Modeling, Machine Learning, and Additive Printing for the Solar Cell Grid
A Meditation on Wire Widths

Oliver Nakano-Baker

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

J. Devin MacKenzie, Chair

Lucien Brush

Kevin Jamieson

Program Authorized to Offer Degree:
Materials Science & Engineering

University of Washington

Abstract

Modeling, Machine Learning, and Additive Printing for the Solar Cell Grid
A Meditation on Wire Widths

Oliver Nakano-Baker

Chair of the Supervisory Committee:
Associate Professor J. Devin MacKenzie
Materials Science & Engineering

The front electrode grid of a solar cell solves a simple problem: it must move current from the solar cell area to a sink. In doing so, it must minimize the resistive power losses incurred in transit while minimizing the size of the shadow it casts on the underlying photovoltaic material. The optimal geometry to achieve this may become complex depending on the material and scaling properties of the system. It may share morphological aspects with the natural systems that solve similar problems, such as leaves, veins, and waterways.

The underpinnings of solar cell grid optimization are a set of well-known equations describing the principal sources of shadowing and resistive power loss. We challenged two assumptions that have been long entrenched in the field: first, that colinear line arrays are strictly optimal when designing solar cell grids. In fact, there is a class of isotropic grids that can outperform linear arrays in certain conditions. Second, that constant grid line height describes most grid applications. Instead, in the era of additive and printed electronics, many grids may be subject to virtuous scaling - the property that larger wires become more efficient conductors.

Under virtuous scaling, complex ramified electrode structures become optimal. A dynamic graph approach was developed to model these electrode patterns, and a locally greedy approach to optimize them realized novel electrode patterns with 1% performance over standard linear grids. The continuous width variation inherent in these patterns presented a

control challenge for an additive process attempting to print them. An electrohydrodynamic (EHD) inkjet printing approach was proposed; the powerful but difficult-to-control additive technique would be combined with machine learning (ML) model-based control.

Rapid serial experiments on the EHD produced a large dataset of training samples. ML models were trained on this data to predict the jetting and deposited feature characteristics of the EHD as a function of the input waveform. Transfer learning was demonstrated between EHD task datasets using a performance-gated mixture-of-experts ensemble, and zero-shot models were trained for use in a model-based control algorithm. Model-based control of the EHD, paired with a single proportional tuning adjustment, was able to achieve $5\mu m$ error when printing lines and $1\mu m$ error when printing dots, both over at least one decade of feature sizes. This capability enabled EHD printing of micron-scale ramified solar cell electrodes with ML-produced recipes. These steps demonstrate a complete journey from concept to prototype, in which computational tools model, optimize, and ultimately manufacture a novel engineered electrode structure to improve solar cell performance.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AC: Alternating Current. In context of an EHD ML waveform, the alternating component of a waveform.

AFG: Arbitrary Function Generator. Hardware capable of producing high-fidelity, arbitrary waveforms on demand.

AI: Artificial Intelligence. A field of study focused on artificial autonomous agents that can act "intelligently", i.e. in a goal-oriented fashion and reactive to its environment.

AUC: Area Under the Curve - generally, area under the ROC. A metric for the ability of a model to separate two classes of datapoint. ROC-AUC = 1.0 indicates a perfectly accurate model on a classification task.

CELL: In this work, "cell" may mean any of: a biological cell comprising or being part of a living organism, a solar photovoltaic cell capable of converting photons to electrical current, one agent in a cellular automation model that acts based on its local surroundings based on a rule-set.

CFD: Computational Fluid Dynamics. A class of computational physics models that approximate the behavior of fluids in motion.

DC: Direct Current. In context of an EHD ML waveform, the bias or offset component of a waveform.

DEP: Dielectrophoresis. Describes a phenomenon where a non-charged polarizable particle experiences a force when exposed to a nonlinear electric field.

DMD: Dynamic Mode Decomposition. A dynamical modeling framework that employs a principle-components representation of modes of motion in a dynamic system to characterize the relationships between said modes in time. The model can, in principle, "play forward" a compressed version of a dynamical system given a starting condition.

DMDC: DMD with control. A DMD variant that admits external energy inputs, e.g. an external control of a dynamical system.

EHD: Electrohydrodynamic, describing a system with fluid flow influenced by electric fields. In this work specifically, an electrohydrodynamic inkjet printer. A method of additive material deposition whereby ink is jetted onto a substrate. Electrodes charge the ink and ground the substrate, resulting in a voltage differential that pulls ink droplets or jets from a nozzle to the substrate. EHD printing is capable of printing inks with a wide range of viscosities and properties and can produce features down to the micron scale.

FTL: Follow-the-Leader, a learning algorithm for in online convex optimization that selects models based on their performance on past data.

GATHERING PROBLEM: For this work, the problem that is solved by transport networks that suffuse an area or volume in order to gather a resource and deliver it to a point sink.

IPA: Isopropanol, or isopropyl alcohol, commonly rubbing alcohol. Solvent used to clean silicon substrates in this work.

KNN: k-nearest neighbor model. An ML model that estimates the value of new inputs by comparing it to the k most similar inputs it has seen before.

MAE: Mean Absolute Error. $mean(Y - Y')$, or the average of all errors between true values and a model's predictions.

MAPE: Mean Absolute Percent Error. $100 * mean(\frac{Y-Y'}{Y})$, or the average of all percent errors between true values and a model predictions.

ML: Machine Learning. A field of study focused on artificial models that can be trained on or fit to observational data in order to reproduce, organize, or make predictions about future similar data.

MLP: Multi-Layer Perceptron. As implemented by SciKit-Learn, a fully-connected neural network with a single hidden layer.

MOE: Mixture of Experts. An ensemble machine learning model where one or more "expert" models are selected to perform inference from a large pool of potential models.

MOE-FTL: Mixture of Experts with a Follow-the-Leader gate. A mixture-of-experts model that selects a single expert model for inference by choosing the model with the best performance on recently-observed data.

MSE: Mean Squared Error. $mean((Y-Y')^2)$, or the average of all errors squared between true values and a model's predictions.

MSLE: Mean Squared Log Error. $mean((log(Y+1) - log(Y'+1))^2)$, or the average of all errors between the logs, squared, of true values and a model's predictions. Because $(log(Y+1) - log(Y'+1) = log(\frac{Y+1}{Y'+1})$, it can also be interpreted as the mean log ratio squared of true to predicted values.

NN: Neural Network. An ML model that employs one or more learnable linear layers, followed by a nonlinear activation function.

PC: Personal Computer. A desktop windows computer.

PEARSON CORRELATION: $\frac{Cov(X,Y)}{\sigma_X \sigma_Y}$, a measure of the linear correlation between two sets of data.

PTFE: Polytetrafluoroethylene, commonly Teflon, a fluoropolymer used to make syringe filter media that was used to filter ink in this work.

PV: Photovoltaics or a photovoltaic cell. A material or device containing a material that reacts to incident photon radiation by producing voltage.

RF: Random Forest model. An ensemble machine learning model comprised of many simple decision trees trained on different random slices of a dataset.

ROC: Receiver Operating Characteristic Curve. A plot describing the true positive vs false positive rate of a binary classifier as its decision threshold is varied.

SIJ: Super Ink-Jet. A commercial EHD inkjet printer manufactured by SIJ, Inc., Japan

SVD: In this work, a Compact Singular Value Decomposition. The SVD of matrix M is $M = U\Sigma V*$. It is an decomposition of the real or complex matrix into component eigenvectors and eigenvalues, where Σ is a square diagonal matrix containing eigenvalues of M.

UW: The University of Washington, the university where the research in this work was carried out.

VIRTUOUS SCALING: A property of channels in a transport network where an increase in the size of a channel makes it disproportionately better at transporting resources.

WCET: The Washington Clean Energy Testbeds, a UW-affiliated open research and laboratory space in Seattle, WA, where much of the hardware used in this research was located

ACKNOWLEDGMENTS

With gratitude, I would like to acknowledge the many friends, co-authors, and co-conspirators who have joined me on this quest for a PhD, including, at least: Clay Boyd, Holly Brunner, Le Cai, Caitlin Cramer, Michael Crump, Jonathan Francis-Landau, Dennis Godin, Tatum Hennig, Liangkui Jiang, Xuepeng Jiang, Chandler King, Joey Law, Richard Lee, Felippe Pavinatto, Sid Rath, Jacob Rodriguez, Brandon Rotondo, Daniel Shae, Shalabh Shukla, Deniz Yucesoy.

A special thank you to the mentors who helped steer this thing over the years: Lucien Brush, Sami Dogan, Hanson Fong, Kevin Jamieson, Krithika Manohar, Igor Novosselov, Hantang Qin, Mehmet Sarikaya, and Hadi Zareie. The guys at SIJ, Kazuhiro Murata and Kotaro Shimizu, who made this all possible. Thank you to my advisor, J. Devin MacKenzie, for your guidance, patience, and trust. This turned out to be a funny one, didn't it?

Thank you to my family. Irene and Ed, Anna and Ryan, Lauren and Michael and Lola, for bottomless support and encouragement. Donna and Bruce, who traveled at the drop of a hat to save us from... every damn thing: illness, babies, and dissertations. Katie, for holding my confidence for me when I misplaced it here and there. Nora and Sylvie, for whom this project has been like a demanding older brother. After this, we're sending him away!

This material is based upon work supported by: generous grants by the UW Clean Energy Institute, SEMI-FlexTech and the ARL; advanced computational, storage, and networking infrastructure provided by the Hyak supercomputer system at UW; the Clean Energy Institute's Clean Energy Testbeds, a National Nanotechnology Coordinated Infrastructure (NNCI) site at the University of Washington, with partial support from the National Science Foundation via awards NNCI-1542101 and NNCI-2025489.

DEDICATION

To the gals: Katie, Nora, and Sylvie.

Figure 1: Oliver and Nora learn machine learning.

Chapter 1

INTRODUCTION: THE BEST GRID

What is the best front solar cell grid? It may seem like an innocuous question. Easily overlooked is the humble electrode. In modern solar cell development, new and promising active materials take up much of the spotlight. However, in some circumstances, solar cell grid optimization is an unsolved problem. Improvements to the state of the art can be economically meaningful, and the technologies used to manufacture front grids continue to evolve.

Today, in 2022, the world is pivoting quickly to renewable energy sources to reduce the future effects of global climate change due to atmospheric CO_2 from fossil fuels.[97] In this transition, solar power has, unsurprisingly, emerged as a mainstay of the new global energy diet. Power from the sun is free and plentiful in much of the world, so, naturally, solar photovoltaics are among the most dominant of all emerging renewable power sources, second only to wind and hydroelectric in the US in 2022.[1] Utility-scale solar energy has increased by at least 20% each year for the past decade in the US,[123] and PV projects account for over half of all future planned installations.[2] With solar panels providing an ever-increasing share of utility-scale power, incremental improvements in efficiency and output can result in significant power gains and competitiveness. For example, a mere 1% relative increase in module efficiency would have corresponded to an additional 3.65 TWh of solar energy produced in the US in 2020, worth about $389 million at the going utility rate.[3]

The original thrust of this project aimed somewhat lower than transforming the world: all I wanted to know was how to build the best solar cell modules for one research group. When I arrived at the lab of J. Devin MacKenzie at the University of Washington in 2017, front electrode optimization was the first task I was handed, in the form of a few pages of hand-drawn equations by Caitlin Cramer and a clear directive - figure out the shape

of an ideal front electrode. It was a task meant to occupy a month or two of my time as I came to grips with a new lab and coursework and so on. Five years later, I don't yet have an answer! In fact, depending on the conditions of manufacturing and solar cell use, the search for "optimal" front grid layouts may be genuinely intractable. Add in the question of manufacturing the things, and the problem becomes truly massive. Luckily, there are some interesting findings on the path to the intractable bits. Electrodes have deep connections to biology and at least one computationally unsolvable problem. When you dig deep enough into the minutiae of solar cell grids, their structure, and their manufacture, you find curiosities. Before diving into that well, we should become acquainted with the main components of a solar cell and a basic set of operating requirements for its front electrode.

1.1 The Parts of a Solar Cell

Solar photovoltaic panels directly convert light to power by capturing photons in an active layer of semiconductor material. Incident photons excite single electrons in the active layer, promoting them from lower level energy states in the material's valence band to higher level states in the conduction band, where the electrons become mobile. The materials that can play host to this process due to a favorable energy band gap - doped silicon, organic semiconductors, III-V composites, CdTe, CIGS, quantum dots, and lead-halide perovskites among them - are the most common protagonists in PV research.[98] However, electron excitation is just the first step in a series of engineered processes that must take place to make practical use of solar energy.

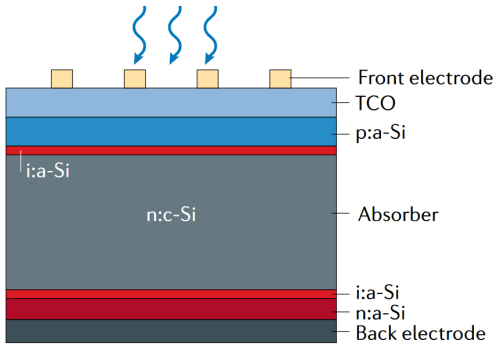


Figure 1.1: Schematic cross-section of a typical doped silicon full-back-contact solar cell device stack. Reproduced from [98].

The second step in this process involves the separation of positive and negative charge carriers (i.e., holes and electrons) before they recombine. This is done first by the standing field in the absorbing layer, then individual carriers are captured by electron- and hole-

transport layers on either side of the absorber. Charge carriers at the back side of the cell pass to a solid metal electrode at this point, but those at the front must navigate a structure that is both electrically conductive and optically transparent. This typically involves a continuous transparent conducting layer of some kind - a heavily-doped silicon layer or transparent conductive oxide such as indium tin oxide (ITO) - and grid of metal lines forming the front electrode.[79, 150]

From there, charge carriers pass through a series of wires that gather current from one or more individual PV devices in a solar cell module and transport it to a battery, a load, or an electrical grid. For utility-scale PV, the charge carriers may navigate transmission lines, city-scale power networks, or long term storage in the batteries of cars, homes, factories, or utility-scale storage. From the moment a photon excites an electron in a PV absorbing layer, the engineering challenge of solar power is a series transport problems. How can these many systems be engineered to efficiently and economically distribute the power weaned from photons to points of application?

1.2 Front Grid Requirements

The transparent front electrode is the second of these transport problems, after charge separation. It presents a characteristic trade-off, in that this layer must be both conductive and transparent. In materials terms, this is a contradiction. The very property that creates a conductive material - a high-density continuous band of electron energy states at the Fermi energy level - makes it highly optically absorbent.[49] (We know from experience that metals are not see-through!)

There are two approaches to solving this: the first is to use one of a class of transparent, conductive metal oxides have been found to be both decent conductors and relatively optically transparent.[51] The second is to use a discontinuous array of metal grid lines, where the individual lines cast some shadow but allow most light through. Thus while the intrinsic role of the PV front electrode is straightforward as a simple metallic conductor, it presents a characteristic design challenge: large metal lines that minimize resistive losses to overall power conversion efficiency cast a larger shadow on the underlying solar cell material, reducing the harvestable photogenerated current. The op-

timization of these two characteristic losses has been well studied both numerically and experimentally.[96, 125, 119, 40, 126, 152, 26, 28, 31, 149, 22, 85, 41, 104] These existing models will be explored in more detail in chapter 2

In general, the principal power losses created by a solar cell's front electrode array arise from the array's shadow, resistive losses in the lines, and resistive losses in the underlying front semitransparent conductive sheet or heavily doped semiconductor as current transports laterally from near the sites of photogeneration in cell active layers to the grid. The goal of front grid optimization is simply to minimize the sum of these losses, thereby maximizing the total electrical power return from the solar cell. This problem, where costs must be balanced to optimize the size and morphology of a transport network, is emblematic of a larger class of such problems which can be encountered in nature and throughout our daily lives! Let's call it "the gathering problem".

1.3 The Gathering Problem

Consider some of the structures presented in Figure 1.2 - a diverse collection of geological and biological systems that range in size from microscopic to continental. All of them are, in some way, solving a problem in this work called the "Gathering Problem." The puzzle is this: there is an area or volume across which a resource is distributed. For it to be useful, the resource must be transported to a sink - that is, a single point, set of points, or subset of the volume. When the resource arrives at a sink, a reward occurs. However, transporting the resource comes at a high cost when the resource must move within the substrate that produces it. So we introduce a second phase to the system that is well suited to transporting the resource but produces nothing by itself. The transport phase may also come with its own associated costs. The system's goal is to maximize reward by providing as many resources to its sink(s) as possible while minimizing the transport costs.

A commonality across transport networks is the remarkable complexity that arises from a simple premise. The resource may be a perfectly uniform field prior to the introduction of the sink, and the gathered resource after the sink is just some flow quantity, a mass of fungible stuff. But the intervening system between these two homogeneous states is complex, even fractal in nature, with complex morphologies that arise from a common set of scaling

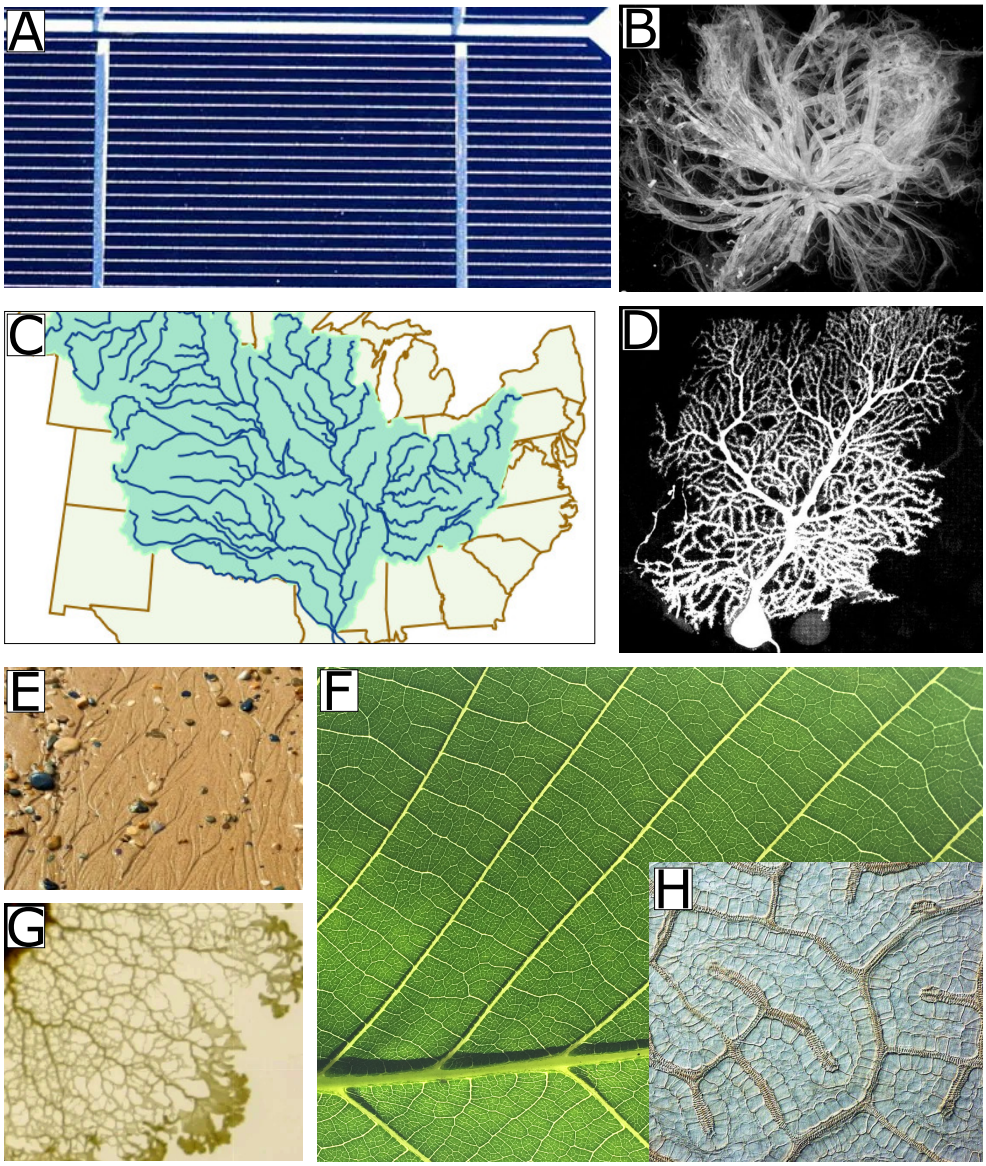


Figure 1.2: Transport Networks for The Gathering Problem. [A] H-bar front grid of a solar cell. [B] X-ray of an insect's tracheal system.[59] [C]The watershed of the Mississippi River. (credit: `https://www.usgs.gov`) [D] Micrograph of a single Purkinje brain cell. (credit: Andrew Shuster) [E] Water runoff pattern in sand. [F] Venous pattern in a leaf. [G] Transport network in the slime mold *physarum polycephalum*[107] [H]Micrograph of leaf vasculature. (credit: Athena McKown)

laws and a characteristic balancing of the costs of transport. It may be tempting, especially with a neuron in the gallery, to draw analogies between these patterns and intelligence, entropy, and evolution. Be that as it may, this work is more interested in the practical application of such patterns, the tools needed to design them, and how to create systems that deploy them to address engineering transport challenges.

Systems that solve the gathering problem often deal with continuum flows: quantities of a media, such as nutrients, water, or heat, that move through the system continuously. In this work, the gathered resource is electrical current. Some of the systems pictured are co-optimized for fault or damage tolerance, but as we approach solar cell optimization we will be focused solely on reward maximization. Specifically, the reward function can be defined as the total power output of the solar cell system, calculated $T = IV$ where total power T in Watts is the product of current I in Amperes and voltage V in Volts collected at the sink. (In this work, the variable P will often be used to describe power losses, while T is reserved to represent the final, all-in power reward. Hopefully, this alleviates more confusion than it creates.) Each gathering system is defined by its particular trade-offs of costs and losses, and the solar cell is no different: the material properties of the solar cell junction and the electrical conductivities of the solar cell and transport grid will ultimately give rise to the morphology of optimal transport networks. The most commonly used of these symbols are summarized in Table 1.1. From these properties, we can define a simple methodology to calculate the principle sources of power loss in solar cell front electrodes.

1.4 Calculating Solar Cell Losses

Power losses in solar cell front grids come in two flavors: shadowing losses and resistive losses. In general, shadow losses in solar cells can be estimated based purely on geometry, assuming perpendicular incident light, no internal reflections, and zero transmission through the metal bars. (Assumptions rarely survive contact with the enemy, but for these initial modeling efforts, these will do.) Resistive losses must contend with a continuous accumulation of solar current everywhere in the solar cell, so the basic formula for current-based resistive power loss, $P = I^2R$, must be modified to account for this variable current. Using the variables illustrated in Figure 1.3, the resistive losses of solar current moving through a sheet would

Table 1.1: Symbols and Units for Front Grid Numerical Models.

Symbol	Description	Units
R	Solar cell radius	cm
b	Grid pitch	cm
w	Grid wire width	cm
h	Grid wire height	cm
ϵ	Sink radius	cm
ρ_{metal}	Metallization resistivity	$\Omega\ cm$
ρ_{TC}	Transparent conductor sheet resistance	Ω/sq
J_{sol}	Solar current density	A/cm^2
V_{op}	Operating voltage	V

be

$$P = \int_0^A I(\alpha)^2 \delta R. \tag{1.1}$$

Here $\delta R = \frac{\rho_{sheet}}{x}\delta\alpha$ is a "unit of resistance" from an increment of sheet material parallel to the electrical current vector. $I(\alpha) = J_{solar}\alpha x$ is the total current accumulated in the sheet. Note that x would be a function of α for any non-rectilinear geometry. Also, while ρ_{sheet} here is a sheet resistance with units $[\Omega/sq.]$, the formula can be adapted to any expression of electrical resistance. I.e., for a bulk material with a known height we would have $\delta R = \frac{\rho_{material}}{hx}d\alpha$ where $\rho_{material}$ has units $[\Omega - cm]$.

The equations that result from calculating the individual shadowing and resistive losses for several grid designs are shown in Figure 1.4. Notably, the equations for the hexagonal and square grids are the same. Also note that the triangle equations also become equivalent if $b_{triangle} \to \frac{3}{2}b_{square}$.

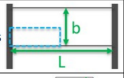
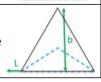
	Fractional Power Loss			
Geometry	Shadow	Contact	Sheet	Lines (assuming linear bus bars)
Bars 	$1\frac{w}{b}$	$[\frac{J_{sc}}{V_{oc}}] \cdot 1 \cdot \rho_{contact} \cdot \frac{b}{w}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{12} \cdot \rho_{sheet} \cdot b^2$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{12} \cdot \rho_{wire} \cdot L^2 \cdot \frac{b}{wh}$
Hex 	$2\frac{w}{b}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{2} \cdot \rho_{contact} \cdot \frac{b}{w}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{32} \cdot \rho_{sheet} \cdot b^2$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{12} \cdot \rho_{wire} \cdot L^2 \cdot \frac{b}{wh}$
Square 	$2\frac{w}{b}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{2} \cdot \rho_{contact} \cdot \frac{b}{w}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{32} \cdot \rho_{sheet} \cdot b^2$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{12} \cdot \rho_{wire} \cdot L^2 \cdot \frac{b}{wh}$
Triangle 	$3\frac{w}{b}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{3} \cdot \rho_{contact} \cdot \frac{b}{w}$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{72} \cdot \rho_{sheet} \cdot b^2$	$[\frac{J_{sc}}{V_{oc}}] \cdot \frac{1}{18} \cdot \rho_{wire} \cdot L^2 \cdot \frac{b}{wh}$

Figure 1.4: Equations for the principle sources of front grid power loss for several electrode geometries. All expressions are unitless, i.e., they express the fraction of total ideal power gathered by the solar cell that is lost in shadow or transport.

1.5 Looking Ahead

There: done! The solar cell grid problem has been quantified, all its secrets laid bare in cold math, and all that is left now is to solve some equations and go home. That is more or less what I thought at this juncture. In a sense, I was correct: the solutions to these types of equations had long been understood, their optimal values mapped out since the 70s. However, those old solutions used old assumptions, and today's energy manufacturing landscape had some new angles. Solar cells did not have to look how they used to - bus bars passed through to the back side now, and alternatives to silicon were on the rise. The success of perovskites and the promise of an all-printed solar cell brought a new set of additive techniques to the fore.[10, 4] Would we soon be printing electrodes? Would we be stuck with indium tin oxide conductive layers, or could we create electrodes small enough to replace them? In the new additive landscape, there were questions to be answered. What would an optimal printed electrode look like, and how could we go about creating it? I started back at the beginning.

In chapter 2, a numerical description of the principal sources of power loss and their trade-offs were developed for several front grid geometries. Isotropic designs were considered strictly inferior in the past, but we made a case for revisiting these designs and identified

some regimes where they were preferable. A tapered-line solution to optimal grid design was reviewed, with connections to the scaling of transport networks in biological systems.

Then in chapter 3, the morphology of those biological systems was explored in more depth using a computational framework that optimized a graph with cellular automaton nodes. The concept of "virtuous scaling" was introduced to justify exploring these intractable problems, leading to an optimal class of ramified transport network designs.

In chapter 4, the practical challenge of creating these varied grid designs was attacked using an electrohydrodynamic (EHD) ink jet printer, a squirrely and adaptable machine capable of carrying additive-printed electronic designs down to the micrometer scale. A few informative but dead-end attempts to model the EHD process would lead, eventually, to a machine learning control approach.

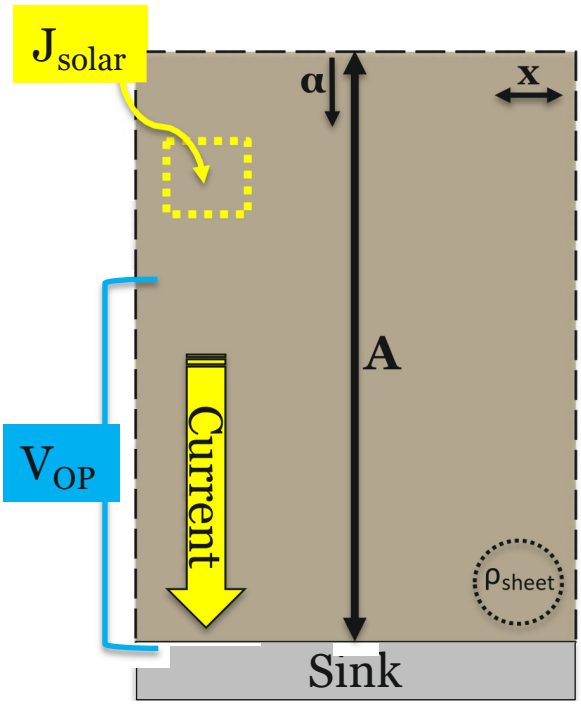


Figure 1.3: Schematic view of the variables to calculate spreading resistance in a solar cell.

The remainder of the project constituted data gathering in chapter 4 and model training in chapter 6 of an ensemble of machine learning models that could predict, and thus control, an EHD printing process. In chapter 7, these models were put to the test and deployed in a model-based control algorithm to direct the additive manufacture of some very particular grid patterns.

This project was wide-ranging, from the mathematical underpinnings of diffusion system optimization to the specific programming tricks that let us grapple with the open-ended pathing problem to the dynamics of a micron-scale electrohydrodynamic process. But, ultimately, this work is about morphology - the shape of transport networks and the practicality

of using them to solve engineering problems. However, all of that complexity grows from the underpinning math, and that is where we will begin.

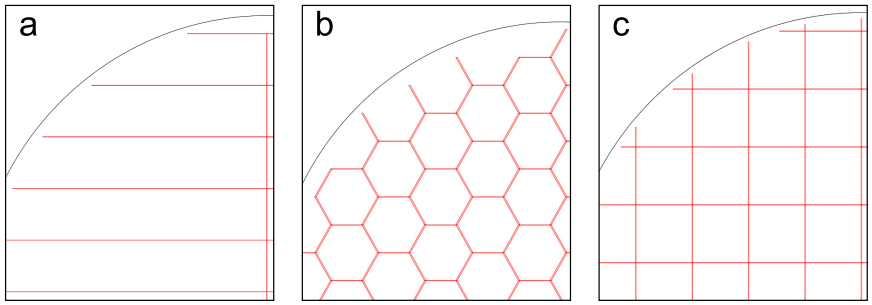


Figure 2.1: Optimized layouts of 5 cm diameter circular front cell electrode designs using (a) H-bar, (b) hexagonal, and (c) square grid layouts.

Chapter 2

NUMERICAL MODELS FOR TRANSPORT GRIDS

The first angle of attack for optimization of the solar cell front grid used the simplest possible numerical description of grids that deploy space-filling, straight-line networks of metal grid lines. The most common of these utilize parallel lines intersecting a perpendicular bus bar, referred to as an "H-bar" design. A second class is comprised of isotropic grids constructed from overlapping layers of parallel lines at different angles - these include all regular tessellating planar unit cells: closed hexagons, triangles, or squares. Examples of H-bar, hexagon, and square patterns are shown in Figure 2.1.

The isotropic grids have long been understood to have inferior properties to H-bar grids for a basic front electrode array application.[96] At equivalent pitch with all other materials and dimensions held equal, isotropic grids incur a 2x increase in shadowing losses while enjoying no benefit to their current transport efficiency as compared to H-bar patterns. For this reason, virtually all contemporary solar cell modules utilize some flavor of front H-bar electrodes. [150]

2.1 Revisiting Isotropic Grids

Two features of emerging solar cell design motivated a reexamination of the isotropic family of grid designs in [94]. First, the shift to point sinks in modern cell layouts, and second, the competitiveness of high-resistance front layers in thin-film solar cells. The first came from the use of through-stack vias by many commercial modules; vias shuttle current to a rear-side bus and are point sinks for the front-side charge carriers. [71] In these architectures, H-bar electrodes result in an indirect current pathway to the sink where isotropic grids would transport along an effectively straight path and with the same efficiency of transport as the H-bar.

The second feature was the competitiveness of high transparent conductor (TC) resistance (ρ_{TC}) solar cell module designs. TCs constitute layers of a device that confer an ability to transport current in the plane of the device while being mostly transparent to light, excluding front metallization. Thin film photovoltaic modules, such as perovskites, cadmium telluride, and organic photovoltaics, commonly use dedicated TC layers such as indium tin oxide (ITO) on their front face to effect charge carrier transport from the active material to the edge of the cell or an electrode grid. ITO layers are expensive and give rise to unavoidable optical extinction losses due to the inherent trades offs between metallic-like conductivity derived from their significant carrier concentrations and absorption and reflectivity. Sourcing materials for their manufacture can be geopolitically challenging, leading to much research into their replacement [158, 122] i.e. by conductive polymers or nanowire layers.

For many such layers, Jacobs et al. [63] contend that it is advantageous to increase the sheet resistivity of the TC, and thus minimize optical extinction, beyond the traditional $\sim 10\Omega/sq$, with values up to $10k\Omega/sq$ optimal for some systems. Additionally, perovskite photovoltaics' high charge carrier mobility could motivate the deployment of TC-free, high-resistance, low-extinction modules. [89, 77] Isotropic grids with fine pitch should excel in these high-ρ_{TC} design spaces due to lower diffusion distances, aiding in the minimization or elimination of ITO layers.

New additive manufacturing pathways for metal lines are being explored [29, 38, 75]

and there is interest in thin film tandem architectures which can require multiple TC layers within a solar cell stack. [73] These developments perturb the established material properties and design assumptions in commercial solar cell modules. Considering solar's growing market and shifting design landscape, it is an ideal time to reexamine old assumptions around front contact design, including the supremacy of the H-bar layout.

2.1.1 Numerical Optimization of Solar Cell Grids

Strategies for the optimization of solar cell grids have evolved significantly over the past 50 years. Early studies by Napoli et al. [96] and Serreze [125] established numerical models for the individual sources of power loss in straight- and tapered-line grids respectively. Such models are suited to rapid numerical optimization, and this study employs similar methods to revisit isotropic transport grids. We assume the simplest possible conditions, including single values for line width and pitch and constant wire height. However, alternate design approaches exist that improve grid performance at the cost of increased complexity and expanded design space.

Most importantly, it is optimal to vary grid line widths locally according to the relationships outlined by Scharlack [119], with every wire scaled to an optimal width relative to solar current, local wire current, and material constants. The relationship has been utilized by later works including spoked radial grids for circular cells. [26, 31]

More recently, grid optimization is often expanded to include module-scale bus bar and material cost optimization.[85, 104] Some grid design strategies abandon the notion of fixed-pitch grids, adopting free-form approaches. These strategies range from strategic addition or subtraction of grid elements[22] to free-form design of ramified transport networks[28, 53] and can potentially be adopted to arbitrary solar module geometries.[54] This approach will be revisited in chapter 3.

For now, we confine this study to basic numerical equations for the principal losses in a front electrode isotropic grid pattern and verify the model using the commercially available Griddler simulation package. [153] The numerical models are used to design optimal H-bar and isotropic grids for high-ρ_{sheet} point-sink solar cells, and their relative performance is

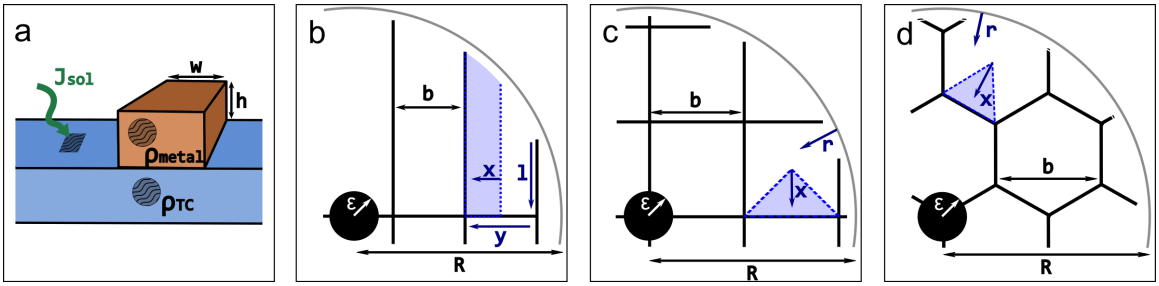


Figure 2.2: Schematic layout of grid cross-section (a), H-bar (b), square (c), and hexagonal (d) grids. Unit tiles used for TC resistive loss calculations are demarked in dotted blue.

evaluated using Griddler's finite element (FE) solar cell simulator. Areas of point-sink solar cell design space are identified where an isotropic grid pattern outperforms H-bar grids.

2.1.2 Numerical Methods for Circular Solar Cell Loss Calculations

To compare H-bar and isotropic patterns, numerical models were used to estimate the main sources of power loss in the front metallization: shadowing, grid line resistance, and TC resistance. Symbols and units for these calculations are defined in Table 1.1 and illustrated in Figure 2.2.

Numerical optimization of the numerical expressions was used to rapidly identify near-optimal configurations of H-bar and isotropic grids on circular solar cells with a single central current sink. Both grid types were optimized across a range of material properties and cell radii, and optimal designs were modeled using Griddler software to determine their peak power output. These FE-calculated power estimates were used to compare the performance of H-bar and isotropic grids.

A notable omission from this study is the consideration of contact losses between TC and grid, as well as downstream contact losses such as where lines intersect with bus bars. These power losses are typically smaller by an order of magnitude than the other power losses considered here and are a secondary source of loss for most practical solar cell modules. [85] All code used to model and optimize grids in this study can be found at `github.com/`

`onakanob/iso_grid_optimizer`.

Calculating power losses

We derive closed numerical equations for the shadow, line resistance, and TC sheet resistance losses in H-bar and isotropic grid arrangements on a circular solar cell. As standardized in former works [96, 119, 40, 126, 152], grid resistive losses were calculated in watts as

$$P_{lost} = \int_j^k I(x)^2 \delta R(x) \tag{2.1}$$

where j and k are locations where current begins and ends respectively; $I(x)$ is the cumulative current in Amps at location x, and $\delta R(x)$ represents an increment of resistance at location x. Generally, this can be expressed as

$$\delta R(x) = \frac{\rho}{l(x)} \delta x \tag{2.2}$$

where ρ is the sheet resistance (in Ω-per-square) of the transporting medium, $l(x)$ is the width of the transporting medium, and δx is an incremental distance along the direction of current flux. When transporting through an array of metal lines, the effective grid sheet resistance ρ of both H-bar and isotropic grids are

$$\rho_{grid} = \frac{\rho_{metal} b}{wh} \tag{2.3}$$

where ρ_{metal} is the metal resistivity, b is the pitch of the grid as illustrated in Figure 2.2, and w and h are the width and height respectively of the metal lines. The effective grid sheet resistivity holds only in the direction parallel to the metal lines in an H-bar grid, but it is isotropic in the plane of the device for triangle, square, and hexagon grids. The equivalence and isotropic nature of these grids is demonstrated in Appendix 2.1.6.

The maximum theoretical power gathered by a circular solar cell (i.e., with a lossless front grid) would be

$$P_{max} = J_{sol} V_{op} \pi R^2 \tag{2.4}$$

where J_{sol} is the solar current density, V_{op} is the cell operating voltage, and R is the radius of the circular cell. Shadow loss can be expressed as a fractional decrease in the max power

due to the proportion of device stack area covered by metal lines. For H-bars, this is simply the ratio of width to pitch; for isotropic grids, the shadow loss is doubled. We have:

$$P^{Hbar}_{shadow\,loss} = P_{max}\frac{w}{b} \tag{2.5}$$

$$P^{iso}_{shadow\,loss} = 2P_{max}\frac{w}{b} \tag{2.6}$$

Below, we outline simple models for the resistive losses in the two grid types on a circular cell.

Isotropic Grid Losses

The three non-H-bar grids all have isotropic conductivity in the solar cell plane and are equivalent to each other in terms of their grid resistive losses; see subsection 2.1.6 for a demonstration of this effect. As long as the solar cell is large enough to contain many unit cells, i.e. $R \gg b$, current can effectively be assumed to travel radially from any point in the solar cell to the central sink. Cumulative current generated between the solar cell edge and radial position r is

$$I^{iso}_{cumulative}(r) = J_{sol}\pi(R^2 - r^2) \tag{2.7}$$

This expression can be combined with Equation 2.3 for the grid effective sheet resistance into Equation 2.1 and Equation 2.2, taking the width $l(x) = 2\pi r$ because current at position r moves radially through a circumferential slice of material. The total power lost due to grid resistance is

$$P^{iso}_{grid\,loss} = \int_{\epsilon}^{R} I^{iso}_{cumulative}(r)^2\frac{\rho_{grid}}{2\pi r}\delta r \tag{2.8}$$

where ϵ is the radius of the center point sink. The trend is resolved using numerical integration in the software model.

Resistive losses in the TC are generated as current diffuses across a triangular unit cell, illustrated in Figure 2.2. The variable x is defined to be the distance from the tip of this triangle on a path perpendicular to the nearest line, i.e. $x = b/2$ defines the coordinate where current reaches the conductive grid. It is assumed that current is evenly distributed to achieve uniform flux at every x, and we solve this equation for both square and hexagonal

grids. The cumulative current in the TC can be written

$$I^{square}_{cumulative}(x) = J_{sol}x^2 \quad I^{hex}_{cumulative}(x) = \frac{1}{\sqrt{3}}J_{sol}x^2 \tag{2.9}$$

Using Equation 2.2 with $\rho = \rho_{TC}$, $l^{square}(x) = 2x$, and $l^{hex}(x) = \frac{2}{\sqrt{3}}x$, substitute in to Equation 2.1:

$$\begin{aligned} P^{square\,unit\,cell}_{TC\,loss} &= \int_0^{\frac{b}{2}} J^2_{sol}x^4\frac{\rho_{TC}}{2x}\delta x \\ &= \frac{1}{128}J^2_{sol}b^4\rho_{TC} \end{aligned} \tag{2.10}$$

and

$$\begin{aligned} P^{hex\,unit\,cell}_{TC\,loss} &= \int_0^{\frac{b}{2}} \frac{1}{3}J^2_{sol}x^4\frac{\sqrt{3}\rho_{TC}}{2x}\delta x \\ &= \frac{\sqrt{3}}{384}J^2_{sol}b^4\rho_{TC} \end{aligned} \tag{2.11}$$

This accounts for loss in one unit cell. To estimate power loss across the entire solar cell, scale up using a ratio of areas, with $A = \pi R^2$, $A_{square} = \frac{b^2}{4}$, and $A_{hex} = \frac{b^2}{4\sqrt{3}}$:

$$\begin{aligned} &P^{iso}_{TC\,loss} \\ &= P^{square\,unit\,cell}_{TC\,loss}\frac{A}{A_{square}} = P^{hex\,unit\,cell}_{TC\,loss}\frac{A}{A_{hex}} \\ &= \frac{\pi}{32}J^2_{sol}b^2R^2\rho_{TC} \end{aligned} \tag{2.12}$$

Power loss density is obtained by dividing by circular area A. This is not possible for grid losses, which depend on R. With identical models describing their shadow, grid-, and TC-resistive losses, the various isotropic grids are effectively interchangeable in this analysis.

H-bar Grid Losses

Resistive losses in the H-bar grid are calculated in two parts. First, losses in a parallel array of lines with width w, height h, and pitch b is calculated, with current traveling from the area of the cell to a single cross-cell bus. Then, losses in the bus are calculated as current travels to the center sink . Using the sheet resistance description of the line array from Equation 2.3 the loss in a strip of the line array intersecting and perpendicular to the bus,

of length L and width δy, can be calculated. Recognizing that the current at distance l from the cell's edge will be $J\,l\,\delta y$, from Equation 2.1 we have

$$P_{segment}(L) = \int_0^L (J_{sol}\,l\,\delta y)^2\,\frac{\rho_{grid}}{\delta y}\delta l = \frac{1}{3}J_{sol}^2 L^3 \rho_{grid}\delta y \tag{2.13}$$

The length of the array lines as one travels away from the central sink can be expressed as $L = f(y) = \sqrt{R^2 - y^2}$ where y is the distance from the sink where the array intersects the bus line. (Note that as $y \to R$ at the end of the bus bar, the length of the array, and its associated resistive losses, go to zero.) The total resistive loss in the array comes from integrating this value numerically across all four quadrants of the cell:

$$P_{grid\,loss}^{Hbar\,array} = 4\int_0^R \frac{1}{3}J_{sol}^2 (R^2 - y^2)^{3/2}\rho_{grid}\,\delta y \tag{2.14}$$

Resistive losses in the bus bar come from similarly integrating from the cell's edge to the sink the current incident from the array on the bus. The current in the bus at position y is

$$I_{bus}(y) = \int_y^R 2J_{sol}\sqrt{R^2 - y^2}\delta y \tag{2.15}$$

and the total power loss from both sides of the bus is solved using numerical integration in implementation as

$$P_{grid\,loss}^{Hbar\,bus} = 2\int_\epsilon^R I_{bus}(y)^2\rho_{grid}\,\delta y \tag{2.16}$$

where ϵ is the radius of the center point sink.

Device TC resistive losses for an H-bar grid are derived in past works. [96, 119] For this study, loss is calculated over the R-radius cell area as

$$P_{TC\,loss}^{Hbar} = \frac{\pi}{12}J_{sol}^2\,b^2\,R^2\rho_{TC} \tag{2.17}$$

And finally, in addition to the shadow cast by the major grid lines per Equation 2.5, the bus casts a single shadow with power cost

$$P_{shadow\,loss}^{Hbar\,bus} = 2R\,J_{sol}\,V_{op}\,w \tag{2.18}$$

These models include some non-physical assumptions, most notably a total conservation of current, constant voltage across the solar cell, and negligible contact losses between TC and line. Nonetheless, we verify as described in section 2.1.3 that optimization of the numerical models yields near-optimum grid layout designs.

Optimizating the Numerical Grid

Numerical models were implemented as functions in Python for isotropic and H-bar grids. All numerical integration was carried out using SciPy's [121] integrate.quad() function, which invokes the Fortran QUADPACK library. Optimization of grid geometry means solving

$$argmax_{\{w,b\}} P_{max} - \sum_{l \in losses} P_l(w,b) + \lambda b^2 \tag{2.19}$$

Where λ is a small constant that encourages a larger grid pitch. This regularization is a practical term introduced because in the isotropic grid, the TC term $P_{TC\,loss} \propto b^2$ will drive an unbounded reduction in both w and b, pushing the grid to the lower design bounds consistently with minimal improvement in actual performance. Regularization of the grid geometry yields patterns that are less computationally strenuous for FE with minimal reduction in performance.

Optimal values of w and b are identified using SciPy's optimize.minimize() function, which employs the L-BFGS-B algorithm. For all grids, line width w had bounds (1 μm, $\frac{R}{2}$) and pitch b had bounds (1 mm, $\frac{R}{2}$) where R was the cell radius. The pitch lower bound was introduced to maintain practical computation burden in FE and was the only boundary condition to affect the grids generated in this study; the smallest designed line width in practice was 2 μm.

2.1.3 Validation of Numerical Grid Designs

The outputs of the numerical model and optimization loop were validated using Griddler, a free solar cell FE modeling package. We first verify the optimality of found grid patterns and the equivalence of the isotropic grid patterns. Then we compare the performance of H-bar and isotropic grids across a range of solar cell and material properties.

A static operating voltage is manually chosen for the numerical model so that its optimal designs are also optimal in Griddler simulation. Figure 2.3 shows that for a $1k\Omega/sq$ TC sheet resistance design, the two models are in close agreement on optimal pitch and width values when $V_{OP} = 420mV$.

For all experiments, a set of standard centered conditions were specified for a circular

5 cm radius solar cell with a central current sink, 2 mΩ/sq metal grid lines, 0.5 mm sink radius, 420 mV operating voltage, and 20 mA/cm2 solar current. Griddler uses a double diode model for active material element simulation, and diode saturation currents of 450 fA/cm2 and 10 nA/cm2 were chosen. Griddler's internal optimizer was used to identify max-power operating voltage and current for each grid design.

Finally, the center condition TC sheet resistance value was chosen where isotropic and H-bar grids show equivalent performance. Figure 2.4a shows that this occurs at about 15 Ω/sq. TC sheet resistance swept values range from 1 to 10^4 Ω/sq, covering from the low end of high-performance ITO layers to the upper range of TC materials considered by Jacobs et al.[63].

2.1.4 Isotropic Grid Results

For a solar cell at the center conditions described above, the model identified an optimal H-bar geometry of 7.8 mm pitch and 0.62 mm line width. Numerical power predictions are increased by a factor of 1.53 to bring center point predictions in line with the FE result. This factor is used to aid visualization as in Figure 2.3, but it does not impact optimization results, which are unaffected by constant scaling. All performance values in Figures 2.4 and 2.5 are un-scaled outputs from Griddler simulations.

In Figure 2.3, predicted power output from the numerical and FE methods are plotted around this optimal point, varying grid pitch and line width independently. The numerical model neglects the effects of voltage on current generation, causing some mismatch between the two models. However, with the tuned value V_{OP} = 420 mV, there is close agreement between the numerical and FE models on the optimal grid design. These trends illustrate the resilience of the grid design to perturbations - for instance, a 50% decrease in line width from the optimum results in just a 7% reduction in power density, from 6.9 to 6.4 mW/cm^2. Therefore we do not expect small errors in grid design from our algorithm to be significant to the question of isotropic vs H-bar grids.

We compare the power output of circular solar cells with H-bar and isotropic front grid patterns given a range of cell geometries and material properties. Square grids were used to

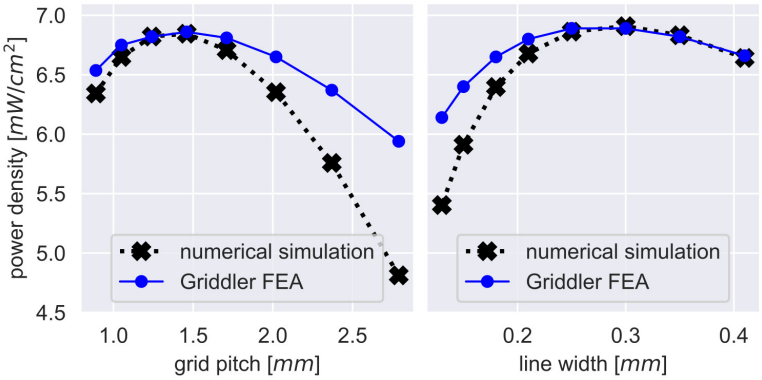


Figure 2.3: Optimum values for grid pitch and line width found using numerical optimization are also near local optima in Griddler's solar cell FE when V_{OP} = 420 mV.

simulate all isotropic patterns. All simulations used solar cells at the centered conditions, but varying one parameter at a time. In Figures 2.4 and 2.5, each of the cell's TC sheet resistance, grid line resistance, radius, and solar current density is varied systematically, and FE results for the optimal grid power output of H-bar and isotropic grids are compared.

At center conditions, TC sheet resistivity is set at 15 Ω/sq so that both the H-bar pattern and the isotropic grid produce an equivalent 8.61 mW/(cm^2). While H-bars perform well at low sheet resistances below 15 Ω/sq, the isotropic grid significantly outperforms optimal H-bar patterns for high TC sheet resistance values and shows more modest gains for large cell radii and elevated solar current.

2.1.5 Isotropic Grids: Discussion

While the H-bar grid continues to dominate performance with industry-standard TC sheet resistance values of <15Ω/sq, isotropic grids matched or outperformed H-bars as front electrodes on center-sink circular solar cells with high TC sheet resistances and, to a lesser extent, large cell areas and high solar currents. The principal advantage of the H-bar design – its 2x lower shadow loss coefficient – was present in these designs, but a combination of other factors gave the edge to isotropic patterns. These factors likely included a flat reduction in TC sheet losses, spreading transport burden across more metal elements, an easier

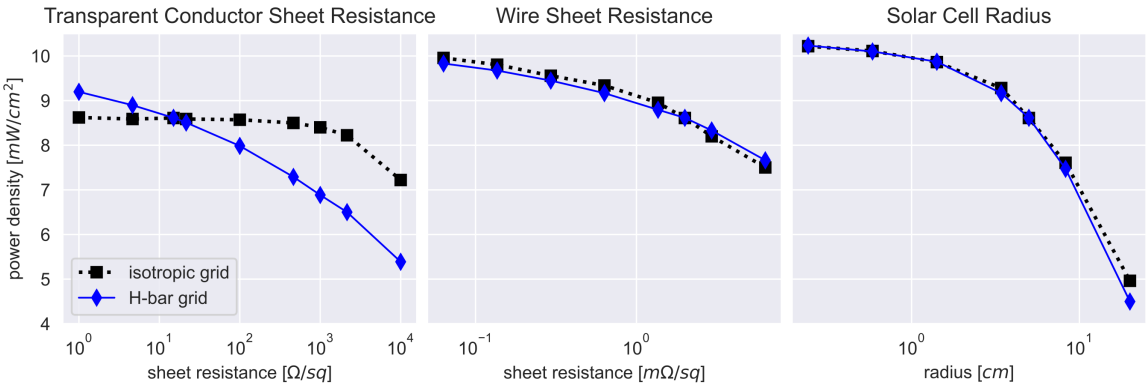


Figure 2.4: Comparison of power density from H-bar and Isotropic front grid solar cells with varying TC sheet resistance, wire sheet resistance, and solar cell radius.

optimization landscape enabling fine-pitch patterns, and a reduction in voltage variance across the device stack.

TC resistive losses in both grids vary $\propto b^2$, with the isotropic grid experiencing lower loss by a constant factor of 8/3. The scaling factor implies that for any TC sheet resistance, a sufficiently small grid pitch will render resistive losses negligible. Conversely, a sufficiently large pitch will cause TC losses to dominate device behavior. Contemporary solar cells usually deploy a front conductive transparent layer such as ITO with sheet resistance $<$ $10\,\Omega$/sq., where millimeters-scale grid lines are sufficient to eliminate most TC losses and the isotropic grid's constant advantage has minimal utility. Under higher TC resistivity, reduction of the pitch value becomes critical to grid optimization. With very high TC resistivity, both grids in this study were clamped by the lower bound pitch of 1mm, at which point the isotropic grid's flat 8/3 advantage becomes a significant performance driver.

The lower TC losses in isotropic grids arise due to the triangular unit cell across which current diffuses to grid lines, as seen in Figure 2.2c and d. This reduces the average distance from any point in the TC layer to the nearest grid wire from $\frac{1}{4}b$ for H-bars to $\frac{1}{6}b$ for square or hexagonal isotropic grids. More importantly, it reduces the average current density in the TC because current can spread with the triangle's cross-section as it travels along the

direction x towards a grid line, reducing resistive power loss.

An idealized solar cell might maintain a uniform max-power voltage across the entire device stack; in reality, a spread of voltages around the max-power point arises across the solar cell due to resistance effects. The isotropic grid, by reducing resistance loss and thus voltage drops in the TC layer, is closer to the ideal case; this results in higher max-power voltage without losing too much photocurrent. To illustrate, at the lowest sheet resistance value studied (R_{sheet} = 46Ω/sq), the optimal isotropic and H-bar grids both produced about 8 mW/cm^2. The two cells had nearly identical open circuit voltage of 622-624 mV, but the H-bar grid produced a much higher short circuit current, 18.6 mA/cm^2 vs the isotropic 16.6 mA/cm^2, due to its reduced shadowing. Despite the H-bar's current advantage, the max-power conditions for the isotropic grid of 523 mV and 15.58 mA/cm^2 matched the H-bar's 477 mV and 17.14 mA/cm^2 in terms of power output due to the isotropic grid's higher operating voltage. All effects of variable voltage and solar current are omitted from the numerical model, likely leaving some room for additional optimization in both grid designs.

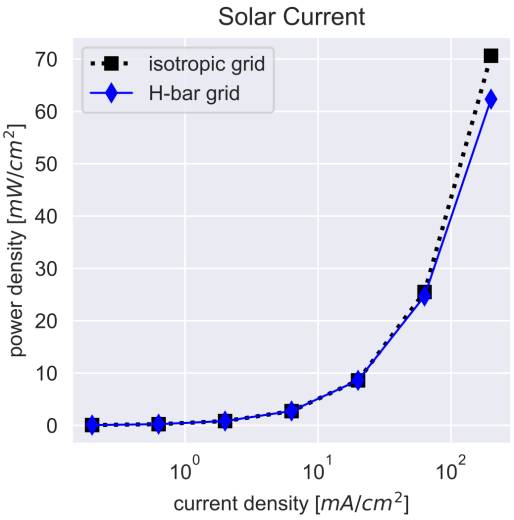


Figure 2.5: Comparison of power density from H-bar and Isotropic front grid solar cells with varying incident solar current.

The final weakness of the H-bar grid was its reliance on concentrated current flux through the single bus bar element – this introduces an additional constraint to its optimization problem and is largely a product of the assumptions in this study. In short, the H-bar grid wants larger line width to reduce bus bar loss but smaller pitch to reduce TC loss – this couples bus bar and TC losses, setting up a trade-off in its optimization space that the isotropic grid does not need to deal with. The isotropic grid is, in fact, unconstrained in its specific choice of width and pitch except by TC loss considerations. $P^{iso}_{grid\,loss} \propto b/w$ and

$P^{iso}_{shadow\,loss} \propto w/b$, so the isotropic grid optimizer only needs to find an optimal width-to-pitch ratio, and can easily miniaturize the pattern to accommodate higher TC resistances, limited only by the simulation's lower bound and regularizing λ term.

This study assumed that current is generated in a circular area centered on the sink, a geometry chosen for its simplicity in assessing the relative performances of grid types. Such a module layout would seldom be encountered in industrial solar cell applications, where semi-square wafers or half cells and often multiple sinks per module are the norm. The extension of the model to any particular layout of sinks is left to the individual practitioner. However, the circular area assumption is likely to be a good approximation of any layout of sinks approaching a close packing. If isotropic grids are to find successful application in contemporary commercial solar modules, such as half-cut and the new C3 modules, a bus bar layout would need to be identified that leverages the area efficiency of close-packed point sinks while conforming to the overall rectilinear arrangement of those modules. The design of optimal bus bar layouts for isotropic grid solar cells is the next step to enable the comparison and competition of isotropic grids against real H-bar solar cell grid designs.

In Conclusion, the foundational numerical models that established today's solar cell electrode designs deemphasized the utility of isotropic grids - arrays of triangular, square, or hexagonal wire patterns - due to their higher degree of shadowing for equivalent conductive performance, compared to H-bar grids. However, in certain corners of the solar cell design space, isotropic grids are capable of outperforming H-bars. Isotropic grids have interesting properties, including an equivalence in shadow/resistance trade-offs across all possible planar isotropic grid designs.

We have attempted to fill this gap in the modeling literature by presenting a simple numerical framework for modeling the shadow and resistive losses in circular, center-sink H-bar and isotropic grids, and for optimizing the pitch and width of the designs. Using Griddler FE software as validation, we demonstrate that isotropic grids produce more power for solar cells with high transparent conductor sheet resistance and point sinks. These designs could help enable a transition away from transparent conductive oxides and their associated processing, material, and optical costs, in favor of lightly doped electron or hole transport layers combined with fine front electrode grids. While the cutting edge of grid

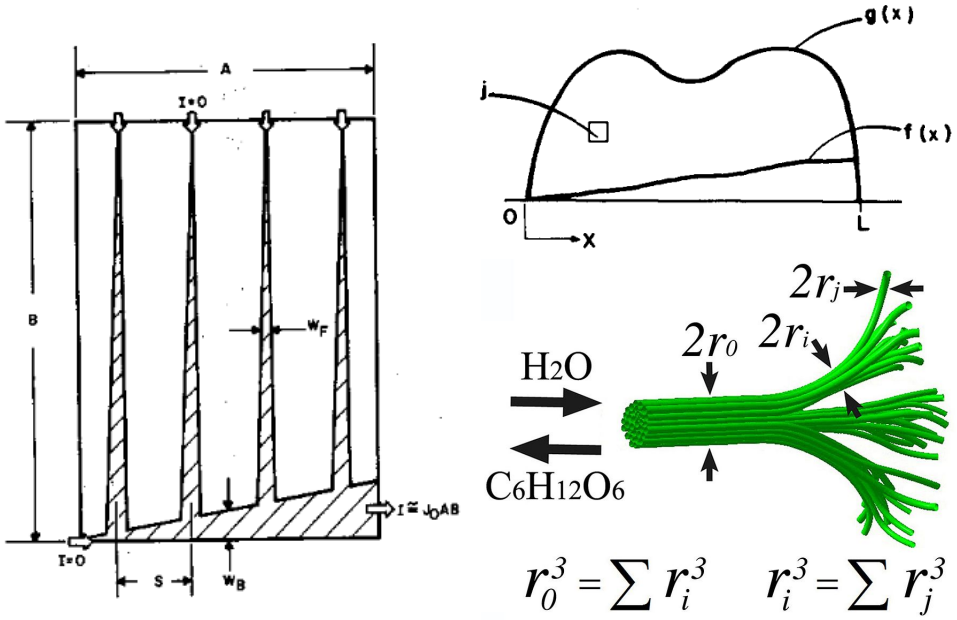


Figure 2.6: Networks that scale with Murray's Law. [Left] Serreze's[125] linear line taper. [Top Right] Shcarlack's[119] universal line scaling. And [Bottom Right] Murray's Law scaling in a plant being network[162]. All three maintain a constant optimal flux density throughout the network.

design has moved towards ramified networks and free-form design, the family of isotropic grids may still provide a performant baseline design, simple to optimize and deploy, for the next generation of center-sink, ITO-free solar cells.

2.1.6 All Isotropic Geometric Grids are Equivalent

It was insinuated but not justified above that all of the various isotropic grid designs are equivalent to each other. That result is derived and discussed in Appendix A

2.2 Introduction to Tapering: Murray, Serreze, and Scharlack

As mentioned above, despite their relative prevalence, straight-lined electrode grids are almost certainly sub-optimal. As early as 1978, Serreze[125] identified a rule set for designing tapered solar cell bus and grid lines, where the rate of taper was determined by the solar

current, metal conductivity, and area of the current collection. The next year, Scharlack[119] generalized these relationships to admit any solar cell shape. Scharlack's optimal scaling law for electrode wire width applied to wires with constant sheet conductivity and is simply stated as

$$f(x) = \alpha \int_0^x g(\xi)\, d\xi \tag{2.20}$$

where α is a constant determined by materials and illumination and ξ is the distance traveled along grid line of length x. This relationship results in lines with linear scaling if sheet width around the wire is constant, as in a linear grid with a constant pitch, but can be extended to arbitrary shapes as well. Because constant solar flux j is assumed, we can simplify the expression to $\alpha \int_0^x g(\xi)\, d\xi = \beta I_{wire}$, where $\beta = j\alpha$, to see that the scaling law effectively maintains "a constant current flux in the grid line".

Serreze's and Scharlack's scaling law echoed a principle first stated some fifty years earlier, in 1926 by Murray, [92, 91]. The three relationships are shown in Figure 2.6. Murray noted that when blood veins intersect, or arteries diverge, in animals, the sum of the small vessels' radii cubed equals the radius cubed of the large vessel. That is, fluid flow networks in biological systems scale according to the relationship

$$\sum_{i \in inputs} r_i^3 = r_{output}^3 \tag{2.21}$$

The law also holds with *inputs* and *outputs* reversed, for diverging intersection. This result has been observationally verified as approximately correct across real and simulated large vascular systems,[130, 103] in plants,[83] and in slime mold colonies[6] to name a few. Murray scaling as been extended to optimize non-biological systems such as fuel cells,[124] heat transfer networks,[148] and microporous materials.[162] However, the true optimal power scaling relationship varies from one application to the next.

Murray derived the scaling result by balancing two power sinks: the work done by a fluid under the Hagen-Poiseuille fluid flow model[135] pf and a volumetric work term $b\,vol.$. With flow rate f, fluid viscosity η, vessel length l, and radius r. The total work done by the system is then

$$E = pf + b\,vol. = \frac{f^2 l 8\eta}{\pi r^4} + bl\pi r^2 \tag{2.22}$$

with a minimum at

$$b = \frac{2f^2 8\eta}{\pi^2 r^6}. \tag{2.23}$$

This establishes the relationship of optimal flow f to vessel radius r. Murray's law can be stated more generally to be:

$$f \propto r^a \tag{2.24}$$

Where the special case of $a = 3$ is true only under conditions of perfect laminar flow. The scaling law for work done by the flow component will vary for other flow regimes, and indeed, follow-on studies established that the power term a increases to ≈ 3.5 under turbulent conditions[143] and may vary locally based on the particular conditions throughout a network.[87, 134]

In the case of diffusive flow through a pipe, the resistance varies linearly with the cross-sectional area, so $a = 2$ and $f \propto r^2$. However, almost all literature on solar grid optimization assumes constant wire height. Under these conditions, the cross-section varies linearly with wire width w alone.

2.2.1 Murray's Law and Scharlack's Scaling are Equivalent

We verify these scaling rules in the present system of equations for the cases of constant and scaling wire height. Consider a small segment of wire of length δx with height h, width w, and resistivity ρ_{wire} that transports I amps of current. Resistive loss is $P_{resistor} = I^2\,R = I^2\frac{\rho_{wire}}{hw}\delta x$. Shadow loss from the wire footprint is $P_{shadow} = I(w)\,V = J_{sol}V_{op}w\,\delta x$.

The optimal wire width w^* can be defined as $w^* = argmin_w(I^2\frac{\rho_{wire}}{hw} + J_{sol}V_{op}w)\delta x$. The solution depends on how h scales with w. For the case of constant height,

$$0 = J_{sol}V_{op} - I^2\frac{\rho_{wire}}{hw^{*2}} \tag{2.25}$$

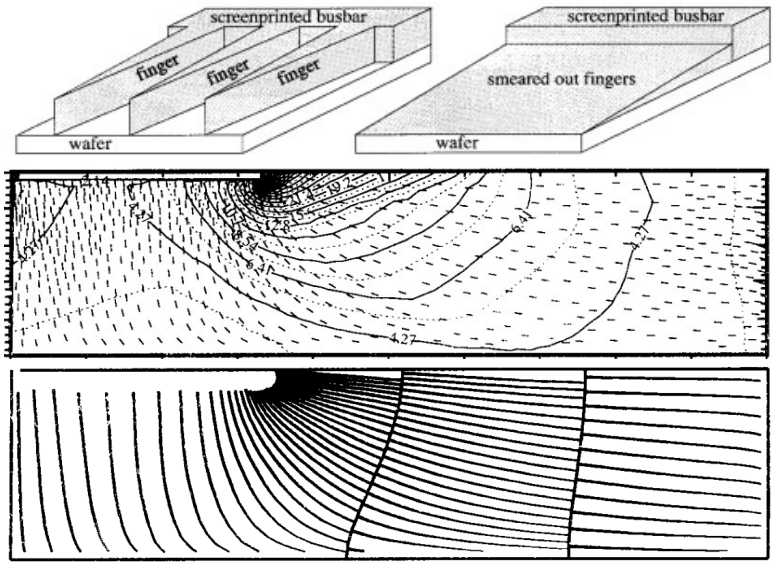


Figure 2.7: Grid routing method proposed by Burgers[26]. Optimal line scaling is established as if using Scharlack scaling [top] but is defined as a field across the solar cell based on current flux [middle]. Finally, traces are "painted in" to the field [bottom].

and

$$w^* = \sqrt{\frac{\rho_{wire}}{hJ_{sol}V_{op}}}I \tag{2.26}$$

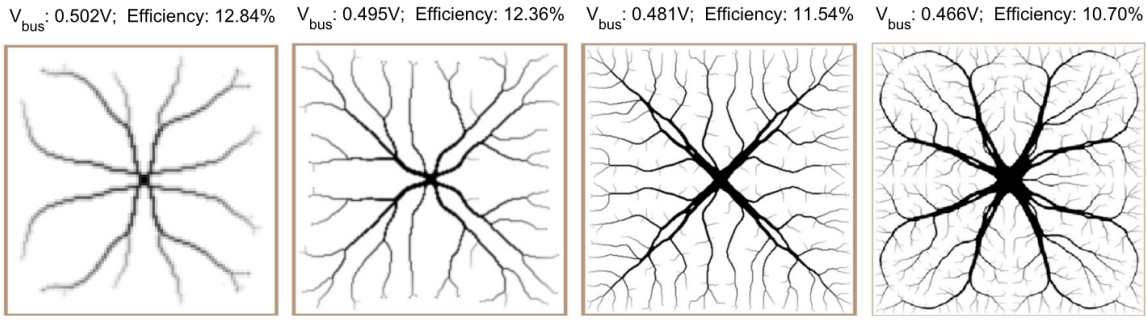
This recovers Scharlack's result of width scaling with current, maintaining an optimal current flux density in the wire of

$$J^*_{wire} = \frac{I}{w^*h} = \sqrt{\frac{J_{sol}V_{op}}{\rho_{wire}h}}. \tag{2.27}$$

An optimal network would maintain this flux density everywhere.

2.3 Line Placement

Most discussions of grid optimality thus far have existed in an abstract space where pitches and lengths are defined locally and extend over some variable length and width, if those macro quantities are delt with at all. The fiddly business of placing actual lines into a



(a) 1.5cm × 1.5cm (b) 3cm × 3cm (c) 6cm × 6cm (d) 9cm × 9cm

Figure 2.8: Examples of topology-optimized solar cell grids from Gupta et. al.[53]

discrete pattern has been left as a downstream exercise. Obviously, this cannot continue forever if the goal is the production of practical electrode patterns! How should one route grid lines across a real solar cell?

A few works have established methods for placing grid lines into arbitrary, concrete solar cell shapes. The most straightforward by Burgers[27, 26, 28] utilize a "smearing" technique illustrated in Figure 2.7 that first maps the optimal line widths across a solar cell as field, then paints lines manually, maintaining the optimal pitch-to-width ratio. More recent efforts by Gupta et. al.[53, 55, 54] use free-form topology optimization to generate the diffusion networks visualized in Figure 2.8.

These approaches make sense, but they also produce some apparent inefficiencies: the discontinuous cross-bars in the field approach, for instance, or the 'islands' of solar material in the grid lines of the topological patterns. But a more fatal flaw may also be present here because each of the above approaches carries the familiar assumption of constant wire height, and under constant h, the specific layout of grid lines does not matter.

2.3.1 Path Independence in Tapered Diffusion Networks

To illustrate the independence of path and power loss in a tapered diffusion network, consider again the infinitesimal slice of material from subsection 2.2.1. One such "leaf" of a solar

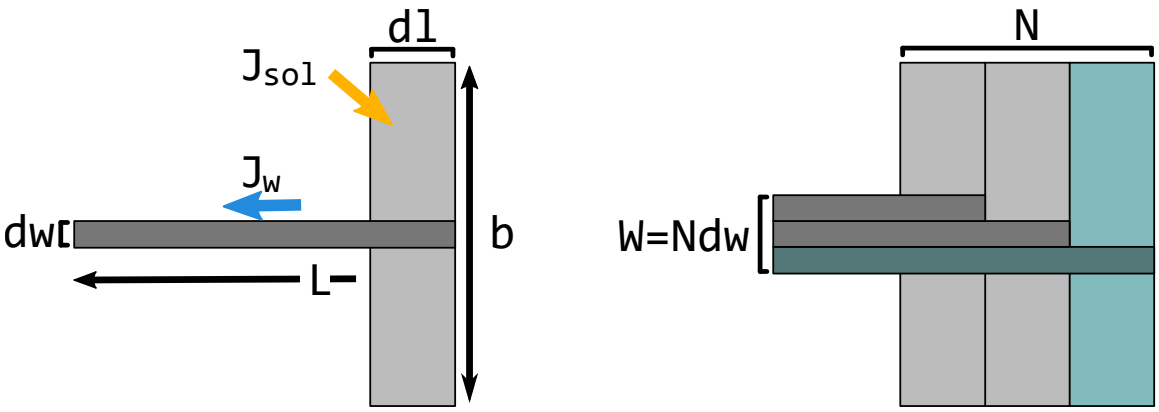


Figure 2.9: Leaves: a schematic of path-independent power loss in constant-height wires. Each "leaf" of active material (left) carves out a constant-width stem until it reaches the sink (right), regardless of the specific path taken.

cell and its grid line is illustrated in Figure 2.9. Assuming constant height h, the optimal width dw of the metal line "stem" of the leaf is set by Equation 2.26. The total current produced by the leaf is $I = bJ_{sol}dl$ and the resistance of the stem is set geometrically as $R_{wire} = h\,L\,dw$, where L is the distance from the leaf to the power sink. We can calculate the power loss in this system to be $P_{loss} = I^2R = b^2J_{sol}^2hLdl^2dw$ - the free parameters in the expression are the pitch b, and L.

When upstream or downstream leaves add their additional current to the grid line, they each supply their own "stem", scaled so as to maintain the optimal current density defined in Equation 2.27. Observe that for each leaf, the resistance of its own stem is unaffected by the stems of other leafs because the current density in each is the same everywhere, set only by the material parameters of the system. Therefore, the resistive losses of each leaf are independent of the overall routing of current through the solar cell. The only variable that matters is path length L!

2.3.2 Virtuous Scaling in Transport Networks

Path independence would seem to lift the burden of routing from a solar cell designer, but the above conditions above hold only for the linear-scaling constant-height case. That is, path independence is true in Murray networks per Equation 2.24 for the special case of $a = 1$. Constant height is a persistent assumption in solar cell design because metallization has been manufactured using deposition, evaporation, or subtractive processes for most of the field's history. These methods result in line heights that are independent of width.

However, many transport systems aimed at the gathering problem, including Murray's law biological networks and most of those shown in Figure 1.2, exhibit "virtuous scaling". That is, these systems have a coefficient $a > 1$, indicating that their transport elements become more efficient as they grow in width. This is true for fluid flow systems, but it is also likely to hold for volumetric engineered systems or additively fabricated solar cell grids.

Let us take the virtuous scaling case used in much of this work of $h \propto w$. This establishes a constant aspect ratio for the cross section of all wires in the system, and a Murray scaling factor of $a = 2$. For instance, by reworking the optimal width relationship for the special case of $h = 2$, we get:

$$0 = J_{sol}V_{op} - 2I^2\frac{\rho_{wire}}{w^{*3}} \tag{2.28}$$

$$w^* = (\frac{2\rho_{wire}}{J_{sol}V_{op}})^{1/3}\, I^{2/3} \tag{2.29}$$

In this case, Scharlack's scaling result no longer holds. $J^*_{wire} = \frac{I}{w^{*2}} = (\frac{J_{sol}V_{op}}{2\rho_{wire}})^{2/3}\, I^{-1/3}$. Optimal flux now varies throughout the network as a function of current, and the operating current flux is actually reduced at larger current loads and for larger grid lines. Under these virtuous scaling conditions, the routing problem is on the table again, and we can begin to consider approaches to find an optimal grid line layout.

Chapter 3

CELLULAR GRID OPTIMIZATION

In the previous chapter, subsection 2.3.2, it was noted that the problem of laying out constant-height grids with tapered lines on a solar cell is easily solved. The path doesn't matter, so all current should proceed directly to the sink as quickly as possible! However, constant height is a poor assumption when it comes to additively manufactured conductive grids - these structures are likely to exhibit virtuous scaling. Virtuous scaling is the term in this report that refers to networks that become more efficient transporters as they increase in size or cross-section. I.e., their Murray scaling constant $a > 1$. Metal electrodes are likely to have this property if they are manufactured additively for two reasons:

First, it is often the case in additive manufacturing that a larger feature height is possible for features with a wider footprint on a substrate. This is simply a physical effect: a larger footprint can support larger features. I.e. for a deposited layer having a set angle of repose on the substrate, we would expect cross-section area $A \propto w$ roughly, where w is feature width. This effect will also be observed when stringing together multiple manufacturing techniques to bridge multiple feature size scales: screen printing (a sub-mm-scale process[8, 42]) can support taller features than μm-scale electrohydrodynamic inkjet (EHD). (EHD printing is discussed in chapter 4.) A manufacturing path that used both of these methods could utilize wide but high-aspect-ratio screen printed lines with low-aspect-ratio EHD-printed fine grid lines.

Second, in addition to the variable height of additive features, the quality of printed material may vary with line width. Many metal inks utilize nanoparticles which are fused by sintering.[137] The effective conductivity of the material in such structures will often increase with width, as open-circuit breaks in the percolating network of nanoparticles are less likely to dominate performance when the line width is larger.

Thus, both the effective material properties and cross-section area are likely to improve as

wires grow in width. Under these virtuous scaling conditions, an intractable and interesting problem arises: what is the best grid line routing across a solar cell to minimize shadowing and resistive losses?

3.0.1 Disambiguation: Vertices, Cells, and Elements

Throughout this chapter, I will build up a model for simulating and optimizing the layout of solar cell grid lines. The smallest functional pieces of this model will at times be referred to as vertices, cells, or elements. Each term is, in some sense correct. In the graph theoretical sense, the "vertices" of a graph are the space-embedded elements that are connected to each other by edges, describing the foundational connectivity of the model. "Cells" refer to the small decision-making units in a cellular automaton simulation, which is one valid description of the model, as well as being a direct analogy to the biological cells that solve the gathering problem in nature. And "element" refers in the field of finite-element analysis to a representation of a small piece of material that simulates local physics. I will generally call upon the term that is germane to the field of study being discussed in each given section. It should be understood, however, that the three terms can generally be treated as interchangeable within this chapter.

Where biological cells or solar cells are being discussed, I will use the full term. Likewise, I will forgo the use of some other tempting nouns such as "node" and "unit" to describe the elements/vertices/cells of the simulation. God help us if cell phones enter the picture.

Similarly, "grid" will refer to the real or simulated front metal electrode construct as a whole, "graph" will refer to the underlying graphical data structure being used to model the grid's connectivity, and "solar cell model" will refer to the entire simulated system, including the current-generating active material, the grid, and the sinks.

3.1 Constructal Theory

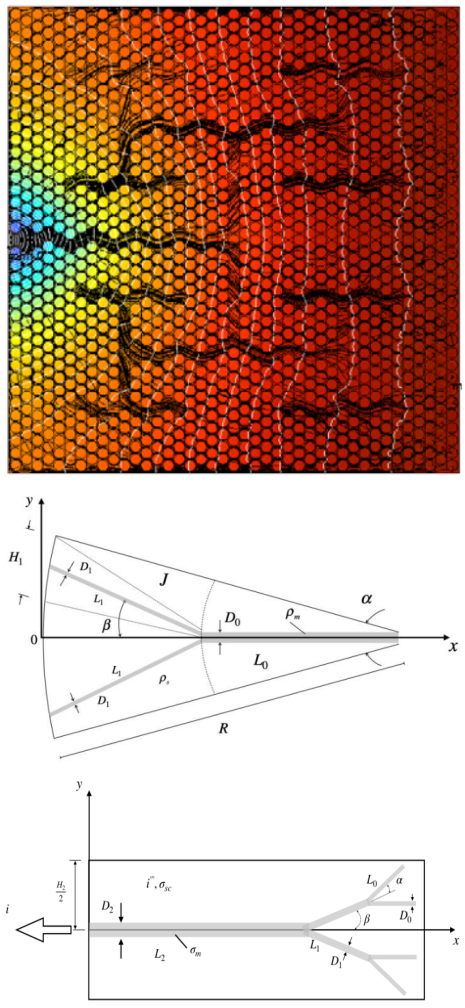


Figure 3.1: Examples of constructally optimized solar cell grids. [Top] by [88]; [Middle] by [101]; and [Bottom] by [100]

One general solution to routing transport networks through a volume was first presented by Adrian Bejan in 1996 and '97.[12, 13] Termed constructal theory, the framework provides a set of guidelines for designing flow systems that solve the gathering problem - some examples of constructal structures can be seen in Figure 3.1. A complete introduction to the field can be found in Appendix B

It is notable that many of the natural systems that seem to conform to constructal theory are comprised of cells that have no view of the larger transport problem but act only based on local stimuli. In this chapter, instead of a top-down strategy, a less principled but more generalizable method was used to optimize the specific layout of solar cell front grids. The general equations from chapter 2 were modified to describe finite elements, and a dynamic directed graph model of current transport was layered on top of the solar cell model that used backpropagation to distribute a reward across the graph and greedy hyper-local decision making to evolve the network over successive generations. The resulting patterns were not optimal but possessed a familiar, self-consistent, ramified topology and appeared to be highly performant as solar cell grids.

3.2 The Grid-Graph Model of Solar Cell Electrodes

A model to explore ramified diffusion networks for solar cell front electrode application was built up in three parts. First, a system of equations is defined that describes the behavior of solar cell current and power loss in a finite element. Second, a routine for calculating total power output from directed graphs of finite elements was tested. Finally, a greedy algorithm for optimizing the graph to increase power output was outlined. The software package that implements this collection of algorithms is called "GridGraph".

3.2.1 Finite Elements: Solar Cell and Stem

A general framework for modeling solar cell grid lines is attractive for its portability and adaptability. Here, the numerical expressions used previously to optimize large grid networks were adapted to describe single finite elements. Instead of the standard boundary conditions of finite element analysis, the electrode's connectivity was described as a directed graph, and wire connections as edges in the graph. Among other things, this made it easy to optimize the network with always-optimal grid lines that displayed virtuous-scaling transport properties.

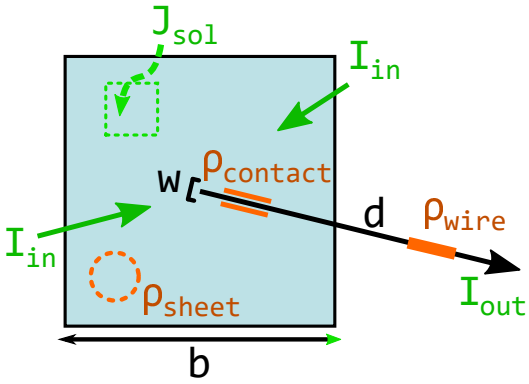


Figure 3.2: A single square element in a cellular grid model.

The following system of equations governs the current generated and power lost in a square finite element with side length b:

$$I_{generated} = J_{sol} b^2 \tag{3.1}$$

$$P_{shadow} = J_{sol} V_{op} w d \tag{3.2}$$

$$P_{sheet} = \frac{1}{12} J_{sol}^2 \rho_{sheet} b^4 \tag{3.3}$$

$$P_{contact} = I_{generated}^2 \frac{\rho_{contact}}{bw} \tag{3.4}$$

$$P_{grid} = (I_{out})^2 \frac{\rho_{wire} d}{wh} \tag{3.5}$$

where d is the length of the wire 'stem' that connects the element to the next downstream element in the grid. For a regular square mesh, $d = a$. In practice, the contact resistance is omitted from models and optimizations that follow. Contact losses make up a minimal proportion of practical solar cells,[85] and their omission improves the readability of the model and code with minimal impact on grid designs. However, as will be seen, this and any other loss source can readily be added back into the model as long as they can be expressed as an analytical or differentiable function.

Because current will daisy-chain many times on the path to a sink, each element accepts current from any upstream elements, adds its own generated solar current, and passes the sum

$$I_{out} = \sum_{i \in inputs} I_{out}^i + I_{generated} \tag{3.6}$$

to its downstream neighbor. For cases where the elements are not square, the sheet resistance is estimated using a square of equivalent area A: Equation 3.1 becomes $J_{sol} A$ and Equation 3.3 becomes $\frac{1}{12} J_{sol}^2 \rho_{sheet} A^2$. Most critically, the width w of the grid line stem is a function of local current - it scales according to the known optimal point determined by material properties and the Murray line scaling parameter a, as discussed in section 2.3. For a constant-height $a = 1$ line, this would be the analytical relationship in Equation 2.26. For a constant-cross section $a = 2$ line, one would use Equation 2.29. In both cases, the current that determines optimal line scaling is the total local current I_{out}.

3.2.2 The Solar Graph Model

A graph is a mathematical object comprised of a set of vertices V and edges E. Each vertex $v \in V$ represents one of a class of objects which may have locations, labels, or properties.

Each edge $e \in E$ "connects" two vertices in the graph, and may have properties of its own. (A graph where multiple edges are allowed between the same two vertices is a multigraph, and where edges may connect more than two vertices is a hypergraph. Neither of these figures into the present model!) An edge $e = \{v_i, v_j\}$ in an undirected graph is a simple unordered set of two vertices, representing a reversible association between v_i and v_j. By contrast, an edge $e = (v_i, v_j)$ in a directed graph represents an ordered pair of two vertices. Here v_i is the "head" and v_j the "tail" of the edge. [141, 151]

The graphical model for a solar cell front grid electrode takes the form of a directed graph with each vertex representing a finite element of solar cell real estate, as described above in subsection 3.2.1, and each edge representing the metal "stem" of one element. So here, the edge (v_i, v_j) is the grid line carrying current from element i to element j. Each vertex is the head of exactly one edge, though it may be the tail of several edges, representing a set of multiple upstream elements. Formally, the set of inputs from Equation 3.6 for vertex j in the grid defined by the directed graph $G = (V, E)$ will be the set of all vertices U_j for which $(v_{i \in U}, v_j) \in E$.

Debt-Passing Model of Solar Cell Power

To calculate the power output of the solar cell modeled by the graph $G = (V, E)$, a "debt-passing" model is deployed that tracks two conserved quantities within the graph: electric current I and power debt P^d. Each vertex in the model is a finite element that generates some quantity of current and power loss, both of which are accumulated in the network. An element-wise update function is defined in algorithm 1 that updates the total local current I_j and power debt P_j^d at element j.

Algorithm 1 Update of element j

Given: U_j are the upstream neighbors of vertex j

$I_j \leftarrow \sum_{i \in U_j} I_i + I_j^{generated}$

$P_j^d \leftarrow \sum_{i \in U_j} P_i^d + P_j^{loss}$

In algorithm 1, $I_j^{generated}$ is the same as Equation 3.1 and P_j^{loss} is the sum of all local

sources of power loss, i.e. Equation 3.2, Equation 3.3, and Equation 3.5. Algorithm 2 then determines the power output from a grid.

Algorithm 2 Power Output of a Solar Graph Model

Given: V_{op}, $sinks$ ▷ Operating voltage
$T \leftarrow 0$ ▷ Total power
for $s \in$ sinks **do**
  $Q \leftarrow walk(s)$
  for $E_j \in reverse(Q)$ **do** ▷ E_j is one vertex element
    update(E_j) ▷ as per algorithm 1
  end for
  $T \leftarrow T + I_s V_{op} - P_s^d$
end for
Return: T

The *walk* operation is any common traversal strategy of the tree rooted at the power sink s that travels *upstream* in the grid G, returning the vertices in the order they are encountered. (Depth-first or breadth-first will both work.) Processing element updates in the reverse order guarantees that upstream vertices are always updated prior to downstream vertices - i.e., the sink s will be the final vertex updated.

Note that V_{op}, the cell's operating voltage, must be chosen arbitrarily and that its only practical effect in the model is to set the relative reward from electrical current vs the cost of power losses. A physics-based approach to choosing V_{op} is presented later, in subsection C.1.2. For now, however, this framework will serve to illustrate approaches to optimizing the solar cell grid to maximize the total return power T.

The grid was optimized using greedy local current routing updates at the cellular level, each vertex "choosing" where to send its allotment of current to maximize power returns. Cells in the grid made their routing decisions based on a localized power return metric that was calculated using gradient back-propagation of the reward T through the network.

3.2.3 Greedy Grid Optimization

Optimizing the solar cell graph comes down to choosing, for each element in the simulation, the best downstream element to route current towards. As with biological analogs to the task, such as tree roots and slime mold colonies, the algorithm petitions each cell in the system to make its own locally-optimal decisions. Specifically, each cell will direct its stem (i.e., a directed edge in the graph leading from it to one of its neighbors) to the neighboring cell that offers the greatest incremental power return. Node j with neighbors N_j will select target k if $k = argmax_{i \in N_j}(\frac{\partial T}{\partial I_i})$. $\frac{\partial T}{\partial I_i}$ is the incremental increase in total power T when a small amount of current is added to cell i. The implementation of this process is algorithm 3.

Algorithm 3 Power-Greedy Update to a Solar Cell Graph

Do: Calculate model power output ▷ per algorithm 2

Given: $Q \leftarrow walk(s), V_{op}$ ▷ from algorithm 2

for $E_j \in Q$ **do** ▷ Update gradients

  if E_j is a sink **then**

    $\partial T_j = V_{op}$

  else if E_j has target k **then**

    $\partial T_j = \partial T_k - \frac{\partial}{\partial I} P_j^{loss}$

  else

    $\partial T_j = 0$

  end if

end for

for $E_j \in Q$ **do** ▷ Update targets

  if E_j is not a sink **then**

    $target_j \leftarrow argmax_{k \in N_j} \partial T_k$

  end if

end for

P_j^{loss} is the power lost locally and is a function of both local current and the length

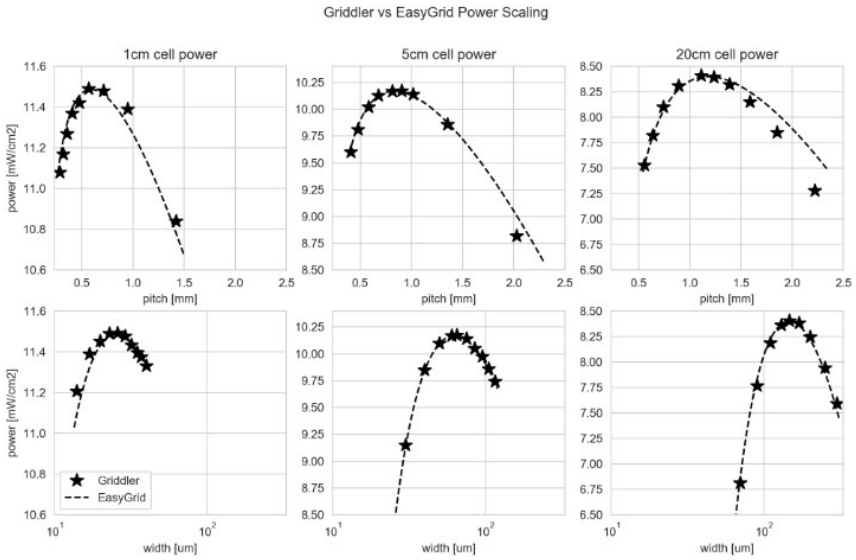


Figure 3.3: Comparison of solar cell power predictions for H-Bar geometries by Griddler and the grid simulation described here. The models agree on the optimal H-Bar design in terms of grid pitch and width for solar cells of 1cm, 5cm, and 20cm side lengths.

of the current stem, i.e. the distance between vertices j and k. To search for an optimal grid layout, 3 is run repeatedly until convergence, when two subsequent updates produce no target changes. A description of the software to implement this routine can be found in Appendix C.

3.2.4 Validation

Unlike the isotropic graphs examined in chapter 2, there is no straightforward path to independently verify a virtuously scaling grid pattern in software. Not, at least, with the same Griddler package as used before. The changes in effective wire height are modeled as changes to the material properties of the 2D elements used by the package, and the number of different values the metal lines take on exceeds the capability of the software. Instead, constant-height H-bar patterns are simulated in Griddler and GridGraph with the same underlying material properties and varying geometries. The results of these comparisons

are shown in Figure 3.3.

After applying a constant scaling factor to account for the lack of voltage variance in the GridGraph model, there was very close agreement between the models. The models agreed on the optimal pitch and width values for 1-, 5-, and 20-cm length square solar cells, and they showed similar scaling of power output as width and pitch diverged from the optimum by a factor of ± 2.

3.2.5 Discussion

How much benefit can we expect to see in solar cell output when switching from a classical H-Bar configuration to the free-form grids produced by the GridGraph optimization loop? Table 3.2.5 has a comparison between the two patterns for square solar cells, between 1- and 10-cm in side length, with a single edge-centered sink. The patterns show up to a 1.0% increase in total power output. As noted in chapter 2, a 1% increase in solar power output would have corresponded to about $389 million in utility-scale power in the US in 2020. These numbers are almost sure to increase year over year as clean energy initiatives gain momentum. Therefore the improvements in solar cell efficiency demonstrated by the switch from straight-line to free-form ramified grid network are small but significant.

Table 3.1: Comparison of Power Outputs from Tapered H-Bars vs Graphs in GridGraph Simulation of Square Solar Cells.

Length	H-Bars	GridGraph	Increase
cm	W	W	%
1	0.0144	0.0145	0.7%
2	0.0555	0.0557	0.4%
5	0.3253	0.3287	1.0%
10	1.2213	1.2331	1.0%

Several example grid solutions are presented in Figure 3.4. The appearance of the grids is visually familiar, conjuring comparisons to the scaling ramified networks presented back

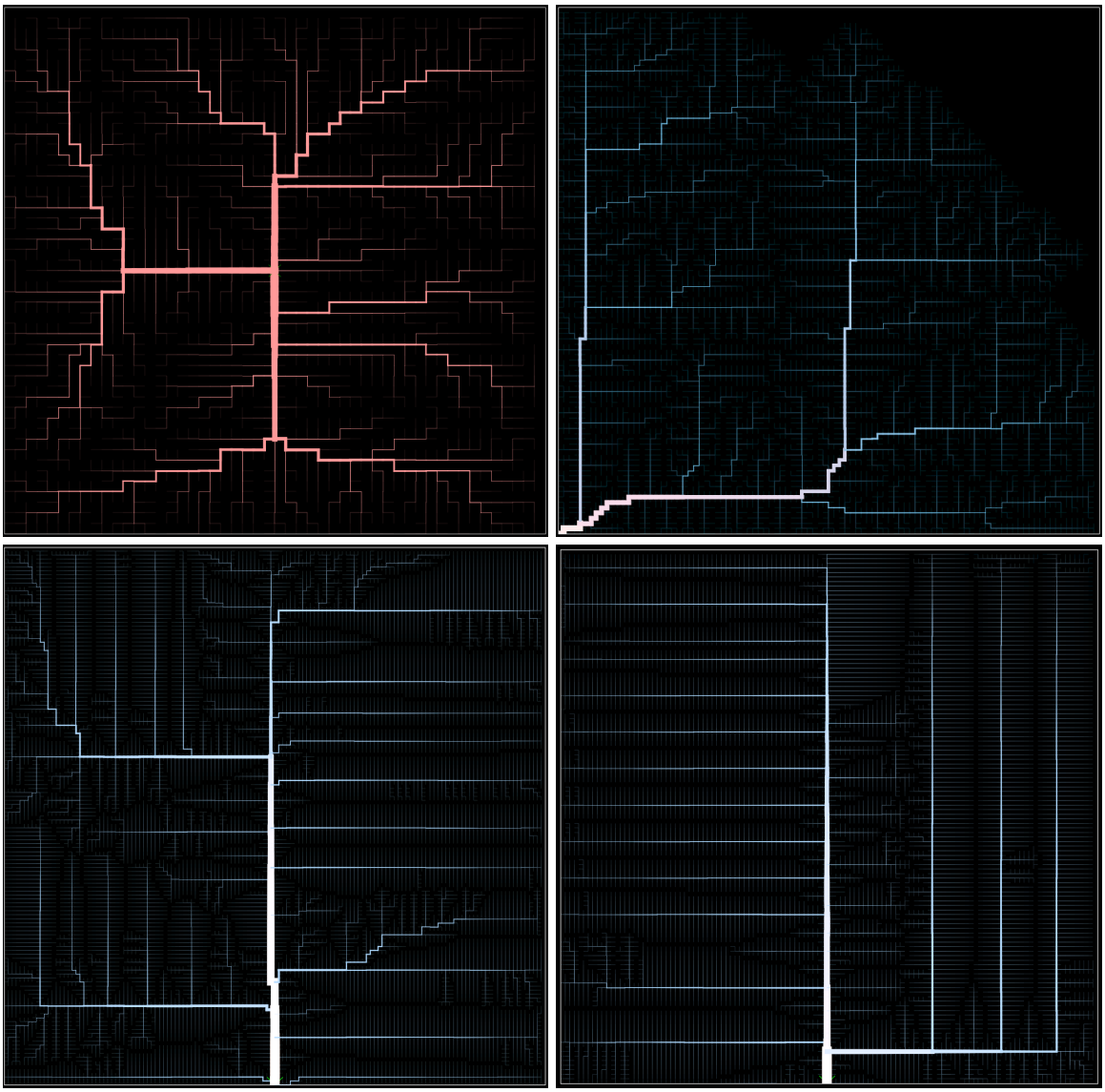


Figure 3.4: Examples of grid patterns generated by the GridGraph simulation with center [Top Left], corner [Top Right], and edge-located sinks [Bot. Left]. [Bot. Right] is a pattern generated using the 2-diode physics model described in subsection C.1.2; the remaining patterns use the debt-passing model described above.

in Figure 1.2. Some subjective observations can be drawn from these patterns. It is common in all patterns that a main trunk or bus line forms that carries the majority of the inbound current to the sink. The trunk tends to be centrally located in the cell and to accept periodic inputs from second-tier tributaries. The tributaries appear to intersect the trunk lines at relatively regular intervals, setting up the appearance of a fractal relationship between three or four tiers of grid lines: the smallest lines spaced by element size, the second and possibly third-tier tributaries, and then the third- or fourth-tier main bus. These emergent, tiered relationships are consistent with the multi-order constructs espoused by constructal theory.

However, many patterns also contain more than one set of length scales. In particular, the bottom two patterns in Figure 3.4 have different behavior regimes on their left- and right-hand sides. The apparent chirality of these patterns probably arises from a tie-breaking convention in the greedy optimization loop (algorithm 3) that prioritizes neighbor vertices in the order they were initialized in the event of two neighbors presenting exactly equal power reward. This mechanism results in different orientations of the tributary layer of lines, and subsequently can result in multiple fractal-like relationships within the same pattern.

All four patterns shown here were optimized on a square-layout mesh with 4-neighbor connectivity, which restricted the patterns to a certain regime of right-angle intersections and city-like layouts. We see some indication in the resulting patterns that the choice of static mesh is a limiting factor in the performance of the grids. In particular, a characteristic asymptotic line shape emerges in multiple grid patterns. See, in Figure 3.4, the top-right image, bottom right corner. This characteristic curve shows up in other patterns shown here and at different scales. A similar curve can be found in most of the natural patterns shown in Figure 1.2. The square mesh is strictly sub-optimal for this shape - it results in inflated path length, increasing both resistive and shadow losses. The model chooses this shape anyway, suggesting that there is a compelling advantage to the curve. A mesh that could better accommodate such curves should result in better grid performance.

3.2.6 The Routing Problem is Hard

Table 3.2: GridGraph size vs. estimated time to brute force optimal graph solutions

N	Elements	# Graphs	Est. Time
2	4	8	6 ms
3	9	1728	0.8 s
4	16	8,957,952	79 minutes
5	25	7.4E+11	18 years
6	36	9.9E+17	34,600,000 years
7	49	2.1E+25	1E15 years

Repeated runs with the graph optimization loop quickly reveal that repeated optimization attempts can yield different results, especially with randomized starting connections. Each routing decision made by the model is "sticky", as illustrated by Figure 3.5. Thus routing decisions made early in the simulation can lock in the major watersheds and trunk locations for the rest of the optimization run. While some grid solutions may appear obviously bad - for instance, if all nodes are not connected to a sink - there is no obvious test for the optimality of any given pattern except by comparing it against every other possible pattern. This problem, therefore, has the hallmark properties of NP-Hardness. That is, the problem appears to be both unsolvable and unverifiable in polynomial time, as it scales with the number of vertices in the simulation.

3.2.7 Alternate Optimizer: Brute Force Search

If the only way to find the true optimal arrangement of grid lines is to try all combinations, why not go ahead and do that? The futility of the exercise may be obvious to someone who has worked with exponentially scaling problems before, but it is entertaining to observe just how quickly this problem explodes in scope.

Table 3.2.7 shows the number of possible grids it is possible to generate on a square mesh with 4-neighbor connectivity when there are $N = 1, 2, ..., 7$ elements per side. For the software as-implemented, a 4-element graph can test one grid layout in just under 1 ms

on a consumer-grade desktop computer. Assuming that rate holds for much larger graphs, the time to explore the full space of possible grids grows to a geological time scale for the 6-by-6 and is larger than the age of the universe for the 7-by-7. Considering that some of the patterns shown in Figure 3.4 are over 100 elements to a side, it should be apparent that brute force search is not a feasible approach to the problem for solar cells with non-trivial surface area. Unless a first-principles optimality proof for the gathering problem is discovered, we should consider true optimal solutions to be un-provable. The best we can do for this project, and possibly ever, is to search for incremental improvements. An account of some alternate solvers and models that were developed for GridGraph can be found in section C.1

3.3 Ramified Grids - What Comes Next?

Work to date showed that the open-ended optimization of the solar cell front grid can produce small but meaningful improvements to the overall power output of solar cells, on the order of 1%. The greedy cellular optimization method presented here resulted in performant, ramified electrode patterns, and the software is extensible, allowing the introduction of different physics and optimization methods. Longer term, the model's exploration capabilities could be improved, and extensions to the framework could allow applications to other areas of research with larger potential benefits. A more detailed list of potential extensions to the work can be found in Appendix D.

Having created solar cell grid patterns that should outperform the current art, an obvious question arises. Can these patterns actually be put to practical use? Are they even manufacturable? The final sections of this report attempted to answer these questions.

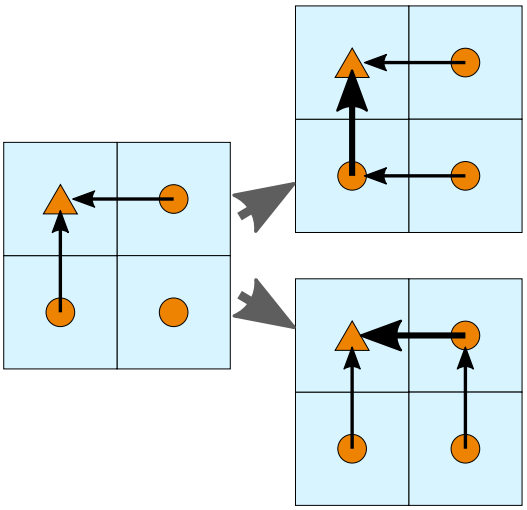


Figure 3.5: Four element example of the "stickiness" of the graph optimization walk with virtuous scaling. The triangle represents a power sink. The bottom right element can get equal power return by going left or right, but once the decision is made the other choice becomes much less desirable. Because of this, every cellular routing decision is self-reinforcing.

Chapter 4

CONTROLLED-SIZE PRINTING WITH THE ELECTROHYDRODYNAMIC INKJET PRINTER

In the previous chapter, an optimization algorithm was presented that generated detailed ramified transport grids which could serve as high-performance solar cell front electrodes. These networks were generated - and indeed are only helpful - under conditions of virtuous line scaling such as is often found in additively manufactured metal lines. In chapter 2, the case for very small line widths was made, with wires down to the micron scale offering benefits insofar as they allow solar cell design to reduce or remove dependence on transparent conductive materials. To truly make use of these combined requirements, an additive manufacturing technique for grids of metal lines would need to be capable of printing metal lines with controlled, on-demand thicknesses, down to the micron length scale. One of the only contemporary technologies that can perform in this space is electrohydrodynamic (EHD) inkjet printing.

4.1 Introduction to Electrohydrodynamic Inkjet Printing

EHD is an emerging additive manufacturing technology that utilizes a controlled voltage at the tip of an inkjet nozzle to create very small drop sizes. Unique among inkjet methods, EHD is capable of micron- and sub-micron-scale printing, enabling additive affordable creation of optically active 2D surfaces, micron-scale electronics and interconnects, and biomedical devices.[106, 38, 86]

Capable of printing at micron-and-below size scales, electrohydrodynamic inkjet (EHD) has the potential to revolutionize the rapid additive manufacturing of electronic, optical, and biomedical devices. However, the parametric complexity of EHD processes, the inherent noisiness of open-bed processing, and the high levels of precision and modulated control required in many applications prevent EHD processing from finding commercial manufacturing applications.

4.1.1 EHD Dynamics

The charge/discharge and ink drop formation physics at the EHD tip is a complex dynamic process, and successes in in-situ monitoring[132, 161] and process modeling[66, 36] have been unable to extend to the small micron-scale end of the EHD process window. The complexities of system modeling derive largely from the dominance of surface tension energy at the small length scales where EHD operates. To overcome this energy and form extremely small femtoliter-scale drops, the EHD printer makes use of a Taylor cone, whereby a protrusion of the ink itself serves as a field concentrator, allowing it to "punch through" the surface energy barrier.[62, 58]

The interplay of surface and electric forces at a curved liquid/air interface is central to the main operating principle of the EHD. As the fluid collects charge, an increasing electric field encourages it to move towards the grounded substrate of the printer. Eventually, the magnitude of this electric force at the point of greatest curvature (where field line divergence leads to the largest local field gradient) will exceed the surface tension force and cause the formation of a conical extrusion known as a Taylor cone. The effect is self-amplifying, and the Taylor cone grows until it contacts the substrate. The resulting discharge momentarily damps the electric driving force, causing the Taylor cone to collapse and the process to start over. This cyclic behavior is often referred to as the "pulsating jet" or "pulsed cone" mode of EHD operation, and arises over many continuous usage conditions.[69, 37, 74, 7, 111] The specifics of drop formation may vary when nanoparticles are the charge-bearing elements in the ink instead of a polarizable solvent,[120] but the cyclic behavior typically remains. Any physics model that claims to predict or reconstruct the physics of an EHD print head under practical manufacturing conditions needs to capture these dynamic effects.

4.1.2 Modeling the EHD Process

Several studies have identified numerical relationships or scaling laws in EHD print systems. In particular, Chen et. al.[32] and Choi et. al.[36] described scaling laws for the pulsed jet deposition mode of EHD printing. Park et. al.[106] and Qian et. al.[113] used numerical and correlational models to predict feature sizes from voltage inputs, with subjectively good

agreement between prediction and result when equipment parameters were held constant.

Several recent studies have also employed finite element or CFD methods to model the evolution of single EHD drop or jet formation cycles through time.[155, 48, 108] The model used in this work was a 2-dimensional CFD model by Jiang is constructed in COMSOL Multiphysics and resolves the evolution of a non-compressible fluid over time while under the influence of five forces: gravity, nozzle pressure, viscosity, surface tension, and electric field. The model enforced continuity and momentum conservation and used Maxwell equations to calculate electrical stresses. A detailed accounting of the model can be seen in [66].

The COMSOL model used a "leaky dielectric" assumption whereby free charges gather near the surface of the charge liquid and employed a boundary layer with gradient properties between liquid and free space to model ink geometry evolution. Only a single Taylor cone formation cycle was provided, as the model does not provide for continual removal of deposited ink. (In reality, substrate velocity clears the print bed for subsequent drops.) While this model provided a near-physically accurate dataset for modeling single drop dynamics, it was not suitable to capture the long-term natural dynamics of multiple drops in an actual print run.

EHD Dynamic Modeling

Properly instrumented, additive processes can provide a wealth of data to drive machine learning (ML) modeling, prediction, and control; the EHD is no exception and may be monitored in situ via signal characterization, imaging, and laser scattering. To capture the dynamics of the EHD process, a machine learning framework that could learn to predict the forward evolution of the EHD system from process data would be tremendously valuable. The perfect state space knowledge of CFD models presents an opportunity to construct data-driven models that rely and thrive on high-dimensional state space knowledge.

Several frameworks have emerged in the study and control of fluid flows – two notable examples are dynamic mode decomposition (DMD)[56] and sparse identification of nonlinear dynamics (SINDY)[23], which provide structured frameworks for learning dynamics operators in dynamical systems. The EHD system incorporates several independent control

inputs, with tip voltage being the primary actuation, motivating the use of control variants of DMD and SINDy[24, 112]. Such dynamical models have never been applied to an EHD process. In this chapter, DMD was evaluated on the task of learning and recreating the detailed dynamics of the EHD printer tip.

DMD is an algorithm for identifying the principle orthogonal components in a time-series dataset and their linear dynamics. DMDc extends the method to admit independent inputs (actuation) to the dynamic system – in the case of EHD, the controlled tip voltage is the main actuator. Because DMD and DMDc require knowledge of the entire state space (i.e., momentum at every point in space in a fluid system), the outputs of fluid dynamics simulations of an EHD print head were used in lieu of real process data. The goal of the project was to successfully reconstruct the fluid ink column at the EHD print head using DMD modes and dynamics, and again with DMDc. If successful, this model would have served as a launching point to develop similar models with sparse, in-situ observable inputs from real-world EHD runs. Unfortunately, as will be seen, critical gaps in the capabilities of DMD to manage nonlinear time dynamics and variable systems prevented the algorithm from finding footing in EHD modeling.

An introduction to DMD, DMDc, and the methods chosen to find dynamic modes and reconstruct state vectors in this work can be found in Appendix E.

4.2 Methods: EHD Dynamics Data

Ultimately, the goal of modeling the EHD process is to allow rapid exploration of input waveforms to produce controlled line thickness results. This capability would allow the EHD to print complex patterns in a single pass, ensuring good feature alignment. Examples of functional patterns that require this capability are shown in Figure 4.1. The dynamic system machine learning approach would have the added benefit of revealing the underlying physics of the system while producing results with far lower computational overhead than a full CFD model, but it required a corpus of training data. Three sources of data were tapped to train the dynamic model, though only two were ultimately used. A comparison of high-speed photography and the 2D model frame of an EHD mid-jet to the flattened state vector used in this study can be seen in Figure 4.2.

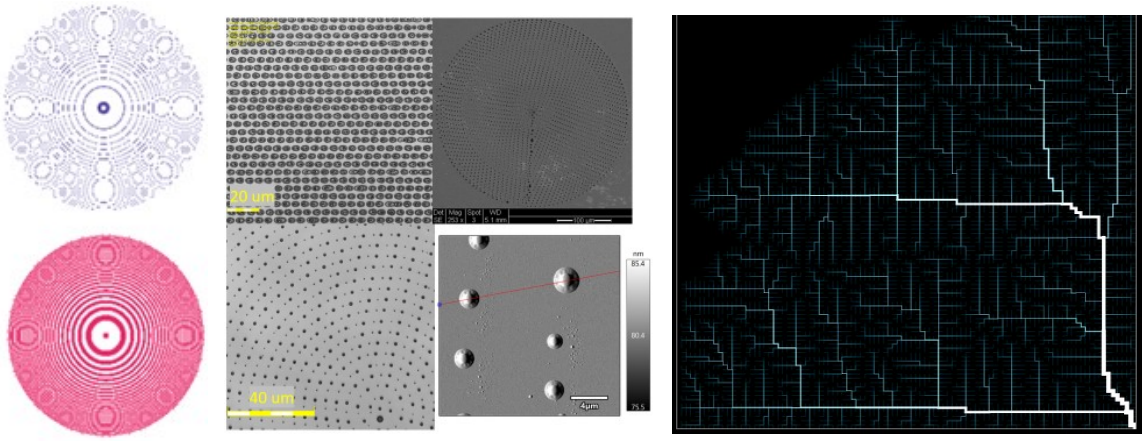


Figure 4.1: Examples of target micron-scale pattern capabilities with the EHD. Left: controlled-diameter dots for metasurface lensing. Right: controlled-width metal lines for a ramified charge transport network solar cell front grid.

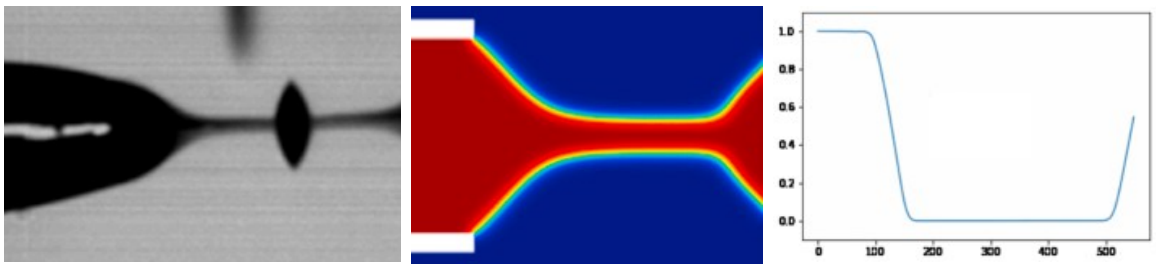


Figure 4.2: Comparison of photographic, 2D[66], and flattened 1D representations of EHD jetting.

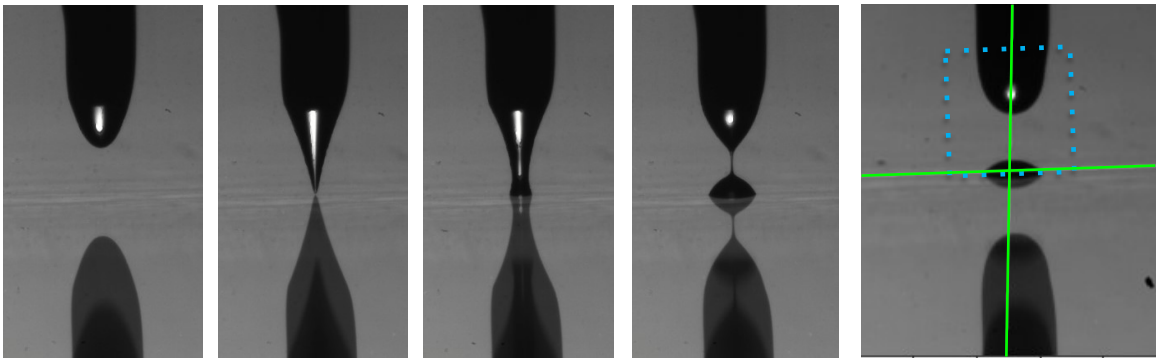


Figure 4.3: High-speed images of a drop being ejected from an EHD tip over about 0.5 seconds. In the final frame, image processing elements are shown such as the substrate mirror plane (horizontal line), the drop axis centerline (vertical line), and the jet measurement region of interest (blue rectangle). Image Credit: Liangkui Jiang

4.2.1 EHD High-Speed Videos

A series of high-speed videos of ink drop ejection from an EHD tip were captured in the Qin lab at Iowa State University – the ink and 100 um tip appear dark against a light background in these images and the quartz substrate is a mirror, as shown in the figure above. An image analysis pipeline was written to measure the width of the ink column along the z-axis that runs from the print tip to the substrate in each frame of each video. The square of the width is a relative measure of mass distribution along the vertical axis – the series of measurements of mass over time and height comprise the state input training data for the DMD model.

The analysis allowed measurements to be taken of the ink column, some of which are listed in the table below. Jet duration, velocity, and thickness, along with the geometry of the deposited drop, all varied with applied voltage. Interestingly, although drop velocity generally increased with higher voltage as expected, the remaining parameters showed complicated variation with the applied voltage, with the largest drops being deposited at the lowest voltage, and the highest setting of 1200 V exhibiting multiple droplets firing in

DC	Height	Velocity	Duration	Jet Thickness	Drop Height	Drop Width
V	um	cm/s	s	um	um	um
800	96.50	18.87	0.351	32.5	20.50	79.48
800	106.81	18.30	0.295	28.4	18.09	67.24
800	100.72	18.14	0.296	28.2	17.92	67.39
1000	121.00	19.48	0.196	12.8	11.00	45.85
1000	119.20	14.81	0.184	12.8	10.09	40.77
1200	122.78	20.23	multi	16.2	10.47	56.19
1200	124.47	20.35	multi	18.2	16.28	74.69
1200	113.27	20.23	multi	20.0	11.42	61.63

Table 4.1: Measurements from image analysis of high-speed video of EHD jet events

quick succession. The results emphasized the need for models to unpack the dynamics of the system, and for those models to correctly capture pulsed-jet-mode printing events.

While these high-speed images provided rich dynamical data, they raise a critical problem for training DMD models: small variances in the geometry of each experiment vis. the distance and angle between the print tip and substrate mean that no two videos yielded state vectors of the same length. From the perspective of a DMD model, each video would need to be a completely new dynamical system, precluding the transfer of a single DMDc model between runs with different control inputs. For this reason, the high-speed images were first used as reference footage to construct finite element CFD simulations of the jetting process, and the synthetic datasets were used in the first attempt at fitting DMD models. Appendix F details the FEA models that generated DMD training time series data.

4.2.2 Evaluation of DMD Modeling

Successful dynamics models would be able to "play forward" each training CFD dataset from only the first frame, using a compressed dynamics representation. A DMD model attempts to find an operator A that describes the time evolution of state X of a dynamical

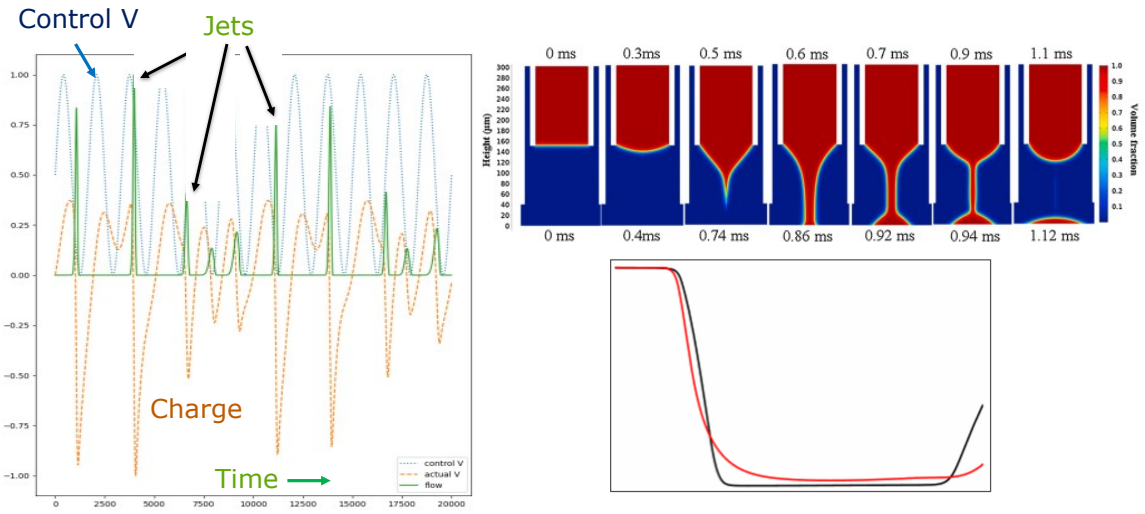


Figure 4.4: Computational models of EHD jetting. Left: 1-dimensional multi-jet fluid model. Upper right: 2-D polar-symmetric CFD model from Jiang et. al.[66] Lower right: DMD model (red) and 2D CFD model (black) state comparison, showing small errors in DMD reconstruction.

system, i.e. $X' \approx AX$. DMD identifies this operator in a subspace defined by the principal dynamic components of X, identified by performing an eigenvalue decomposition on the dynamics of the system, then casting the dynamics into subspace $\tilde{A}$:

$$X \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V}^{*}\widetilde{A} \approx \widetilde{U}X'\widetilde{V}\widetilde{\Sigma}^{-1}$$

In a first investigation of the capabilities of this method, the reconstruction accuracy of DMD and DMDc on single-jet and multi-jet simulations would evaluated using the $\tilde{A}$ operator.

4.3 Dynamic Modeling Results

DMD was first evaluated on a single run dataset extracted from a constant-voltage simulation in the 2D CFD model, flattened to one dimension by summing across pixels equidistant from the substrate. DMDc was evaluated on a collection of runs from the 1D numerical

model over longer timeframes, comprising multiple cycles of EHD jetting and alternating, constant, and random Perlin noise voltage control inputs. DMDc was evaluated twice – first in training on each individual generated dataset, and again when trained on a single concatenated stack of state vectors from all generated runs.

In the case of both DMD and DMDc evaluations, success was defined to be a mean absolute error (MAE) of ¡ 0.1 when reconstructing the training run from its first frame. Reconstructions were evaluated against a baseline "model" that predicts zero mass for all elements and frames of the reconstruction. Results of model reconstruction are summarized in Table 4.2.

	Model Mean Absolute Error			
Dataset	**Baseline (all-zero)**	**DMD**	**DMDc (Individual)**	**DMDc (All)**
2D CFD	0.231	0.007	0.035	NA
AC1	0.218	0.192	0.300	0.580
AC2	0.184	0.180	0.250	0.310
AC3	0.197	0.190	0.280	0.248
AC4	0.211	0.208	0.340	0.598
DC	0.247	0.205	NA	0.294
Perlin1	0.200	0.205	0.305	0.278
Perlin2	0.165	0.169	0.270	0.224
Perlin3	0.179	0.177	0.350	0.221
Perlin5	0.199	0.157	0.269	0.400

Table 4.2: MAE for reconstructions of EHD model runs by DMD, DMDc trained on single runs, and DMDc trained on all runs

DMD (and DMDc, as a sanity check) were able to reconstruct the single jet cycle from the 2D COMSOL model run. However, neither DMD nor DMDc algorithms were able to reconstruct any of the multi-cycle datasets produced by the 1D EHD model. A selection of

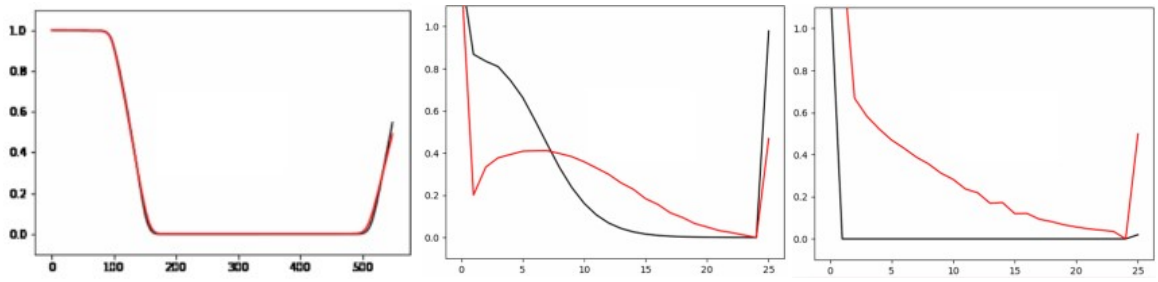


Figure 4.5: Selected frames from ground truth (black) datasets and their and DMD reconstructions (red). Training sets were 2D (left) and 1D (middle and right) models. DMD and DMDc were able to reconstruct a single jet cycle, but any multi-cycle behavior caused immediate divergence from the training data.

frame comparisons between ground truth and reconstruction can be seen in Figure 3.

The simplifying assumptions behind the 1D model take it well outside of physicality – additionally, only cursory manual tuning was attempted insofar as the model needed to exhibit periodic jet formation in an objectively similar fashion to that observed in an actual EHD printer. Future refinements to the model, especially its treatment of curvature and field concentration effects, could bring it in line with the COMSOL Multiphysics model in terms of accuracy. Nonetheless, the 1D model's dynamics were well defined and consistent within the range of inputs used to create the dataset, and so the failure of DMDc to fit the dataset speaks to a fundamental mismatch between the model and the task.

4.3.1 The Problem with Non-Linear Time Effects

While DMD was fully able to recreate a jet cycle, it failed when confronted with multiple cycles. Notably, the behavior of the EHD at the time of jetting onset is decidedly nonlinear. As shown in [66], a threshold voltage is required to overcome surface tension at the nozzle tip and initiate a Taylor cone formation. The Taylor cone is a field concentrator and grows rapidly in a positive feedback loop once initiated, but the energy barrier needs to be overcome before every jet pulse during long runs.

DMD and DMDc, which assume a priori that the operator $\tilde{A}$ is linear, may be fundamentally unequipped to deal with this nonlinear jet onset behavior in time. Therefore, the models fail to reconstruct jet dynamics as soon as more than one cycle is included in a dataset. In canonical uses of the method such as turbulence and strange attractors, the total energy remains stable (or decays) and is always visible in the state variables. EHD, while comparatively simple, involves long periods of total dormancy in the mass-distribution state variable while the hidden state (voltage, or accumulation of charge) increases to a threshold value. This hidden threshold behavior may be beyond the capabilities of DMD to reproduce.

DMD can capture arbitrary and non-linear effects in its X and X' state variables, but a modification to the method is needed to allow for nonlinear effects in the Koopman operator. Perhaps in future work, substitution of a neural network for the forward evolution of time dynamics, such as that demonstrated by Yeung et. al.,[157] could recover the EHD's pulse behavior.

4.3.2 The Problem with EHD Spatial Variability

A second problem with dynamic models - one encountered briefly when trying to train models using the high-speed image dataset - is the variation inherent in repeated runs of the EHD printer. Very precise tip height control is not present in any of the tools we have access to, so the state vector X will vary in dimension from run to run. Additional variations in ink and substrate are likely to produce marginally (or significantly!) different dynamic modes from one run to the next. In other words, the dynamic modes from past DMDc training runs may not transfer to the future.

To solve this problem, methods would be needed that delocalize the dynamics of the ink column. Recent forays into applying convolutional networks as the spatial inputs to the DMD process[90, 142] could possibly accomplish this, at the cost of a significantly expanded dataset requirement.

4.4 End-to-end Machine Learning

While dynamic mode decomposition models could reproduce the time series evolution of single jet events, they failed to converge on the prolonged time-series behavior encountered

in practical printing environments. Because this modeling approach could not replicate the time-series behavior of training data, we did not expect these methods to be capable of predicting unseen system outcomes or generalizing the physics of the system. Tackling this challenge would at least require significant modification to the Koopman operations that approximate dynamics both in space and time.

These difficulties in bringing dynamical models to bear on the EHD process may be surmountable! But they will not be trivial by any stretch of the imagination. Instead, we decided to pivot our modeling approach to a well-understood alternative that trades away interpretability by deploying the most successful black-box models of the contemporary machine learning field. We would apply end-to-end supervised ML toward model-based printer control for the remainder of the project.

Supervised ML describes the process of using powerful function-approximating models to reproduce tabulated datasets. The "supervision" process involves using a training dataset where "correct outputs" are known for a set of inputs. An effective model can generalize to new, previously-unseen inputs and accurately predict outputs in novel situations. For an EHD process, the "inputs" would describe the voltage and waveform delivered to the print head, and the outputs might characterize the printed features that end up on the substrate. A low-error model of this type could be used to implement "model-based control" of the printer, where the control algorithm runs different inputs past the model until it gets an answer that it likes before executing that hero input in the physical system.

The most successful recent ML models have often trained on extensive open datasets of images, text, or other data types. However, no such dataset exists for EHD printing in the available literature. If a supervised ML approach were to be successful, I would have to create the training data myself.

Chapter 5

BUILDING AN EHD PROCESS DATASET

Taking an ML approach to model the EHD process meant one thing above all else: we would need data. Supervised ML excels at taking advantage of large datasets to predict trends, and the data is the critical enabler. Data needed to be created, curated, and collected before any ML work could occur. Creating the software and hardware interfaces, formalizing a workflow, and turning the crank on EHD experiments comprised the bulk of the effort needed to create an ML-driven EHD control scheme.

This chapter outlines the EHD data generation strategy; Figure 5.1 illustrates the main steps of the process. Rapid serial exploration of precisely controlled waveforms with custom hardware generated large-area arrays of hundreds of EHD patterns, each representing the output from a unique waveform input. Then ex-situ optical microscopy and custom image analysis were used to parse and filter the results, resulting in a curated table combining thousands of experiments created across multiple experimental runs. This chapter details the methods used and challenges encountered during dataset creation.

Before anything else could happen, a decision was needed: which process variables would be controlled by the machine learning model, and which would not? Many physical inputs and print parameters can significantly affect the morphology of EHD printed features, especially the size of the print nozzle aperture, its standoff distance from the grounded substrate, and substrate velocity relative to the print head. Aperture size is difficult to modify without changing equipment, but the X, Y, and Z axis operations are, in principle, easy to modulate. However, these macro stage directions were locked behind proprietary software on the SIJ-150 EHD system that would be performing experiments.

So, setting aside the robotic control of the printer, the primary control interface was the voltage applied over time to the print head. This waveform is solely responsible for driving ink from the print nozzle to the substrate in the SIJ-150. The space of possible waveforms

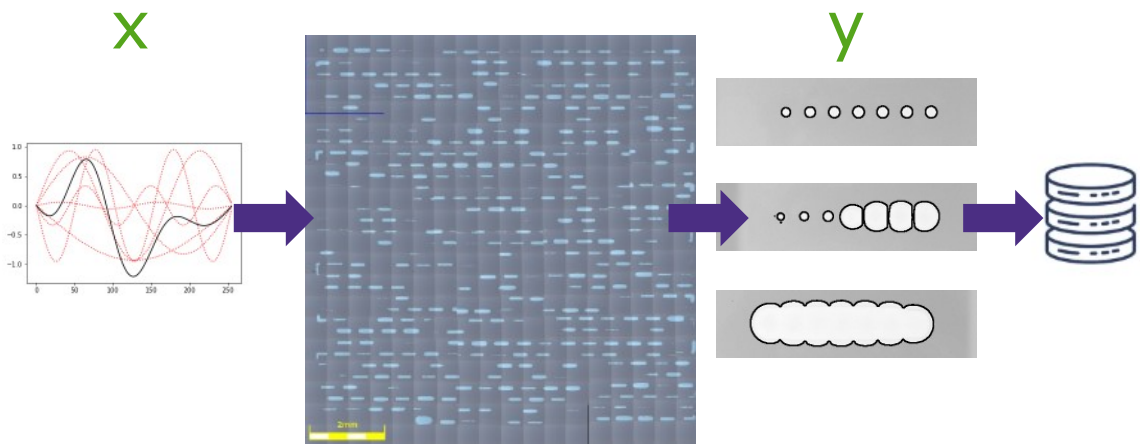


Figure 5.1: Process map of the data generation strategy for supervised training of an EHD process model. Rapid serial waveform trials, with ex-situ microscope characterization and image analysis of the printed features, which are logged to a growing database. A vector of float values are the weights that define the input waveforms X, and measurements in μm comprise the target values Y.

is theoretically huge if sufficiently flexible voltage hardware is available. So, the first order of business was to set up this voltage control hardware.

As illustrated by Figure 5.2, an arbitrary function generator was chosen to generate programmatic waveforms and inserted into the SIJ printer's control loop. Details of this setup can be found in Appendix G. The system was able to pass custom waveforms to the printer and print several features using the waveform at a frequency of 1 new waveform every 3 to 4 seconds. One hour of run time could now produce roughly 1,000 custom-generated waveforms. This would suffice to generate datasets for ML training.

5.1 Rapid Serial Experiments with an Arbitrary Function Generator

With a system in place that could generate custom waveforms and deliver them to the EHD print head, the next order of business was the creation of a standardized experimental protocol. By locking in a specific pattern on the SIJ-150 and standardizing experimen-

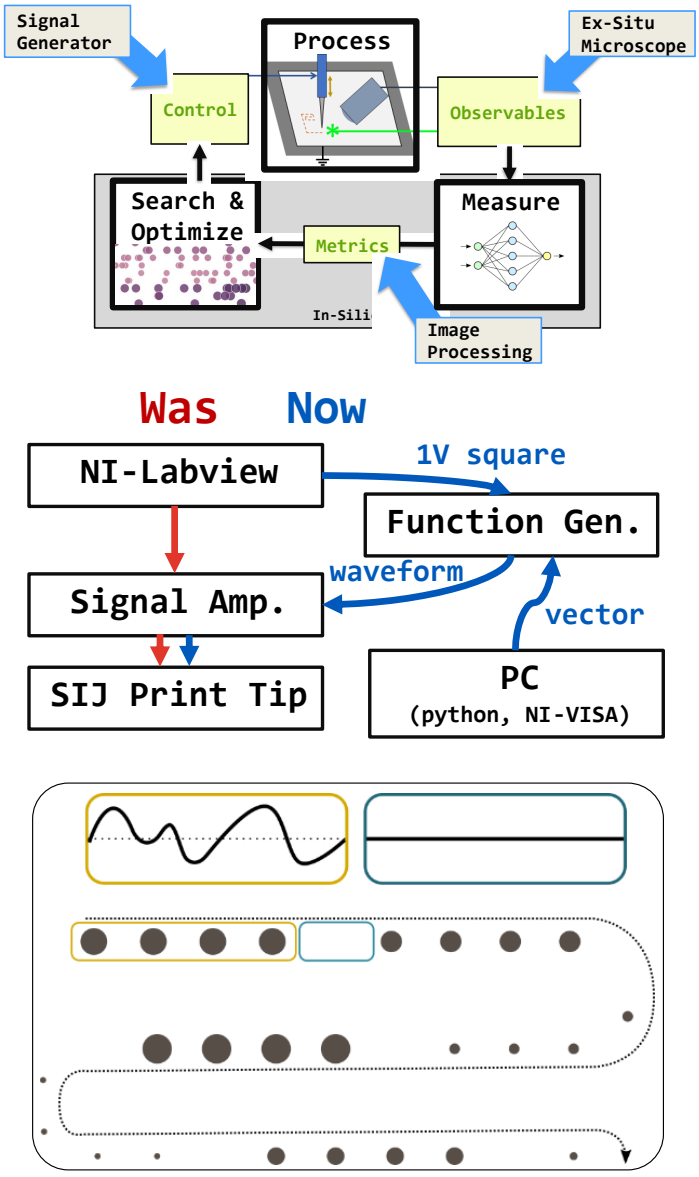


Figure 5.2: Hardware schematic for making rapid serial datasets. [Top] The process control loop, with signal generator inputs to the process, ex-situ microscopy, and classical image processing to generate learnable metrics. [Middle] A modified SIJ-150 control path with a PC-controlled AFG allows pseudo-real-time process control. [Bottom] Rapid serial experimentation strategy, with tandem repeats of randomized waveforms.

tal timings between the printer, signal generator, and the computer script responsible for choosing waveforms, the downstream steps of image and data processing would be easy to automate. Some details of the experimental setup can be found in section G.2.

Choices of material systems were locked in during early experimental runs. Polished silicon wafers were used as the substrate of choice for all runs because the atomically flat surface of polished silicon removed surface roughness as an uncontrolled variable. Doped silicon also brings a higher baseline conductivity to the process than other standard choices of substrate such as silica glass and polymer, decreasing the resistivity between the printing surface and the grounded stage of the SIJ system. Si substrates were prepared either by washing in isopropanol and de-ionized water or were used out of the box without significant surface treatment. Pressurized air dried and removed dust from the substrate surface before every experiment.

The printing ink of choice was AGK-104, a silver nanoparticle formulation by KGK Nano, Japan, that was one of several recommendations by SIJ for printing conductive line traces. Common wisdom in the lab held that AGK-104 was a well-behaved ink and a good choice for printing conductive traces. Nano-silver is conductive and relatively nonreactive, and the high mass loading of this particular ink lent itself to creating robust continuous lines with a single printing pass. The ink sinters at less than 200 C in about an hour, making it an easy choice for development work. AGK-104 contains about 60 wt% of 10-20 μm silver nanoparticles coated in a proprietary ligand and suspended in nonpolar tetradecane solvent. A hydrophobic 0.2 μm PTFE syringe filter removed particle agglomerations from the ink before each print run.

Metal nanoparticle inks with nonpolar solvents like AGK-104 have a particular response to EHD processing, as observed in EHD printing of a nano-gold colloidal solution by [120]. The field-driven ejection of such inks is thought to be caused primarily by a dielectrophoretic force that acts by polarizing the metal nanoparticles in the ink, then accelerating them in regions of nonuniform potential. In the SIJ print head, nonlinear potentials arise in at least three regions. First, a field concentration region lives at the tip of the metal electrode, located several millimeters above the nozzle. Second, the nozzle tip is a field concentrator, and the degree of concentration effect will change with the ink meniscus profile. Third, a

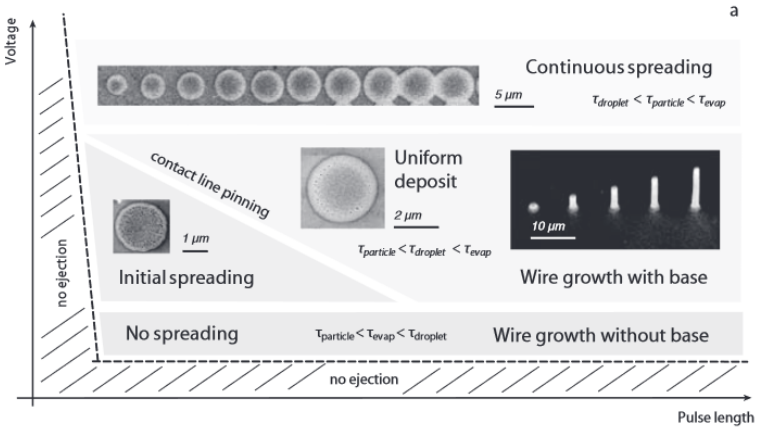


Figure 5.3: Plot of characteristic deposition modes when printing metal nanoparticle inks in nonpolar solvent. Reproduced from [120].

nonlinearity will occur in the boundary layer near the ink-air interface due to the change in dielectric constant between the solvent and the air.

The sharp Taylor cone effect that was first described by [138] depends on the polarization of solvent molecules at an air interface and will generally not occur in ink systems with nonpolar solvents. Many of the accepted EHD printing modes described in the literature, such as jetting, sprays,[39] and pulsed cone printing[37] may not be accurate characterizations of the printing landscape driven by dielectrophoresis. Instead, [120] proposes the printing landscape reproduced in Figure 5.3, with characteristic print onset boundaries. On the voltage axis, a critical onset voltage represents the net force needed to overcome surface tension at the print nozzle aperture. On the time axis, the critical pulse length represents a combination of characteristic rates: polarization of particles, particle drift diffusion to the ink interface, and droplet formation. Unsurprisingly, these characteristic thresholds reappear to some extent in the datasets produced by printing AGK-104 ink.

5.1.1 Experimental Design is Destiny

In an early example of a recurring theme in this project, early decisions made while setting up the data-gathering hardware would have significant downstream impacts when it came

to bounding and biasing the machine learning models that would eventually emerge from the datasets produced by the rapid serial experimental setup. The choice of 300 μm pattern pitch created an implicit upper limit of printed feature size of 200 μm, with a soft barrier closer to 100 μm to allow for a safety margin between adjacent lines of prints. Additionally, the 50 μm^{-1} dot frequency created a transition at the feature size matching the dot pitch of 50 μm: features with widths less than this size would result in isolated dots on the substrate, while features with widths larger than this threshold would merge on the substrate with their neighbors, resulting in continuous regions of ink.

Even more significantly, the specific hardware setup of the modified SIJ-150 forced certain variables to be held constant, allowed others to be controlled once at the start of each experiment, and rendered others hidden or uncontrolled. A summary of some significant experimental parameters and the scope of control exerted across the EHD dataset is summarized Table 5.1.

A significant degree of variability in the ink's surface contact angle was expected to arise from the combination of uncontrolled surface and atmospheric conditions. A reduced contact angle will reduce the contact pinning effect at each droplet's edge, increasing droplet spread on the substrate. Generally, a higher contact angle is preferable because it allows smaller features and higher metal concentrations to deposit in a single pass. Using a nonpolar solvent ink may have reduced the sensitivity of contact angle to surface charge dynamics and humidity variability. However, the contact angle was still expected to be a significant source of noise in the dataset.

One final potentially significant source of variation arose from the operation of loading ink into an SIJ printing nozzle. This is a touchy manual operation that involves pipetting 1 μL of fluid into an extruded glass needle, and air bubbles will often become trapped in the ink column during the loading process. These bubbles do not necessarily prevent printing, but they could change the response profile of the printer. (We will hypothesize some massive defects effects caused by bubbles in the ink column in subsequent chapters.)

Finally, the AGK-driven setup gave rise to some practical challenges. Loss of signal and software delays between the PC and AFG could cause experiments to be dropped, delayed, or in rare cases, cause the entire system to crash. These events needed manual correction in

Table 5.1: Factors affecting EHD print results

Factor	**Scope of Control**	**Value(s)**
Atmospheric conditions	Uncontrolled temperature & humidity	Lab conditions
Ink formulation	Constant	AGK-104 nano-silver ink
Ink preparation	Constant	0.2 micron PTFE syringe filter
Ink temperature	Uncontrolled	Lab room temperature
Nozzle stand-off height	Changes every experiment, rough manual measurement	10 to 40 microns
SIJ nozzle aperture	Uncontrolled within manufacturing tolerances	2-6 micron aperture
SIJ nozzle size	Changes every experiment	Super-fine, standard, or Large SIJ nozzles
Substrate material	Constant	p-doped 110 polished Si
Substrate treatment	Changes every experiment	IPA and H_20 wash. Pressurized air.
Substrate velocity	Constant, machine-controlled	100 um/s

the generated datasets. While initial plans involved using this system as a final conduit to control the printer during test prints, the PC-to-AFG link was prone to 1-second latency gaps between sending a new waveform and printing it. The delay rendered any attempt to synchronize with XY stage control virtually impossible. And most persistent of all problems, many print runs were prone to tip clogs that needed to be manually cleared in the middle of each experimental series, causing data loss and possible process drift.

Despite these problems, the SIJ-AFG system enabled a dataset of several thousand individual waveforms over multiple weeks of experiments, at least an order of magnitude larger than would have been possible without the system. It was fundamentally irreplaceable in enabling a mass exploration of the effects of input waveform on printed EHD features.

5.2 Exploring Waveform Space

In addition to the benefits in throughput and programmatic control that the AFG provided, it also allowed experiments to reach beyond the specific repertoire of waveforms that the base SIJ printer software could produce. Rather than being limited to square and sinusoidal waveforms, the function generator allowed experiments to deploy virtually any conceivable waveform, opening up a vast range of potential inputs. Why was this additional flexibility necessary?

An early hypothesis was that EHD "super waves" might exist within the printing landscape. Such waveforms would take advantage of electric or acoustic resonances within the tip/ink/substrate system to produce superior printing performance. For example, if resonance could reduce the back-pressure threshold for the ink to begin extruding from the nozzle head, then printing could be performed at lower voltages, allowing controlled deposition of sub-micron droplets. Alternatively, a family of waveforms resistant to nozzle clogging might be identified. Discovery of these waveforms would require a broadly capable parametric description of waveform space.

On the other hand, a naive random approach to waveform generation would be unwieldy. For instance, randomly selecting the 8,000 float values that make up each wave function in the AFG would generally produce a consistent flavor of white noise. More principled approaches to generating random waves exist, such as Perlin noise and wavefunction collapse,

which would "smooth out" the functions produced by the random generator. However, these heuristic, gradient-based approaches still lack an underlying representational basis besides the waveform itself. Datasets consisting of these random waveforms quickly fall prey to the curse of dimensionality,[16] whereby the representational density of a dataset existing in a high-dimensional space becomes sparse as the volume of possible inputs scales exponentially with the number of dimensions. At 8,000 dimensions, the number of data points needed to explore waveform space fully is cosmically larger than anything our system could produce in a reasonable time.

Instead, the first datasets were generated using a spectral approach. Various base functions were good candidates to describe the spectral decomposition of waveforms, with wavelet functions being one popular choice. Instead, we chose to start with the classical choice, describing waveforms in Fourier space by constructing them as a sum over harmonic sine wave functions.

5.2.1 Harmonic waves

The datasets created using the "harmonics" wave generator used a random process that produced waveforms U according to

$$U_X(t) = \sum_{i=1}^{N} x_i sin(i\pi t) \mid t \in (0, 1)$$

where t between 0 and 1 describes the time window of the waveform. This is a sum over N harmonics, starting with a principle wave with a frequency double the time window width. Since all experiments used a time window of $1/4s$, the harmonic components had frequencies in the harmonic series $(2, 4, 6, ..., 2N)Hz$. X was an N-dimensional vector with each element x_i drawn uniformly $i.i.d.$ from the range $[-1, 1]$.

These functions have the practical property of being zero-bounded; $U(0) = U(1) = 0$, which returns the printer to a "safe" zero-voltage state at the beginning and end of each wave. The sinusoidal basis functions are also orthogonal (though not orthonormal), and all have the same amplitude of 1.

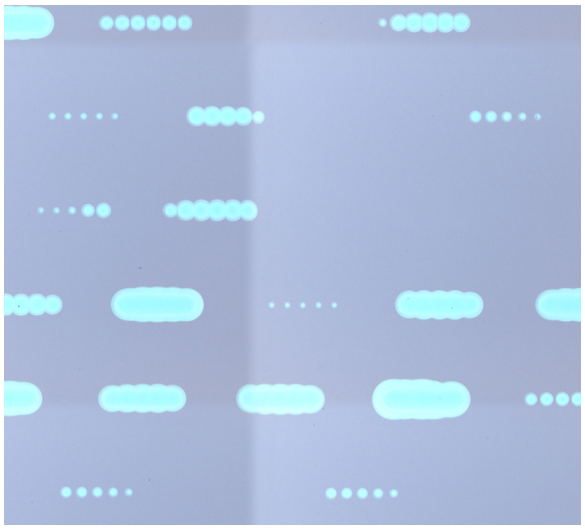


Figure 5.4: Example of rapid experiments printed using the 6-component harmonic wave generator.

In practice, U needed to reside in the domain $-1 < U < 1$ for delivery to the function generator. A bootstrap experiment defined a scaling function that scaled the waveform generator so that 10% of generated waveforms would clip into the domain limit. Using this scaling parameter $P = \frac{0.8}{4.4*10^{-5}N^3-5.4*10^{-3}N^2+0.31N+0.19}$, the waveform generator would return the waveform PU_X to the SIJ amplifier. The vector X containing the randomly generated weights of the harmonic basis functions was saved and would serve as the input value for ML training.

An initial test dataset with dimensionality $N = 16$ proved too sparse to allow ML models to converge - after that, all harmonics datasets used $N = 6$ harmonics. In practice, these waveforms tended to produce a wide range of feature sizes with no or minimal clogging over long runs. Figure 5.4 displays some example features from a large-nozzle harmonic generator run.

With six harmonic components, the fastest component frequency was 10 Hz - this would likely be too slow to excite any of the interesting acoustic resonances in a micron-scale SIJ print nozzle. Even before exploring the space of higher frequencies, however, a more practical limitation arose: with response times on the order of 1-2 seconds, it eventually became clear that the programmatic interface to the AFG would be too slow for practical use in a multi-feature print operation. For real-world ML-driven printing, all models would need to speak in a waveform language that the SIJ printer's native software could understand. This motivated a move to a simpler regime of waveform generation: square waves.

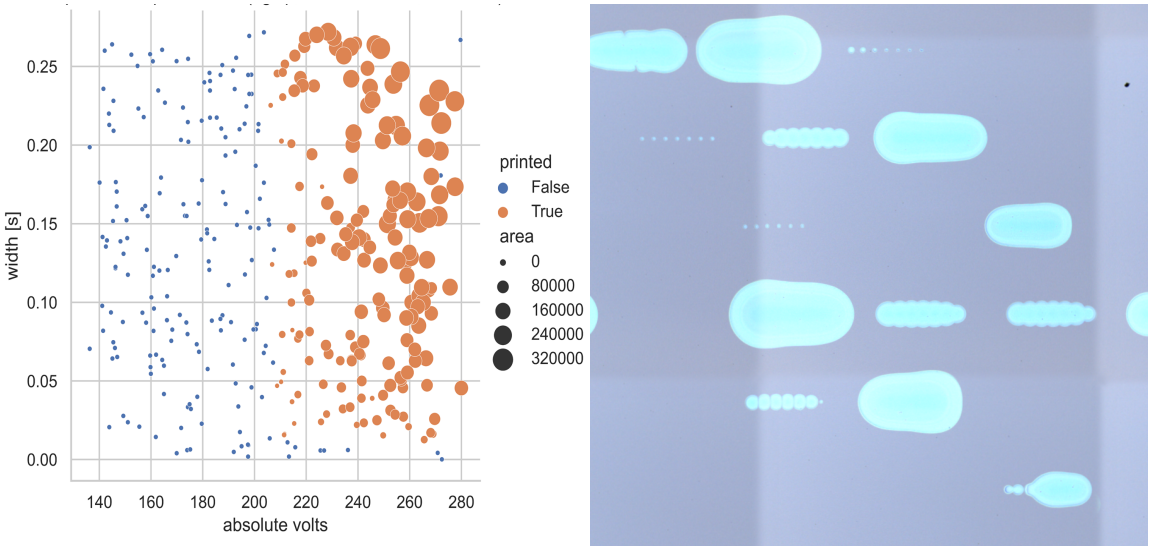


Figure 5.5: (Left) plot of experimental results and (right) example deposited features from rapid experiments using square wave inputs.

5.2.2 Square waves

The "square" waveforms were simple square wave functions with randomized pulse voltage and width. These waveforms were chosen to be directly transferable to SIJ recipes, allowing outputs from any models trained on the datasets to be directly transferable to the real system. They also represent an easily interpretable experimental space that could be plumbed for insight into the time-voltage dynamics of the EHD.

Figure 5.5 shows data from the first experimental square wave run, along with examples of some of the printed features produced by the square waves. This particular print produced a greater proportion of very large features and no-jet events than the harmonic waveforms had. Notably, clear boundaries became apparent in the process window that closely resembled those observed by [120]. A threshold value existed on both the voltage and the pulse width axis below which no jetting occurred (the blue dots.) These thresholds appeared to be more or less independent of each other. I.e., the voltage threshold of about 210V was largely independent of pulse width, and the pulse threshold of about 20 ms

appeared independent of voltage.

The threshold values appeared to be characteristic of the system. In particular, according to one working theory, they were indicative of the energy (voltage) and kinetic (pulse width) barriers that governed the boundary between waveforms that would eject ink and those that would not. Because of this, two additional steps were added to the beginning of all experimental runs after August 2022. First, the critical onset voltage would be measured using a pulse width of 0.5s - this would be the lowest voltage that would cause any volume of ink to eject from the nozzle tip. Then, the critical onset pulse width would be measured at 1.5 times the threshold voltage. Again, this was the shortest time that the set voltage could coerce any ink onto the substrate. Throughout subsequent experiments, onset voltages were measured from 66.0 to 116.1V, and critical pulse widths ranged between 1 and 96ms.

As the first attempts at using ML models trained on square wave datasets to control the printing process showed results, it became clear that an additional set of experiments dedicated to printing continuous lines would be necessary to generate models for an ML-controlled demonstration of printed conductive traces. The final wave generator for sinusoid-driven lines was the result.

5.2.3 Sine Line waves

As with the square wave prints, the "sin-line" wave generator was designed to create waveforms that would translate directly to the SIJ-150 printer. Waveforms were sine waves defined in a two-parameter space of frequency and amplitude. Frequency was drawn from a uniform distribution between 10 and 1,000 Hz (the upper limit of frequency control on the SIJ printer), and voltage was selected in a range between 0.85 and 1.0 times a maximum value that is chosen manually during the experiment. The lowest value of the sine wave was always pinned at 0V.

Line print experiments were executed differently from the dot experiments performed up to this point. The waveform played continuously for the duration of the print, and the substrate velocity of most of the sin_line datasets was increased to 500 $\mu m/s$, resulting in a more spread experimental layout on the substrate. Figure 5.6 displays an example image

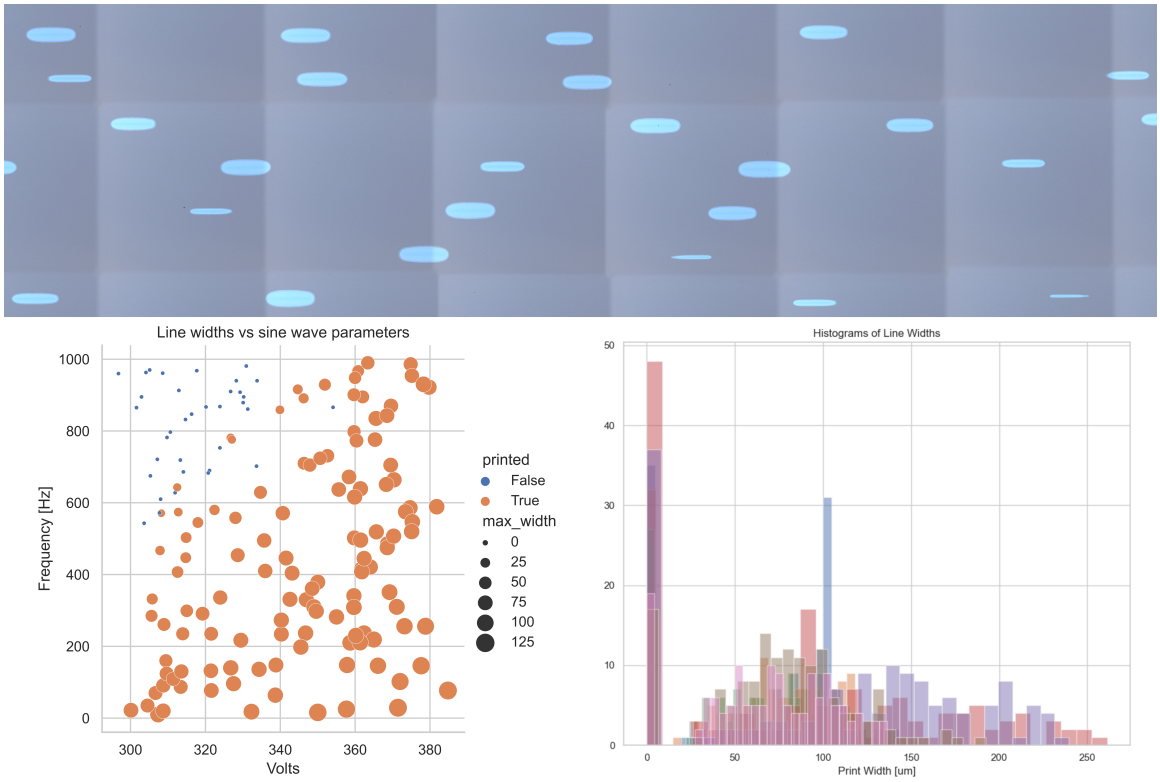


Figure 5.6: (Top) example printed features, (bottom left) experimental results, and (bottom right) histogram of all feature sizes from sine-wave line printing experiments.

from one sin_line run.

The bottom left of Figure 5.6 shows a characteristic dataset from a sine wave line run. Print size consistently fell off as frequency increased, especially above 600 Hz. An overlay histogram of line widths from all runs performed this way, in the bottom right of Figure 5.6, reveals that there was a practical lower limit across all experiments of about 20 μm line width and a fall-off in occurrence above about 150 μm line width. The lower limit was likely due to a combination of the arbitrary choice of velocity and a minimum flow rate thresholding effect in continuous printing that we will discuss in more detail in subsequent chapters. For now, suffice it to say that the collection of sine line experiments produced a dataset that mapped about one order of magnitude of line width variation to waveform

voltage and frequency and provided training support for ML models to control line print patterns.

5.2.4 Running the AFG

It may be worth recording a few lessons learned from SIJ-AFG experiments in case future practitioners wind up in similar straights. The primary learned skill relevant to running the system was the ability to quickly tune the voltage window of the random waveform generator to produce useful results. Of course, massive voltage numbers were to be avoided, lest feature sizes go over 300 μm and spill into the adjacent lane of experiments. But an excess of low voltages also needed to be avoided in the interest of clog prevention and maintaining an interesting dataset. A prospective goal of the machine learning exercise was to classify which waveforms would result in jetting. Therefore, datasets needed to include some proportion of non-jetting waveforms to allow training and testing on the classification task. Unfortunately, executing multiple back-to-back non-jetting waveforms tended to fully clog the printer, preventing all prints until the nozzle could be unclogged.

To enable voltage fine-tuning and de-clogging, a real-time voltage control was added to the random search waveform generator. The voltage could be modified mid-run to adjust the balance of non-jetting inputs, or large bursts of voltage would serve to clear full clogs. The real-time voltage control also compensated for process drift, particularly useful to offset the consistent phenomenon where slightly higher voltages were needed as a run proceeded over minutes or hours of continuous printing. The experimental code logged the voltage setting for each generated waveform for later integration into a training dataset.

The final generation pipeline would proceed as follows. A random vector $X \in \mathbb{R}^N$ was generated in software, describing a parametric wave function. This vector constituted six floats in the range $(-1, 1)$ for the harmonic generator, two floats in the ranges $(V_{min\,voltage}, 1)$ V and $(T_{min\,pulse}, T_{max\,pulse})$ seconds for the square wave generator, and two floats in the ranges $(V_{min\,voltage}, 1)$ V and $(10, 1000)$ Hz for the sine wave generator. Then a waveform $U(X) \in \mathbb{R}^{8000}$ would be generated from the parametric vector and bounded within $-1 \leq U \leq 1$ if needed. Finally, a scaled waveform U' was delivered to the EHD printer amplifier

per

$$U' = \frac{V\,U(X)}{G}$$

where V is the user-supplied max voltage and $G = 300$ is the gain of the amplifier. X, U, and V were all logged for future inclusion in a database as model inputs. A repository of software that implements the waveform generation loop is available at `https://github.com/onakanob/Waveform-Parmeterization`.

After using the generation loop to print extensive experiments of varied features, the geometric properties of the printed features, which would comprise the outputs Y for the ML training task, needed to be characterized.

5.3 Measuring EHD Printed Features

Real-time feedback is a boon to virtually any controllable process. Additive processes, with their compact print heads and open-bed access, are especially ripe for in-situ instrumentation. Processing in-situ data streams is one of the most common applications of machine learning in the additive space,[50] and real-time imaging has played an essential role in developing our understanding of the EHD process in particular, from the first observations of Taylor cones [138] to more recent efforts to codify the energetic and dynamic factors underlying polar solvent printing. [32] [36] Our collaborators in the lab of Hantang Qin demonstrated real-time in-situ characterization of EHD processes using high-speed video of the print head [132] and laser scattering through a substrate [161].

Some unique challenges exist in characterizing the EHD process as it is implemented in the SIJ-150 printer. As opposed to past efforts that utilized print heads and droplet sizes on the order 10s to 100s of microns, our printer utilizes sub-5 μm nozzle diameters and typical standoff distances from the substrate in the range 5 to 40 μm and as small as 1 μm. The printer also uses a bulky motorized stage and an overhead gantry that make the geometry of fitting optics into the print housing challenging. Nonetheless, attempts to deploy several in-situ monitoring solutions were made, detailed in section G.3. These projects met with mixed success, but none of them were reliable enough to produce the extensive measurements that

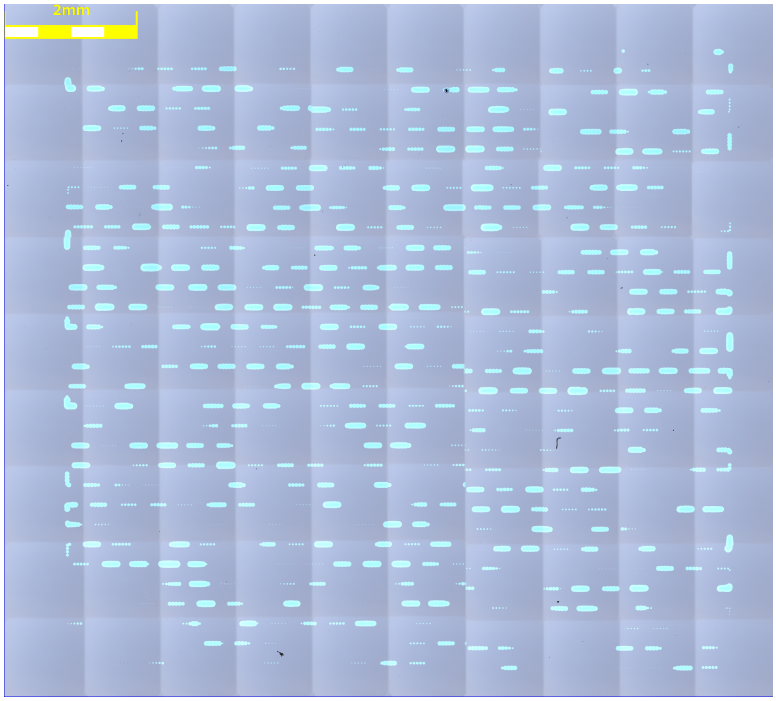


Figure 5.7: A full mosaic image of one experimental EHD run using harmonic waveforms. It contains 680 experiments.

would be needed in a ML-supporting dataset. Instead, an ex-situ measurement strategy would produce the feature measurements for the EHD dataset.

The print experiments described in the previous section generated large-area (up to 1 cm^2) prints containing hundreds of individual experiments. Each pattern would be post-processed using ex-situ microscopy. Each print needed to be imaged, isolated, and characterized using an image analysis pipeline. Then, the entire set of results would be quality-checked, indexed against saved waveforms from the waveform generator, and appended to a multitask repository of print runs.

An Olympus OLS41 optical profilometer created stitched mosaic images that covered the entire print area. Figure 5.7 provides one example of a stitched image. (These pictures contain no surface profile data - the profilometer was simply the nearest instrument with motorized stitching capabilities!) These large-area images served as inputs to a series of

software modules; that image-processing pipeline is discussed in detail in section G.4.

In ML parlance, every run and its constituent experiments constituted one "task." The overall EHD dataset contained many tasks, each executed with slight variations of equipment, ink condition, atmosphere, temperature, and so on. Training machine models that could contend with the variance that would arises from one EHD task to the next would occupy the remainder of the project. A unified interface for loading, folding, and preprocessing the multitask dataset proved necessary. To that end, in addition to image analysis and data collation tools, the dataset repository contains a data loader object, importable as "ehd_dataset.EHD_Loader". The data loader provides a standardized set of tools for applying filters to data points and returning folded training and validation sets of filtered tasks. The complete behavior and API of the loader is available at `https://github.com/onakanob/ehd-dataset`.

The EHD_Loader can return inputs "X" and outputs "Y" in several pre-set formats, summarized in Table 5.2. (Some of these return types only make sense in the context of specific ML challenges that arose during the project.)

5.4 The EHD Multitask Dataset

In total, 25 individual EHD mono-task runs were completed over about a year of experimentation. Each run comprised about 150 to 700 unique waveform experiments, with conditions ranging over three SIJ tip sizes, a range of standoff distances, and three waveform generators. In the context of current ML datasets, the compiled multitask EHD dataset is small in both the dimensionality of its representations and the extent of data it provides. However, it may be one of the more valuable contributions made by this project to the additive- and applied-ML field. Open datasets of process-property relationships in additive manufacturing, especially EHD printing, are largely absent from the literature. Hopefully, by sharing this data, future projects can use it as a stepping-off point for their own ML efforts or take it as inspiration to expand and improve the corpus of EHD datasets with diverse hardware setups.

The datasets, conditions, and number of experiments are summarized in Table 5.3. A complete download of the compiled dataset and all raw and processed images and run logs

Table 5.2: List of data formats in EHD_Loader

Type	**Name**	**Description**
xtype	vector	The raw parametric vector that produces the waveform
xtype	wave	The raw wave vector that was delivered to the AFG
xtype	last_wave	Concatenation of the last wave to play, its measured result Y, and the current wave
xtype	normed_squares	Vector of volts and pulsewidth, each value normalized by their threshold value
xtype	v_normed_squares	Vector of volts and pulsewidth, only the voltage normalized by its threshold value
xtype	transformed_sines	Vector of volts and frequency, with a specialized transform applied to the voltage
ytype	area	Total printed area in μm^2
ytype	print_length	Length of the print area bounding box
ytype	max_width	Width of the print area bounding box
ytype	mean_width	Area divided by print_length
ytype	obj_count	Number of detached printed features in the experiment
ytype	jetted	True if area > 0, else False
ytype	jetted_selectors	A 2-element binary vector: [1, 0] if jetted, [0, 1] if not. (Required for certain classifier models.)

Table 5.3: Index of all EHD experimental runs in the multitask dataset

Path	Date	SIJ Tip	Standoff [um]	Velocity [um/s]	Wavegen	Samples	Area Corr	Hz	V Thresh [V]@0.5s	W thresh [s]@1.5 Vt
29-Mar-2022 lg 1cm 300 points	29-Mar-22	Large	20	100	harmonics	263	0.5	4		
2-May-2022_run 1	2-May-22	Large	20	100	harmonics	121	0.64	4		
2-May-2022_run 2	2-May-22	Large	20	100	harmonics	528	0.62	4		
23-May-2022_squares	23-May-22	Large	30	100	square	352	0.55	4		
24-May-2022 large harmonics	24-May-22	Large	30	100	harmonics	682	0.59	4		
8-Aug-2022_lg-square-20um	8-Aug-22	Large	20	100	square	686	0.85	4		
10-Sep-2022_std-square-10um	10-Sep-22	Std	10	100	square	149	0.79	4	85.5	0.004
10-Sep-2022_std-square-20um	10-Sep-22	Std	20	100	square	643	0.82	4	100.8	0.003
13-Sep-2022_std-square-30um	13-Sep-22	Std	30	100	square	458	0.77	4	107.4	0.05
13-Sep-2022_std-square-40um	13-Sep-22	Std	40	100	square	250	0.79	4	116.1	0.049
22-Sep-2022_sfine-square-10um	22-Sep-22	SFine	10	100	square	299	0.82	4	66	0.01
22-Sep-2022_sfine-square-20um	22-Sep-22	SFine	20	100	square	548	0.59	4	84	0.01
11-Oct-22_sfine-sines-10um	11-Oct-22	SFine	10	100	harmonics	307	0.52	4	78.9	0.06
7-Feb-23_Lg-square-25um	7-Feb-23	Large	25	100	square	196	0.85	4	113.4	0.042
27-Feb-23_sf-sin-line-20um	27-Feb-23	SFine	20	200	sin_line	226	0.27		111.3	0.021
12-Mar-23_std-square-20um	12-Mar-23	Std	20	100	square	200	0.71	4	99	0.01
12-Mar-23_std-lin-line-20um	12-Mar-23	Std	20	500	sin_line	170	0.72		98.7	0.011
12-Mar-23_std-square-20um-2	12-Mar-23	Std	20	100	square	188	0.75	4	102.6	0.012
12-Mar-23_std-lin-line-10um	12-Mar-23	Std	10	500	sin_line	160	0.83		89.7	0.011
12-Mar-23_std-lin-line-30um	12-Mar-23	Std	30	500	sin_line	156	0.68		111	0.013
1-Apr-23_std-lin-line-20um	1-Apr-23	Std	20	500	sin_line	152	0.77		105.3	0.096
1-Apr-23_std-lin-line-30um	1-Apr-23	Std	30	500	sin_line	155	0.53		112.5	0.001
1-Apr-23_std-lin-line-15um	1-Apr-23	Std	15	500	sin_line	169	0.43		98.4	0.001
1-Apr-23_std-lin-line-25um	1-Apr-23	Std	25	500	sin_line	150	0.75		86.1	0.005
1-Apr-23_std-lin-line-10um	1-Apr-23	Std	10	500	sin_line	159	0.37		74.4	0.001

can be found at `https://digital.lib.washington.edu/researchworks/handle/1773/49974`. Unless otherwise specified, subsets of this data served as training inputs to all subsequent models in this report.

The dataset is not exhaustive in its exploration of combinations of per-run parameters like standoff height and tip choice because the complete list of runs was not planned from the start to be explorative. Instead, choices were made organically throughout the project in response to specific needs. The runs can be roughly divided into three epochs. The first, lasting until August 2022, was comprised strictly of large-nozzle runs, primarily using the harmonic wave generator. These were exploratory runs intended to explore the process space and set the benchmark for ML training in EHD process space. SIJ produces three nozzle sizes (Large, Standard, and Super-Fine), with aperture diameters that range from about 2 to about 7 μm. (The exact tolerances for each size are a trade secret.) Large nozzles are less prone to clogging and tend to produce larger features that are easier to characterize in optical microscopy.

The second epoch of experiments began in September 2022 and pivoted to square wave process space in the interest of training models that could be transferred to the SIJ-150's native software. Voltage and pulse-width threshold measurements were performed at the beginning of each run, and all models which rely on these measurements would be unable to train on datasets gathered before this transition.

Finally, after initial ML-driven control efforts showed poor transfer of knowledge from dot-printing to line-printing tasks, the sine-line waveform generator became the primary experimental driver. Beginning in March 2023, this third epoch of experiments filled out the final training dataset. Sine line datasets were also strictly confined to the Standard nozzle size and used an increased substrate velocity of 500 $\mu m/s$ to take advantage of an inverse relationship between velocity and line width. Due to that high velocity, all sine line datasets but one contain fewer than 200 experiments. The dataset titled "1-Apr-23_std-lin-line-20um" was excluded from most ML training activities because of a highly irregular pattern of feature sizes. (More on that in chapter 8.)

A reduction in the number of experiments per run occurred gradually over the course of dataset generation. In early square wave experiments, model accuracy showed diminishing

returns when increasing the number of training points above 100-200. So in later runs, increasing the number of runs took priority, translating to fewer experiments per run. The shift improved the breadth of the multitask dataset without compromising the accuracy of individual trained models, at least for the lower-dimensional square and sine-wave input types. Then, the increase in velocity when printing lines locked in this reduced experiment count by bringing the number of experiments that could fit in the 1 cm print area to about 200.

Fully assembled, the dataset collated the results from over 7,000 EHD experiments on dot and line printing tasks under a handful of print runs that captured some of the natural variability inherent in running the SIJ printer. It provided the data needed to support the training and validation of ML models on the tasks of jetting and feature size prediction and, ultimately, to deploy a model-based control loop to create recipes for EHD runs.

Chapter 6

SUPERVISED MACHINE LEARNING FOR THE ELECTROHYDRODYNAMIC INKJET PRINTER

The first choice to model the EHD printing process used dynamical models, a class of numerical model that would ideally capture and help to reveal the inner physical dynamics of the printing process. As described in chapter 4, however, key difficulties arose when attempting to apply these dynamical models to the EHD. Most notably, the system's non-linear behavior due to strong energy and kinetics threshold effects in the system defeated the naive application of DNDc models to predict printer behavior. Furthermore, inherent variability in the geometry of the EHD setup would have posed challenges to framing the dynamical models in a transferable way. So instead of pursuing the dynamical approach, we pivoted to a black box, data-centric approach with supervised machine learning (ML).

The difficulties of EHD modeling do not disappear when we move to end-to-end ML, but new tools became available to attack those problems. The requirement to retain detailed state space information was removed, so we could move away from detailed simulations and high-speed photography and consequently increase the throughput of data gathering. The rapid, serial experimentation discussed in the previous chapter provided a much larger and more varied set of training data. With larger datasets, supervised ML models like random forests and neural networks become practical - these models are powerful function approximators, effective at reproducing patterns with arbitrary, nonlinear characteristics. Unfortunately, the details of the physics of the system are lost when these models get involved. The EHD, all of its dynamism and process variation, is packed inside a black box model. When the dust settles, it will be observation and intuition that extract physical meaning from the project, not the models themselves.

Given a new dataset, a sensible first step will often be to train a model on the dataset - essentially, this involves tuning the parameters of some numerical model to reproduce the

data (or ideally, the function underlying the data) in the training set. We might test model performance by withholding some data from training and using them to test the model, and in an EHD printing context, we might say that such a model could have predicted the downstream outcomes had the printer kept printing on the day the dataset was gathered.

This retrospective approach can only get us so far. The datasets of the past represent a confluence of random factors that occurred on that particular day. Today, the printer needs to be tuned and controlled in the light of a new dice roll of tip, ink, substrate, and atmospheric conditions. Can yesterday's models be used to control today's EHD manufacturing run? In ML terms, this frames EHD control as a transfer learning problem. Just as a human technician will draw on past experience to quickly tame the process, transfer learning attempts to use learned patterns from past data, and perhaps past models, to efficiently bring a control loop up to speed in a new environment.

To evaluate different modeling approaches on their ability to perform transfer learning in the EHD dataset, I created a standardized evaluation loop that would automatically put any prospective model through its paces. The test graded each model on two tasks (jetting classification and feature size regression) using four metrics (F1 and AUC scores, Pearson correlation, and mean absolute error). For an introduction to some key ML topics such as performance metrics, bias variance trade-off, and few-shot learning, see Appendix H.

6.1 ML and Few-Shot Learning in Additive Manufacturing

In constructing ML models for EHD control, persistent challenges of sparse data and process variability would arise. These problems are not unique to the EHD process, and others have demonstrated applied ML in the additive manufacturing space, including in few-shot and transfer learning applications. The high-throughput and close control of additive manufacturing would seem to make it a natural fit to generate and exploit machine learning models, and ML has found diverse applications for additive manufacturing including fault detection, topology optimization, and quality control.[146, 114] Using ML models to control or characterize traditional inkjet processes is a growing field, with examples including droplet control in traditional inkjet[72, 154] and unsupervised clustering of in-situ video streams.[60] ML applications to EHD printing have been relatively sparse, though Ball et. al.[11] trained a

neural network to predict droplet sizes within a 3-variable, 3-level full factorial DOE.

Transfer learning has recently been applied to the characterization and prediction of powder bed sintering additive manufacturing processes, including for the processing of in-situ images[34, 35, 47] and acoustic sensors,[105] and for predicting manufacturing outcomes in simulated powder melt beds.[61] These applications use transfer learning to bridge gaps between disparate materials and powder compositions. However, the methodology has apparently not yet been extended to modeling inkjet or EHD processes.

Compared to these past examples, the EHD ML use case has an unusually varied input space. The model inputs X are parametrically-defined waveforms, able to scale from very simple to high-dimension representations. And the degree of run-to-run variability is extreme, owing to the EHD's small scale and hypersensitivity to small changes in the geometry of its setup. Furthermore, lacking any real-time in-situ feedback, any practical use of models to drive real prints would need to operate in a one- or zero-shot learning environment. To develop models that can perform under these conditions, it would be helpful to have a standardized and extensive test for the performance of any modeling approach under limited-data conditions.

6.2 N-Walk Evaluation

The goal of the standardized model testbed was to provide the best possible evaluation of a model's ability to deliver accurate predictions under real-world conditions. And from the start, the real-world destiny of ML models trained on the EHD dataset was to drive model-based control of our SIJ brand printer. With that environment would come a host of limitations discussed in previous chapters: a lack of in-situ imaging, limits on the type and number of waveforms that could be produced, and an inability to adjust recipe parameters mid-stream. However, I organized the model tests with a more expanded set of applications in mind.

The model evaluation loop, referred to as the "N-walk", simulates a perfect in-situ imaging system. After all, the lack of such a system on the SIJ-150 was driven by limitations of equipment, time, and money, not by a lack of feasibility. Others have demonstrated in-situ measurements of additive- and EHD-patterned substrates.[50, 161] So, the evaluation

loop assumes that the measurements of all patterns are known immediately after printing them. A live evaluation run might proceed like this: an EHD printer begins an experimental run, choosing a waveform u parameterized by d parameters $x \in \mathbb{R}^d$ selected uniformly *i.i.d.* from an allowed range. u is passed to the print head and causes ink to jet (or not) onto the substrate. The critical dimension y of the pattern is measured by processing an in-situ data stream. This process repeats N times, generating a supervised training dataset of inputs $X \in \mathbb{R}^{N \times d}$ and outputs $Y \in \mathbb{R}^N$.

After training on the N-long dataset, to evaluate a model's transfer learning performance, its accuracy is evaluated on the task of predicting the outcome of the *next* print. In practice, we would perform a series of M test prints and evaluate performance on the test set. If the model can perform within tolerance, we can entrust it to provide predictions for a model-based control loop after seeing N examples in the future. The goal is to minimize N while still meeting performance goals.

In practice, evaluations were performed over folds of the multi-task EHD dataset. (One "task" refers to a dataset from a single experimental run.) For each fold $l \in \{0, 1, ..., L\}$ over L mono-task datasets, for each value of $N \in \{2, 5, 10, 20, ...\}$, a random index t in the time-ordered task dataset D^l is chosen as the starting point. Then training data are drawn from the dataset for the next N indices, i.e., $D_{train} = D^l_{t \to (t+N)}$. The next M data are reserved as the test set: $D_{test} = D^l_{(t+N) \to (t+N+M)}$. In practice, this process repeats for several choices of t to make the performance estimate more robust. In each loop, the model type is trained on D_{train} and tested on D_{test}.

The last piece of the evaluation loop is a pretraining step. Before launching into the N-walk loop, models capable of transfer learning in the N-walk have the opportunity to fit the EHD tasks that are not the evaluation dataset D^l. That is, $D_{pretrain} = concat_{i \in \{0,1,...,L\}} D^i \,|\, i \neq l$. Restated in Pythonic pseudo-code with most of the constants used in practice, the N-walk evaluation loop is:

```
def n_walk(model, Datasets):
        results = []
        for l in len(Datasets):
```

```
        D_eval = Datasets[l]
        D_pretrain = Datasets[~l]
        model.pretrain(D_pretrain)

        for N in [2, 5, 10, 20, 50, 100]:
                M = 20
                for _ in num_trials(N):
                        t = randint()
                        train_set = D_eval[t:t+N]
                        test_set = D_eval[t+N:t+N+M]

                        temp_model = model.copy()
                        temp_model.retrain(train_set)
                        results.append(
                temp_model.evaluate(test_set))
    return results
```

In practice, $num\_trials(N) \propto C - log\, N$ is just a scaling function that prioritizes more repeats of smaller sample sizes N, typically scaled to perform about 100 training repeats in total for each l. The behaviors pretrain(), retrain(), and evaluate() are built into the model class, and the evaluation loop is agnostic to their implementation. A model that does not perform any action on the pretrain() step is a "cold-start" model - it only considers information introduced during the evaluation runs. A model that does not perform any action during the retrain() step is a zero-shot model; performance on the evaluation will depend entirely on knowledge gained from other tasks. Transfer learning models will make use of both training steps.

The evaluate() operation evaluates the model performance when predicting the $M = 20$ new inputs in the test set, reporting AUC and F1 scores for a classification task and r-score and MAE for a regression task. Plotting these metrics with respect to N generates a plot like that illustrated in Figure 6.1. This is an idealized vision of how the three model types

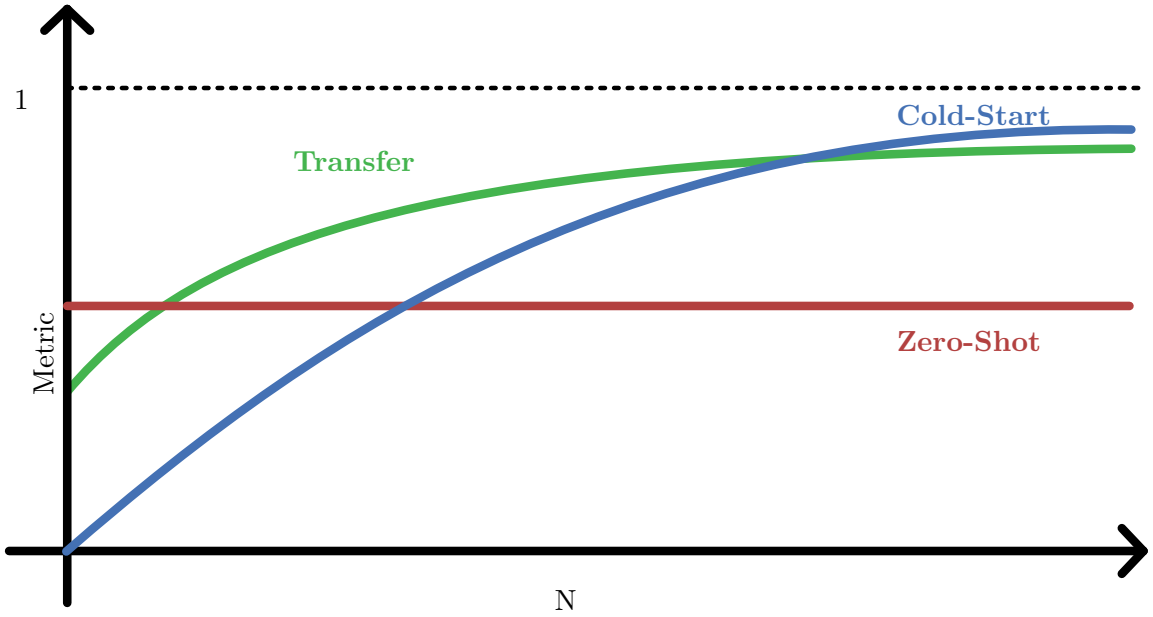


Figure 6.1: Expected performance trends in the N-walk evaluation loop.

might behave in this space. The zero-shot model is reminiscent of a high-bias or under-fit model in Figure H.1 because it inherits training bias from its pretraining operation that it has no way of correcting for the task at hand. It is entirely independent of N because it has no retraining step. While it may be effectively blindfolded to the process running in front of it, this model should have an advantage for very small N because it is incapable of overfitting to the small training set.

The cold-start model is the other extreme, underperforming due to variance for small N, where its training data provides insufficient support. For sufficiently large N, on the other hand, the cold-start model should become fully specialized for the task and become a top performer. Ideally, transfer models will provide a happy medium between these two extremes by using pretrained information to bolster accuracy for small N but de-biasing itself on the target task as N grows large.

One way to frame the overall ML task in the multi-task EHD dataset is: to identify a model that pushes the envelope of state-of-the-art performance in this graph up, to the

left, or both. That upper surface, which may be comprised of different model types for different N, represents the best estimated performance possible given a finite set of training data. In an actual process, switching from one model to another might be advantageous as a run proceeds and data accumulates. This evaluation loop could provide a roadmap for the inflection points from one optimal model to the next.

To present few-shot online learning as a practical approach to printer control, in addition to meeting performance goals stated above on any N, we set a secondary goal to match those metrics when limited to $N < 100$ retraining data points. At the current throughput, 100 data point represents about 8 to 10 minutes of data gathering.

6.3 Hyperparameter Tuning

It was necessary to tune the hyperparameters of cold-start, zero-shot, and transfer models to maximize EHD transfer learning performance. Hyperparameters describe an ML model's construction and training process: layer size, learning rate, and regularization constants are examples. Hyperparameter choice significantly impacts the performance of any model on a given dataset, and hyperparameter search is usually the most time and computationally expensive step of any ML project. Therefore, if the goal is to compare several modeling approaches on a single problem fairly, each competitor must be given the same tuning budget.

A devious trap in hyperparameter tuning is leakage from tuning to test. If any test data is present in the data used to tune hyperparameters, it can positively bias the final estimate of a model's capabilities by overfitting the hyperparameters to the test data. Therefore, it is typical to reserve a test set - say 10% - from the target dataset and use it after hyperparameter tuning is complete to evaluate model performance.

Unfortunately, in a transfer learning setting, a model's performance must be gauged not only on new *data* but on a completely novel *task*. While the EHD hardware could generate a decent number of data points, generating many mono-task *datasets* remains decidedly slow. After filtering by nozzle size and waveform type, most training datasets would amount to no more than seven task sets. Withholding just one as a test set would bias a performance estimate due to minutiae from that single withheld run. Withholding more than one would

significantly shrink the number of remaining tuning and training sets and likely have a significant effect on model performance. So the N-walk hyperparameter tuning loop and evaluation was designed to use all tasks in both steps, accepting some leakage in the process. This effect applied equally to all model types.

With all of the structure surrounding this ML task, innovating on the architecture of the models themselves would have been an unnecessary distraction. Well-worn ML models were used when possible in their Scikit Learn implementation.[109] In an initial evaluation of the dataset and the cold-start modeling task, these models were deployed with no hyperparameter tuning, simply taking default settings across the board. Later, to maximize performance, hyperparameter tuning was performed in a unified manner across all models.

Optuna [5] was the hyperparameter tuning management package of choice. Optuna was a practical choice because it comes with multithreading baked in and uses greedy runtime parameter generation that can be bent to allow model types to define their own hyperparameter search space under a unified interface. This method chooses hyperparameters using the Bayesian Tree-structured Parzen Estimator.[17] Training would run for 4 hours using 12 threads on an Intel i7-12700K, 3.6 GHz processor, and a GeForce RTX 3060 Ti graphics card where applicable.

The hyperparameter tuning loop would proceed much as described above, with some modifications. First, the optimization package would evaluate each set of hyperparameters on a single random fold of the multi-task dataset that selected two tasks for validation instead of one. Second, after pretraining on the other datasets, the model would retrain and evaluate several copies of the model as described above, but only with $N = 50$ retraining sets, to expedite the tuning process and prioritize performance for $N < 100$.

For regression models, the hyperparameter optimization loop attempts to minimize the MSE of predictions on the validation set. For classification, it attempts to maximize the product of AUC, F1, and precision on the jet prediction task. Adding the precision metric to the optimization goal was an attempt to minimize false positives, i.e., predicting that ink would deposit when it would not. The thinking was that these errors were more damaging than false negatives - predicting a no-jet for a waveform that would have produced a jet.

When the four-hour time limit had passed, hyperparameters that had produced the single

highest-performing trial would initialize a new set of models in a full N-walk evaluation loop for $N \in \{2, 5, 10, 20, 50, 100\}$. Finally, the routine would retain an ensemble of the highest-performing models from the hyperparameter tuning run for use in downstream control tasks.

6.4 EHD ML: Opening Salvos

The first dataset produced by the SIJ-150 used 16-component harmonics and contained about 110 data points. It is now excluded from the greater EHD dataset because absolute voltages were not recorded during the run. An initial exploratory ML run on this data produced abysmal results: the best model (a default-parameters random forest) could only conjure an r-score of 0.268 on print area regression. The result was not surprising - 110 data points were brutally sparse coverage in a 16-dimensional search space! Subsequent experiments reduced the number of harmonic components to six and produced a small cohort of large-nozzle harmonics datasets. The first ML experiments with these 6-harmonic datasets were informative.

6.4.1 Machine Learning on Harmonics Datasets

I will begin this section with a disclaimer: the original experiments performed at this time looked different than those presented in this section. For a start, they used the microns-squared print area as the regression prediction target, which produced very large and uninterpretable error numbers. The image processing pipeline was an early version, and that was the only prediction target I could record! Here, figures will display the same trends using the more manageable print width, in microns, as a regression target. This should allow easier comparison to data from later in the project.

I will also be leaving out some of the truly dead-end models that were implemented but quickly discarded. Linear models did not stand a chance: the same inherent nonlinearity of response that reared its head in the dynamical modeling attempt made these a non-starter. There were attempts at clever data engineering, alternate waveform representations, and bundling together multiple time-adjacent data that went similarly nowhere. I will let those wrecks lie at the roadside where they likely belong.

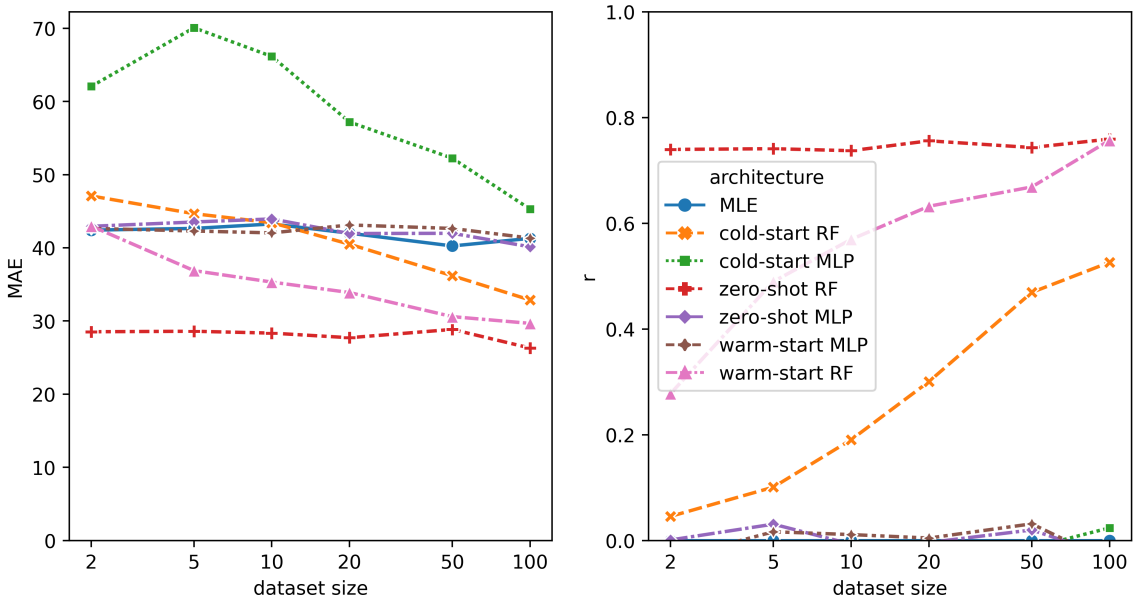


Figure 6.2: Performance of un-tuned regression models on 6 harmonic waveform datasets.

Instead, let us begin with the initial outing of two model types that showed early promise: a random forest (RF) (the top performer from the ill-fated 16-harmonic experiment) and a fully-connected multi-layer perceptron (MLP), which had the potential to perform well as datasets grew increasingly large. Both models took Scikit Learn's default hyperparameters and attempted to train on each of the three data diets: zero-shot, cold-start, and (for the regression task) warm-start. The results are shown in Figure 6.2 and Figure 6.3 for the size regression and jetting classification tasks.

The MLE model type is a naïve input-blind estimator that serves as a performance baseline. A random forest is an ensemble of decision trees trained on subsets of the training data - the model delivers the average of all trees for a regression problem or the consensus class across all trees for classification. The default Scikit Learn implementation uses 100 classifiers, and the "retrain" step for a warm-start model adds 100 new trees to the model that fit the new data. Finally, an MLP model is a neural network of fully-connected, gated regressors that fits data using gradient descent. Here again, for this early exploration, the

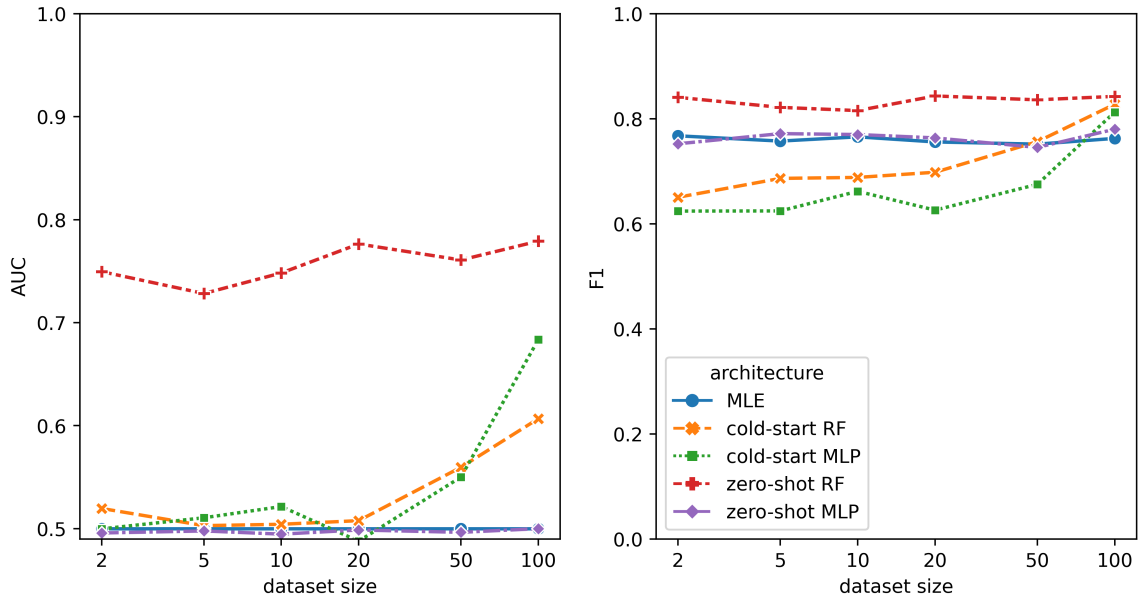


Figure 6.3: Performance of un-tuned classification models on 6 harmonic waveform datasets.

model used default parameters: a single 100-neuron layer, ReLU activation, the ADAM solver, a learning rate of 1e-3, and α = 1e-4. Only the maximum number of training iterations was increased from 200 to 1 million to avoid comparing against models that had not been allowed to converge.

All of these models (except the MLE) could undoubtedly see performance gains with their hyperparameters tuned. We can still draw some conclusions from this outing, however. First of all, the MLP models were distinctly poor performers, converging more slowly than their RF counterparts or not at all. This was likely due to a few factors: MLPs may have an increased reliance on hyperparameter tuning, particularly the regularization parameter and iteration limits that prevent them from drastically overfitting the training data. Therefore, they may be more prone to overfitting when training data is sparse. The random forest will still tend to produce values that it has seen before in the dataset under these conditions, where the unsupported MLP could output virtually any answer when given an input that is distant from all past training points.

Far and away, however, the zero-shot random forest models dominated performance across the board, demonstrating the lowest prediction error, greatest correlation, highest F1 score, and the only AUC scores meaningfully above the MLE baseline. What accounted for this discrepancy? The chief advantage of the zero-shot models was the large data pool to which they had access - the smallest validation fold allowed the zero-shot models to see 900 data points, while the largest N for the cold start models was 100. However, the run-to-run variability of the EHD printer would cap the performance of these zero-shot models. They would be forever biased towards past observations, never able to adjust to the unique combination of environmental factors on a given day. (I still sought realistic quantification of the extent of this task-to-task variation but suspected that it was significant.)

Why did the random forest outperform the MLP on the zero-shot task to such an extent? One culprit could be a severe case of overfitting. A zero-shot pretraining dataset was a naïve combination of several tasks lumped together, possibly presenting a mishmash of multiple response patterns spread sparsely over the 6-dimensional input space. The 100-tree random forest is a fundamentally limited model and may have produced a more generalized representation of this training space than the relatively more flexible 100-neuron MLP.

In a faint sign of life, the cold-start models begin to tick up in performance at $N = 50$ and 100. The MLP classifier, in particular, showed an early surge in performance at the $N = 100$ mark. The charts stop at 100 because not all datasets had enough samples to report performance beyond that. However, the metrics for the cold-start MLP model, shown with dwindling task folds up to N=500 in Figure 6.4.1, suggested that more performance was available if dataset sizes could continue to grow. The jetting and size prediction tasks appeared to be tractable problems, but it was unclear if practically-sized datasets would provide the support needed to fulfill project goals.

Unfortunately, the best models still fell far short on key metrics: AUC, F1, and r > 0.9 and MAE < 10 were but a distant hope. Could significant gains be made with hyperparameter tuning? Undoubtedly yes. However, the cold- and warm-start model trends indicated that the need for more training data was the current limiting factor. Before investing time to grow the dataset, a sanity check was in order.

The 6-harmonic inputs still appeared to be high-dimensional with respect to the dataset

N	AUC	F1	MAE
2	0.5	0.62	62
5	0.51	0.62	70
10	0.52	0.66	66
20	0.49	0.63	57
50	0.55	0.68	52
100	0.68	0.81	45
200	0.73	0.82	39
500	0.84	0.96	32

sizes that the AFG+SIJ hardware could produce. They were not transferable to the SIJ printer's native control software. Besides, the lack of a live in-situ measurement system meant that zero-shot control would be a requirement. It was time for an adjustment in approach to a waveform that was lower in dimension and reproducible with the native SIJ-150 software.

6.4.2 Exploring Square Wave Printing

The square wave input space is about as compact as it gets: the only two variables were the DC voltage of the square pulse and its duration, so the density of training points was now much greater than in the 6-harmonics dataset. Conveniently, these datasets could now be fully represented in a 2-dimensional plot - an early example appears in Figure 5.5. These datasets should be easy to fit - the demarcation between jetting and no-jetting was clear, and within the "jetting" region, increases to either voltage or pulse width would cause a monotonic increase in printed width.

A disclaimer before venturing further - the results presented here are another case of revisionist history, showing N-walk validation performed on a complete multi-task square wave set that accumulated over many weeks of experiments. In fact, many of the observations and decisions discussed below were made using a smaller and noisier subset of this data. Those early modeling results, in turn, informed downstream data collection strategy.

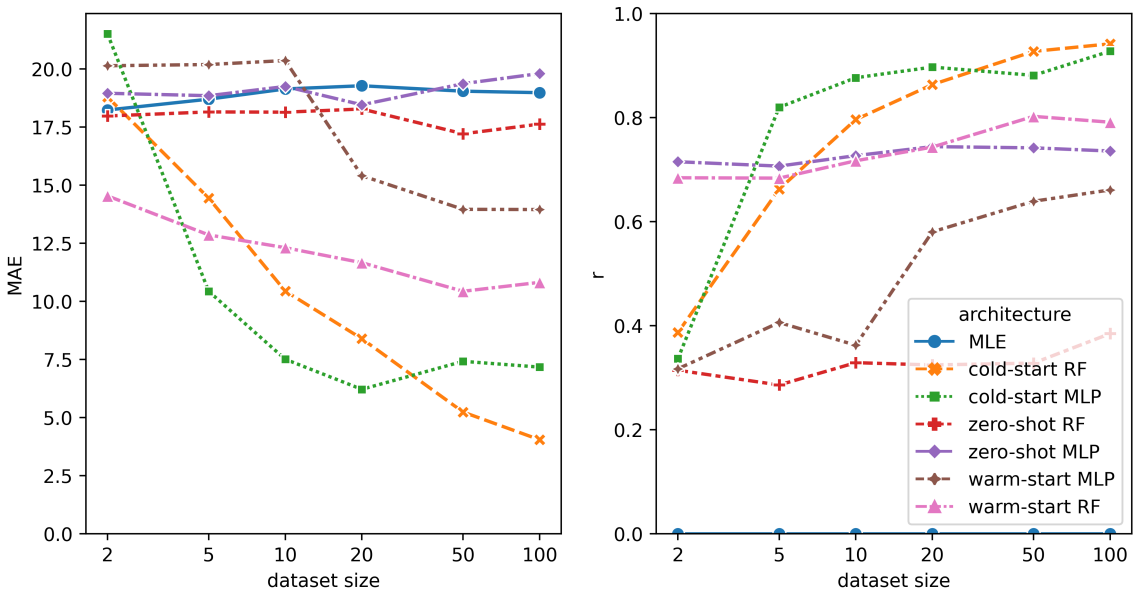


Figure 6.4: Performance of un-tuned regression models on square waveform datasets.

However, the core story has survived, and results that take advantage of as many multi-task folds as possible provide a more precise illustration of performance trends.

As shown in Figure 6.4 and Figure 6.5, models exhibited a marked increase in performance on the square wave dataset compared to the earlier harmonics results. The cold-start models were the stars of the show now, with the cold-start MLP hitting all target metrics with a training set of merely $N = 20$ and the cold-start RF breaking below an MAE of 5 μm at $N = 100$. That corresponded to a mere 2 pixels of error on average in the large-area experimental images.

With no hyperparameter tuning yet performed, the simplified input space of square wave training data sets enabled cold-start ML models to meet all of the project's initial performance goals. The square wave datasets included features with sizes ranging from 2 to 200 μm in diameter using waveforms that were directly reproducible by the SIJ-150 printer. Some drawbacks remained to the square wave approach. Compared to the harmonics prints, it was prone to nozzle clogs, especially when running close to its lower voltage limit, which

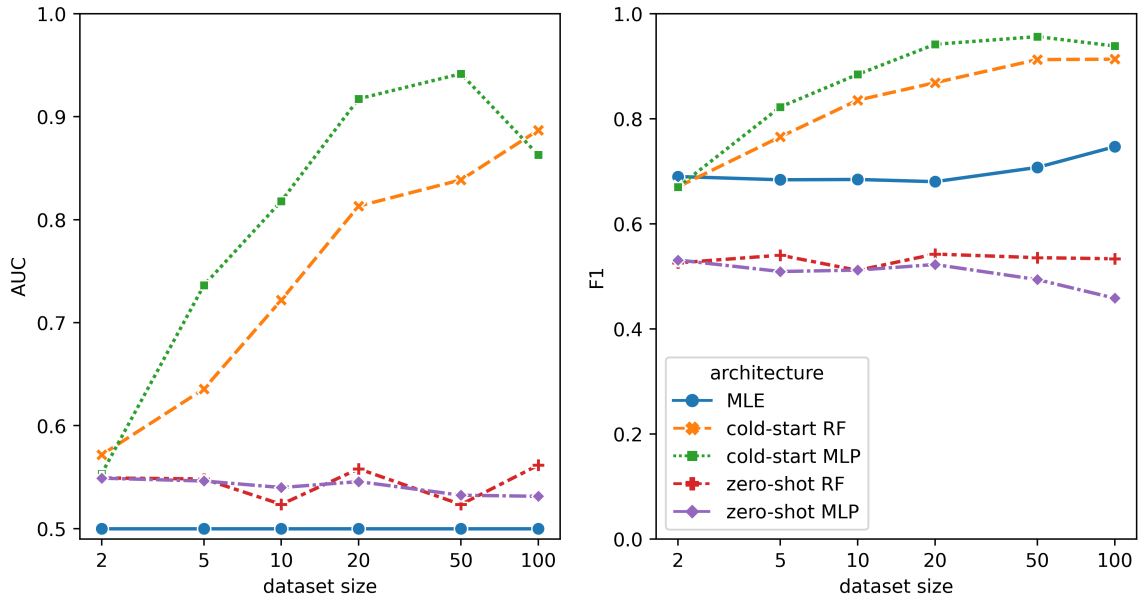


Figure 6.5: Performance of un-tuned classification models on square waveform datasets.

is also where the smallest feature sizes occur. This would make it it more difficult to reliably produce small feature sizes, locking patterns into a larger line width regime. It also meant abandoning any exploration of the dynamic sensitivities and resonances in the EHD tip. For now, however, square waves provided the clearest path to demonstration of practical printing with ML-controlled waveforms.

Despite the apparent strong performance of models trained on square wave data, it was not yet time to declare mission success. Cold start models were suited for an online, mono-task learning environment that could not be replicated with available hardware. A practical model-based control loop on SIJ hardware would need to perform in zero-shots. However, zero-shot and warm-start models were consistently poor performers in the square-wave datasets. Bringing the transfer-learning models up to speed would require confronting the run-to-run variation of the EHD and the resulting heterogeneity in the multi-task dataset.

6.5 Dealing with Variation through Homogenization

From the start of this project, run-to-run variation was a suspected - perhaps feared - bugbear. Dealing with variation in the SIJ tool's response was a known factor to its human operators, but the degree to which it would confound efforts to model it with ML was not yet known. The early results of square wave modeling seemed to point squarely towards a process variability root cause - what else could explain the abysmal performance of zero-shot models in a space with only two input dimensions and that had been readily learned with a relatively small ¡ 100 samples by cold-start models?

With the entire collection of square wave data plotted on the same axes, as in Figure 6.6, the extent of variation at play in these datasets becomes apparent. The square wave datasets are highly heterogeneous. Heterogeneity is used here in its statistical sense: no two monotask datasets behave as if they were drawn from the same underlying distribution. Not, at least, in the space defined by voltage and pulse width. Instead, for a given waveform input (say, 200 V for 100 ms), the printer would vomit ink on one day but print nothing at all on the next.

Heterogeneous data management is a salient topic in modern ML, especially regarding the aggregation and management of vast data from multiple sources, continuous streams, or data lakes.[147] The term is often invoked to describe dimensional or distributional differences in an ML task's input or output interface. For instance, heterogeneous datasets may have different dimensions from each other or be sampled from different populations, as in datasets of indoor vs. outdoor images. Or, a set of heterogeneous tasks may deal with different prediction target types, as when a single dataset of molecules might include both solubility regression and toxicity classification tasks. By contrast, the square datasets are all identical in the dimensionality, units, and task types of their input and output spaces. Instead, the underlying transfer function mapping input to output changes from one EHD run to the next.

A sensible approach to multi-task learning with this data would be to homogenize the input space of the various datasets, and a prime candidate to discover a homogenization function is through the characterization of the print/no-print boundary. This transition is

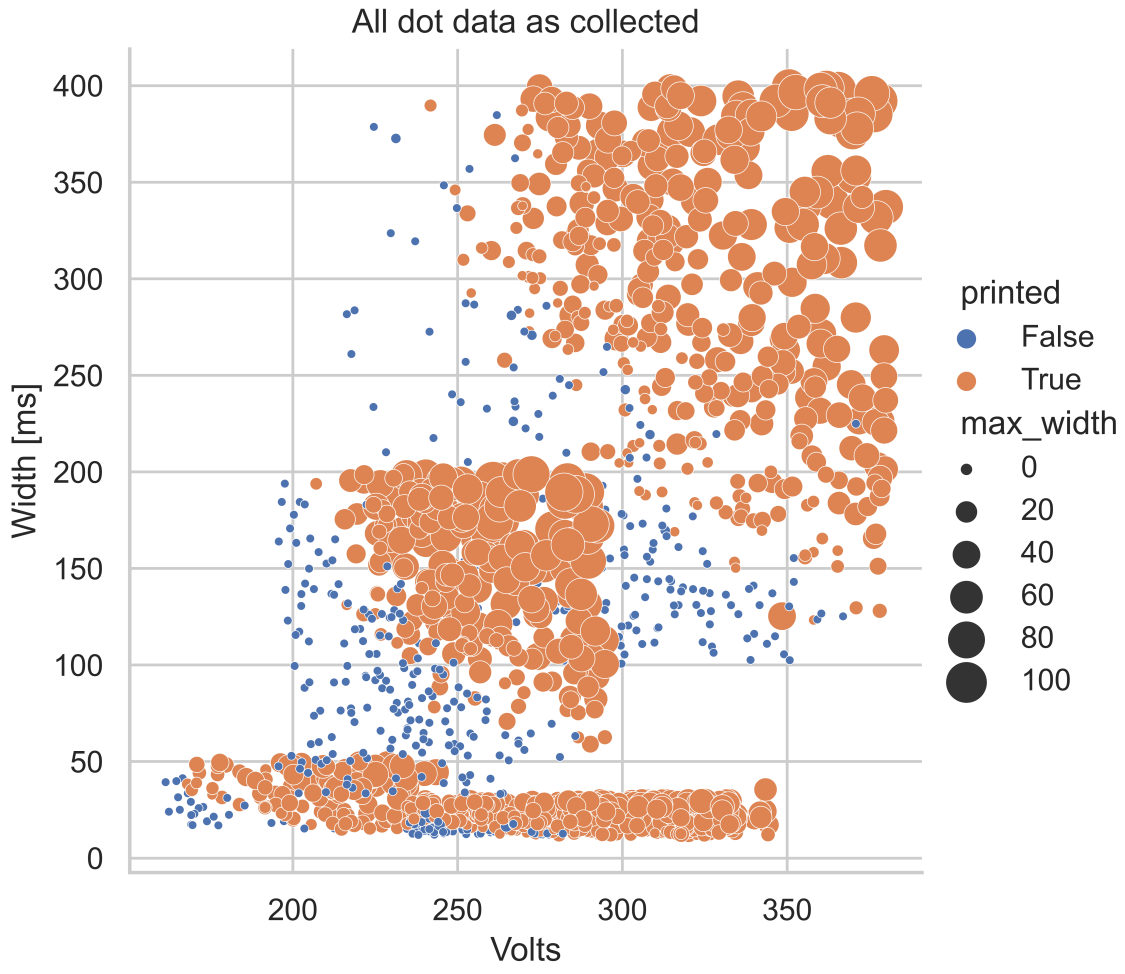


Figure 6.6: Plot of square wave experiment inputs and print sizes. No normalization was performed - voltage and pulse width are in real units.

present and shares a similar shape across every square dataset - an artifact of the manual tuning procedure that attempted to have a few (but not too many) no-print waveforms in every experimental run. To characterize the boundary, measurements of two thresholds were added to the data gathering procedure for all experiments - V_t is the threshold voltage, or the smallest voltage producing a print when the hold time is long (0.5 seconds in practice.) W_t is the threshold pulse width, or the shortest hold time producing a print when $V = 1.5V_t$.

The first homogenization procedure to try was to normalize each input value by its threshold, i.e., $V_{norm} = \frac{V}{V_t}$ and $W_{norm} = \frac{W}{W_t}$. Conceptually, after this transformation, every dataset would have its print onset boundary pinned at $[V_{norm}, W_{norm}]$ coordinates $[1, \infty]$ and $[1.5, 1]$. That is not quite what happened! Figure 6.7 displays an overlay of all square wave datasets in the normalized frame; the widths and voltages in practice tended to be at least double their threshold values. Nevertheless, this transformed space appeared to be markedly less heterogeneous than the previous spread.

Figure 6.8 and Figure 6.9 show comparisons of MLP models trained on data with and without this normalization. Moving to the normalized inputs caused a notable improvement in zero-shot performance on all metrics, and the warm-start MLP regressor improved by a similar margin, becoming roughly equivalent to the zero-shot model on both metrics. As expected, data preprocessing had no meaningful effect on cold-start model performance since those models perform per-task input normalization during training.

In early estimates of this performance, zero-shot MLP classifiers achieved AUC of 0.89 and F1 of 0.92 - this was sufficient to declare success on the classification problem. With the benefit of a larger dataset, this chart shows a reduced performance estimate in the range AUC of 0.75 and F1 of 0.8. Most significant uncertainty in the classification models likely remains immediately around the print/no-print boundary, where imperfect overlap between datasets creates ambiguity in the training data - see the region near 2.3 V/V_t and 6 W/W_t in Figure 6.7. These zero-shot models could discriminate between jetting and no-jetting waveforms and were used in initial outings of the ML-directed printer control loop.

Zero-shot regression performance showed an interesting trend, with zero-shot correlation values being extremely strong, at least r 0.93, and beating or matching the performance of all other models in this space. However, zero-shot absolute size error remained high in

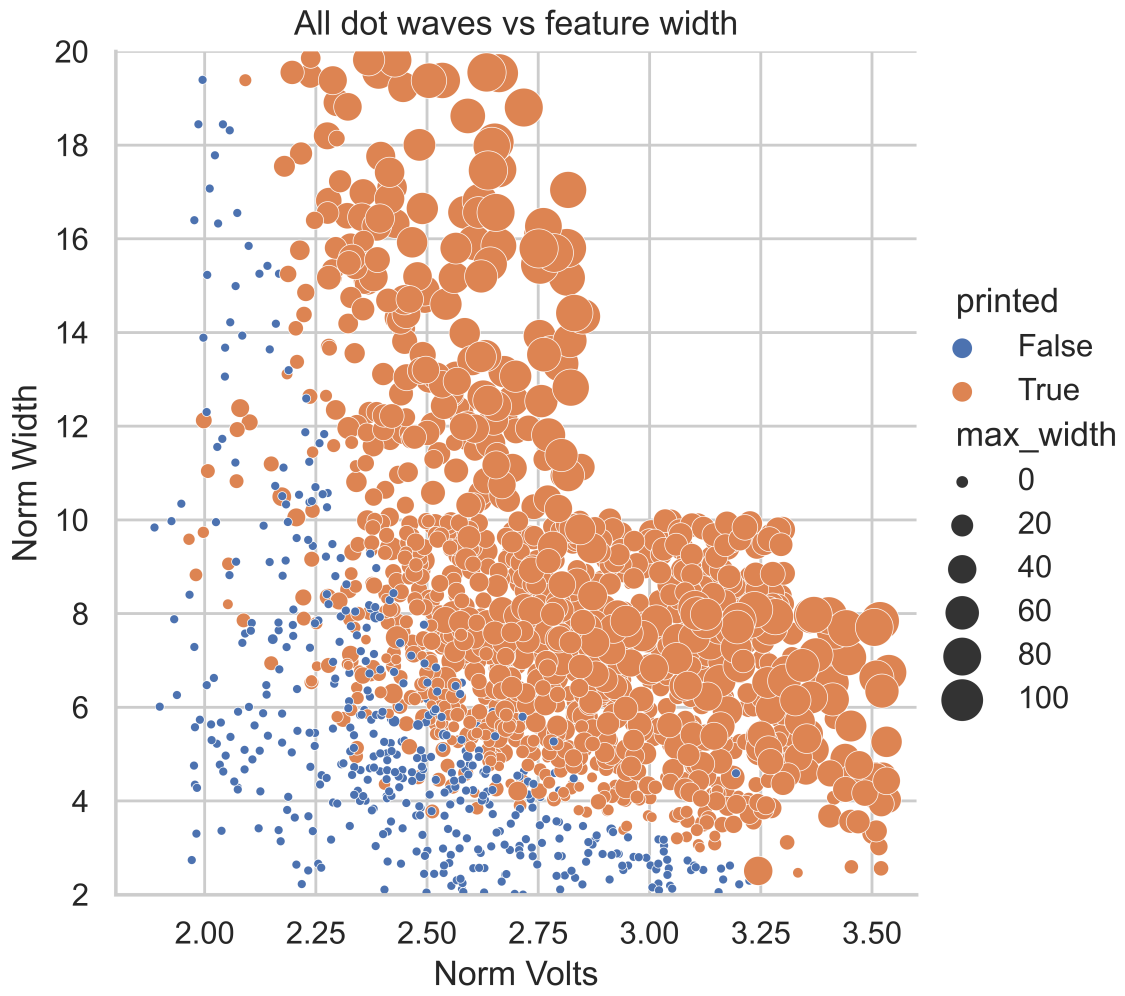


Figure 6.7: Plot of square wave experiment inputs and print sizes after normalizing voltage and pulse width by their measured threshold values.

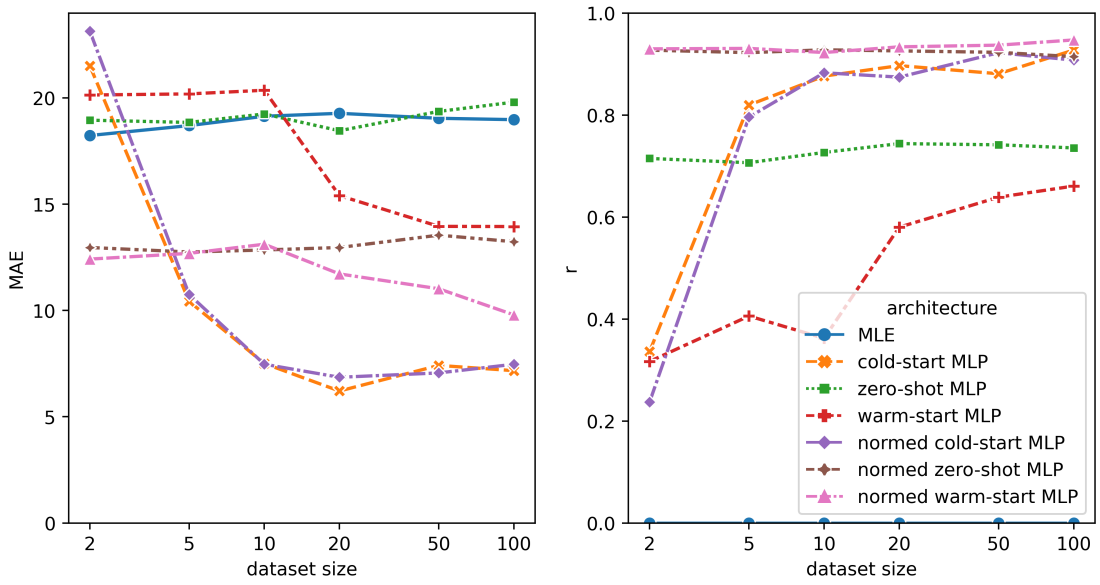


Figure 6.8: Performance of un-tuned regression models on square waveform datasets using threshold normalization.

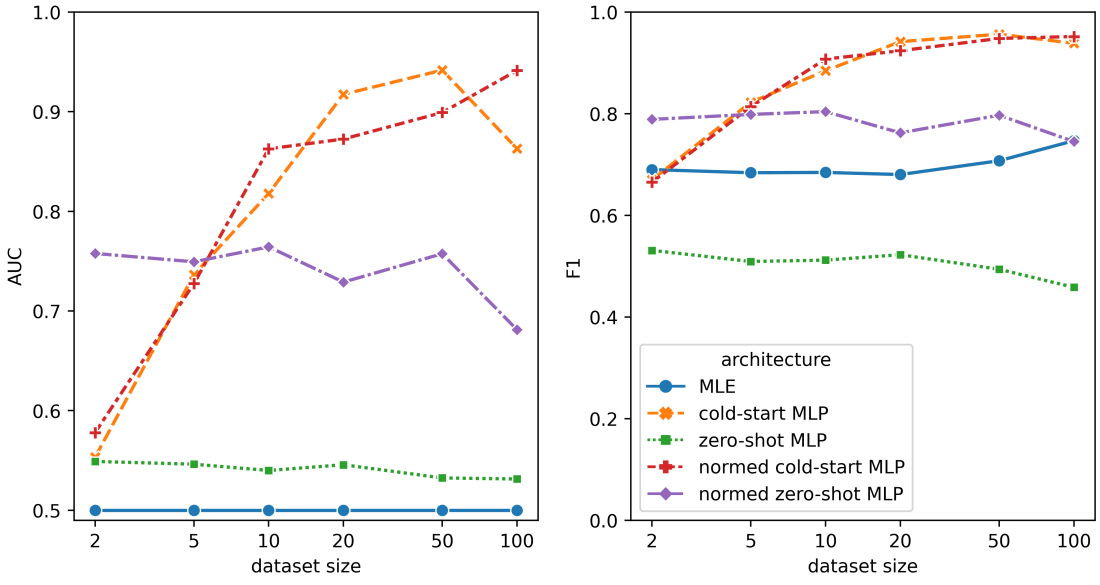


Figure 6.9: Performance of un-tuned classification models on square waveform datasets using threshold normalization.

zero-shot learning, missing by 12-13 μm on average. This mismatch suggested that zero-shot models were effective at *ranking* the relative size of different waveforms but tended to miss when estimating the absolute amount of ink deposited. They would produce correlated but biased outputs. Nonetheless, these models were deemed successful enough to be used in initial printer control experiments.

To summarize the results of applying ML models in a real process: recipes set by zero-shot models tended to produce biased but correlated print sizes, just as their N-walk results suggested. chapter 7 discusses how these ML models could serve as control knobs to rapidly tune the EHD process. Based on those experiments, the classification task would be dropped for subsequent experiments. Print/no-print classification may be helpful to place rails on a process in a true production setting, but in practice, it tended to make the control loop too risk-averse and unable to even attempt prints with small feature sizes. Beyond this point, models would train on the regression task alone.

ML-controlled dot printing eventually proved to be relatively successful. However, early attempts to apply ML models that had been trained on dot datasets to the task of printing lines yielded disastrous results. Early line prints often exhibited over 10x error between model expectation and reality. In response, an additional set of experiments was collected to fill out the dataset with sine wave-driven lines. Unfortunately, that meant the problem of heterogeneous data needed to be solved again in the line-printing input space.

6.5.1 Sine Line Homogenization

The sine line dataset, described in section 5.2, was similar to the square wave dots dataset in that it used one free parameter to control maximum voltage and another to control a time-related parameter. Square wave printing involved control of pulse width, which directly affected ink deposition time. The sine wave generator, on the other hand, varied the frequency of an AC signal up to 1000 Hz. A naïve use of the same normalization procedure that had successfully homogenized the square wave space did nothing to unify the input spaces of the various sine wave task sets. Figure 6.10 illustrates the extreme variation from one experiment to the next in the sine line datasets and the ineffectiveness

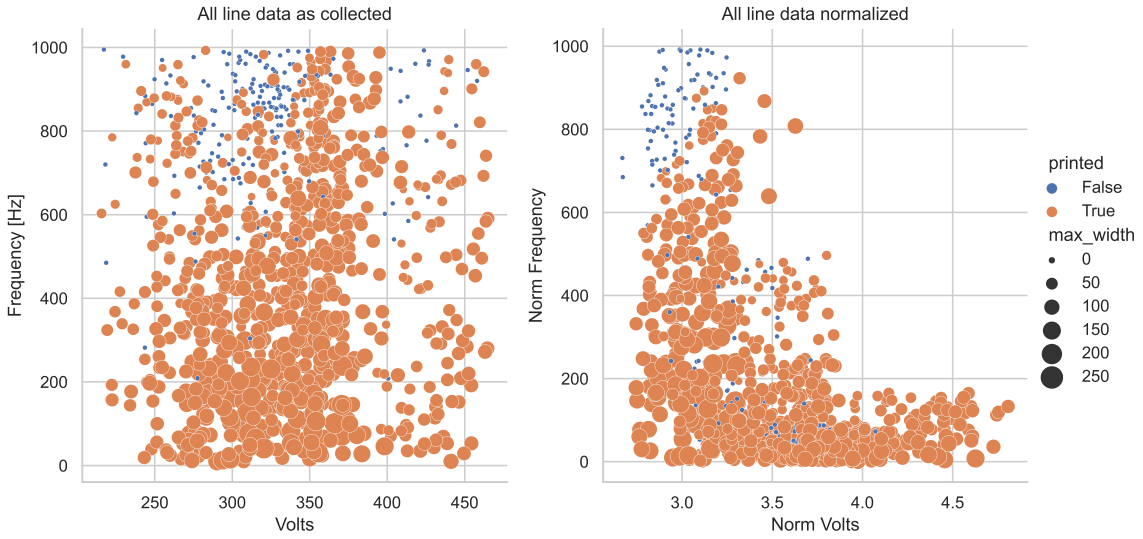


Figure 6.10: Plots of square wave experiment inputs and print sizes. (Left) experiments with real units, and (right) using the same normalization strategy that worked with square waves. Neither view of the dataset provides coherent overlap between the print and no-print regions of the different experimental runs.

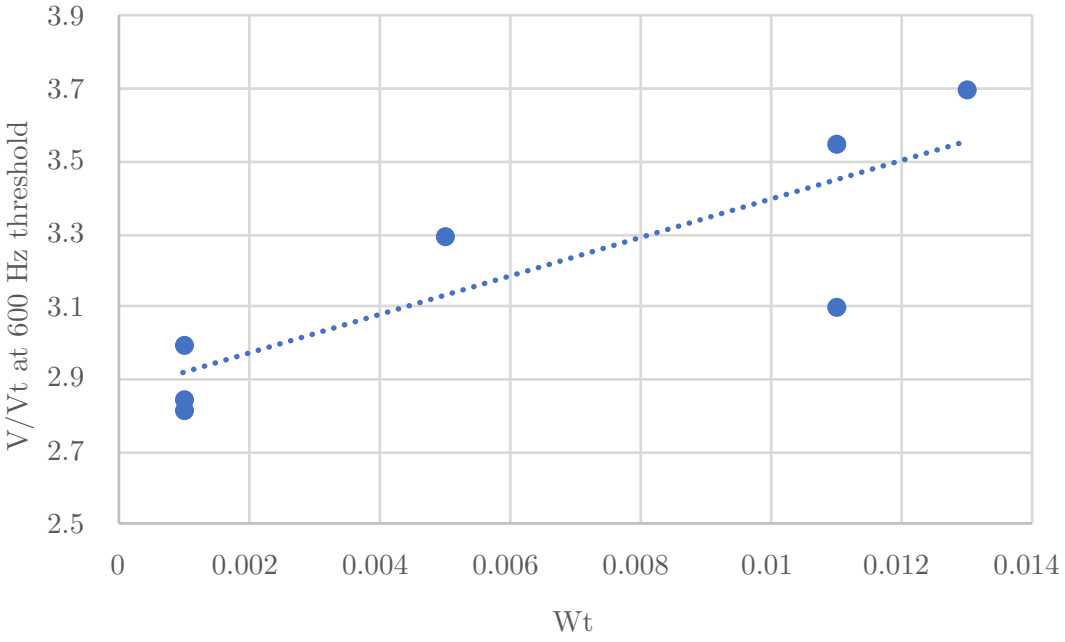


Figure 6.11: Linear correlation between the measured pulse threshold and the print/no-print boundary at 600 Hz in sine wave datasets after voltage is normalized.

of the old normalization procedure.

One problem with normalizing the sine wave frequency is that frequency can interact with many aspects of the EHD system in a constant printing regime. One of the primary points of interaction in a nanoparticle ink with a nonpolar solvent, such as the AGK-104 that was being used here, is the nanoparticle's dielectric relaxation time. This is the time for an external field to induce polarization in the dissolved nanoparticles. That polarization is a strict precondition for the dielectrophoretic acceleration of the ink, so the effects of a waveform on the EHD print process will necessarily fall off as frequencies approach and go past the relaxation rate. In practice, this effect becomes noticeable between 600 to 800 Hz when printing with the AGK ink.

Relaxation time is a property of the ink formulation and should not change from one

print to the next. (In the case of these experiments, the ink was drawn from the same bottle across the entire dataset.) Therefore, any attempt to modify the frequency axis of the sine wave input space would likely do more harm than good by misaligning the relaxation time effects from one task set to the next. Nonetheless, there was a definite relationship between the measured pulse width threshold and dataset behavior.

Every sine line dataset showed a notable increase in voltage needed to produce a jet above 600 Hz input frequency. That inflection point held some promise as a landmark to homogenize the datasets. Figure 6.11 shows the relationship between the normalized voltage V/V_t of the 600 Hz print/no-print boundary in each dataset and the measured pulse width threshold; longer threshold times indicated that a higher voltage would be needed to eject ink. Using the regression between the 600Hz boundary location and voltage, a custom transform was defined per

$$V_{transformed} = \frac{V}{V_{threshold}} - (69.536 * w_{threshold} + 2.8111)$$

that effectively pins the print/no-print boundary at the coordinate $[0, 600]$ in the sine line input space. The homogenized combined dataset is shown in Figure 6.12. Rather than relying on this tortured method of dataset homogenization, future projects using frequency control for an EHD would do well to measure points directly relevant to the input range. For instance, measurements of threshold voltage at one or more frequencies could likely improve the homogenizing transform.

With the custom-transformed dataset, the performance of cold-start and zero-shot MLP regression models took on similar characteristics to models trained on the normalized square dataset. Figure 6.13 illustrates these trends. Regression correlation was strong in the zero-shot model but not as strong as it had been in the square dataset. The regressors showed an average $r > 0.8$, which was nearly on par with cold-start model results. As with the square wave models, the absolute error remained high at about 30 μm. MAE increased overall when moving from dots to lines, but this may have been an effect of increased proportionality. The smallest features in the dot datasets were on the order of 2 μm wide, while the smallest printed lines were 20 μm. A tuned version of the zero-shot models trained on the custom-transformed sine wave data would carry into the second round of ML-directed line printing

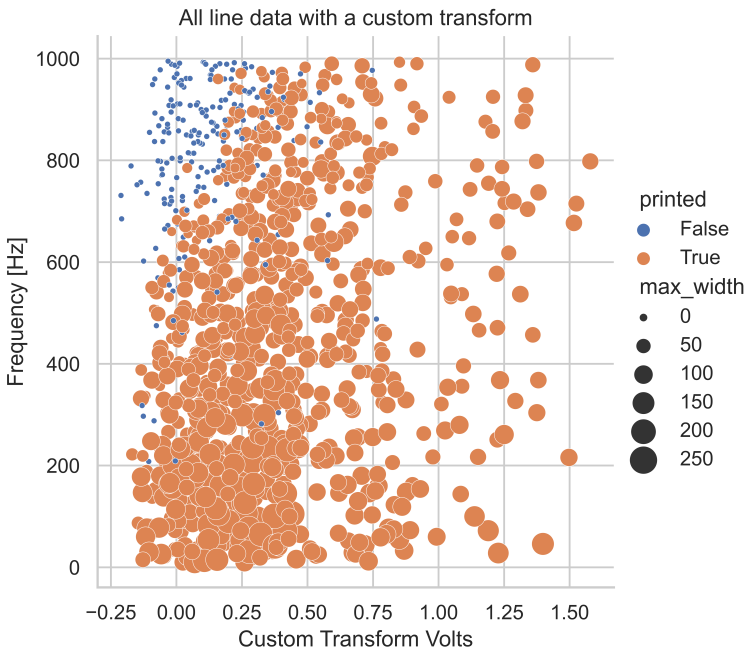


Figure 6.12: Plot of sine wave experiment inputs and print sizes after normalizing voltage with a custom transform.

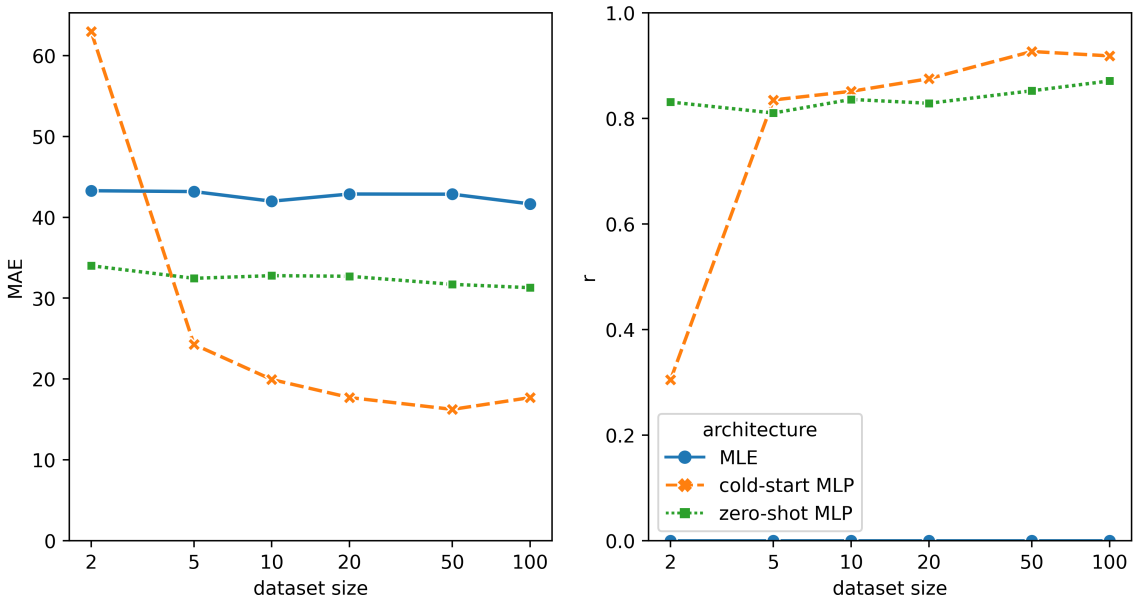


Figure 6.13: Performance of un-tuned regression models on sine wave datasets using a custom voltage transform.

experiments.

6.6 Dealing with Variation with a Mixture of Experts

Zero-shot models had proved capable of making print feature size predictions that were strongly correlated with reality as long as a capable data transformation could be identified that would homogenize the input space of the EHD datasets. With some tweaking, these models would go on to capably control real-world print runs. Nevertheless, there were gaps in their performance - most notably, the absolute error of zero-shot models remained high, barely better than the baseline MLE. Lining up the print/no-print boundary appeared to only partially homogenize the multiple datasets - additional factors affected the final footprint of printed features on the substrate.

Two likely candidates for factors independently influencing feature sizes were the SIJ nozzle aperture diameter, which could vary by $\pm$ 1-2 μm from one session to the next according to the manufacturer, and surface cleanliness, which could influence the ink contact angle and thus the amount a given volume of ink would spread on the substrate.

Curiously, the transfer learning methods had so far done little to improve MAE in few-shot learning. The warm start models summarized in Figure 6.8 were either equivalent to zero-shot models or worse. They seemed to inherit the worst of both worlds, combining the bias of zero-shot transfer with the overfit of small-N training. The warm-start MLP on normed inputs was able to make improvements over the similar zero-shot model when $N \geq 20$, but by that point it had been far outpaced by the cold-start model. Perhaps the square wave space was so low-dimensional, and the EHD process so inherently variable from task to task, that it was slower for a warm start model to *unlearn* its inherited bias from past tasks than it was for a cold neural network to learn the new task from scratch.

There were other possible approaches to transfer learning beyond the warm start. All of the pretraining to date had utilized a combined dataset of past experiments - as a result, warm start models came burdened with the same high error that plagued the zero-shot models. Instead, could a transfer model start from an ensemble of successful mono-task models? A model for such an architecture comes from the field of mixture-of-experts (MoE) models.

MoE models are a classical approach to modeling heterogeneous datasets that have successfully scaled from small bespoke problems to learning on vast datasets. In an MoE, many expert models are trained on different datasets or slices of a large dataset. On a given input to the ensemble model, a subset of experts makes a prediction. The different flavors of MoE are differentiated by the process used to select an expert or set of experts as a function of the input query. The first MoE was formulated by Hampshire and Waibel[57] for a phoneme classification task. Audio samples were gathered from multiple speakers, and each model trained on samples from a single speaker, becoming an "expert" on the spectral range of inputs of that individual. At deployment, a dedicated prediction engine would route novel inputs to one of the pretrained models based on spectral similarity to the original training dataset.

Subsequent developments often focused on extending the methods for routing inputs to specific experts, i.e., by using an exclusive gating layer[64] or a conditional Bayes classifier that is trained online with the expert models.[67] More recent work extended the MoE to a functional layer for deep architectures, using online gate layers or unsupervised clustering to route training data to different experts.[44, 129] All of these models utilize input-side routing, using their population of expert models to divide the input space in the presence of a heterogeneous sparse or very large range of inputs. Other MoE models focus on output heterogeneity, i.e., by deploying multiple task-specific gating networks[80] or rule-based multitask routing[9] to address many tasks with a single collection of expert models.

Unfortunately, neither input nor output characteristics alone would be helpful when routing EHD inputs to expert models because the input and output spaces were essentially equivalent across all EHD tasks, especially for homogenized inputs. Heterogeneity arose purely in the mapping from input to output. Luckily, because EHD regression was framed as a transfer learning task, an MoE model could assume that it would have access to some number of new training examples. This suggested an alternate expert selection strategy using a follow-the-leader criteria. At deployment, the algorithm will test the performance of each model and select an exemplar model that achieves the best performance on the retraining dataset.

Follow-the-leader is a simple model selection strategy in online convex optimization,

Table 6.1: Selected Hyperparameters

Data Type	**Model**	**learning rate init.**	α	**layer_1 neurons**	**batch size**
Norm Squares	cold-start	5.5E-04	1.5E-03	74	74
Norm Squares	zero-shot	4.5E-04	6.8E-05	200	72
Norm Squares	MoE-FtL	3.5E-04	1.0E-04	181	60
Transformed Sines	cold-start	8.6E-04	4.5E-04	425	64
Transformed Sines	zero-shot	4.5E-05	1.1E-04	218	71
Transformed Sines	MoE-FtL	2.0E-04	3.6E-04	330	72

where it has been characterized as the "simplest, most natural, and impractical online learning algorithm."[84] The strategy chooses model parameters based strictly on past performance, and it performs poorly in some circumstances, especially when the problem space is prone to evolve.[127, 20] Similar applications of accuracy estimation across an ensemble of models have been used to prune ensembles based on training accuracy[156] or by down-selecting based on dynamic measures of model confidence.[99]

Here, FtL would be applied to choose a selection vector across a set of pre-trained experts, and it could succeed if the solution space (i.e., the hidden process variables in an EHD print run) remained stable. In principle, the Mixture-of-Experts with a Follow-the-Leader gate (MoE-FtL) architecture would seek to identify the past dataset that was most similar to the deployment environment and use only that dataset to make future inferences.

Specifically, MoE-FtL begins M pretrained models $F = \{f_1, f_2, ..., f_M\}$ and a novel set of N inputs X and outputs Y drawn from a new task or process. The champion model f_c is selected per

$$c = argmin_m \sum_{i=1}^{N} (f_m(x_i) - y_i)^2$$

then only the champion performs inference for test inputs.

6.6.1 Contest of Experts Results

Each of the models in F was an MLP trained on data from a single task in the pretraining dataset, and the N-walk process evaluated each transfer learning model. Models trained on transformed data inputs, using the same square normalization procedure and sine wave voltage transform that had yielded the best zero-shot model performance. MLP models had four hyperparameters tuned for cold-start, zero-shot, and MoE-FtL applications for four hours each, as described in section 6.3.

All MLP models were two-layer fully connected networks with *tanh* activation functions, and their second layer of neurons was always 1/2 the size of the first layer. The ADAM solver was used with default beta values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was adjusted adaptively using cutbacks of $1/5\times$, and all models used an early stopping condition with a randomly selected 10% validation set. (If $N < 10$ for any training operation, early stopping was deactivated.)

Hyperparameters for the best-performing models are summarized in Table 6.1. Hyperparameter tuning tended to converge on strongly preferred batch size and learning rate values. Network size and α values had a less immediate effect on model performance and showed more significant variability amongst the highest-performing models from each tuning run.

Figure 6.14 and Figure 6.15 show the performance of the three tuned networks on the N-walk test. As before, zero-shot models provided high-correlation, high-error predictions in both datasets, and cold-start models converged to competitive performance on datasets with $N \geq 20$. But MoE-FtL models provided a new best performance in the low-N half of the evaluation, delivering the best possible MAE for $N \leq 10$ and matching the high correlation of the zero-shot models for every N. The MoE-FtL model is critically dependent on having a varied and reliable set of expert models to choose from when selecting an exemplar, so a limited number of monotask training data could severely throttle its performance. The ensembles above achieved this performance with six sine line experts and five square dot experts, respectively; there is likely ample opportunity for improvement if those numbers could increase.

A notable feature of the cold-start MLP becomes evident in these plots and has been

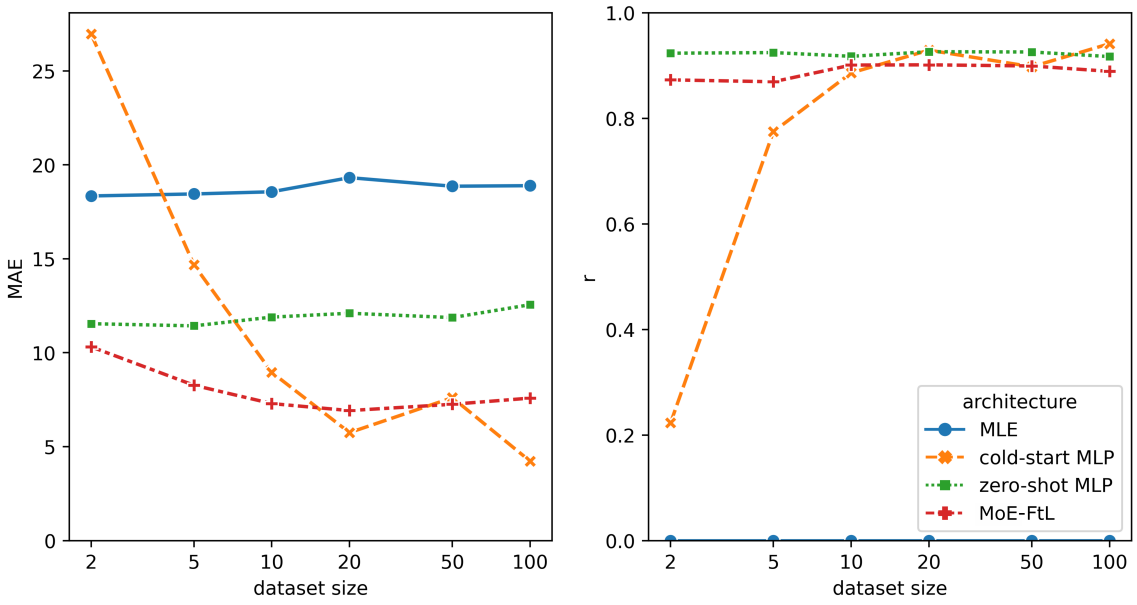


Figure 6.14: N-walk performance of tuned cold-start, zero-shot, and transfer learning MoE-FtL models on the square wave datasets.

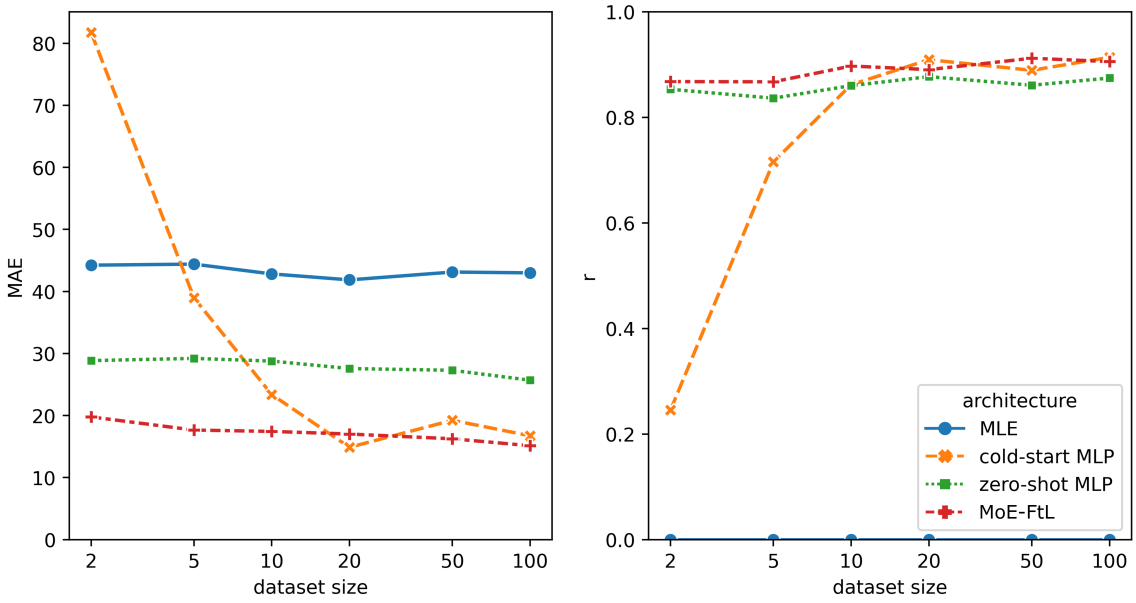


Figure 6.15: N-walk performance of tuned cold-start, zero-shot, and transfer learning MoE-FtL models on the sine wave datasets.

present to some degree in every N-walk experiment: performance experiences a noticeable decrease above $N = 20$. The feature is too consistent to be an artifact of experimental noise, and in this case, the hitch is sufficient to hand the MAE lead back to the MoE-FtL model in the sine line dataset for N=50 and 100. This type of backward trend should not happen in general - the cold-start models should perform strictly better with diminishing returns as N grows. A likely explanation comes from in-run drift and the effect of manual adjustments on the distribution of training points within each monotask dataset.

The SIJ-150 printer shows a consistent trend of reduced voltage sensitivity as a continuous print goes past about 10 minutes of run time. To accommodate, the human technician needs to gradually increase printing voltages as each run proceeds, usually by no more than 5-10V. This effect builds gradually, but when producing the multi-task EHD dataset in practice, the effect will cause a clog somewhere between the 70^{th} and 130^{th} waveform, causing the operator to bump up the overall operating voltage of experiments. Without the adjustment, prints would begin to skew heavily to no-prints, and clogs would run rampant. The result, however, is that most experimental runs contain two or more distinct distributions of inputs. The first is produced by the printer at its starting conditions, and the second (and possibly third, fourth, etc.) set is shifted upwards on the voltage axis.

At $N = 20$ and below, the majority of continuous-time samples drawn from a dataset will be contained entirely within either the first or second in-run distributions. By contrast, for large N, such as the full-dataset training sets of experts in the MoE-FtL model, a significant number of samples will be present from multiple distributions. However, at intermediate sample sizes, say $50 \leq N \leq 100$, a significant number of train/test subsets drawn from each monotask dataset will straddle the transition from one distribution of inputs to the next. This will have caused many test sets to include inputs that fall beyond the range of their training data, causing model performance to degrade. Therefore the step back in performance observed at $N = 50$ could be explained by that sample size creating the greatest proportion of distribution-straddled samples.

With the addition of the MoE-FtL to the set of models, the performance trends on the N-walk test finally resemble the expected behavior outlined in Figure 6.1. With no retraining data, only the zero-shot model is capable of making inferences, and it wins by default.

With any amount of retraining data available, the transfer-learning MoE-FtL architecture delivers the lowest average prediction error up to a critical training set size where the cold-start model edges it out. Square wave and sine line datasets, which both use 2-dimensional inputs, see this transition at about $N = 20$. Using the earlier estimate of critical dataset size, we might expect to see the transition from transfer to cold-start learning at around $N = 20^{\frac{d}{2}}$ in homogenized d-dimensional EHD control datasets.

All machine learning models, training procedures, and control methods are available at `https://github.com/onakanob/ehd-ml`.

6.6.2 Improving and advancing EHD ML

A few immediate opportunities exist to explore and improve the modeling space. Certain model types, abandoned earlier in the project, might be good performers in a MoE-FtL ensemble. In particular, the early RF models appeared to be less susceptible to the $N = 50$ performance dip; RF is likely more robust on just-out-of-distribution novel inputs, and gradient-boosted RF ensembles, in particular, could end up being best-in-class on EHD regression problems.

The MoE-FtL should be evaluated in true one-shot learning. The minimum $N = 2$ is simply a script limitation; many models will not function properly with only a single training input, but the MoE-FtL could choose a model based on a single sample assessment. It would be a reasonable restriction to limit single-point samples to waveforms that jetted ink.

The MoE-FtL could also be evaluated with a reduction in the number of training samples used to fit the expert models. Each expert would train on a continuous-time subset of samples drawn randomly from the existing datasets. $N = 20$ training points would be a sensible choice, and realistically this introduces a new pretrain size hyperparameter. This adjustment would increase the number of experts in the ensemble and reduce the degree of process drift present in each expert's supporting data.

In addition, multi- or soft-gating approaches could reduce variance when choosing the contest winners from the overall MoE-FtL ensemble. A k-winners approach would choose the lowest-k MSE models from the ensemble and average their outputs. A soft gate approach

would weight the contributions of each expert based on their performance. For instance, retraining would constitute calculating weights per

$$w = softmax\big((F(X) - Y)^{-\frac{1}{2}}\big)$$

and inference would multiply elementwise the outputs from M-large ensemble F and the weights:

$$\hat{y} = \sum_{M} wF(\hat{x})$$

Perhaps the strength of the weighted determination could even scale with the number of retraining samples, as in this naïve implementation:

$$w = softmax\big((F(X) - Y)^{-\frac{1}{N}}\big)$$

Exemplar models would be favored more strongly as N, and thus confidence in the contest results, grows large. Any increase in the number of inputs to the MoE-FtL output will also increase output bias, and the optimal choice of sampling strategy is likely to be sensitive to the number, variety, and granularity of expert models in the ensemble.

Finally, a sensible advancement to the MoE-FtL would be to incorporate cold-start training into its retraining loop. As N grows large, new models could be trained and appended to the pretrained ensemble of models. With a constant process, such an approach would smoothly transition from transfer learning to online learning as the new data becomes large enough to support performant models. This approach could also introduce some resilience to process drift, like that observed around $N = 50$ in the cold-start models, by allowing the model to fall back on pretrained trends when online models fall off in accuracy.

Compared to the state-of-the-art in MoE modeling, and indeed to most contemporary ML, the EHD problem was almost laughably simple, with its low-dimensional inputs and small datasets. However, the intense degree of hidden heterogeneity, process variation, and in-dataset drift presented unique challenges that may be representative of other manufacturing and processing environments. And the premium placed on rapid learning with tiny datasets emphasized few-shot transfer and simplicity in the modeling space. Expert-designed transformations were able to homogenize the EHD process inputs and enabled

high-correlation zero-shot modeling of the process space. Then, the MoE-FtL architecture - an MoE with a posterior performance-based single-pass gate - nearly matched the best model performance in transfer learning with only 2 to 20 retraining data points.

These model types provide an initial path forward for machine learning in a printing process with in-situ monitoring. However, to institute ML-driven control of the SIJ-150 printer, we would rely solely on zero-shot models and a print recipe generator.

Chapter 7

EHD PRINTING WITH MODEL-BASED CONTROL

Zero-shot EHD modeling had landed in a peculiar spot. These models strictly used past experiments to predict future printer behavior, receiving no new information except initial voltage and pulse width threshold measurements. Zero-shot models tested well in terms of correlation, but their absolute error remained high. Would these models be able to drive an actual micron-scale print process to a useful degree?

The hardware that had generated ML training datasets would not all be available for zero-shot control experiments. Switching rapidly between different waveforms to achieve varied feature sizes was central to the task, and no practical solutions were forthcoming when it came to synchronizing the standalone arbitrary function generator with the SIJ-150 printer. We would be falling back on base-level software to drive the system.

This next phase of experiments would be more bootstrapped, more accelerated, compared to previous work. We had a handful of artifacts: ensembles of ML models trained on square wave data; commercial printers and microscopes capable of producing micron-scale features; and a handful of 2D grid designs that demanded a wide variety of feature sizes. The task now was to tie these systems together with software and duct tape and to demonstrate that precise control of an additive EHD process with ML could be a practical reality.

7.1 Model-Based Control: Version One

The EHD model-based control algorithm sought to use a pre-existing predictive model to decide which inputs to provide the printer to print features of a specified width. Two feature types would be targeted: isolated dots with specified diameters and continuous lines with specified line widths. The algorithm would fully specify a collection of waveforms offline and then translate them to an SIJ recipe file which could direct the printer. This fully

zero-shot attempt would take no feedback from the live process and make no adjustments after the printing began. The only inputs to the system would be initial measurements of the voltage and pulse width thresholds after the ink was loaded into the print nozzle. The control module occupied the "predict" operation shown in Figure 7.1, using trained models and several target widths to produce an SIJ-readable recipe deck.

Two types of ML model ensembles were available: both had been trained on normalized square dot data and were a collection of the top performers from a hyperparameter optimization run. The classifiers would judge whether a given waveform would produce a printed feature at all, and the regressors would estimate the width of a dot printed by the square wave input. Regressor models were trained on only data points where an ink jet had been produced.

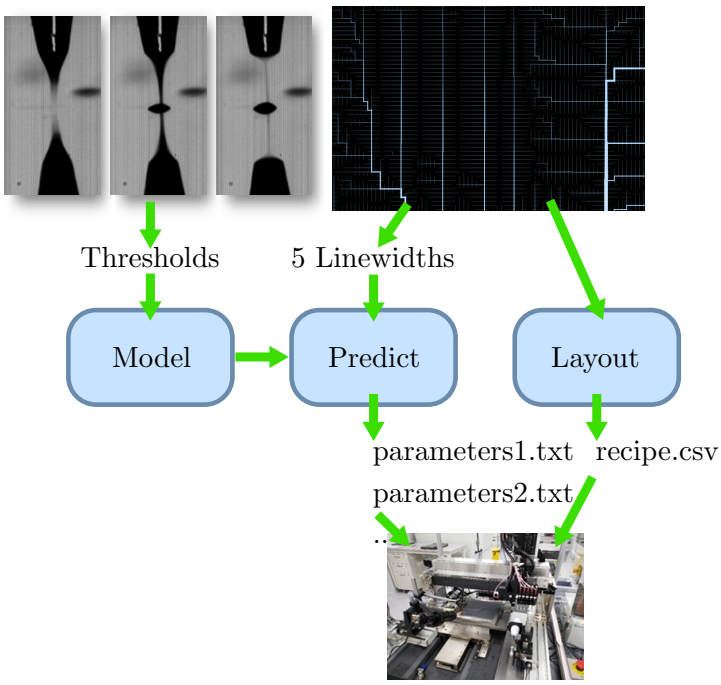


Figure 7.1: Schematic of the major operations to combine a multi-width grid pattern and an ML model to generate recipes for the SIJ-150 printer.

Each ensemble also reported a variance based on the spread of answers given by the models in the ensemble. Regression variance (var_r) was the standard deviation of model estimates with the top and bottom values removed. For the M classifiers g in the classifier ensemble, where each model produces either 0 or 1, indicating a failed (0) or successful (1) ink jet, the variance was the smaller distance from the ensemble mean to either 0 or 1. That is:

$$var_c(x) = min\big(1 - \frac{1}{M}\sum_{i=1}^{M} g_i(x), \frac{1}{M}\sum_{i=1}^{M} g_i(x)\big)$$

where x is an input waveform. The variances served as a confidence indicator - smaller values indicate better agreement between models.

A cost function was defined per

$$cost(w^*, x) = \left(F(x) - w^*\right)^2 + \beta_1\, var_c(x) + \beta_2\, var_r(x) + \beta_3\, x[1]$$

where F is the regressor ensemble, $x[1]$ is the pulse width portion of the waveform x, and constants β are user-tuned weights that balance the importance of the various terms in the cost function. The four terms, in order, motivate minimization of: (1) the difference between predicted feature width $F(x)$ and target width w^*; (2) uncertainty in the classification ensemble; (3) uncertainty in the regression ensemble; and (4) the pulse width. In practice, for this first outing of the control algorithm, the cost function weights were $\beta_1 = 0.25$, $\beta_2 = 0.025$, and $\beta_3 = 0.15$.

Limiting the variance of model outputs is intended to improve the overall accuracy of predictions and to nudge the control system away from possible no-prints. The fourth term, pulse width, was a bit of a wildcard. Minimizing pulse width had some direct benefits, such as increasing print speed and steering the algorithm away from low-voltage, clog-prone inputs. But its primary purpose in this trial was to stand in for general process optimization terms.

One of the main benefits of an offline, model-based recipe generator would be the ability to optimize for multiple goals at once. For instance, a cost function could include terms to steer the process towards conditions that increase line conductivity, maximize aspect ratios, or avoid high voltages for a sensitive ink. Absent a model for any of those effects with the current printer, the width minimization term was a simple choice to introduce process optimization into the control landscape.

With a cost function defined, the algorithm took a pre-calculated approach to selecting control inputs. About 10,000 potential input waveforms, concatenated in tensor X_{grid}, were chosen in a grid over V, W control space. For each coordinate, a jetting prediction was made by the classification ensemble - all points with inference values $G(x) < 0.5$ (i.e., where the ensemble thought jetting would not occur) were removed from the set of possible inputs. Then for each of several target widths, the input x was chosen from X_{grid} that minimized

the cost function. In pseudo-code:

```
def control(Widths):
    # Choose waveforms to produce an array of widths
        # First set the search space
        Waves = make_a_grid()

        # Remove points predicted not to print
        jets = classifiers.predict(Waves) > 0.5
        Waves = Waves[jets]

        # Choose a controlling wave for each target width
        inputs = []
        for w in Widths:
                best_idx = argmin(cost(w, Waves))
                inputs.append(Waves[best_idx])

        return inputs
```

7.1.1 Dot Printing Results

Figure 7.2 displays the first printed dots test pattern produced with ML-derived recipes. This pattern was intended to be a logarithmic walk of 5 target sizes between 2 and 100 μm diameter, but absolute size accuracy was not the only goal of the optimization loop. Due to the input of the classifier models, the algorithm excluded waveforms that could produce very small features and would target no smaller than 8 μm. Due to the upper limit of the pre-computed grid search, the largest diameter that the control loop could target was 58 μm. For the four target widths within these bounds, the diameters printed by the ML-produced waveforms are shown in the left pane of Figure 7.2 and Table 7.1.

These results contained a significant degree of error present. A true zero-shot approach seemed prone to extensive bias, producing results that were off by factors of up to 4.4 and

Table 7.1: Target and Measured Dot Sizes

Target [μm]	Mean Width [μm]
8	28.9
15	65.3
40	83.8
58	109.3

error from 21 to 51 μm. But these results were downright encouraging compared to the first attempted prints of continuous lines.

7.1.2 Line Printing Methods and Results

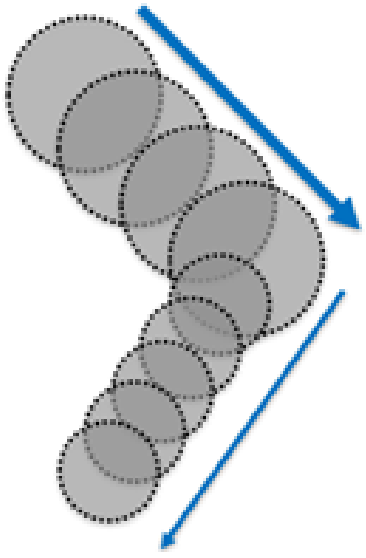


Figure 7.3: Illustration of the strategy for converting dot predictions to printed lines.

Two technical adjustments were needed to perform ML-guided line printing. The first was needed to adopt patterns to the SIJ printing environment. Ramified solar cell front grids like those produced in chapter 3 were a good test case for the ML-driven control of line thickness. These patterns demanded the precise placement of hundreds of interconnected lines, each with a specified thickness. Hardware limitations prevented any attempt at exactly matching every line in a pattern, limiting each print run to at most nine different waveforms. To choose which line widths would be targeted, the whole bag of line widths from a pattern was divided into a handful of bins, each with a target width decided by the generalized Lloyd algorithm, which searches for an optimal quantization of a set of vectors. (Ideally, the algorithm minimizes the average MSE between the widths in the original pattern and their nearest bin width.)

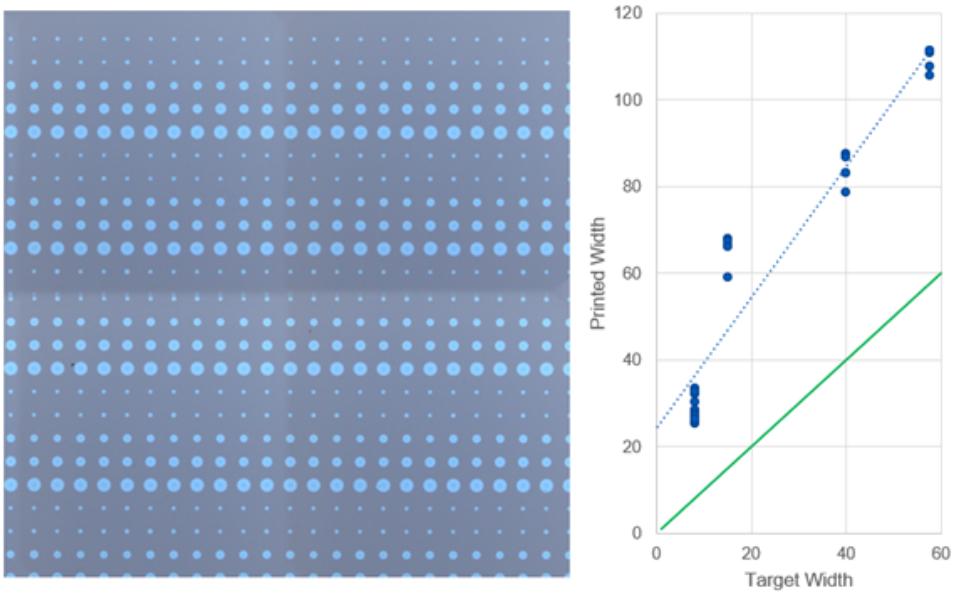


Figure 7.2: (Left) dot test pattern, and (right) target vs measured dot diameters from version 1 ML-controlled recipes.

The second technical addition was a fix to enable the use of models trained on square wave dot datasets to direct continuous line prints. Figure 7.3 illustrates the approach: dots would be printed with a controlled overlap to trace out continuous lines on the substrate. A square waveform $x = [V, W]$ was chosen as in the previous section, with w^* equal to the target line width. Then a 25% repeating square wave would be defined for the printer to execute. The wave was defined with voltage V and frequency $f = \frac{1}{4W}$ to produce the specified square wave, and print would execute at a velocity of Bfw^* microns per second, where B was a pitch parameter that determined the degree of overlap between neighboring dots. In the first attempted outings of the algorithm, $B = 0.5$, creating the 50% overlap pictured in Figure 7.3.

Figure 7.4 illustrates a result from the first line printing attempt. The grid pattern was quantized into just three bins for this pattern, with target and measured widths listed in Table 7.2. With an error over 100 microns, the printed lines were completely different from their target sizes.

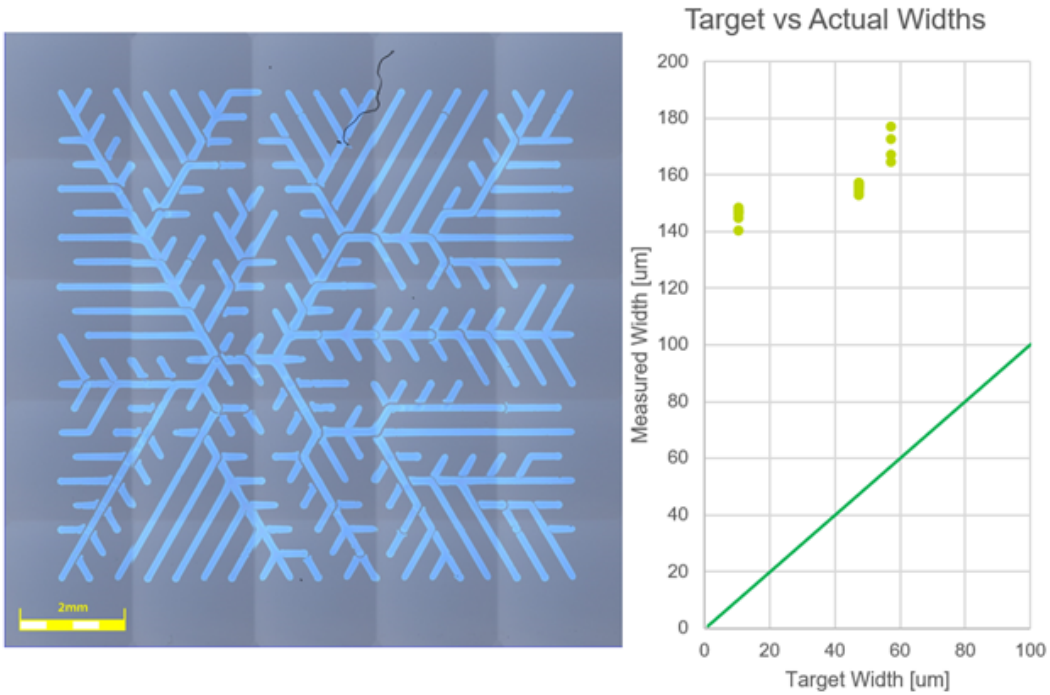


Figure 7.4: (Left) printed test grid pattern, and (right) target vs measured line widths from version 1 ML-controlled recipes.

Table 7.2: Target and Measured Line Widths

Target [μm]	Average Width [μm]
10.7	145.6
47.8	155.4
57.6	170.3

These zero-shot models would not be very useful when printing precise patterns of dots or lines. What was the cause of the gap - large for dots, massive for lines - between model predictions and printed reality?

A good proportion of the prediction error, at least in the dot test pattern, could be attributed to pure model bias. As had been apparent in N-walk model evaluation with cold-start models, the absolute error was necessarily high in zero-shot learning because the model had no way to compensate for day-of effects like contact angle or the specifics of the ink that had been loaded into the print nozzle. This error had fallen in the range 10 to 20 μm in offline model evaluation, and the result here was only slightly on the high end of that expected error.

A second effect was also likely: state memory that carried from one ink jet to the next. Memory effects could include a host of physical phenomena that would make it easier for an ink jet to form in the immediate aftermath of another jet. This could include lingering polarization of the nanoparticles in the ink, hydrostatic pressure in the ink column, and physical extension of the meniscus from the nozzle aperture. All training datasets had been produced at a steady dot deposition rate of 2 Hz, but the drop ejection frequency was slightly higher for the dots test pattern and significantly higher for the lines - perhaps on the order of 10 Hz and 100 Hz, respectively. State memory effects were a likely cause of the massive size overshoot in the initial line prints.

For all ML-guided printing, it was apparent that some degree of online adjustment would be needed to walk the process closer to target widths. Luckily, the high Pearson correlation values observed during N-walk testing appeared to be intact here. Even while the absolute error was high, the printed features were more or less proportional to each other and their original targets. Perhaps a variant of classic proportional control methods could be used to tune the process.

Line printing, however, would need some additional attention. With the much higher printing frequencies involved, line printing appeared to be a fundamentally different process than isolated dots. These results motivated us to collect a new set of waveform experiments, the sine line tasks described in subsection 5.2.3, to find the custom homogenizing transform described in subsection 6.5.1, and retrain regression ensembles on a dedicated line-printing

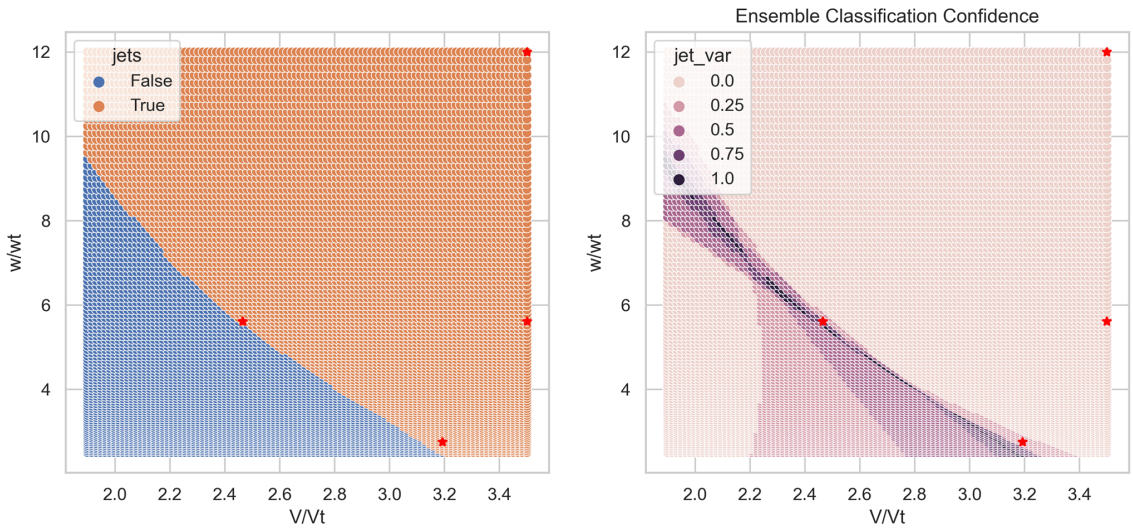


Figure 7.5: Map of classifier predictions (left) and confidence (right) over an area of normalized square waves.

dataset before re-attempting controlled line prints.

7.2 A Visual Tour of EHD Process Space

The pre-calculated grid of possible input values to the EHD printer provides a good visualization of the decision landscape that drives waveform choice. Figure 7.5 displays the outputs of the classification ensemble, the jet/no-jet decision boundary on the left and the ensemble confidence metric on the right. Figure 7.6 shows the error between feature width predictions and a target width of 15 μm on the left and ensemble variance on the right. These are the combined factors that drive waveform choice through cost function minimization - the control algorithm sought to minimize the square of prediction error (i.e., print the size that the pattern specifies) while also minimizing both confidence metrics. Additionally, because the waveform search only includes points with classification output > 0.5, the algorithm is limited to the orange points in the classifier outputs. The classification confidence term nudges waveforms away from the print/no-print boundary, and the regression confi-

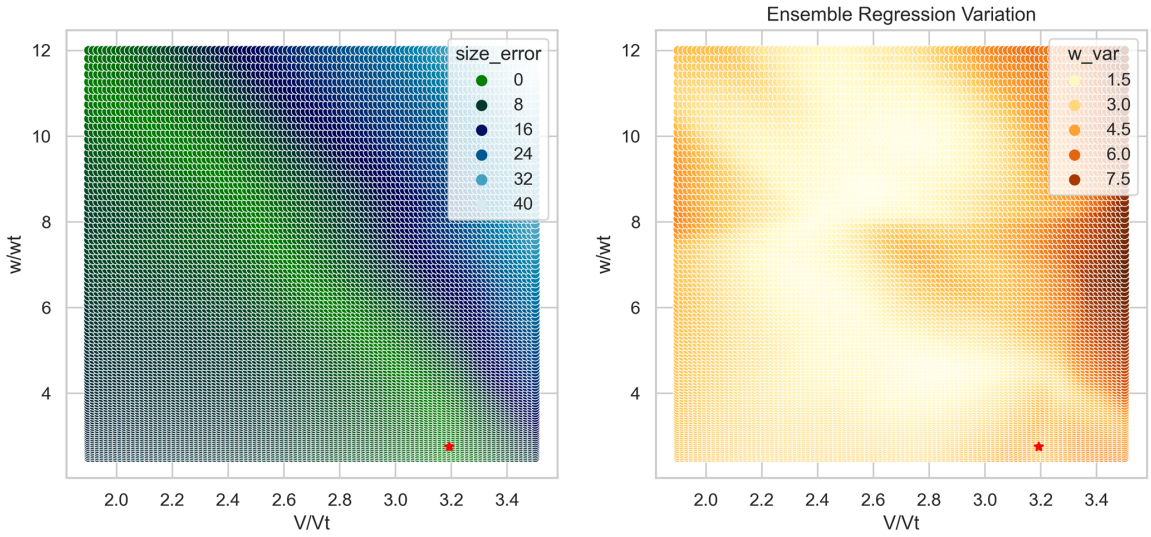


Figure 7.6: Map of regression prediction error (left) and confidence (right) over an area of normalized square waves and a target width of 15 μm.

dence term pushes waveforms away from the periphery of the search space. (Ideally, these terms limit output to regions of process space where the models in the regression ensemble have a high density of training data support, which in turn should smooth out and improve their predictions.)

The four red stars in Figure 7.5 indicate the four chosen waveforms from the first dot printing experiment, with target widths of 8, 15, 40, and 58 μm. 8 μm, the leftmost point, is the smallest possible model prediction on the grid and 58 μm, in the top right, is the largest. Rather than tracing a straight line between these two extremes, the middle-valued waveforms fall along a swooping trajectory that detours through to the bottom-right quadrant of the chart. The steering algorithm chooses this path because of the final term in the cost function, $\beta_3 x[1]$, which motivates the control algorithm to opt for the smallest possible pulse width.

Interestingly, as the algorithm moves from the 8 μm to the 15 μm target width, the rapid pulse term causes the chosen waveform to increase in voltage but to decrease in pulse width.

For a human operating an EHD printer, this would be a nontrivial operation: doubling a printed feature's size by *reducing* the waveform pulse width and increasing voltage to compensate is no easy task. It would typically take many repeated trials to arrive at the second waveform. While the actual error was high in practice, the ML-guided algorithm could successfully produce the correct proportional difference between these two operations. This feat was a crucial sign of life for the ML-guided printing approach.

The left side of Figure 7.6 illustrates how the model arrived at the waveform for the 15 μm dot. It chose from points along a top-left to bottom-right green isoline of waveforms that produce 15 μm width predictions, and of course, the algorithm chose the smallest pulse width that did not involve too much ensemble uncertainty. The shape of the 15 μm isoline is linear or slightly convex, in contrast to the highly concave boundary between jetting and no-jetting. The overlap of these two shapes, one from the regression and one from the classification ensemble, could explain why the smallest possible point of 8 μm lies where it does in process space. The 8 μm isoline must be the smallest isoline to intersect the region of predicted printing by the classifiers, and we would expect it to intersect the print/no-print boundary at a tangent, intersecting the space of allowed waveforms at precisely one coordinate. That intersection, the controlling waveform for the 8 μm target dot, would correspond to an optimal balance point between voltage and pulse width to produce small features.

Now, to attempt to infer any universally optimal processing conditions based on this single set of models would be a stretch. But in just this case, according to these particular model ensembles, the smallest print occurred at a normalized voltage of 2.45 and normalized pulse width of 5.5. Taking typical values of $V_t = 100V$ and $W_t = 14\,ms$, we might expect the smallest-feature square wave to land at 245V and 80 ms.

7.2.1 Control along a cost-optimal path

The path traced out by the procession of algorithmically chosen process points through waveform space is a cost-function-optimal manifold defined by the combined components of the optimization routine: ML ensembles, cost function, and waveform selection algorithm.

The initial outputs of that routine were prone to print size errors and needed to be adjusted based on observation. In other words, this was now a classical control problem. However, rather than choosing one control knob as the primary input (choosing to only adjust voltage or pulse width, in this case), we would like to keep the process confined to the cost-optimal path. That path represents the best guess (contingent on model accuracy) for the optimal balance between model confidence, accuracy, and other process imperatives that could drive improved cost, time, or efficiency or link to external requirements in a manufacturing environment.

One approach would be to approximate the cost-optimal path using a parametric function or spline in process space, then step up and down along the approximated path to fine-tune the process. But there is a more straightforward option: we could use the target width w^* as the primary control input to the printer. This naturally locks the process onto the cost-optimal path by only ever using algorithmic outputs to drive the process, and it frees us from introducing new code or modeling between the established ML algorithm and the print process. It also makes it easy to accommodate updates to the model ensembles if, for instance, transfer learning is added to the loop in the future.

On the other hand, controlling directly with the model brings some risks. Most critically, the path it charts out through process space is not guaranteed to be continuous. It is likely, in fact, that the model-defined cost-optimal path will jump around in the process volume; and this effect would only grow more dominant if the process space were to take on additional dimensions. These discontinuities would introduce hitches in the printer control landscape, where small adjustments to the input parameter w^* result in disproportionate leaps in waveform choice and feature size.

7.3 Model-Based Control: Version Two

7.3.1 Proportional Control Updates

Several changes occurred between the first and second attempts at model-based control. The most significant, of course, was the inclusion of an entirely new dataset of sine line waveforms and line features. A different dedicated regression ensemble would now be deployed for dots,

using square pulses, than for lines, using sine waves. No classifiers were deployed this time, and regressors were trained on entire datasets, including non-jetting samples. This was intended to allow more exploration in the small features region of process space. (In a development environment, I would rather try and fail to print small dots than be prevented from the attempt by the classifier decision boundary.)

The pre-computed grid of waveforms was no more, along with its limitations on the resolution of control steps and upper bounds on voltage, pulse width, and frequency. Instead, a numerical optimizer would choose a waveform for each feature width set point. The waveform chosen for target w^* was

$$\hat{x} = argmin_x \, cost(x, w^*)$$

with new cost functions defined for square waves,

$$cost_{dot}(w^*, x) = \big(F(x) - w^*\big)^2 + \beta_1 \, var_r(x)^2 + \beta_2 \, x[1]$$

and for sine waves,

$$cost_{line}(w^*, x) = \big(F(x) - w^*\big)^2 + \beta_1 \, var_r(x)^2 + \beta_2 \, (\upsilon^* - x[1])^2$$

where υ^* was a target frequency.

For both dots and lines, $\hat{x}$ was chosen by the SciPy[121] implementation of the bounded Nelder-Mead algorithm. The parameters used in each cost and optimization function are summarized in Table 7.3. All parameters were tuned by hand.

Unlike the first round, this set of dot prints attempted to use the longest possible pulse width to avoid a potential failure mode where the SIJ-150 would play a waveform twice when pulses were below 10 ms. The algorithm targeted 600 Hz to stabilize the line print outputs around a common frequency while retaining enough flexibility that low frequencies could still be invoked to produce large feature widths.

Finally, an outer loop was needed to adjust the printing process by adjusting the target width as the primary input. The first design for this process was straightforward and required no additional tuning or parameter tuning: the algorithm would make proportional adjustments based on the observed width of each print. In other words, given a true target

Table 7.3: Control Loop Parameters.

Param	Dot printing	Line printing
β_1	0.1	0.012
β_2	-0.3	0.001
v^*	n/a	600
x_0	[3, 3]	[0, 500]
bounds	[(1, 10), (1, 30)]	[(-3, 3), (20, 1000)]

width w^*, a previous set point of $\hat{w}_i$, and a measured feature width of w_{meas}, the next control input would be

$$\hat{w}_{i+1} = \hat{w}_i \frac{w^*}{w_i^{meas}}$$

The first iteration would run with $\hat{w}_0 = w^*$, and at each step, the printer would execute the waveform chosen by

$$\hat{x} = argmin_x \, cost(x, \hat{w})$$

This was a somewhat naïve approach! The control steps would tend to be relatively large, and there was no damping parameter to prevent the system from flying out of control. There was an implicit assumption of linearity in the system's response to the $\hat{w}$ parameter. So if the regression model ensembles' outputs could maintain a linear correlation with actual feature sizes over a range of inputs, this approach was likely to enable very fast convergence on the target width.

With no in-situ monitoring in place, running the update loop was a laborious process that required me to home the printer, unmount the substrate, walk to a microscope, locate the printed test pattern, measure each feature, walk to a computer, enter the latest w^{meas} values, generate a new recipe, copy it to print PC, walk to the SIJ printer, re-mount the substrate, re-load a new run, reset the nozzle height, unclog the nozzle, and execute the new recipe. So this was not exactly a sustainable solution, and the number of control updates would never be able to exceed one or two trips around the circuit. (It also helped to book out all those tools on the weekend.)

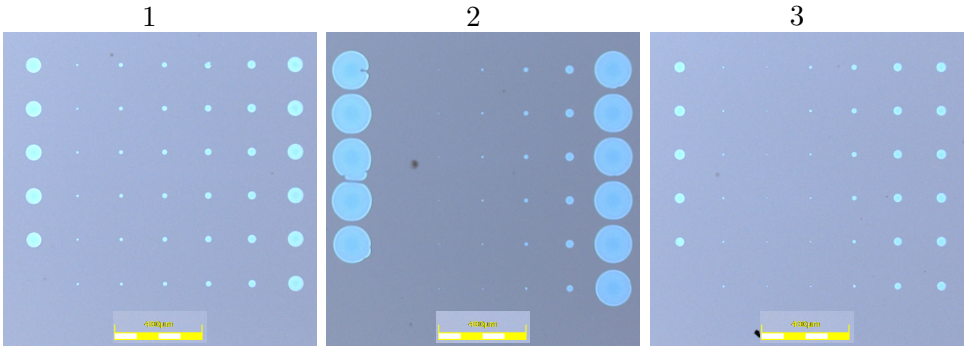


Figure 7.7: Micrographs of three printed dot test patterns.

The lengthy measurement process also introduced variance into the process: nozzle height, ink age, and substrate placement were all reset from each trial to the next. Despite the drawbacks, this setup at least had the potential to demonstrate the feasibility of using proportional updates model-based optimization set points to achieve targeted feature sizes.

7.3.2 Control Version Two Results

Figure 7.7 displays three sequential test patterns from the first test of the new control loop in dots, with targets and measurements shown in Table 7.4. The tests ran from left to right, first printing a row with waveform 6 (the largest target) as a de-clog step, then stepping through the target widths from 2 to 70 μm. The bookend targets were not successful in finding their desired feature widths. The 2 μm target vacillated between printing too large and printing nothing. The 70 μm target, despite starting only 3 μm from the mark, appeared to be destined for an uncontrolled oscillation that is the hallmark of an over-tuned control loop. (When a width of zero was measured, the control value $\hat{w}$ automatically adjusted by a factor of 2 to avoid divide-by-zero errors in the control step.)

The middle four set points, however, hit the mark almost immediately. Starting from an average error of nearly 10 μm, they converged to an average error of $< 2\mu m$ after only one proportional adjustment and $< 1\mu m$ after two adjustments. The first control adjustment for the 5 μm target point was a frightening factor of 0.28×, and yet the resulting feature

Table 7.4: Dot Widths Over Proportional Control Updates

Target [um]	Trial 1 [um]	Trial 2 [um]	Trial 3 [um]
2	10	0	10
5	18	6	6
10	23	8.5	10
20	31	17.5	22
40	39	39	40
70	73	177	46
Mean Error	8.2	19.2	5.8
Mean Error (middle four)	9.5	1.5	0.75

was within 1 μm of the target. The experiment demonstrated that the proportional update loop would cause these values to converge on their target widths in a single step and remain there stably; it was a success.

Figure 7.8 shows the final test pattern of the first successful line control algorithm, with targets and trial measurements shown in Table 7.5. This set of target widths was taken from a 5-bin quantization of a solar cell grid design. As before, the largest target width features showed unpredictable behavior from one trial to the next. However, for the four target widths in the range $[20, 143]$, the mean size error decreased from 25 μm in the initial attempt to 4.75 μm in the third trial. This particular print appeared particularly susceptible to drift from one trial to the next - nonetheless, the control loop was able to drastically improve print size accuracy after just one or two adjustments by the proportional update rule.

7.3.3 Discussion of the Updated Control Method

Single-shot adjustments with a simple proportional update rule for width set points successfully controlled the feature sizes of both dots and lines to within 2 and 10 μm, respectively. However, both methods fell apart at the extrema of target widths, with dots failing to con-

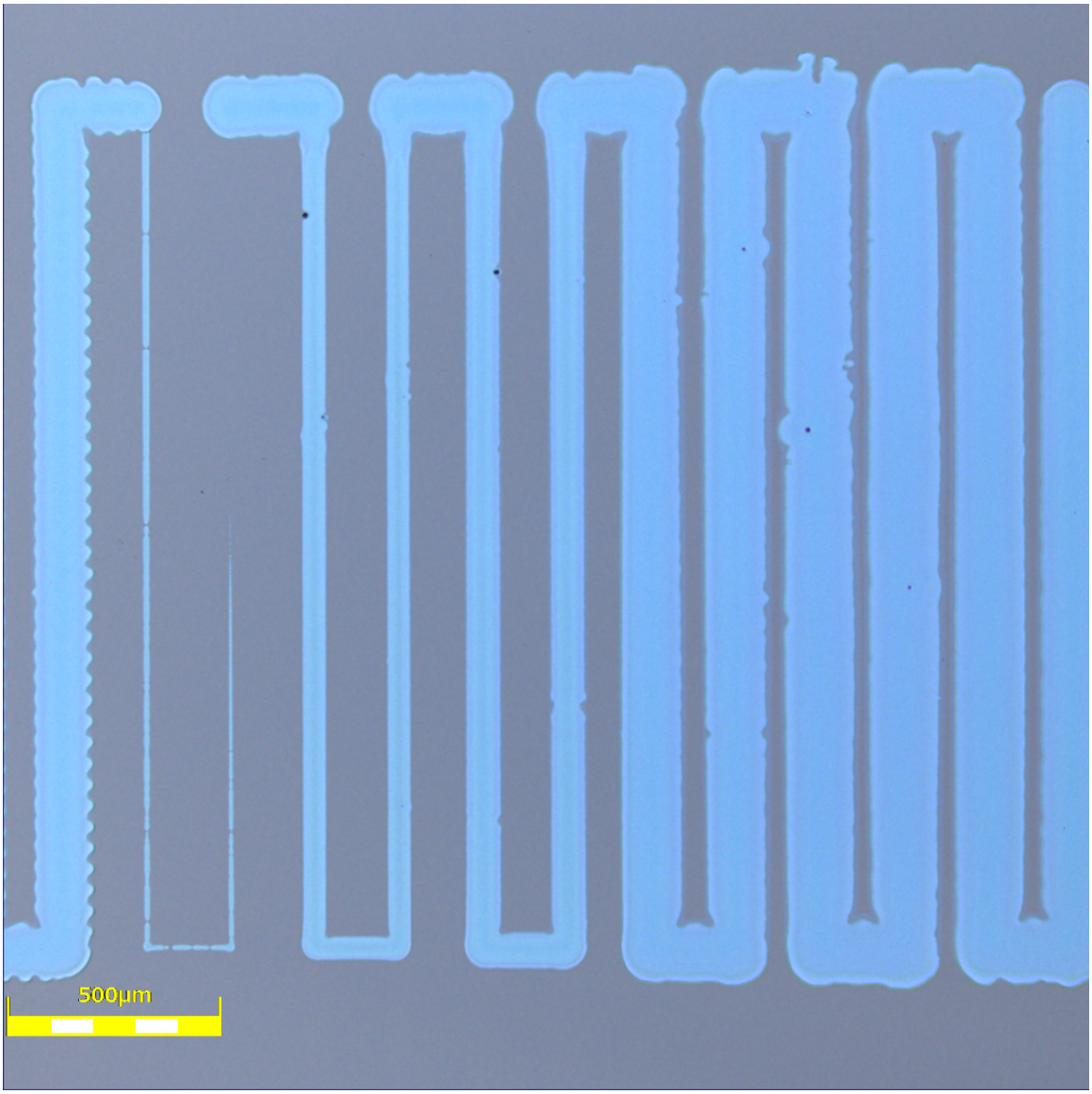


Figure 7.8: Final pattern from a test of controlled line printing.

Table 7.5: Line Widths Over Proportional Control Updates

Target [um]	Trial 1 [um]	Trial 2 [um]	Trial 3 [um]
20	60	18	16
54	83	60	55
86	105	78	80
143	155	122	135
200	180	145	155
Mean Error	24.0	18.4	12.8
Mean Error (top four)	25	9.25	4.75

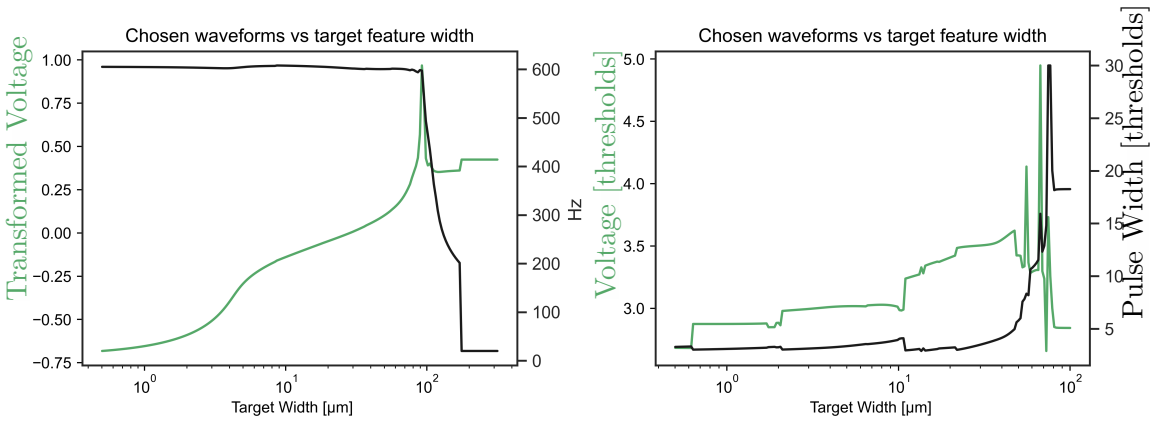


Figure 7.9: Target feature widths vs chosen waveform voltages (in green) and time parameters from the sine line (left) and square wave dot (right) control loops.

trol below 5 μm or above 40 μm and lines failing to hit targets over 140 μm. Figure 7.9 displays chosen waveforms for different target feature widths for the sine wave lines (left) and square wave dots (right), and it may provide some insight into these limits.

Because waveforms were set by a numerical optimizer, control values are not guaranteed to be global optima in waveform space. The outputs of the control loop were susceptible to getting stuck in local minima, jumping between different families of solutions, and would have been sensitive to starting position. To make matters worse, the underlying optimization landscape contained all of the noise and uncertainty produced by an ensemble of models, and the variance metric, in particular, would have introduced a fair amount of noise to the optimization process. The square wave models seemed particularly susceptible to such effects, and notable jumps in the ML-selected waveform occurred at 2, 10, and 50 μm. In addition, both models became highly unstable as the target size grew large before hitting a rail at almost 200 μm for the sine waves and about 80 μm for the squares. The square waves seemed to rail out below 2 μm. These effects were likely caused by falloffs in the amount of training data for these sizes. Square wave dot features mainly fell between 2 and 100 μm in diameter in the training set, but with sparse representation above about 60 μm, and sine line widths were strictly between 20 and 250 μm, with a falloff in their representation above 200 μm. Outside of those bounds, the output of the ML ensembles would have become unstable, the optimizer's behavior erratic, and any attempt at model-based control futile.

The lower limit of 20 μm for line printing also corresponded to a lack of supporting data in the ML training datasets, but it was unique from the other size limits in that it appeared to be a physical limitation of the system. The histogram in Figure 5.6 shows that lines with a width ¡ 20 μm were not encountered during any line printing experiments - that minimum was likely due to a combination of factors, including a minimum sustainable ink flow from a standard-size SIJ nozzle combined with the chosen print velocity of 500 $\mu m/s$.

Nonetheless, a principle result of the experiment was promising: ML-guided control of the EHD process was successful with zero-shot models plus proportional control steps provided that target feature sizes were within the well-supported domain of the ML training datasets.

7.4 ML-Directed Demonstration Patterns

7.4.1 Multi-width lenses and interconnects

With one online control step, ML-driven printing was able to produce reliable printed features with about one decade of size control for both square wave dots and sine wave lines. This was sufficient to attempt printing some larger test patterns. Figure 7.10 displays examples of several test prints. At the top, two varieties of optimized ramified solar cell grids are shown on hexagonal and square meshes, along with visualizations of the grid designs. Both patterns are 2 cm to a side, and while the quantized widths do not fully replicate the line tapers from the original patterns, they reproduce both the largest and the smallest features called for in the original patterns. At the bottom is a portion of a meta-surface lens pattern designed to focus infrared light.[159] This pattern is scaled up to accommodate the smallest dot size that could be produced with that day's setup - it deploys 4, 8, 12, and 16 μm diameter features. Unlike previous additive attempts to produce patterns with varied sizes on EHD printers, all demonstration patterns were printed in a single pass, producing each successive feature size on demand by switching between the tuned ML-produced waveforms.

Unfortunately, these patterns were plagued by missing features or, in the case of the lense, are simply incomplete. This was due to our old friends: process drift and clogging. A less obvious error is that these grids patterns are not optimal for the properties of the printed lines. All grid optimizations done to this point had used textbook values and geometric assumptions to model conductivity and scaling properties, but the true tradeoffs between conductivity and line width (and thus shadowing) were still unknown. An optimal grid pattern needed to optimize its morphology for the specific physical system that would be used to produce it.

7.4.2 Printed Line Conductivity

The final practical to check of printed metal line function was a verification of their conductivity. EHD-printed lines need to move current across the solar cell surface for solar cell front grid applications - to create metal lines with decent conductivity, test patterns were produced using repeated overprinting.

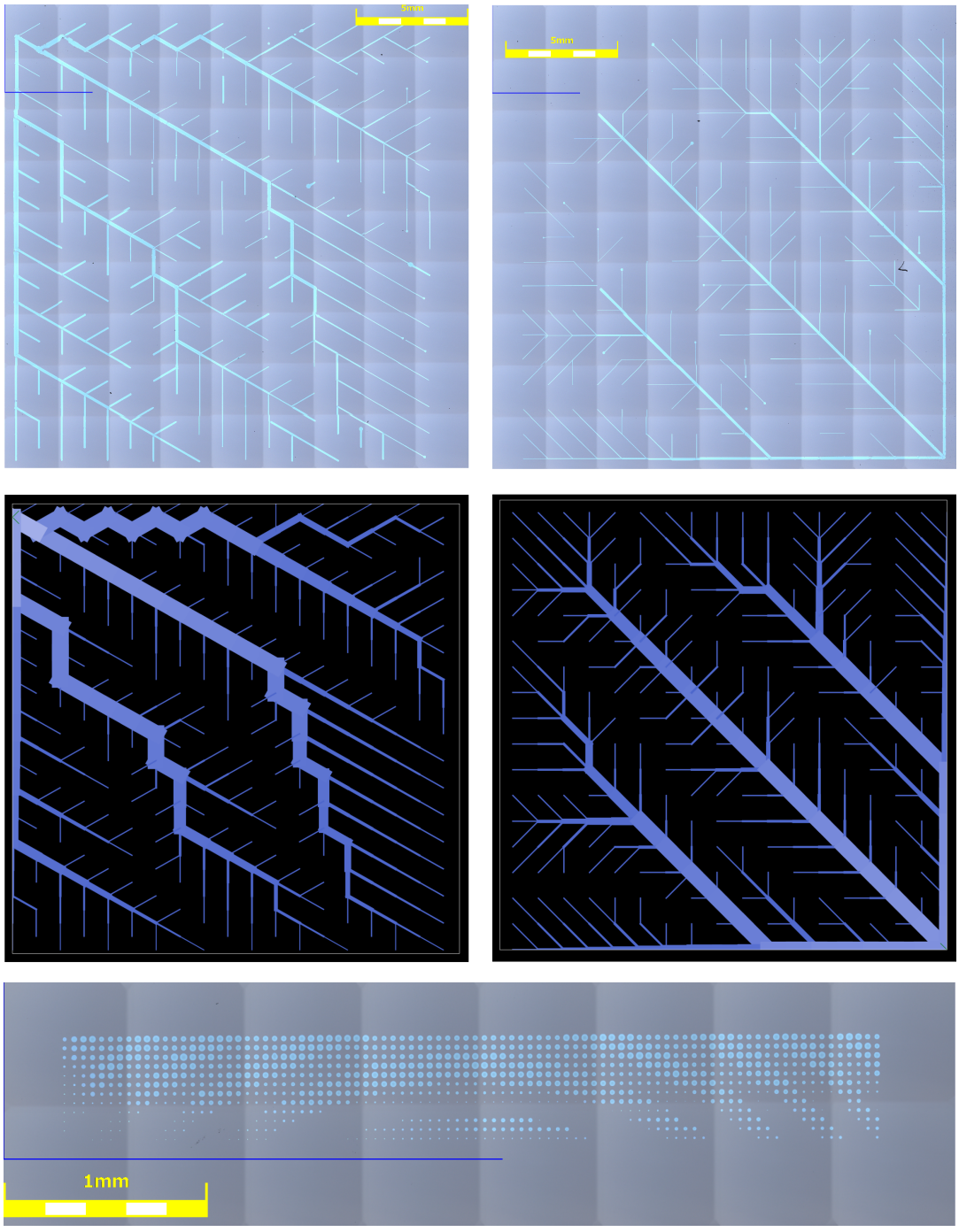


Figure 7.10: Demonstration patterns of lines (top) and dots (bottom) with ML-controlled feature size.

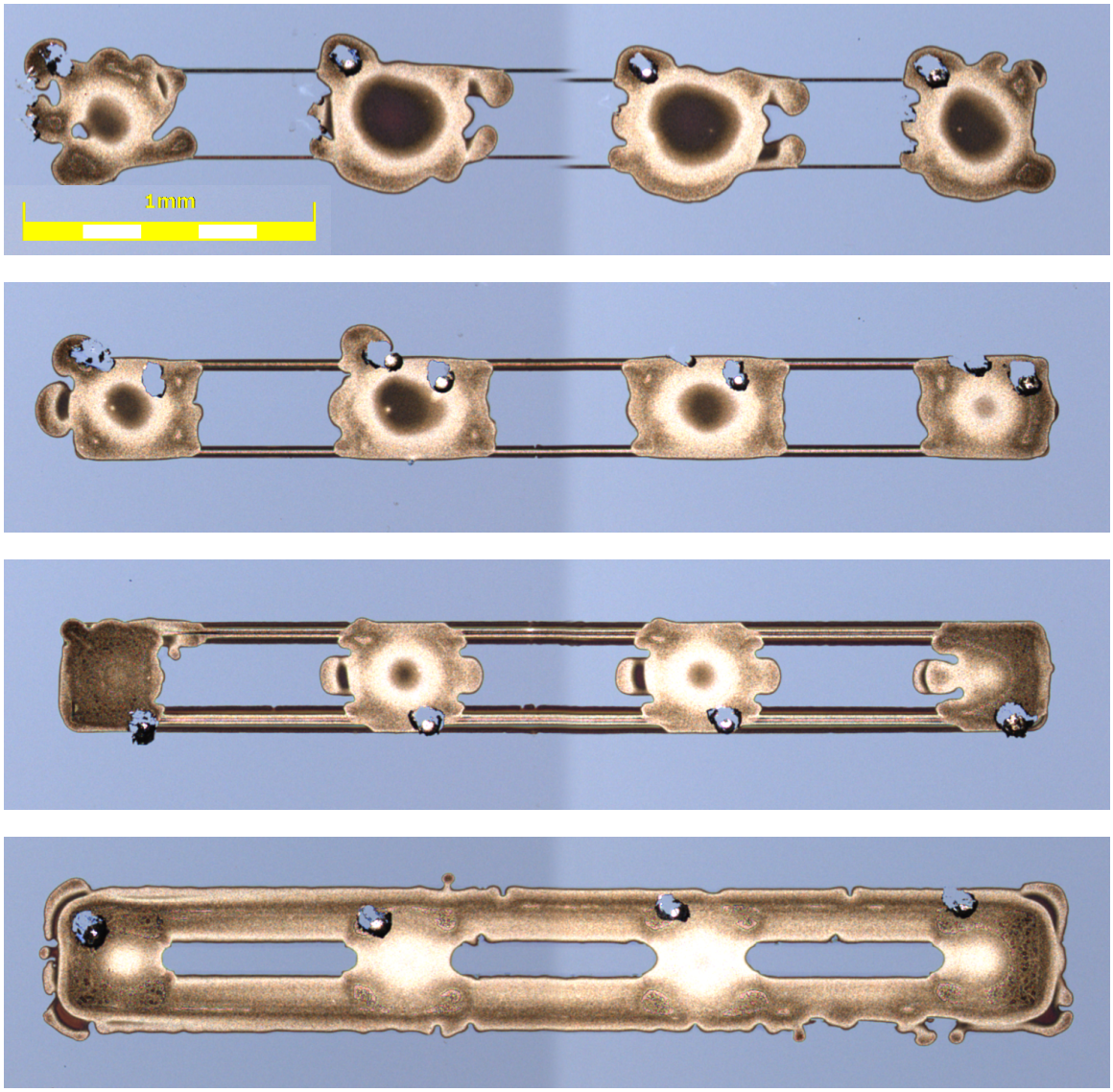


Figure 7.11: 4-point conductivity test patterns printed using ML-derived waveforms and AGK-104 ink. From top to bottom, line widths are 14, 39, 87, and 185 μm.

Table 7.6: Electrical properties of printed lines

Width [μm]	Resistivity [Ω/cm]	Conductivity [S-cm]
14	411.0	2.4E-03
39	212.3	4.7E-03
46	289.4	3.5E-03
70	64.7	1.5E-02
87	61.1	1.6E-02
99	27.3	3.7E-02
143	26.6	3.8E-02
182	7.9	1.3E-01
185	9.0	1.1E-01

Maximizing each line's aspect ratio will generally improve line conductance without introducing additional shadowing, and EHD printing can produce very tall features at the cost of additional printing time. [120] and others have noted that EHD-printed features can grow to almost arbitrarily tall with repeated deposition in one location without spreading on a substrate. Inks tend to stay within the footprint of the first print pass, at least partly due to a coffee staining effect that causes particles to pile up around the perimeter of the first print pass. Precise size control when depositing on a bare substrate is especially beneficial for overprinting because the initial print pattern locks in the footprint for future over-prints of the same pattern. To test the conductivity of deposited silver lines with AGK-104 nanoparticle ink, 4-point conductivity test patterns were created using multiple ML-generated waveforms. Each pattern was printed with ten passes over the same path, and the four contact pads in each pattern were deposited with a high voltage wave.

Figure 7.11 displays some examples of printed conductivity test patterns, and relationships between line thickness and electrical transport properties are shown in Table 7.6 and Figure 7.12. The trend of increasing conductivity with increasing line width is consistent with an exponential trend.

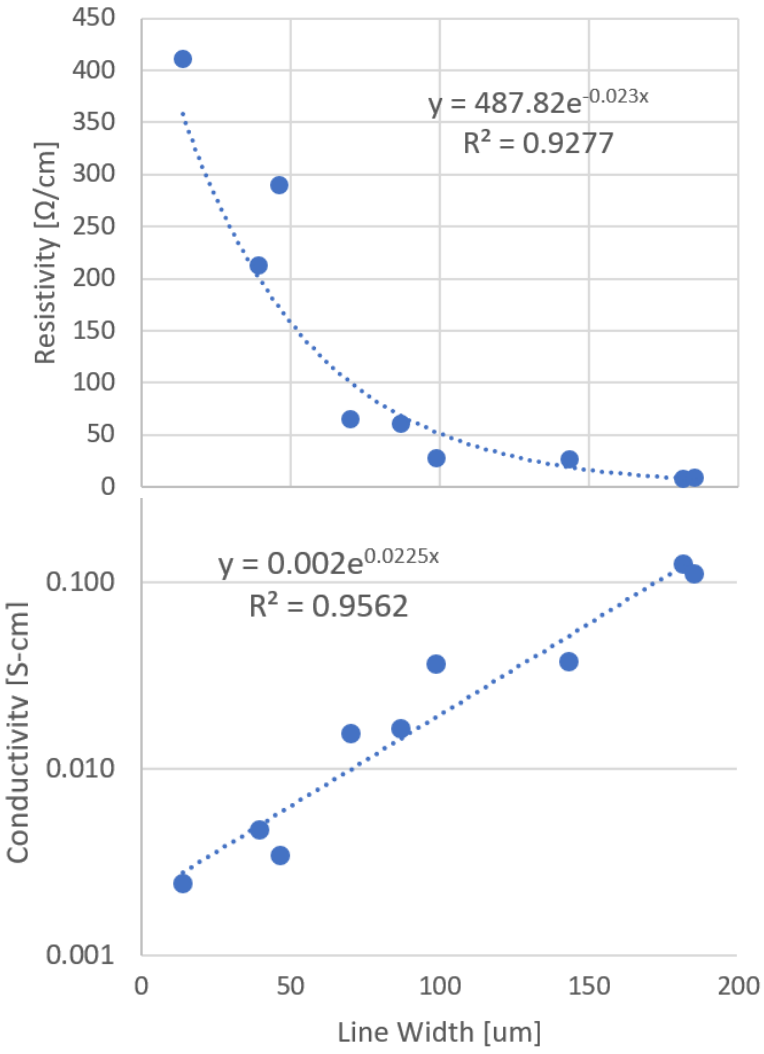


Figure 7.12: Electrical properties of printed lines. Resistivity (top) and conductivity (bottom) have an exponential relationship to line width.

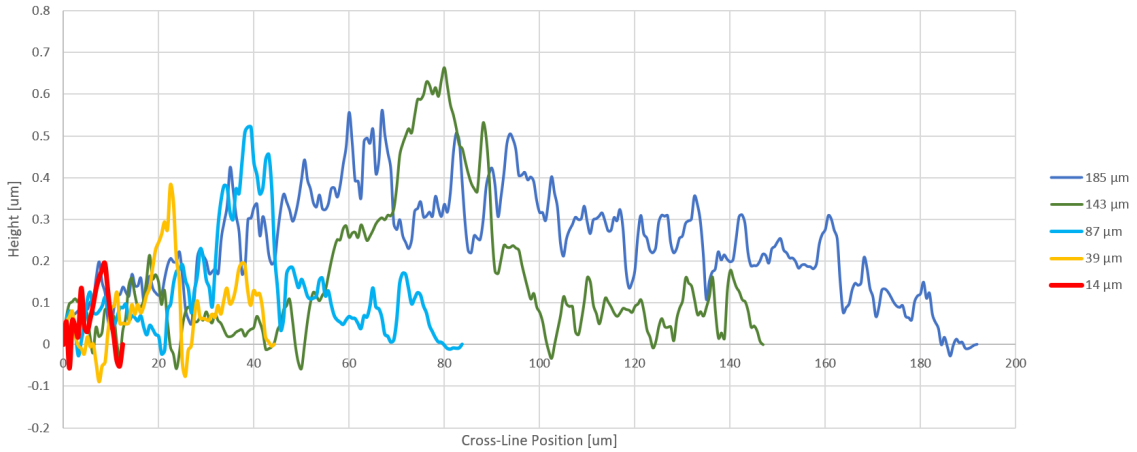


Figure 7.13: Cross-sectional height of five EHD-printed lines of varying width, measured by optical profilometry.

As expected, the effectiveness of printed line charge transport increased as line width increased, at least when lines were limited to a uniform number of print passes. Exponential scaling is something new in the context of diffusion networks - most past studies have dealt with fluid or diffusive flow and employed polynomial models as a result. These trends suggest that there is a statistical mechanism underlying the relationship between width and conductivity. The most likely culprit would be the proportion of percolating pathways that remain continuous through the sintered particle network of a metal nanoparticle-based wire.

Table 7.7: Heights and Aspect Ratios of EHD-Printed Metal Lines

width [µm]	max height [µm]	aspect ratio [x:1,000]
14	0.194184785	13.87034176
39	0.381980852	9.79438082
87	0.521256684	5.991456141
143	0.664384091	4.646042593
185	0.560125516	3.027705491

The print height profiles shown in Figure 7.13 reveal the distribution of material laterally across a selection of printed wires. While overall line height increased with width, it did so with decreasing efficiency. In fact, the measurements summarized in Table 7.7 reveal a reduction in height:width aspect ratio as lines became wider. All metal lines but the largest exhibited profiles that dropped to substrate level at least once, suggesting gaps in the connectivity of the network of nanoparticles that make up each line. These observations lend some credence to the theory that percolation effects were a dominant factor in determining the width-scaling properties of the EHD-printed silver nanoparticle lines.

The effective sheet resistance of the printed metal provides another way to quantify the scaling effect. Evaporated metal lines exhibit a more or less constant sheet resistance independently of line width - not so for EHD printed lines. Sheet resistance for the lines under 50 μm in width was about 0.9 $\Omega/\square$ on average, and for lines over 150 μm, it was 0.16 $\Omega/\square$ on average. Virtuous scaling in an electrode was the principle requirement for a ramified transport network to be optimal over a tapered linear grid, and these results verified the virtuous scaling assumption in EHD-printed metal lines. The resistivity data also provided the first quantitative model for the actual material properties of ML-guided printed lines with this particular set of substrate, ink, and hardware. That model, in turn, could be used to guide the future design of electrode grids on this printer.

Chapter 8

DISCUSSIONS

ML-driven prints of ramified solar cell grids exist in the world now - with dimensions produced to specification and lines that conduct electricity, this could be a natural stopping point. Much work remains undone from each of the component projects in this dissertation, however: a method for laying out tapered isotropic grids in discrete geometry would bring those patterns to performance parity with the tapered H-bar patterns of the 1970s. Better optimization routines and loss function management around wire width limits could improve the performance and stability of the GridGraph optimizer. And the promising MoE-FtL few-shot learning loop has yet to be demonstrated in a physical system.

Before wrapping up this volume, it may be worthwhile to explore some of those missed opportunities, unexplored side streets, and future opportunities in some detail. Then, we will conclude by wrapping the grid printing process back on itself, ouroboros-like. Having charted the flow of information from theory through design and to the printing process itself, design and manufacturing can now collaborate to optimize a printed electrode based on the characterization of this particular metal nanoparticle materials system.

8.1 Extensions for Grid Optimization

The cellular grid optimization framework is aimed fairly strictly at one application in this work, but there are ample opportunities to extend and augment this approach to system optimization. The graph structures that underly the dynamic grid are adaptable and relatively universal. However, a couple of assumptions need to be baked out of the current model before it can be truly transportable. An obvious application of the model would be towards electrochemical electrode structures, where the potential for improved performance may be orders of magnitude larger than in solar cells. However, the model would need to extend into three dimensions and be made multi-phase.

As opposed to the solar cell case, where resources spring up out of the substrate unbidden, batteries and fuel cells extract power from the interface between two or more diffusing entities. At a minimum, the anode, electrolyte, and cathode in such systems each bring diffusion tradeoffs to the process. Some use cases may have even more demands, as in anode-air batteries where the cathode is responsible for current transport but also facilitates 2-way gas species transport through its porous space. In the grid model, the positions of the various diffusion network species would need to fight against each other for limited volume, and power-generating interactions would occur along the edges that connect elements of different species. This advancement would also allow the GridGraph to optimize layout for interdigitated back-contact solar cell electrodes.

In addition to expanding single simulations into higher dimensions and multiple networks, there could also be a benefit to linking multiple grid networks together across multiple scales. For instance, a holistic model for a solar cell might co-optimize the front electrode with a back-side bus layout. The front simulation optimizes the placement of grid lines for a multi-via layout, where each via is a sink. The back-side simulation would optimize the placement of through-vias based on the power inputs from the front-side graph and its own optimization of material and operating voltage. Because GridGraph uses explicit gradient passing, these multi-tiered simulations could pass voltages and cost function values between each other.

Finally, the grid optimization procedure could be much improved. This aspect of the project was deemphasized for the solar cell case because it was unclear how much additional performance remained to be squeezed from increased optimizer performance. However, in a more complicated multi-phase or higher-dimensional problem, the room for improvement is likely much greater. The greedy optimizer used here was susceptible to getting stuck in local minima. Some basic stochastic methods from simulated annealing could make an immediate improvement on solutions.

To push the performance of optimization performance, the grid problem could be packaged as a reinforcement learning problem. The task is difficult on the cellular level, but deep learning methods might be able to identify global trends, such as watershed boundaries or main trunk placement, that would bias the greedy solver towards better minima. This

approach would be reminiscent of deep learning solutions to other NP-Hard problems that live on graphs, such as circuit optimization and protein folding. Recent improvements in those arenas have used existing software such as molecular dynamics or electronic design automation suites, supplementing their solvers with additional information that biases answers toward improved results.[144, 78, 25] A similar approach to grid optimization could use the grid simulation architecture, but augment raw gradient signals passed to each cell with ML-derived estimates of optimal routing direction.

8.2 Adapting MoE-FtL for a Continuous Process

The contest-of-experts MoE-FtL model provided a simple and direct way to perform few-shot prediction of a deeply heterogeneous process and could be a valuable approach to transfer learning in wandering processes in general. The EHD dataset provided an arena that is easily divided into episodes that make it easy to test the MoE-FtL, but it is also interesting to consider how the model might be deployed in longer episodes or for a continuous process. This type of modeling would be especially useful for situations where a process is known to drift due to unmeasurable factors.

This adaptation is simple: episodic data would be gathered in continuous-time chunks of the continuous process, and data from each episode would train one expert model. Simultaneously, a periodic rolling assessment of the experts in the ensemble would provide the basis to select a champion for the next period of predictions. The size of the two continuous sampling windows are two new hyperparameters, and both are subject to the double descent problem.

For the training sampling window, there is a tradeoff between gathering more data to support expert model training and admitting more variation from the process into the sampling window. Similarly, for the retraining sampling window, the increased sample size will boost confidence in the expert selection process but will also delay the deployment of a new expert model. The two sampling windows will be sensitive to the complexity and dimensionality of the training space, the rate of process drift, and each other.

Finally, in such a process, there is potential for the MoE-FtL to grow computationally expensive as new expert models are continuously added to the ensemble. A very large pool

of experts also increases the risk of false positive selections - poor models that win the contest due to lucky answers on the evaluation set. A principled method for curating the ensemble that balances model diversity against size could be helpful in those cases.

As an aside, the MoE-FtL approach is so intuitive and straightforward that it beggars belief that we discovered it in this work. Whether as an ensemble method, a Bayesian process over a collection of estimators, or some other forecasting framework, it has almost certainly been demonstrated and named somewhere else. The omission of prior work may simply be due to a failure of background review, but certainly not malice.

8.3 Printer Improvements and Tapered Lines

For all the time spent characterizing and controlling the EHD, the final demonstration of the printer's capabilities was a somewhat limited affair. Critically, it still cannot do the one thing that, as we have known since the 70s, a solar cell grid should do: the current ML-controlled EHD cannot produce a taper. Producing a smooth, controlled, tapered line may be an extremely challenging task - the current patterns approximate it crudely by stepping through several preset line widths, and a more realistic implementation in the short term may be to increase the number of steps, achieving smoothness through reduced aliasing.

Nonetheless, if taper is the goal, there will need to be some improvements to the EHD hardware, principally its input and output interfaces. Efforts to create online monitoring of the EHD process were abandoned early in the project, which may have been a mistake. The lack of in-situ feedback completely halted any attempts at practical transfer learning or real-time control. However, findings from this project should heavily motivate the inclusion of in-situ measurements in future projects, even if they are relatively simple.

First, the print/no-print boundary is the most valuable signpost in the EHD process space, so even if a camera could only tell whether ink was flowing, it would already provide valuable calibration data for the EHD process. A laser-based scattering rig could be adapted to provide this function for sub-micron droplet sizes if standoff distances are sufficiently large.

Second, transfer learning with a MoE-FtL model was successful with just a few input points, and on-substrate measurement of printed feature sizes could easily take advantage

of this to perform live transfer learning. The optics for these measurements need not be in line with the process. A vertical camera with 2 μm/pixel resolution mounted anywhere above the SIJ substrate could be used to select champion models at the start of runs or to re-calibrate during brief pauses in the middle of runs.

In practice, I would perform these updates in two steps. First, use an optimum sampling strategy of several waveforms to select a champion model from a MoE-FtL ensemble. Then, using that champion, perform a single test print of target sizes to determine proportional control updates, as was done with the zero-shot models in this work.

Finally, to take full advantage of those inputs, the waveform generation and print control interfaces need to be unified. Any future project would do well to coordinate with a tool vendor (or make your own!) to place control of positioning and waveforms under a single software umbrella. This capability is critical to take full advantage of in-situ monitoring and to coordinate the smooth transitions through process space that will be necessary to print smoothly tapered lines.

8.4 Dielectrophoretic Printing Explains Some Things

The EHD is a strange creature, and each ink presents unique challenges. Much time was spent wrestling with the AGK-104 ink system throughout the EHD dataset creation project, and particular idiosyncrasies of ink behavior became clear during that time. In particular, the printer tended to clog when presented with a waveform or two that failed to drive ink ejection, the process consistently drifted towards needing more voltage to achieve consistent results, and the printer showed a large range of critical response times, sometimes nearly a 100x difference from one print to the next. All of these effects can be explained by a particular model of ink actuation by dielectrophoresis (DEP), which I will outline here. These unvalidated hypotheses are based on experience and observation, but they remain to be confirmed by direct experiments.

According to [120], the ejection of a colloidal nanoparticle ink with an entirely nonpolarizable solvent is driven by dielectrophoretic acceleration of nanoparticles across the ink-air interface. DEP describes the two-step effect where an electric field polarizes noncharged particles, then accelerates them in a region of nonlinear field.[76] Conceptually, field nonlin-

earities create an effect where one side of the polarized particle sees a stronger field than the other, causing particles to move towards the region of steeper gradient. So the direct effect of DEP on a colloidal system is to accelerate particles within a nonpolar solvent. However, a combination of entrainment and gradient-driven flow may also cause the bulk fluid to accelerate along with its constituent nanoparticles.

DEP is sometimes applied to directly manipulate particles in a solvent, for instance, by manipulating proteins or isolating red blood cells from plasma in a microfluidic device.[45, 93] When inducing particle drift, DEP can segregate polarizable particles and create concentration gradients within a solvent colloidal mixture. It can also drive bulk flow of a colloidal mixture if the fluid is free to flow.[43] These two regimes - particle drift and bulk flow - are probably both present in the EHD printer: when printing voltage below V_t, the critical voltage to induce ink jetting, electrical fields applied to the ink column will act on metal nanoparticles creating drift but no fluid flow. (We know that flow is absent because no ink is deposited!)

At least three sources of nonlinear field are present in an EHD nozzle. [120] noted that the fluid-air interface at the tip nozzle will tend to accelerate particles towards the substrate. There is also a field concentration effect at the print nozzle tip, the same field effect that causes traditional EHD ink jetting in polarizable fluids, which will become more acute as the meniscus tip extends, reducing the effective curvature of the nozzle tip. Another tip concentration effect is caused by the metal electrode that runs down the inside of the glass SIJ nozzle, terminating a couple of millimeters short of the printing aperture. DEP forces will drive particles toward these concentrators and the nozzle tip under subcritical voltage, likely causing the rapid clogging observed during experiments.

It is interesting to note that clogging could still occur when $V > V_t$. When printing just above the critical voltage, if the rate of fluid flow through the nozzle were less than the rate of particle agglomeration at the nozzle tip, then clogs would still tend to form. This effect may be at play in the smallest 16 μm line in the test pattern shown in Figure 7.8, which tapered off and eventually clogged towards the end of its run. This imbalance in particle and bulk flow rate would create an effective minimum sustainable flow rate, corresponding to the print conditions where bulk fluid flow could "catch up" to the rate of DEP-driven

particle drift. This minimum flow rate, combined with a static tip speed of 500 $\mu m/s$, could at least partially explain why no line widths below 20 μm were observed in any of the sine wave line experiments.

A second effect of DEP-driven printing may be related to the ink column and the effects of bubbles. A curious case of two experimental runs shows up in the dataset: the largest critical width threshold print occurred on April 1, with a threshold pulse width of 96 ms and a routine critical voltage of 105.3V. (The dataset it produced was highly inconsistent and subsequently was excluded from ML model training.) However, the very next set of experiments was back to normal: 1 ms threshold pulse width and a slightly higher 112.5V threshold voltage.

These two experiments, which were performed back to back, used identical equipment: the same substrate, ink, and standard-size SIJ nozzle. The only difference was that between the first and second print, the nozzle height was increased from about 20 to about 30 μm above the surface, and an air bubble was cleared from the nozzle. (The air bubble was trapped in the ink column during print one, then I manually "printed past" the bubble before starting print two.)

So the presence of air in the ink column produced a 100x increase in the response time of the EHD printer but did not affect the critical voltage. This trend makes no sense in a "pull" model of EHD printing, where the main action of the applied voltage is to pull ink directly from the nozzle tip by acting across the fluid-air interface. Under these conditions, a bubble in the ink column would instead increase the system's responsiveness by reducing the mass of ink needing to respond to each voltage pulse.

However, the observation makes perfect sense if the EHD operates according to a "push" model. If the main action of an applied voltage is to pull the entire ink column towards the nozzle tip, then a bubble close to the tip would add a compressible element to the system. Before the ink could jet, pressure would need to build in the trapped air by physically displacing the entire ink column above it down towards the nozzle tip. This would create a time delay between voltage onset and ink flow. When the air bubble was ejected from the ink column, the delay disappeared.

One mechanism that could explain the effect of DEP forces on a large column of ink

that is distant (say, 1mm+) from the EHD nozzle tip could be gradient-driven diffusion. First, the applied voltage pulls particles towards the field concentrators close to the nozzle tip, creating a concentration gradient up through the ink column. Then, everywhere that concentration gradient exists, an osmotic restoring force will attempt to drive fluid down towards the tip, creating back-pressure at the meniscus that scales relative to the height of the whole ink column.

This mechanism also explains the in-process drift that often requires higher voltage as runs proceed. As the ink column depletes over the course of a run, the osmotic back pressure will correspondingly decrease. Tellingly, this effect would be driven by print volume, not print time.

If this speculation is true, it suggests that one of the principal sources of variation and heterogeneity in the EHD is the ink-loading step. Ink is introduced into the print nozzle by hand with a micropipette. The exact amount of ink that gets loaded into each nozzle is random, as is the placement of any air bubbles in the ink column. In fact, given the extreme effect of a single air pocket on printing characteristics observed in the April 1 prints, it may not be a stretch to say that ink loading variance could be the primary driver of print-to-print variability in the EHD printer. To offset these effects, it would be helpful to introduce engineering controls that limit the introduction of air bubbles into the ink column and control the effective height of the ink column. Alternatively, in addition to in-situ measurements of jetting and feature size, a measure of the height of the ink column in the EHD nozzle could allow recipes to compensate for the ink depletion effect in a controlled manner.

Either way, follow-up modeling and experimentation to verify and quantify the effects of bulk-fluid DEP-driven printing with dielectric colloids could significantly enhance understanding and control of EHD processes, with or without all the ML bells and whistles.

8.5 Grid and EHD Applications to Biosensors

In the way of PhDs, I happened to complete a whole different project on the way to the solar cell grid. In that work, we conceptualized, optimized, and manufactured the first proof of concept for a multiplex, peptide-sensitized biosensor that may be able to diagnose

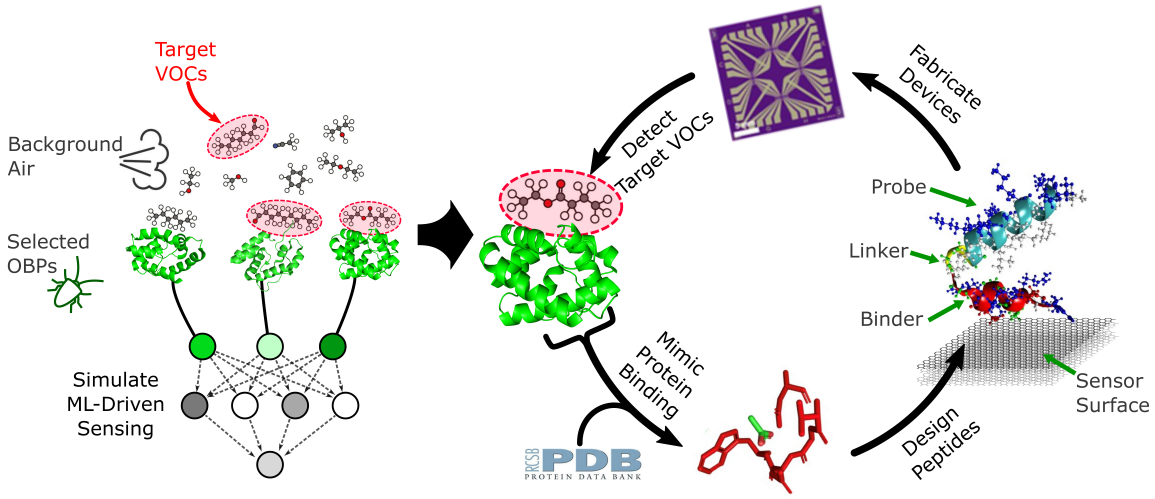


Figure 8.1: Overview of the design and manufacture of a peptide-sensitized eNose sensor for disease detection in human breath. On the left, machine learning models are trained to identify the volatile chemical markers of disease using simulated odorant proteins as sensing elements. On the right, protein binding slot amino acids are transplanted to the surface of a carbon nanotube transistor, enabling the electronic transduction of a scent signal.

COVID-19 disease based on the smell of human patient breath.[95]

Exhaled human breath contains a rich mixture of volatile organic compounds (VOCs) whose concentration can vary in response to disease or other stressors. Using simulated odorant-binding proteins and machine learning methods, we designed a multiplex of short VOC- and carbon-binding peptide probes that detect the characteristic "VOC fingerprint". Specifically, we targeted VOCs associated with COVID-19 in a compact, molecular sensor array that directly transduced vapor composition into multi-channel electrical signals. Rapidly synthesizable, chimeric VOC- and solid-binding peptides were derived from selected proteins using multi-sequence alignment with protein database structures. Selective peptide binding to targeted VOCs and sensor surfaces was validated using surface plasmon resonance spectroscopy and quartz crystal microbalance. VOC sensing was demonstrated by peptide-sensitized, exposed-channel carbon nanotube transistors. Figure 8.1 illustrates the

process.

Two major techniques demonstrated in the current work could potentially improve and accelerate the design and manufacture of peptide-sensitized biosensors. The first is in the design of interdigitated electrodes to maximize surface interface area. A simple version of this was demonstrated by [82], where a compact fractal design improved sensitivity by imposing uniform field conditions across the sensing interface.

The second point of application would be in the deposition of inks containing diverse peptide species onto closely spaced sensor areas. Each carbon nanotube transistor needs to receive a different peptide sensitizer in the final array, and the size of peptide-laden droplets will dictate the minimum physical separation between neighboring transistors. The EHD's ability to deposit micron-scale droplets precisely on a surface may be key to miniaturizing the multiplex eNose device. Additionally, the concentration and conformation of peptides and proteins in solution are manipulable using dielectrophoretic effects.[93] Therefore, the conformation and surface adhesion of peptides in an EHD-deposited droplet may be sensitive to the waveforms used to print them, necessitating the type of multi-objective optimization of the EHD process that was demonstrated in this work.

8.6 Closing the Printed Grid Loop

In the beginning, we began with a simple conception of a solar cell grid: just a straight line of metal running through a homogeneous plane of photovoltaic material and an integral form of Ohm's law describing the sources of power loss in the system. Next, we reviewed a century of research that charted out optimal designs in this space, from numerical descriptions of the balance of line thickness and pitch to the design of line-and-bus H-bar electrodes. We noted an omission in that literature regarding isotropic grid patterns. Finally, we discussed optimal flux density scaling laws noted first by Murray[92] and then in the continuous case by Scharlack[119]. Networks that use metal lines tapered according to those scaling rules were as optimal as one could get when designing solar cell grids.

However, there was an unsolved meta-problem. For transport networks with virtuous scaling - where thicker lines became more efficient transporters - the best layout for a network of tapered lines was a ramified network like those found in biological systems. These

networks allowed current to converge to a few large trunks, expediting efficient transport over longer distances, and selecting optimal networks was an open problem. The numerical description of solar cell grid power loss was packaged into a graph-structured cellular simulation. A greedy optimization loop in that playground was able to find improved grid designs relative to a tapered H-bar baseline.

Containing many line widths that spanned over an order of magnitude, ramified electrode patterns seemed impractical to produce at first glance. A solution was proposed to print these patterns with an electrohydrodynamic inkjet (EHD) - a tool theoretically capable of producing controlled conductive line traces with many materials and at scales down to the micron. To tame the variable EHD process and achieve control over the line widths it deposited, a machine learning (ML) approach was proposed. After creating a multi-task dataset of compiled EHD test runs, supervised ML models were trained and deployed in a zero-shot model-based control algorithm. That algorithm, using single-shot proportional control updates, was able to create recipes capable of reproducing grid patterns that demanded many wire widths. The results are shown in Figure 7.10.

However, these patterns were still not quite optimal. The precise manner in which the conductivities of printed lines scale with their thickness governs optimal line width at the cellular level and decides the optimal routing of current through a ramified network. All grid designs to date used a placeholder function for these values, typically taking a reasonable estimate for bulk metal conductivity and scaling line conductivity as a polynomial function of width.

However, with ML-guided printing online and conductivity characterization complete on a series of physical printed lines, the full grid design loop could finally be closed. A numerical model that described the scaling behavior of EHD-printed traces would plug into the greedy grid optimizer, which would optimize grid patterns for a particular set of hardware and materials.

The resistivity trend shown in Figure 7.12 had an exponential best fit of

$$R = 487.82\, e^{-226w}$$

for w in cm and R in Ω/cm. An exponential trend would fail to capture the trend for very

small widths - R should go asymptotically to infinity as $w \to 0$ - but this model reflected reality in the range of inputs that the printer could easily produce. From subsection 2.2.1, the optimal wire width w^* minimizes the combined power loss from line drop and shadowing in a wire segment δx long per

$$w^* = argmin_w\big(I^2R + V_{op}J_{sol}w\delta x\big)$$

where $R = \frac{\rho_{wire}\delta x}{hw}$. Substituting the analytical equation from the experimental fit yields

$$w^* = argmin_w\big(I^2 487.82\, e^{-226w}\delta x + V_{op}J_{sol}w\delta x\big)$$

$$0 = -110,247\ I^2e^{-226w} + V_{op}J_{sol}$$

with real solution

$$w^* = \frac{1}{226}\, log\big[\frac{110,247\ I^2}{V_{op}J_{sol}}\big]$$

These new trends were inserted into the GridGraph optimization loop by replacing old functions for calculating width and resistance at the element level with the closed-form expressions above.

Figure 8.2 shows examples of grid designs from before and after adopting the data-derived model for wire conductivity. The designs that use realistic scaling (top) tend towards fewer thin wires and a higher density of thick bus bars. The updated model brings the estimated solar cell efficiency from 20 to 18 mW. All grids were optimized for a 1.2 cm wide square solar cell with silicon-like properties (20 mA/cm solar flux and 0.8 V operating potential) and a high-resistance conductive sheet layer of 2,000 $\Omega/\square$. The old model assumed metal conductivity of 2e-4 Ω-cm. The designs were limited to lines between 20 and 200 μm for the models using the realistic conductivity model and down to 10 μm for the old conductivity model. Figure 8.3 shows these patterns as manufactured by the ML-guided EHD printer with routing updates to alleviate the clogging problems in earlier print runs.

These designs close the solar cell design loop by co-optimizing the wire widths, grid layouts, and printing process parameters that all conspire to create a current-transporting grid. With each of the tools discussed in this work pre-baked and ready to go, the flow of information through the design steps is rapid. Reducing experimental data to an analytical

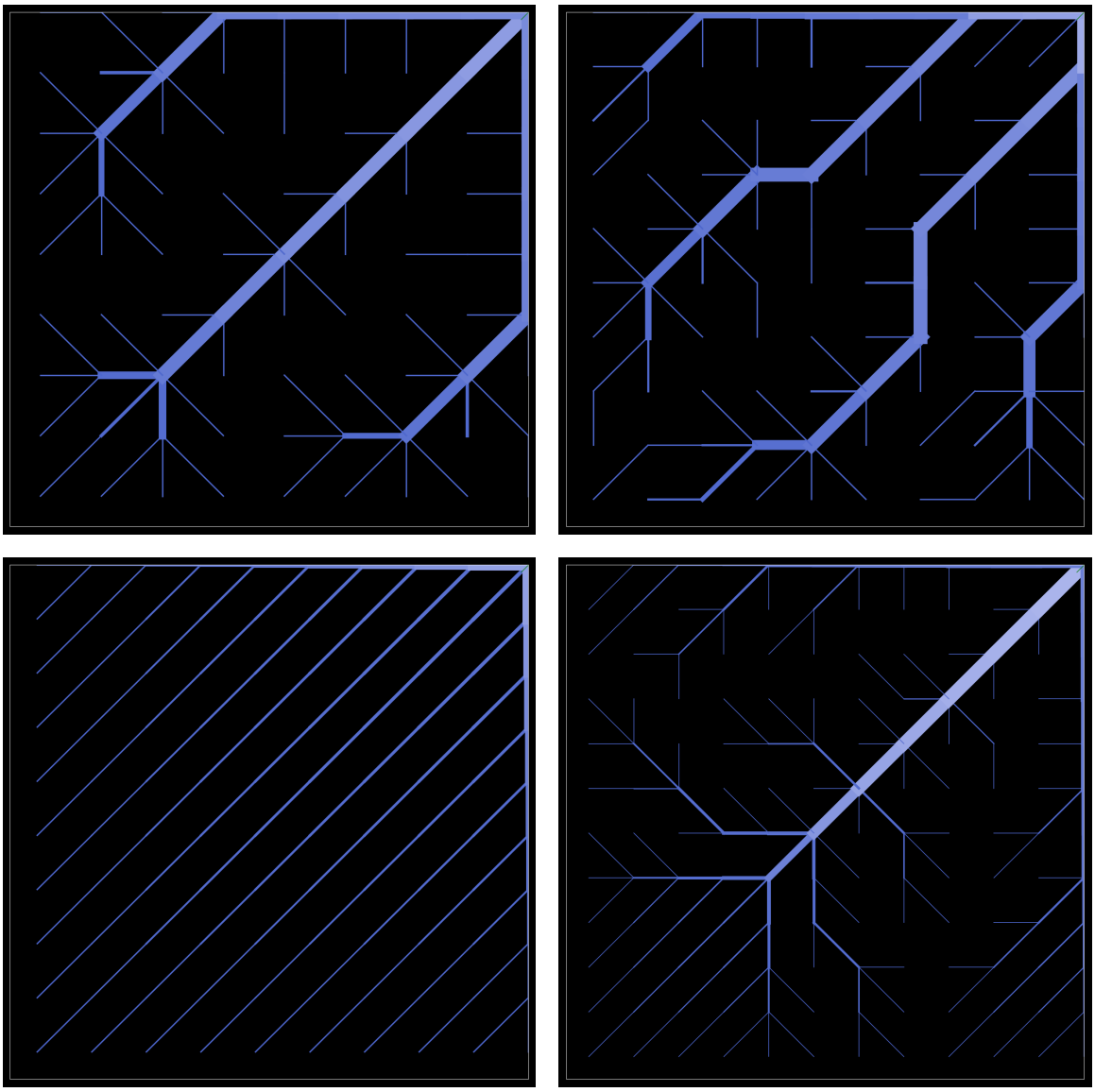


Figure 8.2: Comparison of example 1.2cm solar cell grid designs with an old proxy model for conductivity (bottom) and a new model based on experimental measurements (top). (top left) is a 9x9 pattern yielding 18 mW; (top right) is a 10x10 pattern yielding 18.2 mW; (bottom left) is a 10x10 pattern yielding 19.5 mW; and (bottom right) is a 12x12 pattern yielding 20.1 mW.

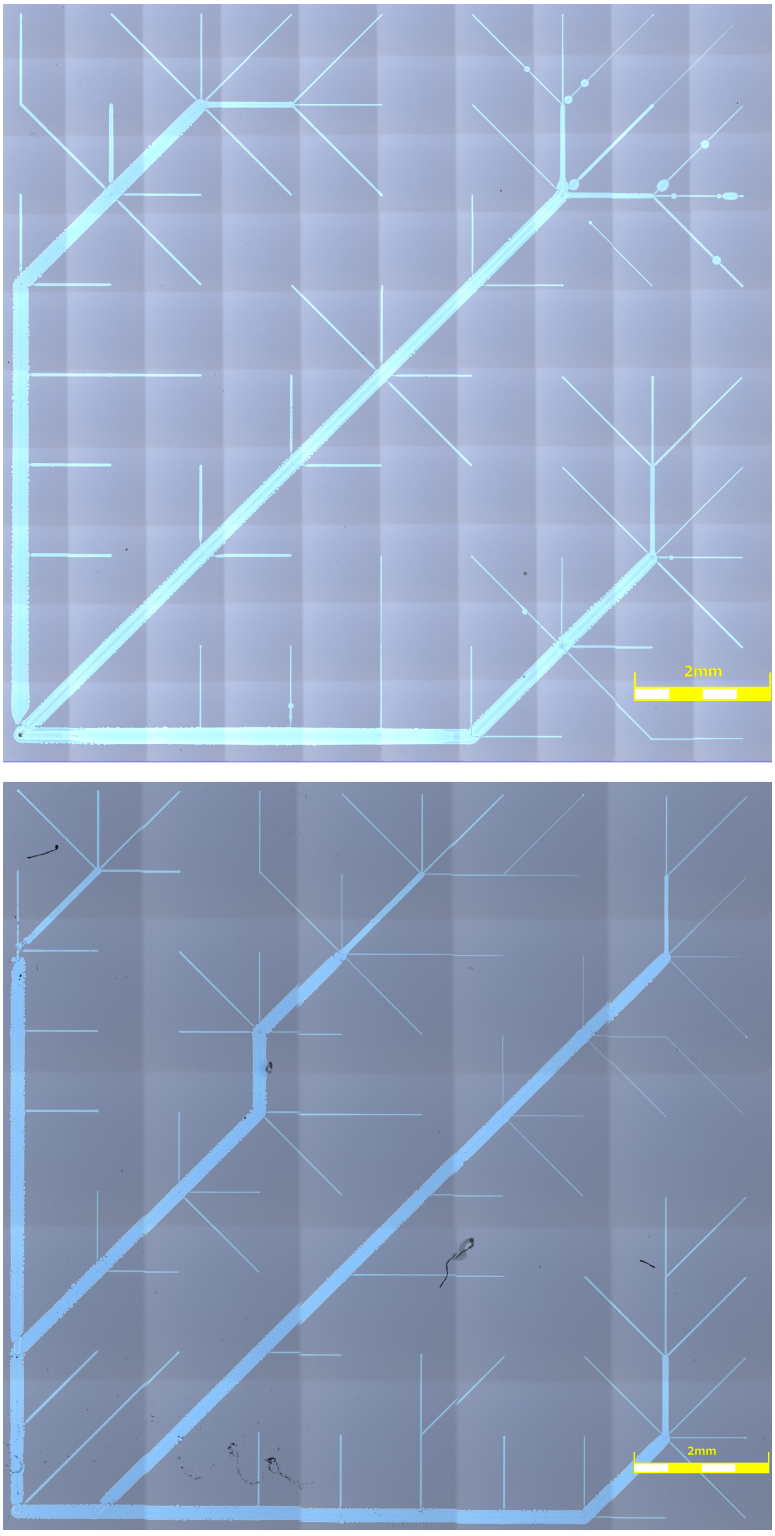


Figure 8.3: Images of printed 1.2 cm solar cell grids designed with a material model based on experimental measurements.

description of optimal wire size is a manual process, costing minutes. Execution of a grid optimization loop with the new analytical expressions takes about 20 seconds. And with a bank of target wire widths from the grid process (for these example patterns, we are looking for lines with widths 20, 48, 84, 110, and 197 μm), the ML-guided process optimization loop will execute faster than the time it takes to type the command.

Of course, there remain limitations and caveats to each of these steps that might prevent them from being put into a production environment right now. The specificity of each solution means that this design loop will only work for the particular confluence of hardware and materials used to develop it. Moving to a new system or application would necessitate, at a minimum, collecting new process data, retraining ML models, and re-characterizing the properties of the new printing output. So in offering this dissertation, my hope is not to provide a complete toolset that could be used for future projects - indeed, the methods presented are far from being turnkey. Instead, this may provide some examples of how complex models such as cellular automata and machine learning can be interfaced with practical problems in mechanical and materials design so that they can operate in the real world.

Hopefully, this project may provide some examples of how complex models of complex systems can work together and an argument for why doing so is worthwhile. Often, the challenge of joining two complex systems arises when trying to identify a workable interface between two models that apparently speak different languages. The interfaces in this project arose organically: from theory and measurement to cellular model, the interface was a set of analytical expressions. From cell model to printer, it is a rich embedded graph. From machine learning to deposited ink, the waveform interface turned out to be essentially a musical tone. And from a running process to the control loop, most days, the interface is a ruler held up to an LCD screen! The brief for this project may have been to optimize the flow of electrons across a 2-dimensional plane, but it has ended up mapping the flow of information through a web of connected models, materials, processes, and datasets.

The benefit of all that complexity is agility. A measurement of a wire's conductivity, itself a product of the connectivity of a percolating network of sintered nanoparticles, is measured this morning. By the afternoon, we can produce a new wire grid that is optimized for that

particular nanoparticle network and printable on this particular EHD printer. A traditional manufacturing approach to this task would lean heavily on fixed processes, locking down each interface in turn to produce a functional, rigid structure. Here, a handful of interfaces in the grid design-and-manufacture process remain free to move against each other, creating an articulating beast. That is not to say that the beast should enjoy free reign of the process. Instead, at every step, rails, confidence gauges, and reduced dimensionality all serve to limit its freedom of movement to a range where data and experience say it is safe to step.

Finally, before closing, I should pay respects to the hidden protagonist of this work, the guy you are so used to seeing in pictures that you forget he is even there: the humble variable for width, w. Width was there from the beginning, mediating the tradeoff between resistance and shadow, and it has been central to every problem considered since. Width was present in virtually every model, expression, and cost function; it was the x- or y-axis in almost every chart, the primary measure of success and failure for every chapter. w even managed, somehow, to become the primary control input to the entire EHD printing system. For such a simple quantity, measured in centimeters and gauged by a ruler, the consideration of w has given rise to a staggering explosion of emergent complexity. That is because, when it comes to transport systems, choosing and achieving w is the crux of the problem.

With batteries, solar cells, and compact energy systems quickly becoming the foundation of the world economy, the task of design optimization for new-wave energy transport systems will only grow in importance. A close look at these systems reveals they are subject to the same tradeoffs that have driven the evolution of the circulatory and respiratory systems in virtually every multicellular organism on Earth. If we want to take advantage of the same morphological solutions to transport that nature has, it will require developing models and tools to optimize and create these transport structures. Furthermore, it will require sustainable technological solutions to control the morphology of energy devices down to the micron scale. In this work, by exploring the use of models, computation, machine learning, and additive processing to tame the deceptively simple w, I hope to have nudged the state of the art towards a more efficient future.

BIBLIOGRAPHY

[1] US Energy Information Administration. Electric Power Monthly - U.S. Energy Information Administration (EIA). https://www.eia.gov/electricity/monthly/index.php.

[2] US Energy Information Administration. Solar power and batteries account for 60% of planned new U.S. electric generation capacity. https://www.eia.gov/todayinenergy/detail.php?id=51518.

[3] U.S. Energy Information Administration. Renewable Energy. *June 2021 Monthly Energy Review*, pages 175–194, June 2021.

[4] Jihoon Ahn, Hyewon Hwang, Sunho Jeong, and Jooho Moon. Metal-Nanowire-Electrode-Based Perovskite Solar Cells: Challenging Issues and New Opportunities. *Advanced Energy Materials*, 7(15):1602751, August 2017.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, Anchorage AK USA, July 2019. ACM.

[6] Dai Akita, Itsuki Kunita, Mark D Fricker, Shigeru Kuroda, Katsuhiko Sato, and Toshiyuki Nakagaki. Experimental models for Murray's law. *Journal of Physics D: Applied Physics*, 50(2):024001, January 2017.

[7] Seongpil An, Min Wook Lee, Na Young Kim, Changmin Lee, Salem S. Al-Deyab, Scott C. James, and Sam S. Yoon. Effect of viscosity, electrical conductivity, and surface tension on direct-current-pulsed drop-on-demand electrohydrodynamic printing frequency. *Applied Physics Letters*, 105(21):214102, November 2014.

[8] Mari Aoki, Kyotaro Nakamura, Tomihisa Tachibana, Isao Sumita, Hideaki Hayashi, Hideaki Asada, and Yoshio Ohshita. 30um fine-line printing for solar cells. pages 2162–2166. IEEE, June 2013.

[9] Raquel Aoki, Frederick Tung, and Gabriel L. Oliveira. Heterogeneous Multi-task Learning with Expert Diversity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2022.

[10] Nigmat Ashurov, Boris L. Oksengendler, Sergey Maksimov, Sayyora Rashiodva, Artur R. Ishteev, Danila S. Saranin, Igor N. Burmistrov, Denis V. Kuznetsov, and

Anvar A. Zakhisov. Current state and perspectives for organo-halide perovskite solar cells. Part 1. Crystal structures and thin film formation, morphology, processing, degradation, stability improvement by carbon nanotubes. A review. *Modern Electronic Materials*, 3(1):1–25, March 2017.

[11] Amit Kumar Ball, Raju Das, Shibendu Shekhar Roy, Dakshina Ranjan Kisku, and Naresh Chandra Murmu. Modeling of EHD inkjet printing performance using soft computing-based approaches. *Soft Computing*, 24(1):571–589, January 2020.

[12] Adrian Bejan. Street network theory of organization in nature. *Journal of Advanced Transportation*, 30(2):85–107, March 1996.

[13] Adrian Bejan. Constructal-theory network of conducting paths for cooling a heat generating volume. *International Journal of Heat and Mass Transfer*, 40(4):799–816, March 1997.

[14] Adrian Bejan and Sylvie Lorente. The constructal law of design and evolution in nature. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1545):1335–1347, May 2010.

[15] Adrian Bejan and Sylvie Lorente. Constructal law of design and evolution: Physics, biology, technology, and society. *Journal of Applied Physics*, 113(15):151301, April 2013.

[16] Richard Bellman. Dynamic Programming. *Science*, 153(3731):34–37, July 1966.

[17] J Bergstra, D Yamins, and D D Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *30th International Conference on Machine Learning*, 2013.

[18] Benjamin Blonder, Cyrille Violle, Lisa Patrick Bentley, and Brian J. Enquist. Venation networks and the origin of the leaf economics spectrum: Venation networks and leaf economics. *Ecology Letters*, 14(2):91–100, February 2011.

[19] Benjamin Blonder, Cyrille Violle, and Brian J. Enquist. Assessing the causes and scales of the leaf economics spectrum using venation networks in *Populus tremuloides*. *Journal of Ecology*, 101(4):981–989, July 2013.

[20] Avrim Blum. 10-806 Foundations of Machine Learning and Data Science.

[21] Stevo Bozinovski. Reminder of the First Paper on Transfer Learning in Neural Networks, 1976. *Informatica*, 44(3), September 2020.

[22] S. Braun, G. Micard, and G. Hahn. Solar Cell Improvement by using a Multi Busbar Design as Front Electrode. *Energy Procedia*, 27:227–233, 2012.

[23] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016.

[24] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse Identification of Nonlinear Dynamics with Control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[25] Ahmet F. Budak, Zixuan Jiang, Keren Zhu, Azalia Mirhoseini, Anna Goldie, and David Z. Pan. Reinforcement Learning for Electronic Design Automation: Case Studies and Perspectives: (Invited Paper). In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 500–505, Taipei, Taiwan, January 2022. IEEE.

[26] A R Burgers. How to design optimal metallization patterns for solar cells. *Progress in Photovoltaics: Research and Applications*, 7:457–461, 1999.

[27] A. R. Burgers and J. A. Eikelboom. Optimizing Metalization Patterns for Yearly Yield. *26th IEEE PVSC*, 1997.

[28] A.R Burgers, J.H Bultman, A.C Tip, and W.C Sinke. Metallisation patterns for interconnection through holes. *Solar Energy Materials and Solar Cells*, 65(1-4):347–353, January 2001.

[29] Dean Buzby. Fine Line Screen Printing of Thick Film Pastes on Silicon Solar Cells. *SGIA Journal*, pages 47–51, 2011.

[30] D.S.H. Chan and J.C.H. Phang. Analytical methods for the extraction of solar-cell single- and double-diode model parameters from I-V characteristics. *IEEE Transactions on Electron Devices*, 34(2):286–293, February 1987.

[31] Ali Cheknane, B. Benyoucef, J.-P. Charles, R. Zerdoum, and M. Trari. Minimization of the effect of the collecting grid in a solar cell based silicon. *Solar Energy Materials and Solar Cells*, 87(1-4):557–565, May 2005.

[32] C.-H. Chen, D. A. Saville, and I. A. Aksay. Scaling laws for pulsed electrohydrodynamic drop formation. *Applied Physics Letters*, 89(12):124103, September 2006.

[33] Lingen Chen, Shengbing Zhou, and Fengrui Sun. Constructal minimization of emitter grid resistance of solar cell with variable cross-section collectors. *APPL PHYS*, 48:7, 2010.

[34] Longwei Cheng, Fugee Tsung, and Andi Wang. A Statistical Transfer Learning Perspective for Modeling Shape Deviations in Additive Manufacturing. *IEEE Robotics and Automation Letters*, 2(4):1988–1993, October 2017.

[35] Longwei Cheng, Kai Wang, and Fugee Tsung. A hybrid transfer learning framework for in-plane freeform shape accuracy control in additive manufacturing. *IISE Transactions*, 53(3):298–312, March 2021.

[36] Hong Kyoon Choi, Jang-Ung Park, O Ok Park, Placid M. Ferreira, John G. Georgiadis, and John A. Rogers. Scaling laws for jet pulsations associated with high-resolution electrohydrodynamic printing. *Applied Physics Letters*, 92(12):123109, March 2008.

[37] Jaeyong Choi, Yong-Jae Kim, Sukhan Lee, Sang Uk Son, Han Seo Ko, Vu Dat Nguyen, and Doyoung Byun. Drop-on-demand printing of conductive ink by electrostatic field induced inkjet head. *Applied Physics Letters*, 93(19):193508, November 2008.

[38] Kyung-Hyun Choi, Khalid Rahman, Nauman Malik, Arshad Khan, Ki-Rin Kwon, Yang-Hoi Doh, and Hyung-Chan Kim. Electrohydrodynamic Inkjet – Micro Pattern Fabrication for Printed Electronics Applications. In Bo Cui, editor, *Recent Advances in Nanofabrication Techniques and Applications*. InTech, Rijeka, Croatia, December 2011.

[39] Michel Cloupeau and Bernard Prunet-Foch. Electrohydrodynamic spraying functioning modes: A critical review. *Journal of Aerosol Science*, 25(6):1021–1036, September 1994.

[40] Mario Conti. Optimal design of front-contact metallization for photovoltaic solar cells. *Solid-State Electronics*, 24(1):79–83, January 1981.

[41] F. Djeffal, T. Bendib, D. Arar, and Z. Dibi. An optimized metal grid design to improve the solar cell performance under solar concentration using multiobjective computation. *Materials Science and Engineering: B*, 178(9):574–579, May 2013.

[42] M. Dressler, T. Studnitzky, and B. Kieback. Additive manufacturing using 3D screen printing. In *2017 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pages 476–478, Verona, Italy, September 2017. IEEE.

[43] Josie L. Duncan, Jeff Schultz, Zeke Barlow, and Rafael V. Davalos. Introducing electric field fabrication: A method of additive manufacturing via liquid dielectrophoresis. *Additive Manufacturing Letters*, 4:100107, February 2023.

[44] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning Factored Representations in a Deep Mixture of Experts. 2013.

[45] Maria E. P. Emmerich, Anne-Sophie Sinnigen, Peter Neubauer, and Mario Birkholz. Dielectrophoretic separation of blood cells. *Biomedical Microdevices*, 24(3):30, September 2022.

[46] Huijun Feng, Lingen Chen, Zhihui Xie, and Fengrui Sun. Constructal optimization of a disc-shaped body with cooling channels for specified power pumping. *International Journal of Low-Carbon Technologies*, 10(3):229–237, September 2015.

[47] Felix Gabriel Fischer, Max Gero Zimmermann, Niklas Praetzsch, and Christian Knaak. Monitoring of the powder bed quality in metal additive manufacturing using deep transfer learning. *Materials & Design*, 222:111029, October 2022.

[48] Rafał Gałek and Piotr Strzelczyk. Velocity profiles of an electrohydrodynamic flow generator: CFD and experiment. *Journal of Electrostatics*, 99:19–30, May 2019.

[49] Jinwei Gao, Krzysztof Kempa, Michael Giersig, Eser Metin Akinoglu, Bing Han, and Ruopeng Li. Physics of transparent conductors. *Advances in Physics*, 65(6):553–617, November 2016.

[50] G. D. Goh, S. L. Sing, and W. Y. Yeong. A review on machine learning in 3D printing: Applications, potential, and challenges. *Artificial Intelligence Review*, 54(1):63–94, January 2021.

[51] Roy G. Gordon. Criteria for Choosing Transparent Conductors. *MRS Bulletin*, 25(8):52–57, August 2000.

[52] Martin A. Green. Solar cell fill factors: General graph and empirical expressions. *Solid-State Electronics*, 24(8):788–789, August 1981.

[53] D. K. Gupta, M. Langelaar, M. Barink, and F. van Keulen. Topology optimization of front metallization patterns for solar cells. *Structural and Multidisciplinary Optimization*, 51(4):941–955, April 2015.

[54] Deepak K. Gupta, Marco Barink, and Matthijs Langelaar. CPV solar cell modeling and metallization optimization. *Solar Energy*, 159:868–881, January 2018.

[55] Deepak K. Gupta, Matthijs Langelaar, Marco Barink, and Fred van Keulen. Optimizing front metallization patterns: Efficiency with aesthetics in free-form solar cells. *Renewable Energy*, 86:1332–1339, February 2016.

[56] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, J. Nathan Kutz, ,Dept. of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, and ,Dept. of Applied Mathematics, University of Washington, Seattle, WA 98195. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[57] John Hampshire and Alex Waibel. Connectionist architectures for multi-speaker phoneme recognition. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[58] Yiwei Han, Chuang Wei, and Jingyan Dong. Droplet formation and settlement of phase-change ink in high resolution electrohydrodynamic (EHD) 3D printing. *Journal of Manufacturing Processes*, 20:485–491, October 2015.

[59] B. R. Helm and G. Davidowitz. Mass and volume growth of an insect tracheal system within a single instar. *Journal of Experimental Biology*, 216(24):4703–4711, December 2013.

[60] Jida Huang, Luis Javier Segura, Tianjiao Wang, Guanglei Zhao, Hongyue Sun, and Chi Zhou. Unsupervised learning for the droplet evolution prediction and process dynamics understanding in inkjet printing. *Additive Manufacturing*, 35:101197, October 2020.

[61] Xufeng Huang, Tingli Xie, Zhuo Wang, Lei Chen, Qi Zhou, and Zhen Hu. A Transfer Learning-Based Multi-Fidelity Point-Cloud Neural Network Approach for Melt Pool Modeling in Additive Manufacturing. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 8(1):011104, March 2022.

[62] Yanhua Huang, Liangkui Jiang, Beiwen Li, Pavithra Premaratne, Shan Jiang, and Hantang Qin. Study effects of particle size in metal nanoink for electrohydrodynamic inkjet printing through analysis of droplet impact behaviors. *Journal of Manufacturing Processes*, page S1526612520302425, April 2020.

[63] Daniel A. Jacobs, Kylie R. Catchpole, Fiona J. Beck, and Thomas P. White. A re-evaluation of transparent conductor requirements for thin-film solar cells. *Journal of Materials Chemistry A*, 4(12):4490–4496, 2016.

[64] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, February 1991.

[65] A Jain. Exact analytical solutions of the parameters of real solar cells using Lambert W-function. *Solar Energy Materials and Solar Cells*, 81(2):269–277, February 2004.

[66] Liangkui Jiang, Li Yu, Pavithra Premaratne, Zhan Zhang, and Hantang Qin. CFD-based numerical modeling to predict the dimensions of printed droplets in electrohydrodynamic inkjet printing. *Journal of Manufacturing Processes*, 66:125–132, June 2021.

[67] Rong Jin and Huan Liu. SWITCH: A Novel Approach to Ensemble Learning for Heterogeneous Data. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg,

Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201, pages 560–562. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[68] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021.

[69] R. Juraschek and F.W. Röllgen. Pulsation phenomena during electrospray ionization. *International Journal of Mass Spectrometry*, 177(1):1–15, August 1998.

[70] Raphael Kay, Anthony Mattacchione, Charlie Katrycz, and Benjamin D. Hatton. Stepwise slime mould growth as a template for urban design. *Scientific Reports*, 12(1):1322, December 2022.

[71] Emmanuel Van Kerschaver and Guy Beaucarne. Back-contact solar cells: A review. *Progress in Photovoltaics: Research and Applications*, 14(2):107–123, March 2006.

[72] Seongju Kim, Minsu Cho, and Sungjune Jung. The design of an inkjet drive waveform using machine learning. *Scientific Reports*, 12(1):4841, December 2022.

[73] Lukas Kranz, Antonio Abate, Thomas Feurer, Fan Fu, Enrico Avancini, Johannes Löckinger, Patrick Reinhard, Shaik M. Zakeeruddin, Michael Grätzel, Stephan Buecheler, and Ayodhya N. Tiwari. High-Efficiency Polycrystalline Thin Film Tandem Solar Cells. *The Journal of Physical Chemistry Letters*, 6(14):2676–2681, July 2015.

[74] Min Wook Lee, Seongpil An, Na Young Kim, Ju Hyeoung Seo, Joo-Youl Huh, Ho Young Kim, and Sam S. Yoon. Effects of pulsing frequency on characteristics of electrohydrodynamic inkjet using micro-Al and nano-Ag particles. *Experimental Thermal and Fluid Science*, 46:103–110, April 2013.

[75] Sen Lin, Xiaopeng Bai, Haiyang Wang, Haolun Wang, Jianan Song, Kai Huang, Chang Wang, Ning Wang, Bo Li, Ming Lei, and Hui Wu. Roll-to-Roll Production of Transparent Silver-Nanofiber-Network Electrodes for Flexible Electrochromic Smart Windows. *Advanced Materials*, 29(41):1703238, November 2017.

[76] Linbo Liu, Ke Chen, Nan Xiang, and Zhonghua Ni. Dielectrophoretic manipulation of nanomaterials: A review. *ELECTROPHORESIS*, 40(6):873–889, March 2019.

[77] Tianjun Liu, Weidong Tang, Sally Luong, and Oliver Fenwick. High charge carrier mobility in solution processed one-dimensional lead halide perovskite single crystals and their application as photodetectors. *Nanoscale*, 12(17):9688–9695, 2020.

[78] Daniela Sanchez Lopera, Lorenzo Servadei, Gamze Naz Kiprit, Souvik Hazra, Robert Wille, and Wolfgang Ecker. A Survey of Graph Neural Networks for Electronic Design Automation. In *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, Raleigh, NC, USA, August 2021. IEEE.

[79] Eduardo Lorenzo Pigueiras, Gerardo L. Araújo, and Eduardo Lorenzo, editors. *Solar Electricity: Engineering of Photovoltaic Systems*. Progensa, Sevilla, 1. engl. ed edition, 1994.

[80] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1930–1939, London United Kingdom, July 2018. ACM.

[81] Alexandre Malley-Ernewein and Sylvie Lorente. Constructal design of thermochemical energy storage. *International Journal of Heat and Mass Transfer*, 130:1299–1306, March 2019.

[82] Yousof Mashraei, Shilpa Sivashankar, Ulrich Buttner, and Khaled Nabil Salama. Integration of Fractal Biosensor in a Digital Microfluidic Platform. *IEEE Sensors Journal*, 16(24):8775–8783, December 2016.

[83] Katherine A. McCulloh, John S. Sperry, and Frederick R. Adler. Water transport in plants obeys Murray's law. *Nature*, 421(6926):939–942, February 2003.

[84] Brendan McMahan and Stephen Joe Jonany. Online Convex Optimization Example And Follow-The-Leader.

[85] Daniel L. Meier, Vinodh Chandrasekaran, Atul Gupta, Vijay Yelundur, and Ajeet Rohatgi. Silver Contact Grid: Inferred Contact Resistivity and Cost Minimization in 19% Silicon Solar Cells. *IEEE Journal of Photovoltaics*, 3(1):199–205, January 2013.

[86] S Mishra, K L Barton, A G Alleyne, P M Ferreira, and J A Rogers. High-speed and drop-on-demand printing with a pulsed electrohydrodynamic jet. *Journal of Micromechanics and Microengineering*, 20(9):095026, September 2010.

[87] Baptiste Moreau and Benjamin Mauroy. Murray's law revisited: Quemada's fluid model and fractal trees. *Journal of Rheology*, 59(6):1419–1430, November 2015.

[88] A M Morega, J C Ordonez, P A Negoias, M Morega, and R Hovsapian. Optimal Electrical Design of Spherical Photovoltaic Cells. *COMSOL Conference*, page 8, 2006.

[89] Carlo Motta, Fedwa El-Mellouhi, and Stefano Sanvito. Charge carrier mobility in hybrid halide perovskites. *Scientific Reports*, 5(1):12746, October 2015.

[90] Takaaki Murata, Kai Fukami, and Koji Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882:A13, January 2020.

[91] Cecil D Murray. The physiological principle of minimum work I. The vascular system and the cost of blood volume. *Physiology*, 12(3):207–214, 1926.

[92] Cecil D Murray. The physiological principle of minimum work II. Oxygen exchange in capillaries. *Physiology*, 12(5):299–304, 1926.

[93] Asuka Nakano and Alexandra Ros. Protein dielectrophoresis: Advances, challenges, and applications: Nanoanalysis. *ELECTROPHORESIS*, 34(7):1085–1096, April 2013.

[94] Oliver Nakano-Baker, Clay Boyd, Caitlin Cramer, Lucien Brush, and J. Devin MacKenzie. Isotropic Grids Revisited: A Numerical Study of Solar Cell Electrode Geometries. *IEEE Transactions on Electron Devices*, 69(7):3783–3790, July 2022.

[95] Oliver Nakano-Baker, Hanson Fong, Shalabh Shukla, Richard V. Lee, Le Cai, Dennis Godin, Tatum Hennig, Siddharth Rath, Igor Novosselov, Sami Dogan, Mehmet Sarikaya, and J. Devin MacKenzie. Data-driven design of a multiplexed, peptide-sensitized transistor to detect breath VOC markers of COVID-19. *Biosensors and Bioelectronics*, 229:115237, June 2023.

[96] L. S. Napoli, G. A. Swartz, S. G. Liu, N. Klein, D. Fairbanks, and D. Tamutus. High level concentration of sunlight on silicon solar cells. *RCA Review*, 38:76–108, March 1977.

[97] United Nations. Renewable energy – powering a safer future. https://www.un.org/en/climatechange/raising-ambition/renewable-energy.

[98] Pabitra K. Nayak, Suhas Mahesh, Henry J. Snaith, and David Cahen. Photovoltaic solar cell technologies: Analysing the state of the art. *Nature Reviews Materials*, 4(4):269–285, April 2019.

[99] Tien Thanh Nguyen, Anh Vu Luong, Manh Truong Dang, Alan Wee-Chung Liew, and John McCall. Ensemble Selection based on Classifier Prediction Confidence. *Pattern Recognition*, 100:107104, April 2020.

[100] Jorge A. Ojeda, Sarah Messina, Erik E. Vázquez, and Federico Méndez. Geometry Optimization of Top Metallic Contacts in a Solar Cell Using the Constructal Design Method. *Energies*, 13(13):3349, June 2020.

[101] Jorge Armando Ojeda, Sarah Ruth Messina, Erick Eduardo Vázquez, and Federico Méndez. Constructal design of top metallic contacts on a disc-shaped solar cell. *Journal of Applied Research and Technology*, 19(5):492–507, October 2021.

[102] OpenAI. GPT-4 Technical Report. 2023.

[103] Page R Painter, Patrik Edén, and Hans-Uno Bengtsson. Pulsatile blood flow, shear force, energy dissipation and Murray's Law. *Theoretical Biology and Medical Modelling*, 3(1):31, December 2006.

[104] T. Panda, S. Sadhukhan, S. Acharyya, P Banerjee, A. Nandi, S. Bose, N. Mondal, G. Das, S. Maity, P. Chaudhuri, and H. Saha. Impact of multi-busbar front grid patterns on the performance of industrial type c-Si solar cell. *Solar Energy*, 236:790–801, April 2022.

[105] Vigneashwara Pandiyan, Rita Drissi-Daoudi, Sergey Shevchik, Giulio Masinelli, Tri Le-Quang, Roland Logé, and Kilian Wasmer. Deep transfer learning of additive manufacturing mechanisms across materials in metal-based laser powder bed fusion process. *Journal of Materials Processing Technology*, 303:117531, May 2022.

[106] Jang-Ung Park, Matt Hardy, Seong Jun Kang, Kira Barton, Kurt Adair, Deep kishore Mukhopadhyay, Chang Young Lee, Michael S. Strano, Andrew G. Alleyne, John G. Georgiadis, Placid M. Ferreira, and John A. Rogers. High-resolution electrohydrodynamic jet printing. *Nature Materials*, 6(10):782–789, October 2007.

[107] Fernando Patino-Ramirez, Chloé Arson, and Audrey Dussutour. Substrate and cell fusion influence on slime mold network dynamics. *Scientific Reports*, 11(1):1498, December 2021.

[108] Arkadeep Paul and Shibendu Shekhar Roy. Numerical simulation to predict printed width in EHD inkjet 3D printing process. *Materials Today: Proceedings*, 62:373–379, 2022.

[109] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. Scikit-learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON*, page 6.

[110] Sonia Pérez-Fernández, Pablo Martínez-Camblor, Peter Filzmoser, and Norberto Corral. Visualizing the decision rules behind the ROC curves: Understanding the classification process. *AStA Advances in Statistical Analysis*, 105(1):135–161, March 2021.

[111] Thanh Huy Phung, Seora Kim, and Kye-Si Kwon. A high speed electrohydrodynamic (EHD) jet printing method for line printing. *Journal of Micromechanics and Microengineering*, 27(9):095003, October 2017.

[112] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic Mode Decomposition with Control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, January 2016.

[113] Lei Qian, Hongbo Lan, and Guangming Zhang. A theoretical model for predicting the feature size printed by electrohydrodynamic jet printing. *Applied Physics Letters*, 112(20):203505, May 2018.

[114] Jian Qin, Fu Hu, Ying Liu, Paul Witherell, Charlie C.L. Wang, David W. Rosen, Timothy W. Simpson, Yan Lu, and Qian Tang. Research and application of machine learning for additive manufacturing. *Additive Manufacturing*, 52:102691, April 2022.

[115] Anne-Gaëlle Rolland-Lagan, Mira Amin, and Malgosia Pakulska. Quantifying leaf venation patterns: Two-dimensional maps. *The Plant Journal*, 57(1):195–205, January 2009.

[116] Henrik Ronellenfitsch, Jana Lasser, Douglas C. Daly, and Eleni Katifori. Topological Phenotypes Constitute a New Dimension in the Phenotypic Space of Leaf Venation Networks. *PLOS Computational Biology*, 11(12):e1004680, December 2015.

[117] Liam M. Rooney, William B. Amos, Paul A. Hoskisson, and Gail McConnell. Intracolony channels in E. coli function as a nutrient uptake system. *The ISME Journal*, June 2020.

[118] Lawren Sack and Christine Scoffoni. Leaf venation: Structure, function, development, evolution, ecology and applications in the past, present and future. *New Phytologist*, 198(4):983–1000, June 2013.

[119] Ronald S. Scharlack. The optimal design of solar cell grid lines. *Solar Energy*, 23(3):199–201, 1979.

[120] Niklas C. Schirmer, Stefan Ströhle, Manish K. Tiwari, and Dimos Poulikakos. On the Principles of Printing Sub-micrometer 3D Structures from Dielectric-Liquid-Based Colloids. *Advanced Functional Materials*, 21(2):388–395, January 2011.

[121] SciPy 1.0 Contributors, Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, Ilhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020.

[122] Kallista K. Sears, Mathilde Fievez, Mei Gao, Hasitha C. Weerasinghe, Christopher D. Easton, and Doojin Vak. ITO-Free Flexible Perovskite Solar Cells Based on Roll-to-Roll, Slot-Die Coated Silver Nanowire Electrodes. *Solar RRL*, 1(8):1700059, August 2017.

[123] SEIA. Solar Industry Research Data. https://www.seia.org/solar-industry-research-data.

[124] S.M Senn and D Poulikakos. Laminar mixing, heat transfer and pressure drop in tree-like microchannel nets and their application for thermal management in polymer electrolyte fuel cells. *Journal of Power Sources*, 130(1-2):178–191, May 2004.

[125] H. B. Serreze. Optimizing Solar Cell Performance by Simultaneous Consideration of Grid Pattern Design and Interconnect Configuration. *Conference Record of the IEEE Photovoltaic Specialists Conference*, pages 609–614, January 1978.

[126] Mahmoud M. Shabana, Mohamed B. Saleh, and Moataz M. Soliman. Optimization of grid design for solar cells at different illumination levels. *Solar Cells*, 26(3):177–187, March 1989.

[127] Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2011.

[128] Vipul Sharma, Anastasia Koivikko, Kyriacos Yiannacou, Kimmo Lahtonen, and Veikko Sariola. Flexible biodegradable transparent heaters based on fractal-like leaf skeletons. *npj Flexible Electronics*, 4(1):27, December 2020.

[129] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. 2017.

[130] T. F. Sherman. On connecting large vessels to small. The meaning of Murray's law. *The Journal of General Physiology*, 78(4):431–453, October 1981.

[131] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[132] Rahul Singh, Xiao Zhang, Yongxin Chen, Junxin Zheng, and Hantang Qin. In-situ real-time characterization of micro-filaments for electrohydrodynamic ink-jet printing using machine vision. *Procedia Manufacturing*, 17:45–52, 2018.

[133] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *Proceedings of the 32 nd International Conference on Machine Learning*, (arXiv:1503.03585), November 2015.

[134] David Stephenson, Alexander Patronis, David M. Holland, and Duncan A. Lockerby. Generalizing Murray's law: An optimization principle for fluidic networks of arbitrary shape and scale. *Journal of Applied Physics*, 118(17):174302, November 2015.

[135] S P Sutera and R Skalak. The History of Poiseuille's Law. page 20, 1993.

[136] Richard S Sutton. Reinforcement Learning Architectures. page 6.

[137] Hong Wei Tan, Jia An, Chee Kai Chua, and Tuan Tran. Metallic Nanoparticle Inks for 3D Printing of Electronics. *Advanced Electronic Materials*, 5(5):1800831, May 2019.

[138] G. I. Taylor and M. D. Van Dyke. Electrically driven jets. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 313(1515):453–475, December 1969.

[139] Atsushi Tero, Ryo Kobayashi, and Toshiyuki Nakagaki. A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of Theoretical Biology*, 244(4):553–564, February 2007.

[140] Atsushi Tero, Seiji Takagi, Tetsu Saigusa, Kentaro Ito, Dan P. Bebber, Mark D. Fricker, Kenji Yumiki, Ryo Kobayashi, and Toshiyuki Nakagaki. Rules for Biologically Inspired Adaptive Network Design. *Science*, 327(5964):439–442, January 2010.

[141] Richard J. Trudeau and Richard J. Trudeau. *Introduction to Graph Theory*. Dover Books on Advanced Mathematics. Dover Pub, New York, 1993.

[142] Israr Ul Haq, Tomoharu Iwata, and Yoshinobu Kawahara. Dynamic mode decomposition via convolutional autoencoders for dynamics modeling in videos. *Computer Vision and Image Understanding*, 216:103355, February 2022.

[143] H B M Uylings. Optimization of diameters and bifurcation angles in lung and vascular tree structures. *Journal of Mathematical Biology*, 39:12, 1977.

[144] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. AlphaFold Protein Structure Database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, January 2022.

[145] Jose V. C. Vargas and Adrian Bejan. Thermodynamic optimization of internal structure in a fuel cell. *International Journal of Energy Research*, 28(4):319–339, March 2004.

[146] C. Wang, X.P. Tan, S.B. Tor, and C.S. Lim. Machine learning in additive manufacturing: State-of-the-art and perspectives. *Additive Manufacturing*, 36:101538, December 2020.

[147] Lidong Wang. Heterogeneous Data and Big Data Analytics. *Automatic Control and Information Sciences*, 3(1):8–15, August 2017.

[148] Xiang-Qi Wang, Arun S. Mujumdar, and Christopher Yap. Thermal characteristics of tree-shaped microchannel nets for cooling of a rectangular heat sink. *International Journal of Thermal Sciences*, 45(11):1103–1112, November 2006.

[149] Liu Wen, Li Yueqiang, Chen Jianjun, Chen Yanling, Wang Xiaodong, and Yang Fuhua. Optimization of grid design for solar cells. *Journal of Semiconductors*, 31(1):014006, January 2010.

[150] S. R. Wenham, Martin A. Green, Muriel E. Watt, Richard Corkish, and Alistair Sproul, editors. *Applied Photovoltaics*. Earthscan, London ; New York, 3rd ed edition, 2012.

[151] E.A.B.S.G. Williamson. *Lists, Decisions and Graphs*. S. Gill Williamson.

[152] F. Willing, S. Wiedeman, J. Newton, J. O'Dowd, and K. Jansen. Power optimization for a-Si solar modules. In *IEEE Conference on Photovoltaic Specialists*, pages 1432–1437, Kissimmee, FL, USA, 1990. IEEE.

[153] Johnson Wong. Griddler: Intelligent computer aided design of complex solar cell metallization patterns. In *2013 IEEE 39th Photovoltaic Specialists Conference (PVSC)*, pages 0933–0938, Tampa, FL, USA, June 2013. IEEE.

[154] Dazhong Wu and Changxue Xu. Predictive Modeling of Droplet Formation Processes in Inkjet-Based Bioprinting. *Journal of Manufacturing Science and Engineering*, 140(10):101007, October 2018.

[155] Maxwell Wu, Patrick M Sammons, and Kira Barton. Numerical Modeling of High Resolution Electrohydrodynamic Jet Printing Using OpenFOAM. page 15.

[156] Liying Yang. Classifiers selection for ensemble learning based on accuracy and diversity. *Procedia Engineering*, 15:4266–4270, 2011.

[157] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems. In *2019 American Control Conference (ACC)*, pages 4832–4839, Philadelphia, PA, USA, July 2019. IEEE.

[158] Jong-Su Yu, Inyoung Kim, Jung-Su Kim, Jeongdai Jo, Thue T. Larsen-Olsen, Roar R. Søndergaard, Markus Hösel, Dechan Angmo, Mikkel Jørgensen, and Frederik C. Krebs. Silver front electrode grids for ITO-free all printed polymer solar cells with embedded and raised topographies, prepared by thermal imprint, flexographic and inkjet roll-to-roll processes. *Nanoscale*, 4(19):6032, 2012.

[159] Alan Zhan, Taylor K. Fryett, Shane Colburn, and Arka Majumdar. Inverse design of optical elements based on arrays of dielectric spheres. *Applied Optics*, 57(6):1437, February 2018.

[160] Jiang Zhang. Universal patterns and constructal law in open flow networks. *International Journal of Heat and Technology*, 34(S1):S75–S82, January 2016.

[161] Xiao Zhang, Benjamin Lies, Hao Lyu, and Hantang Qin. In-situ monitoring of electrohydrodynamic inkjet printing via scalar diffraction for printed droplets. *Journal of Manufacturing Systems*, 53:1–10, October 2019.

[162] Xianfeng Zheng, Guofang Shen, Chao Wang, Yu Li, Darren Dunphy, Tawfique Hasan, C. Jeffrey Brinker, and Bao-Lian Su. Bio-inspired Murray materials for mass transfer and activity. *Nature Communications*, 8:14921, April 2017.

[163] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, January 2021.

Appendix A

ALL ISOTROPIC GEOMETRIC GRIDS ARE EQUIVALENT

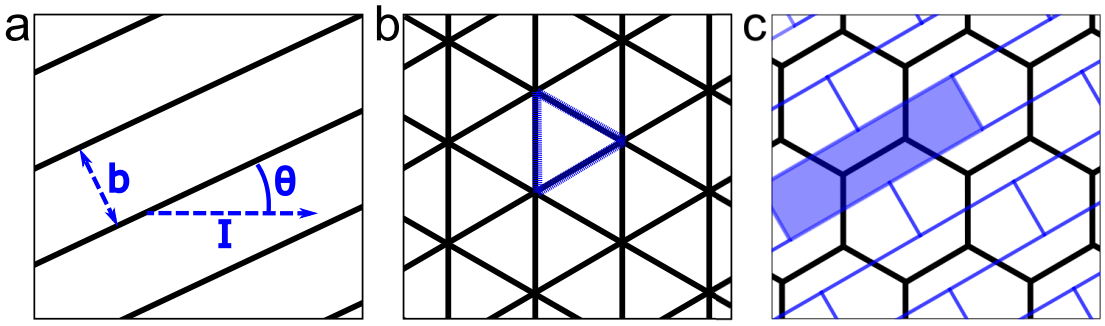


Figure A.1: Geometric conventions in describing the lack of angular dependence on grid conductivity: (a) grid angle, (b) overlay of 3 linear grids in a triangle isotropic grid pattern, and (c) a unit tile for calculation of a linear sub-grid in a hexagonal grid.

It was stated in chapter 2 that every isotropic grid design is equivalent to each other. This fact may not be self evident, but can be illustrated as follows.

To avoid confusion, a planar array of parallel conductive grid lines will be referred to as a "linear grid." Consider the "sheet conductivity" κ (in units squares/Ω or Ω^{-1}) of a linear grid with width w, height h, and material resistivity ρ_{metal}: $\kappa_{sheet} = \frac{1}{\rho_{sheet}} = \frac{wh}{\rho_{metal}\, b} = \frac{1}{\rho_{wire}\, b}$. Here $\rho_{wire} = \frac{\rho_{metal}}{wh}$ is the resistivity of one line in the array, i.e. in units Ω/cm, and b is the grid pitch. These values apply only in the direction parallel to the lines in the array - indeed if current flux has no convergence, a design should simply align its grid with the current flux. With ϕ as the angle between the linear grid and the direction of current flux,

effective grid line pitch and conductivity take on angular dependencies.

$$\rho_{\phi,\,wire} = \frac{\rho_{metal}}{w\,h\,cos\phi} \tag{A.1}$$

$$b_\phi = \frac{b}{cos\phi} \tag{A.2}$$

The angular sheet conductivity for the linear grid becomes

$$\kappa_\phi = cos^2\phi\,\frac{wh}{\rho_{metal}\,b} \tag{A.3}$$

Square and triangle grids are both comprised of multiple overlayed linear grids. Because all sub-grids are continuous and in contact with each other, we model the complete system as several linear grids in parallel with offset angles $\Phi = \{\phi_1,\, \phi_2,\, ...\}$. The resulting conductivity is

$$K_{grid} = \sum_{\phi\in\Phi} cos^2\phi\frac{wh}{\rho_{metal}\,b} \tag{A.4}$$

If the involved grids have the same line properties and pitch, then the right-hand fraction carries out of the summation. With current traveling along direction θ, $\Phi_{squares} = \{\theta,\, \theta + \pi/2\}$ and $\Phi_{triangles} = \{\theta,\, \theta + \pi/3,\, \theta + 2\pi/3\}$, and the summation reduces to a constant value - the lack of dependence on θ indicates that the grids have isotropic conductivity. Specifically,

$$K_{grid}^{square} = \frac{wh}{\rho_{metal}\,b^{square}} \tag{A.5}$$

$$K_{grid}^{triangle} = \frac{3}{2}\frac{wh}{\rho_{metal}\,b^{triangle}} \tag{A.6}$$

and if the grids are sized so that $b^{triangle} = \frac{3}{2}b^{square}$, their conductivities will be equal.

The hexagonal grid solution is less clear because its component grid lines are not continuous. Instead, consider the unit tile shown in Figure A.1(c). The effective conductivity of the cell is 1/3 that of the metal line it contains because of the lengths relationship $l_{wire} = \frac{1}{3}l_{tile}$, and the tile cross section is $\frac{1}{2}$ of the grid pitch b. The sheet conductivity of a tiling of these partial-wire cells takes these scaling factors into account. (Such a tiling would not form a percolating electrical grid by itself.)

$$K_\phi^{hex} = cos^2\phi\frac{l_{wire}}{l_{tile}}\frac{wh}{\rho_{metal}}\frac{2}{b} = \frac{2}{3}cos^2\phi\frac{wh}{\rho_{metal}\,b} \tag{A.7}$$

As with the triangle grid, $\Phi_{hex} = \{\theta, \theta + \pi/3, \theta + 2\pi/3\}$ and $\sum_{\phi \in \Phi_{hex}} cos^2\phi = 3/2$. As a result, the hexagonal grid has an isotropic sheet conductivity equal to that of the square grid, so long as their pitches are equal:

$$K_{grid}^{hex} = \frac{wh}{\rho_{metal}\, b^{hex}} \tag{A.8}$$

This relationship holds only because the current flux is constant in each of the overlapping hexagonal sub-grids and the overall grid is fully connected (it percolates). Any such system with self-similar sub-grids and Φ_{grid} comprised of more than 2 whole number divisions of the unit circle will also be isotropic - it follows that for any such isotropic grid a pitch value exists which renders it identical in conductivity to the isotropic grids discussed here.

Though we do not discuss it here in detail, it is apparent that linear grid angle has no effect on its shadowing behavior, and that the effect of line segment length and pitch scaling applies identically to shadowing. Furthermore, the multiplication trick used to find pitch values that bring conductivity in line between two different isotropic grids will also equilibrate the shadowing coverage of the two grids. Therefore, as far as we can tell, all isotropic grids will be fully equivalent to each other in their conductivity/shadowing trade-offs, regardless of the particular number and angles of linear sub-grids. We believe (but do not demonstrate) that this property extends in expectation to the properties of uniformly random arrays of conductive lines, such as nanowire assemblies.

Appendix B

INTRODUCTION TO CONSTRUCTAL THEORY

Constructalism, formalized by Adrian Bejan in the 90s, aimed to describe and optimize the ramified systems for mass and heat transportation that can be found throughout natrual systems. Bejan observed the formation of such networks naturally in street traffic networks, then formulated a numerical solution for conductive heat transport networks in packaged electronics.[13] In that paper, a 'fourth law' (of thermodynamics) is proposed: "For a finite-size system to persist in time (to live), it must evolve in such a way that it provides easier access to the imposed (global) currents that flow through it". The solution to the heat transport problem, an $a = 1$ scaling problem in the Murray parlance with straight line elements, has much in common with the Murray and Scharlack optimal transport formulation. It identifies an optimal balance point between the sources of loss in the transport system. The constructal law could be seen as extending Murray's formulation from "how" to branch in a network to "when" to branch. (Optimal branching occurs so as to evenly spread the work burden of transport between each assembly level, or branched tier, in the network.)

Constructal theory has since been extended to balance diffusion networks in numerous systems including fuel cells,[145, 124] heat sinks,[148, 46] thermochemical batteries,[81] virtual networks,[160] and natural transport networks.[14, 15] It has been applied to several geometries of solar cell optimization,[33, 100, 101] recovering the same scaling relationships discussed above in section 2.2.

Despite its seemingly universal applicability, constructalism can fall short in solving the "where" of network design. It assumes universality on the part of each element of its design, where real-world problems must deal with heterogeneous reality. (The first four-vertex example in subsection 3.2.6 already breaks the tiered assumptions of a contractual solution.) And the geometric patterns used to form space-filling constructs must be decided by each individual practitioner, limiting the utility of the theory to optimize the specific

layout of any particular pattern.

A key observation of constructal theory, and indeed this thesis up to this point, has been that ramified transport networks appear to be near-optimal structures for a variety of biological tasks. Biological structures can occasionally be directly co-opted to efficiently solve transport engineering problems![128] However, biological systems are self-assembled and their building blocks - usually cells - can typically only "see" their local conditions. The building blocks of transport networks in leafs[115, 116] or biofilms[117] do not have a view of the overall macro-scale problem. Therefore the algorithms by which these natural formations arrive at their topology should be implementable in networks of autonomous cells. For example, computational models of slime mold colonies, which utilize virtuously scaling flow channels to form venous nutrient gathering networks, were able to reproduce maze-solving and space-filling behavior using greedy graphical models.[139, 6]

Appendix C

METHODS AND SOFTWARE FOR SOLAR GRID SEARCH

A Python implementation of GridGraph can be found at `https://github.com/onakanob/GridGraph`. That repository also includes a visualization tool that can be used to explore the grid-finding algorithm with different geometric parameters. Some examples of solved grids using this widget can be seen in Figure C.1. The wires in the visualization have widths that scale with the simulation wire widths and their color shades towards red as total current through the wire increases. The circles each represent one element and sinks are shown as triangles. The area of each element is determined by a Voronoi tessellation of the underlying mesh and the size of the circles scale relative to the element area. The color of the nodes shade from yellow to black, representing the differential term $\partial T/\partial I$ in units [Watts/Amp].

The software is built with three layers of inheritance, visualized in Figure C.2. The base level owns two graphs, a mesh and a dynamic graph, as well as operations for graph traversal and modification. The mesh is an undirected graph that is static throughout an optimization run - it defines the set of allowed connections between all nodes in the model. The dynamic graph is the directed graph that defines flows from the simulation to the sink(s) - dynamic graph edges are always a subset of the mesh and will change from iteration to iteration as optimization passes proceed.

The second level of physics models contains methods for solving current generation and power loss equations at the element level, for defining vertex-level gathering and transport functions in a given graph structure, and for calculating the final power output of a design. This layer is also responsible for defining the update loops that present a reward function at each vertex. Two date there are two modules that fit this interface: the power-debt-passing model discussed above, and a more in-depth 2-diode model of solar cell behavior, discussed in subsection C.1.2.

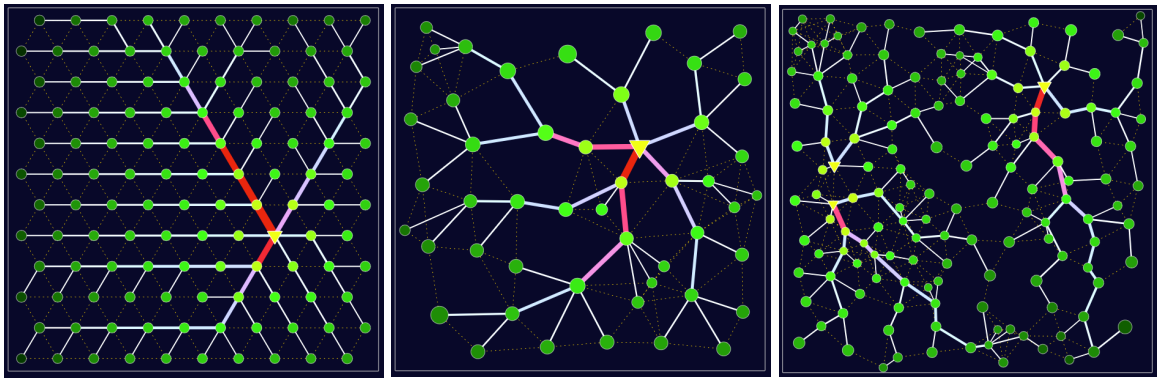


Figure C.1: Three grid solutions for a 1-cm solar cell with the visualizer widget from `github.com/onakanob/GridGraph`. The underlying meshes use [Left] hexagonal and [Middle and Right] randomly placed elements. The [Right] pattern employs three sinks.

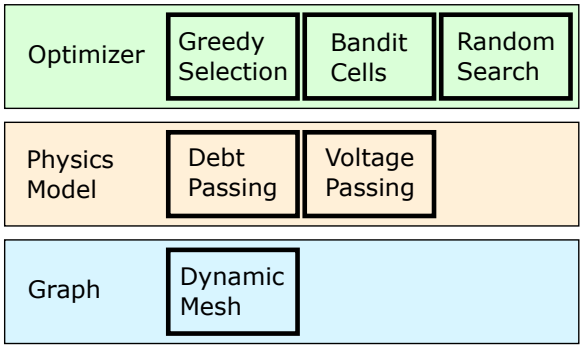


Figure C.2: Schematic of class inheritance in GridGraph's implementation. A modular design approach makes it easy to swap in alternate physics models and optimization strategies.

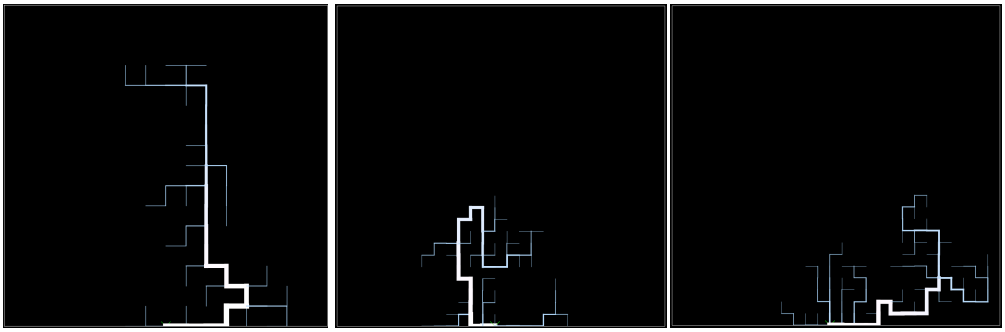


Figure C.3: Grids generated by cell-wise bandit search algorithm.

The third level of abstraction contains methods for optimizing the dynamic graph, using the reward function outputs from the second level to evolve the graph structure over generations. Three optimization algorithms have been implemented to-date: the greedy routing described in algorithm 3, a brute-force search, and a naive reinforcement learning approach with cell-level.

C.1 Alternate approaches to graph optimization

Some alternate modules were explored, in an attempt to rectify weaknesses in the original GridGraph implementation. A bandit solver would attempt to find better solutions to the routing problem - it failed due to the exponential scaling of the problem. Second, a double diode physics model was trialed that would account for fill factor at each cell in the solar cell graph.

C.1.1 Alternate Optimizer: Bandit Cells

The "stickiness" of early routing choices in the greedy model results in an early locking-in of specific choices of main-trunk routing and watershed formation. This hyper-sensitivity to the first few generations of solver steps is likely to lead to sub-optimal outcomes, so strategies that encourage active exploration during the optimization process could improve outcomes. One possible approach to balancing exploration involves the use of reinforcement learning methods at the cellular level.

In a relatively naive implementation, a bandit search algorithm was used to guide wire routing through individual vertices in the electrode graph, as described in by [136]. For each generation of the grid, each vertex made a random selection of target neighbor. After power output was collected and the incremental signal $\frac{\partial T}{\partial I}$ was distributed to individual nodes as described in subsection 3.2.3, a set of action-value weights across neighbors were updated at each vertex. These weights were used to bias the target selections in future generations so that each vertex preferred to route current to neighbors that had given them higher average power returns in the past.

Periodically, a greedy selection was made, with every vertex choosing the neighbor with the max average $\frac{\partial T}{\partial I}$. Examples of three such patterns are shown in Figure C.3 - these patterns were each generated after roughly 1 hour of compute time and between 10- and 100-thousand generations of exploration on the UW Hyak supercomputer cluster. The patterns are significantly better than random selection, but the extent of the connected graphs seem to stall out before the entire space could be explored. They also fail to display the linearity and centered main-trunk locations of the greedy solutions, and the patterns improved relatively little given an extra 2 or 3 hours of run time.

The bandit search algorithm is likely (indeed, guaranteed) to converge given time. But the question is how much time. The search is subject to the same explosive scaling as a brute force search, and at least in this example and subject to these hardware constraints, the practical limitations seem to preclude the method from approaching the performance of the greedy cellular optimization.

A combination of the two methods could prove powerful in the future. It is possible that the outputs from the bandit process could make better initial seeds for the greedy optimization loop than random or cold starts. Ideally, the two methods would offset each others' weaknesses, the bandit method bringing principled exploration to the process while the cellular optimization loop quickly sniffs out locally-optimal solutions to the noisy outputs from the bandit search. For now, this approach is left to a future project.

C.1.2 Alternate Model: 2-Diode Elements

The largest, glaring shortcoming of the debt-passing model of solar cell physics is the omission of voltage modeling. In real photovoltaics, there is a balance between actual solar current and operating voltage that must be navigated to extract the maximum power from a solar cell. As the voltage imposed over a solar cell junction increases, the density of photocurrent from the active material decreases - in the extreme, excessive over-voltage can push current back through the photovoltaic material, causing direct losses of current and power. The behavior of the photovoltaic junction is often represented by a 2-diode model,[30] with:

$$J_{sol} = J_{sc} - J_1 exp[\frac{qV_{op}}{kT}] - J_2 exp[\frac{qV_{op}}{2kT}] - \frac{V_{op}}{R_{shunt}} \tag{C.1}$$

where V_{op} is the operating voltage, q is the elemental electron charge, k is Boltzmann's constant, T is the temperature, and J_{sc} is the short-circuit current. The J_1 term represents a diode model of surface recombination, the J_2 term is a second diode modeling bulk recombination within the PV-junction, and V/R_{shunt} describes current escaping back through the junction through shorts. In practice, operating voltage V_{op} is chosen to maximize cell power $P = J_{sol}V_{op}$, and the difference between this practical power output and an idealized power calculated from short circuit current and open circuit voltage V_{oc} is the fill factor $FF = \frac{V_{op}J_{sol}}{V_{oc}J_{sc}}$.[52, 65]

Solar cell electrodes cause a spread in voltage across the solar cell area, from the external operating voltage at the sink to higher values elsewhere. This means that most of the active material is operating slightly above or below the true max-power voltage, and the fill factor of the overall solar cell is partially a function of the voltage patterns created by the electrode.

These critical effects were ignored in the initial model because they introduce a number of difficulties. Current is no longer conserved, and with the auto-optimizing wire widths of the grid model, there is no analytical solution to the relationship between V_{op} and J_{sol} introduced by Equation C.1.2. The solver needs time and iterations to converge to a solution.

Luckily, in practice, a dynamic grid over 2-diode finite elements is relatively stable! An

implementation of this physics model is available in the code at `github.com/onakanob/GridGraph`, and an example of a grid solution from the 2-diode model can be seen in Figure 3.4, lower right. The solver iterates as before, but now also takes a single step to converge the I/V characteristic of the model. It also simultaneously optimizes the operating voltage at each sink to maximize the fill factor. The system has no guarantees of numerical stability, but it converges in practice. In future work on this project, this 2-diode model will probably be the physics module of choice.

Appendix D

NEXT STEPS FOR THE GREEDY CELLULAR GRID

D.1 Exploring Ramified Electrodes

The patterns that emerge from the GridGraph optimization loops have a certain look about them - there is a familiarity to the emergent length scales and regions of repeated patterns within each solution. While these patterns can be described subjectively, I have yet to offer any measure or quantitative analysis of these patterns. Even though the present approach has little hope of every finding a truly optimal graph for a large-mesh solar cell area, methods to quantify and compare the outputs of the optimization loop could still give insights into optimal designs and how they vary with changes to the material properties and geometric bounds of the problem. Past researchers have taken a graph-analytical approach to make observational studies of veins in leaves[115, 18, 118, 19, 116] and slime mold networks,[140, 107, 70] and the synthetic nature of the GridGraph solar cell patterns could allow for direct comparison to these biological systems, but with larger datasets and systematically varied initial conditions.

The prime candidate for graph analysis of the solar grid is spectral analysis, which could be used to illustrate the characteristic repeated feature sizes across length scales. Some of the metrics employed by [116] such as nesting ratios (we would use branching ratios, perhaps) and "topological length" could likely also be adapted to characterize the solar grids.

D.2 Enhanced Exploration

The incremental utility of making further improvements to the graph optimizer is uncertain at this point. If there is significantly less than 1% additional performance yet to be gained between the patterns being produced now and truly optimal designs, then the effort and computation time needed to find those patterns may not be justified. However, I suspect

that some more exploration of design optimization is still justified. Firstly, because there are likely to be larger potential performance gains in some of the other possible applications of this type of algorithm, particularly the 3-dimension-embedded graphs discussed in section D.3. Secondly, because more optimal designs are also likely to be less varied in terms of feature sizes and distributions, making them more easy to fabricate. And thirdly, because there are likely to be interesting scientific and/or mathematical properties of ramified networks that could be explored analytically using this tool-set, and those underlying properties may be more easily identified in networks that are closer to optimal morphology.

Some ideas for enhancing the exploration capabilities include:

- Simulated annealing to knock the graph out of local optima.
- A switch to slow wire width adjustments, instead of the local selection criteria used in this model. Allowing the graph to take a slower approach to the final grid design could reduce the impact of early routing decisions.
- Use of a more powerful model to solve the reinforcement problem. For instance, a graph convolutional network could learn to weight the edges of a diffusion graph, and the simulation described here could be used to train the network under self play conditions. (This would be a computationally intensive undertaking.)

D.3 Higher Embeddings and Additional Poles

The final set of model enhancements change some underlying assumptions of the model to allow it to tackle an expanded set of optimization tasks, beyond just the front-side solar cell. One small example of extended applications can be seen in section D.4, but other applications require a retooling of the model. The first modification is to raise the elements of the model into three dimensions to allow the optimization of volumetric grids such as those found in batteries, fuel cells, electro-chemical cells, etc. Like solar cells, these devices rely on the transport of charge carriers or chemical species to function, and they are often rate limited by the active surface area available in the device. Optimization of these transport structures could improve power density or process throughput, and with the

additional design freedom that comes with three dimensions, the potential improvements to performance may be larger than the hand full of percents seen in solar cells.

A more challenging improvement is the ability to optimize bi- or multi-polar grids. These grids have two sinks that define two networks within the same space, where some process takes place at the interface between the two electrodes. Such devices will often experience an improvement in performance if that surface length or area is maximized. The simplest application would be to improve the layout of interdigitated electrodes, which find applications in a wide variety of sensors. For instance, introducing fractal-like patterns to some biosensors has shown to increase their sensitivity.[82]

Many multi-polar gathering problems in nature involve two networks with differing transport properties that must suffuse each other - the lungs are one such example, which seek to maximize interfacial area between blood and the atmosphere to facilitate gas exchange. The lungs are a three-pole system, involving flows through arteries, veins, and alveoli. The multi-polar, multi-flow treatment is also relevant to fuel cells and metal-air batteries, which must allow gaseous access to the reactive surface through a cathode, making for a total of four simultaneous transport processes. (Positive and negative charge carriers, atmosphere, and electrolyte transport.) Finally, a bi-polar optimization process would allow for the design of layouts for integrated back-contact solar cells, a fabrication process that removes all shadowing metal from the solar cell front surface by moving both electrodes to an interdigitated layout on the back side.

The mathematical treatment of the reward function in these systems becomes more complex, as reward is no longer produced at every vertex in the grid but arises only from the edges that connect the two (or more) sub-networks. However, the potential improvements to the performance of sensor and energy devices could be large.

D.4 Routing current through vias

GridGraph has, to date, been aimed mostly at solar cell optimization problems, with a special eye towards front contact grids. However, the ubiquitous nature of the gathering problem suggests that a solver well suited to one such problem could probably also find efficient solutions to others. One recent extension of the model involved the optimization of

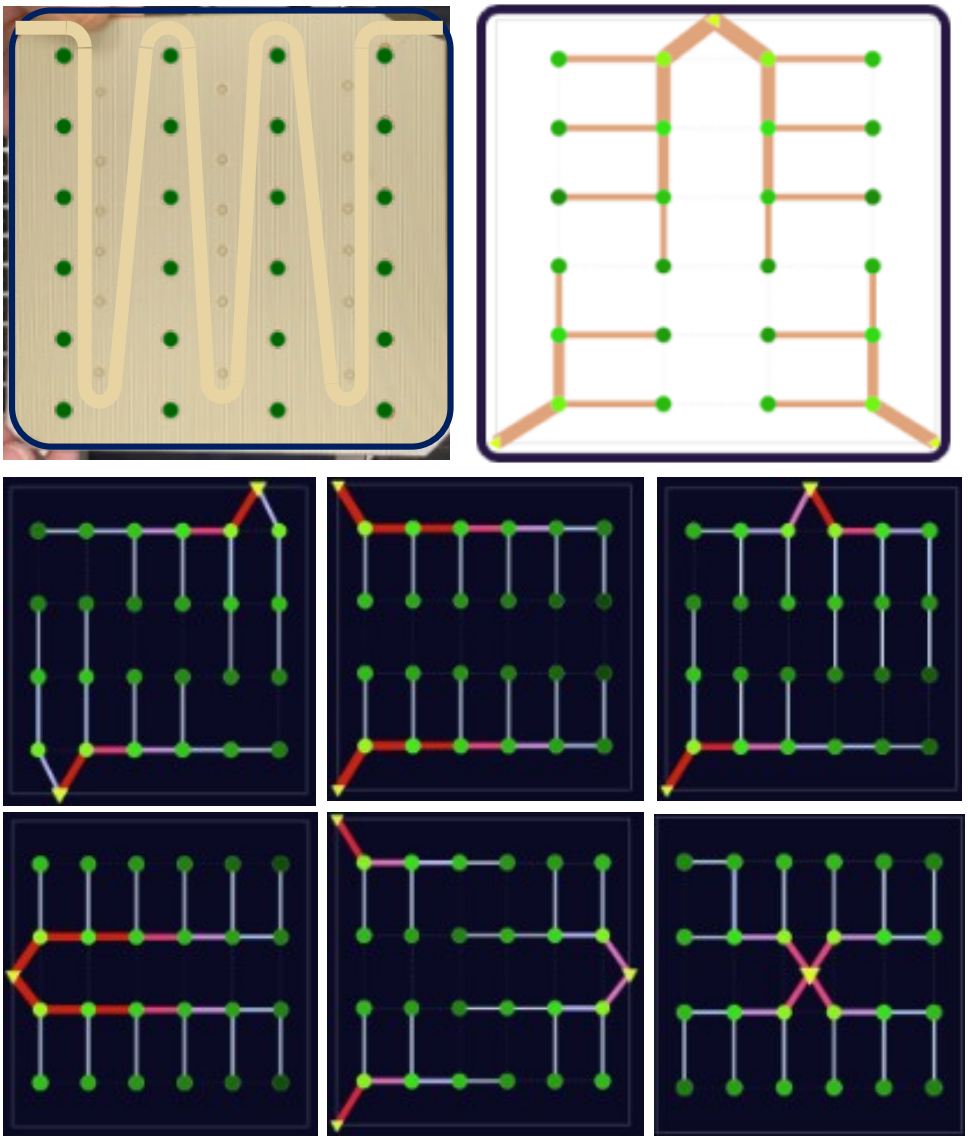


Figure D.1: GridGraph optimization of back-contact vias in a solar cell module utilized by SILFAB SOLAR. [Top Left] shows the real layout of via locations at the back side of one solar cell. The remaining images are various solutions to the routing problem with different current sink locations.

the back-contact vias in a solar cell module. Each via is its own sink for electrode patterns on the front side of the module, but they must all in turn be routed to a common solar cell module-level current sink. Figure D.1 shows some examples of GridGraph-optimized networks with different sink placements. Extension to this and other problems related to solar cell optimization could provide insights to numerous energy and mass-diffusing systems.

Appendix E

INTRODUCTION TO DYNAMIC MODE DECOMPOSITION

DMD is a method for identifying the principal spatiotemporal modes in a high-dimensional dynamical system. Specifically, it approximates the linear dynamics operator A satisfying $X' \approx AX$ in a compact, orthogonal basis set found by performing the truncated singular value decomposition (SVD) of the time series data X:

$$X \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V^*} \tag{E.1}$$

where rows of X' are the next time step of each row vector in X. The dynamics in this truncated base are:

$$\widetilde{A} = \widetilde{U}X'\widetilde{V}\widetilde{\Sigma}^{-1} \tag{E.2}$$

And the state vector at time t can be reconstructed with:

$$x_t = \widetilde{U}\widetilde{A}^t\widetilde{U}^T x_0 \tag{E.3}$$

where the $\widetilde{A}$ is raised to the t-th power. Traditionally, DMD will additionally identify real-space modes using the eigenvectors of $\widetilde{A}$ and evolve the system using powers of the complex eigenvalues of $\widetilde{A}$ - these approaches are computationally favorable, removing the need to perform repeated matrix powers. However, the need to identify frame-zero values for eigenvalues in the DMD mode basis can be performed in several ways, all of them leading to increased error in reconstruction. In this study, X is sufficiently low-dimensional to admit the $\widetilde{A}^t$ reconstruction method above, and this method is used to give DMD the best-case shot at capturing EHD dynamics.

To capture the effect of different control voltages on the EHD process, the control variant of DMD as presented by Proctor, Brunton, and Kutz [112] was explored. DMDc defines the input space as the concatenation of state and control variables, with

$$X' = AX + B\Upsilon \tag{E.4}$$

$$\Omega = \begin{bmatrix} X \\ \Upsilon \end{bmatrix} \tag{E.5}$$

where Υ is the vector of control inputs. The corresponding SVD is

$$\Omega \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V^*} = \begin{bmatrix} \widetilde{U}_1 \\ \widetilde{U}_2 \end{bmatrix} \widetilde{\Sigma}\widetilde{V^*} \tag{E.6}$$

Because this decomposition includes elements of the control inputs which are not part of the reconstruction, a separate SVD is used to provide the spatial vectors used to play the system forward:

$$X\prime \approx \hat{U}\hat{\Sigma}\widehat{V^*} \tag{E.7}$$

Then the dynamics and play-forward formulas for the system are

$$\widetilde{A} \approx \hat{U}^* X\prime \widetilde{V}\widetilde{\Sigma}^{-1}\widetilde{U}_1^*\hat{U} \tag{E.8}$$

$$\widetilde{B} \approx \hat{U}^* X\prime \widetilde{V}\widetilde{\Sigma}^{-1}\widetilde{U}_2^* \tag{E.9}$$

$$x_{t+1} = \hat{U}\widetilde{A}\hat{U}^T x_t + \hat{U}\widetilde{B}u_t \tag{E.10}$$

where u_t is the control input at time t. With the systems established, this framework can, in theory, 'play forward' the system given novel inputs u, allowing DMDc to perform forward prediction and enabling the development of control algorithms for the system.

Appendix F

FINITE ELEMENT MODELS OF EHD JETS

Two CFD simulations were used as inputs to the dynamic model training process. The first was a single run from a 2-dimensional model developed by project collaborators in the lab of Prof. Hantang Qin,[66] and the second was a custom 1-dimensional model that allowed rapid exploration across different control inputs. The 2D model was used to demonstrate the fitness of DMD to model EHD dynamics, and the 1D model provided training data for a DMDc model.

F.1 One 2D EHD Model

The individual frames of a single jet pulse from the Jiang et. al. study[66] were parsed using the same image analysis routine that was applied to high-speed videos in subsection 4.2.1. The final dataset from the 2D COMSOL model comprised a state vector mapping ink volume (and by proxy mass, from the non-compressibility assumption) on the axis running from tip to substrate. This flattened state vector was the training input to a DMD algorithm.

F.2 1D FEA

To assess the impact of a wider set of input parameters, as well as the long-term dynamical evolution of the print head, a 1-dimensional numerical fluid model was developed, modeling the vertical distribution of mass and momentum between tip and substrate. This model made non-physical assumptions to expedite its development and is intended as a test interface for machine learning methods, not as a close approximation of any real system. An example of critical state evolution from a model run from the 1D EHD model could be seen in Figure 4.4. As with the other data sources, the output of the 1D CFD was a state vector representing mass distribution from print tip to substrate over time.

To assess the impact of a wider set of input parameters, as well as the long-term dy-

namical evolution of the print head, a 1-dimensional numerical fluid model was developed, modeling the vertical distribution of mass and momentum between tip and substrate. The model was constructed in Python NumPy, with a state consisting of mass and momentum vectors and a scalar voltage, representing the effective degree of polarization between the substrate and the ink. This effective voltage evolves according to:

$$V_{eff}\prime = C_{recharge}(V_c - V_{eff}) - C_{discharge} m_{eject} \tag{F.1}$$

Where $C_{recharge}$ and $C_{discharge}$ are constants governing the rates of control-driven polarization and ink-driven depolarization respectively, V_c is the control voltage, and m_{eject} is the quantity of ink present in the substrate-facing element of the mass state vector.

The model enforces a uniform full column of ink at its tip-facing element and zero mass at its substrate-facing element so that the tip and substrate function as a perfect source and sink. With mass and momentum conserved, fluid acceleration is calculated from the sum of four principal forces: gravity, viscosity, surface tension, and coulombic attraction, as per the following equations:

$$F_{gravity} = C_{gravity} M \tag{F.2}$$

$$F_{viscosity} = C_{viscosity} M \, \nabla^2 \frac{U}{M} \tag{F.3}$$

$$F_{surface} = -C_{surface} \, \nabla\kappa \, \pi r^2 \tag{F.4}$$

$$F_{charge} = C_{charge} \frac{V_{eff} M \kappa}{x^3} \tag{F.5}$$

where M and U are the mass and momentum state vectors, r is the cross-sectional radius of the ink column, and x is the distance to the substrate. $C_{gravity}$, $C_{viscosity}$, $C_{surface}$, and C_{charge} are fitting constants. Surface curvature κ is estimated geometrically as

$$\kappa = \frac{sin\theta}{r} \tag{F.6}$$

where

$$\theta = arctan\frac{\nabla r}{\delta x} \tag{F.7}$$

and δx is the size of a vertical element. Source code for the 1D Python model can be found at `https://github.com/onakanob/EHD_1D`. The 1D model dataset generated state vectors including mass and momentum between tip and substrate, as well as theV_{eff}. The only control input was the control voltage V_c.

Appendix G

HARDWARE AND IMAGING METHODS FOR EHD DATASET CREATION

G.1 Hardware: Integrating the SIJ-150 with a Function Generator

Voltage waveform generation is an extremely well-worn practice, and there are a variety of approaches to creating controlling waveforms that can drive the EHD process. The physical system that outputs the waveform would need to be flexible, capable of generating arbitrary inputs at various frequencies and voltages. It would need to either accept inputs from a PC Python environment or log and output the inputs that it had generated so that the waveforms could be captured, matched against measurements of the print outcomes that they had caused, and tabulated for later use as training data for ML models. Lastly, the system needed to achieve a high throughput of waveforms to support the creation of ML models. We anticipated that some models could need on the order of 1,000 training data to show good performance. At a rate of 3 seconds per experiment, the apparatus could generate 1,200 experiments in an hour - this was an approximate throughput target.

The printer available at the WCET to perform this work was an SIJ-150 model EHD printer manufactured by Super InkJet, Japan. SIJ's printer is modular in its construction. It uses a National Instruments function generator that takes inputs from a LabView software client and passes waveforms between roughly -10V and 10V to a 300x signal amplifier. The printer's base software only allows the user to define sine waves and square waves with a 25%, 50%, or 75% duty cycle. Each recipe can use a maximum of nine predefined waveforms, and modifications to those recipes can only be done manually by the user during the run.

A direct rewrite of the EHD control software would be needed to achieve the throughput and logging capabilities that the project required. But the manufacturer suggested an alternate approach: we would introduce a new dedicated signal generator into the control loop and use the existing signal generator as a process heartbeat. We purchased a Tektronix

AFG1062 arbitrary function generator (AFG) and inserted it into the control loop of the SIJ-150, as illustrated in the middle pane of Figure 5.2. The AFG produces arbitrary waveforms which are defined using up to 8000-element vectors. The vectors can be generated and logged programmatically in a Python environment, and an open API saves each waveform to a buffer in AFG memory. (It takes about 2 seconds to import a wavefunction to the AFG using this interface.) A 1V to 2V square wave impulse from the EHD controller serves as a trigger, causing the AFG to pass the custom waveform to the signal amplifier and onto the SIJ print tip. Because the amplifier has a 300x gain, an input of 1V from the AFG would produce a 300V potential at the top of the print nozzle.

G.2 Rapid Serial Experiments with an Arbitrary Function Generator

Experiments with the AFG proceeded along a serpentine path on the substrate. Every set of experiments employed a tip velocity of 100 microns per second, with a pitch of 300 microns between each serpentine pass. The entire pattern occupied one square centimeter on the substrate. The SIJ-150 would deliver a 2 Hz square or sine wave between 0 and 1.3 V to the trigger input of the AFG, and the AFG was set to single burst mode with a waveform duration of $\frac{1}{4}s$. For waveforms defined by vectors of 8,000 float values, each "tick" of the waveform lasts $3.125 * 10^{-5}$ seconds, or roughly 31 μs. This setup ensured that each play-through of a particular waveform would be separated from its neighbors by $\frac{1}{4}s$, and printed dots would be deposited at a pitch (i.e., spatial frequency) of 50 μm along the printed line.

New waveforms were delivered to the AFG every few seconds. The waveform would "play" for 1-3 seconds (100 to 300 μm), or roughly three to seven repeats of each waveform. Then a zero-voltage flat line waveform would provide a space of 1-2 seconds (100 to 200 μm) before the following waveform was executed. In total, the length of the serpentine printed line within the 1 cm square area was roughly 34 cm in length and would take roughly 1 hour to traverse, producing between 600 and 700 experiments depending on the precise length of each experimental set. The bottom panel of Figure 5.2 illustrates the process of rapid serial experimentation.

The physical experiment layout required tradeoffs between efficiency in materials, time,

and space and implicit physical bounds placed on the process. The 1 cm^2 area was chosen to keep each experimental pass within an area that would be possible to image in a reasonable time using an optical microscope with a motorized stage and an automated stitching process. The largest line widths in optimized solar cell designs tended to be on the order of 100 μm wide. The choice of 300 μm serpentine pitch allowed printing of 100 to 200 μm feature widths without causing overlap between adjacent rows. Then, the 100 $\mu m/s$ velocity brought the time to complete one full pass to an hour for ease of managing the tool schedule. Finally, the print frequency of 2 Hz provided a middle ground between cold start printing, where no ink had passed through the tip in several seconds, and a continuous print schedule that could see many droplets deposited within one second.

G.3 Attempts at In-Situ EHD Monitoring

To perform in-situ measurements of a live EHD process, two principle strategies were outlined. First, a microscope mount was devised to enable in-situ photography of the SIJ-150 printer nozzle. Second, a project was launched in collaboration with the Qin lab at Iowa State Univeristy and the University of Wisconson at Madison to implement laser scattering measurements of EHD droplets in-flight that would, in principle, allow for the characterization of ink jets below the optical resolution limit of 0.3 μm.

G.3.1 In-Situ Optics

An optical microscope with a long-focal-length macro lens was purchased to take images of the print tip and printed features on the substrate, providing real-time feedback on process outputs. It could potentially automate the dataset generation process, significantly reducing the manpower overhead in training ML models. A mounting system was devised early in the project, pictured in Figure G.1. The equipment was carefully specified so that a 45-degree mount angle would allow the lens housing to sit close to the print head, placing the focal plane in line with the EHD process without touching (and shorting out) either the print head or substrate.

Unfortunately, to integrate the microscope into an automated workflow, custom software would need to take direct control of the microscope camera, capturing images in real time.

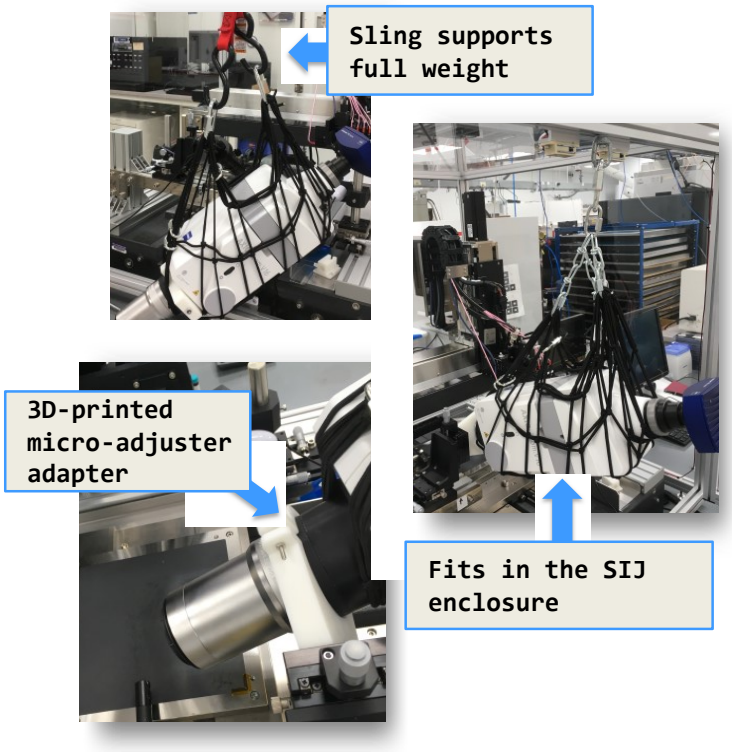


Figure G.1: Pictures of a sling-mounted microscope intended for in-situ imaging of a live EHD process. A flexible sling and 3D-printed collar support the scope's weight while allowing microadjusters to focus the image.

This feature, we discovered, was locked behind a licensing paywall that had not been described when the microscope was initially purchased from Leica. Other unsolved problems remained with the system. The ability of the microscope to capture printed features on the moving substrate remained unproven, and the input control loop lacked a practical method for adjusting process inputs based on fed-back in-situ data. For this reason, we decided not to pursue the extra funds needed to bring the microscope online, and development efforts turned towards implementing an in-situ laser scattering system.

G.3.2 Laser Scattering

The details of the laser scattering developed for in-situ EHD monitoring are pending publication, and I will refrain from reproducing them here in detail. An example of the direct read-out from the system, deployed on an SIJ printer at the WCET, can be seen in Figure G.2.

A major challenge throughout the project was separating the scattering signal from the primary beam. Even with a spot size theoretically under 10 μm, the Gaussian cross-section of the laser beam extended over 100 μm out onto the sensor. As printing voltages increased, however, a mixture of scattering and refraction could be observed from the inkjet and correlated with feature sizes on the substrate.

However, persistent problems chased the laser drop watcher system. Most critically, the spreading of the laser spot size dictated standoff distances that were an order of magnitude larger than those preferred when printing with the SIJ printer. In addition, printing voltages above 1000 V were necessary to achieve measurable results, while the experimental sets generated elsewhere in the project typically used ¡400 V amplitudes. Therefore, the laser system did not see use as an input to the ML training effort. Instead, we decided to pursue an ex-situ measurement strategy.

G.4 The Ex-Situ Image Analysis Pipeline

A workflow was needed to process large-area images of many EHD experiments into useable tabulated measurements for ML model training. The software solutions would combine manual checks with automated image analysis operations. Code can be found at

https://github.com/onakanob/ehd-dataset, and the major steps in the pipeline are summarized below.

G.4.1 Alignment

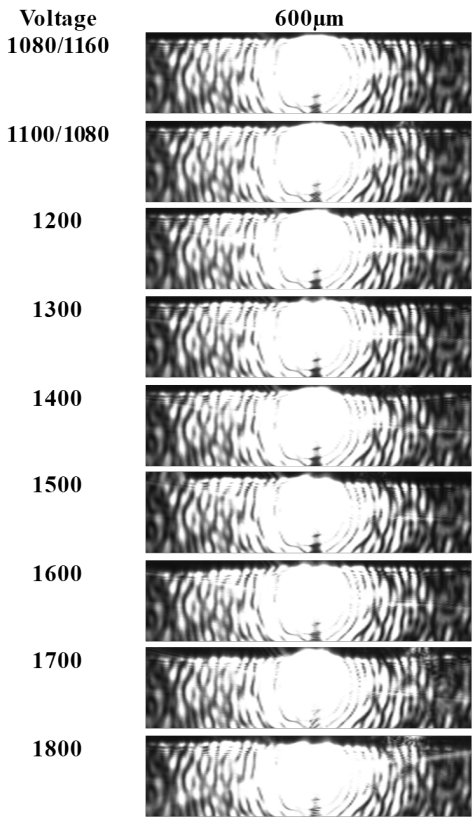


Figure G.2: Laser scattering signals at different printing voltages. Primary signal is overwhelming, but the background shows a diffuse signal.

The analysis pipeline populates a JSON file that will eventually direct the analysis of each experiment patch within the large-area image. The first step involves the alignment of the tool path file, which the printer used to perform X/Y translational operations, with the image. The user adjusts scale, rotation, and translational parameters in the JSON file. A software utility displays the micrograph with an overlay of the tool pattern to aid in this alignment process. The scale of the image is set during this step, with typical values of 450 to 500 pixels/mm, or just over 2 μm per pixel.

Once the tool path is aligned, the user locks in initial guesses of the positions of individual experiments within the large image by specifying their displacement along the tool path. An initial offset, spacing, and experiment count are defined at this stage, along with initial guesses at image processing parameters.

G.4.2 Placement

Next, the user extracts patches from the image individually for manual verification. Figure G.3 shows a frame of a software utility that displays each patch, one by one, in printed

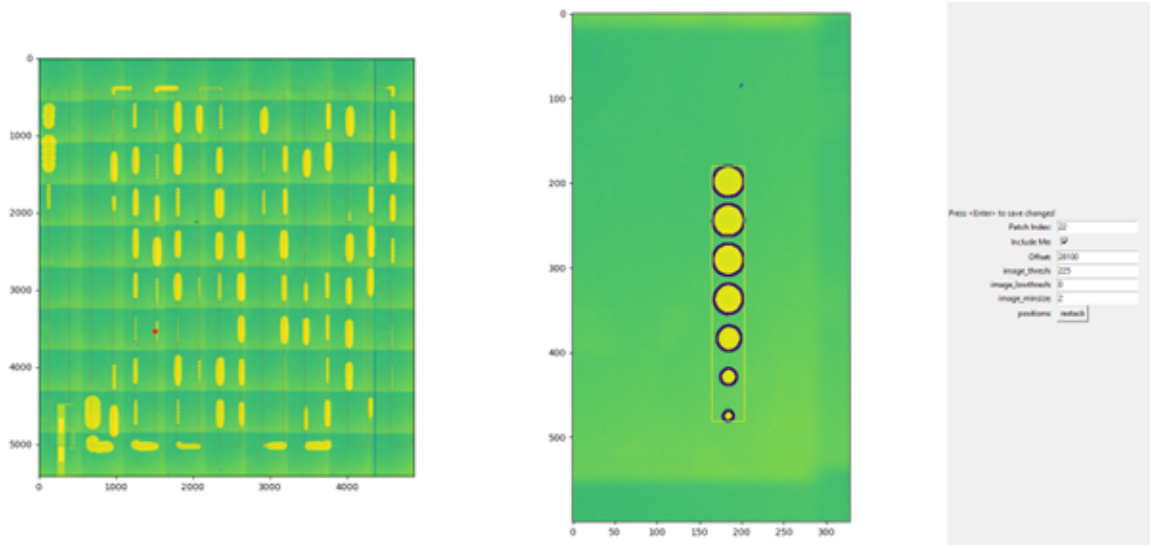


Figure G.3: Example of the pattern alignment (left) and patch image processing (right) graphical user interface for EHD dataset parsing.

order. The user can fine-tune the patch placement along the tool path, adjust image analysis parameters, and exclude individual patches from the dataset.

Three parameters govern the image analysis of individual patches. First, upper and lower brightness thresholds segregate each image into background and foreground. Substrate pixels should lie between the threshold values. The printed metal ink traces are brighter (greater than the upper threshold) or darker (less than the lower threshold) than the underlying silicon substrate. A minimum pixel count filter is applied to each region of printed ink to remove noise from the printed area. Finally, a minimum-area bounding box delineates the remaining detected area. Users can adjust the three parameters to fine-tune the analysis results of each patch.

G.4.3 Post-processing and compilation

When all patches are aligned, a batch script parses the entire set of patches. Image analysis is performed on each patch using manually-adjusted parameters. The total printed area,

length and width of the bounding box, the number of isolated patches of ink, and the average width (i.e., the total area divided by the bounding box length) are all logged.

At this stage, the user can manually designate clogs in the measurements table. Full clogs show as long periods of non-jetting, followed by at least one very large feature that resulted from a high-voltage pulse that cleared the clog. The print tip will generally start each print run in a clogged state, so the first feature is typically a clogging event; it is up to the user to manually flag any subsequent clogging events that happened in the course of the run. Clogged experiments are flagged but not automatically removed from the dataset in case they become useful in future analyses. However, unless otherwise specified, clogged data points were removed from all of the downstream analyses and ML training loops that will be discussed in the rest of this study.

Because the first few waveforms are almost always clogged, it is usually the case that the initial waveforms in the experimental log do not show on the substrate and should be skipped for analysis. The image patch measurements pick up some time into the experiment after the initial clog was cleared. To determine this offset, the Pearson correlation between the total area under the waveform curve and the total area of printed ink is calculated at every possible offset between the waveform and image analysis logs. The offset producing the greatest correlation between the two quantities, which can be as low as 0.27 or as high as 0.85 in practice, is chosen. With inputs matched against outputs, the two halves of the dataset can be collated and added to the multitask dataset.

G.4.4 Errors and misfits

In addition to persistent clogging, other errors occurred during printing and image processing that caused experiments to be excluded or errors to creep into the EHD dataset. The stage alignment of the Olympus microscope was imperfect, causing occasional mismatches between adjacent pictures in the mosaic. Experiments that straddled two adjacent images would often appear discontinuous, yielding a nonsensical bounding box. Likewise, experiments that turned a corner at the hairpin ends of the serpentine tool path would generate inflated bounding boxes and needed to be excluded. Debris on the substrate could cause

false positives on the dark end of the pixel threshold, and unevenness in background illumination would occasionally cause the substrate to appear brighter than some small-area prints. These errors would also trigger exclusion from the database.

More insidious were minor size errors that accumulated due to the manual adjustments in image processing settings that were needed to correctly threshold and bound certain prints. Even for a well-defined pattern, the edges of features are not perfectly sharp. Increasing and decreasing threshold values would necessarily cause measured print boundaries to become slightly larger and softer. Likewise, for dot prints approaching pixel size, it was common for one or more legitimate prints to be excluded from analysis to filter out debris or other false positives in the image patch. The "max_width" measure for these experiments will be accurate, but many of the other "ytype" return types will not be. For future projects reliant on area, print length, etc., a re-parse of the dataset would be advisable.

Appendix H

INTRODUCTION TO SUPERVISED MACHINE LEARNING

The terms artificial intelligence (AI) and machine learning (ML) are often used synonymously, but in truth they are distinct concepts. AI denotes any artificial agent that acts to maximize a goal or reward within a system. ML can be considered a sub-type of AI where action is performed on data, typically with the goal of making predictions about the system. A "model" in this context would be an agent built in software to perform either AI or ML, and "data" is a set of numerical observations drawn from the system of interest. Neural networks (NN) are a popular modern implementation of ML models – they are multi-layer software models that incorporate alternating linear and non-linear operations, utilizing from tens to billions of parameters that can be tuned to reproduce training data. However, numerous effective ML model types exist, including decision trees, neighbor models, and linear regression. In this analysis, suffice to say that these models range from relatively simple to extremely complex, with complexity measured roughly in the number of free parameters.

"Supervised" ML is the practice of training an ML model on tabular data that includes some known "correct answers". The reward function in supervised ML is a measure of prediction error that between model outputs and these known answers which the model seeks to minimize. The inputs "X" should be the columns of a dataset that are readily available at deployment time, and the outputs "Y" should be quantities that need to be predicted at deployment. For the EHD problem, X is composed of numerical parameters that define input waveforms, which are the control input interface during an EHD print run. Outputs Y are some measurement, area or width for instance, or the printed features. These measurements are the property that we ultimately want to control, so they are the prediction target when training ML models.

The overarching goal for EHD control is an AI problem: what inputs should the control agent provide to the printer to produce the goal, an on-demand printed feature of a

precise size? The action interface in this system is the waveform voltage delivered to the printer tip. If a successful supervised ML model can be trained on the $Y = f(X)$ interface described above, it will become possible to implement a model-based control solution to the AI problem. Given a successful predictor f that accurately outputs feature size y given waveform x, and a target feature size of $y*$:

$X \leftarrow$ possible input waveforms

$Y' \leftarrow f(X)$ ▷ Y' are the predicted feature size outcomes.

$i \leftarrow argmin\ Y' - y*$

do X_i

The key to this control approach is the creation of an accurate model f. Meaningful trends in the EHD system can be framed as both classification and regression problems, so we will need metrics by which to evaluate the success of prospective models f. It will also be useful to define a naive baseline ML model, which can be used as a reference signal when attempting to judge results from more complex models. The "MLE" model, or maximum likelihood estimator, will serve this purpose. MLE always gives the same answer, regardless of input x. For a classification task, the MLE will guess the class that it has seen most often in the training data. For regression task, MLE will guess the mean of the answers from the training data.

Classification models seek to cluster query data into two or more classes. EHD control tasks such as predicting if a given waveform input will cause ink to jet or the nozzle to clog are classification problems. A common method of evaluating classification models, illustrated in Figure H.1 [B], is known as the receiver operator characteristic (ROC) curve. This is calculated by sweeping the decision boundary of a classifier model from minimum (everything is True) to maximum (everything is False) extremes. The balance of a model's false positives and false negatives over the course of that sweep draw the ROC curve. As a model approaches perfect separation of two classes, the area under the ROC curve (AUC) approaches 1.0; a model that flips a coin to choose between two classes will receive an AUC-ROC score of 0.5 in expectation. The classification MLE is insensitive to the ROC sweep

because it always gives the same answer, and should always receive an AUC-ROC of 0.5.

Another standard evaluation of classification performance is the F1 score, defined as

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

This is the geometric mean of precision (fraction of $True$ answers correctly classified) and recall (fraction of $True$ predictions correctly classified). It is a good measure of a model's ability to correctly and fully identify a class (the $True$ half of a $True/False$ dataset). Because EHD datasets will typically skew towards printing rather than not printing if the experimenter is paying attention, a classifier MLE will always guess $True$ and so will always achieve a 100% recall. This means that the MLE F1 score will be determined by the proportion of $True$ answers in the dataset, and it will skew towards optimism.

When printing with an EHD, a waveform that does not jet represents a significant failure mode - where a feature was intended, no pattern was produced! So jet prediction is practically useful; we will attempt to achieve AUC and F1 measures of at least 0.9 on the jet prediction task.

In contrast to classification, regression models seek to predict a numerical quantity with precision. Many possible measures of error are in common use, including mean absolute error (MAE), mean square error (MSE), mean square log error (MSLE), and mean absolute percent error (MAPE). MSE, defined as $MSE = \frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - \hat{y}_i\right)^2$ where $\hat{y}_i$ is the correct answer for input x_i, is the primary cost function for most of the regression model training in this study. However, MAE will be the first primary metric for evaluating models. MAE is defined as $MAE = \frac{1}{N}\sum_{i=1}^{N} abs\big(f(x_i) - \hat{y}_i\big)$. It is more easily interpreted in real-world units than MSE for this application. For instance, if a model predicting feature width measured in μm achieves an MAE of 12, that means it missed its size predictions by, on average, 12 μm.

Correlation constants are another useful way to characterize the relationship between true and predicted answers that are decoupled from the dimensionality of the task data. Pearson's correlation coefficient r measures the linear correlation between predicted values and true values of a model. It is equal to the normalized covariance of the two variables

and will always be between -1 and 1, with $r = 0$ indicating no correlation between the two and $r = 1$ indicating perfect correlation. (It is worth noting that $r = 1$ does *not* indicate that the model is always right! Biased models, for instance $f(x) = 2\hat{y}$ or $f(x) = \hat{y} + 10$, will nonetheless be perfectly correlated.)

A regression MLE will always guess the mean of the training data. This means that if a test set is not biased relative to the training set, the MAE of the MLE model will be the mean absolute deviation of the test set in expectation. Because it always guesses the same value, the output of the MLE can have no correlation with any dataset: its r will always be 0.

Without knowing how much noise was baked into the evaluating dataset, it was difficult to choose a target MAE ahead of time. A bare minimum baseline could be to shoot for MAE $< 10\mu m$, or to predict within 4 pixels in the ex-situ microscope images. An MAE $= 5\mu m$, or within 2 pixels, would likely start to approach the scale of measurement error within the dataset. We definitely wanted to see $r > 0.85$ in a highly performant regression model.

H.1 Bias vs Variance Tradeoff

A central problem in machine learning is the trade-off between "underfit" biased models, where the model is insufficiently powerful to capture an underlying trend in data, and "overfit" high-variance models, where the model is overpowered with respect to the data and essentially memorizes the training dataset, losing the forest for the trees. This balance is illustrated in Figure H.1 [C] – a central challenge of ML engineering is to choose a level of model complexity that optimizes the balance between these two effects. This is sometimes called the double descent problem because increasing and decreasing the model size from this optimum point will both hurt performance. Improvements may be made to a biased model by deploying a more descriptive architecture, i.e. by adding free parameters. A high-variance model might benefit from simplifying the model architecture by removing parameters or by adding more training data.

"More data" could be the mantra of contemporary machine learning. Big data is dominant in 2023, and it seems unlikely to change any time soon. Recent triumphs in ML and

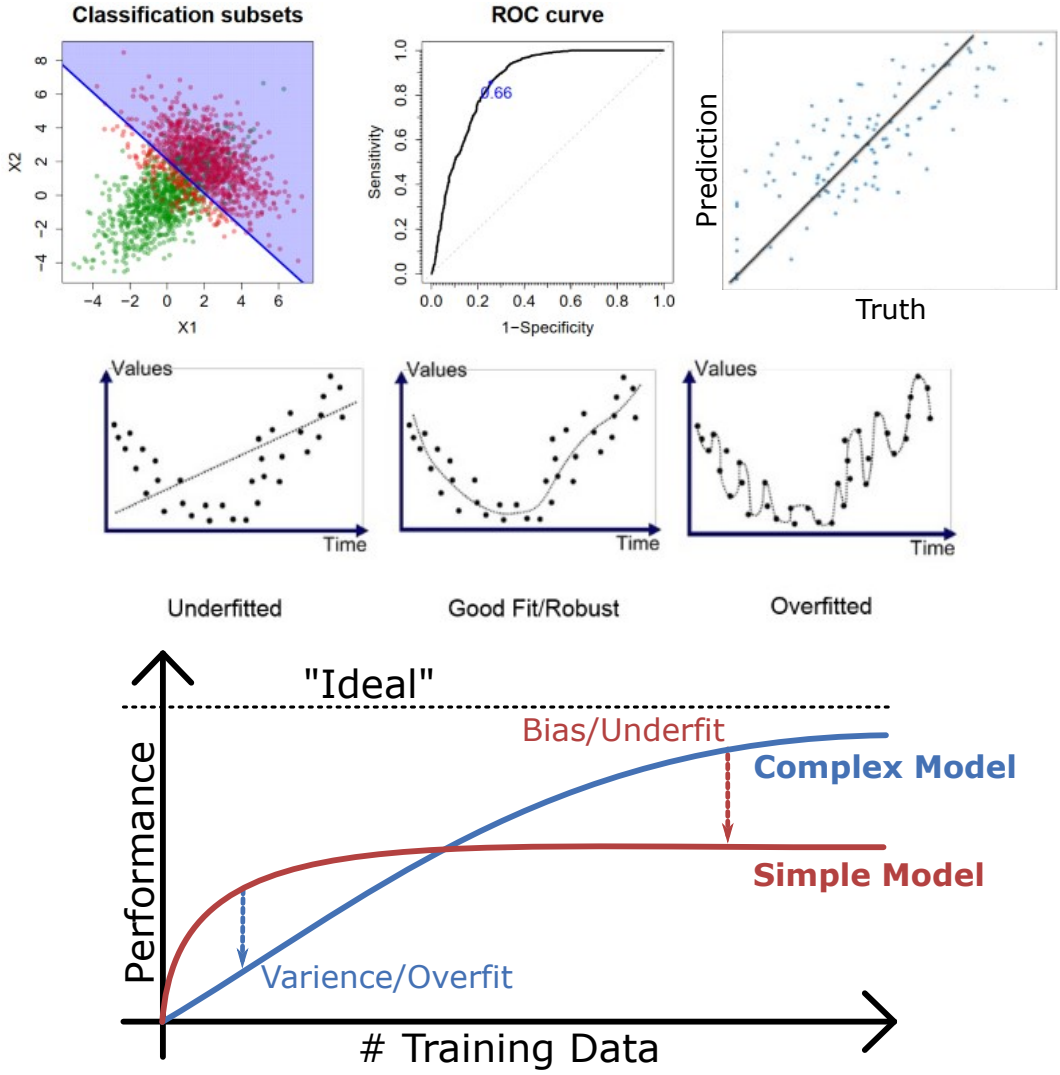


Figure H.1: Illustrations of ML concepts: [A and B] the ROC curve illustrates a model's ability to separate two classes of data.[110] [C] correlation values measure the agreement between model and true values of data points in a regression task. [D] Underfit models fail to capture trends in data, while overfit models hew too closely to training points. Credit:`https://medium.com/greyatom/` [E] A generalized view of model performance as the number of available training data points increases. More complex (i.e., having more free trainable parameters) may achieve better performance but require a larger supporting dataset to do so.

AI have hinged on massive data sets, from beating human players at the game of go [131] to folding proteins,[68] generating images,[133] and reproducing natural written language.[102] What these models have in common is access to truly massive supporting datasets, either by generating those datasets in-silico as in self-play reinforcement learning or by taking advantage of humanity-scale collections of protein data, pictures, or text.

Creating data on this scale is not always practical. Even when practical, it may not always be advantageous. For example, in a manufacturing setting, even if it is possible to accumulate a thousand calibration data samples to train a model before launching a process, the cost in time and materials to run those calibration points will offset any gains in downstream yield that the model might enable. (One thousand samples is both very *small* in ML terms and untenably large in manufacturing terms.)

H.2 Few-Shot and Transfer Learning

It would be nice to use data points from past experiments to train models for the future, but that way lies challenges. Simply combining multiple experiments into a single dataset neglects the hidden process variables such as ink, tip, substrate, and environmental conditions that vary from run to run. (For the EHD dataset, we will soon see that these per-run dice rolls can be an almost overwhelming source of variation.) A method is needed to take advantage of many disparate datasets to train models powerful enough to capture the dynamics of the EHD while also allowing for rapid calibration at test time with the smallest possible set of new data points. The ML discipline that describes methods for training models with a minimum amount of supporting data is known as few-shot learning.

Few-shot learning focuses on training models with small datasets, and often employs some form of "transfer", where knowledge is imported from larger, pre-trained models. The concept is illustrated in Figure H.2. There are a plethora of approaches and strategies to implement transfer learning. Some focus on feature manipulations, modifying or transforming the input representations in a problem to efficient or descriptive vector space. The vector space could be found by pre-training networks on other similar problems, or by employing unsupervised methods to de-noise or cluster inputs. Some methods may transplant entire pre-trained networks or selected layers of pre-trained networks into a new problem,

performing re-training of some portion of the network's learned weights and biases.[163, 21]

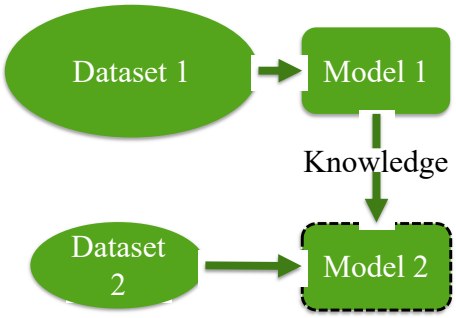


Figure H.2: A schematic view of transfer learning. Knowledge from model 1, trained on a large dataset, is transferred to model 2, increasing performance on a task with a smaller supporting dataset.

Starting an ML training loop with a pre-trained network - also called a "warm start" - does not absolve the practitioner of the need to balance bias and variance. In fact it can make the problem harder by requiring two different solutions to the double descent problem: once in pre-training with a large supporting dataset, then again after the warm start with a presumably smaller dataset. The challenge only grows as this second dataset shrinks: from few-shot learning with small data, to one-shot learning with only a single new example, to the platonic ideal of zero-shot learning, where an old model performs a new task with no retraining at all.