Machine Learning in Adversarial Settings: Attacks and Defenses

Hossein Hosseini

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

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Radha Poovendran, Chair

Payman Arabshahi

Sreeram Kannan

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Machine Learning in Adversarial Settings: Attacks and Defenses

Hossein Hosseini

Chair of the Supervisory Committee:
Professor Radha Poovendran
Electrical and Computer Engineering

Deep neural networks have achieved remarkable success over the last decade in a variety of tasks. Such models are, however, typically designed and developed with the implicit assumption that they will be deployed in benign settings. With the increasing use of learning systems in security-sensitive and safety-critical application, such as banking, medical diagnosis, and autonomous cars, it is important to study and evaluate their performance in adversarial settings.

The security of machine learning systems has been studied from different perspectives. Learning models are subject to attacks at both training and test phases. The main threat at test time is evasion attack, in which the attacker subtly modifies input data such that a human observer would perceive the original content, but the model generates different outputs. Such inputs, known as adversarial examples, has been used to attack voice interfaces, face-recognition systems and text classifiers.

The goal of this dissertation is to investigate the test-time vulnerabilities of machine learning systems in adversarial settings and develop robust defensive mechanisms. The dissertation covers two classes of models, 1) commercial ML products developed by Google, namely Perspective, Cloud Vision, and Cloud Video Intelligence APIs, and 2) state-of-the-art image classification algorithms. In both cases, we propose novel test-time attack algorithms and also present defense methods against such attacks.

TABLE OF CONTENTS

LIST OF FIGURES

DEDICATION

to my loving wife

Chapter 1

INTRODUCTION

Deep neural networks have shown remarkable performance in a variety of tasks such as visual classification, translation and game playing [99, 155, 120]. Such models are typically designed and developed with the implicit assumption that they will be deployed in benign settings. Learning systems are, however, increasingly applied in security-sensitive and safety-critical systems such as banking, medical diagnosis, and autonomous cars. Hence, it is important to study and evaluate their performance in adversarial settings.

The security of machine learning (ML) models has been studied from different perspectives [16, 15, 22, 8, 132]. Learning models are subject to attacks at both training and test phases. The main threat during training is poisoning attack, which is injecting fake samples into the training data or perturbing the training samples in order to influence the model [87, 23]. During test time, the adversary may try extract private training data, e.g., medical data, from the trained model [152]. To protect the training data, several papers have proposed algorithms for privacy preserving ML models [151, 5]. A related threat is extracting model parameters by querying the model multiple times [172].

It has been shown that ML models can be deceived when tested by perturbed inputs or out-of-distribution samples [21, 103, 166, 65, 125, 131, 191]. This property has been used to attack voice interfaces [34], face-recognition systems [148] and text classifiers [139]. Such attacks can be classified according to the adversary's access to the system. In white-box model, the adversary is assumed to have some information about the learning algorithm, while the black-box model assumes that the adversary only has an oracle access to the system [132]. One attack approach in black-box scenario is to generate adversarial data based on a substitute model and transfer them to the target model [129].

We investigate the vulnerabilities of machine learning systems in adversarial settings and develop robust defensive mechanisms. Specifically, we explore techniques to manipulate input data in order to cause the model to make a mistake. We study two classes of models, 1) commercial ML products developed by Google, namely Perspective, Cloud Vision, and Cloud Video Intelligence APIs, and 2) state-of-the-art image classification algorithms. We show that an adversary can subtly modify input data in such a way that a human observer would perceive the original content, but the model generates different outputs. We then propose selective classification approach for rejecting adversarial examples. We show that using an adversarial training procedure for classifying adversarial examples into an "outlier" label improves the trade-off between standard accuracy and adversarial robustness. In the following, we review our works and summarize our contributions.

1.1 Attacking Real-World Machine Learning Systems

In recent years, ML techniques have been extensively deployed for a variety of applications. Success of ML algorithms has led to an explosion in demand. To further broaden and simplify the use of ML algorithms, cloud-based services offered by Amazon, Google, Microsoft, BigML, and others have developed ML-as-a-service tools. Thus, users and companies can readily benefit from ML applications without having to train or host their own models. To assess the robustness of current commercial machine learning systems, we analyzed Google cloud APIs, namely Perspective API developed for detecting toxic language, Cloud Vision API developed for image classification, and Cloud Video Intelligence API developed for video annotation. In the following, we review our attacks on Google APIs.

Attacking Google Perspective API. Social media platforms provide an environment where people can freely engage in discussions. Unfortunately, they also enable several problems, such as online harassment. In 2017, Google and Jigsaw started a project called Perspective, which uses ML to automatically detect online insults, harassment, and abusive speech. The API enables the developers to use the toxic detector running on Google's servers, to identify harassment and abuse on social media or more efficiently filtering invective from the comments on a news website. Jigsaw has partnered with online communities and publishers to implement this toxicity measurement system.

Table 1.1: Demonstration of the Attack on Perspective Toxic Detection System. All phrases in the first column of the table are chosen from the examples provided by the Perspective website.

Original Phrase (Toxicity Score)	**Modified Phrase (Toxicity Score)**
Climate change is happening and it's not changing in our favor. If you think differently you're an **idiot**. (84%)	Climate change is happening and it's not changing in our favor. If you think differently you're an **idiiot**. (20%)
They're **stupid**, it's getting warmer, we should enjoy it while it lasts (86%)	They're **st.upid**, it's getting warmer, we should enjoy it while it lasts (2%)
They are liberal **idiots** who are **uneducated** (90%)	They are liberal **i.diots** who are **un.educated** (15%)
idiots. backward thinking people. **nationalists**. not accepting facts. susceptible to **lies**. (80%)	**idiiots**. backward thinking people. **nationaalists**. not accepting facts. susceptible to **l.ies**. (17%)
They are **stupid** and **ignorant** with no class (91%)	They are **st.upid** and **ig.norant** with no class (11%)
It's **stupid** and wrong (89%)	It's **stuipd** and wrong (17%)
If they voted for Hilary they are **idiots** (90%)	If they voted for Hilary they are **id.iots** (12%)
Anyone who voted for Trump is a **moron** (80%)	Anyone who voted for Trump is a **mo.ron** (13%)
Screw you trump supporters (79%)	**S c r e w** you trump supporters (17%)

A demonstration website[1] has been launched, which allows anyone to type a phrase in the Perspective's interface and instantaneously see how it rates on the "toxicity" scale, where toxic is defined as "a rude, disrespectful, or unreasonable comment that is likely to make one leave a discussion." The Perspective website has also open sourced the experiments, models and research data in order to explore the strengths and weaknesses of using machine learning as a tool for online discussion.

One type of the vulnerabilities of machine learning algorithms is that an adversary can change

[1] `https://www.perspectiveapi.com`

the algorithm output by subtly perturbing the input, often unnoticeable by humans. Such inputs are called adversarial examples. In our paper [79], we demonstrated the vulnerability of the Google's Perspective API against adversarial examples. In the text classification task of Perspective API, adversarial examples can be defined as modified texts that contain the same highly abusive language as the original text, but receive a significantly lower toxicity score from the learning model.

We applied our attack on sample phrases provided in Perspective website and showed that we can consistently reduce the toxicity scores to the level of non-toxic phrases, by introducing modifications such as misspelling abusive words or adding punctuations between the letters (Table 1.1). The existence of such adversarial examples is very harmful for toxic detector systems and seriously undermines their usability, especially since these systems are likely to be employed in adversarial settings.

Attacking Google Cloud Vision API. In 2017, Google introduced the Cloud Vision API for image analysis. A demonstration website[2] has been launched, where for any selected image, the API outputs the image labels, identifies and reads the texts contained in the image and detects the faces within the image. It also determines how likely is that the image contains inappropriate contents, including adult, spoof, medical, or violence contents.

In our paper [86], we evaluated the robustness of Google Cloud Vision API to input perturbations. In particular, we investigated whether we can modify an image in such a way that a human observer would perceive its original content, but the API generates different outputs for it. For modifying the images, we add either impulse noise or Gaussian noise to them. Such images might be generated naturally, since real-world image sensors often suffer from noise, blur, and other imperfections.

Our experimental results showed that by adding sufficient noise to the image, the API is deceived into returning labels which are not related to the original image. We showed that the attack is consistently successful, by performing extensive experiments on different image types, including natural images, images containing faces and images with texts. Figure 1.1 illustrates the attack

[2] `https://cloud.google.com/vision`

Original image

Output Label: **Teapot**

Original image

Output Label: **Property**

Original image

Output Label: **Airplane**

Noisy image (10% impulse noise)

Output Label: **Biology**

Noisy image (15% impulse noise)

Output Label: **Ecosystem**

Noisy image (20% impulse noise)

Output Label: **Bird**

Figure 1.1: Illustration of the attack on Google's Cloud Vision API. By adding sufficient noise to the image, we can force the API to output completely different labels. Captions are the labels with the highest confidence returned by the API. For noisy images, none of the output labels are related to corresponding original images. Images are chosen from the ImageNet dataset.

by showing original and noisy images along with the most confident labels returned by the API. Our findings indicate the vulnerability of Google cloud vision API in real-world applications. As an example, an autonomous car may wrongly identify the objects in rainy weather. Moreover, the API can be subject to attacks in adversarial environments. For example, a search engine may suggest irrelevant images to users, or an image filtering system can be bypassed by adding noise to inappropriate images.

We also evaluated different methods for improving the robustness of the system. We only have

a black-box access to the API and cannot change the algorithm. Hence, we assessed whether noise filtering can improve the API performance on noisy inputs, while maintaining the accuracy on clean images. Our experimental results showed that when a filter is applied on input images, the API generates mostly the same outputs for restored images as for original images. This observation suggests that the cloud vision API can readily benefit from noise filtering, without the need for updating the image analysis algorithms.

Attacking Google Cloud Video Intelligence API. With the growth of online media, surveillance and mobile cameras, the amount and size of video databases are increasing at an incredible pace. Therefore, there is a need for automatic video analysis to handle the growing amount of video data. Automatic video analysis can enable searching videos for a specific event, which is helpful in applications such as video surveillance or returning video search results on the web. It can be also used for pre-scanning videos, e.g., in YouTube and Facebook platforms, where distribution of certain types of illegal content is not permitted.

Following the success of deep learning-based visual classification research, there has been a surge in research for video annotation. Internet companies such as Facebook and Twitter are also developing products for analyzing the videos on their platforms. In 2017, Google introduced the Cloud Video Intelligence API for video analysis. A demonstration website[3] has been launched which allows anyone to test the API with videos stored in Google Cloud Storage. The API then quickly returns the video labels (key objects within the video), shot changes (scene changes within the video) and shot labels (description of every shot). By detecting the video shots and labeling them, the API can make videos searchable just as text documents. Thus, users can search for a particular event and get related videos along with the exact timings of the events within the videos.

In our papers [85, 82], we examined the robustness of video analysis algorithms and proposed targeted attacks against two fundamental classes of algorithms, namely video classification and shot detection. We then applied the attacks on Google Cloud Video Intelligence API and showed that by periodically inserting an image within the video at a very rate, the API will return the

[3] `https://cloud.google.com/video-intelligence`

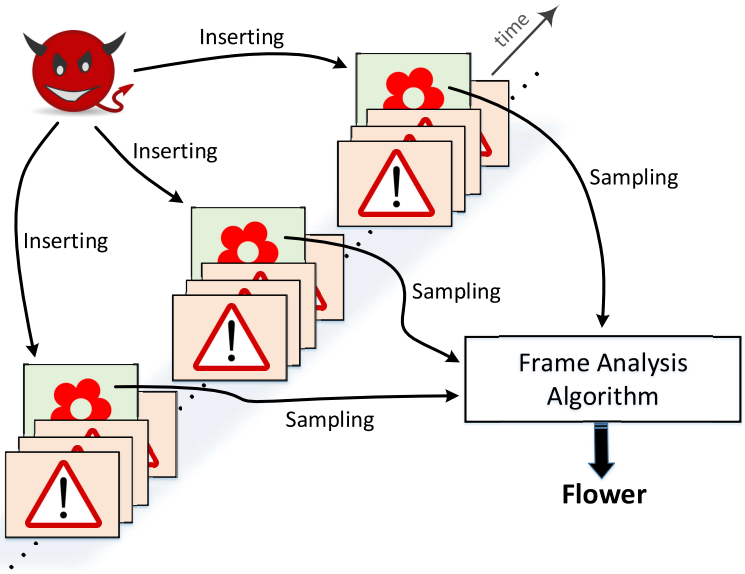


Figure 1.2: Illustration of image insertion attack on video classification algorithms. The adversary modifies the video by placing her chosen image in the sampling locations of the algorithm. As a result, the generated video labels are only related to the inserted image.

adversary's desired outputs. Such vulnerability seriously undermines the applicability of the video analysis system in real-world applications. For example, a search engine may wrongly suggest manipulated videos to users, a video filtering system can be bypassed by slightly modifying a video which has illegal content, or a video search algorithm may miss the important events in surveillance videos. The image insertion attack is illustrated in Figure 1.2. We provide the attack details in Chapter 2.

1.2 Assessing Robustness of Neural Networks Against Input Transformations

Humans have an incredible capability in recognizing unfamiliar objects, by identifying their important features, mainly their shapes. They can also identify objects in various forms such as different scales, orientations, colors or brightness. One of the features of human visual system is the "shape bias" property, i.e., they weight shape more heavily than other dimensions of perceptual similarity,

such as size, texture or color. Convolutional Neural Networks (CNNs) are also designed to take into account the spatial structure of image data. In fact, experiments on image datasets, consisting of triples of a probe image, a shape-match and a color-match, have shown that one-shot learning models display shape bias as well [140].

In our papers [83, 84, 81], we studied how CNNs compare to humans in recognizing shapes and analyzed their capability of "semantic generalization." In doing so, we first examined the performance of CNNs on negative images in different settings. We then extended our work by designing adversarial color-shifted images that can fool state-of-the-art models with high accuracy. In the following, we review our approaches of evaluating the generalizability of CNNs.

1.2.1 Analyzing Shape Bias Property of CNNs

In order to evaluate shape bias property in CNNs, we proposed to train and test CNNs with specifically designed images that represent the same object, but with different colors. One simple way of generating such images is applying image complementing transformation. The transformed image, called negative image, is an image with reversed brightness. Unlike typical transformations used for training or testing machine learning models, image complementing causes a large pixel-wise perturbation to the original images. It, however, maintains the structure (e.g., edges) and semantics of the images, as negative images are often easily recognizable by humans.

In [83], we studied the role of data augmentation, network depth, diversity of training data and complexity of features in the capability of model in recognizing negative images. We show that when training on regular images and testing on negative images, the accuracy of CNNs is significantly lower than when they are tested with regular images. Specifically, we found that the accuracy on negative images is relatively good, only if there is significant diversity within the training data.

In [84], we showed that it is possible to design different CNNs that achieve similar accuracy on original images, but perform significantly differently on negative images, implying that CNNs do not intrinsically require shape bias property to achieve high accuracy on test data. We also showed that, when negative images of some classes are included in training data, CNNs can correctly

recognize negative images of other classes. This is however true, only if the model is trained with batch normalization. Hence, CNNs can indeed learn to be invariant to color, when properly tuned.

We then examined the role of initialization and demonstrated that a different initialization can significantly affect the generalization. Specifically, a model that is initialized by training with original and negative images of a dataset shows shape bias on another dataset as well. Moreover, the shape bias property does not fade away after fine-tuning only with original images of the second dataset. Our results can help better understand the performance of CNNs and develop techniques that more closely imitate the human vision system. The details of experiments are provided in Chapter 3.

The results of our research indicate that current deep leaning models underperform when test data is not exactly distributed as the training data, a scenario that frequently happens in practice. We argue that this is due to the fact that current training methods cause the network to memorize the inputs and, thus, put the burden on the training data to be rich. As a result, the model does not effectively learn the structures of the objects and cannot semantically distinguish between classes. We also suggest that merely computing the accuracy on the test data, that is distributed similarly as the training data, is not representative of the behavior of machine learning models in the wild.

For a particular transformation, we can train the model also on the transformed data to get high accuracy on them. However, relying on training data to cover all aspects of possible novelties at the inference time poses a fundamental limit in adaptation of machine learning systems in real-world applications. While many computer vision problems are data rich, for some critical applications, e.g., training autonomous cars, gathering diverse training data is costly. In essence, learning to reason is the key to "semantical generalization" and can compensate for the lack of diversity in training data.

1.2.2 Semantic Adversarial Examples

Deep neural networks are known to be vulnerable to adversarial examples, i.e., images that are maliciously perturbed to fool the model. Over the past several years, many techniques have been proposed to generate adversarial examples by finding a small perturbation that can fool the model.

Figure 1.3: Samples of CIFAR10 original images (top) and semantic adversarial examples (bottom) on VGG16 network. Adversarial images are generated by converting original images into the HSV color space and randomly shifting the Hue and Saturation components, while keeping Value the same. All images in first row are correctly classified by the model.

Such techniques, including Fast Gradient Sign Method (FGSM) [65], DeepFool [121], Projected Gradient Descent (PGD) [100], and Carlini and Wagner attacks [37], usually involve an optimization problem that solves for a perturbation that maximizes the model prediction error. Generated images, however, contain artificial perturbations that make them somewhat distinguishable from natural images. This property is used by several defense methods to counter adversarial examples by applying denoising filters or training the model to be robust to small perturbations.

In practice, however, the adversary may not be constrained with slightly modifying the image. In our paper [81], we introduced a new class of adversarial examples, namely "Semantic Adversarial Examples," as images that are arbitrarily perturbed to fool the model, but in such a way that the modified image semantically represents the same object as the original image. We formulate the problem of generating such images as a constrained optimization problem and develop an adversarial transformation based on the shape bias property of human cognitive system.

In our method, we first convert the image from RGB into the HSV color space, composed of Hue, Saturation and Value color channels. We then randomly shift the hue and saturation components, while keeping the value the same. This approach generates images that contain the original object with different colors and colorfulness. Our experimental results on CIFAR10 dataset show that the accuracy of VGG16 network on adversarial color-shifted images is 5.7%. Figure 1.3 shows

samples of original images and their corresponding color-shifted images that are misclassified by the model. As can be seen, the original object is recognizable in all of the adversarial images. The attack details are provided in Chapter 3.

1.3 Selective Classification for Rejecting Adversarial Examples

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake [63]. In security-sensitive and safety-critical applications, misclassifying or even accepting adversarial samples can lead to a harmful situation and, hence, it is crucial to be able to reject malicious data points. For example, in medical diagnosis applications, the cost of rejecting a sample might be additional medical tests, which is far more acceptable than taking the risk of deciding based on a potentially perturbed sample. As another example, the vision system in self-driving car can quickly notify the human driver to take over if it identifies that the input is not trusted. Such a classifier that "abstains" from classifying certain inputs is called selective classifier [60].

In our paper, we analyzed the binary classification problem presented in [173] and proved that selective classification improves the robustness. Particularly, we showed that in a setting where the robustness of standard (non-selective) classifier cannot be improved, i.e., when adversarial examples are inevitable, selective classification provides a trade-off between standard accuracy and robustness. We then examined a setting where input features contain both robust and non-robust features and hence robustness of standard classifier can be improved by assigning higher weights to robust features. We showed that in this case selective classifier achieves better trade-off between standard and adversarial accuracy compared to any non-selective classifier.

We then adapted adversarial training procedure to train a selective classifier with an extra class called *outlier*. During training, adversarial examples are generated with perturbation sizes uniformly chosen in range of zero to the maximum perturbation size. The model is trained to classify examples with small perturbations into their true label. Examples with large perturbations are, however, first evaluated by the model. The model is then trained to retain its confidence in the true label and assign the rest of the probability to the outlier label.

We performed experiments on convolutional networks trained with MNIST and CIFAR10 datasets. The models are trained and tested with adversarial examples generated using projected gradient descent (PGD) method [100]. In our selective classifier, an attack is successful if the adversarial image is classified into any label other than the true or outlier labels. On MNIST dataset, we show that our proposed method achieves state-of-the-art adversarial accuracy of 99.2% and 99.6% in white-box and black-box settings respectively, while reaching 99.3% standard accuracy. On CIFAR10 dataset, we implement selective classification on the VGG network and show that the model achieves 81.0% standard accuracy and state-of-the-art adversarial accuracy of 52.4% and 94.7% in white-box and black-box settings, respectively.

Finally, we evaluated the models against characteristic behaviors of gradient masking provided by [13]. In particular, we showed that the models perform well in black-box setting and also against a random attack. Moreover, adversarial accuracy in white-box setting decreases to zero by increasing perturbation size. The results indicate that our defense mechanism does not cause gradient masking.

1.4 Organization of this Thesis

This thesis is organized as follows. Chapter 2 presents the attacks on Google video intelligence API. Chapter 3 investigates the robustness of deep convolutional networks against input transformations and presents the concept of semantic adversarial examples. Chapter 4 proposes a method for detecting adversarial examples based on selective classification framework.

Chapter 2

ATTACKING AUTOMATIC VIDEO ANALYSIS ALGORITHMS: A CASE STUDY OF GOOGLE CLOUD VIDEO INTELLIGENCE API

2.1 Introduction

With the growth of online media, surveillance and mobile cameras, the amount and size of video databases are increasing at an incredible pace. For example, in 2015, YouTube reported that over 400 hours of video are uploaded every minute to their servers [184]. Therefore, there is a need for automatic video analysis to handle the growing amount of video data. Automatic video analysis can enable searching the videos for a specific event, which is helpful in applications such as video surveillance or returning the search results on the web. It can be also used for prescanning user videos, for example in YouTube and Facebook platforms, where distribution of certain types of illegal content is not permitted.

Following the success of deep learning-based visual classification research, there has been a surge in research for video annotation [90, 93, 194, 126]. Internet companies such as Facebook [137] and Twitter [25] are also developing products for analyzing the videos on their platforms. Recently, Google has introduced the Cloud Video Intelligence API for video analysis [109]. A demonstration website [88] has been launched which allows anyone to test the API with videos stored in Google Cloud Storage. The API then quickly returns the *video labels* (key objects within the video), *shot changes* (scene changes within the video) and *shot labels* (description of every shot). By detecting the video shots and labeling them, the API can make videos searchable just as text documents. Thus, users can search for a particular event and get related videos along with the exact timings of the events within the videos.

In this chapter, we examine the robustness of video analysis algorithms. Specifically, we propose targeted attacks against two fundamental classes of video analysis algorithms, namely video

classification and shot detection. We then apply the attacks on Google Cloud Video Intelligence API. [1] We show that an adversary can subtly manipulate a video in such a way that a human observer would perceive the content of the original video, but the API will return the adversary's desired outputs. Such vulnerability will seriously undermine the performance of video analysis systems in real-world applications. For example, a search engine may wrongly suggest manipulated videos to users, a video filtering system can be bypassed by slightly modifying a video which has illegal content, or a video search algorithm may miss the important events in surveillance videos. Our findings further indicate the importance of designing ML systems to maintain their desired functionality in adversarial environments.

Our contributions are summarized in the following:

- We develop a model for state-of-the-art video classification and shot detection algorithms. We then formulate the adversary's objective function for mounting targeted attacks on black-box video analysis systems and discuss different approaches for video modification.

- We propose different methods for deceiving video classification and shot detection algorithms and demonstrate the effectiveness of our attacks on Google Cloud Video Intelligence API. In our experiments, we queried and tested the API with different videos, including our recorded and synthetically generated videos, videos downloaded from web, and the sample videos provided by API website. Selected videos vary in content, length, frame rate, quality and compression format.

- Through experiments, we show that the API's algorithm for generating video and shot labels processes only the first frame of every second of the video. Therefore, by inserting an image at the rate of one frame per second into the video, the API will *only* output video and shot labels that are related to the inserted image.

- We also show that the pattern of shot changes returned by the API can be mostly recovered by finding the peaks in the vector of histogram changes of consecutive frames. Based on our

[1] The experiments are performed on the interface of Google Cloud Video Intelligence API's website in April 2017.

equivalent model, we develop a method for slightly modifying the video frames, in order to deceive the API into generating our desired pattern of shot changes.

- We propose countermeasures against our attacks. We show that introducing randomness to video analysis algorithms improves their robustness, while maintaining the performance.

2.2 Preliminaries

In this section, we first provide a background on digital video data and then describe the Google Cloud Video Intelligence API.

2.2.1 Video Data

A digital video consists of audio data and a series of frames (still images) that are displayed in rapid succession to create the impression of movement. The frequency (rate) at which consecutive frames are displayed is called Frame Rate and is expressed in frames per second (fps). Modern video formats utilize a variety of frame rates. The universally accepted film frame rate is 24 fps. Some standards support 25 fps and some high definition cameras can record at 30, 50 or 60 fps [118].

Digital videos require a large amount of storage space and transmission bandwidth. To reduce the amount of data, video data are typically compressed using a lossy compression algorithm. Video compression algorithms usually reduce video data rates in two ways: 1) Spatial (intraframe) compression: Compressing individual frames, and 2) Temporal (interframe) compression: Compressing groups of frames together by eliminating redundant visual data across multiple consecutive frames, i.e., storing only what has changed from one frame to the next [118]. We refer to compression and decompression algorithms as *encoder* and *decoder*, respectively, and call the concatenation of encoder and decoder as *codec*.

2.2.2 Google Cloud Video Intelligence API

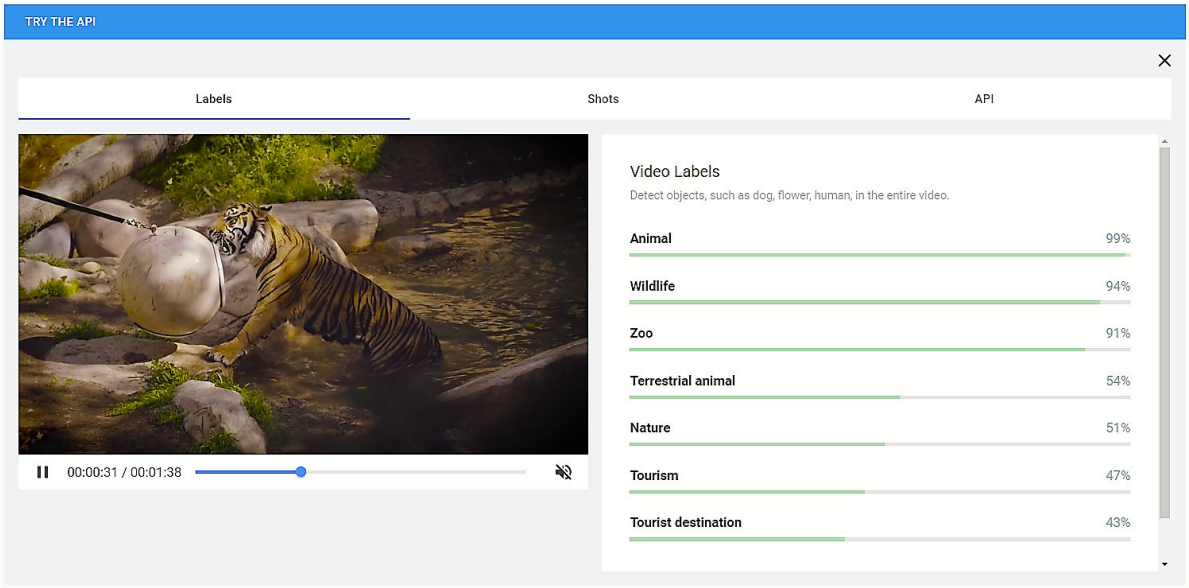
Success of ML algorithms has led to an explosion in demand. To further broaden and simplify the use of ML algorithms, cloud-based service providers such as Amazon, Google, Microsoft, BigML,

and others have developed ML-as-a-service tools. Thus, users and companies can readily benefit from ML applications without having to train or host their own models.

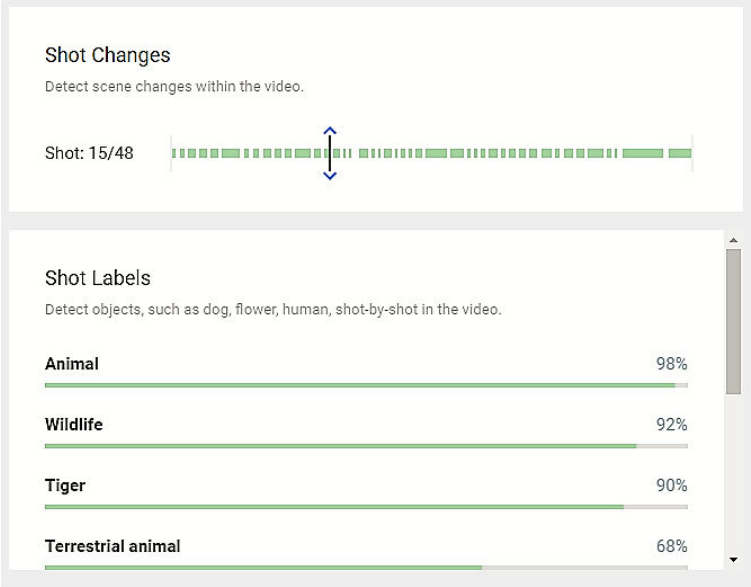
Google has recently launched the Cloud Video Intelligence API for video analysis [88]. The API is designed to help better understand the overall content of the video, while providing temporal information on when each entity was present within the video. Therefore, it enables searching a video catalog the same way as text documents [109]. The API is made available to developers to deploy it in applications that require video searching, summarization or recommendation [109]. The API is said to use deep-learning models, built using frameworks like TensorFlow and applied on large-scale media platforms like YouTube [109].

A demonstration website has been also launched which allows anyone to select a video stored in Google Cloud Storage for annotation [88]. The API then provides the video labels (objects in the entire video), shot changes (scene changes within the video) and shot labels (description of the video events over time). As an illustration, Figure 2.1 shows the screenshots of API's outputs for the video "Animals.mp4," provided by the API website. Through experiments with different videos, we verified that the API's outputs are indeed highly accurate.

In our experiments, we queried and tested the API with different videos, including our recorded and synthetically generated videos, videos downloaded from web, and the sample videos provided by API website. Selected videos vary in content, length, frame rate, quality and compression format. For mounting the attacks, we modify videos, store them on Google cloud storage and then use them as inputs to the API. In one of our attacks, we insert images into the videos. Figure 2.2 shows some of the sample images that were used in our experiments. If not said otherwise, the manipulated videos are generated with frame rate of 25 fps, where each frame is a color image of size 300×500.

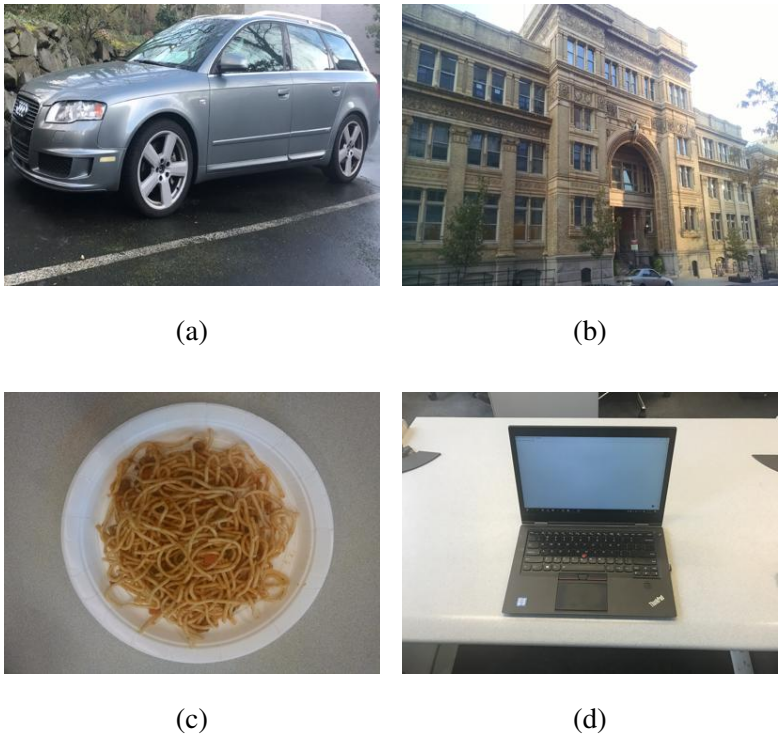


(a) Screenshot of the video labels.



(b) Screenshot of the shot changes and shot labels. The shot changes are the white bars appeared within the green strip. Shot labels are generated for each shot.

Figure 2.1: Screenshots of the API's outputs for "Animals.mp4" video, provided by API website.



(a) (b)

(c) (d)

Figure 2.2: Sample images of (a) a car, (b) a building, (c) a food plate, and (d) a laptop, used in experiments.

2.3 Video Analysis Methods

In this section, we review the current methods for video classification and shot detection and provide a system model from the adversary's perspective for each task.

2.3.1 Video Classification Methods

With the proliferation of mobile devices, video is becoming a new way of communication between Internet users. Accelerated by the increase in communication speed and storage space, video data has been generated, published and spread explosively. This has encouraged the development of advanced techniques for various applications, including video search and summarization [187].

However, due to the temporal aspect of video data and the difficulty of collecting sufficient well-tagged training samples [200, 180], using machine learning for videos classification has remained a challenging task. Automatic video annotation would be a breakthrough technology, enabling a broad range of applications. It can help media companies with quickly summarizing and organizing large video catalogs. It can also improve video recommendations, as it enables the search engines to consider video content, beyond the video metadata. Another use case would be in video surveillance, where many hours of videos must be searched for a specific event. Moreover, Internet platforms, such as YouTube and Facebook, would be able to automatically identify and remove videos with illegal content.

Following the successful use of deep neural networks for image classification [99, 155, 73], researchers have attempted to apply deep learning techniques to the video domain. In recent years, several new datasets have been published to help advancing the field [93, 32]. Most recently, a new large dataset [6] and a competition [41] is announced by Google to further promote the research in video classification methods. The current research for video analysis has mainly focused on selecting the network architectures and also the inputs to the networks. From network architecture perspective, convolutional neural networks [90, 194], recurrent neural networks [197], and temporal feature pooling [93, 197] are viewed as promising architectures to leverage spatial and temporal information in videos.

For network inputs, a simple approach is to treat video frames as still images and apply ML algorithms to recognize each frame and then average the predictions at the video level [197]. However, processing all video frames is computationally inefficient even for short video clips, since each video might contain thousands of frames. Moreover, consecutive video frames significantly overlap with each other in content and not all frames are consistent with the overall story of the video. Therefore, in order to learn a global description of the video while maintaining a low computational footprint, several papers proposed subsampling the video frames [197, 175, 163, 55, 187]. Hence, the focus of recent research is mostly on developing advanced ML techniques, such as deep neural networks, on subsampled frames. Figure 2.3a illustrates the model of video classification algorithms.

To compensate for the loss of motion information, [154] suggested incorporating explicit motion information in the form of optical flow images computed over adjacent frames. Optical flow encodes the pattern of apparent motion of objects and is computed from two adjacent frames sampled at higher rates than the video frames [198]. It has been however observed that the additional gain obtained by incorporating the optical flow images is very small [197]. Also, in [186], the authors proposed using audio spectrogram along with the visual data to improve the video understanding.

2.3.2 Shot Detection Methods

Shot detection is used to split up a video into basic temporal units called shots, which is a series of interrelated consecutive pictures taken contiguously by a single camera and representing a continuous action in time and space. Shot detection is widely used in software for post-production of videos. It is also a fundamental step of automatic video annotation and summarization, and enables efficient access to large video archives [71]. In film editing, two methods are usually used to juxtapose adjacent shots: 1) Abrupt Transitions: A sudden transition from one shot to another, i.e. the last frame of one shot is followed by the first frame of the next shot, and 2) Gradual Transitions: The two shots are combined so that one shot is gradually replaced by another [71].

Shot detection algorithms try to find the positions in the video, where one scene is replaced by another one with different visual content. Conventional approaches for shot detection usually involve the two steps of measuring the similarity of consecutive frames and then determining the shots' boundaries. Two simple methods for measuring the similarity of frames are sum of absolute values of pixel-wise difference of frames and the difference between the histograms of frames. Compared to computing the pixel-wise difference of frames, the histogram-based method is more robust to minor changes within the scene. After scoring the difference of consecutive frames, typically a fixed or adaptive thresholding is used for localizing the shot changes [71]. Once a shot is detected, it can be treated as a short clip and the same algorithm for video classification can be applied for annotation. Figure 2.3b illustrates the model of shot detection algorithms.

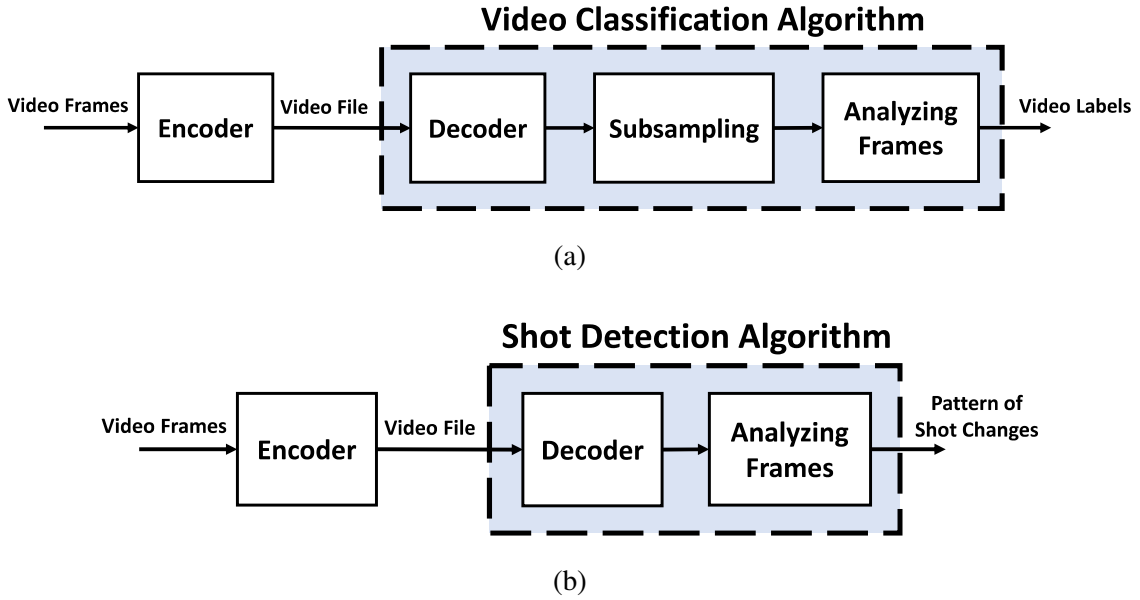


Figure 2.3: Models of video analysis algorithms. a) Video classification algorithm takes a video file, decodes it to obtain video frames, samples some of the frames and then processes subsampled frames to generate the video labels. b) Shot detection algorithm takes a video file, decodes it to obtain video frames and then processes the frames to generate the pattern of shot changes.

2.4 Threat Model

We assume that the video analysis system takes a video file and outputs video labels and the pattern of shot changes. We further assume that the system can only be accessed as a *black-box*. That is, the adversary possesses no knowledge about the training data or the specific ML models or video processing algorithms used, and can only query the system with any video of her choice and obtain the outputs. The goal of the adversary is to mount *targeted attacks*, i.e., given any video, the adversary intends to manipulate the video in such a way that the system generates only the adversary's desired outputs. The modification to the video should be very small, such that a human observer would perceive the content of the original, unmodified video. The proposed attacks are as follows:

- Targeted attack on video labeling algorithm: Deceiving the system to *only* output the adversary's desired video labels,

- Targeted attack on shot detection algorithm: Deceiving the system to output the adversary's desired pattern of shot changes.

The adversary's problem can be formulated as follows. Let F be the video analysis algorithm and $\{X_t\}_{t=1:T}$ be the sequence of video frames, where X_i is the i-th frame. For simplicity, we write $\{X_t\}_{t=1:T}$ as X. Let y^* denote the adversary's desired output. Adversary's goal is to cause the algorithm to yield her desired outputs by making minimal changes to the video. Therefore, the adversary's objective function is as follows:

$$\text{Find } X^* \text{ s.t. } F(X^*) = y^* \text{ and } \|X - X^*\| \leq \epsilon,$$

where X^* is the modified video. The term $\|X - X^*\|$ represents the amount of perturbation made to the video and ϵ is the maximum allowed perturbation, for which a human observer would still be able to perceive the content of the original video.

A video file can be modified in different ways. In the following, we review some methods of video modification and explain the maximum allowed perturbation corresponding to each method.

Inserting images within video frames at a low rate: The adversary can insert images of her choice within the video frames. However, the insertion rate must be low, so that the content of the original video would be perceivable. Moreover, it is preferable that the inserted image would be unnoticeable to a human observer. Speed of processing in the human visual system largely depends on the contrast of successive images shown [169, 135]. Empirical studies have shown that human visual system needs about 50 ms to process and comprehend every single image [44]. Therefore, in a video with frame rate of more than 20 fps, individual frames cannot be distinctly understood by humans. As a result, if an adversary inserts images at the rate of 1 fps within frames of such a video, the human observer would not perceive the content of inserted images.

Removing video frames at a low rate: Consecutive video frames are typically highly correlated, especially for videos with high frame rates, e.g., greater than 30 fps. Therefore, it is possible to

remove video frames at a low rate (and replace them with adjacent frames) without significantly reducing the video quality. It is known that with frame rates greater than 20 fps, human visual system is fooled that the sequence of frames represents an animated scene, rather than being a succession of individual images [33]. Therefore, to preserve the video smoothness, removing video frames should not cause the rate of distinct video frames to drop below 20 fps.

Slightly modifying individual frames: Instead of adding or removing frames, the adversary can modify individual frames. The modification to each frame can be quantified by the PSNR value of the modified frame with respect to the original frame. For images x and x^* of size $d_1 \times d_2 \times 3$, PSNR value is computed as follows [182]:

$$PSNR = 10 \cdot \log_{10} \left(\frac{255^2}{\frac{1}{3\,d_1\,d_2} \sum_{i,j,k} (x_{i,j,k} - x^*_{i,j,k})^2} \right),$$

where (i, j) is the image coordinate and $k \in \{1, 2, 3\}$ denotes the coordinate in color space. PSNR value is measured in dB.

Due to the inherent low-pass filtering characteristic of humans visual system, humans are capable of perceiving images slightly corrupted by noise [17], where acceptable PSNR values for noisy images are usually considered to be more than 20 dB [7]. However, similar to image insertion, an adversary is allowed to add high-density noise to video frames, but at a low rate, e.g., one frame per second. In contrast, the succession of noisy images corrupted even by low-density noise is very disturbing, especially when the video is passed through a lossy video codec. The reason is that video compression algorithms rely on similarity of consecutive frames to compress the video. A random uncorrelated noise added to frames increases the difference of consecutive frames. As a result, the compression algorithm reduces the quality of individual frames to achieve the same compression ratio.

In our attacks, we modify videos by low-rate image insertion within the frames or slightly perturbing frames, e.g., by adding low-density noise to frames at a low rate or smoothing some of the frames. The modifications are done in such a way that the content of the original video would be perceivable. In the following, we describe the attacks.

2.5 Targeted Attack on Video Classification Algorithms

In this section, we first present different approaches of attacking video classification algorithms and then describe our attack on Google Cloud Video Intelligence API.

2.5.1 Attack Approaches

For attacking a video classification algorithm, the adversary slightly manipulates the video. As shown in Figure 2.3a, the manipulated video passes through three blocks of codec, subsampling and frame processing. In the following, we discuss adversary's considerations for modifying videos.

For deceiving the system, the adversary can target the frame analysis algorithm. In this case, the adversary slightly modifies video frames such that, after subsampling, a human observer would classify the subsampled video as its original label, but the frame analysis algorithm yields adversary's desired labels. This attack type can be thought to be a form of generating adversarial examples [166] for frame analysis algorithm. However, as stated in Section 2.2.1, video files are typically compressed using lossy compression algorithms. The compression algorithm is likely to filter out the adversarial noise and thus render this approach ineffective. Therefore, the modification to video should be done in such a way that it would "survive" the codec operation. Moreover, even without the codec operation, this attack approach is challenging due to the adversary's black-box access to the model.

Another approach is to modify the video such that, after subsampling, a human observer would classify the subsampled video as adversary's desired label. This approach is preferable, since it is effective, regardless of frame analysis algorithm. For mounting this attack, the adversary inserts images with her desired content within the video frames, in locations where the subsampling function will sample. The subsampling locations need to be determined by querying the system with several specifically chosen videos. The success of inferring the subsampling function and the required number of queries depend on the function randomness and how it is related to video characteristics, such as video length, frame rate, codec, etc.

We demonstrate the effectiveness of this approach by attacking the video and shot labeling

algorithms of Google Cloud Video Intelligence API. Through experiments, we first infer the API's algorithm for sampling the video frames and then mount the image insertion attack. While the proposed attack is demonstrated on the Google API, this approach is applicable against any video classification system that is based on deterministic subsampling, e.g., [197, 175, 163, 55, 187] to name a few.

2.5.2 Inferring API's Algorithm for Sampling Video Frames

In order to infer the API's sampling algorithm, we first need to determine whether it uses deterministic or stochastic algorithms. For this, we queried the API with different videos. We found that when testing the API with the same video several times, it generates exactly the same outputs, in terms of the video and shot labels, the corresponding confidence scores, and the pattern of shot changes. Our observations imply that *the API's algorithms for processing the input videos are deterministic.* Knowing that API's algorithms are deterministic, we designed an experiment for testing the API with videos which are different from each other on certain frames. We first generated a one-minute long video with all frames being the same image of a "building." We then modified the video in two different ways: 1) replacing the *first* frame of every second with an image of a "car," and 2) replacing the *second* frame of every second with the same image of the "car."

Table 2.1 provides the set of labels for generated videos. As expected, all labels of the original video are related to "building," since it only contains images of a building. However, all labels of the first modified video are related to "car." In contrast, all labels of the second modified video are related to "building." We also tested the API with modified videos for which the i-th frame, $3 \leq i \leq f$, of every second is replaced by image of the "car," where f is the number of frames per second. We observed that, for all of them, the API outputs video labels only related to "building." We selected videos with different characteristics and repeated the above experiments. We found that, regardless of the video content, length, frame rate, quality or compression format, *the API's algorithm for generating the video labels can be reduced to a function that gets as input only the first frame of every second of the video.* This suggests that the subsampling function samples video frames at the rate of 1 fps. Specifically, it samples the first frame of every second of the video.

Table 2.1: Video labels generated by Google Cloud Video Intelligence API for different videos. The results suggest that the API's algorithm for generating video labels processes only the first frame of every second of the video.

Video	**Video labels returned by API**
Original Video: A one-minute long video with all frames being the same image of a "building."	Building, Classical architecture, Neighbourhood, Facade, Plaza, Property, Apartment, Architecture, Mansion, Courthouse
Modified Video 1: Replacing the *first* frame of every second of the original video with an image of a "car."	Audi, Vehicle, Car, Motor vehicle, Land vehicle, Luxury vehicle, Sedan, Audi A6, Wheel, Mid-size car, Audi A4, Bumper, Family car, Audi RS 4
Modified Video 2: Replacing the *second* frame of every second of the original video with an image of a "car."	Building, Classical architecture, Neighbourhood, Facade, Plaza, Property, Apartment, Courthouse, Mansion, Architecture

Then, we examined how the API samples video frames for generating the *shot* labels. Similar to the case of video labels, we tested the API with a one-minute long video with all frames being the same image of a "building," and with modified videos, for which the i-th frame, $1 \leq i \leq f$, of every second is replaced by image of the "car." For the original video, there is only one shot and the shot labels are related to "building." As expected, the API generated 60 shots (one shot per second) for each of the modified videos, because each replaced frame is different from the adjacent frames and thus triggers a shot. For the first modified video, where image of the "car" is replaced in the first frame of every second, all shot labels were related to "car" for all 60 shots. In contrast, for $2 \leq i \leq f$, all the shot labels were related to "building." Our observations suggest that, similar to video labels, *the API generates shot labels by processing only the first frame of every second of the video.* Figure 2.4 illustrates the API's sampling algorithm.

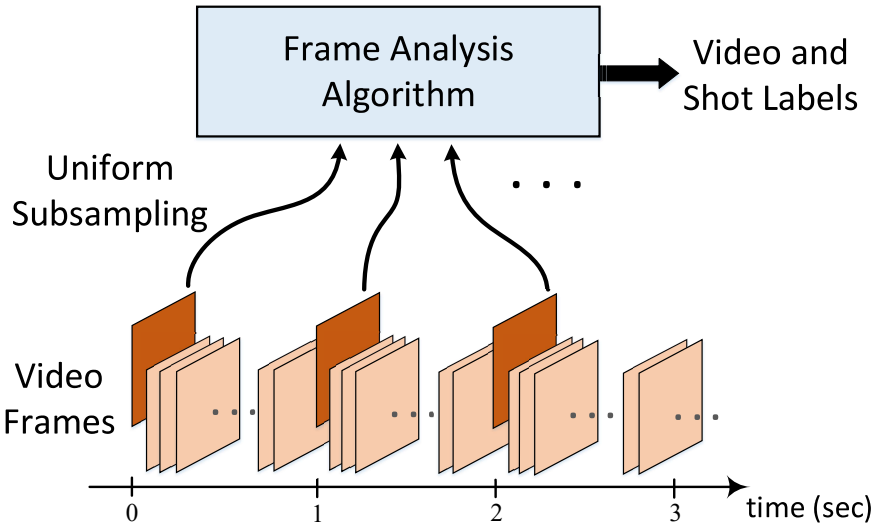


Figure 2.4: The video and shot labeling algorithm of Google Cloud Video Intelligence API subsamples the video frames uniformly and deterministically. Specifically, it samples the first frame of every second of the video.

2.5.3 Image Insertion Attack on API

Now, we describe the image insertion attack for changing the video and shot labels returned by the Google Cloud Video Intelligence API. The goal is to mount a targeted attack on the video classification algorithm. That is, given any video, the adversary intends to slightly manipulate the video, in such a way that a human observer would perceive the content of the original video, but the API outputs the adversary's desired video and shot labels. The attack procedure is as follows. The adversary is given a video and the target video label. She selects an image which represents the desired label, and inserts it into the first frames of every second of the video.

For validating the attack on the API, we generated manipulated videos, stored them on Google cloud storage and used them as inputs to the API. Table 2.2 provides the API's output labels for the manipulated videos of three videos "Animals.mp4," "GoogleFiber.mp4" and "JaneGoodall.mp4," provided by API website (the table shows only the label with highest confidence score). As can be seen, regardless of the video content, the API returns a video label, with a very high confidence

Table 2.2: Results of Image Insertion Attack on Google Cloud Video Intelligence API. Sample videos are provided by API website [88]. Images are inserted in the first frame of every second of the video. The table only shows the video label with the highest confidence returned by API.

Video Name	**Inserted Image**	**Video Label Returned by API (Confidence Score)**
"Animals"	"Car"	Audi (98%)
	"Building"	Building (97%)
	"Food Plate"	Pasta (99%)
	"Laptop"	Laptop (98%)
"GoogleFiber"	"Car"	Audi (98%)
	"Building"	Classical architecture (95%)
	"Food Plate"	Noodle (99%)
	"Laptop"	Laptop (98%)
"JaneGoodall"	"Car"	Audi (98%)
	"Building"	Classical architecture (95%)
	"Food Plate"	Pasta (99%)
	"Laptop"	Laptop (98%)

score, that exactly matches the corresponding inserted images. We tested the API with several videos with different characteristics and verified that the attack is consistently successful. We also mounted the image insertion attack for changing the shot labels returned by the API and verified that, by inserting an image in the first frame of every second of the video, *all* the shot labels returned by the API are related to the inserted image.

2.5.4 Improving Attack on API

In proposed attack, the image insertion rate is very low. For example, for a typical frame rate of 25, we insert only one image per 25 video frames, resulting in an image insertion rate of 0.04. Therefore, the modified video would contain the same content as the original video. Nevertheless, the attack can be further improved using the following methods.

Lower insertion rate. We found that the attack still succeeds if we insert the image at a lower rate of once every two seconds, i.e., by inserting the adversary's image into the video once every two seconds, the API outputs video labels that are only related to the inserted image. However, by further lowering the insertion rate, for some of the videos, the API's generated video labels were related to both the inserted image and also the content of the original video.

Averaging with the original frame. Instead of replacing the video frame, the adversary can *superpose* the image with the frame. Let x_V be a video frame and x_A be adversary's image. We generate $x'_V = \alpha x_V + (1 - \alpha)x_A$ as the new frame and replace it for x_V. We found that by setting $\alpha = 0.75$, we can deceive the API to only output the video labels that are related to the inserted image. As an illustration, Figure 2.5a provides an example of a video frame averaged with the image of a car, presented in Figure 2.2. The weighted averaging of the adversary's image with the video frame significantly reduces the visual effect of the inserted image in the video, yet the API annotates the video as if it only contains the adversary's image. [2]

Misclassification attack by adding noise. In misclassification attack, the adversary's goal is to cause the API to output *different* video labels then the original labels. For mounting the misclassification attack, it is enough to just add noise to video frames. We performed the experiments with

[2]We also tested the API with videos comprising of adversarial examples [166] and observed that the API is robust to this attack.

(a) (b)

Figure 2.5: Illustration of superposed and noisy images used for replacing original video frames. a) The video frame is averaged with the image of a car, presented in Figure 2.2, where the weight of video frame is 0.75 and the weight of the car image is 0.25. b) The video frame corrupted by 5% impulse noise. In both cases, there is only a small modification to the video frame.

impulse noise. Impulse noise, also known as salt-and-pepper noise, is commonly modeled by [78]:

$$\tilde{x}_{i,j,k} = \begin{cases} 0 & \text{with probability } \frac{p}{2} \\ x_{i,j,k} & \text{with probability } 1-p \\ 255 & \text{with probability } \frac{p}{2} \end{cases}$$

where x, $\tilde{x}$ and p are the original and noisy frames and the noise density, respectively. This model implies that each pixel is randomly modified to one of the two fixed extreme values, 0 or 255, with the same probability. Through experiments, we found that by adding only 5% impulse noise to the first frame of every second of the video, we can deceive the API to generate entirely irrelevant video labels. Figure 2.5b provides an example of a video frame corrupted by 5% impulse noise. As can be seen, the noise effect is hardly noticeable. We performed the experiments with other image noise types as well and found that impulse noise is the most effective one, i.e., it can cause the API to misclassify the input video by introducing very small perturbation to the video frames.

2.5.5 *Applications of Image Insertion Attack*

We showed that by inserting an image at the rate of 1 fps into the video, the video labels and all the shot labels generated by the API are about the inserted image. Instead of periodically inserting the same image, an adversary can replace the first frame of every second a video with frames from the corresponding positions of another video. The API then generates the same set of video labels for both videos, although they only have one frame in common in every second. In other words, the adversary can replace one frame per second of a video with the corresponding frame of another video and the API would not be able to distinguish the two videos.

Such vulnerability of video classification systems seriously undermines their applicability in real-world applications. For example, it is possible to poison a video catalog by slightly manipulating the videos. Also, one can upload a manipulated video that contains adversary's images related to a specific event, and a search engine wrongly suggests it to users who asked for videos about the event. Moreover, an adversary can bypass a video filtering system by inserting a benign image into a video with illegal content. Therefore, it is important to design video analysis algorithms to be robust and perform well in presence of an adversary. In the following, we provide a countermeasure against the image insertion attack.

2.5.6 *Countermeasure Against Image Insertion Attack*

One approach to mitigate the effect of the attack is to introduce randomness into the algorithms. In essence, inferring an algorithm that generates different outputs for the same input would be substantially more challenging, especially in the black-box setting. For video classification algorithms that subsample the video frames, an obvious countermeasure to the image insertion attack is to sample the frames randomly, while keeping the sampling rate the same.

Assume that the algorithm randomly samples the video frames at the rate of 1 fps and the adversary replaces K video frames per second with her chosen image. Let the frame rate be r fps. Thus, a fraction of $\frac{K}{r}$ sampled images are the inserted images by the adversary, which is equal to 4% for $r = 25$ and $K = 1$. Suppose further that the system chooses L samples out of the r

frames, uniformly at random and without replacement for analysis. We compute the probability that at least one of the adversary's images is chosen by the system, as well as the expected number of sampled adversary's images.

Probability of sampling an adversary's image. Let Y denote a random variable representing the number of images that are sampled from the adversary's images. We are interested in computing $Pr(Y \geq 1)$, or equivalently $1 - Pr(Y = 0)$. Let χ_i denote the event that the i-th sample is from original video frames. We have

$$\begin{aligned} 1 - Pr(Y = 0) &= 1 - Pr(\chi_1 \cap \cdots \cap \chi_L) \\ &= 1 - \prod_{i=1}^{L} Pr(\chi_i | \chi_1, \ldots, \chi_{i-1}) \qquad (2.1) \\ &= 1 - \prod_{i=1}^{L} \left(\frac{r - K - (i-1)}{r - (i-1)} \right) \qquad (2.2) \end{aligned}$$

where (2.1) follows from the conditioning on the χ_i's and (2.2) holds since the probability of choosing a valid sample, given that the first $(i-1)$ samples are valid, is equal to the number of remaining valid samples $(r - L - (i-1))$ divided by the total number of remaining samples $r - (i-1)$. For $K, L \ll r$, this can be approximated by $Pr(Y \geq 1) \approx 1 - e^{\frac{-KL}{r}}$, which can be further approximated by $Pr(Y \geq 1) \approx \frac{KL}{r}$, when $KL \ll r$. Also, for $L = 1$, we have $Pr(Y = 1) = \frac{K}{r}$, meaning that the chance that adversary's image will be sampled increases linearly by the number of replaced images.

Expected number of selected images from adversary. Let $T(L, K, r) \triangleq \mathbf{E}(Y)$ denote the expected number of adversary's images that are chosen, and let χ_1 be defined as above. Let $\overline{\chi}_1$ denote the complement of χ_1. We have that

$$\begin{aligned} T(L, K, r) &= \mathbf{E}(X|\chi_1)Pr(\chi_1) + \mathbf{E}(X|\overline{\chi}_1)Pr(\overline{\chi}_1) \\ &= T(L-1, K, r-1)(1 - \frac{K}{r}) + T(L-1, K-1, r-1)\frac{K}{r}. \end{aligned}$$

where $T(L, K, K) = \min\{L, K\}$. Hence, the expected number of adversarial samples can be computed recursively. For $K, L \ll r$, we have $\mathbf{E}(Y) \approx \frac{KL}{r}$. With larger L, the number of frames

sampled from adversary's image increases, however the ratio of adversarial frames to all sampled frames remains approximately the same.

Analyzing the exact effect of sampling a small fraction of frames from adversary's images would require knowledge about the frame analysis algorithm and can be an interesting direction for future works. However, the process of the adversary uniformly replacing the video frames and the system randomly subsampling them is equivalent to the process of the adversary randomly replacing and the system uniformly sampling. Therefore, we can evaluate the performance of randomly subsampling the video frames using the current system, by randomly replacing the video frames with adversary's image. Through experiments, we found that even by randomly replacing two video frames every second, the API outputs mostly the same video labels as for the original video and none of the labels are related to the inserted image. The results imply that the video classification algorithm can benefit from randomly subsampling the video frames, without losing the performance.

2.6 Targeted Attack on Shot Detection Algorithms

In this section, we first present our approach of attacking shot detection algorithms and then describe our attack on Google Cloud Video Intelligence API.

2.6.1 Attack Approach

For attacking a shot detection algorithm, the adversary needs to modify the video such that the algorithm detects a shot only at the adversary's desired locations. Since we assume that the adversary only has a black-box access to the system, for every video, she needs to query the system multiple times, each time with the video modified differently. This approach may require a large number of queries to the system, which can make the attack ineffective in real-world applications.

However, we noticed that different shot detection algorithms usually yield similar pattern of shot changes. Therefore, the adversary can rely on *transferability* of manipulated videos, i.e., a specific modification to a video causes different algorithms to detect the same pattern of shot

changes. Hence, the adversary can first design her own shot detection algorithm and try to deceive it by manipulating the video. The adversary then uses the manipulated video as input to the target algorithm. We demonstrate the effectiveness of our approach by attacking the shot detection algorithm of Google Cloud Video Intelligence API. We first develop a histogram-based method for shot detection and then propose an attack for deceiving the algorithm to output our desired pattern of shot changes. We show that the manipulated videos can successfully deceive the API.

2.6.2 Histogram-Based Algorithm for Shot Detection

We first develop a method for shot detection and then compare the results with the ones generated by the API. Since shot changes are usually detectable on the gray-scale video, for simplicity, we transform the frames to gray-scale images. We adopt a *histogram-based algorithm*, where histogram of a frame is a vector that, for each gray-value, contains the number of pixels with that value. In this method, we first compute the difference between histograms of consecutive frames. If a frame has a large histogram difference with its previous frame, we declare that a shot is initiated at that frame. For measuring the difference of two histograms, we compute the L_1 norm of their difference, i.e., the sum of the absolute values of histogram changes.

Our method is described more formally in the following. Let $\{X_t\}_{t=1:T}$ be the set of video frames and $\{H_t\}_{t=1:T}$ be the set of histograms of gray-scale frames. We denote by $\delta H_t = H_t - H_{t-1}$ the element-wise difference of the histograms, which represents the change in statistics of the consecutive frames. Let v be a vector of histogram changes of the video, obtained as the L_1 norm of the difference of histograms, i.e., $v_t = \|\delta H_t\|_1, 2 \leq t \leq T$. We locate the shot changes by finding the local maxima of the vector v, for which the amplitude of the histogram change is more than a threshold. The threshold is set to be greater than the typical histogram change of consecutive frames within a shot.

Through experiments with different videos, we explored whether the output of our method can resemble the pattern of shot changes returned by the API. Figure 2.6 provides the results for three sample videos, "Animals.mp4," "GoogleFiber.mp4" and "JaneGoodall.mp4." For each video, the figure shows the vector v and the corresponding shot changes generated by the API (green pattern)

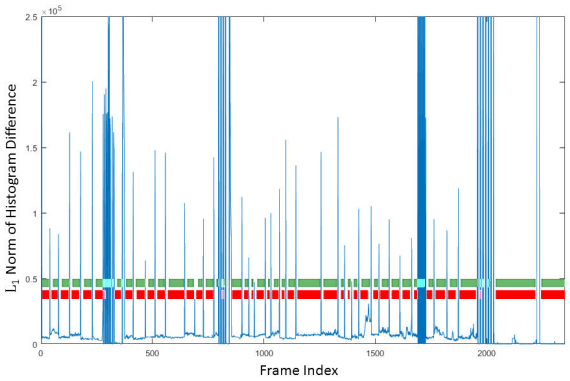
and our histogram-based algorithm (red pattern). For our method, we set the threshold of 3×10^4 for declaring the shot change, which is determined by manually checking the histogram plots of several videos. We also discard very small shots (less than 10 frames). As can be seen, our method declares a shot change when there is a large local maximum in the vector of histogram changes. Also, the shot changes generated by our algorithm are mostly aligned with the API's generated pattern of shot changes. The results imply that *the API's algorithm of detecting shot changes can be reduced to an algorithm of finding the large peaks in the pattern of histogram changes of consecutive frames.*

2.6.3 Shot Altering Attacks on API

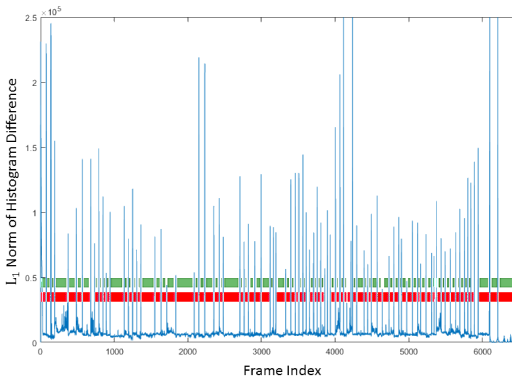
In this section, we present shot removal and shot generation attacks with the goal of deceiving the shot detection algorithm to miss real shot transitions and detect fake shots, respectively. Using the combination of shot removal and shot generation attacks, an adversary can deceive the API to output any desired pattern of shot changes for any video.

Shot Removal Attack. To force the API to miss a shot change, we need to modify an abrupt transition in the video to a gradual transition. For this, we apply a local low-pass filter on the frames located on the shot boundaries. We iteratively smooth those frames so that their histogram difference becomes less than the pre-specified threshold of our shot detection algorithm. Once we can deceive our histogram-based algorithm, we test the smoothed video on the API.

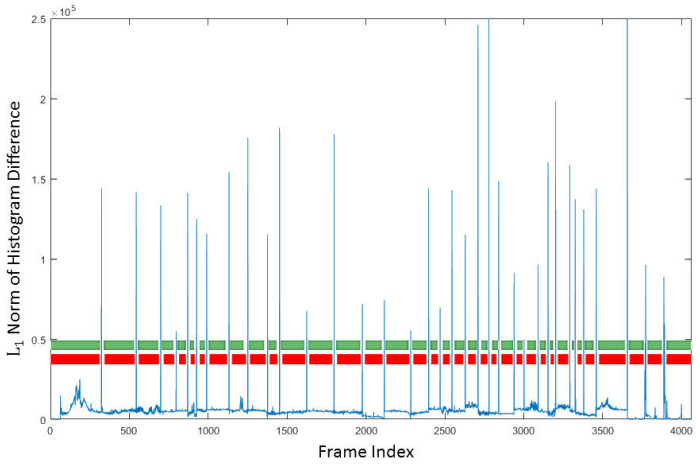
The smoothing attack is described more formally in the following. To remove the shot transition between the frames X_t and X_{t+1}, we first smooth them as follows: $X'_t = \frac{1}{h}\sum_{i=-h}^{h} X_{t+i}$ and $X'_{t+1} = \frac{1}{h}\sum_{i=-h}^{h} X_{t+1+i}$, where $h = 1$. We then compute v_t, the L_1 norm of the difference of histograms of X'_t and X'_{t+1}. If v_t is greater than the threshold, we increase h by one and repeat the process. We continue smoothing the frames, until the histogram change of the two frames becomes less than the threshold.



(a) “Animals.mp4” video

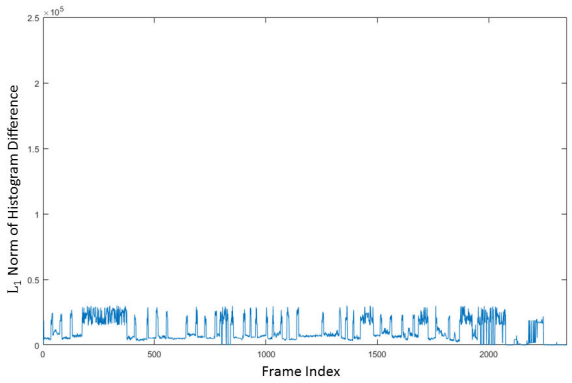


(b) “GoogleFiber.mp4” video

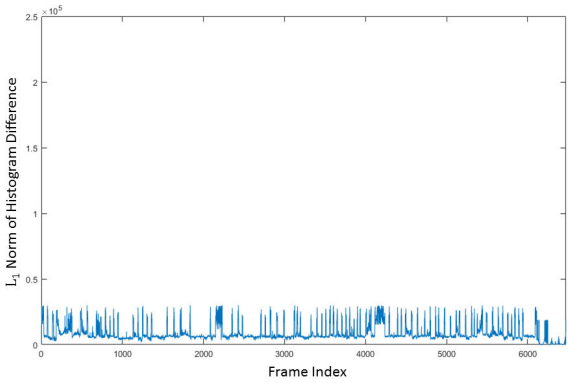


(c) “JaneGoodall.mp4” video

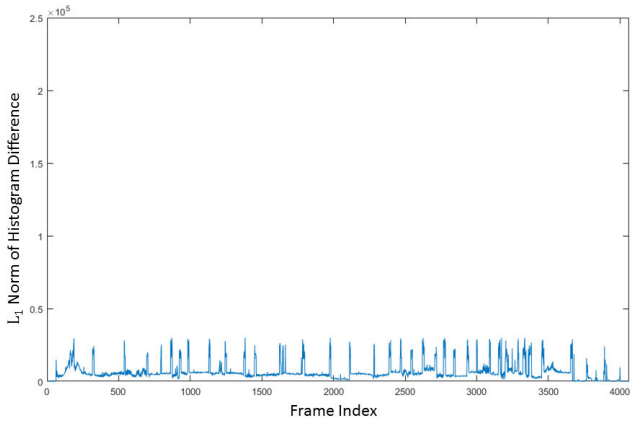
Figure 2.6: The histogram changes of consecutive video frames (blue curve), the shot changes generated by Google Cloud Video Intelligence API (green pattern) and our histogram-based method (red pattern). The shot changes returned by our method are located at large local maxima of the vector of histogram changes and are mostly aligned with API’s output.



(a) "Animals.mp4" video



(b) "GoogleFiber.mp4" video



(c) "JaneGoodall.mp4" video

Figure 2.7: The histogram changes of consecutive frames for the smoothed videos. For the sake of comparison, we keep the y-axis of subfigures the same as the ones for Figure 2.6. All the peaks of the histogram changes are now smaller than our threshold of detecting a shot (3×10^4). Therefore, our histogram-based method does not detect any shot change. We verified that the Google Cloud Video Intelligence API also generates only one shot for the entire smoothed video.

Through experiments with different videos, we examined whether the smoothed videos *transfer* to the API, i.e., the API also fails to detect the shot changes. We present the results on three sample videos, "Animals.mp4," "GoogleFiber.mp4" and "JaneGoodall.mp4," where the attack goal is to

remove *all* shot changes, i.e., we want to deceive the API to believe the entire video is composed of only one shot. Figure 2.7 shows the histogram changes of the smoothed videos. For the sake of comparison, we keep the y-axis of subfigures the same as the ones for Figure 2.6. As can be seen, all the peaks now are smaller than the threshold of 3×10^4. Hence, our histogram-based method does not detect any shot change. We verified that the API also generates only one shot for the smoothed videos. Since our attack only smooths few frames on the shot boundaries and keeps the rest of the video the same, the perturbation to the video is hardly noticeable. Figure 2.8 illustrates an example of two frames on a shot boundary and their corresponding smooth frames. Unlike the original frames, the API does not detect any shot change between the smoothed frames.

Shot Generation Attack. The goal of shot generation attack is to slightly modify the video such that the API wrongly detects a shot change at a specific frame. As explained in Section 2.5, inserting an image in between the video frames causes the API to declare a shot change at that frame. However, we do not need to insert an entirely different image within the frames to fake a shot change. We found that even by slightly increasing or decreasing all the pixel values of a frame, the API is deceived to detect a shot change at that frame.

As an example, we generated a one minute long video, where all frames were the same image of a building. As expected, the API outputs only one shot for this video. We then selected *one* of the video frames and increased all the pixel values by 10. When testing the API with the modified video, it detects a shot change at that frame. Figure 2.9 shows the original and modified frames. Notice that the frames are hardly distinguishable.

2.6.4 Applications of Shot Altering Attacks

Shot detection is important for temporal segmentation of the video and enables applications such as video searching and summarization. Fragility of the shot detection algorithm seriously undermines the benefits of the video analysis system. We showed that by slightly manipulating the video, an adversary can cause the API to generate her desired pattern of shot change. Shot altering attacks disrupt the functionality of the API by preventing it from correctly indexing and annotating the

(a) Last frame of a shot

(b) First frame of next shot

(c) Smoothed frame of (a)

(d) Smoothed frame of (b)

Figure 2.8: An Illustration of shot removal attack. Original frames represent an abrupt shot transition, while smoothed frames form a gradual shot transition.

video content. Essentially, without correctly detecting the individual shots, the system cannot reliably produce the story of the video by cascading the proper shot labels. As an example of the attack, an adversary can prevent the API from finding important events in video surveillance data. In the following, we provide a countermeasure against shot altering attacks.

2.6.5 Countermeasure Against Shot Altering Attacks

Several papers have proposed methods for gradual shot detection in videos [71, 157, 9]. We are however interested in robust techniques for *adversarial* inputs, i.e., videos that are maliciously modified to deceive the algorithm. Similar to the image insertion attack, the system robustness can

(a) (b)

Figure 2.9: An Illustration of shot generation attack. We generate a video, where all frames are an image of a "building" (subfigure (a)). We then select *one* of the video frames and increase all pixel values by 10 (subfigure (b)). When testing the API with the modified video, it detects a shot change at the location of the modified frame.

be improved by introducing randomness to the algorithm, e.g., random subsampling of the video frames and measuring the difference of sampled frames as an indicator for whether the shot has changed. However, compared to the task of video labeling, the system needs to subsample video frames at higher rates, e.g., 10 fps, to be able to detect small shots. Also, since the scene may change within the shot, samples within the shot may have large differences. Therefore, compared to consecutive frames, a higher threshold needs to be set when measuring the difference of subsampled frames. Moreover, the accuracy can be improved by running the randomized algorithm several times and detecting the shot changes by averaging the results.

There is a trade-off between accuracy and robustness of the shot detection algorithm. Sampling at lower rates increases the robustness to spurious changes within the shots, but causes the algorithm to potentially miss the small shots occurred between the two consecutive sampled frames. Moreover, with subsampling, the information of the exact moment of shot transition will be lost; but, since the intersampling times are less than one second, the estimated time suffices for most applications. Random subsampling, however, will not affect the shot labels; therefore, the overall description of the video remains mostly the same.

Chapter 3

ASSESSING ROBUSTNESS OF NEURAL NETWORKS AGAINST INPUT TRANSFORMATION

3.1 Introduction

It has been shown that state-of-the-art Convolutional Neural Networks (CNNs) trained with stochastic gradient descent (SGD) have enough capacity to "memorize" the training data, even when training images or labels are randomized [201]. Memorization is often associated with overfitting training data and, hence, large generalization error, defined as the difference between training error and test error. Yet, CNNs are known to be able to generalize well to images that are similarly distributed as training data.

For an image classifier, the space of possible inputs is much larger than the size of training data. Hence, models with identical performance on samples that are similarly distributed as training images can perform qualitatively differently on other distributions. Of course, most of possible inputs are random images that the model is not expected to recognize. Therefore, it remains to be determined what distributions we expect the model to generalize to and how different training methods affect generalization capability.

The distributions that we are interested in can be determined according to the human cognitive system. It is known that humans display "shape bias" when assigning a name to new items [102], i.e., they weight shape more heavily than other dimensions of perceptual similarity, such as size or texture. In [140], the authors studied shape bias property in CNNs by performing experiments on small datasets, in which images were arranged in triples of a probe, a shape-match and a color-match. It was shown that state-of-the-art one-shot learning models display shape bias as well, although the magnitude of bias varies greatly with different initializations and also fluctuates throughout training.

In order to conduct larger scale experiments, we propose to train and test CNNs with specifically designed images that represent the same object, but with different colors. One simple way of generating such images is applying image complementing transformation. The transformed image, called *negative image*, is an image with reversed brightness. Image complementing maintains the structure (e.g., edges) and semantics of images, as negative images are often easily recognizable by humans. Figure 3.1 shows representative samples of original and negative images of MNIST [106], notMNIST [31] and CIFAR10 [98] datasets.

We then examine the shape bias property of CNNs, by training and testing them on negative images. Through extensive systematic experiments on MNIST, notMNIST and CIFAR10 datasets, we assess the role of different factors, such as training data, network architecture, initialization method and regularization techniques, on the ability of model in generalizing the shapes. Our contributions are summarized in the following.

- We show that it is possible to design different CNNs that achieve similar accuracy on original images, but perform significantly differently on negative images. For instance, we designed three CNNs that achieve 0%, 28.3% and 99.5% accuracy on MNIST negative images, while yielding the same accuracy of 99.5% on original images. The results suggest that CNNs do not intrinsically require shape bias property to achieve high accuracy on test data.

- We then show that, when negative images of some classes are included in training data, CNNs can correctly recognize negative images of other classes. This is however true, only if the model is trained with batch normalization. For instance, we trained the CNN with all MNIST regular images and also negative images of 9 classes, and tested it on negative images of the excluded class. The test accuracy is 0% and 97.7% respectively without and with batch normalization. Hence, CNNs can indeed learn to be invariant to color, when properly tuned.

- We also investigate the effect of augmenting training data with original and negative images of an unrelated dataset. We show that such data augmentation also significantly improves

generalizability to negative images. For instance, the CNN accuracy on negative images of MNIST training data increases from 29.1% to 99.8%, when MNIST training data is augmented with original and negative images of notMNIST dataset. The results suggest that augmenting training data with unrelated datasets that even have unrelated labels can help the model to learn better image representations.

- We examine the role of initialization and demonstrate that a different initialization can significantly affect the generalization. Specifically, a model that is initialized by training with original and negative images of a dataset shows shape bias on another dataset as well. Moreover, the shape bias property does not fade away after fine-tuning only with original images of the second dataset. For instance, when initialized by a CNN that is trained on original and negative images of notMNIST dataset, the accuracy on MNIST negative images increases from 29.1% to 95.4%.

- We finally show that CNNs do not learn the shape bias property from a dataset that lacks any structure, e.g., a dataset with random images. Moreover, a model with shape bias cannot generalize this property to random images, i.e., it does not perform similarly on random images and their negatives. The results imply that, although CNNs can fit any training data, they only learn and generalize the structures.

3.2 Related Works

Recently, there has been an interest in exploring various aspects of the generalization of CNNs by both theoretical and empirical analysis [201, 10, 123, 113, 189, 192]. It has been shown that CNNs trained with SGD are capable of memorizing the training data, contradicting their low generalization error [201]. In [10], however, the authors argued that networks trained with SGD learn structures before memorizing. This follows the works suggesting that SGD generally results in simpler models [156, 195].

Deep neural networks are known to be capable of approximating any measurable function given sufficient capacity [43, 76]. These works determine the set of hypotheses a model can express, but

Figure 3.1: Examples of regular images and their corresponding negative images of datasets MNIST, notMNIST and CIFAR10.

do not specify what hypothesis can be reached by training a network on a dataset using a particular method [124, 95]. We assess the capability of CNNs in generalizing shapes when color information is lost, a property known as shape bias [102, 140].

From the practical perspective, it is important to determine how CNNs perform on related distributions as of the training data. This problem has been studied in literature of transfer learning [127, 196] and domain adaptation [18, 143, 66, 47]. The goal is to use models and features learned on one dataset/domain for another dataset/domain with minimal fine-tuning [127]. We examine transferability of features from original to negative images. To the best of our knowledge, generalizing to negative images is not studied in transfer learning or domain adaptation literature.

3.3 Setup for Simulations

Experiments are performed on image datasets MNIST [106], CIFAR10 [98] and notMNIST [31]. MNIST is a dataset of handwritten digits. CIFAR10 dataset consists of natural color images in 10 classes. notMNIST is a dataset similar to MNIST with 10 classes of letters A-J taken from different fonts. notMNIST dataset contains more than $500,000$ images, from which we randomly select $60,000$ images for training and validation.

In [140], the authors studied the shape bias property in CNNs by performing experiments on images that were arranged in triples of a probe, a shape-match and a color-match. Images were chosen from a cognitive psychology probe data and also a real-world dataset consisting of 150 and 90 images, respectively. In order to conduct larger scale experiments, we propose to train and test

CNNs with negative images. A negative image is defined as the complement of the original image, in which light pixels appear dark and vice versa. Let X be an image and $X_{i,n} \in [0, 1]$ be the i-th pixel in the n-th color channel. The negative image is defined as X^*, where $X^*_{i,n} = 1 - X_{i,n}$.

Image complementing is a simple transformation that preserves the shape and semantics of images, as negative images are often easily recognizable by humans. In fact, it has been shown that human accuracy on negative images of German Traffic Sign Recognition Benchmark decreases only about 1% compared to original images [83]. Moreover, since in MNIST, notMNIST and CIFAR10 datasets, classes are very distinct, image complementing is unlikely to change the ground-truth label. In the rest of the chapter, we refer to original images as regular images.

We consider a small version of VGG-16 model [155], namely sVGG, with 6 convolutional layers followed by two fully connected layers. Similar to VGG-16, we use 3×3 convolution kernels and 2×2 max-pooling. [1] For MNIST, we also consider multi-layer perceptrons (MLPs) with 1 or 2 layers, ReLU activation function and 1000 hidden nodes per layer. Models are trained using SGD and, unless otherwise stated, with batch normalization in all layers. In all experiments, 20% of training data are held out and used for validation. We stop training, when validation accuracy is the highest and training accuracy is 100%. In cases that training accuracy does not reach 100%, models are trained for 500 epochs. When validation accuracy is not meaningful (e.g., training with random images), we report test accuracy of the last epoch. Results are averaged over five experiments.

3.4 Testing on Negative Images

In this section, we consider the following questions: 1) Do CNNs display shape bias by designs? and 2) Do they need to achieve shape bias at all in order to yield high accuracy on test data? To answer these questions, we examine the performance of CNNs on negative images by designing three experiments on MNIST and CIFAR10 datasets. The experiments on MNIST are illustrated in Figure 3.2 and the results on CIFAR10 are provided in Table 3.1.

[1] The sVGG structure is as follows: (conv $3 \times 3 \times 16$, ReLU, conv $3 \times 3 \times 16$, ReLU, max pool 2×2, conv $3 \times 3 \times 32$, ReLU, conv $3 \times 3 \times 32$, ReLU, max pool 2×2, conv $3 \times 3 \times 48$, ReLU, conv $3 \times 3 \times 48$, ReLU, max pool 2×2, FC-128, ReLU, FC-128, ReLU, FC-10, softmax).

Figure 3.2: Illustration of the three experiments on MNIST. The figure shows one image and the corresponding label as a representative of each class. For example, in experiment 3, regular training images of digit 0 and negative training images of digit 9 are mapped to class zero, and the accuracy on negative test images with correct labels is 0% and accuracy on negative test images with modified labels is 99.5%. In all cases, accuracy on training data is 100%.

In the first experiment, we train the sVGG model only with regular images. As can be seen in Figure 3.2 and Table 3.1, the accuracy on negative test images is significantly lower then the accuracy on regular images. We repeated the experiment without using batch normalization in training, and obtained 12.2% and 21.4% accuracy respectively on MNIST and CIFAR10 negative test images, which signifies the importance of batch normalization in generalization. We will explore the role of batch normalization more in next section. We also performed the experiment with MNIST and using a softmax classifier and 1- and 2-layer MLP models and achieved 0% and around 8% accuracy on negative images, respectively for softmax classifier and MLPs. Hence, regarding generalization to images with similar shapes and different colors, CNNs (with batch normalization) indeed perform better than MLPs.

Table 3.1: Accuracy of sVGG model on regular and negative images of CIFAR10 test data. In experiment 1, the model is only trained with regular images. In experiment 2, the model is trained with both regular and negative images. In experiment 3, the model is also trained with both regular and negative images, but the labels of negative images are changed to $(i + 1)\ mod\ 10$, where i is the ground-truth label. In all cases, accuracy on training data is 100%.

Test Data	**Experiment 1**	**Experiment 2**	**Experiment 3**
Regular images	79.7%	78.3%	75.4%
Negative images	38.7%	78.5%	4.2%
Negative images with modified labels	NA	NA	75.7%

In the second experiment, we train the models with both regular and negative images. In this case, the accuracy of CNN model on negative images is similar to the accuracy on regular images. We again did the experiment with MNIST and using a softmax classifier and the MLP models. For the softmax classifier, the accuracy even on training data is about 15%, since the data is not linearly separable. The two MLP models, however, yield high accuracy on both regular and negative images.

In the third experiment, we train the models with both regular and negative images, but the labels of negative images are changed to $(i + 1)\ mod\ 10$, where i is the ground-truth label. In this case, the model cannot classify images solely based on the shape pattern, since training images in separate classes have the same shapes. Therefore, while the model can certainly achieve 100% training accuracy, the question is whether it can generalize to regular images and also to negative images with modified labels. The answer is yes for both MNIST and CIFAR10 datasets. Specifically, for MNIST, the accuracy on regular and negative test images (with original labels) is 99.5% and 0%, respectively, and the accuracy on negative images with modified labels is 99.5%. We did the experiment with MNIST and MLP classifiers and obtained similar results.

The results on CIFAR10 is similar to MNIST. Note that, unlike MNIST, CIFAR10 consists of natural images. Therefore, it is interesting that the CNN model generalizes to both regular test images and negative test images with modified labels. For instance, negative images of "automobile" class can be classified as "bird," yet the model would achieve high accuracy on both classes of "automobile" and "bird." Specifically, compared with the case where the model was only trained on regular images, the accuracy on regular test images decreases only about 4%, with most of newly misclassified images are labeled as $(i + 1)\ mod\ 10$. Similar result holds for negative test images, i.e., only about 4% of negative test images are classified as their true label. [2]

To conclude, we designed three CNNs that perform similarly on regular images, but very differently on negative images. The results show that CNNs do not intrinsically display shape bias. Specifically, in the second and third experiments, we guided the models to respectively weight shape and color more, and yet they can achieve similar accuracy on regular test images. The results also suggest that accuracy on images that are distributed similarly as training data is not representative of the behavior of machine learning models in the wild. That is, models with identical performance on regular images can behave qualitatively differently on a distinct yet related distribution.

3.5 Training with Negative Images

In this section, we investigate whether CNNs can learn to be "invariant to color." That is, if negative images of some classes are included in training data, does the CNN model correctly recognize negative images of other classes? To answer this question, we first train the sVGG model with all regular images and also negative images of some classes, and test it on negative images of excluded classes. We then extend the experiments by combining two datasets and examine whether the CNN model can transfer features learned on one dataset to another. The latter experiment is conducted on MNIST and notMNIST datasets, since they contain images of similar types.

---

[2] The possibility for the model to yield high accuracy both on regular test images and on negative test images with modified labels can be attributed to the dataset bias discussed in [170].

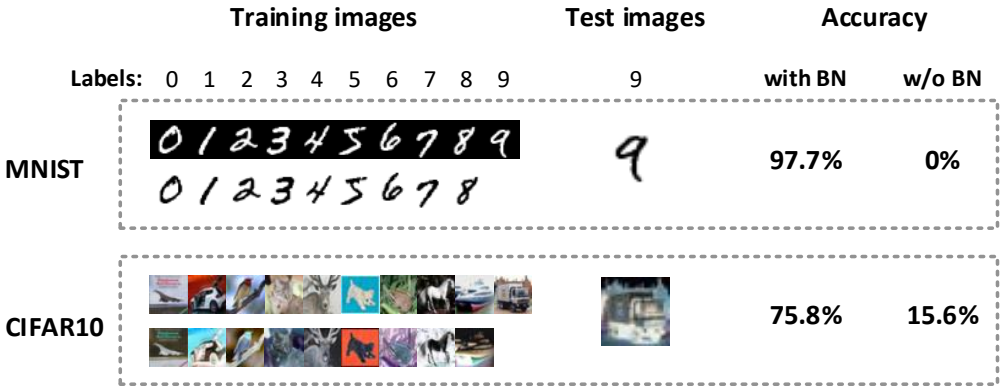


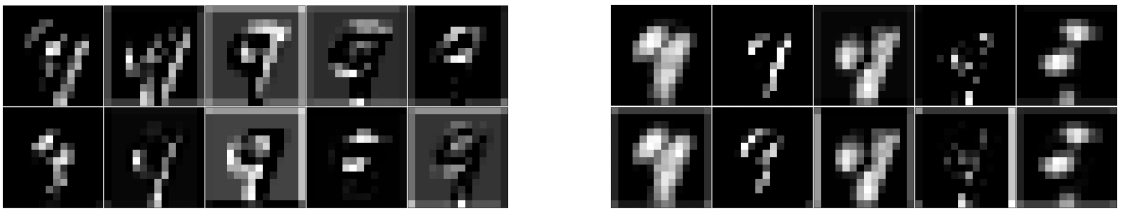
Figure 3.3: An illustration of partially training the model with negative images. BN is batch normalization. We train the CNN with all regular images and also negative images of 9 classes, and test it on the negative images of the excluded class. The experiment is conducted on all classes and the average accuracy is reported. In all cases, accuracy on training data is 100%.

3.5.1 Experiments on One Dataset

For both MNIST and CIFAR10 datasets, we train the sVGG model with all regular images and negative images of 9 classes of training data. We then test the model on negative training images of the excluded class. We do the experiment on all 10 classes and report the average accuracy. Figure 3.3 illustrates the experiment.

Recall from Figure 3.2 that, for MNIST dataset, when the model is only trained with regular images, the accuracy on negative images is 28.3%. By including negative images of 9 classes into the training data, the accuracy on negative images of the excluded class jumps to 97.7%. We repeated the experiment with MNIST and with 1- and 2-layer MLPs. For MLP models, when including negative images of 9 classes in training data, the accuracy on negative images of the excluded class is 0%. Therefore, unlike MLPs which classify inputs based on raw pixel intensities, CNNs are able to classify objects based on their shapes.

To gain an insight into intermediate feature layers, we visualized the outputs of convolutional layers of the sVGG model trained on MNIST. Two cases are considered: 1) training only with



(a) CNN model only trained with regular images.

(b) CNN model trained with all regular images and also negative images of classes 0 to 8.

Figure 3.4: Visualizations of outputs of second convolutional layer. Out of 16 outputs, five of them with most number of active neurons are shown. In each figure, top and bottom rows show outputs on a regular image of digit 9 and its negative, respectively. As can be seen, by simultaneously training with regular and negative images, the model becomes invariant to color.

regular images, and 2) training with all regular images and also negative images of classes 0 to 8. The trained models are then tested with a regular and a negative image of digit 9. Figure 3.4 shows outputs of second convolutional layer with the most number of active neurons. As can be seen, the outputs of the first model are different for regular and negative images. However, for the second model, the first two convolutional layers already provided the invariance to color.

The results on CIFAR10 is similar to MNIST. The accuracy of sVGG on negative images of the excluded class jumps from 38.7% to 75.8%, when negative images of 9 classes are included in training data. Essentially, by including some of negative images in training data, the model learns to weight edge patterns more then the color information. Hence, it can generalize significantly better to other negative images as well.

Role of regularization and batch normalization. We examined the effect of L_2 and Dropout [161] regularizations on model ability to generalize to negative images of the excluded class. Our experimental results show that regularizing the model slows down and stabilizes the training process, but does not affect the generalization.

We also evaluated the role of batch normalization [89] and found that it fundamentally impacts

Table 3.2: Accuracy of different models with and without batch normalization (BN). In case 1, models are trained with regular images and tested on negative images. In case 2, models are trained with all regular images and also negative images of 9 classes, and tested on negative images of the excluded class.

		Case 1		**Case 2**	
Dataset	**Model**	**w/o BN**	**with BN**	**w/o BN**	**with BN**
MNIST	MLP	8%	8%	0%	0%
	CNN	12.2%	28.3%	0%	97.7%
CIFAR10	CNN	21.4%	38.7%	15.6%	75.8%

the model behavior on negative images. Essentially, in our experiment, since the model is trained and tested on different distributions, it must deal with the covariate shift problem [149]. Batch normalization is specifically designed to reduce the internal covariate shift of deep neural networks, by fixing the means and variances of layer inputs within each mini-batch. Table 3.2 summarizes the results of the role of batch normalization on model capability to generalize shapes. As can be seen, CNNs display shape bias only when batch normalization is used. This behavior was consistent across all of our experiments that involved training and testing on different distributions. In the rest of the chapter, we only report the experimental results with batch normalization.

Role of diversity of training data. We now examine whether enhancing the diversity of negative images improves the generalization. For this, we train the sVGG model with all MNIST regular images and also $10,000$ negative images, in two cases: 1) negative images chosen from classes 0 to 3, and 2) negative images chosen from classes 0 to 7. Figure 3.5 illustrates the results. As can be seen, increasing the number of classes while keeping number of negative images the same improves the generalization performance. Essentially, more number of classes enhances the diversity of negative images, and hence helps the model to better recognize negative images of unseen classes.

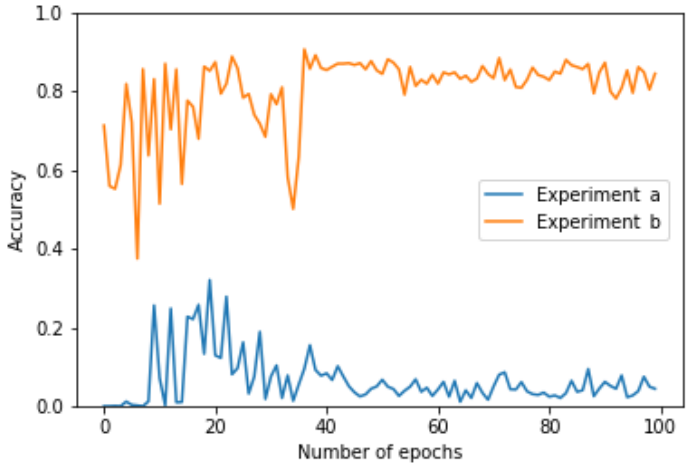
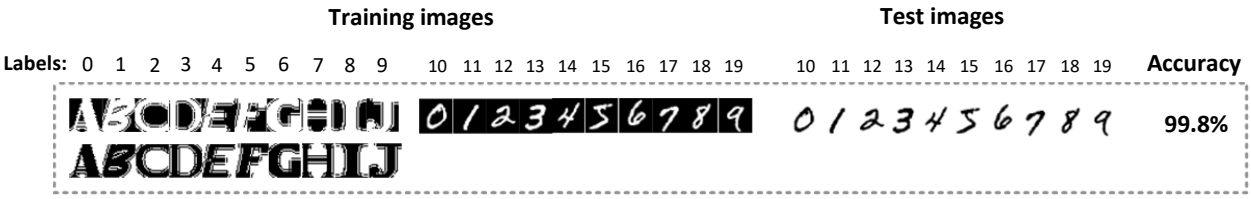


Figure 3.5: Evaluating the effect of diversity of negative images in training data on accuracy on negative images of excluded classes. The sVGG model is trained with all MNIST regular images and $10,000$ negative images. In experiment a, negative images are chosen from classes 0 to 3, while in experiment b, negative images are chosen from classes 0 to 7. More number of classes with same number of negative images improves generalizability to negative images of unseen classes.

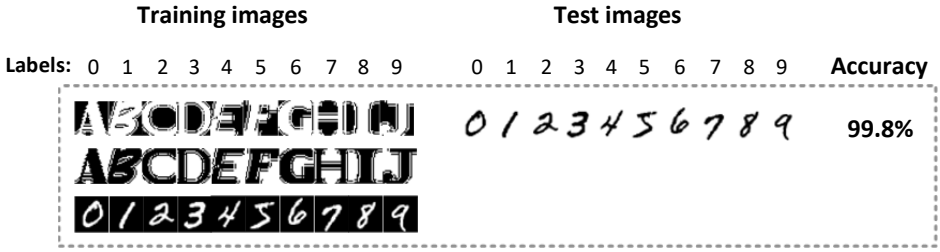
3.5.2 Experiments on Two Datasets

We extend the experiments by combining two datasets and training on negative images of one of them. The sVGG model is trained with regular and negative images of notMNIST and regular images of MNIST. The model has 20 number of classes, i.e., each class in the two datasets is assigned to a unique label. The experiment is illustrated in Figure 3.6a. When tested on MNIST negative images, the model accuracy reaches 99.8%, almost as if it was also trained on them. In essence, the model learns the class representations using MNIST images and improves its basic understanding of objects, e.g., the shape bias property, using regular and negative images of notMNIST.

Accuracy versus different number of negative images. We also investigate whether any number of notMNIST negative images will always lead to better generalization to MNIST negative images. Figure 3.7 shows the accuracy on MNIST negative images versus different number of notMNIST regular and negative images included in training data. As can be seen, training with few negative



(a) Images of MNIST and notMNIST datasets are mapped to different labels.



(b) Images of MNIST and notMNIST datasets are mapped to same labels.

Figure 3.6: An illustration of simultaneously training the model with regular and negative images of notMNIST and regular images of MNIST dataset.

images causes the accuracy to decrease. As an example, recall from Figure 3.2 that, when trained with regular images, the accuracy on MNIST negative images is 28.3%. By including only 10 to 100 notMNIST negative images in training data, the model accuracy on MNIST negative images drops to 0%. The accuracy, however, reaches 99% when the model is trained with 10000 notMNIST regular and negatives images. This implies that the CNN model learns to classify objects by their shapes, only when it is trained with a large enough number of regular and negative image pairs.

3.6 Role of Data Augmentation and Initialization on Shape Bias

In this section, we examine the transferability of features from one dataset to another, by studying the effect of augmenting MNIST dataset with notMNIST dataset and the role of initialization on model generalizability. We also investigate whether the features can be transferred to or from a dataset with random images.

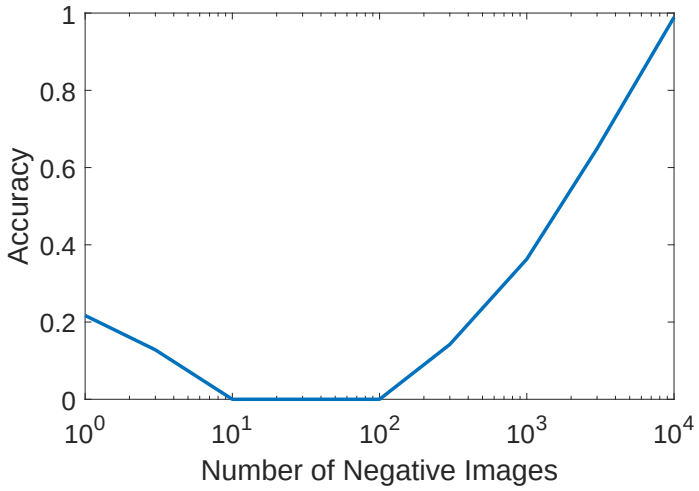


Figure 3.7: Accuracy of sVGG model on MNIST negative images versus number of notMNIST negative images included in training data. The model is trained with regular and negative images of notMNIST and regular images of MNIST.

3.6.1 Role of Data Augmentation

Similar to Section 3.5.2, we train the sVGG model with regular and negative images of notMNIST and regular images of MNIST, but with 10 number of classes. That is, all images are mapped to labels 0 to 9, e.g., images of digit 0 of MNIST and images of letter A of notMNIST are assigned the label 0. The experiment is illustrated in Figure 3.6b. In this case, we essentially augment the MNIST dataset with regular and negative images of notMNIST. Similar to the case of training with 20 labels, when tested on negative images of MNIST, the model accuracy reaches 99.8%.

Assigning images of the two datasets to the same labels has a practical implication. Most data augmentation techniques include transformed version of images into the training data. Our results, however, show that one can augment a target dataset with another dataset that contains similar type of images, though with unrelated labels. Moreover, in the case that the two datasets do not have common classes, each class of the external dataset can be randomly labeled. Augmenting a target dataset with unrelated data is especially helpful, when the dataset size is small and it is difficult to obtain samples from the classes of target dataset.

Generalization to/from random images. We examine whether the CNN can learn the shape bias property from a dataset that lacks any structure, e.g., a dataset with random images. We also investigate how a model with shape bias property performs on random images and their negatives. In experiments, we generate a dataset, with the same size as MNIST, consisting of random images with uniform distribution in $[0, 1]$. Images are labeled randomly between 0 to 9.

For answering the first question, we train the sVGG model with random images and their negatives and with the regular images of MNIST. The model is trained with 10 labels, i.e., random images and MNIST images are mapped to the same set of classes. Our experimental results show that the model accuracy on MNIST negative images is about 30%, implying that including random images and their negatives in the training data does not improve the model generalization to MNIST negative images. In the second experiment, we train the sVGG model with notMNIST regular and negative images and also with random images. The test accuracy on negatives of random images is 10%. In conclusion, although CNN models can fit any dataset, they only learn and generalize the structures.

3.6.2 Role of Initialization

Now, we examine the effect of initialization on the ability of CNNs to generalize to negative images. For this, we first train the sVGG model with regular and negative images of notMNIST dataset for 100 epochs, and then fine-tune with MNIST regular images for another 100 epochs. We consider two cases for the first phase of training: 1) training with notMNIST regular and negative images with correct labels, 2) training with notMNIST regular and negative images, with labels of negative images being modified to $(i + 1)\ mod\ 10$, where i is the ground-truth label.

Figures 3.8a and 3.8b show the model accuracy on MNIST regular and negative images versus epoch number for the two cases. While accuracy on MNIST images in not meaningful during the first phase of training, notably for the first case, it is similar for both MNIST regular and negative images. This implies that, when correctly trained with regular and negative images of one dataset, the model performs similarly on regular and negative images of another dataset as well. In the first case, after fine-tuning the model with MNIST regular images, the accuracy on negative images

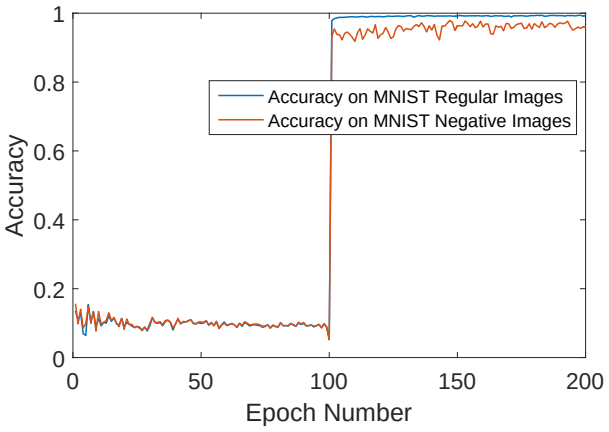
increases and remains on par with the accuracy on regular images, i.e., the shape bias property does not fade away when the model is fine-tuned only with regular images. In the second case, the model accuracy on MNIST negative images converges to about 30%, which is similar to the case where the model was initialized randomly.

We also use KL-divergence metric to measure the similarity of the model output probability vectors on regular and negative images. Let $P(X)$ be the model output probability vector for an image X. The KL-divergence from $P(X)$ to $P(1-X)$ is defined as follows:

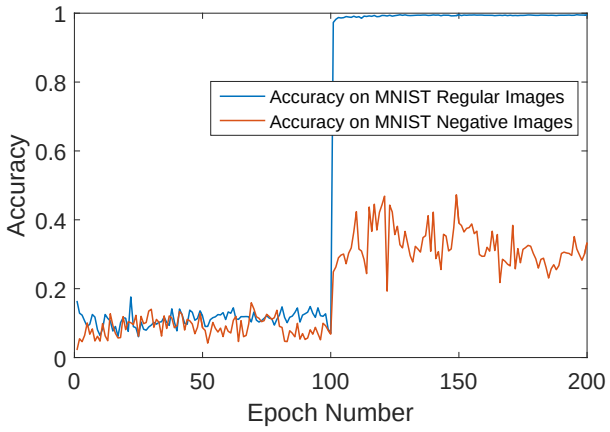
$$D_{\mathrm{KL}}(P(X)\|P(1-X)) = \sum_i P(X)_i \log \frac{P(X)_i}{P(1-X)_i}.$$

We define $D = \mathbb{E}_X[D_{\mathrm{KL}}(P(X)\|P(1-X))]$, i.e., the average KL-divergence of model outputs on all pairs of regular and negative images. Figure 3.8c shows the value of D versus epoch number for the two cases. As can be seen, in the first case (training with correct labels) and during the first phase of training, the KL divergence remains low and on par with the KL divergence in the second phase. This further demonstrates that CNN learns to be invariant to color. However, in the second case (training with notMNIST negative images with wrong labels), the average KL divergence is significantly higher throughout the training. In conclusion, a different initialization can lead to a qualitatively different model, e.g., in the first case, it is as if the shape bias property is encoded into the model at initialization.

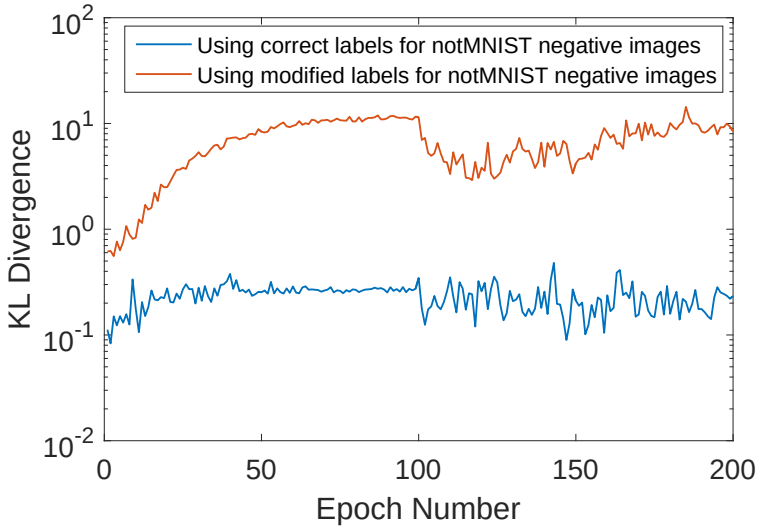
The experiments also show that data augmentation yields better results compared to fine-tuning, as the model seems to fully retain features of the first dataset. Specifically, with fine-tuning and data augmentation, we obtained about 95.4% and 99.8% accuracy on MNIST negative images, respectively. The proposed approach of data augmentation can also mitigate catastrophic forgetting in neural networks [64], a phenomenon in which models "forget" how to perform the first task, when retrained on a second task.



(a) Accuracy versus epoch number in the first case.



(b) Accuracy versus epoch number in second case.



(c) KL divergence between model outputs on MNIST regular and negative images in both cases.

Figure 3.8: Role of initialization and fine-tuning on shape bias. The CNN model is trained with notMNIST regular and negative images for 100 epochs and then fine-tuned with MNIST regular images for another 100 epochs. Two cases are considered for the first phase of training: 1) training with notMNIST images with correct labels, and 2) training with notMNIST images with labels of negative images being modified to $(i + 1)\ mod\ 10$, where i is the ground-truth label. Throughout training, we test the model on MNIST regular and negative images. Only in the first case, the model performs similarly on MNIST regular and negative images, implying that it learns to be invariant to color.

3.7 Semantic Adversarial Examples

Image classifiers are vulnerable to adversarial inputs, i.e., it is possible to carefully modify an image such that the model will classify it into a wrong label, while a human observer perceives the original object [21, 166]. Generating adversarial examples has been mostly limited to finding small perturbations that maximize the model prediction error [65, 121, 100, 37]. Such modified images, however, contain artificial perturbations that make them somewhat distinguishable from natural images. This property is used by several defense methods to make deep learning models robust against small perturbations [171, 116] or to map the perturbed image back into the space of natural images by applying preprocessing filters [159, 70].

In practice, however, the adversary may not be constrained with slightly modifying the image. That is, the adversary may perturb the image a lot, but in such a way that the modified image semantically represents the same object as the original image (because otherwise, we cannot expect the model to classify it correctly). To construct such images, we need to identify the types of transformations that human vision is invariant to and investigate how do start-of-the-art deep learning models compare to humans. We develop a method for constructing adversarial examples based on the shape bias property. Specifically, we introduce a new class of adversarial examples, namely *semantic adversarial examples*, as images that are arbitrarily perturbed to fool the model, but semantically represent the original objects.

3.7.1 Problem Statement

Adversarial Examples. We consider the misclassification attack. Current techniques for generating adversarial examples try to find a perturbation that maximizes the network prediction error. Let F be the machine learning classifier and x be the given image. The adversarial perturbation is typically found by solving the following optimization problem [166]:

$$\begin{aligned} &\min \|\delta\| \\ &\text{s.t. } F(x+\delta) \neq F(x). \end{aligned} \tag{3.1}$$

The added perturbation in adversarial examples is usually small and, hence, the modified image is likely to belong to the same class as the original image. The image, however, does contain an artificial perturbation that makes it somewhat distinguishable from natural images. This property is used by several defense methods to counter adversarial examples by explicitly applying denoising operations [159, 70] or training the model to do so implicitly [171, 116].

Semantic Adversarial Examples. In practice, the adversary may not be constrained with slightly modifying the image. That is, the image can be modified by any transformation, conditioned that the transformation preservers the semantics of the image. Let Ω be the human vision system. The problem of generating semantic adversarial examples is stated as follows:

$$\begin{aligned} &\text{find } x^* \\ &\text{s.t. } \Omega(x^*) = \Omega(x) \text{ and } F(x^*) \neq F(x). \end{aligned} \tag{3.2}$$

The problem (3.2) can be seen as mapping any given image into the space of natural images that are misclassified by the model, but contain the original object. In this sense, wrongly-classified clean images are adversarial examples with zero perturbation.

Identifying and studying adversarial transformations is important from the learning perspective, since it helps to investigate how the model compares to human visual system and also to analyze the model generalization performance. Moreover, such adversarial transformations will be able to evade state-of-the-art defense methods that try to reverse the added perturbation. Therefore, it is important also from the security perspective to identify the attack space and develop more robust defense mechanisms.

3.7.2 Proposed Method

In this section, we first review the shape bias property of human cognitive system. We then provide a background on HSV color space and finally propose a method for generating semantic adversarial examples by shifting the image color components.

Shape Bias Property of Human Cognitive System. To construct semantic adversarial examples, we need to identify the properties of human vision system. One such property is the "shape bias,"

stating that humans prefer to categorize objects according to their shape rather than color [102]. Therefore, we expect machine learning models to also correctly classify images that contain the original object with different colors. In the following, we propose a method to generate such images.

HSV Color Space. HSV (Hue, Saturation and Value) is an alternative to RGB (red, green and blue) color space and is known to more closely represent the way human vision perceives color [97]. The hue channel corresponds to the color's position on the color wheel. As hue increases from 0 to 1, the color transitions from red to orange, yellow, green, cyan, blue, magenta, and finally back to red. Saturation measures the colorfulness, i.e., setting saturation to 0 yields a gray-scale image and increasing it to 1 generates the most colorful image with same colors. Value shows the brightness, which is maximum value of red, green and blue components.

Color-Shifted Images as Semantic Adversarial Examples. HSV can be seen as a color space, in which color components (hue and saturation) are decoupled from the object structure (brightness). Therefore, by changing the hue and saturation components and keeping the value the same, we can generate images that contain the original object with different color and colorfulness. Let x_H, x_S and x_V respectively denote the hue, saturation and value components of image x. The problem of generating semantic adversarial examples by changing the image color can be stated as follows:

$$\begin{aligned} &\text{find } x^* \\ &\text{s.t. } x^*_V = x_V \text{ and } F(x^*) \neq F(x). \end{aligned} \tag{3.3}$$

For solving (3.3), we can generate random images, set their value component to x_V and then choose the ones that are misclassified by the model. Equally, we can start from the given image and randomly perturb the hue and saturation components in such a way that the modified image can fool the model. These methods, however, will generate images with visible noise.

In order to generate smooth and natural-looking images, we *shift* the hue and saturation components of all pixels by the same amount. This approach generates an image where all pixels are equally colored if the shift in saturation is positive or decolored if the shift is negative. The color of all pixels (the hue component) is also shifted by a fixed amount. We call such images as

color-shifted images.

We also note that significantly decreasing or increasing the saturation component generates gray-scale or too colorful images, respectively, and hence causes the image to be less like a natural color image. To generate better looking images, we add a requirement that the saturation component be minimally shifted. Figure 3.9 shows a sample image of CIFAR10 dataset and several color-shifted images. The center image is the original one. As can be seen, images differ in color and colorfulness across variations in hue and saturation components, respectively. Also, while the original object is recognizable in all images, images closer to the middle column look more like natural color images.

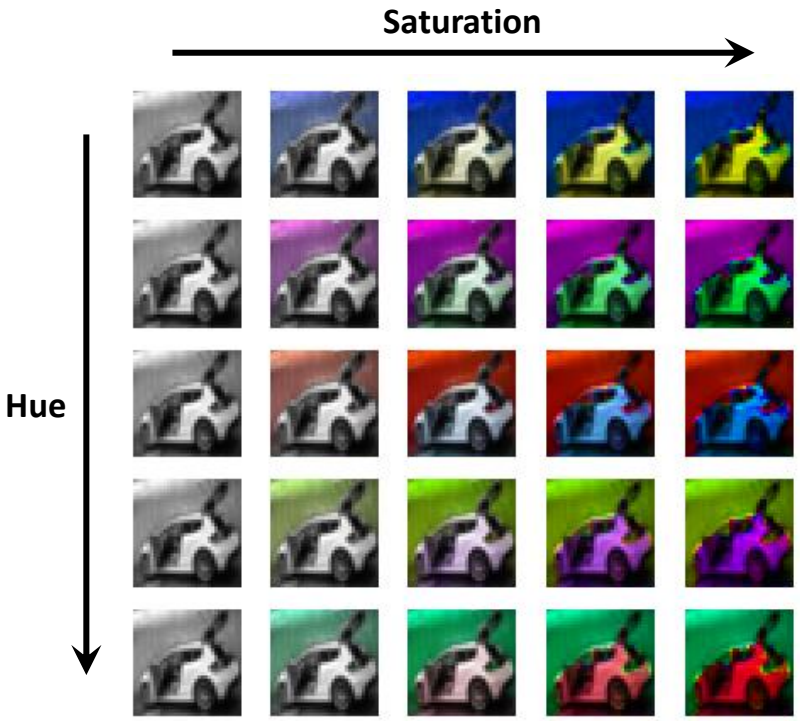


Figure 3.9: Illustration of shifting color components (hue and saturation) in HSV color space on a sample image of CIFAR10 dataset. The center image is the original one. Shifting hue and saturation components changes the color and colorfulness, respectively. As can be seen, the original object is recognizable in all images.

Let δ_H and δ_S denote the shifts in hue and saturation components, respectively. Adversarial color-shifted images can be generated by solving the following problem:

$$\begin{aligned} &\min \ |\delta_S| \\ &\text{s.t.} \begin{cases} x^*_H = (x_H + \delta_H) \bmod 1 \\ x^*_S = \text{clip}(x_S + \delta_S, 0, 1) \\ x^*_V = x_V \end{cases} \\ &\text{and } F(x^*) \neq F(x). \end{aligned} \tag{3.4}$$

Note that δ_H and δ_S are scalars. The color in hue component changes in a circle, i.e., hue of 1 is equal to hue of 0. Hence, we compute the modulo of hue component with 1 to map it to $[0, 1]$. The saturation component, however, should be clipped to the interval of $[0, 1]$.

Algorithm. To solve (3.4), we do a random search over parameters δ_H and δ_S, and obtain the modified image as described in first constraint of (3.4). We continue generating color-shifted images until the modified image is misclassified by the model or the maximum number of trials, denoted by N, is reached. The shift in hue component is chosen as $\delta_H \sim U(0, 1)$, where $U(a, b)$ denotes uniform distribution in $[a, b]$. However, to find adversarial images with smaller saturation shift, δ_S is chosen as $\delta_S \sim U(-\frac{i}{N}, \frac{i}{N})$, where i is the iteration number. That is, we start with $\delta_S = 0$ and linearly increase the interval after each trial. Algorithm 1 describes the method.

3.7.3 Experimental Results

Experiments are performed on image dataset CIFAR10, which consists of natural color images in 10 classes of airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck [98]. We apply the attack on the pretrained VGG16 network [155], a robust network proposed by Madry et al. [116], and the VGG16 network trained with both original and color-shifted images. In experiments, we set maximum number of trials $N = 1000$.

Table 3.3 provides the results for different models. The accuracy of pretrained VGG16 network drops to 5.7% when tested on adversarial color-shifted images. Figure 3.10 shows attack success

Algorithm 1 Generating Adversarial Color-shifted Images

Input: Classifier F, Image x, Maximum number of trials N
Output: Adversarial color-shifted image x^* or $\emptyset$
$x_H, x_S, x_V \leftarrow$ Hue, Saturation and Value components of image x, respectively.
$x^* \leftarrow x$
for $i = 0, ..., N-1$ **do**
    $\delta_H \leftarrow$ a number uniformly chosen in $[0, 1]$
    $\delta_S \leftarrow$ a number uniformly chosen in $[-\frac{i}{N}, \frac{i}{N}]$
    $x^*_H = (x_H + \delta_H) \bmod 1$
    $x^*_S = \text{clip}(x_S + \delta_S, 0, 1)$
    if $F(x^*) \neq F(x)$ **then**
        return x^*
    end if
end for
return $\emptyset$

rate versus number of trials. For more than 14% of images, the model can be fooled by trying only one modified image, obtained by randomly shifting only the hue component. The results show that although the model achieves accuracy of 93.6% on test data, it is fragile to images with maliciously-shifted color components.

We also applied targeted attack on VGG16 network. In our method, the attack space is limited, since we search over only two parameters. Nevertheless, we could achieve 35.4% success rate on images that are correctly classified by model, i.e., adversarial color-shifting can change the model prediction to an average of more than three classes. Figure 3.11 shows sample images of CIFAR10 dataset, each with three color-shifted versions classified into different labels.

The model proposed by Madry el al. yields the state-of-the-art results against adversarial examples [13], by providing robustness against worse-case perturbations [116]. We apply our attack also on this model to ensure that our method does not add noise-like perturbation to the image.

Table 3.3: Accuracy of different CNNs on test data and adversarial color-shifted images. The VGG-augmented is a VGG16 network trained with both original and color-shifted images.

Network	**Accuracy on test images**	**Accuracy on adversarial color-shifted images**
Pretrained VGG16 [155]	93.6%	5.7%
Madry et al. Model [116]	87.3%	8.4%
VGG-augmented	89.9%	69.1%

We observed that the accuracy of the robust CNN is 8.4% on adversarial color-shifted images, implying that even if the model is robust to perturbations around data points, it does not provide robustness to semantic adversarial examples.

We also applied the attack on a VGG16 network which, at each epoch, is trained with original and color-shifted images with $\delta_H \sim U(0,1)$ and $\delta_S \sim U(-1,1)$. The accuracy on adversarial color-shifted images is 69.1%, indicating that the model shows more robustness when trained with same types of images. However, as pointed out in [91], robustness achieved by data augmentation may not be an indication that the model has learned higher level semantic features in the dataset. That is, the network is likely to be vulnerable to other types of semantic adversarial examples. We will explore other attack and defense methods in future works.

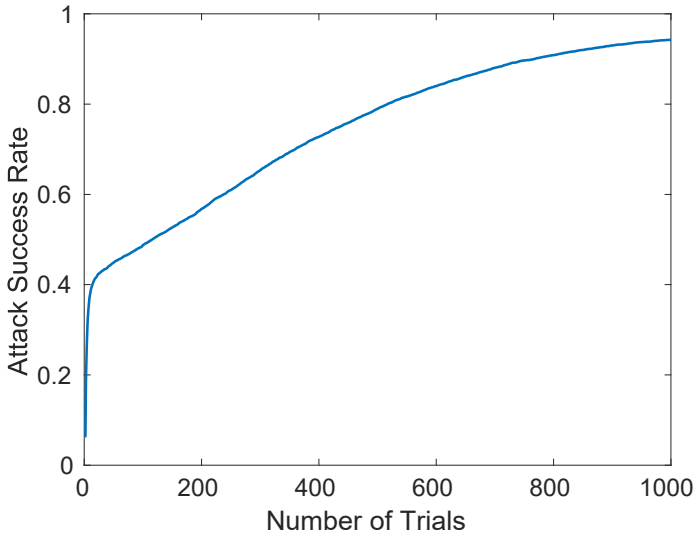


Figure 3.10: Attack success rate versus number of trials. For more than 14% of images, the model is fooled after trying only one color-shifted image, obtained by randomly shifting the hue component.

Figure 3.11: Sample images of CIFAR10 dataset, along with three color-shifted versions classified into different labels by VGG16 network. The leftmost-column shows original images.

Chapter 4

SELECTIVE CLASSIFICATION FOR REJECTING ADVERSARIAL EXAMPLES

4.1 Introduction

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake [63]. One particular method for generating adversarial examples for image classifiers is adding small perturbations to legitimate inputs such that the modified image is misclassified by the model, but a human observer perceives the original content [21, 166].

Several methods have been proposed for developing robust algorithms that classify adversarily perturbed examples into their ground-truth label, but later broken using adaptive iterative attacks [35, 13]. One promising direction for defending against adversarial examples is adversarial training [65], which has produced state-of-the-art results on robustness of deep neural networks [116]. The obtained adversarial accuracy is, however, significantly lower than the standard accuracy. Moreover, it is observed that adversarial robustness is improved at the cost of a reduction of standard generalization [173].

In security-sensitive and safety-critical applications, misclassifying or even accepting adversarial samples can lead to a harmful situation and, hence, it is crucial to be able to reject malicious data points. For example, in medical diagnosis applications, the cost of rejecting a sample might be additional medical tests, which is far more acceptable than taking the risk of deciding based on a potentially perturbed sample. As another example, the vision system in self-driving car can quickly notify the human driver to take over if it identifies that the input is not trusted. Such a classifier that "abstains" from classifying certain inputs is called selective classifier [60].

In this chapter, we propose a selective classification method for rejecting adversarial examples

with bounded L_∞ perturbation. Our contributions are summarized in the following.

- We first analyze the binary classification problem presented in [173] and prove that selective classification improves the robustness. Particularly, we show that in a setting where the robustness of standard (non-selective) classifier cannot be improved, i.e., when adversarial examples are inevitable, selective classification provides a trade-off between standard accuracy and robustness. We then examine a setting where input features contain both robust and non-robust features and hence robustness of standard classifier can be improved by assigning higher weights to robust features. We show that in this case selective classifier achieves better trade-off between standard and adversarial accuracy compared to any non-selective classifier.

- We then adapt adversarial training procedure to train a selective classifier with an extra class called *outlier*. During training, adversarial examples are generated with perturbation sizes uniformly chosen in range of zero to the maximum perturbation size. The model is trained to classify examples with small perturbations into their true label. Examples with large perturbations are, however, first evaluated by the model. The model is then trained to retain its confidence in the true label and assign the rest of the probability to the outlier label.

- We perform experiments on convolutional networks trained with MNIST and CIFAR10 datasets. The models are trained and tested with adversarial examples generated using projected gradient descent (PGD) method [100]. In our selective classifier, an attack is successful if the adversarial image is classified into any label other than the true or outlier labels. On MNIST dataset, we show that our proposed method achieves state-of-the-art adversarial accuracy of 99.2% and 99.6% in white-box and black-box settings respectively, while reaching 99.3% standard accuracy. On CIFAR10 dataset, we implement selective classification on the VGG network and show that the model achieves 81.0% standard accuracy and state-of-the-art adversarial accuracy of 52.4% and 94.7% in white-box and black-box settings, respectively.

- We evaluate the models against characteristic behaviors of gradient masking provided by [13]. In particular, we show that the models perform well in black-box setting and also against a random attack. Moreover, adversarial accuracy in white-box setting decreases to zero by increasing perturbation size. The results indicate that our defense mechanism does not cause gradient masking.

4.2 Preliminaries

In this section, we provide an overview on the iterative optimization-based method for generating adversarial examples and the adversarial training approach for defending against them.

4.2.1 Notations

Let $F : X \to Z$ be a function that takes an image $X \in \mathbb{R}^d$, where d is the number of pixels, and outputs the vector $Z \in \mathbb{R}^k$, where k is the number of classes and $\sum_i Z_i = 1$. Let also $f : X \to y$ be a classifier that takes the image X and outputs the label $y \in \{0, \cdots, k-1\}$, where $y = \operatorname{argmax}_i Z_i$. Let $\ell(X, y; \theta)$ denote the loss of the classifier with parameters θ on (X, y). When holding θ fixed and viewing the loss as a function of (X, y), we simply write $\ell(X, y)$. We alternatively write $\ell(X, Z)$ for the loss of the classifier on input X and output probability vector Z.

4.2.2 Adversarial Examples

We consider a class of adversarial examples for image classifiers where small (imperceptible) perturbations are added to images to force the model to misclassify them. Similar to [116], we consider the case where the L_∞ norm of the perturbation is bounded. Formally, the attacker's problem is stated as follows:

$$\begin{aligned} &\text{given } X, \text{ find } X' \\ &\text{s.t. } \|X' - X\|_\infty \leq \epsilon_{\max} \text{ and } f(X') \neq y, \end{aligned} \tag{4.1}$$

where X and X' are the clean and adversarial samples, respectively, y is the true label, and $\epsilon_{\max}$ is the maximum allowed absolute change to each pixel.

Several optimization-based attacks have been proposed for generating adversarial examples, including fast gradient sign method [65], iterative gradient method [100], DeepFool [121], and Carlini and Wagner attacks [37]. It has been observed that the specific choice of optimizer is less important than choosing to use iterative optimization-based methods [116].

We generate adversarial examples using the Projected Gradient Descent (PGD) method [100, 116], which provides a unified framework for iterative attacks independent of the specific optimization function. Let n be the number of steps and X^j be the image at step j. We have $X^0 = X$ and $X' = X^n$. At step j, the image is updated as follows:

$$X^{j+1} = \Pi_{X+\mathcal{S}}(X^j + \epsilon_{\text{step}} \operatorname{sign}(V^j)), \tag{4.2}$$

where V^j is the attack vector at step j, ϵ_{step} is the added perturbation per step, and $\Pi_{X+\mathcal{S}}$ is the projection operator where $\mathcal{S}$ is the set of allowed perturbations. In the case of bounded L_∞ constraint, projector clips each pixel of the image within $\epsilon_{\max}$ of the corresponding pixel of original image X. Instead of starting with clean image X, the attacker can start from a randomly chosen point inside $X+\mathcal{S}$, i.e., $X^0 = X+V$, where $V_i \sim U[-\epsilon_{\max}, \epsilon_{\max}]$ [171, 116]. We characterize PGD attacks with the tuple $(\epsilon_{\max}, \epsilon_{\text{step}}, n, \text{random\_start})$, where $\text{random\_start} \in \{\text{False}, \text{True}\}$ determines whether the attack is started from the original or noisy image.

The attack vector takes different forms depending on the attack goal. Generally, the attacker's goal is to maximize the loss on the defender's desired output probability vector or alternatively minimize the loss on attacker's desired probability vector. These approaches lead to two common attacks respectively known as misclassification and targeted attacks, for which attack vectors are specified as follows:

- $V^j = \nabla_X \ell(X^j, y)$, for misclassification attack,
- $V^j = -\nabla_X \ell(X^j, y_t)$, for targeted attack,

where $y_t \neq y$ is the attacker's desired target label.

4.2.3 Adversarial Training

One countermeasure against adversarial examples is to modify the training procedure to reflect the fact that perturbations of each training sample should be classified as the original class. This approach is known as adversarial training [65] and is formulated as a min-max optimization problem as follows [116]:

$$\min_{\theta} \mathbb{E}_{(X,y)\sim\mathcal{D}}\big[\max_{\delta\in\mathcal{S}} \ell(X+\delta, y; \theta)\big] \tag{4.3}$$

where $\mathcal{D}$ is the underlying data distribution over pairs of images X and corresponding labels y and $\mathcal{S}$ is the set of allowed perturbations. This approach iteratively generates batches of adversarial examples and trains the model to correctly classify them. It has been shown that the modified objective function makes the classifier more robust to adversarial examples [116].

4.3 Rejecting Adversarial Examples

Natural images are highly compressible and hence reside in low-dimensional subspace of input space. However, while training datasets occupy only a small fraction of input space, image classifiers are allowed to be tested with *any* input. This causes classifiers to be vulnerable to maliciously perturbed and out-of-distribution inputs, such as adversarial or fooling images [166, 125]. Hence, one approach to defend against adversarial examples is to design a classifier that, in addition to learning the boundaries of different classes, learns the distribution of training data so as to reject invalid inputs at inference time.

In this paper, we propose a method for training a selective classifier to reject adversarial examples. We augment the set of classes with an *outlier* class, which is the class number k and also denoted by $\varnothing$, and train the model to classify adversarial examples into either their true label or outlier, while mapping clean images to their ground-truth label. The standard and adversarial accuracy of the selective classifier is defined in the following.

Definition 1: *Let $f : X \to y$ be a selective classifier that takes an image $X \in \mathbb{R}^d$ and outputs the label $y \in \{0, \cdots, k\}$, where $y = k$ is the outlier label. The standard accuracy is defined as the*

probability that clean samples are classified correctly, i.e.,

$$p_{clean} = \Pr_{(X,y)\sim\mathcal{D}}(f(X) = y). \tag{4.4}$$

Let A be an algorithm that given X generates adversarial example $A(X)$ subject to $\|A(X) - X\|_\infty \leq \epsilon_{\max}$. Adversarial accuracy is defined as the probability that an adversarial example is classified into either the true label or outlier, i.e.,

$$p_{adv} = \Pr_{(X,y)\sim\mathcal{D}}(f(A(X)) = y \text{ or } f(A(X)) = k). \tag{4.5}$$

4.4 Improving Accuracy-Robustness Trade-off by Selective Classification

The empirical observations of difficulty of defending against adversarial perturbations have led to studies of fundamental limitations of robustness of machine learning classifiers [53, 146, 173]. It has been argued that no classifier can be made robust [53] and that reduction in accuracy on clean data is the inevitable price of robustness [173]. We argue that by augmenting the classifier with the reject option, one might be able to improve the trade-off of accuracy on clean data and robustness on adversarial data. In the following, we present our analysis by extending the example given by [173].

4.4.1 Setup

Consider a dataset consisting of input-label pairs (x, y) sampled from the following distribution [173]:

$$\begin{aligned} y &\overset{u.a.r}{\sim} \{-1, +1\}, \\ x_1 &= \begin{cases} +y, & \text{w.p. } p, \\ -y, & \text{w.p. } 1-p \end{cases}, x_2, \ldots, x_{d+1} \overset{i.i.d}{\sim} \mathcal{N}(\eta y, 1), \end{aligned} \tag{4.6}$$

where $\mathcal{N}(\mu, \sigma^2)$ is a normal distribution with mean μ and variance σ^2 and $\epsilon_{\max} \in [0, 1)$ is the maximum allowed adversarial perturbation. We assume $\eta > \epsilon_{\max}$.

The adversary's goal is to perturb x so as to reduce the accuracy on adversarial data. Let x' denote the adversarial input. Since $|x_1| > \epsilon_{\max}$, the sign of first feature remains the same after perturbation, i.e., $x_1 = \text{sign}(x_1) = \text{sign}(x'_1)$. As a result, the first feature is robust to perturbation. The rest of features are positively correlated with y. Hence, the adversary's optimal strategy is to perturb them along the direction of $-y$, i.e., adversarial features are obtained as $x'_i = x_i - \epsilon_{\max} y, i \in \{2, \cdots, d+1\}$, and distributed as $\mathcal{N}((\eta - \epsilon_{\max})y, 1)$.

Let $p_\lambda = \lambda p_{\text{clean}} + (1-\lambda) p_{\text{adv}}$, where $\lambda \in [0, 1]$ balances the standard and adversarial accuracy. A normal classifier is designed to maximize the accuracy on clean data, i.e., assuming $\lambda = 1$. The robust optimization framework of (4.3), however, trains the model to correctly classify the worst perturbation, hence assuming $\lambda = 0$. For $\lambda \in [0, 1]$, the optimal classifier is obtained according to the following Lemma.

Lemma 1. *Let $p_\lambda = \lambda p_{clean} + (1-\lambda) p_{adv}$. The optimal classifier for maximizing p_λ on the dataset defined in (4.6) is*

$$\hat{y} = \textit{sign}\big(w_1 x_1 + \Sigma_{i=2}^{d+1} w x_i\big), \tag{4.7}$$

where $w_1 = \log(\frac{p}{1-p})$ and $w = 2(\eta - (1-\lambda)\epsilon_{\max})$.

Proof. For the dataset defined in (4.6), we have

$$\begin{aligned}\frac{\Pr(y=1|x)}{\Pr(y=-1|x)} &= \frac{\Pr(x|y=1)\Pr(y=1)/\Pr(x)}{\Pr(x|y=-1)\Pr(y=-1)/\Pr(x)} \\ &= \frac{\Pr(x|y=1)}{\Pr(x|y=-1)}.\end{aligned}$$

Since features are independent random variables, $\Pr(x|y)$ is computed as follows:

$$\begin{aligned}\Pr(x|y) &= \Pr(x_1|y) \cdot \Pi_{i=2}^{d+1} \Pr(x_i|y) \\ &= \big(\frac{1}{2} + y x_1 (p - \frac{1}{2})\big) \cdot \Pi_{i=2}^{d+1} \Pr(\mathcal{N}(\mu, 1) = x_i).\end{aligned}$$

where $\mu = (\eta - a\epsilon_{\max})y$ and a is a Bernoulli random variable corresponding to each sample that indicates whether the sample is adversarial or not, i.e., $a = 1$ if and only if the sample is adversarial.

Hence,

$$\begin{aligned}
&\log(\Pr(x|y)) \\
&= \log\big(\frac{1}{2} + yx_1(p-\frac{1}{2})\big) + \sum_{i=2}^{d+1} \log(\Pr(\mathcal{N}(\mu,1)=x_i)) \\
&= \log\big(\frac{1}{2} + yx_1(p-\frac{1}{2})\big) + \sum_{i=2}^{d+1} \frac{-(x_i-\mu)^2}{2} - d\cdot\log(\sqrt{2\pi}).
\end{aligned}$$

Let $r(x) = \log(\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}) = \log(\frac{\Pr(x|y=1)}{\Pr(x|y=-1)})$. We have

$$\begin{aligned}
r(x|a) &= \log(\Pr(x|y=1,a)) - \log(\Pr(x|y=-1,a))) \\
&= \log\big(\frac{\frac{1}{2} + x_1(p-\frac{1}{2})}{\frac{1}{2} - x_1(p-\frac{1}{2})}\big) + \sum_{i=2}^{d+1} 2(\eta - a\epsilon_{\max})x_i \\
&= \log(\frac{p}{1-p})x_1 + \sum_{i=2}^{d+1} 2(\eta - a\epsilon_{\max})x_i. \qquad (4.8)
\end{aligned}$$

We also have $r(x) = \lambda r(x|a=0) + (1-\lambda) r(x|a=1)$. Replacing $r(x|a)$ from (4.8), we obtain

$$r(x) = \log(\frac{p}{1-p})x_1 + \sum_{i=2}^{d+1} 2(\eta - (1-\lambda)\epsilon_{\max})x_i.$$

Therefore, the optimal classifier is obtained as follows:

$$\hat{y} = \text{sign}\big(\log(r(x))\big) = \text{sign}(w_1x_1 + \sum_{i=2}^{d+1} wx_i),$$

where the weights are $w_1 = \log(\frac{p}{1-p})$ and $w = 2(\eta - (1-\lambda)\epsilon_{\max})$. ■

A selective classifier can reject the sample if it is suspected to be altered by an adversary to cause misclassification. The selective classifier is defined as follows:

$$\hat{y} = \begin{cases} \text{sign}(z), & \text{if } |z| > h, \\ \varnothing, & \text{otherwise} \end{cases}, \qquad (4.9)$$

where $z = w_1x_1 + \sum_{i=2}^{d+1} wx_i$ and h is the threshold. In adversarial setting, we have $z' = w_1x_1 + \sum_{i=2}^{d+1} wx_i' = z - dw\epsilon_{\max}y$. The standard and adversarial accuracy of selective classi-

fier are obtained as follows:

$$p^{\text{s}}_{\text{clean}} = \Pr(yz > h), \tag{4.10}$$

$$p^{\text{s}}_{\text{adv}} = \Pr(yz' > -h) = \Pr(yz > dw\epsilon_{\max} - h). \tag{4.11}$$

In the following, we investigate two cases of $p = 0.5$ and $\lambda = 0.5$ to show how selective classification improves the trade-off between standard accuracy and robustness.

4.4.2 Case I: On Existence of Robust Classifiers

Let $p = 0.5$, thus we have $w_1 = 0$ in (4.7), and hence the classifier must only rely on non-robust features to predict the output. Since $w > 0$ regardless of the value of λ, both normal and robust classifiers are obtained as $\hat{y} = \text{sign}\big(\sum_{i=2}^{d+1} x_i\big)$. In this case, we cannot assign higher weights to certain features to increase robustness, since all features are equally fragile. As a result, the adversarial accuracy cannot be improved. The corresponding standard and adversarial accuracy are computed as follows:

$$p_{\text{clean}} = \Pr(y\Sigma_{i=2}^{d+1} x_i > 0) = \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > 0), \tag{4.12}$$

$$\begin{aligned} p_{\text{adv}} &= \Pr(y\Sigma_{i=2}^{d+1}(x_i - \epsilon_{\max} y) > 0) \\ &= \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > \epsilon_{\max}\sqrt{d}). \end{aligned} \tag{4.13}$$

A selective classifier, however, rejects the sample if it is too close to the decision boundary. The standard and adversarial accuracy of selective classifier are obtained as follows:

$$p^{s}_{\text{clean}} = \Pr(y\Sigma_{i=2}^{d+1} w x_i > h) = \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > \frac{h}{w\sqrt{d}}), \tag{4.14}$$

$$\begin{aligned} p^{s}_{\text{adv}} &= \Pr(y\Sigma_{i=2}^{d+1} w(x_i - \epsilon_{\max} y) > -h) \\ &= \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > \epsilon_{\max}\sqrt{d} - \frac{h}{w\sqrt{d}}). \end{aligned} \tag{4.15}$$

The threshold, h, provides a trade-off between accuracy and robustness. Setting $h = 0$ yields the normal classifier and, with $h = dw\epsilon_{\max}$, no adversarial input is misclassified because of adversary's perturbations. The following theorem determines the optimal threshold for any $\lambda \in [0, 1]$.

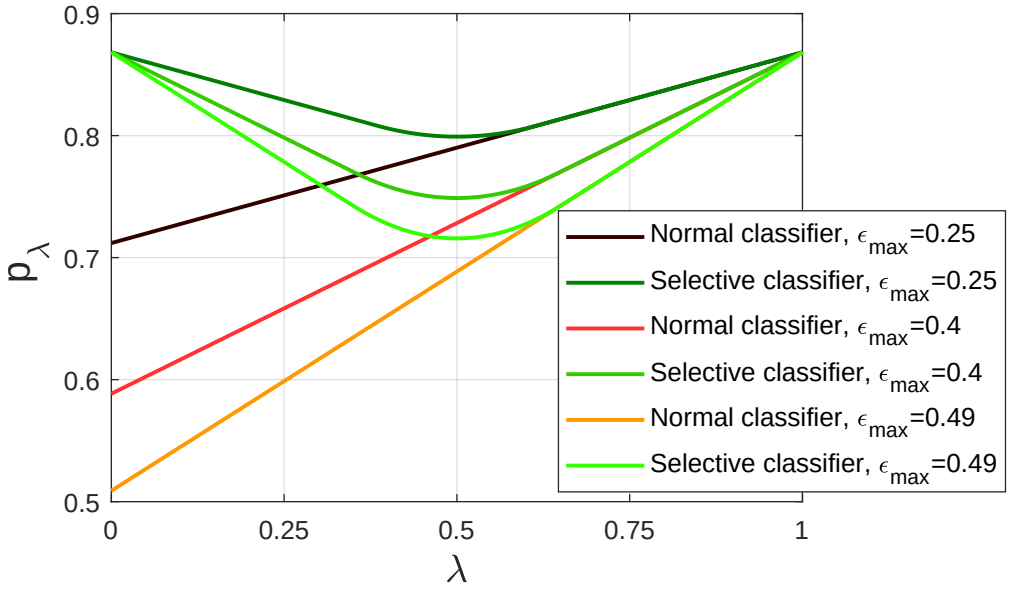


Figure 4.1: Value of $p_\lambda = \lambda p_{\text{clean}} + (1-\lambda)p_{\text{adv}}$ versus λ for the dataset defined in (4.6) assuming $p = 0.5$. Results are shown for dataset parameters $\eta = 0.5$ and $d = 5$ and different $\epsilon_{\max}$. The advantage of selective classifier over normal classifier increases with more powerful adversary, i.e. with smaller λ and larger $\epsilon_{\max}$.

Theorem 1. *Consider the dataset defined in (4.6) with $p = 0.5$. Let $p_\lambda = \lambda p_{clean} + (1-\lambda)p_{adv}$. Let $h = \alpha dw\epsilon_{\max}, \alpha \in [0,1]$. The optimal threshold for maximizing p_λ for the selective classifier defined in (4.9) is obtained by setting α as:*

$$\alpha^* = clip\Big(\frac{1}{2} + \frac{1}{d\epsilon_{\max}(2\eta - \epsilon_{\max})} \log\frac{1-\lambda}{\lambda}, 0, 1\Big). \tag{4.16}$$

Proof. Assume $p = 0.5$, thus we have $w_1 = 0$ in (4.7). Let $p_\lambda = \lambda p_{\text{clean}} + (1-\lambda)p_{\text{adv}}$. For

selective classifier, standard and adversarial accuracy are computed as follows:

$$\begin{aligned}
p_{\text{clean}} &= \Pr(y\Sigma_{i=2}^{d+1} w x_i > h) \\
&= \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > \frac{h}{w\sqrt{d}}) = Q(-\eta\sqrt{d} + \frac{h}{w\sqrt{d}}), \\
p_{\text{adv}} &= \Pr(y\Sigma_{i=2}^{d+1} w(x_i - \epsilon_{\max} y) > -h) \\
&= \Pr(\mathcal{N}(\eta\sqrt{d}, 1) > \epsilon_{\max}\sqrt{d} - \frac{h}{w\sqrt{d}}) \\
&= Q(-\eta\sqrt{d} + \epsilon_{\max}\sqrt{d} - \frac{h}{w\sqrt{d}}),
\end{aligned}$$

where $Q(\cdot)$ is the Q-function defined as $Q(z) = \Pr(Z > z)$ with Z being the standard normal distribution.

Let $h = \alpha d w \epsilon_{\max}, \alpha \in [0, 1]$. We have

$$\begin{aligned}
p_\lambda =& \lambda Q(-\eta\sqrt{d} + \alpha\epsilon_{\max}\sqrt{d}) \\
&+ (1-\lambda) Q(-\eta\sqrt{d} + (1-\alpha)\epsilon_{\max}\sqrt{d}).
\end{aligned}$$

Since $(-\eta\sqrt{d} + \alpha\epsilon_{\max}\sqrt{d}) < 0$ and $(-\eta\sqrt{d} + (1-\alpha)\epsilon_{\max}\sqrt{d}) < 0$ and Q-function is concave for negative inputs, the maximum of p_λ is obtained by setting $p'_\lambda = 0$ with respect to α. The derivative of Q function is computed as $Q'(z) = -cz'e^{-z^2/2}$, where $c = 1/\sqrt{2\pi}$. Therefore:

$$\begin{aligned}
p'_\lambda =& - c\lambda\epsilon_{\max}\sqrt{d}\, e^{-(-\eta\sqrt{d} + \alpha\epsilon_{\max}\sqrt{d})^2/2} \\
&+ c(1-\lambda)\epsilon_{\max}\sqrt{d}\, e^{-(-\eta\sqrt{d} + (1-\alpha)\epsilon_{\max}\sqrt{d})^2/2}.
\end{aligned}$$

Setting $p'_\lambda = 0$, we obtain $\alpha^* = \frac{1}{d\epsilon_{\max}(2\eta - \epsilon_{\max})} \log\frac{1-\lambda}{\lambda} + \frac{1}{2}$. Since threshold is non-negative, we have $\alpha \geq 0$. We also have $h \leq dw\epsilon_{\max}$ and, hence, $\alpha \leq 1$. The reason is $h = dw\epsilon_{\max}$ is equal to the maximum contribution of adversarial perturbation, i.e., with $\alpha = 1$, no adversarial example is misclassified due to adversary's perturbation. Thus, α^* is clipped between 0 and 1. The proof of Theorem 1 is complete. ■

The theorem implies that for $\lambda \in [0, \lambda_0]$, where $\lambda_0 > 0.5$, the optimal threshold is positive and, hence, selective classifier strictly outperforms normal (and robust) classifiers for maximizing p_λ.

This case shows that in a setting where adversarial examples are inevitable, selective classification provides a trade-off between standard accuracy and robustness. Figure 4.1 shows p_λ versus λ for different dataset parameters and for normal and selective classifiers. As can be seen, selective classifier outperforms normal classifier and is more advantageous with more powerful adversary, i.e., with smaller λ and larger $\epsilon_{\max}$.

4.4.3 Case II: On Price of Robustness

Now assume $p \neq 0.5$. We define the attack success rate as the probability that adding adversarial perturbation causes a correctly classified clean sample to be misclassified. For the classifier defined in (4.7), the attack success rate, q, is computed as follows:

$$\begin{aligned} q &= \Pr(yz > 0 \text{ and } yz' < 0) \\ &= \Pr(yz > 0 \text{ and } yz < dw\epsilon_{\max}) \\ &= \Pr(w_1 x_1 y + w\sqrt{d}\mathcal{N}(\eta\sqrt{d}, 1) \in (0, dw\epsilon_{\max})). \end{aligned} \tag{4.17}$$

From (4.17), it can be seen that attack success rate is zero only if $w = 0$. In this case, the classifier discards all normally-distributed features and yields standard and adversarial accuracy of $\max(p, 1-p)$. In general, attack success rate decreases by decreasing w to zero, causing the contribution of non-robust features to diminish.

Similarly, the attack success rate in selective classifier is obtained as follows:

$$\begin{aligned} q &= \Pr(yz > h \text{ and } yz' < -h) \\ &= \Pr(yz > h \text{ and } yz < dw\epsilon_{\max} - h) \\ &= \Pr(w_1 x_1 y + w\sqrt{d}\mathcal{N}(\eta\sqrt{d}, 1) \in (h, dw\epsilon_{\max} - h)). \end{aligned} \tag{4.18}$$

In this case, regardless of w, attack success rate of zero can be achieved by setting $h = dw\epsilon_{\max}/2$. Hence, in contrast to robust classifier, in selective classifier no feature is discarded, indicating that the model uses some degree of representational power of all features. Selective classifier, however, might wrongly reject clean samples, thus causing standard accuracy to drop. In the following,

we show that selective classifier indeed yields better trade-off between accuracy and robustness. Assume $\lambda = 0.5$, i.e., we intend to maximize $p_{\text{clean}} + p_{\text{adv}}$. The following theorem determines the optimal threshold for the classifier.

Theorem 2. *The optimal threshold for the selective classifier defined in (4.9) for maximizing* $p_{clean} + p_{adv}$ *is* $h^* = d\epsilon_{\max}(\eta - \epsilon_{\max}/2)$.

Proof. Assume $p \neq 0.5$. We have

$$\begin{aligned} p_\lambda =& \lambda p_{\text{clean}} + (1-\lambda) p_{\text{adv}} \\ =& \lambda(p \cdot Q(m_1) + (1-p) \cdot Q(m_2)) \\ &+ (1-\lambda)(p \cdot Q(m_3) + (1-p) \cdot Q(m_4)), \end{aligned}$$

where

$$\begin{cases} m_1 = -w_1/(w\sqrt{d}) - \eta\sqrt{d} + \alpha\epsilon_{\max}\sqrt{d}, \\ m_2 = w_1/(w\sqrt{d}) - \eta\sqrt{d} + \alpha\epsilon_{\max}\sqrt{d}, \\ m_3 = -w_1/(w\sqrt{d}) - \eta\sqrt{d} + (1-\alpha)\epsilon_{\max}\sqrt{d}, \\ m_4 = w_1/(w\sqrt{d}) - \eta\sqrt{d} + (1-\alpha)\epsilon_{\max}\sqrt{d}. \end{cases}$$

The derivative of p_λ is computed as follows:

$$\begin{aligned} p'_\lambda =& \lambda(-pc\epsilon_{\max}\sqrt{d}e^{-m_1^2/2} - (1-p)c\epsilon_{\max}\sqrt{d}e^{-m_2^2/2}) \\ &+ (1-\lambda)(pc\epsilon_{\max}\sqrt{d}e^{-m_3^2/2} + (1-p)c\epsilon_{\max}\sqrt{d}e^{-m_4^2/2}). \end{aligned}$$

Setting $\lambda = 0.5$, we obtain

$$\begin{aligned} p'_\lambda =& 0.5(1-p)c\epsilon_{\max}\sqrt{d}e^{-m_1^2/2 + w_1} \\ &\cdot \left(e^{2w_1(0.5-\alpha)\epsilon_{\max}/w} + 1\right)\left(e^{(2\eta - \epsilon_{\max})(0.5-\alpha)\epsilon_{\max} d} - 1\right). \end{aligned}$$

Note that p_λ is symmetric around $\alpha = 0.5$. Also, $p'_\lambda > 0$ for $\alpha \in [0, 0.5)$ and $p'_\lambda = 0$ for $\alpha = 0.5$. Hence, $\alpha^* = 0.5$ maximizes p_λ for $\lambda = 0.5$. With $\lambda = 0.5$ and $\alpha = 0.5$, we obtain $h^* = d\epsilon_{\max}(\eta - \epsilon_{\max}/2)$ and the proof is complete. ■

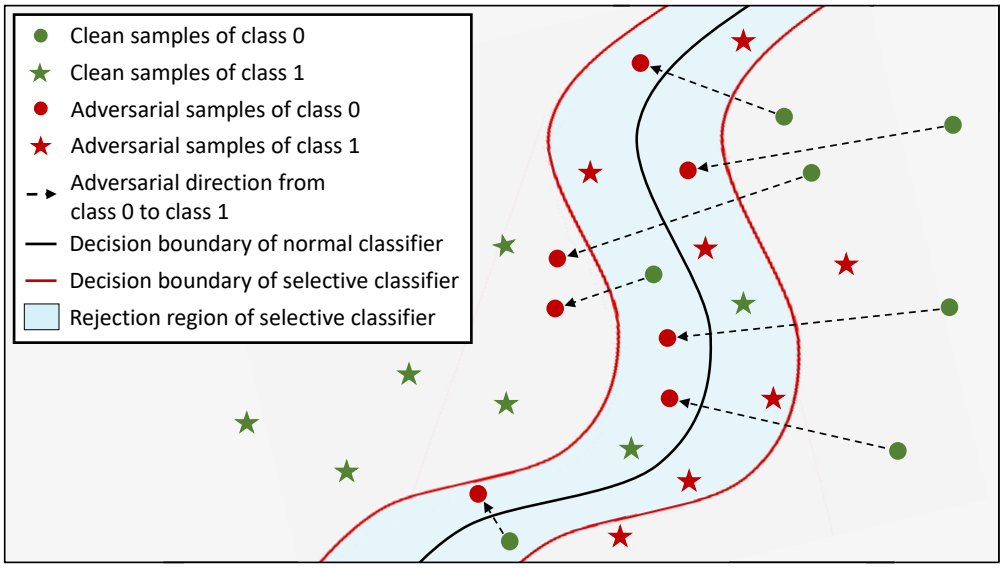


Figure 4.2: An illustration of decision boundaries of a binary classifier (only adversarial directions from class 0 to class 1 are shown). Selective classifier improves the trade-off between standard accuracy and robustness. The reason is adversarial examples are more likely to occur around the decision boundary compared to clean examples and, hence, are more likely to be rejected by the model.

Since $\eta > \epsilon_{\max}$, the optimal threshold is positive, indicating that selective classifier achieves better trade-off between standard and adversarial accuracy compared to any non-selective classifier. Figure 4.2 illustrates decision boundaries of normal and selective classifiers. Intuitively, adversarial examples are more likely to occur around the decision boundary compared to clean examples and, hence, are more likely to be discarded by the model. As a result, selective classifier increases the adversarial accuracy at the cost of decreasing standard accuracy by a smaller value.

4.5 Threat Model

We consider an attack against a selective classifier to be successful if a given image can be perturbed within the maximum allowed perturbation such that the perturbed image is classified to neither ground-truth label nor outlier. Therefore, the attacker's problem of (4.1) is modified as

follows:

$$\begin{aligned} &\text{given } X, \text{ find } X' \\ &\text{s.t. } \|X' - X\|_\infty \leq \epsilon_{\max}, \\ &\quad f(X') \neq y \text{ and } f(X') \neq \varnothing. \end{aligned} \tag{4.19}$$

We consider white-box and black-box attack settings described in the following.

White-Box Setting. The adversary knows training data, training procedure and weights of the target classifier.

Black-Box Setting. The adversary knows training data, model architecture and training procedure of the target classifier but, unlike the white-box setting, does not have any access to the trained model itself.

In black-box setting, the adversary relies on the transferability property of adversarial examples to attack the target classifier [166], i.e., she trains an independently initialized classifier with the same architecture, training data and training procedure as the target model. The attacker then generates adversarial examples on the source classifier and uses them on the target classifier [130]. Despite the weaker adversary model compared to the white-box setting, defending against black-box adversaries has remained a challenging task. Indeed, [171] has pointed out that "black-box security is a reasonable and more tractable goal for deployed ML models."

4.6 Proposed Method

Adversarial training (4.3) is shown to cause the model to gradually overfit to adversarial examples of training data and to not fully generalize to adversarial examples of the validation set [116]. Misclassified adversarial examples are, however, distributionally distinguishable from their corresponding clean samples (since they are classified differently by the model). Hence, during training, the model can be trained to classify currently misclassified adversarial examples into the outlier class. In order to maintain the accuracy on clean samples, the outlier probability is set to $1 - Z_y$, where Z_y is the model's confidence on true label. The new model will be, however, still vulnerable to new adversarial perturbations. Hence, we need to iteratively generate adversarial examples and

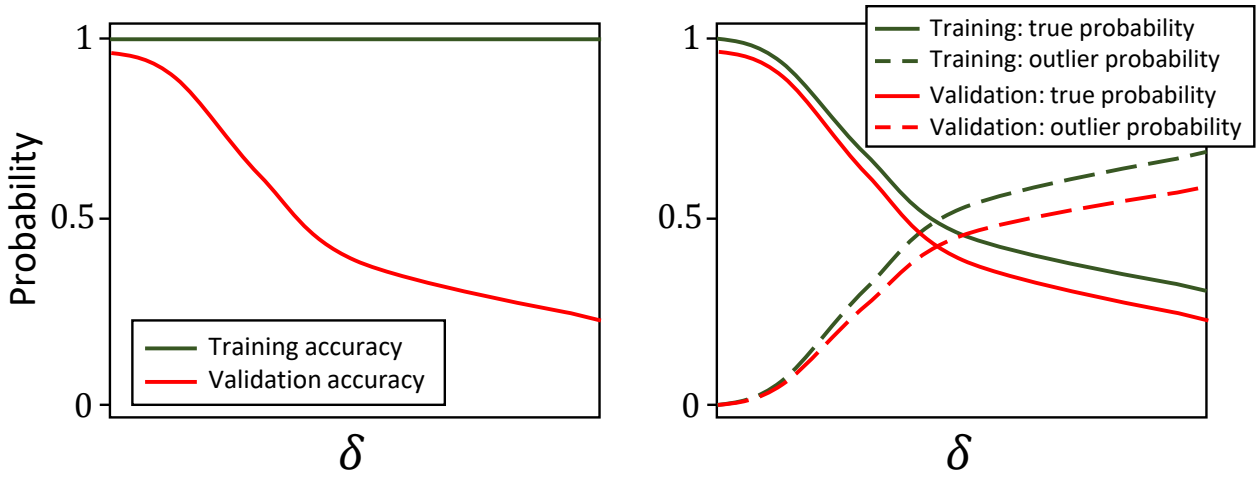


Figure 4.3: Illustration of accuracy versus perturbation size for models trained with adversarial training **(Left)** and selective classification **(Right)**. Adversarial training causes the model to overfit to adversarial examples of training data, i.e., accuracy on adversarial validation samples is low and decreases as perturbation increases. Selective classifier, however, is trained to split the probability between true and outlier classes such that it retains its confidence on the true label and assigns the rest of probability to the outlier class.

train the model to classify them into outlier class with appropriate probability. Figure 4.3 illustrates accuracy versus perturbation size for models trained with adversarial training and selective classification.

Choosing perturbations during training. Let $\delta = X - X'$. We train the model with clean images as well as adversarial examples with $\|\delta\|_\infty \sim U(0, \delta_{\max}]$, where the maximum perturbation size, $\delta_{\max}$, is set to a value greater than $\epsilon_{\max}$ to account for a slight distribution shift of test data. Assigning a nonzero outlier probability to adversarial examples regardless of the perturbation size results in sharp decision boundaries. That is, by adding a small noise to input, the model's prediction abruptly changes from the true label to outlier, causing a reduction in standard generalization. Hence, to smooth out the loss surface around the training data points, we train the model to classify examples with small perturbations, for which $\|\delta\|_\infty < \delta_{\text{mid}}, \delta_{\text{mid}} \in (0, \epsilon_{\max})$, into their true labels.

Generating adversarial examples. To solve (4.3), [116] generated adversarial examples based on the misclassification attack. With a model that is trained to classify perturbed images to an outlier class, this approach quickly results in examples that are misclassified by the model into that label. Such examples, however, are not adversarial, because the attacker intends to avoid both the true and outlier labels. Moreover, since the model already classifies such generated examples as outlier, retraining with them does not effectively update the parameters and the model remains vulnerable to attacks that take the outlier class into account.

To adapt the attack to selective classifier, we propose *selective misclassification attack*, in which the adversary maximizes the loss on both the true and outlier classes. The adversary's loss function is required to be zero if the input is classified into either the true label or outlier and, hence, is defined as the product of corresponding losses. Thus, the attack vector is obtained as follows:

$$V^j = \nabla_X(\ell(X^j, y) \cdot \ell(X^j, k)). \tag{4.20}$$

Training formulation. The training procedure of selective classification is formulated as the following alternative optimization problem:

$$\begin{cases} \delta_1^* = \operatorname{argmax}_{\delta \in \mathcal{S}_1} \ell(X+\delta, y; \theta^*) \cdot \ell(X+\delta, k; \theta^*), \\ \delta_2^* = \operatorname{argmax}_{\delta \in \mathcal{S}_2} \ell(X+\delta, y; \theta^*) \cdot \ell(X+\delta, k; \theta^*), \\ \theta^* = \operatorname{argmin}_\theta \mathbb{E}_{(X,y)\sim\mathcal{D}}[\lambda \ell(X, y; \theta) \\ \qquad + \frac{1-\lambda}{2}(\ell(X+\delta_1^*, y; \theta) + \ell(X+\delta_2^*, Z'; \theta))], \end{cases} \tag{4.21}$$

where $\mathcal{S}_1$ and $\mathcal{S}_2$ are the set of perturbations with $\|\delta\|_\infty \in [0, \delta_{\text{mid}}]$ and $\|\delta\|_\infty \in [\delta_{\text{mid}}, \delta_{\max}]$, respectively, and $\lambda \in [0, 1]$ balances the training on clean and adversarial samples. Also, Z' is a probability vector defined as $Z'_y = F(X + \delta_2^*)_y$, $Z'_k = 1 - Z'_y$ and $Z'_i = 0$ for $i \notin \{y, k\}$, where F is the model on which adversarial examples are generated.

Adversarial examples are generated using PGD attack with cross-entropy loss function. To prevent the model from overfitting to adversarial examples of training data, we apply the attack on randomly perturbed images with $\|\delta\|_\infty \in [0, \delta_{\text{mid}}]$. Adding more perturbation causes initial

random images to be classified as outlier, on which it is difficult to successfully apply the selective misclassification attack. Also, to speed up the training in early epochs, at each epoch we generate adversarial examples only on samples that are correctly classified by the model.

4.7 Experimental Results

In this section, we provide the experimental results on MNIST and CIFAR10 image datasets. We compare selective classifier with robust model trained using adversarial training approach [116], which is shown to achieve state-of-the-art adversarial accuracy [13]. We also provide the results on models that are trained only on clean data, which we refer to as normal models.

Training Setup. We use the same architecture for normal and selective classifiers. For CIFAR10 dataset, VGG16 network [155] is used. For MNIST dataset, we train a convolutional network with three convolutional layers with 16, 32 and 64 filters and one fully connected layer of size 128. Perturbation sizes are reported for pixel values in range of $[0, 1]$. Adversarial examples are generated using PGD attack with maximum L_∞ perturbation of $\epsilon_{\max}$. Following [116], we use $\epsilon_{\max} = 0.3 = 76.5/255$ for MNIST and $\epsilon_{\max} = 8/255$ for CIFAR10. For training the selective classifier on MNIST, we set $\delta_{\text{mid}} = 10/255$, $\delta_{\max} = 80/255$, $\epsilon_{\text{step}} = 1/255$ and $n = 4 \cdot \|\delta\|_\infty / \epsilon_{\text{step}}$. For CIFAR10, we set $\delta_{\text{mid}} = 4/255$, $\delta_{\max} = 10/255$, $\epsilon_{\text{step}} = 1/255$ and $n = 2 \cdot \|\delta\|_\infty / \epsilon_{\text{step}}$.

We set a target accuracy on standard validation data and output the model for which standard accuracy is greater than the target value and adversarial accuracy on validation data is highest. The target validation accuracy of selective classifier is set to 99.0% for MNIST and 80.0% for CIFAR10 dataset. Table 4.1 shows the test accuracy of normal, robust and selective models. Selective classification causes a reduction in accuracy with respect to normal training, but performs better compared to adversarial training.

Test Setup. We generate adversarial examples on test data using selective misclassification and targeted attacks. In white-box setting, for each sample, we first inspect the classifier's output and then add the perturbation only if it is correctly classified. We follow the same approach in black-box setting as well, i.e., clean examples that are wrongly classified by the source model are transferred to the target model without adding any perturbation. In experiments, we observed

Table 4.1: Standard and adversarial accuracy of different classifiers on MNIST and CIFAR10 datasets. Maximum perturbation size for adversarial examples is set to 0.3 for MNIST and $8/255$ for CIFAR10. Bold numbers indicate the best performing classifiers.

Dataset	**Training Method**	**Model**	**Standard Accuracy**	**White-box Accuracy**	**Black-box Accuracy**
MNIST	Normal Training	ConvNet	$\mathbf{99.4\%}$	0.0%	8.7%
	Adversarial Training	ConvNet [116]	98.5%	92.0%	97.5%
	Selective Classification	ConvNet	99.3%	$\mathbf{99.2\%}$	$\mathbf{99.6\%}$
CIFAR10	Normal Training	VGG	$\mathbf{92.5\%}$	0.0%	4.6%
	Adversarial Training	Resnet [116]	79.4%	43.7%	57.7%
	Selective Classification	VGG	81.0%	$\mathbf{52.4\%}$	$\mathbf{94.7\%}$

that such examples are indeed more likely to fool the target classifier when not distorted. Also, in targeted attack, although the attacker intends to cause the model to output a specific label, we consider the attack to be successful if the perturbed example is classified to any label other than the true label or outlier.

The models are tested against PGD attack with cross-entropy and CW loss functions [37]. As stated in Section 4.6, we found that setting $\mathrm{random\_start} = \mathrm{True}$ causes adversarial examples to be classified as outlier. Hence, we present the results only for the case of $\mathrm{random\_start} = \mathrm{False}$. For misclassification attack on selective classifier, the CW loss function is modified to minimize the average of logits of true and outlier classes, while maximizing the largest value of other logits. Table 4.1 summarizes attack results on different models in white-box and black-box settings.

Table 4.2 provides detailed results of the PGD attack on selective classifier.

Results on MNIST. In white-box setting, the model achieves state-of-the-art adversarial accuracy of 99.2% against the strongest PGD attack with parameters $(\epsilon_{\max}, \epsilon_{\text{step}}, n) = (0.3, 0.01, 10000)$. In comparison, robust classifier gives 92.0% adversarial accuracy against PGD attack [116]. For black-box attack, we transferred adversarial examples from a normal and an independently initialized classifier with the same architecture and training method as the target model. Adversarial accuracy of selective classifier on samples that are generated on normal model is 100% regardless of attack parameters. For samples that are transferred from a selective model, the classifier achieves state-of-the-art adversarial accuracy of 99.6% against the best attack.

Results on CIFAR10. In case of CIFAR10 dataset, before inference we add a small noise to images as follows. Given an image X, we obtain the noisy image as $X' = X + \delta \cdot V$, where V is a random vector which takes values of $\{-1, +1\}$ with equal probability, i.e., $V_i \sim \text{Rademacher}(0.5)$. The value of δ is set to the largest value for which validation accuracy is greater than the target value. Since the model is trained to be robust against small perturbations, adding noise only slightly reduces the standard accuracy, but increases adversarial accuracy by a larger value and, hence, improves the trade-off of accuracy and robustness. We also observed that adding noise at each step of PGD attack reduces the attack success rate. Thus, we run the attacks without noise.

In white-box setting, selective classifier achieves state-of-the-art adversarial accuracy of 52.4% against the strongest PGD attack with parameters $(\epsilon_{\max}, \epsilon_{\text{step}}, n) = (8/255, 0.01/255, 10000)$. In comparison, a VGG network trained with adversarial training is reported to achieve 80.3% and 37.0% standard and adversarial accuracy, respectively [114], and a robust Resnet model achieves standard and adversarial accuracy of 79.4% and 43.7%, respectively. [116] also applied adversarial training on a wider version of Resnet, in which layer widths were increased by a factor of 10, and achieved 87.3% standard accuracy and 45.8% adversarial accuracy. The results show that, when comparing models with similar capacity, selective classifier provides better standard and adversarial accuracy compared to robust models.

In black-box setting, similar to MNIST, the accuracy of selective classifier on examples that are transferred from a normal model is 100% regardless of attack parameters. The classifier also

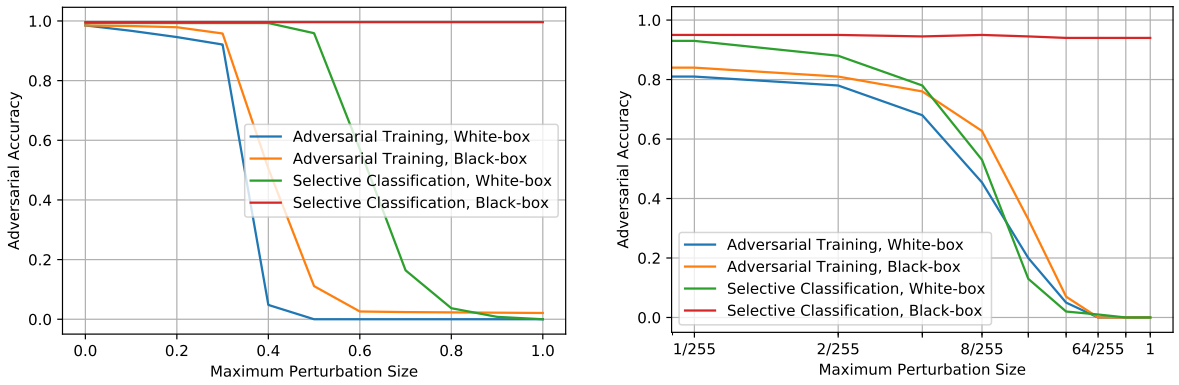


Figure 4.4: Adversarial accuracy versus maximum perturbation size on MNIST **(Left)** and CIFAR10 **(Right)** datasets.

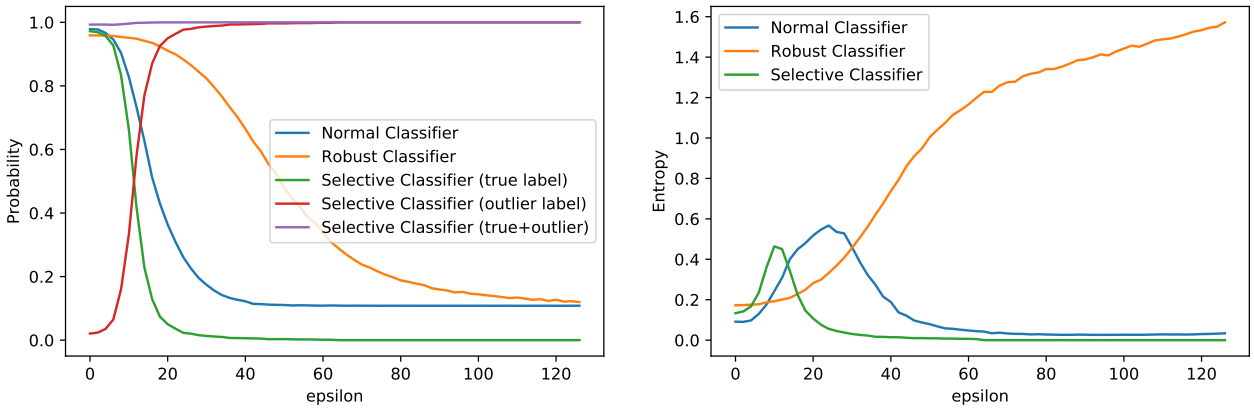
achieves state-of-the-art accuracy of 94.7% on examples that are generated on an independently initialized selective model. The results show that, in both MNIST and CIFAR10 datasets, the black-box attack success rate is almost zero and the small misclassification rate is mostly due to the model's error on clean data.

Investigating Gradient Masking. [130] noted that some defenses break gradient-based attacks by causing the gradients to not to be "useful". [13] discussed some characteristic behaviors of such defenses. In the following, we address each point and argue that the robustness provided by our method is not due to gradient masking.

- **One-step versus iterative attacks.** We ran PGD with different number of steps and verified that iterative attacks significantly outperform one-step attacks.

- **Attacks with large distortion.** Figure 4.4 shows adversarial accuracy versus maximum perturbation size for MNIST and CIFAR10 datasets. As can be seen, in both adversarial training and selective classification methods, white-box adversarial accuracy monotonically decreases with larger perturbation and tends to zero when the maximum perturbation size is increased to 1.

- **Black-box versus white-box setting.** As can be seen in Figure 4.4, attacks in white-box



(a) (b)

Figure 4.5: Statistics of output probability vector on noisy images versus L_∞ norm of noise, ϵ. (a) average probability of true (and outlier) labels, (b) average entropy of output probability vector.

setting are consistently better than those in black-box setting. In fact, for both MNIST and CIFAR10 datasets, our method maintains high adversarial accuracy in black-box setting even with large perturbation sizes. With perturbation size of 1, any image can be transformed into any other image (though, such an image is not adversarial, since the ground-truth label has changed). Hence, no model can achieve better accuracy than 0%. We, however, observed that generated images contain artificial perturbations that do not resemble clean examples and, hence, are classified as outlier by the target classifier.

- **Random attack.** We examine the selective classifier against random attack as follows. For each example, we test the model with 10000 randomly perturbed images and consider the defense to be successful if all perturbed images are classified into either the true label or outlier. For both MNIST and CIFAR10 datasets, the models achieve 100% adversarial accuracy against the random attack, i.e., none of the random perturbations of test samples could fool the selective classifier.

4.8 Analyzing Adversarial Examples of Selective Classifier

In this section, we empirically analyze the behavior of different models on random and adversarial perturbations. We provide the results on normal, robust and selective classifiers trained on CIFAR10 dataset.

Analyzing behavior of classifiers against noise. Given an image X, we generate noisy image $X' = X + \epsilon \cdot V$ for different values of ϵ, where V is a random variable which takes values of $\{-1, +1\}$ with equal probability, i.e., $V \sim \text{Rademacher}(0.5)$. We have $\|X' - X\|_\infty = \epsilon$. Figure 4.5a shows the average confidence of different classifiers on true label, $\mathbb{E}_{X,V}[Z(X')_y]$, versus ϵ. For selective classifier, we also plot the average confidence on the true and outlier labels combined, i.e., $\mathbb{E}_{X,V}[Z(X')_y + Z(X')_k]$. Figure 4.5b shows the average entropy of output probability vector, $\mathbb{E}_{X,V}[H(Z(X'))]$, versus ϵ.

In normal model, by increasing the perturbation size, the confidence in true label gradually decreases. The entropy, however, first increases and then decreases, implying that the probability is transiting from correct label to a wrong label and the model eventually classifies the noisy sample into another label with high confidence. Showing high confidence on wrong labels for invalid samples is a known failure of deep learning algorithms [125].

The robust model is less confident on clean samples and more robust to noise compared to the normal model. Moreover, by increasing the perturbation size, the entropy increases, indicating that the model spreads the probability across all classes. Showing less confidence on clean samples is analogous to label smoothing method [165] which is shown to improve robustness to simple attacks [183]. Also, spreading probability to all classes is a feature of models with the so-called "rubbish class," where the classifier is trained to assign a uniform distribution to invalid data and discards samples with low confidence at test time [27]. These two observations might provide insights into designing alternative approaches for adversarial training.

In selective classifier, by adding a small perturbation, the model's confidence on true label drops to zero and the outlier probability increases to 1. Thus, the entropy first increases and then decreases (to zero). Moreover, the sum of probabilities of true and outlier labels is always near

1. The results show that selective classifier rejects noisy images, while maintaining high accuracy on clean samples. Similar pattern is observed when adding noise to adversarial examples, i.e., selective classifier rejects noisy adversarial examples even when noise magnitude is small. The sensitivity to noise enables the classifier to reject adversarial examples transfered from other models, since such examples can be thought to be slightly perturbed versions of the true adversarial examples generated on target model itself.

Visualizing Loss Surface. Let R_1 and R_2 be the directions of adversarial and random perturbations of a data point X, i.e., $R_1 = X' - X$ and $R_2 = \epsilon V$, where $V \sim$ Rademacher(0.5) and $\epsilon = \mathbb{E}(|R_1|)$. Let $W_{\alpha,\beta} = X + \alpha \cdot R_1 + \beta \cdot R_2$, for $\alpha, \beta \in [-2, 2]$. We obtain the output probability vector, $Z(W_{\alpha,\beta})$, and compute the probability of the true label, $Z(W_{\alpha,\beta})_y$, as well as the attack success probability defined as $1 - Z(W_{\alpha,\beta})_y - Z(W_{\alpha,\beta})_k$.

Figure 4.6(a-b) shows the results for one of CIFAR10 test images that the model classifies correctly and an adversarial example can be obtained. As can be seen, the probability of true label decreases when α approaches 1. Around the adversarial point with $\alpha = 1$ and $\beta = 0$, attack success probability is high. However, by slightly moving along the random or adversarial directions, attack success probability drops to zero.

Now, we evaluate the target classifier on data points $W_{\alpha,\beta}$, obtained from the source model. Figure 4.6(c-d) shows the results. As can be seen, the true probability decreases similarly along random and adversarial directions, implying that the adversarial direction of source classifier looks like a random direction to the target classifier. Also, the attack success probability is almost zero at points around the adversarial example, meaning that the adversarial example of the source classifier fails to transfer to the target classifier.

4.9 Related Work

Robustness of classifiers to adversarial examples has been studied from theoretical perspective. [54] derived an upper bound on robustness to adversarial perturbations. [53, 146, 46, 117] showed that under certain conditions on data distribution, any classifier is vulnerable to adversarial perturbations. [162] empirically observed a trade-off between accuracy and robustness of several

ImageNet models. [173] showed this fundamental trade-off on an example dataset. We further extended the example setting and proved that selective classification improves the trade-off of accuracy and robustness.

Detecting out-of-distribution examples has been explored in various contexts [144, 20, 75, 108, 60]. These methods, however, are not designed for adversarial settings. Several methods for detecting adversarial examples have been proposed [190, 56, 67, 119, 110, 136, 115], but later broken [35, 74, 13]. A recent competition on defending against adversarial examples allows classifiers to "abstain" from classification [28].

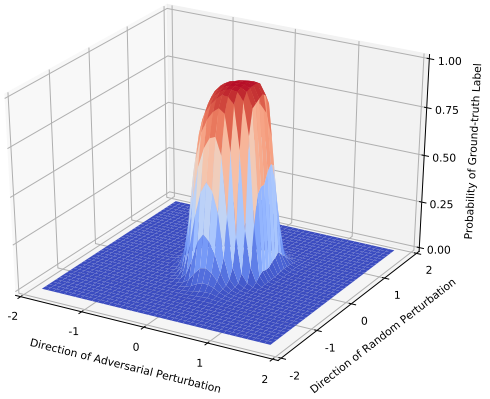
Adversarial training is shown to be an effective approach for improving robustness of classifiers [65]. [100] provided observations on adversarial training on ImageNet. [171] showed ensemble adversarial training improves robustness in black-box setting. [116] formulated adversarial training as a minmax optimization problem and showed it can successfully increase model robustness. Several other defenses have been broken with adaptive iterative attacks [37, 74, 35, 36, 174, 13, 12, 49]. In this paper, we adapted adversarial training for selective classification and showed it increases robustness. Recent works improved adversarial training using spectral normalization [52] and Bayesian neural networks [114]. These approaches are orthogonal to ours and can be incorporated in the selective classification framework as well.

4.10 Conclusion

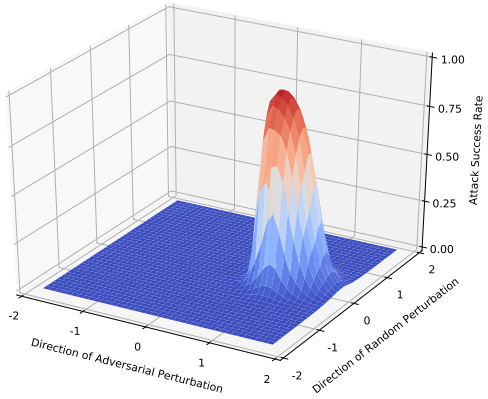
We analyzed selective classification framework both in theory and practice and showed that it improves the trade-off between standard accuracy and adversarial robustness. Specifically, we first investigated a binary classification problem and proved that rejecting low-confidence examples increases the robustness. We then proposed a model that classifies adversarial examples into either their true label or outlier. The experimental results on MNIST and CIFAR10 datasets against powerful iterative attacks show that the proposed method achieves state-of-the-art accuracy in both white-box and black-box attack settings.

Table 4.2: Adversarial accuracy of selective classifiers trained on MNIST and CIFAR10 datasets in white-box and black-box settings and for misclassification and targeted attacks. xent stands for cross-entropy loss function. CW is the loss function proposed by [37]. Bold numbers indicate the best performing attacks.

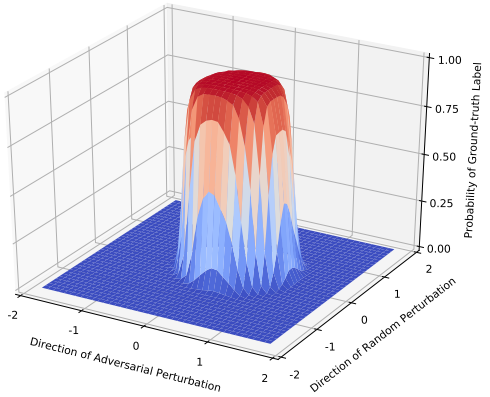
Dataset	**PGD Attack Parameters: Loss type**, $(\epsilon_{\max}, \epsilon_{\text{step}}, n)$	**Adversarial Accuracy**			
		White-box Setting		**Black-box Setting**	
		Misclassification	**Targeted**	**Misclassification**	**Targeted**
MNIST	xent, $(0.3, 0.01, 100)$	99.3%	99.8%	99.7%	99.9%
	xent, $(0.3, 0.01, 1e3)$	99.3%	99.8%	99.7%	99.9%
	xent, $(0.3, 0.01, 1e4)$	99.3%	99.8%	99.7%	99.9%
	xent, $(0.3, 0.001, 1e4)$	$\mathbf{99.2}\%$	$\mathbf{99.7}\%$	$\mathbf{99.6}\%$	99.9%
	CW, $(0.3, 0.01, 1e3)$	99.3%	99.8%	99.7%	99.9%
	Random perturbation, $\epsilon_{\max} = 0.3$, 10000 Trials	NA	NA	100%	NA
CIFAR10	xent, $(8/255, 1/255, 100)$	85.9%	94.6%	96.6%	96.9%
	xent, $(8/255, 1/255, 1000)$	83.3%	94.2%	96.5%	96.9%
	xent, $(8/255, 1/255, 1e4)$	81.5%	93.1%	97.1%	97.2%
	xent, $(8/255, 0.1/255, 1000)$	58.5%	82.5%	$\mathbf{94.7}\%$	$\mathbf{95.1}\%$
	xent, $(8/255, 0.1/255, 1e4)$	55.6%	$\mathbf{77.9}\%$	95.7%	96.6%
	xent, $(8/255, 0.01/255, 1e4)$	$\mathbf{52.4}\%$	78.5%	96.0%	96.9%
	CW, $(8/255, 1/255, 100)$	94.3%	95.1%	97.2%	97.5%
	CW, $(8/255, 1/255, 1000)$	94.1%	94.2%	97.5%	97.9%
	CW, $(8/255, 1/255, 1e4)$	91.6%	93.7%	97.6%	97.8%
	CW, $(8/255, 0.1/255, 1000)$	60.4%	83.3%	94.9%	95.7%
	CW, $(8/255, 0.1/255, 1e4)$	58.7%	81.9%	96.3%	97.5%
	CW, $(8/255, 0.01/255, 1e4)$	53.6%	78.1%	97.0%	97.6%
	Random perturbation, $\epsilon_{\max} = 8/255$, 10000 Trials	NA	NA	100%	NA



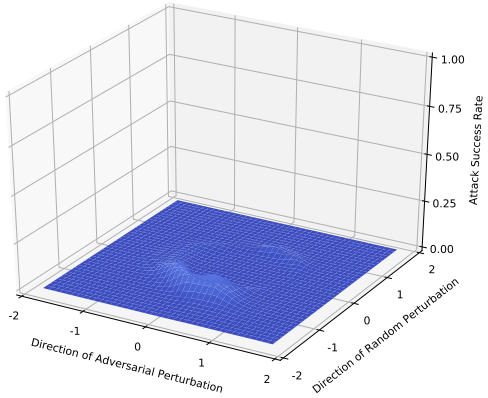
(a) Ground-truth probability of source model.



(b) Attack success probability on source model.



(c) Ground-truth probability of target model.



(d) Attack success probability on target model.

Figure 4.6: Visualization of ground-truth probability as well as attack success probability, defined as $1 - Z_y - Z_k$, on $X + \alpha \cdot R_1 + \beta \cdot R_2, \alpha, \beta \in [-2, 2]$. $R_1 = X' - X$ is the direction of adversarial perturbations for the source classifier and $R_2 = \epsilon V$ is a random direction, where $V \sim$ Rademacher(0.5) and $\epsilon = \mathbb{E}(|R_1|)$. For the source classifier, attack success probability is high for $\alpha = 1$ and $\beta = 0$, and drops to zero by slightly moving along the random or adversarial directions. For target classifier, the true probability drops similarly along random and adversarial directions, implying that adversarial direction of source classifier looks like a random direction to the target classifier. Also, attack success probability is almost zero at all points, meaning that adversarial example of source classifier fails to transfer to target classifier.

BIBLIOGRAPHY

[1] http://venturebeat.com/2015/04/08/snapchat-research, 2015.

[2] goo.gl/8q5CGb, 2017.

[3] http://www.theverge.com/2017/3/8/14857126/google-cloud-video-api-ai-machine-learning, 2017.

[4] goo.gl/7ckrNP, 2017.

[5] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[6] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[7] Aishy Amer, Amar Mitiche, and Eric Dubois. Reliable and fast structure-oriented video noise estimation. In *Image Processing. Proceedings. 2002 International Conference on*, volume 1. IEEE, 2002.

[8] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[9] Evlampios Apostolidis and Vasileios Mezaris. Fast shot segmentation combining global and local visual descriptors. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6583–6587. IEEE, 2014.

[10] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242, 2017.

[11] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[12] Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

[13] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[14] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[15] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[16] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[17] Peter GJ Barten. *Contrast sensitivity of the human eye and its effects on image quality*, volume 72. SPIE press, 1999.

[18] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[19] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902, 2015.

[20] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[21] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[22] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, 2014.

[23] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[24] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

[25] Twitter blog. Increasing our investment in machine learning. `https://blog.twitter.com/official/en_us/a/2016/increasing-our-investment-in-machine-learning.html`, 2016.

[26] Damian Borth, Adrian Ulges, Christian Schulze, and Thomas M Breuel. Keyframe extraction for video tagging & summarization. In *Informatiktage*, volume 2008, pages 45–48, 2008.

[27] Jane Bromley and John S Denker. Improving rejection performance on handwritten digits by training with rubbish, 1993.

[28] Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352*, 2018.

[29] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[30] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018.

[31] Y Bulatov. Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html*, 2011.

[32] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[33] Stuart K Card, Allen Newell, and Thomas P Moran. The psychology of human-computer interaction. 1983.

[34] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16), Austin, TX*, 2016.

[35] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[36] Nicholas Carlini and David Wagner. Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017.

[37] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[38] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers. *arXiv preprint arXiv:1704.08082*, 2017.

[39] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[40] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[41] Google Cloud and YouTube-8M Video Understanding Challenge. `https://www.kaggle.com/c/youtube8m`, 2017.

[42] Costas Cotsaces, Nikos Nikolaidis, and Ioannis Pitas. Video shot detection and condensed representation. a review. *IEEE signal processing magazine*, 23(2):28–37, 2006.

[43] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[44] Antoine Del Cul, Sylvain Baillet, and Stanislas Dehaene. Brain dynamics underlying the nonlinear threshold for access to consciousness. *PLoS biology*, 5(10):e260, 2007.

[45] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[46] Elvis Dohmatob. Limitations of adversarial robustness: strong no free lunch theorem. *arXiv preprint arXiv:1810.04065*, 2018.

[47] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[48] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[49] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*, 2018.

[50] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[51] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[52] Farzan Farnia, Jesse M Zhang, and David Tse. Generalizable adversarial training via spectral normalization. *arXiv preprint arXiv:1811.07457*, 2018.

[53] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

[54] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

[55] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016.

[56] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[57] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[58] Yarin Gal and Lewis Smith. Idealised bayesian neural networks cannot have adversarial examples: Theoretical and empirical study. *arXiv preprint arXiv:1806.00667*, 2018.

[59] Ruth Ellen Galper. Recognition of faces in photographic negative. *Psychonomic Science*, 19(4):207–208, 1970.

[60] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4878–4887, 2017.

[61] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[62] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[63] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, and Peter Abbeel. Attacking machine learning with adversarial examples. *Open AI Blog*, 2017. `https://blog.openai.com/adversarial-example-research/`.

[64] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[65] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[66] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 999–1006. IEEE, 2011.

[67] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[68] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[69] Nicolás Guil, Julio Bregáins, and Adriana Dapena. Computational intelligence in multimedia processing. *Advances in Computational Intelligence*, pages 520–527, 2011.

[70] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[71] Alan Hanjalic. Shot-boundary detection: Unraveled and resolved? *IEEE transactions on circuits and systems for video technology*, 12(2):90–105, 2002.

[72] Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John Fisher, and Lars Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[74] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv preprint arXiv:1706.04701*, 2017.

[75] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[76] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[77] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

[78] Hossein Hosseini, Farzad Hessar, and Farokh Marvasti. Real-time impulse noise suppression from images using an efficient weighted-average filtering. *IEEE Signal Processing Letters*, 22(8):1050–1054, 2015.

[79] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.

[80] Hossein Hosseini and Radha Poovendran. Deep neural networks do not recognize negative images. *arXiv preprint arXiv:1703.06857*, 2017.

[81] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. *arXiv preprint arXiv:1804.00499*, 2018.

[82] Hossein Hosseini, Baicen Xiao, Andrew Clark, and Radha Poovendran. Attacking automatic video analysis algorithms: A case study of google cloud video intelligence api. In *Proceedings of the 2017 on Multimedia Privacy and Security*, pages 21–32. ACM, 2017.

[83] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, and Radha Poovendran. On the limitation of convolutional neural networks in recognizing negative images. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pages 352–358. IEEE, 2017.

[84] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, and Radha Poovendran. Assessing shape bias property of convolutional neural networks. *arXiv preprint arXiv:1803.07739*, 2018.

[85] Hossein Hosseini, Baicen Xiao, and Radha Poovendran. Deceiving googles cloud video intelligence api built for summarizing videos. *arXiv preprint*, 2017.

[86] Hossein Hosseini, Baicen Xiao, and Radha Poovendran. Google's cloud vision api is not robust to noise. *arXiv preprint arXiv:1704.05051*, 2017.

[87] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 43–58. ACM, 2011.

[88] Cloud Video Intelligence. `https://cloud.google.com/video-intelligence`, 2017.

[89] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[90] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[91] Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

[92] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[93] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[94] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[95] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[96] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[97] Andreas Koschan and Mongi Abidi. *Digital color image processing*. John Wiley & Sons, 2008.

[98] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[99] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[100] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[101] Victor AF Lamme. Towards a true neural stance on consciousness. *Trends in cognitive sciences*, 10(11):494–501, 2006.

[102] Barbara Landau, Linda B Smith, and Susan S Jones. The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321, 1988.

[103] Pavel Laskov et al. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 197–211. IEEE, 2014.

[104] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[105] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[106] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[107] Chen-Yu Lee, Saining Xie, Patrick W Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 5, 2015.

[108] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[109] Fei-Fei Li. Announcing google cloud video intelligence api, and more cloud machine learning updates. `https://goo.gl/jeLXSd`, 2017.

[110] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *ICCV*, pages 5775–5783, 2017.

[111] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[112] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[113] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[114] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *arXiv preprint arXiv:1810.01279*, 2018.

[115] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[116] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[117] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *arXiv preprint arXiv:1809.03063*, 2018.

[118] Final Cut Pro 7 User Manual. `https://documentation.apple.com/en/finalcutpro/usermanual`, 2017.

[119] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[120] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[121] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

[122] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

[123] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*, 2017.

[124] Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015.

[125] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[126] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1029–1038, 2016.

[127] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[128] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[129] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[130] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[131] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[132] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[133] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3646–3653, 2014.

[134] Richard J Phillips. Why are faces hard to recognize in photographic negative? *Perception & Psychophysics*, 12(5):425–426, 1972.

[135] Mary C Potter, Brad Wyble, Carl Erick Hagmann, and Emily S McCourt. Detecting meaning in rsvp at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[136] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580, 2018.

[137] Joaquin Quinonero Candela. Powering facebook experiences with ai. `https://code.facebook.com/posts/804694409665581/powering-facebook-experiences-with-ai`, 2016.

[138] Ambrish Rawat, Martin Wistuba, and Maria-Irina Nicolae. Adversarial phenomenon in the eyes of bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

[139] Sravana Reddy, MA Wellesley, and Kevin Knight. Obfuscating gender in social media writing. In *Proceedings of the 1st Workshop on Natural Language Processing and Computational Social Science*, pages 17–26, 2016.

[140] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. *arXiv preprint arXiv:1706.08606*, 2017.

[141] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference*, pages 1–14. ACM, 2009.

[142] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[143] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. *Computer Vision–ECCV 2010*, pages 213–226, 2010.

[144] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2013.

[145] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324, 2014.

[146] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

[147] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2015.

[148] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[149] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[150] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[151] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321. ACM, 2015.

[152] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *arXiv preprint arXiv:1610.05820*, 2016.

[153] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *arXiv preprint arXiv:1802.01421*, 2018.

[154] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[156] Jonas Sjöberg and Lennart Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995.

[157] Alan F Smeaton, Paul Over, and Aiden R Doherty. Video shot boundary detection: Seven years of trecvid activity. *Computer Vision and Image Understanding*, 114(4):411–418, 2010.

[158] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

[159] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[160] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[161] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[162] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. *arXiv preprint arXiv:1808.01688*, 2018.

[163] Chen Sun, Sanketh Shetty, Rahul Sukthankar, and Ram Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pages 371–380. ACM, 2015.

[164] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[165] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[166] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[167] A Murat Tekalp. *Digital video processing*. Prentice Hall Press, 2015.

[168] Shaohua Teng, Wenwei Tan, and Wei Zhang. Cooperative shot boundary detection for video. *Computer Supported Cooperative Work in Design IV*, pages 99–110, 2008.

[169] Simon Thorpe, Denise Fize, and Catherine Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520, 1996.

[170] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[171] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[172] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[173] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). *arXiv preprint arXiv:1805.12152*, 2018.

[174] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

[175] Balakrishnan Varadarajan, George Toderici, Sudheendra Vijayanarasimhan, and Apostol Natsev. Efficient large scale video classification. *arXiv preprint arXiv:1505.06250*, 2015.

[176] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[177] Carl Vondrick, Deva Ramanan, and Donald Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. *Computer Vision–ECCV 2010*, pages 610–623, 2010.

[178] Meng Wang, Xian-Sheng Hua, Richang Hong, Jinhui Tang, Guo-Jun Qi, and Yan Song. Unified video annotation via multigraph learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):733–746, 2009.

[179] Meng Wang, Xian-Sheng Hua, Jinhui Tang, and Richang Hong. Beyond distance measurement: constructing neighborhood similarity for video annotation. *IEEE Transactions on Multimedia*, 11(3):465–476, 2009.

[180] Meng Wang, Bingbing Ni, Xian-Sheng Hua, and Tat-Seng Chua. Assistive tagging: A survey of multimedia tagging with human-computer joint exploration. *ACM Computing Surveys (CSUR)*, 44(4):25, 2012.

[181] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. *arXiv preprint arXiv:1706.03922*, 2017.

[182] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[183] David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, page 311, 2016.

[184] Industry Keynote with YouTube CEO Susan Wojcicki (VidCon 2015). `https://www.youtube.com/watch?v=O6JPxCBlBh8`, 2015.

[185] Hong Ren Wu and Kamisetty Ramamohan Rao. *Digital video image quality and perceptual coding*. CRC press, 2005.

[186] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. Multi-stream multi-class fusion of deep networks for video classification. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 791–800. ACM, 2016.

[187] Zuxuan Wu, Ting Yao, Yanwei Fu, and Yu-Gang Jiang. Deep learning for video classification and captioning. *arXiv preprint arXiv:1609.06782*, 2016.

[188] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, pages 1689–1698, 2015.

[189] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *arXiv preprint arXiv:1705.07809*, 2017.

[190] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[191] Weilin Xu, Yanjun Qi, and David Evans. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

[192] Zhennan Yan and Xiang Sean Zhou. How intelligent are convolutional neural networks? *arXiv preprint arXiv:1709.06126*, 2017.

[193] Yang Yang, Yi Yang, and Heng Tao Shen. Effective transfer tagging from image to video. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(2):14, 2013.

[194] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[195] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[196] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[197] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[198] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition*, pages 214–223, 2007.

[199] Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015.

[200] Zheng-Jun Zha, Meng Wang, Yan-Tao Zheng, Yi Yang, Richang Hong, and Tat-Seng Chua. Interactive video indexing with statistical active learning. *IEEE Transactions on Multimedia*, 14(1):17–27, 2012.

[201] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[202] Tianzhu Zhang, Changsheng Xu, Guangyu Zhu, Si Liu, and Hanqing Lu. A generic framework for video annotation via semi-supervised learning. *IEEE Transactions on Multimedia*, 14(4):1206–1219, 2012.

[203] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.