School of Information Technologies

Department of Computer Systems

Miguel Angel Chaparro Quiñones 184627IASM

Robustness Analysis of Deep Neural Networks for Computer Vision

Master's Thesis

Supervisor: Aleksei Tepljakov
Ph.D.
Research Scientist

TALLINN 2020

Declaration of Originality

Declaration: I hereby declare that this thesis, my original investigation and achievement, submitted for the Master's degree at Tallinn University of Technology, has not been submitted for any degree or examination.

Deklareerin, et käesolev diplomitöö, mis on minu iseseisva töö tulemus, on esitatud Tallinna Tehnikaülikooli magistrikraadi taotlemiseks ja selle alusel ei ole varem taotletud akadeemilist kraadi.

Miguel Angel Chaparro Quiñones

Date: May 20, 2020

Signature: .....

To my parents, which I owe everything for what I am today, and to my lovely wife for her unconditional love and support.

Abstract

Deep learning applications are achieving extraordinary results in different fields, mostly due to the rapid rate of innovation and the amount of research involved in this area. One fundamental aspect to consider when implementing deep learning techniques, particularly inside mission-critical fields, is to be able to guarantee the robustness of the models, meaning that there is a measurable and precise certainty that the application will behave as expected.

Robustness evaluation is an extensive and ongoing study with many research gaps, such as; determining the informativeness of a data set, defining which metrics should be used, and the lack of interpretability of the results obtained. Due to the vast amount of rapidly changing research around these topics, having a clear understating of state of the art regarding how to improve or measure the robustness of a model can be quite challenging.

For this thesis the author will do a thorough analysis of the current state of the art in robustness evaluation for deep neural networks with a focus in computer vision, this work includes deep learning background, current robustness methodologies, analysis of research gaps and concludes with an experiment that will illustrate a practical example and tools available to measure robustness.

The thesis is in English and contains 57 pages of text, 11 chapters, 23 figures, 9 tables.

Nomenclature

DNN	Deep Neural Network
NNs	Neural Network
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
ReLU	Rectified Linear Unit
CUDA	Compute Unified Device Architecture
x'	Adversarial instance
ε	Epsilon

Contents

List of Figures

List of Tables

1. Introduction

Deep Neural Networks (DNNs) remarkable success is notorious among different fields such as computer vision [15, 16] audio [17, 18] and natural language processing [19, 20].

However, starting with Szegedy et al. [21] many studies have demonstrated that deep neural networks are highly vulnerable to slightly perturbed inputs, while being imperceptible to the human vision, can induce specific and unintended behaviour. After this discovery, a vast amount of research is focused on ensuring the robustness of the models against these perturbations.

The analysis presented in this thesis is centred around image classification robustness, image classification is tightly related to human perception, as can be observed when a model misclassifies an image is an indication that its robust features are mapped to a class which according to the human vision is considered not to be the correct class. Robustness inside image classification domain demands to be treated differently from other domains since most domains do not care about what the human vision system considers, such as malware classification. This might be the key motive why understanding the robustness of a computer vision model is a complex endeavour.

2. Background

2.1. Biological Vision

Neural Networks neurons are inspired by biological ones inside our nervous system [2], a biological neuron has thousands of dendrites, dendrites receive signals from other neurons as witnessed in Figure 1, when this signal is passed along a dendrite it causes a slight change in the voltage difference between the cell's interior and its surroundings, some signals provoke a small positive variation in voltage and other a small negative variation. If the cumulative effect of these signals reaches a threshold, the neuron will fire an action potential away from its cell body conveyed by its axon, meaning that it will transmit a signal to other neurons in the network.

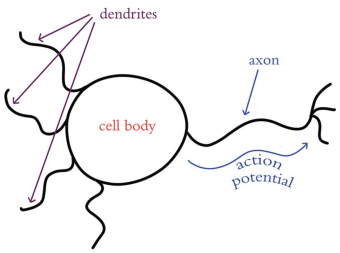

Figure 1. The anatomy of a biological neuron [1]

Some history

Five hundred fifty million years ago, the total number of species on the planet suffered an incremental explosion [2], there is evidence that this explosion of new species was driven by the development of light detectors in the trilobites, this is the first record of a primitive visual system, this system provided a large advantage compared to the other species such as the ability to facilitate the location of food and preys, the hypothesis concludes that the trilobite's prey, as well as its predators, had to evolve rapidly to survive.

After the trilobites, the complexity of the vision system has increased considerably; for instance, in mammals, a large proportion of the outer grey matter of the brain is involved in visual perception. In the 1950s at Johns Hopkins University, the physiologists David Hubel and Torsten Wiesel began executing pioneering research

on how visual information is processed in the cerebral cortex, the results of this research [22] contributed to their later awarded Nobel Prize. Hubel and Wiesel's methodology consisted of using a projector to display specific images to anaesthetized cats while simultaneously recording the activity of individual neurons connected to the primary visual cortex, which the first section that receives visual input from the eyes. The results exposed that straight edge lines produced an electric activation in the neurons, as can be witnessed in Figure 2 with this finding Hubel and Wiesel comprehended that the neurons that receive visual input from the eye are most responsive to simple straight-edged, which they denominated *simple neurons*.

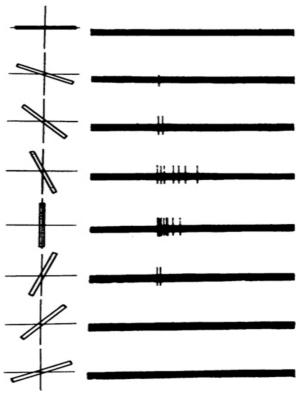

Figure 2. A simple cell in the visual cortex of a cat that fires at different rates depending on the orientation of the line shown to the cat. The line's orientation is shown in the left-hand column, while the right-hand column shows the electrical activity in the cell over one second [2]

Biological vision profoundly influences computer vision; computer vision tries to emulate human perception, which is a challenging task. Thus, by understanding the functioning of human perception, it might conduct to achieve the desired behaviour inside computer vision. The timeline milestones of both fields can be witnessed in Figure 3.

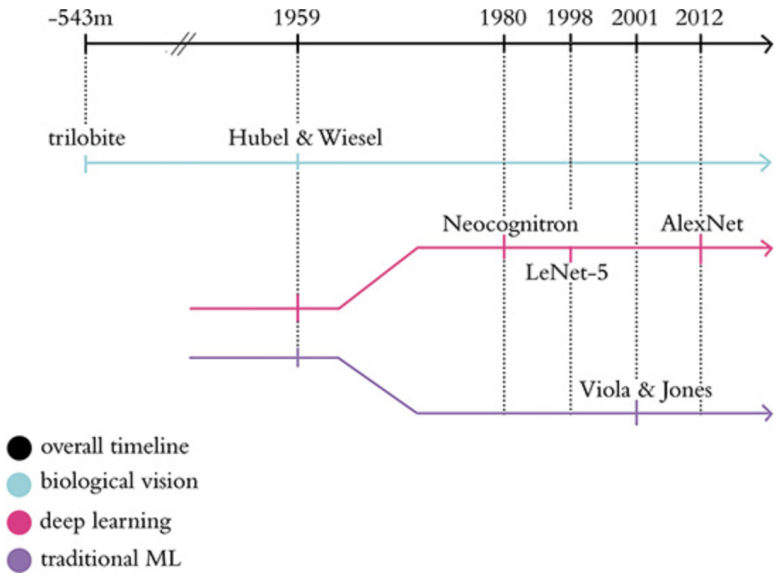

Figure 3. Timeline of biological and machine vision [1]

2.2. Deep Learning

Deep Learning is a subfield of machine learning; machine learning algorithms rely on *feature engineering* [23] which is a common practice used to extract desired features from the data, this process is a manual time consuming and labour-intensive task mainly when used for non-linear problems such as computer vision and natural language processing. This is one of the critical areas in which deep learning excels because algorithms inside deep learning can generate these features automatically [24].

Neural networks are the building block of deep learning systems, the most common type of architecture is feedforward neural networks that accept several inputs, computes a weighted sum, and applies a step function to obtain a final prediction as denoted in Figure 4.

Deep neural networks can be seen as stacked neural networks, resulting in a network composed of several layers.

Deep Neural Networks (DNNs), in contrast to traditional algorithms, do not have explicitly programmed steps, DNNs employs algorithms that learn from previous experience and common patterns observed to map an instance among a finite set of classes. These sort of algorithms are additionally referred to as a model.

$$y = \sum(weight * input) + bias$$

Figure 4. Neuron definition

The concept of Neural Networks is not a recent discovery, it dates back to the mid-1950s, the main reason deep learning research exploded in the recent years is due to the amount of data available, cost drop of computation resources and hardware advances, especially Graphical Processing Units GPUs performance and their increasing affordability.

GPUs excels in performing parallel computations, GPUs are generally used to perform millions of matrix operations per second to render polygons used in gaming. Madhavan et al. [25] indicated that training neural networks was based on performing numerous matrix operations, and introduced the idea of using these graphics cards to accelerate training, by this addition, deep neural networks architectures were feasible for the first time. Some common Deep Learning concepts can be observed in Table 1, it is essential to understand these topics to be able to proceed with the robustness evaluation of a model.

Deep Learning Concepts	
Loss Function	Measures the accuracy of the model, if the loss function is very small, the model is accurate.
Activation Function	The activation function determines if the neuron triggers or not. Traditionally the Sigmoid and Tanh functions have been used to train networks; however, since Hahnloser et al. [26], the ReLU function has been used more often. Currently, ReLU is by far the most popular activation function used in deep learning.
Optimizer	Ensures the neural network is performing fast and correctly by updating the biases and weights accordingly.
Gradient Descent	Optimization method to discover the minimum of a function.
Learning Rate	One of the most important parameters to tune for training a Neural Network represents the rate at which the model updates the network's weights.
Batch	Subset of the total data used for training, the data is divided in several groups of equal size.
Batch Normalization	Normalization technique to improve stability of the network, takes the mean and standard deviation of the activation layer and use those to normalize the activations. Additionally improves generalization.
Generalization	Refers to the ability of the model to infer predictions to new previously unseen data. High generalization is always desired.
Epoch	One iteration where the model sees the whole training set to update its weights, in other words, one complete forward and backward pass for all the training samples.
Dropout Rate	Normalization technique used to throw away activations at random, so that no activation can memorize any part of the input, it helps over-fitting.

Table 1. Additional Deep Learning concepts

Tuning a neural network can be a daunting task due to the vast amount of hyper-parameters available and the math surrounding them. It has been proven by empirical efforts, some default parameters that present excellent results, and most of the existing deep learning libraries already have these default values.

2.3. Model Architectures

The architecture of a model can be seen as the blueprints of the Deep Neural Network. It defines the internal structural components, such as the number of layers, loss and activation functions. Several commonly used architectures specialize in specific tasks.

Convolutional Neural Network (CNN) is a special type of Neural Network that has shown remarkable performance on image classification, they have been used on visual tasks since the late 1980s [27] the architectural design of the CNN was inspired by Hubel and Wiesel's work [22].

According to Jeremy Howard [28], selecting the architecture of the model should not be an important part of the Deep Learning process since there are many available architectures as can be witnessed in Table 2, these architectures will work on most common scenarios. Only for specific tasks, a finely crafted model should be required. This is one of the reasons the amount of Deep Learning applications are increasing since now is simpler to develop a functional model, most Deep Learning libraries such as TensorFlow and Pytorch already include utility methods to import these common architectures with a single line of code.

Table 2. Common architectures [14]

Name	Year	Parameters	Depth
LeNet	1998	0.060 M	5
AlexNet	2012	60 M	8
VGG	2014	138 M	19
GoogleNet	2015	4 M	22
Inception-V4	2016	35 M	70
ResNet	2016	25.6 M	152
DenseNet	2017	25.6 M	190

Each of these architectures can have different versions such as ResNet and ResNet-50, 50 refers to the total number of layers of this architecture, ResNet additionally has 18, 34, 101, and 152 versions.

By increasing the number of layers inside the architecture, the amount of time

required to train the model additionally increases, architectures with more layers are more prone to over-fitting when having limited amount of data and tend to be more accurate when using more data. Over-fitting is a problem of Machine Learning (ML), not just Deep Learning (DL), the problem refers when a function is too closely fit to a particular dataset, specific for DL it means that the model has memorized the data presented during the training process and is not able to generalize to new data, the accuracy is high on the trained data. However, it is low when new unseen data is presented.

Transfer learning

Transfer learning or pre-trained models is a commonly used technique that allows models to use the weights that have been trained on a different model, usually with a more complex architecture and bigger dataset such as ImageNet, which requires plenty of time and resources to train, this means that the model can use less data and train more accurate results in less time. One crucial aspect to consider before using pre-trained models is to observe the data similarity of both models, for instance, using MNIST pre-trained weights in a model that uses x-ray images may not provide any performance in the results.

Pre-train models usually remove the last layer, which is the layer that specializes in a particular classification task such as ImageNet classification and replacing it with one or multiple layers with randomized weights. The importance and deep understanding of how to use pre-trained models is ongoing research. He et al. [29] concludes that pre-trained models provide no performance benefits on various task and architectures, argues that both pre-trained and training from scratch results in models with similar accuracy. Hendrycks et al. [30] demonstrate that this is true only for unperturbed data, showing that pre-train models substantially improves the quality of various complementary model components such as the model *adversarial robustness* while training for longer on a clean dataset allow models without pre-train to display similar results, training for more epochs on a corrupted dataset conduct to the model deterioration.

2.4. Datasets

Data is the cornerstone of any deep learning application. It is of high importance, the anatomy of the data being used inside the model to interpret the results obtained.

When training a model, the data is required to be split among **training set**, **validation set** and optionally a **test set**, the accuracy of the model should be measured only on the validation set otherwise, the results are not valid. When the model does not have enough data and is trained for many epochs, the accuracy of the model tends to drop; due to over-fitting, some mitigation techniques for over-fitting include adding a dropout rate. This means randomly throwing away activations so that the model will not memorize any part of the inputs rate, increasing the amount of data by augmentation and reducing the complexity of the architecture.

ImageNet

ImageNet [31] dataset is one of the most popular datasets among Deep Learning research, is an extensive collection of human-annotated images, it has 14 million images spread across 22,000 categories, and it commenced with ILSRC (ImageNet Large Scale Visual Recognition Challenge) [32] in 2010.

The objective of the contest is to label and categorize an image among multiple classes. The method these classes were generated was the result of WordNet, an open-source word classification database using hierarchically synonyms.

ILSRC competition lasted for seven years (2010 - 2017) its primary goal was to promote the development of improved computer vision techniques and to benchmark state of the art. In the year 2011, ImageNet best results obtained 75% accuracy with no Neural Networks, in 2012 first place achieved 85% accuracy, the winning team was the only one that implemented Neural Networks as witnessed in Figure 5, the following years all the teams that had a top submission were using Neural Networks.

Deep Learning revolution is widely attributed to have its origin inside the ILSVRC challenge after the 2012 results.

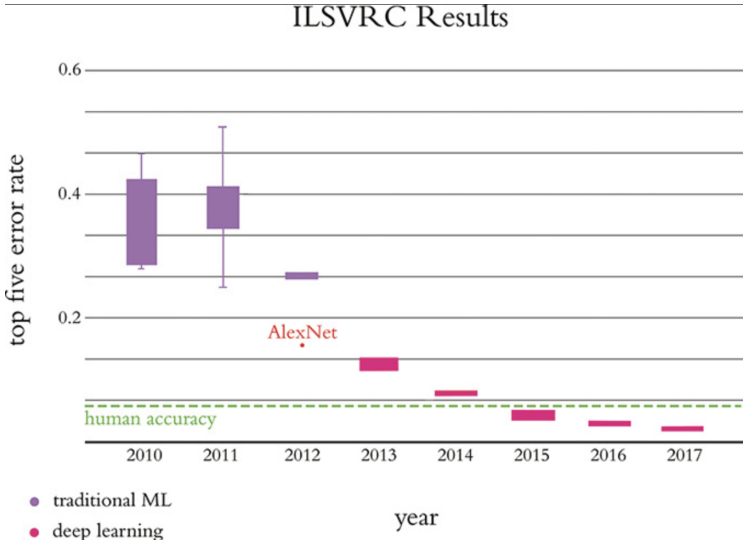

Figure 5. ILSVRC timeline milestones [2]

3. Computer Vision Robustness

Computer Vision Robustness is a broad, rapidly changing research, the number of scientific papers in this field is drastically increasing each year as it can be witnessed from Figure 6. To facilitate the comprehension of this field, the author created a diagram shown in Figure 7 that represents a high-level representation of the main areas.

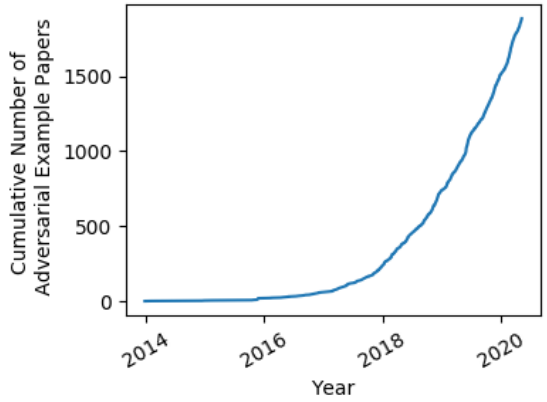

Figure 6. Cumulative number of adversarial papers [3]

Why adversarial instances exist?

Szegedy et al. [21] work, concluded that the main reason that the models can be fooled is due to the lack of generalization in low probability space of data, which may be caused by the high complexity of the deep neural network model structures. However, Goodfellow et al. [33] demonstrate that even linear models are vulnerable to adversarial attacks. Some additional insight can be gained by studying the decision boundary of the model's; the adversarial examples are, in most cases, close to the decision boundary, which may be because the boundary is too flat [34], curved [35], or inflexible [36]. Understanding why adversarial or corrupted instances are capable of provoke a model misclassification is a research gap that can guide us towards more robust models.

4. Model Interpretability

Interpretability refers to the ability to understand the reasoning behind a model, current state of Deep Neural Networks could be considered a *black box statistics model* [37] that can generate results based on patterns it learns through many iterations. Understanding the decisions that lead to those results is a difficult task that is becoming a hot topic for research [38, 39] Fan et al. [38] work states that the main reasons behind the complexity of interpretability are due to the **lack of human expertise** when handling an intricate problem such as pseudo-random predictions, **commercial cost,** there are strong motives for big corporations to hide their models to avoid being reverse-engineered and interpretability high financial cost that is required in terms of computational resources, real-world **data** has many levels of dimensions that are not easy to interpret. Finally, the **algorithms** complexity, Deep Learning relies highly on nonlinear algorithms and recursiveness, resulting in a non-convex optimization problem that is complicated to comprehend.

Several works around DNNs interpretability [4, 40, 41], provide rich visual representations of what the neurons across each layer perceives from the robust features, as can be observed in Figure 8, understanding how the DNNs interprets the results is of high importance in the field of robustness since the reason behind the neuron's activation to specific patterns or textures to provoke a decision boundary misclassification is the foundation of adversarial attacks.

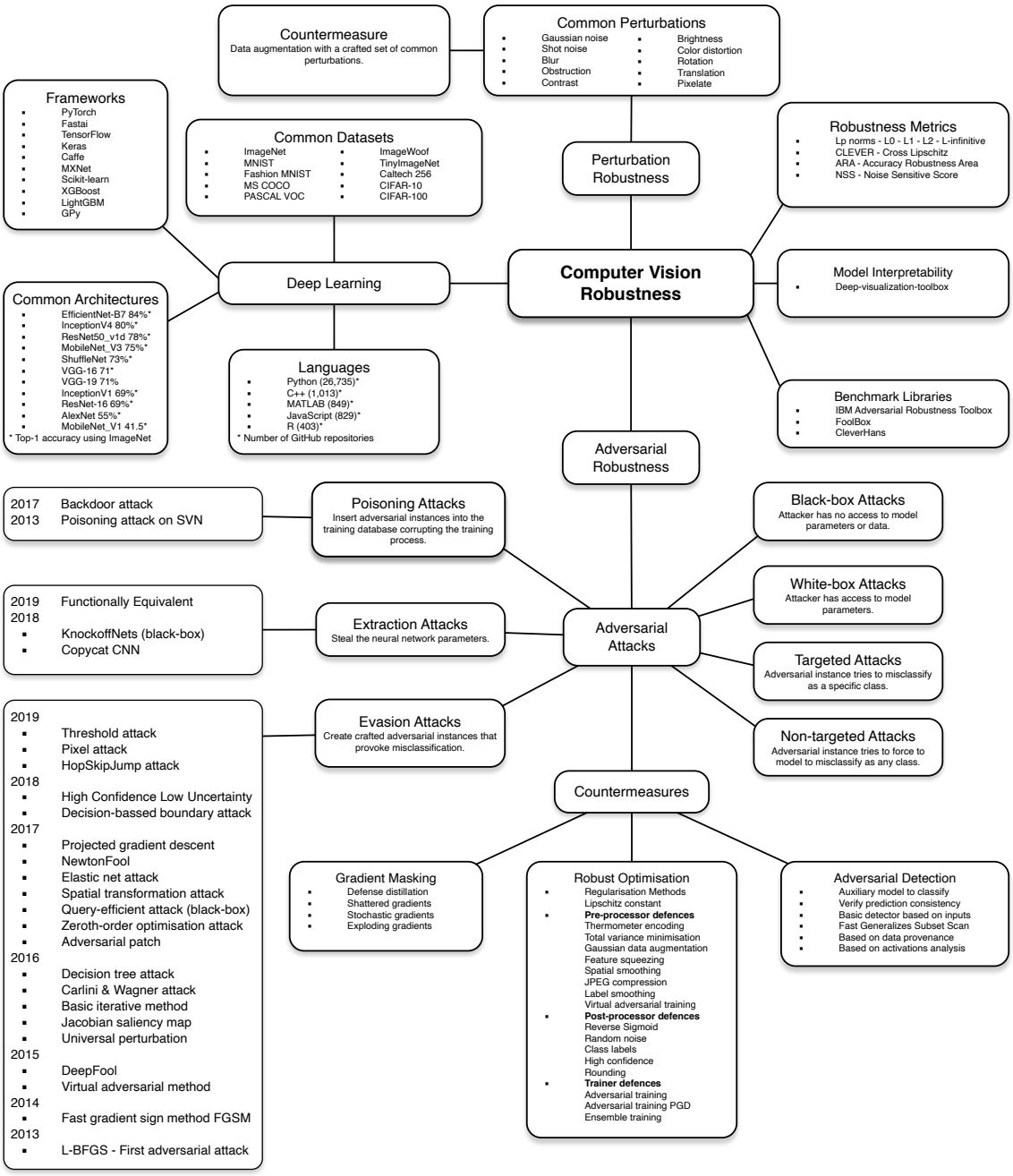

Figure 7. Big picture diagram of robustness for computer vision

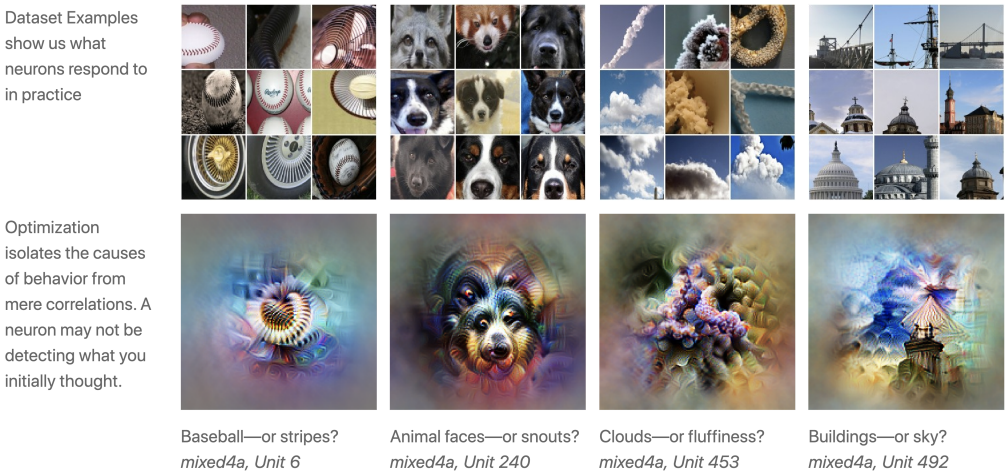

Figure 8. Comparison of how a neuron visualize a feature [4]

Adversarial robustness could be correlated to interpretability Ross et al. [5], demonstrated that an adversarial defence using gradient smoothing additionally increased the interpretability in the input gradients of the model as can be witnessed in Figure 9, their findings concluded that when the gradients are interpretable, adversarial instances may be used as explanations. If the victim model had more interpretable input gradients, then the adversarial examples which are generated directly from their input gradients would be more interpretable as well, resulting in an adversarial example that is more obviously transformative away from the original class label and towards another.

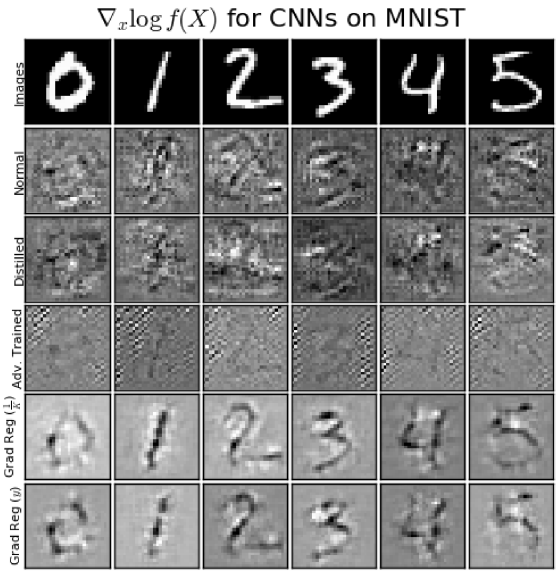

Figure 9. Input gradients of MNIST dataset, distilled models at T=50 (third from top), adversarial trained (fourth from top) and gradient regularization (bottom top). Regularized model gradient appears smoother and easier to interpret by human perception [5]

5. Robustness Metrics

Having a standardized metric to measure the robustness of a model is a research gap that does not have a consensus answer. Currently, most research uses these two approaches, $Lp - norms$ distances, used to determine the minimum distance from the original instance to the closest adversarial example, this is called *minimum adversarial distortion* [42] and testing the accuracy of the model under adversarial attacks [43].

L_0 quantify the number of pixels that are different between two images, L_2 additionally known as Euclidean distance which is the shortest distance between two vectors, computes the squared difference between two images and L_∞ quantify the largest difference between corresponding pixels in the images, determines the maximum value in a vector.

Both approaches have some significant drawbacks, $Lp - norms$ are not able to capture the human similarity perception accordingly, and measuring the accuracy under adversarial attacks is highly affected by the attack specifications and cannot

comprehensively reflect the actual robustness regarding model-intrinsic properties.

Several works [6, 7, 44] have concluded that $Lp - norms$ are not suitable for defining similarity as can be observed in Figure 10, and may lead one to conclude that an adversarial example is similar to a benign instance when it is not.

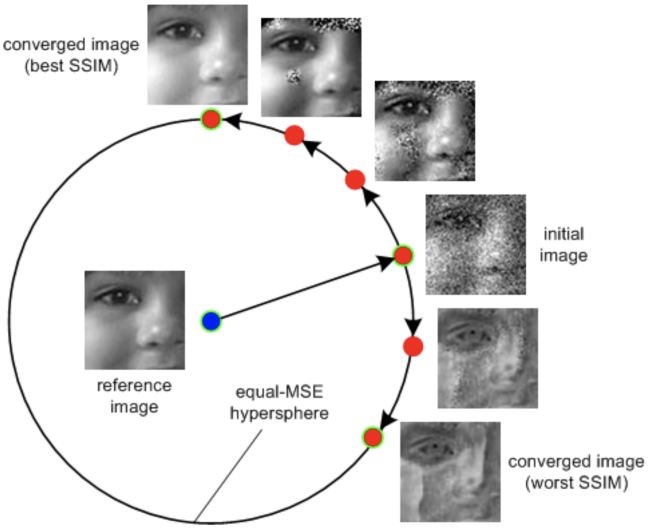

Figure 10. Images equally far away from a reference image in the L_2 distance [6]

Sharif et al. [7] created an experiment where they presented three sets of images as witnessed in Figure 11, to 399 participants, each image has three variants C_B (Benign) is the benign non-adversarial instance, C_{AI} (Adversarial Imperceptible) shows an adversarial instance that fool state-of-the-art DNNs and was *not* designed to mislead humans, and finally C_{AP} (Adversarial Perceptible) which is an adversarial instance that fool state-of-the-art DNNs, designed to mislead humans. Both C_{AP} and C_{AI} share same $Lp - norms$ distance with respect of C_B, the results indicated that the humans were able to have high accuracy in the C_B and C_{AI} instances and low accuracy for the C_{AP} instances, demonstrating that $Lp - norms$ can be insufficient for ensuring perceptual similarity in some cases.

(a) L_0 (b) L_2 (c) L_∞

Figure 11. Set of images with similar Lp-norms perturbation [7]

For these reasons the author consider this topic should be of main priority on the robustness research, even though the numbers of scientific papers related to adversarial robustness is increasing each year as can be witnessed on Figure 6, if there is not an standardized robustness metric, the validity of the new proposals to enhance robustness cannot be ensured.

There is a vast amount of ongoing research [45, 46, 47, 48, 7] around electing new robustness metrics, selecting which metric should be used as a standardized measure of robustness is not a straightforward endeavour, nevertheless, one conclusion from the ongoing metric research is that the selected metric should have at least the following attributes: it should be able to **generalize** across different data sets, models, defence and attacks, **invariance** under re-parameterization and acceptable **computing complexity**.

6. Adversarial Robustness

Adversarial examples are slightly modified versions of a correctly classified input instance that are carefully crafted with the whole purpose of generating a class misclassification inside the targeted model, which additionally report high confidence on the wrong prediction. The formalized definition of adversarial in most research [33, 49, 50] defines adversarial instance as x', that is a slightly modified version of a regular instance x, such that a neural network model classifies them differently.

An adversarial examples x' satisfies two properties [51]: (1) for some $d(\cdot)$, distance metric $d(x, x') < \varepsilon$, but (2) for the neural network, $f(x) \neq f(x')$. As long as ε is set to be small enough, the adversarial perturbation introduced should not change the actual true classification of the object in the image.

The term adversarial example was first used in 2013 by Szegedy et al. [21] they generated small perturbations on a particular image that was previously correctly classified and were able to tool the state of the art deep neural networks with high probability, these misclassified samples were named as *adversarial examples*.

Deep Learning is usually considered a *black box* technique since it is not evident how to interpret the decision behind a particular output. From adversarial examples, some knowledge concerning the internal working of a DNN could be acquired, especially since adversarial attacks are focused on finding problematic decision boundaries.

Adversarial examples represent worst-case domain shift, in Deep Learning domain shifts occur when training data and test data do not have the same distribution, with adversarial examples, the adversary maliciously on purpose does this worst-case domain shift and causes the model to behave poorly. Figure 12 represents the variety of adversarial examples that can be crafted using different attack mechanisms.

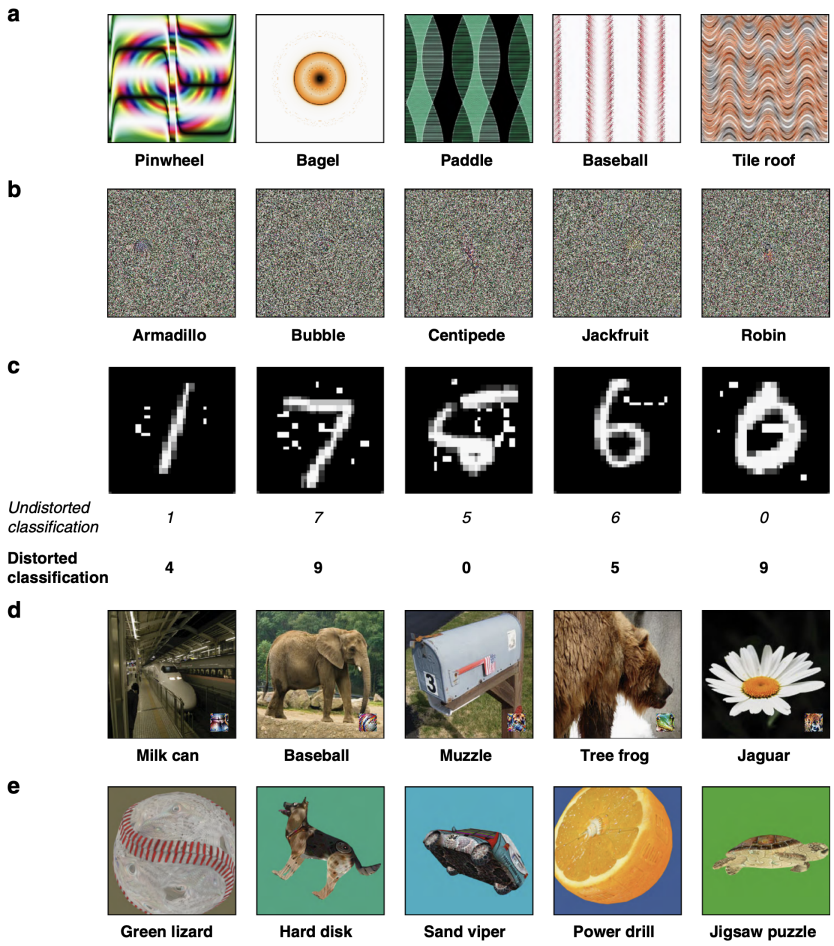

Figure 12. Examples of adversarial images, (a) shows indirectly encoded images [8], (b) shows directly encoded images [8], (c) shows perturbed adversarial images [9], (d) example of a patch attack [10], (e) shows 3D objects that can be printed in real-world and produce misclassification [11]. Figure from [12]

6.1. Adversarial Attacks

There are several methods to generate adversarial instances; a summary of the most common adversarial attacks can be observed inside the Table 3. Adversarial attacks can be classified by their threat model [52], the threat model defines the rules of the attacker, the capabilities and the end goal of the attack. The goal defines what the adversarial instance seeks from the attack, such as confidence reduction, misclassification, and targeted misclassification. An attacker can have single or multiple goals. Capabilities refer to what resources are available to the attacker such as training data, architecture and model parameters, and this additionally includes some boundaries where the attacker claim that is able to produce misclassification

such as specifying the type of $L_p norms$ and their upper and lower bounds.

Inside **black-box attacks** the attacker has limited knowledge. The limited capabilities of the adversary are the most common on real-world scenarios and often considered as more practical-research. These attacks are bounded around querying the model on inputs and observing the labels and accuracy returned. In order to alleviate the lack of knowledge, black-box attacks usually rely on *adversarial transferability* [53], where the attacker craft adversarial examples using a substitute model A, submit the same examples to model B which is the victim, and model A is likely to misclassify the inputs. Most mitigation techniques for black-box attacks are based on limiting the number of queries and information a deployed model provides and reducing the transferability property of the victim model [54]. **White-box attacks** have complete knowledge of the targeted model, such as the architecture, parameters, learned weights and in some cases even labeled trained data. For this type of attacks the common strategy of the attacker is to model a replica using the victim weights and derive the adversarial instances. **Evasion attacks** are the most common attacks, the adversary aims to provoke a misclassification by crafting an adversarial instance, evasion attacks can be either white-box or black-box. **Poisoning attacks** refer to attacks directly on the model training data, affecting the training of the model itself, these variety of attacks attempt to poison the training data by injecting adversarial instances to compromise the learning process, as expected, all poisoning attacks require a white-box setting, where the attacker has full access to the victim model. **Extraction attacks** are black-box attacks that do not influence the training dataset. Their main goal is to explore and extract knowledge of the learning algorithm such as parameters and type of architecture.

Table 3. Common adversarial attacks

Attack Name	Description	Year
Threshold Attack [55]	evasion, black-box, un-targeted, L_∞	2019
Pixel Attack [56]	evasion, black-box, un-targeted, L_0, low-cost, easy-implementation	2019
HopSkipJump Attack [57]	evasion, black-box, targeted, un-targeted, $L_2 L_\infty$, only requires class predictions, advanced version of Boundary Attack	2019
Projected Gradient Descent (PGD) [58]	evasion, iterative extension of FGSM, similar to BIM, main difference is that PGD projects the attack results back on the $\epsilon - norm$ ball around the original input at each iteration	2017
NewtonFool [59]	evasion, un-targeted, tries to decrease the probability $F_y(x)$ of the original class $y = C(x)$ using gradient descent	2017
Elastic Net Attack (EAD) [60]	evasion, L_1, modification of (C&W)	2017
Spatial transformation Attack [61]	evasion, performs a combination of exactly one rotation and one translation to the input image to craft the adversarial instance, same rotation and translation parameters are used for all the input batch	2017
Query-Efficient Attack [62]	evasion, black-box version of (C&W)	2017
Zeroth-Order Optimization Attack (ZOO) [63]	evasion, black-box, only has access to the input images and the confidence scores.	2017
Boundary Attack [64]	evasion, black-box, only requires queries of the output class	2016
Adversarial Patch [65]	evasion, used to create printable adversarial instances for the real-world	2016
Carlini & Wagner Attack (C&W) [66]	evasion, white-box, targeted, un-targeted, $L_2 L_\infty$,	2016
Basic Iterative Method (BIM) [67]	evasion, extension of FGSM, applies the attack multiple times, iteratively.	2016
Jacobian Saliency Map (JSMA) [9]	evasion, targeted, L_0	2016
Universal Perturbation [68]	evasion, un-targeted, creates a constant perturbation that successfully alters the classification of a specified fraction of inputs.	2016
DeepFool [45]	evasion, un-targeted, L_2, projects the input into the nearest decision boundary, iteratively.	2015
Fast Gradient Sign Method (FGSM) [33]	evasion, targeted, un-targeted, $L_1 L_2 L_\infty$, very efficient to compute, only one gradient evaluation is required, popular choice for adversarial training.	2014
Functionally Equivalent Extraction [69]	extraction, direct extraction, no training, focus on accuracy and high fidelity	2019
Copycat CNN [70]	extraction, learning, has a goal of accuracy and fidelity, query labels	2018
KnockoffNets [71]	extraction, learning, has a goal of accuracy and query probabilities	2018
Poisoning Attack on SVM [72]	poisoning, used for Support Vector Machines	2013
Backdoor Attack [73]	poisoning, attacks training dataset and corresponding ground-truth labels	2017

6.2. Countermeasures

As the number of adversarial attacks increases, there is additionally an increment in defence mechanisms to countermeasure them. A summary of common defence strategies can be observed in Table 4. There is not a single universal defence that will improve the robustness of a model for all types of attacks. Each defence is robust against a specific attack under certain specific conditions such as the $Lp-norms$ bounds one defence that claims to be robust for a certain attack on a specific type of $Lp-norms$ may not improve the robustness against the same attack with a different $Lp-norms$ type or boundary.

Countermeasures for adversarial attacks can be grouped in three main categories: **gradient masking**; since most attacks exploit back-propagation to use gradients to craft the adversarial instance, gradient masking consists on hiding gradient information to the attacker by obfuscating them, **robust optimization** consists of different techniques that can enhance the model performance against adversarial instances, such as regularization, adversarial training, adding random noise, ensemble training, and gradient smoothing, **adversarial detection** this type of defences specialize in detecting in runtime adversarial instances among the inputs.

One of the most common techniques is adversarial training which consists on augmenting the training data with adversarial examples to improve the robustness of the network, there are some drawback with this approach, the attack used by the victim model to defend and the one used by the adversary should be the same, if the attackers adapt its strategy with a different attack, the defence is not very effective anymore, this is the case for most defences and remains an open research gap. Is harder to generalize model robustness to adaptive attacks, since the classifier is required to be aware of all attacker strategies.

Table 4. Common adversarial defences

Defence Name	Description	Year
Thermometer Encoding [74]	robust optimization, data pre-processing, input discretization, encodes each feature as a fixed-size binary vector	2018
Total Variance Minimization [75]	robust optimization, data pre-processing, randomly selects a small set of pixels to reconstruct an image, reconstructed image does not contain adversarial perturbation since these perturbation are small and localized	2018
Pixel Defend [76]	robust optimization, purifies a maliciously crafted image by moving it back towards the distribution seen in the training data, can be used to protect already deployed models in combination of other defences	2017
Gaussian Data Augmentation [77]	robust optimization, data pre-processing, augmentation technique that additionally improve adversarial robustness	2017
Feature Squeezing [78]	robust optimization, data pre-processing, reduces the precision of the components of x by encoding them on a smaller number of bits	2017
Spatial Smoothing [78]	filter out adversarial signal using local spatial smoothing	2017
JPEG compression [79]	robust optimization, data pre-processing, dramatically reduce adversarial attacks	2016
Label Smoothing [80]	modifies the labels during the training, the difference between maximum and minimum components is reduced, thus reducing gradients	2016
Virtual Adversarial Training [81]	robust optimization, data augmentation, is not used to create adversarial instances, it creates samples that, if included in the training set for adversarial training, result in local distributional smoothness of the trained model	2015
Reverse Sigmoid [82]	robust optimization, data post-processing, limits the information provided to the adversary, omitting real probability scores, provides useful, yet misleading class probabilities, forcing the attacker to discard the probabilities score	2018
Random Noise [83]	robust optimization, data post-processing, extraction defence	2018
High Confidence [84]	robust optimization, data post-processing, extraction defence, avoid returning rich outputs, omit confidence values	2016
Rounding [84]	robust optimization, data post-processing, similar to High Confidence, extraction defence, round the confidence scores returned to avoid giving out to much information	2016
Adversarial training [21, 58]	robust optimization, data augmentation, generate adversarial inputs and add them to the train set	2013
Defence Distillation [85]	gradient masking, transform data, uses distillation training, broken by (C&W) attack	2015
Fast Generalizes Subset Scan Detector [86]	detection, finds anomalous patterns in general categorical data sets	2018
Activation Analysis Detector [87]	poisoning attack detection, backdoor detection, works on text and images, model does not require a verified dataset	2018
Data Provenance Detector [88]	poisoning attack detection, uses contextual information about the origin and transformation of data point in the training set to identify poisonous data	2018

7. Perturbation Robustness

Deep Neural Networks exhibit unexpected instability on common perturbations; common perturbations can be considered as small changes that can be encounter in a real-life situation such as rotation, snow, fog, brightness, contrast and pixelation. One contribution that allowed to increase the research around this area is the perturbation benchmark introduced in Hendrycks et al. [13] additionally known as ImageNet-C benchmark due to the name of the new dataset introduced, which was generated by adding a set of 15 types of algorithmically generated visual corruptions as witnessed in Figure 13.

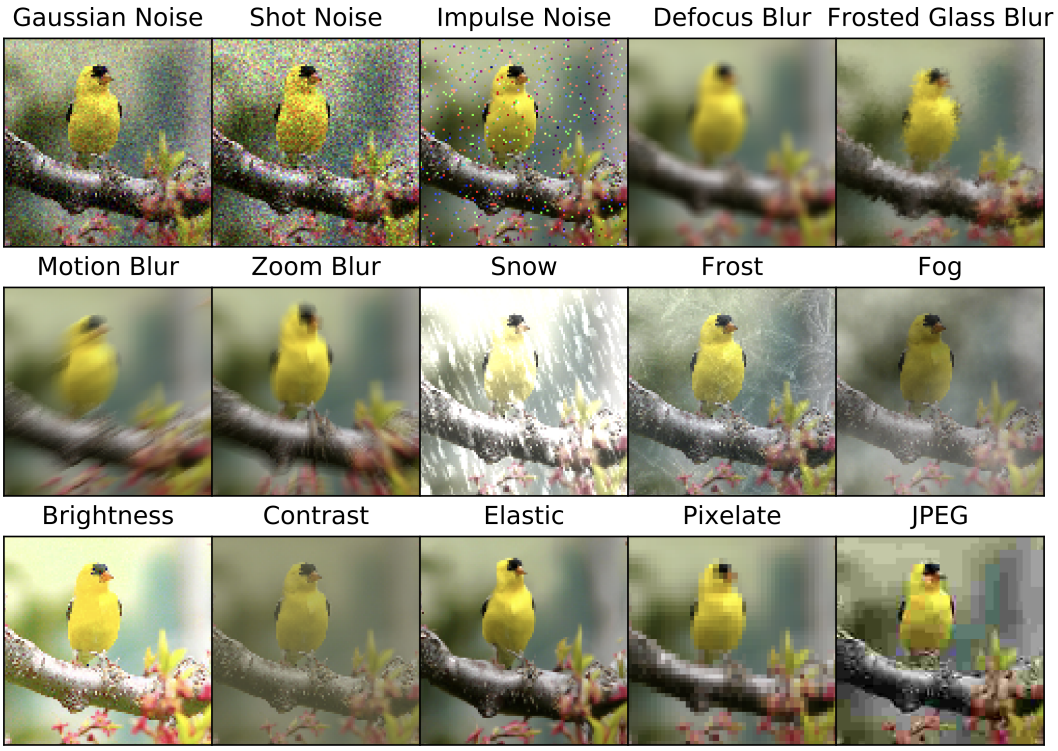

Figure 13. ImageNet-C perturbations[13]

There has been some research [89, 90, 13] to determine if adversarial and common perturbations are correlated, the results demonstrate that a fully augmented model is not more robust than the regular model to adversarial attacks, indicating that robustness to common perturbation does not protect the model from adversarial attacks. The results additionally concluded that some viable mitigation techniques to improve common perturbation robustness are: histogram equalization, multi-scale architectures, and data augmentation with common perturbations data.

8. Tools Available for Measuring Robustness

Robustness benchmarking tools gather known attacks and defences and implement them in the form of a library agnostic to the type of DL framework. To the author's knowledge, the amount of benchmark tools is limited.

Some of the most popular open-source libraries are Cleverhans [91], Foolbox [92], and IBM Adversarial Robustness Toolbox (IBM ART) [93].

IBM ART was selected to perform the experiments in this work since among the others, it has a bigger variety of attacks, defences and even metrics implemented, is compatible with PyTorch, the framework used by the author, has great documentation and finally it has excellent community support, the author had several discussions with the creators of the library using IBM Slack channel with a positive and rapid response.

9. Proposed Methodology

For the experimental part of this research multiple tests were conducted to analyze the required level of difficulty to provoke a model misclassification. The experiment consists of the workflow described inside Figure 14.

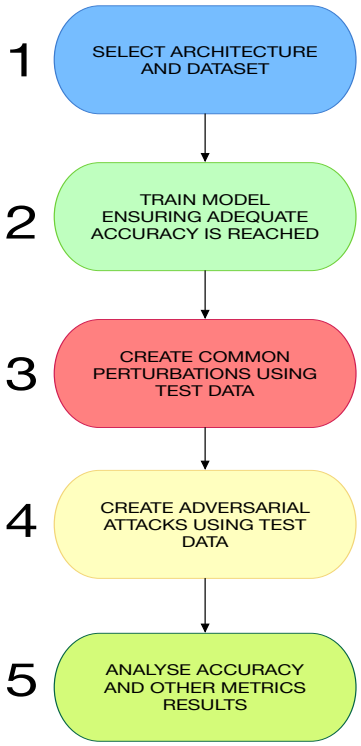

Figure 14. Workflow of the experiment.

Key points of the experiment:

1. The architecture for all the experiments is ResNet-50. ResNet-50 was selected since is one of the state-of-the-art architectures that has shown excellent results inside the literature reviewed.
2. Four datasets were selected; each dataset has unique characteristics that are explained in the following chapter.
3. Before testing robustness, the models should have at least 90% accuracy to ensure there is not a misclassification due to the model's poor performance.
4. One image per class is selected from the validation set to test against common perturbation and adversarial attacks.
5. All four datasets have ten classes, due to this, on each evaluation, the test dataset consists of ten images.
6. Four metrics are analyzed for each test L_1, L_1,L_∞, and *accuracy*.
   (a) $L_p norms$ are gathered to understand if there is a correlation among accuracy and level of $L_p distance$.

(b) *accuracy* indicates the level of robustness of the model.

7. Five types of common perturbations are applied, each perturbation is applied in a range of different magnitudes as witnessed in Figure 7.

8. The adversarial attack selected is Projected Gradient Descent (PGD), since PGD is recommended as good entry to measure robustness [58].

Projected Gradient Descent (PGD) attack is an iterative method in which, after each iteration the perturbation is projected on an $L_p norms$ to find an adversarial instance. For this experiment IBM ART library is used to generate the attack using its default parameters, the default parameters for the attack can be observed in Table 5.

Table 5. PGD default parameters

Name	**Value**	**Description**
norm	np.inf	The Lp norm of the adversarial perturbation.
eps	0.3	Maximum perturbation the attacker can introduce.
eps_step	0.1	Attack step size at each iteration
max_iter	100	Maximum number of iterations to craft the adversarial instance.
targeted	False	Targets or untargeted attack.
num_random_init	0	Number of random initializations within the epsilon ball.
batch_size	1	Size of the batch on which adversarial samples are generated.
random_eps	False	When True, epsilon is drawn randomly from truncated normal distribution.

10. Experiment and Evaluation

Environment setup

The models were trained in a server provided by the Department of Computer Systems of Tallinn University of Technology with the following specification: CPU Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 16GB DDR4, 1TB HDD, GPU NVIDIA GeForce GTX 1080 with a clean installation of Ubuntu 20.04 LTS as OS.

For this experiment 60GB of HDD space was utilized to store the different datasets and save the model weights.

The sever was accessed principally remotely, using SSH connection with Ngrok Pro client [94] to facilitate the tunnelling from outside Tallinn University of Technology VPN. Since each time the server rebooted Ngrok VPN client was shutdown, Ngrok was configured to autostart using the YAML file shown in Figure 15, the **remote_addr** and **subdomain** are paid features from the Pro version, by using the free version Ngrok domains will be randomized on each restart, besides that, they share similar functionality.

Nvidia drivers were installed using CUDA Toolkit 10.2 [95] and UEFI Secure Boot had to be disabled in order to allow remote drivers installation.

```
authtoken: {ngrok_token}
region: eu
tunnels:
        default:
                proto: tcp
                addr: 22
                subdomain: ict_lab_server
                remote_addr: 3.tcp.eu.ngrok.io:21832
        thesis:
                proto: http
                addr: 8888
                subdomain: cnn_robustness_analysis
```

Figure 15. Ngrok YAML configuration file

Software Environment

The experiments were performed using Python + Jupyter Notebooks + Conda environment with Fastai as DL library, the main list of dependencies are shown in Table 6. The full Conda environment along with the Jupyter Notebooks code is available in the author GitLab repository [96].

Table 6. Main software dependencies

Name	Version
python	3.7.7
fastai	1.0.61
torch	1.4.0
torch cuda	10.1
adversarial-robustness-toolbox	1.2.0
jupyterlab	1.2.6
conda	4.8.3

Datasets

For this experiment the author used four datasets, Imagenette-160 [97], Imagewoof-160 [97], MNIST [98] and Pentomino (Arcade Universe) [99].

Imagenette-160 is a subset of ten easily classified classes from ImageNet, similar to that, Imagewoof-160 is a subset of ten ImageNet dog breed classes that are more difficult to classify since the classes share similar characteristics. MNIST (Modified National Institute of Standards and Technology database) is a famous handwritten digits dataset, commonly used as a benchmark due to its small size and fast training time. Imagenette-160, Imagewoof-160 and MNIST was downloaded from the Fastai datasets URLs.

The last dataset, Pentomino is a dataset crafted from Arcade Universe [99] which is an open-source library that can generate crafted datasets that are valid for image

classification tasks, the output images have a finite set of classes and share similar attributes such as amount of channels and size. The Pentomino dataset is a sequence of images where each image contains exactly three shapes. Two shapes are identical, and the third is different from the others. The target of the classification task is to identify the third shape. There are ten possible shapes, and thus ten possible classes for the task, an example of the dataset can be observed in Figure 18. The author added this open-source dataset as part of this experiment to analyze the behaviour of common perturbation on simple shapes.

Some highlights among the four datasets used in the experiment:

1. MNIST dataset has a small resolution and has a grayscale colour; classification complexity is considered as low. Thus should, in theory, be considerably simple to generate adversarial instances. MNIST will represent the experiment baseline due to its simplicity.

2. Imagewoof-160 and Imagennete-160 datasets share similar characteristics. The difference relies on Imagewoof-160 being a more challenging dataset to classify. The author will analyze whether models that are trained to classify a more difficult dataset present stronger robust attributes.

3. Pentomino dataset contains simple shapes which should result in a simple classification task, however, due to the additional complexity of the dataset; that the image class should correspond only to the shape that does not have a duplicate element, the results obtained could be insightful.

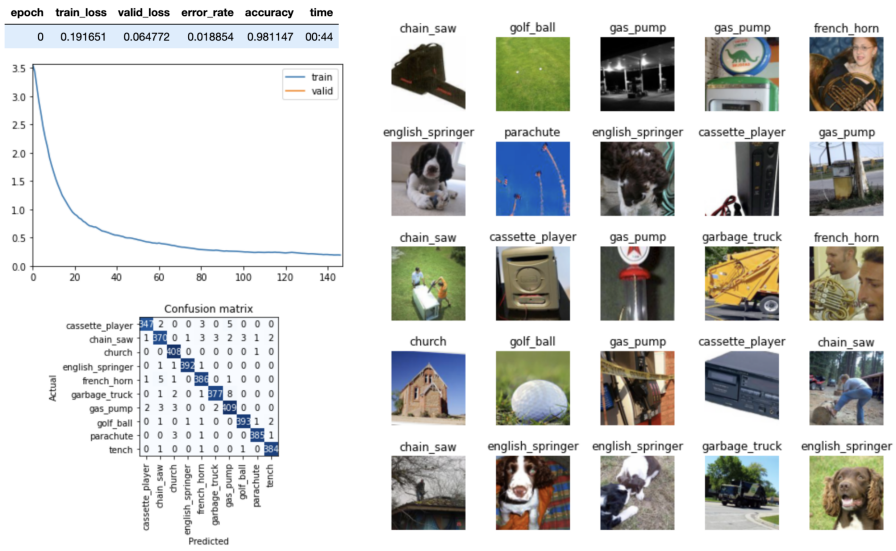

Figure 16. Imagenette-160 dataset, left section denotes the training accuracy and confusion matrix, right section denotes a sample of the images.

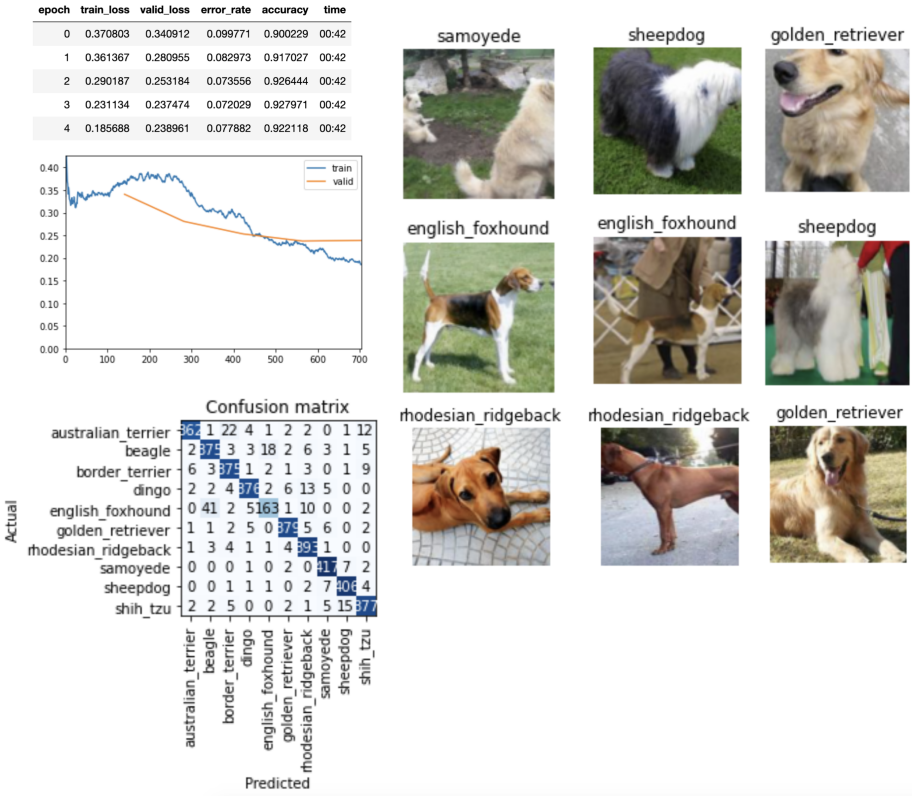

Figure 17. Imagewoof-160 dataset, left section denotes the training accuracy and confusion matrix, right section denotes a sample of the images.

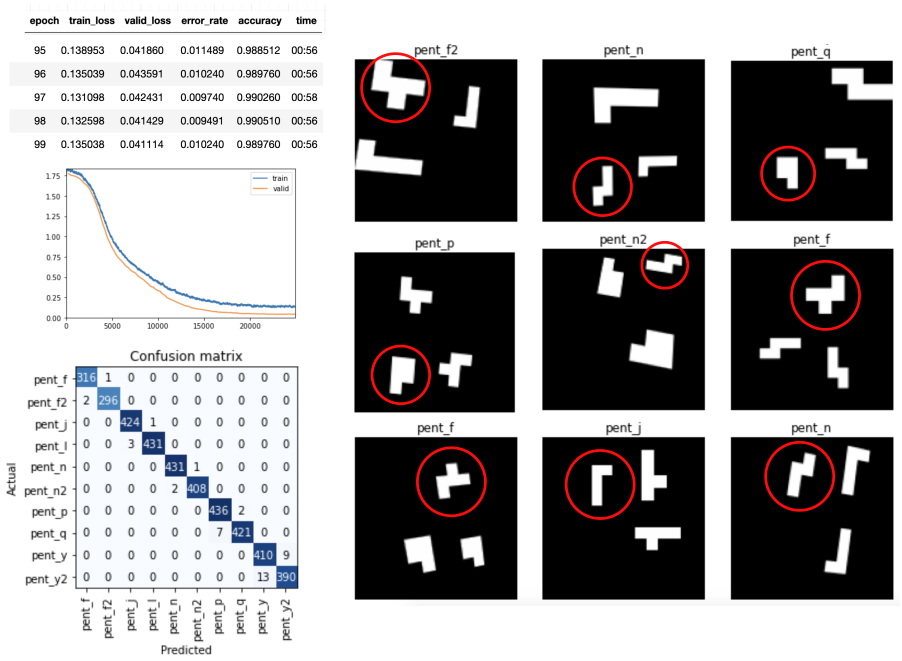

Figure 18. Pentomino dataset, left section denotes the training accuracy and confusion matrix, right section denotes a sample of the images, a red circle was added to highlight the shape that is different from the other two, this shape represent the class of the image.

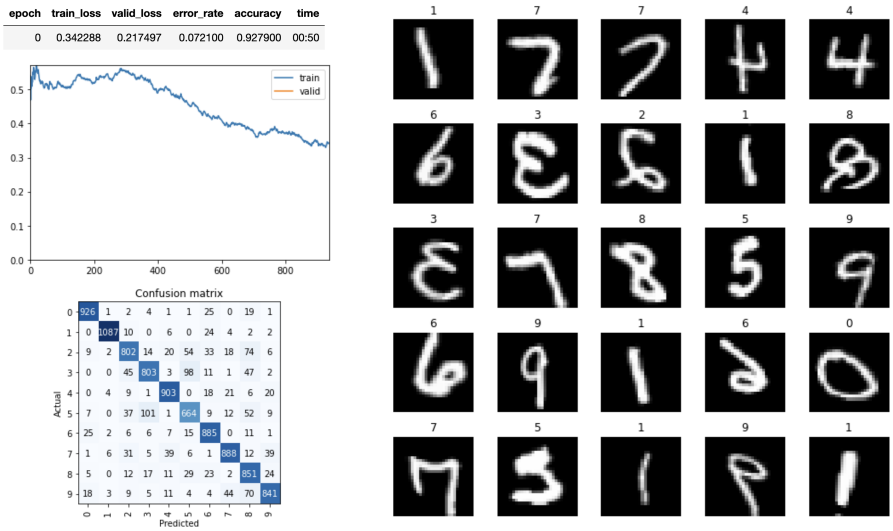

Figure 19. MNIST dataset, left section denotes the training accuracy and confusion matrix, right section denotes a sample of the images.

Table 7. List of perturbations applied

No.	Perturbation	Range	Description
1	Rotation	[-60, 20, 100, 180] degrees	Rotates de image
2	Brightness	[0.1, 0.3, 0.4, 0.5] change	Change of 0 will transform image to black, change of 1 will transform the image to white
3	RGB Randomize	Red [0.2, 0.5, 0.99] thresh	Randomize one of the channels of the input image, values will not exceed the thresh
4	Jitter	[-0.05, -0.03, 0.00, 0.03, 0.05] magnitude	Changes the pixels by randomly replacing them with pixels from the neighborhood, which is controlled by the magnitude
5	Contrast	[0.5, 1.17, 2.74, 6.41] scale	Scale of 0 will transform image to gray, scale over 1 will transform image to upper-contrast, scale of 1 does not adjust the image

Training Methods

The training of all four datasets share similar training conditions, ResNet-50 architecture, pre-trained on ImageNet, Adam optimizer, CrossEntropyLoss, resizing according to each data set size and data normalization with MNIST stats [0.131], [0.308] for MNIST dataset, and ImageNet stats [0.485, 0.456, 0.406], [0.229, 0.224, 0.225] for the rest.

Results

During training, all datasets were able to reach at least 90% accuracy, additionally, as it is expected, the time taken to train each dataset is correlated to the complexity of the dataset, for MNIST and Imagennete-160 took one epoch and less than one minute to reach 92% accuracy, for Imagewoof-160 took four epochs and three minutes, finally for the most complex dataset, Pentomino, it took around ninety epochs and one hour and thirty minutes to reach 98% accuracy for this dataset a dropout rate of 30% had to be added to avoid over-fitting.

The results after applying the five common perturbations to each dataset can be observed in Figure 20 for both Imagewoof-160 and Imagennete-160 and inside Figure 21 for the MNIST and Pentomino dataset.

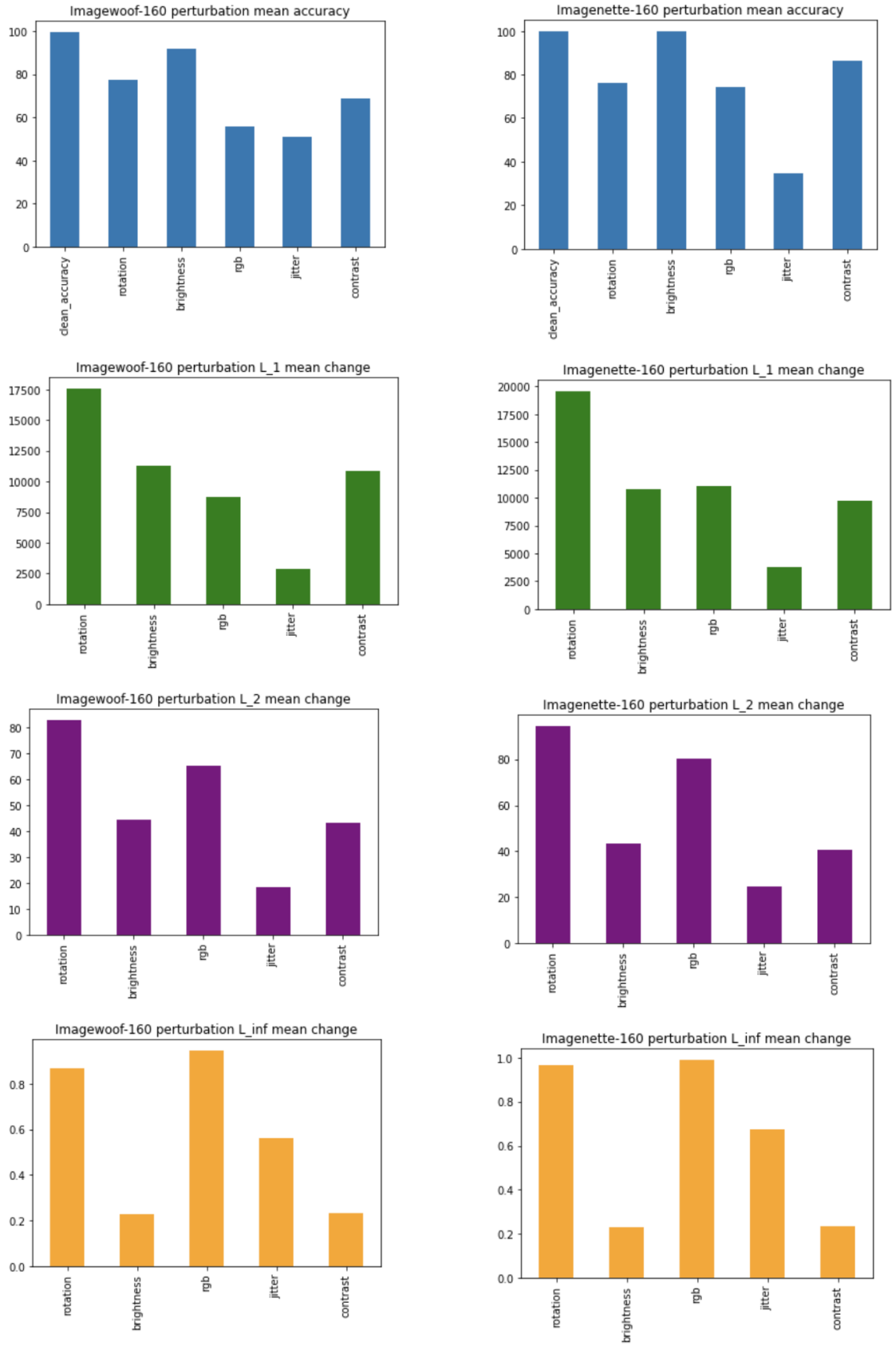

Figure 20. Perturbation attack results, left section Imagewoof-160, right section Imagennete-160. From top to bottom, mean accuracy plot in blue, mean L_1 distance plot in green, mean L_2 distance plot in purple and mean L_∞ distance plot in orange.

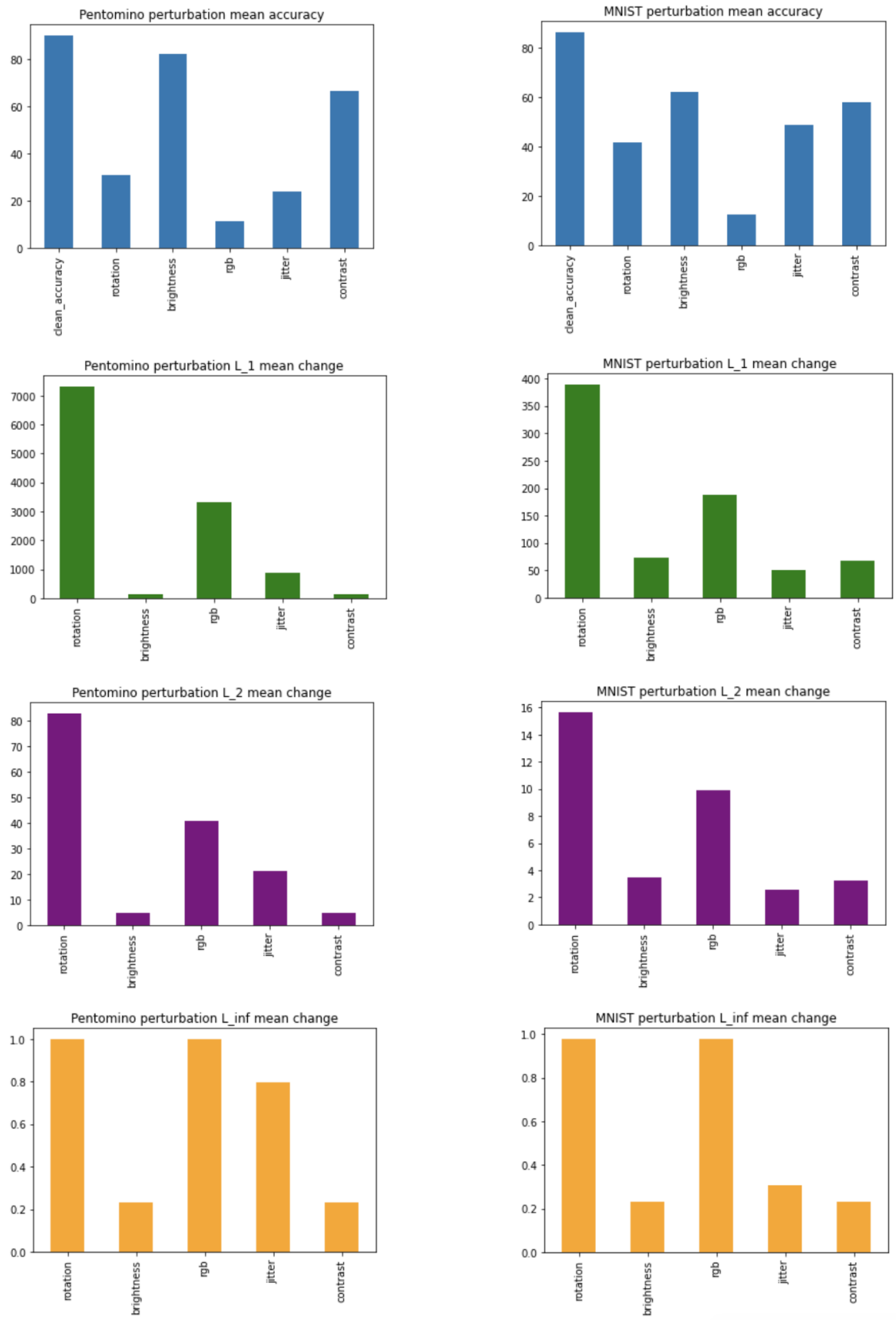

Figure 21. Perturbation attack results, left section MNIST, right section Pentomino. From top to bottom, mean accuracy plot in blue, mean L_1 distance plot in green, mean L_2 distance plot in purple and mean L_∞ distance plot in orange.

The accuracy results can be observed inside Table 8, additionally an example of the applied perturbation can be observed in Figure 22.

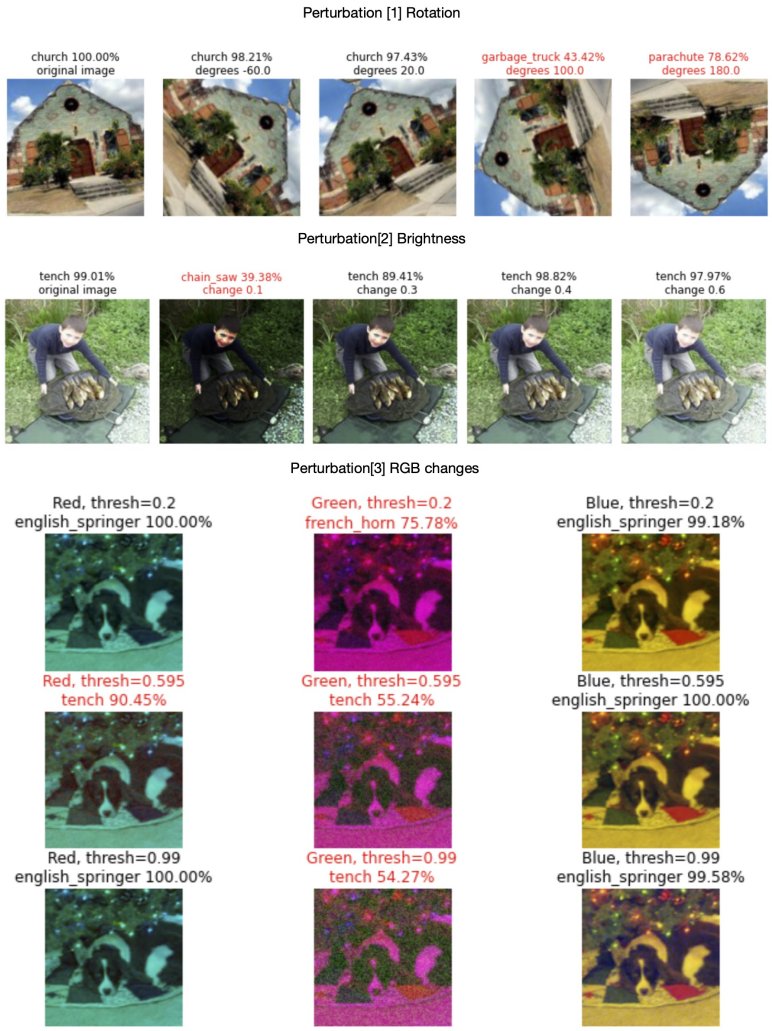

Figure 22. Representation of three perturbations applied to Imagennete-160, from top to bottom, rotation, brightness and RGB randomize, each image with a red title represent a misclassification from the ground truth.

Conclusions after the common perturbations attack are as follows:

- Common perturbations are valid and concerning issue in terms of accuracy, since all datasets were reasonably simple to fool.
- The two strongest perturbations where rotation and jitter, which might be related to patterns applied on both perturbations to change the pixels.
- Imagennete-160, presented more robust features against Imagewoof-160, which

might indicate the similarity of the classes could lead to a lack of robustness, further work on this could be beneficial.

- Pentomino and MNIST worst perturbation was due to RGB randomize since they are both grayscale images, the model is not able to process colour channels, indicating that adding a preprocessing filter such as binary thresholding before the model inference might conduct to better results.
- Grayscale images present strong robustness against contrast and brightness perturbations.
- Pentomino had the worst accuracy, indicating that simple shapes might lack robust characteristics since their decision boundary might be quite close to among classes.
- The author was not able to find a correlation between accuracy and $L_p norms$, for instance L_∞ shows a high distance for the RGB randomize when there is low accuracy such as Pentomino and MNIST. However, it has high values aditionally with increased accuracy such as Imagennete-160 and Imagewoof-160. Indicating that $L_p norms$ might not be the correct metrics for robustness.

Table 8. Common perturbations accuracy results

Dataset Name	**Clean Accuracy**	**Accuracy After Perturbation**	**Accuracy difference**
Pentomino	89.98%	42.816%	47.114%
Imagewoof-160	99.64%	66.56%	33.07%
MNIST	99.99%	72.50%	27.49%
Imagennete-160	99.99%	73.11%	26.87%

From the adversarial attack results, the following conclusions can be taken:

- PGD is a powerful attack can drastically drop the accuracy of a model even if it had excellent accuracy, it was able to fool all datasets.
- Benchmark tools such as IBM ART simplify testing different attacks.
- The attack is crafted using only ten relatively small images, however, it took more than two minutes to generate results, confirming that the attacks are computationally expensive, meaning that, targeting bigger high-resolution datasets might not be computationally feasible.

- The attack was not able to provoke a misclassification for Pentomino dataset indicating that the dataset has some unique characteristics that are able to avoid PGD attack using default parameters, for this a more aggressive tuning might be required, adversarial attacks tuning is an area that requires further analysis.

Adversarial attack accuracy results can be witnessed in Table 9, and an example of the resulting images after the attack can be observed inside Figure 23.

Table 9. Adversarial attack results

Dataset Name	Clean Accuracy	Accuracy After PGD	Difference	Time for the Attack
Pentomino	89.98%	98.19%	1.45%	2min 26seconds
Imagewoof-160	99.64%	57.73%	41.91%	2min 38 seconds
MNIST	99.99%	23.96%	76.03%	2min 14seconds
Imagennete-160	99.99%	29.99%	70%	2min 25seconds

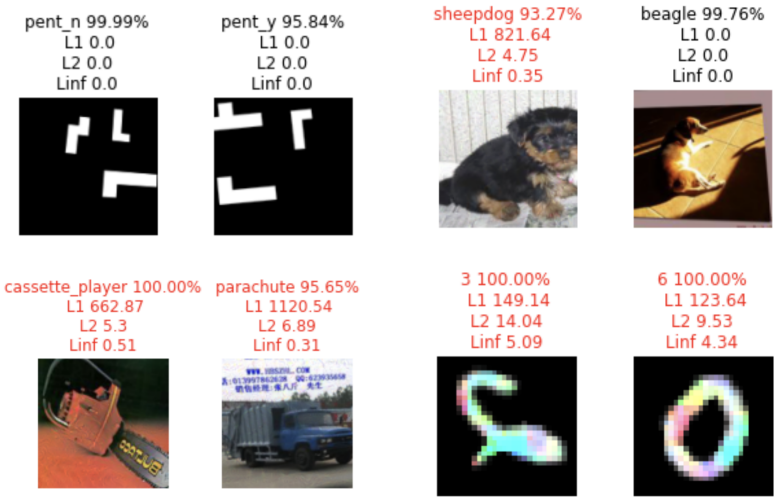

Figure 23. Adversarial instances, from left to right and top to bottom, Pentomino, Imagewoof-160, Imagennete-160 and MNIST dataset, red titles denote that the class of the image was different from the ground truth.

11. Conclusions and Future Work

After this work, the author concludes adversarial and common perturbations are an alarming security threat that should have more research.

To the author's knowledge there is a vast amount of adversarial robustness literature and considerable less regarding perturbation robustness, as was concluded in the experiment even for models that have high accuracy, the amount of effort required to provoke a misclassification by common perturbations is considered low. Thus the author considers more research should inquire in making models more robust to common scenarios and not only on worst-case scenarios which are the adversarial instances.

During this work, several research directions were considered, such as improving interpretability and adaptive defences as mitigation mechanism, the author considers the direction with the highest priority should be identifying a reliable, standardized robustness metric to ensure future work is viable and accurate.

The number of attacks and defences available is drastically increasing, for each attack, there is a defence and then a counter-defence attack that can break it, the author acknowledges the impact of adversarial attacks and highly encourage the research on this direction, however, the author concludes that adversarial attacks are a representation of the worst-case scenario. By considering this, the research should always concentrate corresponding time on common perturbation robustness, through consolidating efforts in both fields, the research community might be capable of coming up with a robust design against all type of adversarial instances.

Future Work

This work provides an analysis of robustness inside computer vision; future work could address robustness inside other domains such as audio, video, natural language processing and malware detection.

The experiment demonstrated the basic workflow of adversarial instances, with particular focus on common perturbation using $L_p norms$, in this sense, more detailed examples could be added regarding defences and how they behave against adversarial

attacks, likewise, further analysis on the different robustness metrics could enhance the practical examples presented. Finally, the work presented could be converted in a PyPI package that allows verifying the robustness accuracy of any model in a simplified manner.

Adversarial and perturbation robustness is a rapidly changing research; the amount of scientific papers in this field is drastically increasing each year as witnessed from Figure 6. Nicholas Carlini, one of the key personalities around this field is a research scientist at Google Brain and co-author of the (C&W) adversarial attack [66], created a blog [3] that contains a starting point for robustness literature. The author highly encourages anyone starting in this field to review the content of this blog.

References

[1] K. Warr, *Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery*. O'Reilly Media, 2019. [Online]. Available: https://books.google.ee/books?id=QKegDwAAQBAJ

[2] A.-W. D. . A. Series, *Deep Learning Illustrated*. OREILLY, 2019, https://www.oreilly.com/library/view/deep-learning-illustrated/9780135116821/.

[3] CarliniBlog. Nicholas carlini blog. [Online]. Available: https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[4] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[5] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," *CoRR*, vol. abs/1711.09404, 2017. [Online]. Available: http://arxiv.org/abs/1711.09404

[6] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[7] M. Sharif, L. Bauer, and M. K. Reiter, "On the suitability of lp-norms for creating and preventing adversarial examples," *CoRR*, vol. abs/1802.09653, 2018. [Online]. Available: http://arxiv.org/abs/1802.09653

[8] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014. [Online]. Available: http://arxiv.org/abs/1412.1897

[9] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *CoRR*, vol. abs/1511.07528, 2015. [Online]. Available: http://arxiv.org/abs/1511.07528

[10] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," *CoRR*, vol. abs/1801.02608, 2018. [Online]. Available: http://arxiv.org/abs/1801.02608

[11] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *CoRR*, vol. abs/1707.07397, 2017. [Online]. Available: http://arxiv.org/abs/1707.07397

[12] Z. Zhou and C. Firestone, "Humans can decipher adversarial images," *Nat. Commun.*, vol. 10, no. 1334, pp. 1–9, Mar 2019.

[13] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," 2019.

[14] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," 2019.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[16] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[17] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[18] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, 2012.

[19] H. Xu, M. Dong, D. Zhu, A. Kotov, A. Carcone, and S. Naar-King, "Text classification with topic-based word embedding and convolutional neural networks," 10 2016.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[22] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *Journal of Physiology*, vol. 160, no. 1, p. 106, Jan 1962.

[23] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," *CoRR*, vol. abs/1701.07852, 2017. [Online]. Available: http://arxiv.org/abs/1701.07852

[24] L. M. S. Y. K. V. Subramanian, *Deep Learning with PyTorch 1.x.*, 2019. [Online]. Available: https://learning.oreilly.com/library/view/deep-learning-with/9781838553005/1eeb87fe-f960-4641-b8e0-93f3e9ea1a92.xhtml

[25] A. Madhavan and A. Ng, "Large-scale deep unsupervised learning using graphics processors," vol. 382, 01 2009, p. 110.

[26] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, pp. 947–951, 2000.

[27] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[28] J. Howard and S. Gugger, "fastai: A layered api for deep learning," 2020.

[29] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," 2018.

[30] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," *CoRR*, vol. abs/1901.09960, 2019. [Online]. Available: http://arxiv.org/abs/1901.09960

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.

[33] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[34] A. Fawzi, S. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: from adversarial to random noise," *CoRR*, vol. abs/1608.08967, 2016. [Online]. Available: http://arxiv.org/abs/1608.08967

[35] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Analysis of universal adversarial perturbations," *CoRR*, vol. abs/1705.09554, 2017. [Online]. Available: http://arxiv.org/abs/1705.09554

[36] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *Mach. Learn.*, vol. 107, no. 3, pp. 481–508, 2018. [Online]. Available: https://doi.org/10.1007/s10994-017-5663-3

[37] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," 2017.

[38] F. Fan, J. Xiong, and G. Wang, "On interpretability of artificial neural networks," 2020.

[39] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[40] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018, https://distill.pub/2018/building-blocks.

[41] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," 2015.

[42] N. Carlini, G. Katz, C. Barrett, and D. Dill, "Ground-truth adversarial examples," 09 2017.

[43] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang, and X. Chen, "Interpreting and evaluating neural network robustness," *CoRR*, vol. abs/1905.04270, 2019. [Online]. Available: http://arxiv.org/abs/1905.04270

[44] J. Gilmer, R. P. Adams, I. Goodfellow, D. Andersen, and G. E. Dahl, "Motivating the rules of the game for adversarial example research," 2018.

[45] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CoRR*, vol. abs/1511.04599, 2015. [Online]. Available: http://arxiv.org/abs/1511.04599

[46] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," 2018.

[47] H. Chen, H. Zhang, S. Si, Y. Li, D. S. Boning, and C. Hsieh, "Robustness verification of tree-based models," *CoRR*, vol. abs/1906.03849, 2019. [Online]. Available: http://arxiv.org/abs/1906.03849

[48] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," *CoRR*, vol. abs/1902.02918, 2019. [Online]. Available: http://arxiv.org/abs/1902.02918

[49] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017.

[50] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *CoRR*, vol. abs/1802.00420, 2018. [Online]. Available: http://arxiv.org/abs/1802.00420

[51] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," 2019.

[52] S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru, "A Survey of Black-Box Adversarial Attacks on Computer Vision Models," *ArXiv e-prints*, Dec 2019. [Online]. Available: https://arxiv.org/abs/1912.01667

[53] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016. [Online]. Available: http://arxiv.org/abs/1605.07277

[54] G. Adam, P. Smirnov, B. Haibe-Kains, and A. Goldenberg, "Reducing adversarial example transferability using gradient regularization," *CoRR*, vol. abs/1904.07980, 2019. [Online]. Available: http://arxiv.org/abs/1904.07980

[55] D. V. Vargas and S. Kotyan, "Model agnostic dual quality assessment for adversarial machine learning and an analysis of current neural networks and defenses," *CoRR*, vol. abs/1906.06026, 2019. [Online]. Available: http://arxiv.org/abs/1906.06026

[56] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[57] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," 2019.

[58] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017.

[59] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, pp. 262–277. [Online]. Available: https://doi.org/10.1145/3134600.3134635

[60] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," 2017.

[61] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," 2017.

[62] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Query-efficient black-box adversarial examples," *CoRR*, vol. abs/1712.07113, 2017. [Online]. Available: http://arxiv.org/abs/1712.07113

[63] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo," *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec17*, 2017. [Online]. Available: http://dx.doi.org/10.1145/3128572.3140448

[64] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017.

[65] T. B. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017.

[66] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[67] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016.

[68] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *CoRR*, vol. abs/1610.08401, 2016. [Online]. Available: http://arxiv.org/abs/1610.08401

[69] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," 2019.

[70] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul 2018. [Online]. Available: http://dx.doi.org/10.1109/IJCNN.2018.8489592

[71] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," 2018.

[72] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012.

[73] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR*, vol. abs/1708.06733, 2017. [Online]. Available: http://arxiv.org/abs/1708.06733

[74] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S18Su--CW

[75] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyJ7ClWCb

[76] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *CoRR*, vol. abs/1710.10766, 2017. [Online]. Available: http://arxiv.org/abs/1710.10766

[77] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," 2017.

[78] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *CoRR*, vol. abs/1704.01155, 2017. [Online]. Available: http://arxiv.org/abs/1704.01155

[79] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," *CoRR*, vol. abs/1608.00853, 2016. [Online]. Available: http://arxiv.org/abs/1608.00853

[80] G. Warde-Farley, "Perturbation, optimization and statistics," May 2016, [Online; accessed 17. May 2020]. [Online]. Available: https://pdfs.semanticscholar.org/b5ec/486044c6218dd41b17d8bba502b32a12b91a.pdf

[81] T. Miyato, S. ichi Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," 2015.

[82] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against model stealing attacks using deceptive perturbations," *CoRR*, vol. abs/1806.00054, 2018. [Online]. Available: http://arxiv.org/abs/1806.00054

[83] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," 2018.

[84] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," *CoRR*, vol. abs/1609.02943, 2016. [Online]. Available: http://arxiv.org/abs/1609.02943

[85] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2015.

[86] S. Speakman, S. Sridharan, S. Remy, K. Weldemariam, and E. McFowland, "Subset scanning over neural network activations," *CoRR*, vol. abs/1810.08676, 2018. [Online]. Available: http://arxiv.org/abs/1810.08676

[87] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *CoRR*, vol. abs/1811.03728, 2018. [Online]. Available: http://arxiv.org/abs/1811.03728

[88] N. Baracaldo, B. Chen, H. Ludwig, A. Safavi, and R. Zhang, "Detecting poisoning attacks on machine learning in iot environments," 09 2018.

[89] A. Laugros, A. Caplier, and M. Ospici, "Are adversarial robustness and common perturbation robustness independent attributes ?" 2019.

[90] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk, "Adversarial examples are a natural consequence of test error in noise," 2019.

[91] I. J. Goodfellow, N. Papernot, and P. D. McDaniel, "cleverhans v0.1: an adversarial machine learning library," *CoRR*, vol. abs/1610.00768, 2016. [Online]. Available: http://arxiv.org/abs/1610.00768

[92] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," 2017.

[93] M. Nicolae, M. Sinn, T. N. Minh, A. Rawat, M. Wistuba, V. Zantedeschi, I. M. Molloy, and B. Edwards, "Adversarial robustness toolbox v0.2.2," *CoRR*, vol. abs/1807.01069, 2018. [Online]. Available: http://arxiv.org/abs/1807.01069

[94] inconshreveable, "ngrok - secure introspectable tunnels to localhost," May 2020, [Online; accessed 19. May 2020]. [Online]. Available: https://ngrok.com

[95] Nvidia, "CUDA Toolkit 10.2 Download," Oct 2015, [Online; accessed 19. May 2020]. [Online]. Available: https://developer.nvidia.com/cuda-downloads

[96] M. Chaparro, "Robustness Analysis of Deep Neural Networks / project," May 2020, [Online; accessed 18. May 2020]. [Online]. Available: https://gitlab.pld.ttu.ee/miguel-chaparro-thesis/project

[97] fastai, "imagenette," Mar 2020, [Online; accessed 18. May 2020]. [Online]. Available: https://github.com/fastai/imagenette

[98] LeCun, "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges," May 2013, [Online; accessed 19. May 2020]. [Online]. Available: http://yann.lecun.com/exdb/mnist

[99] caglar, "Arcade-Universe," Jun 2015, [Online; accessed 19. May 2020]. [Online]. Available: https://github.com/caglar/Arcade-Universe