Technische Universität Berlin

Faculty IV - Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Department of Machine Learning and Intelligent Data Analysis

Marchstraße 23
10587 Berlin, Germany
www.ml.tu-berlin.de

Master Thesis

The Effects of Stability Training on the Robustness of Deep Neural Networks

Jan Laermann

Matriculation Number: 365898
9 April 2019

Examiner	Prof. Dr. Klaus-Robert Müller
	Prof. Dr. Thomas Wiegand
Supervisor	Dr. Wojciech Samek
	Dr. Nils Strodthoff

Fraunhofer
Heinrich Hertz Institute

Fraunhofer Institute for Telecommunications
Heinrich Hertz Institute, HHI
Einsteinufer 37
10587 Berlin, Germany

This thesis originated in cooperation with the Fraunhofer Institute for Telecommunications (HHI).

First of all I would like to thank Prof. Dr. Klaus-Robert Müller at Technische Universität Berlin for giving me the opportunity to carry out state-of-the-art research in this field.

Special thanks to Dr. Wojciech Samek and Dr. Nils Strodthoff for their guidance and support while working with them in the Machine Learning Group at Fraunhofer HHI.

Furthermore I would like to thank the rest of the group and peer students for their valuable feedback and encouragement.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den 9 April 2019

. . . . . .

(Unterschrift Jan Laermann)

Declaration

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, 9 April 2019

. . . . . .
(Signature Jan Laermann)

Abstract

Stability training is an optimization method introduced by Zheng et al. [2016] to improve the prediction stability of deep neural networks against small perturbations in the input. The proposed method allows a network to improve generalization over distortions introduced by common image processing, such as compression or rescaling. In this thesis, we explore stability training's capacity as a general alternative to data augmentation. We validate the method's performance against a number of distortion types and transformations commonly used with data augmentation. In addition, we propose modifications to the stability optimization objective and validate that they improve on some of the method's weaknesses. Finally, we evaluate stability training as a method to increase adversarial robustness and compare its effectiveness to that of standard adversarial training. We conclude that stability training is generally more useful as a method to improve generalization as it offers consistently significant improvements against a broader range of distortions, while also exhibiting significantly less severe negative side effects. Furthermore, our experiments suggest stability training to be a viable strategy of improving adversarial robustness.

Zusammenfassung

Zhang et al. [2018] führten *stability training* als eine Optimierungfunktion ein, um die Stabilität von *deep neural networks* gegen kleine Störungen im Eingangssignal zu verbessern. Die vorgeschlagene Methode erlaubt einem Netzwerk besser über Störsignale zu generalisieren, die durch gängige Bildberarbeitungsprozesse hervorgerufen werden. In dieser Thesis untersuchen wir die Möglichkeit, stability training als eine alternative zu *data augmentation* zu nutzen. Wir evaluieren das Verhalten beider Methoden im Bezug auf verschiedene Typen von Störsignalen sowie Transformationen die gebräuchlich beim Training mit data augmentation sind. Wir führen außerdem Modifikationen der Optimierungsfunktion von stability training ein und zeigen wie diese gewisse Schwächen der Methode lindern. Zuletzt evaluieren wir stability training als eine Methode, um Robustheit gegen *adversarial examples* zu steigern, und vergleichen dabei die Effektivität mit der von klassischem *adversarial training*. Zusammenfassend folgern wir, dass stability training, im Vergleich zu data augmentation, eine allgemein nützlichere Method zur Verbesserung der Generalisierungseigenschaften eines Netzwerkes darstellt, da sie konsequent signifikante Verbesserungen über ein breiteres Spektrum an Störsignalen produziert und zugleich deutlich weniger schwerwiegende Nebenwirkungen verursacht. Außerdem suggerieren unsere Experimente, dass stability training eine brauchbare Strategie ist, um die die Robustheit gegen adversarial examples zu steigern.

Contents

List of Figures

List of Tables

1. Introduction

Deep neural networks (DNN) are complex learning systems, which have been used in a variety of tasks with great success in recent times. In some fields, like visual recognition or playing games, DNNs can compete with or even outperform their human counterparts [Silver et al., 2016, Vinyals et al., 2017, He et al., 2015], showcasing their utility and effectiveness. While their complexity serves their purpose as general function approximators, they still need to find the right fit for the data. If they do not generalize well, because they simply internalize artifacts in the data, then they are said to overfit.

To counteract overfitting, regularization techniques, such as weight decay [Krogh and Hertz, 1991] or dropout [Srivastava et al., 2014] have been used. These regularization techniques introduce constraints on the complexity of a model. If the constraints are too strict, the model will be unable to capture the complexity of the data and is therefore underfitting the data. Another approach to counteract overfitting is to increase the amount of training examples. As labeled data is often scarce, adding perturbed copies of existing data samples has been shown to greatly increase the generalizability of a given model [Yaeger et al., 1997, Taylor and Nitschke, 2017]. This is called data augmentation. Unfortunately, as this effectively extends the data corpus, this improvement can lead to a performance loss on the original data distribution. In this work, we propose stability training [Zheng et al., 2016] as an alternative strategy to data augmentation that offers similar advantages with limited negative side effects. We also introduce modifications to the method, which mitigate weaknesses and extend its suitability.

Standard regularization techniques, such as weight decay or dropout, work by penalizing model complexity. Weight decay adds an optimization objective by imposing a penalty on strong neuron connections, forcing solutions to be diffused over the entire network [Zhang et al., 2018]. Dropout aims to regularize model complexity as well, but achieves its goal by randomly disabling connections between neurons for each sample during training. This imposes a constraint on the network, which effectively favors solutions that exhibit redundancies and therefore perform more robustly [Hinton et al., 2012]. Both these methods impose restrictions on the model's complexity and therefore on its capacity to fit complex problems. Hence, it becomes a trade-off between a necessary model complexity and a requirement for regularization that grows proportional to it. Data augmentation, instead, is network independent and is applied, as the name suggests, to the data instead of the model. The goal of this method is to increase the feature complexity inherent in the data, which implicitly forces a generalized solution as the complexity differential between model and data diminishes. This strategy has been proven to be efficient and effective, and was shown to match or outperform explicit regularization methods [Hernández-García and König, 2018], such as those mentioned above.

In real-world applications, DNNs often face un-curated data. This data might exhibit distortions that have not been encountered by the network before. In the case of image classification, examples could be different compression techniques that do not change semantic meaning of the image but introduce minute changes to pixel values. These small changes

often confuse unprepared models and can cause them to misclassify. An intuitive approach to improve this issue would be to add instances of the most prominent distortions to the training phase. This can be seen as a form of data augmentation. It has been shown, however, that this strategy is suboptimal. With an increasing number of distortion instances added to the training corpus, the model underfits the increasingly complex data and therefore performance on the original classification task deteriorates.

Stability training [Zheng et al., 2016] aims to improve robustness against said distortions without sacrificing classification performance. Instead of adding distortion instances to the training corpus, stability training generates images perturbed by Gaussian noise and feeds them simultaneously with the reference sample to the network. The network is then tasked to adjust itself, such that its output to the perturbed image becomes more similar to its output to the reference sample. This implicitly enforces a constraint on the sensitivity of a model to small changes in the input data. The method was shown to be effective at increasing a model's prediction stability with respect to image transformations that frequently occur in un-curated data from the internet.

We believe, that the inspiring work by Zheng et al. [2016] is very promising for varying scenarios beyond what was described by the authors and aim with this thesis to explore their method further. As mentioned above, data augmentation leads to underfitting if the data complexity is increased too severely due to the added training examples. Stability training addresses this problem successfully, but leads to a doubling of the effective batch size and therefore of the memory demand as well. Stability training was shown to generalize its robustness improvements learned from Gaussian noised samples to different transformations as well. However, while it would not bring the additional memory cost, data augmentation's utilization of these samples was not compared to stability training by the authors. Additionally, we wish to explore its utility in combination with perturbations other than Gaussian noise.

Furthermore, Zheng et al. [2016] showed that the artifacts which stability training aims to improve robustness against, can be seen as a form of naturally occurring adversarial example. In the domain of image classification, adversarial examples can be thought of as carefully crafted optical illusions for DNNs [Goodfellow et al., 2017] that are often undetectable by the human eye. A naive approach to specifically increase robustness against these adversarials is to add them to the training procedure. As this is a form of data augmentation, it would be interesting to see stability training's performance in contrast. We set results of both methods into context by comparing them with standard adversarial training [Goodfellow et al., 2015]. Lastly, we will investigate potential trade-offs and side effects that stability training might introduce and how its benefits and disadvantages compare to those of standard data augmentation in general.

As such, this work aims to explore the capabilities and limitations of stability training. We compare it against data augmentation in terms of its effects on both model generalizability and prediction stability. We also propose modifications to the original optimization objective that help mitigate weaknesses with a certain class of input distortions and allow us to simultaneously train multiple stability objectives independently. Lastly, we aim to explore stability training's capacity as a method to increase robustness against adversarial examples. In particular our contributions are as follows:

- We establish stability training as a generally superior alternative to data augmentation, which produces comparable or superior robustness improvements that affect a broader range of distortions even when trained with distortion types that are known to perform

well with data augmentation, while exhibiting significantly lower risk to deteriorate prediction performance. It is also able to generalize robustness learned from specific distortions across different distortions.

- We propose a *symmetrical* stability objective that increases stability training's effectiveness in learning from data transformations, like rotation, and offers superior performance to data augmentation in scenarios which are likely to be encountered in real-world applications.
- We evaluate stability training as an alternative to adversarial training to increase robustness against adversarial examples generated via the fast gradient sign method [Goodfellow et al., 2015].

2. Background

In this chapter, we introduce different concepts required as background knowledge in order to understand or give context to stability training and our interest in exploring it further. We start with an introduction into machine learning and specifically deep learning by explaining how neural networks are generally trained and how they differ from traditional learning algorithms.

A common problem for deep neural networks and traditional methods is that they try to infer a general rule from patterns found in a finite set of data. These patterns might be common among the overall data distribution or to the encountered data set alone. We therefore discuss different regularization strategies aimed at balancing how strictly a DNN is constraint by them. We differentiate between regularization methods that explicitly impose constraints on the model and those that augment the model's input to implicitly achieve a similar goal.

In this thesis, we are interested in a model's prediction robustness to perturbations of the input, that is, how sensitive it is to distortions of the input that we would like it to be invariant to. We therefore discuss a special subset of inputs that has been purposefully crafted to confuse a neural network. In the image domain, these *adversarial examples* are sometimes regarded as optical illusions for DNNs that are nearly imperceptibly different to natural images for the human eye [Goodfellow et al., 2017]. In particular, we discuss a specific method of generating these examples, known as *fast gradient sign method*, and traditional strategies to minimize a model's sensitivity to such *attacks*.

2.1. Deep Learning

Deep learning is a subfield of artificial intelligence, and more specifically machine learning. Deep learning algorithms differ from traditional task-specific alternatives in a variety of ways, importantly also in their capacity to learn complex patterns from raw data. Deep learning architectures such as deep feed-forward convolutional neural networks or recurrent neural networks do not require predefined rules in order to complete a task. Instead, they learn from patterns within the data presented to them by extracting embedded features from the data itself.

Training a network can be done in different fashions, typically divided into three main categories of supervised, unsupervised and reinforcement learning. Broadly speaking the difference of these categories is that, supervised learning strategies provide the network with example data and corresponding targets, which enables the network to adjust its own predictions in order to meet these targets. Typical examples are classifying images by the categories of objects present in them or recognizing handwriting in order to transcribe it into digital format. It is supervised in the sense that, some supervisor needs to provide these pairs of ground truth information.

Unsupervised learning does not provide such targets. Instead, these learning strategies focus on finding commonalities within the data itself in order to allow the network to categorize

data itself. For example, deciding if an email belongs into the spam folder could be learned this way by learning the defining qualities of a proper mail opposed to those present in spam.

Self-organization learning and clustering are terms often used interchangeably with unsupervised learning. Reinforcement learning describes the category of learning strategies, where the learning algorithms does not have access to targets in order to validate the correctness of each of its predictions, but instead interacts with an environment that enforces its own rules upon the network. Self-driving cars or self-taught game agents are examples, where the network learns by interacting with such an environment and learning from rewarding or punishing experiences.

All of these approaches are loosely inspired by our understanding of biological processes occurring during learning, in the sense that they mimic a networked structure of neurons that communicate via action potentials as first described in Hebbian learning [Hebb, 1949].

The fundamental building block of a neural network, the perceptron, was formalized by Rosenblatt [1958], describing the functionality of such a neuron that takes several inputs (the data) and depending on an adaptive rule decides under which conditions to output a signal of its own. Stacking these perceptrons in parallel and chaining them sequentially as layers of a network were one of the first advances and introduced the so called multilayer perceptron [Bishop, 2006].

These fully connected networks, that is, a network where each neuron in any given layer has an output connection to each neuron in the next layer, can be trained to classify input data among a given set of classes without the need to provide a set of predefined rules. Instead, the network's architecture was designed, such that for each possible class the network had a neuron in its (last) output layer, where it could output its prediction of how probable this class was given an input. The output at each of these output neurons depends on the output of neurons that came before them, back until the beginning of the network where the input layer of neurons is fed the actual data. Therefore, the network produces an output, given a certain input, which can be interpreted as the network's prediction.

In supervised learning tasks, we can calculate the error of our prediction with respect to the true class label. In multi-class classification, the measure for this error is often chosen to be the discrete cross-entropy [Murphy, 2012]. This measurement, often called the *loss*, can be used as an optimization objective to minimize the prediction error. Neural networks are often optimized via gradient-based methods like stochastic gradient descent [Robbins and Monro, 1951] This method iterates backwards through the network and adjusts the neurons, such that the error between the model output and the target output is minimized, based on the impact they had to produce this error. This gives the multilayer perceptron the capacity to be a universal approximator [Hornik et al., 1989] of functions in the euclidean space, given a sufficient number neurons and at least a single layer.

Modern architectures have since increased in developed further, introducing more complex building blocks, such as convolution or pooling layers, or restrictions on the connections between neurons. The key advantage of neural networks here is, however, their ability to learning directly from data itself, which led to their widespread use and success in a large variety of fields in recent years. Still, a key problem that remains, is the ability to transfer knowledge to novel scenarios.

2.2. Generalization performance

In classification applications, neural networks are tasked to learn from a set of examples. For this to work, it is vital that the examples are representative of the true distribution of data on which the network is tasked to operate. As the set of training examples is finite, however, it is bound to contain a sampling error that defers from the true distribution. The predictive capabilities of a model must therefore be evaluated on previously unseen data and its performance is then referred to as its generalization error or generalization gap.

To achieve this, a portion of the available data is typically reserved to be used solely for evaluation purposes. As a some continuous learning strategies incorporate their generalization error into the training procedure, it is also common to reserve another portion for intermediate validation. This ensures that any decisions made on the performance on the validation portion do not bias the generalization error estimation on the testing portion. Splitting the data into such portions is called cross-validation.

If a model becomes too sensitive to sampling error in the data, we call it overfitting. An overfitted model will perform poorly on data outside of the training set, as those samples will not share the feature expressions introduces by the sampling error. The more complex a model is, the more capable it is to detect and extract features less prominent features in the data. That is, because with unlimited bounds to its complexity the model will be able to perfectly fit every sample from the training set, therefore also fitting the sampling error. As such, the model complexity must therefore be large enough to capture the complexity of the data, but not too large as to avoid overfitting. This is known as the bias-variance trade-off, where being too restrictive with regard to model complexity biases the set of possible solution, but being too lenient allows the network to overfit, creating a high variance in the prediction quality.

2.2.1. Regularization

Regularizors are methods that implement a way to balance the previously introduced bias-variance trade-off. There are several options with varying effects on the model and its predictive capabilities. In this section we distinguish between classical explicit regularization methods that add a regularization term to the learning objective, and those that achieve a similar goal, but rather augment the training data to implicitly counteract unwanted learning behavior.

Weight decay

The weight decay regularization method (also known as ridge regression or Tikhonov regularization) is a simple technique that penalizes large connection weights between neurons. The most commonly used form of weight decay, L^2 weight decay [Goodfellow et al., 2016], adds a regularization term to the original optimization objective J_0, also called loss function, such that

$$J = J_0\left(f_w(x), y\right) + \lambda||W||_2^2, \tag{2.1}$$

where f is our model, x represents the input data, y the true target label, λ a scaling term, and w a weight parameter in the weight matrix W of our network. Large absolute values in the weight matrix therefore increase the value of our training objective that we try to minimize.

Gradient descent evaluates the impact of each weight parameter onto the total loss J in order to produce an update step. If we look at the impact of weight decay, we find

$$\nabla_w \lambda ||W||_2^2 = \nabla_w \lambda \sum w^2 = 2\lambda w, \tag{2.2}$$

which shows a linear impact of w onto the update step it receives. The total weight update can then be written as

$$w \leftarrow w - \alpha \left[2\lambda w - \nabla_w J_0(f(x), y) \right]. \tag{2.3}$$

A more intuitive regularization term might come to mind in the form of

$$J = J_0 \left(f_w(x), y \right) + \lambda \sum |w|, \tag{2.4}$$

which is known as L^1-regularization.

L^1-regularization does not provide a linear correspondence between weight value and update. Rather, its impact is given by

$$\nabla_w \lambda \sum |w| = \lambda \text{sign}(w), \tag{2.5}$$

which drives the weight matrix to be sparse, meaning that as many weights as possible will be forced to 0 in order to minimize the loss.

Dropout

Dropout [Srivastava et al., 2014] can be thought of as an inexpensive way to train an ensemble of smaller networks and combine their knowledge into a larger unified network. Ensemble training is a strategy to overcome biases of single networks and leverage the power of group knowledge. For example, Szegedy et al. [2015] used an ensemble of size six to win the ILSVRC 2014 challenge. However, for large networks and datasets this quickly becomes cumbersome. This procedure provides an implicit force onto the network that prevents co-adaptions to arise, a phenomenon where certain feature detectors in the network depend on the existence of other feature detectors to work properly.

Dropout can approximate ensemble training by randomly disabling non-output neurons in the network during training. The such created virtual submodels all share the same weights, which makes this approach different form actual ensemble training, but grants a significant improvement in runtime and memory usage.

During inference, ensemble training would then let each model cast a vote on its prediction, thus the prediction of the ensemble would be given by the mean of all predictions,

$$\frac{1}{k} \sum_{i=1}^{k} p_i(y|x). \tag{2.6}$$

Dropout, however, allows us to simply re-enable all neurons in the network, weighting their output with their probability to be enabled during training, and take the network's prediction as the mean prediction of the virtual ensemble [Hinton et al., 2012].

Data augmentation

The sampling error introduced by sampling examples from the true distribution is directly correlated with the size of the sampled set. Increasing the size of the sampled set thus reduces the sampling error and would hence yield better models. However, training data is often scarce, especially in classification tasks, where one does not only need data, but also correct class labels.

For visual object recognition, the need for more training data coincides with the desire for the model to be robust against transformations of the image. For example, we would like our classifier to be invariant to small rotations and translations of the image. Therefore, we can simply generate new training data by augmenting existing samples and assigning the original class label. Even injecting noise into the image can be seen as a form of data augmentation [Sietsma and Dow, 1991] that helps neural networks improve their natural low robustness against it [Tang and Eliasmith, 2010].

When comparing two algorithms it is therefore essential to apply the same data transformations in both cases. The regularizing effects of the augmentations are otherwise likely to overshadow the true difference in performance of the learning algorithms [Goodfellow et al., 2016].

2.2.2. Adversarial Examples

Deep neural networks have reached human performance in classifying objects from several test data sets. Szegedy et al. [2013] were able to show that even these models can be fooled easily by carefully augmented inputs. They constructed a perturbed input x' of the original reference image x, such that the difference between the model's responses to both images would be large. They found that networks could be fooled almost all the time, even if the similarity between the original examples and the *adversarial example* was such that human observers could not tell them apart. While there arise many interesting implications from this, Goodfellow et al. [2015] found that training on adversarial examples could decrease the generalization error of the network on the original test set. This is known as adversarial training. Specifically, the optimization objective L_{adv} is given as

$$L_{adv} = \mu L_0(x;\theta) + (1-\mu)L_0(x';\theta); \mu \in [0,1], \tag{2.7}$$

where L_0 is the original loss function. Note that this training objective doubles the batch size, as for each input x there will be a perturbed input x' passed through the network simultaneously.

There exist various methods for different purposes for generating adversarial examples. A simple yet effective technique is called fast gradient sign method [Goodfellow et al., 2015]. Here the sign of the elements of the gradient of the loss function with respect to the input is added to the input image,

$$x' = x + \epsilon \operatorname{sign}\left(\nabla_x J(f_w(x), y)\right), \tag{2.8}$$

where ϵ is a scaling factor to be chosen.

The gradient $\nabla_x J$ tells us which changes to x would increase J. Therefore, adding $\nabla_x J$ onto the input x will increase our loss by implicitly confusing the network to produce false predictions. The sign method is simply an implementation detail, which can reduce computational load. The scaling factor ϵ regulates the intensity of the perturbation, the smaller ϵ,

the smaller the actual change that is being applied to the image and the harder to visually distinguish the adversarial example from the original image.

3. Stability Training

Zheng et al. [2016] proposed a new method in an effort to stabilize deep neural networks with respect to small input perturbations. In their work, they claimed that common image processing steps such as compression, resizing and post-processing by the user introduce distortions, which confuse networks and thus reduce their utility in large scale data applications as these tend to be used on un-curated data sets. Furthermore, they claimed that the artifacts introduced via such methods produce images with similar qualities to purposefully crafted adversarial examples (see Section 2.2.2). To address this issue, they formulated a general training framework called *stability training*.

In this chapter, we want to introduce stability training, and propose and motivate our extensions. We first discuss how stability training integrates into the original training objective, by splitting the implicit regularization mechanic, as known from data augmentation (see Section 2.2.1), from the machinery that drives learning the original task.

Stability training increases robustness to input perturbations by minimizing its variance in response to such perturbations and the original input itself. We therefore formally introduce different perturbation sampling methods that will be used throughout our experiments in Chapter 4.

3.1. Stability Objective

Stability training has the goal to stabilize predictions of a deep neural network $\mathcal{N}$ in response to small input transformations. The method aims to achieve this by flattening the learned output function $f(x)$ of the network around inputs $x \in [0,1]^{w \times h}$, where x is an image of size $w \times h$. An images x', which is visually similar to x and semantically equivalent therefore ought to produce similar outputs $f(x')$. That is, given an appropriate choice of a distance metric D for the outputs of f and d describing the distance in input space, the goal can be formalized as

$$\forall x' : d(x, x') \text{ small } \iff D(f(x), f(x')) \text{ small.} \tag{3.1}$$

The full optimization objective can then be constructed as a composite loss function of the original task L_0, for example cross-entropy between the network's prediction $y = f(x)$ and ground truth label $\hat{y}$, and a separate stability objective L_{stab} which enforces the condition form above. Given an original image x and a perturbed version x' of it, the objective is then given by

$$L(x, x'; \theta) = L_0(x; \theta) + \alpha L_{stab}(x, x'; \theta), \tag{3.2}$$

$$L_{stab}(x, x'; \theta) = D(f(x), f(x')), \tag{3.3}$$

where α is a hyper-parameter to regulate the impact of the stabilization term onto the loss function and θ describes the parameters of our neural network $\mathcal{N}$.

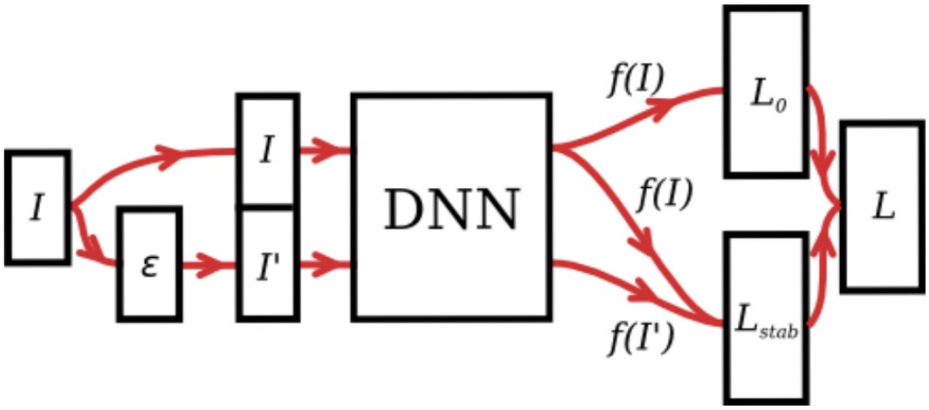


Figure 3.1.: Stability pipeline (taken from [Zheng et al., 2016]).

The choice of a distance function D is task specific. The Kullback-Leibler divergence [Kullback and Leibler, 1951] is a measure of how different a probability distribution is from a reference probability distribution. It therefore offers itself as a measure of distance between the network's outputs $y' = f(x')$ and $y = f(x)$, such that

$$D(y, y') = D_{KL}(s(y) \parallel s(y')) = H(s(y), s(y')) + H(s(y)), \tag{3.4}$$

where H refers to the cross-entropy and entropy [Shannon, 1948], respectively, and s is the softmax function [Gibbs, 1902, Bridle, 1990] that transforms a vector of real numbers into a probability distribution. The stability loss function is then given by

$$L_{stab}(x, x'; \theta) = D_{KL}(s(y) \parallel s(y')) \tag{3.5}$$

$$= \sum_j P(y_j|x; \theta) \log \left(\frac{P(y_j|x; \theta)}{P(y'_j|x'; \theta)} \right). \tag{3.6}$$

We then optimize our model by minimizing the complete loss function from Equation (3.2), such that we find

$$\theta^* = \arg\min_\theta \sum_{x_i \in \mathcal{D}; d(x,x') < \epsilon} L(x, x'; \theta). \tag{3.7}$$

3.1.1. Symmetrical stability training

Using Kullback-Leibler divergence as the distance measure D has certain implications. As $D_{KL}(s(y) \parallel s(y')) \neq D_{KL}(s(y') \parallel s(y))$, we see that this drives the network to change its response to the perturbed input x' to be similar to that of x.

This works well for perturbations, where the resulting image x' can be clearly identified as a image with an additional noise signal, as is the case for Gaussian noise for example. That is, because we want the network to respond to the noisy image more like it responds to the reference image. For some perturbations, however, this is not the case. If the perturbed image is generated via rotation of the reference image, for example, there is no good reason to favor the model's response to either input. In this case, the asymmetry in our objective described

above will introduce an unwarranted bias into the optimization of our network. We group distortions that exhibit this trait into a category that we call *transformative* distortions.

To alleviate this issue, we propose a partially and a fully symmetrical optimization objective, which we evaluate in Section 4.2.3. We define a symmetrical stability objective as

$$\begin{aligned} L_{sym\_stab}(x, x'; \theta) = L_0(x; \theta) + \alpha_1 L_{stab}(x, x'; \theta) \\ + \alpha_2 L_{stab}(x', x; \theta), \end{aligned} \tag{3.8}$$

and a fully symmetrical optimization objective as

$$L_{sym\_full}(x, x'; \theta) = \frac{1}{2} \begin{pmatrix} L_0(x; \theta) + \alpha_1 L_{stab}(x, x'; \theta) \\ + L_0(x'; \theta) + \alpha_2 L_{stab}(x', x; \theta) \end{pmatrix}. \tag{3.9}$$

Both are designed to work around the asymmetry introduced through the choice of distance function D. The fully symmetrical objective simply treats both x and x' as if they were reference images. It applies the original objective from Equation (3.2) to both scenarios and averages the result.

In contrast to the fully symmetrical loss, the symmetrical stability objective only affects the stability term of the objective. This is more in line with the initial motivation of stability training, where evaluation on the original task L_0 was restricted to samples from the original data set $\mathcal{D}$. However, as perturbed samples generated via perturbation methods within the transformative category cannot be easily distinguished from their reference counterparts, it is not trivial whether this restriction should still apply.

3.1.2. Multiple independent distortions

We also propose a generalization of the stability optimization objective from Equation (3.2) to incorporate an arbitrary number of distortions independently. The new generalized loss function is given by

$$L_{multi}(x, x'; \theta) = L_0(x; \theta) + \sum_{i}^{n} \alpha_i L^i_{stability}(x, x'_i; \theta), \tag{3.10}$$

where n refers to the number of independent distortions to be used. Note, that this increases the batch size with each added distortion. Instead of producing one perturbed image, with n distortions applied to it, we create n perturbed images each distorted with a single distortion and evaluate a respective stability objective with respect to the reference image.

Stability training combined with data augmentation

As we mentioned earlier in Section 3.1.1, we expect stability training to perform poorly on *transformative* training distortions, while data augmentation is generally known to perform well on them. A natural step would therefore be to formulate a combined optimization objective that exhibits strengths of both methods. We can simply adapt the standard stability training procedure to

$$L(x', x''; \theta) = L_0(x'; \theta) + \alpha L_{stab}(x', x''; \theta); \quad x' = t_d(x), \quad x'' = t_s(x'), \tag{3.11}$$

where t_d and t_s are the distortions we want to apply for the data augmentation and stability training part, respectively. This allows us to extend the scope of our original task to include data generated via t_d, for example via rotation, and still split the regularization achieved via stability training's distortion t_s.

3.1.3. Contrast to data augmentation

When training with data augmentation, one would create a new dataset $\mathcal{D}^*$, which incorporates the original training dataset $\mathcal{D}$ and all the perturbed versions of the samples within it. The training procedure would then evaluate the original objective L_0 on the entire set of $\mathcal{D}^*$. While decreasing the generalization error, this may also lead to inferior performance on the original data set $\mathcal{D}$.

Stability training is fundamentally different from data augmentation in this regard. The original task L_0 is only being evaluated on samples from the original dataset $\mathcal{D}$, while perturbed samples are only used with the purpose of stabilizing the network's output via the stability objective L_{stab}. Note, that this is importantly not the case for the fully symmetrical optimization objective from Equation (3.9).

While decoupling learning to solve the original task from stabilizing class predictions allows stability training to achieve a low generalization error without sacrificing performance on L_0, this also increases memory consumption, as for each forward pass through the model, there needs to be a perturbed input x' for every x. This effectively doubles the batch size, as shown in Figure 3.1.

3.2. Perturbation Sampling

During training we need to generate a perturbed version x' of the original reference image x (see Figure 3.1) for each sample in the dataset. Both images are then passed forward through our network and the respective responses used to evaluate our stability objective from Equation (3.2). The choice of the perturbation has implications on the effect stability training has on the robustness of the network. Data augmentation procedures typically use perturbations which are expected to be encountered during deployment of the network. However, this strategy is limited by the number different perturbations one can feasibly add to the data pre-processing pipeline. There arises a trade-off between the gain in generalizability and robustness, and the loss in performance on the actual task. Increasing the number of perturbations will shadow the features inherent in the original data distribution, thus rendering the network less rather than more effective. A more desirable solution would be a perturbation method that allows the network to increase its robustness to input distortions universally.

Table 3.1.: Category assignment of distortion types.

Category	Distortions
Noise and Artifacts	Gaussian noise, JPEG Compression, thumbnail resizing, fast gradient sign adversarial examples
Transformative	rotation, random offset cropping

[Zheng et al., 2016] proposed to use Gaussian noise for this purpose. They claimed, as Gaussian noise could represent several types of distortion, it could allow the network to share

its robustness gains across many kinds of perturbations. In this section, we formally introduce Gaussian noise and several alternative sampling methods for generating the perturbed input x'. Based on the observations from Section 3.1.1, we assign them to their respective category as shown in Table 3.1. To provide an intuitive understanding of any given perturbation distorts the input at different intensity levels, we show exemplary perturbations for each method. Note, that the distortion intensities depicted coincide with those used for testing model robustness Chapter 4.

3.2.1. Gaussian Noise

We add pixel-wise uncorrelated Gaussian noise to each image during training. This method allows to potentially represent various forms of input perturbation. For every pixel k in the reference image x, the value of the same pixel in the perturbed image x' is therefore given by

$$x'_k = x_k + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma_k^2), \quad \sigma_k > 0, \quad x \in \mathcal{D}_{train}. \tag{3.12}$$

Here, σ^2 denotes the variance of the Gaussian distribution from which we sample our noise, which is fixed across all pixels and remains a hyper-parameter to be optimized. We later refer to a setting using Gaussian noise and a certain σ as `GAUSS-`σ.

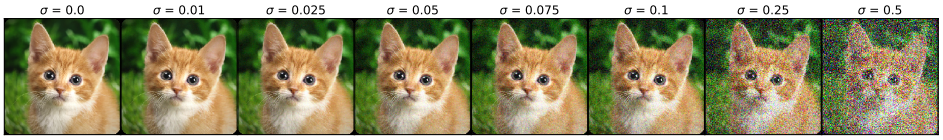


Figure 3.2.: Pixel-wise uncorrelated Gaussian noise distortion applied with increasing standard deviation σ from left to right.

3.2.2. JPEG Compression

JPEG compression is a technique often used to reduce the size of digital photographs without losing much perceived visual fidelity. The exact mechanics are beyond the scope of this thesis. It shall suffice to say that the method offers the user to adjust the level of quality, ranging from 100 to 1, that is to be retained in the compressed result. The perturbed image x' is generated as a compressed version of x given

$$x' = \text{JPEG}(x, q), \quad q \in [0, 1], \quad x \in \mathcal{D}_{train}, \tag{3.13}$$

where q refers to the JPEG quality level. We will refer to a setting using JPEG compression with a certain quality setting q as `JPEG-`q.

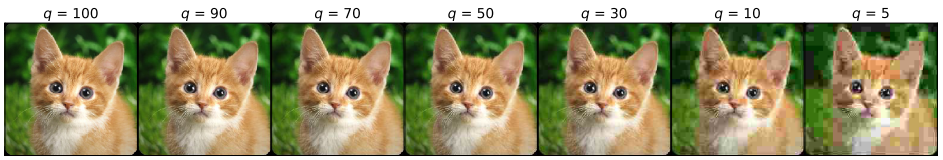


Figure 3.3.: JPEG Compression distortion applied with increasing JPEG quality level q from left to right.

3.2.3. Rotation

Standard image rotation by $\gamma°$. Each reference sample from the training dataset is transformed to generate x', such that

$$x' = R(\gamma)x, \quad \gamma \in \mathcal{U}\{-\rho, \rho\}, \quad x \in \mathcal{D}_{train}, \tag{3.14}$$

where ρ is the maximum degree of rotation from which to sample γ at random, and R is the rotation matrix. We will refer to settings using rotation with a certain maximum rotation ρ as `ROT-`ρ.

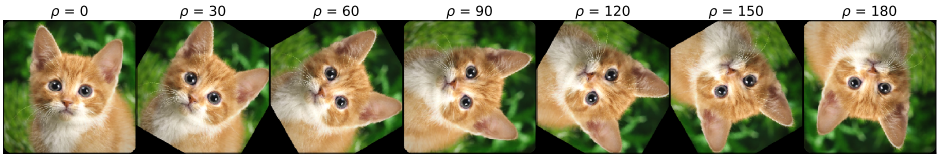


Figure 3.4.: Random rotation applied distortion with increasing maximum rotation degree ρ from left to right.

3.2.4. Random offset cropping

Here, we alter the common pre-processing procedure described in Section 4.1.1. After scaling the image, such that its shorter side is 256 pixels long, we crop a quadratic area of 224 pixels in width. In contrast to our standard pre-processing, however, we offset the center of the crop by c pixels from the center of scaled image, where c is chosen randomly, such that $c \in \mathcal{U}\{-o, o\}$. We will refer to settings that use random offset cropping with an absolute maximum offset o as `CROP-`o.

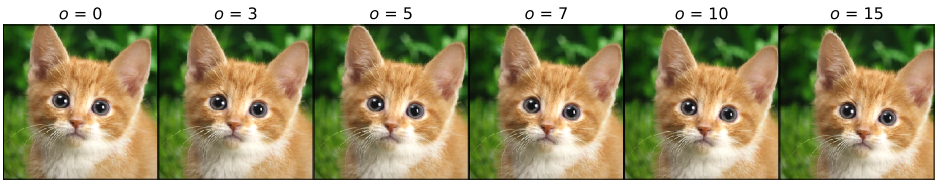


Figure 3.5.: Random offset crop distortion applied distortion with increasing offset o from left to right.

3.2.5. Thumbnail resizing

In this method, the image is down sampled to a smaller image size and then re-upscaled to its original size. This introduces noise to the image, as during down scaling pixel values have to be fused, and during up scaling additional pixel values have to be interpolated. We use bilinear interpolation in our experiments.

We will use `THUMB-`A to refer to this method, where the width of the intermediate quadratic image is given by A.

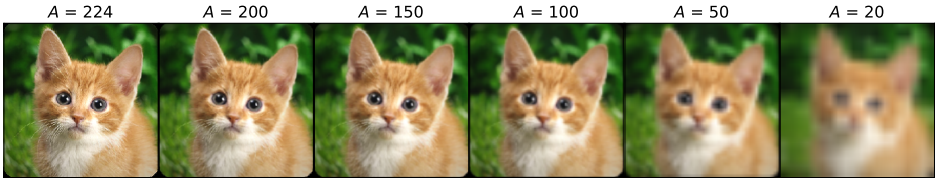


Figure 3.6.: Thumbnail resizing distortion applied distortion with increasing thumbnail size A from left to right.

3.2.6. Fast gradient sign method

We dynamically generate adversarial examples, as discussed in Section 2.2.2, during training of the current model. In this case, we use the *fast gradient sign method* [Goodfellow et al., 2015], such that for every pixel k in the reference image, the same pixel in the distorted image is given by

$$x'_k = x_k + \epsilon \operatorname{sign}\left(\nabla_x J(\theta, x_k, y)\right), \tag{3.15}$$

where $\nabla_x J(\theta, x, y)$ is the gradient of the loss of the network output with respect to the input. The weighting term ϵ then scales the intensity of the distortion. We refer to this setting as `FGSM-`ϵ.

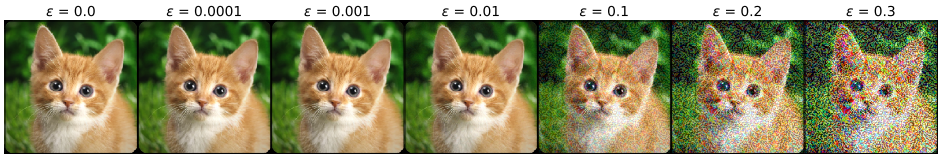


Figure 3.7.: Fast gradient sign method distortion applied distortion with increasing scaling coefficient ϵ from left to right.

4. Experimental Analysis

In real world applications, models are often presented with un-curated input data, such as images that have undergone some form of post-processing as for example compression or aesthetic visual filtering. In the context of compression, these procedures are designed to change the input in a way that reduces information contained in the data, but limits changes to semantics and perceived change by a human. These procedures introduce perturbations into the image which can confuse deep neural network classifiers.

A natural attempt to combat these issues is to add instance of these distortions to the training corpus, i.e. to use data augmentation. While often successfully increasing robustness against distortions similar to those added to the training corpus, we show that this procedure can lead to significantly worse performance on the original data distribution.

The following sections are divided as follows. First, we introduce how we conducted our experiments and our setup. We will then give a description of each of our experiments and the motivation behind them. For each of them, we present our results and point out important observations. We conclude with a discussion of our results to put them into context of our overarching question of how stability training compares to data augmentation as means to increase robustness and decrease a model's generalization error.

4.1. Implementation details

In this section, we discuss the common context and conduct of our experiments. We will provide a motivation for our choice of the dataset and network architecture, and describe how we optimize our models and apply the relevant distortions. Lastly, we explain how we compare results and evaluate their fitness, as well as how we condense our results into meaningful and informative plots.

4.1.1. Dataset

Zheng et al. [2016] emphasized the need for robust performance with respect to prediction stability of image classifiers on large scale data. As a consequence, they chose the ImageNet dataset form the ILSVRC 2012 challenge [Russakovsky et al., 2015]. This dataset contains 1.2 million high resolution (average resolution of 473x406 pixels) images, split into 1000 classes for the training set, and an additional 50 samples of each class for the validation set. Due to time and computational limitations we looked for a decent alternative.

Tiny ImageNet[1] is a subset of ILSVRC2012, where for the training set 500 images and for validation 50 images of a subset of 200 classes where chosen and down-scaled to a size of 64x64 pixels. As we are interested in performance on data which might occur in real-world applications, we decided to create our own hybrid dataset called Tiny ImageNet Fullsized (TIFS).

[1] From Stanford University's cs231 course: `https://tiny-imagenet.herokuapp.com/`

TIFS contains the same classes and number of samples as the Tiny ImageNet dataset, but each sample is the original from the ImageNet dataset (see Appendix A). As no labels are provided with the official ImageNet test set, we used the images from the validation set as the basis of our test set and split the training set in a cross-validation fashion, such that for each class we randomly assigned 450 samples to the training set and 50 to the validation set. We purposefully kept the number of samples per class equal to avoid any bias.

Images contained in the ILSVRC2012 ImageNet dataset and thus in our TIFS dataset come at varying sizes. Due to the architecture of our chosen model (see Section 4.1.2), every image in the dataset is first resized, such that the shortest side is 256 pixels in length. Next, we crop out a 224 pixels wide quadratic area, such that the center points of the crop and the image coincide. The image is then converted from RGB integer color values ranging 0 to 255, to floating-point values ranging 0 to 1. Finally, we normalize the image, such that the mean pixel value across the entire dataset is 0 and whiten the dataset, such that the mean variance is 1. Note, that this differs slightly from the commonly used pre-processing pipeline used in ImageNet training, in that we crop at a *fixed* position of the rescaled image even during training. Distortions are purposefully applied before normalization occurs. This helps simulate how such distorted images would be encountered in real-world applications.

4.1.2. Network

In our experiments, we used a state-of-the-art architecture in form of a convolutional deep residual network (ResNet18) [He et al., 2016]. This type of network, although a larger variant of it, won the ILSVRC2015 challenge due to its introduction of skip-connections, which allow deep models to be trained efficiently. These connections allow the gradient flow to skip layers, which helps prevent the vanishing gradient problem.

Veit et al. [2016] explain that deep networks usually produce worse results than shallow ones, as the information produced by early layers becomes muddied after passing through to the deeper ones. They claim a residual network can be thought of as an ensemble of shallow networks of different complexities, which together provide the benefits of deep models while being easy to optimize.

We chose this network over alternatives like the Inception network [Szegedy et al., 2015] or VGGNet [Simonyan and Zisserman, 2014] as it allows us to use a deeper model. It also allows us to perform experiments in a much more reasonable amount of time as the network components are comparably small.

4.1.3. Optimization

To optimize our model, we solved the problem from Equation (3.2) if not stated differently for specific experiments (see Sections 4.2.3 and 4.2.4). We used mini-batch stochastic gradient descent with Nesterov accelerated momentum [Nesterov, 1983] and batch normalization [Ioffe and Szegedy, 2015].

The different distortions come with their own set of hyper-parameters to tune. We optimize each configuration, that is the stability regularization coefficient α and the distortion dependent parameter, via grid search. The search ranges are displayed in Table 4.1. As the learning rate λ, we found 0.01 to consistently provide us with the best results and therefore decided to keep it fixed.

We found that introducing the stability objective late during the training phase produces

Table 4.1.: Grid search ranges for hyper-parameters used in our experiments.

Hyper-parameter	Start value	End value	Scale
Regularization coefficient α	0.01	10.0	logarithmic
JPEG compression quality q	90	10	linear
Gaussian noise standard deviation σ	0.01	1.	logarithmic
Obstacle side length l	10	150	linear
Crop offset o	0	15	linear
Rotation degree ρ	0	180	linear
Thumbnail size A	20	200	linear
FGSM coefficient ϵ	0.001	1.0	logarithmic

similar results to applying it from the beginning.However, we did not restrict weight updates to the last fully connected layers, as is common when fine-tuning models, as we found that this can limit the regularization effect.

We initially trained a model for 30 epochs on the clean training set described in Section 4.1.1 with no distortions added to the pre-processing procedure. The weights of the model were saved after each epoch and their performance validated on the validation set. The model iteration with the highest validation accuracy was then evaluated on the test set and used as the baseline model in the following experiments. All subsequent models are initialized with the weights of our baseline model and continue training from thereon.

In an individual run using either stability training or data augmentation, the model was trained in the same fashion as our initial baseline, expect being limited to 10 epochs. Note that only the training set is affected by any distortions applied to the data.

4.1.4. Experimental conduct

Experiments are conducted in a consistent manner, which we describe in the following. First, we introduce some terminology: We call the type of distortion to be used for regularization with stability training or data augmentation during training our *training distortion*. As described in Section 4.1.3, distortions bring their own parameter to regulate their intensity. We therefore refer to a *specific* training distortion and intensity value pair by the short hands introduced in Section 3.2, for example we will refer to JPEG compression with intensity level 30 as `JPEG-30`. The combination of a specific training distortion, the choice of the regularization function, i.e. data augmentation or stability training, and a specific selection of all other relevant hyper-parameters (see Table 4.1), we call a *training setting*.

Evaluating a training setting

Training in a given training setting is conducted as described in Section 4.1.3, that is we initialize our model with the parameters of our previously trained baseline model and continue training according to the selected setting. We then evaluate he performance of the final model, i.e. its average accuracy, on a *test setting*. A test setting is, analogously to the training setting, a specific selection of a distortion type and intensity level pair to be applied to the entire test dataset, where we call this distortion type our *test distortion*. To produce a richer overview of the robustness of a model, we group the results of all test settings by their test distortion.

Table 4.2.: Distortion intensities used to evaluate prediction robustness.

Distortion parameter	Values
JPEG compression quality q	100, 90, 70, 50, 30, 10, 5
Gaussian noise standard deviation σ	0., 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5
Obstacle side length l	0, 25, 50, 57, 100, 150
Crop offset o	0, 3, 5, 7, 10, 15
Rotation degree ρ	0, 30, 60, 90, 120, 150, 180
Thumbnail size A	224, 200, 150, 100, 50, 20
FGSM coefficient ϵ	$1e-5 * (\frac{5}{1e-5})^{\frac{i}{75}}; \forall i \in [0, 75]$

For example, if we choose stability training with training distortion `GAUSS-0.05` to be our training setting and JPEG to be our test distortion, we will evaluate the performance, of the model on all test settings of JPEG (see Table 4.2) and present a single plot showing the result of all test settings. As such, we do not have a single number indicating a model's robustness to a test distortion, but rather multiple measurements for several intensity levels of that distortion type.

Determining the best model

As we do not have single number indicating a model's robustness to a specific distortion, it is not trivial to determine the best model, when comparing different training settings. For example, training with `GAUSS-0.05` might produce superior results on the test setting `JPEG-90` compared to training with `GAUSS-0.1`, but the reverse might be the cause for testing on `JPEG-10`.

We therefore define three scenarios, *no* distortion, *practical* distortion, and *extreme* distortion and determine the best model given each of them. These scenarios are just a selection of three test settings belonging to the same test distortion, which we deem interesting. No distortion is equal to the original task. The samples from the test dataset are not augmented and performance in this scenario indicate the capacity of a regularization technique to retain the knowledge of the baseline model. Practical distortion is chosen subjectively and aims to resemble a distortion that we consider likely to be encountered in real world applications. Or in certain settings, like FGSM, where the distortion is on the verge of being detectable by a human, but has not yet surpassed that threshold. Extreme distortion is chosen to explore the effects in the extremes. This is mostly relevant for academic purposes, with the intention of providing a richer illustration of the generalization performance of a given model.

For each scenario, we are then able to determine the training setting that performed best and plot the evaluation performance on all test settings. This allows us to explore how the training setting that performed best given a specific test setting, i.e. the scenario, impacts the model's performance across all intensity levels of the same test distortion type. We do the same for the training setting that produced the worst results in a scenario to help us explore potential negative effects that might occur. For stability training, we also draw a region indicating the absolute upper and lower bounds of any training setting at any test setting. This allows us to find potential outliers that we might otherwise miss.

As an example to put everything into context, we can think of investigating how well thumbnail resizing as a training distortion works to improve the baseline model's robustness against

Table 4.3.: Intensities of all distortion types chosen to be scenarios.

Distortion setting	No distortion	Practical distortion	Extreme distortion
JPEG compression	JPEG-100	JPEG-30	JPEG-5
Gaussian noise	GAUSS-0.0	GAUSS-0.05	GAUSS-0.25
Random offset cropping	CROP-0	CROP-3	CROP-15
Random rotation	ROT-0	ROT-30	ROT-150
Thumbnail resizing	THUMB-224	THUMB-150	THUMB-50
FGSM	FGSM-0.0	FGSM-0.001	FGSM-0.1

artifacts introduced via JPEG compression. In this case, we first train several copies of our baseline for 10 epochs with stability training or data augmentation using various training settings THUMB-A (see Table 4.1). We would then evaluate all resulting models on all test settings JPEG-q (see Table 4.2). Next, we determine for stability training and data augmentation individually, which of their training settings performed best and worst under each of the three scenarios (see Table 4.3). Finally, for each scenario we plot the results of the four respectively chosen training settings and indicate the absolute bounds of all stability training settings.

In an attempt to produce comparable conditions, we conducted our experiments such that models trained with either method underwent the same number of weight update iterations. That is, for both methods we used the same mini batch size of 128. Note, that this choice implies that stability training is fed twice the amount of raw data, as it virtually generates a second batch of the same size containing perturbed images. However, as it uses one batch solely to learn the original classification task and the other to stabilize the predictions, we decided that it was more important to provide both, stability training and data augmentation, with the same number of weight updates, rather than the same number of samples.

4.2. Experiments

In this section, we will describe and motivate our experiments, present our results, and discuss the implications. First, we will explore how well stability training and data augmentation can utilize Gaussian noise as a universal perturbation approximator to improve a model's robustness against a diverse set of distortions in Section 4.2.1. Next, we explore stability training's capacity to replace data augmentation by evaluating how well models generalize when trained with various other training distortions in Section 4.2.2. In Sections 4.2.3 and 4.2.4, we evaluate our additions to and modifications of stability training from Sections 3.1.1 and 3.1.2, which aim to improve on the weaknesses we identified previously. Lastly, we will investigate stability training's ability to improve robustness on adversarial examples and compare performance with data augmentation and an adversarial training procedure.

4.2.1. Gaussian noise as a universal perturbation approximator

In this experiment, we evaluate the performance of stability training with Gaussian noise on a number of test cases. We tested different test distortions and intensities thereof in each test setting. We trained with a variety of hyper-parameter configurations, as shown in

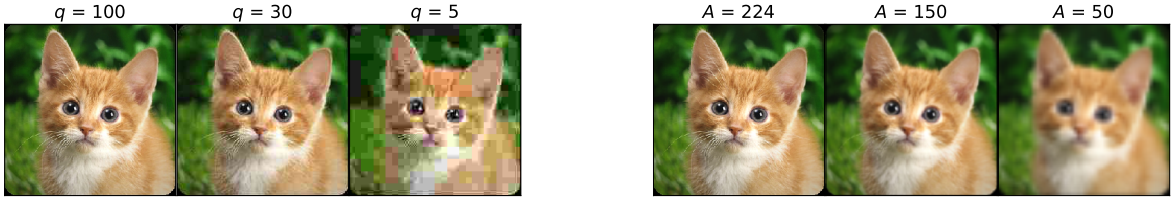
Table 4.4, and benchmark our results against data augmentation performance. To generate the perturbed images x', we added pixel-wise uncorrelated Gaussian noise to the input as described in Section 3.2.1. Following, we present our results, where we evaluate these models under different conditions. Note, that the models are the same for each experiment, and we only changed the test distortion for evaluation. To remind the reader, as described in Section 4.1.4, we only present the best and worst models for each test distortion and scenario to increase the clarity of the illustrations.

Hyper-parameter	Values
Regularization coefficient α	0.1, 1.0, 10.0
Gaussian noise standard deviation σ	0.01, 0.05, 0.1, 0.25

Table 4.4.: Hyper-parameter values for training with Gaussian noise.

Noise and artifact robustness

First, we looked at test performance on distortions from the noise and artifact category (see Table 3.1). This category encompasses distortions of the image that introduce a distractor signal to the image. Figure 4.1 shows examples of an image for each scenario from the test dataset.



(a) JPEG compression examples.

(b) Thumbnail resizing examples.

Figure 4.1.: Examples of an image from the test dataset in each of the three scenarios, from left to right: *no*, *practical*, and *extreme* distortion.

Looking at our results on JPEG compression in Figure 4.2, we can observe that stability training offers improved robustness, especially for high intensity distortions, across all three scenarios. Data augmentation, on the other hand, is unable to produce comparable results.

Remember that the *best* model shown for a given scenario, is that model which performed best at the distortion intensity associated with the respective scenario (indicated by the dotted vertical line in our plots). We observe that stability training consistently produces similar improvements across the whole spectrum of JPEG quality levels regardless of the chosen scenario.

Furthermore, we see that data augmentation under certain circumstances deteriorates the model's robustness compared to the baseline. The worst model produced by data augmentation is substantially worse across all distortion intensities, while stability training's worst model only has a marginally negative effect.

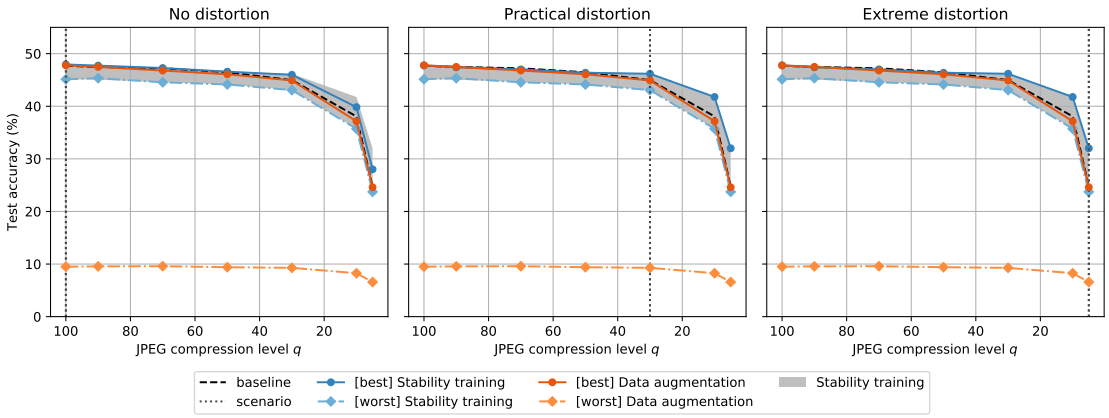


Figure 4.2.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion GAUSS-σ on increasingly distorted versions of the test dataset (x-axis) via JPEG-q. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

We conducted a similar experiment with test datasets perturbed via the Thumbnail resizing distortion, described in Section 3.2.5, to confirm our findings were not solely due to specifics of the JPEG compression algorithm. In Figure 4.3 we show our results, which paint a similar picture as our previous test.

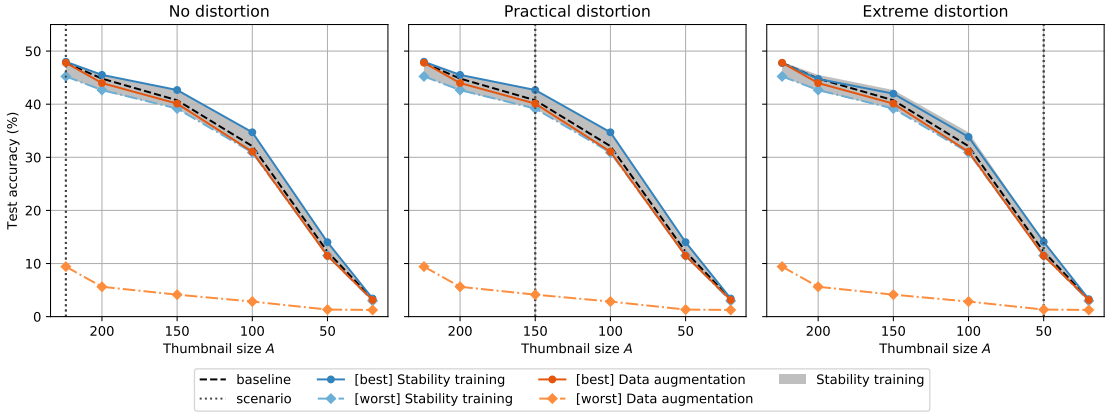


Figure 4.3.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion GAUSS-σ on increasingly distorted versions of the test dataset (x-axis) via THUMB-A. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

The best stability training configurations offer up to 5% improvement compared to baseline across all intensity levels, while the worst configuration we found produces only negligibly worse performance. Data augmentation, on the other hand, can provide no improvement at all at best and at worst deteriorates model performance to near random chance.

We observe again that stability training performs best in the range of distortion intensities around that intensity level chosen for our practical scenario. This is consistent across all scenarios. That is, stability training offers significant robustness improvements in the range

around `JPEG-30` and `THUMB-150`, regardless of the scenario under which we chose the model.

Transformation robustness

We now focus on the second category of input distortions: transformations. Distortions from this category produce a perturbed image x', which is qualitatively qualitatively similar to its reference x, as described in Sections 3.1.1 and 3.2. This experiment was conducted in order to further investigate possible negative side effects of training with Gaussian noise, in cases where the test distortions cannot be simulated by the training distortion.

We evaluate model robustness to rotation and random offset cropping applied to the test dataset as described in Section 3.2. We choose `ROT-30` and `CROP-3` as our distortions for the *practical* scenario and `ROT-150` and `CROP-15` for the *extreme* scenario, respectively. Our results are depicted in Figures 4.4 and 4.5.

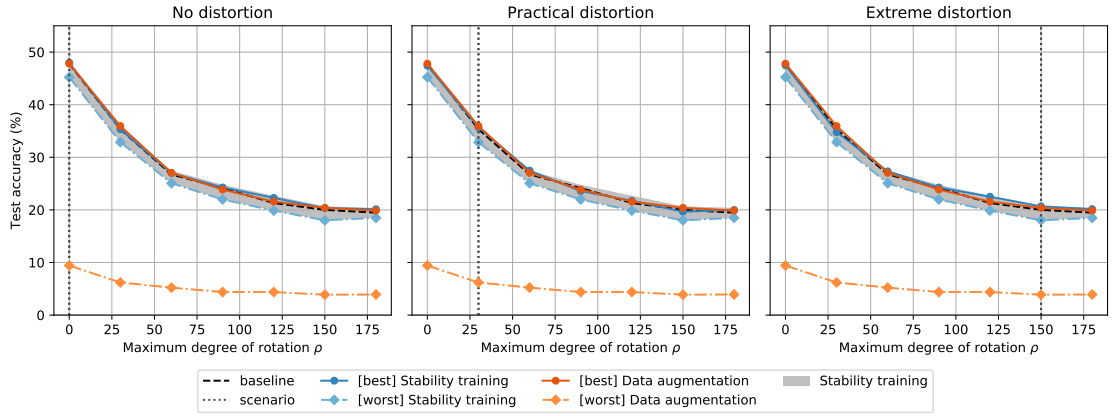


Figure 4.4.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion `GAUSS-`σ on increasingly distorted versions of the test dataset (x-axis) via `ROT-`ρ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

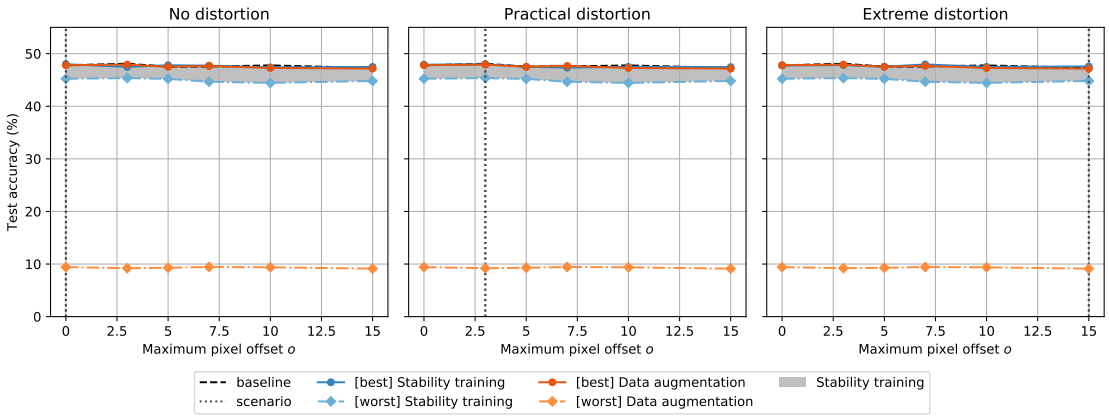


Figure 4.5.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion `GAUSS-`σ on increasingly distorted versions of the test dataset (x-axis) via `CROP-`o. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Data augmentation and stability training both cannot use their regularization with Gaussian noise to improve on their performance with respect to either rotation or offset cropping. While both methods can potentially produce models that perform on par with baseline, stability training's worst configuration is only marginally worse than baseline. Data augmentation, however, exhibits the same issue we witnessed before, where the worst model severely decreases the model's performance even on the original, un-augmented dataset.

Discussion

Our previous experiments revealed that stability training is well suited for improving a model's robustness towards noise or artifact inducing distortions by training with Gaussian noise. We were able to show that, while data augmentation is mostly unable to produce any significant improvements, stability training was able to offer substantial improvements to a model's robustness against practical and high intensity distortions.

Investigating robustness improvements towards transformations, we found that neither stability training nor data augmentation were able to generalize from Gaussian noise. As these distortions are qualitatively different from those in the noise and artifacts category, which Gaussian noise belongs to, this was to be expected. Importantly, however, we observed that data augmentation not only did not offer any improvements, but could severely deteriorate the model's performance even on the original task given an unfortunate choice of hyper-parameters.

It is important to take the results of the *worst* models into account. While one might want to suggest that these configurations could simply be eliminated during hyper-parameter search, we argue that this is unfeasible. Remember that the models used for evaluation in the previous experiments where those that performed best on a separate validation dataset as described in Section 4.1.3. As the goal of training with Gaussian noise is to generally increase robustness against various types of distortions, it is impossible to validate the model's robustness against all of them at that stage. It is therefore important to know the bounds of the potential impact the trained model will have on various types of distortions. Our results above show that stability training often offers improvements, but more importantly only brings almost

negligible risk to harm robustness. We present additional results investigating this behavior in greater detail in Section 4.2.2.

4.2.2. Distortion alternatives

As described in Section 2.2.1, data augmentation is typically used with data distortions that one would like the model to be invariant to. Here, images with instances of the relevant distortion are then added to the training dataset. This procedure makes no distinction between perturbed and original images. When the model tries to extract features of a class from the patterns it observes in the data, those patterns that are introduced by the artificial perturbations will simply be added to the concept that class (here, we use the term *concept of a class* to refer to the combination of features the model uses to identify that class). This works well for *transformative* types of perturbations (see Table 3.1), as, for example, a slightly rotated image is equally natural as the un-rotated image it originated from. However, perturbations like added Gaussian noise or compressions (the category we term *noise and artifacts*) induce a distractor signal to the image. Ideally, the model should learn to extract the characteristics of these distractor signals rather than add them to the concept of a class, as, for example, a Gaussian noised version of an image is qualitatively different from the original image.

Stability training provides a model with the reference and perturbed image at once. It optimizes the model such that its response to the perturbed image is more like its response to the reference image. We therefore hypothesize that stability training is better suited to improve robustness against distortions of the *noise and artifact* category.

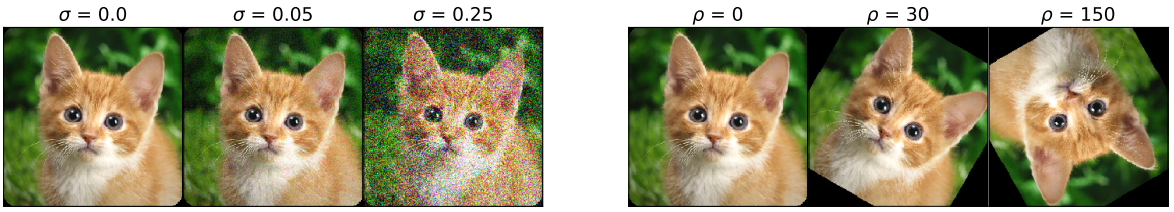
In our following experiments, we trained models with a variety of training distortions from both categories and evaluated their performance on various test distortions (see Table 4.2) in general. In our results, we distinguish between those experiments where training and test distortion are of the same type and those where they differ. This allows us to investigate how well a method can increase robustness against a given distortion type, on the one hand, and, on the other hand, what side effects on other distortions this might induce.

Using targeted perturbations

In a controlled environment, one might not care as much about having a single general perturbation approximator in the form of Gaussian noise, as used in Section 4.2.1, but instead is more interested in maximum robustness against a specific type of distortion. In this section, we discuss how well stability training performs compares to data augmentation, when the distortion type that will be seen during application of the model is known at training time. Therefore, we conduct experiments where we test model performance across different intensities of the same distortion that was used during training to regularize the model. Specifically, we present our results for Gaussian noise and rotation. The parameters used during training can be found in Table 4.5

Hyper-parameter	Values
Regularization coefficient α	0.1, 1.0, 10.0
Gaussian noise standard deviation σ	0.01, 0.05, 0.1, 0.25
Absolute maximum random rotation ρ	30, 90, 150

Table 4.5.: Hyper-parameter values for training with Gaussian noise and random rotation in the distortion alternatives experiment.



(a) Gaussian noise examples.

(b) Random rotation examples.

Figure 4.6.: Examples of an image from the test dataset in each of the three scenarios, from left to right: *no*, *practical*, and *extreme* distortion.

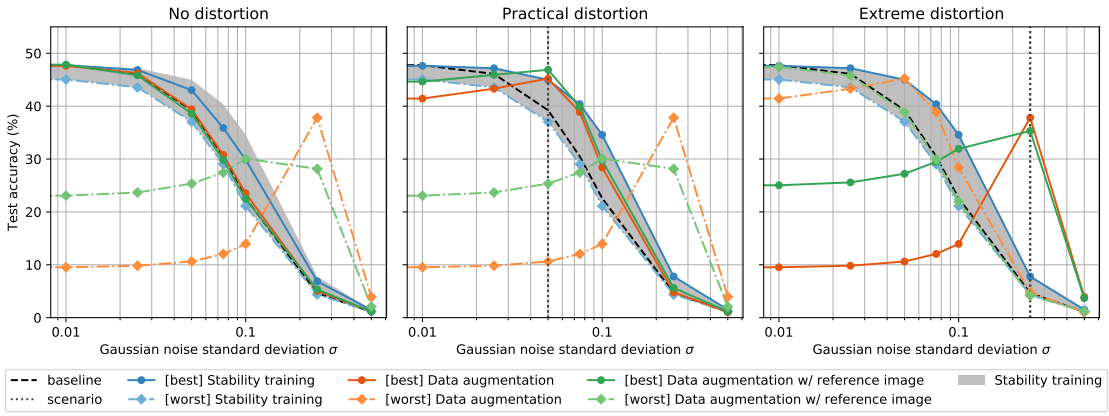


Figure 4.7.: Average test accuracy (y-axis) of stability training, data augmentation, and data augmentation with reference image, with training distortion `GAUSS-`σ on increasingly distorted versions of the test dataset (x-axis) via `GAUSS-`σ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

When testing performance on Gaussian noised test data, we see an interesting picture. We chose `GAUSS-0.05` as the practical and `GAUSS-0.25` for the extreme scenario. Examples of the distortions and the respective introduced noise are shown in Figure 4.6a.

For distortion intensities up to our practical scenario, stability training offers equal or better performance on the scenario intensity itself, but, importantly, also provides a significant increase in robustness against Gaussian noise across all levels of intensity. Data augmentation can potentially provide a competitive performance at the exact intensity level which we selected for via our scenarios, and in some cases can outperform stability training, as we

can witness in the extreme scenario. It sacrifices, however, general performance consistency. Data augmentation shows a tendency to suffer from catastrophic forgetting [French, 1999, McCloskey and Cohen, 1989] of the previously learned task. Looking at the parameter configurations behind the data augmentation runs reveals that these peaks we observe, correlate with the intensity level present during training. It seems that the new task overwrites the ability previously acquired by the baseline model.

To validate our hypothesis, we tested how test performance changed when we introduced the reference image back into the training corpus. Specifically, we defined a parameter p which represents the probability that a sample during training will be distorted, with 1 being the previously described data augmentation setting and 0 representing no data augmentation at all.

Figure 4.7 shows that introducing the reference image back into the training phase can help data augmentation reduce the effect of forgetting its original task. Given the right parameters we observe that data augmentation can achieve close to baseline performance and is sometimes able to improve on it. Importantly, however, this reduces the peak performance advantage data augmentation produced on test settings that were similar to the training setting. If one chooses to maximize generalization performance and minimize cases where the regularization via data augmentation negatively impacts performance, then stability training will still at least be comparable and often offer superior performance.

Figure 4.8 reveals the strengths of data augmentation. When training for and testing with a transformative distortion, data augmentation offers far superior performance and also does not seem to suffer from its previously encountered issue of forgetting its original task. While stability training offers some generalization improvement in this setting for all scenarios, it is often only at best comparable to the worst performance we achieved with data augmentation.

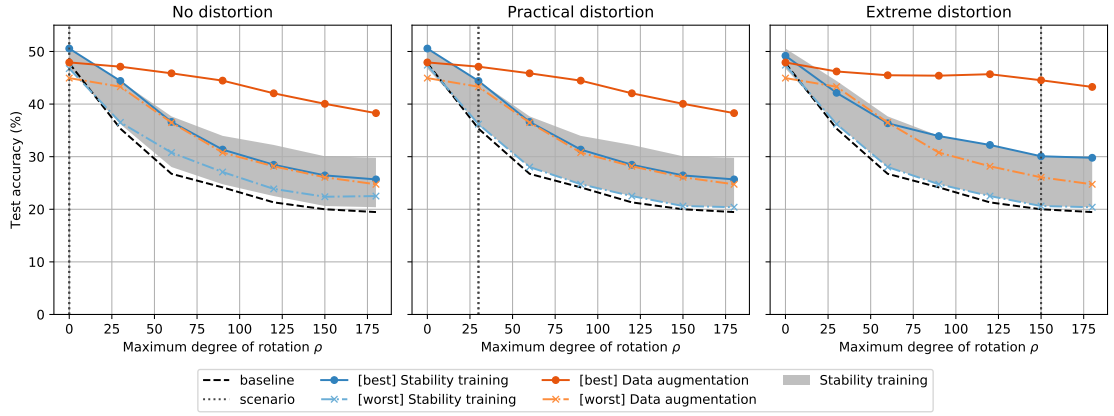


Figure 4.8.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion `ROT-`ρ on increasingly distorted versions of the test dataset (x-axis) via `ROT-`ρ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Stability training's loss objective introduces a directed loss optimization path, which forces a model to make its response to a noisy image more like its response to a clean reference image. Due to this property, we suspected stability training to perform worse with transformative training distortions of training distortions as the stability objective introduces a hurtful bias

towards one of the two images being more relevant than the other. In Section 3.1.1, we therefore proposed two alternative stability objectives to alleviate this issue, which we validate in Section 4.2.3.

Cross distortion effects

In Section 4.2.1, we observed that data augmentation can exhibit negative side effects on distortion types that were not used during training. Stability training, on the other hand, showed little risk of producing undesirable impacts.

In this experiment, we investigate if this issue is restricted to training with Gaussian noise or translates to other training distortions as well. We evaluated models which were trained with thumbnail resizing or rotation, and evaluated their performance on different distortions from either the same or the other category. Specifically, we present the side effects of both methods and both training distortions on encountering JPEG compressed images.

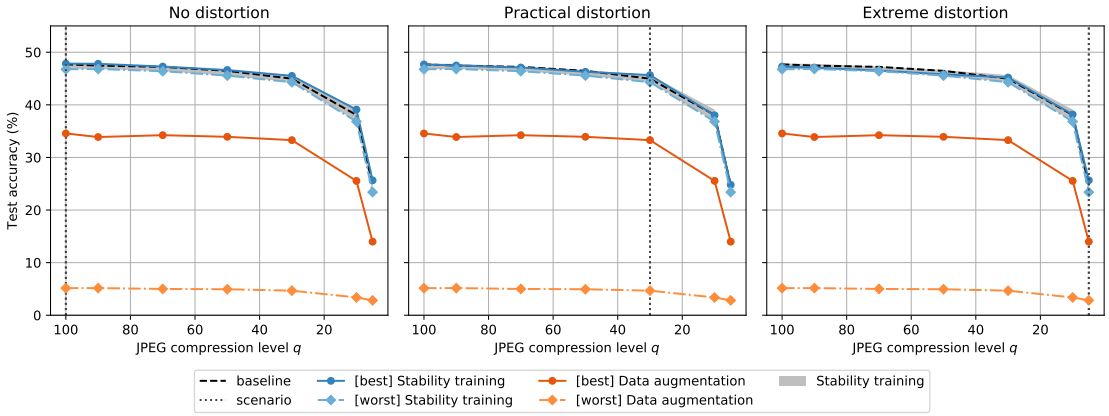


Figure 4.9.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion `THUMB-`A on increasingly distorted versions of the test dataset (x-axis) via `JPEG-`q. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

In Figure 4.9, we see that stability training does neither improve nor impair baseline performance. We also would not expect to see any improvements, as the training distortion (see Figure 3.6) does not bear any resemblance to the test distortion (see Figure 3.3). Data augmentation, however, greatly impairs the model in all scenarios. Even the best model offers substantially worse performance across all intensity levels, further worsening the issue we observed before.

Training with rotation to represent transformative distortions (see Figure 4.10) does reinforce this trend even more. While data augmentation impairs model performance, stability training is even able to generalize a robustness improvement from the distortions introduced via rotation to those introduced by the JPEG compression algorithm. The model trained with stability training also exhibits improved performance on the original task with no augmentation, indicating stability training's capacity to utilize a variety of distortions to increase a model's overall generalization performance.

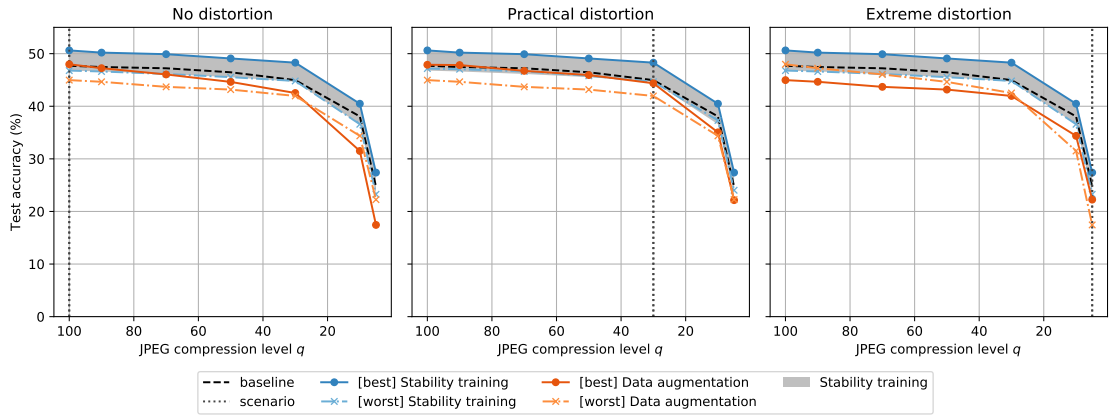


Figure 4.10.: Average test accuracy (y-axis) of stability training and data augmentation with training distortion `ROT-`ρ on increasingly distorted versions of the test dataset (x-axis) via `JPEG-`q. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Discussion

Our results using targeted perturbations show that stability training offers superior performance. Training and testing with Gaussian noise showed that, while data augmentation sometimes offers greater robustness improvements to specific test settings, stability training produces a model that is more robust across the spectrum of all test distortion intensities. As in our previous experiment, we observed an issue of catastrophic forgetting with data augmentation. This issue was only slightly mitigated by reintroducing the reference image into the training procedure.

Transformative distortions, however, is where data augmentation generally outperforms stability training. While stability training does increase robustness in these settings significantly, it offers fewer improvements compared to data augmentation by a substantial margin. Interestingly, however, we were able to observe that stability training can increase performance on the original task, indicating a form of general regularization.

The cross-distortion effects paint a clear picture. In most test settings, where we did not expect to see any generalization across different distortions, both methods, in fact, did not improve the model's robustness. However, data augmentation revealed to often produce substantial undesirable side effects, offering significantly worse performance on test distortions it was not designed to increase robustness against, compared to the baseline. Models trained with stability training, on the other hand, showed that, while offering increased robustness to the intended test distortion (see Section 4.2.2), did not harm performance on different test distortions at all. In fact, as we showed in Figure 4.10, stability training was sometimes even able to generate improvements on previously unseen distortions.

Overall we find that the choice of values of hyper-parameters clearly impacts the effectiveness of stability training, contrary to their effect in data augmentation, however, it seems to never impair the model's performance significantly. While not being a true lower bound, even with a unfavorable selection of parameters (i.e. one that does not improve generalization performance) the resulting model will perform comparably to the baseline.

Also, in Figures 4.7 and 4.8, we saw that for virtually every distortion and intensity level up to a practical scenario, stability training was also able to offer a superior generalization performance across the entire range of intensities.

4.2.3. Symmetrical stability objectives

We hypothesized that stability training's optimization objective (see Equation (3.2)) could introduce a hurtful bias into our training for these types of distortions. We could confirm this issue observing our results in Section 4.2.2, where we used stability training with rotated images during training. To alleviate this problem, we proposed two symmetrical stability objectives in Section 4.2.3, where $L_{fully_symmetrical}$ from Equation (3.9) symmetrizes the entire optimization objective, while L_{sym_stab} from Equation (3.8) only applies to the stability objective.

Hyper-parameter	Values
Symmetrical objective coefficient pairs α_1, α_2	(0.1, 0.1), (1.0, 1.0), (10.0, 10.0)

Table 4.6.: Hyper-parameter value pairs of the symmetrical stability objective and the fully symmetrical optimization objective.

In this experiment, we want to evaluate our proposed modifications. We used rotation as our training distortion and evaluated the robustness performance of models trained with our modified optimization objective. We trained the additional models with the selection of hyper-parameters shown in Table 4.6.

On the one hand, we present our results on rotation used as a test distortion in Figure 4.11 and compare our method directly against regular stability training and data augmentation. On the other hand, we also present evaluation results on Gaussian noise in Figure 4.12 to see if our methods introduce negative side effects on distortions from the noise and artifacts category.

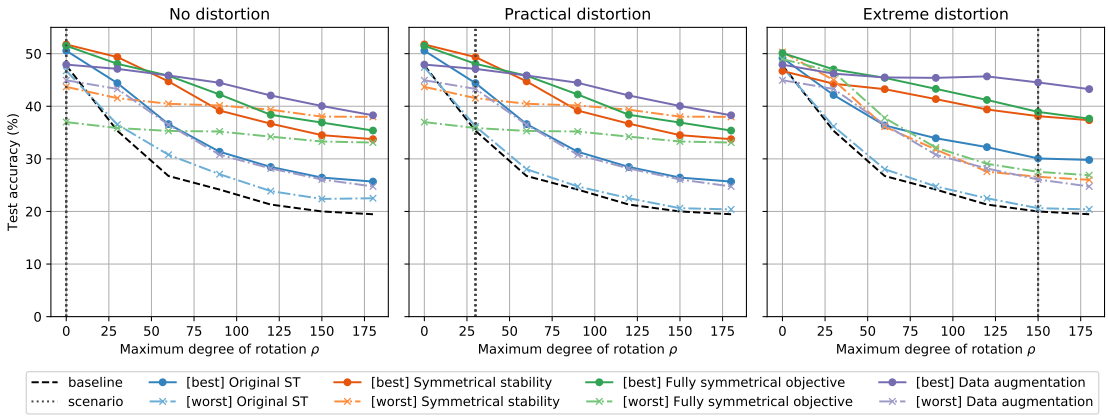


Figure 4.11.: Average test accuracy (y-axis) of regular stability training, stability training with modified optimization objective L_{sym_stab}, stability training with with modified optimization objective L_{sym_full}, and data augmentation, with training distortion `ROT-`ρ on increasingly distorted versions of the test dataset (x-axis) via `ROT-`ρ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

We can observe in Figure 4.11 that both our objectives do in fact remedy the shortcoming of the original stability objective when training with rotation. Across all intensity levels, both objectives can significantly increase the model's robustness towards rotation compared to classical stability training. For the *no* and *practical* distortion scenario, our methods outperform data augmentation by a measurable margin.

The symmetrical stability loss function does seem to provide equivalent performance improvements compared to the fully symmetrical objective, but with limited risk of deteriorating baseline performance. This, however, might be due to our choice of fixing $\alpha_1 = \alpha_2$ (see Table 4.6).

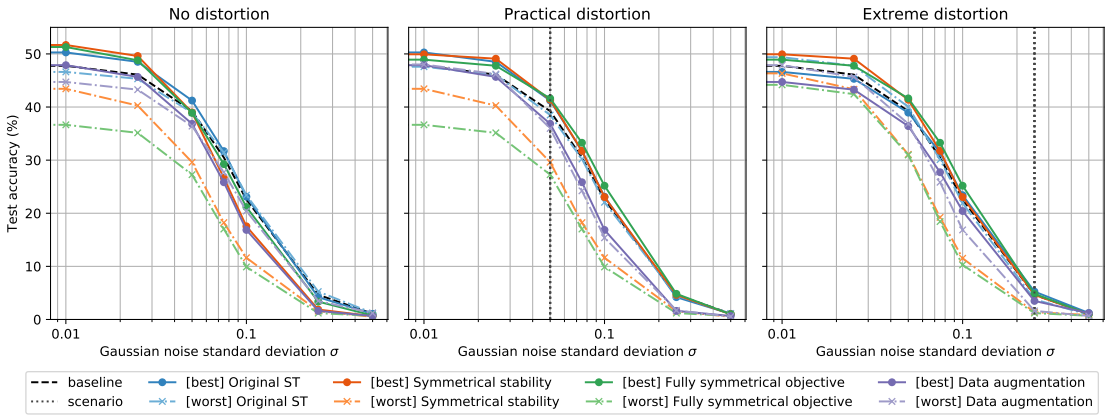


Figure 4.12.: Average test accuracy (y-axis) of regular stability training, stability training with modified optimization objective L_{sym_stab}, stability training with with modified optimization objective L_{sym_full}, and data augmentation, with training distortion `ROT-`ρ on increasingly distorted versions of the test dataset (x-axis) via `GAUSS-`σ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Our results from Figure 4.12 reveal that our modified objectives offer the same level of improvement as regular stability training does and therefore also outperforms data augmentation in this regard. Both method also bear lower risk to exhibit negative side effects under certain conditions. At low intensity levels, we observe that the worst model produced by the symmetrical stability loss is at least on par with the worst models created through the other methods. With increasing test distortion intensity, however, the worst model often offers significantly better performance than even the best model found by regular stability training and is comparable to the best data augmentation model. For the models chosen under the *no* and *practical* distortion scenarios the worst model produced by the fully symmetrical objective, however, shows significantly worse performance compare to baseline.

Discussion

We were able to show that our symmetrical loss functions from Section 3.1.1 mitigate the weaknesses of regular stability training that can be observed with transformative train distortions. As discussed in Section 3.1, the regular stability loss function introduces a bias towards the reference image x. Both, the symmetrical stability loss and the fully symmetrical loss, counteract this bias by treating the reference image x and the perturbed image x' similarly. Instead of forcing the network to adapt such that when given the perturbed image x' it produces outputs that are similar to the outputs it produces in response to the reference image x, our modified objectives aim to find a compromise.

Our results show that both methods offer significantly improved performances compared to regular stability training. For intensities up to our practical scenario, we found that even data augmentation was outperformed.

Concerning the risk of inducing potentially harmful side effects, we found that the symmetrical stability objective also bears the least amount of risk harming the performance, often even outperforming the best models of other methods in high intensity distortion robustness.

The symmetrical stability objective therefore offers the best performance across all scenarios. By restricting the symmetry to the stability objective, we allow the method to retain its original characteristic. Stability training usually splits learning the original task in the reference images and the stabilization objective. The fully symmetrical optimization objective eliminates this distinction, which we believe to be the reason for the higher risk of producing negative side effects.

4.2.4. Multiple optimization objectives

Data augmentation techniques typically apply multiple augmentations at once. We extended stability training's objective, such that multiple distortions can be optimized independently of each other in Section 3.1.2. This method yields more granular control over the impact of each distortion compared to naively applying each distortion in conjunction.

In our following experiments, we want to evaluate the suitability of both methods to simultaneously improve the robustness against multiple distortions. We will use the two train distortions Gaussian noise and rotation. The distortions will either be applied together on the same image to produce a single perturbed image x' which is used for data augmentation or *naive* stability training (naive indicates that we do not refer to our generalized extension of stability training mentioned above, but to the original procedure), or separately to produce two perturbed images x'_1 and x'_2 to be used with our modified stability objective from Equation (3.10).

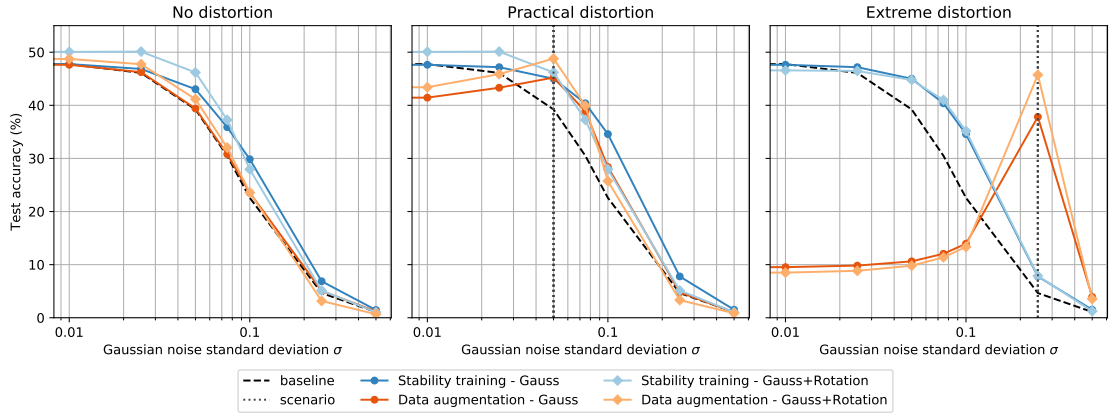


Figure 4.13.: Average test accuracy (y-axis) of regular stability training and data augmentation either exclusively with the training distortion `GAUSS-`σ, or combined with `ROT-`ρ, on increasingly distorted versions of the test dataset (x-axis) via `GAUSS-`σ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

First, we will focus on the effect of naively adding a second training distortion to both methods. We produce two test cases of Gaussian noise and rotation, where we train and evaluate on the same distortion, similar to what we described in Section 4.2.2. We compare these models with the results of adding the *other* distortion during training as well. That is, for example, we compare training and testing on Gaussian noise with training on a single perturbed image x' that was rotated and had Gaussian noise added to it.

In Figure 4.13 we can see that both, stability training and data augmentation, benefit in terms of robustness against Gaussian noise when naively adding rotation during training. In the no distortion and practical distortion scenarios of Figure 4.14, however, we can observe that the extra Gaussian noise added during training can significantly impair the robustness gained via data augmentation, eliminating the advantage the method had over stability training.

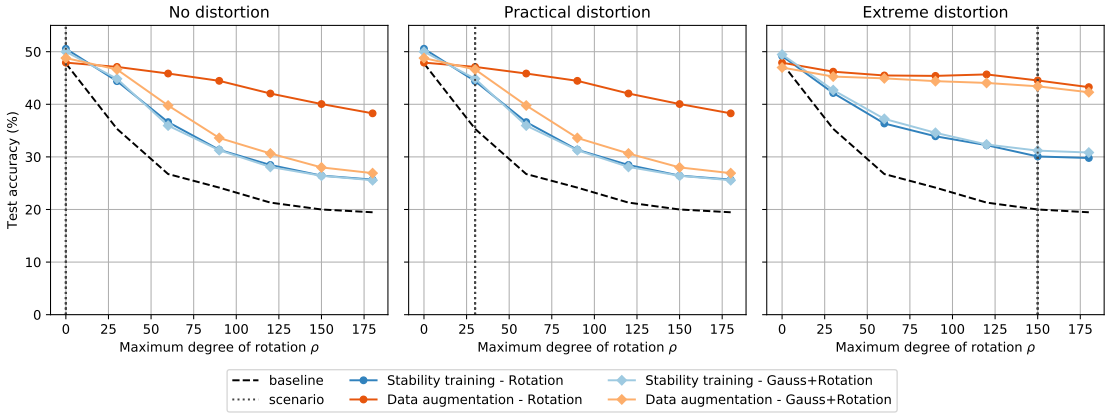


Figure 4.14.: Average test accuracy (y-axis) of regular stability training and data augmentation either exclusively with the training distortion `ROT-`ρ, or combined with `GAUSS-`σ, on increasingly distorted versions of the test dataset (x-axis) via `ROT-`ρ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Next, we compare these results with our extended stability objective, which splits training distortions onto separate input images. As data augmentation proved to effectively increase robustness against distortions of the transformative category and stability training showed its strength in generalizing from Gaussian noise, we also trained models with a combination of both methods, as described in Section 3.1.2.

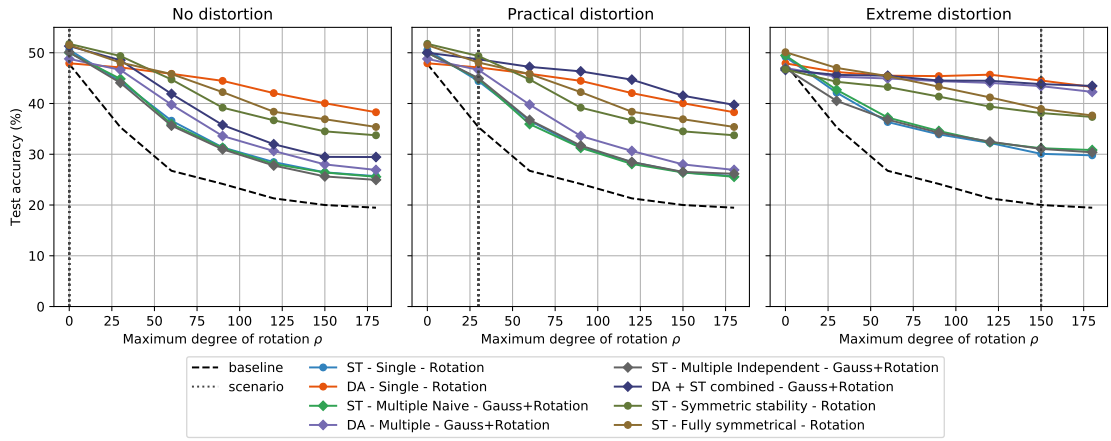


Figure 4.15.: Average test accuracy (y-axis) of regular stability training and data augmentation either exclusively with the training distortion `ROT-`ρ, or combined with `GAUSS-`σ, on increasingly distorted versions of the test dataset (x-axis) via `ROT-`ρ. Compared with stability training using a modified optimization objective L_{multi}, a combined optimization objective of regular stability training and data augmentation, and stability training with symmetrical objectives L_{sym_stab} and L_{sym_full}. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Our results evaluating on rotation in Figure 4.15 reveal no benefit of splitting training distortions in order to optimize them individually. Looking at our results on Gaussian noise in Figure 4.16 as well, we observe that the combination of data augmentation and regular stability training seems to offer the best of both methods. The models resulting from this procedure offer similar robustness to rotation as regular data augmentation, while also exhibiting the same level of robustness towards Gaussian noise as regular stability training offers. Lastly, we see in Figure 4.15 that our previously tested symmetrical stability loss produces models that offer still superior robustness against low to practical degrees of rotation compared to that of the combined method.

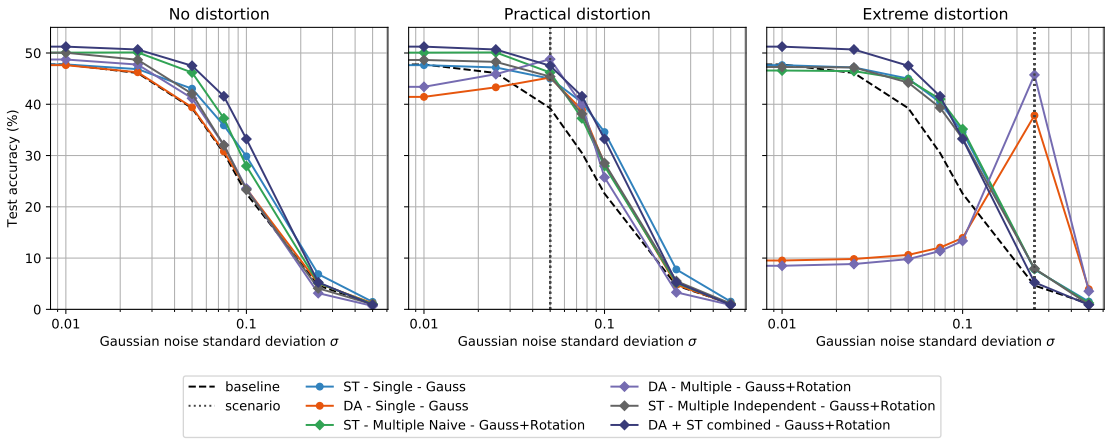


Figure 4.16.: Average test accuracy (y-axis) of regular stability training and data augmentation either exclusively with the training distortion `ROT-`ρ, or combined with `GAUSS-`σ, on increasingly distorted versions of the test dataset (x-axis) via `GAUSS-`σ. Compared with stability training using a modified optimization objective L_{multi} and a combined optimization objective of regular stability training and data augmentation. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

Discussion

When using data augmentation, one often applies several distortions to the image simultaneously. We showed that this procedure can have negative side effects, similar to those discovered in our previous experiment in Section 4.2.2. Stability training, on the other hand, does not exhibit this behavior.

We were unable to show that splitting training distortions onto separate input image offers any advantage when using stability training. However, it also did not harm performance compared to naively applying multiple distortions to a single perturbed image x'. Further research has to be conducted to determine if this feature can be leveraged to improve robustness across multiple distortions more effectively.

Combining data augmentation and stability training could be shown to be a sensible strategy to easily combine the strengths of both methods. At the same time, however, we showed that the strengths that data augmentation offers with respect to distortion types of the transformative category, are surpassed by the benefits achieved by our symmetrical stability objective in the range of intensities around the practical scenario. Further research also needs to be conducted to confirm if the combination of stability training and data augmentation bears the same level of risk of hurting the baseline performance, as we commonly observed with regular data augmentation in previous experiments.

4.2.5. Adversarial robustness

A recently upcoming topic of discussion regarding prediction stability and robustness of deep neural networks are adversarial examples. Adversarial examples show case particular weaknesses of deep neural networks. Carefully chosen changes to the input can confuse a network

and greatly impair its ability to correctly classify the data.

A naive brute force approach to improve robustness against adversarial examples is to generate them during training and integrate them into the training corpus. This can be seen as a form of data augmentation. As it has proven to mitigate the issue of catastrophic forgetting (see Figure 4.7), we randomly reintroduce the reference image for the data augmentation experiments, with a chance of $p = 0.75$ to see the adversarial example. As our previous experiments indicate that stability training offers robustness advantages on a broader spectrum of perturbations, we also wanted to analyze its effect on adversarial examples, specifically. We compare both methods with a standard technique to improve robustness against adversarial examples, called adversarial training (see Section 2.2.2). As we see from Equation (2.7), this method differs from data augmentation in that it simultaneously evaluates the reference image and an adversarial example based on the reference at the same time. It also differs from stability training, because it evaluates the original loss L_0 on both images.

Hyper-parameter	Values
Regularization coefficient α	0.1, 1.0, 10.0
Adversarial training ratio μ	0.5, 0.75
FGSM coeffecient ϵ	0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1
JPEG quality level q	90, 50, 10
Gaussian noise std. σ	0.01, 0.05, 0.1, 0.25

Table 4.7.: Hyper-parameter values for training with Gaussian noise, JPEG compression and adversarial examples in the adversarial robustness experiments.

In this experiment, we trained our models with dynamically generated adversarial examples via the fast gradient sign method (see Section 3.2.6). We conduct the experiments treating adversarial examples like any other previous perturbed input. We selected `FGSM-0.001` and `FGSM-0.1` as the practical and extreme scenario, which we depict in Figure 4.17

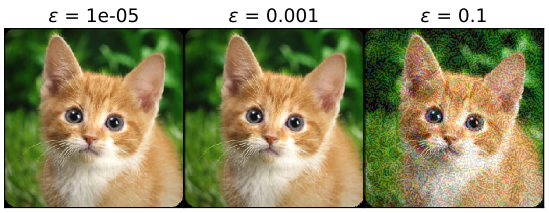


Figure 4.17.: Examples of a distorted image from the test dataset in each of the three scenarios, from left to right: *no*, *practical*, and *extreme* distortion.

Explicit adversarial robustness

We first want to focus on the effectiveness of all three methods in explicitly increasing a model's robustness against adversarial examples. In Figure 4.18 we plot our comparative results.

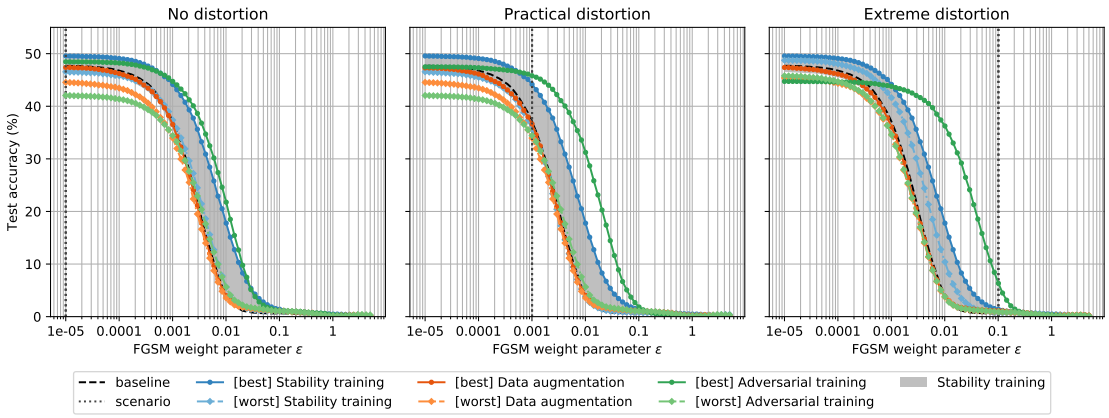


Figure 4.18.: Average test accuracy (y-axis) of regular stability training, data augmentation with the training distortion `FGSM-`ϵ on increasingly distorted versions of the test dataset (x-axis) via `FGSM-`ϵ, compared with adversarial training. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

We can clearly observe that data augmentation does not improve robustness in any case, rather it seems to hurt baseline performance. Stability training, on the other hand, offers significant improvements consistently. Models that have been trained with stability training and were chosen for their performance in a given scenario show general improvements across the entire range of tested intensities.

Looking at the no distortion scenario, we see that stability training and adversarial training can offer similar performance improvements. The practical and extreme scenarios show that when choosing the model based on its performance in these scenarios, adversarial training often outperforms stability training by a significant margin for large test distortion intensities. However, at the same time, stability training offers better performance on low intensities even if the model was chosen for its performance on the extreme distortion. We can also observe that even the worst stability training settings do not have any significant negative impact on the baseline performance. Adversarial training, however, can harm baseline performance with an unfavorable parameter selection. As we see in the extreme scenario, even the best model deteriorates performance on lower intensity levels.

Implicit adversarial robustness via stability training

As stability training proved itself to be able to generalize robustness well from Gaussian noise regularization, we compared how well this method increases robustness against FGSM examples. Additionally, as we saw in Section 4.2.2, stability training was sometimes able to generalize robustness across distortion types. We therefore also evaluate the side effect of training with JPEG compression can have on adversarial robustness. With this experiment, we intend to compare the degree to which adversarial robustness is improved by stability training and data augmentation without specifically setting out to do so. We present our results in Figure 4.19.

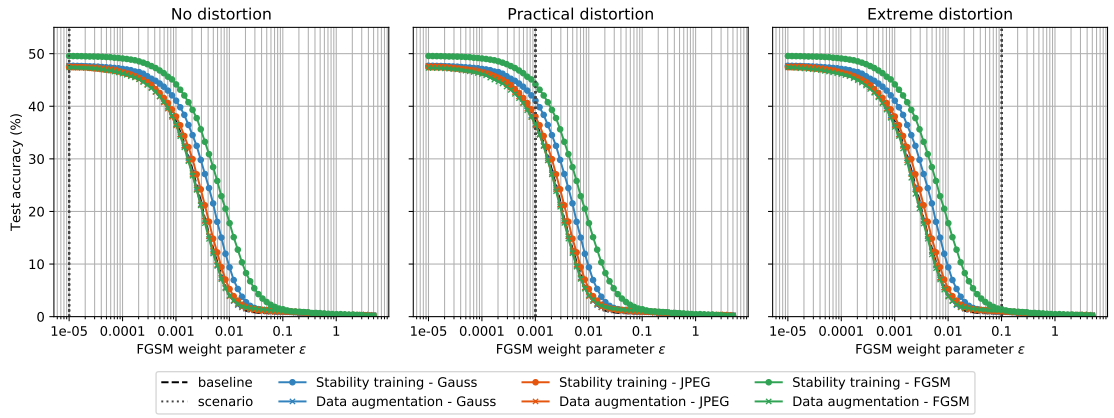


Figure 4.19.: Average test accuracy (y-axis) of regular stability training, data augmentation with the training distortion of either `FGSM-`ϵ, `GAUSS-`σ, `JPEG-`q on increasingly distorted versions of the test dataset (x-axis) via `FGSM-`ϵ. Subplots show the best and worst models for both methods, chosen based on their performance in the respective scenario (dotted vertical line).

While most settings do not offer any improved robustness, stability training with dynamically generated FGSM examples produces the overall best performance in every case. It even allows the model to improve upon its performance on the original task which we see as an indication for a general regularization. It is worth noting, however, that stability training with JPEG compression offers tiny improvements on baseline robustness as well. Distorting an image with randomly generated Gaussian noise is orders of magnitude less computationally demanding than computing the data gradient for each step during training as required by FGSM. Additionally, the load required to compute this gradient increases with the complexity of the model. Data augmentation is not able to improve robustness significantly in any test setting. The positive side effect of increased adversarial robustness from training with stability training and Gaussian noise is therefore a noteworthy feat.

Discussion

We were able to establish stability training as a valid method to increase a model's robustness towards adversarial examples generated via the fast gradient sign method. We showed that while simple data augmentation does not offer any improvements, but instead can make a model more sensitive to these examples, stability training offers robustness improvements, which are arguably even more desirable than those of standard adversarial training. Stability training was again able to offer improved robustness without the risk of deteriorating the baseline model performance at any intensity level, which is unmatched by both alternatives. Additionally, while the improvements are not substantial, it is noteworthy to see that regular stability training with Gaussian noise as the training distortions could be shown to measurably improve robustness of the model against FGSM examples. This is an indication for universally improved robustness.

Lastly, it is important to note that these results do not pose as a proper evaluating of adversarial robustness. Carlini et al. [2019] point out that solely using fast gradient sign based evaluation procedures does not suffice for a proper analysis of adversarial robustness of

a given model. Therefore further research needs to be conducted to confirm these preliminary findings.

5. Conclusion

The goal of this thesis was to explore the capabilities of stability training, understand its strengths and weaknesses, and propose and evaluate potential modification. We saw potential in stability training as an alternative to data augmentation in a wider variety of cases than originally covered by its authors and therefore continued to compare both methods with each other.

We started with Gaussian noise as a universal perturbation approximator. Due to the central limit theorem, Gaussian noise has the capacity to potentially simulate a variety of different noise and distortion types. It therefore lends itself to potentially enable a model to generalize off of it against a multitude of different distortions, thus minimizing extra computational cost. In Section 4.2.1 we showed that stability training was able to leverage this property effectively and use it to increase a model's robustness against a variety of distortions, while data augmentation at best failed to improve upon baseline performance and at worst drastically diminished it. Furthermore, we showed that stability training impacts model performance and robustness in a way that is much more general. We showed, that while data augmentation can be used to increase robustness against a very specific distortion configuration, when dealing with distortions that add a noise signal or introduce visual artifacts, it often even fails to generalize over a spectrum of intensities of the same distortion.

In Section 4.2.2 we showed that standard data augmentation can suffer severely from the phenomenon of catastrophic forgetting, where the regularization procedure erases part of the knowledge acquired by the network about the original task. We showed that introducing the reference image back into the regularization phase does help combat this issue, but at the cost of peak performance. The result would be a model that performs comparably well on a distortion type it was specifically regularized for, but does not have the same cross distortion benefits that stability training offers.

Furthermore, we showed that stability training's performance is suboptimal when facing distortions that do not allow to identify the perturbed input as a combination of the reference input and some noise signal, but rather apply a transformation onto the input, as for example rotation does. We discussed how this might be due to the directed, asymmetrical nature of the stability objective that is inappropriate for these types of augmentations. In Section 4.2.3 we then proposed our own modification of the stability objective and showed empirically that this eliminates this shortcoming, and in fact, enables stability training to outperform data augmentation in this regard.

In Section 4.2.5 we presented that stability training can be used to much greater effect compared to data augmentation in increasing robustness to adversarial examples that have been generated via the fast gradient sign method. When comparing it with the robustness gained from standard adversarial training, we found that stability training offers arguably more desirable effects as it provided more consistent improvements. We could even show, that regular stability training with Gaussian noise could yield improvements here, however, dynamically generating adversarial training examples does still offer the highest improvements, while at the cost of a significant higher demand of computational power.

Lastly, we compared the ability of both methods to utilize multiple distortions at once. In Section 4.2.4 we underlined the effect we saw earlier with data augmentation. That is, data augmentation can potentially severely harm performance if an inappropriate augmentation is chosen during training. This can easily happen in real world application, where un-curated data might contain perturbations that were not specifically accounted for during training of the model. Our modification of the stability objective to accommodate multiple distortions independently could not be shown to improve naive stacking of several distortions meaningfully. Combining stability training with data augmentation, such that both tackle those distortions they are best at, had us hoping of combining the best of both worlds into one procedure. In the end, we could show, however, that for practical levels of distortion intensity, regular stability training or, if appropriate, our symmetrical derivative does provide better absolute and generalized performance.

In conclusion, we feel comfortable to recommend stability training as an alternative to data augmentation in virtually every case. Stability training can offer at least competitive and often superior performance both in regard to specific types of distortion and across different unseen types, while also suffering from significantly less severe and often negligible undesirable side effects. Two caveats, however, are that stability training does increase the effective batch size for each mini-batch as for each step there needs to be the reference and augmented image present. And secondly, that stability training requires an extra parameter to be tuned. We found, however, that this also yields greater control over the regularization and preliminary testing often pointed us towards appropriate values of the coefficient that generally worked well in practice.

A. Tiny ImageNet Fullsized

Following, we provide a python3 script to generate the TIFS dataset used throught this thesis. It requires the ILSVRC-2012 dataset to be present.

build_tifs_from_ilsvrc2012.py

```
from pathlib import Path
import argparse
import os
import random
import shutil

parser = argparse.ArgumentParser()
parser.add_argument('--loc', type=lambda loc: Path(loc), required=True,
                    help='Location of ILSVRC2012 dataset')
parser.add_argument('--out', type=lambda loc: Path(loc), required=True,
                    help='Location of newly generated TIFS dataset')
parser.add_argument('--seed', type=int, help='Seed for subset generation')

TINY_CLASSES = ['n03355925', 'n03992509', 'n02423022', 'n06596364',
                'n07920052', 'n02437312', 'n01910747', 'n07614500',
                'n09332890', 'n01882714', 'n02364673', 'n02190166',
                'n12267677', 'n02917067', 'n03599486', 'n04099969',
                'n07747607', 'n02410509', 'n02279972', 'n02094433',
                'n02948072', 'n02124075', 'n02509815', 'n02123045',
                'n01917289', 'n03649909', 'n07753592', 'n02699494',
                'n02815834', 'n02906734', 'n04596742', 'n02233338',
                'n04067472', 'n03447447', 'n04507155', 'n02666196',
                'n07720875', 'n03393912', 'n02808440', 'n01742172',
                'n04328186', 'n02927161', 'n02843684', 'n03837869',
                'n03042490', 'n01644900', 'n04597913', 'n02395406',
                'n02963159', 'n02226429', 'n03838899', 'n02132136',
                'n02795169', 'n02481823', 'n03388043', 'n04285008',
                'n03980874', 'n04532670', 'n02977058', 'n02480495',
                'n02099712', 'n03854065', 'n02002724', 'n04265275',
                'n01784675', 'n03733131', 'n03255030', 'n02403003',
                'n02165456', 'n09193705', 'n02415577', 'n04399382',
                'n04417672', 'n07749582', 'n03977966', 'n04501370',
                'n09246464', 'n04023962', 'n03250847', 'n02999410',
                'n02321529', 'n03983396', 'n07875152', 'n02769748',
                'n01774750', 'n02074367', 'n01629819', 'n04008634',
                'n04398044', 'n07871810', 'n07768694', 'n03444034',
                'n02085620', 'n02788148', 'n02125311', 'n01641577',
                'n07873807', 'n03976657', 'n02814860', 'n04366367',
```

```
                'n02504458', 'n02988304', 'n02814533', 'n01770393',
                'n03902125', 'n02802426', 'n04456115', 'n07579787',
                'n03089624', 'n04532106', 'n02950826', 'n03930313',
                'n04254777', 'n02056570', 'n01768244', 'n03662601',
                'n02486410', 'n04070727', 'n02268443', 'n03100240',
                'n01774384', 'n04275548', 'n01984695', 'n04149813',
                'n01443537', 'n03404251', 'n03424325', 'n04560804',
                'n03584254', 'n03796401', 'n04487081', 'n04486054',
                'n02206856', 'n02129165', 'n07734744', 'n07711569',
                'n02236044', 'n02730930', 'n04118538', 'n07715103',
                'n03763968', 'n03670208', 'n01855672', 'n02113799',
                'n03814639', 'n04251144', 'n04133789', 'n07615774',
                'n02837789', 'n09256479', 'n04311004', 'n04179913',
                'n03804744', 'n01944390', 'n03617480', 'n03770439',
                'n02123394', 'n01698640', 'n02841315', 'n01983481',
                'n03937543', 'n03891332', 'n03201208', 'n02106662',
                'n04259630', 'n02099601', 'n03085013', 'n02892201',
                'n02231487', 'n03400231', 'n03544143', 'n04465501',
                'n01950731', 'n07695742', 'n03160309', 'n03014705',
                'n03179701', 'n02058221', 'n04074963', 'n04371430',
                'n02909870', 'n04562935', 'n02281406', 'n04356056',
                'n03126707', 'n03970156', 'n02793495', 'n09428293',
                'n02669723', 'n03706229', 'n02823428', 'n03026506',
                'n04146614', 'n03637318', 'n02883205', 'n04376876',
                'n07583066', 'n01945685', 'n04540053', 'n02791270']

if __name__ == '__main__':
    args = parser.parse_args()
    if args.seed is not None:
        random.seed(args.seed)
    args.loc = Path(args.loc)
    for phase in ('train', 'val'):
        for i, cls in enumerate(TINY_CLASSES):
            path = args.loc / phase / cls
            chosen_samples = random.sample(os.listdir(args.loc / phase / cls),
                                           500 if phase == 'train' else 50)
            out_path = args.out / phase /cls
            os.makedirs(out_path, exist_ok=True)
            for f in chosen_samples:
                shutil.copy(path / f, out_path)
            print(f'{phase}: ({i+1}/{len(TINY_CLASSES)}) classes generated',
                  end='\n' if i-1 == len(TINY_CLASSES) else '\r')
```

Listing 1: Code to generate the TIFS dataset. Requires ILSVRC2012 dataset to be present.

Bibliography

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 978-0-387-31073-2.

J. S. Bridle. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In F. F. Soulié and J. Hérault, editors, *Neurocomputing*, NATO ASI Series, pages 227–236. Springer Berlin Heidelberg, 1990. ISBN 978-3-642-76153-9.

N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On Evaluating Adversarial Robustness. Feb. 2019. URL `https://arxiv.org/abs/1902.06705v2`.

R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, Apr. 1999. ISSN 1364-6613. doi: 10.1016/S1364-6613(99)01294-2. URL `http://www.sciencedirect.com/science/article/pii/S1364661399012942`.

J. W. Gibbs. *Elementary principles in statistical mechanics*. 1902.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

I. Goodfellow, Y. Bengio, and A. Courville. Regularization for Deep Learning. In *Deep Learning*, pages 221–266. MIT Press, 2016. URL `http://www.deeplearningbook.org`.

I. Goodfellow, N. Papernot, S. Huang, R. Duan, P. Abbeel, and J. Clark. Attacking Machine Learning with Adversarial Examples, Feb. 2017. URL `https://openai.com/blog/adversarial-example-research/`.

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. pages 1026–1034, 2015. URL `https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. pages 770–778, 2016. URL `https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html`.

D. O. Hebb. The Organization of Behavior: A Neuropsychological Theory. In *The Organization of Behavior: A Neuropsychological Theory*, pages 62–87. Wiley, Jan. 1949. ISBN 978-0-471-36727-7. Google-Books-ID: dZ0eDiLTwuEC.

A. Hernández-García and P. König. Data augmentation instead of explicit regularization. *arXiv:1806.03852 [cs]*, June 2018. URL `http://arxiv.org/abs/1806.03852`. arXiv: 1806.03852.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs]*, July 2012. URL `http://arxiv.org/abs/1207.0580`. arXiv: 1207.0580.

K. Hornik, M. B. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, June 2015. URL `http://proceedings.mlr.press/v37/ioffe15.html`.

A. Krogh and J. A. Hertz. A Simple Weight Decay Can Improve Generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pages 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 9781558602229. URL `http://dl.acm.org/citation.cfm?id=2986916.2987033`. event-place: Denver, Colorado.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, Mar. 1951. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177729694. URL `https://projecteuclid.org/euclid.aoms/1177729694`.

M. McCloskey and N. J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In G. H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, Jan. 1989. doi: 10.1016/S0079-7421(08)60536-8. URL `http://www.sciencedirect.com/science/article/pii/S0079742108605368`.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. ISBN 0-262-30432-5.

Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 0003-4851. URL `https://www.jstor.org/stable/2236626`.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471(Electronic),0033-295X(Print). doi: 10.1037/h0042519.

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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27 (3):379–423, 1948.

J. Sietsma and R. J. F. Dow. Creating Artificial Neural Networks That Generalize. *Neural Netw.*, 4(1):67–79, Jan. 1991. ISSN 0893-6080. doi: 10.1016/0893-6080(91)90033-2. URL `http://dx.doi.org/10.1016/0893-6080(91)90033-2`.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL `https://www.nature.com/articles/nature16961`.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Sept. 2014. URL `http://arxiv.org/abs/1409.1556`. arXiv: 1409.1556.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, Jan. 2014. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2627435.2670313`.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, Dec. 2013. URL `http://arxiv.org/abs/1312.6199`. arXiv: 1312.6199.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper With Convolutions. pages 1–9, 2015. URL `https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html`.

Y. Tang and C. Eliasmith. Deep Networks for Robust Visual Recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1055–1062, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL `http://dl.acm.org/citation.cfm?id=3104322.3104456`. event-place: Haifa, Israel.

L. Taylor and G. Nitschke. Improving Deep Learning using Generic Data Augmentation. Aug. 2017. URL `https://arxiv.org/abs/1708.06020v1`.

A. Veit, M. Wilber, and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *arXiv:1605.06431 [cs]*, May 2016. URL `http://arxiv.org/abs/1605.06431`. arXiv: 1605.06431.

O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv:1708.04782 [cs]*, Aug. 2017. URL `http://arxiv.org/abs/1708.04782`. arXiv: 1708.04782.

L. S. Yaeger, R. F. Lyon, and B. J. Webb. Effective Training of a Neural Network Character Classifier for Word Recognition. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 807–816. MIT Press, 1997. URL `http://papers.nips.cc/paper/1250-effective-training-of-a-neural-network-character-classifier-for-word-recognition.pdf`.

G. Zhang, C. Wang, B. Xu, and R. Grosse. Three Mechanisms of Weight Decay Regularization. Oct. 2018. URL https://arxiv.org/abs/1810.12281v1.

S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the Robustness of Deep Neural Networks via Stability Training. pages 4480–4488, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Zheng_Improving_the_Robustness_CVPR_2016_paper.html.