Mathematisch-
Naturwissenschaftliche
Fakultät

BethgeLab
Computational Neuroscience
& Machine Learning

Master's thesis

Metalearning Robustness

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Benjamin Mitzkus, benjamin.mitzkus@student.uni-tuebingen.de

Thesis period: 02/2020 - 07/2020

Supervisor: Robert Geirhos
First reviewer: Prof. Dr. Matthias Bethge, Universität Tübingen
Second reviewer: Prof. Dr. Felix Wichmann, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Benjamin Mitzkus (Matrikelnummer 3570443), 31. Juli 2020

Abstract

Recent work shows that standard trained convolutional neural networks are not robust to corrupted training data, dropping considerable amounts of performance as the input signal gets noisy. As real world data contains different sources of noise (sensor noise, weather conditions, lighting conditions, occlusions), this poses a huge problem in practice. A method to tackle this shortcoming is to apply style transfer to the training data. However, so far the exact influence of the parameterization of style transfer on the robustness of the resulting model has not been assessed. In this work, different parameterizations of the magnitude the style transfer is applied with were systematically tested and evaluated w.r.t. the robustness of the resulting model. Results suggested that most robustness is gained by applying style transfer on maximum magnitude to the training data. Additionally, the importance of the composition of the dataset the styles are drawn from was evaluated. The results showed that using a small subset of only 64 different, randomly chosen styles was already enough to observe the desired generalization to corrupted testing data. Instead of selecting uniformly from the style dataset, a selection probability distribution was learned. There exists a clear qualitative difference between styles with high selection probability and styles with low selection probability, indicating that a distinction between beneficial and detrimental styles w.r.t. the robustness of a model trained on data stylized with these can be drawn. Applying style transfer using the styles with maximum likelihood according to the learned distribution resulted in an increase of 1.4% of mean performance under corruption compared to applying style transfer with the same number of randomly selected styles.

Acknowledgements

I would like to thank the following people, who helped me with this research project:

Robert Geirhos, who was the supervisor for this project, for valuable discussions, guidance and his shared expertise. Data and figures in section 4.3 were collected by him and published here with his permission. Roland Zimmermann provided the implementation of the hyperparameter optimization method used in my experiments. Wieland Brendel for discussions and feedback on the experimental design of this thesis. My reviewers Matthias Bethge and Felix Wichmann for the opportunity to work on this project as well as the development of the research idea.

I want to express my gratitude to my parents for their continuous support throughout my studies, and for emotional and motivational support.

Contents

1 Introduction

In recent years, convolutional neural networks (CNNs) have become overwhelmingly popular on image recognition tasks including image classification, object detection and semantic segmentation, often surpassing human-level performance on a single task, which led to wide-spread usage of CNNs in real-world applications. Real-world data usually contains different sources of noise, ranging from sensor noise over occlusions to distortions in the visual signal, such as weather/lighting conditions. It has been shown that the performance of fully trained CNNs drops rapidly when tested on images that are modified in a way that was not part of the training regime. Geirhos et al. [2018b] showed that image classification performance drops to chance level on a variety of image distortions. Michaelis et al. [2019] evaluated the performance of object detection models on corrupted testing data, showing that performance drops on average by 50%, irrespective of the model architecture or dataset that was used for training. Moreover, training CNNs on a subset of distortions doesn't result in an increase in performance on other distortions [Geirhos et al., 2018b]. In order to make this generalization failure measurable and comparable accross training setups, Hendrycks and Dietterich [2019] established the ImageNet-C benchmark consisting of a set of common image corruptions that are applied to the test set and allow to measure the performance of models as soon as the input gets noisy. A preview of some of these image corruptions is given in figure 1.1.

The susceptibility of CNNs to noise in the input poses a problem to safety critical real-world applications, as it is unforeseeable which kinds of input distortions might occur over the lifespan of an image processing system. Imagine for instance a smart restock robot in a supermarket, that was trained for several days and is now able to detect and refill empty shelves while avoiding any collisions with customers or inventory. One day, the supermarket owner decides to switch his lightbulbs to economy-friendly LEDs. For the human visual system, the difference is hardly noticeable as we have learned to adapt to the changing spectrum of daylight. The robot however starts bumping into obstacles and customers and fails to detect missing items. The slight change in lighting condition might not make a perceptible difference to humans, but the pixel values that the optical sensor of the robot measures have changed drastically, making retraining necessary in order to avoid dangerous accidents. This scenario is accurately represented by some of the image corruptions (brightness, contrast) that are part of the common corruption benchmark.

The failure of standard trained CNNs to generalize to corrupted image data stems

partly from the fact that networks classify images based on local texture of objects rather than object shape, while human decisions are strongly biased towards object shapes [Geirhos et al., 2018a]. As most image corruptions used to evaluate the robustness of CNNs as well as most naturally occuring corruptions of visual signals alter, occlude or destroy local texture, but leave object shapes mostly intact, humans are able to recognise objects under strong corruptions while the performance of CNNs drops to chance level (compare Geirhos et al. [2018b,a]).

Intuitively, this seems like a problem of just collecting a dataset that is large and diverse enough to force the classifying network to learn robust representations of objects. However, collecting labelled training data is expensive. The most frequently used datasets for training CNNs contain hundreds of thousands of annotated images [Russakovsky et al., 2015, Lin et al., 2014]. The creation of such datasets requires labelling the images, drawing bounding boxes around objects or even creating pixel-based annotations for segmentation tasks, which is a time consuming task that usually has to be done by humans.

A cheaper way of increasing the variation in a training dataset is to employ automatic data augmentation pipelines. Automatic data augmentation provides a way of altering images while leaving the semantic content intact. It is therefore not necessary to relabel the augmented images. For instance, modifying color, brightness or contrast of an image would not change the object class or the segmentation mask, but could provide dozens of variants of the original image. Other label-preserving transformations include geometric distortions (rotation, translation, shear, elastic transformation), additive noise or image blur. Style transfer [Gatys et al., 2015] is a particularly interesting technique that allows to exchange the (usually photorealistic) style of an image with an artistic style extracted from another image, while the semantic content is left unchanged (compare the preview in figure 1.1). It has been shown that style transfer reduces the texture bias of CNNs and thereby increases the robustness [Geirhos et al., 2018b]. All these data augmentations can be combined to form a fully automatic, computationally cheap data preprocessing pipeline that increases variation in the training set without the need to label new data instances. Networks achieving state-of-the-art performance on visual tasks usually take advantage of such a preprocessing pipeline [Touvron et al., 2020, Chen et al., 2017, He et al., 2016].

Although being an important tool to facilitate the generalization of CNNs to unseen data, composition and parameterization of extensive data augmentation pipelines have moved to the focus of interest only recently. This stems from the fact that manually designing such a pipeline requires expertise and domain knowledge - for instance, image flipping is a very useful augmentation when it comes to object classification, but changes semantic information on digit recognition tasks. Furthermore, finetuning the configuration of the pipeline requires retraining CNNs in order to evaluate the effect of the data augmentation, which is time consuming. Recent work therefore applied metalearning methods in order to automate the search

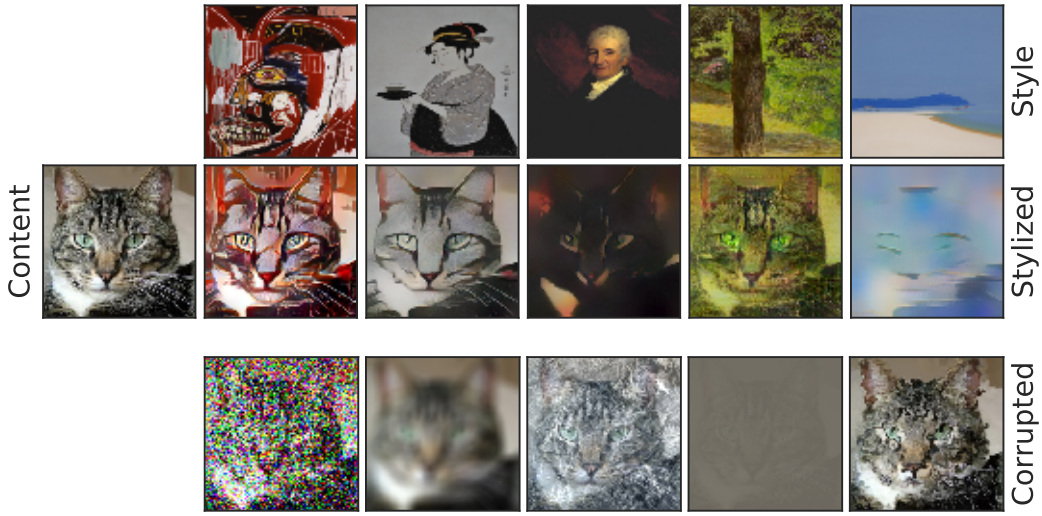


Figure 1.1: Effects of style transfer with different styles (top row) on the same content image are visualized in the middle row. Bottom row shows the effects of a subset of the common image corruptions used to measure model robustness.

for optimal augmentation policies [Cubuk et al., 2019, Ho et al., 2019, Cubuk et al., 2020], achieving state-of-the-art performance on ImageNet. While an increase in model robustness was reported for models trained on the optimized augmentation policies [Yin et al., 2019], the research focus was not to train robust models but to increase clean performance.

This work aims to extend the positive findings on robust models trained with data augmentation policies that were optimized using metalearning methods. So far, style transfer was not included in the set of augmentation techniques that were subject to the augmentation policy optimization, although leading to a significant robustness increase when applied to the training data. The reason for this is that style transfer is computationally expensive compared to other data augmentations, as it involves a forward pass through a transferer CNN.

In this work, a hyperparameter optimization approach is used in order to find an optimal parameterization of style transfer w.r.t. the robustness of the trained model. Style transfer is controlled by two parameters: The magnitude parameter controls the strength of the stylization, it can be seen as an interpolation parameter between 0 (meaning no stylization) and 1 (a fully stylized version of the image). The second parameter is the artistic style that is used for stylization. Applying style transfer with different styles produces vastly different results (compare figure 1.1).

In the literature on robustness and style transfer, it is usually applied at full magnitude. However, it is unclear whether a fixed magnitude setting results in most robust models. Using a schedule that increases the magnitude over training time and

thereby gradually increases the impact of the augmentation might be beneficial, as it provides a smooth transition between clean and augmented data. Such schedule effects are reported in Ho et al. [2019]. Furthermore, an analysis on the optimal magnitude value w.r.t. model robustness has not been done. While it seems plausible that stronger data augmentation leads to more robustness, this hypothesis has not been tested. The optimal magnitude value might be different for different styles, so on top of optimizing one parameter used for all styles, automatic hyperparameter optimization will be used in this work in order to search for optimal individual magnitudes for each style.

Concerning style selection, usually the style is randomly picked from the Painter-By-Numbers[1]dataset, containing almost 80.000 paintings by different artists. It is very likely that not all of these styles are needed for the robustness increase reported in Geirhos et al. [2018a] and Michaelis et al. [2019]. This work will examine how many styles are sufficient. Additionally, it is plausible to assume that not all styles are equally useful as a data augmentation. As shown in figure 1.1, some styles cause drastic modifications to the appearance of an image, while preserving the semantic content. Expectedly, these styles should be beneficial as a data augmentation. Other styles alter the image in a way that it is hardly recognizable to humans, and might therefore be harmfull to the model's performance as well. Instead of choosing styles with equal probability, a probability distribution for style selection will be learned. Expectedly, some styles will be chosen more frequently than others, which might provide insights into common qualities of styles that increase a models robustness most. Training on data stylized on the styles with maximum likelihood according to the learned distribution should result in an increase in robustness compared to training on data stylized on the same amount of randomly selected styles.

Contributions of this work can be summarized as follows:

- Comparison of various magnitude settings and schedules w.r.t. model robustness show that using stylization with fixed, maximum magnitude leads to highest model robustness. A positive linear relationship between magnitude and robustness is shown.

- The effect of style subset size on resulting model robustness is tested systematically, showing that a very small subset (64) of all available styles is sufficient for the robustness increase.

- Insight into common features of styles that have the largest effect on the robustness of a model is gained. There are striking qualitative similarities between styles that are beneficial to the model robustness and styles that have no positive effect on the robustness.

[1] `https://www.kaggle.com/c/painter-by-numbers`

- An efficient preprocessing pipeline applying style transfer with arbitrary styles to any image dataset on-the-fly is developed. An implementation for metalearning of data augmentation hyperparameters and augmentation selection probabilities is provided. Code for both is made openly available at `https://github.com/Hvitgar/MetalearningRobustness`.

2 Related Work

2.1 Model Robustness

The vulnerability of CNNs to small changes in the input has been the focus of several studies in different research areas. There are different notions of robustness. Adversarial robustness deals with the search for the smallest change in the input that causes misclassification. There is a whole research area on adversarial attacks and defenses. Adversarial attacks are usually imperceptible to humans, which makes them particularly dangerous in safety critical applications. The focus of this work however is robustness to common image corruptions. These corruptions are easily perceptible to humans, but also unavoidable if CNNs are applied in the wild. Natural corruptions of visual data include weather or lighting condition changes, occlusions and sensor noise.

Dodge and Karam [2016] investigate how image quality affects the performance of state-of-the-art neural networks for image classification. They find that neural networks are particularly vulnerable to Gaussian noise and blur. Geirhos et al. [2018b] compared the robustness of CNNs and humans to a variety of different image corruptions, finding that the network performance drops much faster than human performance. They also showed that CNNs trained on a subset of distortions do not generalize to other distortions at test time. In order to compare models w.r.t. their robustness, Hendrycks and Dietterich [2019] established the ImageNet-C benchmark consisting of 15 image corruptions on five severity levels and showed that standard trained networks perform poorly on corrupted test images.

Michaelis et al. [2019] showed that this phenomenon is not limited to image classification models. They evaluated several state-of-the-art object detection/segmentation models on various on the same set of image corruptions, measuring a decrease of 50% of a model's clean performance on corrupted data.

2.2 Data Augmentation

In deep learning, data augmentation serves several purposes. Most state-of-the-art models use handcrafted data augmentation strategies [Touvron et al., 2020, Chen et al., 2017, He et al., 2016] to reduce the generalisation error. Furthermore, when training data is expensive to generate or can't be generated at all (which is usually the case in medical applications), data augmentation provides a way to increase

the training set size [Hussain et al., 2017]. Advanced augmentation policies found with AutoAugment [Cubuk et al., 2019] have been shown to also increase the model robustness [Yin et al., 2019].

Among the most used augmentation methods are geometric transformations (scaling, translating, shearing, rotating), color space transformations (contrast/brightness adjustment), random cropping, flipping or elastic distortions [He et al., 2016, Chen et al., 2017, Ho et al., 2019, Cubuk et al., 2019, Touvron et al., 2020].

Additionally, although being introduced as an artistic tool that allows to merge the contents of one image with the style of another image, style transfer [Gatys et al., 2015] recently became popular as a complex, label preserving data augmentation method beyond linear transformations or color space augmentations. In the context of model robustness, applying style transfer to training images has been shown to increase the performance on perturbed test data compared to vanilla models [Geirhos et al., 2018a, Michaelis et al., 2019].

2.3 Hyperparameter Optimization

Most machine learning algorithms have hyperparameters which have a crucial impact on the outcome of the learning process. Fine-tuning them by hand is a time-consuming task and might lead to suboptimal results. Therefore, hyperparameter optimization is used to find good configurations. Standard hyperparameters subject to optimization algorithms are the learning rate, weight decay or momentum parameters of the optimization procedure [Bergstra and Bengio, 2012, Smith, 2018]. More recently, hyperparameter search was extended to include searching for the best optimizer for a given task, optimal data augmentation strategies or even the network architecture that produces the best predictions [Bello et al., 2017, Zoph and Le, 2016]. As the number of hyperparameters grows, manually tuning them to find good settings becomes a tedious task. On top, manually testing configurations requires training models for days in order to compare their performance. Therefore, there is a focus on research on automated hyperparameter optimization in the machine learning community. The algorithms commonly used for hyperparameter optimization can be grouped into the categories search-based, Bayesian and gradient-based methods. Each category will be discussed briefly in the following.

Search-based methods Straight forward methods for hyperparameter search include **grid search** and **random search** [Bergstra and Bengio, 2012]. For grid search, the hyperparameters have to be discretized. Every possible hyperparameter combination is then evaluated to find the configuration that performs best. Random search starts off at a random configuration. It then iteratively takes small steps in the hyperparameter space that maximize the objective. Both methods can produce sufficient results on low-dimensional hyperparameter spaces, but suffer from the

curse of dimensionality. The complexity of grid search scales exponentially with the number of dimensions of the search space [Bergstra and Bengio, 2012]. For random search, the rate of convergence scales exponentially with the dimensionality of the search space [Li et al., 2016]. In high dimensional spaces, search-based methods will need too many search iterations which is infeasible when evaluating the objective involves training of a CNN.

Search strategies like Hyperband [Li et al., 2016] or Population Based Training [Jaderberg et al., 2017] combine random search with a evolutionary explore/exploit strategy. Starting off with a set of random configurations, good configurations will be kept in the population, either unchanged or slightly modified (exploit). Bad configurations will be overwritten by either randomly sampled configurations or copies of existing, good configurations (explore). Both methods profit from reusing intermediate model checkpoints from earlier search branches and are therefore efficient tools for hyperparameter optimization in both discrete and continuous parameter search spaces. Still, since these methods rely on random search, they suffer from the same limitation as classical random search, namely exponentially scaled convergence rate with increasing search space dimensionality.

AutoAugment [Cubuk et al., 2019] fully automated the process of finding a good augmentation policy. The authors trained a reinforcement learning network as a controller that aims to maximize the reward on a proxy task, typically the validation accuracy of a relatively small neural network (compared to state-of-the-art models) trained with the optimized augmentation policy on a subset of SVHN,CIFAR10,CIFAR100 or ImageNet. The data augmentations included geometric transformations (translate, shear, rotate), color space transformations (invert, equalize, solarize, posterize, contrast, color shift, brightness) and cutout. Augmentations consisted of (operation, probability, magnitude) tuples. The probability parameter controls the likelihood that a certain operation is applied, magnitude controls the strength of the operation. An augmentation policy consisted of five sub-policies of two such tuples. Although both probability and magnitude were discretized by the authors, the search space was huge (10^{32}). For a full search, they had to train a total of 15.000 Wide-ResNet-40 models to convergence in order to determine the validation accuracy which was used as reward for the RNN controller. The resulting best policy was then used to train a larger model on the full dataset. The method achieved state-of-the-art on all four datasets.

In their follow up work, the authors introduced **RandAugment** [Cubuk et al., 2020], which tackles two of the practical limitations of AutoAugment. First, the policy search was infeasible to run for the ordinary researcher/user. Due to the large amount of sub-models that were trained during policy search, optimization took over 200 GPU days on CIFAR10 and over 600 GPU days on ImageNet. Second, they showed that optimal data augmentation policies differ between datasets and models. Using a proxy task (data subset, smaller model) during the policy search will therefore not necessarily find optimal policies for the real task. In fact, they showed that

optimal augmentation magnitude is directly dependent on the dataset size and the number of model parameters. For RandAugment, they significantly reduced the search space, eliminating the need for a separate policy search phase on the proxy task. The RandAugment data augmentation policy is just parameterized by two parameters: the number N of operations to be applied and the magnitude M for all augmentations. The N operations will be randomly selected from the set of available augmentations, all will be applied with the same magnitude M. This simplification reduced the search space size to 10^2 while not losing any performance gains reported in AutoAugment.

With **Population Based Augmentation** (PBA, Ho et al. [2019]), another approach for data augmentation policy search was introduced. PBA is based on the same exploit/explore strategy as Population Based Training. The policy search phase is split into stages. In the beginning, a number of models is trained with randomly initialized augmentation policies which are parameterized the same way as the augmentation policies in AutoAugment. At the end of each phase, the worst quartile of policies (w.r.t. validation accuracy) is dropped and the best quartile of policies is duplicated. With a low probability, each policy's parameters will then be modified or replaced with random parameters. While introducing stages further increases the size of the search space, the possibility to reuse models from earlier stages vastly reduced the computational cost of policy search. Furthermore, PBA learns an augmentation schedule instead of a fixed policy that is used throughout the whole training process. Intuitively, it might be useful to use lower magnitudes or a subset of less disruptive augmentations during early training stages and increase magnitudes or introduce harder augmentations later on. The authors showed that having a schedule instead of fixed policies or randomly shuffled subpolicies increases the model performance. They also found that their policies show a clear trend that magnitudes increase over training time and the set of augmentations used in each phase varies. Results matched the performance of AutoAugment while the cost for policy search was greatly reduced. However, PBA suffers from the same limitations as AutoAugment. Due to the number of models that have to be trained in parallel for the evolutionary policy search, they also used proxy tasks and models and transferred the best policies to the full datasets.

Bayesian methods Bayesian hyperparameter optimization aims to learn a probabilistic model of the true objective function. Configurations sampled w.r.t. that model are then applied to the true objective function, generating ground truth datapoints that are used to further refine the probabilistic model [Rasmussen, 2003]. Bayesian methods estimate good configurations with fewer update steps than grid/random search based methods, but updating the probabilistic model adds considerable computational cost to the hyperparameter search. However, compared to the cost of evaluating the objective function which usually involves training a CNN for days, the cost of updating the Bayesian model can be neglected. Bayesian Optimization

and Hyperband (BOHB, Falkner et al. [2018]) utilizes the Hyperband optimization for global search and the Bayesian approach for local refinement.

Gradient-based methods For very high dimensional or continuous search spaces the methods described above would fail as their computational cost scales with the dimensionality of the search space and both methods rely on the discretization of the parameter space. Gradient-based hyperparameter optimization can overcome that issue. Calculating the gradients of hyperparameters requires an estimate of how the optimal model parameters change w.r.t. the hyperparameters (best-response-function). In the hyperparameter optimization methods described above, this function is evaluated by training a model to convergence on a configuration of the hyperparameters, which is computationally infeasible for complex models in combination with high dimensional search spaces. In order to avoid training models to convergence for every hyperparameter update step, the implicit function theorem is used in Lorraine et al. [2019] to estimate how the hyperparameter updates affect the optimal model weights. Under assumptions (mainly that the model is close to convergence), the implicit function theorem allows to estimate the hypergradients without being able to evaluate the best-response-function. As long as small hyperparameter update steps are taken only after several network weight update steps, the assumptions for the IFT are met. As hyperparameter gradients are computed w.r.t. minimizing the validation loss, this method can only be applied to hyperparameters that directly influence the validation loss (for instance data augmentation parameters), but not to optimizer parameters like the learning rate.

3 Methods

The primary goal of this work is to provide insights into the ways style transfer can increase a model's robustness to image corruptions applied to the test set. Experiments will apply metalearning methods (described in section 3.2) to the parameters of style transfer (described in section 3.1) in order to find a configuration that increases the model's robustness to common image corruptions (described in section 3.3). The methods involved in the process are described in detail below.

3.1 Style Transfer

Style transfer was introduced as an artistic tool in Gatys et al. [2015]. The general idea is to combine the contents of one image with the artistic style of another image. Later work [Geirhos et al., 2018a, Michaelis et al., 2019] showed that it can be used as a data augmentation, increasing diversity in the training set beyond geometric or color transformations, which boosted the robustness to unseen image corruptions at test time.

In this work, style transfer is used as a data augmentation applied to the training data. It is parameterized by the style that is used for style transfer and the magnitude of the stylization (α). The implementation based on Adaptive Instance Normalization (AdaIN) described in Huang and Belongie [2017] applies style transfer using arbitrary styles in real time, which is crucial for this work. As the parameters of the style transfer are exactly the hyperparameters that are subject to the optimization procedure, they will change during the training process, meaning that style transfer must be applied on-the-fly. An overview over the style transfer pipeline is given in figure 3.1. For this method, feature maps for both content and style are computed via a VGG encoder network. In feature space, style modifications are done by aligning mean and variance of content features f_c with style features f_s (instance normalization):

$$\text{AdaIN}(f_c, f_s) := \sigma(f_s)\frac{f_c - \mu(f_c)}{\sigma(f_c)} + \mu(f_s) \tag{3.1}$$

where $\mu(\cdot)$, $\sigma(\cdot)$ are mean and standard deviation over the spatial dimensions of the feature map.

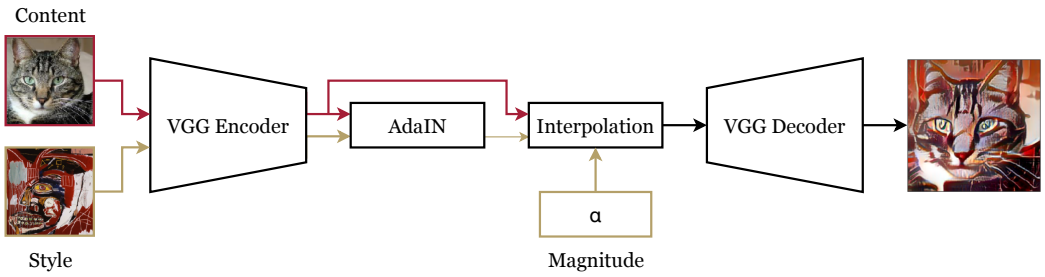


Figure 3.1: Styletransfer via adaptive instance normalization is performed in VGG feature space. The content features are normalized by mean and standard deviation of the style features. Afterwards, a linear interpolation (controlled by α) between normalized content features and style features is computed. The VGG decoder reconstructs the stylized image from that interpolation.

The magnitude of stylization is controlled via linear interpolation between f_s and the normalized content features controlled by the magnitude parameter α:

$$f := (1-\alpha) \cdot \text{AdaIN}(f_c,\ f_s) + \alpha \cdot f_s \tag{3.2}$$

For a given content image, the parameters controlling the outcome of the style transfer are the style that determines the style features f_s and the magnitude parameter α. In recent work on style transfer and model robustness [Geirhos et al., 2018a, Michaelis et al., 2019], styles were drawn randomly from the Painter-By-Numbers[1]dataset which consists of almost 80k artistic images and α was set to one throughout training, but it is plausible to assume that not all styles are equally useful in order to increase the model robustness and that a subset of styles is sufficient for the reported generalisation effect. In this work, both parameters will be optimised with the metalearning method described in the following section.

3.2 Gradient-based hyperparameter optimisation

In general, metalearning consists of an inner train loop that optimizes the model parameters and an outer train loop that optimizes the hyperparameters. Inner and outer loop might have different objectives and optimization procedures.

The metalearning procedure used to optimize the style transfer parameters (style selection and magnitude) in this work is based on Lorraine et al. [2019]. This method provides a way to optimize hyperparameters in the outer loop via backpropagation, contrary to recently published work that relied on random search or reinforcement

[1] `https://www.kaggle.com/c/painter-by-numbers`

learning approaches for the outer loop optimization [Cubuk et al., 2019, 2020, Ho et al., 2019]. This allows training model parameters and hyperparameters end-to-end in parallel. The model is first trained to convergence with fixed hyperparameters. After the initial training phase, hyperparameters can be optimized. Inbetween hyperparameter update steps, the model parameters are optimized for k iterations such that they can adapt to the hyperparameter changes.

First, an overview over the hyperparameter optimization method as introduced in Lorraine et al. [2019] is given. In the following subsections, it will be described how this method was adapted in this work to learn optimal magnitude parameters (section 3.2.1) and selection probabilities over the styles that are used for stylization (section 3.2.2).

Let $\mathcal{L}_T, \mathcal{L}_V$ be the training/validation loss, w the model parameters and λ the hyperparameters that are subject to the optimization. Hyperparameter optimization can then be formalized as follows:

$$\lambda^* := \arg\min_\lambda \mathcal{L}_V^*(\lambda) \tag{3.3}$$

$$\text{where } \mathcal{L}_V^*(\lambda) := \mathcal{L}_V(\lambda, w^*(\lambda)) \tag{3.4}$$

$$\text{and } w^*(\lambda) := \arg\min_w \mathcal{L}_T(\lambda, w) \tag{3.5}$$

The optimal hyperparameters λ^* are the ones that minimize the validation loss given optimal model parameters w^*. Optimal model parameters are the ones that minimize the train loss. As data augmentation is applied only to the training data, $\mathcal{L}_T$ is directly dependent on the hyperparameter choice while $\mathcal{L}_V$ is only indirectly (via the optimal model parameters w^*) dependent on λ.

Without training a model to convergence for every update step however, it is impossible to determine the best response function $w^*(\lambda)$. Lorraine et al. leverage the implicit function theorem to calculate the gradients of the implicit function $w^*(\lambda)$. This allows for an estimate of how the best response function behaves when the hyperparameters change.

In order to perform backpropagation through the optimization described in section 3.2, all parts involved in calculating $\mathcal{L}_T$ and $\mathcal{L}_V$ must be differentiable. Let $\mathcal{M}_w$ be the classification model parameterized by the model weights w, $\mathcal{D}_T, \mathcal{D}_V$ the training and validation dataset and $(i_T, l_T) \in \mathcal{D}_T$ an image/label pair from the training dataset (analogue notation will be used for the validation dataset). The classification loss is a scalar function that involves the model's prediction of the class of an image and the ground truth label:

$$\mathcal{L}_T : \mathcal{M}_w(i_T), l_T \to \mathbb{R} \text{ (analogously for } \mathcal{L}_V) \tag{3.6}$$

If data augmentation is applied, the train loss can be reformulated to

$$\mathcal{L}_T : \mathcal{M}_w(\text{AUG}_\lambda(i_T)), l_T \to \mathbb{R} \tag{3.7}$$

with the augmentation function AUG_λ mapping an image to its augmented version. The hyperparameters λ affect the way in which this augmentation function alters the input. In this work, style transfer is used as the augmentation function. Style transfer is parameterized by α, denoting the strength of the stylization applied to an image and the image the style is transferred from (compare figure 3.1). In the following, it will be discussed that both L_T and L_V are differentiable w.r.t. both style transfer parameters.

3.2.1 Hyperparameter optimization for alpha

The magnitude parameter α of style transfer controls the influence of the style on the stylized image. It ranges between 0 (no style transfer is performed) and 1 (the resulting image fully adapts the artistic style). Since this parameter is used for interpolation in VGG feature space (compare equation (3.2)), it is easy to perform backpropagation w.r.t. α. The hyperparameter update step just involves backpropagation through the VGG decoder, the linear interpolation function between content and style features and the linear AdaIN layer (equation (3.1)).

3.2.2 Hyperparameter optimization for style selection

Determining particularly good or bad styles w.r.t. corruption robustness can be done by learning probabilities $\pi_i, 1 \leq i \leq N$ of a categorical distribution over a set of N styles. During training, the styles used for style transfer are then drawn from that distribution. However, selection based on samples from a categorical distribution is a discrete process which does not allow for backpropagation.

Jang et al. [2016] introduced the Gumbel-Softmax distribution as a continuous, differentiable relaxation of the categorical distribution. Given a sample $g \sim \text{Gumbel}(0,1)$, the probabilities π_i and a temperature parameter $\tau \in \mathbb{R}^+$, the Gumbel-Softmax sample can be computed as follows:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{N} \exp((\log(\pi_i) + g_j)/\tau)} \text{ for } i = 1, \ldots, N \tag{3.8}$$

Sampling from the Gumbel(0,1) distribution can be done by drawing $u \sim \text{Uniform}(0,1)$ and computing $g = -\log(-\log(u))$.

The influence of the temperature τ is shown in figure 3.2. As τ approaches zero, expectation over samples from the Gumbel-Softmax distribution approach the expectation over one-hot vectors drawn from a categorical distribution with the same class probabilities π_i while samples approximate one-hot samples from a categorical distribution.

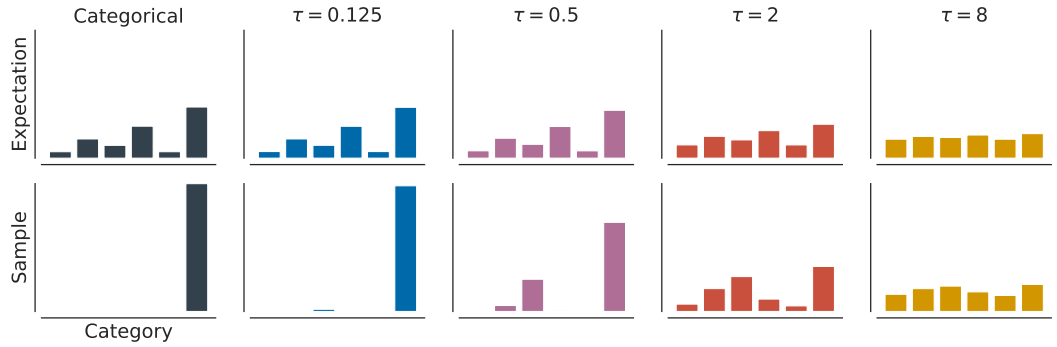


Figure 3.2: The Gumbel-Softmax distribution is a continuous relaxation of the categorical distribution. For low temperatures, Gumbel-Softmax samples approximate one-hot samples from a categorical distribution while the expectation over samples approximates the class probabilities. For higher temperatures, samples as well as the expectation converge towards a uniform distribution over the classes.

As backpropagation w.r.t. π_i through the Gumbel-Softmax distribution is possible, a distribution over the styles can be learned. In order to have a one-hot vector for style selection, but still be able to backpropagate through Gumbel-Softmax to the class logits $\log(\pi_i)$, it is necessary to use the gradient of the continous relaxation y (the sample from Gumbel-Softmax) for the backward pass, but use its discrete one-hot representation z for the forward pass:

$$y \sim \text{Gumbel-Softmax}(\pi, \tau) \tag{3.9}$$

$$z_i = \begin{cases} 1 & \text{if } i = \arg\max_j y_j \\ 0 & \text{else} \end{cases} \quad \text{for } i = 1, \ldots, N \tag{3.10}$$

The learned class probabilities should represent the usefulness of a style w.r.t. the hyperparameter optimization objective.

3.3 Measuring corruption robustness

An established way to estimate the robustness of a model is to apply perturbations at test time to the model's input [Hendrycks and Dietterich, 2019]. In this work, the `imagecorruptions` python package[2] was used. It provides 15 common image corruptions on five severity levels. A preview of all corruptions is given in figure 3.3. In order to determine the robustness of a model w.r.t. unseen corruptions, these 15 corruptions cannot be applied to the test set. However, the hyperparameter

[2] `https://github.com/bethgelab/imagecorruptions`

Figure 3.3: Image corruptions provided in the ImageNet-C benchmark [Hendrycks and Dietterich, 2019] applied at maximum severity to the same content image. While humans are able to recognize image contents in most cases, the performance of CNNs drops drastically on higher severity levels.

optimization outer loop requires an estimate of the model robustness that can be used as the outer loop objective. The `imagecorruptions` package provides an additional four holdout corruptions that are meant to be used for cross validation and will be used to estimate the corruption robustness of a model during hyperparameter optimization.

The measure used in the following to determine the corruption robustness of a model is mean performance under corruption (mPC), which was introduced in Michaelis et al. [2019]. It averages the accuracy of the model over the 15 common corruptions shown in figure 3.3 applied to each testing image on all 5 severity levels. Higher mPC means that the model is more robust to corrupted image data.

4 Experiments

A series of experiments was set up to investigate the influence of different parameter configurations of style transfer on the robustness of the resulting model. There are two parameters involved in style transfer: The magnitude parameter α controls the strength of the applied stylization. A magnitude value of 0 corresponds to no style transfer applied to the input, while a magnitude value of 1 results in a maximally altered artistic style of the image (compare equation (3.2)). The second parameter of style transfer is the image from which the artistic style is extracted (compare equation (3.1), figure 3.1). In other works that apply style transfer in order to train robust models, magnitude was set to 1 at all times and the style was drawn uniformly from a set of almost 80.000 styles available in the Painter-By-Numbers[1]dataset.

Experiments on the magnitude parameter include setting α to various fixed values (section 4.1.1), increasing α step-wise over the training (section 4.1.2) and optimizing α w.r.t. model robustness (section 4.1.3). Experiments on the style selection include investigating the effect of the number of available styles on the robustness (section 4.2.1), learning a probability distribution for the selection of particularly useful styles and using distilled, very small style subsets consisting of the best styles according to the learned distribution (section 4.2.2).

The last set of experiments (section 4.3) investigates how results generalize from a subset of ImageNet to the full dataset. Methods evaluated in this work are compared to human performance on corrupted data.

The aim of the experiments is to provide answers to the following questions:

- What is the best way to set the magnitude parameter α of the style transfer?
- Are there particularly good/bad styles w.r.t. model robustness? Is a subset of styles sufficient for the increase in robustness reported in other work [Geirhos et al., 2018a, Michaelis et al., 2019]?

All experiments were conducted in the realm of image classification. Due to the computational overhead resulting from the metalearning outer loop, most experiments are conducted on a sixteen-class subset of ImageNet (referred to as ImageNet16 in the following). The number of examples per class is balanced in ImageNet16, the training set contains 1000 examples per class (with two exceptions) and the validation set contains 50 examples per class, resulting in a total of 15487

[1] `https://www.kaggle.com/c/painter-by-numbers`

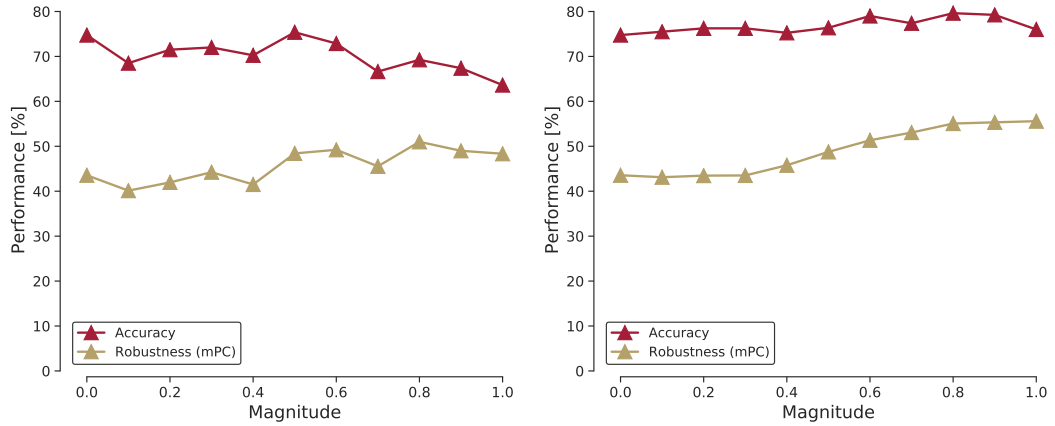


Figure 4.1: Clean accuracy and mean performance under corruption for models trained using different levels of fixed magnitude. **Left:** Results from models that used stylized training data from the start. **Right:** Results from models that were pretrained on clean data before being introduced to stylized training data.

training images and 800 testing images. For reproduction, the full subset description (file paths) for the ImageNet16 dataset is given at `https://github.com/Hvitgar/MetalearningRobustness`. The last set of experiments investigates generalization of the results to the full ImageNet dataset as well as generalization to a set of image corruptions that pose a hard problem even to humans.

If not stated otherwise, all experiments are conducted with a ResNet18 [He et al., 2016]. Relevant training parameters, training scripts and code for reproduction are published at `https://github.com/Hvitgar/MetalearningRobustness`. Whenever style transfer is applied to the training data, half of the training data is left as clean data. It has been shown in Michaelis et al. [2019] that there exists a tradeoff between clean accuracy and robustness when style transfer is applied at training time. This tradeoff can be reduced by mixing clean and stylized training data.

4.1 Magnitude parameter

The following experiments investigate the influence that different settings of the magnitude parameter have on the robustness of the model. Settings that were used include fixing the parameter throughout training, gradually increasing the parameter over the training time and using metalearning methods to optimize the parameter.

4.1.1 Fixed magnitude

The first set of experiments aims to answer the question, whether stylization on higher magnitude values leads to an increase in robustness. For this set of experiments, the magnitude parameter was set to a fixed value for the whole training time. Magnitudes ranged between 0 and 1 in steps of size 0.1. Models trained with fixed magnitudes will be referred to as ResNet18+FM.

The robustness of the resulting models, evaluated on the corrupted test set, is plotted in figure 4.1. On the left, results when style transfer was applied from the beginning of the training are shown, on the right results when the models were pretrained to convergence on clean data before style transfer was applied to the training data are shown. Clearly, both clean accuracy as well as performance under corruption are better for models pretrained on clean data. Especially for larger magnitudes, using style transfer at the start of training results in diminished test accuracy (drop of 7.5 percentage points between $\alpha = 0$ and $\alpha = 1$). Therefore, for all following experiments models were pretrained to convergence before style transfer was introduced to the training procedure.

Interestingly, for models pretrained (converged) on clean data, even clean performance increases with increasing magnitude values. This is due to the fact that style transfer introduces a lot of variation to the ImageNet16 dataset which is rather small (1000 examples per class with few exceptions). Models trained on clean data alone seem to overfit, applying style transfer acts as a regularization, preventing overfitting on the training set which results in higher accuracy on the testing data. This effect is also reported on the full ImageNet dataset [Geirhos et al., 2018a], but as overfitting is much more likely on small datasets, the regularization effect observed from applying style transfer is larger on small datasets too.

The results from the experiments on fixed magnitude values indicate that applying strong stylization, especially early on in the training, hurts the performance of the model on clean data. The increase in robustness of the models with increasing magnitude levels came at the cost of decreasing performance on clean data (especially visible for models that were not pretrained on clean data). Style transferred data can be seen as a whole new domain of training data, which is vastly different from the clean training images. The magnitude parameter controls the gap between clean and stylized domain. When setting this parameter instantly to large values (> 0.5 for non-pretrained, > 0.7 for pretrained models), there is a tradeoff between clean accuracy and robustness which is not present at lower magnitudes.

4.1.2 Increasing magnitude

Using a parameter schedule for data augmentation might be beneficial. Intuitively, applying less disruptive data augmentation early in the training and stronger, more disruptive data augmentation at the end of training might result in better

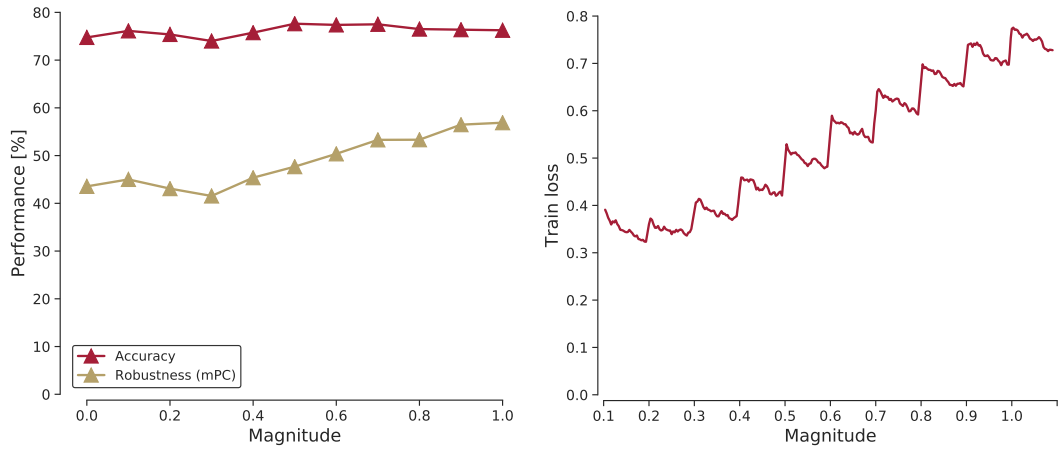


Figure 4.2: Results measured on ResNet18+IM. **Left**: Robustness increases with increasing magnitude, clean performance slightly increases as well due to reduced overfitting to the training set. **Right**: Train loss increases with increasing magnitude, while test performance isn't reduced - an indicator for overfitting on the clean training set.

overall performance, as the gap between clean data and augmented data increases over time. This pattern is observed in Ho et al. [2019], where the magnitudes of transformations increased over training time for optimal augmentation policies. The authors claim that having a parameter schedule increases performance compared to fixed parameterization. Therefore, for this experiment the magnitude parameter was increased step-wise by 0.1 every 30 epochs in order to keep the domain gap between clean and stylized training data small. Increasing the magnitude step-wise during training provides a smooth transition from the clean domain to the stylized domain, benefitting from the increasing robustness at higher magnitudes without losing clean accuracy. The model trained with this procedure will be referred to as ResNet18+IM.

The resulting clean accuracy and model robustness are shown in figure 4.2 on the left. After increasing the magnitude to 1, the resulting model has the highest robustness (mPC = 56.9%) as well as the highest clean accuracy (P = 76.3%) of all models discussed so far (compare table 4.1).

On the right of figure 4.2, the train loss is plotted. The regularization effect discussed before can be seen here as well: With increasing magnitude, the train loss goes up, while the clean test accuracy remains unchanged (compare figure 4.2 on the left). Increased train loss in combination with unchanged test performance indicates that in fact the model pretrained on clean data only (which was used as the starting point for this experiment) overfitted to the training set. Applying style transfer to the training data increases the training set size and therefore introduces variance to the training examples, which counteracts overfitting observed especially on small

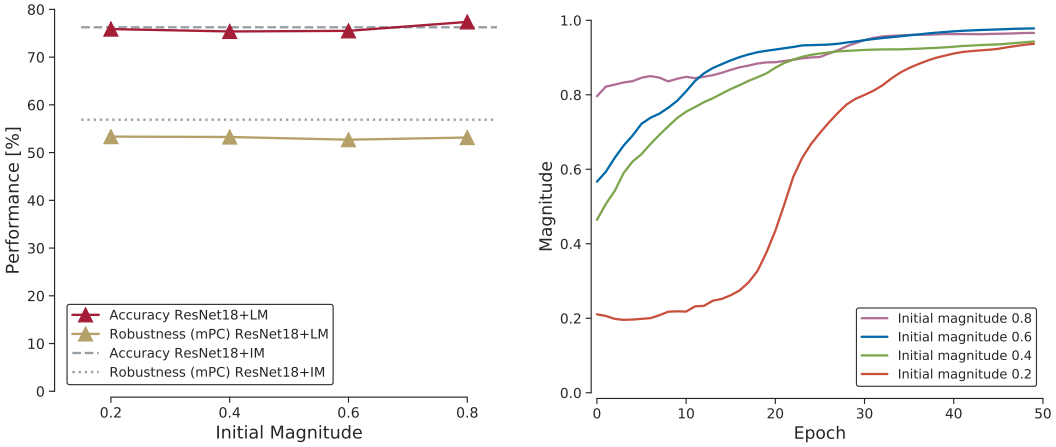


Figure 4.3: Results measured on ResNet18+LM. **Left**: Clean performance and mPC of the converged models are similar irrespective of the initial magnitude. **Right**: Magnitude values over the course of the hyperparameter optimization procedure.

datasets like ImageNet16.

4.1.3 Learned magnitude

As shown above, the way in which the magnitude parameter is set influences the robustness of the resulting model. For the next experiments, instead of selecting the magnitude (or a magnitude schedule) by hand, it was optimized it via hypergradient estimation [Lorraine et al., 2019] in parallel to training the model. The aim of this experiment is to investigate whether optimization of the magnitude parameter can lead to a value/schedule that increases the robustness compared to the experiments above. Starting from a model pretrained on clean data, the initial magnitude was set to $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$. These models will be referred to as ResNet18+LM in the following.

The objective of optimal hyperparameters is to increase the robustness of the model being trained. Therefore, the outer loop training objective used to estimate hyperparameter gradients has to be related to a robustness measure. As the whole idea of robustness is to test on corruptions that have not been part of the training set, instead of using the common corruptions, four holdout validation corruptions (speckle noise, gaussian blur, spatter, saturate) from the `imagecorruptions` python package[2] are used as an estimate of the robustness to unseen image corruptions.

Results are shown in figure 4.3. On the left, the resulting performances are compared to the performance from the ResNet18+IM model at magnitude $\alpha = 1$ discussed above. Models with optimized magnitude parameter matched clean accuracy, but were less

[2] `https://github.com/bethgelab/imagecorruptions`

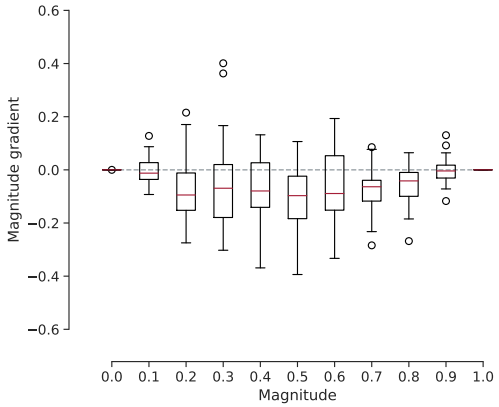


Figure 4.4: Magnitude gradients estimated on the validation split on different magnitude levels. Boxes show the variance over batches taken from the validation split. Most of the mass of the boxes is below zero on all magnitude levels, meaning that optimizing the magnitude based on these gradients would result in an increased value.

robust. On the right, the magnitude is plotted over the hyperparameter optimization process. For all initial magnitude values, the learned parameter trended quickly to $\alpha \approx 0.9$. The depicted schedule is very close to the schedule used for ResNet18+IM, but convergence to large α is way quicker for the optimized magnitude, meaning that during training, there is less time for the models to adapt to higher levels of style transferred data.

The results presented above together with the results from the ResNet18+IM experiments suggest that the most robust models are trained using the maximum magnitude value. In order to analyze this finding further, the hypergradients of the magnitude are plotted in figure 4.4. For the plot, hypergradients were estimated over one full iteration through the validation split of the training set on models converged on several magnitude levels. The plot illustrates that most of the time the gradients were negative, meaning that optimizing the magnitude would result in an increasing value as gradient descent is performed in order to minimize the validation loss. This is exactly the behaviour observed in figure 4.3 and on the ResNet18+IM model.

Optimizing a single magnitude value used for all styles does not result in an increase in model robustness, mainly due to the strong evidence for the optimal solution $\alpha = 1$. However, learning individual magnitude parameters for every style might lead to a better understanding of which styles contribute best at what magnitude level to a robust model.

In order to optimize the magnitude for every single style, the styles must be kept in GPU memory to perform backpropagation efficiently. Therefore, for the following

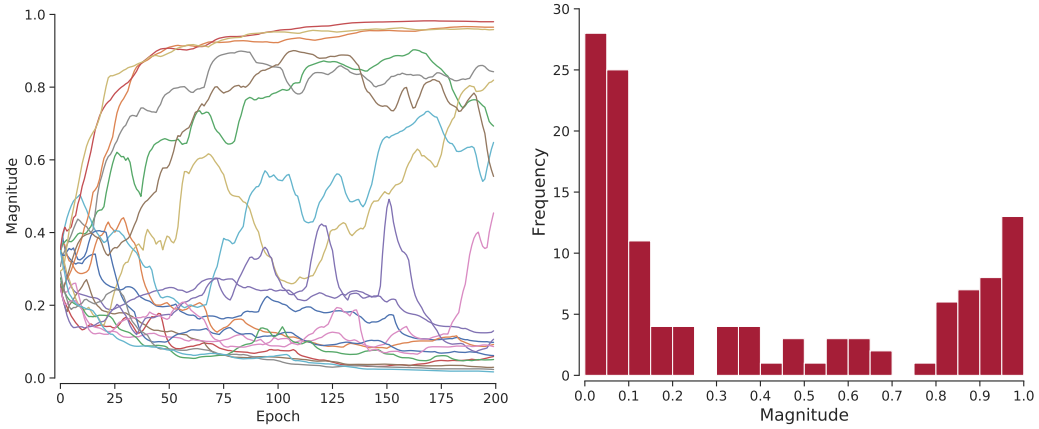


Figure 4.5: Results from optimizing one magnitude parameter per style. **Left**: Magnitude parameters of 20 randomly selected styles over the course of the hyperparameter optimization. For some styles, a clear convergence pattern towards low or high magnitude values exists, while the parameter oscillates for others. **Right**: Histogram over learned magnitude values for all 128 styles at the end of training. Low magnitudes < .25 dominate, magnitudes between .25 and .75 are rare.

experiment a random subset of 128 styles was used to reduce memory consumption. It will be argued in section 4.2.1 that using a subset of size ≥ 64 does not reduce the robustness of the resulting model significantly compared to using all styles.

The resulting model only shows an increase in robustness over a vanilla ResNet18 trained solely on clean data (mPC = 49.9% with individual learned magnitudes, mPC = 43.6% for a vanilla ResNet), but does not reach ResNet18+IM robustness (mPC = 56.9%). Performance on clean data is also increased compared to the vanilla model (78.4% vs. 74.8% clean accuracy), mainly because of the regularization effect of stylized training data that reduces the overfitting present in the vanilla model. A sample of learned magnitude values is depicted in figure 4.5 on the left. For some styles, there is a clear trend towards minimum or maximum magnitude, while some others oscillate between high and low magnitude over training. Magnitudes at the end of training are summarized in the histogram in figure 4.5 on the right. For most styles (72), magnitude is below 0.25 at the end of the training. Only 21 styles end up with a learned magnitude value between 0.25 and 0.75, while 35 styles end up with a learned magnitude greater than 0.75, which stands in contrast to the results reported above, where higher magnitude always resulted in higher robustness. Still, the pattern that was observed in the experiments above holds here as well: either use clean data (low magnitude) in order to optimize for clean accuracy, or maximally stylized data (high magnitude), which increases the robustness the most.

This experiment suggests that there is no clear connection from validation loss (performance under holdout corruptions) to the optimal magnitude value a certain

style is applied with, as a more robust solution exists with the ResNet18+IM model. In fact, even setting the magnitude to 1 for all styles (as it is done on the ResNet18+FM model) results in a more robust model. Therefore, finetuning the level of stylization applied to the data via gradient-based hyperparameter optimization with the presented method can not be used to increase the model's robustness/generalization to novel image corruptions at test time.

4.2 Style selection

In this section, experiments examining the influence of the styles that are used for style transfer are described. It is clearly not necessary to use all available styles to observe the maximum robustness gain from style transfer. But it is unclear how many styles are sufficient. For the first experiment, styles are picked with equal probability from random subsets of different sizes in order to estimate minimum size of the style subset needed for the generalization effect.

As already observed in the experiment on individual magnitude optimization (4.1.3), there are styles that contribute more to the robustness increase than others (reflected in the fact that some styles are applied at magnitude 1 while others are applied at magnitude 0). In the second experiment, it will be examined whether a probability distribution for the style selection can be learned, that favors particularly useful styles for training robust models instead of picking styles with uniform probability.

4.2.1 Random subsets

The Painter-By-Numbers dataset consists of almost 80.000 images providing different artistic styles. It is highly unlikely that all these styles are needed for the training of robust models. In order to examine the number of styles sufficient to increase the robustness, models were trained on subsets of increasing size. Subsets were chosen randomly from all available styles, the styles were picked uniformly from the subset for stylization of the training data.

Results are shown in figure 4.6. Interestingly, using 64 styles is already enough to observe comparable robustness to a model trained using all styles. Using less styles results in a lower clean performance (due to a diminished regularization effect counteracting overfitting) and robustness (due to less variation in the datasets), while using more styles does not increase accuracy or robustness further. As discussed by Cubuk et al. [2020], optimal hyperparameter settings are in general not transferable accross tasks/datasets/models. For a dataset consisting of more classes such as ImageNet, more styles might be needed to create more variance in these classes. On the other hand, for datasets with a high number of examples per class, the overfitting effect reported above might not be as bad, meaning that the regularization effect of style transfer is not as necessary and few styles are sufficient as well.

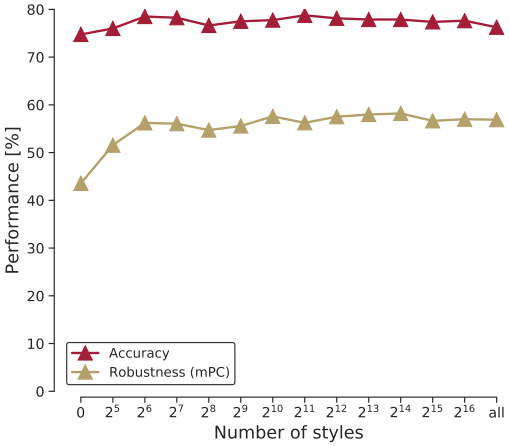


Figure 4.6: Effect of the size of the set of available styles used for style transferring the training data. Using more than 64 styles does not result in a significant increase in model robustness.

4.2.2 Refinement via subset optimization

As it was shown above, a small amount of styles is already enough to observe the effects of style transfer. Still, not all styles might be equally useful for this effect. Instead of selecting styles based on uniform probabilities, it might be beneficial to learn a probability distribution that favors useful styles over neutral or even harmfull styles. As shown in figure 4.9, there are styles that completely erase image content and therefore cannot be considered useful for any task.

Two experiments were conducted on style distribution learning. Both aimed to learn a distribution over a total of 128 available styles. As the distribution was optimized in parallel to being used for the selection of styles, it was important to make sure that styles that were assigned low probabilities early on in the training were still selected and used in order to get a gradient estimate on their probability. Otherwise, if the probability of a style was low enough once, it would rarely get any updates, even if there might be support for a positive effect of that style later in the optimization. To avoid this, the temperature parameter was set to a high value (8) at the start of the training and halved every few epochs. Thereby, the learned distribution was similar to a uniform distribution irrespective of the learned probabilities early on in the training, and converged more and more to the learned categorical distribution over the styles as the temperature decreased (compare figure 3.2). As hyperparameter optimization objective, once again performance on holdout corruptions was used.

The first experiment can be considered a proof-of-concept experiment. Half of the styles were replaced by zeroes, contents of images stylized with these would be completely destroyed. The other half of the styles is left intact. The resulting learned distribution over this set of styles is shown in figure 4.7 on the left. The left 64 styles

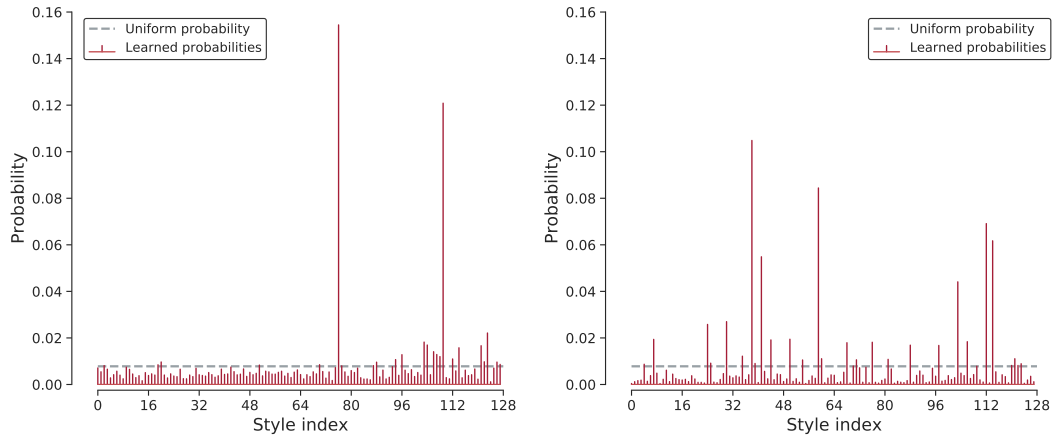


Figure 4.7: Learned probability distribution over the styles. **Left**: Styles 0-63 were overwritten with zeros, destroying any image contents they are applied to. **Right**: Style distribution learned on a subset consisting of 128 randomly selected styles. There is a clear distinction between a few highly frequently used styles (with $p_i > 0.05$) and the majority of styles that were selected with probability $p_i << 0.01$.

are the ones destroying image contents. It can be seen that the resulting distribution would pick from the bad styles with probabilities lower than uniform probability in almost all cases, while styles in the right half get assigned values above uniform probability as well. The learned distribution reflects the fact that some styles are harmfull to the model's performance.

For the second experiment, a distribution over the full subset of 128 styles was learned. Figure 4.7 (right) shows the resulting selection probabilities. Interestingly, the learned distribution seems to favor very few styles heavily, while most of the styles are picked with probabilities lower than uniform. Clean accuracy and performance under corruption at the end of the distribution optimization matched the performance of a model trained with a uniform distribution over 128 styles.

All of the most favored styles ($p_i > 5\%$) were among the styles with individual learned magnitude $\alpha_i > 0.75$ in the experiment above (ResNet18+LM). In figure 4.8 (left), the learned probabilities are plotted against learned individual magnitudes on the same styles. There is a slight positive correlation between magnitude and probability of a style (r = 0.27). Fitting a linear regression model to the data (golden line in figure 4.8 (left)) reveals a significant linear relationship ($p < 0.05$) between learned probabilities and magnitudes, even when the 6 potential outliers with learned probability >0.04 are excluded. This indicates that there is some agreement on the usefulness of a style between both experiments. A style that is assigned a high probability (and is therefore considered useful) is expected to also be applied at high magnitude as this produces the most robust models. With few exceptions, styles with low magnitude get assigned low probabilities and styles with high magnitude get assigned high

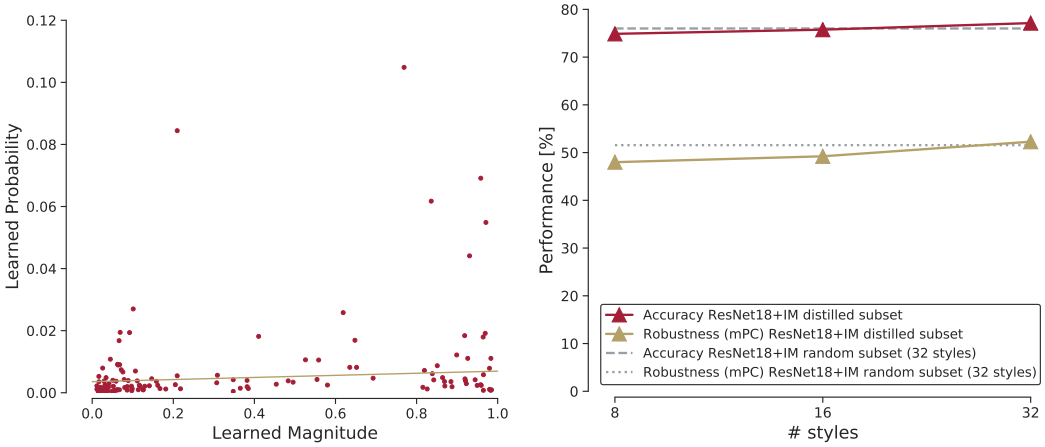
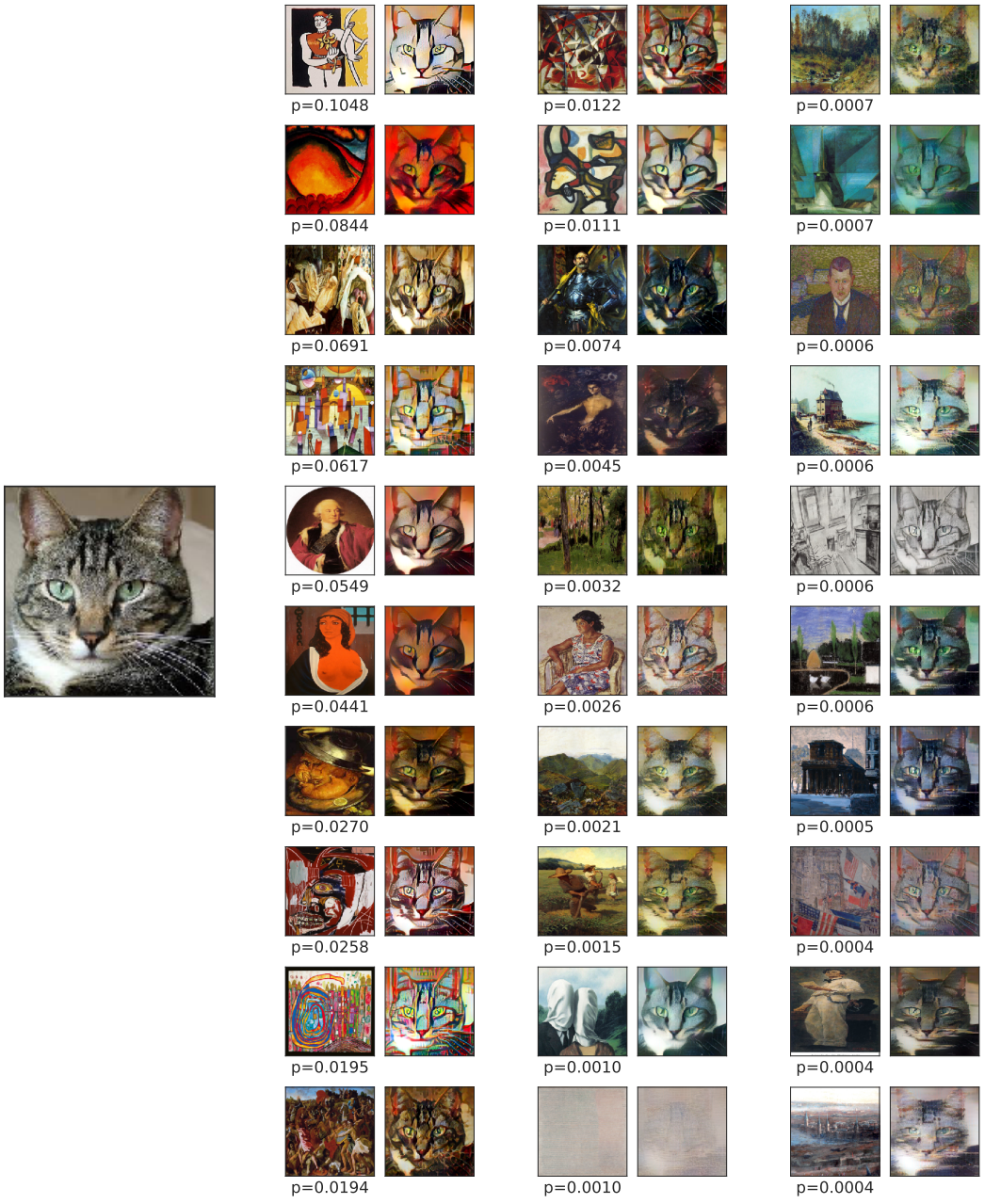


Figure 4.8: Left: Learned probabilities plotted against learned magnitudes of a subset of 128 styles. Styles applied frequently got assigned high magnitude as well, styles assigned a low magnitude (reducing the effect of stylization) are also assigned low probability values. Regression line fitted to the data is plotted in golden, showing a slight positive correlation even when potential outliers (with probability >0.04) are excluded. **Right**: Results of applying stylization with styles picked from a small subset of size [8, 16, 32] formed of the most frequent styles w.r.t. a learned distribution over a subset of 128 styles.

probabilities as well. However, this might not be true the other way around. Even if a style is assigned a low probability, it might be best to apply that style at high magnitude when it is used.

Figure 4.9 visualizes the ten most favored (left), least favored (right) and ten randomly chosen styles (middle) for comparison applied to the same content image. Qualitatively, the most frequent styles seem to change the local texture and color composition of the image, fur structure and whiskers are masked. These are image features that are lost quickly when applying blur/noise corruptions or digital corruptions like jpeg compression or pixelate as well. For the least frequently used styles, the stylized version looks like a merely colorshifted version of the original image. Fur marks and whiskers as well as some of the fur structure is preserved. As also discussed in Geirhos et al. [2018a], vanilla classification networks tend to put a lot of focus on local textures instead of shape, while shape is a much more robust classifying feature than local texture when the image gets distorted. It seems that styles that modify local texture but keep the contours intact are preferred over styles that also preserve some of the local texture. Interestingly, a style that blends out image content to a point where it is almost unrecognizable for humans (middle column, last row) is still assigned a higher selection probability than the monotone styles from the right column.

Common qualities of the styles used most frequently are several radiant colors and



(a) Content image

(b) Most frequent styles

(c) Randomly selected styles

(d) Least frequent styles

Figure 4.9: The same content image stylized on magnitude $\alpha = 1$ using the ten most frequent, ten random or the ten least frequent styles according to the learned probability distribution. Selection probabilities are printed below each style.

strong contours, while the least frequently used styles share blurry contours, washed out colors or lots of gray/white/light blue.

For the last set of experiments, models were trained with stepwise increasing magnitude on subsets of the most used styles according to the distribution depicted on the right of figure 4.7. Subset sizes were 8, 16 or 32. The performance of models trained on images stylized with these small style subsets is shown on the right of figure 4.8. The generalization effect observed on 32 random styles can almost be reached by using the top 8 styles according to the learned distribution, it is surpassed by using the best 32 styles. Due to computational resources it was out of scope to repeat that experiment but learn a distribution on a significant part of the Painter-By-Numbers dataset, but the results indicate that selecting styles w.r.t. a learned distribution might lead to slightly higher robustness than the best values reported on ResNet18+IM.

4.3 Transfer to ImageNet

The best performing setup on ImageNet16 (increasing the magnitude by 0.1 every 6 epochs for a total training time of 60 epochs) was used to train a ResNet50 on the full ImageNet dataset. This model will be referred to as ResNet50+IM in the following. Results are reported in table 4.1. On top of clean performance and mPC, mean corruption error (mCE, introduced in Hendrycks and Dietterich [2019]) was measured as well. Model performances are compared to a vanilla ResNet50 and the ResNet50 SIN+IN model from Geirhos et al. [2018a], which is trained on the concatenation of the clean training set and the training set stylized with random styles on magnitude $\alpha = 1$. The procedure is the same as the one described for the ResNet18+FM experiment.

Compared to the vanilla ResNet50, a clear increase in mPC (and therefore a decrease in mCE) can be seen, the method increases the robustness of the trained model. This increase is traded for a decrease in clean performance however. The ResNet50 SIN+IN model is both more robust and performs better on clean data.

Models were also tested against a set of more challenging distortions that was introduced in Geirhos et al. [2018b], including additive uniform noise, contrast, low/high pass filtering, eidolon manipulations and phase noise. A preview is given in figure 4.10. All distortions are parameterized and can therefore be applied at different strengths. The authors found that a ResNet50 trained on stylized data only (ResNet50+SIN) perfomed best on these distortions. Results on the distortions for vanilla Resnet50, ResNet50+IM and ResNet50+SIN are plotted in figure 4.11, human performance on the data (measured by and reported in Geirhos et al. [2018a]) is included for comparison as well. On most of these challenging image distortions, ResNet50+IM performs comparable to the vanilla ResNet50 and is outperformed by ResNet50+SIN. Significant differences in performance are only observed on high-

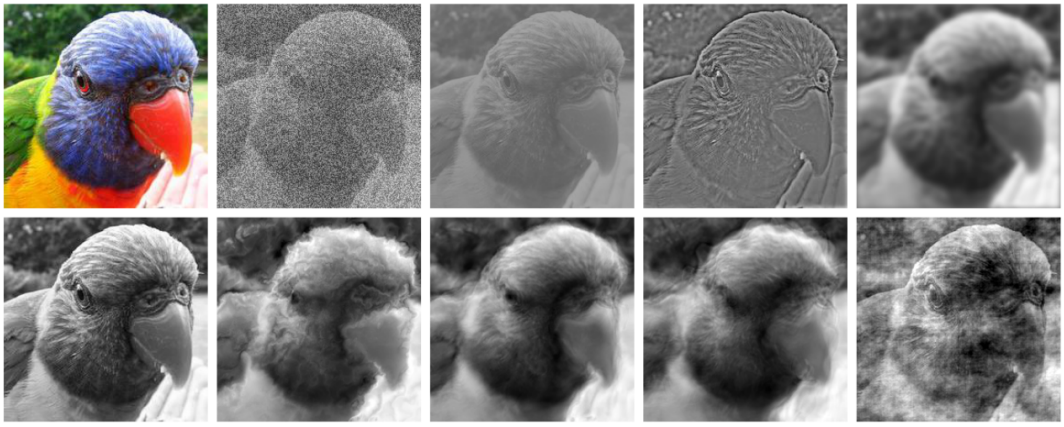


Figure 4.10: Visualisation of image distortions. Original images (example top left) are grayscaled (bottom left) before applying the distortion. Distortions were (from left to right): Additive uniform noise, low contrast, high-pass filter, low-pass filter (top row); Eidolon manipulations I, II, III, phase noise (bottom row). Adapted from Geirhos et al. [2018a] with the author's permission.

and low-pass filters. ResNet50+IM is less susceptible to high-pass filtered images compared to a vanilla ResNet50, while on higher distortion strengths ResNet50+SIN is more robust to high-pass filtered images. High-pass filtering destroys local image texture, but keeps edges (shapes) intact (compare figure 4.10). Applying style transfer forces the model to focus on shape features instead of local texture, which explains the performance gain observed for the models trained with stylized images. As the low-pass filter retains the most of the local texture compared to the other distortions in the set, the opposite effect is observed here: vanilla ResNet50 and ResNet50+IM, which were trained mostly on non/less strong stylized images outperform ResNet50+SIN.

Geirhos et al. [2018a] presented a method to make explicit whether classification decision are based on object shape or texture by creating conflicting cues consisting of the shape of one class (for instance a cat), but the texture of another (for instance elephant). Their experiments exhibited a strong texture bias in standard trained CNNs, while humans tested on the same cues classified objects based on shape in almost all cases (see figure 4.12). As object shape is much more robust to all corruptions discussed in this paper, it is not surprising that humans outperform CNNs on these corruptions (compare figure 4.11). The authors also showed that the texture bias in CNNs can be reduced by training models on stylized training data. The texture bias of ResNet50+IM is lower than vanilla ResNet50 but higher than ResNet50+SIN (figure 4.11), which makes sense as the data used for training the ResNet50+IM model can be seen as an interpolation between data used for ResNet50 and ResNet50+SIN respectively.

There are two possible explanations for this. First, the effect observed using increasing

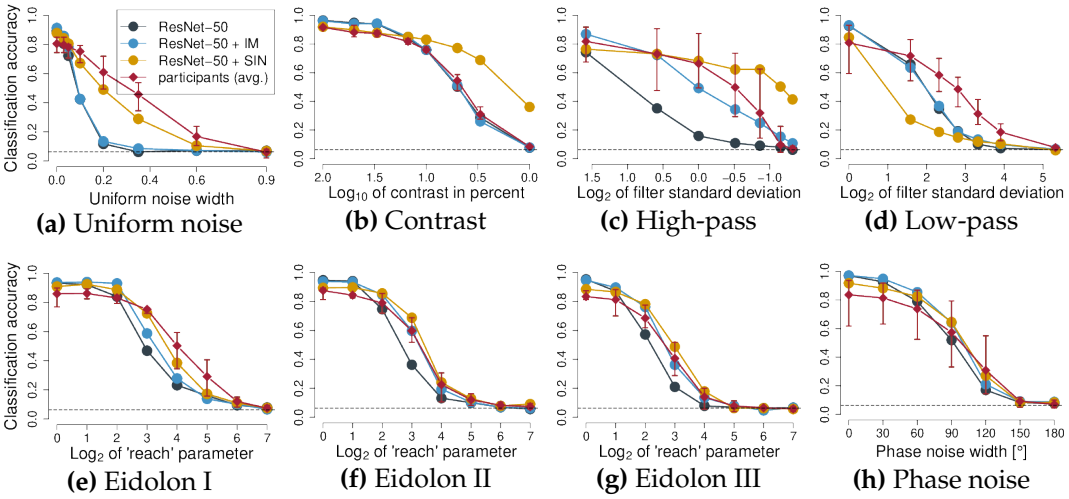


Figure 4.11: Classification accuracy on distorted images for vanilla ResNet50, ResNet50+IM (ours), ResNet50+SIN and human participants (as reported in Geirhos et al. [2018a]). Models trained on stylized data have an advantage over the model trained on clean data. Humans perform equally well or better on most distortions.

magnitude on ImageNet16 stems mainly from reducing the existing overfitting in the pretrained network. This effect would not transfer to the full ImageNet dataset as overfitting happens less on a dataset of that scale. Second, training setups between ResNet50+IM and ResNet50+SIN are different: ResNet50+IM is trained for 6 epochs on a magnitude level before the magnitude is increased by 0.1, meaning that on the highest magnitude levels (>= 0.7) that contribute most to the robustness of the model, only 24 epochs are trained. Meanwhile, ResNet50+SIN was trained for 45 epochs on magnitude $\alpha = 1$. Repeating the experiment but increasing the number of epochs per magnitude level might result in an increase in observed robustness. Due to the computational cost of applying style transfer on the fly, increasing the number of epochs was out of scope for this work.

Additionally, ResNet50+SIN used a learning rate schedule starting with an initial learning rate of 0.1 diveded by factor 10 every 15 epochs, allowing for drastic parameter changes at the beginning. Such a schedule is not useful in combination with a magnitude schedule (as highest learning rates would be used in combination with lowest magnitudes) - however, using an adaptive optimizer such as Adam [Kingma and Ba, 2014] in combination with a magnitude schedule might result in higher robustness as well.

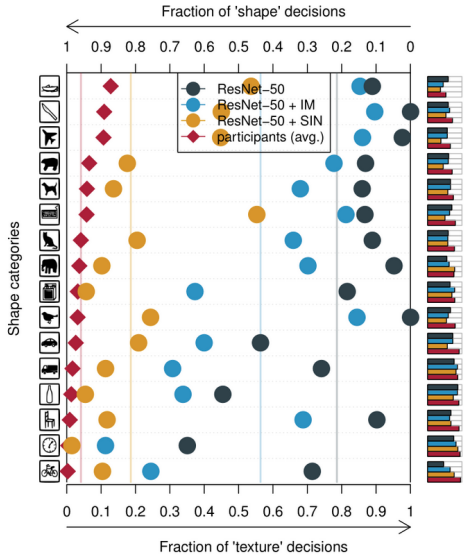


Figure 4.12: Fraction of shape decisions vs. texture decisions taken by CNNs and humans on cue-conflict data. Fractions are build for decisions that corresponded to either correct shape or correct texture. The barplots on the right display the ratio of correct decisions (either texture or shape) to all decisions taken. Humans exhibit a clear shape bias, a standard ResNet50 is biased towards the texture decision. If stylized data is presented to the models during training, decisions are shifted towards shape.

Name	Clean accuracy	mPC	mCE	Comments
		ImageNet16		
ResNet18	74.75	43.55		
ResNet18+FM	76.00	55.59	-	best mPC at $\alpha = 1$
ResNet18+IM	76.25	**56.89**	-	best mPC at $\alpha = 1$
ResNet18+LM	75.88	53.34	-	best mPC at initial magnitude $\alpha = 0.2$
ResNet18+LM	**78.38**	49.86	-	individual learned magnitudes per style
ResNet18+IM	77.12	52.28	-	on a distilled 32-style subset consisting of the most frequent styles according to the learned distribution
		ImageNet		
ResNet50	76.13	39.17	76.70	
ResNet50+IM	73.29	43.72	70.91	
ResNet50 SIN+IN	**76.72**	**45.22**	**69.30**	reported in Geirhos et al. [2018a]

Table 4.1: Summary of all described experiments. For experiments with different configurations, results from the configuration with highest robustness are reported. For clean accuracy and mPC: highest=best, for mCE: lowest=best

5 Discussion

The experiments showed that image corruptions at levels where humans can still easily classify images pose huge problems to CNNs trained with standard methods. Applying style transfer to the training data and thereby introducing variance to the data helps increasing the robustness of a model to unseen image corruptions. Experiments were conducted on different parameterizations of style transfer, the robustness of the resulting models was compared. First, a broad overview over the experiments is given, followed by a discussion of the results, their relevance and practical implications.

Magnitude parameter Regarding the magnitude parameter, settings included using fixed magnitude on different levels, increasing the magnitude step-wise over training, using hyperparameter optimization to learn either one magnitude value for all styles or individual magnitudes per styles. Results indicate that higher magnitude leads to higher robustness. Additionally, a regularization effect of style transfer was observed that also leads to higher clean accuracy compared to models trained on clean data solely. There is only a slight advantage in robustness when using increasing magnitude compared to fixing it throughout training on the ImageNet16 subset, this effect could not be reproduced on the full ImageNet dataset. Optimization of the magnitude parameter supported the finding that the highest magnitude leads to the most robust models. Learning individual magnitudes per style did not result in an increase in robustness, however it indicated that there are styles that are clearly beneficial to the robustness (and are therefore assigned high magnitude values) and styles that have no positive effect (and are therefore turned off by reducing their magnitude to 0). In all cases, pretraining on clean data to convergence before applying style transfer leads to higher clean accuracy and robustness of the resulting models and is therefore strongly advised.

The results on optimal parameterization of the magnitude of style transfer have very practical implications. In this work, as the parameter was allowed to change during training, style transfer had to applied on-the-fly to the training data, resulting in an increase of computational effort involved in training the models. As the maximum magnitude value produces the most robust models, and there is no indication of a significant effect of a magnitude parameter schedule on the model robustness, this on-the-fly application of style transfer is not necessary. Training images can be stylized before training on maximum magnitude and stored on disk, during training stylized and clean data can simply be loaded and mixed, as it was done in

Geirhos et al. [2018a], Michaelis et al. [2019]. The results of experiments in this work suggest that the robustness gain from style transfer reported in other work could not have been improved using a different parameterization.

Additionally, as learning individual magnitudes did not lead to increased robustness either, it is not necessary to reapply the hyperparameter optimization process to each model architecture and training dataset. As argued in Cubuk et al. [2020], this would be necessary as optimal policies are specific to the architecture and dataset they were optimized on. The experiments on optimal magnitude parameterization support the hypothesis that the simplest way to apply style transfer (on maximal magnitude for all styles) produces the most accurate and robust models. This is particularly important as application of style transfer in combination with metalearning methods is very expensive compared to other, more standard ways of data augmentation, which are mostly linear transformations of the image data. On top, the effort invested in hyperparameter optimization usually scales with the complexity of network architectures. As state-of-the-art results often come at the cost of a more complex architecture, this would mean that also the overhead of applying hyperparameter optimization increases.

Style subset composition Experiments on the style subset that is used for style transfer showed that the robustness gain can already be reached by using only a randomly chosen subset of 64 out of almost 80.000 available styles. Learning a probability distribution on a subset of styles and selecting styles based on it resulted in a slight increase in robustness compared to a randomly selected subset of styles. Styles that were assigned high selection probabilities differed qualitively from styles with low selection probabilities. Features of frequent styles were appearance of several radiant colors as well as strong contours. The appearance of images styled with these styles changed drastically. While contours were kept intact and to some extent even enhanced, local image texture and color composition of the stylized images were completely different to their original counterpart. Features of styles that were picked least frequently include a washed out, monotone color palette, gray/light blue/light green as the dominant color and blurry image quality. Images stylized with these styles appeared like a merely colorshifted version of the original.

The experiments on the size of the style subset showed that a small number of styles (64) is enough for training a model with the desired robustness increase. It is not necessary to use more styles, as this did not increase robustness further. This is a very interesting outcome in terms of resource management: the almost 80.000 images in the Painter-By-Numbers dataset occupy 38GB of disk space. The experiments on style subset size suggest that using 64 randomly chosen styles is already sufficient to observe the positive effects of style transfer, which would reduce the memory footprint by three orders of magnitude.

Clearly, selecting a random subset of 64 styles is not the optimal way to maximize the robustness gain. By learning a probability distribution over a small subset of styles and just using the most frequent styles according to that distribution, a slight increase in robustness could be observed. over using the same number of randomly picked styles. This indicates that there are in fact styles that contribute more to a model's robustness than others. As there was a significant positive correlation between learned magnitude and learned probability, this further supports the assumption that pre-filtering a small subset of useful styles out of the Painter-By-Numbers dataset can in fact increase the generalization ability of a network to unseen data or common image corruptions.

Repeating the experiment on a larger scale on the full ImageNet dataset might provide interesting insights into common features of styles that boost model robustness. Although a significant increase in model robustness could not be reported for models trained on a distilled subset of styles, it is plausible that using only styles that are considered useful in terms of increasing a model's robustness might lead to decreased training time needed to reach that increase compared to using all styles or a subset of randomly picked styles.

Furthermore, the qualitative differences between styles that were assigned high probabilities and styles that were assigned low probabilities provide insight into style features that benefit a model's robustness the most. Frequently selected styles showed much more radiant colors, the visual quality of the images was much better and there was more contrast and contours in these styles compared to the styles with low selection probability. More strikingly, most frequent styles are almost exclusively abstract paintings, while least frequent styles consist mostly of natural drawings. Introspectively, the most frequent styles depicted in figure 4.9 are connected to a feeling of stimulation, they are more fascinating to humans than the low probability versions. This impression holds as well for the images stylized using the respective styles. There might be a connection between an aesthetical feeling of interest and the usefulness as a style used for style transfer, as the frequent styles seem to be further away from the domain of natural images, while styles with low selection probability appear to be very close to natural images.

Interestingly, even styles that modify the image such that it becomes hard to recognize the content for humans are considered more useful (assigned higher probabilities) than just natural drawings. The pattern that can be observed here seems to be: It is better for a style to cause a lot of change in the image, even if visual quality is diminished, than to just modify the color composition of the image (as was mostly the case for low probability styles).

It would be interesting to learn more about the characteristics of styles that increase the generalization ability of CNNs the most by learning a distribution on a much larger subset of the styles. Due to computational demand, this was out of scope for this work, but the results on the small subset indicate that there might be image statistics that are common to frequent or less frequent styles. Identifying such

characteristics or statistics can enable research in the direction of generative models, that, when applied in an adversarial training setting, can lead to further insight into the directions of variation that benefit robustness the most, but are missing in standard training sets.

Other ways to increase robustness Style transfer is not the only technique that increases robustness of a model. Yin et al. [2019] reported an increase in robustness when augmenting the data with a policy over standard data augmentations (including translation, shearing, rotation, color/contrast/brightness modifications) that was optimized with AutoAugment [Cubuk et al., 2019]. In their work, they analyze the fourier spectrum of images augmented with different common image corruptions and a model's robustness to these corruptions, showing that adversarial training or gaussian noise augmentation only leads to robustness w.r.t. high frequency corruptions. Interestingly, their experiments showed that using low frequency data augmentations during training does not increase the robustness to low frequency corruptions at test time. Instead, using a wide variety of different data augmentations seemed to increase the robustness to both high and low frequency image corruptions. Rusak et al. [2020] achieved lower mCE than ResNet50 SIN+IN by augmenting the training images with additive noise. In contrast to the findings described above, although using high-frequency noise as their main data augmentation they reported state-of-the-art results on most low-frequency corruptions. Lopes et al. [2019] applied patch-wise gaussian noise to training images, combining the robustness increase observed from using additive noise with the clean accuracy increase reported for patch-wise cutout data augmentation. All these augmentation strategies can be applied after style transfer, increasing the variation in the training set even further, which might lead to more robust models as well.

The literature on robustness is diverse, and it remains an open research question what is the best technique or ensemble of techniques in order to train a model that shows robustness to common image corruptions that is comparable to the human visual system. Additionally, even achieving good performance on the common corruptions benchmark introduced by Hendrycks and Dietterich [2019] does not mean that models are robust to natural weather phenomenons, day/night shifts or various lighting conditions. It remains unclear how general robustness to naturally occuring image distribution shifts can be achieved or even assessed. The best way so far is to introduce as much different corruptions to the testing pipeline as possible.

Best practice To summarize the discussion, a best practice guideline on how to use style transfer in your training setup is given. If stylized training images are preprocessed once and stored to disk, the overhead of using stylization is minimal. Style transfer has been shown to increase both clean accuracy and robustness of a variety of models and datasets on image classification, object detection and semantic

segmantation tasks, making it an extremely useful way of augmenting your training data.

1. Start with pretraining on clean data to convergence (apply any other data augmentation technique as usual).
2. If memory requirement is an issue, select a random subset of at least 64 styles from the Painter-By-Numbers dataset.
3. Precompute the stylized training set: Stylize each training image with at least one style at maximum magnitude.
4. Fine-tune the model on half stylized, half clean training data (apply any other data augmentation technique as usual).

Bibliography

I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le. Neural optimizer search with reinforcement learning, 2017.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks, 2017.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

S. Dodge and L. Karam. Understanding how image quality affects deep neural networks. *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, Jun 2016. doi: 10.1109/qomex.2016.7498955. URL `http://dx.doi.org/10.1109/QoMEX.2016.7498955`.

S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale, 2018.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2018a.

R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann. Generalisation in humans and deep neural networks. *CoRR*, abs/1808.08750, 2018b. URL `http://arxiv.org/abs/1808.08750`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.90. URL `http://dx.doi.org/10.1109/cvpr.2016.90`.

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

D. Ho, E. Liang, I. Stoica, P. Abbeel, and X. Chen. Population based augmentation: Efficient learning of augmentation policy schedules. *arXiv preprint arXiv:1905.05393*, 2019.

X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

Z. Hussain, F. Gimenez, D. Yi, and D. Rubin. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA Annual Symposium Proceedings*, volume 2017, page 979. American Medical Informatics Association, 2017.

M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu. Population based training of neural networks, 2017.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2016.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *Lecture Notes in Computer Science*, page 740–755, 2014. ISSN 1611-3349. doi: 10.1007/978-3-319-10602-1_48. URL `http://dx.doi.org/10.1007/978-3-319-10602-1_48`.

R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation, 2019.

J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation, 2019.

C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming, 2019.

C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

E. Rusak, L. Schott, R. S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel. A simple way to make neural networks robust against diverse image corruptions, 2020.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Apr 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL http://dx.doi.org/10.1007/s11263-015-0816-y.

L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018.

H. Touvron, A. Vedaldi, M. Douze, and H. Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet, 2020.

D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer. A fourier perspective on model robustness in computer vision, 2019.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning, 2016.