UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Safe Augmentation: Learning Task-Specific Transformations from Data

Author:
Irynei BARAN

Supervisor:
Arseny KRAVCHENKO

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2019

Declaration of Authorship

I, Irynei BARAN, declare that this thesis titled, "Safe Augmentation: Learning Task-Specific Transformations from Data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Abstract

Faculty of Applied Sciences

Master of Science

Safe Augmentation: Learning Task-Specific Transformations from Data

by Irynei BARAN

Data augmentation is widely used as a part of the training process applied to deep learning models, especially in the computer vision domain. Currently, common data augmentation techniques are designed manually. Therefore they require expert knowledge and time. Moreover, optimal augmentations found for one dataset, often do not transfer to other datasets as effectively. We propose a simple novel method that can automatically learn task-specific data augmentation techniques called **safe augmentations** that do not break the data distribution and can be used to improve model performance. Moreover, we provided a new training pipeline for using safe augmentations for different computer vision tasks. Our method works both with image classification and image segmentation and achieves significantly better accuracy on CIFAR-10, CIFAR-100, SVHN, Tiny ImageNet and Cityscapes datasets comparing to other augmentation techniques.

Acknowledgements

I would like to express my appreciation to Arseny Kravchenko and Orest Kupyn for providing supervision and consultations, and for sharing their experience in the computer vision domain. Additionally, I give my gratitude to the WANNABY company for providing computational resources. Also, I would like to thank Rostyslav Hryniv and Olexii Molchanovskyi for organizing research process and providing invaluable feedback. Without all of you, this work would never be finished.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We, as humans, can analyze and interpret 3D surroundings for all our purposes and intents quite easily. Computer vision is a science that intends to give this ability to machines. Computer vision is being utilized today as a part of many applications, including object acknowledgment, medical imaging, autonomous driving, surveillance, 3D modeling, visible authentications, etc. (Parashar, 2017)

1.1 Motivation

Deep neural networks achieve human-level or even higher performance in many computer vision tasks, such as image classification, image restoration, image or video segmentation, etc. (Zahangir Alom et al., 2018). For example, the human top-5 image classification error on the ImageNet dataset is 5%, whereas the current state-of-the-art deep neural networks achieve nearly 3% (Russakovsky et al., 2015).

However, deep learning models require a massive amount of training data, and this is often a big problem. Data augmentation is one of the approaches that can help to solve the problem by expanding training data by doing transformations that preserve semantic information and class labels. The choice which data augmentation techniques to use for the specific dataset and task is not a trivial one. While some augmentation could help the model to learn and generalize better, others could make the model performance even worse. For instance, a horizontal flip is proven to be useful augmentation for ImageNet-like datasets, but not for the MNIST dataset (LeCun and Cortes, 2010), because horizontally flipped digits are often no longer valid digits. Currently, common data augmentation techniques are designed either empirically or by leveraging expert knowledge. Hence, it takes a lot of time and efforts.

The main idea of our research is to make the process of choosing data augmentations automatic, namely to learn from data which augmentation techniques will lead to model generalization and accuracy improvements.

1.2 Goals of the master thesis

1. To provide an overview of previous works on reducing overfitting.

2. To explore recent examples of automatically learned data augmentations.

3. To present our approach and compare results.

Chapter 2

Related Work

In this section, we provide a brief review of previous commonly used works that have dealt with reducing overfitting and improving generalization of the model. We also explore recent research on performing data augmentation automatically and compare them with our approach.

2.1 Deep learning. Overfitting

Deep learning had made a huge step forward in many domains, including computer vision. In 2012, the very first deep convolutional neural network called AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) won the ImageNet Large Scale Visual Recognition Challenge (Russakovsky et al., 2015) and achieved significantly better accuracy in visual object recognition than any other traditional machine learning or computer vision algorithms. This was a crucial breakthrough, and since then, almost all state-of-the-art algorithms in computer vision domain are based on deep learning.

Deep neural networks(DNNs) contain multiple non-linear hidden layers with a large number of parameters, and thus they can learn very complex relationships between inputs and outputs. However, DNNs often tend to perform very well on training data and poorly on unseen data, especially when trained on a relatively small train set - this problem is called *overfitting* (Hawkins, 2004). To reduce it without requiring a massive amount of training data, different methods have been developed. We provide a short overview of several of them.

Weight decay. The idea is to decrease the model complexity with adding a constraint to the growth of the weights through some kind of weight decay (Krogh and Hertz, 1992). It can be achieved by adding a term to the cost function that penalizes large weights.

$$E(w) = E_0(w) + \frac{\lambda}{2}\sum_i w_i^2, \tag{2.1}$$

where E_0 is the loss function and λ is the parameter that controls how much the magnitude of the weights will be penalized. Weight decay is widely interpreted as a form of L_2 regularization because in case of gradient descent

they are equivalent. However, it is not the case for other types of optimization algorithms, i.e., Adam or K-FAC. It is shown that the weight decay regularization improves the generalization of the model consistently (Zhang et al., 2018a).

Dropout. The idea is to both reduce overfitting and provide a way of approximately combining exponentially many different neural network architectures efficiently. It is done by randomly removing units of the neural network along with their incoming and outgoing connections. The unit is "dropped" with the fixed probability p independently of other units. Dropout is proven to work in different domains, showing significant success in object classification and speech recognition (Hinton et al., 2012). One of the drawbacks of this approach is the training time. Each training case tries to learn different random architecture. Thus, the training time of network with dropout is 2-3 times slower compared to a standard network (Srivastava et al., 2014).

Early stopping. The idea is to stop the training of a neural network before it has overfitted the training dataset. Typically the generalization error is estimated by the average error on a validation set. In the simplest case, there are four steps how to do early stopping:

1. Split the dataset into train set and validation set.
2. Train model on the train set and evaluate the error on the validation set periodically, e.g., every 1, 2 or 5 epoch.
3. Stop the training once the error on the validation set increases compared to the previous evaluation.
4. Use weights of the model with the smallest validation error.

However, training neural networks can be noisy in the real world. Hence, validation error often has more than one local minimum and stopping at first sign of overfitting may not be a good idea. Therefore, some delay in model stopping is almost always used (Prechelt, 1998). Early stopping is perhaps one of the most commonly used forms of regularization in deep learning, because of its simplicity and effectiveness.

2.2 Data augmentation

Data augmentation is another approach that reduces overfitting by artificially increasing the size of the dataset and enhancing it using label preserving transformations. Each transformation function can be expressed as the following mapping:

$$\phi : D \rightarrow T, \tag{2.2}$$

where D is the original dataset and T is the augmented set of D. The term label preserving means that given an image x from class y, augmented image $\phi(x)$ should belong to the same class y. The final training set D' is then

represented as:

$$D' = D \cup T \tag{2.3}$$

Data augmentation has been used for quite a long time, e.g., back in 1997

FIGURE 2.1: Examples of simple label preserving transformations using the image taken from the Tiny ImageNet dataset (provided by Fei-Fei Li, Andrej Karpathy, and Justin Johnson as a part of the cs231n course at Stanford University http://cs231n.stanford.edu/).

Yaeger, Lyon, and Webb were generating additional data using small changes in skew, rotation and scaling during the training of a character classifier (Yaeger, Lyon, and Webb, 1997).

Traditional augmentations. Common practice for image data is to use geometric transformations, such as flipping, cropping, rotating, scaling, etc., and color transformations, such as adjusting color, brightness, resolution, etc. They are often called generic or traditional augmentations (2.1). They all fall under the category of data warping and are usually performed in the data space, e.g., Wong et al. have shown that it is better to perform data augmentation in data space than in feature space as long as label preserving transforms are known (Wong et al., 2016). This type of transformations is easy to use and efficient to implement. One main disadvantage is that you need to have expert knowledge in the image domain to choose transformations that will not break the labels of the images. Traditional augmentations are broadly used and have shown excellent results in reducing overfitting and improving model performance (Perez and Wang, 2017; Taylor and Nitschke, 2017).

Generative Adversarial Networks (GANs). In 2014 Goodfellow et al. have proposed a new class of neural networks that can generate realistic data from scratch using generator and discriminator networks that are trained in the minimax two-player game framework (Goodfellow et al., 2014). GANs can be used as a form of unsupervised data augmentation by generating new data from the source distribution. They have also been used for style transfer, e.g., transferring images from one weather condition to another (2.2). These generated images can be used to help the model to work in different conditions, for instance, to train autonomous cars to drive in night or snow, having collected data from sunny weather only. GANs are also shown to be successfully used for data augmentation in the medical imaging domain by synthetically augment mammogram and MRI images (Bowles et al., 2018; Wu et al., 2018).

FIGURE 2.2: Example of style transfer using CycleGAN (Zhu et al., 2017).

2.3 Comparison with other works

The recent paper by Hernández-García and König has shown that data augmentation alone can achieve the same or even higher performance than explicit regularization techniques (weight decay, dropout, etc.), without wasting model capacity (Hernández-García and König, 2018). Despite described advantages, common data augmentation methods are usually dataset-specific and designed manually, which require prior expert knowledge and time.

Our idea is to learn from data which transformation methods to use for the specific dataset and task. Recently, a lot of interesting works have been done aiming to automate the process of data augmentation. We dived them into the following two groups.

Generate data transformations. Cubuk et al. proposed a new procedure called AutoAugment to learn augmentation policies that lead to the highest accuracy of the model on a given dataset. They created a search space of data augmentation policies and used search algorithm based on reinforcement learning to find the optimal one. The results are great: they were able to achieve state-of-the-art accuracy on CIFAR-10, CIFAR-100, SVHN, and ImageNet datasets. Moreover, It is shown that policies learned from one dataset can be transferred to other similar datasets. One of the drawbacks of AutoAugment might be the computational complexity and long training time due to the extensive search space of possible policies (Cubuk et al., 2018). Ratner et al. learned generative sequence model over user-defined transformations using GAN-like framework. Their idea is to compose and parameterize a set of user-specified transformation functions in ways that are diverse but still preserve class labels. Their approach allows leveraging domain knowledge flexibly and straightforwardly. (Ratner et al., 2017).

Generate augmented data directly. Smart Augmentation proposed by Lemley, Bazrafkan and Corcoran can automatically generate augmented data by merging two or more samples from the same class, in a way that reduces the loss of the original model (Lemley, Bazrafkan, and Corcoran, 2017).

DeVries and Taylor proposed domain-independent data augmentation technique by using simple transformations in the learned feature space. They train a sequence autoencoder to construct a learned feature space in which they extrapolate between samples (DeVries and Taylor, 2017). Tran et al. introduced a novel Bayesian method for generating additional data based on the distribution learned from the training set (Tran et al., 2017). Generative adversarial networks have been extensively used for producing augmented data. For example, Antoniou, Storkey, and Edwards presented DAGAN - An image conditional GAN-based model that learns from one data item how to generate other realistic within-class data items. DAGAN can be applied to unseen classes of data and can also enhance few-shot learning systems. (Antoniou, Storkey, and Edwards, 2017). Another approach called DADA: Deep Adversarial Data Augmentation was proposed by Zhang et al. to train deep learning models in extremely low data regimes. They show that that DADA outperforms both traditional data augmentation and a few GAN-based options (Zhang et al., 2018b).

Chapter 3

The Proposed Method

We present a simple approach of automatically learning which transformation methods do not break the data distribution and can be used to improve the model performance. Our learned set of transformations are called safe augmentations, and we propose to use them for fine-tuning the already pre-trained models.

Our method is computationally efficient and can be easily performed along with the original task. It is shown in the Experiments section (5) that using safe augmentations leads to significant performance improvements of image classification and semantic segmentation tasks on several different datasets.

3.1 Learning safe augmentations

We propose to learn safe augmentations from data using a convolutional neural network (CNN). Consider a dataset D and a set of all available augmentation techniques A. The task is to define which transformations from set A do not break the distribution of the D, i.e., to select $S \subset A$, where S is a set of safe augmentation. Our pipeline can be divided into the four main steps.

Step 1. Train the CNN to solve the following multi-label classification problem. Given a set A, for every batch of images, a random subset $a \subset A$ is applied. The subset a is of random size from 0 to the defined maximal size. In our experiments, we used maximum size of 5. Each transformation from subset a is applied with the probability $p = 1$. The model tries to predict which augmentations were applied. As a loss function, we use L_{augm} - a multi-label one-versus-all loss based on max-entropy, between input x and target y. L_{augm} is equivalent to applying *sigmoid* function along with the binary cross entropy loss.

$$L_{augm}(x,y) = -\sum_i y[i] * \log((1+\exp(-x[i]))^{-1}) + (1-y[i]) * \log\left(\frac{\exp(-x[i])}{(1+\exp(-x[i]))}\right) \tag{3.1}$$

where $i = 0, \ldots, x.length - 1$, and $y[i]$ in $\{0, 1\}$.

Step 2. After the model is being trained, evaluate it on the unseen test data without any augmentations and collect per-label false positives, i.e.,

how many times the model predicts that the specific augmentation technique was applied when, in fact, it was not.

Step 3. Evaluate the model on the unseen test data using the same procedure of applying augmentations as in the training phase. Collect per-label classification accuracy for each transformation technique.

Step 4. Divide all augmentations into two groups: safe and other. If it is hard for the model to distinguish whether a particular transformation was applied and the model never predicts it on the clean set, then we believe that this transformation cannot break the data distribution and can be safely used during the training of the original task. Thus, we consider augmentation as safe if it has relatively low per-label classification accuracy on the test set with augmentations (step 3) and low false positive rate on the clean test set without augmentations (step 2).

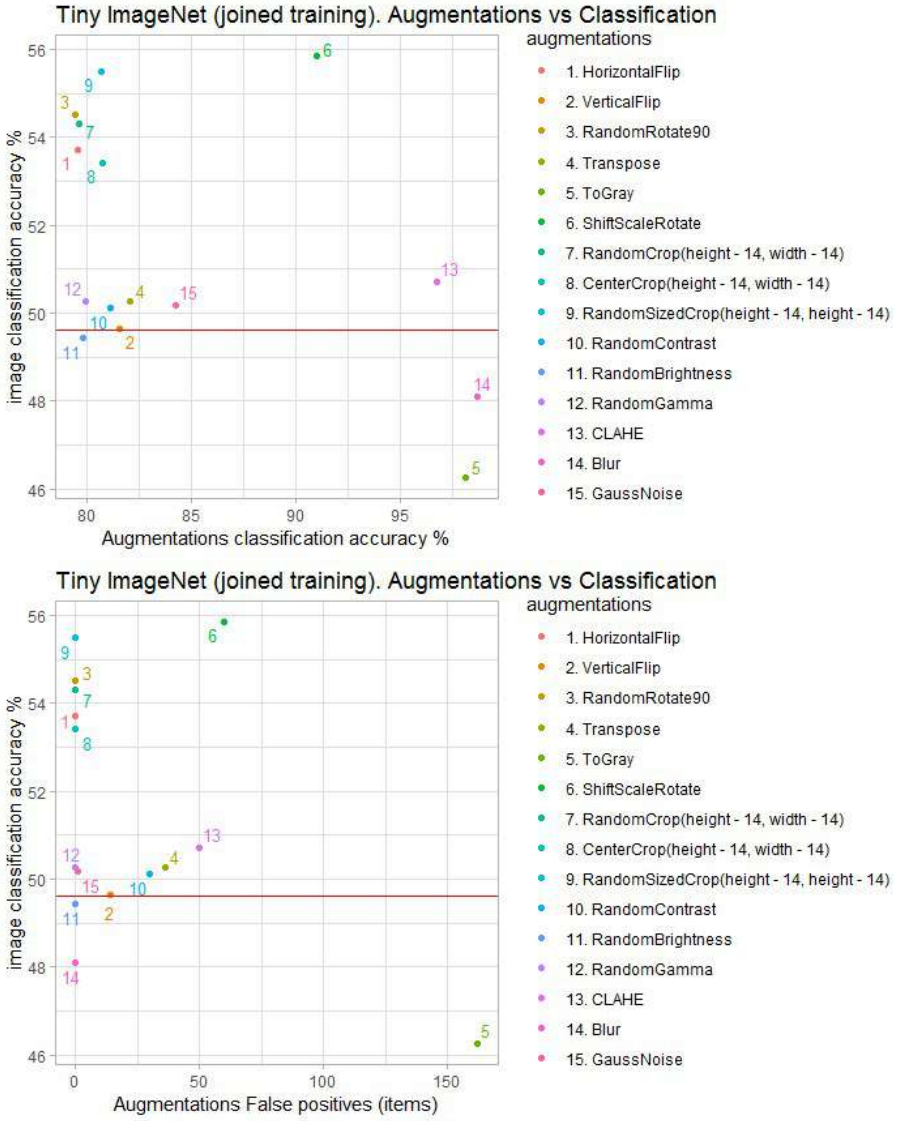

FIGURE 3.1: Image classification vs Augmentation false positives (step 2) and Augmentation classification (step 3). Red line denotes image classification accuracy without augmentations.

For example, figure (3.1) shows the described above metrics for the Tiny ImageNet dataset along with the image classification accuracy of every single augmentation. *Blur* is an example of non-safe transformation with the small false positive rate on the clean test set and very high augmentation classification accuracy on the augmented test set, so it is easy for the model to distinguish when *Blur* was applied. On the other hand, *HorizontalFlip* is a good example of safe augmentation with both low false positive rate and augmentation classification accuracy. It is clearly shown than *HorizontalFlip* significantly increases the image classification accuracy on the Tiny ImageNet dataset, whereas *Blur* decreases it.

Note that we can only evaluate one transformation at a time, i.e., we cannot take into account the impact of different augmentations on each other. So the combination of safe augmentations is not necessarily safe. For example, given a dataset of $32x32$ images and image classification task, we found that *RandomCrop(height - 7, width - 7)* and *CenterCrop(height - 7, width - 7)* are safe transformations. But, when these two functions are applied together, it is likely that such a combination is no longer safe because the augmented image could be too small.

3.1.1 Joint learning

To learn augmentations that not only do not break the data distribution but also improve original task accuracy, we decided to train the multi-label classification problem (step 1) in a joint learning setup. To do that, we propose to modify the architecture of the original models in the following way.

For the image classification task (3.2), the new loss L_{total} is calculated as sum of the augmentation classification loss L_{augm} and the image classification loss L_{class}.

$$L_{total} = L_{augm} + L_{class} \tag{3.2}$$

where L_{class} is the cross-entropy loss.

$$L_{class}(x, y) = -\log\left(\frac{\exp(x[y])}{\sum_j \exp(x[j])}\right) = -x[y] + \log\left(\sum_j \exp(x[j])\right) \tag{3.3}$$

where $i = 0, \ldots, C-1$ and C - number of classes.

For the semantic image segmentation task (3.3) the new loss L_{total} is calculated in a similar way as sum of the augmentation classification loss L_{augm} and the semantic segmentation loss L_{segm}.

$$L_{total} = L_{augm} + L_{segm} \tag{3.4}$$

where L_{segm} is the cross-entropy loss same as L_{class}, but here x is a two-dimensional predicted mask, y is the two-dimensional target mask, and the goal is to label every pixel in x with the correct class.

For each augmented batch of images we calculate defined above L_{total} loss and then perform gradient updates. Joint learning setup helps to find better set of safe augmentations that can be used to improve performance of the

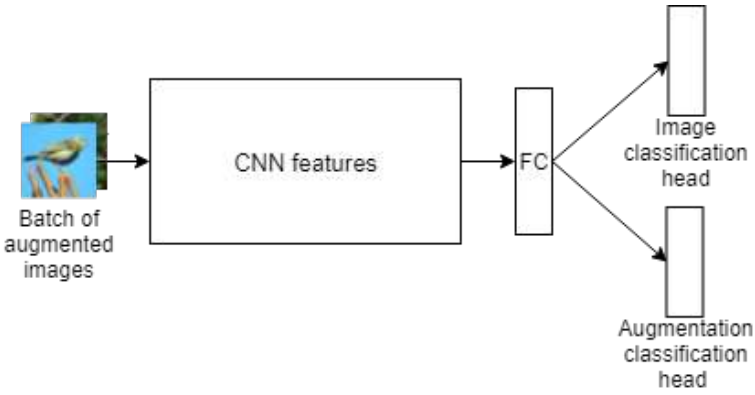

FIGURE 3.2: Example of joint learning setup for the image classification task.

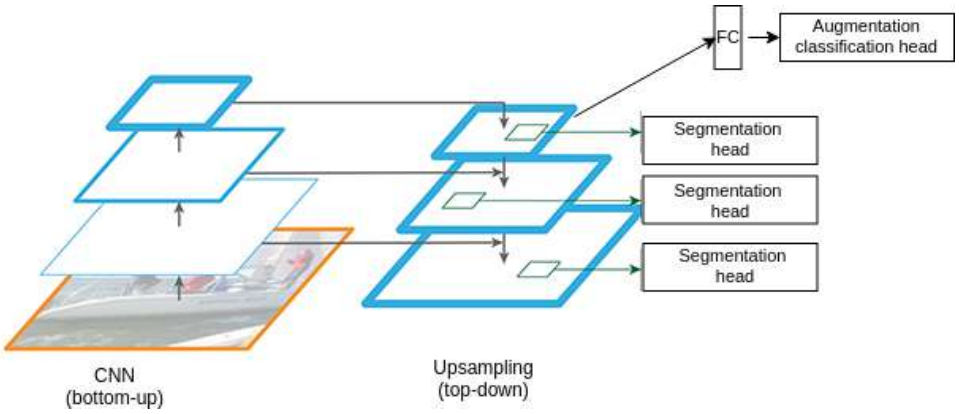

FIGURE 3.3: Example of joint learning setup for the semantic segmentation task using Feature Pyramid Network (Lin et al., 2016).

original task. All the results presented in the Experiments section (5) were obtained using this setup.

3.2 Safe augmentations usage

Having learned the set of safe augmentations S for a given dataset and task, we propose to use them to improve the accuracy of the original task in the following way.

Step 1. Train the original task using a set of all augmentations A. For every batch of images, a random subset $a \subset A$ of fixed size is applied. In our experiments, we used a subset of the size 3. Each transformation is applied with the fixed probability $p = 0.5$.

Step 2. Fine-tune the already pre-trained model on all augmentations using a subset of safe augmentations S. For every batch of images, a random subset $s \subset S$ of fixed size is applied. The subset size and probability of applying transformations are the same as in the previous step.

We showed that training the model using described two-step pipeline leads to the highest accuracy comparing to other alternative pipelines (5). We believe that using all augmentations, including those that break the data distribution can force the model to learn more general features. Thus, they can be used for the model pre-training. After that, we need to fine-tune the model using safe augmentations for learning dataset-specific features.

FIGURE 3.4: Example of using random subsets $s \subset S$ of safe augmentations on images from CIFAR-100 dataset. Each transformation is applied with the probability $p = 0.5$. Each crops is of size $25x25$ pixels.
Set 1: HorizontalFlip, RandomContrast, RandomSizedCrop.
Set 2: RandomCrop, RandomContrast, RandomRotate90.
Set 3: RandomSizedCrop, RandomContrast, RandomCrop.
Set 4: RandomContrast, RandomBrightness, RandomGamma.

Chapter 4

Data

Here we give an overview of the datasets that we used to conduct experiments and evaluate our approach. We used high-quality midsize datasets that are the most popular among researchers. Below we provide a summary of each of them.

4.1 Image classification

For the image classification task, we used four different datasets.

CIFAR-10. The CIFAR-10 dataset consists of 60.000 32x32 color images in 10 classes, with 6.000 images per class. There are 50.000 training images and 10.000 test images. The ten different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks (Krizhevsky, 2009).

CIFAR-100. The CIFAR-100 dataset is almost the same as CIFAR-10, except it contains 100 different classes with 600 images per class (500 training and 100 testing images). The 100 classes are grouped into 20 superclasses. For example, superclass *flowers* contains the following classes: *orchids, poppies, roses, sunflowers, tulips* (Krizhevsky, 2009).

SVHN. The Street View House Numbers (SVHN) dataset is a real-world image dataset of color digits obtained from house numbers in Google Street View images. It consists of 73257 digits for training and 26032 digits for testing, each image is resized to a fixed resolution of 32x32 pixels.(Netzer et al., 2011)

Tiny ImageNet. The Tiny ImageNet dataset contains 200 classes. Each class has 500 training images and 50 validation images. All images are of size 64x64 pixels. This dataset was provided by Fei-Fei Li, Andrej Karpathy, and Justin Johnson as a part of the cs231n course at Stanford University.

4.2 Image segmentation

For the semantic image segmentation task, we used a relatively new dataset called **Cityscapes**. It is a large-scale dataset collected from street views from 50 different cities. Cityscapes is used to train and test approaches for pixel-level and instance-level semantic labeling. It has 5000 fully annotated images of size 1024x2048 pixels. It contains 30 different classes, including *person, car, road, building, sky, etc.* (Cordts et al., 2016)

Chapter 5

Experiments

5.1 Augmentations

We used a main set A of 15 common augmentations from Albumentations library that provides fast image transforms based on highly-optimized OpenCV library (Buslaev et al., 2018).

Transformation functions
HorizontalFlip
VerticalFlip
RandomRotate90
Transpose
ToGray
ShiftScaleRotate
RandomCrop
CenterCrop
RandomSizedCrop
RandomContrast
RandomBrightness
RandomGamma
CLAHE
Blur
GaussNoise

TABLE 5.1: Set of all augmentations A used in experiments.

5.2 Image classification

To evaluate our approach on image classification task, we chose to use the Densely Connected Convolutional Network (DenseNet), which is a modern architecture that requires fewer parameters and achieves decent results on many benchmark datasets. (Huang et al., 2016).

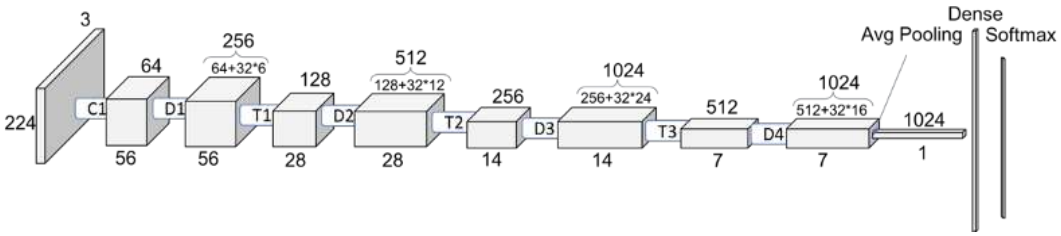

FIGURE 5.1: DenseNet-121 architecture for 224x224 images. Dx: Dense Block x. Tx: Transition Block x (Ruiz, 2018).

5.2.1 Training details

All training was performed on a single GTX 1080 GPU. We perform the phase of learning safe augmentations using the joint learning setup (3.2). As the main model we use DenseNet-121 (5.1). Both image classification and augmentation classification tasks are trained from scratch using stochastic gradient descent (SGD) optimizer. The initial learning rate is set to 10^{-1}, momentum to 0.9 and weight decay to 0.0005. All models are trained for 500 epochs. We use reducing learning rate on the plateau by 0.1 with 10 epochs patience and early stopping with 20 epochs patience.

5.2.2 Results

Here we provide all quantitative results obtained on four different datasets. In order to compare our approach, for every dataset we train the image classification task using different augmentation techniques, namely without augmentations, using random subset of size 3 of all augmentations and using random subset of size 3 of found safe augmentations. We also use defined above sets of transformations to fine-tune the model pre-trained on all transformations. All augmentations in our experiments with the image classification task is applied with the probability $p = 0.5$.

CIFAR-10	CIFAR-100	Tiny ImageNet	SVHN
RandomCrop	RandomCrop	RandomCrop	RandomContrast
CenterCrop	CenterCrop	CenterCrop	RandomBrightness
RandomSizedCrop	RandomSizedCrop	RandomSizedCrop	RandomGamma
HorizontalFlip	HorizontalFlip	HorizontalFlip	GaussNoise
RandomRotate90	RandomRotate90	RandomRotate90	
RandomBrightness	RandomBrightness	RandomBrightness	
RandomGamma	RandomGamma	RandomGamma	
	RandomContrast		

TABLE 5.2: Safe augmentations found using joint learning setup for CIFAR-10, CIFAR-100, Tiny ImageNet and SVHN datasets.

CIFAR-10 and CIFAR-100

We used all crops described in the table (5.1) with the size $(height - 7, width - 7)$, where *height* and *width* represent the size of the image. We trained both augmentation classification task and image classification task using the batch of size 256.

Our approach found almost the same set of safe augmentations for CIFAR-10 and CIFAR-100 datasets, which makes sense because these datasets are very similar (5.2). We manually defined another set called safe augmentations v2 by removing *RandomCrop* and *CenterCrop*. Since our approach can only evaluate transformations independently, it cannot determine the possibly harmful influence of one transformation on another. In fact, safe augmentations v2 achieved the best results in terms of image classification accuracy both on CIFAR-10 and CIFAR-100. Nevertheless, our proposed pipeline of fine-tuning the model using safe augmentations showed great accuracy and outperformed most of the tested augmentation techniques (5.3, 5.4).

	Test loss	Test accuracy (%)
Trained without Augmentations	0.698039	79.551
Trained using Safe Augmentations	0.468199	85.224
Trained using All Augmentations	0.362399	88.252
Fine-tuned with All Augmentations	0.342988	88.681
Fine-tuned with Safe Augmentations	0.358708	88.388
Fine-tuned with Safe Augmentations V2	0.356512	**88.798**

TABLE 5.3: Image classification results on CIFAR-10 dataset with different augmentation techniques. All fine-tuning experiments were performed using the model pre-trained on all augmentations.

	Test loss	Test accuracy (%)
Trained without Augmentations	1.968367	53.418
Trained using Safe Augmentations	1.417157	63.125
Trained using All Augmentations	1.284332	65.488
Fine-tuned with All Augmentations	1.294823	65.839
Fine-tuned with Safe Augmentations	1.282066	65.937
Fine-tuned with Safe Augmentations V2	1.268305	**66.015**

TABLE 5.4: Image classification results on CIFAR-100 dataset with different augmentation techniques. All fine-tuning experiments were performed using the model pre-trained on all augmentations.

SVHN

Same as for CIFAR-10 and CIFAR-100, we used all crops described in the table (5.1) with the size $(height - 7, width - 7)$, where *height* and *width* represent the size of the image. We trained both augmentation classification task and image classification task using the batch of size 256.

Our method found 4 safe augmentations (5.2), which all are color-based transformations. Using the found set of transformations for fine-tuning the model pre-trained on the set of all transformations lead to the highest test accuracy (5.5).

	Test loss	Test accuracy (%)
Trained without Augmentations	0.175104	95.032
Trained using Safe Augmentations	0.181426	95.004
Trained using All Augmentations	0.167121	95.620
Fine-tuned with All Augmentations	0.165516	95.501
Fine-tuned with Safe Augmentations	0.145694	**96.229**

TABLE 5.5: Image classification results on SVHN dataset with different augmentation techniques. All fine-tuning experiments were performed using the model pre-trained on all augmentations.

Tiny ImageNet

	Top-1 (%)	Top-3 (%)	Top-5 (%)
Trained without Augmentations	49.609	68.046	74.804
Trained using Safe Augmentations	57.148	74.726	80.957
Trained using All Augmentations	58.925	76.416	82.382
Fine-tuned with All Augmentations	58.896	76.025	82.060
Fine-tuned with Safe Augmentations	58.867	76.386	82.236
Fine-tuned with Safe Augmentations V2	**59.160**	**76.425**	**82.656**

TABLE 5.6: Image classification results on Tiny ImageNet test set with different augmentation techniques. Top-1, top-3 and top-5 denote the corresponding test classification accuracy. All fine-tuning experiments were performed using the model pre-trained on all augmentations.

We used all crops described in the table (5.1) with the size $(height - 14, width - 14)$, where *height* and *width* represent the size of the image. We trained both augmentation classification task and image classification task using the batch of size 256. Similarly to CIFAR-10 and CIFAR-100, we created a set of safe augmentations v2 by excluding *RandomCrop* and *CenterCrop* from the originally found set of safe augmentations.

Our method found the set of 7 safe augmentations. The proposed pipeline of fine-tuning models with safe transformations outperforms most of the tested techniques (5.6). Note that set of manually created safe augmentations v2 achieves the best accuracy.

5.3 Image segmentation

To evaluate our approach on semantic image segmentation task, we chose to use the Feature Pyramid Network (FPN), which is a fast and accurate architecture that generates multi-scale feature maps for better object detection in different scales (Lin et al., 2016).

5.3.1 Training details

All training was performed on a single GTX 1080 GPU. We perform the phase of learning safe augmentations using the joint learning setup (3.3). As the main model, we use FPN with the DenseNet-121 backbone. Both image classification and augmentation classification tasks are trained from scratch using Adam optimizer. The initial learning rate is set to 10^{-4}. All models are trained for 200 epochs. We use reducing learning rate on plateau by 0.5 with 7 epochs patience and early stopping with 15 epochs patience.

5.3.2 Results

Here we provide all quantitative results. To compare our approach, we train the semantic image segmentation task using different augmentation techniques, namely without augmentations, using a random subset of size 3 of all augmentations and using a random subset of size 3 of found safe augmentations. We also use defined above sets of transformations to fine-tune the model pre-trained on all transformations. All augmentations in our experiments with the semantic segmentation task are applied with the probability $p = 0.5$.

Cityscapes

Transformations
HorizontalFlip
RandomBrightness
RandomGamma
Transpose

TABLE 5.7: Safe augmentations found using joint learning setup for the Cityscapes dataset.

We trained both augmentation classification task and image classification task using the batch of size 16. We rescaled every image to 256x256 pixels due to the limited training resources. We used same set of all augmentations (5.1), except for *RandomSizedCrop*, because we already have *RandomCrop* and do *Resize* for every image. We also changed the size of the crops to $(512, 512)$.

Our method found 4 safe augmentations (5.7). Using the found set of transformations for fine-tuning the model pre-trained on the set of all transformations lead to the highest validation performance in terms of intersection over union metric(5.8).

	Validation loss	Validation IoU (%)
Trained without Augmentations	0.32913	45.340
Trained using default Augmentation	30.389	51.112
Trained using Safe Augmentations	0.29692	51.584
Trained using All Augmentations	0.25249	59.413
Fine-tuned with All Augmentations	0.28026	60.373
Fine-tuned with Safe Augmentations	0.27121	**62.091**

TABLE 5.8: Semantic image segmentation results on Cityscapes dataset with different augmentation techniques. All fine-tuning experiments were performed using the model pre-trained on all augmentations.

Chapter 6

Conclusion

We present a novel approach for learning task-specific augmentations that do not break the data distribution and can be safely used for training of the original task. Additionally, we propose a training pipeline using safe augmentations. Our methods show great results in many different datasets on image classification task and semantic image segmentation task. We compare our approach with a diverse set of possible augmentation techniques.

Future work. Our method of choosing safe augmentations is still a semi-automated one because we must manually divide all augmentations into two groups using the provided metrics. We are currently working on making this process fully automatic. We would also like to explore our approach in a transfer learning setup, to see if we can train one model and use it to get safe augmentations from many similar datasets.

References

Antoniou, A., A. Storkey, and H. Edwards (2017). "Data Augmentation Generative Adversarial Networks". In: *ArXiv e-prints*. arXiv: 1711.04340 [stat.ML].

Bowles, Christopher et al. (2018). "GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks". In: *arXiv e-prints*, arXiv:1810.10863, arXiv:1810.10863. arXiv: 1810.10863 [cs.CV].

Buslaev, A. et al. (2018). "Albumentations: fast and flexible image augmentations". In: *ArXiv e-prints*. arXiv: 1809.06839 [cs.CV].

Cordts, Marius et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *arXiv e-prints*, arXiv:1604.01685, arXiv:1604.01685. arXiv: 1604.01685 [cs.CV].

Cubuk, E. D. et al. (2018). "AutoAugment: Learning Augmentation Policies from Data". In: *ArXiv e-prints*. arXiv: 1805.09501 [cs.CV].

DeVries, T. and G. W. Taylor (2017). "Dataset Augmentation in Feature Space". In: *ArXiv e-prints*. arXiv: 1702.05538 [stat.ML].

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Hawkins, Douglas M. (2004). "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1. PMID: 14741005, pp. 1–12. DOI: 10.1021/ci0342472. eprint: https://doi.org/10.1021/ci0342472. URL: https://doi.org/10.1021/ci0342472.

Hernández-García, A. and P. König (2018). "Data augmentation instead of explicit regularization". In: *ArXiv e-prints*. arXiv: 1806.03852 [cs.CV].

Hinton, Geoffrey E. et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv e-prints*, arXiv:1207.0580, arXiv:1207.0580. arXiv: 1207.0580 [cs.NE].

Huang, Gao et al. (2016). "Densely Connected Convolutional Networks". In: *arXiv e-prints*, arXiv:1608.06993, arXiv:1608.06993. arXiv: 1608.06993 [cs.CV].

Krizhevsky, Alex (2009). "Learning multiple layers of features from tiny images". In: URL: http://www.cs.toronto.edu/~kriz/cifar.html.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Krogh, Anders and John A. Hertz (1992). "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems 4*. Ed. by J. E. Moody, S. J. Hanson, and R. P. Lippmann. Morgan-Kaufmann, pp. 950–957. URL: http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf.

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database". In: URL: http://yann.lecun.com/exdb/mnist/.

Lemley, J., S. Bazrafkan, and P. Corcoran (2017). "Smart Augmentation - Learning an Optimal Data Augmentation Strategy". In: *ArXiv e-prints*. arXiv: 1703.08383 [cs.AI].

Lin, Tsung-Yi et al. (2016). "Feature Pyramid Networks for Object Detection". In: *arXiv e-prints*, arXiv:1612.03144, arXiv:1612.03144. arXiv: 1612.03144 [cs.CV].

Netzer, Yuval et al. (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Parashar, Amrita (2017). "IMPORTANCE OF COMPUTER VISION FOR HUMAN LIFE". In: *International Journal of Advanced Research* 5.3, pp. 2396–2399. DOI: 10.21474/ijar01/3769. URL: https://doi.org/10.21474%2Fijar01%2F3769.

Perez, Luis and Jason Wang (2017). "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *ArXiv e-prints*, arXiv:1712.04621, arXiv:1712.04621. arXiv: 1712.04621 [cs.CV].

Prechelt, Lutz (1998). "Early Stopping-But When?" In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. London, UK, UK: Springer-Verlag, pp. 55–69. ISBN: 3-540-65311-2. URL: http://dl.acm.org/citation.cfm?id=645754.668392.

Ratner, A. J. et al. (2017). "Learning to Compose Domain-Specific Transformations for Data Augmentation". In: *ArXiv e-prints*. arXiv: 1709.01643 [stat.ML].

Ruiz, Pablo (2018). "Understanding and visualizing DenseNets". In: URL: http://www.pabloruizruiz10.com/resources/CNNs/DenseNets.pdf.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Taylor, Luke and Geoff Nitschke (2017). "Improving Deep Learning using Generic Data Augmentation". In: *arXiv e-prints*, arXiv:1708.06020, arXiv:1708.06020. arXiv: 1708.06020 [cs.LG].

Tran, T. et al. (2017). "A Bayesian Data Augmentation Approach for Learning Deep Models". In: *ArXiv e-prints*. arXiv: 1710.10564 [cs.CV].

Wong, Sebastien C. et al. (2016). "Understanding data augmentation for classification: when to warp?" In: *arXiv e-prints*, arXiv:1609.08764, arXiv:1609.08764. arXiv: 1609.08764 [cs.CV].

Wu, Eric et al. (2018). "Conditional Infilling GANs for Data Augmentation in Mammogram Classification". In: *arXiv e-prints*, arXiv:1807.08093, arXiv:1807.08093. arXiv: 1807.08093 [cs.CV].

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

Zahangir Alom, Md et al. (2018). "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches". In: *arXiv e-prints*, arXiv:1803.01164, arXiv:1803.01164. arXiv: 1803.01164 [cs.CV].

Zhang, Guodong et al. (2018a). "Three Mechanisms of Weight Decay Regularization". In: *arXiv e-prints*, arXiv:1810.12281, arXiv:1810.12281. arXiv: 1810.12281 [cs.LG].

Zhang, X. et al. (2018b). "DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification". In: *arXiv e-prints*. arXiv: 1809.00981 [cs.CV].

Zhu, Jun-Yan et al. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*.