Few-Shot Learning

Toward Few-Shot Learning and Data Augmentation

Bachelor Thesis

by

Milad Navidizadeh
2953248

in fulfillment of requirements for degree
Bachelor of Science (B.Sc.)

submitted to
Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science III
Bachelor Thesis for Information Systems and Artificial Intelligence

in degree course
Computer Science (B.Sc.)

First Supervisor: Prof. Dr. Stefan Wrobel
University of Bonn

Second Supervisor: Prof. Dr. Christian Bauckhage
University of Bonn

Bonn, March 21, 2020

Abstract

Here should be short summary of what we doing in thesis

> » *If it takes 200 years to achieve artificial intelligence and then finally there's a textbook that explains how it's done, the hardest part of that textbook to write will be the part that explains why people didn't think of it 200 years ago.* « John McCarthy

Acknowledgement

First of all, I wish to thank all the people whose assistance was a milestone in the completion of this work. These may include Mohsen Seifi, Pouya Azimi Garakani, and Sally Chau who improved this work with their proofreading and great hints.

A special thanks goes out to Prof. Dr. rer. nat. Wrobel, for providing the possibility of this work and willingness to lead this thesis as first supervisor. I would like to thank Prof. Dr.-Ing. Bauckhage for his willingness to assess my bachelor thesis as second supervisor.

I wish to express my deepest gratitude to Mr. Florian Seiffarth for being my mentor during this work. His guidance and patience during our meetings, made this thesis happen and boosted the quality of it.

Last but not least, I would like to express heartfelt thanks to my family who is endlessly supporting me especially another half of me, Jannet Bejia Ferjani, for her accompanying in good and bad days and my parents who provide me the possibility of education and grant me the best gift in my life that they believed in me.

Contents

1 Motivation

Nowadays machine learning and deep learning have become a distinguished approach for visual recognition tasks and have achieved great success in this process. However, they require a large amount of labeled data to learn. Providing this amount of labeled data does not only bring much effort along but also occupies a huge amount of storage. In contrast, we as humans are very good in visual recognition so that, we are able to learn with one or few[1] examples. Imagine a child who is able to recognize a lion in a picture after learning from, few example, how lions look like. We want to simulate and apply this ability to deep learning such that, few example are sufficient to achieve a desirable accuracy. In this bachelor thesis, we focus on few-shot learning in deep learning. We aim at learning and training a model for small set of labeled examples. We enlarge our dataset by generating artificial examples based on the examples at hand. This technique known as data augmentation. These artificial labeled examples help to increase accuracy and prevent overfitting. Data augmentation is our focus in this work to achieve few-shot learning and prevent overfitting. In this following, we will introduce several well-known techniques of data augmentation. The first purpose will be to discover if and how far data augmentation can improve the learning process and accuracy. The second step will be to compare their accuracy. In the end, we aim discovering potential combination of different approaches and techniques of data augmentation to increase accuracy and reduce the error-rate. We will focus on visual recognition tasks and their classification. Additionally, we will concentrate on the implementation of various techniques of data augmentation for convolutional neural networks.

[1] This is known as few-shot learning in deep learning and represents the scenario when there are one or few instances of each class in training-set to learn.

2 Introduction

Neural networks can possibly contain multiple non-linear layers which makes them very expressive models that are able to learn complex relationships between input and output parameter. Even with limited input data, neural networks discover and learn many relations from the data. However, sometimes the discovered and learned relations do not exist or consist of redundant information and relations. Redundant relations potentially arise from data-noise or lack of data-generalization. Non-existent relations can potentially emerge from lack of enough data. These phenomena are known as *overfitting* in deep learning. In other words, learning with few labeled examples or noisy data causes overfitting. Overfitting causes low accuracy and high error-rates. Hence, our goal is to propagate from artificial labeled examples from a few given examples to prevent overfitting and reduce the error-rate and increase the accuracy.

As we mentioned acquiring a huge labeled dataset is expensive and seeks much effort and time. Therefore, we aim to generate artificial examples from few labeled examples. In other words, we augment our which is known as data augmentation. There are a few well-known techniques for data augmentation. We will introduce the following techniques by implementing their strategy and comparing their efficiency and capability:

- **Image Translations** 3.1.1
- **Elastic Distortions** 3.1.2
- **Stroke Warping** 3.1.3
- **Bayesian Approach** 3.2

3 Data Augmentation

In this chapter, we will introduce a few noteworthy related works in this field which mainly consist of two approaches and several techniques. In what follows, we solely focus on image data. While some techniques might be applicable to other types of data as well, we shall explore them when used on image datasets.

As the name clears itself, we are looking for enlarging datasets artificially and generate synthetic data based on the obtainable few samples to increase the accuracy of the prediction.

3.1 Label Preserving Transformations

When training neural networks, label preserving transformations are a commonly used approach with classes of techniques for enlarging datasets by generating generic data. The advantage of using this approach and its techniques, when available, is twofold. The first benefit is the low space complexity of these methods since one does not necessarily need to save the generated data on storage. Also, these transformations are usually of relatively smaller time complexity compared to other approaches, which makes them desirable in many instances in practice.

As the name may suggest, the ultimate goal when using these techniques is to generate synthetic data points after applying a set of suitable transformations to a real data point.

3.1.1 Image Translations

Image translation is one of the simplest and meanwhile applicable well-known technique of the label preserving transformations approach. As the research by Krizhevsky et al. [KSH17] proposes, we extract image translations and their horizontal reflections to generate synthetic data and augment the dataset. Image translation consists of extracting patches that are smaller in size than the original

image. Given a single channel (grayscale) $n \times n$ image and a translation patch of the size m for some $m < n$, one can produce $(n-m+1) \times (n-m+1)$ synthetic instances of the datapoint. Also, taking the horizontal reflections of each newly generated data point into account, one can enlarge the dataset by a factor of two, therefore finally the dataset can be enlarged by factor of:

$$2 \times (n-m+1) \times (n-m+1) \tag{3.1}$$

Figure 1 is an illustration of the translations and horizontal reflections of a 4×4 image, using all possible patches of the size 3×3, which results in eight new data points.

At test (prediction) time, the method extracts the patches with the same size ($m < n$), however this time the patches will be extracted from the four corners and the center of the test image. The network predicts on these five patches and their horizontal reflections (10 patches altogether). In the end, the average on the softmax layer will be determined by the final prediction.

3.1.2 Elastic Distortions

Another well-known technique for data augmentation is elastic distortion. Quite similar to image translation, the ultimate goal is to generate synthetic dataset from a single data point. However, instead of taking patches which are smaller than the original image, the synthetically produced data points are of the same size as the original image as proposed by [SSP03] . This is done by moving pixels and modifying pixel intensity according to both their former and new position. To this end, a few interpolation schemes such as the nearest neighbor, bicubic, spline, and bilinear are available. Due to their practical effectiveness and simplicity, we use the bilinear interpolation scheme in this work, which we discuss in detail below.

Let $\alpha \in \mathbb{R}$, and let $\Delta x(x,y) = \alpha x$ and $\Delta y(x,y) = \alpha y$ denote the horizontal and the vertical displacement of a point (x,y) of an image respectively. Since the scaling parameter α could take a non-integer value, the interpolation task is necessary for adjusting the intensity of pixels. Utilizing the bilinear interpolation, we adjust the intensity of a shifted pixel according to its former location and intensity of its neighbors therein. A formal description of the procedure is as follows. In what follow we will show and summarize the process formally.

Definition 1: Given the pixel p'. We want to displace it with $\Delta x(x,y) = \alpha x$. Let $\Delta y(x,y) = \alpha y$ and $p_{(x,y)}, p_{(x+1,y)}, p_{(x,y-1)}, p_{(x+1,y-1)}$ the neighbors (on origin square)

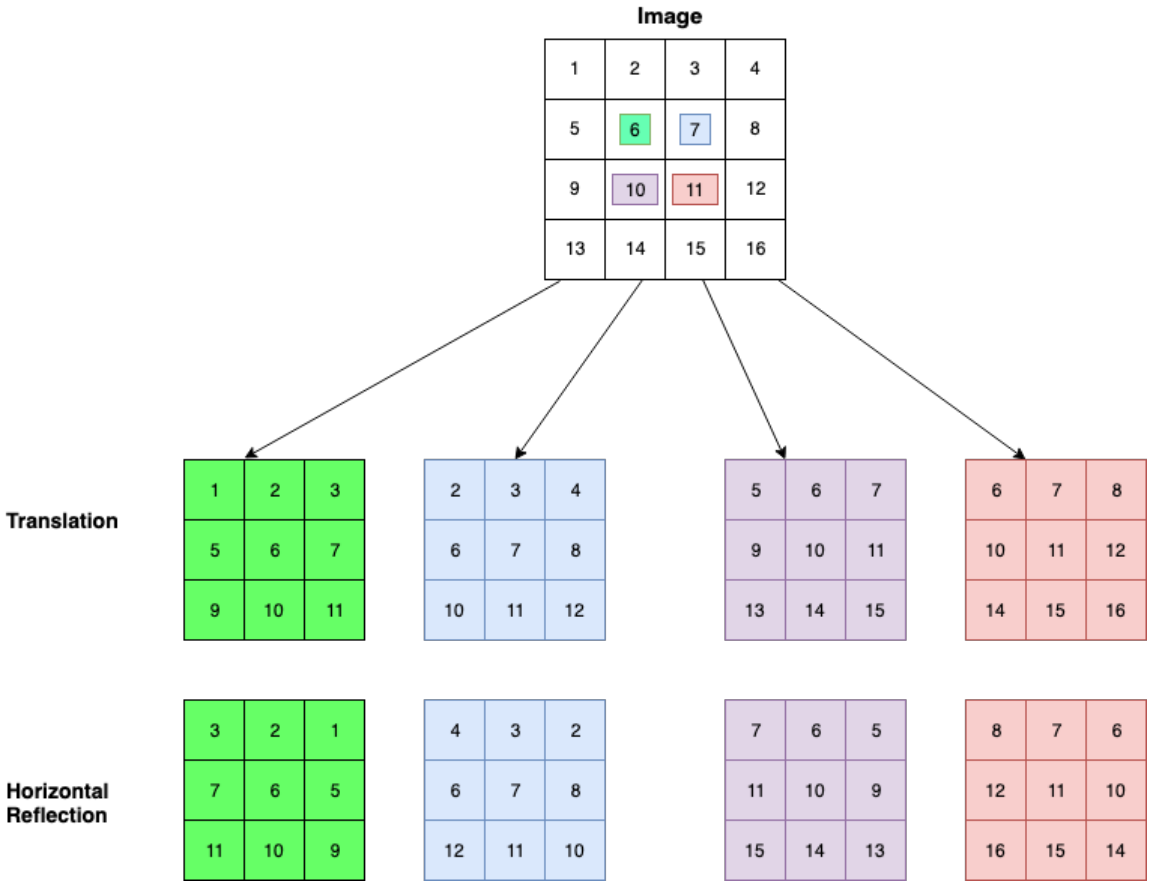

Figure 1: *An example of single channel image with size of* 4×4 *with its translations with size of* 3×3 *patches and their horizontal reflections. The numbers determinate the pixels intensity*

of p' in the new location after displacement and $I(p)$ shows the intensity of pixel p. Then the vertical interpolation yields:

$$V_1 = I(p_{(x,y)}) + \Big(\Delta x(p', p_{(x,y)}) \times I(p_{(x+1,y)})\Big)$$
$$V_2 = I(p_{(x,y-1)}) + \Big(\Delta x(p', p_{(x,y-1)}) \times I(p_{(x+1,y-1)})\Big)$$

And then the horizontal interpolation yields a new intensity for pixel p' after displacement:

$$I(p') = V_1 + \Big(\Delta y(p', p_{(x,y-1)}) \times V_2\Big)$$

In practice, usually, one picks the scaling parameter α from the interval $[-1, 1]$ uniformly at random. At the final stage of the procedure, the fields $\Delta x(x, y)$ and $\Delta y(x, y)$ are convolved with a Gaussian filter with a standard deviation of σ, whose value depends on the size and the entropy of the image. Observe that this technique is called the elastic distortion mainly because the procedure described above results in an elastically deformed instance of the original image.

Similar to image translation, at test (prediction) time, the image will be augmented by a factor of ten with elastic distortion. Finally and in test and prediction time, average on softmax layer for this enlarged ten images determines the prediction (label).

3.1.3 Stroke Warping

Similar to the previously introduced techniques, stroke warping uses predetermined families of transformations. In other words, we enlarge our dataset artificially with the aid of well-known classical computer vision transformations. This technique not withstanding of non-heavy complexity accomplished desirable results even on medical purposes [Bai+19]. The ideas of this technique raised from Tangent Dist [Sim+92] and Tangent Prop [SLD93] works.

In this technique, we perform small changes in images to augment our data. That means we augment our images with skewing, rotating, and shearing (scaling) [YLW97]. Similar to image translation and elastic distortion the augmentation will be started before the training phase and the training will be performed on enlarged data. Figure 2 represents each mentioned transformation to make them visually understandable.

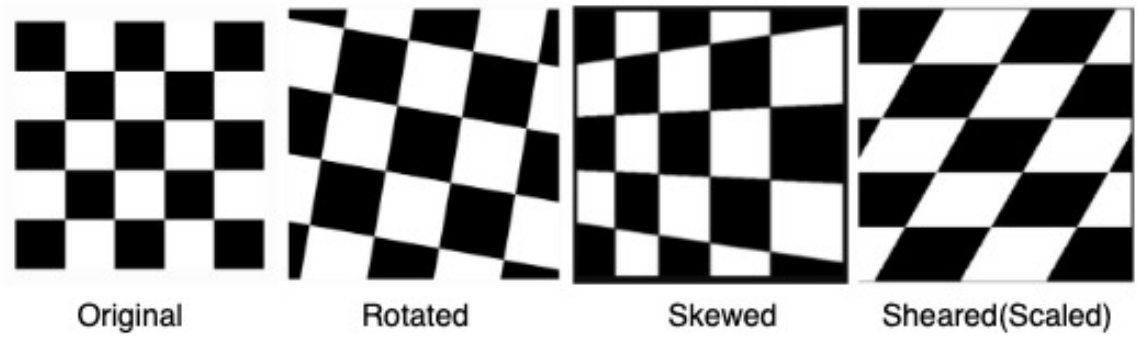

Figure 2: *An example of rotation, skew, and shear (scale) transformations for stroke warping [Blo17].*

At test (prediction) time, the image will be enlarged by factor of ten and prediction will be performed on averaging of softmax layer of ten images, similar to the previous techniques.

3.2 Bayesian Approach

The astute reader should have noticed that, although quite different, all the introduced techniques so far share a fundamental aspect. Precisely speaking, in all these techniques, one obtains new training samples by applying a set of predefined random transformations on the annotated training data, and the augmentation procedure ends before the training phase starts. This widely used process is called the poor man's data augmentation (PMDA) [Tan91]. However, to the best of our knowledge, this is not the furthest that one can go. Indeed, the fact that neural networks are generally capable of learning complicated patterns and nonlinear relationships in images suggests that they should also be able to learn latent variables so that they can enhance the data augmentation process dynamically. In direct contrast to PMDA, in Bayesian data augmentation, the training set evolves dynamically and in an iterative fashion during the training phase, which could considerably enhance the generalization ability of a neural network. This approach uses a number of techniques to achieve this matter as such as maximum a posterior probability (MAP) [GPS89] in Bayesian statistics and Generative Adversarial Networks (GANs). Before diving directly into technical details about the Bayesian data augmentation, we briefly introduce generative models. First, we present the Generative Adversarial Networks (GANs) proposed initially by Goodfellow et al. [Goo+14]. In what follow, we will introduce GANs and in the bayesian approach. Here, we will explain how this approach uses the main idea of GANs and extend it to improve data augmentation.

3.2.1 Generative Adversarial Networks (GANs)

Generally, a Generative Adversarial Net is made up of two parts:

- **Generator:** As the name suggests, the generator in a GAN is responsible for generating new data and, at the same time, learns how to generate more plausible data.
- **Discriminator:** The discriminator of a GAN learns to distinguish real data from synthetic data generated by the generator in the GAN. This helps the generator to generate more plausible data.

Indeed, one can view the interaction between the generator and the discriminator of a GAN as a minimax two-player game. Throughout the game, the generator's goal is to trick the discriminator with synthetically generated data. The discriminator's goal, however, is to detect the model data and to distinguish it from the real data. In the beginning, the discriminator may easily detect the model data. However, after some iterations, the generator becomes sufficiently intelligent so that it can produce synthetic data that is indistinguishable from the real data. Notice that, this means, eventually, the discriminator's accuracy of classifying the real and fake data reduces to 50%, where the classification is effectively random (predicts between two classes real or fake), and therefore the game is over. In some instances, competition in this game enhances the generator's performance up to a point where detecting the model data is impossible even for human eyes. Technically speaking, a GAN's task is to replicate a probability distribution.

The generator will be fed by random input[1] to generate synthetic data. For this matter, they use a loss function to measure the distance between the distribution of the generator's data and the distribution of the real data. Goodfellow et al. [Goo+14] suggest minimax loss in their work. The minimax loss is defined as:

$$Minimax\ Loss = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \tag{3.2}$$

where $D(x)$ denotes the discriminator's estimate of the probability that real data instance x is real and $D(G(z))$ denotes the discriminator's estimate of the probability that a fake instance is real. It is clear that the Generator tries to minimize the (3.2) and the discriminator tries to maximize it. Figure 3 shows a basic architecture of a GAN.

[1] noise

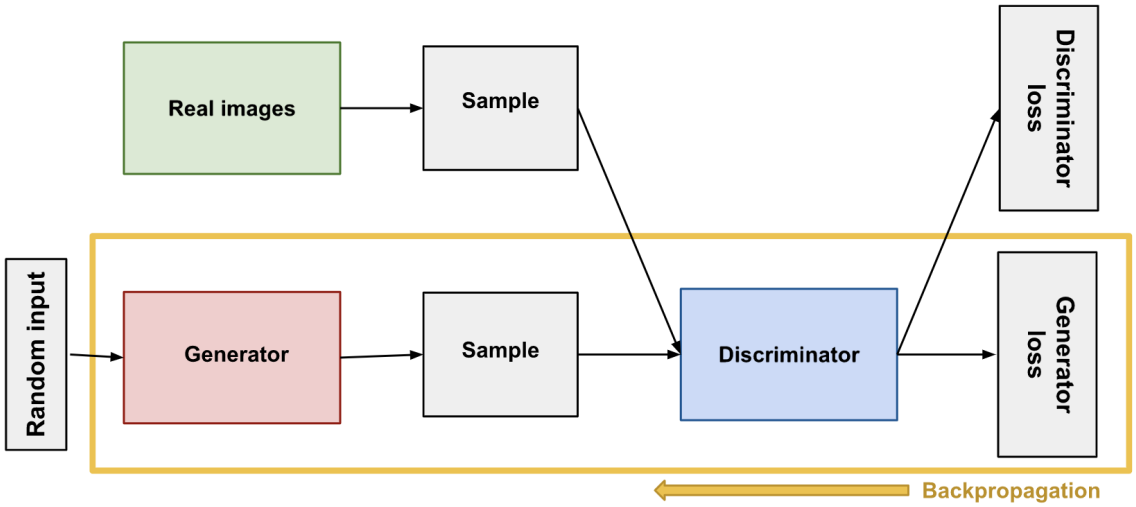

Figure 3: *GAN architecture [Com].*

After a brief introduction of GANs we focus now on the **Bayesian Approach**. One noteworthy work for data augmentation with the aid of the Bayesian model proposed by Toan Tran et al. [Tra+17]. In this approach, our deep learning model tries to estimate the distributions of labeled data. with the estimated distributions and in order to optimize the latent variable used for data augmentation. The approach uses the GANs architecture skeleton to generate synthetic data with the difference that the optimization of latent variables are derived from the Bayesian model. The main idea for data augmentation using latent variables was proposed by the statistical learning community [TW87]. Nevertheless applying the idea directly into deep learning seeks a massive computational effort. Therefore, the estimation of the variable distribution becomes crucial. To be more precise, the approach uses a novel Bayesian data augmentation algorithm, called Generalized Monte Carlo Expectation Maximization (GMCEM). This algorithm augments training data and mutually optimizes the network parameters. The algorithm successively generates synthetic data and uses the Monte Carlo Algorithm to estimate the expected value of the network parameters given the previous estimate instead of calculating the loss function. After the estimation of the expected value, the parameter values will be updated with stochastic gradient descent (SGD). In the end, the algorithm and approach are turned in to reality with the aid of GANs. The proposed GAN consists of one generator and two discriminators. As we discussed in Section 3.2.1 the generator is responsible to generate synthetic data and one of our discriminators distinguishes fake and real data. However, the second discriminator differentiates between the classes of data. Figure 4 represents the utilized network architecture in this approach visually. This proposed architecture is almost similar to the

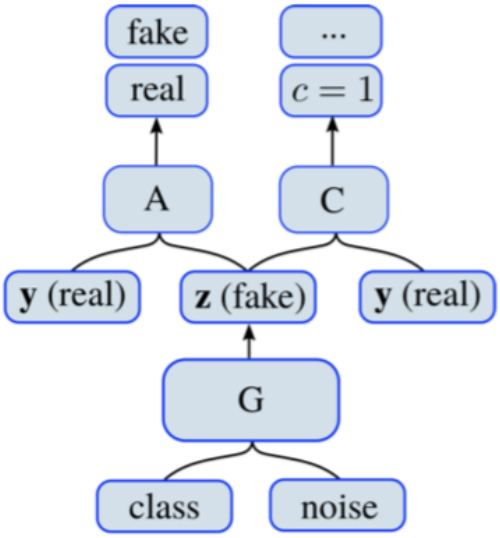

Figure 4: *The network architecture of Bayesian data augmentation approach [Tra+17]. G: Generator, A: Authenticator, C: Classifier.*

Auxiliary Classifier GANs (AC-GANS) [A O16]. Nevertheless, in the AC-GANS the discriminator is responsible for both classifications real-or-fake data and data labels (classes) while in our network utilizes two distinct discriminators for this matter.

In the following, we will formally explain the utilized algorithm from the Toan Tran et al. work [Tra+17]. The goal is to estimate the parameters of the neural networks using labeled data. The training process is defined by the following optimization problem:

$$\theta^* = \arg\max \log p(\theta|\mathbf{y}), \tag{3.3}$$

where the training set is denoted as $\mathcal{Y} = \{\mathbf{y}_n\}_{n=1}^N$ with $y = (t, x)$ and $t \in \{1, ..., K\}$ (Classes-Set) and data samples $\mathbb{R}^D$ and θ are denoted as model (network) parameters. The observed posterior defined as:

$$p(\theta|\mathbf{y}) = p(\theta|t, \mathbf{x}) \propto p(t|\mathbf{x}, \theta)p(\mathbf{x}|\theta)p(\theta). \tag{3.4}$$

Now if we assume that the data samples $\mathcal{Y}$ are conditionally independent, we can define the following loss function which maximize the function given in (3.3).

$$\log p(\theta|\mathbf{y}) \approx \log p(\theta) + \frac{1}{N} \sum_{l}^{N} \left(\log p\left(t_n|\mathbf{x}_n, \theta\right) + \log p\left(\mathbf{x}_n|\theta\right)\right) \tag{3.5}$$

where $p(\theta)$ denotes a prior on the distribution of the deep learning model parameters, $p(t_n|x_n,\theta)$ represents the conditional likelihood of label t_n, and $p(x_n|\theta)$ is the likelihood of the data x.

After estimation and optimization of θ on our training set, we generate synthetic data from y using the latent variable z. Therefore, the augmented $p(\theta|y,z)$ can be estimated. The latent variable z, similar to y, is defined as $z = (t^\alpha, x^\alpha)$ where $t^\alpha \in \{1,...,K\}$ denotes the associated label and $x^\alpha \in \mathbb{R}^D$ is a synthesized sample. In order to avoid a heavy and most likely infinite computation we will use the Generalized Monte Carlo EM Algorithm to estimate the expected value and maximize it instead of the Expectation-Maximization (EM) algorithm. Hence the augmented posterior $p(\theta|y,z)$ for the latent variable z is defined as follow:

$$p(\theta|\mathbf{y},\mathbf{z}) = \frac{p(\mathbf{y},\mathbf{z},\theta)}{p(\mathbf{y},\mathbf{z})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z}|\mathbf{y})p(\mathbf{y})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})}{p(\mathbf{z}|\mathbf{y})} \tag{3.6}$$

where the expectation step will be defined as:

$$p(\theta|\mathbf{y},\mathbf{z}) = \frac{p(\mathbf{y},\mathbf{z},\theta)}{p(\mathbf{y},\mathbf{z})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z}|\mathbf{y})p(\mathbf{y})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})}{p(\mathbf{z}|\mathbf{y})} \tag{3.7}$$

where $\mathbf{z}_m \sim p\left(\mathbf{z}|\mathbf{y},\theta^i\right)$, for $m \in \{1,\dots,M\}$. In (6), if the label t_m^a of the m^{th} synthesized sample $\mathbf{z_m}$ is known, then $\mathbf{x}_m^a$ can be sampled from the distribution $p\left(\mathbf{x}_m^a|\theta,\mathbf{y},t_m^a\right)$. Hence, the conditional distribution $p(\mathbf{z}|\mathbf{y},\theta)$ can be decomposed as:

$$p(\mathbf{z}|\mathbf{y},\theta) = p\left(t^a,\mathbf{x}^a|\mathbf{y},\theta\right) = p\left(t^a|\mathbf{x}^a,\mathbf{y},\theta\right)p\left(\mathbf{x}^a|\mathbf{y},\theta\right) \tag{3.8}$$

where $(t^a,\mathbf{x}^a)$ are conditionally independent of y given that all the information from the training set y is summarized in θ. This means that $p\left(t^a|\mathbf{x}^a,\mathbf{y},\theta\right) = p\left(t^a|\mathbf{x}^a,\theta\right)$, and $p\left(\mathbf{x}^a|\mathbf{y},\theta\right) = p\left(\mathbf{x}^a|\theta\right)$. Now, with respect to θ for the maximization step with concern of removing the independent terms for θ will derive the maximization of $\hat{Q}\left(\theta,\theta^i\right)$ as follow:

$$\begin{aligned}\hat{Q}(\theta,\theta^i) &= \log p(\theta) + \frac{1}{N}\sum_{n=1}^{N}(\log p(t_n|\mathbf{x}_n,\theta)+ \\ &\log p(\mathbf{x}_n|\theta)) + \frac{1}{M}\sum_{m=1}^{m}\log p(\mathbf{z}_m|\mathbf{y},\theta) \\ &= \\ &\log p(\theta) + \frac{1}{N}\sum_{n=1}^{N}(\log p(t_n|\mathbf{x}_n,\theta)+ \\ &\log p(\mathbf{x}_n|\theta)) + \frac{1}{M}\sum_{m=1}^{n}(\log p(t_m^a|\mathbf{x}_m^a,\theta) + \log p(\mathbf{x}_m^a|\theta))\end{aligned} \tag{3.9}$$

After all, we estimate the θ^{i+1} such that $\hat{Q}(\theta^{i+1},\theta^i) > \hat{Q}(\theta^i,\theta^i)$. To reduce the computation complexity, instead of using gradient descent, we employ stochastic gradient decent (SGD) for estimating θ^{i+1}. The iteration will be continued until $|\theta^{i+1} - \theta^i|$ becomes sufficiently small.

The above formal explanations and equations are derived from [Tra+17] and in some points are matched one-to-one.

4 Data Representation

In this section, we briefly introduce the datasets that we use to evaluate the goodness of each data augmentation approach in practice. The reason behind our choice of datasets here is twofold. First, they are well-known and widely used datasets. Many researchers already carried out numerous experiments on them and reported results, and therefore a lot of baseline results are available for the sake of comparison. Additionally, they are simple enough in their structure so that one can relatively effortlessly learn and append non-complex classifiers, which can classify them with desirable accuracy. The second reason is that all these there datasets have ten classes with small differences and they are balanced datasets and that means each class consists of the same number of samples. All of these help us to achieve a better and more realistic benchmark.

4.1 MNIST

The MNIST dataset (Modified National Institute of Standards and Technology) is a large handwritten digits dataset, provided by Yann Le Cun, derived from NITS Special Database 19 [STA].

The MNIST dataset consists of 60,000 train- and 10,000 test-images where each image is grayscale with 28×28 pixels. The dataset has ten classes where each class represents a digit from zero to nine. The data is fairly splitted between classes [LeC]. The MNIST is one of the most popular datasets for deep learning because of its low complexity and compatibility with almost all deep learning models. Hence, many papers attempted to reach a low error-rate on this dataset. One of them manages to reduce the error-rate on the MNIST by up to 0.23% [DS12]. You can find the information about the dataset in table 1. The figure 5 shows an example of the dataset.

Figure 5: *7 examples per class of MNIST dataset, merged in one image [Ath].*

4.2 Fashion-MNIST

The Fashion-MNIST is a dataset of Zalando's[1] article images provided by Han Xiao et al. [HV17] for benchmarking machine learning algorithms.

Pretty much similar to the MNIST dataset, the Fashion-MNIST dataset, consists of 60,000 train- and 10,000 test-images where each image is grayscale with 28×28 pixels. It has ten classes (T-Shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle Boot) and the data is fairly splitted between these classes. One of the best reported accuracy achieved 91.90% on this dataset with convolutional neural network[2]. You can find the information about the dataset in table 1. Figure 6 shows an example of the dataset.

4.3 CIFAR-10

The CIFAR-10 (Canadian Institute for Advanced Research) , collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton is a subset of 80 million tiny images dataset [Uni].

The dataset consists of 60,000 RGB with 32×32 pixels images, which are divided into 50,000 train and 10,000 test datasets. As the name makes it clear, the CIFAR-10 contains ten classes (plane, car, bird, cat, deer, dog, frog, horse, ship, truck) [Kri]. One of the lowest reported error-rates achieved 2.56% with a convolutional neural network [DT17]. You can find the information about the dataset in table 1. Figure 7 shows an example of the dataset.

[1] https://jobs.zalando.com/de/tech/
[2] https://github.com/zalandoresearch/fashion-mnist#benchmark/

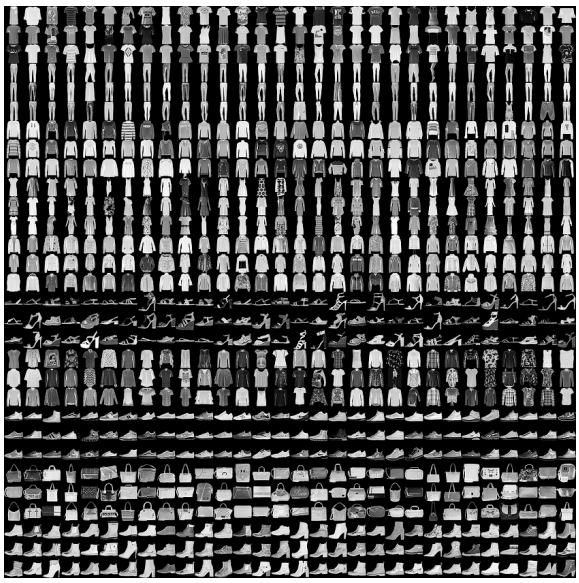

Figure 6: *Examples of Fashion-MNIST dataset, merged in one image.*

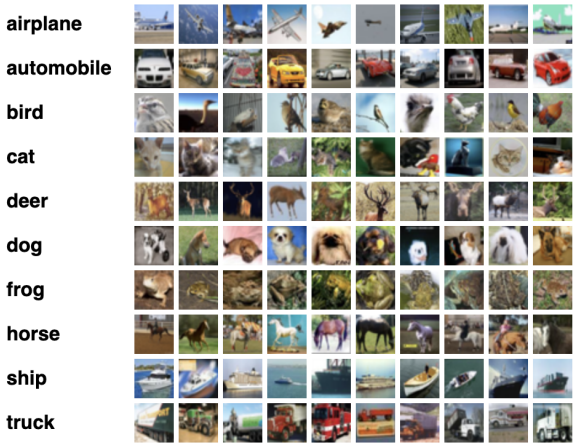

Figure 7: *10 examples per class of CIFAR-10 dataset, merged in one image [Kri].*

Table 1: *Structure of datasets.*

Dataset	NO. Classes	NO. Train	NO. Test	Size (pixel)	NO. Channel
MNIST	10	60,000	10,000	28×28	1 (Grayscale)
Fashion-MNIST	10	60,000	10,000	28×28	1 (Grayscale)
CIFAR-10	10	50,000	10,000	32×32	3 (RGB)

5 Experiments

In this chapter, we explain the manner of implementation of the classifiers, approaches, and techniques pragmatically. We introduce the architecture of the utilized classifiers[1] for each introduced datasets in chapter 4. After that, we explain the procedure of implementation of each introduced approach and technique in chapter 3.

5.1 CNNs Architecture

In this section, we will introduce our classifiers architecture. We picked our CNNs architecture from different sources regarding their low complexity and desirable accuracy. It might be that our chosen CNNs architecture does not provide the best-reported accuracy on datasets, as we use the same CNN-architecture for each dataset and we aim to compare various data augmentation approaches on each dataset, it would not affect our results.

5.1.1 MNIST

For the MNIST dataset, we have chosen a semi-simple CNN architecture from [rep17] with two convolutional layers and two fully connected layers. ReLU function is utilized as the activation function. For training and to calculate the loss function, **CrossEntropyLoss** and to optimize the network's parameters **Adam optimizer** [KB14] from the Pytorch have been chosen. The learning rate for Adam optimizer is set to 0.001. To reduce overfitting, a drop-out layer is placed at the end of the second convolutional layer. Figure 8 represents the explained CNN architecture visually.

[1] Convolutional Neural Networks (CNNs)

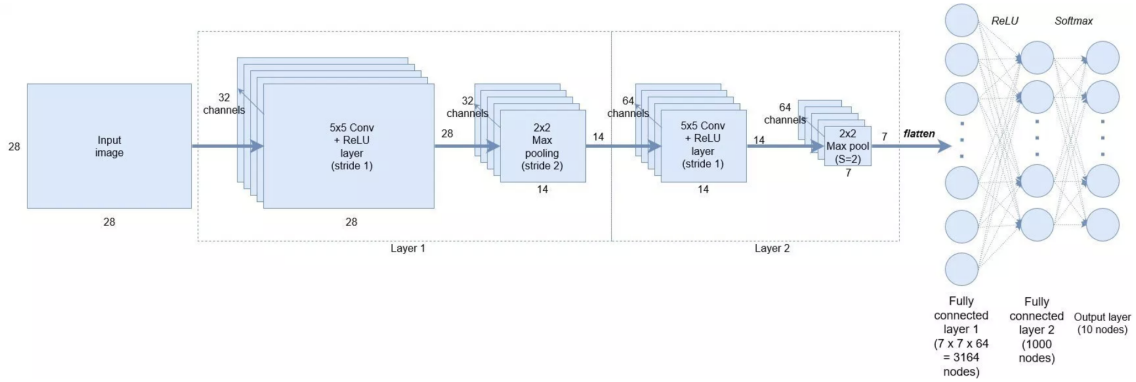

Figure 8: *CNN Architecture for training the MNIST dataset [17].*

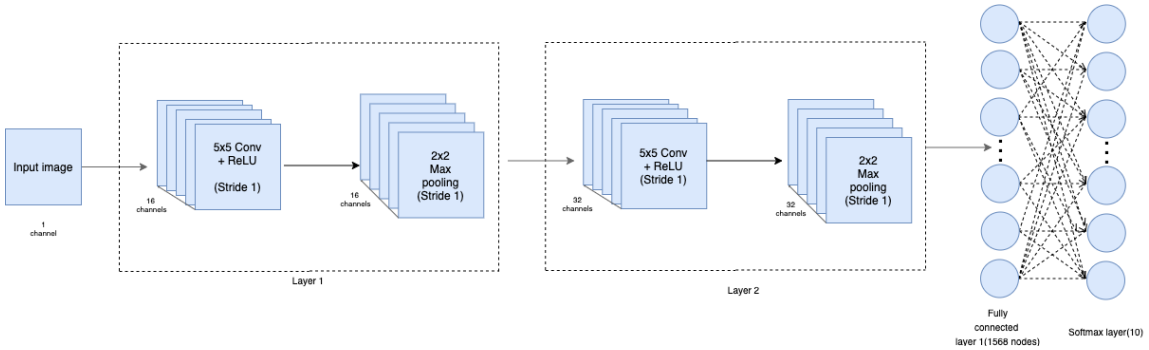

Figure 9: *CNN Architecture for training the Fashion-MNIST dataset.*

5.1.2 Fashion-MNIST

The Fashion-MNIST dataset uses CNN architecture derived from [**TODO**] with two convolutional layers and two fully connected layers similar to the MNIST dataset but with different of up- and downsampling and kernel sizes. ReLU function is utilized as the activation function. The learning rate for Adam optimizer is set to 0.001. Figure 9 represents the explained CNN architecture visually.

5.1.3 CIFAR-10

For the CIFAR-10 dataset, we have chosen a bit more complex CNN architecture from [Zhe18] with six convolutional layers and three fully connected layers. ReLU function is utilized as the activation function. For training and to calculate the loss function, **CrossEntropyLoss** and to optimize the network's parameters **Adam optimizer** similar to the two previous CNNs architectures have been chosen. The learning rate for Adam optimizer is set to 0.001. To reduce overfitting, a drop-out layer is placed at the end of the fourth convolutional layer. Figure 10 represents the explained CNN architecture visually.

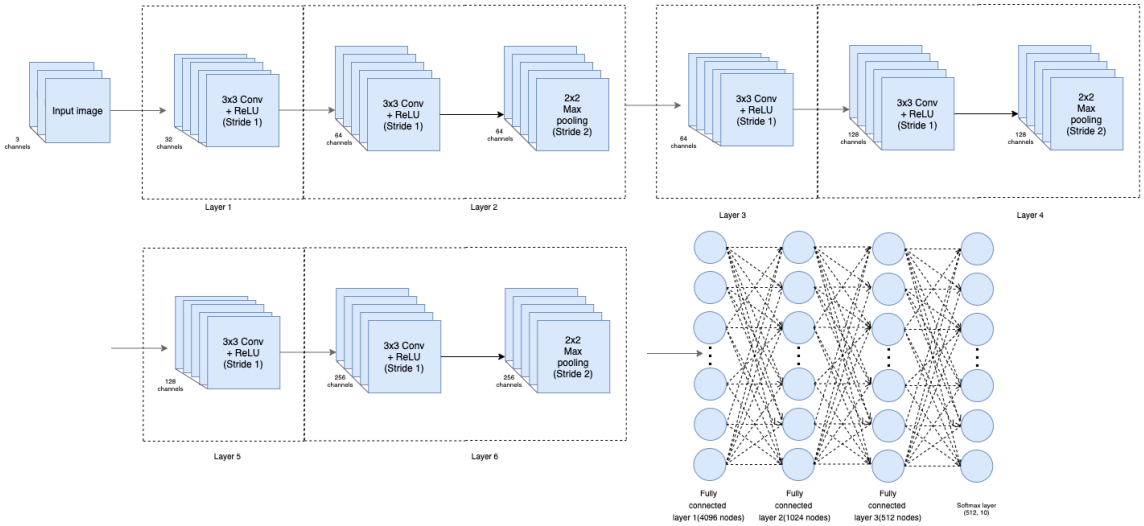

Figure 10: *CNN Architecture for training the CIFAR-10 dataset.*

5.2 Implementations

In what follow, we will briefly explain the manner of implementation of each approach and technique.

In order to put the idea of label preserving transformations and their techniques into practice, we utilized an external library in Python for data augmentation called Augmentor[2]. The main part (Learning) is carried out by a powerful library by Facebook called Pytorch[3].

5.2.1 Image Translations

To augment the data with image translation we picked patches smaller than original image size with a factor approximately 0.8. The patches are not only wide enough to not lose the 80% of the image but also allow to augment the dataset with factor close to 100. The equation (5.1) reproduces equation (3.1) with the MNIST dataset. After augmentation with translations and their horizontal reflections, we resized images with bilinear interpolation to the original image size in the dataset. This matter provides this possibility to be able to change the patches' size without putting any afford to modify CNN architecture.

$$2 \times (28 - (28 \times 0.8) + 1) \times (28 - (28 \times 0.8) + 1) \approx 2 \times 7 \times 7 \approx 100 \qquad (5.1)$$

[2] https://github.com/mdbloice/Augmentor
[3] https://pytorch.org/

For test and prediction, the image augmented with the same size of patches from corners and center with a factor of 10.

5.2.2 Elastic Distortions

To augment with this technique we picked 100 different and random patches from the image with a size of 8×8 and displacement performed in these patches with $\alpha = 8$ to augment the data with factor 100. After displacement, the patches have been smoothed with the Gaussian filter with $\sigma = 4$. For test and prediction, we used the same parameters but this time augmented with a factor of ten instead of 100.

5.2.3 Stroke Warping

For stroke warping, we augmented each image with skewing them horizontally and vertically with different magnitude, rotated them clockwise and counterclockwise with $-10° \leq \alpha \leq 10°$, and scaled with a factor s where $-6 \leq s \leq 6$ and sheared them. With different values for the mentioned parameters, we augmented the data with a factor of 100. Similar to other for test and prediction data augmented with a factor of ten.

5.2.4 Bayesian Approach

Here we used same code[4], implemented by Toan Tran et al. and introduced in their work [Tra+17] with small modification for using it on few-shot dataset.

[4] https://github.com/toantm/pytorch-bda

6 Result & Comparison

In this Chapter, first, we give the main results of our experiments using introduced approaches as well as introduced datasets. Following the main findings is a discussion of the advantages and drawbacks of each method and a comparison of its behaviour on different datasets. Besides providing great insights, this discussion serves as a preface and sets the stage for the novel ideas and approaches proposed in the next chapter 7.

6.1 Result

For the sake of an insightful and comprehensive comparison, we implement each technic or approach using different setups. Precisely speaking, we consider the accuracy of each method using the original dataset, that is, including all the test and train data points, as well as the accuracy using few-shot datasets both with and without augmentation. Our results also consist of a comparison between various k-shot[1] datasets:

$$k = \begin{cases} \{1,5,10\} & \text{if } dataset = MNIST \text{ or } Fashion-MNIST \\ \{10,20,30\} & \text{if } dataset = CIFAR-10 \end{cases} \tag{6.1}$$

Additionally, to compare the result in fair circumstances we augmented the data by a factor of 100 (100X) for the training and by a factor of ten (10X) with each augmentation technique. To provide a more realistic result, we used 10-fold cross-validation [Efr83]. That means for each k-shot learning, we derived 10 different and random k-shot datasets to calculate more realistic accuracy for each technique.

Figures 11, 12, and 13 represent the results and accuracy for MNIST, Fashion-MNIST, and CIFAR-10 datasets, respectively.

[1] k represents the number of samples in training set, existing in each class.

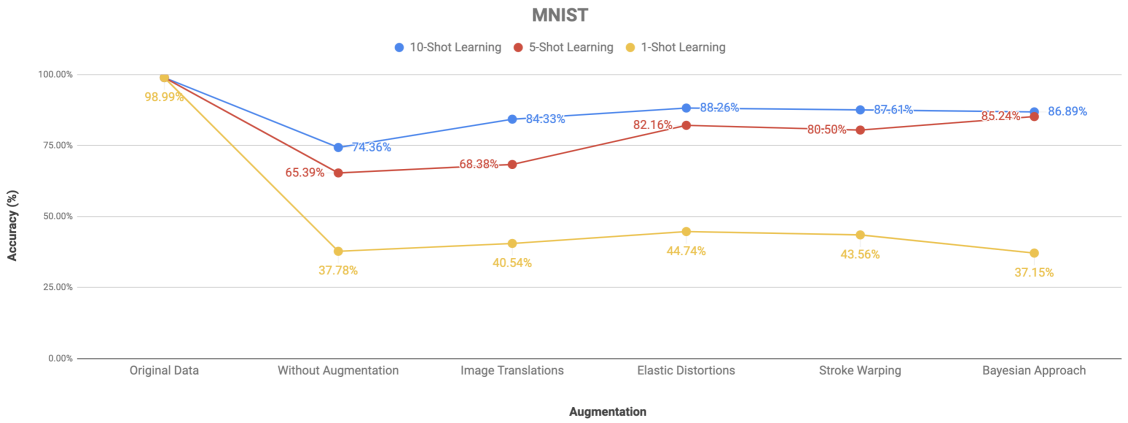

Figure 11: *Result of augmentation techniques on MNIST dataset.*

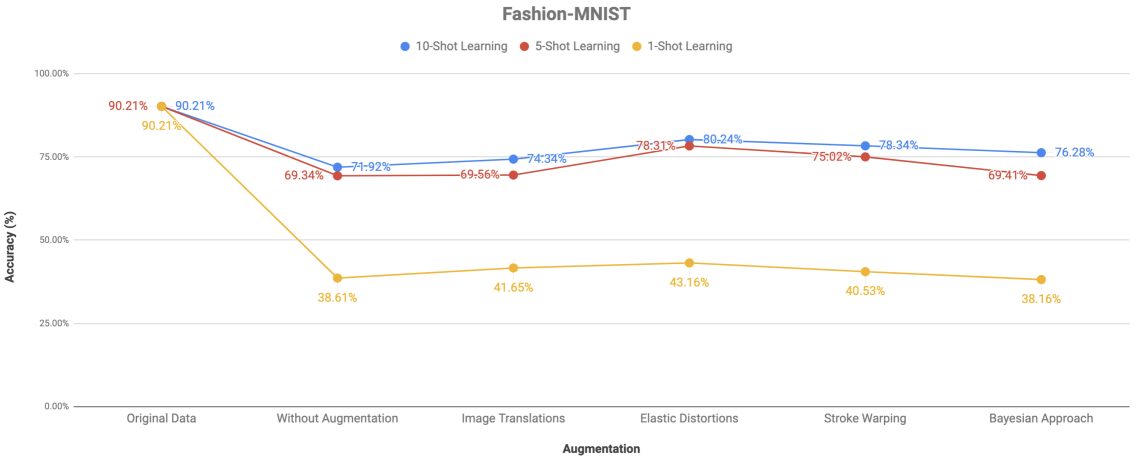

Figure 12: *Result of augmentation techniques on Fashion-MNIST dataset.*

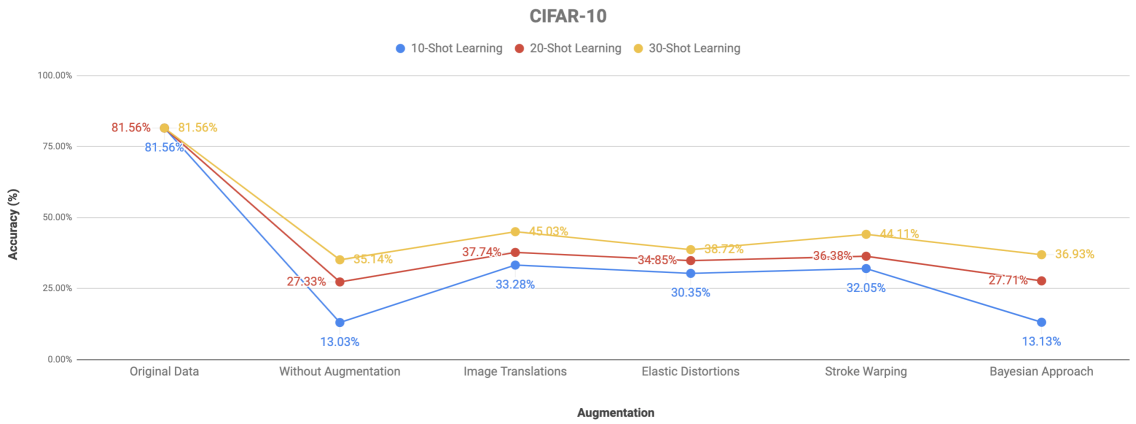

Figure 13: *Result of augmentation techniques on CIFAR-10 dataset.*

Table 2: *Accuracy of augmentation techniques on 10-shot datasets.*

Dataset\Augmentation	Without Augmentation	Image Translations	Elastic Distortions	Stroke Warping	Bayesian Approach
MNIST	74.36%	84.33%	88.26%	87.61%	86.89%
Fashion-MNIST	71.92%	74.34%	80.24%	78.34%	76.28%
CIFAR-10	13.03%	33.28%	30.35%	32.05%	34.13%

The first look at Figures 11 and 12 expose that there is a significant gap between the accuracy of 10 and 5-shot learning and 1-shot learning without augmentation. Besides, it shows that k-shot learning is not linearly correlated with the accuracy for each technique. As there is just one sample for each class, the CNN involves pretty soon with overfitting. Nevertheless, the accuracy is about 30% and that means that the prediction is not random even for 1-shot learning on MNIST and Fashion-MNIST.

Figure 13 displays the considerable gap between accuracy on the original data and few-shot learning on CIFAR-10. This matter exposes the major role of color (RGB images) in learning and accuracy. As it expected, this gap is not observable on MNIST and Fashion-MNIST.

6.2 Comparison

Table 2, derived from the Figures 11, 12, and 13 for 10-shot datasets, determines that the augmentation approaches and techniques behave the same on MNIST and Fashion-MNIST datasets. That means the improvement rate of accuracy is almost the same for all techniques. On the other hand, some techniques behave differently on CIFAR-10 in comparison to MNIST and Fashion-MNIST datasets. In what follows, we will discuss each technique behavior on the datasets and the underlying reasons.

6.2.1 Image Translations

As Table 2 represents firmly, image translation has the best accuracy on CIFAR-10 as it was expected. On the other hand, the accuracy of MNIST and Fashion-MNIST for image translation is not as good as the other techniques. It has the worse performance on these datasets. The reason is that image translation works with image patches smaller than the original image size. Hence, its performance is highly dependent on the datasets and even their classes. Put the matter in another way, image translation sometimes extracts patches from an image of class but the generated synthetic image gets the same as an image in another class. For example, Figure 14 shows a translation of digit nine from the MNIST dataset which looks

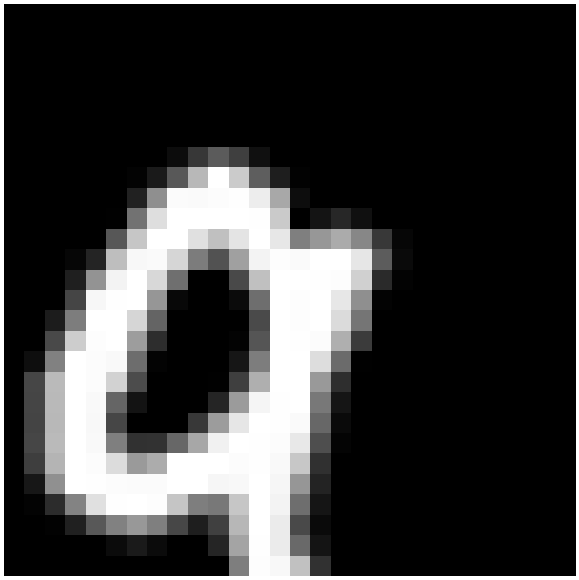

Figure 14: *Example of Image Translations on digit* 9 *which looks like* 0.

like a zero. On the Fashion-MNIST dataset some patches of a shirt image can get similar to a T-shirt image, etc. but the image translation on the CIFAR-10 dataset do not map the synthetic images to another class. It means a patch of e.g. an airplane will not become similar to another class in the CIFAR-10 dataset.

6.2.2 Elastic Distortions

As it represents in Table 2, elastic distortion has the best performance on MNIST and Fashion-MNIST. This behavior is what we expected from elastic distortion regarding its augmentation technique. For example, on the MNIST dataset, it simulates the deformation and flexion caused by handwriting on the digits and generates the synthetic data almost like the real ones in the original dataset. Figure 15 represents this matter visually. Fashion-MNIST obeys the same reason. In this case, the distortions simulate for example the folding on clothes and generate the synthetic data almost like the real other ones in the original dataset. On the other hand, the accuracy of elastic distortion on CIFAR-10 is not as good as the other techniques and the reason is trivial as well. Distortions, for example on cars or airplanes will not generate a total new meaningful image that are close to the other ones in the original dataset.

6.2.3 Stroke Warping

Table 2 demonstrates that stroke warping is the most robust technique between the other mentioned techniques in the label preserving transformation approach.

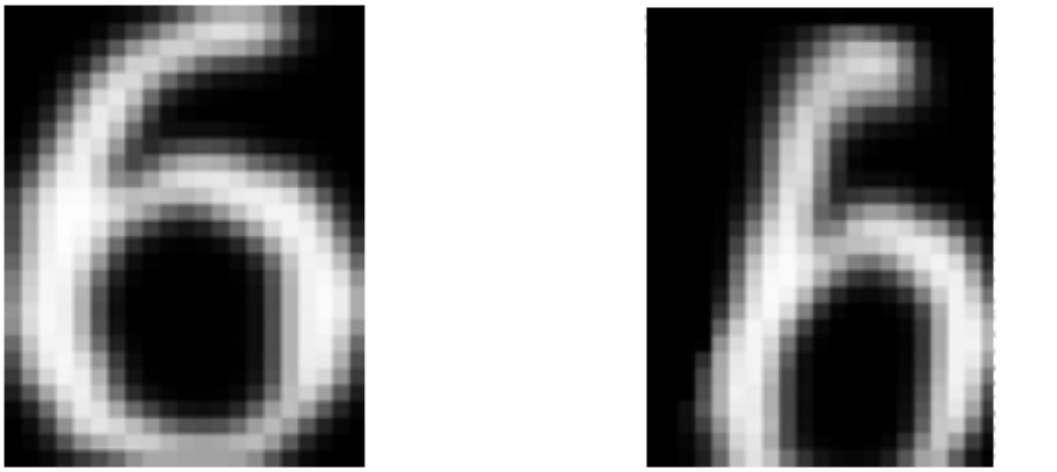

Figure 15: *Example of Elastic Distortions on one of the MNIST data.*

It means the accuracy improvement is almost the same for all three datasets. The reason is that Stroke Warping makes small changes in the whole image and only partially. These small changes help to improve the accuracy however the performance and accuracy of this technique is always ordinary on all datasets in compare to the other techniques.

6.2.4 Bayesian Approach

Table 2 shows that the bayesian approach is another robust approach and class of technique on different datasets. Since we learn during the augmentation how to generate the synthetic data and unlikely of the label preserving transformations the data will not be generated with some pre-defined transformations, this approach improves the accuracy with a factor independent to the dataset. However, this approach improves the accuracy relatively well and is robust on different datasets. In comparison to the other techniques requires a long time during generating and training phase. Additionally, since this approach works with statistical models and distributions (Max-A-Posterior probability, the accuracy stays almost constant in 1-shot learning. On the another hand, if the number of samples is increased, the accuracy increases more, compare to other techniques.

6.3 Conclusion

Between label preserving transformations techniques, image translation, and elastic distortion are highly dependent on datasets and even classes which means they can provide a desirable accuracy on some classes and some classes are not suitable for such an augmentation. This matter makes them strong with a considerable improvement in accuracy on compatible datasets and not so strong on incompatible ones. This based on the number of suitable classes for such an augmentation.

Stroke warping is the most robust technique of the label preserving transformations which means they improve the accuracy with almost the same factor independent on datasets. This technique is suitable for almost all datasets but the improvement of accuracy is always less than the best case of image translation and elastic distortion.

Bayesian approach is another class of techniques which provides robust result and accuracy on all datasets. The advantage is that it learns how to augment the data and generate the synthetic data based on each dataset and obtainable samples. On the another hand, high time-complexity in comparison with other techniques is a disadvantage of this kind of augmentation. Additionally, as it works with statistical models and distributions the number of samples at the beginning playing a significant role in the final result.

7 Contribution Of Work

In this chapter, we will introduce three new techniques of data augmentation, developed in this work. We studied some of the existing techniques and experimented pragmatically on 3 different datasets and compared them comprehensively in previous chapters. Based on all these studies and the aid and inspiration of other existing works and researches, we aim to propose new techniques for data augmentation to rectify the disadvantage of existing ones to reach better results and accuracy. In what follows, we focus on each approach (label preserving transformations and bayesian approach) separately and propose an improvement for each approach and their techniques.

7.1 Ensemble Learning & Label Preserving Transformations

As we precisely cleared in the previous chapter 6 that label preserving transformations techniques are highly dependent on datasets or even on each class (at least image translation and elastic distortion). This matter makes them perform completely differently on datasets specifically in detail in each class. Put the matter in another way, they can have highly good accuracy and prediction in one class, while having extremely accuracy and prediction in another class from the same dataset. Figure 16 represents an example of this matter of predicted class and the actual class of the 10-shot learning on the Fashion-MNIST dataset for different Label Preserving Transformations techniques.

As the astute readers most likely can guess in this technique we will propose a combination of image translation, elastic distortion, and stroke warping. However, the combination will be accompanied by learning from the performance of each technique in each class and try to choose the best technique for each class. In this technique and for such learning we inspired from a well-know approach called Ensemble Learning introduced by Robi Polikar [**TODO**]. This technique begins with the augmentations and learning for each three label preserving transformations

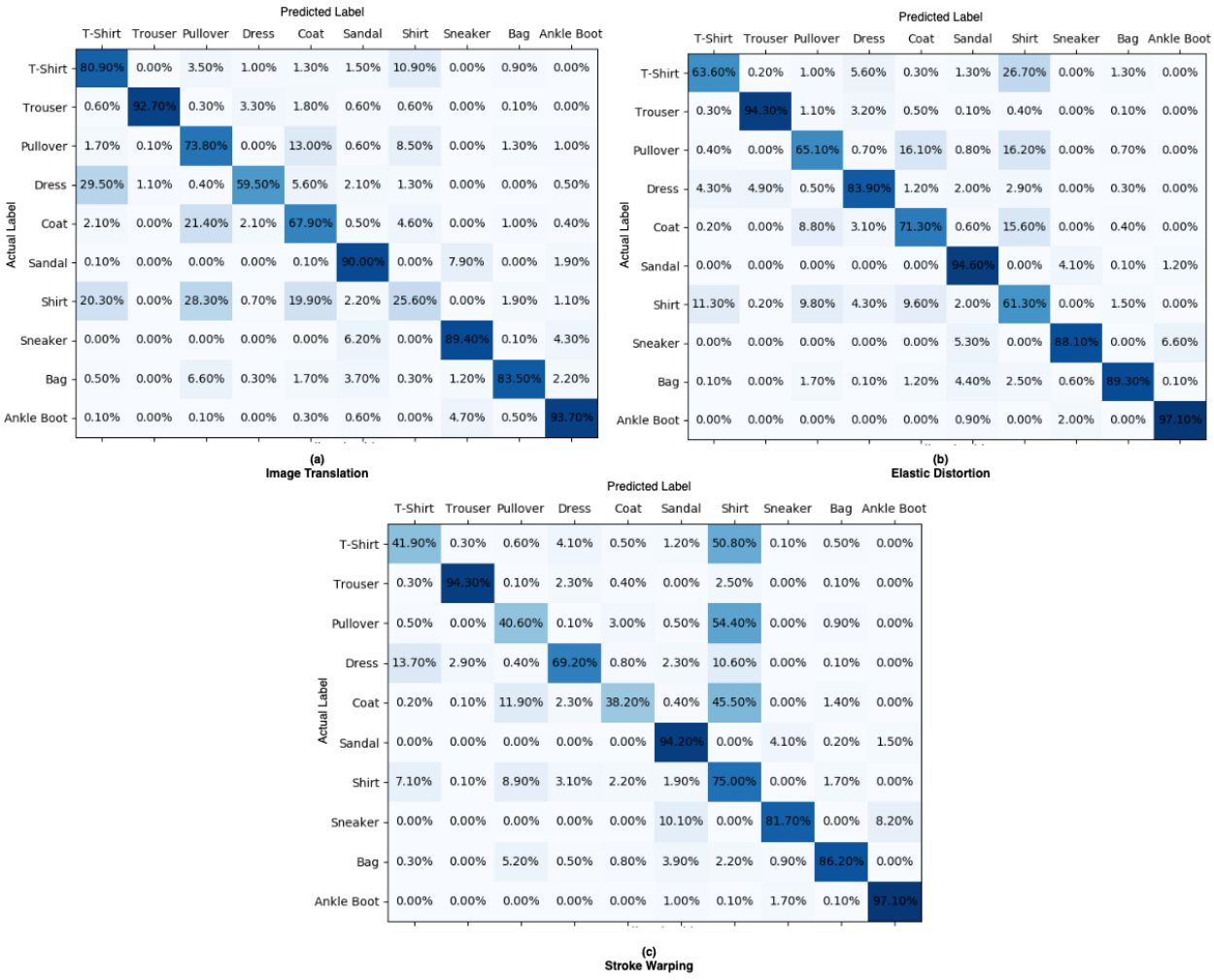

Figure 16: *Prediction accuracy of each class on Fashion-MNIST for Label Preserving Transformations techniques.*

techniques (image translation, elastic distortion, and stroke warping) separately. After training and with testing dataset a heatmap (10×10 prediction matrix) for accuracy on each class will be generated for each technique as like as Figure 16. These matrices expose the performance and accuracy of each augmentation technique for each class. Based on these matrices, a matrix will be generated which represents the probability of correct prediction for each class with each augmentation technique. In the end, we augment again our few-shot dataset, but this time with the highest correct prediction probability technique for each class and train the CNN with this augmented dataset. This is how we augment each class with the best techniques and improve the whole accuracy on dataset. The following equation shows the manner of generation of correct prediction probability formally:

Where IT, ED, and SW be the 10×10 prediction matrices respectively for image translation, elastic distortion, and stroke warping and each element of them denoted by it_{ij}, ed_{ij}, and sw_{ij} respectively for $i, j \in \{1, 2, ..., 10\}$ and $i = j$, than correct prediction probability matrix denoted by CPP and each element by cpp_{kj} will be generated as follow:

$$cpp_{kj} = \begin{cases} \dfrac{it_{ij}}{it_{ij} + ed_{ij} + sw_{ij}} & \text{if } k = 1 \\ \dfrac{ed_{ij}}{it_{ij} + ed_{ij} + sw_{ij}} & \text{if } k = 2 \\ \dfrac{sw_{ij}}{it_{ij} + ed_{ij} + sw_{ij}} & \text{if } k = 3 \end{cases} \tag{7.1}$$

Where $j \in \{1, 2, ..., 10\}$, then augmentation technique for each class will be determined by:

$$Augmentation\ Technique_j = max(cpp_{kj}) \quad , \forall k \in \{1, 2, 3\} \tag{7.2}$$

At the end and in the test and prediction time, we augment the data by factor ten with all three techniques. Again and as same as each technique we average on the softmax layers for ten augmented images for each technique separately. If two or more (three) predicts the same class that would be the final prediction of our model. If each of the softmax layers predict different classes (labels) the final prediction would be the prediction of the softmax layer with maximum probability. If we come to the edge case that the softmax layers predict different classes with the exact same probability first we check if there is one predicted class which

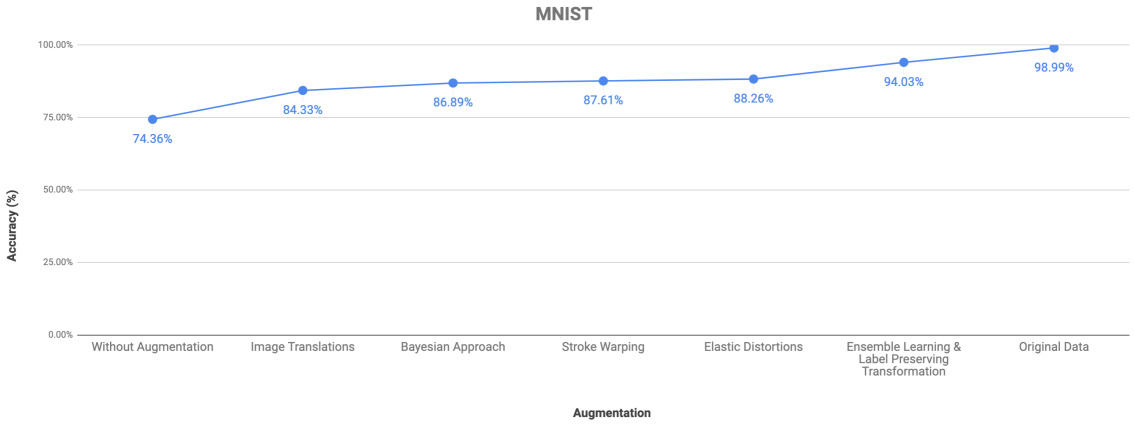

Figure 17: *Comparative result between Ensemble Learning & Label Preserving Transformations augmentation and other augmentation techniques on MNIST dataset.*

augmentation technique for prediction and augmentation technique for learning match. If there is just one prediction (class) that satisfies that case that class would be the final prediction. Otherwise, we augment again the data and repeat these steps until get the unique prediction.

Figures 17, 18, and 19 represent the results of this technique besides other introduced techniques in chapter 3 for datasets MNIST, Fashion-MNIST, and CIFAR-10 respectively. These results are proving that this technique does not only seem better theoretical and in words than other ones but also in application. Additionally, Figure 20 represent the prediction accuracy for each class for this technique. The comparison between this figure and figure 16 profs our statement and shows that accuracy for almost every class is better than the best of the previous techniques.

7.2 Color Randomization & Ensemble Learning

As the name clearly indicates, this technique tackles the datasets such as Cifar-10 with RGB images. As we have stated in Chapter (6) there is a significant gap between the accuracy of few-shot learning and learning on original data for the Cifar-10 dataset. Further investigation on this dataset made it obvious that the colors have been highly instrumental in learning. Figure 21 presents the detailed accuracy of classes' prediction on the original data of Cifar-10. As it demonstrates, for instance, many airplane samples mistakenly have been classified as a ship even on the original dataset. It is trivial that CNN learns the colors and classifies the samples based on that since there is no shape similarity between airplane and ship

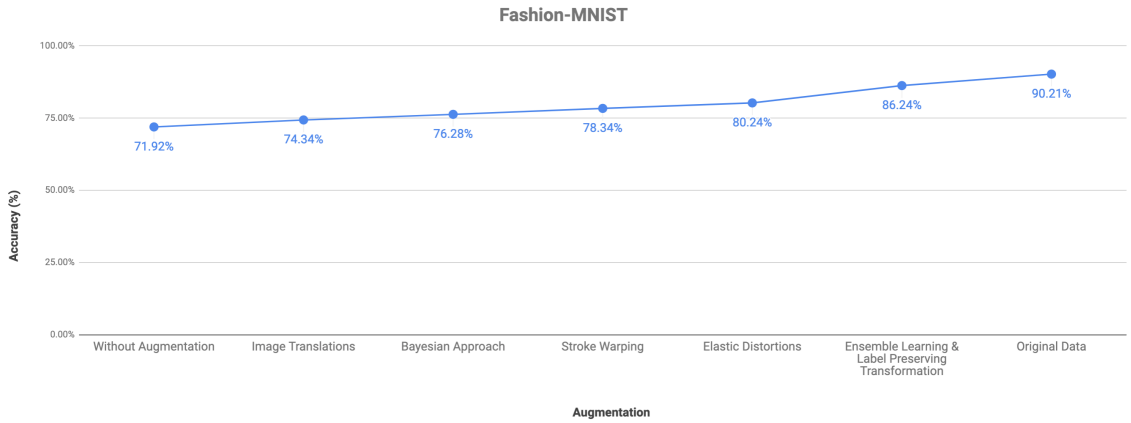

Figure 18: *Comparative result between Ensemble Learning & Label Preserving Transformations augmentation and other augmentation techniques on Fashion-MNIST dataset.*

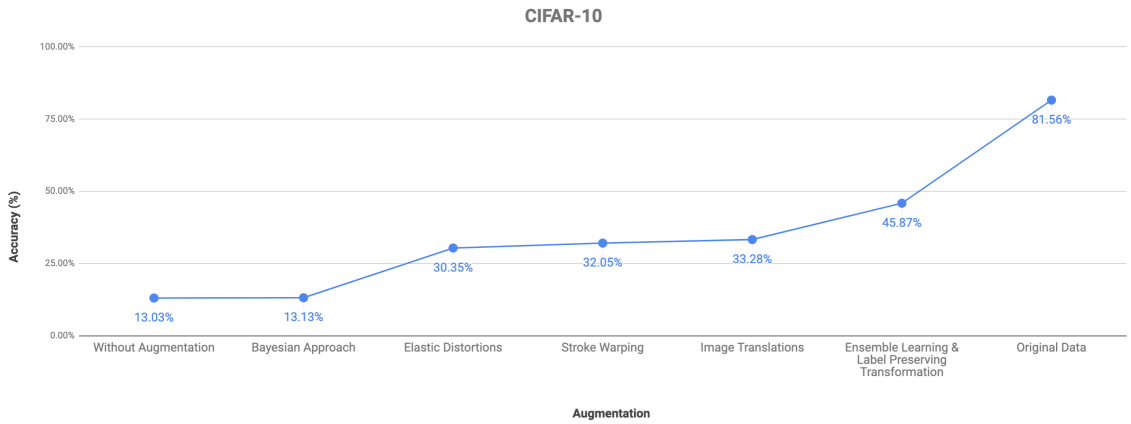

Figure 19: *Comparative result between Ensemble Learning & Label Preserving Transformations augmentation and other augmentation techniques on CIFAR-10 dataset.*

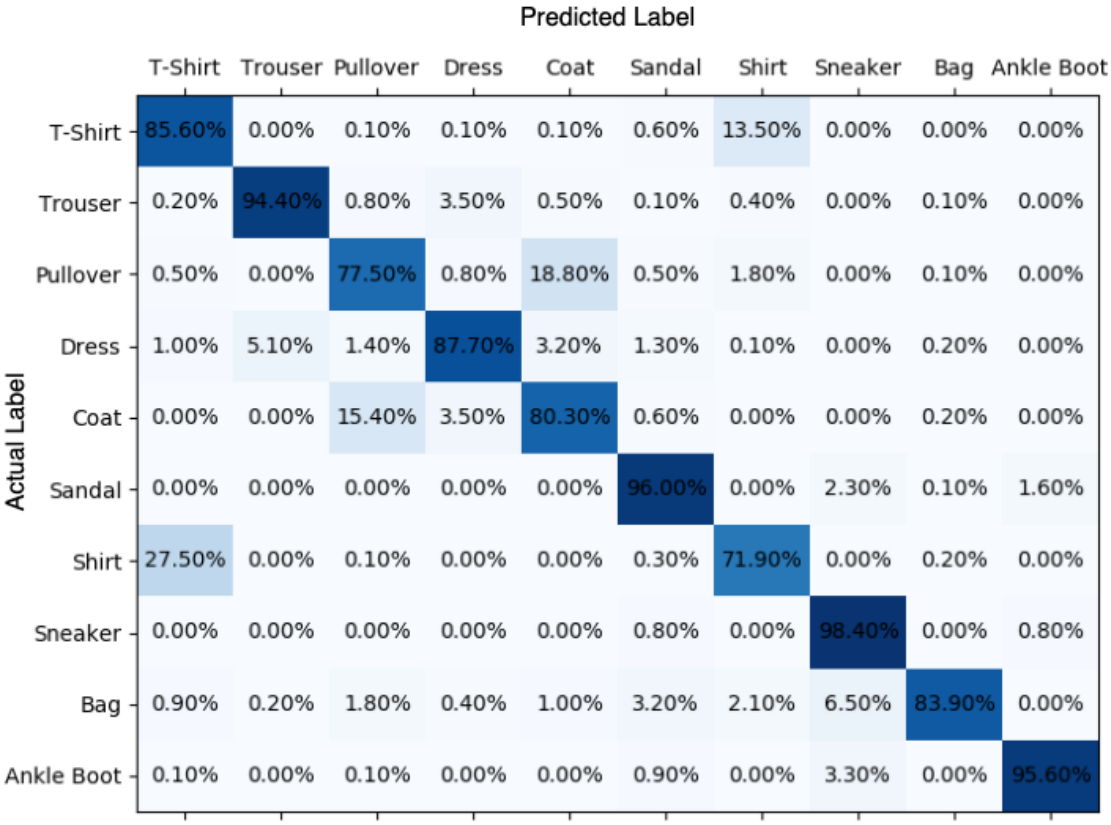

Figure 20: *Prediction accuracy of each class on Fashion-MNIST for Ensemble Learning & Label Preserving Transformations.*

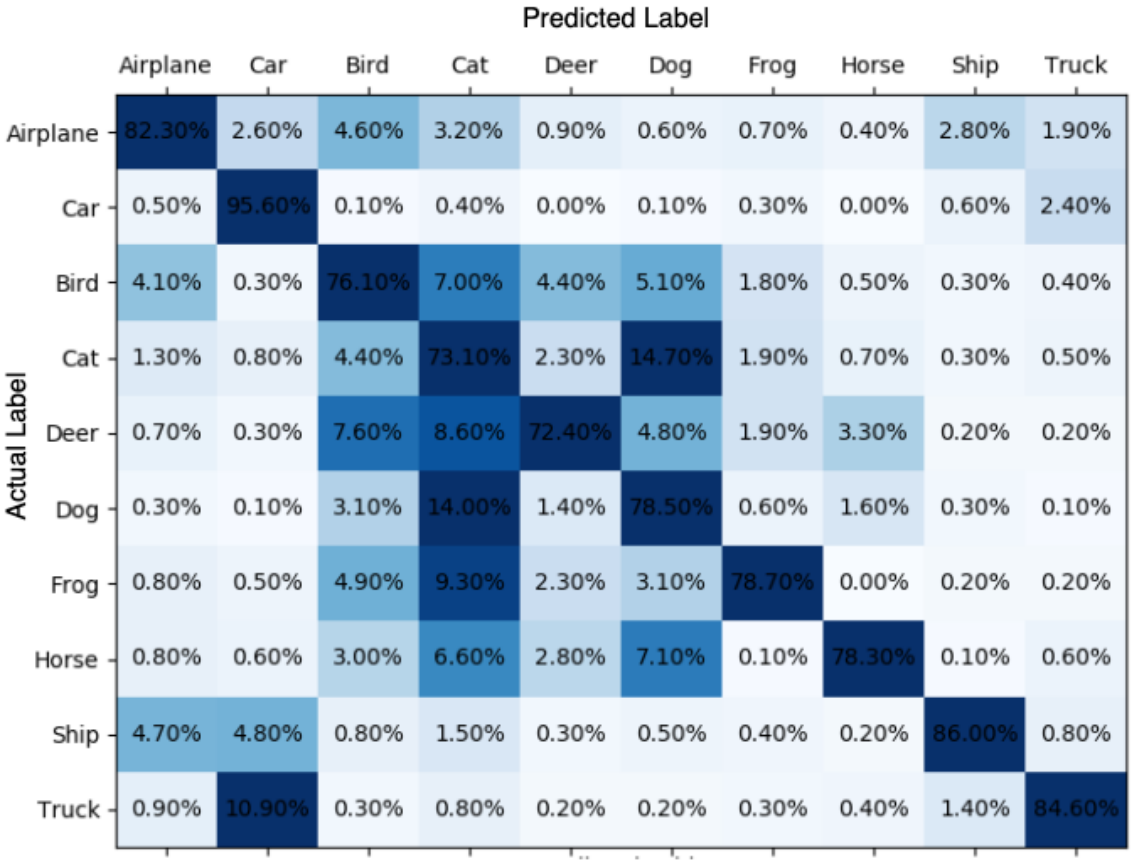

Figure 21: *Prediction accuracy of each class on the original CIFAR-10 dataset.*

but they have almost the same background-color (blue[1]). In this technique, we approach to prevent unnecessary color learning such as background-color in the classifier with augmentation of the data with a random color.

In the first step, data will be augmented as same as the previous technique (Ensemble Learning & Label Preserving Transformations). In the second step, images in the augmented dataset will be converted from the RGB color model to the HSI color model [GW08] as follow:

$$h = \begin{cases} \theta & \text{if } B \leq G \\ 2\pi - \theta & \text{otherwise} \end{cases} \quad \text{with} \quad \theta = \cos^{-1}\left\{ \frac{1/[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}} \right\} \tag{7.3}$$

Where h represents hue and r, g , and b denoted as normalized RGB value:

[1] Airplanes are almost have sky and ships have sea background

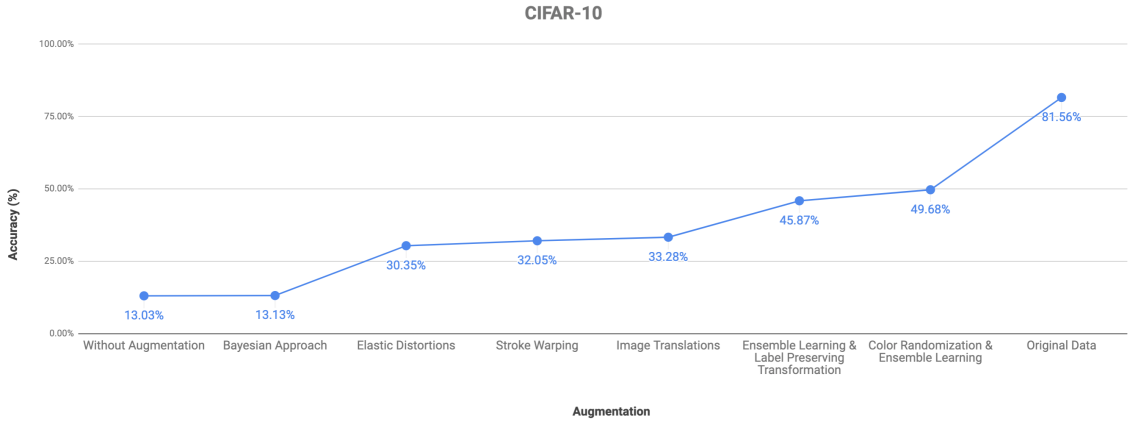

Figure 22: *Comparative result between Color Randomization & Ensemble Learning augmentation and other augmentation techniques on CIFAR-10 dataset.*

$$r = \frac{R}{R+B+G}, \qquad b = \frac{B}{R+B+G}, \qquad g = \frac{G}{R+B+G} \tag{7.4}$$

s which determines saturation of color (Chroma) defined as follow:

$$s = 1 - 3 \cdot \min(r,g,b); \quad \mathbf{s} \in [0,1] \quad \text{(with exception for black with } (s=0)) \tag{7.5}$$

and i which determines intensity defined as follow

$$i = (R+G+B)/(3 \cdot 255); \quad i \in [0,1] \tag{7.6}$$

After converting the images to the HSI model, we will enlarge the dataset by factor 4 for each image by randomizing the color of each sample. Color randomization accomplishes with adding a random value between π and $-\pi$ to the hue of the pixel mentioned in equation (7.3). Following equation presents new hue denoted as h' formally:

$$h' = h + random(\pi, -\pi) \tag{7.7}$$

In the last step, we reconvert the augmented dataset to the RGB color model and start the learning phase as same as before. The test and prediction phase will be accomplished exactly like the previous technique.

We selected the HSI color model to randomize the color without the disadvantageous effect on the pixel's intensity or chroma. As equation (7.7) illustrate this matter that our technique changes only the hue of pixel. It ables us not only to prevent the model from unnecessarily color learning but also not losing important

information such as interest points, edges, corners, and etc. in the picture. Figure 22 represents the result of this technique on Cifar-10 besides other introduced techniques. It demonstrates that this technique is even, more better, than the previous one.

7.3 Random Erasing & Bayesian Approach

Finally and the last technique, we attempt to improve the bayesian approach with augmenting the dataset at the beginning. It means that we augment our few-shot dataset before the bayesian approach starts to generate synthetic data. The data augmentation will be carried out with an augmentation technique called random erasing which we inspired from an introduced work by Zhun Zhong et al. [Zho+17]. In this technique for augmentation, we choose random patches smaller than original images size in the dataset and erase them. We erase 100 random patches with size 5×5 from the images to augment our few-shot dataset with a factor by 100. The influence of such an augmentation with random erasing is similar to the dropout layer in CNNs [WG15]. Nevertheless, in the random erasing instead of disabling random nodes (random discrete pixels), an area will be erased (disabled). Considering this augmentation, some negligible objects can be erased from samples, therefore, it can improve accuracy and cause that the bayesian approach has a more realistic and better estimation of class's distribution and observed posterior introduced in equation (3.4).

Figure 23 represents a comparison between the bayesian approach with and without random erasing. It demonstrates that this technique not only in theory but also pragmatically achieve better performance and accuracy on all introduced datasets.

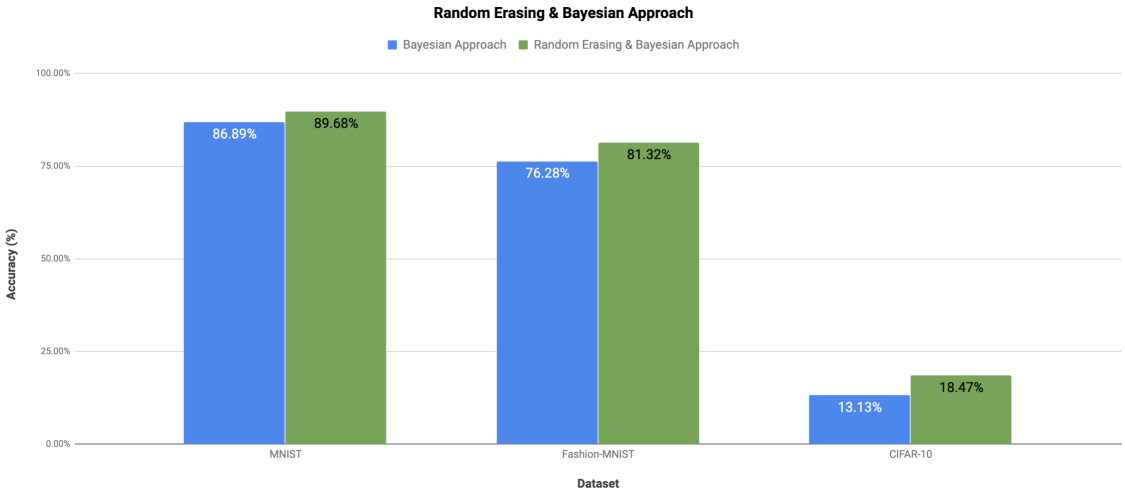

Figure 23: *Comparative result between Bayesian Approach with and without random erasing.*

8 Bibliography

[17] *Convolutional Neural Networks Tutorial in PyTorch*. https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-in-pytorch/. Accessed: 2020-01-15. 2017.

[A O16] and J. Shlens A. Odena C. Olah. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2016. arXiv: 1610.09585 [cs.CV].

[Ath] P. Athul. *Medium Handwritten digit recognition using PyTorch*. https://medium.com/@athul929/hand-written-digit-classifier-in-pytorch-42a53e92b63e. Accessed: 2019-12-16.

[Bai+19] Sung W. Baik et al. "Multi-grade brain tumor classification using deep CNN with extensive data augmentation". In: 30 (Jan. 2019). https://www.sciencedirect.com/science/article/pii/S1877750318307385, pp. 174–182.

[Blo17] Marcus D. Bloice. *Augmentor*. Version 0.2.8. 2017. URL: https://github.com/mdbloice/Augmentor.

[Com] Google Company. *Generative Adversarial Networks*. https://developers.google.com/machine-learning/gan/generator. Accessed: 2020-01-15.

[DS12] Ueli Meier Dan Cireşan and Juergen Schmidhuber. *Multi-column Deep Neural Networks for Image Classification*. 2012. arXiv: 1202.2745 [cs.CV].

[DT17] Terrance DeVries and Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017. arXiv: 1708.04552 [cs.CV].

[Efr83] Bradley Efron. "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation". In: vol. 78. Journal of the American Statistical Association, 1983, pp. 316–331. URL: https://amstat.tandfonline.com/doi/ref/10.1080/01621459.1983.10477973?scroll=top#.XlvbR5NKjOQ.

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [cs.CV].

[GPS89] D. M. Greig, B. T. Porteous, and A. H. Seheult. *Exact Maximum A Posteriori Estimation for Binary Images*. 1st ed. Vol. 51. Journal of the Royal Statistical Society: Series B (Methodological), 1989, pp. 271–279. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1989.tb01764.x.

[GW08] Rafael C. Gonzalez and Richard E. Woods. "Digital Image Processing". In: 3rd ed. Pearson Education, Inc., 2008, pp. 407–414. URL: https://pdfs.semanticscholar.org/15bd/427a1a5f9bc57a7f67fb1b1fc85c5bb39f46.pdf.

[HV17] Kashif Rasul Han Xiao and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[Kri] Alex Krizhevsky. *The CIFAR-10 dataset (Canadian Institute for Advanced Research)*. http://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2019-12-16.

[KSH17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: vol. 60. Commun. ACM, 2017, pp. 84–90. URL: https://doi.org/10.1109/ICDAR.2003.1227801.

[LeC] Yann LeCun. *exdb THE MNIST DATABASE of handwritten digits*. http://yann.lecun.com/exdb/mnist/. Accessed: 2019-12-16.

[rep17] GitHub repository. *adventures-in-ml-code*. https://github.com/adventuresinML/adventures-in-ml-code. Accessed: 2020-01-15. 2017.

[Sim+92] P. Simard et al. "Tangent Prop-A Formalism for Specifying Selected Invariances in an Adaptive Network". In: *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, 1992, pp. 895–903.

[SLD93] P. Simard, Y. LeCun, and .T. Denker. "Efficient Pattern Recognition Using a New Transformation Distance". In: *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993, pp. 50–58.

[SSP03] P. Y. Simard, D. Steinkraus, and J. C. Platt. "Best practices for convolutional neural networks applied to visual document analysis". In: vol. 2. Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, Scotland, UK, 2003. Pp. 958–963. URL: https://ieeexplore.ieee.org/document/1227801.

[STA] STANDFORD. *NIST National Institute of Standards and Technology*. https://www.nist.gov/data. Accessed: 2019-12-16.

[Tan91] Martin A. Tanner. *Tools for Statistical Inference. Observed Data and Data Augmentation Methods*. 1st ed. Vol. 67. ISBN 978-0-387-97525-2. Springer-Verlag New York, 1991.

[Tra+17] T. Tran et al. *A bayesian data augmentation approach for learning deep models*, 2017. arXiv: 1710.10564 [cs.CV].

[TW87] Martin A. Tanner and Wing Hung Wong. *The Calculation of Posterior Distributions by Data Augmentation. Journal of the American Statistical Association*. 1st ed. Vol. 82. ISBN 82(398):528–540. American Statistical Association, 1987, pp. 528–540.

[Uni] New York University. *Visual Dictionary Teaching computers to recognize objects*. http://groups.csail.mit.edu/vision/TinyImages/. Accessed: 2019-12-16.

[WG15] Haibing Wu and Xiaodong Gu. *Towards dropout training for convolutional neural networks*. Vol. 71. https://doi.org/10.1016/j.neunet.2015.07.007. Elsevier BV, Nov. 2015, pp. 1–10.

[YLW97] Larry S. Yaeger, Richard F. Lyon, and Brandyn J. Webb. "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, 1997, pp. 807–816. URL: http://papers.nips.cc/paper/1250-effective-training-of-a-neural-network-character-classifier-for-word-recognition.pdf.

[Zhe18] Zhenye. *Deep Learning with Pytorch on CIFAR-10 Dataset*. https://zhenye-na.github.io/2018/09/28/pytorch-cnn-cifar10.html. Accessed: 2020-01-15. 2018.

[Zho+17] Zhun Zhong et al. *Random Erasing Data Augmentation*. 2017. arXiv: 1708.04896 [cs.CV].

*

List of Figures

List of Tables

Statement of Authorship

I hereby confirm that the work presented in this bachelor thesis has been performed and interpreted solely by myself except where explicitly identified to the contrary. I declare that I have used no other sources and aids other than those indicated. This work has not been submitted elsewhere in any other form for the fulfilment of any other degree or qualification.

Bonn, March 21, 2020

Milad Navidizadeh