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
Bachlor Thesis for Information Systems and Artificial Intelligence

in degree course

Informatik (B.Sc.)

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

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

Bonn, March 1, 2020

Acknowledgement

A special thanks goes out to Dr. rer. nat. Wolfgang Hansmann, for maintaining the guideline up to version 0.4 and Prof. Dr.-Ing. André Miede for his work on an extended template for `classicthesis` that I learned (and borrowed) a lot from.

> » *It has been a long journey to this moment.* «   Sidney Poitier, 1964

Contents

1 Motivation

Nowadays machine learning and deep learning have become a distinguished approach for visual recognition tasks and has achieved great success in this process. However, they seek a large amount of labeled data to learn. Providing this amount of labeled data not only will bring much effort along but also will occupy a huge size of storage and seek large storage. In contrast, humans are very good in visual recognition so that, they can learn with one [1] or few [2] examples. Imagine one kid who can recognize a lion in a picture after looking a few pictures of lions as an example. We want to simulate and apply this human's ability to the deep learning and make them learn with few examples, with desirable accuracy. In this bachelor thesis, we concern ourselves with few-shot learning in deep learning. We aim to learn and train a model when there are few labeled examples obtainable. We approach to generate artificial examples from a few available labeled examples and enalrge our dataset artificially. This technique known as data augmentation. These artificial labeled examples aid us to learn better with more accuracy and prevent overfitting. Data augmentation is our focus in this work to achieve few-shot learning and prevent overfitting. In this thesis, we will introduce different well-known methods 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 to discover the potential possibility of combining the different methods of data augmentation to increase accuracy and reduce error-rate and improve the learning process. We will focus on visual recognition tasks and their classification. Additionally, we will concentrate on the implementation of various methods of data augmentation for convolutional neural networks

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

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

2 Introduction

Neural networks can possibly contain multiple non-linear layers and this makes them very expressive models that can learn very complicated relationships between their inputs and outputs. With even limited input data, neural networks can discover and learn many relations from the data, however, sometimes the discovered and learned relations do not exist or just consist of redundant information and relations. Redudant relation can potentially arise of data-noise or lack of data-generalization. Non exist relations can potentially emerge from lack of enough data. These phenomena known as *overffiting* in deep learning. In other words, learning with few labeled examples or noisy data causes overfitting. Overffiting cause low accuracy and high error-rate. Hence, we approach to propagation of artificial labeled examples from a few given examples to prevent overfitting and reduce the error rate and increase the accuracy.

As we mentioned aquiring a huge labeled dataset is expensive and seeks much effort and time. Therefore we aim to generate artificial example from few obtainable labeled examples. In other words, we augment our data and this strategy is known as data augmentation. There are a few well-known methods for data augmentation. We aim to introduce them in this thesis. Besides we will implement these methods to compare their efficiency and capability. These methods are as follow:

- **Label Preserving Transformation** 5.1
- **Elastic Distrotion** 5.1.2
- **Stroke Warping** 5.1.3
- **Bayesian Approach** 5.2

3 Data Representation

In this chapter, we will briefly introduce the datasets which we use in this work to evaluate the performance of each data augmentation approach pragmatically. There are mainly 2 reasons that we have chosen these datasets. First, they are well-know datasets and there are many researchers on them and many reported results. Additionally, they are simple enough to learn and append non-complex classifiers to classify them with desirable accuracy. The second reason is that all these 3 datasets have 10 classes with small differences. All of these aids us to have a better and more realistic benchmark.

3.1 MNIST

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

The MSNIT dataset consists of 60,000 train- and 10,000 test-images and each image is grayscale with 28×28 pixels. It has 10 classes that represent $0-9$ digits and data is fairly splitted between classes [LeC]. MNIST is one of the most popular datasets for deep learning because of the not too high 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 at the table 1 and the figure 1 shows an example of the dataset.

3.2 Fashsion-MNIST

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

[1] https://jobs.zalando.com/de/tech/

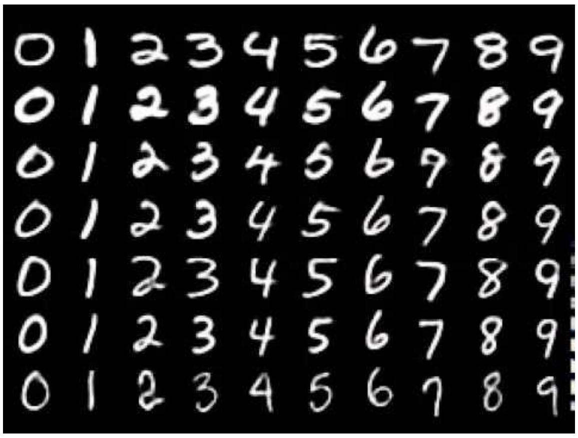


Figure 1: *7 examples per class of MNIST dataset, merged in one image [Ath]*

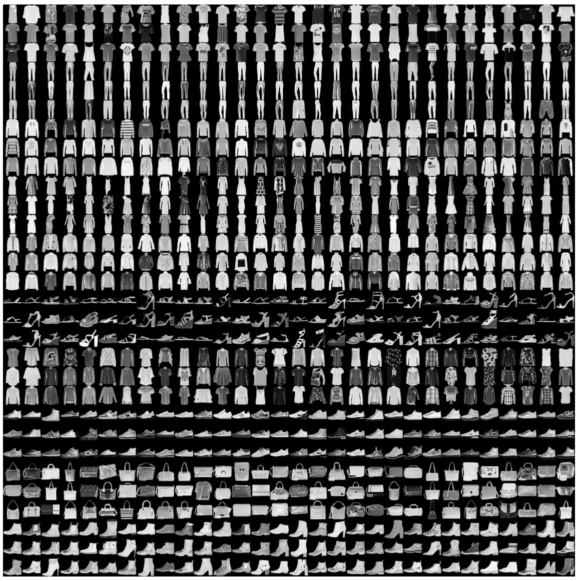


Figure 2: *Examples of Fashion-MNIST dataset, merged in one image*

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

[2] https://github.com/zalandoresearch/fashion-mnist#benchmark/

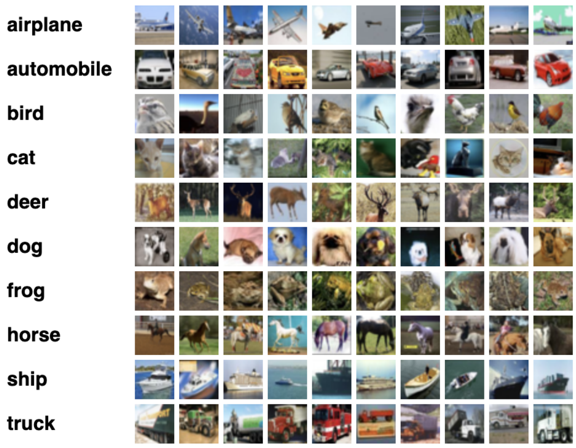


Figure 3: *10 examples per class of CIFAR-10 dataset, merged in one image [Kri]*

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)
Fashsion-MNIST	10	60,000	10,000	28 × 28	1 (Grayscale)
CIFAR-10	10	50,000	10,000	32 × 32	3 (RGB)

3.3 CIFAR-10

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

The dataset consists of 60,000 RGB with 32×32 pixels images, which are divided to the 50,000 train and 10,000 test datasets. As the name makes it clear the CIFAR-10 contains 10 classes ([plane, car, bird, cat, deer, dog, frog, horse, ship, truck]) [Kri]. One of the lowest reported error-rate with a convolutional neural network managed to achieve 2.56% [DT17]. You can find the information about the dataset at the table 1 and the figure 3 shows an example of the dataset.

4 Preliminaries

4.1 Introduction

4.2 Convolutional Neural Network

4.3 Overffiting

5 Data Augmentaion

In this chapter, we will introduce a few noteworthy related works in this field which are mainly consist of 2 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 looking for enlarging datasets artificially and generate synthetic data based on the obtainable few samples to increase the accuracy of the prediction.

5.1 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.

5.1.1 Image Translations

Image translations is one of the simplest and meanwhile applicable well-known technique of the lable preserving transformations approach. As the research by Krizhevsky et al. [KSH17] proposes, we extract image translations and their horizontal reflections to generate the synthetic data and augment the dataset. Image translation consists of extracting patches that are smaller in size than original images. 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 2, therefore finally dataset can be enlarged by factor:

$$2 \times (n - m + 1) \times (n - m + 1)$$

Figure 4 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 8 new data points.

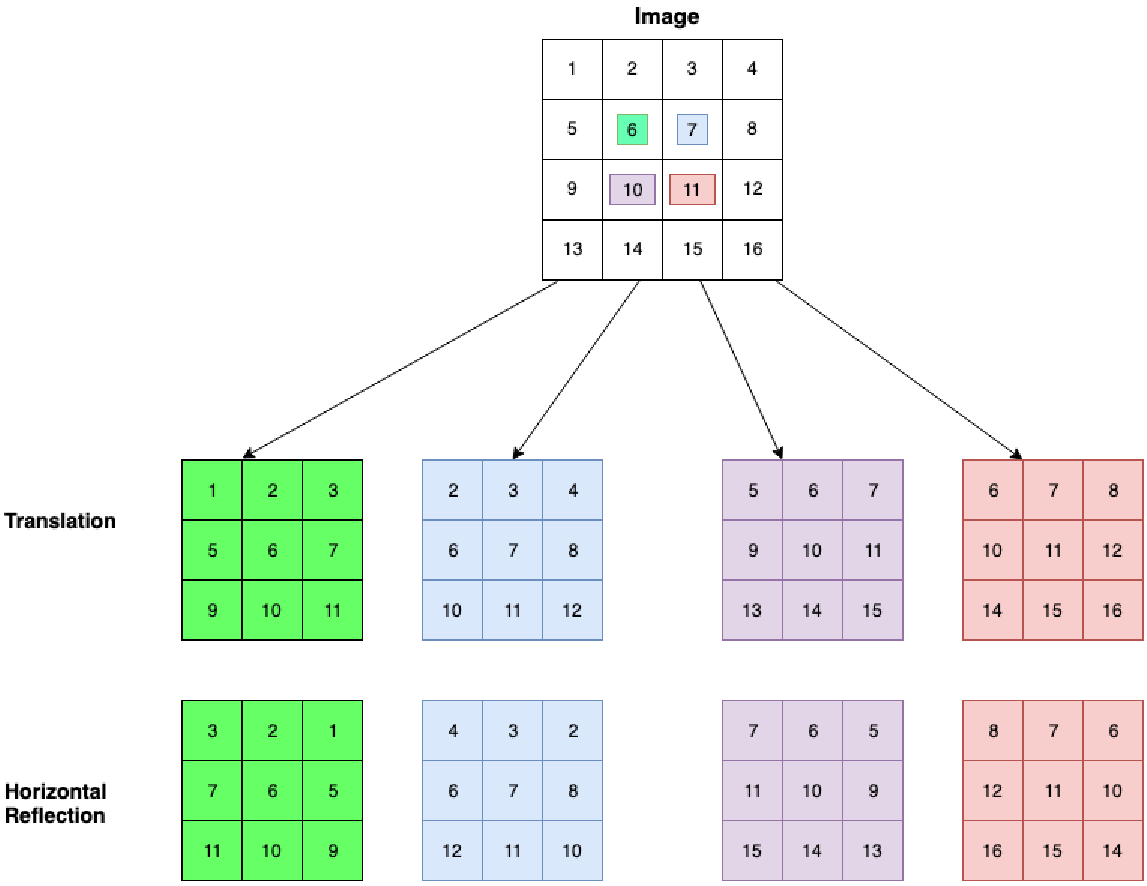
At test (prediction) time, the method extracts the patches with the same size $(m < n)$, however this time the patches will be extracted from 4 corners and center of the test image. The network predicts on these five patches and their horizontal reflections (10 patches altogether). In the end, the average on the softmax layer will be determined the final prediction.

5.1.2 Elastic Distortion

Another well-known technique for data augmentation is elastic distortion. Quite similar to the image translations, the ultimate goal is to generate synthetic data set 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 p' the pixel which we want to dispalacment it with $\Delta x(x,y) = \alpha x$ and $\Delta y(x,y) = \alpha y$ and $p_{(x,y)}, p_{(x+1,y)}, p_{(x,y-1)}, p_{(x+1,y-1)}$ are the neighbors (on



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3
5	6	7
9	10	11

2	3	4
6	7	8
10	11	12

5	6	7
9	10	11
13	14	15

6	7	8
10	11	12
14	15	16

3	2	1
7	6	5
11	10	9

4	3	2
6	7	8
12	11	10

7	6	5
11	10	9
15	14	13

8	7	6
12	11	10
16	15	14

Figure 4: *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*

origin square) of p' in the new location after displacment and $I(p)$ shows the intensity of pixel p. Then the vertical interploation 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 interploation yields a new intensity for pixel p' after displacment:

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

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, value of which 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.

As same as image translations, at test (prediction) time, image will be augmented by factor 10 with elastic distortion. Finally, avrage on softmax layer for this enlarged 10 images determined the prediction (lable).

5.1.3 Stroke Warping

This teqnique as same as previously introduced teqniques uses predetermined families of transformations. In other words, we enlarge our dataset artificially with the aid of well-known classical computer vision transformations. This method notwithstanding of non-heavy complexity accomplished desirable results even on medical purposes [Bai+19]. The ideas of this methode raised from Tangent Dist [Sim+92] and Tangent Prop [SLD93] works.

In this method, we perform small changes in images to augment our data. That means we augment our images with skewing, rotating and shearing (scaling) them [YLW97]. As same as the image translations and elastic distortion the augmentation will be started before the training phase and the training will be performed on enlarged data. Figure 5 represents each mentioned transformation to make them visually understandable.

At test (prediction) time, image will be enlarged by factor 10 and prediction will be preformed on avraging of softmax layer of this 10 images, as same as previous teqniques.

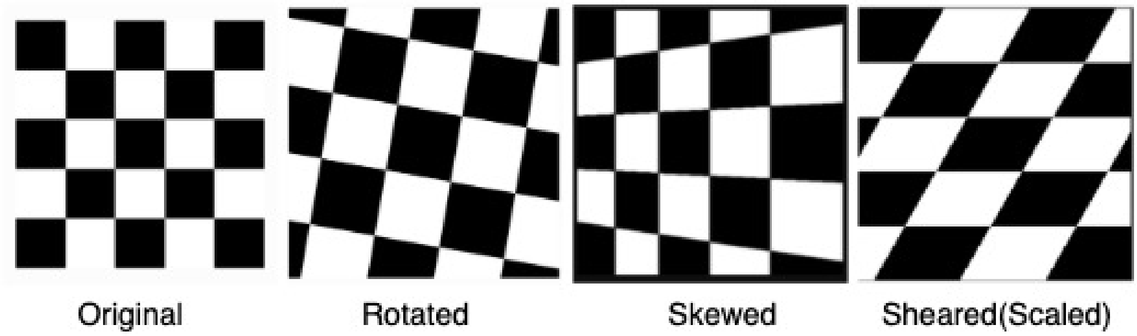


Figure 5: *An exmaple of rotation, skew, and shear (scale) transforamtions for stroke warping respectively from left to right [Blo17]*

5.2 Bayesian Approach

The astute reader should have noticed that, although quite different, all the introduced teqniques so far share a fundamental aspect. Precisely speaking, in all these teqniques, 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 the 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 class of teqniques 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 need to briefly introduce generative models. Specifically, first, we present the Generative Adversarial Networks (GANs) proposed initially by Goodfellow et al. [Goo+14]. In what follow, we will introduce GANs and after that and in the Bayesian approach description we will explain how this approach uses the main idea of GANs and extend it to improve data augmentation.

5.2.1 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 portion of a GAN learns to distinguish real data from the synthetic data generated by the generator in the GAN and with this matter 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 2 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. In general, GANs try to replicate a probability distribution. For this matter, they use loss function to measure the distance between the distribution of the generator's data and the distribution of the real data. As we mentioned Goodfellow el al. [Goo+14] suggest minimax loss in their work. The minimax loss defined as:

$$Minimax\ Loss = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \tag{5.1}$$

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 (5.1) and the discriminator tries to maximize it. Figure 6 shows a basic architecture of a GAN.

Now and after briefly introduction of GANs we can start with the **Bayesian Approach**. One of the noteworthy work for data augmentation with the aid of the bayesian model proposed by Toan Tran et al. [Tra+17]. In this approach,

[1] noise

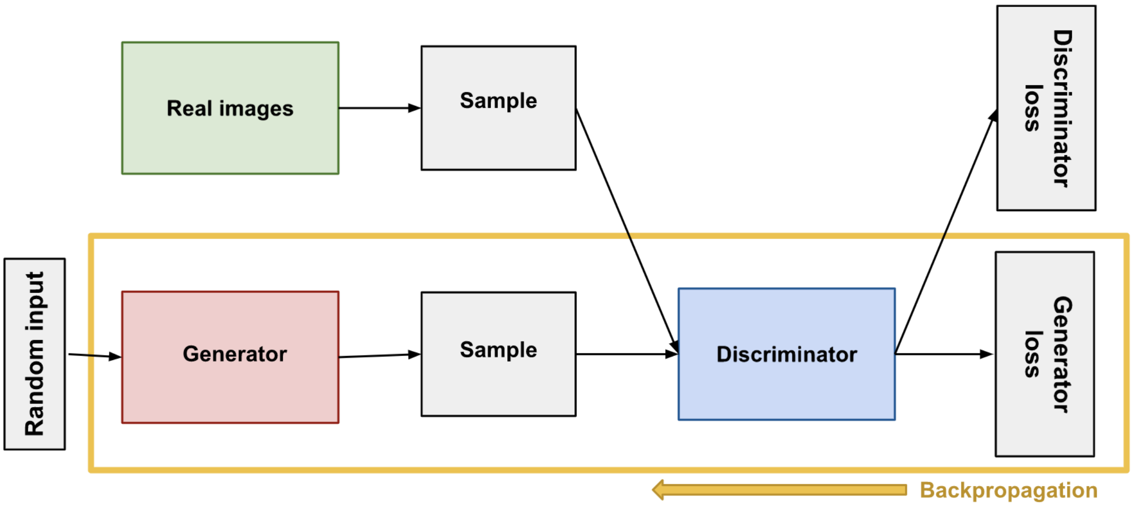


Figure 6: *GAN architecture [Com]*

our deep learning model tries to estimate the distributions of labeled data and with aid of the estimated distributions optimizes 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 derived from the Bayesian model. The principle idea for data augmentation using latent variables proposed by the statistical learning community [TW87]. Nevertheless applying the idea, directly into deep learning seeks a massive computational effort. Therefore we talked before about the estimation. To be more precise, the approach uses a novel Bayesian data augmenation 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 use Monte Carlo to estimate the expected value of the network parameters given the previous estimate instead of calculating 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 turned in to reality with the aid of GANs. The proposed GAN consists of one generator and 2 discriminators. As we in the GANs section (5.2.1) discussed the generator is responsible to generate synthetic data and one of our discriminators distinguishes fake and real data. However, the second discriminator discriminates between the classes of data. Figure 7 represents the utilized netwok architecture in this approach visually. This proposed architecture is nearly similar to the Auxiliary classifier GANs (AC-GANs) [A O16]. Nevertheless, in the AC-GANs discriminator responsible for both classification real-or-fake data and data labels (classes) and in our network we utilized 2 discriminators separately for this matter.

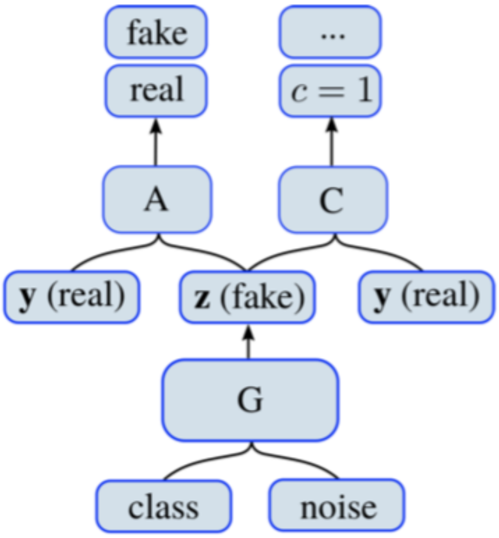


Figure 7: *The network architecture of Bayesian data augmentaion approch [Tra+17]. G: Generator, A: Authenticator, C: Classifier.*

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

$$\theta^* = \arg\max \log p(\theta|\mathbf{y}) \tag{5.2}$$

Where training set 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 denoted as model (network) parameters and observed posterior defined as:

$$p(\theta|\mathbf{y}) = p(\theta|t, \mathbf{x}) \propto p(t|\mathbf{x}, \theta)p(\mathbf{x}|\theta)p(\theta) \tag{5.3}$$

Now if we assume that the data samples $\mathcal{Y}$ are conditionaly independent, we can define the following loss function which maximize the (5.2).

$$\log p(\theta|\mathbf{y}) \approx \log p(\theta) + \frac{1}{N}\sum_{l}^{N}(\log p\left(t_n|\mathbf{x}_n, \theta\right) + \log p\left(\mathbf{x}_n|\theta\right)) \tag{5.4}$$

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

$$p(\theta|\mathbf{y},\mathbf{z}) = \frac{p(\mathbf{y},\mathbf{z},\theta)}{p(\mathbf{y},\mathbf{z})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z}|\mathbf{y})p(\mathbf{y})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})}{p(\mathbf{z}|\mathbf{y})} \tag{5.5}$$

where the expectation step will be defined as follow:

$$p(\theta|\mathbf{y},\mathbf{z}) = \frac{p(\mathbf{y},\mathbf{z},\theta)}{p(\mathbf{y},\mathbf{z})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z}|\mathbf{y})p(\mathbf{y})} = \frac{p(\mathbf{z}|\mathbf{y},\theta)p(\theta|\mathbf{y})}{p(\mathbf{z}|\mathbf{y})} \tag{5.6}$$

where $\mathbf{z}_m \sim p\left(\mathbf{z}|\mathbf{y},\theta^i\right)$, for $m \in \{1,\ldots,M\}$. 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:

$$p(\mathbf{z}|\mathbf{y},\theta) = p\left(t^a,\mathbf{x}^a|\mathbf{y},\theta\right) = p\left(t^a|\mathbf{x}^a,\mathbf{y},\theta\right)p\left(\mathbf{x}^a|\mathbf{y},\theta\right) \tag{5.7}$$

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 the θ for the maximization step with concern of removing the independent terms for θ will derived the maximization of $\hat{Q}\left(\theta,\theta^i\right)$ as follow:

$$\hat{Q}\left(\theta,\theta^i\right) = \log p(\theta) + \frac{1}{N}\sum_{n=1}^{N}\left(\log p\left(t_n|\mathbf{x}_n,\theta\right) + \log p\left(\mathbf{x}_n|\theta\right)\right) + \frac{1}{M}\sum_{m=1}^{m}\log p\left(\mathbf{z}_m|\mathbf{y},\theta\right)$$
$$=$$
$$\log p(\theta) + \frac{1}{N}\sum_{n=1}^{N}\left(\log p\left(t_n|\mathbf{x}_n,\theta\right) + \log p\left(\mathbf{x}_n|\theta\right)\right) + \frac{1}{M}\sum_{m=1}^{n}\left(\log p\left(t_m^a|\mathbf{x}_m^a,\theta\right) + \log p\left(\mathbf{x}_m^a|\theta\right)\right) \tag{5.8}$$

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

As we made it clear, the above formal explanations and equations are derived from [Tra+17] and in some points are matched one-to-one.

6 Pragmatic Expriments

in this Chapter

6.1 CNNs Architecture

In this section, we will introduce our classifiers (CNNs) architecture for various datasets which we introduced in the chapter (3). We picked our CNNs architecture from different sources regarding their non-heavy complexity and desirable accuracy. Maybe our chosen CNNs architecture doesn't provide the best-reported accuracy on datasets, but while 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. Moreover, as we will report in chapter (7) our result are not so far away from best-reported accuracy on the orginal dataset.

6.1.1 MNIST

For the MNIST dataset, we have chosen a semi-simple CNN architecture from [rep17] with 2 convolutional layers and 2 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.

6.1.2 Fashsion-MNIST

The Fashion-MNIST dataset use CNN architectur deriven form [**TODO**] with 2 convolutional layers and 2 fully connected layers as same as MNIST dataset just with this different of up- and downsampling and kernels sizes. ReLU function is

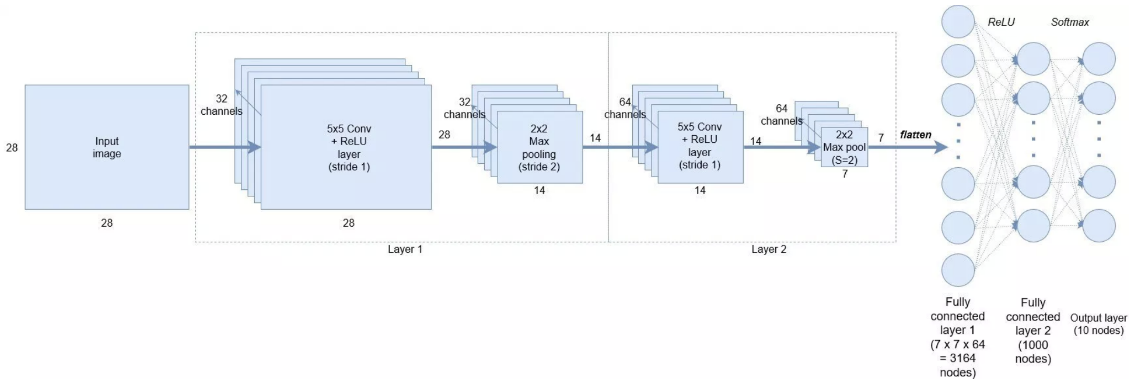

Input image
28
28
32 channels
5x5 Conv + ReLU layer (stride 1)
28
28
32 channels
2x2 Max pooling (stride 2)
14
14
Layer 1
64 channels
5x5 Conv + ReLU layer (stride 1)
14
14
64 channels
2x2 Max pool (S=2)
7
7
Layer 2
flatten
ReLU
Softmax
Fully connected layer 1 (7 x 7 x 64 = 3164 nodes)
Fully connected layer 2 (1000 nodes)
Output layer (10 nodes)

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

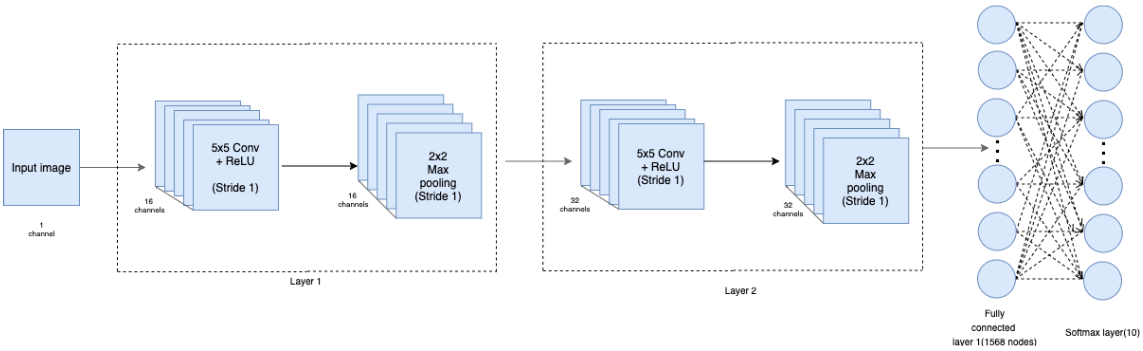

Input image
1 channel
16 channels
5x5 Conv + ReLU (Stride 1)
16 channels
2x2 Max pooling (Stride 1)
Layer 1
32 channels
5x5 Conv + ReLU (Stride 1)
32 channels
2x2 Max pooling (Stride 1)
Layer 2
Fully connected layer 1(1568 nodes)
Softmax layer(10)

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

utilized as the activation function. The learning rate for Adam optimizer is set to 0.001. Figure 9 represents the explained CNN architecture visually.

6.1.3 CIFAR-10

For the CIFAR-10 dataset, we have chosen a bit more complex CNN architecture from [Zhe18] with 6 convolutional layers and 3 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** as same as 2 previous CNNs architecture have been chosen. The learning rate for Adam optimizer is set again 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.

6.2 Implementaions

In what follow, we will point out shortened to the manner of proceed and implementation of each approach and technique.

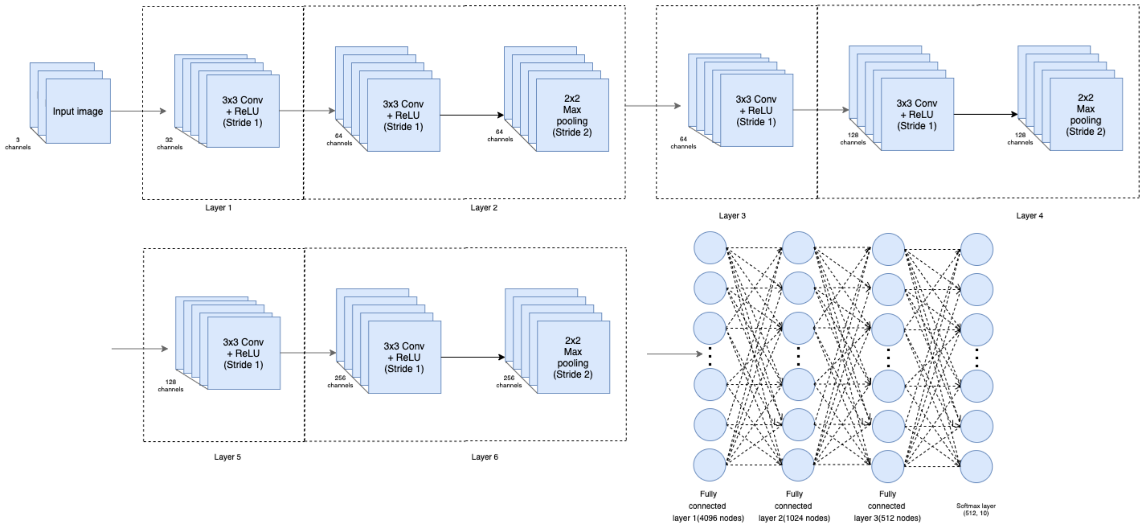

Input image
3 channels
3x3 Conv + ReLU (Stride 1)
32 channels
Layer 1
3x3 Conv + ReLU (Stride 1)
64 channels
2x2 Max pooling (Stride 2)
64 channels
Layer 2
3x3 Conv + ReLU (Stride 1)
64 channels
Layer 3
3x3 Conv + ReLU (Stride 1)
128 channels
2x2 Max pooling (Stride 2)
128 channels
Layer 4
3x3 Conv + ReLU (Stride 1)
128 channels
Layer 5
3x3 Conv + ReLU (Stride 1)
256 channels
2x2 Max pooling (Stride 2)
256 channels
Layer 6
Fully connected layer 1(4096 nodes)
Fully connected layer 2(1024 nodes)
Fully connected layer 3(512 nodes)
Softmax layer (512, 10)

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

In general 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 [1]. As we mentioned the main part (Machine Learning) carried out with a powerful library form Facebook called Pytorch [2].

6.2.1 Image Translations

6.2.2 Elastic Distortion

6.2.3 Stroke Warping

6.2.4 Bayesian Approach

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

7 Result & Comparison

In this chapter, we will publish the results of our pragmatic experiments on each introduced approach and dataset and following that we will discuss and compare the advantage and disadvantages of each approach and their behavior on different datasets. These results and comparisons do not only provide a good insight into each approach but also is a preface of the next chapter and the manner of genesis the idea of new approaches.

To have a good and comprehensive comparison, we tried the approaches and techniques in different environments. To be more precise, we considered the accuracy of the original datasets (with all train and test samples) and accuracy of learning without augmentation on the few-shot dataset to keep tracking of the differences between accuracy of them with augmentation on the few-shot dataset. Due to the comprehensive comparison, we tried various of the k-shot datasets:

$$k = \begin{cases} \{1,5,10\} & \text{if } dataset = \ MNIST\ or\ Fashion-MNIST \\ \{10,20,30\} & \text{if } dataset = \ CIFAR-10 \end{cases} \tag{7.1}$$

Additionally, to compare the result in fair circumstances we augmented the data by factor 100 (100X) for the training and by factor 10 (10X) with each augmentation technique. In the end and to provide a more realistic result we used 10-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.

The figures 11, 12, and 13 represent these results and accuracy for each approach and technique respectively for MNIST, Fashion-MNIST, and CIFAR-10 datasets.

The first look to the 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 accuracy for each technique. As there is just one sample for each class, the CNN

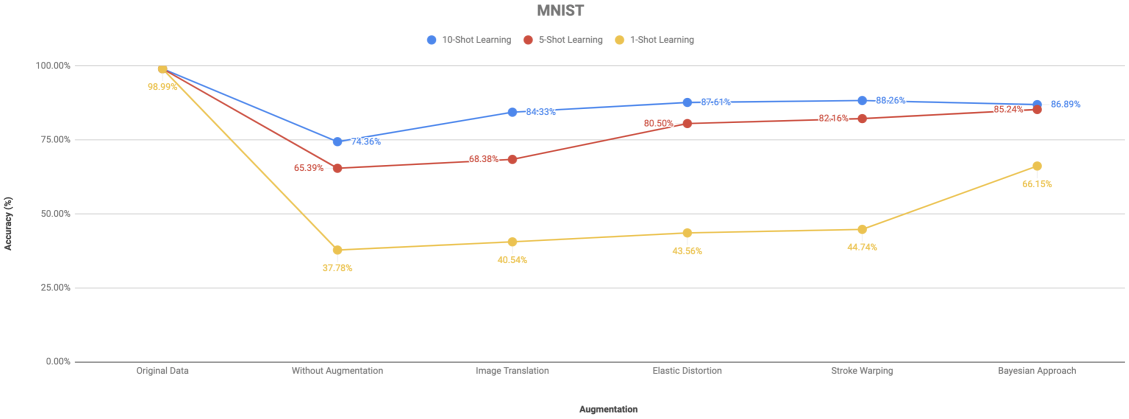


Figure 11: *Result of augmentation techniques on MNIST dataset*

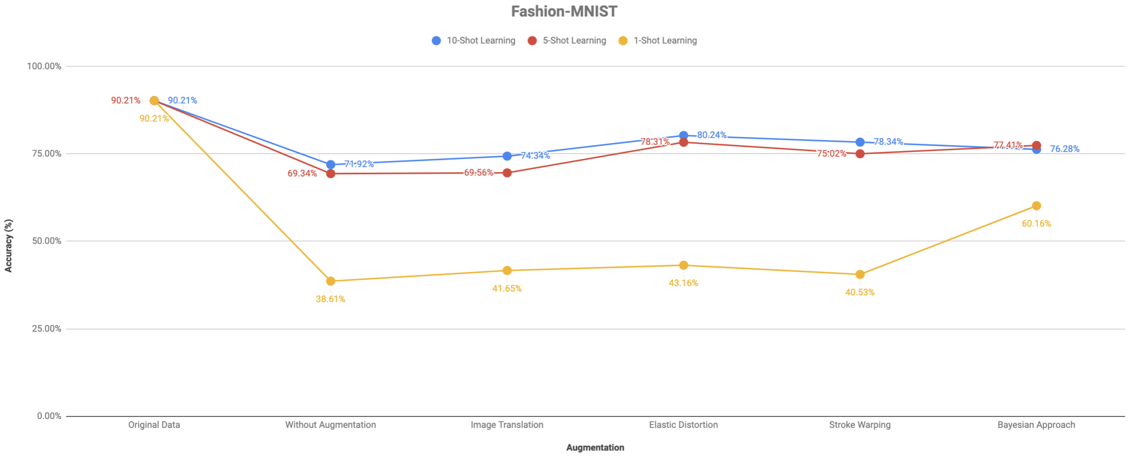


Figure 12: *Result of augmentation techniques on Fashion-MNIST dataset*

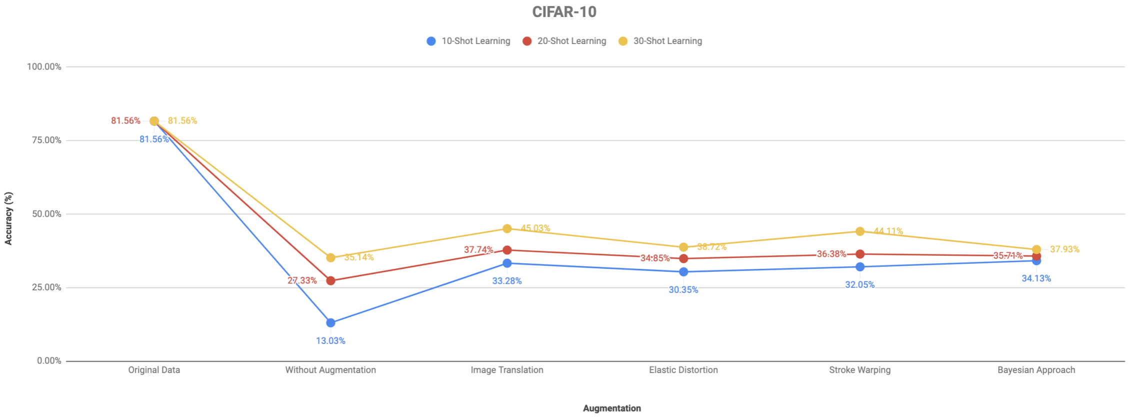


Figure 13: *Result of augmentation techniques on CIFAR-10 dataset*

involves pretty soon with overfitting. Nevertheless, the accuracy is about 30% and that means that the prediction is not randomly even for 1-shot learning on MNIST and Fashion-MNIST.

Another impression of the first look to the figure 13 is the considerable gap between accuracy on original data and few-shot learning on CIFAR-10. This matter exposes the major role of color (RGB images) in learning and accuracy. As long as this gap is not observable on MNIST and Fashion-MNIST.

8 Contribution Of Work

9 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.

[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.

[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 CIFAR10 Dataset*. https://zhenye-na.github.io/2018/09/28/pytorch-cnn-cifar10.html. Accessed: 2020-01-15. 2018.

*

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 1, 2020

Milad Navidizadeh