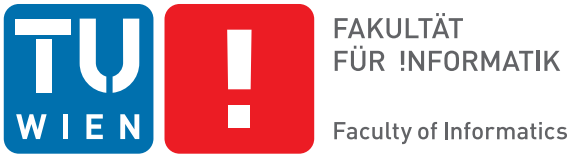

Exercise 3 - Deep Learning

Machine Learning

Group 6

Diogo Braga (E12007525)
Enrico Coluccia (E12005483)
Matthias Kiss (01218325)

January 28, 2021

Contents

1 Introduction **1**

2 Dataset Description **2**
2.1 Fashion MNIST 2
2.2 CIFAR-10 3

3 SIFT based Classification **4**
3.1 Scale Invariant Features Detection 4
3.2 Bag of Words System 5
3.3 Results 5

4 Color Histogram based Classification **9**
4.1 Computing the different Color Histograms 9
4.2 Results 10

5 Convolutional Neural Networks **15**
5.1 FashionMNIST 15
5.1.1 Architectures 16
5.1.2 Learning 17
5.1.3 Results 20
5.1.4 Conclusions 20
5.2 CIFAR-10 20
5.2.1 Architectures 20
5.2.2 Learning 21
5.2.3 Results 24
5.2.4 Conclusions 24
5.3 Featuremaps Visualization 24

6 Conclusion **27**

Chapter 1

Introduction

The goal of this exercise is to classify images and to compare the performance of a Convolutional Neural Network (CNN), which is considered a deep learning approach, to other classical classifiers, i.e. Support Vector Machine (SVM), K-Nearest-Neighbor (KNN), Decision Tree, and Multi-Layer-Perceptron (MLP).

The image data sets we used for this exercise were the FashionMNIST and CIFAR-10 data sets, which are publicly available repositories of images created for the purpose to benchmark different image classification approaches. More on the data sets can be found in chapter 2.

In order to achieve a basis for the comparison of the deep learning and classical approaches we first use a SIFT based descriptor on the images of both data sets (chapter 3) to train the four classical approaches and evaluate their performance based on this descriptor. In chapter 4 we use Color Histogram as a very simple alternative descriptor to SIFT for the classical classifiers and again examine the performance of the 4 classifiers using this descriptor.

Finally we use the raw image pixel data as input for our deep learning approach in chapter 5. Here we compare the performance of different CNN architectures and find that it is possible to achieve better accuracy with CNNs and the raw pixel input than with the 4 classical approaches and both descriptors. However it is not guaranteed that the CNNs deliver better results, when choosing the wrong architecture.

Chapter 2

Dataset Description

In this section the selected image data sets are presented.

2.1 Fashion MNIST

Fashion-MNIST [1] is a data set of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grey-scale image, associated with a label from 10 classes.

The 10 classes (see figure 2.1) are: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

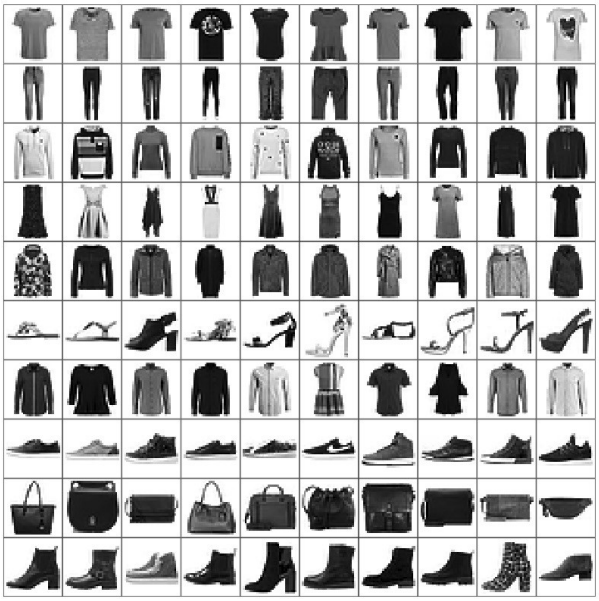

Figure 2.1: CIFAR10 Classes Visualization

[1] https://github.com/zalandoresearch/fashion-mnist

2.2 CIFAR-10

CIFAR-10 [2] is a data set that consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. Thus there are 50000 training images and 10000 test images.

The data set is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The 10 classes are represented in the figure 2.2.

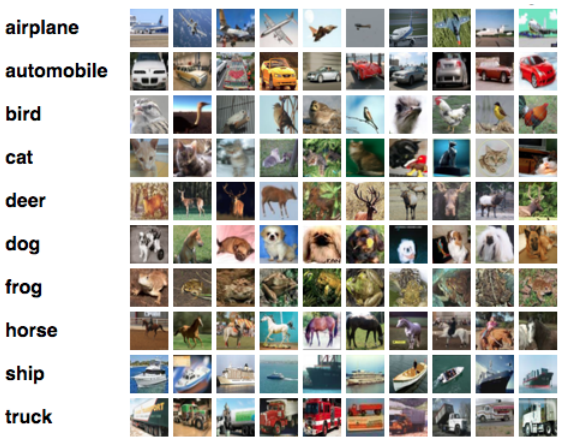

Figure 2.2: CIFAR10 Classes Visualization

[2] https://www.cs.toronto.edu/~kriz/cifar.html

Chapter 3

SIFT based Classification

As first traditional feature based extraction method we chose the SIFT approach by following a bag of (visual) words (BoVW) system to create the histograms and the sets for the classification tasks.

3.1 Scale Invariant Features Detection

Introduced in 2004 by D. Lowe in his paper, Distinctive Image Features from Scale-Invariant Keypoints [1], the Scale Invariant Feature Transform (SIFT) is an algorithm that extracts keypoints and the related descriptors from a given image. The main phases to compute the algorithm are:

- **Scale-space Extrema Detection**: in which by means of a Difference of Gaussians (DoG) we search for the best scale for the keypoint.
- **Keypoint Localization**: we identify the location of the extrema and we delete the edges response from the DoG (as they are not keypoints!).
- **Orientation Assignment**: we create the orientation histogram for each keypoint.
- **Keypoint Descriptor**: we create the final descriptors array for each keypoint.
- **Keypoint Matching**: eventual matching between two or more images.

We can easily understand that this algorithm is powerful and can handle changes in viewpoint and significant changes in illumination. The algorithm is also fast and efficient, it can also run in real-time. Another important characteristic is that SIFT uses only a monochrome intensity image (hence the colours are irrelevant). To compute the descriptors we used the *OpenCV* library.

[1] `https://people.eecs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf`

3.2 Bag of Words System

In bag of words (BoW), we count the number of times each word appears in a document, use the frequency of each word to find the keywords of the document, and make a frequency histogram from it. Thus, we treat a document as a bag of words (BoW). We have the same concept in bag of visual words (BoVW), but instead of real words (text), we use image features, for our particular case descriptors, as the "words". To train the bag of visual word system we used the following steps:

- Compute the features (in this case, the descriptors) for each image in the training set
- Cluster those features with KMeans (the higher the number of clusters, the finer it is)
- Build the histograms for each images in training/test
- Create a model with the training data
- Fit the model with the test data and measure the performance

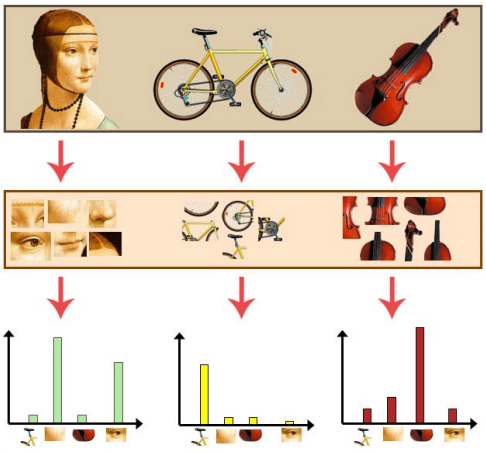

Figure 3.1: Bag of words SIFT based.

3.3 Results

The results were obtained on four different algorithms (i.e. Decision Tree, MLP, SVM, and KNN) with standard hyper parameters. We tried also to perform some optimization but it was extremely time consuming since the gridsearch algorithm did not converge in some cases (over 3 days of computing for one algorithm). In the table 3.1 we can see how the implementation worked better on the FashionMNIST data set as opposed to the CIFAR-10 data set, where the performance of our bag of visual words system was rather poor. We can also see how the algorithms are computationally expensive, in fact it

took over 1 hour of computation for the Support Vector Machine (that also has the best performance) and around 10 minutes for the Decision Tree classifier.

The running time of the whole process was significant. In particular we noticed that both classification tasks and the BoVW system implementation took several minutes to perform all the operations. We measured all the metrics to perform a comparison between all the algorithm presented in this report.

The detailed report of the computations are presented in table 3.2. From the table we can see how the most time consuming task in this kind of system is the creation of the histograms for both the training set and test set. The clustering algorithm took around one minute in both the data sets. The overall run time was around 10 minutes for the CIFAR-10 and around 8 minutes for the Fashion MNIST data set.

Dataset	Classifier	**Accuracy**	**Precision**	**Recall**	**Time (s)**
CIFAR 10	SVM	0.287	0.28	0.29	862
	K-Nearest Neighbors	0.180	0.18	0.18	107
	Decision Tree	0.173	0.17	0.17	2.5
	Multi Layer Perceptron	0.255	0.25	0.25	388
MNIST	SVM	0.664	0.66	0.66	3600
	K-Nearest Neighbor	0.476	0.50	0.48	726
	Decision Tree	0.590	0.58	0.58	206
	Multi Layer Perceptron	0.609	0.61	0.61	919

Table 3.1: Classifiers' performance for the two data sets.

Data set	Extraction	Clustering	Train Hist	Test Hist	**Overall (s)**
CIFAR 10	47	42	357	76	**522**
MNIST	57	57	295	60	**469**)

Table 3.2: Preprocessing and feature extraction time (in seconds) over the two data sets.

From the images below we can visually understand what happened with the SIFT based classification and we can understand in which classes this method has some drawbacks.

From the images in figure 4.1 we can see how the overall performance was quite bad for all the chosen classifiers for the data set CIFAR-10. Although some algorithms (e.g. SVM) performed better than others we can see a common pattern in all the confusion matrices. In fact, for all the classifiers we have a large error in three particular classes: bird, cat and deer. This could be justified by the presence of common descriptors in the related images. In conclusion we can state that the performance for this data set did not exceed our expectations. Some probable reasons that misguided the algorithm could be the presence of too many different characteristics from the same class in the images (e.g. too much "augmentation") or the presence of different backgrounds. We can not say anything about the color influences since we know the SIFT is a monochrome invariant algorithm.

From the images in 4.2 we can see that the performance on the FashionMNIST data set was quite good with an homogeneous performance overall the different classes. The most confused class for all the algorithms was the "shirt" class (confused with similar classes as coat, pullover, top). Here the SIFT based approach confirmed our expectations, in fact without the presence of a background in the images and without too many different views for instance we had a reasonable performance.

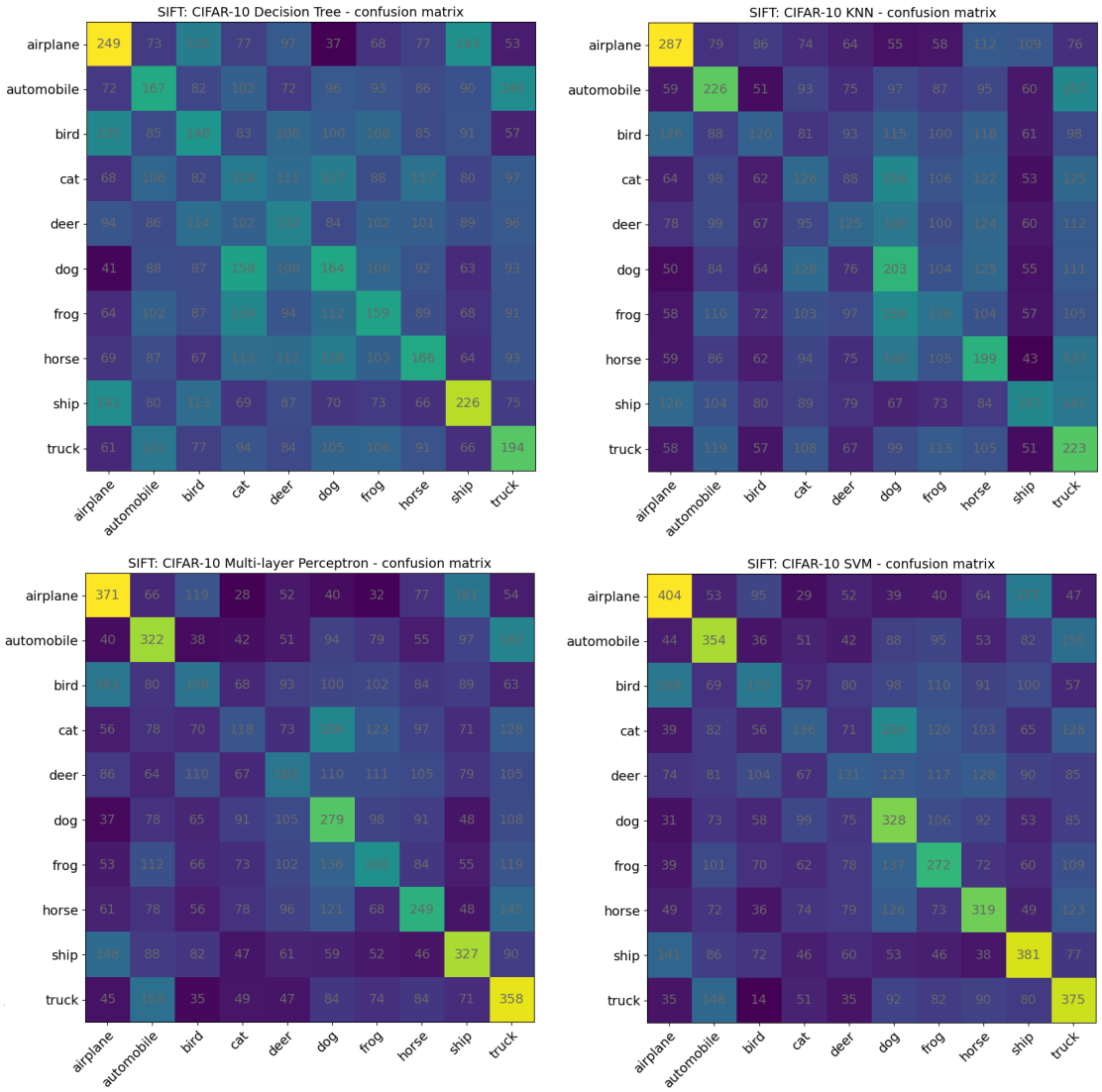

Figure 3.2: Confusion Matrices for the CIFAR-10 data set all four classifiers (SVM, KNN, Decision Tree, and MLP). The count for each classification is written inside the cells.

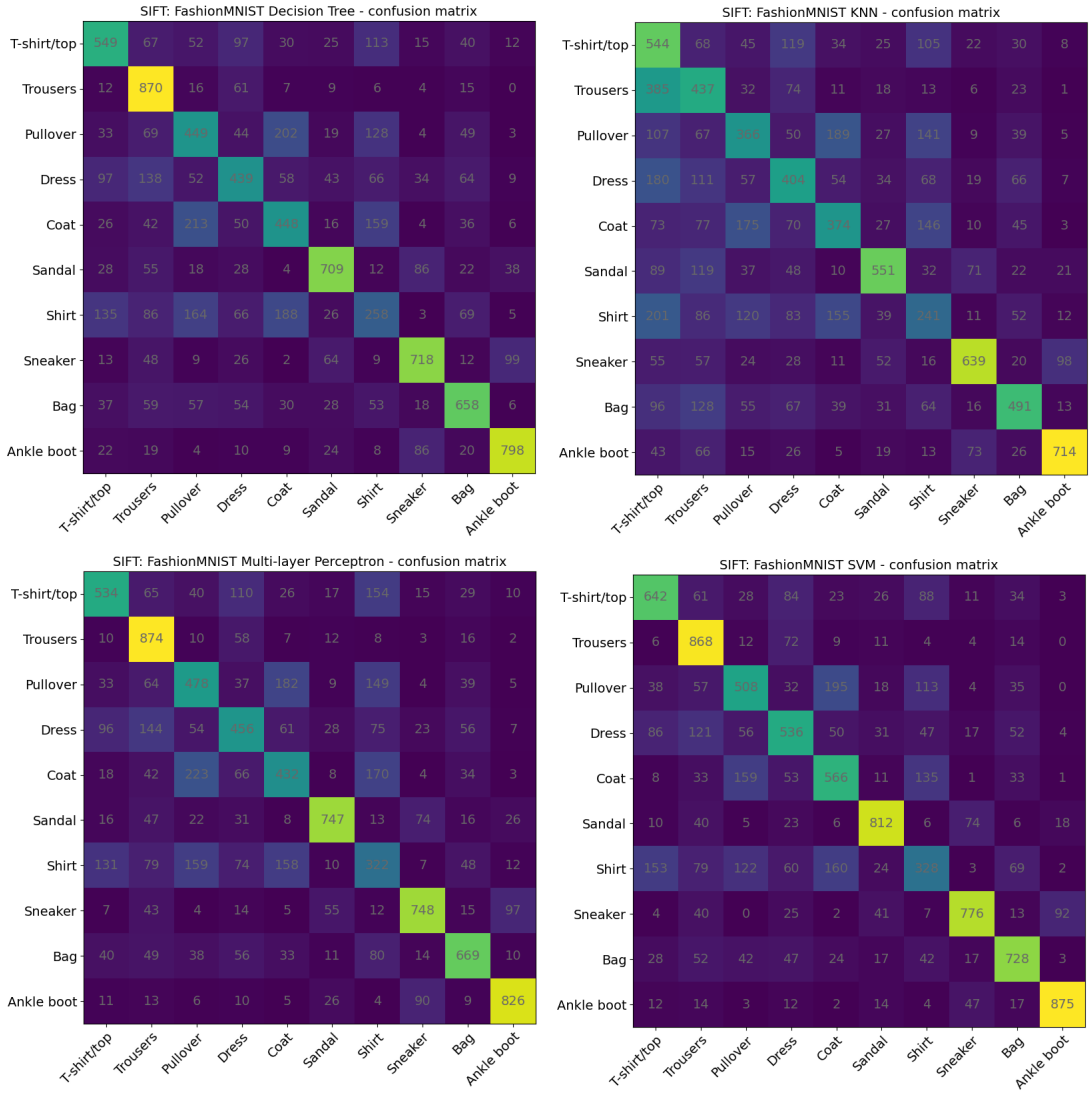

Figure 3.3: Confusion Matrices for the FashionMNIST data set all four classifiers (SVM, KNN, Decision Tree, and MLP). The count for each classification is written inside the cells.

Chapter 4

Color Histogram based Classification

Color histograms function as a very simple descriptor of images. The idea behind them is quite simple: Extract the color values of each pixel from each image for each color channel and compute a histogram that shows the frequency of each color value.

4.1 Computing the different Color Histograms

In the case of the images in the data sets used for this exercise each color has 256 intensities. The FashionMNIST data set only has grey scale images, which means there is only one color channel. The CIFAR-10 data set however has three color channels (RGB = red, green, and blue), with 256 possible values for each channel.
In order to test the color histogram approach we compute multiple different types of color histograms per image, using the set of descriptors generated with each method to independently train our classifiers. We always use the same methodology for the training set as well as the test set. The methods we use are taken from the provided script on the TUWEL page of this lecture and are as follows: For the images of the CIFAR-10 data set, with red, green and blue channels, we first compute the standard color histogram for each color, with 256 bins each. Those values are stored in the order red, green and blue in one single array for each individual image. The second method is identical to the first, but instead of one bin per color we now use only 64 bins, which leads to averages over multiple color values. The third method is to look at two color channels simultaneously (RB, RG, BG) for calculating the histogram. This leads to 3 new histograms, again stored in one array, per image that have 16 bins each that count how often each combination of two channels have the same frequency for the color values in each bin. The third method is the same as the previous one but now we use all three color channels with a bin size further reduced to 8, in order to keep the amount of data generated at a reasonable size. For the FashionMNIST data set that only has one color channel we simply calculate color histograms of the grey value channel with 256, 128, 64 and 32 bins. For the evaluation of this approach we used the same classifiers as for the SIFT based approach in the previous chapter (SVM, KNN, Decision Tree, and MLP). Overall computing the different color histograms is not very expensive with combines computation times in the range of a few seconds for all 4 histogram types per data set.

4.2 Results

Dataset	Classifier	**Accuracies**			
		256 bins	**64 bins**	**2 colors**	**3 colors**
	SVM	0.3521	0.3622	0.4189	0.4168
CIFAR 10	K-Nearest Neighbor	0.2671	0.2784	0.3223	0.3177
	Decision Tree	0.1983	0.2214	0.2675	0.2706
	Multi Layer Perceptron	0.2757	0.3115	0.3492	0.3768
		256 bins	**128 bins**	**64 binsl**	**32 bins**
	SVM	0.4965	0.4819	0.4649	0.4366
MNIST	K-Nearest Neighbors	0.4142	0.409	0.3916	0.3863
	Decision Tree	0.4021	0.3904	0.3771	0.3613
	Multi Layer Perceptron	0.528	0.5355	0.5233	0.4964

Table 4.1: Accuracies of each classifier for both data sets, using standard parameters. The different descriptors for CIFAR-10 are standard color histograms with 256 and 643 bins, as well as histograms using the combined values of two color channels (column "2 colors") as well as of all three colors (column "3 colors"). For the fashionMNIST data set only one color channel was availabel so the only difference between the histograms is the bin size (256, 128, 64 and 32 bins).

First we compared the performance of our four classifiers with standard parameters on each of the different types of color histograms to choose the best descriptor to perform hyper parameter optimization with. We chose not to do hyper parameter optimization for each type of color histogram due to the large computation times needed as a result of the amount of information in each data set. This lead to us choosing the color histogram that combines all three channels for the CIFAR-10 data set and the histogram with 256 bins for the FashionMNIST data set. The average accuracy for each data set and classifier, when using standard parameters, is shown in table 4.1. Next we performed hyper parameter optimization using these two descriptors. Using the optimized classifiers we performed a more detailed study on the results of each classifier for both data sets. Doing automated optimization with the average accuracy as a measure of quality we found the following hyper parameters for each classifier and data set to be better than the standard settings. The hyper parameters are written in the format needed to train the classifier with the *sklearn* python package.

CIFAR-10 Hyper parameters:
1) SVC(C = 10, gamma = 0.001)
2) DecisionTreeClassifier(criterion='gini', max_depth=10, max_features=None, min_samples_leaf=4, splitter='best')
3) KNeighborsClassifier(n_neighbors=50, p=1, weights='distance')
4) MLPClassifier(activation='relu', alpha=0.05, hidden_layer_sizes=(10, 30, 10), learning_rate='adaptive', solver='adam')

Dataset	Classifier	**Accuracy**	**Recall**	**Time (s)**
CIFAR 10	SVM	0.52	0.16	8199
	K-Nearest Neighbors	0.40	0.39	577
	Decision Tree	0.30	0.30	4
	Multi Layer Perceptron	0.35	0.36	177
MNIST	SVM	0.56	0.57	3053
	K-Nearest Neighbor	0.54	0.51	347
	Decision Tree	0.46	0.46	6
	Multi Layer Perceptron	0.54	0.56	306

Table 4.2: Accuracies, recall and time for training plus predictions on test set for each classifier for both data sets, using the optimized hyper parameters. The chosen descriptors are the combination of all three color channels for CIFAR-10 and the 256 bin histogram for FashionMNIST.

FashionMNIST Hyper parameters:
1) SVC(C = 1, gamma = 0.001)
2) DecisionTreeClassifier(criterion='gini', max_depth=10 max_features=None, min_samples_leaf=4, splitter='best')
3) KNeighborsClassifier(n_neighbors=50, p=1, weights='distance')
4) MLPClassifier(activation='relu', alpha=0.05, hidden_layer_sizes=(10, 30, 10), learning_rate='adaptive', solver='adam')

The results after the hyper parameter optimization can be seen in the form of the plotted confusion Matrices in figure 4.1 for the CIFAR-10 data set, and in figure 4.2 for the FashionMNIST data set. Average accuracy and recall, as well as computation times after the hyper parameter optimization can additionally be seen in table 4.2. After the hyper parameter optimization we can see an increase in average accuracy for each classifier. However due to limited computational resources and time, the parameter space was very limited (especially for the SVM classifier) and better settings might have been found otherwise. When looking at the confusion matrices in figures 4.1 and 4.2 we can see some interesting effects:
The very first thing to notice is that accuracy alone is not good enough to determine classifier performance during the hyper parameter optimization, which leads to the terrible results for the SVM with the CIFAR-10 data set. The average accuracy for this classifier was 0.52, because there were very few wrong classifications for most classes. However the reason for this is that almost all predictions were made for one class (dog). If we would have also considered the average recall (see table 4.2) we would have seen it has a value of only 0.16 and would have chosen different parameters for the SVM. There are also some more general remarks to be made in regards of using color histograms as descriptors for images: Some classes performed better using color histograms independently of classifiers. For the CIFAR-10 data set those were for example "frog", "airplane", and "automobile". However there were also some classes that were generally misclassified, for example the class "ship". This makes intuitive sense when looking at the color palette of the classes mentioned. Ships were often misclassified as airplanes, very likely due to the high content

of the color blue, while frogs were the only class that has a mostly green color palette and therefore was mostly classified correctly. For the FashionMNIST data set we can see that the classes "Trousers", "Sandals", and "Sneaker" were classified correctly most often. We assume that for sneakers and sandals the amount of white background was the main reason for this because they are the smallest items. Especially for sandals most of the image is white. The worst results were found for "shirts" and "bags", where shirts were often misclassified as t-shirts, pullovers or coats, which also makes sense when looking at size and colors of the items. In the end we were still surprised by how well the classification of grey-scale images worked with color histograms because the difference in coloring was perceived by us to be very small for most classes.

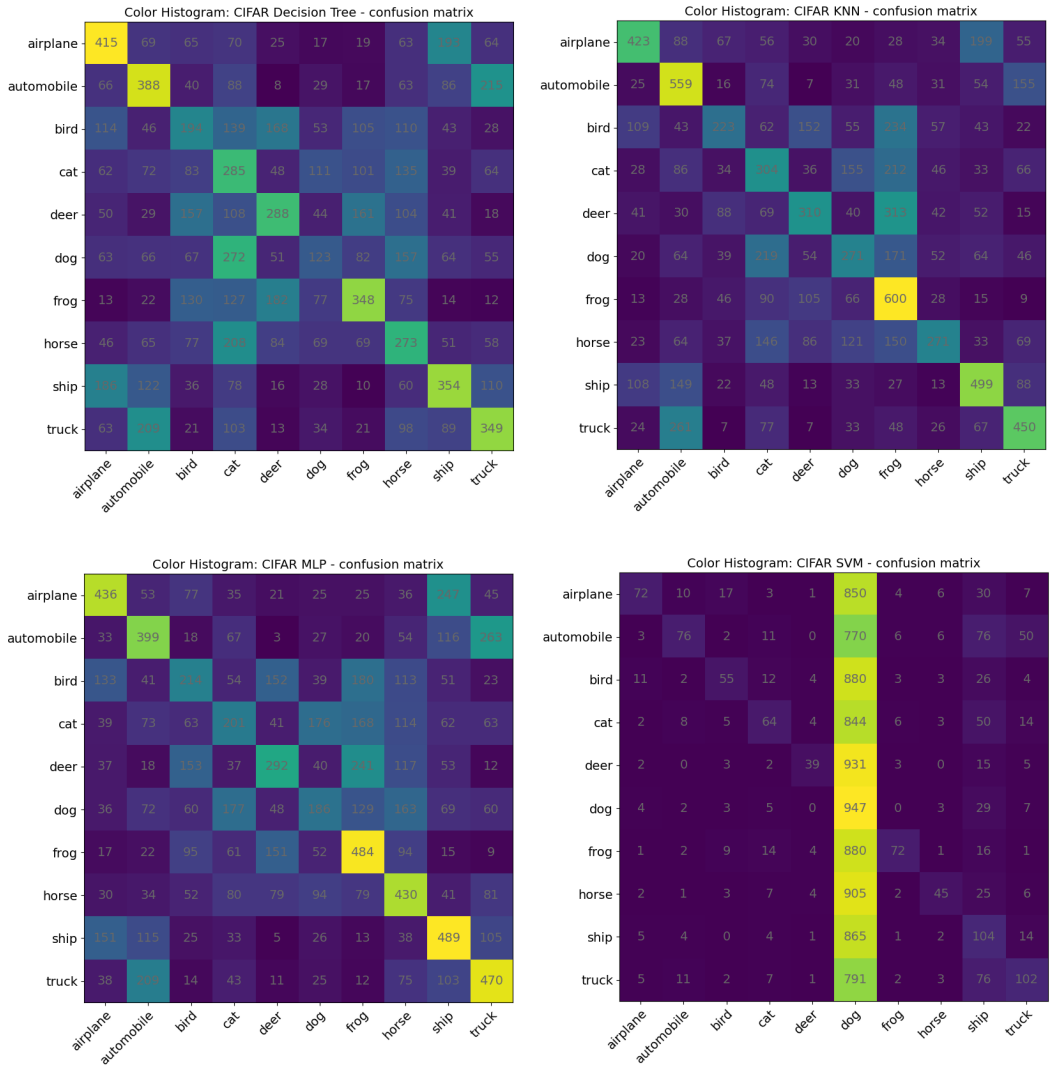

Figure 4.1: Confusion Matrices for the CIFAR-10 data set all four classifiers (SVM, KNN, Decision Tree, and MLP). The count for each classification is written inside the cells. Note that the confusion matrix for the SVM classifier shows that barely any predictions are correct, however this classifier still has a high average accuracy. This shows that accuracy alone is not enough to measure classifier performance.

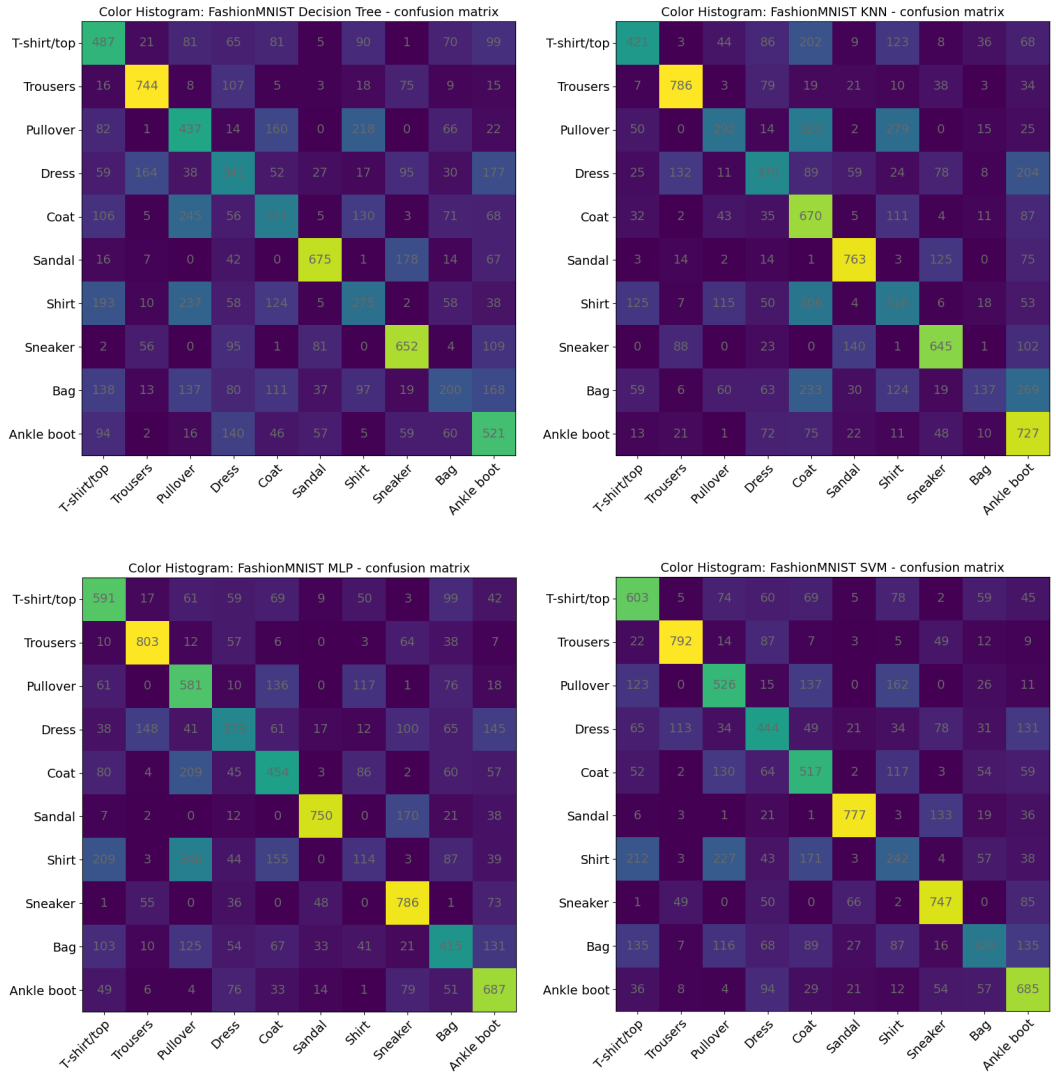

Figure 4.2: Confusion Matrices for the FashionMNIST data set all four classifiers (SVM, KNN, Decision Tree, and MLP). The count for each classification is written inside the cells.

Chapter 5

Convolutional Neural Networks

Image classification is the process of extracting and interpreting information from digital data. This process is included in Computer Vision's scientific-technological area, one of the main sub-areas of Artificial Intelligence. The main objective of this process is to emulate human vision and thus be able to interpret images. This task can be accomplished through the use of artificial neural networks and, in this document, results of two types of networks will be presented and compared, they are Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN).

The benchmark is performed on two different data sets, the FashionMNIST and the CIFAR10. This difference in complexity includes the insertion of colours and the number and variety of images. For each phase, the learning graphs and the results obtained will be presented, and through these, the efficiency of the models used will be compared. These phases will all be tested equally with 15 seasons so that the comparison is fair. All the architectures used in each network are attached in the main folder of this project, and in this document, the architectures and hyper-parameters of the data sets will be discussed. During the document, metrics such as accuracy, loss and the processing time of each model will be taken into account.

5.1 FashionMNIST

FashionMNIST is a data set of images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 greyscale image, associated with a label from 10 classes. It has only black and white colour, but it is an excellent data set to test the created networks' efficiency due to its greater complexity. To compare the efficiency of CNNs against MLPs, we tested both networks on the same data set. For CNNs, we tested four architectures with different complexities. These are presented in the next section. Then the learning curves of the various networks are presented, and at the end, the results are discussed, explaining the variation of the learning curves. We also tested data augmentation with smooth values.

5.1.1 Architectures

Simple:

- Convolution. Input = 28x28x1.
- SubSampling (Max Pooling).
- Dropout.
- Fully Connected #1. Output = 128.
- Output 10.

Plus:

- Convolution #1. Input = 28x28x1.
- SubSampling (Max Pooling) #1.
- Batch Normalisation #1.
- Convolution #2.
- SubSampling (Max Pooling) #2.
- Batch Normalisation #2.
- Dropout #1.
- Fully Connected #1. Output = 128.
- Dropout #2.
- Fully Connected #2. Output = 50.
- Output 10.

***Lenet5*:**

- Convolution #1. Input = 28x28x1.
- SubSampling (Max Pooling) #1.
- Convolution #2.
- SubSampling (Max Pooling) #2.
- Fully Connected #1. Output = 256.
- Fully Connected #2. Output = 84.
- Output 10.

Plusplus:

- Convolution #1. Input = 28x28x1.
- Convolution #2.
- SubSampling (Max Pooling) #1.
- Batch Normalisation #1.
- Dropout #1.
- Convolution #3.
- Convolution #4.
- SubSampling (Max Pooling) #2.
- Batch Normalisation #2.
- Dropout #2.
- Convolution #5.
- Convolution #6.
- SubSampling (Max Pooling) #3.
- Batch Normalisation #3.
- Dropout #3.
- Fully Connected #1. Output = 512.
- Dropout #2.
- Output 10.

5.1.2 Learning

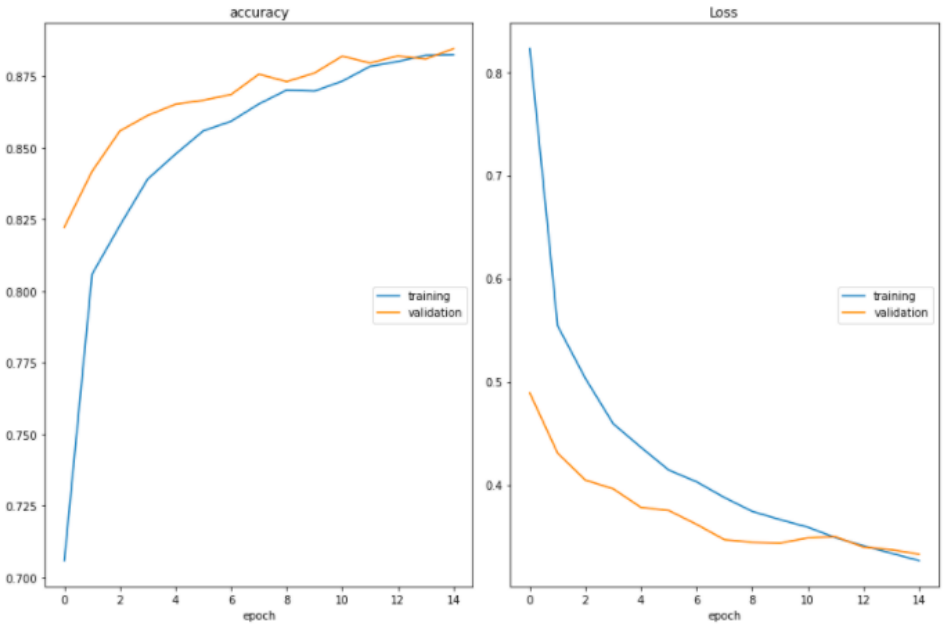

Figure 5.1: Evolution of Accuracy and Loss in FashionMNIST using MLPs

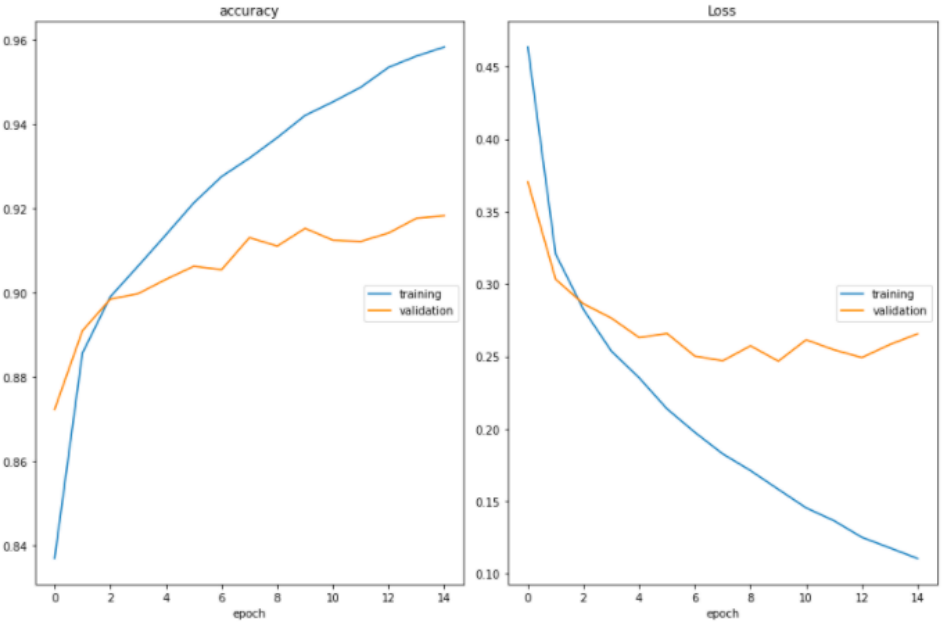

Figure 5.2: Evolution of Accuracy and Loss in FashionMNIST using CNNs (architecture - simple)

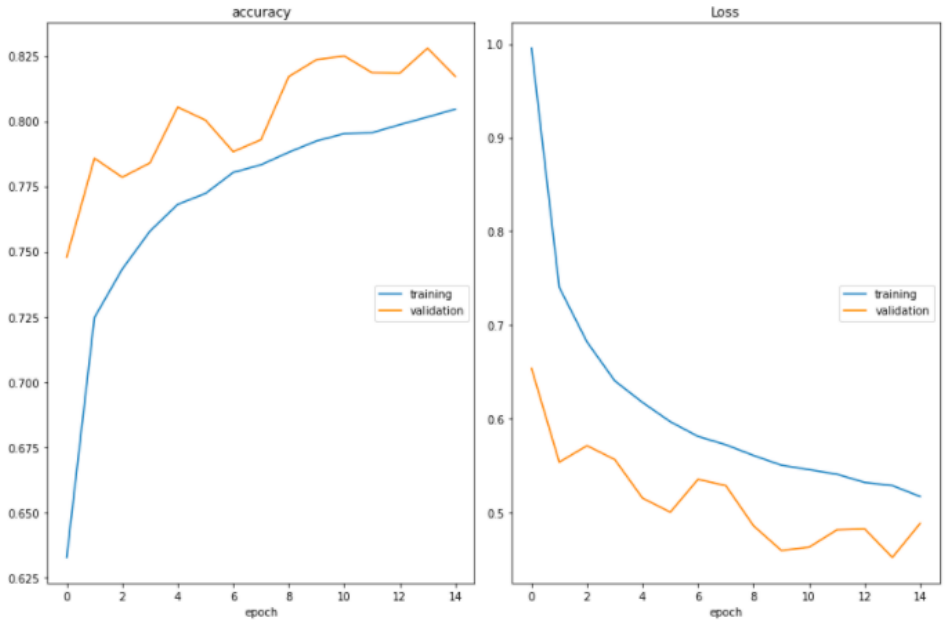

Figure 5.3: Evolution of Accuracy and Loss in FashionMNIST using CNNs and applying Data Augmentation (architecture - simple)

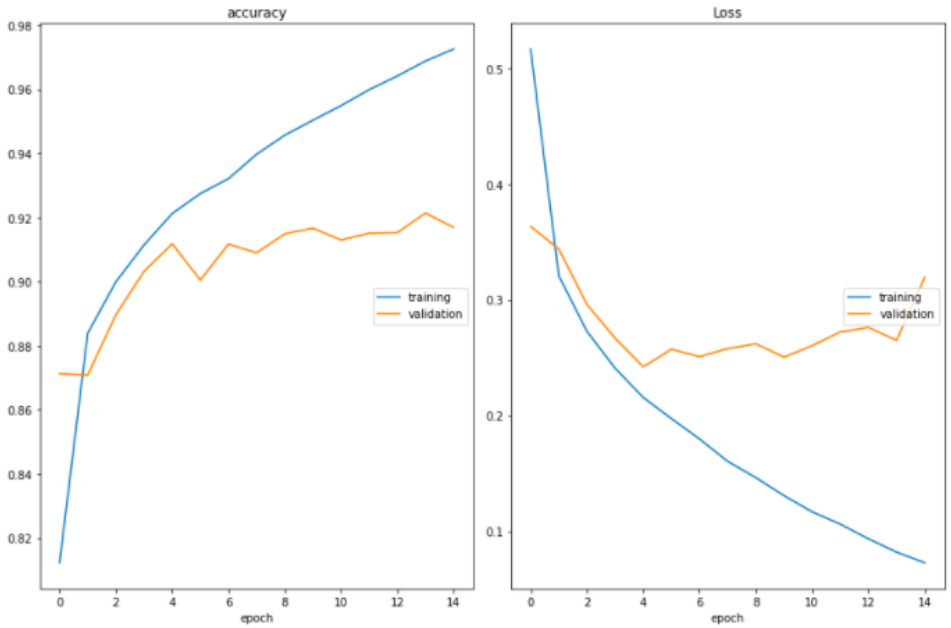

Figure 5.4: Evolution of Accuracy and Loss in FashionMNIST using CNNs (architecture - *Lenet5*)

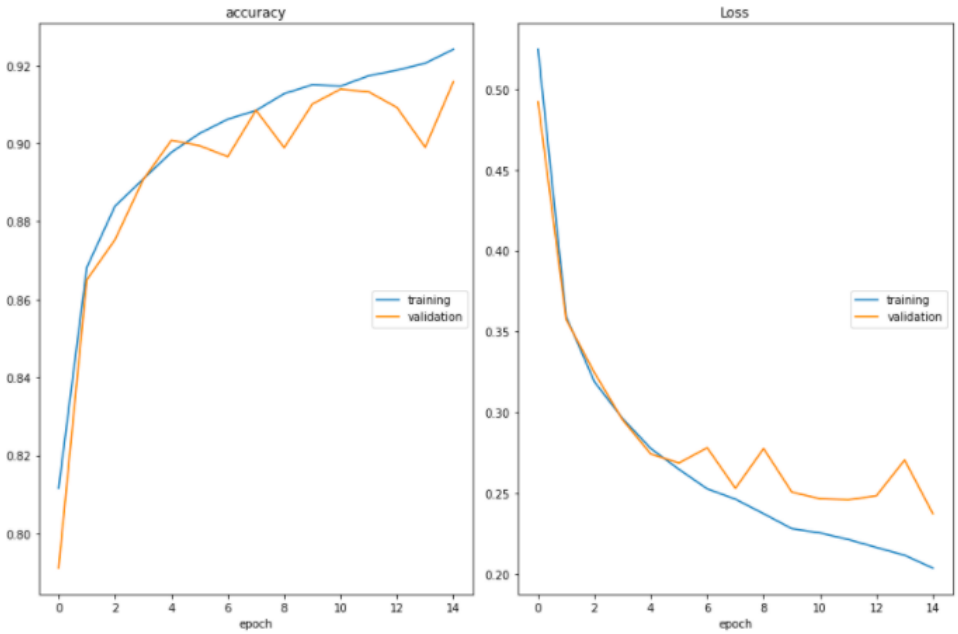

Figure 5.5: Evolution of Accuracy and Loss in FashionMNIST using CNNs (architecture - plus)

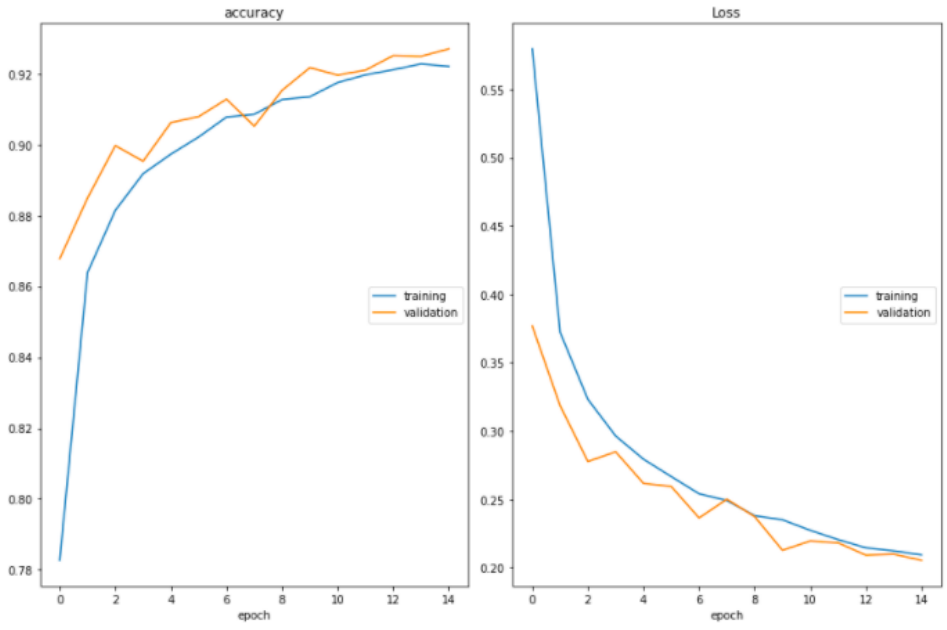

Figure 5.6: Evolution of Accuracy and Loss in FashionMNIST using CNNs (architecture - plus plus)

5.1.3 Results

Architecture	Data Augmentation	Accuracy	Loss	Processing Time
MLP	No	88.46%	0.333	74.88 s
CNN - Simple	No	91.82%	0.266	439.44 s (∼7 min)
CNN - Simple	Yes	81.72%	0.489	646.90 s (∼11 min)
CNN - *Lenet5*	No	91.69%	0.319	1046.09 s (∼17 min)
CNN - Plus	No	91.58%	0.237	730.83 s (∼12 min)
CNN - Plus Plus	No	92.70%	0.205	2573.60 s (∼42 min)

Table 5.1: Results after 15 epochs

5.1.4 Conclusions

In this data set, MLPs are able to perform similarly to CNNs, with a much shorter processing time, which is a favorable point. Still, if we want a little more accuracy, introducing CNNs and some complexity, we get better results (but need to invest more time).

Regarding the learning curves, it is crucial to avoid overfitting the sample data. In MLPs this is not felt. However, in the simplest architecture of CNNs, overfitting is visible and should be significantly penalized, as we are not learning the original distribution of the data, but rather memorize the training data. The *Lenet5* architecture has similar behaviour in this matter. On the other hand, more complex architectures deal with this obstacle much better. This greater efficiency in reducing overfitting in the complex architectures of CNNs is caused by the layers of *dropout* and *pooling*, which decrease the complexity of the features in the data.

The data augmentation process did not show better results, even though the defined parameters were smooth. A large amount of data can explain this decrease in results that the data set already has. In this sense, the increase in data implies a more significant variance and does not present itself as positive for the final results. In the graph of this experiment, it is possible to visualize an underfitting on the training data. Furthermore, there is an expectation of understanding what would happen with more than 15 epochs.

5.2 CIFAR-10

This data set uses features related to colour, more specifically three colour channels, leading to a significant increase in the amount of data and complexity. The benchmarking process is similar to that carried out in the previous data set. For testing the MLPs, we made it possible to create a grey-scale from the three colour channels.

5.2.1 Architectures

Regarding the architectures, we apply the same ones we did for FashionMNIST, only changing the values of entry into the network due to the size of the images and the colour

channels. The idea of using the same networks is in the sense of being able to have a better point of comparison between grey-scale images and colour images.

5.2.2 Learning

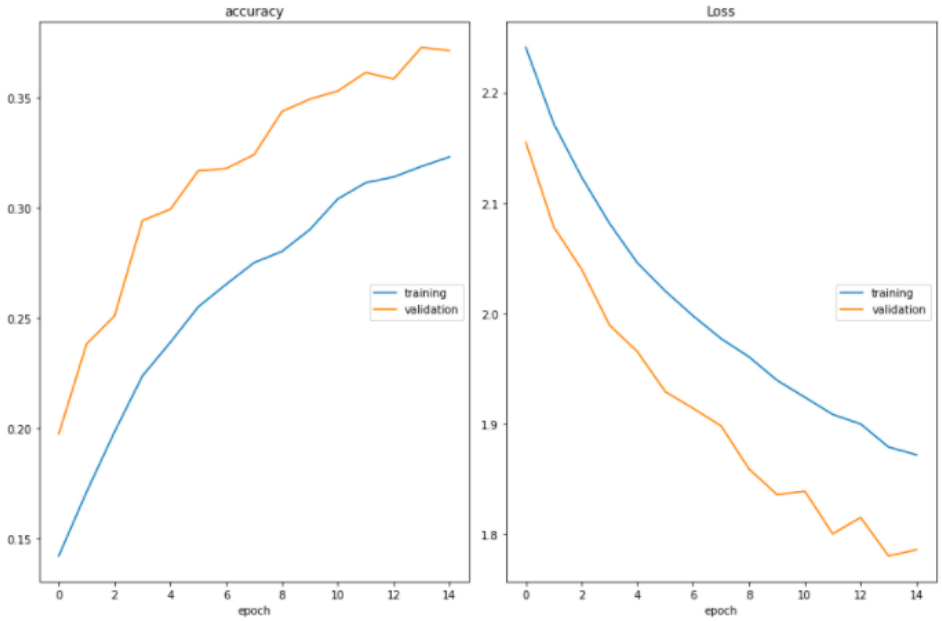

Figure 5.7: Evolution of Accuracy and Loss in CIFAR using MLPs

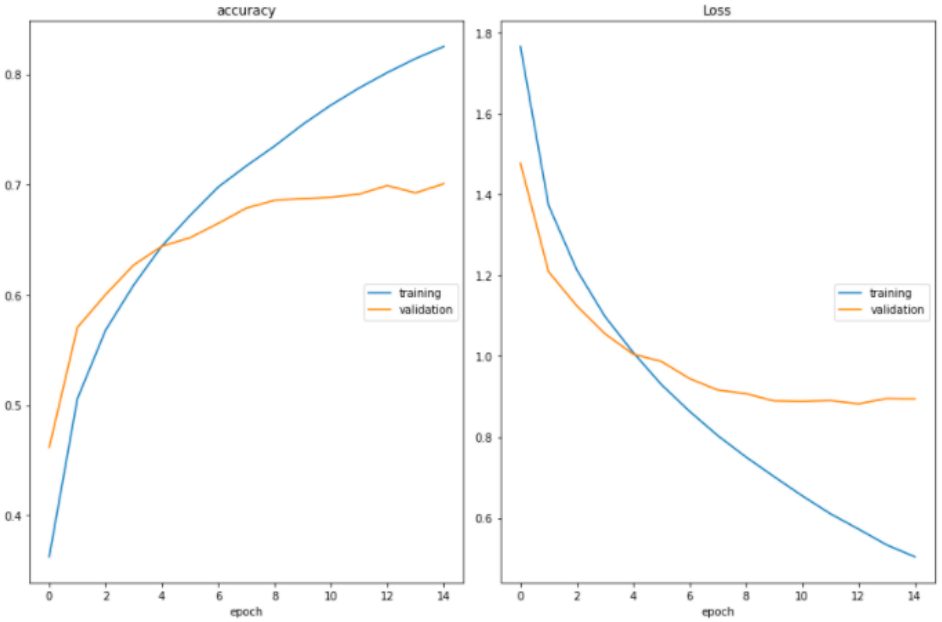

Figure 5.8: Evolution of Accuracy and Loss in CIFAR using CNNs (architecture - simple)

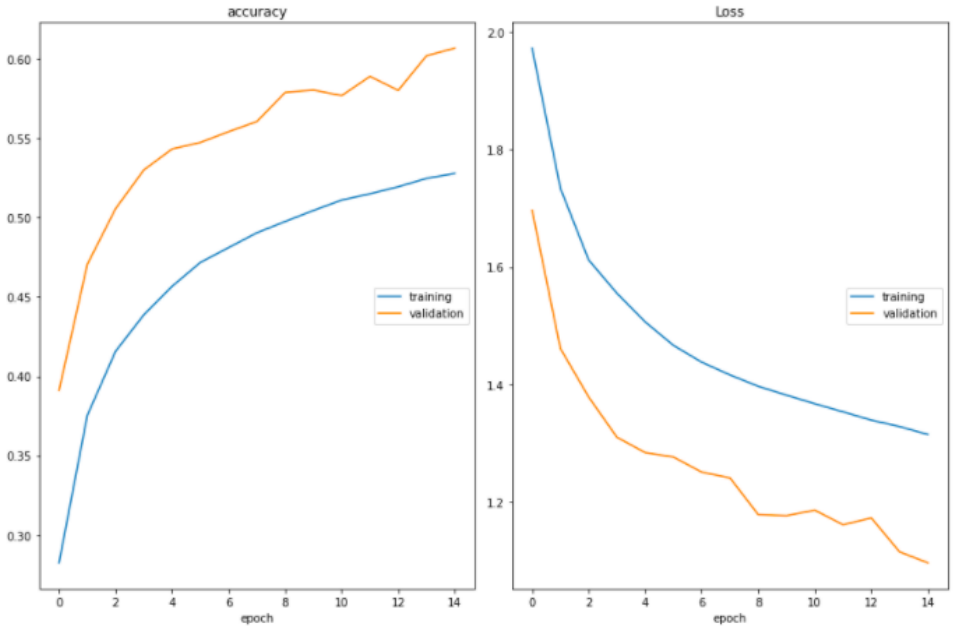

Figure 5.9: Evolution of Accuracy and Loss in CIFAR using CNNs and applying Data Augmentation (architecture - simple)

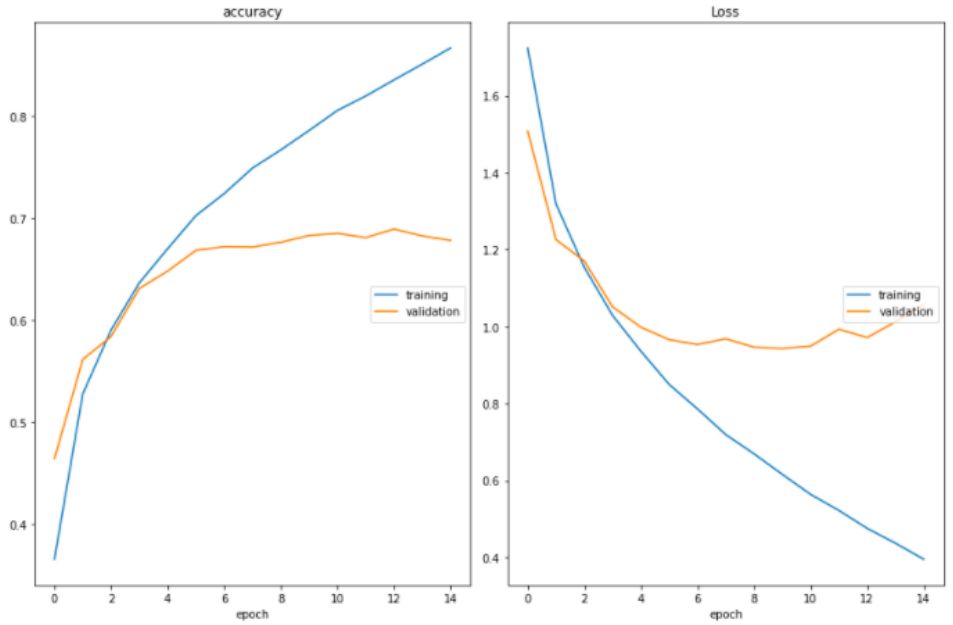

Figure 5.10: Evolution of Accuracy and Loss in CIFAR using CNNs (architecture - *Lenet5*)

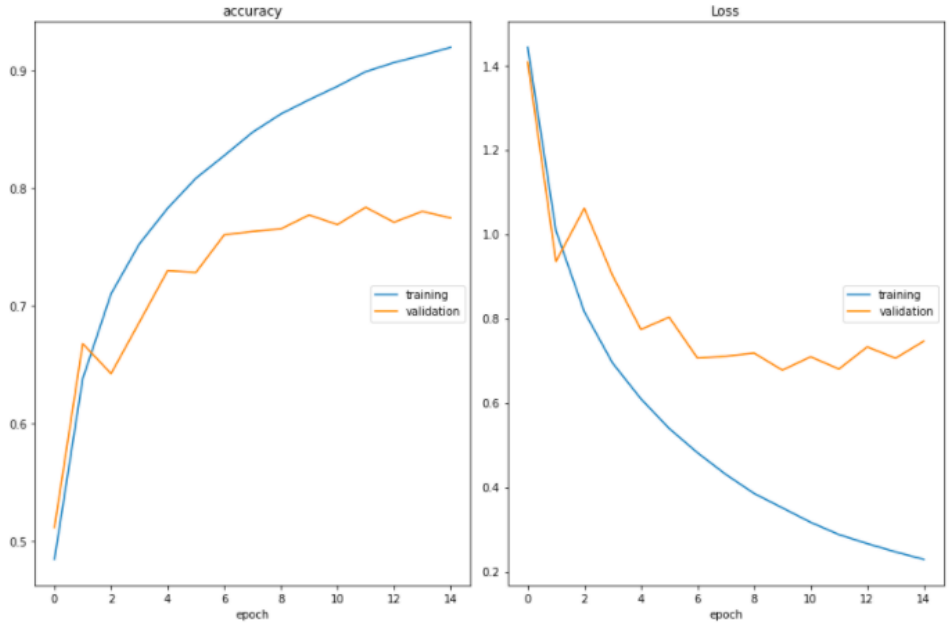

Figure 5.11: Evolution of Accuracy and Loss in CIFAR using CNNs (architecture - plus)

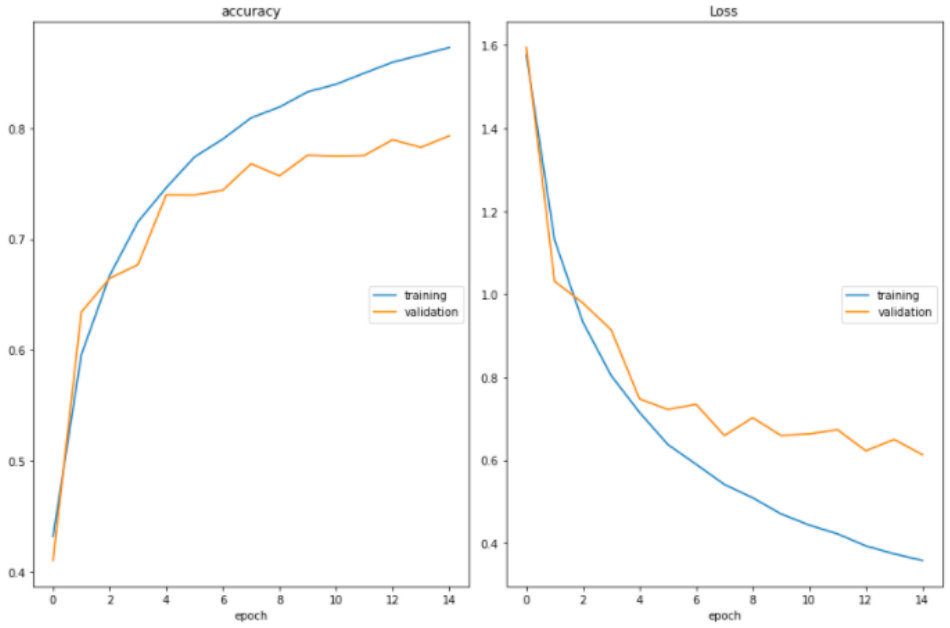

Figure 5.12: Evolution of Accuracy and Loss in CIFAR using CNNs (architecture - plus plus)

5.2.3 Results

Architecture	Data Augmentation	Accuracy	Loss	Processing Time
MLP (greyscale)	No	37.15%	1.786	88.15 s
CNN - Simple	No	70.07%	0.895	2094.20 s (∼35 min)
CNN - Simple	Yes	60.66%	1.097	2059.43 s (∼34 min)
CNN - *Lenet5*	No	67.80%	1.058	1083.34 s (∼18 min)
CNN - Plus	No	77.46%	0.746	3297.38 s (∼54 min)
CNN - Plus Plus	No	79.29%	0.613	4035.80 s (∼67 min)

Table 5.2: Results after 15 epochs

5.2.4 Conclusions

In this data set, concerning MLPs, we present the grey-scale value because, without it, the values were much worse, confirming the MLPs' inability to deal with colour images. In this data set, there is a much more significant difference in accuracy between MLPs and CNNs. It follows, therefore, the greater ability of CNNs to deal with the classification of colour images.

The data augmentation procedure did not show better results than for the previous data set, with the justifications for this happening being the same. However, the expectation of understanding what would happen after 15 epochs remains valid.

Also in this data set, it is possible to view overfitting in the simplest architectures and *Lenet5*. Likewise, this effect should be penalized. In such a way, the most complex networks were tested, despite introducing more convolutions. They also introduce layers that help combat overfitting to training data, thus better generalizing the final result in the original distribution data. It is important to notice the continuous evolution of the curves, which leads to the conclusion that with more training seasons the result would certainly reach higher values.

In general, the accuracy in this data set is smaller than the previous one, and this is normal due to the greater complexity presented in this data set, both in colours and in the amount of data. This was predictable considering the use of the same network architecture.

5.3 Featuremaps Visualization

For the development of CNN network architectures, the visualization of *featuremaps* becomes a very advantageous technique as it makes it possible to understand what happens in the training process, specifically in each layer. The effect of these and the utility they have for the data in question is capabilities that the visualization of the *featuremaps* provides to anyone who tries to find the best possible model for the classification of images in a given data set. In a detailed way, it is possible to visualize the effects of the *convolution*, *pooling* and *batch normalization* layers, which present themselves as an essential part in the concurrence of any Convolutional Neural Network.

These layers create and change the *featuremaps* during the training process. The essential features of the images are highlighted in the convolution layer. In the *pooling* layer, the image is reduced, highlighting only the most relevant features. Afterwards, this process is repeated, but for a smaller number of *featuremaps*, with fewer pixels and activation channels, the result of previously performed processes. What is noticeable in these phases is the refinement of the features that make up the images. *Featuremaps* with negative values are not activated. The effects of each layer are shown in the following figures.

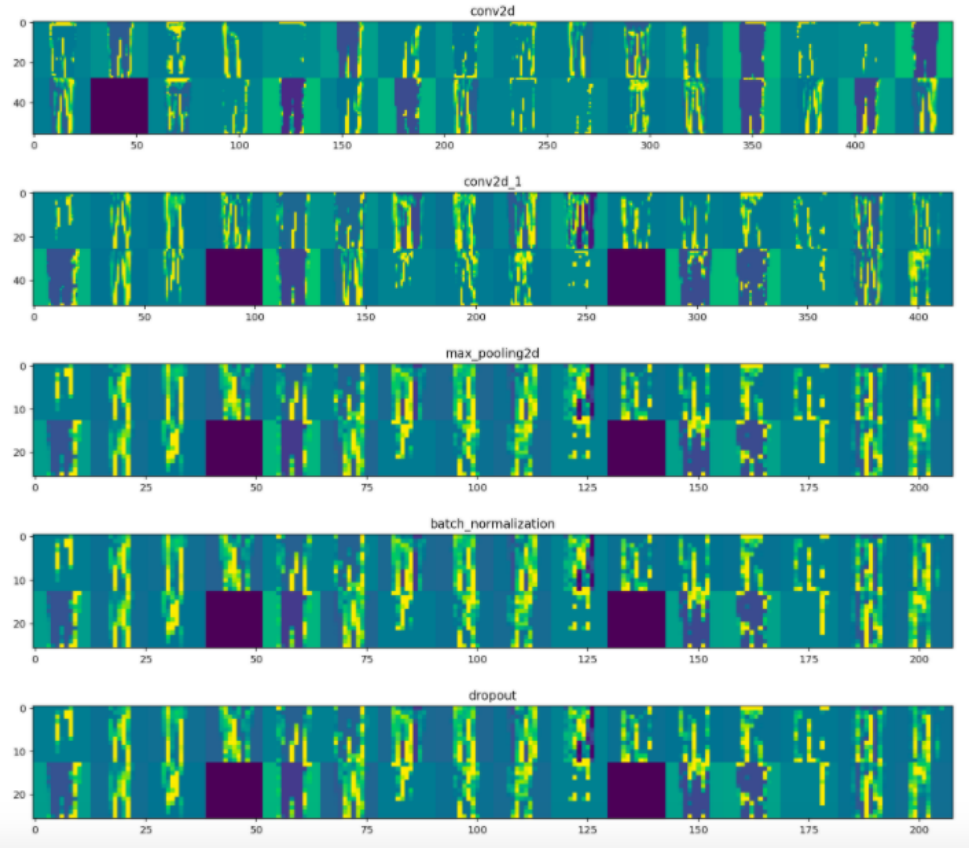

Figure 5.13: Featuremaps Visualization in FashionMNIST (Pants)

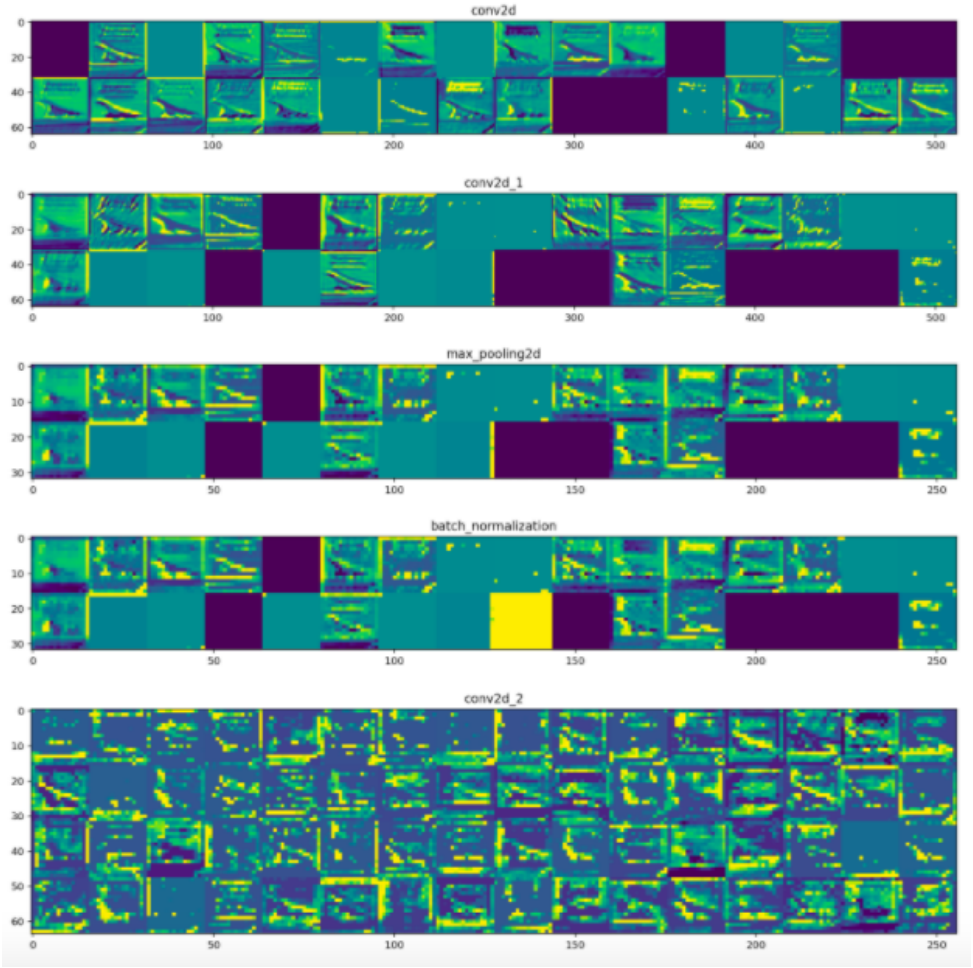

Figure 5.14: Featuremaps Visualization in CIFAR (Frog)

Chapter 6

Conclusion

From the performed analysis we can state that the SIFT based approach had a good performance on the FasionMNIST data set and had some drawbacks on the CIFAR-10 set. The accuracies were between 0.17 and 0.67. This happens because the characteristics of the images are quite different (e.g. background noise, orientation of subjects). We also identified which classes were confused by the classifiers and found some similar patterns over all the confusion matrices. Thus, we understood that the SIFT based approach has got some drawbacks that can lead to very bad performance for some classes. The running time was reasonable, the creation of the training/test set histograms took around 10 minutes for both the data sets. This places the BoVW system between the Colour Histogram and the CNN approach in terms of both performances and computational time.

In general the color histograms classifiers had surprisingly good results (accuracies between 0.35 and 0.56 for all different data sets and classifiers) when considering how limited the information content is for each image. Only information about colors and their frequency is used. It should aslo be mentioned, that computing the color histograms was very fast while the training of the different classifiers took longer than for the SIFT approach. However we found that this can also lead to obvious problems, e.g. when comparing an airplane in blue the sky and a ship in the blue ocean, where both images mainly have the colors blue and maybe white for clouds or sea foam. This leads to the inability to distinguish the two classes, which was also proven by the results of our classifiers using this descriptor.

From this document and related research, it is crucial to verify the importance that hardware has in the execution of CNNs, due to the large amount of data trained and tested in the typically complex architectures in this type of network. This point was a limitation in carrying out tests because due to the lack of a GPU it was not possible to apply other libraries with better performance and, consequently, it was not possible to generate many tests for comparison or to be able to visualize the training process at a good pace.

In MLP networks, the types of layers usually used are activation and also *dropout* and *batch normalization*, where there is not so much complexity at this level. In CNNs, it makes sense to highlight the use of more types of layers, such as *convolution* and *pooling*, each with different functions and advantages within the network, which have already been clarified. In this way, the layers that CNNs typically have in their architecture were

addressed and acquired the knowledge involved.

In this project, the evolution of the networks was mainly carried out by increasing the network's depth to carry out a greater refinement of the features. That said, it is essential to note that both architectures can be evolved in other ways in order to produce better results because, for example, there is some frequency of *featuremaps* that are not activated. In this sense, a possible change would be to increase the input size of the images, which in these tests was 32x32 pixels, but which, with a higher resolution, is expected to provide the networks with more definition in the image features and, consequently, they have better results.

Finally, we conclude that CNNs do have better results than MLPs in image classification. They are more complex and perform slower. However, the results are much more efficient and, consequently, much more useful in interpreting and classification images.