# Machine learning report (Group 19)
# Exercise 3.1: Deep Learning

Ana Oliveira da Costa (01624245)
Anastassiya Pustozerova (11832651)
Sarah Hoppe (11831524)

February 24, 2019

## Introduction

We chose task 3.1 for the third assignment and compared Deep Learning for image classification with the combination of *traditional* classifiers and advanced feature extraction.

**Datasets** As data sets we used the FIDS30 fruits data set, which consists of 971 colour images showing 30 different types of fruits, and the car detection dataset on the *car present or not* task. We took the training data set into account, which consists of 1050 grey images (with 500 images belonging to one class and 550 to the other) and splitted it in a training and validation set.

***Traditional* classifiers and feature extraction** As features for the images we computed colour histograms and combined them with features extracted by the SIFT algorithm. Based on the features extracted by SIFT, a Bag of Visual Words was computed and features were then standard scaled. We chose the classifiers: MLP, Random Forest and k-NN.

**Deep Learning** For Deep Learning we tried six different CNN architectures for the car data set and three different architectures for the fruits data set. For both data sets we additionally tried a simple fully-connected NN and all CNN architectures with data augmentation.

**Setup** We used for classification, feature extraction and deep learning Python 3.7.2 with Numpy 1.16.1, Scikit-learn 0.20.2, Matplotlib 3.0.2, Pillow 5.4.1, Theano 1.0.4, Keras 2.2.4 and a self-built version of OpenCV 4.0.0, which includes non-free packages like SIFT. For the implementation of the Bag of Visual Words we oriented ourselves towards the code of Kushal Vyas[1].

[1] https://kushalvyas.github.io/BOV.html

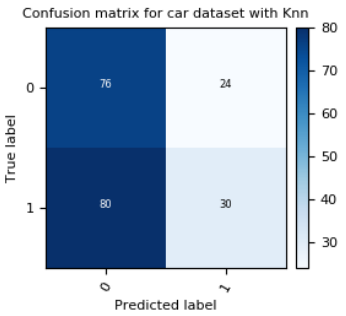
# 1 Car Dataset

## 1.1 *Traditional* classifiers and feature selection

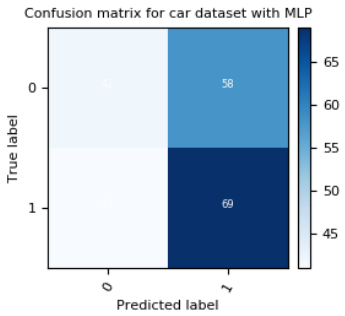
**Protocol** We splitted the data set in a stratified training set of 80% and a validation set of 20%. With a grid search we tried different parameters for MLP, Random Forest and k-NN and cross-validated the classifiers with three stratified folds to find the best parameter settings.
As a subset for hyperparameter optimization for k in k-NN we tested k= 3, 5, 7, 10, 20 and 30. Furthermore we tested uniform weighted neighbours and neighbours weighted based on their distance. For the Random Forest we optimized the number of decision trees and tested 3, 5, 7, 10, 15, 20, 25, 30, 35 and 50. For the MLP we took a single hidden layer and optimized the number of nodes in the layer, the activation function and the solver. For the number of nodes we tested 100, 120, 150, 200, 250, 300, 500, 700, 1000, 1250, 1500, 3000 and 5000, for activation functions identity, logistic, tanh, ReLU function, for solvers LBFGS, SGD and Adam.
Then we computed the confusion matrix and the accuracy for the validation set.

**Results** Testing different k for k-NN, all parameter settings yielded a mean accuracy above 0.46. For validation we took k=3, which yielded the best results with a mean accuracy of 0.489 with a standard deviation of 0.024, and uniform weighted neighbours. These settings yielded during validation an accuracy of 0.505.
As to be expected for the hyperparameter optimization of the random forest, the highest tested number of trees (50) yielded the best results, which had a mean accuracy of 0.743 with a standard deviation of 0.012. On the validation set this setting achieved an even higher accuracy of 0.767
For the MLP the lowest mean accuracy while testing different number of nodes was achieved by 150 with a mean accuracy of 0.569 and a standard deviation of 0.012. The highest mean accuracy was achieved by 1500 with 0.641 with a standard deviation of 0.011. Further, best results were yielded with the identity activation function and SGD as a solver. however, on the validation set these settings achieved an accuracy of 0.529.
The confusion matrices for all three classifiers on the validation set can be found in figure 1. It can be seen that k-NN has a high number of false positives whereas the MLP has a higher number of false negatives.
In table 2 are the training, testing and validation run times for each classifier listed and in table 3 the MLP run times in comparison with the feature extraction run time. Computing the Bag of visual Words was what took overall the longest.

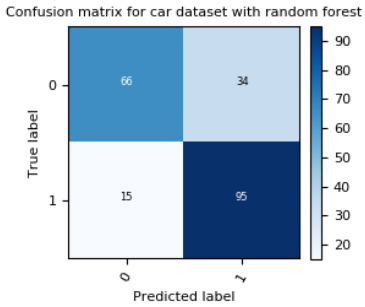
[2]Colour histogram and SIFT




(a) k-NN




(b) MLP




(c) Random Forest

Figure 1: Confusion Matrices for *traditional* classifiers for the car data set

| | k-NN | MLP | Random Forest |
|---|---|---|---|
| Accuracy | 0.505 | 0.529 | 0.767 |

Table 1: Accuracy on the validation set for the car data set for the *traditional* classifiers

| | k-NN | MLP | Random Forest |
|---|---|---|---|
| Training (mean) | 0.002 | 5.471 | 0.062 |
| Testing (mean) | 0.068 | 0.006 | 0.003 |
| Validation | 0.051 | 3.663 | 0.151 |

Table 2: Training, testing and validation run times (in seconds) for the *traditional* classifiers for the car data set

| | Car | Fruits |
|---|---|---|
| Feature Extraction[2] | 4 | 520 |
| Bag of visual Words | 34 | 2048 |
| Training (MLP mean) | 5 | 4 |
| Validation (MLP) | 4 | 19 |
| Overall below | 47 | 2591 |

Table 3: Run times (in seconds) for feature extraction, training and validation and overall run time for Car and Fruits data set.

## 1.2 Deep Learning

**Protocol** First we made experiments with simple fully-connected network, with architecture from the figure 2

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_14 (Dense) | (None, 256) | 1024256 |
| dense_15 (Dense) | (None, 256) | 65792 |
| dense_16 (Dense) | (None, 1) | 257 |

Figure 2: Architecture of the simple Fully-connected network for Car data set

The we tried six different CNN architectures. *CNN0* is from the tutorial by Thomas Lidy[3]. In this design we have 2 convolution layers and one dense layer. Each convolution layer has a rectified linear activation (ReLU), followed by max pooling and 30% dropout. Although it is not specified in the tutorial, this design seems to follow the LeNet architecture [1].

The *CNN1* design is *CNN0* without dropout function. The dropout randomly switches off neurons, which prevents the overfit of the network. *CNN2* is another variation of *CNN0*. In this design we do not include the ReLU activation. The goal of these designs is to test the impact of both the dropout and the ReLU activation in the quality of our predictions.

The *CNN3*, summarized in fig. 4, has two convolution layers, with max pooling and a dropout of 25%, followed by a full layer. The main difference of this design w.r.t. the previous is a different number of filters and a different kernel size in each convolution layer.

Finally, *CNN4* and *CNN5* have both three convolution layers with an increase of number of filters and kernel size. There are two main difference between them: in *CNN4* we only consider kernel sizes of $3 \times 3$ and $5 \times 5$, while in *CNN5* the last convolution layer has kernel of size $7 \times 7$; and in *CNN5* we have two dense functions in the last layer, instead of one like all the other designs.

[3] `https://github.com/tuwien-musicir/DL_Tutorial/blob/master/Car_recognition.ipynb`

Additionally to the testing on the split car data set, we also tested the Deep Learning architectures on the car test data set.

[CNN2]
Model summary:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_5 (Conv2D) | (None, 38, 98, 16) | 160 |
| batch_normalization_3 (Batch | (None, 38, 98, 16) | 64 |
| max_pooling2d_3 (MaxPooling2 | (None, 19, 49, 16) | 0 |
| dropout_4 (Dropout) | (None, 19, 49, 16) | 0 |
| conv2d_6 (Conv2D) | (None, 17, 47, 16) | 2320 |
| dropout_5 (Dropout) | (None, 17, 47, 16) | 0 |
| flatten_3 (Flatten) | (None, 12784) | 0 |
| dense_5 (Dense) | (None, 256) | 3272960 |
| dropout_6 (Dropout) | (None, 256) | 0 |
| dense_6 (Dense) | (None, 1) | 257 |

Figure 3: CNN2 Architecture for Car data set

[CNN3]
Model summary:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_7 (Conv2D) | (None, 40, 100, 48) | 480 |
| max_pooling2d_4 (MaxPooling2 | (None, 20, 50, 48) | 0 |
| dropout_7 (Dropout) | (None, 20, 50, 48) | 0 |
| conv2d_8 (Conv2D) | (None, 16, 46, 62) | 74462 |
| max_pooling2d_5 (MaxPooling2 | (None, 8, 23, 62) | 0 |
| dropout_8 (Dropout) | (None, 8, 23, 62) | 0 |
| flatten_4 (Flatten) | (None, 11408) | 0 |
| dense_7 (Dense) | (None, 256) | 2920704 |
| dense_8 (Dense) | (None, 1) | 257 |

Figure 4: CNN3 Architecture for Car data set

**Results** In table 4 can be seen all the accuracy on the validation set for all Deep Learning architectures with and without data augmentation. Accuracy does not vary much for these two cases. CNN1 is the architecture which performs best in both cases.
The mean training time without augmentation over all architectures was 123.52 seconds and with augmentation 173.54. The mean testing time without augmentation was 0.84, with augmentation 1.50.

CNN4 is the architecture which yielded best results without augmentation and CNN2 is the one which had highest accuracy with augmentation.

On the figure 6 one can see confusion matrix for CNN4 and CNN2 with and without data augmentation. From the history of training the CNN2 and CNN4

[CNN4]
Model summary:

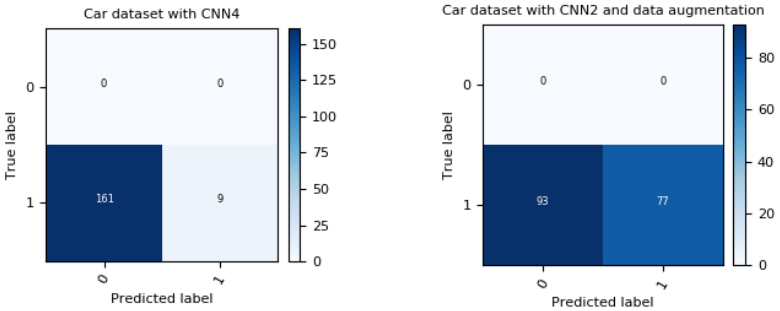
| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_9 (Conv2D) | (None, 40, 100, 48) | 480 |
| max_pooling2d_6 (MaxPooling2 | (None, 20, 50, 48) | 0 |
| dropout_9 (Dropout) | (None, 20, 50, 48) | 0 |
| conv2d_10 (Conv2D) | (None, 16, 46, 62) | 74462 |
| max_pooling2d_7 (MaxPooling2 | (None, 8, 23, 62) | 0 |
| dropout_10 (Dropout) | (None, 8, 23, 62) | 0 |
| conv2d_11 (Conv2D) | (None, 4, 19, 68) | 105468 |
| max_pooling2d_8 (MaxPooling2 | (None, 2, 9, 68) | 0 |
| dropout_11 (Dropout) | (None, 2, 9, 68) | 0 |
| flatten_5 (Flatten) | (None, 1224) | 0 |
| dense_9 (Dense) | (None, 256) | 313600 |
| dense_10 (Dense) | (None, 1) | 257 |

Figure 5: CNN4 Architecture for Car data set

| Architecture | Accuracy |
|---|---|
| Fully-connected NN | 0.895 |
| CNN 0 | 0.976 |
| CNN 1 | 0.981 |
| CNN 2 | 0.971 |
| CNN 3 | 0.971 |
| CNN 4 | 0.967 |
| CNN 5 | 0.948 |
| CNN 0 with augmentation | 0.938 |
| CNN 1 with augmentation | 0.981 |
| CNN 2 with augmentation | 0.957 |
| CNN 3 with augmentation | 0.957 |
| CNN 4 with augmentation | 0.971 |
| CNN 5 with augmentation | 0.914 |

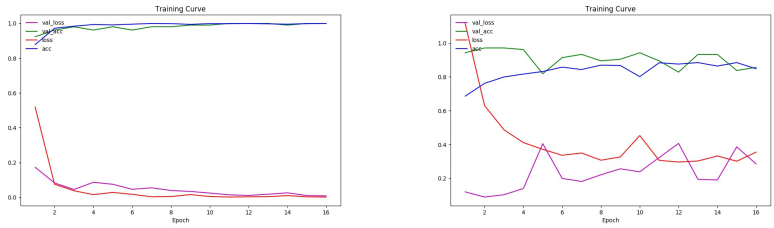
Table 4: Accuracy for different architectures for the Car data set tested on the validation set(split from training set)

(figure 7, 8) we see that with augmented data it becomes more difficult to overfit the training data, what decreases the accuracy of training and validation set for the both CNN. So when we tested the CNN architectures on the original test data, data augmentation improved the results very much.



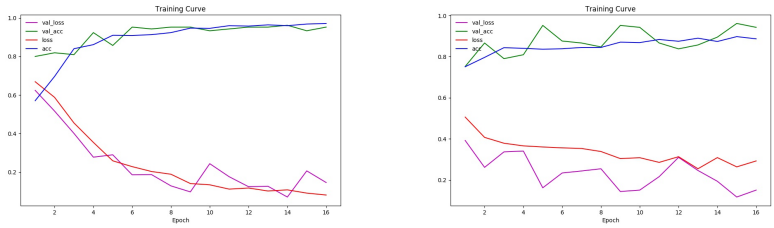

(a) Without data augmentation (b) With data augmentation

Figure 6: Confusion Matrices for CNN classification for the car data set with and without data augmentation.




(a) CNN2 (b) CNN2 With data augmentation

Figure 7: History curve for CNN2 with and without data augmentation.




(a) CNN4 (b) CNN4 With data augmentation

Figure 8: History curve for CNN4 with and without data augmentation.

| Architecture | Accuracy |
|---|---|
| Fully-connected NN | 0.159 |
| CNN 0 | 0.006 |
| CNN 1 | 0.029 |
| CNN 2 | 0.006 |
| CNN 3 | 0.029 |
| CNN 4 | 0.053 |
| CNN 5 | 0.053 |
| CNN 0 with augmentation | 0.671 |
| CNN 1 with augmentation | 0.412 |
| CNN 2 with augmentation | 0.676 |
| CNN 3 with augmentation | 0.641 |
| CNN 4 with augmentation | 0.429 |
| CNN 5 with augmentation | 0.382 |

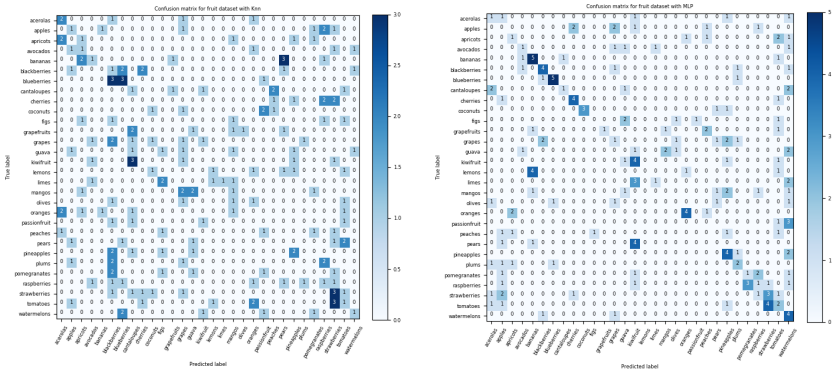
Table 5: Accuracy for different architectures for the Car data set tested on the test set

# 2 Fruit Dataset

## 2.1 *Traditional* classifiers and feature selection

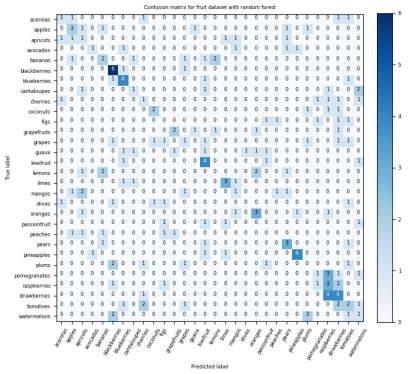
**Protocol** For the FIDS30 data set we used the same procedure for feature selection, hyperparameter optimization, training and validation as for the car data set.
The only problem was, that OpenVC failed to read the input images apricots/26.jpg, apples/49.jpg, apples/41.jpg, /cherries/10.jpg, /bananas/11.jpg, pineapples/11.jpg, raspberries/11.jpg and guava/32.jpg. Thus, we excluded them from the data set.

**Results** For k-NN, k=7 yielded the best results in testing different parameters with a mean accuracy of 0.107 and a standard deviation of 0.016. For the weighting methods, distance weighting was better than uniform weighting with a mean accuracy of 0.112 and a standard deviation of 0.013 versus 0.098 and a standard deviation of 0.008. For validation we got for k-NN with k=7 and distance weighting an accuracy of 0.114.
As to be expected, the highest tested number of trees for the random forest yielded again the best results. We got here a mean accuracy of 0.298 with a standard deviation of 0.017 for the cross-validation of the training set and an accuracy of 0.275 on the validation set.
For the MLP we got the best mean accuracy for a size of 5000 nodes with a mean accuracy of 0.260 and a standard deviation of 0.014. The identity activation function yielded the best mean accuracy over the different activation functions with a mean accuracy of 0.204 and a standard deviation of 0.012 and Adam for




(a) k-NN

(b) MLP




(c) Random Forest

Figure 9: Confusion Matrices for *traditional* classifiers for the fruits data set

the solvers with a mean accuracy of 0.171 and a standard deviation of 0.022 (it is noticable, that the mean accuracy for the other two solvers was in both cases below 0.05). On the validation set the MLP with optimized hyperparameters yielded an accuracy of 0.275.

The confusion matrices for validation for the three classifiers can be seen in figure 9. It is difficult to see, speecific trens or errors, but interestingly, all three classifiers seem to predict tomatoes quite often falsely as strawberries.

In table 7 are the testing, training and validation run times for all three classifiers listed, in table 3 also the feature extraction times. Overall, run times for the fruits data set were longer and again, computing the Bag of Visual Words took the longest.

| | k-NN | MLP | Random Forest |
|---|---|---|---|
| Accuracy | 0.114 | 0.275 | 0.275 |

Table 6: Accuracy on the validation set for the Fruits data set for the *traditional* classifiers

| | k-NN | MLP | Random Forest |
|---|---|---|---|
| Training (mean) | 0.004 | 3.964 | 0.148 |
| Testing (mean) | 0.039 | 0.010 | 0.004 |
| Validation | 0.025 | 18.652 | 0.368 |

Table 7: Training, testing and validation run times (in second) for the *traditional* classifiers for the fruits data set

## 2.2 Deep Learning

**Protocol** The set was split onto training and test sets with 80% and 20% correspondingly. Then the normalization with standard deviation was made for both training and test set.

At first there were made experiments with a simple fully connected network with architecture described in figure 10. The activation function for output was "softmax", as the task is multiclass classification, this was also the reason for peaking "sparse categorical crossentropy" as a loss function. The "batch size" was of 32 and optimal "epochs" appear to be - 40.

Then there were made experiments using convolutional neural network of three different architecture. On the figure 11 one can see architecture for CNN 1. CNN 2 has different number of channels in the second layer - 32 and in the first the same - 16. CNN 3 has additional layer and number of channels per lawyer 16-32-32 correspondingly. For each CNN the "batch size" was of 32, "epochs" = 20. The next step was to make an augmentation of the training data and check how it influence the result.

**Results** The results on time and accuracy can be found in the table 8.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               14400128
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_3 (Dense)              (None, 30)                7710
=================================================================
Total params: 14,440,862
Trainable params: 14,440,862
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: Architecture of the simple Fully-connected network for Fruits data set

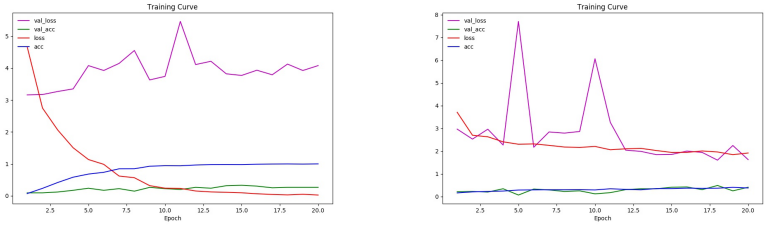
| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_37 (Conv2D) | (None, 148, 248, 16) | 448 |
| batch_normalization_18 (Batc | (None, 148, 248, 16) | 64 |
| activation_54 (Activation) | (None, 148, 248, 16) | 0 |
| max_pooling2d_18 (MaxPooling | (None, 74, 124, 16) | 0 |
| dropout_54 (Dropout) | (None, 74, 124, 16) | 0 |
| conv2d_38 (Conv2D) | (None, 72, 122, 16) | 2320 |
| activation_55 (Activation) | (None, 72, 122, 16) | 0 |
| dropout_55 (Dropout) | (None, 72, 122, 16) | 0 |
| flatten_18 (Flatten) | (None, 140544) | 0 |
| dense_38 (Dense) | (None, 256) | 35979520 |
| activation_56 (Activation) | (None, 256) | 0 |
| dropout_56 (Dropout) | (None, 256) | 0 |
| dense_39 (Dense) | (None, 30) | 7710 |

Total params: 35,990,062
Trainable params: 35,990,030
Non-trainable params: 32

Figure 11: Architecture of CNN 1 for Fruits data set

For the Simple fully-connected network we achieved the accuracy for test set of 0.23 which is worse than result with traditional classifier and features selection.

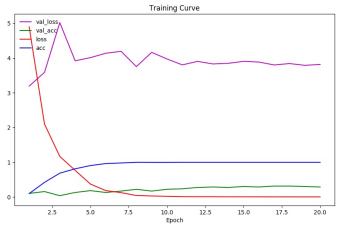
For the CNN 1 we have got the accuracy of 0.28, which is slightly better that the best result of simple classification with feature selection (random forest). With data augmentation the result was better - accuracy of 40%. From the figure 12 one can see that the accuracy for training set in the case without data augmentation is much higher but the loss function is higher as well. Accuracy for validation set doesn't look very different in both cases, however the loss function for validation set is lower in case of augmented data.



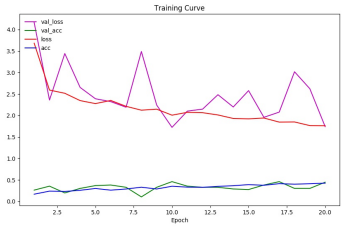

(a) CNN 1

(b) CNN 1 with augmented data

Figure 12: Results on accuracy and loss function for training and test set with CNN1 for Fruit data set

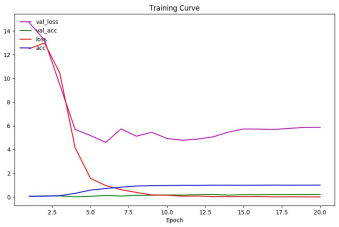



(a) CNN 2

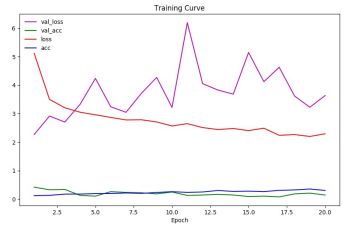



(b) CNN 2 with augmented data

Figure 13: Results on accuracy and loss function for training and test set with CNN2 for Fruit data set




(a) CNN 3




(b) CNN 3 with augmented data

Figure 14: Results on accuracy and loss function for training and test set with CNN 3 for Fruit data set

For the CNN 2 the accuracy for test set was 0.32, for the model trained with augmented data it has increased to 0.46. On the figure 11 is shown that loss function with augmented data for validation set is again lower that for the case without augmentation and the accuracy for validation set is higher with augmented data.

On the figure 14 you can see the similar results for CNN 3 with 3 layers for loss function as in previous cases. However the additional layer doesn't bring better results on accuracy, in this case we have only 19 % for the test set. With augmented data it is even a bit worse - 0.18.

The results on training and testing time were the biggest for CNN with augmented data. On simple data it takes 600 seconds in average to train the model, with augmented around 850 seconds.

# 3 Analysis

For the car data set the results showed that for the simple case of testing a on a split of the training data, Deep Learning architectures performed really good, in all cases better than the traditional classifiers. Because the test data was

| Architecture | Epochs | Accuracy | Training time | Testing time |
|---|---|---|---|---|
| Fully-connected NN | 40 | 0.23 | 91.229 | 0.287 |
| CNN 1 | 20 | 0.28 | 495.686 | 2.166 |
| CNN 2 | 20 | 0.32 | 717.999 | 2.405 |
| CNN 3 | 20 | 0.19 | 599.766 | 2.617 |
| CNN 1 + augmentation | 20 | 0.40 | 745.451 | 4.058 |
| CNN 2 + augmentation | 20 | 0.46 | 919.009 | 4.511 |
| CNN 3 + augmentation | 20 | 0.18 | 968.099 | 7.143 |

Table 8: Results of DL for Fruits data set

more simple, data augmentation was not necessary and did not improve results further. However, for the more difficult test data set, accuracy was without data augmentation very bad but could be improved.
For the fruits data set the simple fully-connected NN doesn't give better results than the traditional classifiers. However, with some Convolutional network the result is getting better than traditional classifiers with feature selection. Data augmentation increases the results even more. It is interesting to notice, that in the case of Convolutional NN with 3 layers we have received worse results than with two layers, while the time performance was worse.
Comparing the run times for the car data set, mean training time was much worse than the time the altogether time taken by feature extraction and training of traditional classifier. For the Fruits data set on the other hand, computing the Bag of visual Words took an especially long time, longer than the training time of every CNN.

# 4 Conclusion

We made different experiments both with Deep Learning architectures and *traditional* classifiers with feature extraction methods.
For both data sets we could show that CNN architectures yielded mostly better results than the traditional classifiers. Further, data augmentation helped to improve the classification performance for difficult tasks. For the run time, however, there is no clear trend to which approach is more efficient. For data sets with many channels, feature extraction in the *traditional* way can be quite time demanding.

# References

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.