Exercise 3.2 Deep learning

e12045110 Maria de Ronde
e12040873 Quentin Andre
e11921655 Fabian Holzberger

July 25, 2021

Datasets

For exercise 3.2 Deep Learning we decided to apply deep learning on image classification. The data sets that we will use are CIFAR-10 [1] and Tiny-ImagenNet [2]. With these two datasets we have variation in the number of classes represented in the data. This enables us to explore the difference in performance when the number of classes increase. In sections and both datasets are described in more detail. The pictures also differ in input size.
We used Python as our programming language and implemented the deep learning using Tensor Flow.

CIFAR-10

CIFAR-10 is a dataset which consists of 60.000 images, of which 50.000 training images and 10.000 test images. Each image has 32×32 colored pixels. There are 10 different classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck) each class has exactly 5.000 images in the training data and 1.000 images in the test data. Each image only belongs to one class. There are no multi-label images.

Tiny ImageNet

Tiny ImageNet is a dataset containing of 100.000 training images, divided in 200 different classes. There are 500 images per class in the training data. Next for the training data there are 10.000 testing and 10.000 validation images as well. Each picture has 64×64 pixels. The images in the test-set are not labelled and therefore we will not make use of them. To load the data some code from Github [3] was used.

Traditional classifiers

In order to have a baseline for our deep classifier some traditional classifiers have been executed. The following traditional classifiers have been trained:

1. **Multinomial Naive Bayes**: `alpha = 1.0`, `fit_prior= True`, `class_prior= None`
2. **Random forest**: `n_estimators = 100`, `criterion='gini'`, `max_depth=None`, `min_samples_split=2`, `min_samples_leaf=1`, `min_weight_fraction_leaf=0.0`, `max_features='auto'`, `max_leaf_nodes = None`, `min_impurity_decrease=0.0`, `min_impurity_split=None`, `bootstrap=True`, `oob_score=False`,

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] http://cs231n.stanford.edu/tiny-imagenet-200.zip
[3] https://github.com/rmccorm4/Tiny-Imagenet-200

`n_jobs=None`, `random_state=None`, `verbose=0`, `warm_start=False`, `class_weight=None`, `cc_alpha=0.0`, `max_samples=None`

3. **Single layer perceptron**: `penalty=None`, `alpha=0.0001`, `l1_ratio=0.15`, `fit_intercept=True`, `max_iter=1000`, `tol=0.001`, `shuffle=True`, `verbose=0`, `eta0=1.0`, `n_jobs=None`, `random_state=0`, `early_stopping=False`, `validation_fraction=0.1`, `n_iter_no_change=5`, `class_weight=None`, `warm_start=False`

4. **Multi layer perceptron**: 2 Relu activation layers 256, 1 softmax activation 10 `epochs=15`, `batch_size=32`, `verbose=0`

Before we could train the traditional classifiers, we extracted features from our images. We performed two type of feature extraction.

1. Color histogram
2. SIFT

Color histograms

Color histograms is one of the simplest feature extraction method for images. It counts the frequency of pixels with a certain color. The bins are based on the RGB coding, each pixel has a value for red, green and blue between 0 and 255. Color histograms do not take spatial information into account. All spatial information gets lost during this feature extraction. In Figure1 an example picture with the corresponding color histogram for both datasets is given.

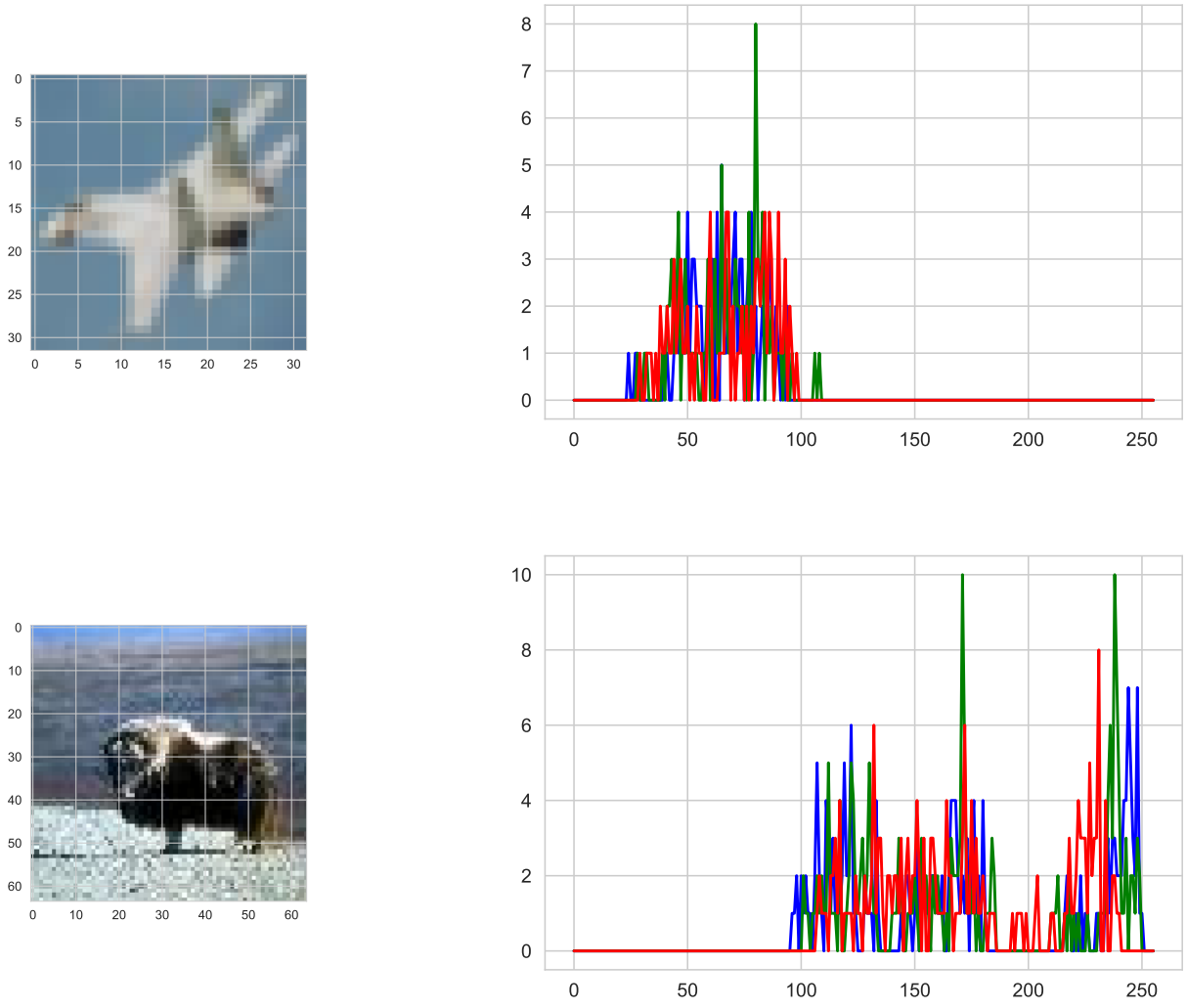


Figure 1: Top: Example picture and color histigram of Cifar dataset. Down: Example picture and color histigram of Tiny Imagenet dataset

We created 4 different datasets using color hisograms, two based on one dimensional histograms (one with 256 bins per channel and one with 64 bins per channel), one on two dimensional histograms (16 bins per channel) and one of 3 dimensional histograms (8 bins per channel). This is based on the example shown in simple-image-feature-extraction [4]. In order to create the color histograms we used the package OpenCV.

Sift back of visual words

First the images are converted into grey scale images. With use of SIFT the key-points and descriptors are detected. K-means-clustering is applied to find 20 clusters in the detected descriptors.
Finally vectors with the visual words are created and vectorized in a histogram (frequency of visual words). The histogram is scaled to a standard normal distribution before running the classifiers.

[4] https://tuwel.tuwien.ac.at/course/view.php?id=35929

Results

In Figure 2 and **??** the accuracy for the different base classifiers can be found for the Cifar-10 and Tiny image dataset respectively. It can be seen that the random forest out performs all classifiers with the color histogram features, the best performance was obtained using the 2-d color histograms.
Accuracy of BOW The confusion matrices shown in Figure 3 are based on the 2-d feature color histograms as this gave the best results. The confusion matrix of the single-layer-perceptron shows pretty much a random classifier. Furthermore it can be seen that trucks are often classified as cars and birds are often classified as ship
. The confusion matrix of the Naive Bayes, Random Forest and multi-layer-perceptron show similar results. They show that airplanes, ships and trucks are predicted best even though airplanes and ships are also often confused. It can also be seen that the different animals get confused.

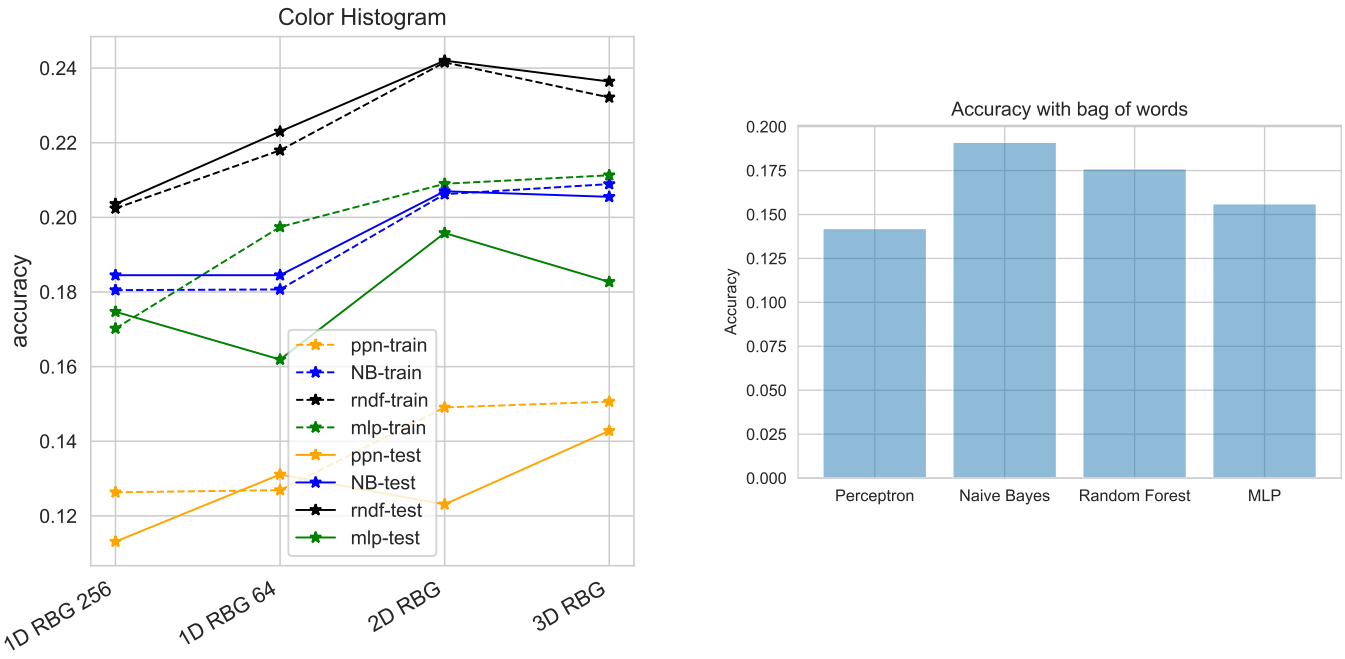


Figure 2: Accuracy Cifar-10

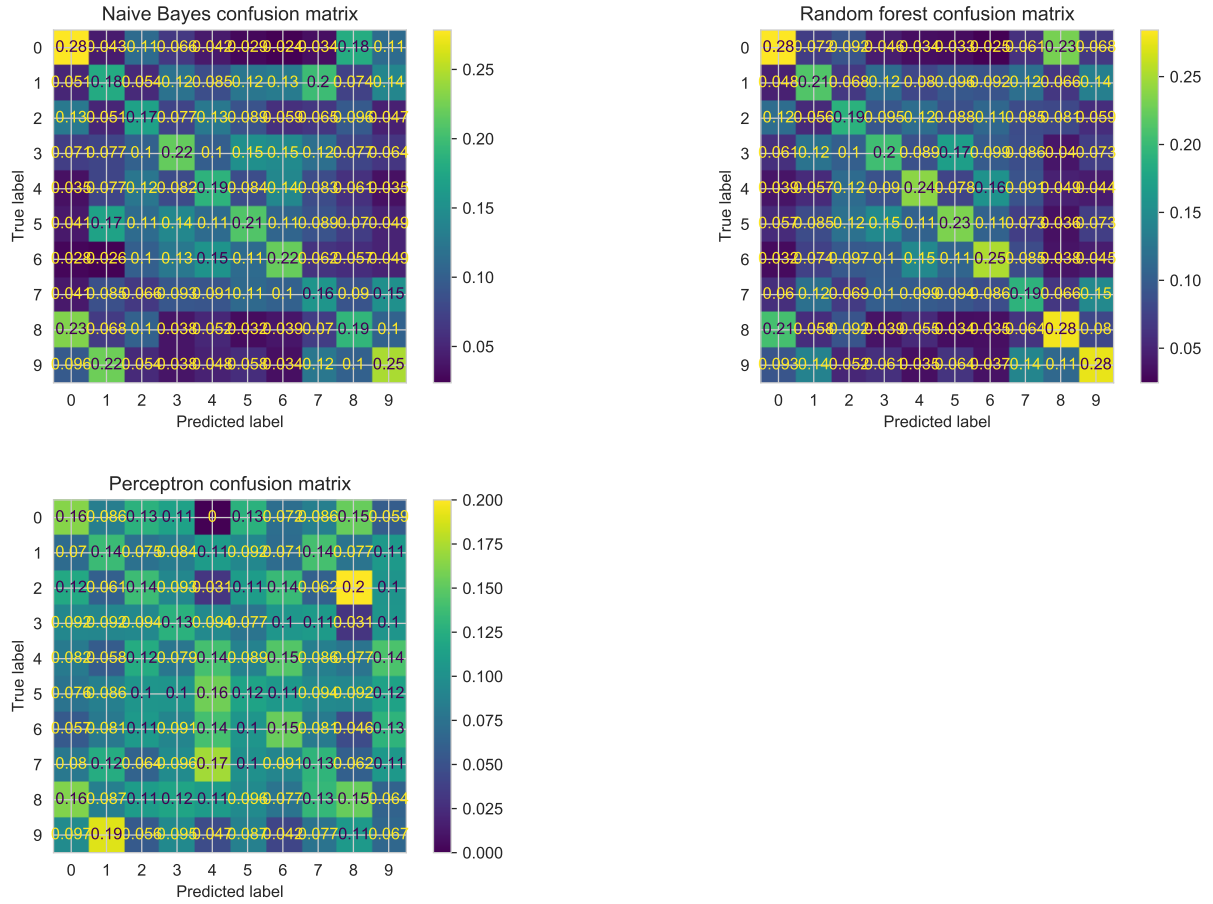


Figure 3: Confusion matrices color histogram Cifar-10: 1. Naive Bayes 2. Random forest 3. Single-layer-perceptron 4. Multi-layer-perceptron

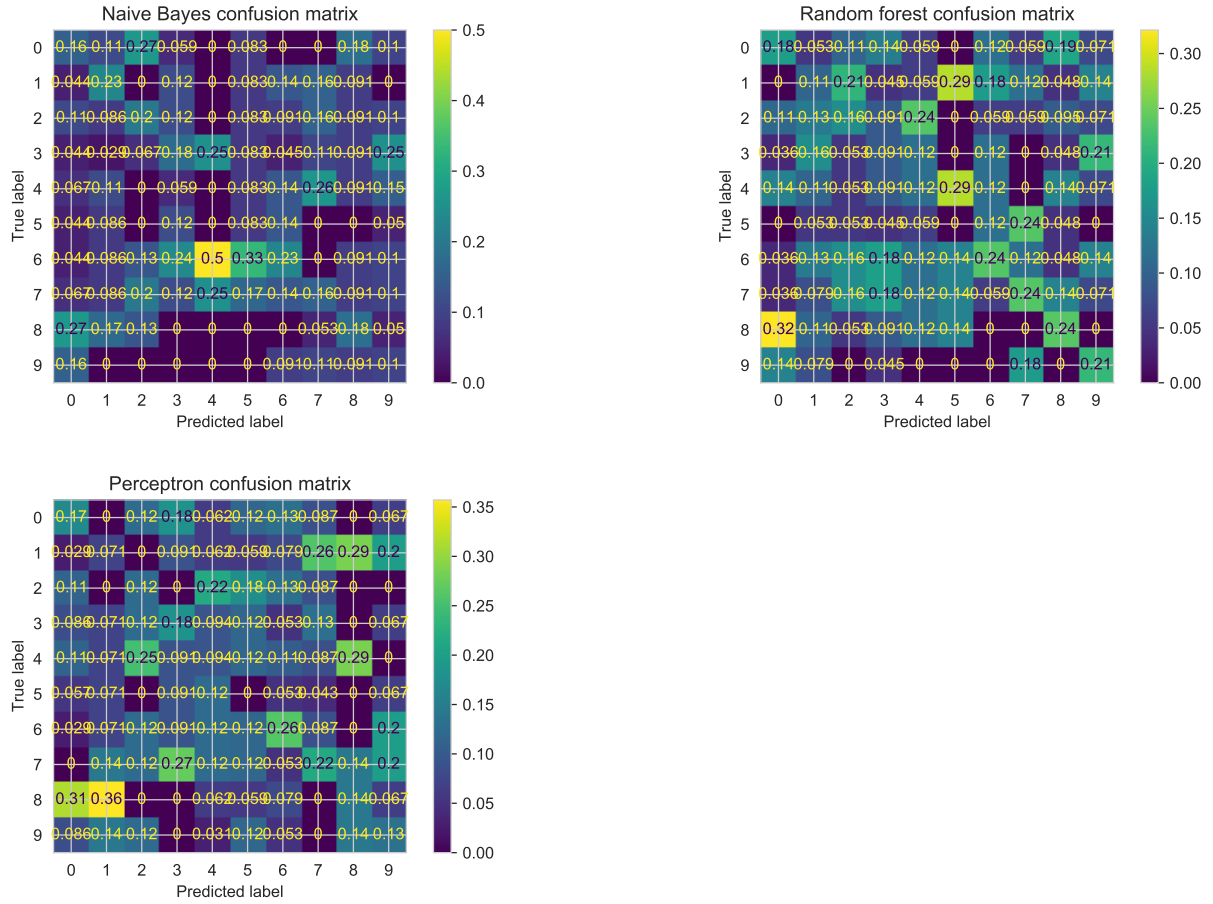


Figure 4: Confusion matrices bag of words Cifar-10: 1. Naive Bayes 2. Random forest 3. Single-layer-perceptron 4. Multi-layer-perceptron

Figure 5: Comparison of time required for different feature extractions and training of models

Convolutional Neural Networks (CNN's)

In this section we introduce the CNN architectures used for this work.

SqueezeNet

For the first architecture we decided on unsing SqueezeNet [2]. SqueezeNet was developed with the goal to make it as compact as possible, but still achiving state of art results. In the original paper it is shown that SqueezeNet can achive the same perfomance as an AlexNet for the ImageNet competition but with $50\times$ less parameters. This makes SqueezeNet particularly well suited for our small project where we need to rely on limited resources.

Wrapped Res50Net for Transfer learning

For the second architecture we have choosen intentionaly a large one with many paramters, that gives the opertunity for transfer learning. Here we aim to use Res50Net [1], which has over 23 Million trainable parameters. We wrap this architecture by a custom network. For the training we freeze all layers till layer 168 and train the remaining layers of Res50Net and the additional custom layers. More details are given in the Results section.

Training Details for CNN's

For the training we are perfomring image augmentation and compare the obtained results to the case when no augmentation is applied. We apply for all CNN's the same random augmentations:

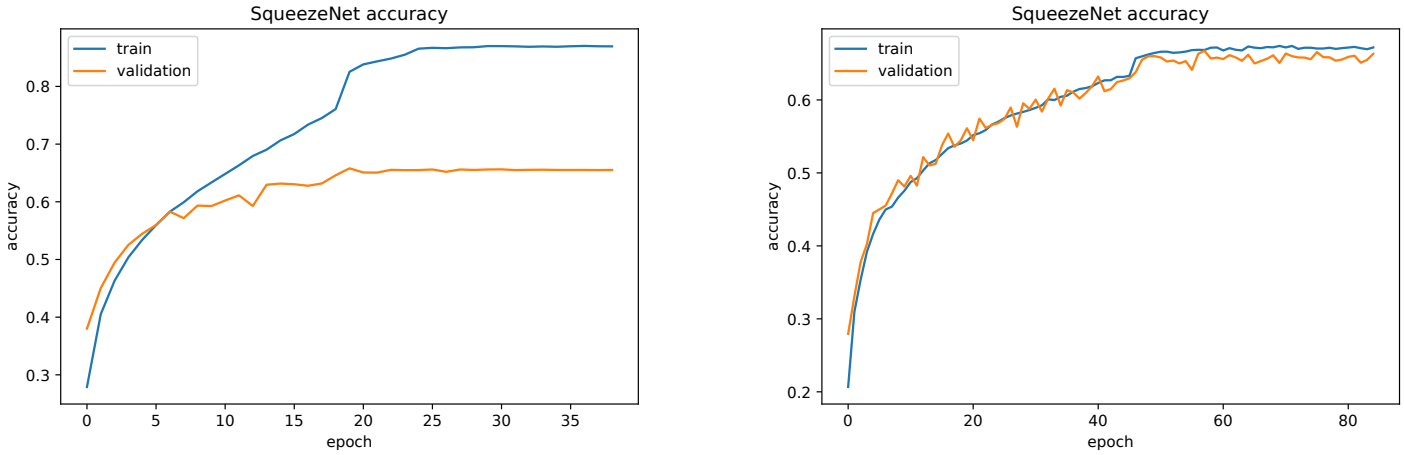
- width/height shift up to 10%
- shearing up to 10%
- zoom up to 10%
- rotation up to 30°
- horizontal flip

For the training it was observed that the ADAM optimizer with learning rate 0.0001 works well for all our cases. While performing the parameter search we decrease the learning rate by a factor 0.1 if the loss function did not decrease for the 5 previous iterations, where the loss function is the cathegorical-crossentropy. Additionally if the loss function does not improve for 15 iterations we stop the parameter search early. The parameter search is exellerated by a NVidia K80 GPU, but test-set evaluation times are measured when employing our CNN's on a Intel i5 CPU. Another important aspect is that we dont normalize the RGB scales of the pixels since this caused non-convergence for the most parameter searches. All CNN'related analysis is perfomed as holdout, meaning we create a test, train validation set split.

Results Cifar10 Dataset

SqueezeNet

For the Cifar10 Dataset the resulting SqueezeNet architecuture has 740.554 trainable parameters. A holdout analysis is performed where we take 10% of the train-data as validation data to evaluate the performance of the CNN at training time.

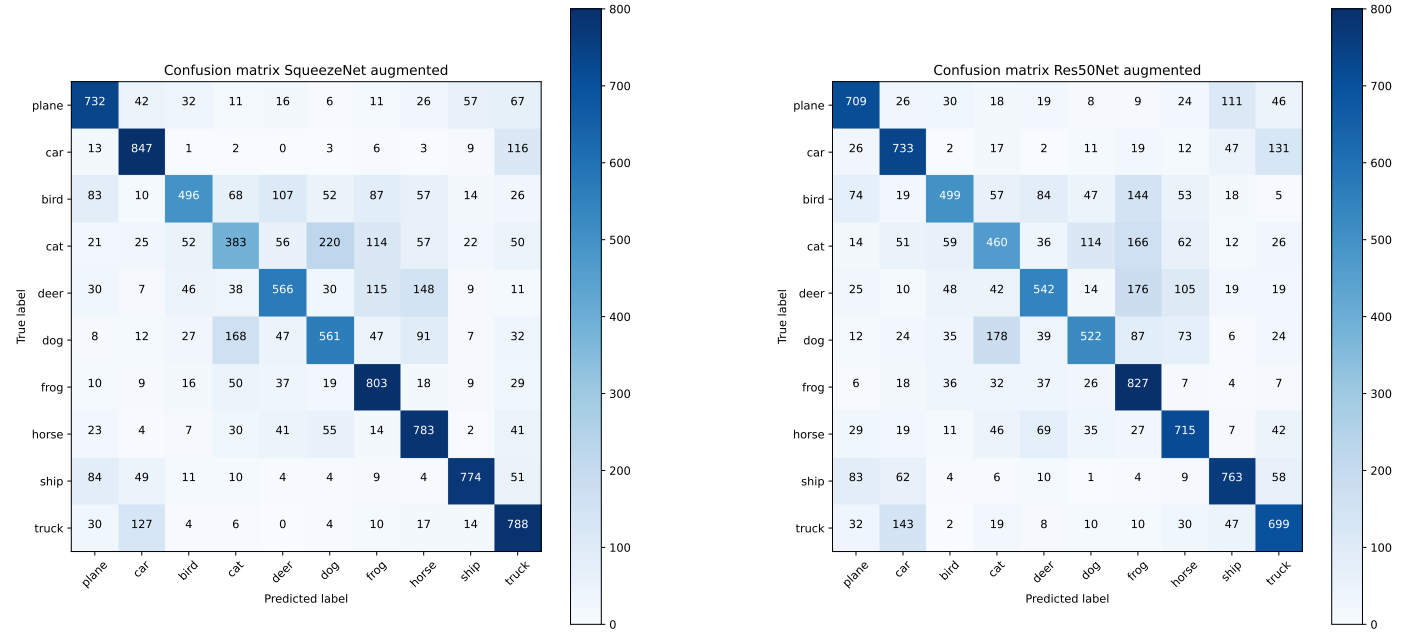


(a) accuracy for SqueezeNet without augmentation

(b) accuracy for SqueezeNet with augmentation

Figure 6: Training-accuracy for SqueezeNet on the Cifar10 Dataset

In figure 6 we can see the effect of the augmentation on the accuracy of the predictions while training. We have a clear overfitting of the CNN when not applying augmentation. This overfitting is largly decreased by applying the image-augmentation as can be seen in the figure. When applying the augmentation the accuracy of SqueezeNet on the Test-set increased from 65.21% to 67.33%. Augmentation of the images has an influence on the efficiency of the parameter search for the Network. Without augmentation the optimal paramters are found after 6 min and less than 40 epochs, whereas augmentation causes the time for parameter search to increase to 46 min and 84 epochs. For the predition of the 313 samples in the test-set SqueezeNet needs about 9s



(a) Confusion matrix for SqueezeNet trained on augmented Datase

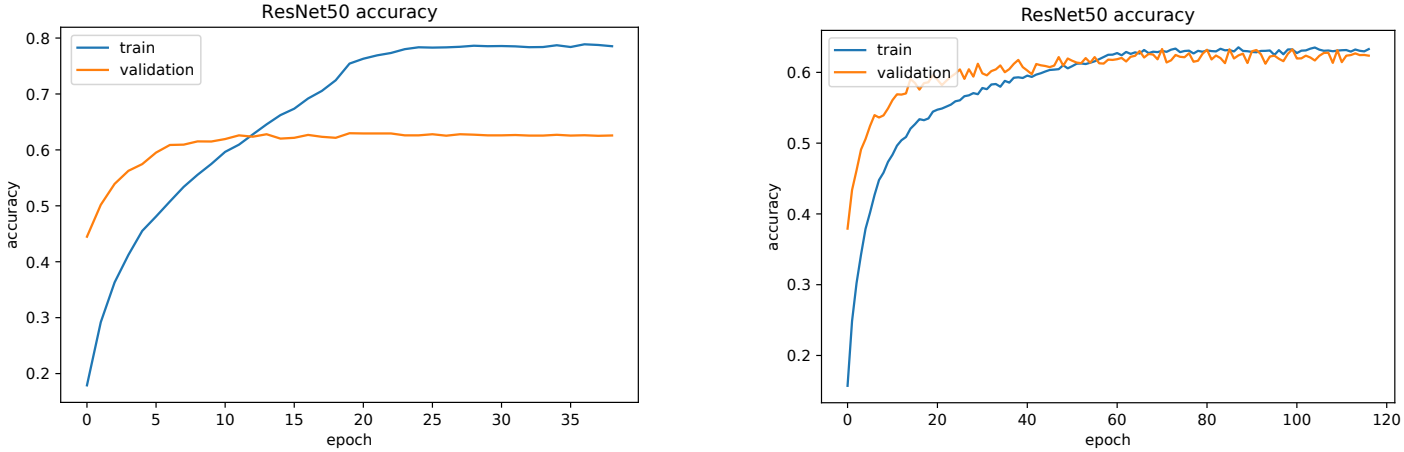
(b) Confusion matrix for wrapped Res50Net trained on augmented Datase

Figure 7: Confusion Matrices of CNN architectures on the Cifar10 Dataset

In figure 7 (a) we show the corresponding confusion matrix for the SqueezeNet trained on the augmented Cifar10 images. The most true positive labels are predicted for the car class with 847 where we note that 127 times a truch sample gets confused to be a car. The least true positive labels are predicted for cats with 383 where this class often gets confused with the dog class. The most false negative preditions are made for the truck class with 423 samples closely followed by the horse class with 421 flsely predicted samples. This is due to the fact that Trucks are often confused with cars and horses are often confused with deers or dogs.

Res50Net

To create our wrapped Res50Net we stip off the top layers and replace that layer by a custom layer of $32 \times 32 \times 3$ for reading our images. The output of the Res50Net gets fed into 3 densle layers with $256, 128, 64$ neurons till it reaches the softmax layers for prediction. Note that our dense layers consist each at the top of a batch normalisation layer and the dense layers with a dropout of 50% for regularisation purposes. For the Res50Net layers in our CNN we enable the training of the 168 bottom layers and freeze the rest of them. By that we end up with a CNN that has in total 24.163.786 parameters from which are 3.986.762 trainable. For the transfer learning we apply the same image augmentation as for SqueezeNet.



(a) accuracy for Res50Net without augmentation

(b) accuracy for Res50Net with augmentation

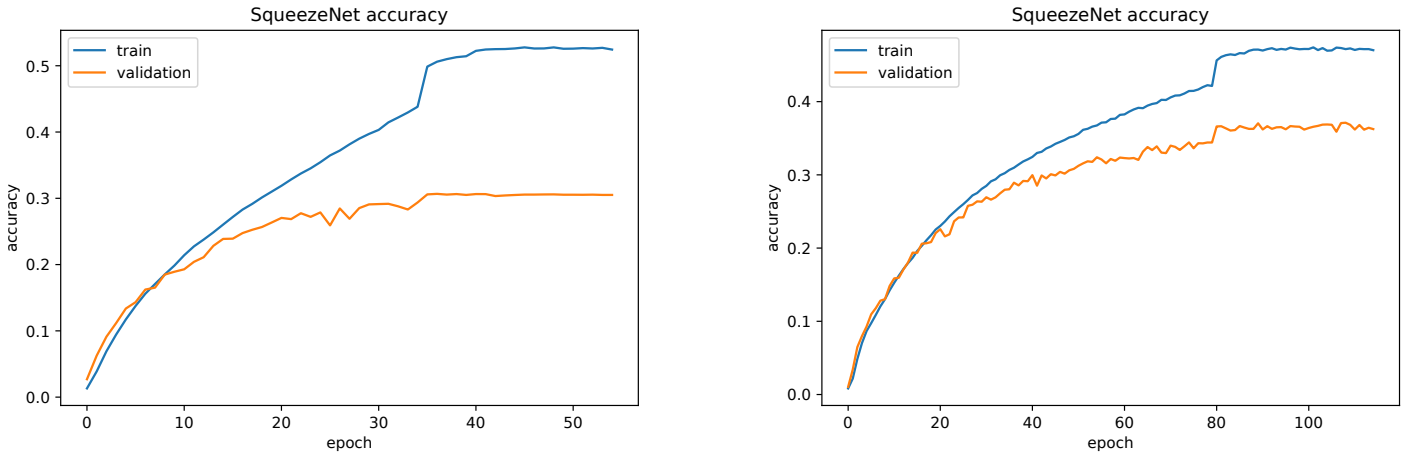
Figure 8: Training-accuracy for Res50Net on the Cifar10 Dataset

In figure 8 we show the accuracy while executing the transfer learning of Res50Net. Again without augmentation the CNN overfits till about 80% accuracy on the trainset while the accuracy on the validation-set is 20% lower. With augmentation we have diminished this effect and additionally improve the accuracy on the test-set from 62.16% for no augmentation to 64.69% with augmentation. The parameter-fitting for the augmented process took 117 epochs with a total time of 64 min compared to that the parameter-fitting with no augmentation took 39 epochs and about 10 min in total. The evaluation on the testset took 38s which is more 4 than longer than for the SqueezeNet.
In figure 7 (b) the corresponding confusion matrix for the Res50Net trained on the augmented Cifar10 images is plotted. The most true positive predictions are made for the frog class with 827. We also see that the frog class has aswell the most true negative predictions where it seems particularly hard for the CNN to differentiate it from other animals. The cat class with the one with lowest true positives at 460, where we note that 148 dogs get wrongly classified as cats. Also notable is that cars get often cofused as trucks and ships are often confused as planes.

Results Tiny ImageNet Dataset

SqueezeNet

On the Tiny ImageNet SqueezeNet has 838.024 trainable parameters. First we plot in figure 9 the auccuracy over the number of training epochs.

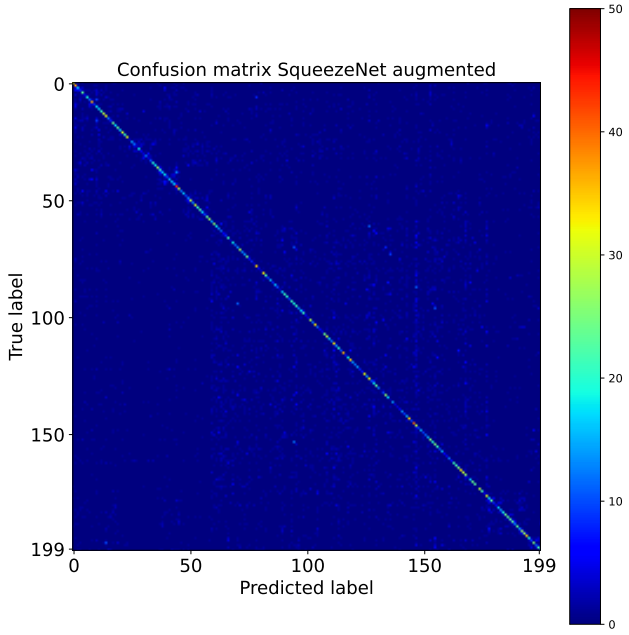


(a) accuracy for SqueezeNet without augmentation

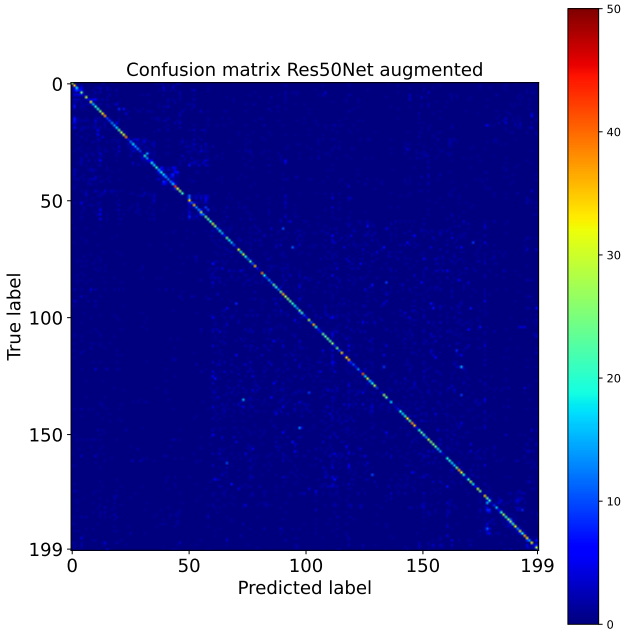
(b) accuracy for SqueezeNet with augmentation

Figure 9: Training-accuracy for SqueezeNet on the Tiny ImageNet Dataset

As before we can examine less overfitting for the augmented case, but this time even with augmentation the SqueezeNet is overfitting as can be seen in the much higher acuracy of on the train-set compared to the accuracy on the validation-set. In both graphs we can identify the first decrease in learning rate by the jump in the training accuracy. Note that this jump occurs at a much later epoch for the augmented case showing more potential for improvement of the accuracy in the augmented case before reaching a local optima. For the evaluation on the test-set the SqueezeNet without augmented training-data has an accuracy of 31.26% and augmentation caused the accuracy on the test-set to increase up to 34%. The traidoff we pay is that for the training with no augmentation the parameter-search takes 55 epochs and 41 min, but with augmentation 115 epochs and 297 min. Since the Tiny ImageNet is larger that Cifar10 our evqaluation time for SqueezeNet on the 10.000 samples test-set is here about 60s.



(a) confusion matrix of SqueezeNet with augmentation



(b) confusion matrix of wrapped Res50Net with augmentation

Figure 10: Confusion matrices for CNN archtectures on the augmented Tiny ImageNet Dataset.

If we analyze the confusion matrix in figure 10 (a) we can measure the quality of our classifier by the size of the diagonal entries and the small offdiagonal entries.
Many classes have a large number of true prositiv predicted labels but to understand the classifier better analyze some special labels more closely. In table 1 is a summary of some important properties of the confusion matrix given.

label	corr. amount	label	corr. amount	act. label	pred. label	times confused
water jug	0	monarch	43	bee	monarch	13
plunger	1	school bus	40	sports car	convertible	12
pop bottle	2	fire salamander	37	CD player	scoreboard	11
barrel	3	black widow	36	altar	organ	11
umbrella	3	lifeboat	36	convertible	beach wagon	11
Labrador	3	brass	34	coral reef	brain coral	10
Chihuahua	3	espresso	34	dam	steel arch bridge	10
bucket	3	rugby ball	34	beach wagon	convertible	10
chain	3	maypole	34	scorpion	centipede	9
dumbbell	4	triumphal arch	33	slug	centipede	9

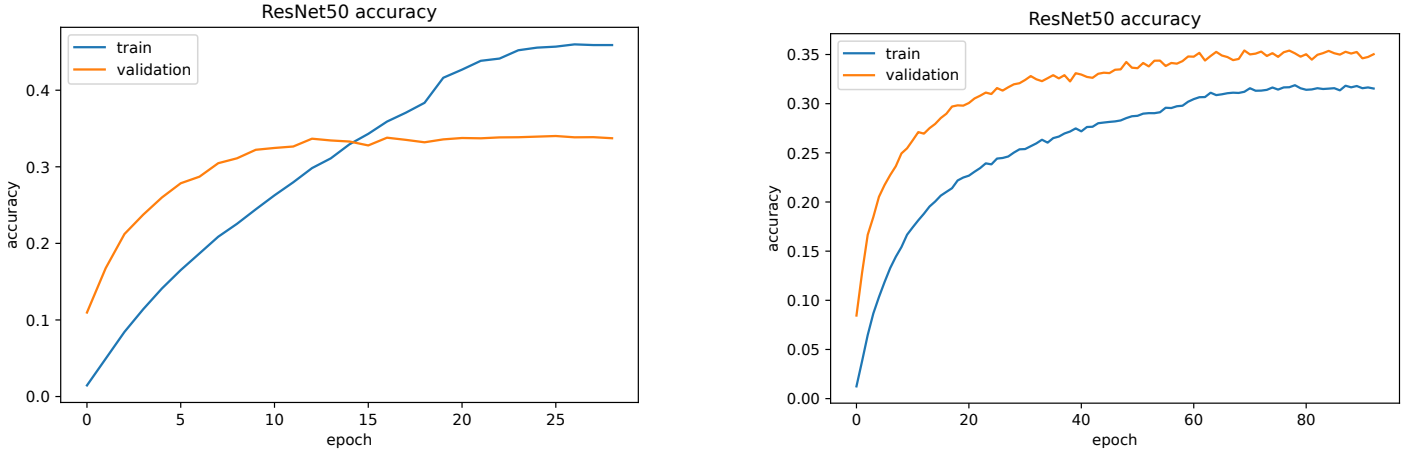
Table 1: Some detailed information about the Confusion matrix of the SqueezeNet with augmented Tiny ImageNet

The first two columns state the classes that have the least ammount of true positive predicted labels. Here we see that for the water jug we have no correct prediction whereas all other classes have at least one true positive predicted sample. For the classes with the highest amount of correct predetions we see in the next two columns

that the monach butterfly and secondly school busses were mostly correct identified. Further we can state that often strong colored subjects are easily identified like school busses ir life boats. In the last three columns the most confused labels are stated. Here we can extract that our SqueezeNet often confuses bees for monachs or secondly sports cars as convertables. This behaviour is as expected since auch subjects are very closely related in shape and color.

Res50Net

For the Res50Net we again create a custom wrapper CNN where the input layer is defined for $64 \times 64 \times 3$ images as reqired for Tiny ImageNet. The Res50Net's first 168 layers are not trained and we append additional three 256 dense layers at the end where each has a batch normalisation layer at top and a dropout of 50%. The total parameters of the wrapped Res50Net are 25.902.920 where 5.713.352 are trainable.



(a) accuracy for wrapped Res50Net without augmentation

(b) accuracy for wrapped Res50Net with augmentation

Figure 11: Training-accuracy for wrapped Res50Net on the Tiny ImageNet Dataset

The training accuracies for the wrapped Res50Net are given in the two plots in figure 11. For this case when using augmented images the accuracy on the validation-set is always higher than on the training-set in contrast to the studies before in this work. For the accuracy of the non augmented case on the test-set we report 34.57%, where the running time was 42 min with 39 epochs. For the augmented case we have a 37.7% accuracy on the test-set, with a running time of 282 min over 93 epochs. For this architecture the prediction time for the 10.000 testsamples is 103s.
The confusion matrix for the augmented case is shown above in figure 10 (b). In the confusion matrix we can see that the network works well, sice the off-diagonal entries are significantly lower than the diagonal ones for most of the columns. Again we cant do a in deoth analysis on such a large confusion matrix. Therefore we extract some of the most important patterns from the confusion matrix and show them in table 2.
When considereing the first two columns in the table we see that the classes plunger and hog never get predicted correctly. All other classes have at least one correct prediction. Further we see that without those 10 classes with the least amount of correct predictions all other classes have already at least 3 correct predictions. Next when analysing the columns 3 and 4 in the table we see the classes that have the most correct predicted labels. Simillar as in SqueezeNet the monarch, school bus and esspresso are apperaring here, which again suggests that large structures or objects with prominent colors can be recognized easier. Lastly when examining the 3 last columns on the right of the table we see the most confused classes. Mostly these confused classes describe objects that are very simillar in shape and color like the pop bottle and beer bottle. But we see also that

label	corr. amount	label	corr. amount	act. label	pred. label	times confused
plunger	0	monarch	42	pop bottle	beer bottle	15
hog	0	obelisk	40	ice cream	plate	15
syringe	1	bullet train	38	moving van	trolleybus	15
wooden spoon	1	dugong	38	tabby	Egypt. cat	14
ox	2	school bus	37	sew. machine	desk	13
bucket	2	brain coral	37	meat loaf	plate	11
projectile	2	espresso	36	beach wagon	convertible	11
ice cream	2	triumphal arch	36	hog	bison	11
tabby	3	maypole	36	orange	lemon	10
beach wagon	3	Christ. stocking	36	turnstile	pay-phone	10

Table 2: Some detailed information about the Confusion matrix of the wrapped Res50Net with augmented Tiny ImageNet

turnstiles are ofthen confused with pay-phones, besides they are not to closely related. By that we see that the wrapped Res50Net CNN has still potential for improvement and shortcomings like discussed could be improved by a more fine grained training.

Results

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.