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 [INCLUDE REFERENCE] and Tiny-ImagenNet[Include REFERENCE. With these two datasets we have variation in the classes represented in the data. This enables us to explore the difference in performance when the number of classes increase. In the following sections both datasets are described in more detail.

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.

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, 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 RBG coding. Spatial information get lost completely during this feature extraction. Below an example of one-d color histograms for picture 10 of the Cifar dataset is given. In FIGURE REF!!! a color histogram for both datasets is given.
We created 4 different datasets, two based on one dimensional histograms (256 bins per channel and 64 bins per channel), one on two dimensional histograms (16 bins per channel) and one of 3 dimensional histograms (8 bins per channel). This based on the example shown in simple-image-feature-extraction INCLUDE REFERENCE https://tuwel.tuwien.ac.at/course/view.php?id=35929 !!!. The color histograms have been created using OpenCV.

Sift back of visual words

First the images are converted into grey scale images. With use of SIFT the keypoints are detected. Afterwards 20 clusters are created with use of Kmeans.
Visual words are created and vectorized in a histogram (frequency of visual words).
Scale the histogram
Classify

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× 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.

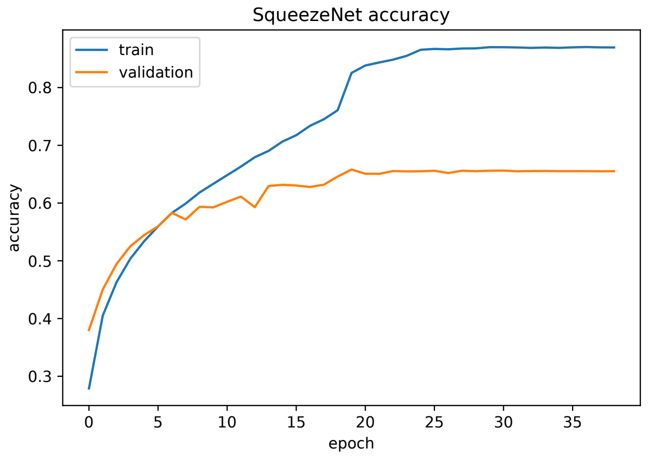
Results Cifar10 Dataset

SqueezeNet

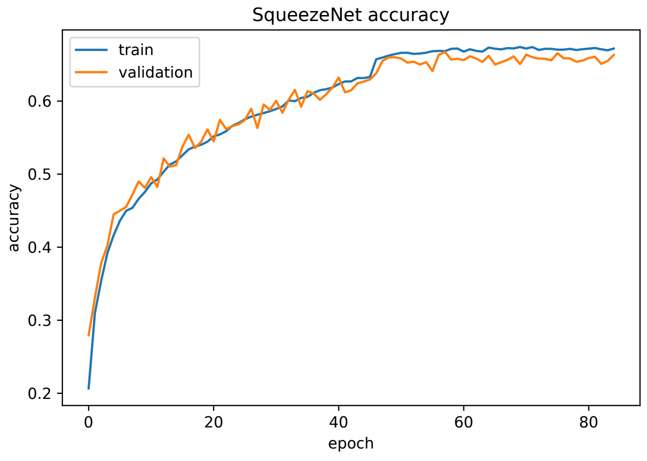
For the Cifar10 Dataset the resulting SqueezeNet architecuture has 740.554 trainable parameters. We augment the Cifar10 Dataset by the following random augmentations:

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

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.



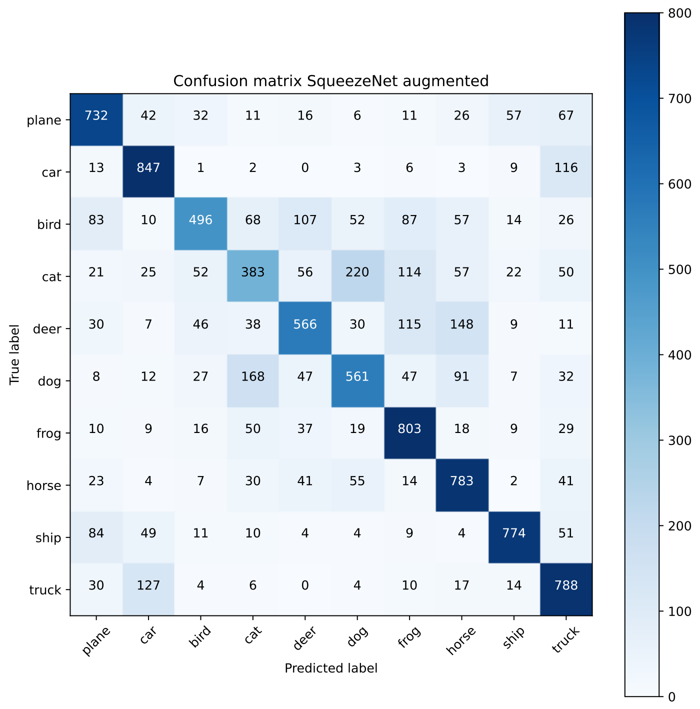
(a) accuracy for SqueezeNet without augmentation



(b) accuracy for SqueezeNet with augmentation

Figure 1: Training-accuracy for SqueezeNet on the Cifar10 Dataset

In figure 1 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



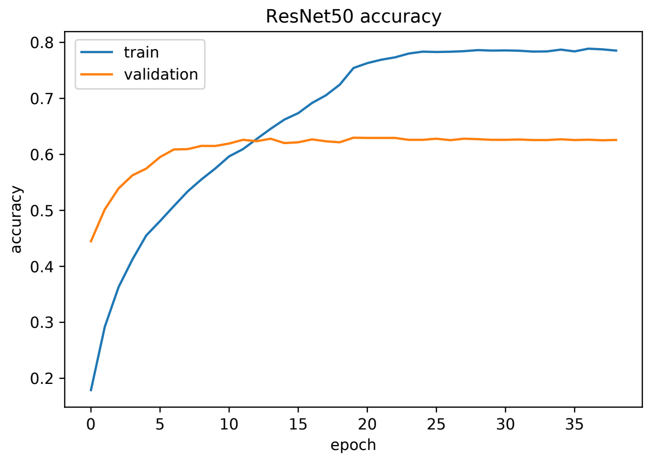
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	732	42	32	11	16	6	11	26	57	67
car	13	847	1	2	0	3	6	3	9	116
bird	83	10	496	68	107	52	87	57	14	26
cat	21	25	52	383	56	220	114	57	22	50
deer	30	7	46	38	566	30	115	148	9	11
dog	8	12	27	168	47	561	47	91	7	32
frog	10	9	16	50	37	19	803	18	9	29
horse	23	4	7	30	41	55	14	783	2	41
ship	84	49	11	10	4	4	9	4	774	51
truck	30	127	4	6	0	4	10	17	14	788

Figure 2: Confusion matrix for the SqueezeNet trained on the augmented Cifar10 Datase.

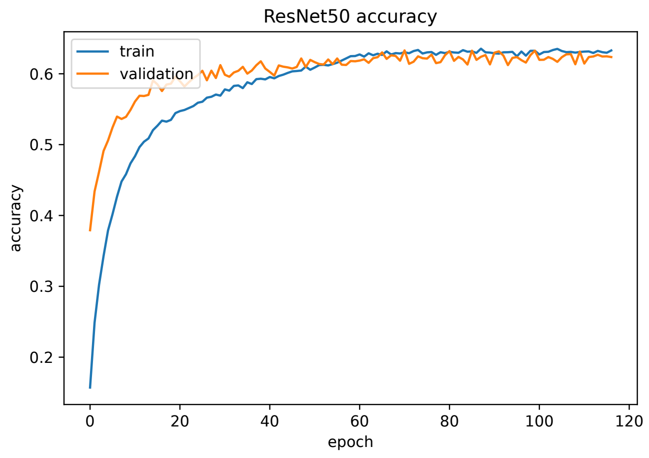
In figure 2 we show the corresponding confusion matrix for the SqueezeNet trained on the augmented Cifar10 images. The most true positive labels are predicted for the car class with 847 where we note that 127 times a truch sample gets confused to be a car. The least true positive labels are predicted 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 3: Training-accuracy for Res50Net on the Cifar10 Dataset

In figure 3 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.

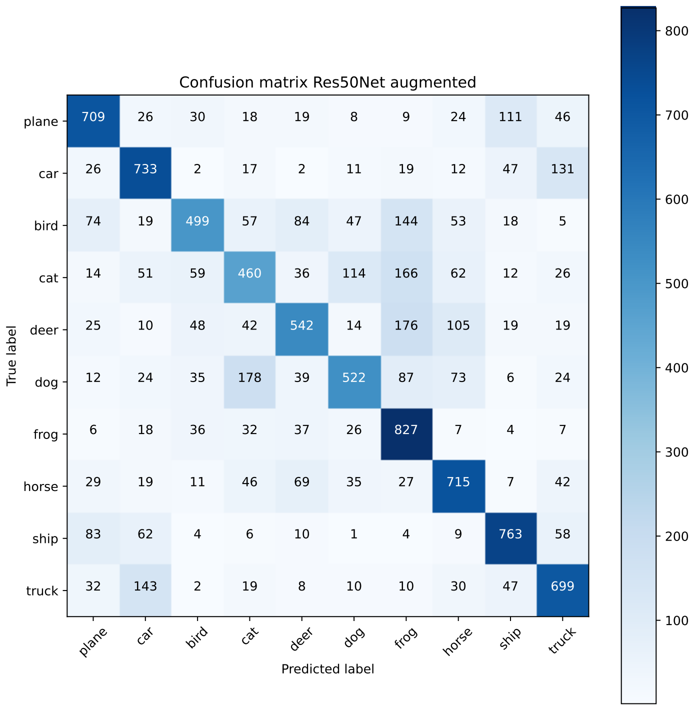


Figure 4: Confusion matrix for the SqueezeNet trained on the augmented Cifar10 Datase.

In figure 4 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

squeezNet

For the second architecture we decided on unsing SqueezeNet [2]. Squeezenet claims to have an accuracy equal to AlexNet but has a lot less parameters to learn. For SqeeuzeNet no weights are available, therefore we recreated the model and re-trained it.

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.