UNIVERSITY OF AMSTERDAM

Final Lab Report: Part 2

CNNs for Image Classification

October 22, 2022

Students:
Roni Kremer
14564807

Tsatsral Mendsuren
14530775

Didier Merk
11037172

Tutor:
Wei Wang, Zehao Xiao

Group:
Practicals Group E

Course:
Computer Vision 1

1 Introduction

In the second part of the final lab for Computer Vision 1, several neural network architectures used for image classification are implemented. Namely, a Two Layer Neural Network (TLNN) and a Convolutional Neural Network (CNN) are used to classify images from the CIFAR-100 and STL-10 datasets into multiple classes.

In this report a custom dataset and data loader are introduced, the architectures of the two neural networks are defined and the loss and accuracy of said networks is evaluated. Finally, the hyperparameters of the two models are fine-tuned in order to achieve a higher prediction accuracy and explore variations of the original neural network architectures.

The two datasets used in this assignment are CIFAR-100 and STL-10. The CIFAR-100 dataset consists of 60,000 32x32 colour images in 100 classes, where each class contains 600 images. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 so-called *superclasses* Coates, Ng, and Lee 2011.

The STL-10 dataset is inspired by the CIFAR-10 dataset, however with some modifications. In particular, each class has fewer labelled training examples than the classes in

the CIFAR-10 datasets, but a very large set of unlabeled examples is provided to learn image models prior to the supervised training. The dataset consists of 10,000 images of 96x96 RGB images in 10 classes Adam Coates 2011. Each class contains 500 training and 800 test images. In this assignment, the focus will be on only five out of the ten classes.

Figure 1: 32 randomly chosen example images from the CIFAR-10 dataset.

2 Image Classification on CIFAR-100

2.1 Architecture of TwoLayerNet and ConvNet

As mentioned in the introduction, two neural networks are used to explore the classification of images from the CIFAR-100 dataset. In this section a short description of these two architectures (TwoLayerNet and ConvNet) are given.

The **TwoLayerNet** is a fully connected neural network with two hidden layers, which used the ReLU activation function. The model takes three variables that determine the input size, the dimensions of the hidden layers, and the number of classes to which the model will assign the images. Initially, the hidden layers in the Two Layer Network are set to a size of 512, input size of 3072 $(3 * 3 * 32)$, and 100 classes. In this configuration, the model has 1,624,676 trainable parameters.

The **ConvNet** is a convolutional neural network based on the architecture of LeNet5 Lecun et al. 1998. It consists of 7 layers in total: 2 convolutional layers with 2 subsampling layers in between, followed by 3 fully connected layers with ReLU activations. To decrease the amount of time it takes to train the model, 2D batch normalization with the

convolutional layers is used, together with max pooling for subsampling. The convolutional layers extract notable features from the images, followed by the dense layers that attempt to classify the images based on their features. In total, the ConvNet has 69,700 trainable parameters.

In the table below we can see the four different architectures and the corresponding test accuracies that have been measured. These accuracies can be seen as a sort of baseline, and in this report the aim is to improve the accuracies of the TwoLayerNet and ConvNet architectures

Model	**Parameters**	**Test acc.**
TwoLayerNet	1,624,676	0.22
FourLayerNet	848,996	0.23
ConvNet	69,700	**0.24**
ConvNetPlus	83,980	0.22

Table 1: In this table the amount of parameters for each architecture and their baseline test accuracies are displayed. In green, the highest test accuracy (the convolutional network) is highlighted.

2.2 Data Pre-processing

To prepare for the data for training, a custom CIFAR100 dataset class is defined, based on information from the PyTorch Dataset documentation. Once the dataset is loaded, a composition of transformations is applied to make the data more suitable for training. For the base, only the images are transformed into PyTorch tensor format and normalize the image pixel values.

After the transformation, a PyTorch DataLoader is used to make the dataset iterable and break down the dataset into batches. Batch size will play a crucial role later on, as it is one of the hyperparameters that will be tuned.

2.3 Training the models

To get a baseline training accuracy, we first had to define a loss function and optimizer. For the loss function, we chose Cross Entropy as it is better suited for classification tasks. We used Adam as our initial optimizer, as it is a safe choice and is known to reliably generate good results.

The models were trained with a batch size of 64 over 20 epochs. To get a performance baseline, the batch size and epoch number were chosen arbitrarily. As show in table 2, the results for the initial models were far from ideal, with the TwoLayerNet reaching a training accuracy of 44% and a test accuracy of only 22%, which may suggest that the model is overfitting. ConvNet had a training and testing accuracy of 34% and 24%, respectively. As later tests show, we can reach a test accuracy of approximately 30-40% on both models just by increasing the number of epochs.

Model	**Test acc.**	**Train acc.**
TwoLayerNet	0.22	**0.44**
FourLayerNet	0.23	0.26
ConvNet	**0.24**	0.34
ConvNetPlus	0.22	0.27

Table 2: Baseline test and training accuracies found using a loss function and optimizer. The accuracies are displayed for the four architectures, with the highest highlighted in green.

2.4 Model Tuning

To achieve higher performance, we explored the influence of different hyperparameters on the base architecture. First, we iterated over 16 combinations of learning rates and batch sizes to determine the best settings for the data loader and optimizer. We introduced two new transformations to the dataset by resizing and centring the images at 32 pixels to better fit the training process.

Finally, we implemented two new model architectures by adding two additional layers to the Two Layer Network and the ConvNet. We resize and centre the images, transform the images into PyTorch tensor format, and normalize

2.4.1 Hyper-parameter search

In the first step, we perform a grid search over 4 variations of learning rates and 4 variations of batch sizes. The learning rates $\boldsymbol{lr}$ were in the range of {0.01, 0.001, 0.0001, 0.00001}. The batches $\boldsymbol{b}$ were in the range of {128, 256, 512, 1024}. This produced 16 variations and allowed us to determine the optimal settings for the two hyperparameters.

The impact of batch size is that a smaller batch size means that there are more parameter updates (when keeping the number of epochs the same). Smaller batches (e.g. 4) work

better when the learning rate is also small (e.g. 0.0001). We also experience that a smaller batch size slows down training significantly. The large batch size (e.g. 1024) is much faster, but results in fewer update steps. Furthermore, with a larger batch size there is lower variance, so higher probability that the training gets stuck in a local minimum.

In figure 2 the best performing values for learning rate and batch size are displayed for both the Convolutional model and the TwoLayer Network. For the ConvNet algorithm the best performing learning rate was found to be 0.001 and the best performing batch size 256. For the TwoLayerNet algorithm the best performing learning rate and batch size were 0.001 and 128 respectively.

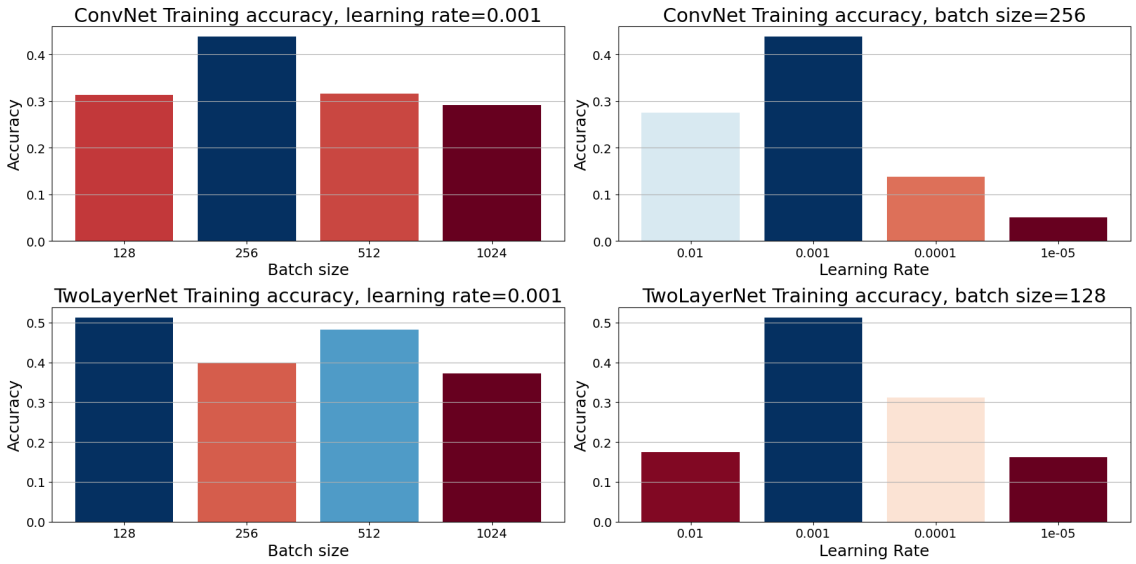

Figure 2: Training accuracies for the best performing learning rate and batch size hyperparameter values. On the y-axis the training accuracy is displayed and on the x-axis the different learning rates and batch sizes.

2.4.2 Architecture Adjustments

We modified the TLNN and CNN architectures by adding two additional layers to both models:

- FourLayerNet takes the original architecture of the TwoLayerNet and adds two hidden layers. In this configuration, the initial hidden layer is of size 256, and the 2 following layers are size 128. We made the additional layers smaller with the goal that they would learn the image features better.

- ConvNetPlus we added two fully connected layers before the final output layer to help the CNN classify images more accurately. As we are dealing with small images, the two convolutional layers in the original architecture were enough to extract meaningful features from the images. Same as in the original CNN, the two additional fully connected layers had ReLU activations.

Despite having additional layers, FourLayerNet and ConvNetPlus performed on average 2-5% worse than their original counterparts. Because we increased the model complexity, it does not mean that the model will perform better. On the contrary, without a large dataset, the more complex models are probably more likely to overfit, thus leading to reduced accuracy on the testing samples.

2.4.3 Loss and Optimizer experiments

For the loss function, we experimented with Mean Squared Error loss and Cross Entropy loss. MSE was very poorly suited for the task at hand, as it works best on regression tasks. We concluded that for multi-class classification, Cross Entropy consistently yields the best results.

We experimented with three optimizers: Stochastic Gradient Descent (SGD), Adam, and Adam with weight decay (AdamW). Although SGD combined with momentum is often used to achieve state-of-the-art performance, we consistently achieved the best results with AdamW. For all three of these optimizers, we used the same learning rate we derived in our hyperparameter search.

2.5 Results

To get our final results, we trained both models over 50 epochs with the best hyperparameters. For the Convolutional network, the batch size and learning rate that yielded the best results were 256 and 0.001, respectively. With these parameters, as shown in table 3, the network was able to reach a training accuracy of 35% and a test accuracy of 33%. This means an improvement of around a 30% for the test accuracy.

For the Two Layer network, the batch size and learning rate of 128 and 0.001 generated the highest accuracy. With the new parameters, the model achieved 49% training accuracy and 22% test accuracy. Although these models were not able to achieve state-of-the-art performance, we gained valuable insight into how specific parameters affect the performance of a neural network.

Model	Original train acc.	Final train acc.	Original test acc.	Final test acc.
ConvNet	0.34	**0.35**	0.24	**0.33**
TwoLayerNet	0.44	**0.49**	0.22	**0.22**

Table 3: Final train and test accuracies compared to their original value. Values highlighted in green show improvement from their original baseline value, values in blue show that no improvement was found.

In figure 3 below the training accuracy for both the Convolutional Net and the Two Layer Network are displayed using the chosen learning and batch size. From this figure it can easily be seen which settings of the hyperparameters cause the highest training accuracy. Figure 6 in the appendix shows the inverted figure.

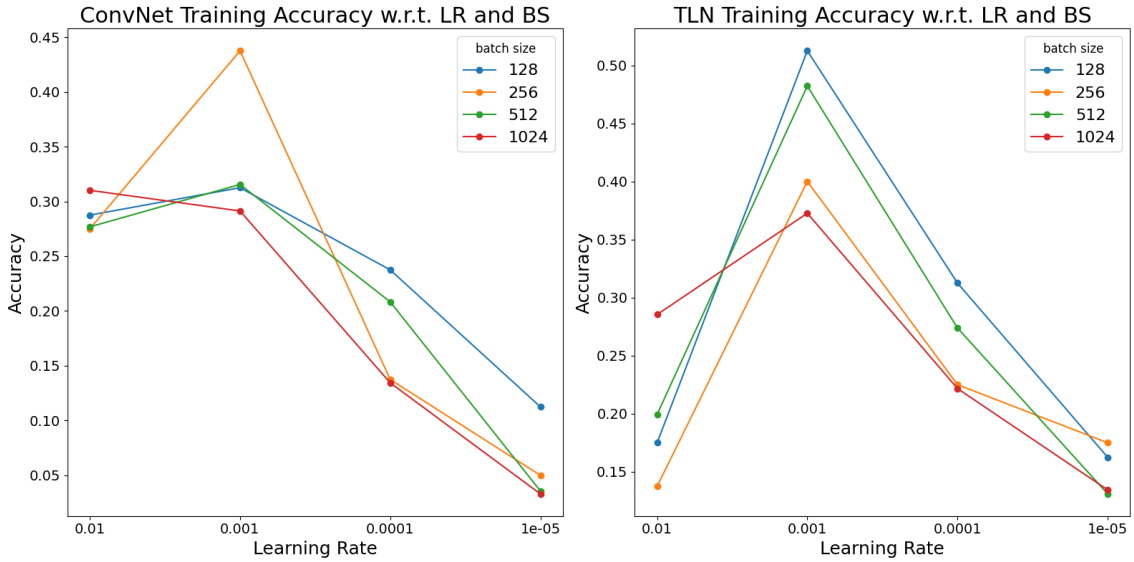

Figure 3: Training accuracies for both the ConvNet (left) and the TwoLayerNet (right). On the x-axis the different learning rates are displayed and on the y-axis the training accuracies are shown. The colour of the line declares the batch size.

3 Image Classification on STL-10

3.1 Data Pre-processing

For the STL-10 dataset, we define a custom dataset class that is similar to the CIFAR-100 dataset. The main difference is that we also introduce a mapping, which lets us select 5 classes from the dataset: 'car', 'deer', 'horse', 'monkey', and 'truck'. We use the same set of transformations as for CIFAR-100, but this time we resize and center the images at 96 pixels.

Figure 4: 25 randomly chosen example images from the STL-10 dataset, with 5 from each class.

3.2 Fine-tuning ConvNet for STL-10

To use ConvNet on the STL-10 dataset, we need to fine-tune the architecture, because the images have a higher resolution. Because the images are larger, the output of the last convolutional layer is now 16 channels of 21x21, instead of 16 by 5x5. To adjust for this, we changed the input size of the first fully connected layer and added an additional fully connected layer after it to improve feature classification.

We trained the STL-10 tuned ConvNet with a batch size of 128 and a learning rate of 0.001, over 50 epochs, and achieved an accuracy of 77% on the test dataset. As seen in figure 5, the model was most successful at classifying images of cars and trucks and had the lowest accuracy with the monkey class.

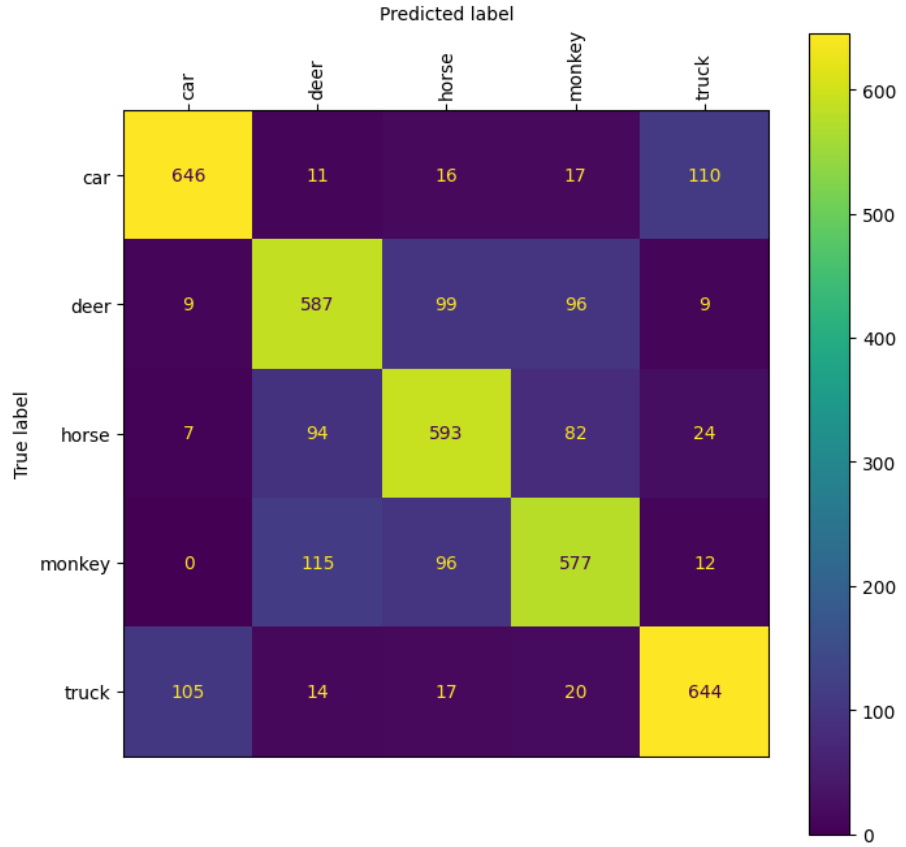

Figure 5: Confusion matrix displaying the results of the fine-tuned ConvNet classification architecture. The y-axis displays the true classes and x-axis displays the predicted class. It can be seen that a horse was predicted to be a deer 94 times, for example.

4 Conclusion

In this study, we experimented with different neural network architectures for multi-class classification of images. We found that even a simple convolutional neural network is more effective for this type of task than a two-layer network. Overall, we managed to achieve 77% accuracy of classification on the STL-10 dataset.

For further research, we could introduce more augmentation into our dataset to have more samples to train and test our models with. There is still work to be done in exploring the performance of the SGD optimizer for this test case, with different learning rates and

momentum values. Finally, we could experiment with other pre-trained convolutional neural networks such as ResNet, DenseNet, and VGG16.

References

Adam Coates Honglak Lee, Andrew Y. Ng (2011). "STL-10 Dataset". In: URL: http://cs.stanford.edu/~acoates/stl10.

Coates, Adam, Andrew Ng, and Honglak Lee (Nov. 2011). "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: Proceedings of Machine Learning Research 15. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík, pp. 215–223. URL: https://proceedings.mlr.press/v15/coates11a.html.

Lecun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

5 Appendix

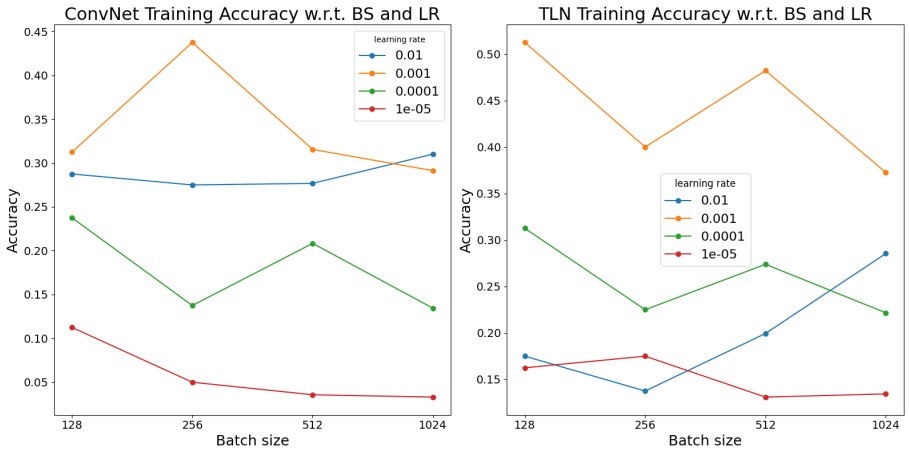

Figure 6: Training accuracies for both the ConvNet (left) and the TwoLayerNet (right). On the x-axis the different batch sizes are displayed and on the y-axis the training accuracies are shown. The colour of the line declares the learning rate.