DLVC Assignment 1

Daniel Zabielski, 12119018 || Sebastian Krähsmaier, 12122535 || Group 42

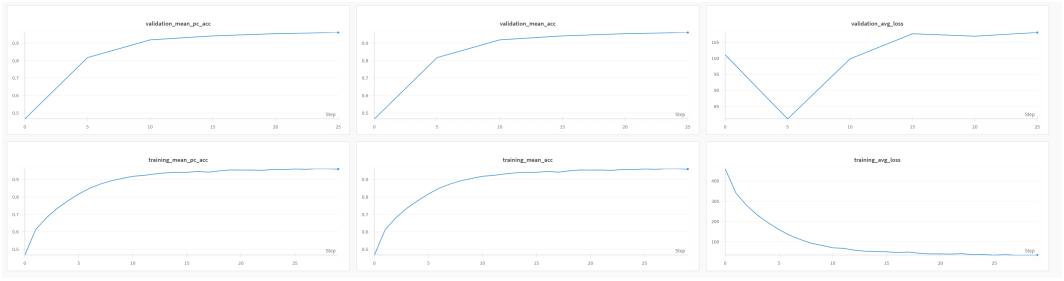
11.05.2025

1 What is the purpose of a loss function? What does the cross-entropy measure? Which criteria must the ground-truth labels and predicted class-scores fulfil to support the cross-entropy loss, and how is this ensured?

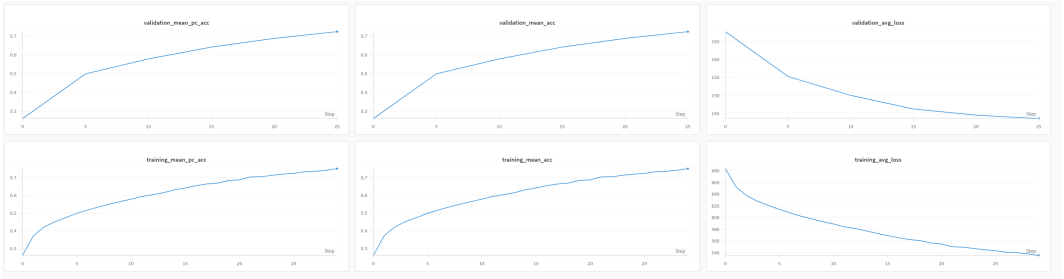
The purpose of a loss function is to evaluate the difference between the ground truth and the prediction of the model. The cross-entropy measures the probability of an image belonging to a specific class. The ground-truth labels should be one-hot encoded, and the prediction should be a probability. To ensure a valid probability, a softmax is applied.

2 What is the purpose of the training, validation, and test sets and why do we need all of them?

To ensure that the model also works on unseen and not just on the training data, the data is split into three parts. On the train-set the model is trained. The validation-set is used to tune the hyperparameters. And lastly, the test-set is used to evaluate the model on unseen data. Additionally, the validation set does not stay the same, for this the train-set gets split into, so called, folds and one of these folds become the validation-set and the old validation-set becomes part of the train-set. Doing this multiple times is called cross-validation, and the average results is taken for the model.



Figur 1: ResNet Results



Figur 2: CNN Results

3 What are the goals of data augmentation, regularization, and early stopping? How exactly did you use these techniques (hyperparameters, combinations) and what were your results (train, val and test performance)? List all experiments and results, even if they did not work well, and discuss them.

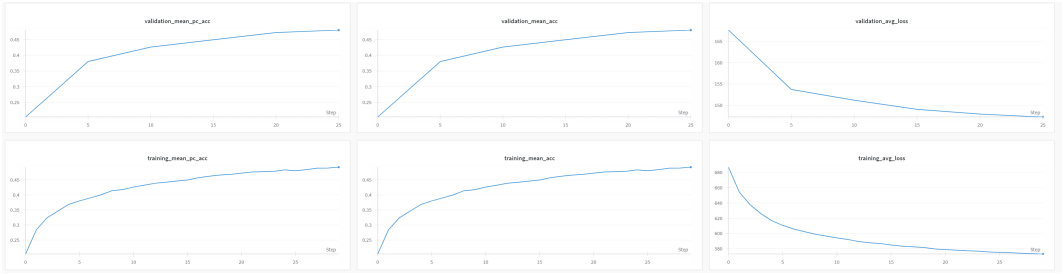
The purpose of data augmentation, regularization, and early stopping is to prevent overfitting. In our implementation, none of these techniques are used, although there are easy to implement as pytorch provides built-in options.

In Figures 1-3, the results of the ResNet, the CNN and the ViT are shown.

The ResNet shows a very high accuracy, but the loss is at its lowest at step 5 and rises afterward. This happens because of overfitting to prevent it, in this specific case early stopping can be implemented.

CNN achieves an accuracy of about 74%, to increase the accuracy, data augmentation can be used to broaden the dataset, as a too small data set or a data set that is too similar can be an issue.

The ViT only achieves an accuracy of about 45%, this can be due to a small dataset, as transformer in general need a lot more data than even CNNs to produce a suitable model.



Figur 3: ViT Results

4 What are the key differences between ViTs and CNNs? What are the underlying concepts, respectively? Give the source of your ViT model implementation and point to the specific lines of codes (in your code) where the key concept(s) of the ViT are and explain them.

The CNN process patches of a full image via sliding windows, whereas the ViT the images gets split into patches which get flattened. The CNN extracts local features, whereas the ViT extracts global relations based on the attention concept. The CNN is hard to parallelize, whereas the ViT architecture provides easy parallelization. In the CNN the layers can be divided into low, mid, high level features, where the ViT does not provide something like that. The amount of data to train a ViT is much larger than that of a CNN. Also, the inductive bias of CNNs is higher, since it assumes spatial locality and translation invariance.

CNN underlying concepts: The first concept isspatial position, CNNs assume that each pixel is related to their nearby pixels — this local connectivity allows the network to capture spatial patterns. The second concept is weight sharing, that means that the same convolutional kernel (filter) is applied across the entire image. This not only reduces the number of parameters but also makes the model translation-invariant. Third, CNNs perform hierarchical feature learning: early layers detect low-level features like edges and corners, while deeper layers learn more abstract features such as shapes and complex objects. Lastly, CNNs incorporate a strong inductive bias by assuming properties such as spatial locality and translation invariance. This means a smaller dataset for training is needed.

ViT underlying concepts: At the core is the use of attention, which allows the model to capture global relationships between all parts of the image. ViTs also rely on patch embedding and positional encoding: an image is split into fixed-size patches, each of which is flattened and interpreted as a vector. Since spatial order is lost during this process, positional embeddings are added to preserve spatial information. The main processing happens through layered transformer blocks, which consist of multi-head attention mechanisms, followed

by feedforward neural networks, along with layer normalization and residual connections. Finally, ViTs are characterized by having no built-in inductive bias. They do not assume spatial locality or translation invariance, on the side this also mean they need a lot of data to be trained.

The source for our ViT model is the exercise of the lecture Machine Learning", where we also had to implement a ViT. The patch embedding is performed in the class PatchEmbed. It transforms the input image into patches of equal size which then serve as tokens. Positional encoding and the class token are initialized in the __init__ method of the VisionTransformer class in lines 187 and 188. Positional encoding is applied to add a concept of position to the input, as transformers are inherently permutation invariant, i.e. the order of the input tokens does not matter. Since the transformer returns an output for each token, we introduce the class token to mark the position which will be used for classification. This is done to prevent any bias towards some parts of the input sequence. All these concepts are applied to the inputs in the prepare_tokens method.