Assignment 1 - Deep Learning for Visual Computing

May 6, 2024

1 Questions

1.1 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 fulfill to support the cross-entropy loss, and how is this ensured?

The purpose of the the loss function is to quantify a distance between the expected and the predicted results, basically answering how good the prediction is. The cross-entropy measures the distance between two probability distributions, in our case between the predicted and expected class. The lower the cross-entropy, the more accurate is the model. Furthermore, it is essential that the ground truths are consistent. Going on, one-hot encoding can also be helpful.

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

There is a need to ensure that the evaluation of the model is not done on data, that the model was already trained on. One wants to avoid data leakage. Therefore, there is a split for training and then verifying the data on the test set, such that the results of the testing are not falsified. Also, there is the need to avoid overfitting. The different methods usually have hyper parameters that have to be tuned. For this the validation dataset can be used.

1.3 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 goal of techniques like data augmentation, regularization and early stopping is to avoid overfitting. By that it is meant that the algorithm shouldn't "learn" to specific features. Therefore, the data

can be transformed, data augmentation or to stop methods earlier when using for example gradient descent, early stopping.

The regularization techniques were applied to all three models. Data augmentation was performed by random crops and left right mirroring of the cifar-10 images. For the CNN and the resnet18, a dropout layer with a probability of 50 % was added to the base architecture. In contrast, the vision transformer already includes two dropout layers within the attention blocks in its base form. These layers were initialized with a dropout rate of 20 %. The results for the base models and their regularized versions are displayed in table 1, which presents a comparison of performance metrics across different configurations on the test set.

Table 1: Performance metrics on test data

Model	Regularization technique	Mean accuracy (%)	Cross entropy loss
ResNet18	None	71.67	1.51
	Dropout layer	70.69	1.37
	Image augmentation	59.77	2.36
CNN	None	70.16	1.17
	Dropout layer	71.67	0.85
	Image augmentation	59.63	1.49
ViT	None	63.49	1.27
	Dropout layer	63.98	1.23
	Image augmentation	56.1	1.34

The logged training and validation performance metrics for the resnet18, convolutional neural network and vision transformer models are illustrated in figures 1, 2 and 3 respectively.

From the table and figures it is easy to see that the image augmentation technique gave the worst results, especially on the validation and test sets. One potential issue might be the randomized cropping of the images. The cifar-10 images are already quite small, 32 x 32. Aggressive cropping might lead to a loss of essential parts of the objects to classify. To prevent this, one could further experiment with the *scale* argument in the *torchvision.transforms.v2.RandomResizedCrop* function, that specifies the upper and lower bounds of the cropping area.

On the other hand, the addition of the dropout layer does not seem to improve or degrade the performance in any way. It can also be observed that for most models the performance on the validation set seems to peak at around epoch 15, irrespective of the chosen metric. Unfortunately, we weren't able to further experiment with the number of epochs, as most models were reaching the 1 hour limit one the cluster close to epoch 25. However, improving the epochs might only make sense when applying regularization techniques, as the models without regularization were approaching near perfect scores on the training set at around epochs 20 to 25. Regarding this, it might be beneficial to combine the regularization techniques dropout and data augmentation.

1.4 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.

CNNs work by applying a series of convolutional layers, focusing on local receptive fields and weight sharing to extract features from the input data. ViTs use the Transformer architecture to apply self-attention to capture global information across the entire image by weighing certain inputs more than other's. CNNs therefore focus more on local patterns, while ViTs can better understand long-range relationships through the mechanism of self-attention. The code for the VIT was obtained and adjusted for our code base as described in the task description [1]. The source code can be found in the file **vit.py** encompassing the architecture. In the vision transformer the images are divided into sequences of smaller patches, which are then handled similarly to tokens or words in the original transformer architecture. (function *img_to_patch()*). Positional encoding is added to these embedded patches to retain information about the original location of each patch within the image. Self attention is performed to allow the model to attend to different parts of the image, and to better understand the context and relationship between these different image areas (*self.attn = nn.MultiheadAttention(embed_dim, num_heads, dropout=dropout)*). The linear layer after self attention allows the model to learn non linear transformations (*self.linear = nn.Sequential(nn.Linear(embed_dim, hidden_dim), nn.GELU(), nn.Dropout(dropout), nn.Linear(hidden_dim, embed_dim), nn.Dropout(dropout))*. Finally the representation of the image, captured by the embedded cls token, is passed through a fully connected layer to output the class probabilities (*out = self.mlp_head(cls)*.

A Appendix

[1] https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial15/Vision_Transformer.html

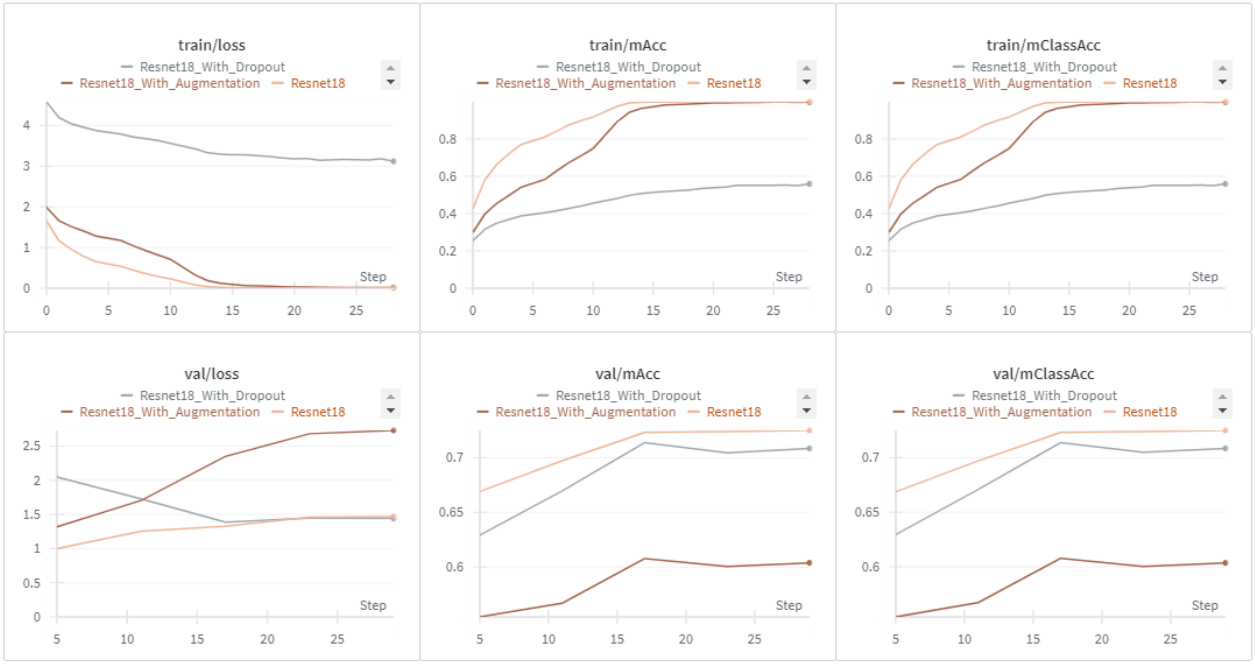

Figure 1: ResNet18 train and validation performance

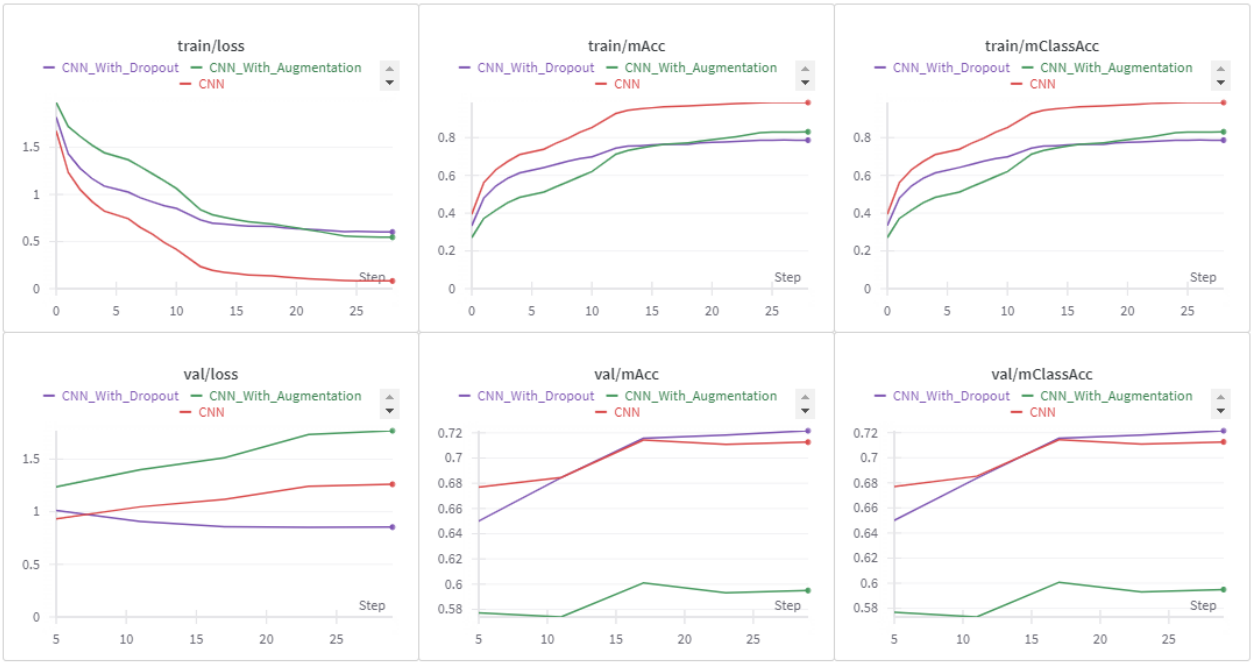

Figure 2: CNN train and validation performance

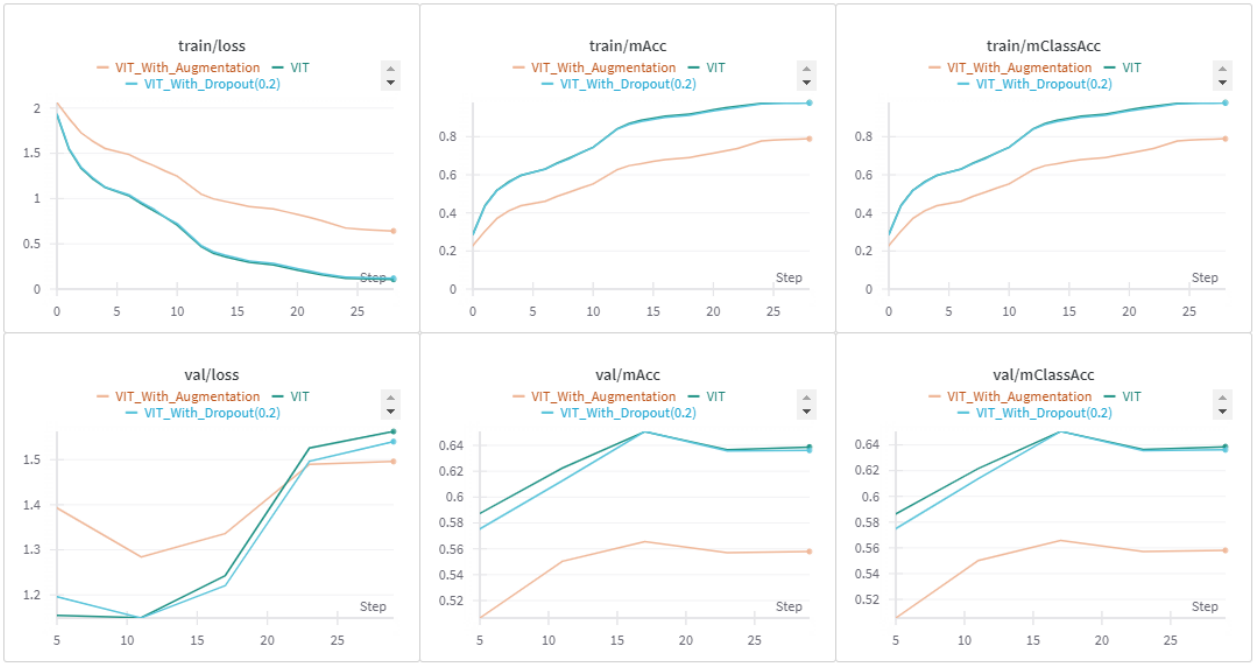

Figure 3: Vision transformer train and validation performance