Deep Learning (IST, 2024-25)  
Homework 2

Group 23

113191 – Miguel Miranda | 102787 – Ricardo Santos

**Question 2.**

**1.**

|  |  |
| --- | --- |
| Figure 1 - Validation accuracy as function of the epoch number for the CNN with learning rate 0.1 | Figure 2 - Training loss as a function of the epoch number for the CNN with learning rate 0.1 |
| Figure 3 - Validation accuracy as function of the epoch number for the CNN with learning rate 0.01 | Figure 4 - Training loss as function of the epoch number for the CNN with learning rate 0.01 |
| Figure 5 - Validation accuracy as function of the epoch number for the CNN with learning rate 0.001 | Figure 6 - Training loss as function of the epoch number for the CNN with learning rate 0.001 |

The configuration with best results was the one with learning rate of 0.01 as it achieved the lowest final training and validation losses with 0.7186 and 1.1986 respectively and the highest final validation and test accuracy with 0.6909 and 0.6797 respectively.

**2.**

|  |  |
| --- | --- |
| Figure 7 - Validation accuracy as function of the epoch number for the CNN with learning rate 0.01 and batch normalization | Figure 8 - Training loss as function of the epoch number for the CNN with learning rate 0.01 and batch normalization |

The configuration which included batch normalization, achieved improved performance compared to the baseline without batch normalization. The final test accuracy improved from 0.6797 to 0.7447, the validation accuracy improved from 0.6909 to 0.7486 and the validation loss decreased from 1.1986 to 0.7016. Using batch normalization led to a smoother convergence and better generalization on unseen data.

**3.**

The network without batch normalization, the total number of trainable parameters was 5340742, while in the network with batch normalization, it dropped significantly to 755718.

This reduction can be attributed to the use of global average pooling in the network with batch normalization. Instead of flattening the feature maps, global average pooling compresses them, which greatly reduces the number of inputs to the MLP layers.

**4.**

We use small kernels in convolution layers because small kernels are more efficient in feature extraction. Using small kernels allows the network to stack multiple convolutional layers, which creates a deeper architecture that captures hierarchical patterns in the data.

Pooling layers reduce the spatial dimensions of the feature maps, which helps reduce computational cost and the number of parameters in subsequent layers. They also summarize the features found in the feature maps.

The use of both small kernels and pooling layers are what contribute to the efficiency and effectiveness of CNNs.