Exercises Week 2

Eetu Knutars

January 21, 2025

1 Playing with a CNN on MNIST dataset

I implemented the CNN with PyTorch, using the same architecture as the example model implemented on TensorFlow. When loading the data, I used 5000 of the train data samples as the validation data. Train data size was then 55000 samples and test data 10000 samples. I trained the model and achieved $\sim$ 97% test accuracy.

In this task I was supposed to visualize and analyse the feature maps of the two Conv2D layers and the AvgPool2D layer. For the example image I use one on the train set that is labeled as 7.

The feature maps of the first convolutional layer are shown in Figure 1. This layer applies a 2D convolution using a 5x5 kernel, and adds padding of size 2 to the image so the resulting image shape is 28x28, same as the input image. The layer takes the one input channel and outputs 6 channels. The feature maps of the

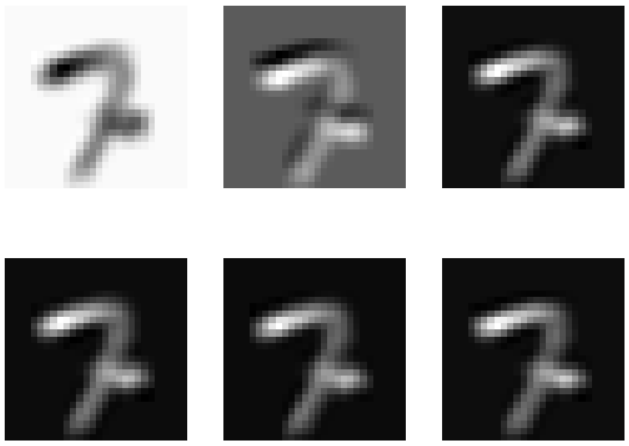
Figure 1: Feature maps of the first Conv2D layer

average pooling layer are shown in Figure 2. This layer applies an average pool with 2x2 kernels and stride of 2. This results a 14x14 sized image, so total image size is reduced to 1/4. The feature maps of the second convolutional layer are shown in Figure 3. This is the second convolutional layer, using a 5x5 kernel with no padding, so the result is 10x10 image. The layer takes 6 channels in and outputs 16 channels.

2 Classifying your own hand writing

In this task I had to draw digit by myself and test the trained CNN to classify them. I drew all nine different digits once and created 28x28 grayscaled images of them. I put them as inputs to the model and got the predicted values as the output.

The model didn't work too well on these digits, misclassifying three of them. The digits and their predicted labels are shown in Figure 4. Number 4 was misclassified as 9, number 7 was misclassified as 1 and number 9 was misclassified as 3. All of them are somewhat understandable misclassification because of the style of my hand writing.

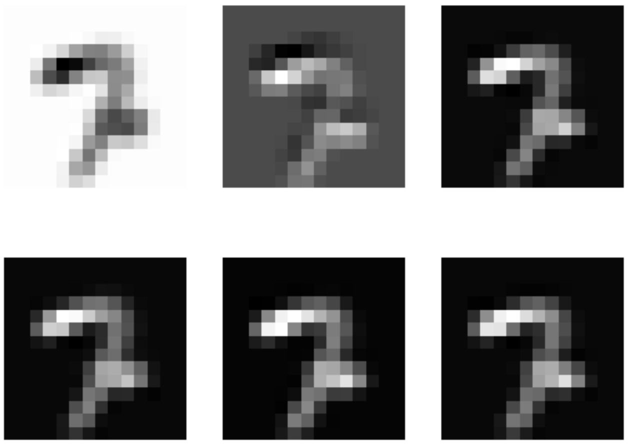

Figure 2: Feature maps of the AvgPool2D layer

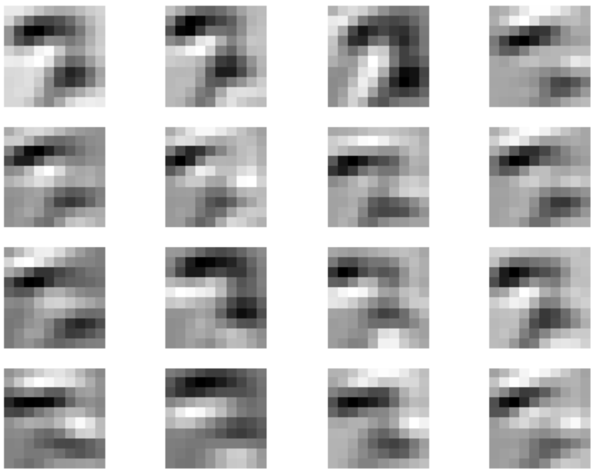

Figure 3: Feature maps of the second Conv2D layer

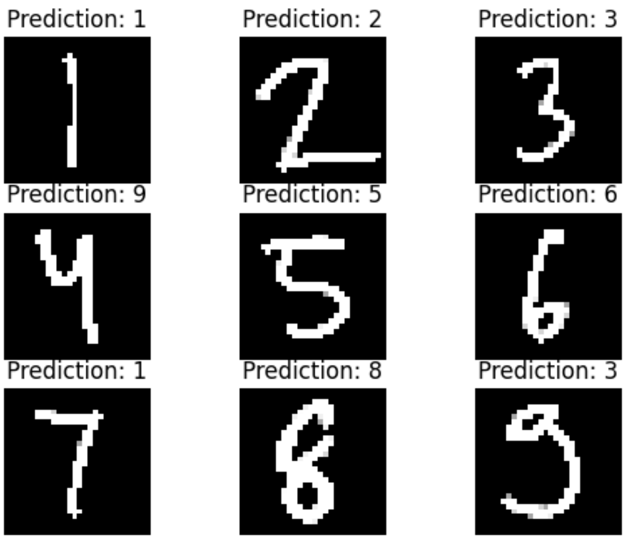

Figure 4: Feature maps of the second Conv2D layer

3 Experiencing with hyperparameters

Learning rate

For training this model I noticed learning rate 0.001 was pretty optimal. Too big learning rate changed the weights too much, so the training loss didn't improve. Smaller learning rate didn't change the weights enough, again not improving the training loss. Scheduler could be used during training, since it adjusts the learning rate on the go.

Epoch number

Too few epoch results the training loss not converging, and hence the test accuracy is not as good as it could be. Having too many epochs does no harm if using validation data, since the model will just stop improving. In this case its wise to stop training after validation loss has not improved in N epochs. If no validation data is used, the model might overfit to the training data.

Number of feature maps

Increasing the amount of feature maps naturally increases the amount of trained parameters in the model, making the training process computationally more expensive. Test accuracy can be improved a bit by increasing model complexity, however if the improvement is marginal then the simpler model may be a better option (Occam's razor)

4 Playing with CIFAR-100 dataset

In this task my goal was to experiment with the CIFAR-100 data set and in particular the coarse labels, which there are 20. The data set contains of 32x32 RGB images, and the goal was to implement a CNN to classify them with at least 40% test accuracy.

In my model I use four convolutional layers, batch normalization after every convolution layer and max pooling after every two convolutional layers. At the end I have two linear (dense) layers and one dropout layer with 50% dropout. I use ReLU activation in the model.

I tried different learning rates, and settled to 0.0005. I use a scheduler that adjusts the learning rate during training. When training the model I do 30 epochs at most and end the training after validation loss stops improving.

After training the model, test accuracy of $\sim$ 41.5% was achieved, not super good accuracy but acceptable for 20 class classification (random selection would have 5% accuracy).

The architecture of the modified model is shown in Figure 5. I only visualize the layers and not the specific dimensions just for the sake of simpler visualisation. The first purple layer in the beginning is the input image and the last purple layer in the end is the output layer. The model consists of convolutional layers (blue layers in the image), batch normalization (yellow layers), ReLU activation (smaller gray boxes), average pooling (orange layers), dense layers (red layers) and a dropout layer (light blue layer).

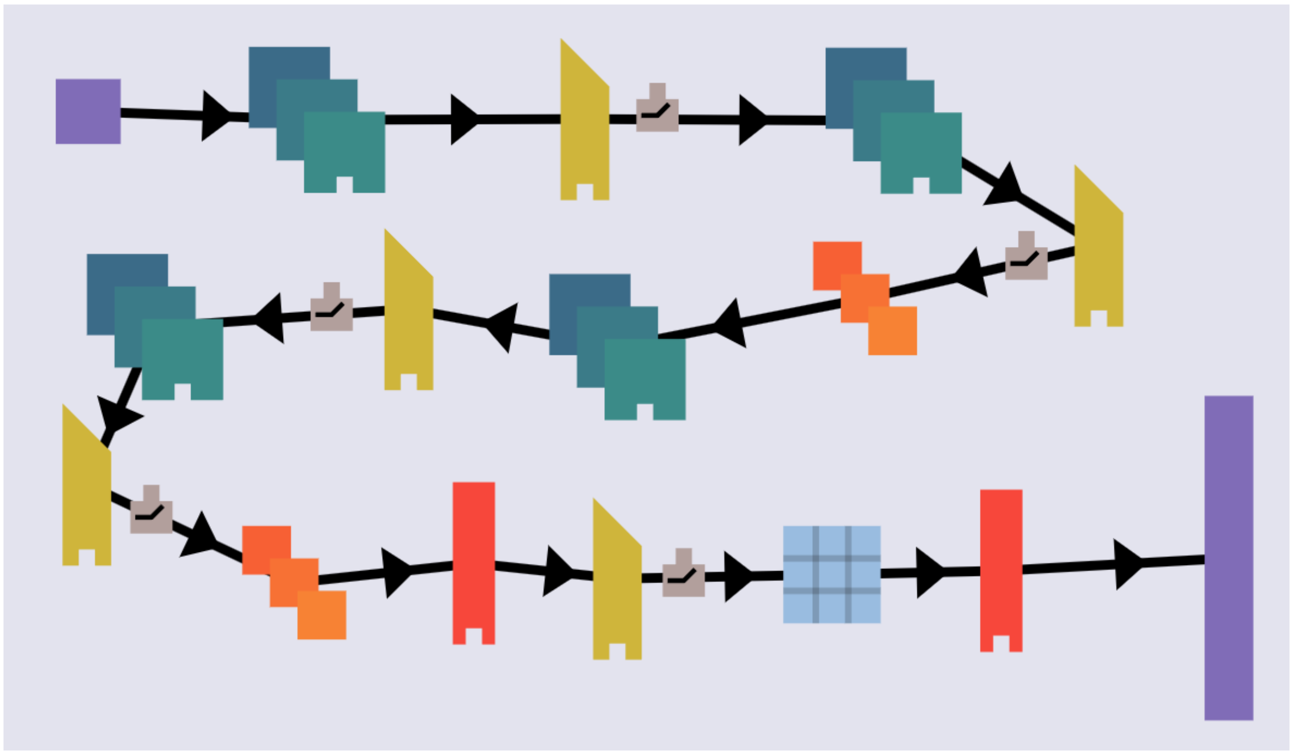

Figure 5: Modified CNN architecture