AssignmentReport-Group101

March 7, 2024

1 Assignment 3 Report

Marijan Soric and Zan Stanonik

2 Task 1

2.1 task 1a)

$$I = \begin{pmatrix} 2 & 1 & 2 & 3 & 1 \\ 3 & 9 & 1 & 1 & 4 \\ 4 & 5 & 0 & 7 & 0 \end{pmatrix} \quad K = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$I \star K \in \mathbb{R}^{3\times 5}$$

$$\begin{cases} (I\star K)_{1,1} = -(1\cdot 2 + 9\cdot 1) \\ (I\star K)_{2,1} = -(1\cdot 1 + 9\cdot 2 + 5\cdot 1) \\ (I\star K)_{3,1} = -(9\cdot 1 + 5\cdot 2) \end{cases} \quad \begin{cases} (I\star K)_{1,2} = -(2\cdot(-2) + 3\cdot(-1) + 2\cdot 2 + 1\cdot 1) \\ (I\star K)_{2,2} = -(2\cdot(-1) + 3\cdot(-2) + 4\cdot(-1) + 2\cdot 1 + 1\cdot 2 + 0\cdot 1) \\ (I\star K)_{3,2} = -(3\cdot(-1)4 + \cdot(-2) + 1\cdot 1) \end{cases}$$

$$\begin{cases} (I\star K)_{1,3} = -(1\cdot(-2) + 9\cdot(-1) + 3\cdot 2 + 1\cdot 1) \\ (I\star K)_{2,3} = -(1\cdot(-1) + 9\cdot(-2) + 5\cdot(-1) + 3\cdot 1 + 1\cdot 2 + 7\cdot 1) \\ (I\star K)_{3,3} = -(9\cdot(-1) + 5\cdot(-2) + 1\cdot 1 + 7\cdot 2) \end{cases}$$

$$\begin{cases} (I\star K)_{1,4} = -(2\cdot(-2) + 1\cdot(-1) + 1\cdot 2 + 4\cdot 1) \\ (I\star K)_{2,4} = -(2\cdot(-1) + 1\cdot(-2) + 1\cdot 1 + 4\cdot 2) \\ (I\star K)_{3,4} = -(1\cdot(-1) + \cdot(-1) + 4\cdot 1) \end{cases} \quad \begin{cases} (I\star K)_{1,5} = -(3\cdot(-2) + 1\cdot(-1)) \\ (I\star K)_{2,5} = -(3\cdot(-1) + 1\cdot(-2) + 7\cdot(-1)) \\ (I\star K)_{3,5} = -(1\cdot(-1) + 7\cdot(-2)) \end{cases}$$

$$I\star K = -\begin{pmatrix} 1\cdot 2 + 9\cdot 1 & 2\cdot(-2) + 3\cdot(-1) + 2\cdot 2 + 1\cdot 1 & 1\cdot(-2) + 9\cdot(-1) + 3\cdot 2 + 1\cdot 1 & 2\cdot(-2) + \\ 1\cdot 1 + 9\cdot 2 + 5\cdot 1 & 2\cdot(-1) + 3\cdot(-2) + 4\cdot(-1) + 2\cdot 1 + 1\cdot 2 + 0\cdot 1 & 1\cdot(-1) + 9\cdot(-2) + 5\cdot(-1) + 3\cdot 1 + 1\cdot 2 + 7\cdot 1 & 2\cdot(-1) + \\ 9\cdot 1 + 5\cdot 2 & 3\cdot(-1)4 + \cdot(-2) + 1\cdot 1 & 9\cdot(-1) + 5\cdot(-2) + 1\cdot 1 + 7\cdot 2 & 1\cdot(- \end{pmatrix}$$

$$I \star K = \begin{pmatrix} -11 & 2 & 4 & -1 & 7 \\ -24 & 8 & 12 & -5 & 12 \\ -19 & 10 & 4 & -3 & 15 \end{pmatrix}$$

[1]:
```
import numpy as np
import scipy
x = np.array([[2, 1, 2, 3, 1],
              [3, 9, 1, 1, 4],
              [4, 5, 0, 7, 0]])
```

```
y = np.array([[-1, 0, 1],
              [-2, 0, 2],
              [-1, 0, 1]])
scipy.ndimage.convolve(x,y,mode='constant')
```

```
[1]: array([[-11,   2,   4,  -1,   7],
            [-24,   8,  12,  -5,  12],
            [-19,  10,   4,  -3,  15]])
```

2.2 task 1b)

The **Max Pooling layer** reduces the sensitivity to small translational variations in the input.

2.3 task 1c)

Given a single convolutional layer with a stride $S = 1$, kernel size of 7×7 $F = 7$, and 6 filters. If you want the output shape (Height $\times$ Width) of the convolutional layer to be equal to the input image, how much padding P should you use on each side?

There are two equations that link output shape to the input shape:

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - F + 2\mathbf{P}}{S} \right\rfloor + 1 \quad H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - F + 2\mathbf{P}}{S} \right\rfloor + 1$$

Since, $W_{\text{out}} = W_{\text{in}}$ and $H_{\text{out}} = H_{\text{in}}$,

$$\lfloor 2\mathbf{P} \rfloor = F - 1 = 6 \Rightarrow \mathbf{P} = \mathbf{3}$$

2.4 task 1d)

You are told that the spatial dimensions of the feature maps in the first layer are 508×508 and that there are 12 feature maps in the first layer. Assuming that no padding is used, the stride is 1, and the kernel used is square and odd number size, what are the spatial dimensions of these kernels? Give the answer as (Height) $\times$ (Width).

$$W_{\text{out}} = \frac{W_{\text{in}} - \mathbf{F} + 2P}{S} + 1$$

With numerical values:

$$508 = \frac{512 - \mathbf{F} + 2 \times 0}{1} + 1$$

$$\mathbf{F} = 512 - 508 + 1 = \mathbf{5}$$

Same result for the width. Thus the kernel is in $\mathbb{R}^{5\times5}$

2.5 task 1e)

If subsampling is done after the first convolutional layer, using filters of size 2×2, with a stride of 2, what are the spatial dimensions of the pooled feature maps in the first pooling layer? Give the answer as (Height) $\times$ (Width).

After the pooling with size 2×2 and stride of 2, the spatial dimensions becomes $\left(\frac{508}{2}, \frac{508}{2}\right) =$ **(254,254)**.

2.6 task 1f)

The spatial dimensions of the convolution kernels in the second layer are 3 × 3. Assuming no padding and a stride of 1, what are the sizes of the feature maps in the second layer? Give the answer as (Height) × (Width).

$$\mathbf{W}_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - F + 2P}{S} \right\rfloor + 1$$

With numerical values:

$$\mathbf{W}_{\text{out}} = \frac{254 - 3 + 2 \times 0}{1} + 1$$

$$\mathbf{W}_{\text{out}} = 252$$

Same result for the width. Thus the sizes of the feature maps in the second layer are **(252,252)**.

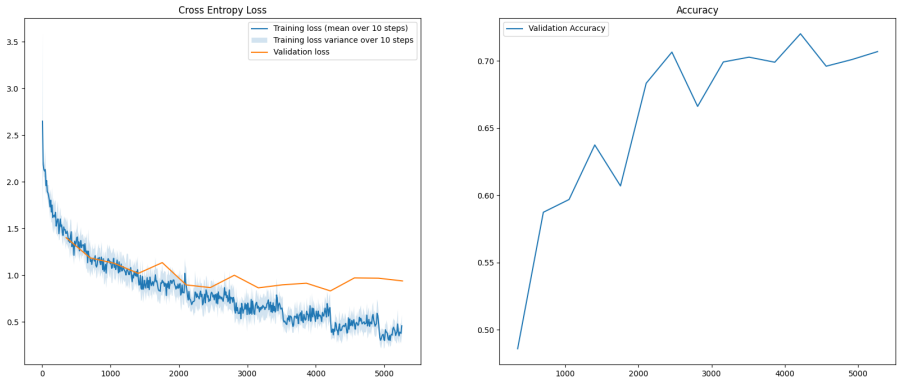
2.7 task 1g)

Let's count how many parameters there are in the network.

Layer	Type	Units / Filters	Input size	Output size	Number of parameters	Value
1	Conv+Pool	32	3x(32x32)	32x(16x16)	32x(3x(5x5)+1)	2 432
2	Conv+Pool	64	32x(16x16)	64x(8x8)	64x(32x(5x5)+1)	51 264
3	Conv+Pool	128	64x(8x8)	128x(4x4)	128x(64x(5x5)+1)	204 928
4	Fully	64	(128x4x3)	64	64x(128x(4x4)+1)	131 136
5	Fully	10	64	10	10x(64+1)	650
Sum	-	-	-	-	-	**390 410**

3 Task 2

3.0.1 Task 2a)



We can notice that the training loss is decreasing, but the validation loss is stagnating and starting to increase which is a sign of overfitting.

Validation loss is 0.83, and the validation accuracy is 71.9%.

3.0.2 Task 2b)

The final losses & accuracies are the following:

Data	Accuracy
Training	86.9 %
Validation	71.9 %
Test	71.8 %

4 Task 3

Layer	Layer Type	Number of Hidden Units / Number of Filters
1	Conv2D	32
	BatchNorm2d	-
	ReLU	-
	Conv2D	32
	BatchNorm2d	-
	ReLU	-
	MaxPool2D	-
2	Conv2D	64
	BatchNorm2d	-
	ReLU	-
	Conv2D	64
	BatchNorm2d	-
	ReLU	-
	MaxPool2D	-
3	Conv2D	128
	BatchNorm2d	-
	ReLU	-
	Conv2D	128
	BatchNorm2d	-
	ReLU	-
	MaxPool2D	-
4	Conv2D	256
	BatchNorm2d	-
	ReLU	-
	Conv2D	256
	BatchNorm2d	-
	ReLU	-
	MaxPool2D	-
	Flatten	-
5	Fully-Connected	64
	BatchNorm1d	-
6	Fully-Connected	num_classes

Layer	Layer Type	Number of Hidden Units / Number of Filters
	BatchNorm1d	-

The network by our design hence has 6 layers, made out of 4 exactly the same block with different filter sizes. It could be argued that there are actually 10 layers, but for us one layer is conv2d-batchnorm2-relu-conv2d-batchnorm2d-relu-maxpool2d, similar to what was suggested in the instructions pdf.

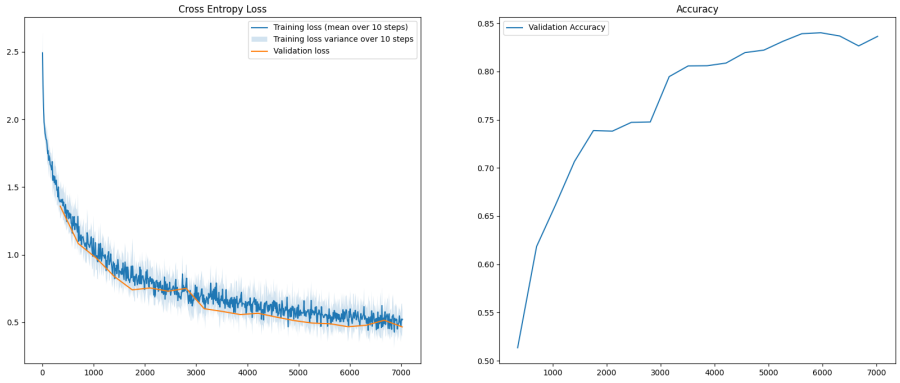
Optimizer used in the final model is AdaGrad, for regularization weight decay of 1e-5 in the optimizer was used. We also tested out dropout, but the results in combination with weight decay looked the same or worse so Dropout was removed.

Learning rate was kept the same as before at 5e-2, same goes for batch size (64). All Linear and Conv2d units of the network were initialized with the Kaiming normal method. Apart from that, only the dataset loaders were changed so that the training loader transforms the images in the following way:

Transformation	Description
RandomHorizontalFlip()	Randomly flip the input horizontally with a 50% chance.
RandomRotation(10)	Rotate the input randomly by a degree chosen uniformly from the range [-10, 10].
RandomCrop(32, padding=4)	Crop the input at a random location with padding of 4 and size of 32x32.
ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2)	Randomly change the brightness, contrast, and saturation of an image.

4.0.1 Task 3b)

Data	Loss	Accuracy
Training	0.48	83.9 %
Validation	0.48	83.9 %
Test	0.48	83.6 %



4.0.2 Task 3c)

What worked The biggest improvements were seen by increasing the number of trainable parameters in the model, changing the optimizer to Adagrad, and by adding transformations to the images of the training set. In my opinion all of these methods worked because of the following: - increasing the number of layers, it allowed the network to learn more parameters because we were limited with the number of epochs, and the intial network was maybe a bit to small to store all the important information. - optimizer, Adagrad in my opinion performed better because it adapts the learning rate for each parameter based on the past ones, which can prevent the algorithm from overshootign the minimum. From iits use it other areas it has also been shown that it performs better on smaller datasets, since it delivers larger updates to infrequent parameters and smaller updates to frequent ones. - tranformations, due to us tranforming the data, we had "more" images to train on, which lead to network capturing the actual pattern in the data that contribute to class predictions, hence increasing the accuracy of the network.

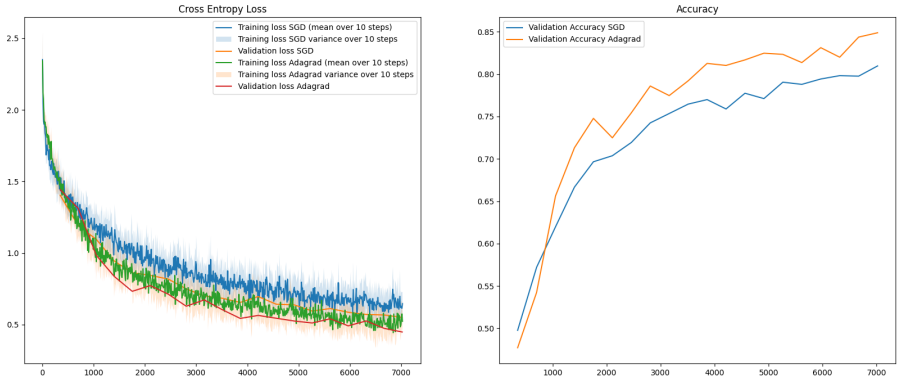
What didn't

- Dropout of 0.2, it didn't seem to improve the accuracy of the network from which we can assume that the weight decay was sufficient to regularize the network.
- Replacing the activation function ReLU, we had tested LeakyReLU and SELU as alternative functions. But they didn't seem to perform better so the original activation function was kept.

4.0.3 Task 3d)

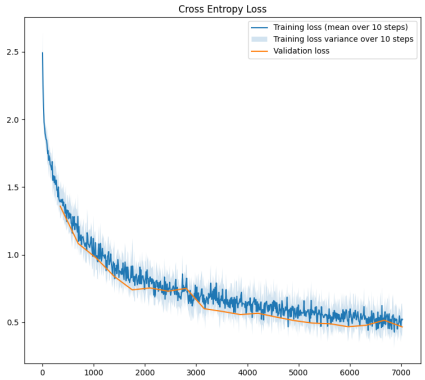
The most significant improvement apart from modifying the network arhitecture (where it is hard to narrow down which exact part contributed the most), was the replacement of SGD with Adagrad. This alone contributed ~4% to the accuracy increase of the model predictions.

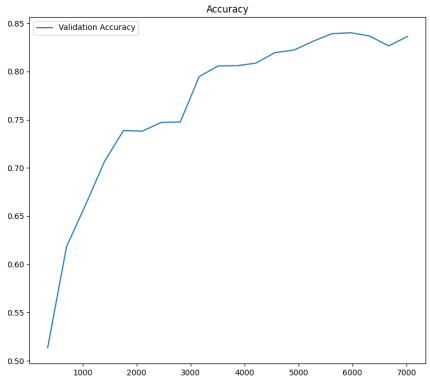
Optimizer	Test accuracy
SGD	80.2%
Adagrad	84.6%



4.0.4 Task 3e)

The plot is the same as for 3b, since I described the final model which performed best.





4.0.5 Task 3f)

For the best model the loss seems to be starting to stagnate, but it isn't quite there yet. I would say that if the model ran for longer we could see this in action, which is probably the result of the overfitting in this specific case. Overall I would say the model performs quite good, since the training and validation accuracies are close to the test one, which means that the model learned actual data patterns and is not overfitting with the help of memorization.

5 Task 4

5.1 Task 4a)

We implemented transfer learning with Resnet18. Hyperparameters used:

Hyperparameter	Value
optimizer	Adam Optimizer
batch size	64
learning rate	5 x 10^{-5}

Data transformation and data augmentation used:

Transformation	Value
Resize	(112,112)
RandomHorizontalFlip	p=0.5
ColorJitter	brightness=0.5
RandomRotation	degrees=15
RandomGrayscale	p=0.2
GaussianBlur	3, sigma=(0.1, 2.0)
Normalize	(mean, std)

Our final test accuracy: **89.9%**.