Excercise 4

Arnas Šniokaitis

November 22, 2025

1 Excercise 1

1.1 Briefly Explain the Term Inductive Bias

An inductive bias is how we assume what determines the output(class for classification). For example if we use a linear model to determine a class, we assume that the value of the data can be linearly computed/determined from their features.

1.2 Explain the purpose of pooling layers in CNNs. Explain what the term "stride" means in this context

Pooling is a method, by which data is combined to a smaller range. This allows us to have fewer parameters and is especially usefull, if local changes are negligable. With pooling we 'Divide and conquer', for instance, image data, since local changes are usually negligable. Stride here means the 'Step size'. If we have a stride of s, then after taking our first propagation, we skip $s-1$ cells, and then continue at the s-th cell.

1.3 Name three specific and distinct strategies (i.e., more specific than "pooling layers") on how you can reduce the spatial size of a feature map

Convolution without padding, max-pooling, average-pooling.

1.4 Consider a network architecture with the following layers, which receives a 3-channel image (RGB) as input: * Assume a scalar bias per convolutional kernel / node where bias is true. Compute the number of parameters for the whole network

To calculate the parameters for the network we simply have to sum up the parameters for each layer. For the first layer we have a parameter for each 'cell' of the convolution. For the first layer we have that it is a 7x7 convolution and since the convvolution has to go through each of the channels, we have that it has $7 * 7 * 3 + 1 = 148$ parameters. Since we have 16 output channels, this imples that we have 16 filters, which each get unique weights as well, which bring the total count for the first layer to $148 * 16 = 2368$ parameters. For the

second layer we accordingly calculate $3 * 3 * 16 + 1 = 145$ and since we have 16 output channels this brings our count for the second layer to $145 * 16 = 2320$. The third layer is a max-pooling layer which doesn't have any arguments. For the fourth layer we accordingly calculate $3 * 3 * 16 + 1 = 145$ Since we have 32 output channels we get $145 * 32 = 4640$. For the fifth layer we calculate $2 * 2 * 32 = 128$ $128 * 32 = 4096$ The global-average Layer doesn't take any arguments. And lastly we have a fully-connected layer with two outputs so we get $32 * 2 + 2 = 66$ parameters. Here we ad two, since each output channel has its own unique bias. For the final layer we have the Softmax function, which as well doesn't take any parameters. So in total across the whole network we have $2368 + 2320 + 4640 + 4096 + 66 = 13490$ total parameters.

1.5 Compute the size of the output of layer 5 for an image of size 32x32

Since we have an RGB image, we have 3 channels, as such the full breadth of our input is 32x32x3.After the first layer, since we have 'same' padding we also get back a 32x32 size, but since we get 16 output channels we get that we have 32x32x16. Similarly after passing layer two, since we use same padding and also have 16 output channels, we get that the size of our input after layer two is also 32x32x16. Since our third layer is a max-pooling layer with a stride of 2. Our size gets cut in half; the number of channels remain unchanged as such we get 16x16x32. For the same reasons as for layers one and two, after passing the fourth layer our data size becomes after passing layer four: 16x16x32 and 16x16x32 for layer five. As such the output after layer 5 is of size: **16x16x32**

2 Excercise 2

2.1 What is regularization, and why is it important for machine learning?

Regularization is a method by which we avoid overfitting; most commonly by drop-out or punishing large parameter values. This is important because our model can become trained to seperate the training dadata "too aggressively" and leads to poor generalization.

2.2 Briefly describe why data augmentation can be considered as a form of implicit regularization

Augmentation can be viewed as regularization because it causes variance in the training data that prevents the model from learning, for instance; for images the values for specific pixels. As such it becomes harder for the modelto overfit the data.

2.3 Which ofhe provided ssrategy will not provide additional learning signals to the provided CNN?

Translation to the left by 4 pixels would not provide additional info. This is because we use a global average. after translation the average doesn't change as

such there is no way by which it can be determined if we provided a translated or original input. As such the model will view them as the same.

2.4 Show that during inference, we have to multiply each activation by 1-p to achieve the sa,e expected value

If the expected value for all of the nodes is A then, since we keep each node with probablilty $1-p$ and since each layer is linear, we get that the expected value drops by $1-p$ $\mathbb{E}[(1-p)X] = (1-p)\mathbb{E}[X] = (1-p)A$ as such when we use the full layer and get the mean A, to make sure that it the same as the drop-out mean, we have to multiply by $(1-p)$.

3 run.py tasks

The following are images aquired during the training of the model.

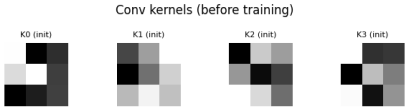

Figure 1: Randomly Initialized Kernels

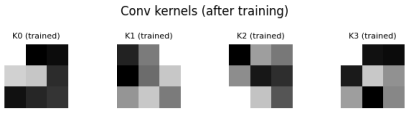

Figure 2: Trained Kernels

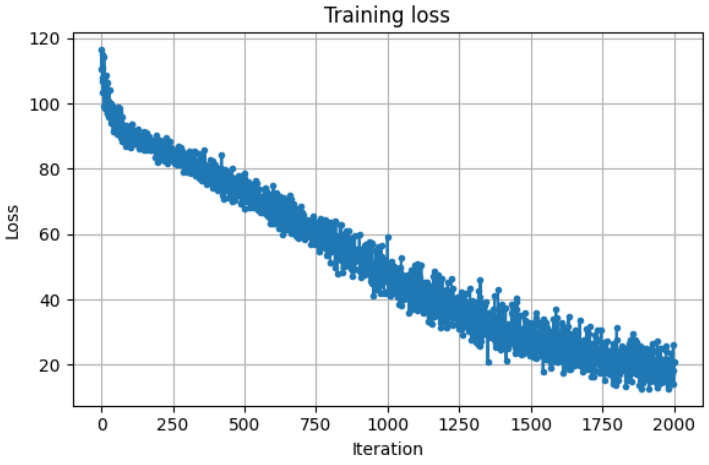

Figure 3: A graph of the error function

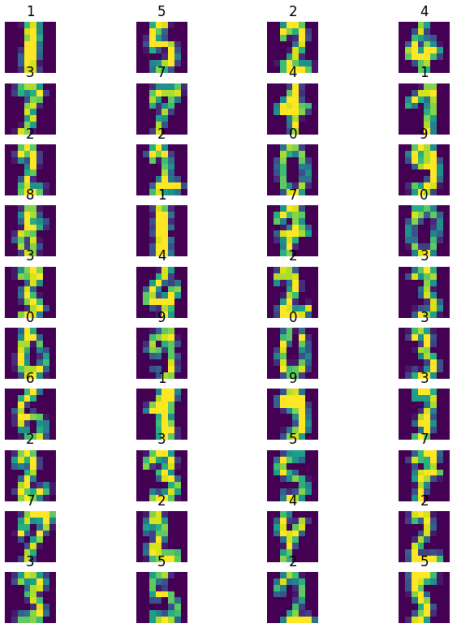

Figure 4: A part of the reaining set for the model

After running the model for 10 times I achieved the following accuracies.

1. 86,97829%
2. 89,48247%
3. 91,31886%
4. 82,97162%
5. 85,47579%
6. 85,64274%
7. 84,97496%
8. 79,29883%
9. 87,14254%
10. 78,29716%

The mean of these 10 iterations is $85,16\%$ The standard deviation is $3,88\%$