Deep Learning - 2022/2023

Homework II – Group 05

95604 João Bagorro 95617 Juliana Yang

Work Distribution

The coding exercises were done by both students which in turn were compared, choosing the best implementation, the testing and bug solving was done as a group. Questions number 1, 2.1, 2.2, 2.3, 2.5 and 3.3 were solved together. Both students contributed equally.

Question 1

1.

a. Given the following formulas:

$$output\ witdh = \frac{input\ witdh - kernel\ witdh + 2 * padding\ witdh}{stride} + 1 = W - N + 1$$

$$output\ height = \frac{input\ height - kernel\ height + 2 * padding\ height}{stride} + 1 = H - M + 1$$

Then we could say that: $z \in \mathbb{R}^{(H-M+1)\times\,(W-N+1)}$

b. Considering z, x and W we can get the formula (1) below:

$z_{ab} = \sum_{i=1}^{M}\sum_{i=1}^{N} w_{ij} x_{(i+a-1)(j+b-1)}$, where $1 \leq a \leq H$ and $1 \leq b \leq W$

Now we need to transform x into x' and w_{ij} into some M_{kl}. We know that each element of z' (z_{ab}) can be obtained by dot product between the corresponding row of z_{ab} in M and x'. So, we can deduce that $z_{ab} = M_{W'(a-1)+b}\ x'$ and by iterating over x we get (2):

$$z_{ab} = \sum_{i=1}^{M}\sum_{i=1}^{N} M_{[W'(a-1)+b],[W(i-1)+j]}\ x_{ij}$$

Finally, from (1) we know that not all elements M is relevant. This means that some elements will get be zero in order to ignore them and other values should be w_{ij}. Therefore, we have for each z_{ab}:

$$M_{[W'(a-1)+b],[W(i-1)+j]} = \begin{cases} 0 & if\ a \leq i \leq M + a - 1\ and\ b \leq j \leq N + b - 1 \\ w_{ij} & otherwise \end{cases}$$

c.

- CNN and Max Pooling:
  - Convolutional Layer: $Nr\ parameters = M \times N$
  - Max Pooling Layer: no parameters
  - Output Layer: $Nr\ parameters = dimension\ of\ output\ \times\ output\ dimension\ of\ MaxPooling = \frac{3(H-M+1)(W-N+1)}{4}$
  - Total: $Nr\ final\ parameters = M \times N + \frac{3(H-M+1)(W-N+1)}{4}$
- FNN:
  - Hidden Layer: $Nr\ parameters = HW \times HW \frac{(H-M+1)(W-N+1)}{4}$
  - Output Layer: $Nr\ parameters = \frac{3(H-M+1)(W-N+1)}{4}$
  - Total: $Nr\ final\ parameters = HW \times \frac{(H-M+1)(W-N+1)}{4} + \frac{3(H-M+1)(W-N+1)}{4}$

2. We have: $Q = x'W_Q = x'$; $K = x'W_k = x'$; $V = x'W_V = x'$.

The attention probabilities are given by computing scaled dot product attention and applying softmax row-wise: $P = softmax(\frac{QK^T}{\sqrt{1}})$.

Finally, the output is: $Z = PV$.

Question 2

1. A CNN has fewer free parameters than a fully-connected network with the same input size and the same number of classes because of the way it processes the data. In a fully-connected network, each neuron in one layer is connected to every neuron in the next layer. This results in a large number of parameters that need to be learned, which increases as the number of layers in the network increases.

In contrast, a CNN uses convolutional and pooling layers, which reduces the number of parameters that need to be learned. In a convolutional layer, each neuron is only connected to a small, localized region of the previous layer, called a filter. This allows the CNN to learn features from small patches of the input data, and then use these features to classify the input. The pooling layers reduce the dimension of the data and also help in reducing the number of parameters. Because each filter is only connected to a small subset of the input, the number of parameters in a CNN is typically much smaller than in a fully-connected network, even if the input size and the number of classes are the same. Additionally, the shared parameters across all the locations in an image in CNN, also helps reducing the overall number of parameters.

All of these factors together lead to CNNs having fewer free parameters than a fully-connected network with the same input size and the same number of classes.

2. Convolutional Neural Networks (CNNs) are good for image and pattern recognition tasks because they can learn translation-invariant features. This means that a feature learned by a CNN at one location in an image is likely to be useful in identifying the same feature at other locations in the image, like edges and textures, this can be called "parameter sharing". Unlike fully connected networks, which do not have this ability and must learn a separate set of parameters for each location.

Additionally, CNNs use a combination of convolutional and pooling layers to reduce the dimensionality of the data as it passes through the network. This reduction in dimensionality not only helps to reduce the number of free parameters in the network as explained in the previous question, but also helps to make the network more robust to minor changes in the position of the image.

So, in summary, CNNs achieves better generalization on image and pattern recognition tasks due to its ability to learn translation-invariant features, dimensionality reduction, and techniques to prevent overfitting.

3. If the input data is from a source composed of independent sensors with no spatial structure, a CNN may not be the best choice for achieving good generalization. This is because the main advantage of CNNs is their ability to learn translation-invariant features, which is particularly useful for image and pattern recognition tasks where there is spatial structure in the data. In the case of independent sensors with no spatial structure, the data does not have any inherent spatial relationships, so the ability to learn translation-invariant features would not be as useful.

Fully connected networks, on the other hand, can still be used to model this kind of data, as they are not restricted to learning spatial relationships. They are capable of learning any kind of relationship, be it spatial or not, in the data. Additionally, the data might contain correlation between the sensor data but will not have any "spatial" relationship in data.

Therefore, in cases where the input data is from independent sensors with no spatial structure, a fully connected network may be better suited to achieve good generalization than a CNN.

4. **Best configuration:** Adam, 0.0005 learning rate

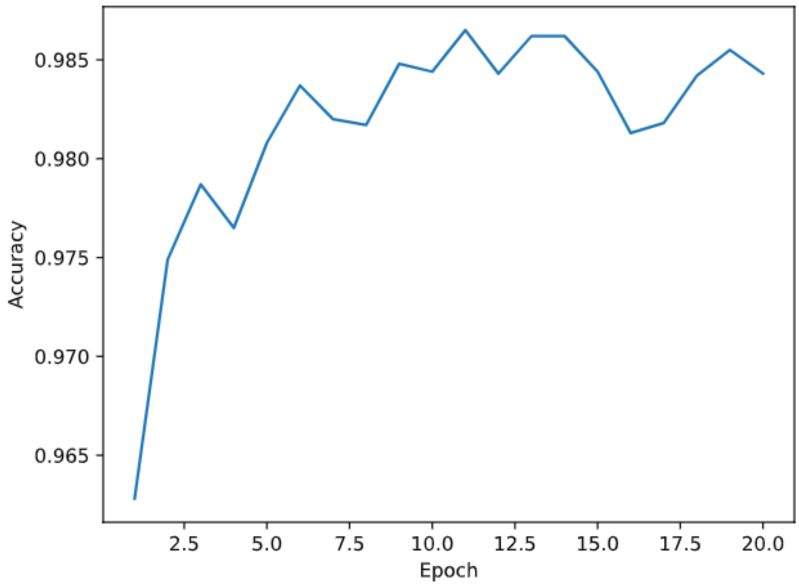

Figure 1- Validation Accuracy of the best configuration

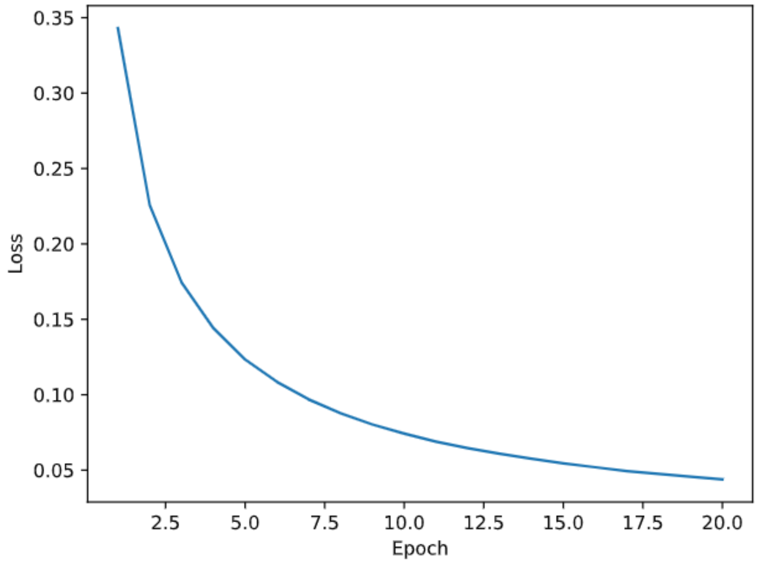

Figure 2 - Training Loss of the best configuration

5. The activation maps after the convolution layer highlight the features of the input image that were most important for the next layer to make a prediction. These features are the parts of the image that contain the most relevant information, such as edges or textures. The activation maps are also smaller in size than the original image, as the convolution operation reduces the spatial dimensions of the feature maps.

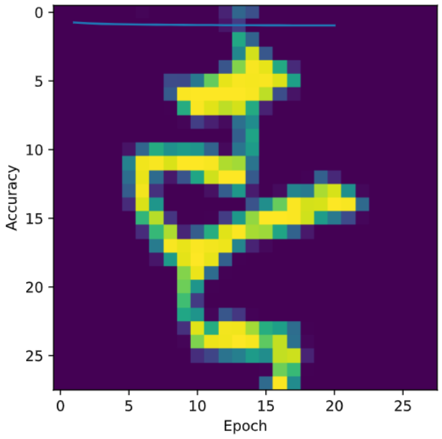

Figure 4 - Original Image

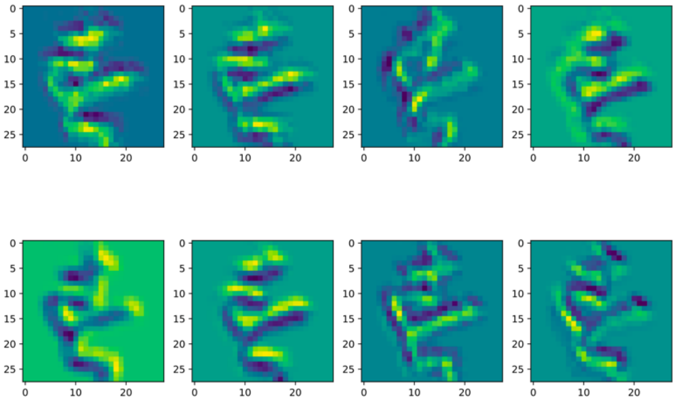

Figure 3 - Activation Maps

Question 3

1. Final Error Rate in the Test set: 0.5001

Final validation error rate: 0.4995

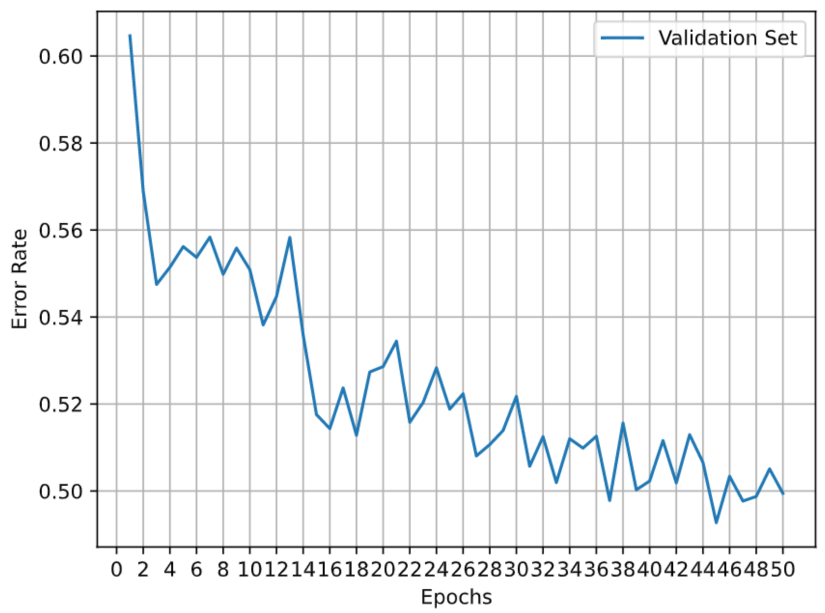

Figure 5 - Validation Error Rate without Attention

Deep Learning - 2022/2023

Homework II – Group 05

95604 João Bagorro 95617 Juliana Yang

2. Final Error rate in the Test set: 0.3782

Final validation error rate: 0.3715

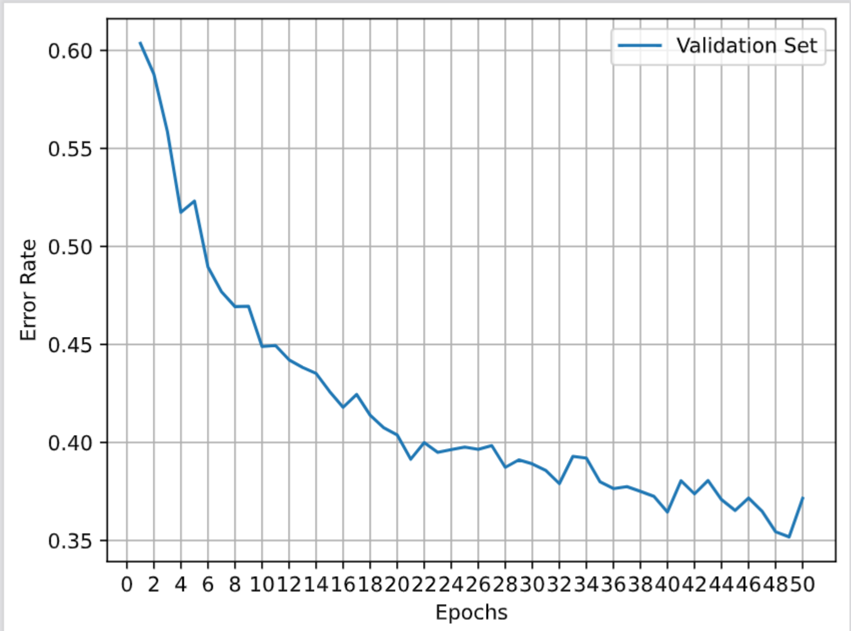

Figure 6 - Validation Error Rate with Attention

3. Looking at the decoding process in the test() section of the hw2-q3.py file we can see that it uses a greedy search algorithm to select the word with the highest attention weight and probability at each decoding step. This method is simple to implement and computationally efficient, but it has the drawback of not considering the context of the previous decoding steps, which may lead to sub-optimal or even incorrect final output. In other words, the model might be getting "stuck" in local best solutions without being able to find the best solution overall. A way to improve this would be to use a beam search algorithm, although it would increase the computation time.