Neural Networks Homework 10

Sayeh Jarollahi (7073520, saja00006@stud.uni-saarland.de)
Mahsa Amani (7064006, maam00002@stud.uni-saarland.de)

January 25, 2025

Exercise 10.1

Proof. a) This will cause symmetry breaking as every neuron in a layer will produce the same output and have the same gradients during backpropagation. Consequently, the network cannot learn effectively, as all neurons remain redundant and do not learn diverse features.
b) When using ReLU activation functions, initializing biases to small positive values such as 0.01 ensures that all ReLU units fire in the beginning and therefore obtain and propagate some gradient. □

Exercise 10.2

Proof. a) CNNs excel with images by effectively capturing spatial hierarchies and local patterns, such as edges and textures, through convolutional filters. They can be applied to text represented spatially, like word embeddings with positional structure, but recurrent or transformer-based models typically better capture sequential and contextual dependencies in text.
b) By flattening the input matrix we will have:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{bmatrix}^T$$

Now considering kernel sliding and the parts of input that kernel covers, we have the following 4×9 matrix for the convolution operation is:

$$\mathbf{W} = \begin{bmatrix} w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 \end{bmatrix}$$

By multiplying $\mathbf{W}$ with the flattened input $\mathbf{x}$ we will get the output vector $\mathbf{y} \in \mathbb{R}^4$, representing the convolution result.

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x}$$

c) By expanding the recursion:

$$R_\ell = R_{\ell-1} + (k-1)\prod_{i=1}^{\ell-1} s_i$$

$$R_{\ell-1} = R_{\ell-2} + (k-1)\prod_{i=1}^{\ell-2} s_i$$

$$R_{\ell-2} = R_{\ell-3} + (k-1)\prod_{i=1}^{\ell-3} s_i$$

and so on, until:

$$R_1 = R_0 + (k-1)\prod_{i=1}^{0} s_i$$

Now we sum up the terms and replace R_0 with 1:

$$R_\ell = R_0 + (k-1)\sum_{j=1}^{\ell}\prod_{i=1}^{j-1} s_i = 1 + (k-1)\sum_{j=1}^{\ell}\prod_{i=1}^{j-1} s_i$$

d)
a) The formula for output dimensions of a convolutional layer is:

$$\text{Output size} = \frac{\text{Input size} - \text{Kernel size}}{\text{Stride}} + 1$$

$$\text{Output height} = \text{Output width} = \frac{32-5}{1} + 1 = 28$$

Also, the number of filters determines the depth of the output. Here, we have 10 filters, so the output dimensions are:

$$28 \times 28 \times 10$$

b) As each filter has a size of $5 \times 5 \times 3$ (since the input has 3 channels), the number of parameters for each filter is:

$$\text{Parameters per filter} = 5 \times 5 \times 3 = 75$$

Also for each of the 10 filters. there is an additional bias term, so:

$$\text{Total parameters} = (75 + 1) \times 10 = 760$$

c) To calculate the receptive field at layer ℓ, we use the formula of the previous question:
Layer 1:

$$R_1 = 1 + (5-1) = 5$$

Layer 2:

$$R_2 = R_1 + (5-1)\cdot 1 = 5 + 4 = 9$$

Layer 3:
$$R_3 = R_2 + (5-1) \cdot 1 = 9 + 4 = 13$$
Layer 4:
$$R_4 = R_3 + (5-1) \cdot 1 = 13 + 4 = 17$$

□

Exercise 10.3

Proof. a) The problem with very deep networks is the degradation problem, where increasing the network depth beyond a certain point leads to a rapid decrease in accuracy during training, even though there exists a theoretically optimal solution. This degradation is not due to overfitting but rather stems from the difficulty in optimizing these deeper networks. Although deeper networks should not inherently result in higher training error (since a solution can theoretically be constructed by copying the parameters of a shallower model and using identity mappings for additional layers), current optimization methods often fail to find such solutions effectively or within a reasonable time.
b) Introducing skip connections, as in residual networks, helps address the optimization challenges in deep neural networks by reformulating the learning problem. Instead of requiring layers to learn the desired mapping $H(x)$ directly, they learn a residual mapping $F(x) = H(x) - x$, which is then added back to the input x via shortcut (identity) connections. This approach simplifies optimization by making it easier to approximate the residual function and ensures that the network can effectively learn even with very deep architectures, as identity mappings are easier to approach if optimal. □

Exercise 10.5

Proof. a) Sharp minima in the loss landscape are problematic because they often correspond to model parameters that are highly sensitive to small changes, leading to poor generalization on unseen data. This sensitivity can result in overfitting and degraded performance on test datasets.
b) The main idea of SAM is to improve generalization by simultaneously minimizing the training loss and the sharpness of the loss landscape. It focuses on finding parameters in regions with consistently low loss rather than merely at single points. This is accomplished through a min-max optimization problem that optimizes model parameters to be robust against worst-case perturbations within a specified neighborhood.
c) In the case of L_2-regularization, the perturbation $\hat{\epsilon}(w)$ will become:
$$\hat{\epsilon}(w) = \rho \frac{\nabla_w L_S(w)}{\|\nabla_w L_S(w)\|_2}$$
Here, ρ is a hyperparameter representing the neighborhood size, $\nabla_w L_S(w)$ is the gradient of the training loss with respect to w, and $\|\nabla_w L_S(w)\|_2$ is the L_2-norm of this gradient. This formula ensures that $\hat{\epsilon}(w)$ is a perturbation in the direction of the gradient, scaled to an L_2-norm of ρ.

$\square$