Foundations of Deep Learning **Fall 2024**

Exercise 9: Neural Network Architectures

Lecturer: Aurelien Lucchi

Problem 1 (Convolutions as Dense Operations):

The goal of this exercise is to better understand the term "parameter-sharing" in the context of convolutional neural networks.

a) Consider the 1-d convolution of an input $\mathbf{x} \in \mathbb{R}^D$ with a kernel $\mathbf{h} \in \mathbb{R}^K$:

$$(\mathbf{x} * \mathbf{h})_k = \sum_{i=1}^{K} x_{K+k-i} h_i \quad \text{for } k \in \{1, \dots, D-K+1\}$$

For fixed $\mathbf{h}$ we can thus view $\mathbf{x} * \mathbf{h}$ as a 1-d convolutional layer in a neural network

$$f_{\mathbf{h}} : \mathbb{R}^D \to \mathbb{R}^{D-K+1}, \, \mathbf{x} \mapsto \mathbf{x} * \mathbf{h}$$

Rewrite this operation as a dense layer, namely find a matrix $\mathbf{W_h} \in \mathbb{R}^{(D-K+1)\times D}$ such that you can write

$$f(\mathbf{x}) = \mathbf{W_h}\mathbf{x} \;\; \forall \mathbf{x}$$

Hint: Use Toeplitz matrix.

b) Compare the layer $f_{\mathbf{h}} : \mathbb{R}^D \to \mathbb{R}^{D-K+1}$, $\mathbf{x} \mapsto \mathbf{W_h}\mathbf{x}$ with a general dense layer $f : \mathbb{R}^D \to \mathbb{R}^{D-K+1}$, $\mathbf{x} \mapsto \mathbf{W}\mathbf{x}$ for some matrix $\mathbf{W} \in \mathbb{R}^{(D-K+1)\times D}$.

What are the number of free parameters in f and in $f_{\mathbf{h}}$? Why does $f_{\mathbf{h}}$ have less free parameters? Think about "parameter-sharing".

Problem 2 (Batch Normalization):
Assume we have a fully connected neural network f with input $\mathbf{x} \in \mathbb{R}^d$ described by the recursive system

- $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^d$
- $\mathbf{h}^{(l+1)}(\mathbf{x}) = \mathbf{W}^{(l+1)}\tilde{\mathbf{h}}^{(l)}(\mathbf{x}) \in \mathbb{R}^{d_{l+1}}$ where $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d_{l+1} \times d_l}$
- $\tilde{\mathbf{h}}^{(l+1)}(\mathbf{x}) = \phi\left(\mathbf{h}^{(l+1)}(\mathbf{x})\right) \in \mathbb{R}^{d_{l+1}}$ where $\phi : \mathbb{R} \to \mathbb{R}$ is a non-linearity applied component-wise
- $f(\mathbf{x}) = \left(\mathbf{W}^{(L+1)}\right)^T \tilde{\mathbf{h}}^{(L)}(\mathbf{x}) \in \mathbb{R}$ where $\mathbf{W}^{(L)} \in \mathbb{R}^{d_L}$

We can describe the batch norm operation at layer $l+1$ as follows. Fix a dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ and take a mini batch $\mathcal{B} = \{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ which we assume for simplicity is just the first B examples.
During the forward-pass for one step of SGD we calculate the pre-activations $\mathbf{h}^{(l+1)}(\mathbf{x}_1), \ldots, \mathbf{h}^{(l+1)}(\mathbf{x}_B)$. Given those values, we calculate mean and variance statistics as

- $\boldsymbol{\mu}^{(l+1)}(\mathcal{B}) = \frac{1}{B}\sum_{i=1}^{B} \mathbf{h}^{(l+1)}(\mathbf{x}_i) \in \mathbb{R}^{d_{l+1}}$
- $\boldsymbol{\sigma}^{(l+1)}(\mathcal{B}) \in \mathbb{R}^{d_{l+1}}$ such that $\sigma_j^{(l+1)} = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}\left(h_j^{(l+1)}(\mathbf{x}_i) - \mu_j^{(l+1)}\right)^2}$

Instead of propagating $h^{(l+1)}(\mathbf{x}_i)$ through the non-linearity ϕ for $i = 1, \ldots, B$, we pass

$$\mathbf{z}_j^{(l+1)}(\mathbf{x}) = \gamma_j^{(l+1)} \frac{h_j^{(l+1)}(\mathbf{x}) - \mu_j^{(l+1)}}{\sigma_j^{(l)}} + \beta_j^{(l+1)} \quad \text{for } j = 1, \ldots, d_{l+1}$$

where $\boldsymbol{\gamma}^{(l+1)}, \boldsymbol{\theta}^{(l+1)} \in \mathbb{R}^{d_{l+1}}$ are learnable parameters. The equations for a layer thus become:

- $\mathbf{h}^{(l+1)}(\mathbf{x}) = \mathbf{W}^{(l+1)}\tilde{\mathbf{h}}^{(l)}(\mathbf{x}) \in \mathbb{R}^{d_{l+1}}$
- $\mathbf{z}^{(l+1)}(\mathbf{x}) = \boldsymbol{\gamma}^{(l+1)} \odot \frac{\mathbf{h}^{(l+1)}(\mathbf{x}) - \boldsymbol{\mu}^{(l+1)}(\mathcal{B})}{\boldsymbol{\sigma}^{(l+1)}(\mathcal{B})} + \boldsymbol{\theta}^{(l+1)} \in \mathbb{R}^{d_{l+1}}$
- $\tilde{\mathbf{h}}^{(l+1)}(\mathbf{x}) = \phi\left(\mathbf{z}^{(l+1)}(\mathbf{x})\right) \in \mathbb{R}^{d_{l+1}}$

It is very important to realize that $\boldsymbol{\mu}^{(l)}(\mathcal{B})$ and $\boldsymbol{\sigma}^{(l)}(\mathcal{B})$ are a function of the batch $\mathcal{B}$ and thus vary in different forward passes, while $\boldsymbol{\gamma}^{(l)}$ and $\boldsymbol{\theta}^{(l)}$ do not.

In this exercise we want to understand the role of the rescaling operation through $\boldsymbol{\gamma}^{(l)}$ better.

a) Where lies the difference between

- Standardizing the layer and rescaling it with learnable parameters (batch normalization).
- Not modifying the layer at all.

You can argue very informally.

b) Why do we even rescale again with $\boldsymbol{\gamma}^{(l)}$? To that end, consider a very simple neural network

$$f(x) = w^{(2)}\phi(w^{(1)}x)$$

where $x \in \mathbb{R}$, $w^{(1)} \in \mathbb{R}$ and $w^{(2)} \in \mathbb{R}$. What happens to the representation power of f if we apply batch norm on the first layer without applying the rescale operation?