NNDL Exercise set 4 – Mathematical exercises

1.

a)

The leftmost image is the input, a 28×28 greyscale image, here represented by the digits 1 and 7. The vision transformer then divides the image into a grid of 16 image patches of size 7×7. Each patch is flattened and linearly projected into a vector, and these vectors are treated as input tokens for a standard transformer architecture. The layers in the transformer consists of multi-head self-attention blocks, which compute the attention weights between all token pairs. For input X and weight matrices W_Q, W_K and W_V, the attention is computed according to the formula

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where $Q = XW_Q$ are the queries, $K = XW_K$ are the keys and $V = XW_V$ are the values. The scaling factor d_k is the dimension of the queries/keys.

Each heatmap demonstrates likely the attention weights of a single head in a transformer layer, higher intensity demonstrating a higher attention a patch gives to another patch. From the heatmaps we can conclude that the model has focused somewhat well on the actual digit instead of the background noise.

b)

In this case the input dimensions differ, so this would need to be accounted for in the patching layer. Otherwise the architecture remains unchanged. I presume the model would not work as well, since we lose spatial information by switching to 1D patches instead of 2D. The MNIST digits contain structure both horizontally and vertically, which would be harder to capture in only 1D.

2.

I chose an article titled Seizure Prediction Based on Transformer Using Scalp Electroencephalogram by Jianzhuo Yan, Jinnan Li, Hongxia Xu, Yongchuan Yu and Tianyu Xu (2022).

In the article the authors develop a transformer-based model to predict epileptic seizures based on scalp electroencephalogram (EEG) signals. More specifically they try to predict the preictal state, which is the phase leading up to a seizure, and distinguish preictal from interictal EEG segments.

EEG data is sequantial, but it is treated multidimensionally as a 3D tensor to capture spatial, temporal and spectral features. Most important preprocessing steps:

- Short-time Fourier transform (STFT) is applied to convert 5 second EEG segments into time-frequency domain representations
- Frequencies around 60 Hz and its harmonics are removed to filter out power line noise
- Overlapping sliding windows are used to increase the number of preictal EEG segments, since the dataset was unbalanced
- The EEG segments are labeled as either preictal or interictal

The architecture is a three-tower architecture, where each tower processes a different (spatial, temporal, spectral) perspective of the EEG tensor. The input sequences are reshaped into 2D sequences along each axis and passed through separate transformer encoders. Standard multi-head self-attention is used. In addition, a gating mechanism is used to weigh and combine the outputs from the three towers. Final classification is done via a fully connected layer. The number of layers or parameters is not specified.

The training is done via supervised learning, where the EEG segments are classified as preictal or interictal. The data contains 21 patients and around 642000 EEG spectrograms. The split was 90 % train, 10 % test. In addition, the authors used a k-of-n (with $k = 24$ and $n = 30$) voting method as a post-processing step in order to reduce false positives.

The model worked well, with average sensitivity 96.01 %, average specificity 96.23 %, average precision 95.86 % and false prediction rate (FPR) of 0.047/h after postprocessing. Compared to other studies using Gated Transformer Networks (GTNs), CNNs or LSTMs the transformer model outperformed previous models in sensitivity and FPR, and provided less false positives and more accurate classification.

In summary, the model is good, even state-of-the-art in seizure prediction. The paper was clear and explained relevant details (preprocessing, architecture, evaluation) well. However, some implementation details were left out, such as the number of layers/parameters or hyperparameters. Bonus points for possibly impactful application domain.

3.

We have $(x, y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$\boldsymbol{\mu} = (0, 0), \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix} \tag{2}$$

for some constant c. We are fitting a linear estimator $\hat{y} = \beta_0 + \beta_1 x$ by least squares, so we want to minimize the sum of squared residuals $r_i = y_i - \hat{y}_i$

$$S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2. \tag{3}$$

The minimum is found where the gradient is zero, so we proceed to calculate the partial derivates.

$$\begin{aligned}
&\frac{\partial S}{\partial \beta_0} = 0 \\
&\implies \sum_{i=1}^{n} -2(y_i - \beta_1 x_i - \beta_0) = 0 \\
&\implies \sum_{i=1}^{n} \beta_0 = \sum_{i=1}^{n} (y_i - \beta_1 x_i) \\
&\implies \beta_0 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_1 x_i) = \bar{y} - \beta_1 \bar{x},
\end{aligned} \tag{4}$$

where $\bar{x}$ and $\bar{y}$ denote the sample mean. Using Eq. (4) we find

$$\begin{aligned}
&\frac{\partial S}{\partial \beta_1} = 0 \\
&\implies \sum_{i=1}^{n} (y_i - \beta_1 x_i - \bar{y} + \beta_1 \bar{x})^2 = 0 \\
&\implies \sum_{i=1}^{n} -2(y_i - \beta_1 x_i - \bar{y} + \beta_1 \bar{x})(\bar{x} - x_i) = 0 \\
&\implies \sum_{i=1}^{n} \beta_1 (\bar{x}^2 + x_i^2 - 2 x_i \bar{x}) = \sum_{i=1}^{n} (y_i x_i - \bar{y} x_i - y_i \bar{x} + \bar{y}\bar{x}) \\
&\implies \beta_1 \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\
&\implies \beta_1 = \frac{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2} = \frac{\mathrm{Cov}(x, y)}{\sigma_x^2}.
\end{aligned} \tag{5}$$

From the covariance matrix $\mathbf{\Sigma}$ we can read off that $\mathrm{Cov}(x, y) = c$ and $\sigma_x^2 = 1$, thus

$$\beta_1 = c, \tag{6}$$

i.e. the linear estimator learns the parameter c. It is the slope β_1 of the fit. This can be interpreted as self-supervised learning because we have constructed an auxiliary learning task from the input data only and used standard supervised learning methods to train the model (minimizing an objective function given data and labels).