Deep Learning (IST, 2021-22)

Homework 2

André Martins, Rita Ramos, João Santinha, José Moreira, Ben Peters, Francisco Melo

Deadline: Monday, January 31, 2022.

> Please turn in the answers to the questions below in a PDF file, together with the
> code you implemented to solve them (when applicable).
> Please submit **a single zip file** in Fenix under your group's name.

Question 1

1. (10 points) Consider a convolutional neural network (CNN) that takes as input images of
   size 28x28x3 and has the following layers:

   - One convolutional layer with 8 kernels of shape 5x5x3 with stride of 1, and a padding
     of zero (i.e., no padding).
   - A max-pooling layer with kernel size 4x4 and stride of 2 (both horizontally and verti-
     cally).
   - A linear output layer with 10 outputs followed by a softmax transformation.

   How many parameters are there in total in this network?

2. (10 points) Suppose that we replace the convolutional and max-pooling layers above by a
   feedforward, fully connected layer with hidden size 100 and output size 10 (followed by the
   softmax transformation). How many parameters are there in total? Compare to the previous
   answer and comment.

3. Let $\boldsymbol{X} \in \mathbb{R}^{L \times n}$ be an input matrix for a sequence of length L, where n is the embedding
   size. We are going to look at the *self-attention* for this sequence. Assume two self-attention
   heads have projection matrices $\boldsymbol{W}_Q^{(h)}, \boldsymbol{W}_K^{(h)}, \boldsymbol{W}_V^{(h)} \in \mathbb{R}^{n \times d}$ for $h \in \{1, 2\}$, with $d \leq n$.

   1. (5 points) Show that the self-attention probabilities for each head can be written as the
      rows of a matrix of the form

      $$\boldsymbol{P}^{(h)} = \text{Softmax}(\boldsymbol{X} \boldsymbol{A}^{(h)} \boldsymbol{X}^\top) \tag{1}$$

      where $\boldsymbol{A}^{(h)} \in \mathbb{R}^{n \times n}$ has rank $\leq d$ (in the above expression, Softmax is applied row-wise).
      Provide an expression for the matrix $\boldsymbol{A}^{(h)}$.

   2. (5 points) Show that, if $\boldsymbol{W}_Q^{(2)} = \boldsymbol{W}_Q^{(1)} \boldsymbol{B}$ and $\boldsymbol{W}_K^{(2)} = \boldsymbol{W}_K^{(1)} \boldsymbol{B}^{-\top}$, where $\boldsymbol{B} \in \mathbb{R}^{d \times d}$ is
      any invertible matrix, then the self-attention probabilities are exactly the same for the
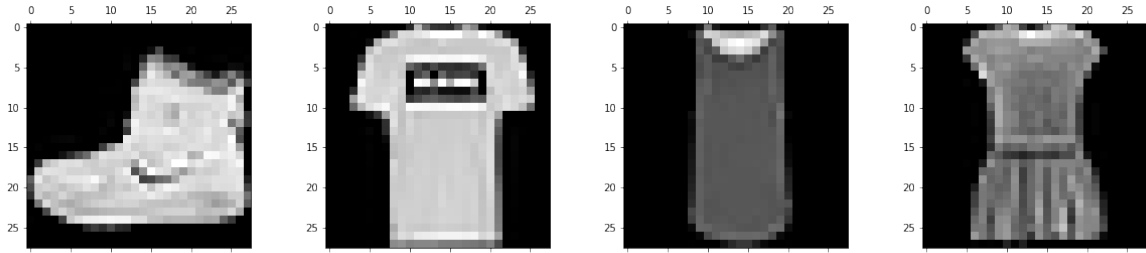   two attention heads.

Figure 1: Examples of images from the FashionMNIST dataset.

Question 2

Image classification with CNNs.   In this exercise, you will implement a convolutional neural network to perform classification using the Fashion-MNIST dataset. Examples of images in this dataset are shown in Figure 1.

As previously done in Homework 1, you will need to download the Fashion-MNIST dataset. You can do this by running the following command in the homework directory:

```
python download_fashion_mnist.py
```

Python skeleton code is provided (`hw2-q2.py`). You will now try out convolutional networks. For this exercise, we recommend you use a deep learning framework with automatic differentiation (suggested: Pytorch).

1. (5 points) What kind of invariances can you capture with convolutional layers?

2. (20 points) Implement a simple convolutional network with the following structure:

   - A first block comprising (order as provided):
     - A convolution layer with 16 output channels, a kernel of size 3x3, stride of 1, and padding chosen to preserve the original image size.
     - A rectified linear unit activation function.
     - A max pooling with kernel size 2x2 and stride of 2.
   - A second block comprising (order as provided):
     - A convolution layer with 32 output channels, a kernel of size 3x3, stride of 1, and padding of zero.
     - A rectified linear unit activation function.
     - A max pooling with kernel size 2x2 and stride of 2.
   - An affine transformation with 600 output features (to determine the number of input features use the number of channels, width and height of the output of the second block. Hint: The number of input features = *number of output channels × output width × output height*).
   - A rectified linear unit activation function.
   - A dropout layer with a dropout probability of your choice.
   - An affine transformation with 120 output features.
   - A rectified linear unit activation function.

- An affine transformation with the number of classes followed by an output LogSoftmax layer.

Hint: use the functions `nn.Sequential`, `nn.Conv2d` and `nn.MaxPool2d`.

Train your model for 15 epochs using `SGD` and tune the learning rate on your validation data, using the following values: 0.001,0.01,0.1. For the best configuration, report it and plot two things: the training loss and the validation accuracy, both as a function of the epoch number.

3. (10 points) Plot the filters in the first and second convolutional layers. In general, what kind of features are usually captured by CNNs in bottom vs top layers? (Note: for this exercise, the image resolution may be too low to notice interesting patterns in the filters.)

NOTE: the skeleton code is designed to output the kernels of the first and second convolutions that you can use to answer the next question. Use the options `-conv_1_name` and `-conv_2_name` to save the kernels after finishing the training of the CNN. If you are using `nn.Sequential` to define each block as:

```
self.convblock1 = nn.Sequential(...)
self.convblock2 = nn.Sequential(...)
```

use `-convlayer1 convblock1[0] -convlayer2 convblock2[0]`.

If not using `nn.Sequential` and the layers are defined as:

```
self.convlayer1 = nn.Conv2(...)
...
self.convlayer2 = nn.Conv2(...)
```

use `-convlayer1 convlayer1 -convlayer2 convlayer2`.

Question 3

Image Captioning. Image captioning is the problem of automatically describing an image with a short text description. This task is usually addressed with an encoder-decoder model. Typically, the encoder is a CNN model that processes and summarises the input image into relevant feature representations. The representations are then passed to a language decoder, commonly an LSTM model, which generates the respective caption word-by-word given the previous words and the image encoder representations.

1. You are going to implement an image captioning model for this task concerning aerial images, thus actually addressing remote sensing image captioning. The input and output should respectively be an image (i.e., remote sensing image) and the corresponding text description. The evaluation metric is BLEU-4 (which calculates precision between n-gram occurrences in the generated and reference captions.).

   (a) (10 points) Unzip the provided `images.zip` file and put the images at the `data/raw_dataset/images` folder. Run the file `script_features.py` (`python3 src/script_features.py`) that extract the images features using a pretrained `ResNet-18` convolutional neural network encoder model. Then, implement a vanilla image captioning model using an encoder-decoder architecture with an auto-regressive LSTM as the decoder. Specifically, in the skeleton code, you will need to implement the method `forward()` of the `decoder.py` (inside the folder `models`) and then run (`python3 src/train.py`).

   Plot the training loss and the validation BLEU-4. Also, report the final BLEU-4 score in the test set.

Hint: At each time-step, the LSTM decoder `self.decode_step` receives as input the embedding of the current word (`self.embedding`), then the LSTM hidden state is processed through a dropout operation (`self.dropout`), followed by an affine transformation (`self.fc`) to produce the scores of the words of the vocabulary.

(b) (20 points) Add an attention mechanism to the decoder (additive attention), which weights the contribution of the different image regions, according to relevance for the current prediction.

Specifically, in the skeleton code, you will need to implement the `forward()` methods of the `decoder_with_attention.py`, i.e., the forward methods of the `DecoderWithAttention` class and the `Attention class`.

Plot the training loss and the validation BLEU-4. Also, report the final BLEU-4 score in the test set.

Hint: Now, the input of the LSTM decoder (`self.decode_step`) is the embedding of the current word (`self.embedding`) concatenated with the attention vector.

You can run the code with the command `python src/train.py -use_attention`

(c) (5 points) For the previous model (encoder-decoder with neural attention), report the generated captions for the images "219.tif", "540.tif", "357.tif". For each of those images, place the image next to the corresponding caption. Hint: Use the file inside the `results` folder: `caps_generated_enc_dec_w_attention.json`.