Homework Assignment II
Group 42

91605: Fátima Agostinho Napoleão
92447: David Azevedo Escobar de Lima
92479: Ielga Katiana Anacleto de Oliveira

Wednesday, February 2nd, 2022

1

1.1

Consider a convolutional neural network (CNN) that takes as input images of size $28 \times 28 \times 3$ and has the following layers:

- **One convolutional layer with 8 kernels of shape $5 \times 5 \times 3$ with stride of 1, and a padding of zero (i.e., no padding).**
- **A max-pooling layer with kernel size 4×4 and stride of 2 (both horizontally and vertically).**
- **A linear output layer with 10 outputs followed by a softmax transformation.**

How many parameters are there in total in this network?

- Image of size = $28 \times 28 \times 3$
- Convolutional layer:
  - Each Kernel is a matrix of shape 5 x 5
  - Each kernel has three channels
  - There are 8 kernels/filters
- The total amount of parameters for the convolutinal layer is :
  - Number of parameters = number of filters x ((kernel width x kernel height x number of channels) + bias) = 8 x ((5 x 5 x 3) + 1))=608
- Output Layer:
  - As the output is fully connected layer, there are 3 filters

  - Number of parameters = numbers of filters x ((input width x input height x number of channels) + bias) = 10 x ((28 x 28 x 3) + 1) = 23530

- The pooling layer has a fixed operation, so there are no learnable parameters.
- The network will have a total of 608 + 23530 = 24138 parameters

1.2

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.

- Input image of size : $28 \times 28 \times 3$
- Fully Connected layer with hidden size 100 and output size 10
- A linear output layer with 10 outputs
- Fully connected layer parameters:
  - Number of parameters = number of units x ((input with x input height x number of channels) + bias) = 100 x ((28 x 28 x 3) + 1) = 235 300
- As saw in the previous question there are 23530 parameters for the output layer
- Summing all trainable parameters results for each layer, we get the total number of 235300 + 23530 = 258830 parameters within the entire network
- In comparison with others layers a fully connected layer has the highest number of parameters, because all inputs units have a separate weigh to each output unit that's why we have a higher number in this network compared to what we had before.

1.3

Let $X \in \mathbb{R}^{L \times n}$ 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 $W_Q^{(h)}$, $W_K^{(h)}$, $W_V^{(h)} \in \mathbb{R}^{n \times d}$ for $h \in \{1, 2\}$, with $d \leq n$.

1.3.1

Show that the self-attention probabilities for each head can be written as the rows of a matrix of the form

$$P^{(h)} = \textbf{Softmax}(XA^{(h)}X^{\top}) \tag{1}$$

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

- Given the input $X \in \mathbb{R}^{L\times n}$ for a sequence L, and the self projections matrices $W_Q^{(h)}$,$W_K^{(h)}$,$W_V^{(h)} \in \mathbb{R}^{n\times d}$, $h \in 1, 2$ and $d \leqslant n$
- We create the three distance representations:
  - Query Vectors : $Q^{(h)} = XW_Q^{(h)}$
  - Key Vectors : $K^{(h)} = XW_K^{(h)}$
  - Value Vectors : $V^{(h)} = XW_V^{(h)}$
- $P^{(h)} = \textbf{Softmax}(Q^{(h)}(K^{(h)})^T)$
- $P^{(h)} = \textbf{Softmax}(XW_Q^{(h)}(XW_K^{(h)})^T) = \textbf{Softmax}\ (XW_Q^{(h)}W_K^{(h)T}X^T)$
- The A^h is given by:
  - $A^h = W_Q^{(h)}W_K^{(h)T}$

1.3.2

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

- $h \in 1, 2$
- Self-attetion probabilities for first head : h = 1
  - $P^{(1)} = \textbf{Softmax}\ (XA^{(1)}X^T)$
  - $A^{(1)} = W_Q^{(1)}W_K^{(1)T}$
- Self-attetion probabilities for the second head : h = 2
- $P^{(2)} = \textbf{Softmax}(XA^{(2)}X^T) = \textbf{Softmax}(XW_Q^{(2)}W_K^{(2)T}X^T)$
- $W_Q^{(2)} = W_Q^{(1)}B$ and $W_K^{(2)} = W_K^{(1)}B^{-T}$
- $P^{(2)} =$**Softmax**$(XW_Q^{(1)}B(W_K^{(1)}B^{-T})^TX^T =$ **Softmax**$(XW_Q^{(1)}B(B^{-T})^TW_K^{(1)T}X^T)$
- Invertible matrix theorems: $B^{-T} = (B^{-1})^T$ and $[(B^{-1})^T]^T = B^{-1}$

- $P^{(2)} = \textbf{Softmax}(XW_Q^{(1)}BB^{-1}W_K^{(1)T}X^T)$
- B is invertible $\implies \mathrm{BB}^{-1} = I$
- $P^{(2)} = \textbf{Softmax}(\ \mathrm{XW}_Q^{(1)}IW_K^{(1)T}X^T)$
  $= \textbf{Softmax}(XW_Q^{(1)}W_K^{(1)T}X^T) = P^{(1)}$

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.

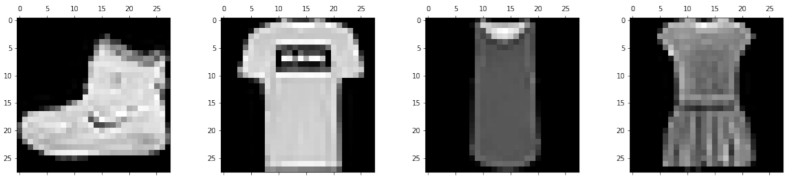


Figure 1: Examples of images from the Fashion-MNIST dataset.

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`).

2.1

What kind of equivariances do convolutional layers exhibit and why are they useful in CNNs for image classification?

- Locality: because features are translational invariant, small kernels can be used to convolve over the input.
- Spatial: this allows for detecting features/objects independently of their position in the input.

Because of these two properties, CNNs easily exploit images' data structures. They are capable of learning them independently of their dimensions and use relatively few parameters.

2.2

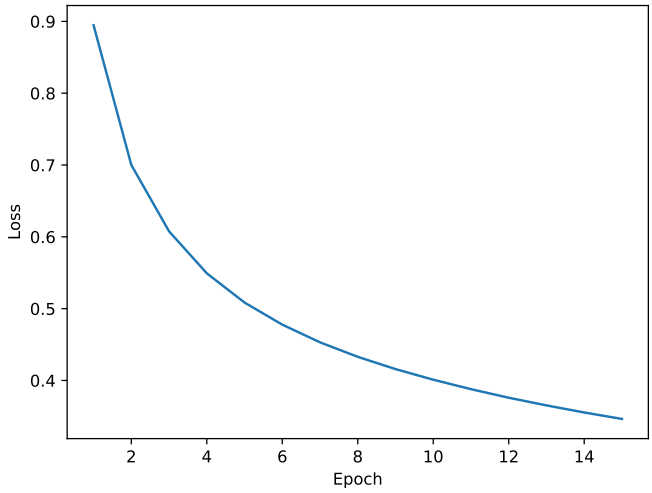
Implement a simple convolutional network with the following structure:

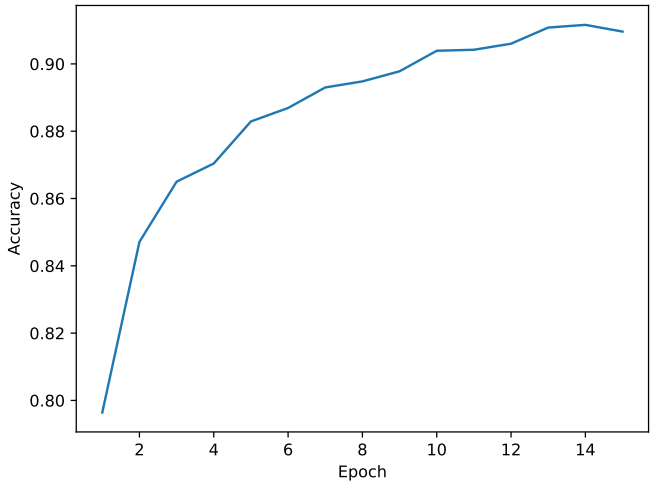
- **A first block comprising (order as provided):**
  - **A convolution layer with** 16 **output channels, a kernel of size** 3×3**, stride of** 1**, and padding chosen to preserve the original image size.**
  - **A rectified linear unit activation function.**
  - **A max pooling with kernel size** 2×2 **and stride of** 2**.**
- **A second block comprising (order as provided):**
  - **A convolution layer with** 32 **output channels, a kernel of size** 3×3**, stride of** 1**, and padding of zero.**
  - **A rectified linear unit activation function.**
  - **A max pooling with kernel size** 2×2 **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.**

Best configuration: learning rate = 0.01.





2.3

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(···)
```

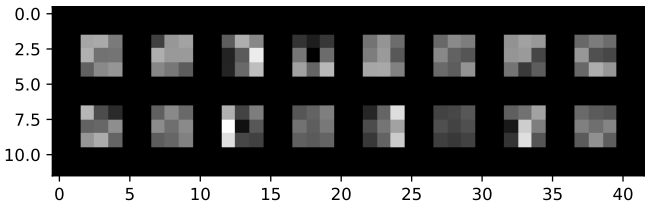
use `-convlayer1 convblock1[0] -convlayer2 convblock2[0].`

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

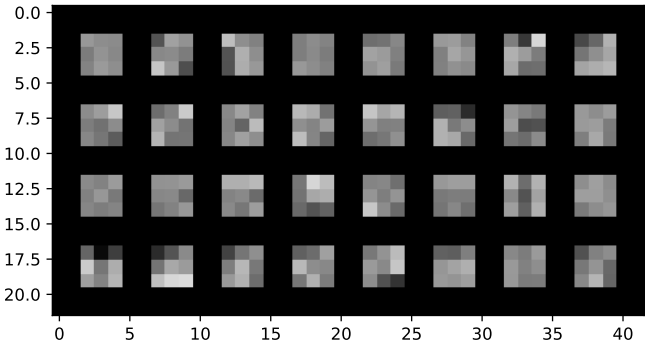
```
self.convlayer1 = nn.Conv2d(···)
···
self.convlayer2 = nn.Conv2d(···)
```

use `-convlayer1 convlayer1 -convlayer2 convlayer2.`

Layer 1:



Layer 2:



In general, lower layers capture lower-level representations, such as edges, corners and simple shapes, while higher layers capture higher-level representations, such as specific objects.

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.

3.1

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)

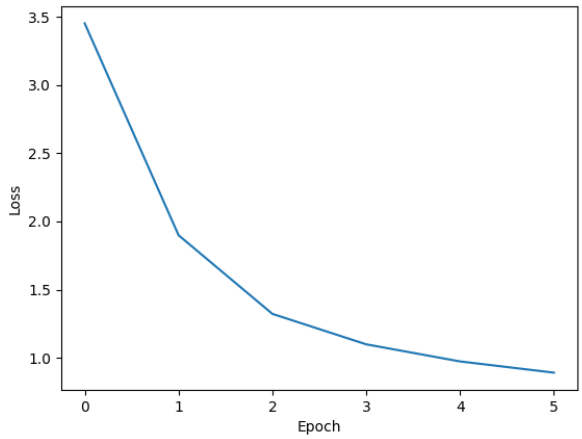
Unzip the provided `images.zip` file and put the images at the `data/raw_dataset/images` folder.

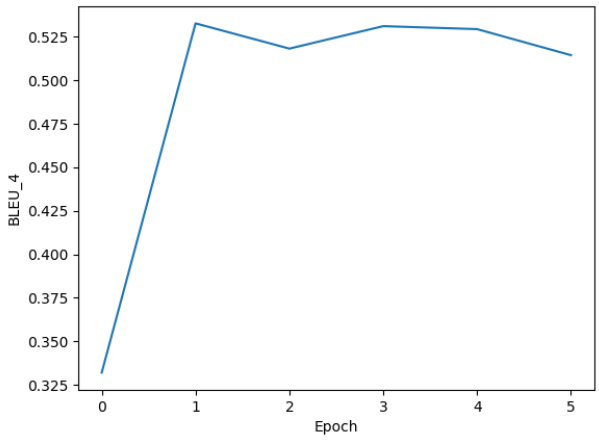
Run the file `script_features.py` (`python src/script_features.py`), that extracts the images' features using a pretrained `ResNet-18` convolutional neural network encoder model.

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.





Final BLEU-4 score on test set: 0.52706.

(b)

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.

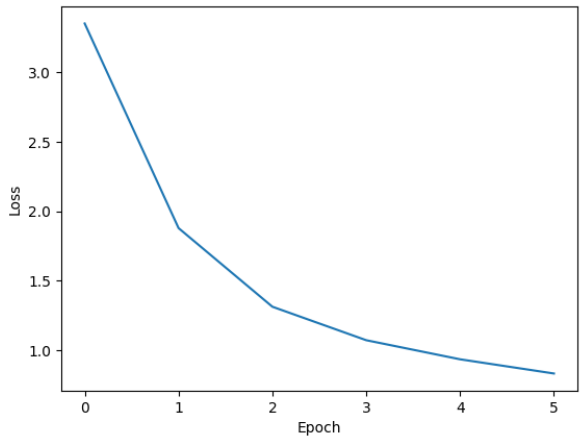
Plot the training loss and the validation BLEU-4. Also, report the

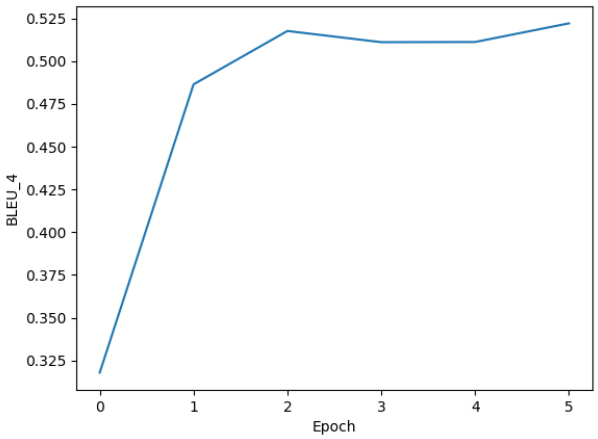
final BLEU-4 score in the test set.

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
```





Final BLEU-4 score on test set: 0.52922.

(c)

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`.

219: "a residential area with many houses arranged neatly and some roads go through this area"

540: "an industrial area with many white buildings and some roads go through this area"

357: "a residential area with many houses arranged neatly and some roads go through this area"