Advanced Human Language Technologies
Exercises on Convolutional Neural Networks

Convolutional Neural Networks

Exercise 1.

Convolutional Neural Networks (CNNs) are commonly used in NLP for tasks such as text classification. Consider a 1D CNN applied to a sequence of word embeddings.

1. Explain the role of convolutional filters in a 1D CNN for NLP.
2. How does the receptive field of a CNN change when stacking multiple convolutional layers?
3. Compare the use of CNNs and RNNs for text classification. What are the advantages and disadvantages of each?

Exercise 2.

Consider a 1D CNN applied to a text sequence. Suppose we have the sentence:

The cat sat on the mat

We encode each word as a 3-dimensional vector and arrange them into a matrix:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$

A 1D convolutional filter of size 3×3 slides over this matrix.

1. If we use $stride = 1$ and no padding, how many output values will be generated?
2. If we use $stride = 2$ instead, how does the output size change?
3. How does adding padding affect the output size?

Exercise 3.

Given an input sentence represented as a matrix of word embeddings:

$$X = \begin{bmatrix} 0.2 & 0.5 & -0.3 \\ -0.1 & 0.7 & 0.4 \\ 0.8 & -0.6 & 0.1 \\ 0.3 & 0.2 & -0.5 \end{bmatrix}$$

where each row represents a word embedding of dimension 3. Consider a convolutional filter with weights:

$$W = \begin{bmatrix} 0.5 & -0.2 & 0.1 \\ -0.3 & 0.8 & -0.6 \end{bmatrix}$$

1. Compute the output of the convolution operation when applying this filter to the input matrix with stride=1
2. What happens if we use a stride of 2 instead of 1?

Exercise 4.

Consider a max-pooling operation applied to the following feature map:

$$H = \begin{bmatrix} 1.2 & -0.3 & 0.5 & 0.7 \\ -0.8 & 2.0 & 0.1 & -1.4 \\ 0.6 & 0.9 & -0.2 & 1.1 \end{bmatrix}$$

Using a pooling window of size 2×2 with stride 2, compute the output of the max-pooling operation.

Exercise 5.

Consider the following NLP pipeline using a 1-D Convolutional Neural Network (CNN):

An input sentence of length $L = 10$ words is represented as a sequence of word embeddings, each of dimension $d = 50$. The input is therefore a matrix of size 10×50.

Two different convolutional filters are applied:

- Filter F_1 has a kernel size of $k_1 = 3$ and applies $c_1 = 4$ feature maps.
- Filter F_2 has a kernel size of $k_2 = 5$ and applies $c_2 = 6$ feature maps.

Both filters slide with a stride $= 1$ and padding $= 0$.

A max-pooling operation is then applied to each feature map, using a pooling size of 2 and a stride of 2.

1. Compute the output dimensions (height $\times$ width) of the feature maps produced by each convolutional filter.
2. Compute the dimensions of the feature maps after the max-pooling operation.
3. Draw a schema of the CNN, detailing the dimensions of each layer.
4. How would the dimensions change if "same" padding* were used?

* "same" padding consists in paddind the sequence with as many slots are needed so that the output of the convolution has the same length than the input

Exercise 6.

A 1-D Convolutional Neural Network (CNN) processes an input sentence represented as a matrix of shape 12×100, where each row corresponds to a word embedding of dimension 100. The input is processed by two **stacked** convolutional layers and a final max pooling layer:

- First Convolutional Layer: 3 filters, each with a kernel size of 4, stride=1, "same" padding.
- Second Convolutional Layer: 5 filters, each with a kernel size of 6, stride=2, padding=0.
- Max-Pooling Layer: size=2, stride=2

1. Compute the output dimensions after each convolutional layer.
2. Compute the output dimensions after the max-pooling layer.
3. Draw a schema of the CNN, detailing the dimensions of each layer.
4. How would the output dimensions change if the last max pool layer used max pool over time instead?

Exercise 7.

In NLP tasks, CNNs extract **local patterns** from word embeddings. Consider a **sentiment analysis** task where a CNN is used to detect positive or negative phrases in a sentence.

Given these phrases:

I absolutely love this movie!
This was a terrible experience.

1. How can CNN filters be designed to detect *positive* and *negative* sentiment patterns?
2. How does *max pooling* help select the most important features?
3. Why might CNNs struggle with very long-range dependencies compared to RNNs?

Exercise 8.

For each NLP task below, decide whether a CNN or an RNN (or both) would be more suitable, and justify your choice:

1. Text classification (e.g., spam detection, sentiment analysis)
2. Machine translation
3. Named entity recognition (NER)
4. Text similarity (e.g., plagiarism detection)
5. Part-of-speech (POS) tagging

-

Exercise 9.

CNNs are useful in NLP since they capture word *n-grams*. Suppose we have a filter size of 3 sliding over a sentence represented as word embeddings.

1. What patterns might a size-3 filter detect in a sentence?
2. If we use multiple filter sizes (e.g., 2, 3, and 4), how does this improve feature extraction?
3. How does max pooling affect the final representation?

Exercise 10.

Given the feature map output from a convolutional layer:

$$F = \begin{bmatrix} 1.2 & 0.5 & 2.3 & 1.8 \\ 0.7 & 1.5 & 2.1 & 0.6 \\ 1.0 & 0.8 & 1.7 & 2.0 \end{bmatrix}$$

Apply max pooling with a pool size of 2×2 and *stride=2*.

1. What does the output look like?
2. Why is max pooling useful in NLP?
3. What information might be lost when applying pooling?

Exercise 11.

CNNs can be used to compare two sentences in tasks like question-answering or paraphrase detection.

1. How can we use two CNNs with shared weights to compare sentence embeddings?
2. Why is a 1D convolution over words effective for sentence matching?
3. How does cosine similarity help in matching two CNN-encoded sentence representations?