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?

SOLUTION

1. Convolutional filters detect local patterns such as n-grams in word embeddings. These filters slide over the text and capture features useful for downstream tasks.
2. The receptive field grows with each additional convolutional layer. For example, a single-layer filter of size 3 covers three words, but stacking two layers increases the receptive field to five words (approximate calculation assuming stride 1).
3. CNNs capture local patterns efficiently and allow parallel computation, making them faster than RNNs. However, RNNs can model sequential dependencies better, which is useful for tasks like language modeling.

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?

SOLUTION

1. With $stride = 1$ and no padding, the 3×3 window will slide from row 1 to row 4, thus generating 4 values.
2. With $stride = 2$, the window will be applied at row 1 and 3, i.e. the output size will be 2.
3. Adding padding allows the window to slide beyond the limits of the input matrix, so the output will be longer than without padding.

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?

SOLUTION

1. The convolution operation is computed as follows:

   - Rows 1-2:

     $$H_1 = X_1 \cdot W_1 + X_2 \cdot W_2$$
     $$= (0.2 \times 0.5) + (0.5 \times -0.2) + (-0.3 \times 0.1) + (-0.1 \times -0.3) + (0.7 \times 0.8) + (0.4 \times -0.6) = 0.32$$

   - Rows 2-3:

     $$H_2 = X_2 \cdot W_1 + X_3 \cdot W_2$$
     $$= (-0.1 \times 0.5) + (0.7 \times -0.2) + (0.4 \times 0.1) + (0.8 \times -0.3) + (-0.6 \times 0.8) + (0.1 \times -0.6) = -0.93$$

   - Rows 3-4:

     $$H_3 = X_3 \cdot W_1 + X_4 \cdot W_2$$
     $$= (0.8 \times 0.5) + (-0.6 \times -0.2) + (0.1 \times 0.1) + (0.3 \times -0.3) + (0.2 \times 0.8) + (-0.5 \times -0.6) = 0.9$$

     Thus, the output feature map with a stride of 1 is: $H = \begin{bmatrix} 0.32 \\ -0.93 \\ 0.9 \end{bmatrix}$

2. When using a stride of 2, the filter moves two rows at a time, applying the filter to rows 1-2, and 3-4

   Thus, the output with a stride of 2 is: $H = \begin{bmatrix} 0.32 \\ 0.9 \end{bmatrix}$

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 1, compute the output of the max-pooling operation.

SOLUTION

We apply a max-pooling operation with a 2×2 pooling window and a stride of 1. The feature map is divided into six 2×2 blocks as follows:

- Block 1, starts at position (1,1): $H_{11} = \begin{bmatrix} 1.2 & -0.3 \\ -0.8 & 2.0 \end{bmatrix}$. The maximum value in this block is 2.0.
- Block 2, starts at position (1,2): $H_{12} = \begin{bmatrix} -0.3 & 0.5 \\ 2.0 & 0.1 \end{bmatrix}$. The maximum value in this block is 2.0.
- Block 3, starts at position (1,3): $H_{13} = \begin{bmatrix} 0.5 & 0.7 \\ 0.1 & -1.4 \end{bmatrix}$. The maximum value in this block is 0.7.
- Block 4, starts at position (2,1): $H_{21} = \begin{bmatrix} -0.8 & 2.0 \\ 0.6 & 0.9 \end{bmatrix}$. The maximum value in this block is 2.0.
- Block 5, starts at position (2,2): $H_{22} = \begin{bmatrix} 2.0 & 0.1 \\ 0.9 & -0.2 \end{bmatrix}$. The maximum value in this block is 2.0.
- Block 6, starts at position (2,3): $H_{23} = \begin{bmatrix} 0.1 & -1.4 \\ -0.2 & 1.1 \end{bmatrix}$. The maximum value in this block is 1.1.

Thus, the output feature map after max-pooling is: $\begin{bmatrix} 2.0 & 2.0 & 0.7 \\ 2.0 & 2.0 & 1.1 \end{bmatrix}$

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:

- F_1 applies $c_1 = 4$ filters with kernel size of $k_1 = 3$ each.
- F_2 applies $c_1 = 6$ filters with kernel size of $k_1 = 5$ each.

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

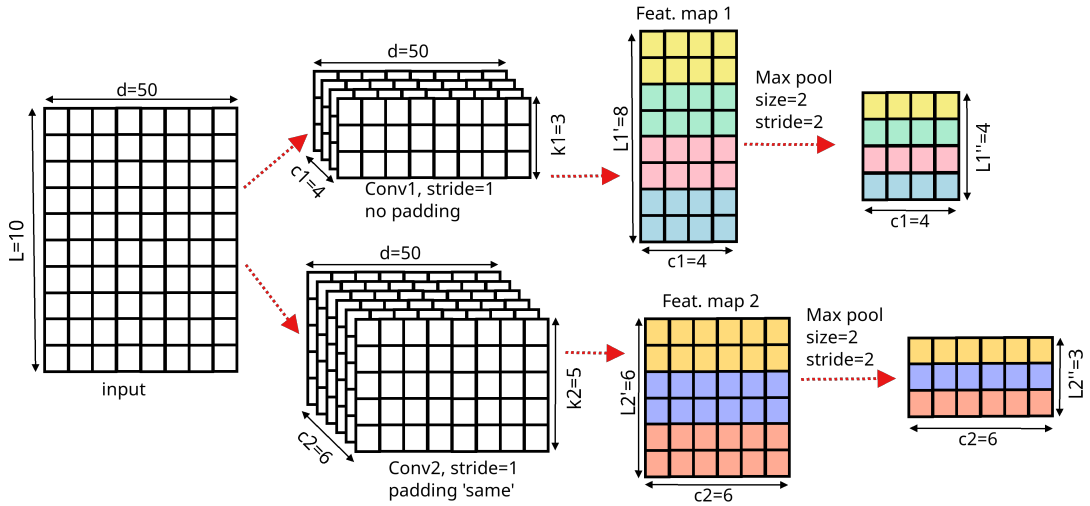
A max-pooling operation is then applied to each produced feature map, using a pooling size of 2 and a stride of 2.

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

SOLUTION

1. The output length L' of a 1-D convolution of size k with no padding is $L' = L - k + 1$. Thus,
   - For F_1 ($k_1 = 3$), the output length is $L'_1 = 8$. Since there are 4 feature maps, the output dimension is 8×4.
   - For F_2 ($k_2 = 5$), the output length is $L'_2 = 6$. Since there are 6 feature maps, the output dimension is 6×6.
2. The length for a max-pooling output with size p and stride s is: L" = $\frac{L'-p}{s} + 1$. Thus,
   - For F_1 ($L'_1 = 8$), the output size is $L''_1 = \frac{8-2}{2} + 1 = 4$. Since there are 4 feature maps, the output dimension is 4×4.
   - For F_2 ($L'_2 = 6$), the output size is $L''_2 = \frac{6-2}{2} + 1 = 3$. Since there are 6 feature maps, the output dimension is 3×6.
3. Architecture schema



4. With "same" padding the filter can slide once for each word in the input, so that output length remains the same as the input length, thus, $L'_1 = 10$ and $L'_2 = 10$.

   After max-pooling:
   - For F_1 the output size would be $L''_1 = \frac{10-2}{2} + 1 = 5$. Since there are 4 feature maps, the output dimension is 5×4.
   - For F_2 the output size would be $L''_2 = \frac{10-2}{2} + 1 = 5$. Since there are 6 feature maps, the output dimension is 5×6.

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?

SOLUTION

1. For a 1-D convolution: $L' = \frac{L+2p-k}{s} + 1$ where p is the padding.

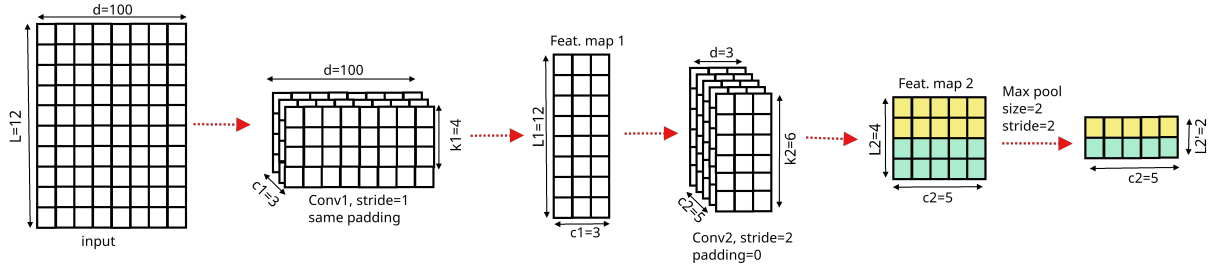
   Conv1 ("same" padding means output length remains L): $L_1 = 12$. Since there are 3 filters, output size is 12×3.

   Conv2 ($padding = 0$) is applied on the output of Conv1: $L_2 = \frac{L_1-6}{2} + 1 = \frac{12-6}{2} + 1 = 4$. Since there are 5 filters, output size is 4×5.

2. After max pooling $L' = \frac{L-p}{s} + 1$

   Pooling after Conv2: $L'_2 = \frac{4-2}{2} + 1 = 2$. Since there are 5 filters, output size is 2×5.

3. Architecture schema



4. If Conv 2 output were max-pooled over time, only the max value for each column in feature map 2 would be kept, obtaining a vector of size 1×5.

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?

SOLUTION

1. CNN filters can be designed to detect *positive* and *negative* sentiment patterns by learning convolutional kernels that capture key n-grams associated with sentiment. For example:
   - A filter detecting positive sentiment may capture phrases like *love this* or *absolutely love*.
   - A filter detecting negative sentiment may capture patterns like *terrible experience*.
2. *Max pooling* helps select the most important features by retaining the most activated values within a region, ensuring that the strongest presence of a feature (e.g., a sentiment-biased phrase) is preserved while reducing dimensionality.

3. CNNs struggle with very long-range dependencies because they rely on fixed-size filters, limiting their receptive field. RNNs, however, process sequences step-by-step, allowing them to capture dependencies over longer distances more effectively.

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

SOLUTION

1. **Text classification**: CNNs are often preferred as they effectively capture local patterns (e.g., sentiment phrases) and are computationally efficient.
2. **Machine translation**: RNNs (or Transformer-based models) are more suitable since they handle sequential dependencies and context across long distances.
3. **Named entity recognition**: RNNs are generally better because entities depend on surrounding context, but CNNs can be used in hybrid models.
4. **Text similarity**: CNNs can be effective for feature extraction, but similarity tasks often benefit from deeper sequence-aware models such as RNNs or Transformers.
5. **PoS tagging**: RNNs are preferable since POS tags depend on syntactic structure, but CNNs can be used for feature extraction in hybrid models.

-

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?

SOLUTION

1. A size-3 filter detects trigrams, capturing local semantic or syntactic patterns such as verb-object pairs (*love this movie*) or negations (*not very good*).
2. Using multiple filter sizes (e.g., 2, 3, and 4) improves feature extraction by capturing different granularity levels, enabling better phrase and pattern detection.
3. Max pooling ensures that the most prominent feature from each filter is retained, reducing the dimensionality while preserving the most significant patterns.

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?

SOLUTION

1. The output after max pooling is:
$$\begin{bmatrix} 1.5 & 2.3 \\ 1.0 & 2.0 \end{bmatrix}$$
2. Max pooling is useful in NLP as it reduces dimensionality while keeping the most salient features, improving computational efficiency.
3. Pooling may lose fine-grained positional information, which could impact tasks requiring precise word order.

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?

SOLUTION

1. Two CNNs with shared weights can be used to encode both sentences into vector representations. These representations can then be compared using a similarity metric (e.g., cosine similarity) to determine their relatedness.
2. A 1D convolution over words is effective for sentence matching because it captures key n-grams and semantic features efficiently while being less sensitive to word order than RNNs.
3. Cosine similarity measures the angle between two CNN-encoded sentence vectors, providing a measure of similarity regardless of absolute magnitude.