National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester 2, 2024/2025

Tutorial 10
Convolutional Neural Networks and Recurrent Neural Networks

Summary of Key Concepts

In this tutorial, we will discuss and explore the following learning points from Lecture:

1. CNN
   (a) Convolution kernel
   (b) CNN receptive field
   (c) Padding and stride
   (d) Design choice
2. RNN Design choice
3. CNN vs RNN

A ConvNets

1. You are given an image $\mathbf{x} \in \mathbb{R}^{5\times5}$ and a filter $\mathbf{W} \in \mathbb{R}^{3\times3}$. **No** extra padding is added.

0.1	0.2	0.1	0.1	0
0.8	0.9	1	1	0.9
1	1	1	1	1
0.9	1	1	0.8	1
0	0.1	0.1	0.2	0

1	1	1
0	0	0
-1	-1	-1

Figure 1: (left) The image $\mathbf{x}$. (right) The filter $\mathbf{W}$.

Get the output feature map from this convolution and write down its output dimensions if the kernel moved with a stride of 1×1. No Padding or Pooling op-

eration required. Additionally, what do you think is the purpose of this filter?

Solution:

Output dimension: 3×3
Output feature map:
$$\begin{bmatrix} -2.6 & -2.6 & -2.8 \\ -0.2 & 0.1 & 0.1 \\ 2.8 & 2.6 & 2.7 \end{bmatrix}$$
The filter performs horizontal edge detection by computing the differences between neighboring pixels in the vertical direction.

2. Now, let's say you have access to a very deep, large CNN model. We feed a single image to the network. Each image has $3(C)$ channels (RGB) with a height and width of 224×224 ($H = 224, W = 224$). Here our input is a 3-dimensional tensor ($H \times W \times C$). The first layer of the big CNN is a convolutional Layer with 96 (C_1) kernels which has a height and width of 11, and each kernel has the same number of channels as the input channel. The stride is 4×4 and no padding is used. Calculate the output size $H_1 \times W_1 \times C_1$ after the first convolutional Layer.

Solution:

For a single image and a single kernel, the output height $H_1 = \left\lfloor \frac{H-K+2P}{S} \right\rfloor + 1 = 54$. Similarly output width is $W_1 = 54$. Hence the output size is $H_1 \times W_1 \times C_1$. Here, H_1 and W_1 are the dimensions of the intermediate feature map, K is the kernel size, P is the padding, and S is the stride.

3. In most of Deep Learning libraries such as PyTorch, images are fed together as a batch B. B can take values such as $8, 16, 32, 64$. Comment on the output shape if we feed the large CNN in part (b) with a batch of images. What are the advantages of using a batch of images rather than a single image?

Solution:

The output shape will be $B \times H_1 \times W_1 \times C_1$. Using a batch of images is computationally efficient due to modern hardware (especially GPUs) being able to parallelize. It is also more stable in gradient descent convergence.

B CNN Receptive field

The receptive field refers to the specific region of the input image that a particular neuron in the network is influenced by. Essentially, it's the area of the input that contributes to the output of a neuron in a convolutional layer. It gives us an idea of where we're getting our results from as data flows through the layers of the network. To further illuminate the concept let's have a look at this illustration

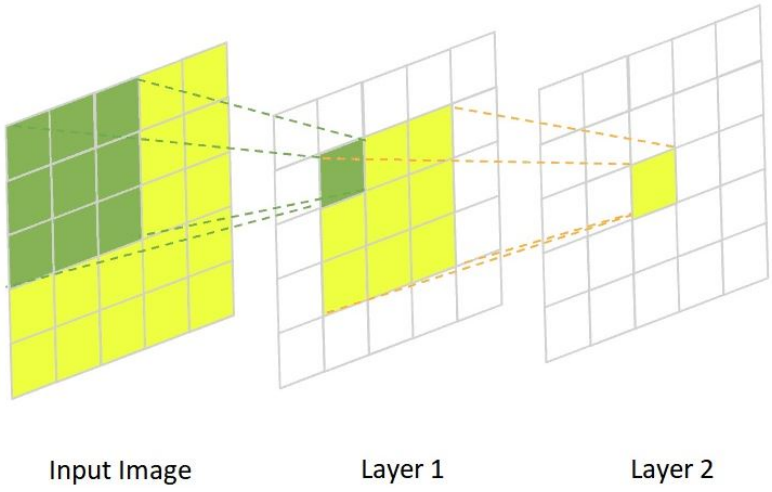


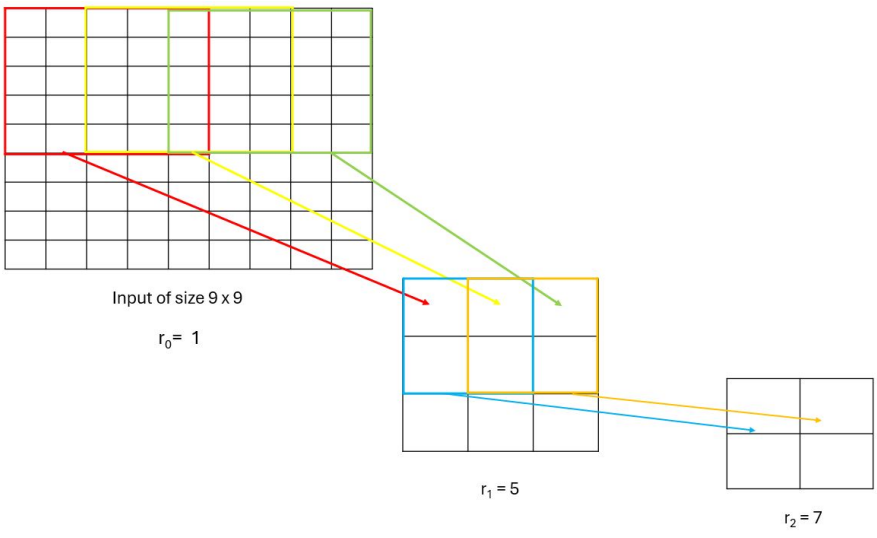
Figure 2: Two-layered CNN with a 3x3 kernel

The green area in the input image represents the receptive field (3x3) of one neuron in Layer 1, as the output of this neuron is influenced by a 3x3 pixel region in the input image. The yellow area in the input image marks the receptive field of one neuron in Layer 2, where a 5x5 pixel area from the input image is considered.

1. Consider a 2-layer convolutional neural network (CNN) with the following configuration:
   - An initial input matrix with size 9×9
   - Layer 1: Uses a kernel of size 5×5 and a stride of 2×2.
   - Layer 2: Uses a kernel of size 2×2 and a stride of 1×1.

   Calculate the receptive field sizes of neurons in the first and second layers.

Solution:



The output of the neurons in the first layer is influenced by the 5×5 pixels in the input image. The neurons in the second layer is then influenced by the 2×2 neurons in Layer 1. To produce these 2×2 neurons in Layer 1, a 7×7 pixel area from the input layer is considered.

Alternatively, to calculate the receptive field for each layer in the CNN, we can use the following formula:

$$r_i = r_{i-1} + (kernel_i - 1) \times j_{i-1} \tag{1}$$

$$j_i = j_{i-1} \times stride_i \tag{2}$$

For the input image, we use these values: $r_0 = 1$, $j_0 = 1$ Using the formula above we can calculate r_1 :

$$r_1 = 1 + ((5 - 1) \times 1) = 5 \tag{3}$$

$$j_1 = 1 \times 2 = 2 \tag{4}$$

With the value of r_1 and j_1 we can calculate r_2

$$r_2 = 5 + ((2 - 1) \times 2) = 7$$

Thus, our final answers are:

- Receptive Field of first layer: 5×5
- Receptive Field of second layer: 7×7

2. How does an increase in the receptive field affect the performance of a Convolutional Neural Network (CNN)?

Solution:

An increase in the receptive field of a Convolutional Neural Network (CNN) allows each neuron in the deeper layers to "see" a larger portion of the input image. This enables the network to capture more global features and spatial context, which can be beneficial for understanding high-level structures in the image, such as objects and their relationships.

In practical terms, a larger receptive field helps the CNN to better recognize patterns that span across larger areas of the image, such as long edges, textures, or multiple objects. This can improve the network's performance in tasks that require contextual information, such as object detection, scene segmentation, and image classification.

However, larger receptive fields usually involve more layers and can lead to increased computational requirements.

C RNN Design

1. What type of RNN model does Image captioning use? What characteristics make this a standard model for Image captioning? Can you think of any other examples for this type of RNN?

   Solution:

   One-to-many model. One-to-many sequence problems are sequence problems where the input data has one time-step, and the output contains a vector of multiple values or multiple time-steps. Thus, we have a single input and a sequence of outputs. Input: One image. Output: Multiple words as captions.

   Other examples:

   1. Music generation (e.g. Input: Non-sequential parameters like genre, key, style, composer etc., Output: A sequence of notes)
   2. Text generation (e.g. Input: Non-sequential parameters like genre, format, author etc., Output: A sequence of words)

2. What type of RNN model does stock market prediction use? What characteristics make this a good model for stock prediction? Can you think of any other examples for this type of RNN?

   Solution:

   Many-to-one model. In many-to-one sequence problems, we have a sequence of data as input, and we have to predict a single output. Stock price prediction is one such use case. Input: Time series data of stock prices, Output: Likelihood of price increase.

   Other examples:

   1. Text sentiment analysis (e.g. Input: Many words. Output: Which class this text belongs to)
   2. Health monitoring (e.g. Input: Time series data of blood pressure, cholesterol, etc., Output: Likelihood of a disease)

3. What type of RNN model does language translation use? What characteristics make this a good model for language translation? Can you think of any other examples for this type of RNN?

   Solution:

   Many-to-many model. Many-to-Many sequence learning can be used for machine translation where the input sequence is in some language, and the output sequence is in some other language. Input: Many words / code of

language A. Output: Many words / code of language B.

Other examples:

1. Phonetic transcription (e.g. Input: An audio signal, Output: A sequence of phones i.e. speech sounds, expressed in IPA (International Phonetic Alphabet))
2. Handwriting recognition (e.g. Input: A sequence of pen stroke data, Output: A sequence of characters)

D CNN vs RNN

1. Let's apply what you learnt to a specific example, performing sentiment analysis on Covid-19 posts on twitter. Explain what characteristics of RNN make it a good model for sentiment analysis and which RNN model you want to use to tackle this problem.

   Solution:

   The Recurrent neural network method is the standard method for dealing with any sequential (e.g., time series) input. As the final state of the RNN encodes the representation of the entire sentence, we use this final state to yield a classification of the sentiment of the sentence. This corresponds to a many-to-one RNN model.

2. Given the same context in part (1), would it be possible to perform sentiment analysis using CNN? Explain why or why not.

   Solution:

   The key thing about sentiment analysis is that we do not just need to capture what words appear, but we also need to capture the general context of the whole sentence. The CNN model is also capable of doing sentence classification because as we add more convolution layers and make the network deeper, we will be able to detect higher-level features and capture the general context of the whole sentence.

3. In part (2), we've looked at ConvNets for sequences (sentiment analysis of text, for instance). Now, let's examine RNNs for image processing.

   "Tokens" are what constitute a sequence; words are tokens in sentences, for instance. Usually, when it comes to words, we represent them as vectors. We also know RNNs take in sequences (temporally-related collection of tokens). If we had to treat an image as a sequence, what would you do to allow RNNs to classify images? In other words, how can you tokenize an image? Your final method of "tokenizing" an image should ensure an RNN can be fed these tokens as inputs.

Express your creativity in coming up with different ways to go about doing so.

Solution:

Possible ways are to treat each row/column as a token, or use overlapping / non-overlapping patches, or treat windows of 10 pixels as a token. Among these, the best (as in, practical) way is to use patches as they retain locality, allowing important features (like ears or eyes) to be kept together.