National University of Singapore
School of Computing
CS3244: Machine Learning
Solution to Week - 10 - Tutorial 8

Deep Learning

Colab Notebook Solutions: Deep Learning

1. **CNN** You are given an image $\mathbf{x} \in \mathbb{R}^{4\times4}$ and a kernel $\mathbf{W} \in \mathbb{R}^{3\times3}$. **No** extra padding is added.

0.5	0.2	0.1	0.7
0.1	0.6	0.9	0.5
0.0	0.8	0.2	0.7
0.2	0.4	0.0	0.4

0.1	0.2	0.6
0.4	0.3	0.5
0.9	0.8	0.7

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

.

(a) Get the output feature map from this convolution and write down its output dimensions if the kernel moved with a stride of $\{1 \times 1\}$ and we used ReLU. No Pooling operation required.

Output dimension: 2×2

Output feature map: $\begin{bmatrix} 1.60 & 2.59 \\ 1.51 & 1.91 \end{bmatrix}$

(b) 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 of has a height of 11 and a width of 11 and each kernel has the same number of channels as the input channel(i.e. kernel size is $\{11 \times 11\}$). Stride is $\{4 \times 4\}$ and no padding is used. Calculate the output size after the first Convolutional Layer.

For single image and a single kernel, the output height $h_1 = \left\lfloor \frac{h - k + p}{s} \right\rfloor + 1 = 54$ (Refer slide 31 of 10.a for the equation). Similarly output height is $w_1 = 54$. Hence the output size is $h_1 \times w_1 \times c_1$, where h_1, w_1 are the dimensions of the intermediate feature map, k is the kernel size, p is the padding, and s is the stride.

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

The output shape will be $b \times h_1 \times w_1 \times c_1$. Using a batch of images is computationally efficient and more stable in gradient descent convergence.

(d) In a CNN, why is it good design choice to stack **TWO** Convolutional layers, each with a $\{3 \times 3\}$ kernel, one after the other on an image instead of a **SINGLE** Convolutional Layer with a $\{5 \times 5\}$ kernel? The Convolutional layers have $\{1 \times 1\}$ stride. (**Hint:** Think about parameter count and power of representations)

$\{5 \times 5\}$ kernel has more parameters than two layers with $\{3 \times 3\}$ kernels each (25 vs. 18). In addition, the collective receptive field (area of sight, just like our own eyes) of a single $\{5 \times 5\}$ kernel is the same as that of two back-to-back $\{3 \times 3\}$ kernels. This holds when the stride for both convolutions is $\{1 \times 1\}$.

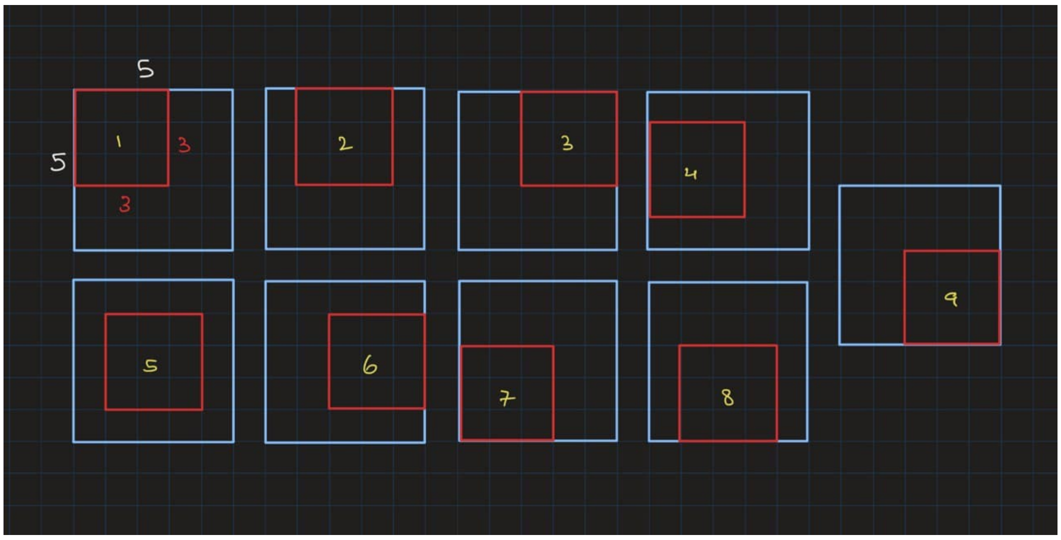

Figure 2: Compare the receptive fields of two $\{3 \times 3\}$ Convolutional layers and a single $\{5 \times 5\}$ Convolutional layer.

2. **Quick Discussions**. Describe what deep learning models/techniques will you use to tackle the following problems.

- You need a model to determine the sentiment of sentences.

  RNN or CNN. A sentence has obvious sequential dependencies between words and hence a proper model would need to capture this interdependence. The recurrent neural network method is the standard method for dealing with any sequential (e.g., time series) input. RNNs natively deal with different length (sized) input (to think about, why?). 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. Also as the RNN tends to model recent past time steps well better than distant ones, several architectures improve on the plain RNN architecture; namely Long Short-term Memory (LSTM), Gated Recurrent Units (GRU), and their Bi-directional variants.

The CNN is also capable of doing sentence classification because the receptive fields of convolution layers are growing with the depth of the network. Deep layers are therefore able to capture the general context of the whole sentence. See Reading Material 1. CNN has a particular advantage in that the convolution operations can be done in parallel over a fixed-sized window.

- You need a model to do translation between two languages.
  RNN. This is a case of the many-to-many architecture, where the output comes strictly after encoding all of the input source sentence's token into a internal sentential representation (as evidenced by the final Θ_{hh} weights being passed after the final input token is observed. This sentence representation is sometimes referred to as a "thought vector". This encoded representation is then decoded to produce the target sentence in the output language. Nowadays, transformer models take the lead of area, and produce the SOTA(State of the art) result for many nlp problems. Can refer to material 3 for more information.

3. **RNN** Suppose we have a very simply RNN model such that

$$\begin{aligned} h_t &= f(w_1 x_t + w_2 h_{t-1} - 1) \\ y_t &= g(w_3 h_t) \end{aligned}$$

where f and g are activation functions satisfying

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$
$$g(x) = x$$

For this question, you can assume $h_0 = 0$.

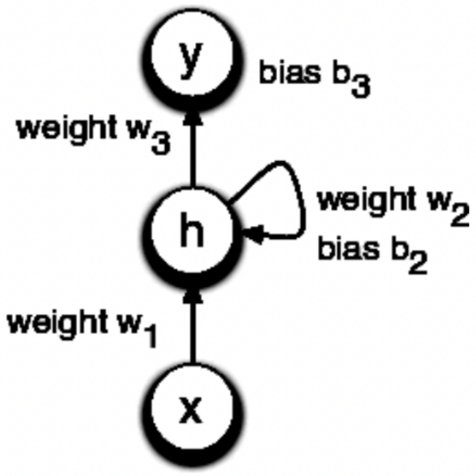

Figure 3: Illustration of the RNN unit.

(a) Find one possible combination of (w_1, w_2, w_3) so that the input sequence $x_{1\to4} = [0, 1, 0, 1]$ produces the output $[0, 1, 1, 1]$.
There are many possible answers. For example, $(w_1, w_2, w_3) = (2, 2, 1)$ is a possible answer.

(b) Find the output for the input sequence [1, 0, 0, 0, 1] using the weights found in part (a).
The output sequence is [1, 1, 1, 1, 1]

You can refer to the following reading materials to learn more:

1. Convolutional Neural Networks for Sentence Classification
2. Residual Networks Behave Like Ensembles of Relatively Shallow Networks
3. Attention Is All You Need