Computer Vision
Assignment №13

Pooya Kabiri

Department of Computer Science

Iran University of Science and Technology

December 2020

1 Theoretical Questions

- **A.**

  We use the following formula:

  $$W_2 = (W_1 - F + 2P)/S + 1$$

  $$H_2 = (H_1 - F + 2P)/S + 1$$

  $$D_2 = K$$

  Because we want to have the same output W and H, we put X for Padding value as an unknown variable:

  $$32 = ((32 - 9 + 2X)/1) + 1 \Rightarrow 9 - 1 = 2X \Rightarrow X = 4$$

  The H value is calculated same as W, So we need 4 Padding for both Width and Height.

  We have 16 filters ($K = 16$), so the whole filter shape is $9 * 9 * 16$, and there are 10 channels of the input (Number of filters in the previous layer), so there are $((9 * 9 * 10) + 1) * 16 = 12,976$ parameters for training (1 is added for bias).

- **B.**

  We use the following formula:

  $$W_2 = (W_1 - F + 2P)/S + 1$$

$$H_2 = (H_1 - F + 2P)/S + 1$$
$$D_2 = K$$

$S = 1$, $P = 0$, $W_1 = 32$, $F = 5$ and $H_1 = 32$, So:

$$W_2 = (32 - 5 + 0)/1 + 1 \Rightarrow W_2 = 28$$

H_2 is also same as W_2 which is equal to 28.
So the output shape is $28 * 28 * 3$.

$S = 1$, $P = 0$, $W_1 = 32$, $F = 3$ and $H_1 = 32$, So:

$$W_2 = (32 - 3 + 0)/1 + 1 \Rightarrow W_2 = 30$$

When applying the same layer:
$S = 1$, $P = 0$, $W_2 = 30$, $F = 3$ and $H_2 = 30$, So:

$$W_3 = (30 - 3 + 0)/1 + 1 \Rightarrow W_3 = 28$$

So the output shape is $28 * 28 * 3$.

- **C.**
  - **Learning Rate:**
    The learning rate is the most important hyper-parameter of an Artificial Neural Network.
    It is a hyper-parameter that controls how much to change the model in response to the estimated error each time the model weights are updated (the amount that the weights are updated during training, also referred to as the step size).
    Choosing a too large learning rate may result in getting stuck in a local optima in the loss function (learning a sub-optimal set of weights too fast) or an stochastic behavior of the loss function (unstable training).
    Choosing a value of learning rate which is too small results in a long and exhausting training procedure which is error prone and can get stuck (training fails).

  - **Batch Size:**
    The Batch Size determines the number of samples which propagate through the network (forward and backward) at every step of the training procedure.

Because the batch size is usually a fraction of the total training set size, using batches uses less memory because at every step less data should be loaded into the memory and not the whole data set. This is important in large scale machine learning.

Also using batches (also called mini-batches) are helpful in speeding up the training process, because the weights of the network are updated after each batch propagates forward in the network, and the error and loss and gradients of the batch are back propagated to optimize the loss function, so in compare to a full-size batch training, the network parameters are updated more regularly resulting in a more promising training process. When using a batch size equal to 1, at each pass only a simple example is propagated through the network. When the cardinality of the training data set is large, the process of calculating gradients for only one training example in each step has a huge computational overhead (Because Neural Networks usually use vectorized implementation of the gradient calculation).

With all being said, using a batch size equal to 32, 64 or 128 is a popular choice.

Depending on the batch size there are 3 types of Gradient Descent algorithm:

* **Batch Gradient Descent:** Batch Size = Size of Training Set
* **Stochastic Gradient Descent:** Batch Size = 1
* **Mini-Batch Gradient Descent:** 1 < Batch Size < Size of Training Set

– **Epoch:**

Number of Epochs determines the number of times that the learning (optimization) algorithm will run on the complete training data.

During one epoch, each and every sample of the training data has propagated through the network and had a chance to update the network parameters.

The value for number of epochs is usually large, which allows the learning algorithm to minimize the cost and error functions of the network as much as possible.

During each epoch, all the mini batches are propagated through the network one-by-one. We can think of epoch as a for loop repeating for a number of times (e.g. 2000), and mini-batches as a nested for loop inside the epoch for loop which iterates over every mini-batch and propagates the mini-batch through the network.

Source: Links provided y assignment document

- **D.**

When Fully-Connected layers are used in a neural network, each neuron of the FC layer receives input (information) from every element of the previous layer, so the neuron cannot have a specialized view but rather a general view on the image data, **Meanwhile,** In a convolutional layer each neuron receives input from a single restricted region of the previous layer, so each neuron can build a much more specialized view on the image data. In this case each neuron is trained to infer a special kind of information from the restricted region of the image which it looks at. For every neuron this restricted area is called the neuron's **receptive field**.

Because of the reasons above, when dealing with spatial data (like images), using convolutional layers are much more promising. That's why they are widely used in Computer Vision tasks such as Image Classification, Object Tracking and Detection, Semantic Segmentation and Instance Segmentation.

Source: Wikipedia

- **E.**

  Pooling layers are useful in reducing the dimension of the feature maps created by convolution layers. This results in the network having less parameters to learn and less amount of computation.

  Also Pooling layers summarizes the features in the feature map which are generated by convolution layers. This leads to the model being more robust to variation in the position of the feature. This layer summarizes the feature map in a way that the precise location of the features is not important anymore.

  Source: Geeks for Geeks

- **F.**

  First, consider a $30 * 30 * 1$ image.

  We calculate the output shape of this image after applying a convolution layer with filter size of 7, and also after 3 consecutive convolution layers of filter size 3 each. Then we compare the result. Assume that the number of filters are equal and equal to 1.

  When using convolution with filter size of 7, we have the following parameters: $S = 1$, $P = 0$, $W_1 = 30$, $F = 7$ and $H_1 = 32$, So:

  $$W_2 = (30 - 7 + 0)/1 + 1 \Rightarrow W_2 = 24$$

  H_2 would have the same value as W_2.

So the output size would be $24 * 24 * 1$.

When using 3 subsequent convolution with filter size of 3 we have the following values :$S = 1$, $P = 0$, $W_1 = 30$, $F = 3$ and $H_1 = 32$.

$$W_2 = (30 - 3 + 0)/1 + 1 \Rightarrow W_2 = 28$$

$$W_3 = (28 - 3 + 0)/1 + 1 \Rightarrow W_3 = 26$$

$$W_4 = (26 - 3 + 0)/1 + 1 \Rightarrow W_4 = 24$$

values of H are same as their corresponding W values.

The output size of the 3 subsequent layers is $24 * 24 * 1$, it is the same as a $7 * 7$ convolution layer.

Now let's consider the total number of multiplications needed for each approach, on an example input image of size $30 * 30 * 1$ containing only a single channel.

For simplicity consider number of filters to be 1 in both approaches ($K = 1$).

The total number of multiplications needed for a convolution layer is equal to the total number of output pixels ($W * H$) multiplied by total numbers of filter parameters ($F * F$).

For the first approach, we have the following number of multiplication (M_1):

$$M_1 = 24 * 24 * 7 * 7 = 28,224$$

Now for the second approach, we have the following number of multiplication each of the 3 layers:

$$M_1 = 28 * 28 * 3 * 3 = 7,056$$

$$M_2 = 26 * 26 * 3 * 3 = 6,084$$

$$M_3 = 24 * 24 * 3 * 3 = 5,184$$

So the total number of multiplications is:

$$M_1 + M_2 + M_3 = 18,324$$

So the second approach also uses much less multiplications in compare to the first approach, so it is computationally more efficient.

2 Time Tracked

The total time tracked for this assignment (both theory and programming) is **7 Hours and 40 Minutes.**