Student Name: Mohammad Ahmad Khattab Mousa

Student Number: 2002639

Course: CSE616: Neural Networks and Their Applications

Academic Year: Spring 2022

Affiliation: Ain Shams University

Solution of Assignment 2

1. Consider an input image of shape 500x500x3. The image is flattened and a fully connected layer with 100 hidden units is used. What is the shape of the weight matrix of this layer (without the bias)? What is the shape of the bias?

Layer	Output Shape
Input Image	500 x 500 x 3
Flatten	1 x 750,000 (500*500*3)
Fully Connected	1 X 100

Weight matrix has a shape of 750,000 x 100 leading to **75 million learnable parameters**.
Bias has a shape of 1 X 100 leading to 100 learnable parameters

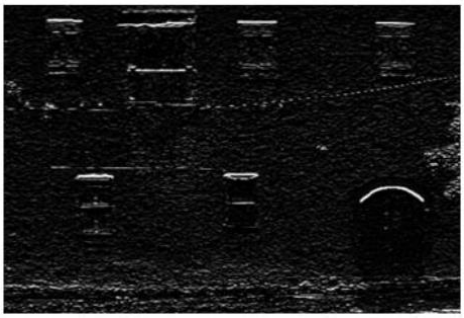
2. You run this image in a convolutional layer with 10 filters, of kernel size 5x5. How many arameters does this layer have?

Filter size = 5 X 5 X 3 ➔ each filter has 5*5*3 (convolution operation) +1 (bias) parameters
Each filter has 76 parameters
Total layer parameters = 76 * 10 = **760 learnable parameters**

3. The top gray image has run through different types of filters and the results are shown in the following images. What type of convolutional filter was used to get each of the resulting images. Explain briefly and include the values of these filters. The filters have a shape of (3,3).

The image was generated using a filter to detect vertical edges which is a High-Pass Filter applied to the horizontal direction. Example of a filter could be as below:

$$\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$



The image was generated using a filter to detect horizontal edges which is a High-Pass Filter applied to the vertical direction. Example of a filter could be as below:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

4. In the Adam optimizer. Show what will happen when the numbers of steps to compute the exponential moving averages gets large.

Intuitively, since beta factors are less than 1 ➔ after too many updates parameter updates, initial & very old values for the gradient will be modulated by a very small number leading to very small contribution to the adaptive learning rate and very recent gradient values will have higher contribution to the adaptive learning rate

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

Investigating the equations, when unbiased first & second tends to biased versions

Rewriting the first moment equation:

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$

$m_t = \beta_1(\beta_1 m_{t-2} + (1-\beta_1) g_{t-1}) + (1-\beta_1) g_t = {\beta_1}^2 m_{t-2} + \beta_1 (1-\beta_1) g_{t-1} + (1-\beta_1) g_t$

$m_t = {\beta_1}^2(\beta_1 m_{t-3} + (1-\beta_1) g_{t-2}) + \beta_1 (1-\beta_1) g_{t-1} + (1-\beta_1) g_t$

$m_t = {\beta_1}^3 m_{t-3} + {\beta_1}^2 (1-\beta_1) g_{t-2} + \beta_1 (1-\beta_1) g_{t-1} + (1-\beta_1) g_t$

$$m_t = {\beta_1}^t m_0 + (1-\beta_1) \sum_{i=1}^{t} {\beta_1}^{t-i} g_i$$

$$\hat{m}_t = \frac{{\beta_1}^t m_0 + (1-\beta_1) \sum_{i=1}^{t} {\beta_1}^{t-i} g_i}{1-{\beta_1}^t}$$

Similarly,

$$\hat{v}_t = \frac{{\beta_2}^t v_0 + (1-\beta_2)\sum_{i=1}^{t} {\beta_2}^{t-i} {g_i}^2}{1-{\beta_2}^t}$$

when $t \to \infty$, $the\ quantity\ (\beta^t)\ will\ be\ much\ smaller\ than\ 1\ \&\ hence\ \ (1-\beta^t) \to 1$
In addition, m_0 & v_0 assumed zero, hence

$$\lim_{t\to\infty} \hat{m}_t = \lim_{t\to\infty} m_t = (1-\beta_1)\sum_{i=t-n+1}^{t} {\beta_1}^{t-i} g_i$$

Similarly,

$$\lim_{t\to\infty} \hat{v}_t = \lim_{t\to\infty} v_t = (1-\beta_2)\sum_{i=t-n+1}^{t} {\beta_2}^{t-i} {g_i}^2$$

Where we could conclude the following when we have too many update iterations:

1. Bias correction is not needed after too many training iterations
2. Adaptive gradient will be dominated by recent gradients meanwhile old gradient values will have a neglectable impact

5. Given a batch of size m, and assume that a batch normalization layer takes an input $z=(z_{(1)},\ldots,z_{(m)})$ as an input. Write down the output equation(s) of this layer. Give two reasons for using the batch normalization layer.

$$\mu_z = \frac{1}{m}\sum_{i=1}^{m} z_i$$

$$\sigma_z^2 = \frac{1}{m}\sum_{i=1}^{m} (z_i - \mu)^2$$

$$y_i = \frac{z_i - \mu_z}{\sqrt{(\sigma_z^2 + \epsilon)}}\gamma + \beta$$

where γ , β are learnable parameters,

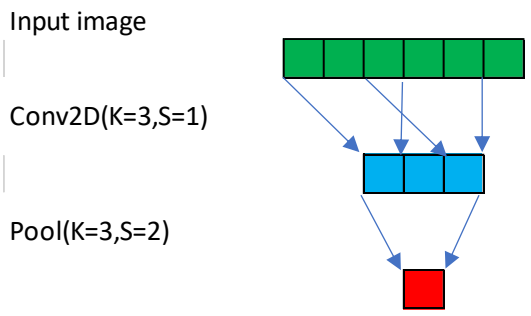
ϵ is a very small constant for numerical stability

Benefits of using batch normalization:

1. Improves gradient flow through the network
2. Allow higher learning rates
3. Reduce the strong dependency on initialization

6. Suppose you have a convolutional network with the following architecture: The input is an RGB image of size 256 x 256. The first layer is a convolution layer with 32 feature maps and filters of size 3x3. It uses a stride of 1, so it has the same width and height as the original image. The next layer is a pooling layer with a stride of 2 (so it reduces the size of each dimension by a factor of 2) and pooling groups of size 3 x 3.
Determine the size of the receptive field for a single unit in the pooling layer. (i.e., determine the size of the region of the input image which influences the activation of that unit.) You may assume the receptive field lies entirely within the image.

Layer	Input shape	Output Shape
RGB Input image		Input shape 256 x 256 x 3
Conv2D, K=3, S=1, in_channels=3, out_channels=32	256 x 256 x 3	254 x 254 x 32
Pooling, K=3, S=2	254 x 254 x 32	126 X 126 x 32



Receptive Field = 5 x 5

7. If an input data block in a convolutional network has dimension $C \times H \times W = 96 \times 128 \times 128$, (96 channels, spatial dim 128x128) and we apply a convolutional filter to it of dimension $D \times C \times HF \times WF = 128 \times 96 \times 7 \times 7$, (i.e. a block of D=128 filters) with stride 2 and pad 3,what is the dimension of the output data block?

Checking: A guide to convolution arithmetic for deep learning (https://arxiv.org/pdf/1603.07285.pdf)

HO=WO=$HO = WO = floor\left(\frac{W+2P-k}{s}\right) + 1 = floor\left(\frac{128+6-7}{2}\right) + 1 = 64$

Output Shape = 128 x 64 x 64

8. What is inverted dropout and what is its advantage?

Inverted dropout is an implementation of the dropout regularization algorithm which avoids any change in the network during the testing/inference phase. This is implemented by the dividing the activations of all unmasked activations by the dropout probability (P) only during training phase (which means multiplying unmasked activations by a factor greater than 1) & hence compensate the effect of masked activations during training time. During testing time, the network is not changed & all activations are allowed to flow without any manipulation.

Advantage: Simplicity of implementation as the dropout layer could be completely removed during testing time without any impact over the result.

9. Explain briefly why fully connected neural networks do not work well for image classification.

FC networks have the following assumptions which are not suitable for image classification:

1. FC networks **generally assumes non-sparse weight matrix**. Pixels of images depend greatly on each other and hence **weigh matrix should be sparse** or otherwise network will suffer from **overfitting due to model over parameterization.**
2. FC networks **assumes 1-D input layer**. Flattening of images will destroy 2D spatial information making learning process more difficult.
3. FC networks don't support the properties of translation equivariance (shifting the position of an object would apply the same spatial shift to the layer activation) and translation invariance (slightly shifting the position of an object shouldn't impact the classification without more training). Hence, image classification will require a very big dataset to learn all different translations.

10. Compute the convolution of the following two arrays: (4 1 -1 3) * (-2 1). Your answer should be an array of length 5. Show your detailed work.

We will need to apply padding to the input (P=1)

Input = [0 4 1 -1 3 0]

Output[0] = 0*-2 +4*1= 4

Output[1] = 4*-2+1*1= -7

Output[2] = 1*-2+(-1)*1= -3

Output[3] = -1*-2+3*1= 1

Output[4] = 3*-2+0*1= -6

Output = (4 -7 -3 1 -6)

11. Describe what setting change in epochs 25 and 60 could have produced this training curve. Be brief.

There was a drop in the learning rate leading to drop in the loss function and enhancement in accuracy. This could be done manually by changing the model learning rate or by using learning rate schedulers.

12. Why are convolutional layers more commonly used than fully-connected layers for image processing?

CNN solve the previously mentioned problems of FC networks in the following ways:

1. CNN enforces **sparse weigh matrix** by having convolution with small kernel size hence for each activation unit a small portion of the image (k*k pixels) is concerned & weights to other portions are considered zero. This greatly reduces the model parameters and avoid overfitting.
2. CNN makes use of the **2D spatial information** of the image without destroying this information.
3. CNN has **translation equivariance** property by **sharing the same convolution kernel parameters between all units within the same activation map**. Hence, shifting the position of an object will apply the same shift to the activation map without more training loops.
4. CNN has **translation invariance** property by using the **pooling layer with the correct receptive field**. This means that slightly shifting the position of one object will lead to a shifted activation map and then when passing through a pooling layer, the same output will be generated without any more training loops. Hence, CNN are more robust to object position change without more training loops.

13. Dropout layers implement different forward functions at train and test time. Explain what they do. Let p be the probability that node value is *retained.*

For Vanilla Dropout:

At Training time, the dropout layer randomly selects (1-p) percentage of the activations to be masked to zero (dropped out) and (p) percentage of the activations are allowed to pass through the network without manipulation (retained).

At testing time, all activations are allowed to pass through the network after multiplying these activations by (P) percentage to compensate the fact that the network weights were trained with the presence of only (P) percentage of these activations. In this way, network weights shouldn't be impacted by the presence of all activations.

For Inverted Dropout:

At Training time, the dropout layer randomly selects (1-p) percentage of the activations to be masked to zero (dropped out) and (p) percentage of the activations are allowed to pass through the network and the values of these unmasked activation is divided by (P) which means that the values of unmasked activations are increased to compensate the impact of masking some activations.

At testing time, all activations are allowed to pass through the network without any manipulation.

14. Explain how standard momentum and Nesterov accelerated gradient differ. How does their performance (convergence as a function of time) differ on convex optimization problems? Use sketches as an aid.

Standard Momentum: Parameter Update Rule:

1. Compute the gradient of the loss function $\nabla_\theta L(\theta_{t-1}, y_{t-1})$
2. Calculate new velocity vector using the below equation:

$v_t = \rho * v_{t-1} - \alpha * \nabla_\theta L(\theta_{t-1}, y_{t-1})$

$\boldsymbol{\alpha}: learning\ rate$, $\boldsymbol{\rho}: velocity\ decaying\ factor$

3. Calculate the new parameter value using the below equation

$\theta_t = \theta_{t-1} + v_t$

Nesterov Momentum: Parameter update Rule:

1. Calculate an approximation of the next parameter value

$\widehat{\boldsymbol{\theta}}_{\boldsymbol{t-1}} = \theta_{t-1} + \rho * v_{t-1}$

2. Compute the gradient of the loss function $\nabla_\theta L(\widehat{\boldsymbol{\theta}}_{\boldsymbol{t-1}}, y_{t-1})$ by using the approximation of the next parameter value instead of the current value
3. Calculate new velocity vector using the below equation:

$v_t = \rho * v_{t-1} - \alpha * \nabla_\theta L(\hat{\theta}_{t-1}, y_{t-1})$

4. Calculate the new parameter value using the below equation

$\theta_t = \theta_{t-1} + v_t$

Performance Enhancement: Based on http://proceedings.mlr.press/v28/sutskever13.pdf

For convex optimization problems, in case current point is close to the minimum and the momentum was large to push the update in the other direction away from the minimum, **standard momentum** will fail to damp this update as the current gradient ($-\nabla_\theta L$) will be in the same direction of the momentum. The network will need to wait for the next update to start to damp this velocity direction.

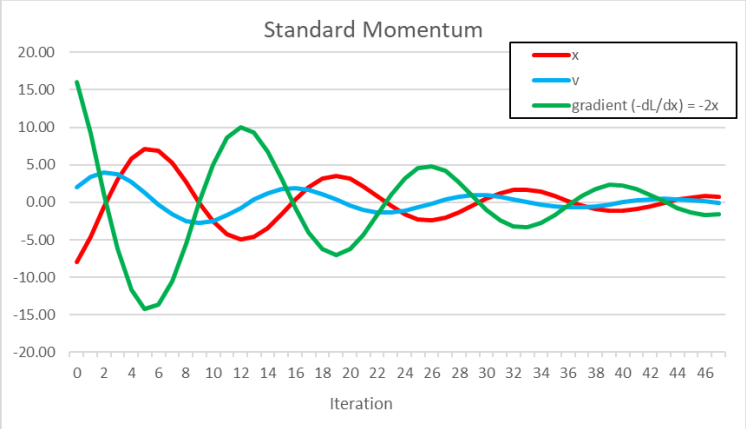
Nesterov momentum uses the look-ahead (approximated) value of the parameter to calculate the gradient, which would probably be in the other side of the curve and hence the gradient will be in the opposite direction of the velocity vector and **will help to damp the velocity vector without waiting for the next update iteration**.

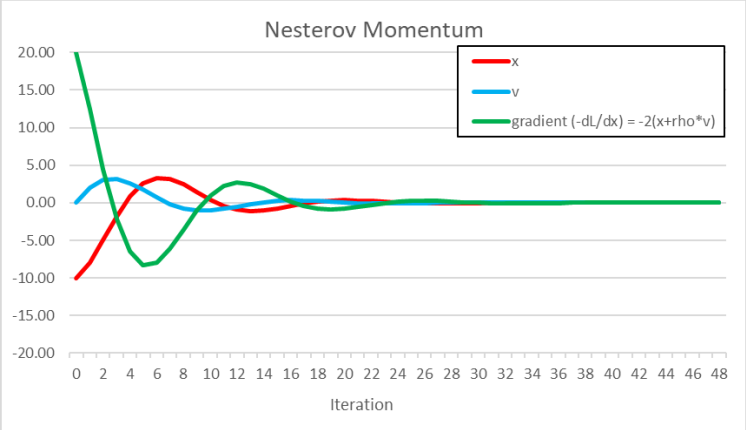
In conclusion, Nesterov momentum will help to have more stable optimization and damp oscillations around the minimum and hence will provide faster convergence for convex loss functions.
To illustrate the difference, I used the case of simple convex quadratic loss function:

$L = x^2$ (which has a minimum at x=0)

$Gradient = -\nabla_x L = -\frac{dL}{dx} = -2x$

$\boldsymbol{\rho = 0.9,\ \alpha = 0.1,\ v_0 = 0, x_0 = -10}$ were chosen to plot the below graphs





15. How does the actual learning rate of ADAGRAD change with the number of steps?

Actual learning rate will be decaying as the RMS of the previous gradients will keep growing with the learning iterations and hence learning updates will be dying with the increase of iteration steps even if there's enough gradient back propagation through the network.