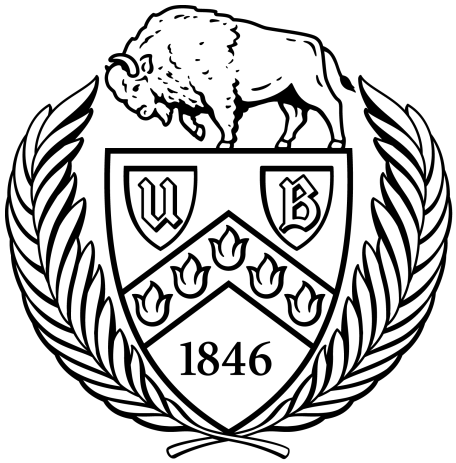
University at Buffalo,
The State University of New York

CSE 676

Deep Learning

Assignment - 2

Yash Narendra Saraf (50290453)



CSE 676: Deep Learning

Assignment - 2

Yash Narendra Saraf
UB ID: 50290453
ysaraf@buffalo.edu

April 13, 2019

Q1 Convolutional Neural Networks [3 points]

(1) [1 point] Define the convolution operator for input $x \in R^2$ and filter $w \in R^2$ as:

$$(x * w)_{ij} \triangleq \sum_k \sum_l x_{k+i,l+j}\ w_{kl}$$

Let, x = $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$; and w = $\begin{bmatrix} 4 & 2 \\ 5 & 1 \end{bmatrix}$**, 1) Compute the result of $x * w$ with padding of size 1 and stride of size 2. 2) What is the relation between input size L, output size O, filter size F, padding size P, and stride S?**

We have padding of size 1. So let us first visualize the matrix x along with the padding.

x (with padding) = $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 0 \\ 0 & 5 & 6 & 7 & 8 & 0 \\ 0 & 9 & 10 & 11 & 12 & 0 \\ 0 & 13 & 14 & 15 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Now since we also have stride = 2, writing convolution equations using stride and new x and w.

Let us define the convolution output as -

$$O_{ij} = (x * w)_{ij} \triangleq \sum_k \sum_l x_{k+i,l+j}\ w_{kl}$$

Here assumed that the matrix indices start from 0.

$$\begin{aligned} O_{0,0} &= x_{0,0}.w_{0,0} + x_{0,1}.w_{0,1} + x_{1,0}.w_{1,0} + x_{1,1}.w_{1,1} \\ &= 0.4 + 0.2 + 0.5 + 1.1 = 1 \\ &= 1 \end{aligned} \tag{1}$$

Since, stride value given is 2, we move the filter by 2 positions to the right.

$$\begin{aligned} O_{0,1} &= x_{0,2}.w_{0,0} + x_{0,3}.w_{0,1} + x_{1,2}.w_{1,0} + x_{1,3}.w_{1,1} \\ &= 0.4 + 0.2 + 2.5 + 3.1 = 15 \\ &= 15 \end{aligned} \tag{2}$$

Similarly, performing other calculations,

$$\begin{aligned} O_{0,2} &= x_{0,4}.w_{0,0} + x_{0,5}.w_{0,1} + x_{1,4}.w_{1,0} + x_{1,5}.w_{1,1} \\ &= 0.4 + 0.2 + 4.5 + 0.1 = 20 \\ &= 20 \end{aligned} \tag{3}$$

Now, we also move down using the stride value -

$$\begin{aligned} O_{1,0} &= x_{2,0}.w_{0,0} + x_{2,1}.w_{0,1} + x_{3,0}.w_{1,0} + x_{3,1}.w_{1,1} \\ &= 0.4 + 5.2 + 0.5 + 9.1 = 19 \\ &= 19 \end{aligned} \tag{4}$$

$$\begin{aligned} O_{1,1} &= x_{2,2}.w_{0,0} + x_{2,3}.w_{0,1} + x_{3,2}.w_{1,0} + x_{3,3}.w_{1,1} \\ &= 6.4 + 7.2 + 10.5 + 11.1 = 99 \\ &= 99 \end{aligned} \tag{5}$$

$$\begin{aligned} O_{1,2} &= x_{2,4}.w_{0,0} + x_{2,5}.w_{0,1} + x_{3,4}.w_{1,0} + x_{3,5}.w_{1,1} \\ &= 8.4 + 0.2 + 12.5 + 0.1 = 92 \\ &= 92 \end{aligned} \tag{6}$$

$$\begin{aligned} O_{2,0} &= x_{4,0}.w_{0,0} + x_{4,1}.w_{0,1} + x_{5,0}.w_{1,0} + x_{5,1}.w_{1,1} \\ &= 0.4 + 13.2 + 0.5 + 0.1 = 26 \\ &= 26 \end{aligned} \tag{7}$$

$$\begin{aligned} O_{2,1} &= x_{4,2}.w_{0,0} + x_{4,3}.w_{0,1} + x_{5,2}.w_{1,0} + x_{5,3}.w_{1,1} \\ &= 14.4 + 15.2 + 0.5 + 0.1 = 86 \\ &= 86 \end{aligned} \tag{8}$$

$$\begin{aligned} O_{2,2} &= x_{4,4}.w_{0,0} + x_{4,5}.w_{0,1} + x_{5,4}.w_{1,0} + x_{5,5}.w_{1,1} \\ &= 16.4 + 0.2 + 0.5 + 0.1 = 64 \\ &= 64 \end{aligned} \tag{9}$$

So, the final result is

$$Output = \begin{bmatrix} 1 & 15 & 20 \\ 19 & 99 & 92 \\ 26 & 86 & 64 \end{bmatrix}$$

Now, the relation between the input size L, output size O, filter size F, padding size P, and stride S can be defined as -

$$O = \frac{L + 2(P) - F}{S} + 1$$

This can be clearly understood from the previous example, as stride values increases the values skips decreases by same ratio. Also padding adds a row and column at both ends and filter size decreases the number by its size.

(2) [2 point] For the LeNet Architecture, the input are images of size 32 x 32, the first layer uses a convolution layer with 6 filters, each with a size of 5 x 5, zero padding and stride of size 1.

(2).1 What is the output size and how many parameters are there in the first layer?

The information given is -

- Input 32×32
- Layer 1 has 6 filters
- Each filter has dimension 5×5

Now, we first calculate the number of parameters -

- Number of parameters per filter = 5 x 5 + 1 = 26 (1 for bias)
- So, Total number of parameters = 26 x Number of filters = 26 x 6 = **156**

Now, we calculate the dimension of output -

- Output dimension of each filter = $O = \frac{L+2(P)-F}{S} + 1$
  $O = \frac{32+2(0)-5}{1} + 1 = 28$
- Each filer will output a 28 x 28 matrix.
- Since, we have 6 filters we will get Output dimension as **28 x 28 x 6**

(2).2 Propose a way to reduce the number of parameters, and calculate how many parameters are there in your proposed schema.

Now, the model parameters can be decreased using number of methods -

First let's go through the LeNet Architecture -

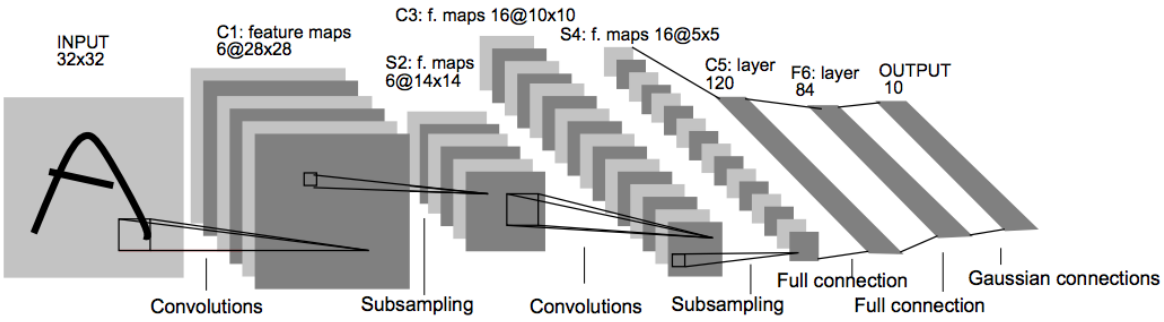


Figure 1: LeNet Architecture

The LeNet Architecture contains 2 convolutions layer followed by sub sampling layer i.e. Max Pooling layers and after flattening the output of second convolutional block it is fed to 2 Fully connected layers.

To reduce the number of parameters following steps can be taken, these steps could also result in somewhat poorer performance due to the model complexity and accuracy tradeoff.

- Reducing Kernal Size to 4 x 4.
- Reducing the number of Kernal from 6 to 5.
- Changing the value of stride from 1 to 2.

Based on above optimization's to the architecture the new number of parameters are -

- Number of parameters per filter = 4 x 4 + 1 = 17 (1 for bias)
- So, Total number of parameters = 17 x Number of filters = 17 x 5 = **85**

- Output dimension of each filter $= O = \frac{L+2(P)-F}{S} + 1$
  $O = \frac{32+2(0)-4}{2} + 1 = 15$
- Each filer will output a 15 x 15 matrix.
- Since, we have 6 filters we will get Output dimension as **15 x 15 x 5**
- The decrease in output dimensionality will directly effect the number of parameters of further layers.
- **Just considering first layer the number of parameters has decreased from 156 to 85 just by 2 small changes.**

Q2 Recurrent Neural Networks [3 Points]

(1) Given the LSTM structure in Figure 1 and the corresponding definition in (1).

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \begin{bmatrix} \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix} & \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \end{bmatrix}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot + g_t$$

$$h_t = o_t \odot tanh(c_t)$$

Let the loss of an LSTM model be L. Assume we have calculated $\frac{\partial L}{\partial i_{t+1}}$, $\frac{\partial L}{\partial f_{t+1}}$, $\frac{\partial L}{\partial o_{t+1}}$, $\frac{\partial L}{\partial g_{t+1}}$, $\frac{\partial L}{\partial c_{t+1}}$, and $\frac{\partial L}{\partial h_{t+1}}$. Derive gradient formulas for $\frac{\partial L}{\partial i_t}$, $\frac{\partial L}{\partial f_t}$, $\frac{\partial L}{\partial o_t}$, $\frac{\partial L}{\partial g_t}$, $\frac{\partial L}{\partial c_t}$, and $\frac{\partial L}{\partial h_t}$

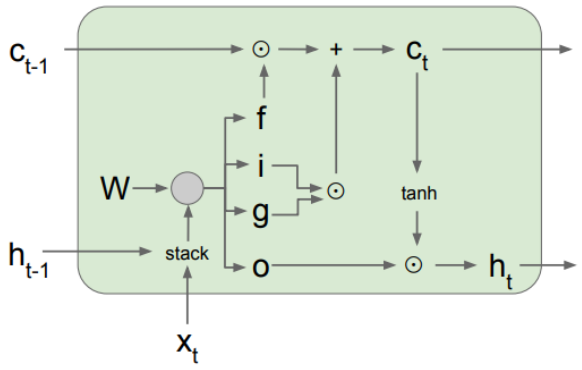


Figure 2: Cell unit for LSTM

We have been given the gradients calculated for timestamp $t+1$. Now, we need to backpropogate based on given values and calculate the gradients for time t.

The problem can be divided into 2 parts. The first part is calculating the $\frac{\partial L}{\partial h_t}$ in terms of gradients from timestamp $t+1$. And then using the value of $\frac{\partial L}{\partial h_t}$ to update all the gradient value at time t.

In LSTM, architecture as can be seen from the figure the Backpropogation through time takes place by 2 nodes i.e. c_{t+1} and h_t.

So, now solving the equations at time stamp t+1 to calculate the gradient value for h_t which would be required for the calculation of all other gradients at time t.

Assigning notations -

$$z_t = \begin{bmatrix} \hat{i_t} \\ \hat{f_t} \\ \hat{o_t} \\ \hat{g_t} \end{bmatrix} = \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{10}$$

$$
\begin{aligned}
z_t &= W \times I_t \\
i_t &= \sigma(\hat{i}_t) \\
f_t &= \sigma(\hat{f}_t) \\
o_t &= \sigma(\hat{o}_t) \\
g_t &= \sigma(\hat{g}_t)
\end{aligned}
\tag{11}
$$

We have the values for $\frac{\partial L}{\partial i_{t+1}}$, $\frac{\partial L}{\partial f_{t+1}}$, $\frac{\partial L}{\partial o_{t+1}}$, $\frac{\partial L}{\partial g_{t+1}}$, $\frac{\partial L}{\partial c_{t+1}}$, and $\frac{\partial L}{\partial h_{t+1}}$ i.e. δi_{t+1}, δf_{t+1}, δo_{t+1}, δg_{t+1}, δc_{t+1}, and δh_{t+1}.

Using these values along with equation 1 and 2 to calculate δh_t.

So, applying partial derivatives in equation 2 for time t+1.

$$
\begin{aligned}
\delta\hat{i}_{t+1} &= \delta i_{t+1} \odot i_t \odot (1 - i_{t+1}) \\
\delta\hat{f}_{t+1} &= \delta f_{t+1} \odot f_t \odot (1 - f_{t+1}) \\
\delta\hat{o}_{t+1} &= \delta o_{t+1} \odot o_t \odot (1 - o_{t+1}) \\
\delta\hat{g}_{t+1} &= \delta g_{t+1} \odot (1 - tanh^2(\hat{g}_{t+1}))
\end{aligned}
\tag{12}
$$

Now, using equation 3 and equation 1 we have.

$$
\delta z_{t+1} = \begin{bmatrix} \hat{i}_{t+1} \\ \hat{f}_{t+1} \\ \hat{o}_{t+1} \\ \hat{g}_{t+1} \end{bmatrix} \tag{13}
$$

Using equation 4 along with equation 1.

We get

$$
\delta I_{t+1} = W^T \times \delta z_{t+1} \tag{14}
$$

As,

$$
I_{t+1} = \begin{pmatrix} h_t \\ x_{t+1} \end{pmatrix} \tag{15}
$$

So,

$$
\delta I_{t+1} = \begin{pmatrix} \delta h_t \\ \delta x_{t+1} \end{pmatrix} \tag{16}
$$

We, will obtain the value of δh_t from above matrix which is being backpropogated by further time steps.

Now, moving to the second part of the solution, i.e. considering the state of the cell at time t.

The δh_t will contain 2 parts i.e. the backpropogated error calculated earlier and the Error difference calculated for that particular time stamp.

So, we have,

$$
\delta h_t = \Delta_t + \delta h_t(From\ time\ t+1) \tag{17}
$$

Since, both the values are known we can calculate the value of δh_t.

Now, solving the equations for time t.

The front pass equations are given as -

$$
\begin{aligned}
c_t &= f_t \odot c_{t-1} + i_t \odot + g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
\tag{18}
$$

Since we have δh_t we can use it to calculate δi_t, δf_t, δo_t, δg_t, and δc_t.

Starting with equation 9. Calculation for δc_t. Using partial derivatives.

$$\begin{aligned} \delta o_t &= \frac{\partial L}{\partial o_t} = \frac{\partial L}{\partial h_t} . \frac{\partial h_t}{\partial o_t} \\ \delta o_t &= \delta h_t . tanh(c_t) \\ \delta o_t &= \delta h_t \odot tanh(c_t) \end{aligned} \tag{19}$$

Now, calculating δc_t using equations 9. The backpropogation path to c_t comes from h_t and c_{t+1}.

So, we have

$$\begin{aligned} \delta c_t &= \frac{\partial L}{\partial c_t} = \frac{\partial L}{\partial h_t} . \frac{\partial h_t}{\partial c_t} + \frac{\partial L}{\partial c_{t+1}} . \frac{\partial c_{t+1}}{\partial c_t} \\ \delta c_t &= \delta h_t \odot o_t \odot (1 - tanh^2(c_t)) + \delta c_{t+1} \odot f_t \end{aligned} \tag{20}$$

Now, calculating gradients i.e. δi_t,δf_t, and δg_t aas they depend on δc_t.

$$\begin{aligned} \delta i_t &= \frac{\partial L}{\partial i_t} = \frac{\partial L}{\partial c_t} . \frac{\partial c_t}{\partial i_t} \\ \delta i_t &= \delta c_t . g_t \\ \delta i_t &= \delta h_t \odot g_t \end{aligned} \tag{21}$$

Similarly,

$$\begin{aligned} \delta f_t &= \frac{\partial L}{\partial f_t} = \frac{\partial L}{\partial c_t} . \frac{\partial c_t}{\partial f_t} \\ \delta f_t &= \delta c_t . c_{t-1} \\ \delta f_t &= \delta h_t \odot c_{t-1} \end{aligned} \tag{22}$$

Similarly,

$$\begin{aligned} \delta g_t &= \frac{\partial L}{\partial g_t} = \frac{\partial L}{\partial c_t} . \frac{\partial c_t}{\partial g_t} \\ \delta g_t &= \delta c_t . i_t \\ \delta g_t &= \delta h_t \odot i_t \end{aligned} \tag{23}$$

So, to summarize the solution - The final equations would be -

We obtain the value of δh_t from equation 8. The value is calculated as the sum of gradient calculated at that timestamp and the backpropogated value from δI_{t+1} which is a stack of δh_t and δx_t.

$$\begin{aligned} \delta o_t &= \delta h_t \odot tanh(c_t) \\ \delta c_t &= \delta h_t \odot o_t \odot (1 - tanh^2(c_t)) + \delta c_{t+1} \odot f_t \\ \delta i_t &= \delta h_t \odot g_t \\ \delta f_t &= \delta h_t \odot c_{t-1} \\ \delta g_t &= \delta h_t \odot i_t \end{aligned}$$

For understanding LSTM Backpropogation the following references were used.

- Arun Mallya Github.io Blog
- Colah's Blog
- A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation

Q3 Variational Autoencoder [3 points]

(1) Assume we have the true posterior distribution of latent variable z given data $x, p(z|x)$, as a mixture of two Gaussian, with contour shown as blue curves in Figure. In VAE, we have use of proposal distribution $q(z|x)$ to approximate $p(z|x)$. The possibly learned distributions $q(z|x)$ are plotted red curves in Figure.

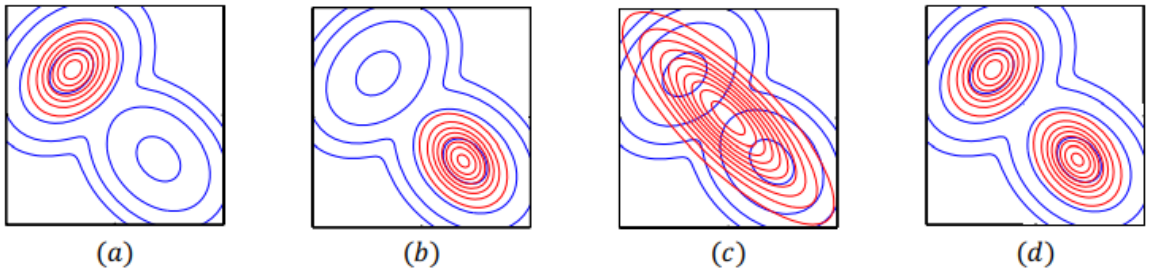


Figure 3: True Posterior distribution $p(z|x)$ (blue) and variational distribution $q(z|x)$ (red).

(1).1 [1 point] Point out what form(s) of $q(z|x)$ can be learned with standard VAE. Choose the corresponding plots (a), (b), (c), and (d) in Figure. This is potentially a multiple choice problem

The **(a), (b), and, (c)** can be learned using the standard VAE.

(1).2 [2 points] Explain your answer

The standard VAE is based on the assumption that the Latent space variables are a Gaussian Distribution. This assumption is made to simplify the calculation of Variational Inference problem.

So, the fourth distribution is not a Gaussian distribution so it can't be learnt.

Follow the following equations for the calculation of the LowerBound term.

$$
\begin{aligned}
KL\Big(\frac{q(x)}{p(z/x)}\Big) &= -\sum q(x)\, log\Big(\frac{p(z/x)}{q(z)}\Big) \\
&= -\sum q(x)\, log\Big(\frac{p(x,z)}{q(z) * p(x)}\Big) \\
&= -\sum q(x)\, \Big(log\Big(\frac{p(x,z)}{q(z)}\Big) + log\Big(\frac{1}{p(x)}\Big)\Big) \\
&= -\sum q(x)\, \Big(log\Big(\frac{p(x,z)}{q(z)}\Big) - log(p(x))\Big) \\
&= -\sum q(x)\, \Big(log\Big(\frac{p(x,z)}{q(z)}\Big)\Big) + log(p(x)) \sum q(x) \\
&= -\sum q(x)\, \Big(log\Big(\frac{p(x,z)}{q(z)}\Big)\Big) + log(p(x)) \\
KL\Big(\frac{q(x)}{p(z/x)}\Big) &= -\sum q(x)\, \Big(log\Big(\frac{p(x,z)}{q(z)}\Big)\Big) + log(p(x)) \\
KL\Big(\frac{q(x)}{p(z/x)}\Big) &= -Variational\, Lowerbound + log(p(x))
\end{aligned}
$$

In the above equation the $log(p(x))$ is the constant term. Here we are trying to minimize the KL divergence. Minimizing KL Divergence is same as maximizing Lower Bound.

$$
\begin{aligned}
Lower\ Bound &= \sum q(x) \left(log\Big(\frac{p(x,z)}{q(z)}\Big)\right) \\
&= \sum q(x) \left(log\Big(\frac{p(x/z).p(z)}{q(z)}\Big)\right) \\
&= \sum q(x) \left(log(p(x/z)) + log\Big(\frac{p(z)}{q(z)}\Big)\right) \\
&= \sum q(x)\, log(p(x/z)) - D_{KL}[q(z)||p(z)] \\
&= E_{q(z)} log(p(x/z) - D_{KL}[q(z)||p(z)]
\end{aligned}
$$

Now while optimizing we try to maximize the Lower Bound. The first term corresponds to the reconstruction loss i.e. how close is the reconstructed output with respect to the input. Since here we assume that $q(z)$ distribution is Gaussian, the second term corresponding to the KL Divergence tries to Normalize the latent space vector as close to the Gaussian Distribution as possible by reducing the divergence.

For reference lecture slides and book Pattern Recognition and Machine Learning by Christopher Bishop has been used.

Q4 Generative Adversarial Networks [1 point]

(1) Run any GAN on the MNIST dataset, and plot the curves of generator and discriminator losses versus the number of epochs. You can reuse any code online

For this question code can be found in the submitted folder directory. The name of the file is **Q4_GAN.ipynb**

The code is written inside jupyter notebook. For reference this GitHub Repository along with Skymind Blog Post was used.

The result graphs for the code can be found below -

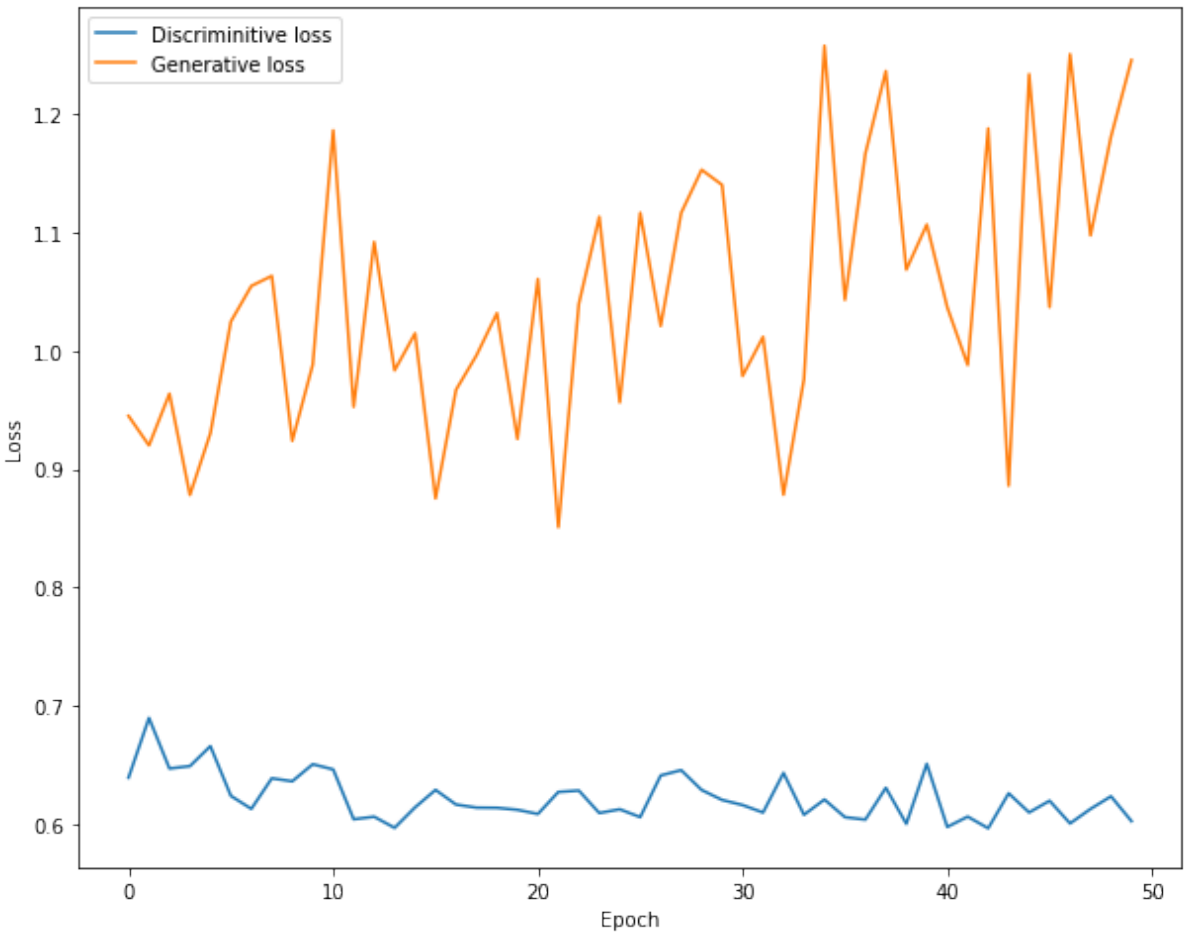


Figure 4: Plot for Discriminator and Generator loses with respect to Number of Epochs