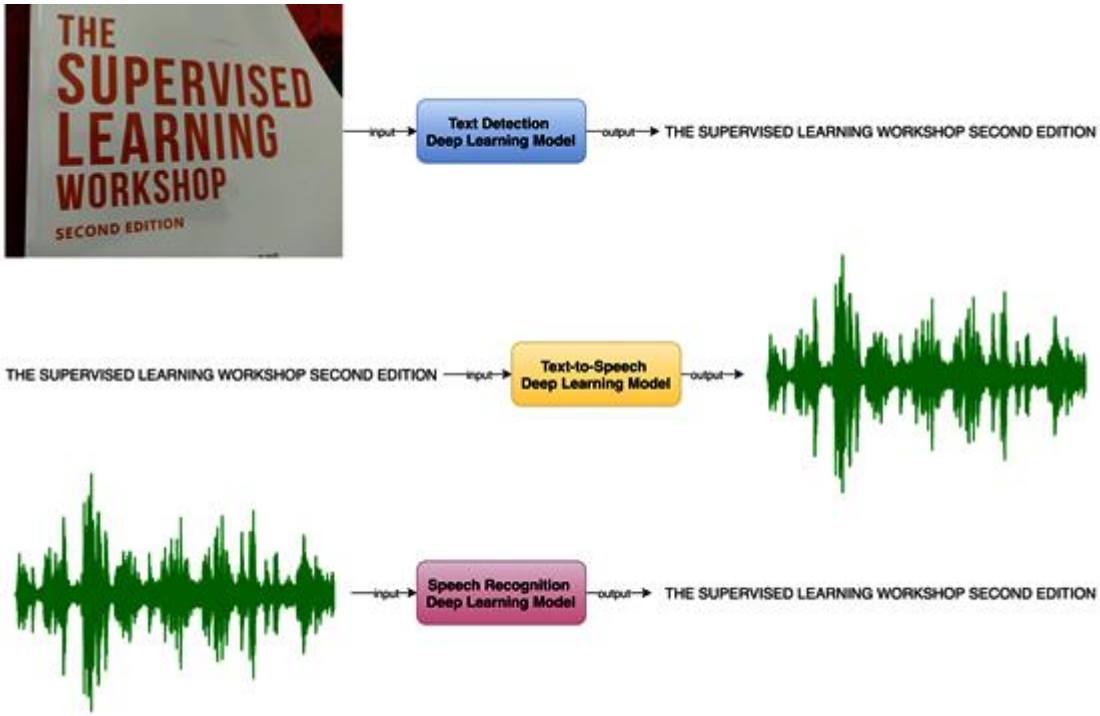
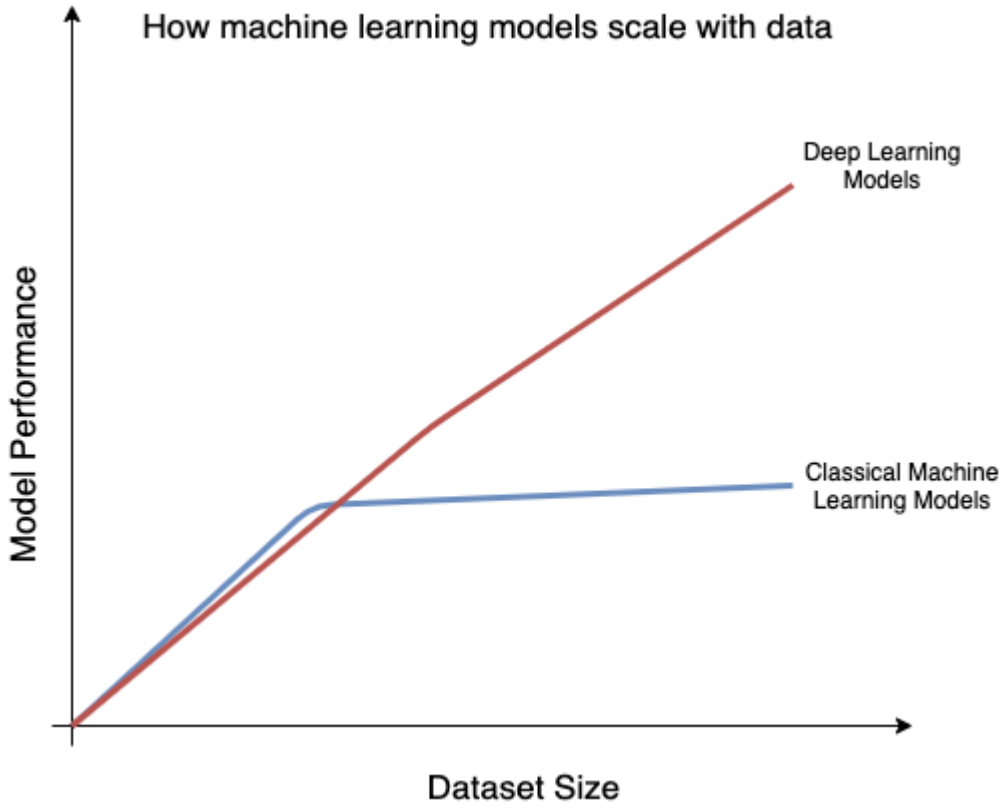
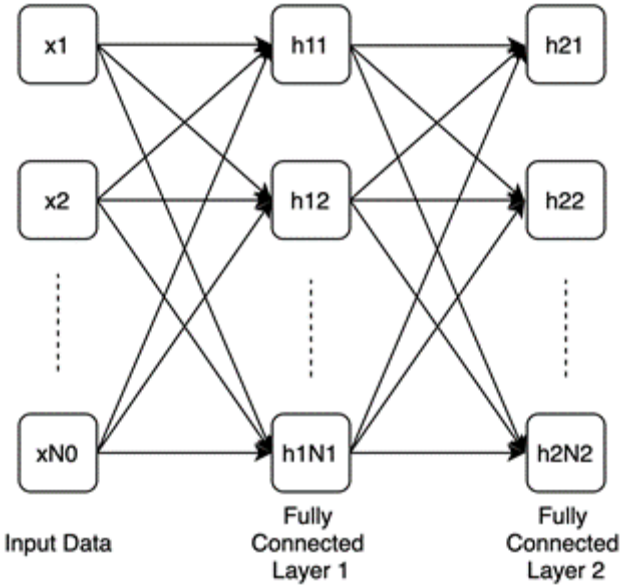
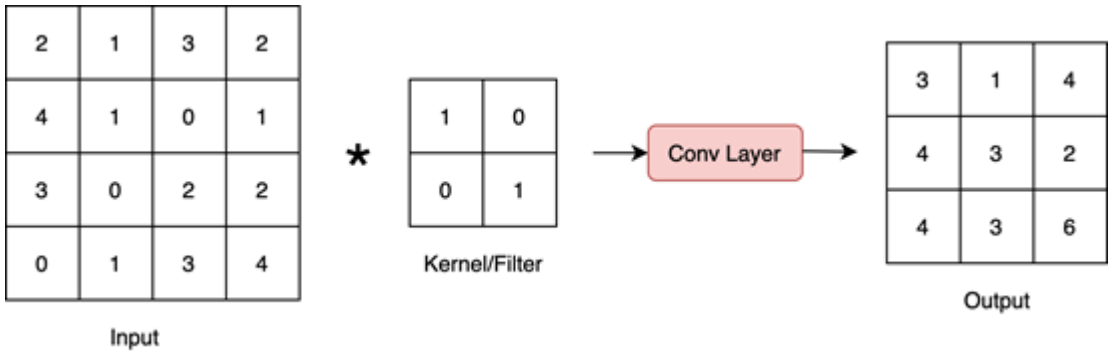
Chapter 1: Overview of Deep Learning Using PyTorch



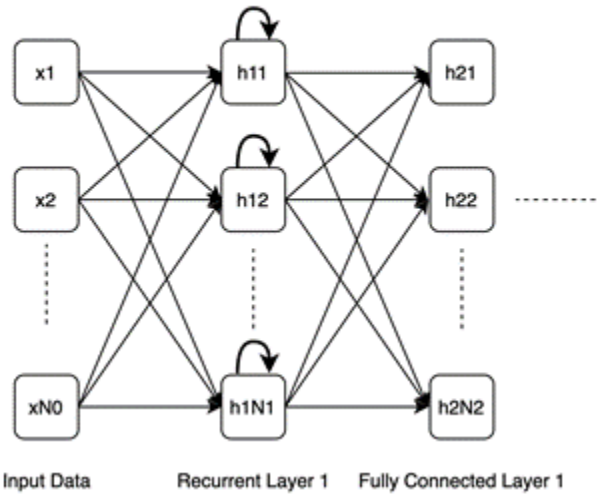


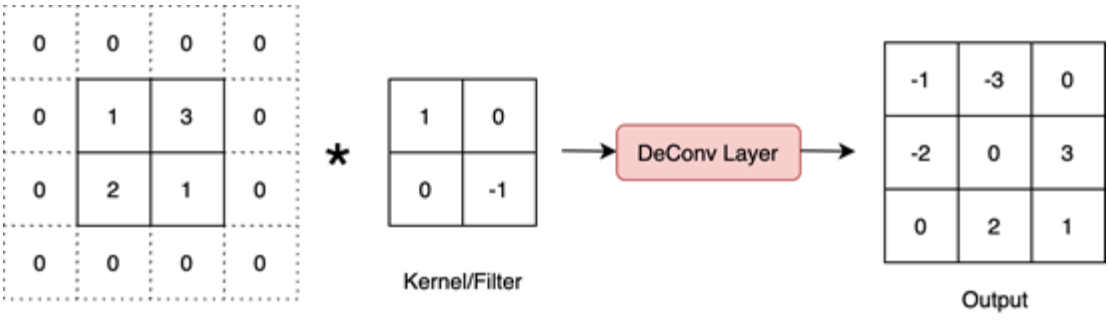
Fully Connected Layer

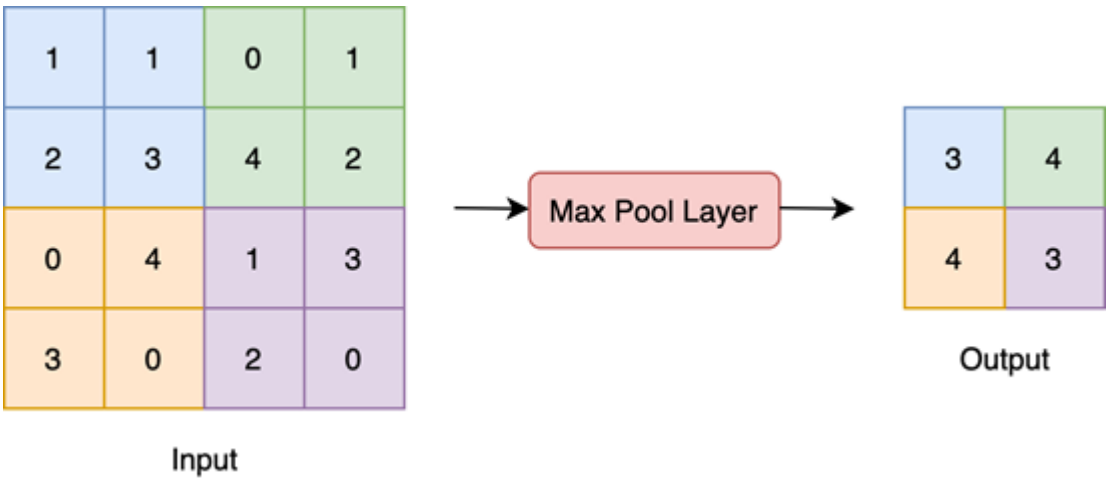


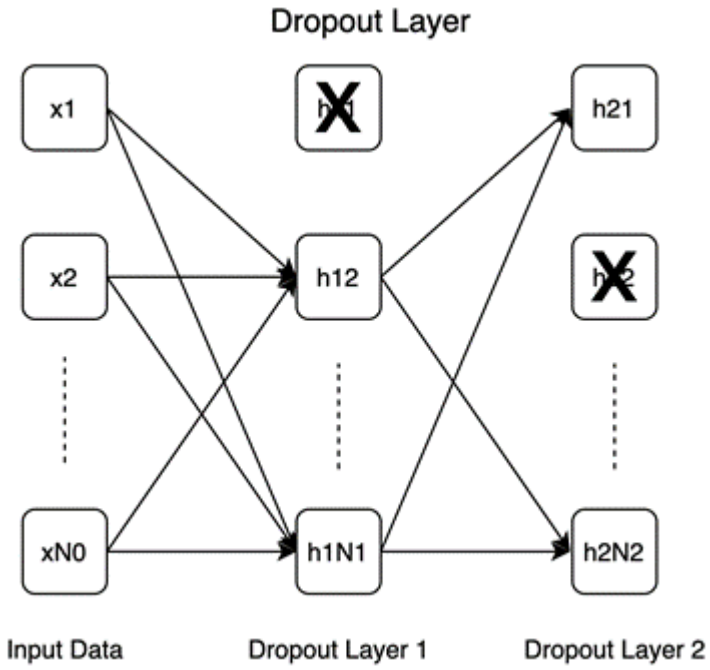


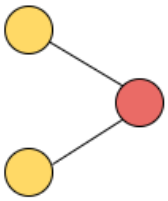
Recurrent layer



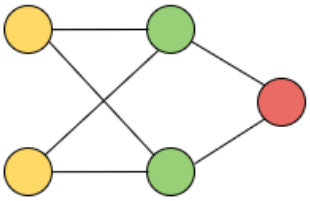

0 0 0 0
0 1 3 0
0 2 1 0
0 0 0 0
Input
*
1 0
0 -1
Kernel/Filter
DeConv Layer
-1 -3 0
-2 0 3
0 2 1
Output


1 1 0 1
2 3 4 2
0 4 1 3
3 0 2 0
Input
Max Pool Layer
3 4
4 3
Output

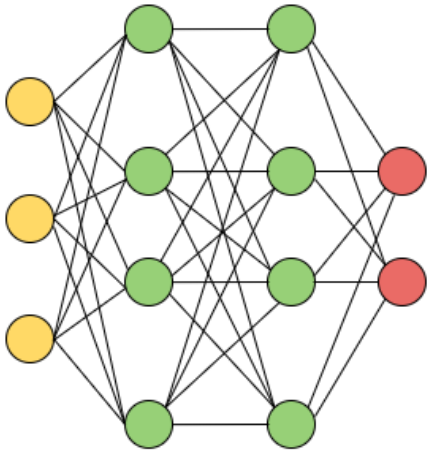

Dropout Layer
x1
x2
xN0
h12
h1N1
h21
h2N2
Input Data
Dropout Layer 1
Dropout Layer 2



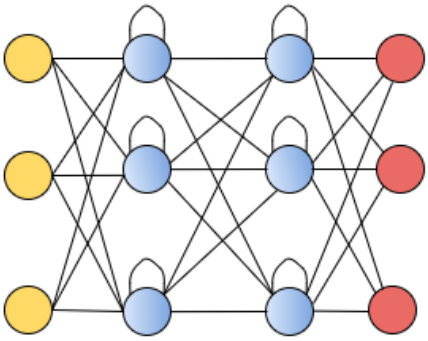
Perceptron



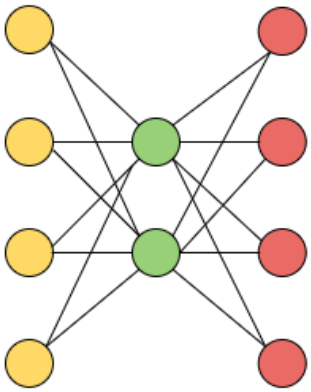
Feed forward network



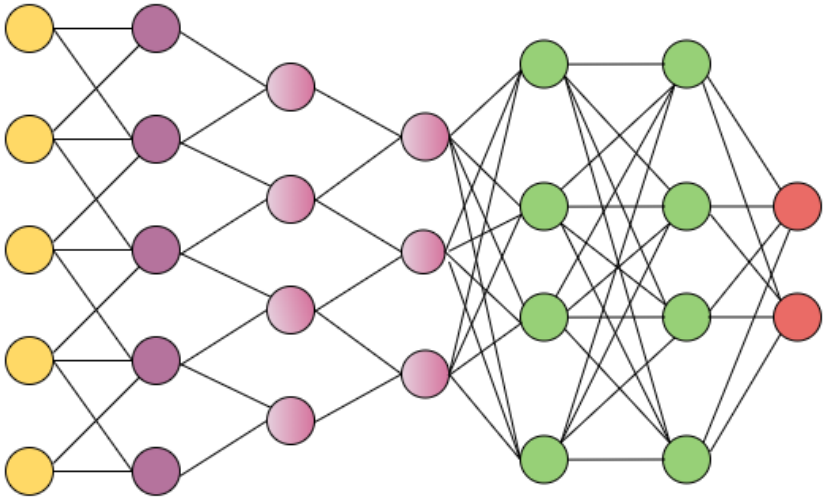
Deep feed forward network



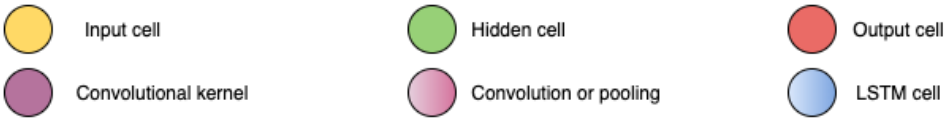
LSTM network

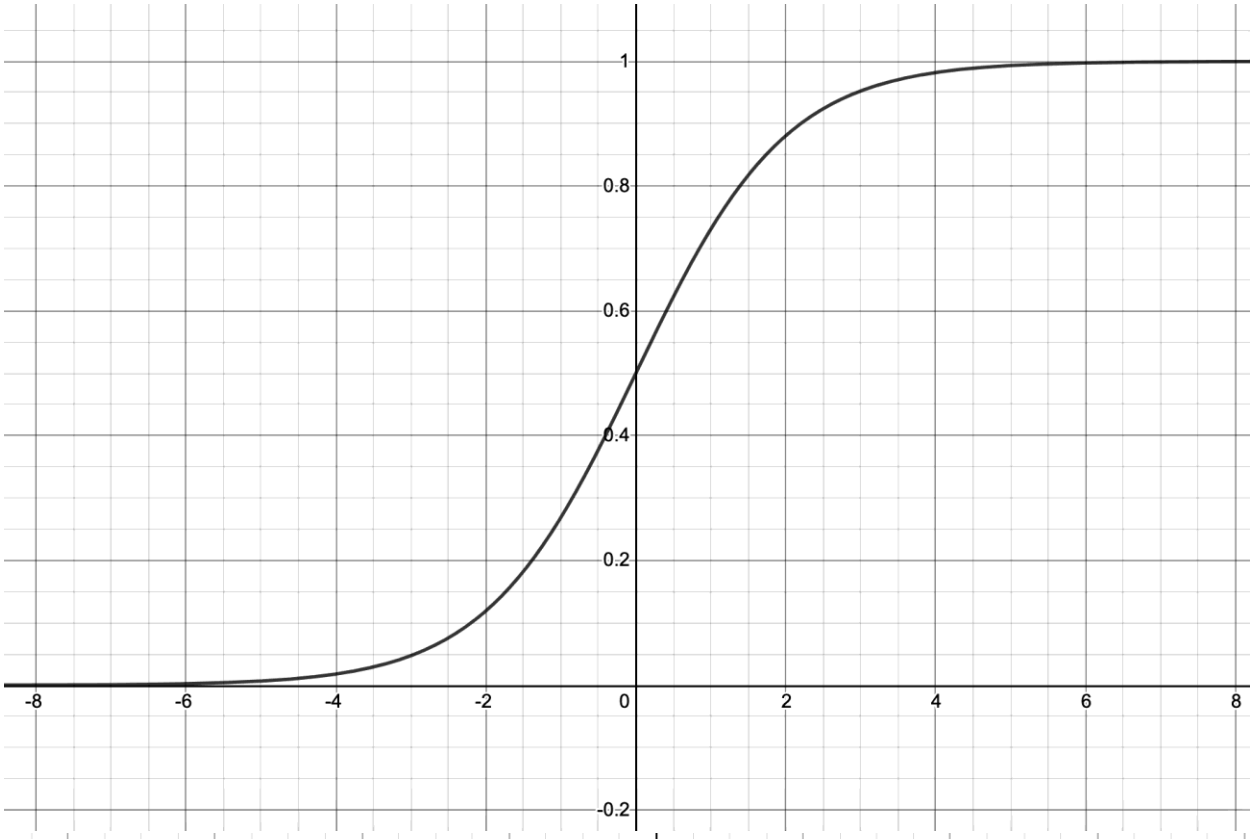


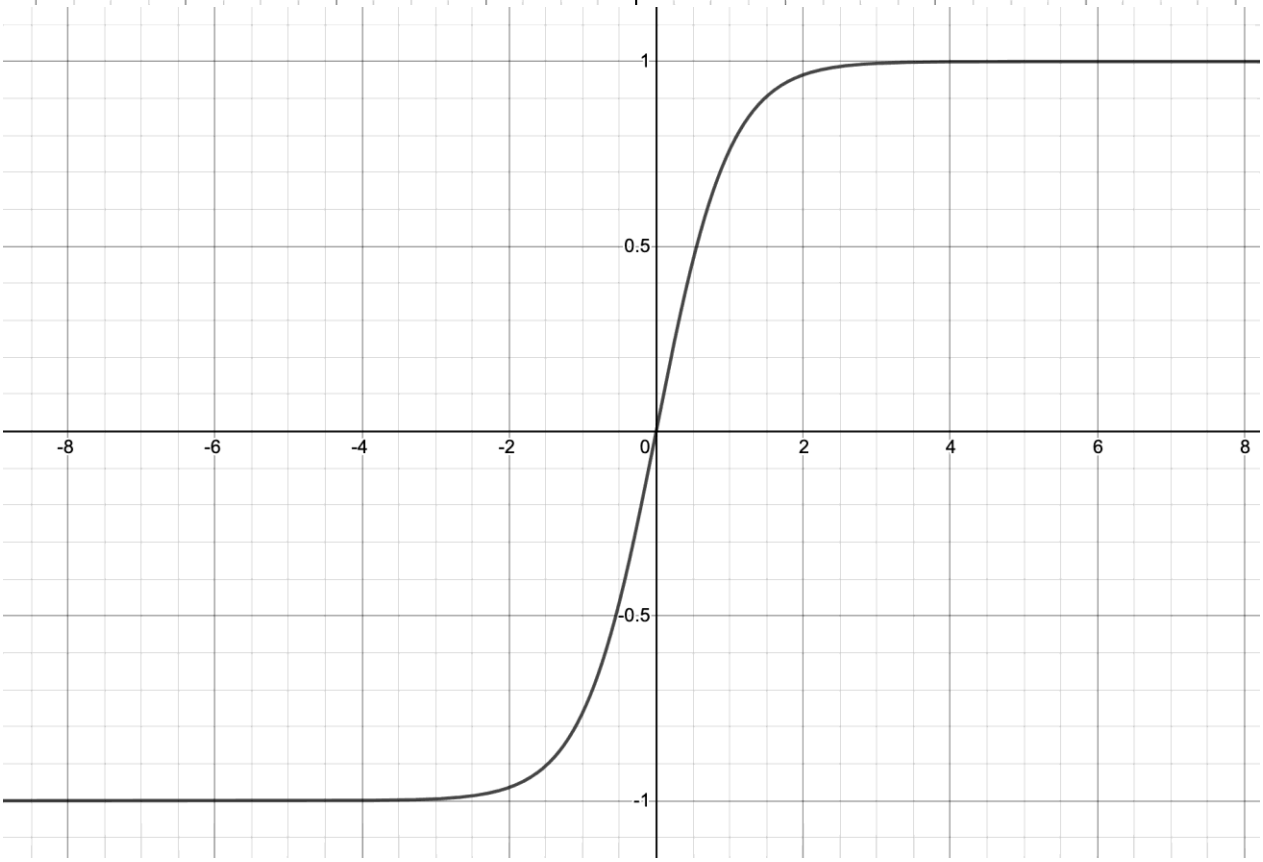
Autoencoder

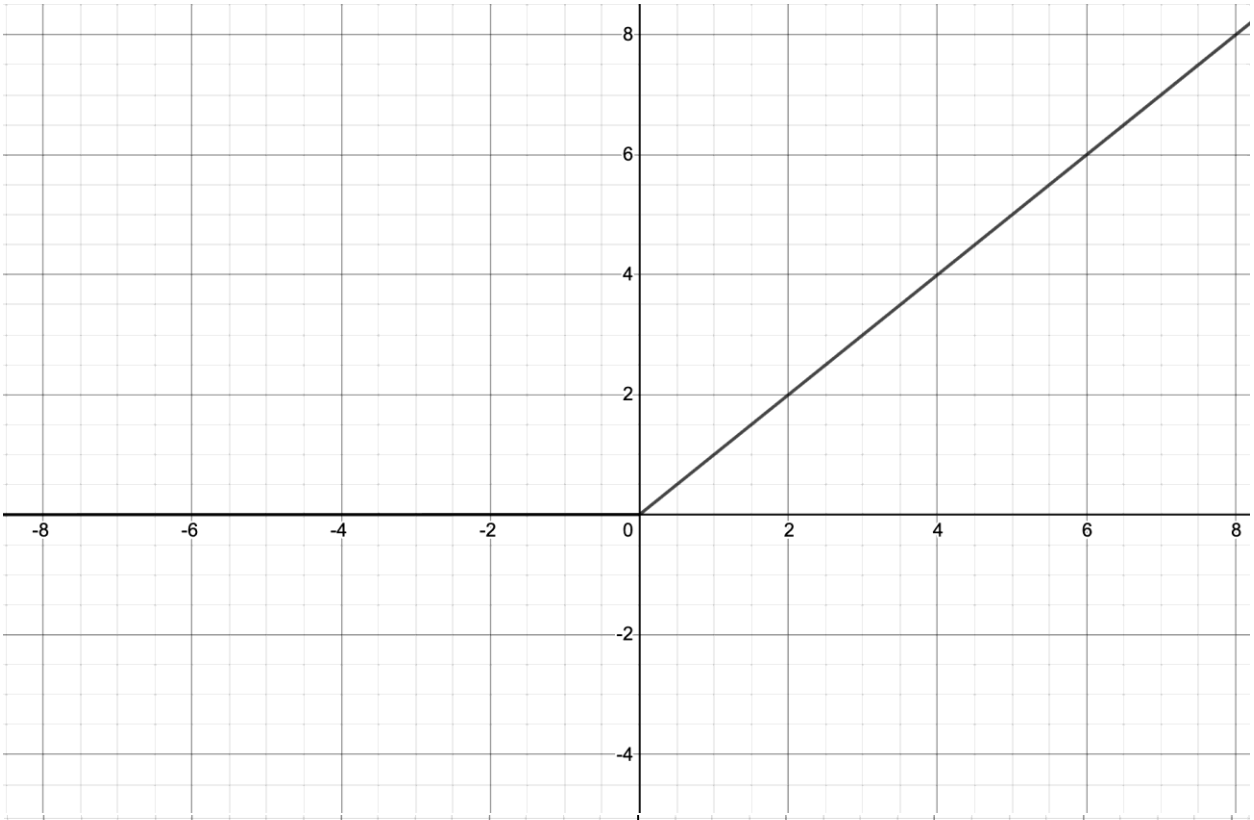


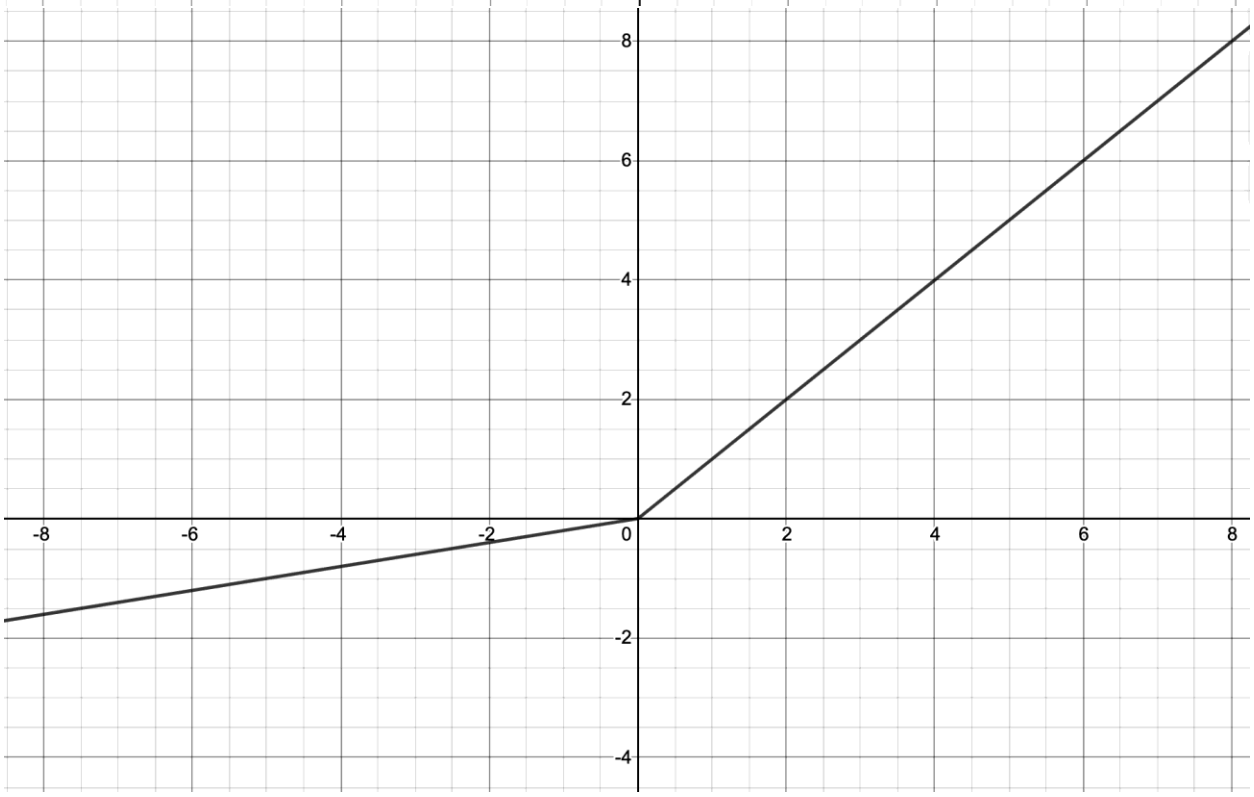
Deep Convolutional Neural Network




1
0.8
0.6
0.4
0.2
-0.2
-8
-6
-4
-2
0
2
4
6
8


1
0.5
-0.5
-1
-8
-6
-4
-2
0
2
4
6
8


8
6
4
2
-8
-6
-4
-2
0
2
4
6
8
-2
-4


8
6
4
2
-8
-6
-4
-2
0
2
4
6
8
-2
-4

```
points = torch.tensor([[1.0, 4.0], [2.0, 1.0], [3.0, 5.0]])
points.storage()
```

```
 1.0
 4.0
 2.0
 1.0
 3.0
 5.0
[torch.FloatStorage of size 6]
```

```
points.size()
```

```
torch.Size([3, 2])
```

```
points.storage_offset()
```

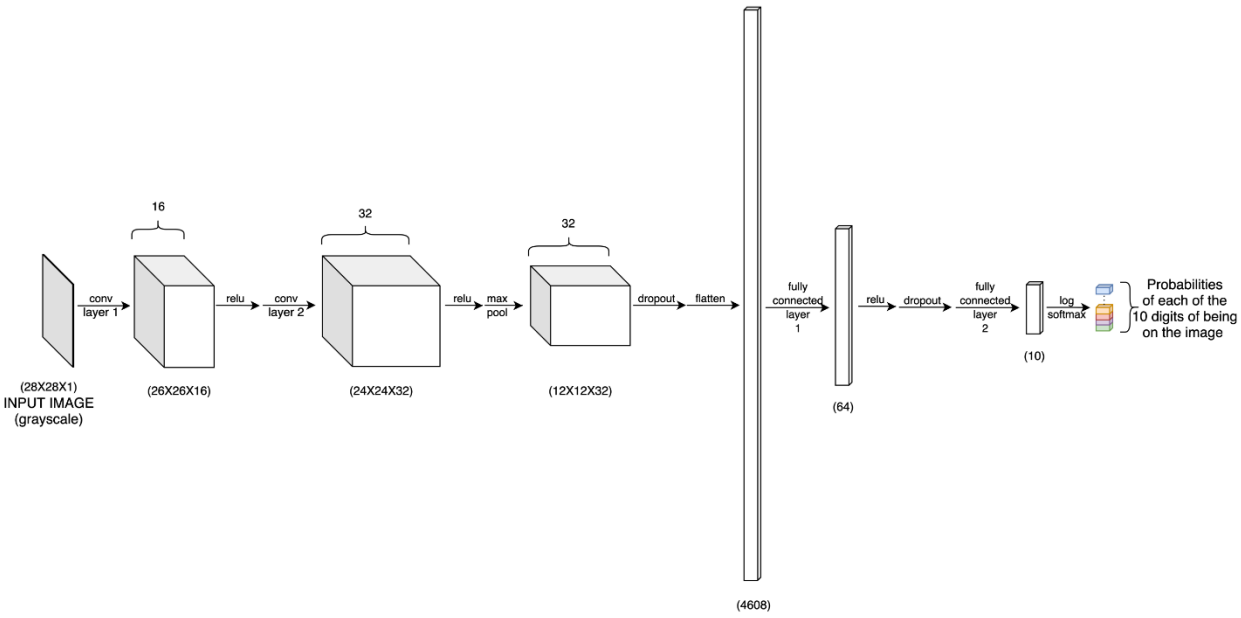
```
0
```

```
points[1].storage_offset()
```

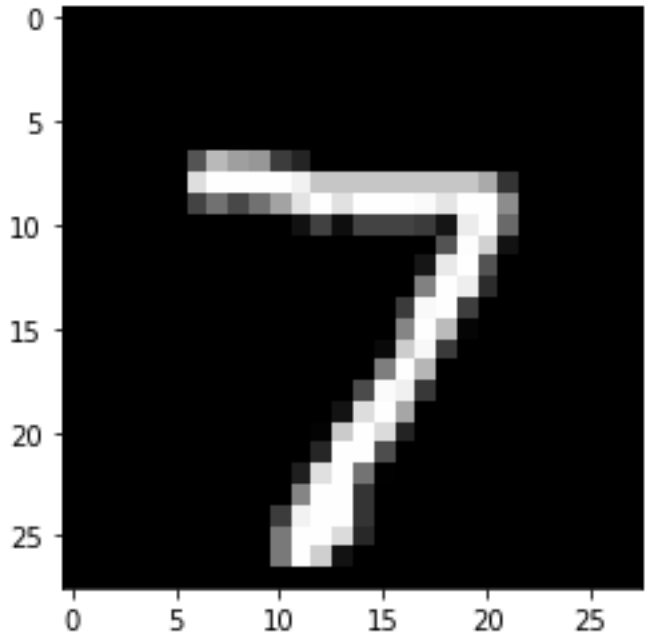
```
2
```

```
points.stride()
```

```
(2, 1)
```


16
32
32
conv layer 1
relu
conv layer 2
relu
max pool
dropout
flatten
fully connected layer 1
relu
dropout
fully connected layer 2
log softmax
Probabilities of each of the 10 digits of being on the image
(28X28X1)
INPUT IMAGE
(grayscale)
(26X26X16)
(24X24X32)
(12X12X32)
(4608)
(64)
(10)

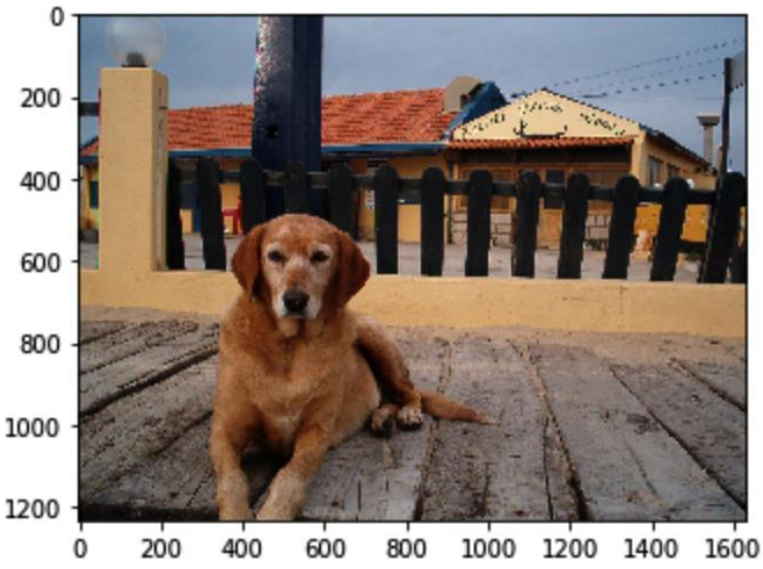
```
epoch: 1 [0/60000 (0%)]  training loss: 2.306125
epoch: 1 [320/60000 (1%)]        training loss: 1.623073
epoch: 1 [640/60000 (1%)]        training loss: 0.998695
epoch: 1 [960/60000 (2%)]        training loss: 0.953389
epoch: 1 [1280/60000 (2%)]       training loss: 1.054391
epoch: 1 [1600/60000 (3%)]       training loss: 0.393427
epoch: 1 [1920/60000 (3%)]       training loss: 0.235708
epoch: 1 [2240/60000 (4%)]       training loss: 0.284237
epoch: 1 [2560/60000 (4%)]       training loss: 0.203838
epoch: 1 [2880/60000 (5%)]       training loss: 0.292076
epoch: 1 [3200/60000 (5%)]       training loss: 0.541438
epoch: 1 [3520/60000 (6%)]       training loss: 0.411091
epoch: 1 [3840/60000 (6%)]       training loss: 0.323946
epoch: 1 [4160/60000 (7%)]       training loss: 0.296546
                                 ⋮
epoch: 2 [56000/60000 (93%)]     training loss: 0.072877
epoch: 2 [56320/60000 (94%)]     training loss: 0.112689
epoch: 2 [56640/60000 (94%)]     training loss: 0.003503
epoch: 2 [56960/60000 (95%)]     training loss: 0.002715
epoch: 2 [57280/60000 (95%)]     training loss: 0.089225
epoch: 2 [57600/60000 (96%)]     training loss: 0.184287
epoch: 2 [57920/60000 (97%)]     training loss: 0.044174
epoch: 2 [58240/60000 (97%)]     training loss: 0.097794
epoch: 2 [58560/60000 (98%)]     training loss: 0.018629
epoch: 2 [58880/60000 (98%)]     training loss: 0.062386
epoch: 2 [59200/60000 (99%)]     training loss: 0.031968
epoch: 2 [59520/60000 (99%)]     training loss: 0.009200
epoch: 2 [59840/60000 (100%)]    training loss: 0.021790

Test dataset: Overall Loss: 0.0489, Overall Accuracy: 9850/10000 (98%)
```

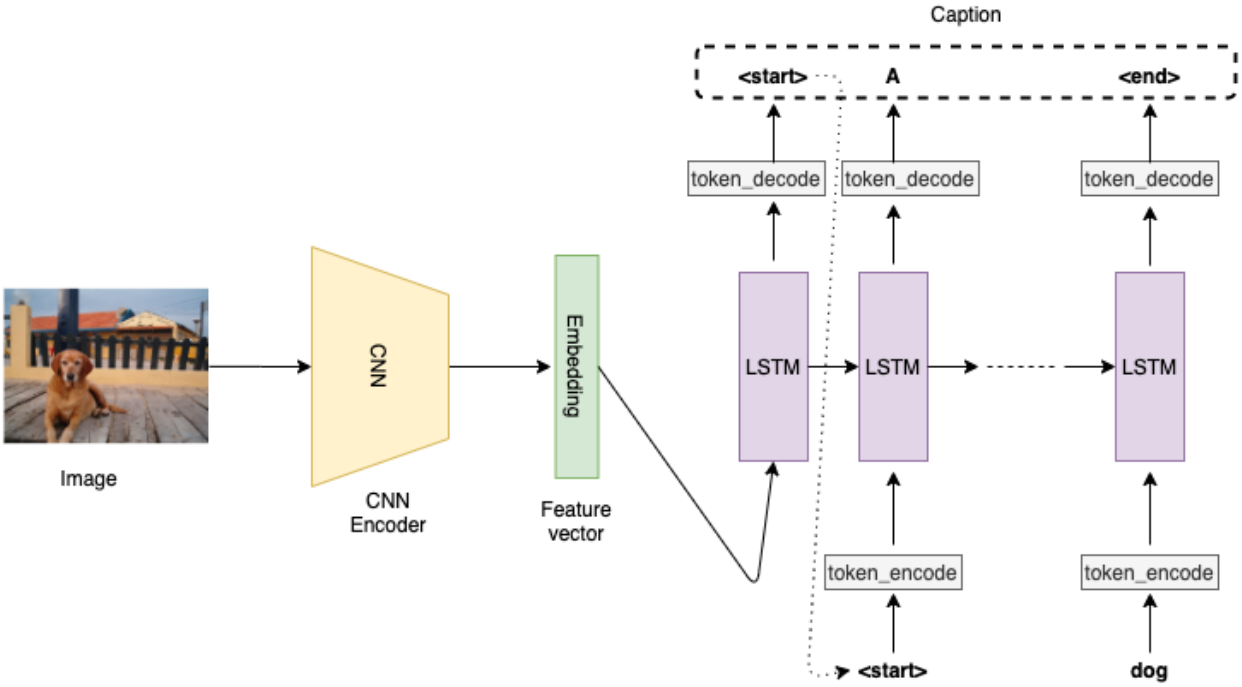



```
Model prediction is : 7
Ground truth is : 7
```

Chapter 2: Combining CNNs and LSTMs

```
<start> a dog is standing on a sidewalk near a building . <end>
```





```
--2020-05-19 06:45:20--  http://msvocds.blob.core.windows.net/annotations-1-0-3/captions_train-val2014.zip
Resolving msvocds.blob.core.windows.net (msvocds.blob.core.windows.net)... 52.176.224.96
Connecting to msvocds.blob.core.windows.net (msvocds.blob.core.windows.net)|52.176.224.96|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19673183 (19M) [application/octet-stream Charset=UTF-8]
Saving to: './data/captions_train-val2014.zip'

captions_train-val2 100%[===================>]  18.76M   220KB/s    in 6m 46s

2020-05-19 06:52:07 (47.4 KB/s) - './data/captions_train-val2014.zip' saved [19673183/19673183]

--2020-05-19 06:52:07--  http://images.cocodataset.org/zips/train2014.zip
Resolving images.cocodataset.org (images.cocodataset.org)... 52.216.143.4
Connecting to images.cocodataset.org (images.cocodataset.org)|52.216.143.4|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13510573713 (13G) [application/zip]
Saving to: './data/train2014.zip'

train2014.zip        63%[===========>        ]   8.03G  --.-KB/s    in 4h 54m
```



```
extracting: ./data/val2014/COCO_val2014_000000014526.jpg
extracting: ./data/val2014/COCO_val2014_000000154892.jpg
extracting: ./data/val2014/COCO_val2014_000000535313.jpg
extracting: ./data/val2014/COCO_val2014_000000008483.jpg
extracting: ./data/val2014/COCO_val2014_000000259087.jpg
extracting: ./data/val2014/COCO_val2014_000000030667.jpg
extracting: ./data/val2014/COCO_val2014_000000132288.jpg
extracting: ./data/val2014/COCO_val2014_000000155617.jpg
extracting: ./data/val2014/COCO_val2014_000000049682.jpg
extracting: ./data/val2014/COCO_val2014_000000382438.jpg
extracting: ./data/val2014/COCO_val2014_000000488693.jpg
extracting: ./data/val2014/COCO_val2014_000000324492.jpg
extracting: ./data/val2014/COCO_val2014_000000543836.jpg
extracting: ./data/val2014/COCO_val2014_000000551804.jpg
extracting: ./data/val2014/COCO_val2014_000000045516.jpg
extracting: ./data/val2014/COCO_val2014_000000347233.jpg
extracting: ./data/val2014/COCO_val2014_000000154202.jpg
extracting: ./data/val2014/COCO_val2014_000000038210.jpg
extracting: ./data/val2014/COCO_val2014_000000113113.jpg
extracting: ./data/val2014/COCO_val2014_000000441814.jpg
```

```
loading annotations into memory...
Done (t=0.79s)
creating index...
index created!
[1000/414113] Tokenized the captions.
[2000/414113] Tokenized the captions.
[3000/414113] Tokenized the captions.
[4000/414113] Tokenized the captions.
[5000/414113] Tokenized the captions.
[6000/414113] Tokenized the captions.
[7000/414113] Tokenized the captions.
[8000/414113] Tokenized the captions.
[9000/414113] Tokenized the captions.
[10000/414113] Tokenized the captions.
                    :
                    :
                    :
                    :
                    :
[407000/414113] Tokenized the captions.
[408000/414113] Tokenized the captions.
[409000/414113] Tokenized the captions.
[410000/414113] Tokenized the captions.
[411000/414113] Tokenized the captions.
[412000/414113] Tokenized the captions.

[413000/414113] Tokenized the captions.
[414000/414113] Tokenized the captions.
Total vocabulary size: 9956
Saved the vocabulary wrapper to './data_dir/vocab.pkl'
```

```
[100/82783] Resized the images and saved into './data_dir/resized_images/'.
[200/82783] Resized the images and saved into './data_dir/resized_images/'.
[300/82783] Resized the images and saved into './data_dir/resized_images/'.
[400/82783] Resized the images and saved into './data_dir/resized_images/'.
[500/82783] Resized the images and saved into './data_dir/resized_images/'.
[600/82783] Resized the images and saved into './data_dir/resized_images/'.
[700/82783] Resized the images and saved into './data_dir/resized_images/'.
[800/82783] Resized the images and saved into './data_dir/resized_images/'.
[900/82783] Resized the images and saved into './data_dir/resized_images/'.
[1000/82783] Resized the images and saved into './data_dir/resized_images/'.
```



```
[82000/82783] Resized the images and saved into './data_dir/resized_images/'.
[82100/82783] Resized the images and saved into './data_dir/resized_images/'.
[82200/82783] Resized the images and saved into './data_dir/resized_images/'.
[82300/82783] Resized the images and saved into './data_dir/resized_images/'.
[82400/82783] Resized the images and saved into './data_dir/resized_images/'.
[82500/82783] Resized the images and saved into './data_dir/resized_images/'.
[82600/82783] Resized the images and saved into './data_dir/resized_images/'.
[82700/82783] Resized the images and saved into './data_dir/resized_images/'.
```

```
loading annotations into memory...
Done (t=0.95s)
creating index...
index created!
```

```
Downloading: "https://download.pytorch.org/models/resnet152-b121ed2d.pth" to /Users/ashish.jha/.cache/torch/checkpoints/resnet152-b121ed2d.pth
```

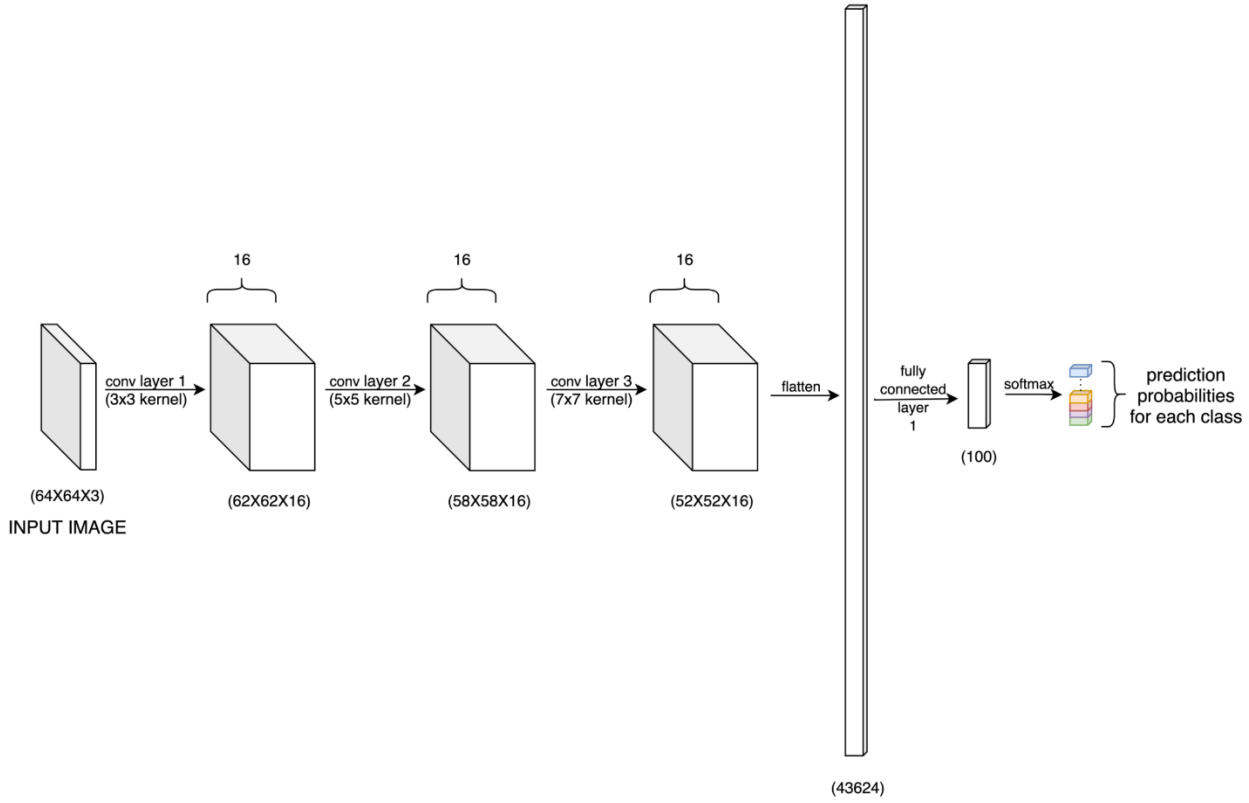
100% 230M/230M [21:30:44<00:00, 3.12kB/s]

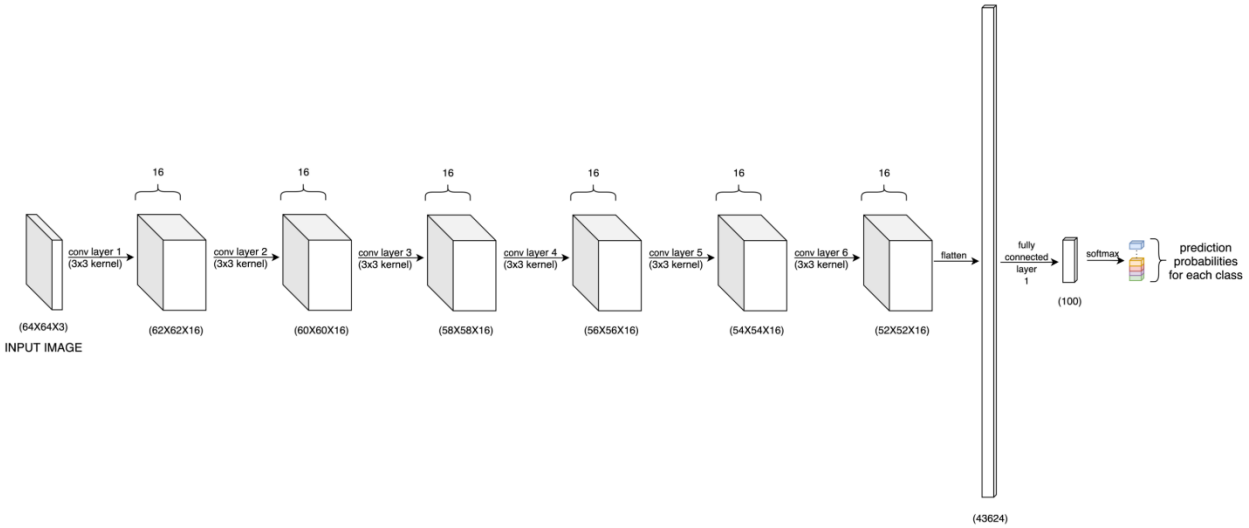
```
Epoch [0/5], Step [0/3236], Loss: 9.2069, Perplexity: 9965.6803
Epoch [0/5], Step [10/3236], Loss: 5.8838, Perplexity: 359.1789
Epoch [0/5], Step [20/3236], Loss: 5.1500, Perplexity: 172.4289
Epoch [0/5], Step [30/3236], Loss: 4.9295, Perplexity: 138.3147
Epoch [0/5], Step [40/3236], Loss: 4.5292, Perplexity: 92.6851
Epoch [0/5], Step [50/3236], Loss: 4.3870, Perplexity: 80.3971
Epoch [0/5], Step [60/3236], Loss: 4.2046, Perplexity: 66.9942
Epoch [0/5], Step [70/3236], Loss: 4.0149, Perplexity: 55.4195
Epoch [0/5], Step [80/3236], Loss: 3.9087, Perplexity: 49.8341
Epoch [0/5], Step [90/3236], Loss: 3.8128, Perplexity: 45.2768
Epoch [0/5], Step [100/3236], Loss: 3.7193, Perplexity: 41.2363
Epoch [0/5], Step [110/3236], Loss: 3.8261, Perplexity: 45.8836
Epoch [0/5], Step [120/3236], Loss: 3.6833, Perplexity: 39.7769
Epoch [0/5], Step [130/3236], Loss: 3.4806, Perplexity: 32.4807
Epoch [0/5], Step [140/3236], Loss: 3.6516, Perplexity: 38.5349
Epoch [0/5], Step [150/3236], Loss: 3.6148, Perplexity: 37.1424
Epoch [0/5], Step [160/3236], Loss: 3.6043, Perplexity: 36.7555
Epoch [0/5], Step [170/3236], Loss: 3.4089, Perplexity: 30.2317
Epoch [0/5], Step [180/3236], Loss: 3.5103, Perplexity: 33.4576
Epoch [0/5], Step [190/3236], Loss: 3.4509, Perplexity: 31.5299
Epoch [0/5], Step [200/3236], Loss: 3.3716, Perplexity: 29.1259
```

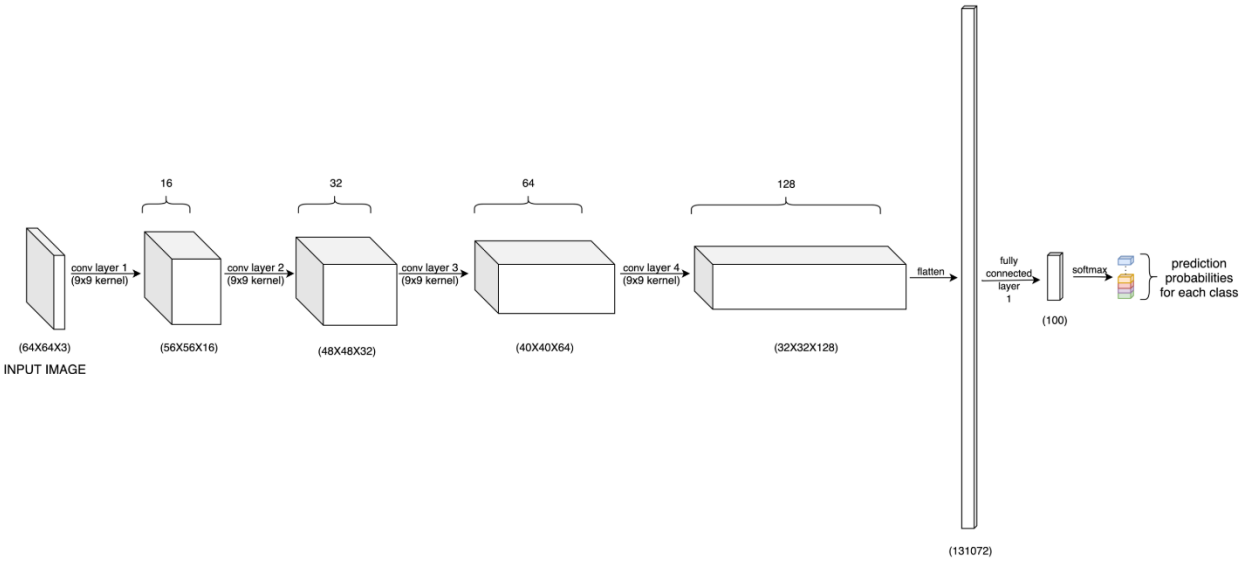


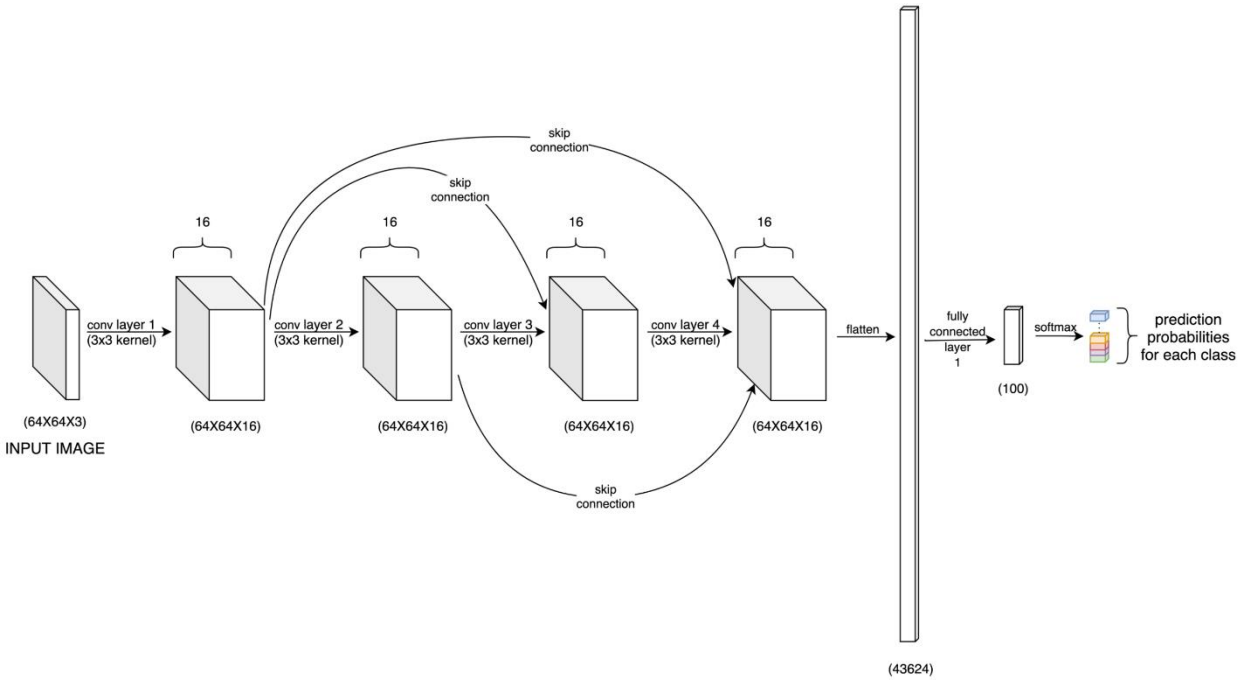
```
Epoch [1/5], Step [3100/3236], Loss: 1.9792, Perplexity: 7.2366
Epoch [1/5], Step [3110/3236], Loss: 2.0225, Perplexity: 7.5575
Epoch [1/5], Step [3120/3236], Loss: 1.9827, Perplexity: 7.2626
Epoch [1/5], Step [3130/3236], Loss: 2.1007, Perplexity: 8.1719
Epoch [1/5], Step [3140/3236], Loss: 2.0461, Perplexity: 7.7378
Epoch [1/5], Step [3150/3236], Loss: 2.1792, Perplexity: 8.8390
Epoch [1/5], Step [3160/3236], Loss: 2.0305, Perplexity: 7.6180

Epoch [1/5], Step [3170/3236], Loss: 2.0086, Perplexity: 7.4526
Epoch [1/5], Step [3180/3236], Loss: 2.0680, Perplexity: 7.9090
Epoch [1/5], Step [3190/3236], Loss: 2.1530, Perplexity: 8.6106
Epoch [1/5], Step [3200/3236], Loss: 1.9798, Perplexity: 7.2412
Epoch [1/5], Step [3210/3236], Loss: 2.0868, Perplexity: 8.0591
Epoch [1/5], Step [3220/3236], Loss: 2.0150, Perplexity: 7.5010
Epoch [1/5], Step [3230/3236], Loss: 2.0978, Perplexity: 8.1480
```

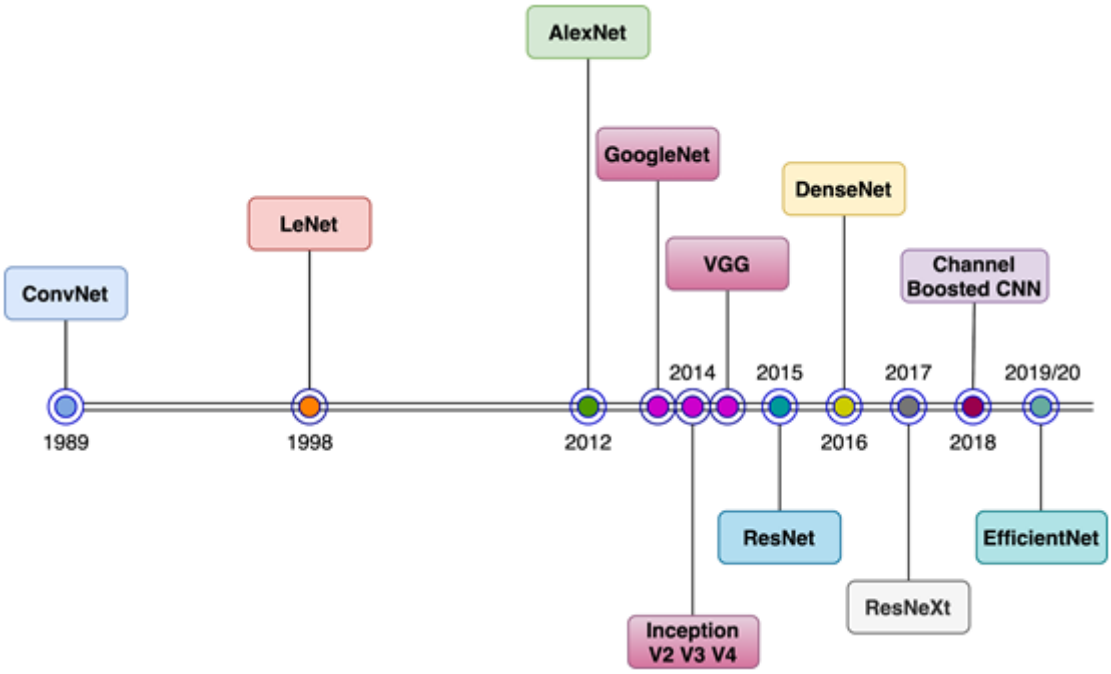
Chapter 3: Deep CNN Architectures

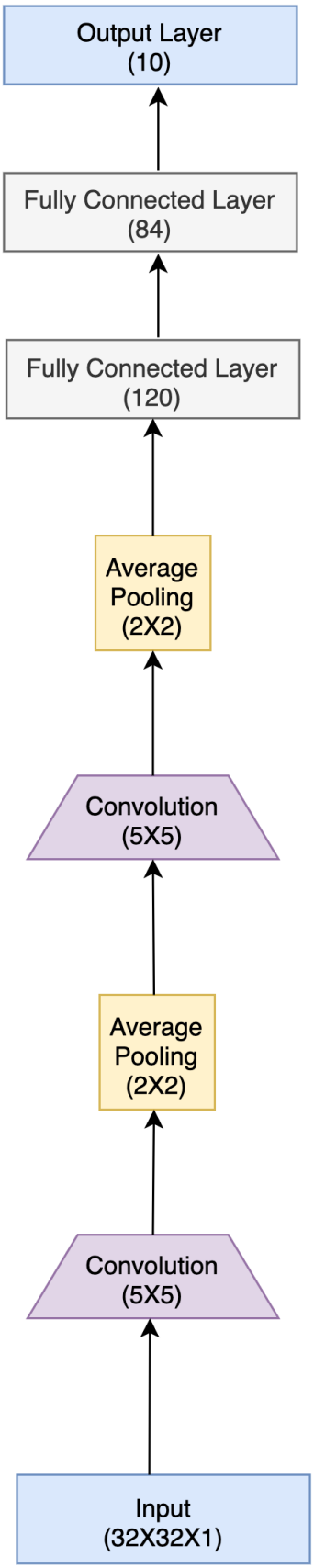





16
32
64
128
conv layer 1
(9x9 kernel)
conv layer 2
(9x9 kernel)
conv layer 3
(9x9 kernel)
conv layer 4
(9x9 kernel)
flatten
fully connected layer 1
softmax
prediction probabilities for each class
(64X64X3)
INPUT IMAGE
(56X56X16)
(48X48X32)
(40X40X64)
(32X32X128)
(131072)
(100)


skip connection
skip connection
16
16
16
16
conv layer 1
(3x3 kernel)
conv layer 2
(3x3 kernel)
conv layer 3
(3x3 kernel)
conv layer 4
(3x3 kernel)
flatten
fully connected layer 1
softmax
prediction probabilities for each class
(64X64X3)
INPUT IMAGE
(64X64X16)
(64X64X16)
(64X64X16)
(64X64X16)
skip connection
(43624)
(100)

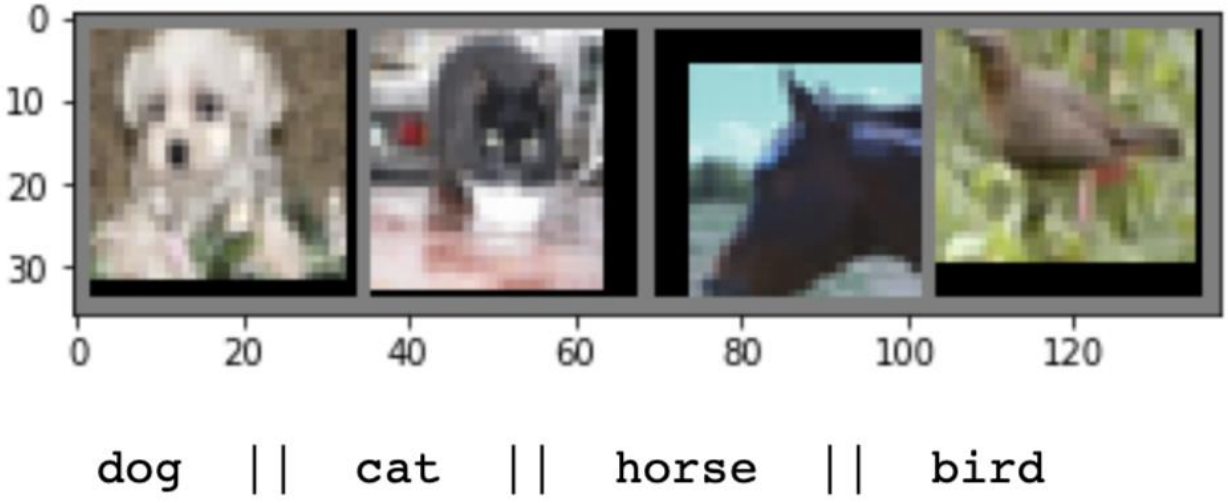

AlexNet
GoogleNet
DenseNet
LeNet
VGG
Channel Boosted CNN
ConvNet
2014
2015
2017
2019/20
1989
1998
2012
2016
2018
ResNet
EfficientNet
ResNeXt
Inception V2 V3 V4


Output Layer
(10)
Fully Connected Layer
(84)
Fully Connected Layer
(120)
Average
Pooling
(2X2)
Convolution
(5X5)
Average
Pooling
(2X2)
Convolution
(5X5)
Input
(32X32X1)

```
LeNet(
  (cn1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (cn2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
```

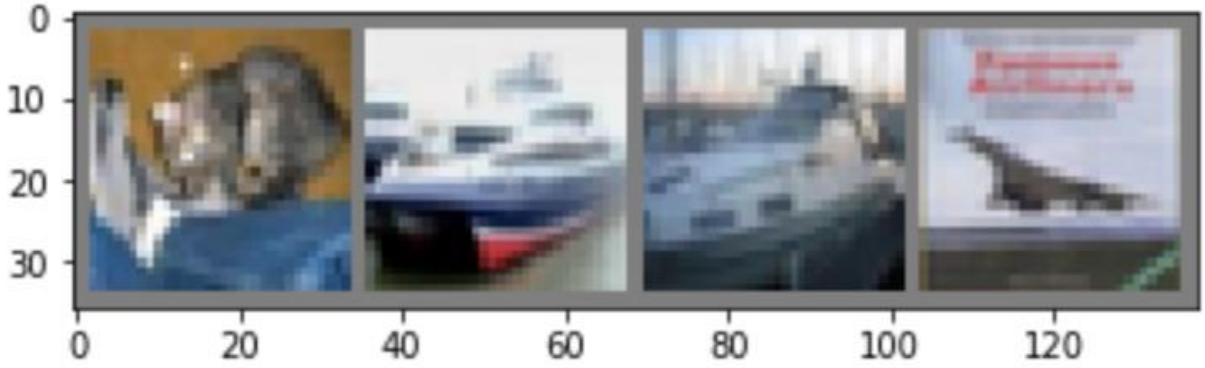
```
170500096/? [02:40<00:00, 934685.86it/s]

Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```



```
[Epoch number : 1, Mini-batches:  1000] loss: 9.901
[Epoch number : 1, Mini-batches:  2000] loss: 8.828
[Epoch number : 1, Mini-batches:  3000] loss: 8.350
[Epoch number : 1, Mini-batches:  4000] loss: 8.125
[Epoch number : 1, Mini-batches:  5000] loss: 7.935
[Epoch number : 1, Mini-batches:  6000] loss: 7.619

LeNet accuracy on 10000 images from test dataset: 48 %
```

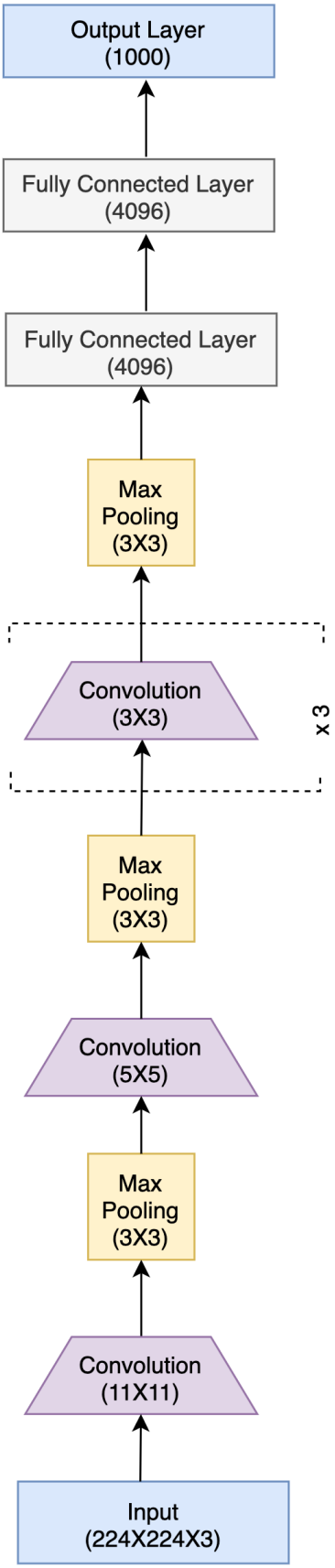
⋮

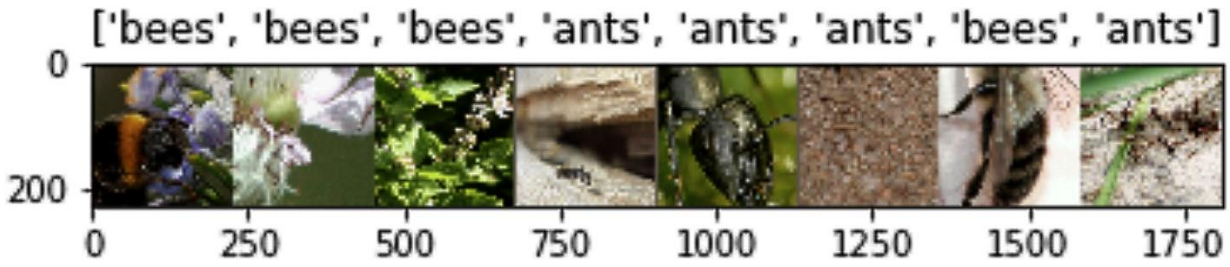
```
[Epoch number : 50, Mini-batches:  1000] loss: 5.027
[Epoch number : 50, Mini-batches:  2000] loss: 5.143
[Epoch number : 50, Mini-batches:  3000] loss: 5.079
[Epoch number : 50, Mini-batches:  4000] loss: 5.159
[Epoch number : 50, Mini-batches:  5000] loss: 5.065
[Epoch number : 50, Mini-batches:  6000] loss: 4.977

LeNet accuracy on 10000 images from test dataset: 67 %

Finished Training
```




```
Label:          cat   ship   ship plane
Prediction:     cat   ship   ship plane
Model accuracy on 10000 images from test dataset: 67 %
```

```
Model accuracy for class plane : 68 %
Model accuracy for class   car : 87 %
Model accuracy for class  bird : 57 %
Model accuracy for class   cat : 56 %
Model accuracy for class  deer : 59 %
Model accuracy for class   dog : 39 %
Model accuracy for class  frog : 83 %
Model accuracy for class horse : 62 %
Model accuracy for class  ship : 82 %
Model accuracy for class truck : 75 %
```


Output Layer
(1000)
Fully Connected Layer
(4096)
Fully Connected Layer
(4096)
Max
Pooling
(3X3)
Convolution
(3X3)
x 3
Max
Pooling
(3X3)
Convolution
(5X5)
Max
Pooling
(3X3)
Convolution
(11X11)
Input
(224X224X3)




```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
  (1): ReLU(inplace=True)
  (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (4): ReLU(inplace=True)
  (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU(inplace=True)
  (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU(inplace=True)
  (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

```
Sequential(
  (0): Dropout(p=0.5, inplace=False)
  (1): Linear(in_features=9216, out_features=4096, bias=True)
  (2): ReLU(inplace=True)
  (3): Dropout(p=0.5, inplace=False)
  (4): Linear(in_features=4096, out_features=4096, bias=True)
  (5): ReLU(inplace=True)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

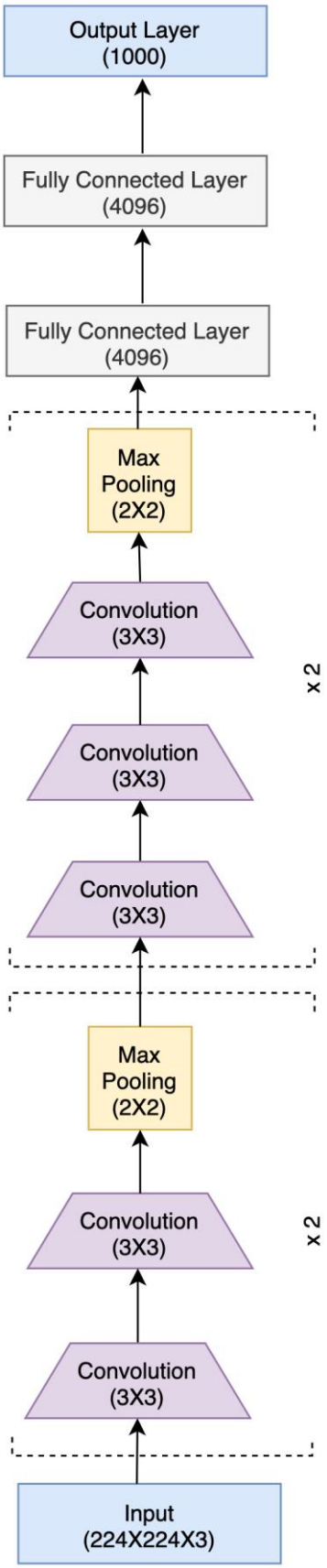
```
Epoch number 0/9
====================
train loss in this epoch: 0.7761217306871884, accuracy in this epoch: 0.4959016393442623
val loss in this epoch: 0.6042805251732372, accuracy in this epoch: 0.6666666666666666

Epoch number 1/9
====================
train loss in this epoch: 0.5759895355975042, accuracy in this epoch: 0.6639344262295082
val loss in this epoch: 0.4689261562684003, accuracy in this epoch: 0.7908496732026143

Epoch number 2/9
====================
train loss in this epoch: 0.5033335646644967, accuracy in this epoch: 0.75
val loss in this epoch: 0.3966531710687026, accuracy in this epoch: 0.8431372549019608

                                        .
                                        .
                                        .
                                        .

Epoch number 8/9
====================
train loss in this epoch: 0.3300624494669867, accuracy in this epoch: 0.860655737704918
val loss in this epoch: 0.27101927756764044, accuracy in this epoch: 0.934640522875817

Epoch number 9/9
====================
train loss in this epoch: 0.3026028309689193, accuracy in this epoch: 0.8729508196721312
val loss in this epoch: 0.2609025729710565, accuracy in this epoch: 0.9215686274509803

Training finished in 4.0mins 30.213629007339478secs
Best validation set accuracy: 0.934640522875817
```

pred: bees || target: bees

pred: ants || target: ants

pred: ants || target: ants

pred: ants || target: ants


Output Layer
(1000)
Fully Connected Layer
(4096)
Fully Connected Layer
(4096)
Max
Pooling
(2X2)
Convolution
(3X3)
Convolution
(3X3)
Convolution
(3X3)
x 2
Max
Pooling
(2X2)
Convolution
(3X3)
Convolution
(3X3)
x 2
Input
(224X224X3)

```
{0: 'tench, Tinca tinca', 1: 'goldfish, Carassius auratus', 2: 'great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias', 3: 'tiger shark, Galeocerdo cuvieri', 4: 'hammerhead, hammerhead shark'}
```

```
Downloading: "https://download.pytorch.org/models/vgg13-c768596a.pth" to /Users/ashish.jha/.cache/torch/checkpoints/vgg13-c768596a.pth
```

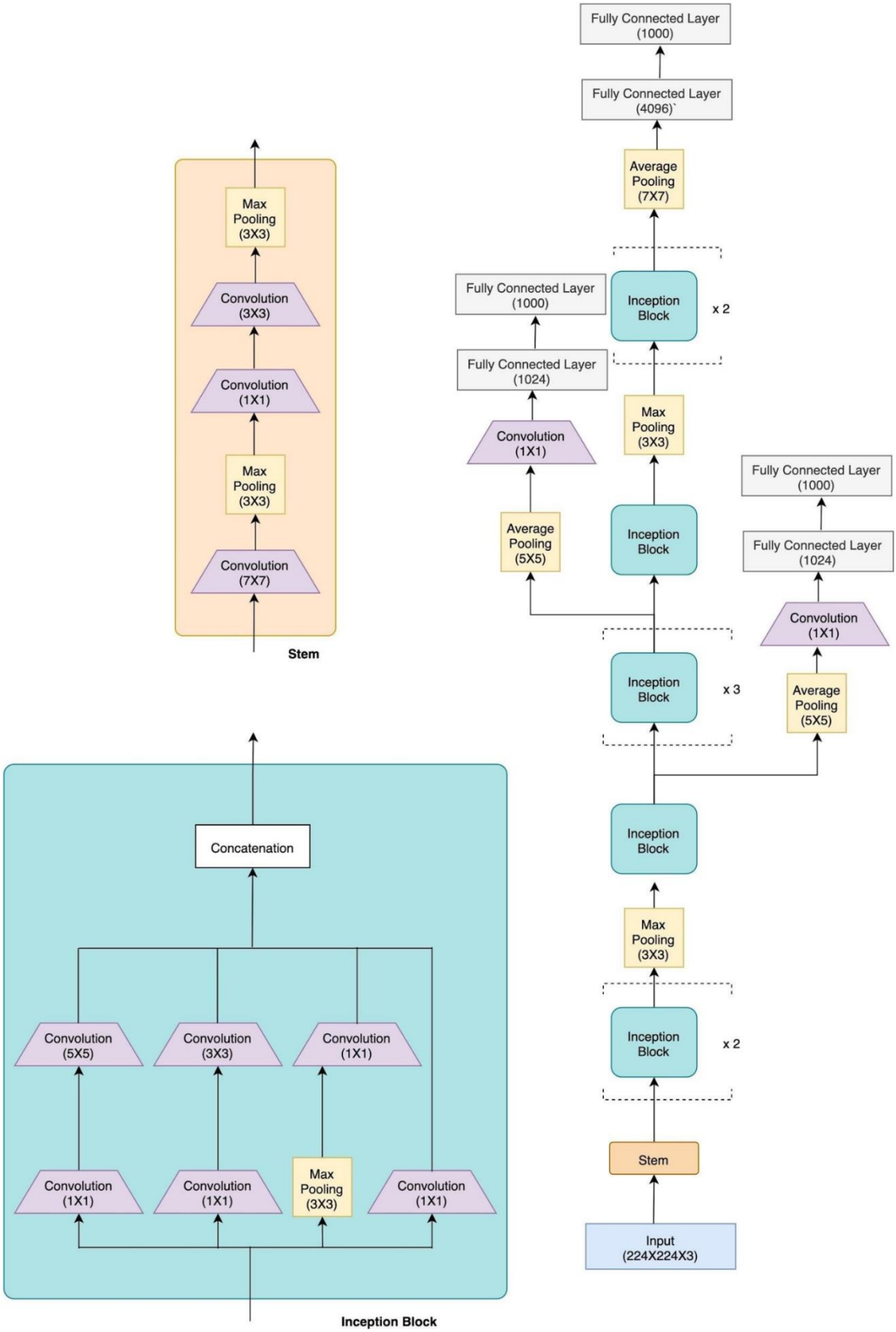
100% 508M/508M [21:36<00:00, 411kB/s]

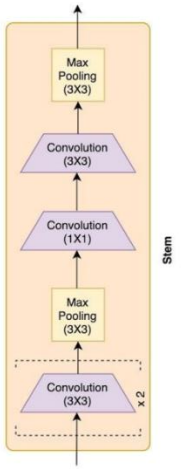
pred: bee

pred: ant, emmet, pismire

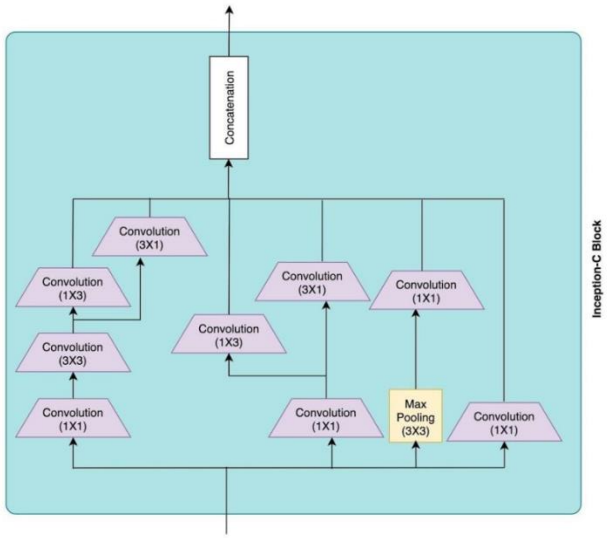
pred: ant, emmet, pismire

pred: ant, emmet, pismire

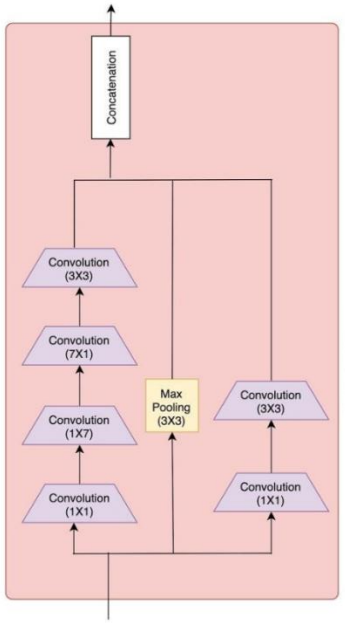

Fully Connected Layer (1000)
Fully Connected Layer (4096)`
Average Pooling (7X7)
Inception Block
x 2
Fully Connected Layer (1000)
Fully Connected Layer (1024)
Convolution (1X1)
Max Pooling (3X3)
Average Pooling (5X5)
Inception Block
Fully Connected Layer (1000)
Fully Connected Layer (1024)
Convolution (1X1)
Average Pooling (5X5)
Inception Block
x 3
Inception Block
Max Pooling (3X3)
Inception Block
x 2
Stem
Input (224X224X3)
Max Pooling (3X3)
Convolution (3X3)
Convolution (1X1)
Max Pooling (3X3)
Convolution (7X7)
Stem
Concatenation
Convolution (5X5)
Convolution (3X3)
Convolution (1X1)
Convolution (1X1)
Convolution (1X1)
Max Pooling (3X3)
Convolution (1X1)
Inception Block



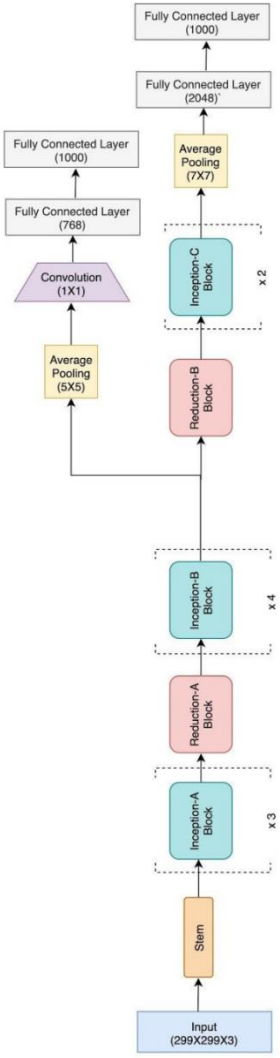
Stem

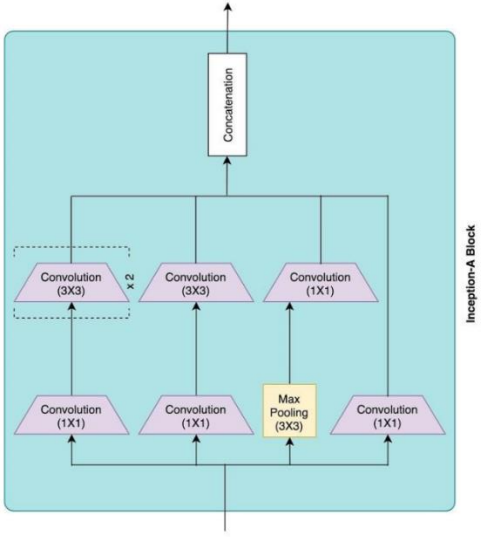


Inception-C Block

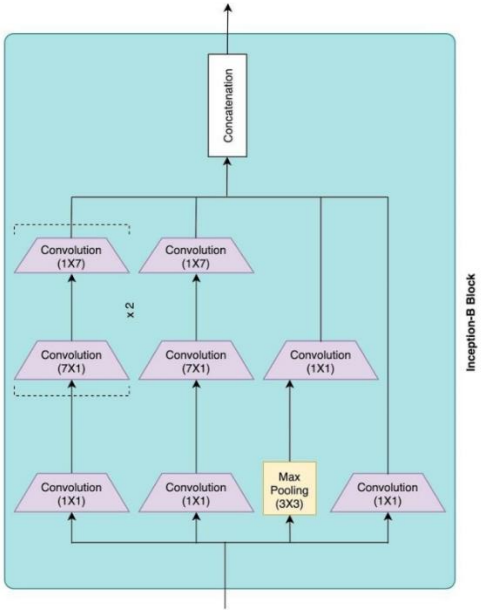


Reduction-B Block

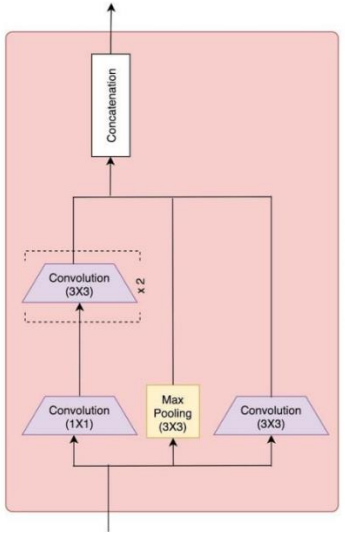




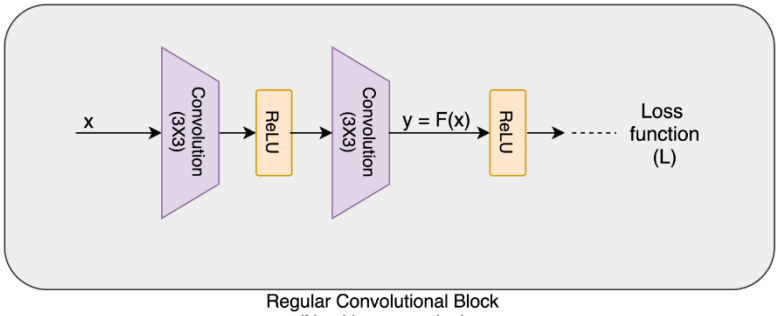
Inception-A Block



Inception-B Block

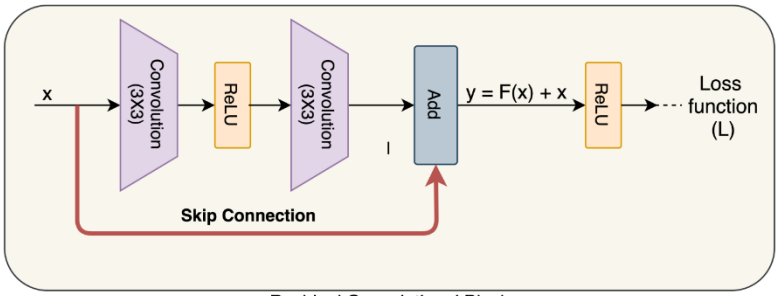


Reduction-A Block



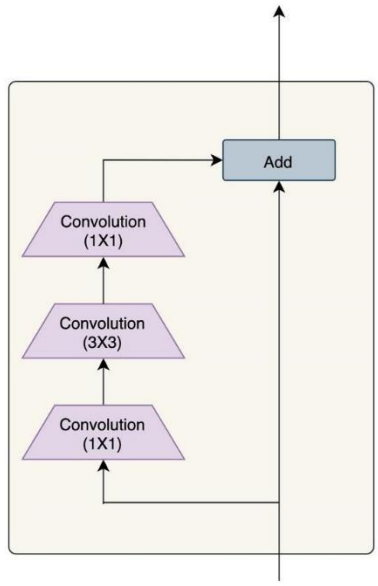
Regular Convolutional Block
(No skip connection)

$$\frac{\delta L}{\delta x} = \frac{\delta L}{\delta y} * \frac{\delta y}{\delta x}$$
$$= \frac{\delta L}{\delta y} * F'(x)$$

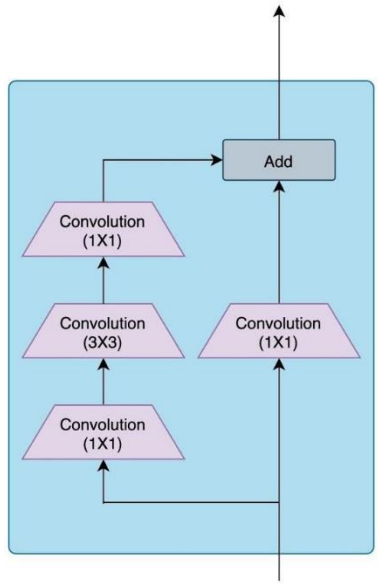


Residual Convolutional Block
(Skip connection)

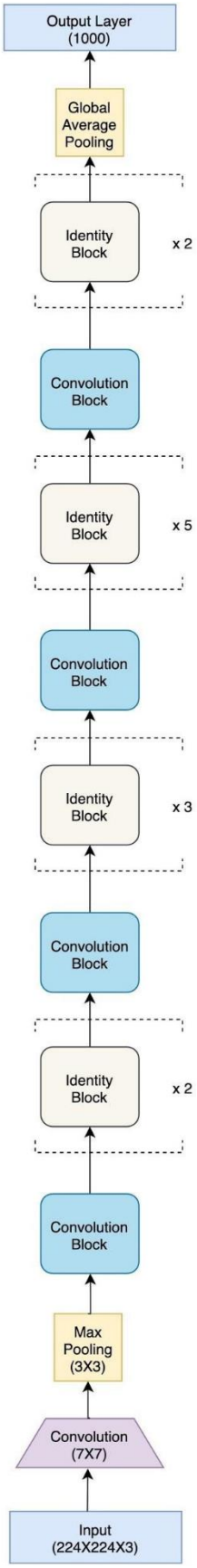
$$\frac{\delta L}{\delta x} = \frac{\delta L}{\delta y} * \frac{\delta y}{\delta x}$$
$$= \frac{\delta L}{\delta y} * F'(x) + \overbrace{\frac{\delta L}{\delta y}}^{\text{helps stabilize gradients}}$$

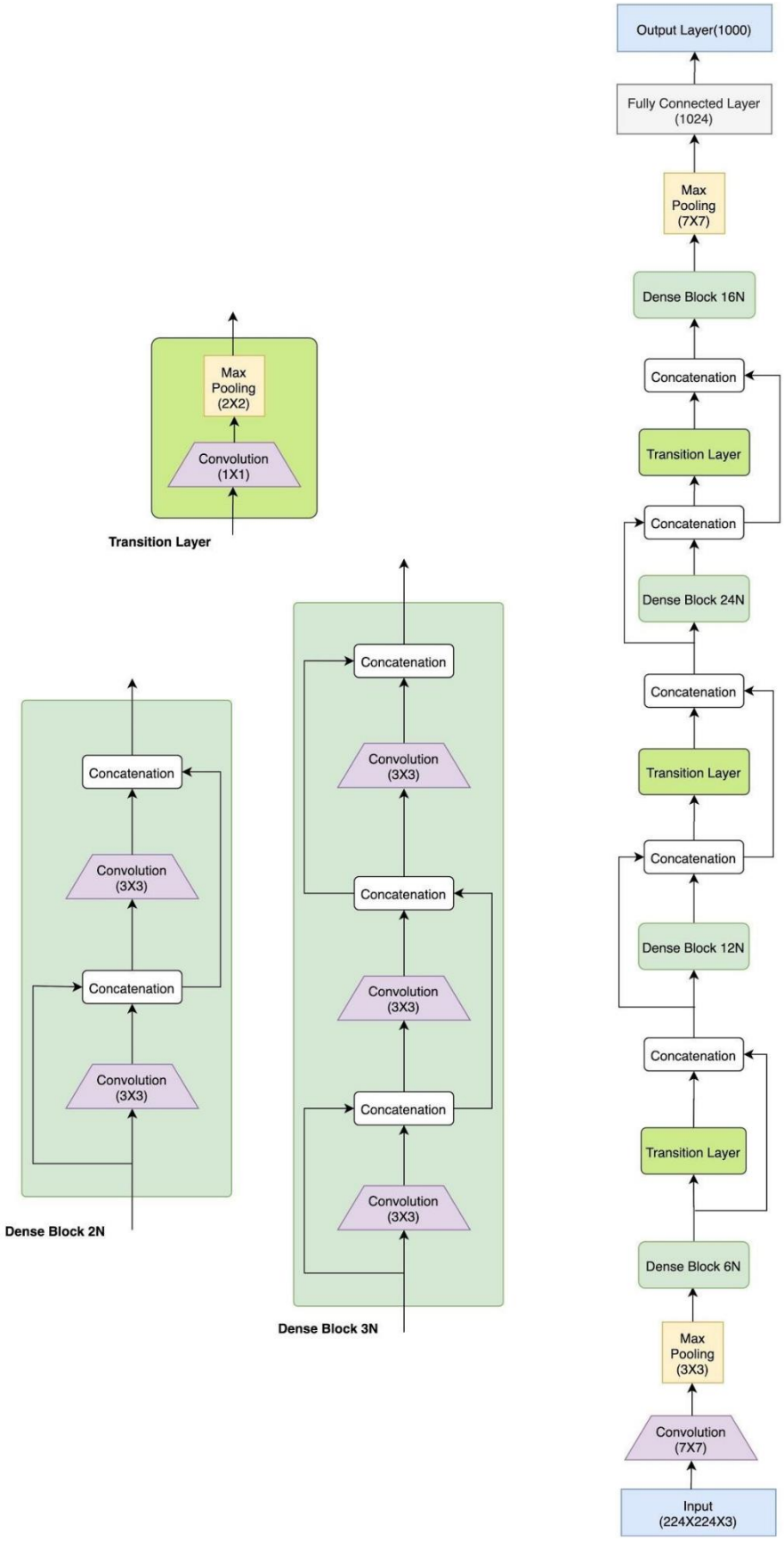


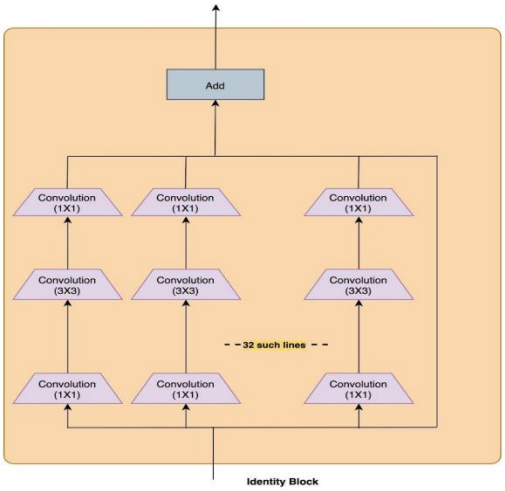
Identity Block



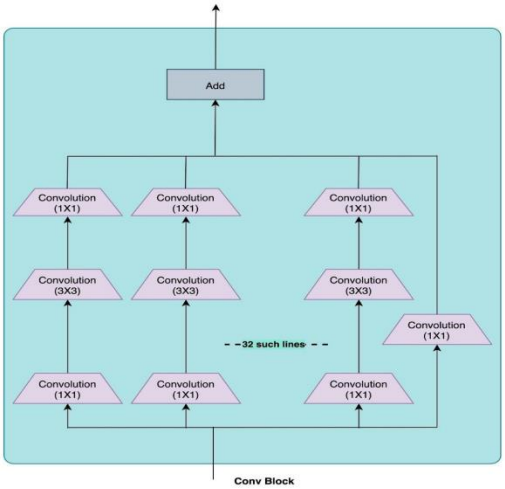
Convolution Block



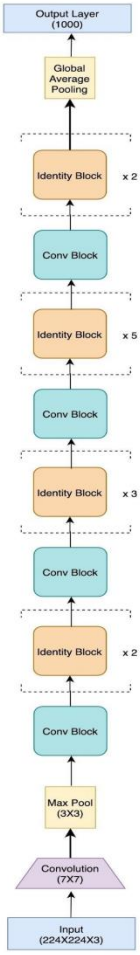

Output Layer(1000)
Fully Connected Layer (1024)
Max Pooling (7X7)
Dense Block 16N
Concatenation
Transition Layer
Concatenation
Dense Block 24N
Concatenation
Transition Layer
Concatenation
Dense Block 12N
Concatenation
Transition Layer
Dense Block 6N
Max Pooling (3X3)
Convolution (7X7)
Input (224X224X3)
Max Pooling (2X2)
Convolution (1X1)
Transition Layer
Concatenation
Convolution (3X3)
Concatenation
Convolution (3X3)
Concatenation
Convolution (3X3)
Dense Block 2N
Concatenation
Convolution (3X3)
Concatenation
Convolution (3X3)
Concatenation
Convolution (3X3)
Dense Block 3N



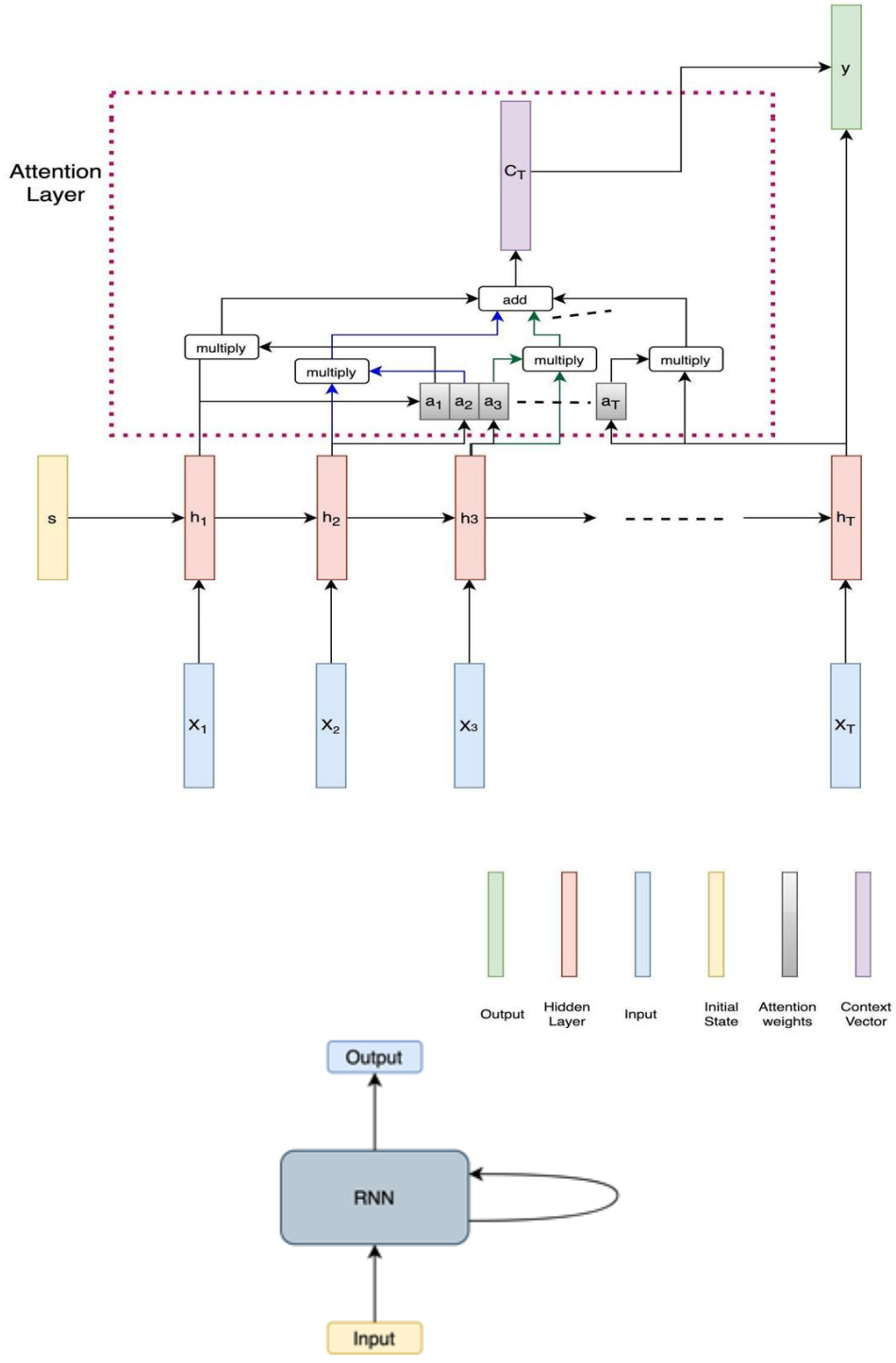
Identity Block

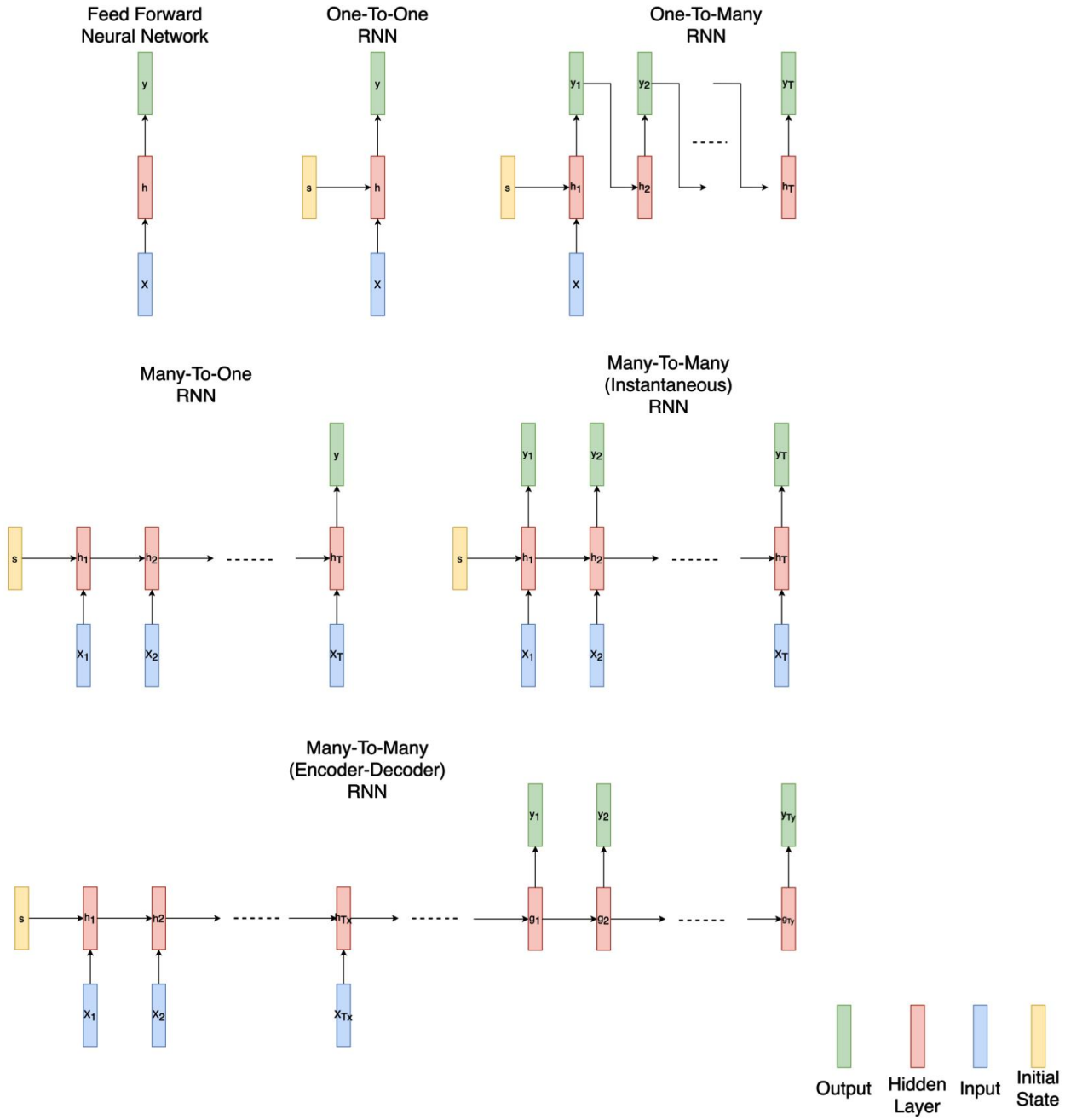


Conv Block

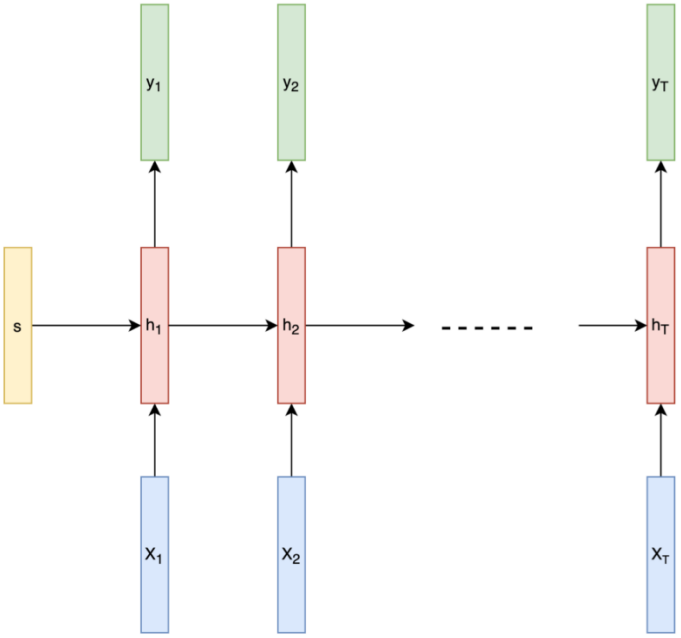


Chapter 4: Deep Recurrent Model Architectures

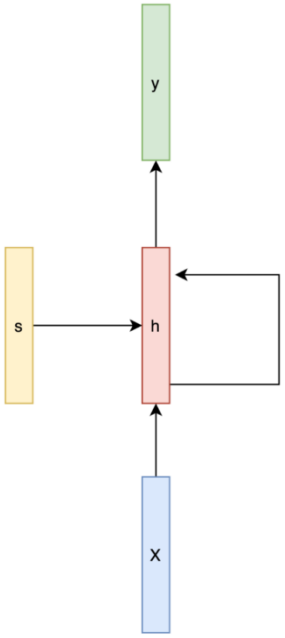


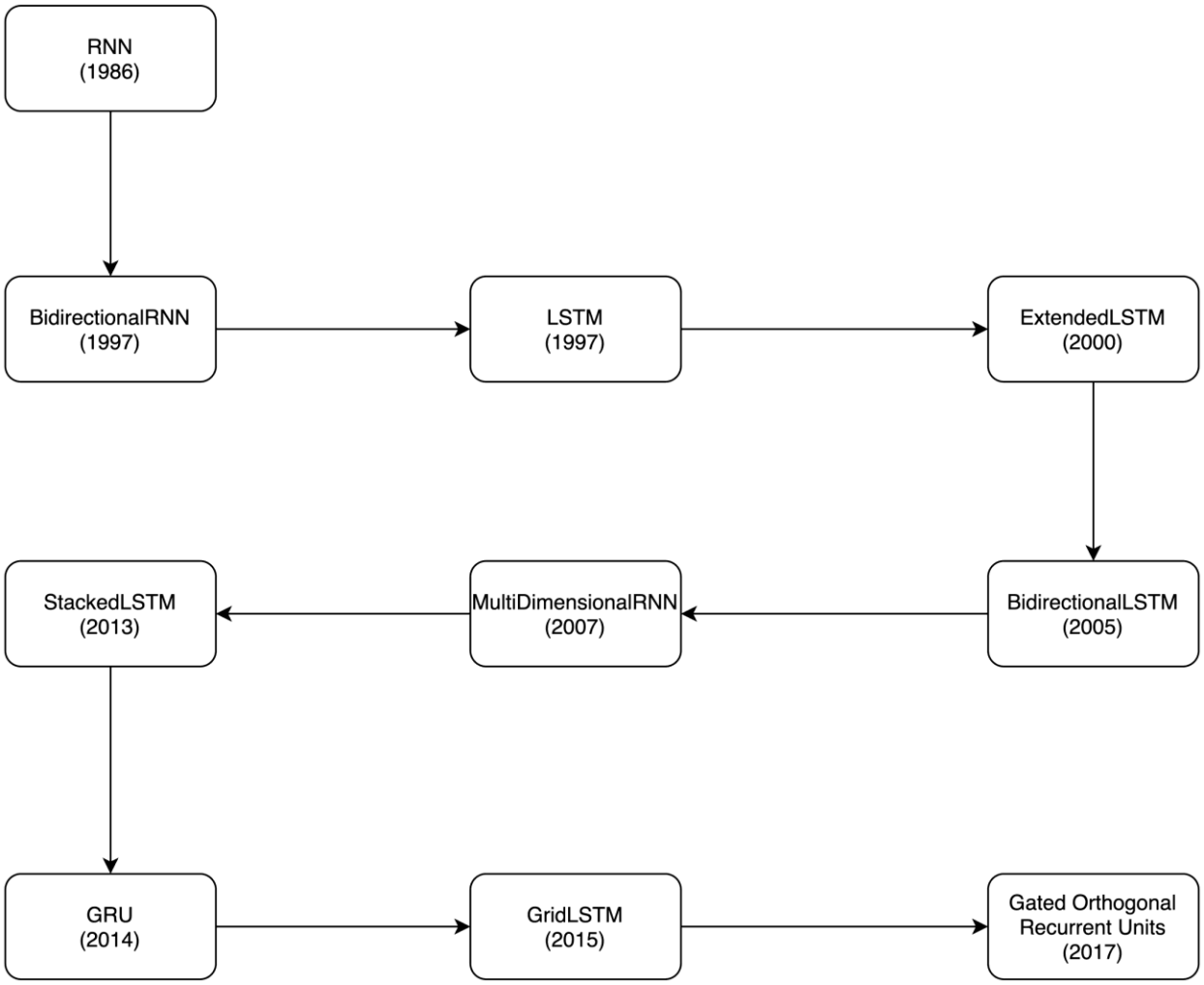

Feed Forward Neural Network
One-To-One RNN
One-To-Many RNN
Many-To-One RNN
Many-To-Many (Instantaneous) RNN
Many-To-Many (Encoder-Decoder) RNN
Output
Hidden Layer
Input
Initial State

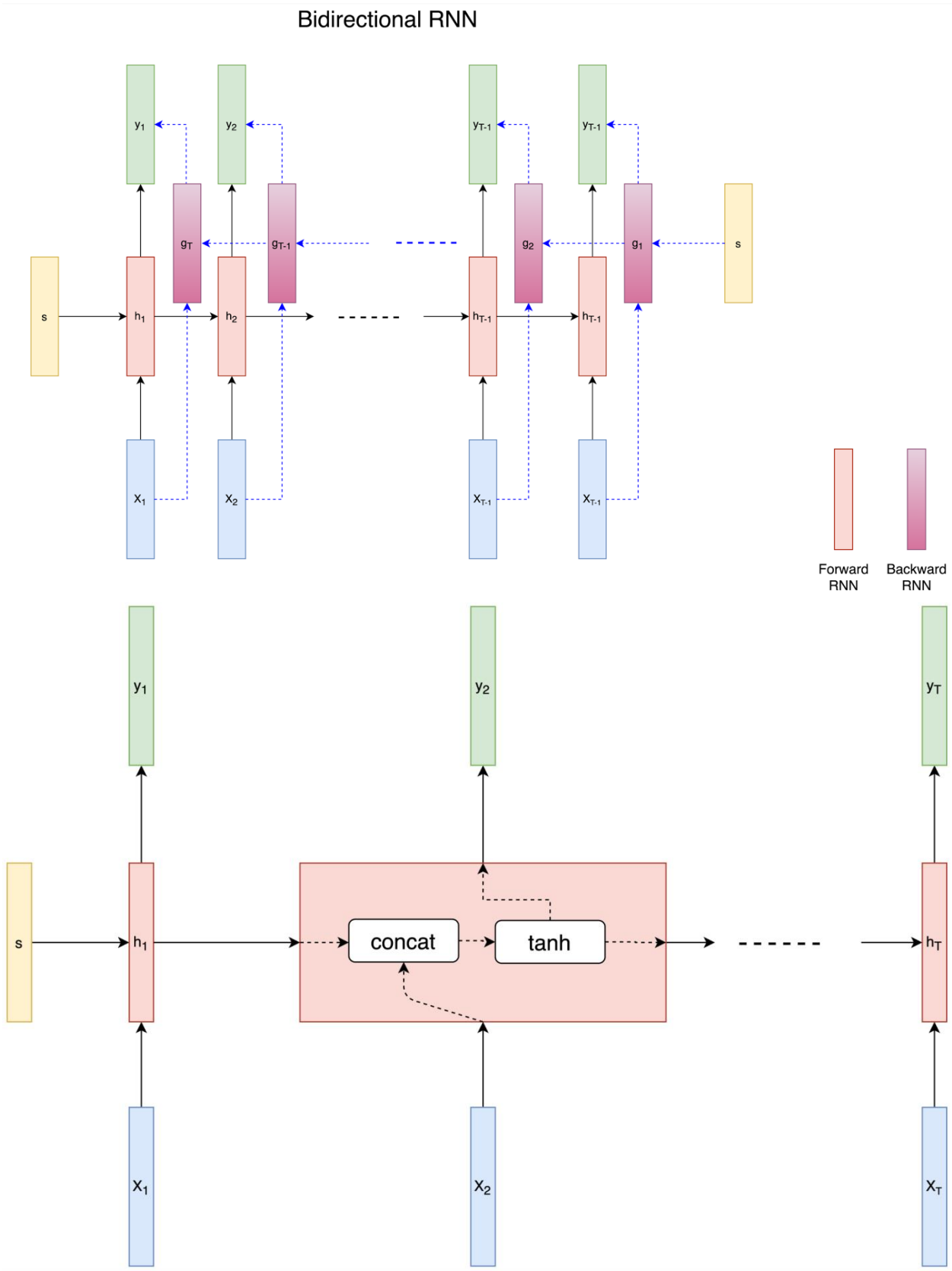
RNN (Time-Unfolded)

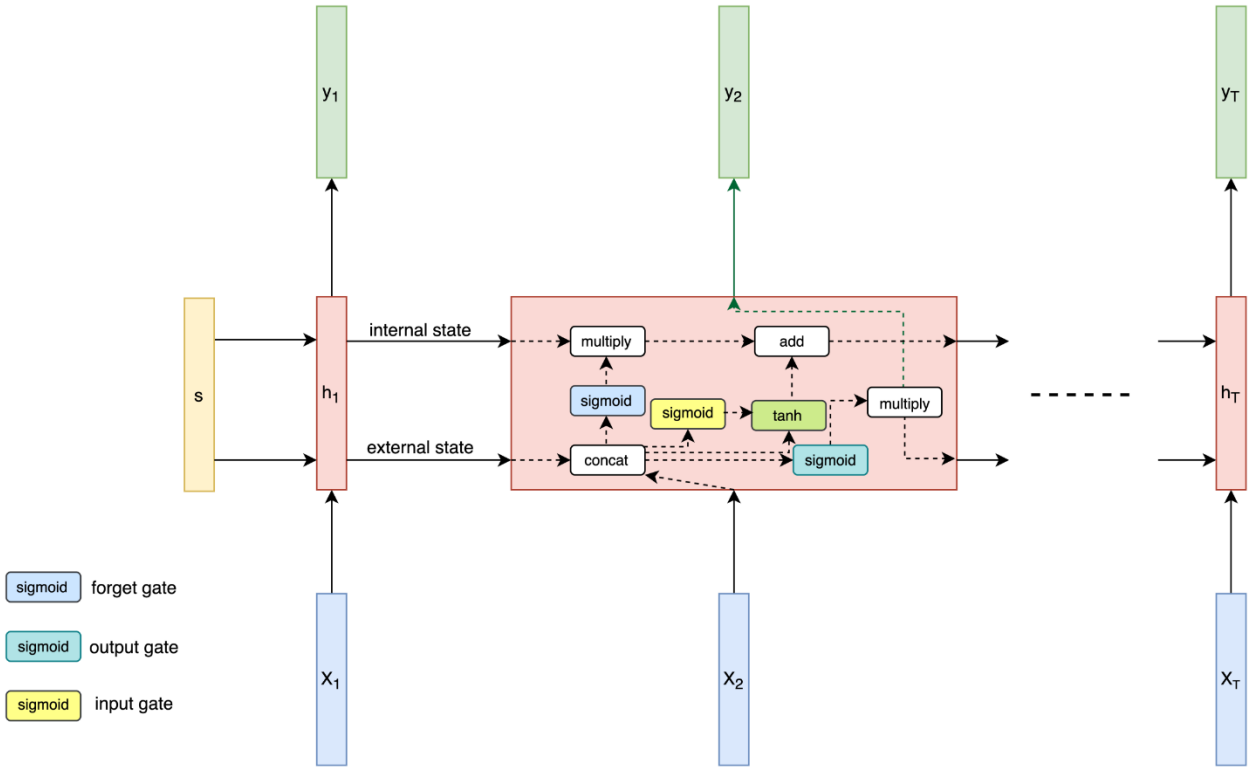


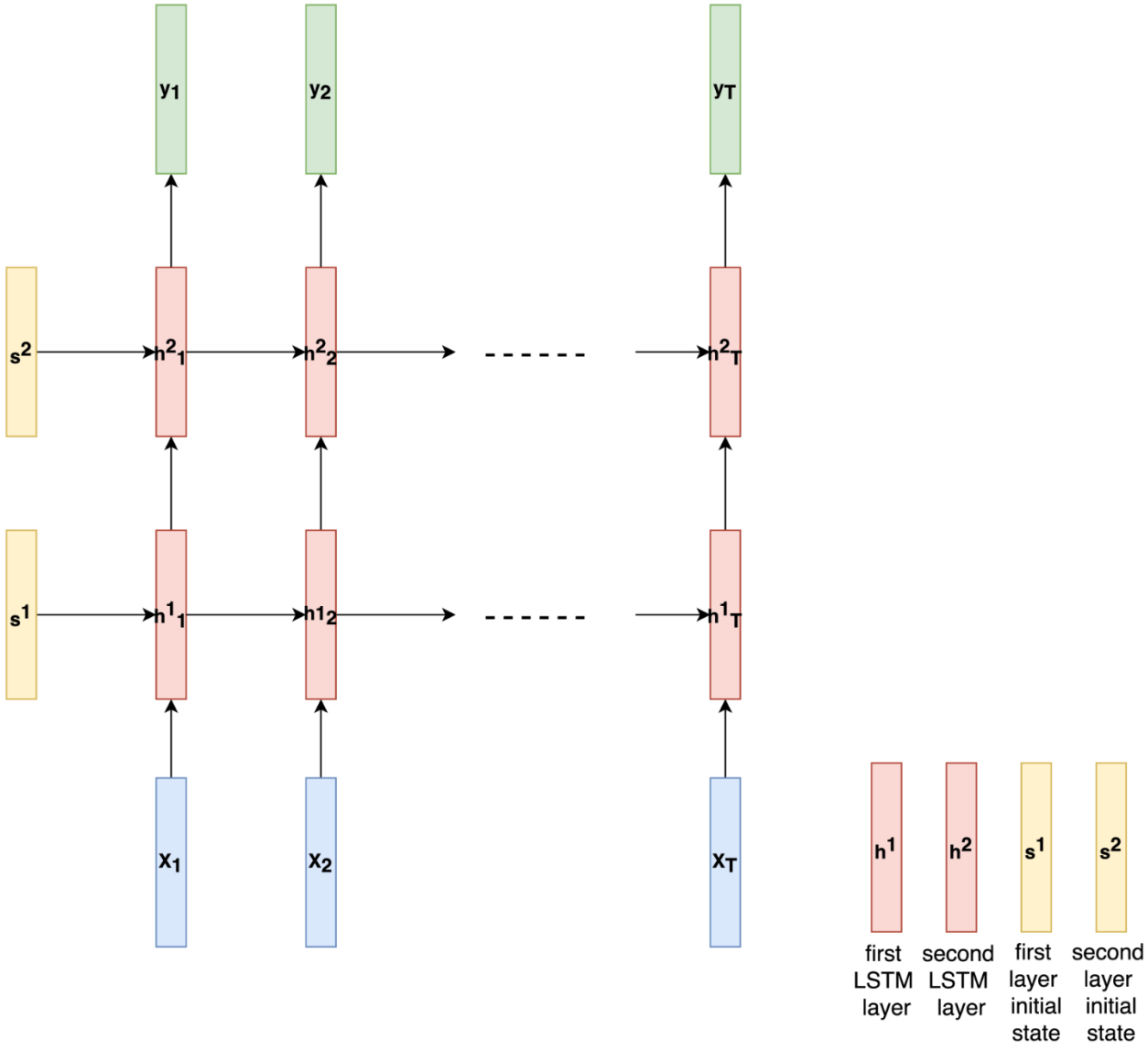
RNN (Time-Folded)

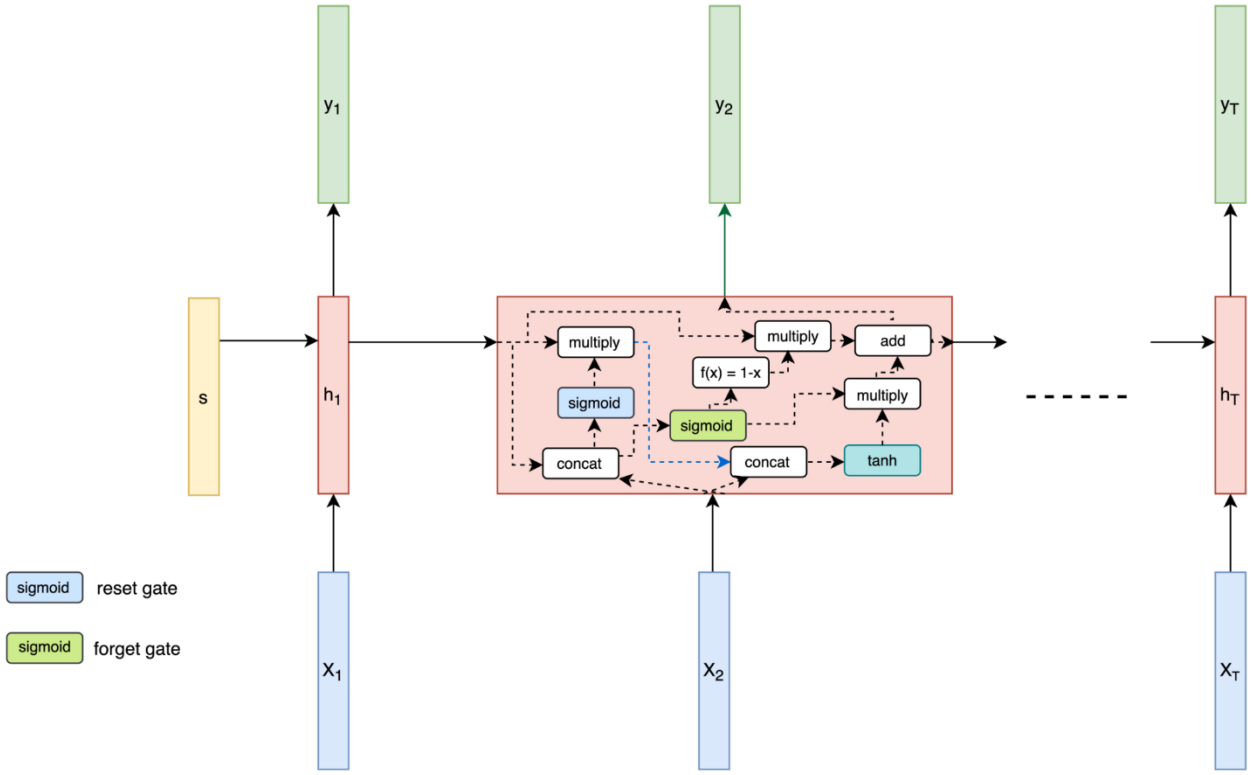



RNN
(1986)
BidirectionalRNN
(1997)
LSTM
(1997)
ExtendedLSTM
(2000)
BidirectionalLSTM
(2005)
MultiDimensionalRNN
(2007)
StackedLSTM
(2013)
GRU
(2014)
GridLSTM
(2015)
Gated Orthogonal
Recurrent Units
(2017)


Bidirectional RNN
y1
y2
yT-1
yT-1
gT
gT-1
g2
g1
s
s
h1
h2
hT-1
hT-1
x1
x2
xT-1
xT-1
Forward RNN
Backward RNN
y1
y2
yT
s
h1
concat
tanh
hT
x1
x2
xT


y1
y2
yT
s
h1
internal state
external state
multiply
add
sigmoid
sigmoid
tanh
multiply
concat
sigmoid
hT
X1
X2
XT
sigmoid forget gate
sigmoid output gate
sigmoid input gate


y1
y2
yT
s2
h2 1
h2 2
h2 T
s1
h1 1
h1 2
h1 T
x1
x2
xT
h1
h2
s1
s2
first LSTM layer
second LSTM layer
first layer initial state
second layer initial state




```
100%|██████████| 12500/12500 [00:03<00:00, 3393.39it/s]
100%|██████████| 12500/12500 [00:03<00:00, 3707.12it/s]
```

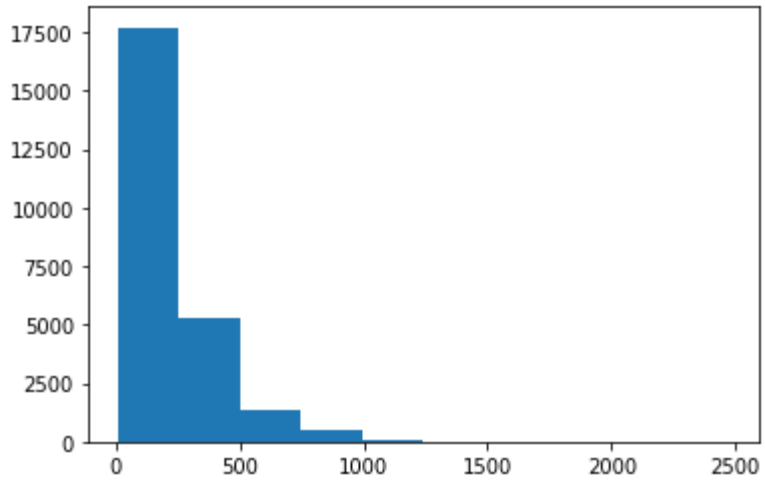
```
Number of reviews : 25000
```

```
[('the', 334691), ('and', 162228), ('a', 161940), ('of', 145326), ('to',
135042), ('is', 106855), ('in', 93028), ('it', 77099), ('i', 75719), ('th
is', 75190)]
```

```
[('the', 1), ('and', 2), ('a', 3), ('of', 4), ('to', 5), ('is', 6),
('in', 7), ('it', 8), ('i', 9), ('this', 10)]
```

```
for a movie that gets no respect there sure are a lot of memorable quotes listed for this gem
imagine a movie where joe piscopo is actually funny maureen stapleton is a scene stealer the
moroni character is an absolute scream watch for alan the skipper hale jr as a police sgt
```

```
[15, 3, 17, 11, 201, 56, 1165, 47, 242, 23, 3, 168, 4, 891, 4325, 3513, 15, 10, 1514, 822, 3,
17, 112, 884, 14623, 6, 155, 161, 7307, 15816, 6, 3, 134, 20049, 1, 32064, 108, 6, 33, 1492,
1943, 103, 15, 1550, 1, 18993, 9055, 1809, 14, 3, 549, 6906]
```




```
Example Input size:  torch.Size([32, 512])
Example Input:
 tensor([[    0,     0,     0,  ...,     1,   875,   520],
        [    0,     0,     0,  ...,   482,   800,  1794],
        [    0,     0,     0,  ...,     3,  1285, 70251],
        ...,
        [    0,     0,     0,  ...,     4,     1,  1374],
        [    0,     0,     0,  ...,     2,  8268, 17117],
        [    0,     0,     0,  ...,  6429,   271,   116]])

Example Output size:  torch.Size([32])
Example Output:
 tensor([1., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 0., 0., 1., 1., 1.,
        1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.])
```

```
epoch number: 1 | time elapsed: 136.13723397254944s
training loss: 0.627 | training accuracy: 66.23%
validation loss: 1.048 |  validation accuracy: 19.65%

epoch number: 2 | time elapsed: 150.36637210845947s
training loss: 0.533 | training accuracy: 73.80%
validation loss: 0.858 |  validation accuracy: 54.43%

epoch number: 3 | time elapsed: 186.54570603370667s
training loss: 0.438 | training accuracy: 80.39%
validation loss: 0.551 |  validation accuracy: 78.56%

                        :
                        :
                        :
                        :

epoch number: 8 | time elapsed: 199.3309519290924s
training loss: 0.198 | training accuracy: 92.92%
validation loss: 0.971 |  validation accuracy: 66.19%

epoch number: 9 | time elapsed: 185.76586294174194s
training loss: 0.315 | training accuracy: 87.93%
validation loss: 0.950 |  validation accuracy: 62.34%

epoch number: 10 | time elapsed: 188.91670608520508s
training loss: 0.193 | training accuracy: 93.08%
validation loss: 1.042 |  validation accuracy: 62.71%
```

```
0.05216024070978165
0.17682921886444092
0.7510029077529907
0.9689022898674011
0.9829260110855103
```

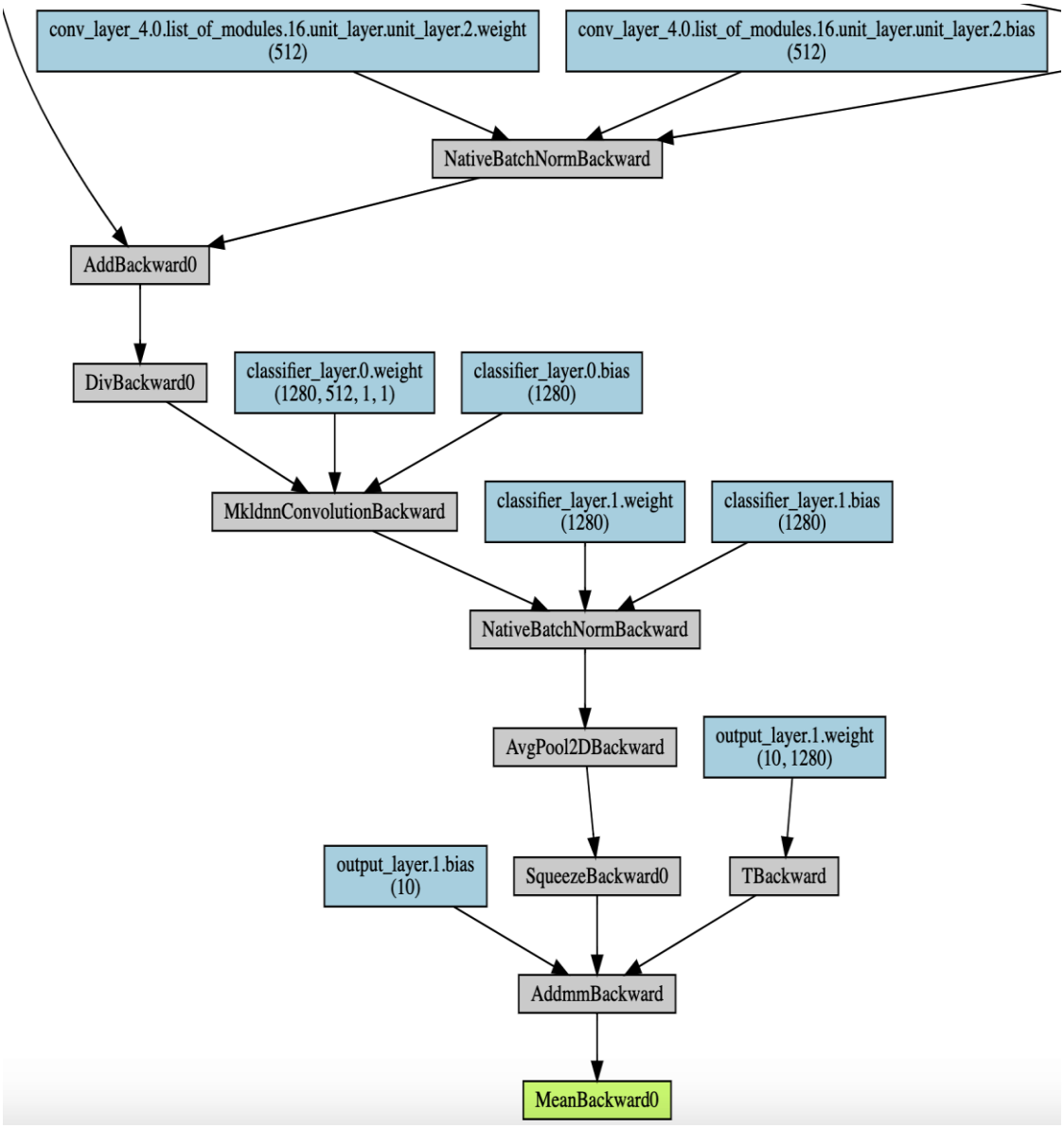
```
epoch number: 1 | time elapsed: 1212.3228149414062s
training loss: 0.686 | training accuracy: 54.57%
validation loss: 0.666 |  validation accuracy: 60.02%

epoch number: 2 | time elapsed: 1138.5317480564117s
training loss: 0.650 | training accuracy: 61.54%
validation loss: 0.607 |  validation accuracy: 68.02%

epoch number: 3 | time elapsed: 1141.8038160800934s
training loss: 0.579 | training accuracy: 69.60%
validation loss: 0.654 |  validation accuracy: 67.09%
```

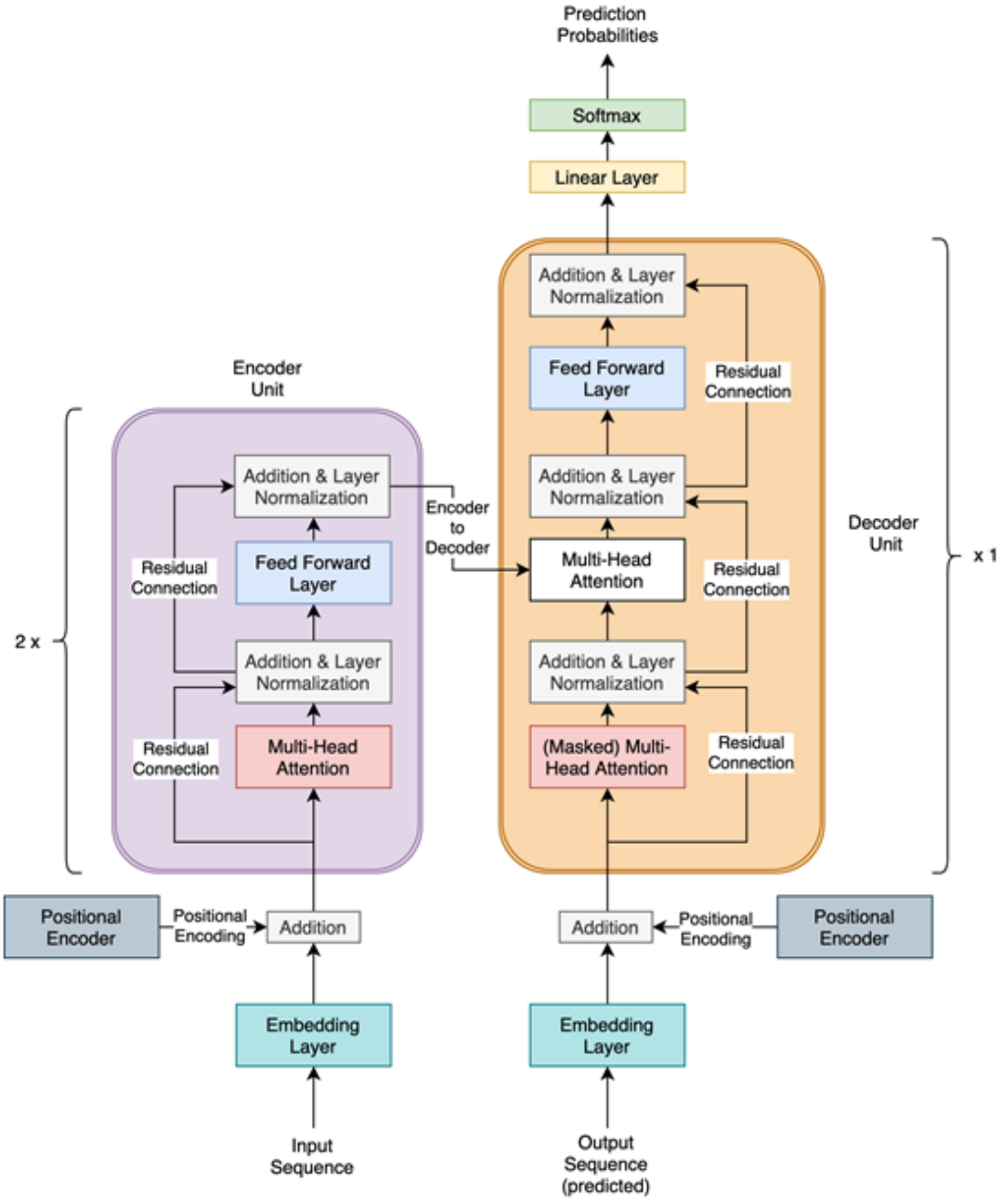


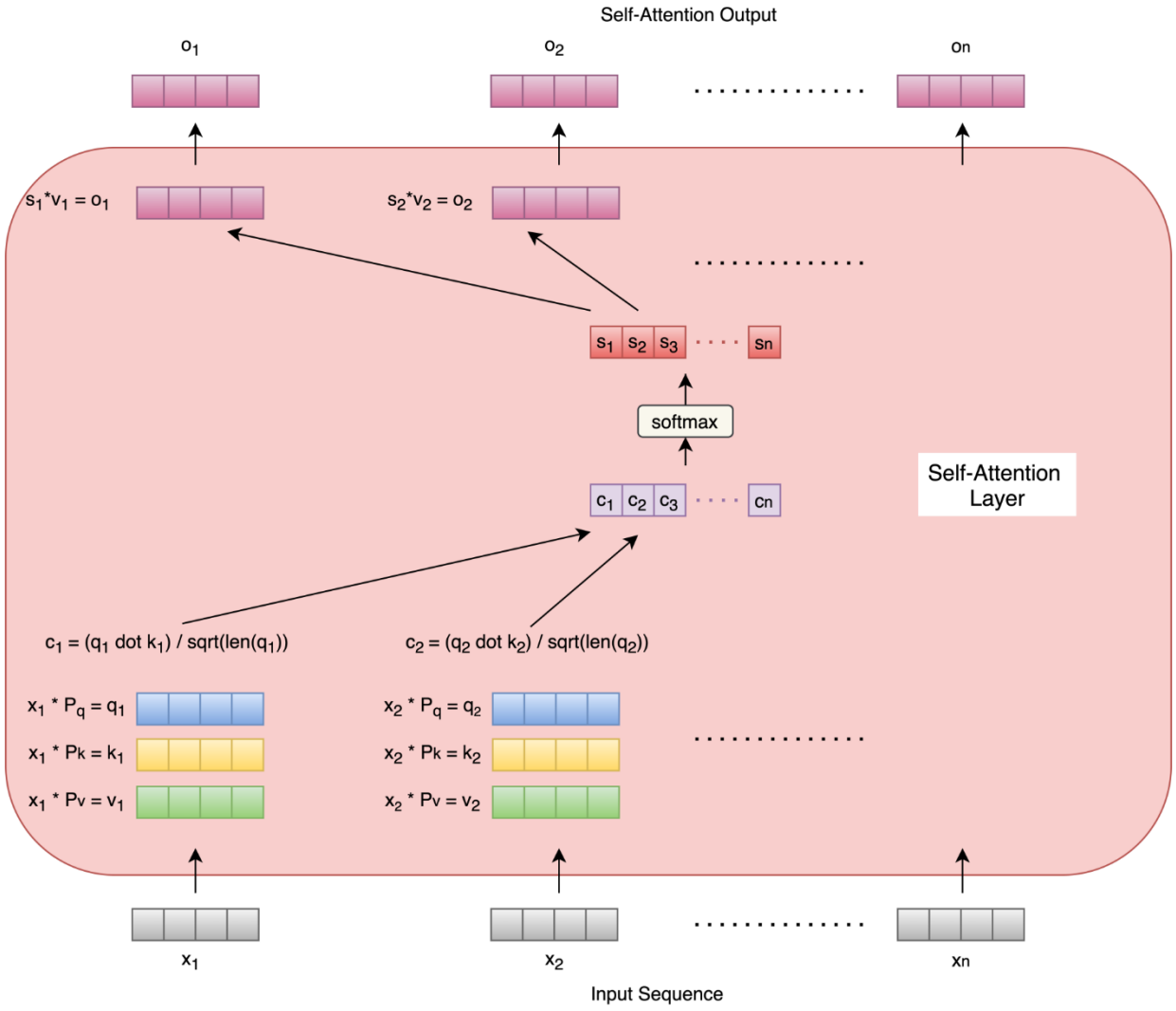
```
epoch number: 8 | time elapsed: 1066.7158658504486s
training loss: 0.383 | training accuracy: 83.04%
validation loss: 0.653 |  validation accuracy: 74.60%

epoch number: 9 | time elapsed: 1046.7357511520386s
training loss: 0.389 | training accuracy: 83.21%
validation loss: 0.586 |  validation accuracy: 75.98%

epoch number: 10 | time elapsed: 1029.34814786911s
training loss: 0.351 | training accuracy: 84.87%
validation loss: 0.549 |  validation accuracy: 77.66%

test loss: 0.585 | test accuracy: 76.19%
```

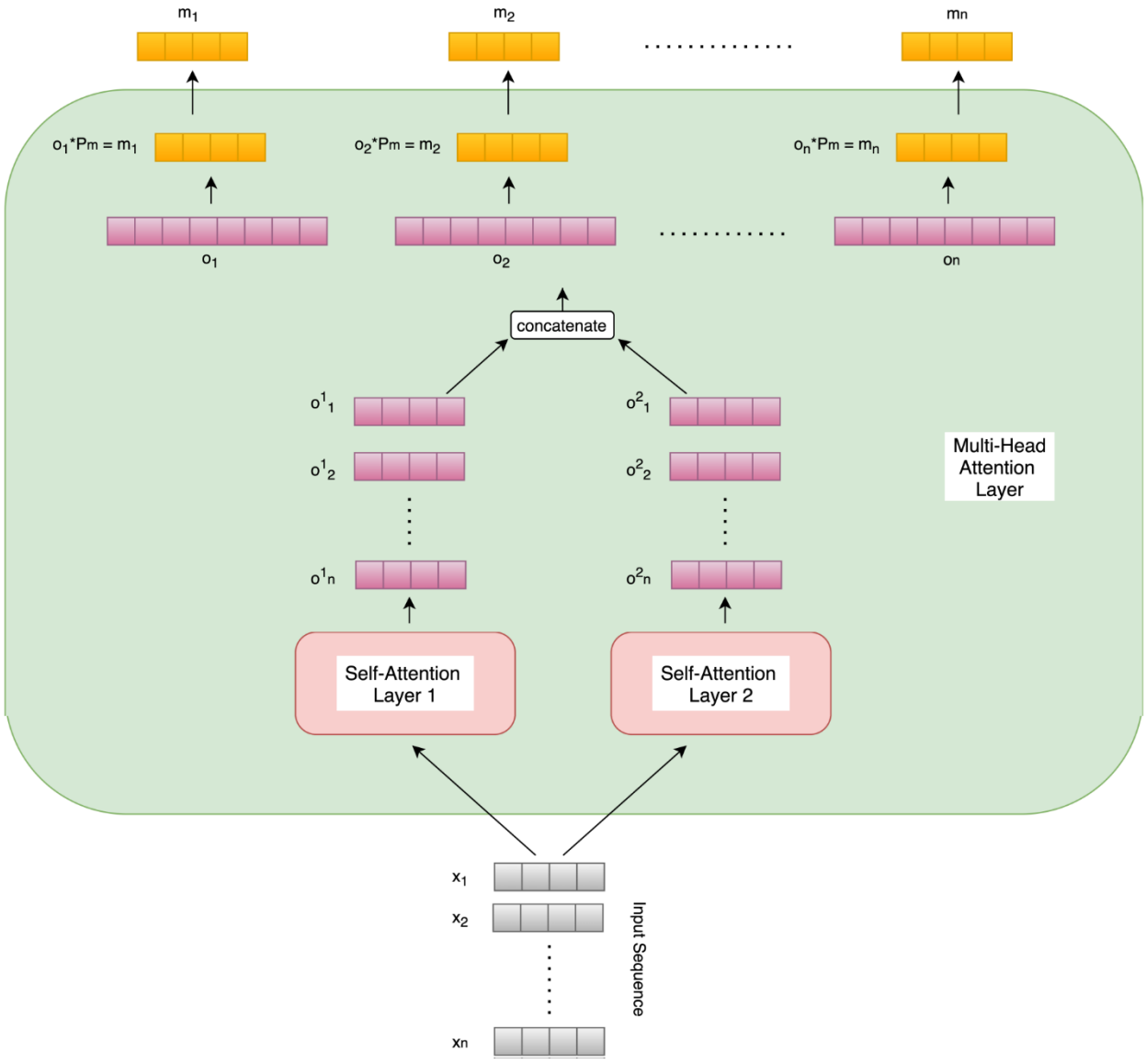
```
0.06318538635969162
0.015872443094849586
0.37745001912117004
0.8425034284591675
0.9304025769233704
```

Chapter 5: Hybrid Advanced Models




Prediction Probabilities
Softmax
Linear Layer
Addition & Layer Normalization
Feed Forward Layer
Residual Connection
Addition & Layer Normalization
Multi-Head Attention
Residual Connection
Addition & Layer Normalization
(Masked) Multi-Head Attention
Residual Connection
Decoder Unit
x 1
Encoder to Decoder
Encoder Unit
2 x
Addition & Layer Normalization
Residual Connection
Feed Forward Layer
Addition & Layer Normalization
Residual Connection
Multi-Head Attention
Positional Encoder
Positional Encoding
Addition
Positional Encoding
Positional Encoder
Embedding Layer
Embedding Layer
Input Sequence
Output Sequence (predicted)


Self-Attention Output
o1
o2
on
s1*v1 = o1
s2*v2 = o2
s1 s2 s3 sn
softmax
c1 c2 c3 cn
Self-Attention Layer
c1 = (q1 dot k1) / sqrt(len(q1))
c2 = (q2 dot k2) / sqrt(len(q2))
x1 * Pq = q1
x1 * Pk = k1
x1 * Pv = v1
x2 * Pq = q2
x2 * Pk = k2
x2 * Pv = v2
x1
x2
xn
Input Sequence


m1
m2
mn
o1*Pm = m1
o2*Pm = m2
on*Pm = mn
o1
o2
on
concatenate
o11
o12
o1n
o21
o22
o2n
Multi-Head Attention Layer
Self-Attention Layer 1
Self-Attention Layer 2
x1
x2
xn
Input Sequence

```
epoch 1, 100/1018 batches, training loss 8.50, training perplexity 4901.66
epoch 1, 200/1018 batches, training loss 7.16, training perplexity 1286.24
epoch 1, 300/1018 batches, training loss 6.76, training perplexity 865.43
epoch 1, 400/1018 batches, training loss 6.55, training perplexity 702.21
epoch 1, 500/1018 batches, training loss 6.45, training perplexity 631.90
epoch 1, 600/1018 batches, training loss 6.31, training perplexity 548.01
epoch 1, 700/1018 batches, training loss 6.25, training perplexity 516.28
epoch 1, 800/1018 batches, training loss 6.11, training perplexity 450.42
epoch 1, 900/1018 batches, training loss 6.09, training perplexity 441.72
epoch 1, 1000/1018 batches, training loss 6.08, training perplexity 436.78

epoch 1, validation loss 5.82, validation perplexity 336.19

epoch 2, 100/1018 batches, training loss 5.98, training perplexity 394.64
epoch 2, 200/1018 batches, training loss 5.90, training perplexity 364.08
```



```
epoch 5, 700/1018 batches, training loss 5.22, training perplexity 185.69
epoch 5, 800/1018 batches, training loss 5.07, training perplexity 158.79
epoch 5, 900/1018 batches, training loss 5.13, training perplexity 169.36
epoch 5, 1000/1018 batches, training loss 5.19, training perplexity 179.63

epoch 5, validation loss 5.23, validation perplexity 186.53
```

testing loss 5.14, testing perplexity 171.47

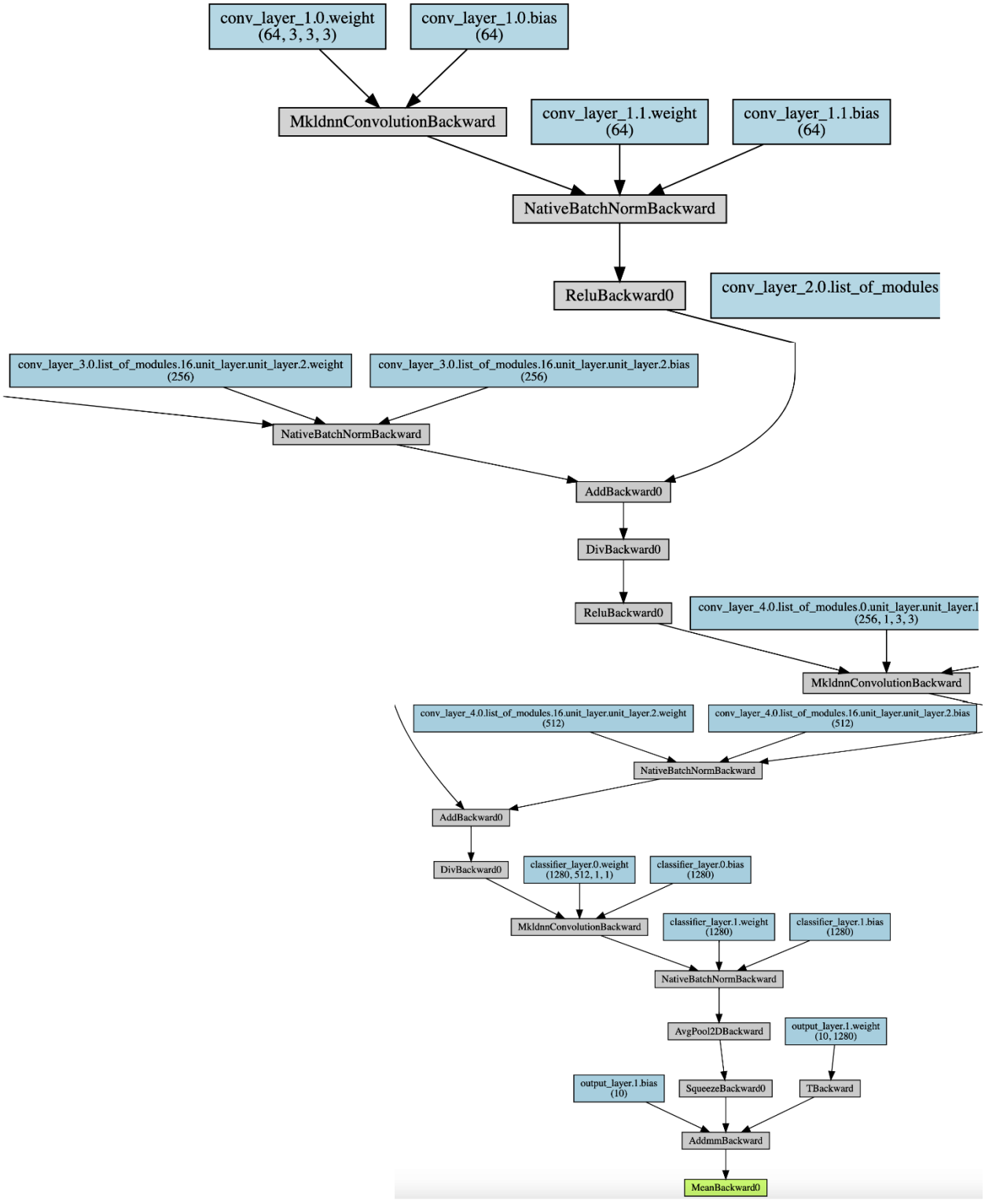
```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to dataset/cifar-10-python.tar.gz
```

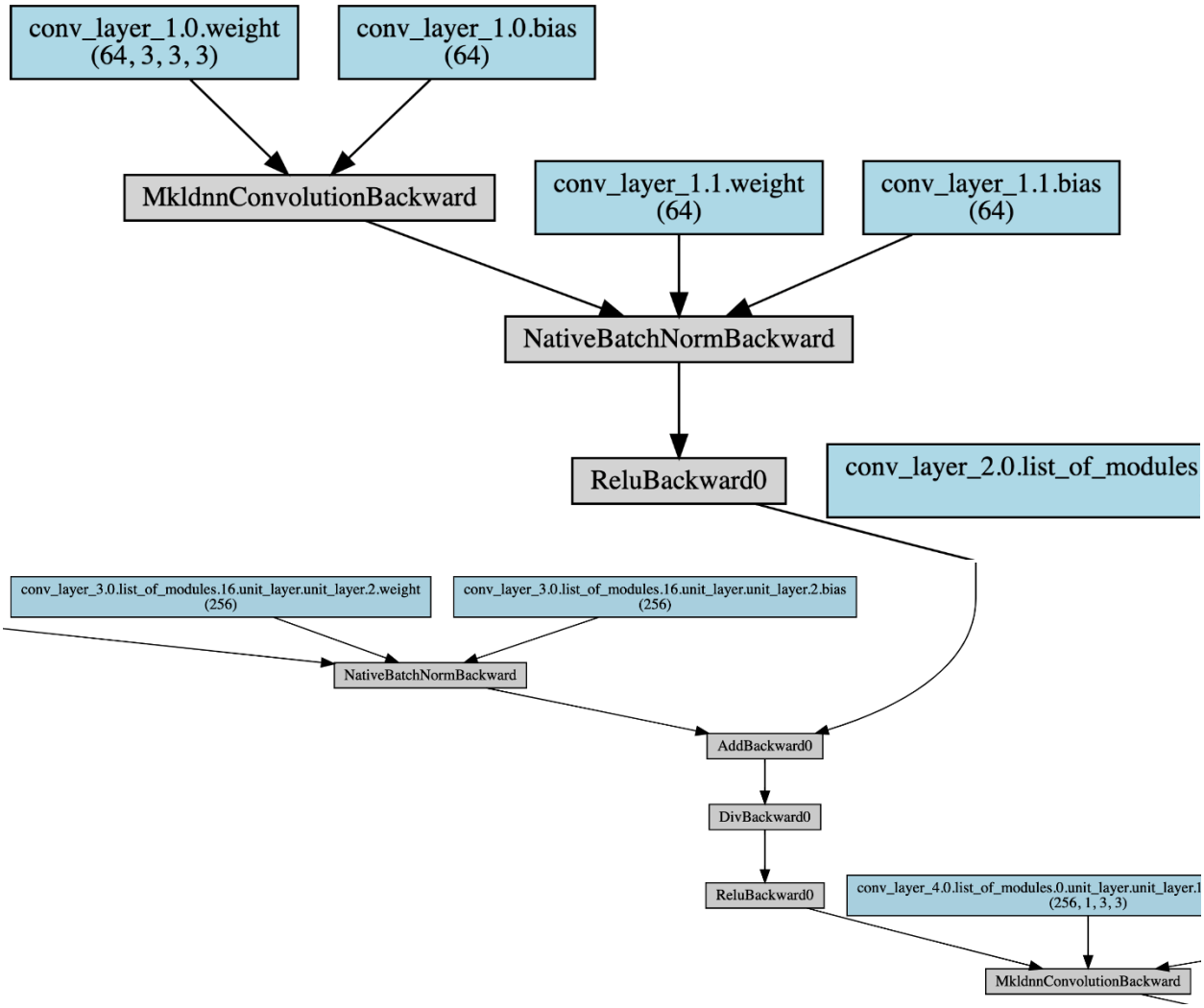
```
170500096/? [03:10<00:00, 889623.81it/s]

Extracting dataset/cifar-10-python.tar.gz to dataset
epoch 1, loss: 1.8047572374343872, accuracy: 32.8125
epoch 1, loss: 1.8053011894226074, accuracy: 39.0625
epoch 1, loss: 1.5705406665802002, accuracy: 40.625
epoch 1, loss: 1.7380733489990234, accuracy: 29.6875
epoch 1, loss: 1.7764639854431152, accuracy: 32.8125
epoch 1, loss: 1.425702691078186, accuracy: 37.5
epoch 1, loss: 1.3414183855056763, accuracy: 51.5625
test acc: 43.24%, best test acc: 0.00%
model train time:  3522.6173169612885
epoch 2, loss: 1.5954769849777222, accuracy: 45.3125
epoch 2, loss: 1.3833452463150024, accuracy: 53.125
epoch 2, loss: 1.370549201965332, accuracy: 43.75
epoch 2, loss: 1.3685939311981201, accuracy: 54.6875
epoch 2, loss: 1.4633197784423828, accuracy: 48.4375
epoch 2, loss: 1.2918241024017334, accuracy: 50.0
epoch 2, loss: 1.317800521850586, accuracy: 50.0
test acc: 51.04%, best test acc: 43.24%
model train time:  6938.013380050659
epoch 3, loss: 1.0907424688339233, accuracy: 59.375
```



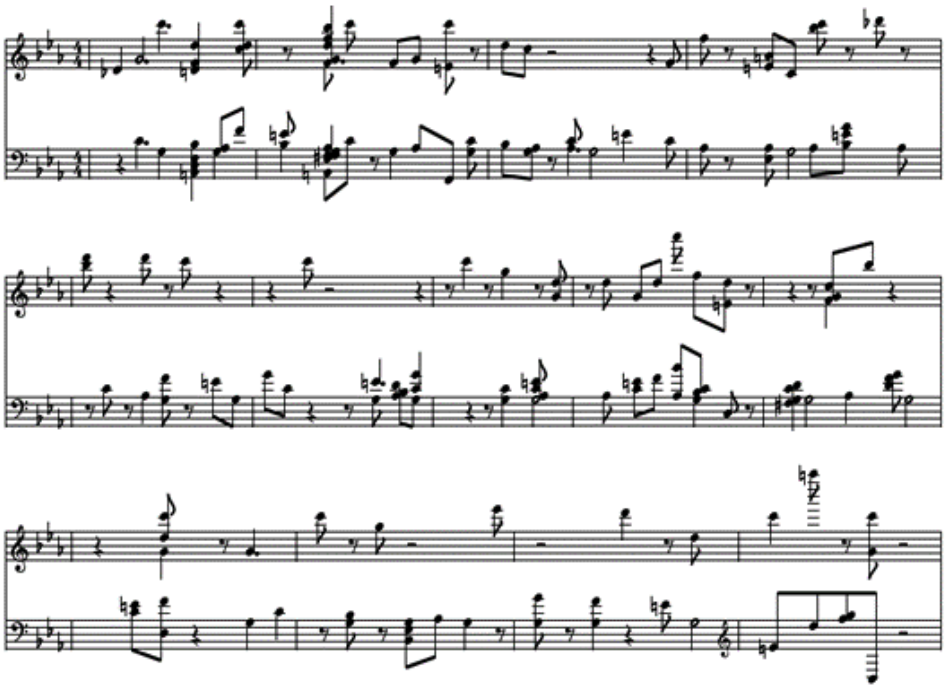
```
epoch 5, loss: 1.2000718116760254, accuracy: 62.5
test acc: 67.45%, best test acc: 67.73%
```

best model accuracy: 67.73%, last epoch: 4


conv_layer_1.0.weight
(64, 3, 3, 3)
conv_layer_1.0.bias
(64)
MkldnnConvolutionBackward
conv_layer_1.1.weight
(64)
conv_layer_1.1.bias
(64)
NativeBatchNormBackward
ReluBackward0
conv_layer_2.0.list_of_modules
conv_layer_3.0.list_of_modules.16.unit_layer.unit_layer.2.weight
(256)
conv_layer_3.0.list_of_modules.16.unit_layer.unit_layer.2.bias
(256)
NativeBatchNormBackward
AddBackward0
DivBackward0
ReluBackward0
conv_layer_4.0.list_of_modules.0.unit_layer.unit_layer.l
(256, 1, 3, 3)
MkldnnConvolutionBackward
conv_layer_4.0.list_of_modules.16.unit_layer.unit_layer.2.weight
(512)
conv_layer_4.0.list_of_modules.16.unit_layer.unit_layer.2.bias
(512)
NativeBatchNormBackward
AddBackward0
DivBackward0
classifier_layer.0.weight
(1280, 512, 1, 1)
classifier_layer.0.bias
(1280)
MkldnnConvolutionBackward
classifier_layer.1.weight
(1280)
classifier_layer.1.bias
(1280)
NativeBatchNormBackward
AvgPool2DBackward
output_layer.1.weight
(10, 1280)
output_layer.1.bias
(10)
SqueezeBackward0
TBackward
AddmmBackward
MeanBackward0


conv_layer_1.0.weight
(64, 3, 3, 3)
conv_layer_1.0.bias
(64)
MkldnnConvolutionBackward
conv_layer_1.1.weight
(64)
conv_layer_1.1.bias
(64)
NativeBatchNormBackward
ReluBackward0
conv_layer_2.0.list_of_modules
conv_layer_3.0.list_of_modules.16.unit_layer.unit_layer.2.weight
(256)
conv_layer_3.0.list_of_modules.16.unit_layer.unit_layer.2.bias
(256)
NativeBatchNormBackward
AddBackward0
DivBackward0
ReluBackward0
conv_layer_4.0.list_of_modules.0.unit_layer.unit_layer.1
(256, 1, 3, 3)
MkldnnConvolutionBackward

Chapter 6: Music and Text Generation with PyTorch



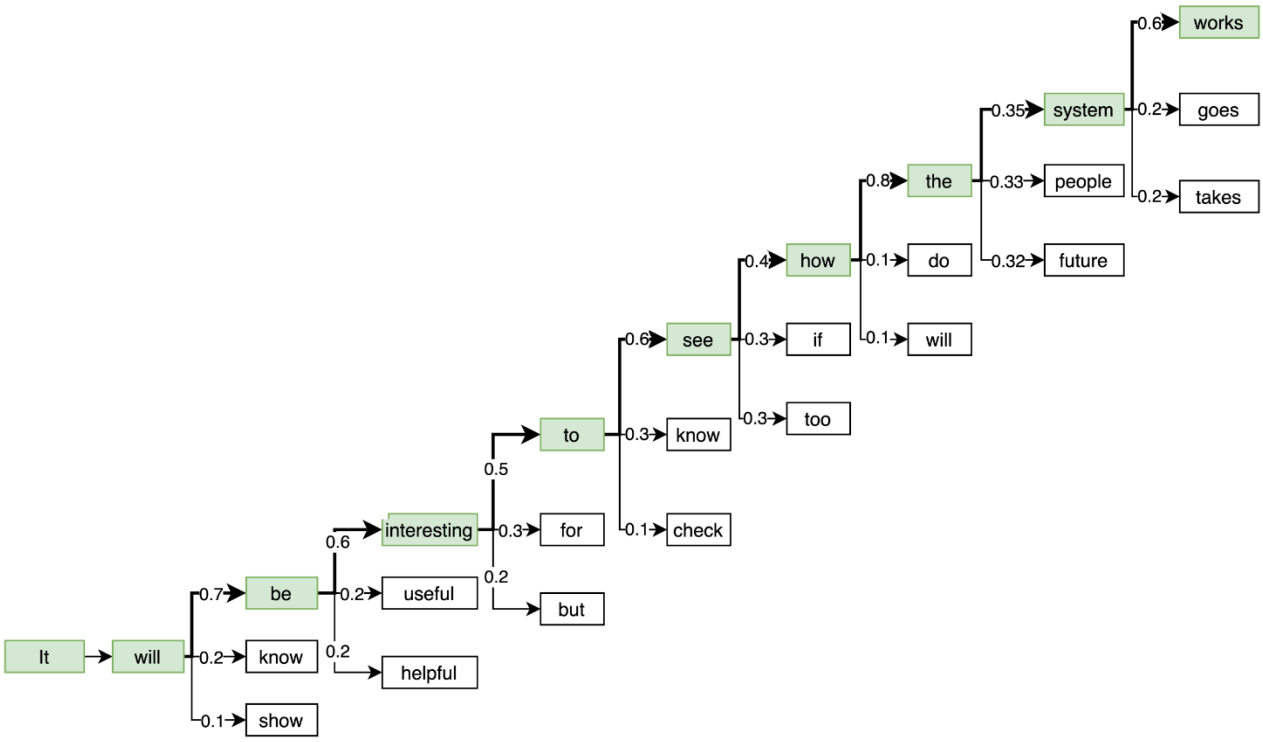
```
epoch 1, 100/1018 batches, training loss 8.63, training perplexity 5614.45
epoch 1, 200/1018 batches, training loss 7.23, training perplexity 1380.31
epoch 1, 300/1018 batches, training loss 6.79, training perplexity 892.50
epoch 1, 400/1018 batches, training loss 6.55, training perplexity 701.84
epoch 1, 500/1018 batches, training loss 6.45, training perplexity 634.57
epoch 1, 600/1018 batches, training loss 6.32, training perplexity 553.86
epoch 1, 700/1018 batches, training loss 6.24, training perplexity 513.65
epoch 1, 800/1018 batches, training loss 6.13, training perplexity 459.07
epoch 1, 900/1018 batches, training loss 6.11, training perplexity 450.48
epoch 1, 1000/1018 batches, training loss 6.07, training perplexity 433.88

epoch 1, validation loss 5.82, validation perplexity 337.70

epoch 2, 100/1018 batches, training loss 5.98, training perplexity 395.15
epoch 2, 200/1018 batches, training loss 5.90, training perplexity 363.99
```



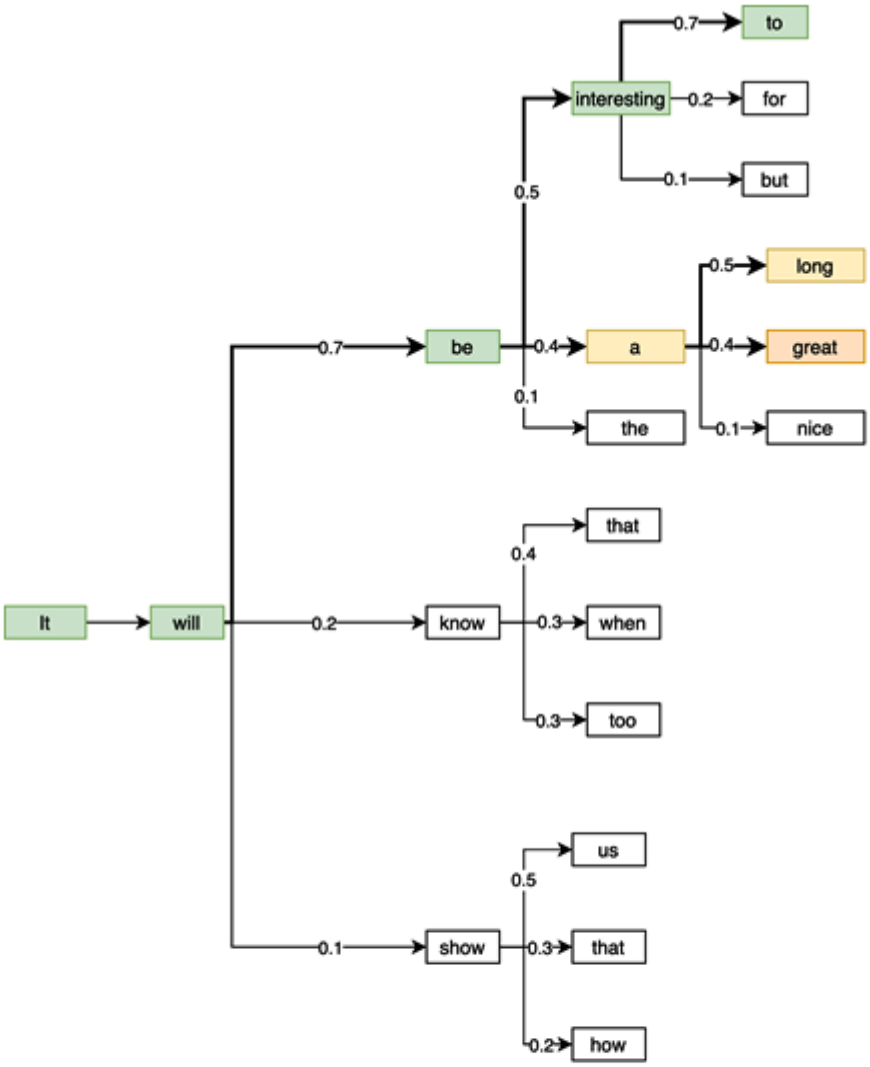
```
epoch 50, 100/1018 batches, training loss 4.45, training perplexity 85.55
epoch 50, 200/1018 batches, training loss 4.38, training perplexity 79.68
epoch 50, 300/1018 batches, training loss 4.39, training perplexity 80.61
epoch 50, 400/1018 batches, training loss 4.39, training perplexity 80.27
epoch 50, 500/1018 batches, training loss 4.39, training perplexity 80.31
epoch 50, 600/1018 batches, training loss 4.38, training perplexity 80.17
epoch 50, 700/1018 batches, training loss 4.41, training perplexity 82.47
epoch 50, 800/1018 batches, training loss 4.26, training perplexity 71.00
epoch 50, 900/1018 batches, training loss 4.33, training perplexity 76.24
epoch 50, 1000/1018 batches, training loss 4.36, training perplexity 78.51

epoch 50, validation loss 4.98, validation perplexity 145.72
```

```
It will be used to the first season , and the
```

It will be interesting to see how the new system works



It will be interesting to see how the new system



It will be = 0.7
It will know = 0.2
It will show = 0.1

It will be interesting = 0.7 * 0.5
It will be a = 0.7 * 0.4
It will know that = 0.2 * 0.4
It will be the = 0.7 * 0.1
It will know when = 0.2 * 0.3
It will know too = 0.2 * 0.3
It will show us = 0.1 *0.5
It will show that = 0.1 * 0.3
It will show how = 0.1 * 0.2

It will be interesting to = 0.7 * 0.5 * 0.7
It will be a long = 0.7 * 0.4 * 0.5
It will be a great = 0.7 * 0.4 * 0.4
It will be interesting for = 0.7 * 0.5 * 0.2
It will be a nice = 0.7 * 0.4 * 0.1
...

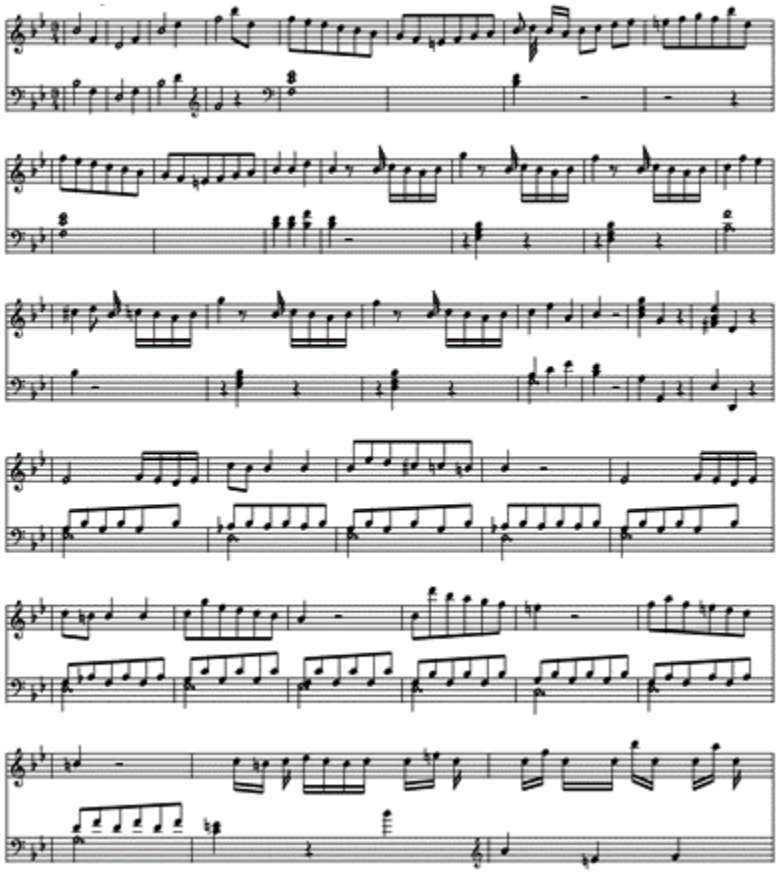
```
It will be interesting to
It will be a long
It will be a great
```

```
It will also be a
It will be a long
It will also be interesting
```

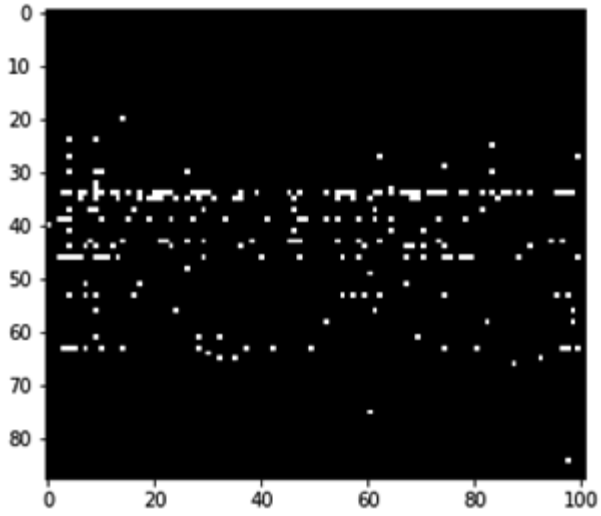
```python
for i in range(3):
    torch.manual_seed(i)
    op_greedy = mdl.generate(ip_ids, max_length=5)
    seq = tkz.decode(op_greedy[0], skip_special_tokens=True)
    print(seq)
```

```
It will be interesting to
It will be interesting to
It will be interesting to
```

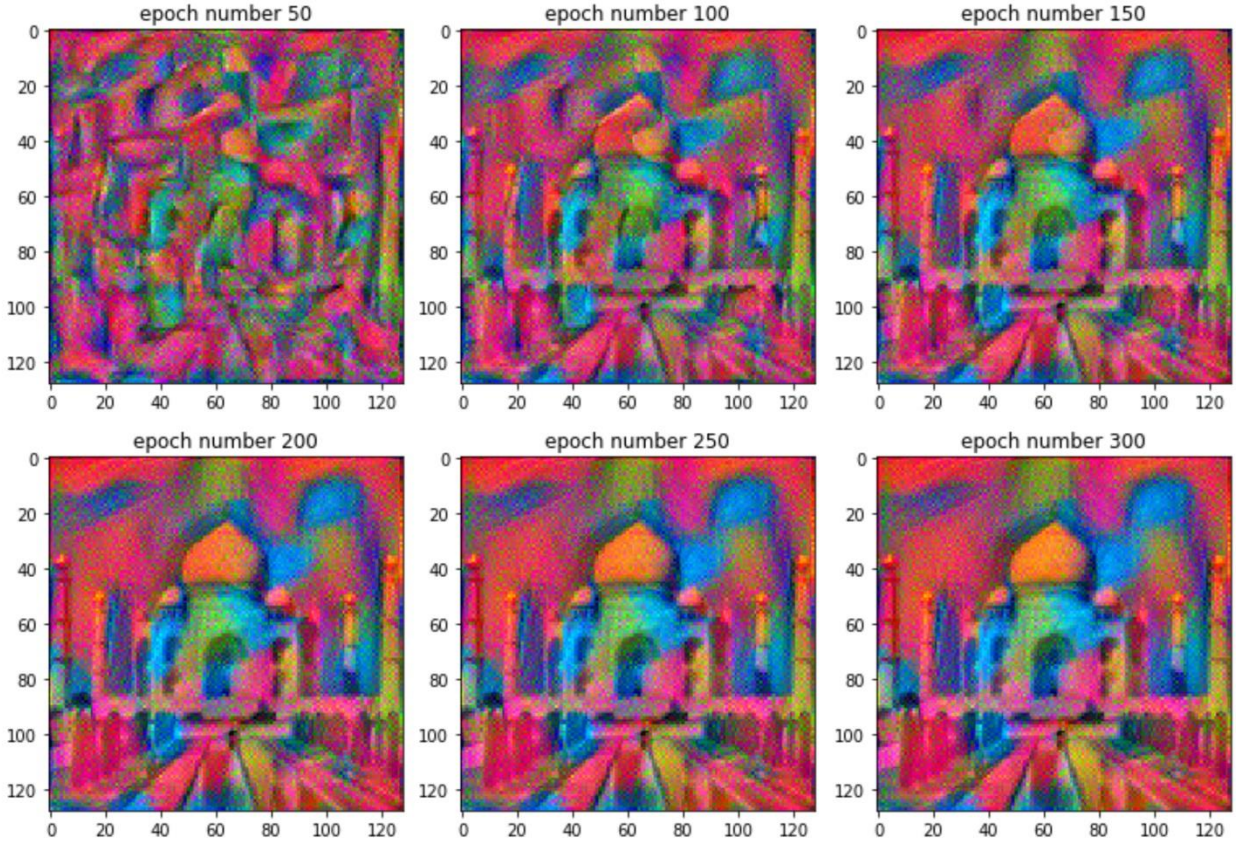
It will require work in
It will be an interesting
It will likely be important torch.Size([3, 1587, 88])

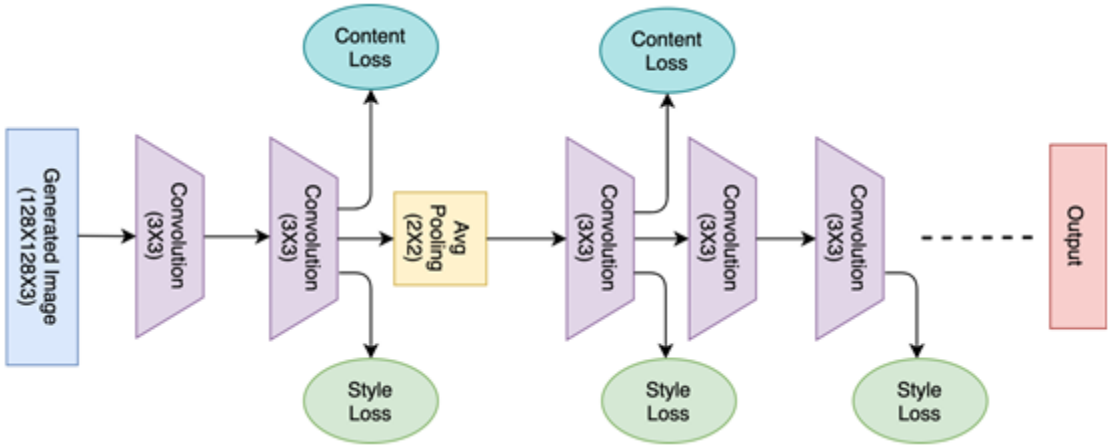


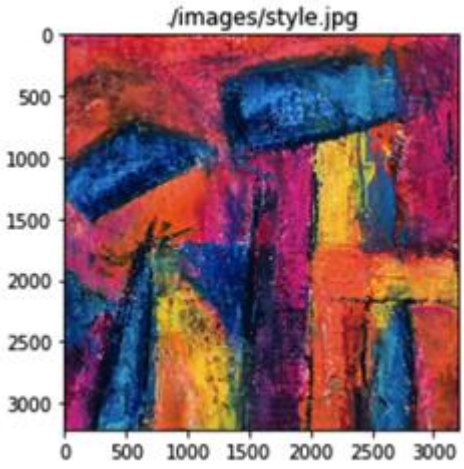

```
ep 0 , train loss = 1.2445591886838276
ep 0 , val loss = 1.3352128363692468e-06

ep 1 , train loss = 2.1156165103117623
ep 1 , val loss = 1.6539533744088603e-06

ep 2 , train loss = 1.6429476936658223
ep 2 , val loss = 6.44313576921296e-07

ep 3 , train loss = 1.3036367297172546
ep 3 , val loss = 7.910344729101428e-07

ep 4 , train loss = 0.6105860968430837
ep 4 , val loss = 1.2166870756004527e-06

ep 5 , train loss = 0.582861324151357
ep 5 , val loss = 5.687958283017817e-07

ep 6 , train loss = 0.28131235639254254
ep 6 , val loss = 4.83049781240143e-07

ep 7 , train loss = 0.1561812162399292
ep 7 , val loss = 5.472248898085979e-07

ep 8 , train loss = 0.14845856527487436
ep 8 , val loss = 4.1753687837465244e-07

ep 9 , train loss = 0.1285532539089521
ep 9 , val loss = 3.899009367655375e-07
```

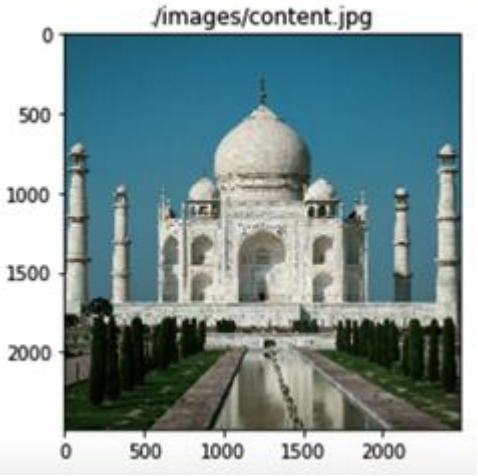

0
10
20
30
40
50
60
70
80
0
20
40
60
80
100

Chapter 7: Neural Style Transfer



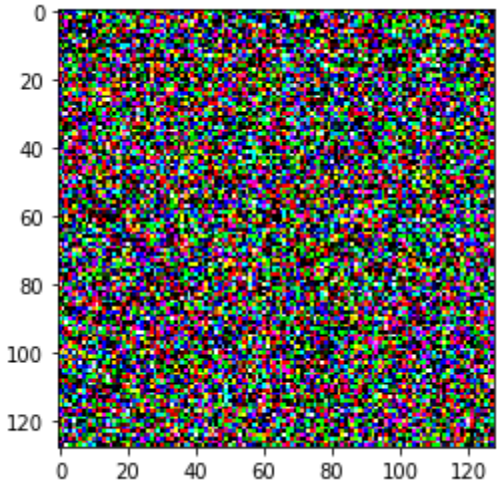


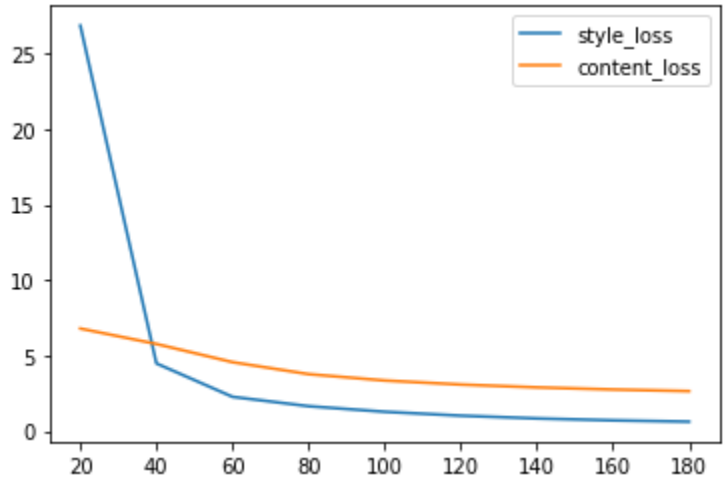

./images/style.jpg
0
500
1000
1500
2000
2500
3000
0
500
1000
1500
2000
2500
3000


./images/content.jpg
0
500
1000
1500
2000
0
500
1000
1500
2000

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): ReLU(inplace=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU()
  (4): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU()
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU()
  (9): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
)
```


0
20
40
60
80
100
120
0
20
40
60
80
100
120


style_loss
content_loss
25
20
15
10
5
0
20
40
60
80
100
120
140
160
180

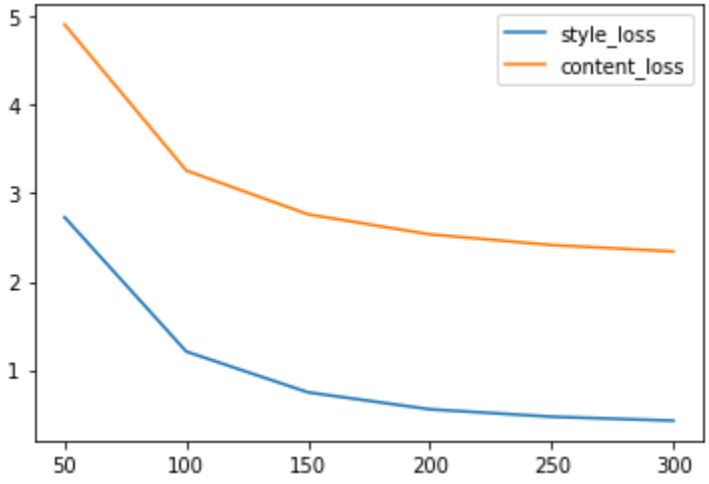
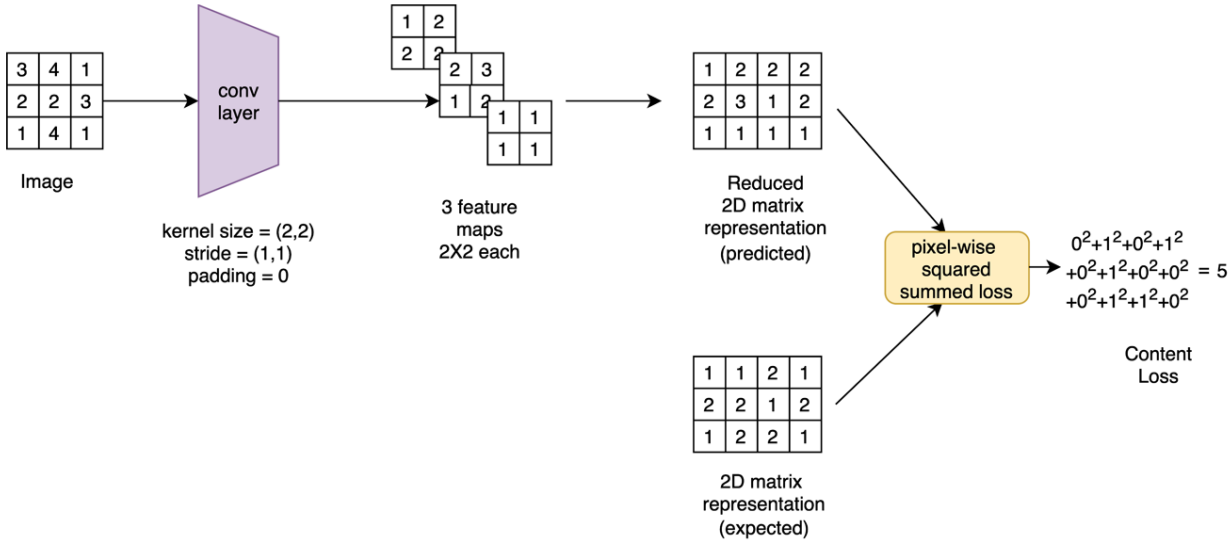
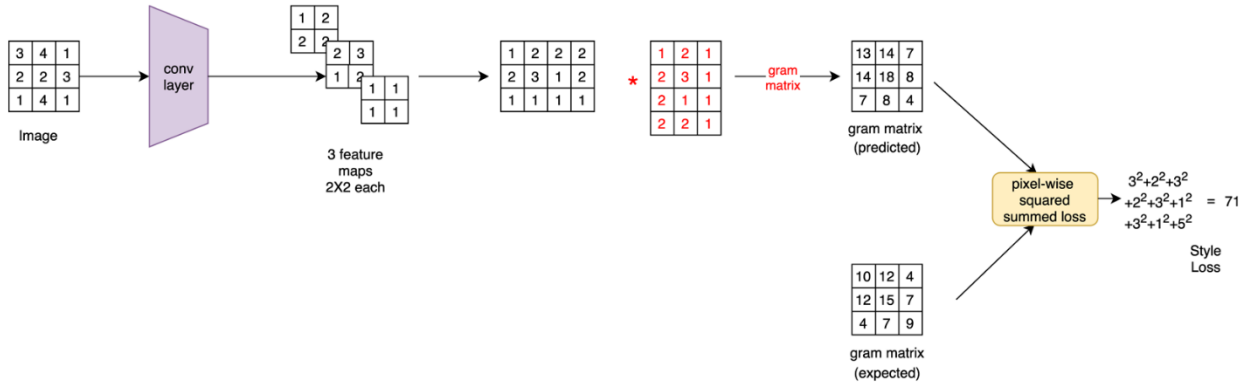

style_loss
content_loss
5
4
3
2
1
50
100
150
200
250
300

Image A
(Content)

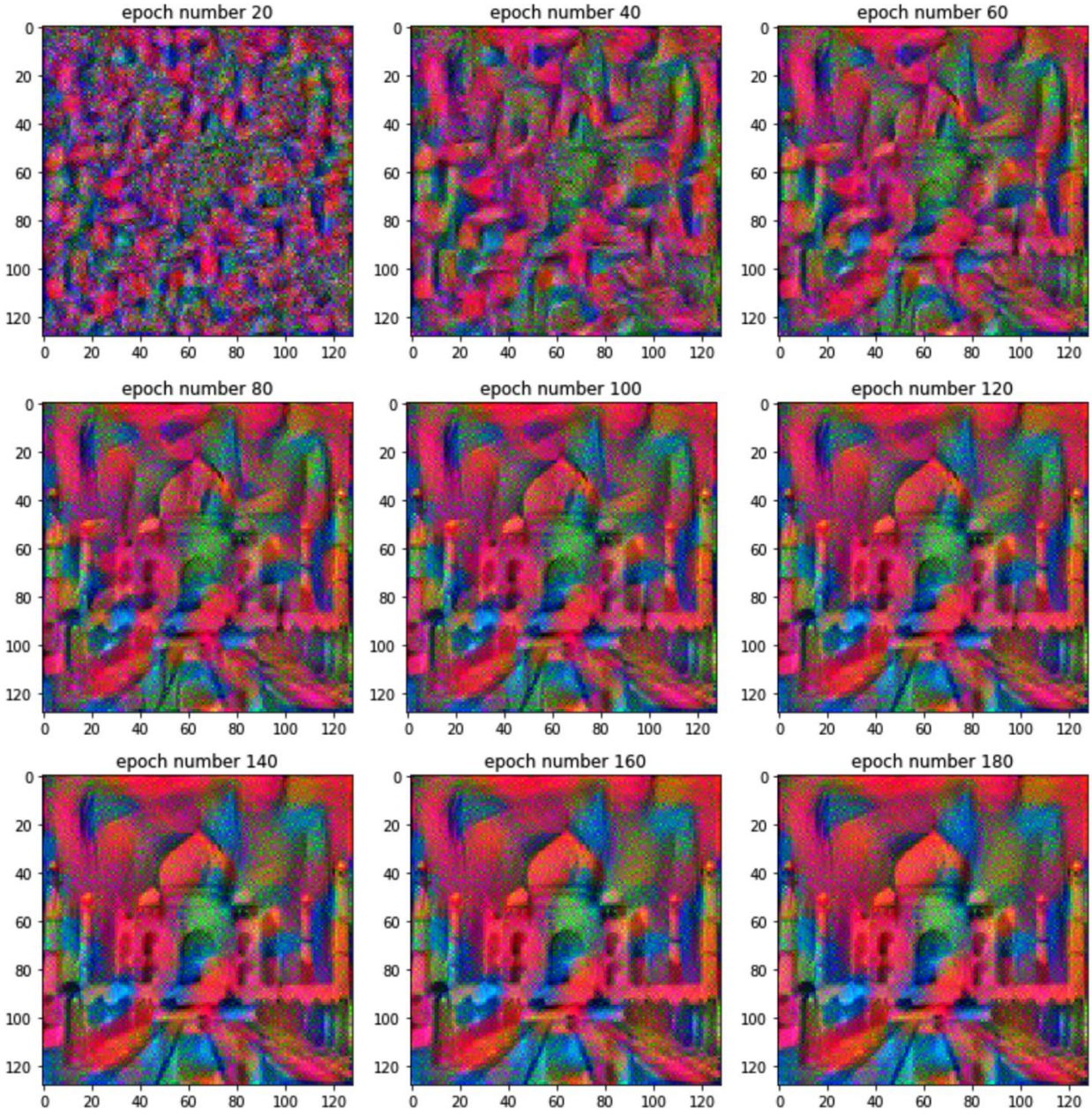
Image B
(Style)

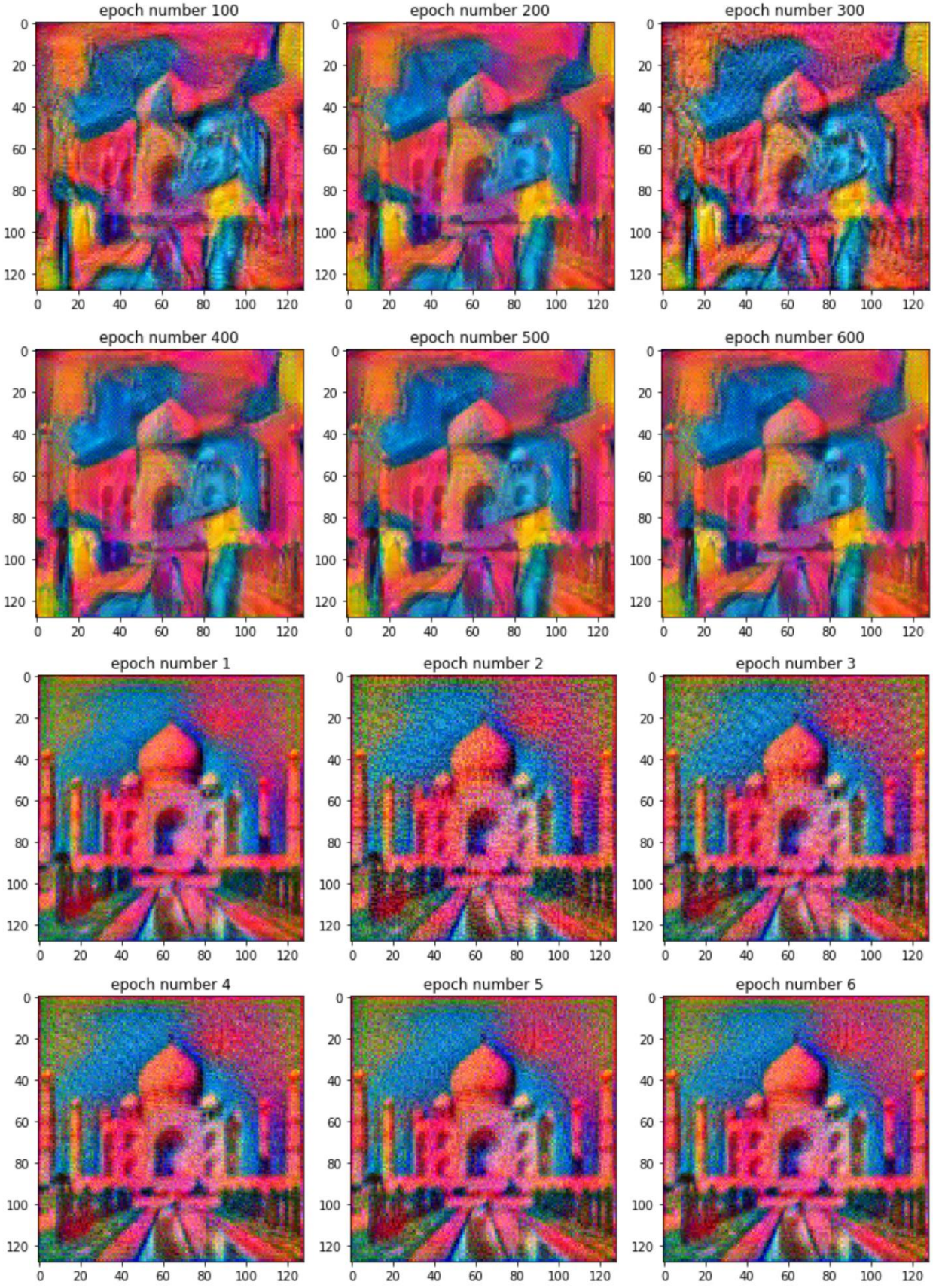
Image C
(Resultant)



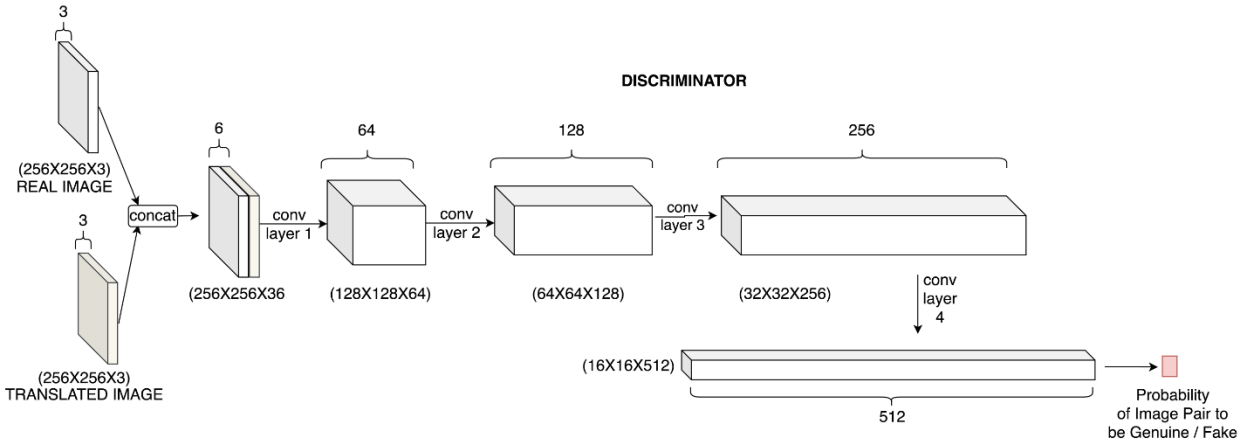


```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace=True)
  (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (17): ReLU(inplace=True)
  (18): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace=True)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace=True)
  (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (24): ReLU(inplace=True)
  (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (26): ReLU(inplace=True)
  (27): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (29): ReLU(inplace=True)
  (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (31): ReLU(inplace=True)
  (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (33): ReLU(inplace=True)
  (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (35): ReLU(inplace=True)
  (36): AvgPool2d(kernel_size=2, stride=2, padding=0)
)
```


epoch number 20
epoch number 40
epoch number 60
epoch number 80
epoch number 100
epoch number 120
epoch number 140
epoch number 160
epoch number 180


epoch number 100
epoch number 200
epoch number 300
epoch number 400
epoch number 500
epoch number 600
epoch number 1
epoch number 2
epoch number 3
epoch number 4
epoch number 5
epoch number 6

Chapter 8: Deep Convolutional GANs

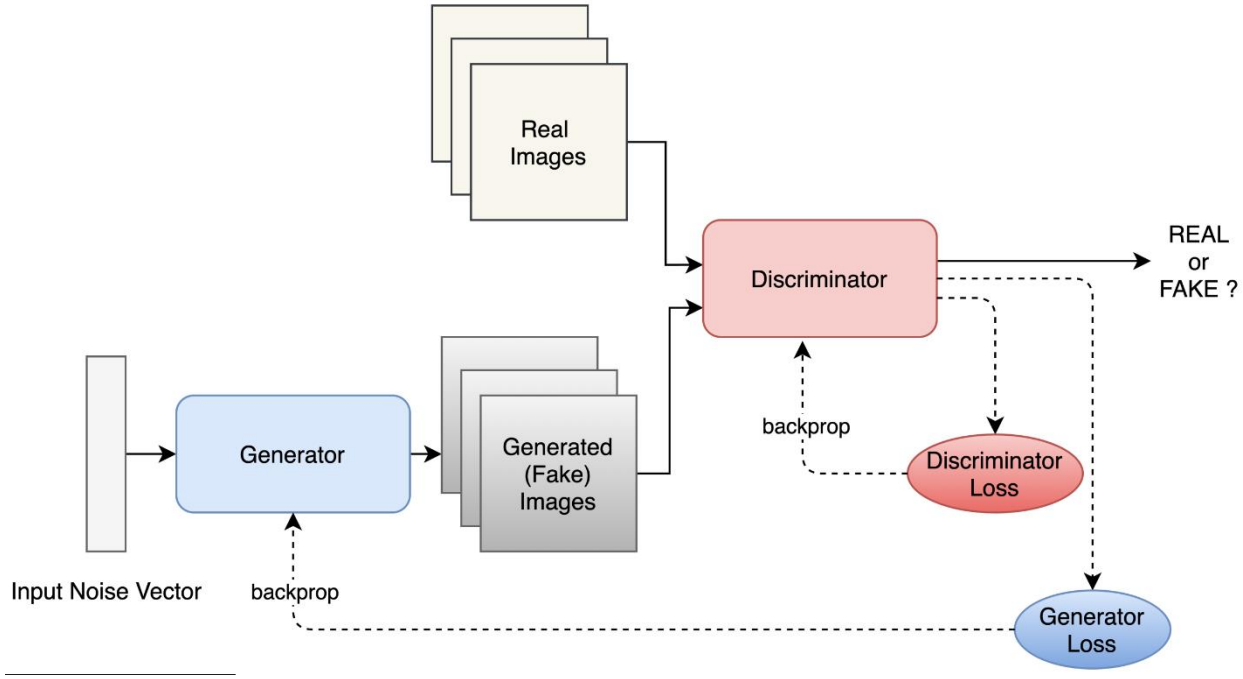


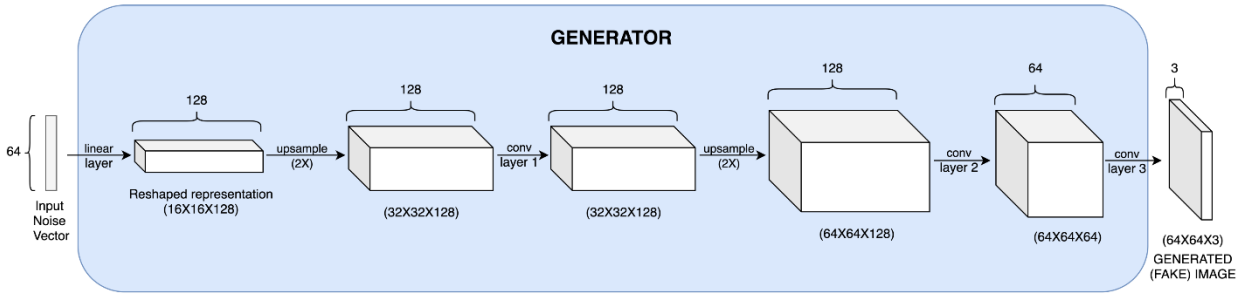
conv layer 1: in_channels=16, out_channels=64, kernel_size=4, stride=2, padding=ON

conv layer 2: in_channels=64, out_channels=128, kernel_size=4, stride=2, padding=ON

conv layer 3: in_channels=128, out_channels=256, kernel_size=4, stride=2, padding=ON

conv layer 4: in_channels=256, out_channels=512, kernel_size=4, stride=2, padding=ON



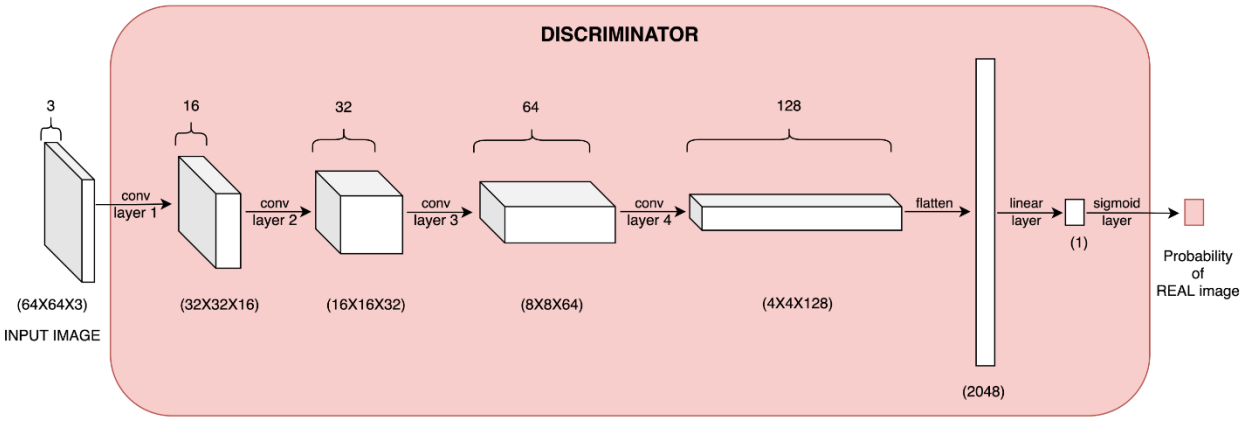


upsample: upsampling along spatial dimensions based on nearest neighbour strategy

conv layer 1: in_channels=128, out_channels=128, kernel_size=3, stride=1, padding=ON

conv layer 2: in_channels=128, out_channels=64, kernel_size=3, stride=1, padding=ON

conv layer 3: in_channels=64, out_channels=3, kernel_size=3, stride=1, padding=ON



conv layer 1: in_channels=3, out_channels=16, kernel_size=3, stride=2, padding=ON

conv layer 2: in_channels=16, out_channels=32, kernel_size=3, stride=2, padding=ON

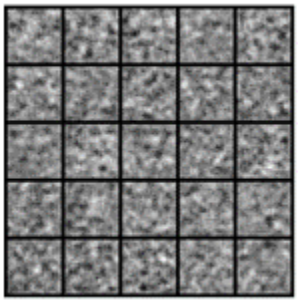
conv layer 3: in_channels=32, out_channels=64, kernel_size=3, stride=2, padding=ON

conv layer 4: in_channels=64, out_channels=128, kernel_size=3, stride=2, padding=ON

```
epoch number 0 | batch number 0 | generator loss = 0.683123 | discriminator loss = 0.693203
epoch number 0 | batch number 200 | generator loss = 5.871073 | discriminator loss = 0.032416
epoch number 0 | batch number 400 | generator loss = 2.876508 | discriminator loss = 0.288186
epoch number 0 | batch number 600 | generator loss = 3.705342 | discriminator loss = 0.049239
epoch number 0 | batch number 800 | generator loss = 2.727477 | discriminator loss = 0.542196
epoch number 0 | batch number 1000 | generator loss = 3.382538 | discriminator loss = 0.282721
epoch number 0 | batch number 1200 | generator loss = 1.695523 | discriminator loss = 0.304907
epoch number 0 | batch number 1400 | generator loss = 2.297853 | discriminator loss = 0.655593
epoch number 0 | batch number 1600 | generator loss = 1.397890 | discriminator loss = 0.599436
```

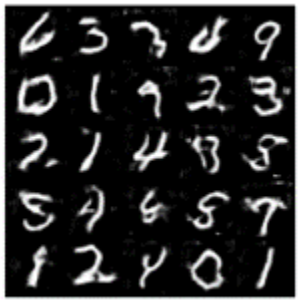
```
epoch number 10 | batch number 3680 | generator loss = 1.407570 | discriminator loss = 0.409708
epoch number 10 | batch number 3880 | generator loss = 0.667673 | discriminator loss = 0.808560
epoch number 10 | batch number 4080 | generator loss = 0.793113 | discriminator loss = 0.679659
epoch number 10 | batch number 4280 | generator loss = 0.902015 | discriminator loss = 0.709771
epoch number 10 | batch number 4480 | generator loss = 0.640646 | discriminator loss = 0.321178
epoch number 10 | batch number 4680 | generator loss = 1.235740 | discriminator loss = 0.465171
epoch number 10 | batch number 4880 | generator loss = 0.896295 | discriminator loss = 0.451197
epoch number 10 | batch number 5080 | generator loss = 0.690564 | discriminator loss = 0.285500
```

Epoch 0

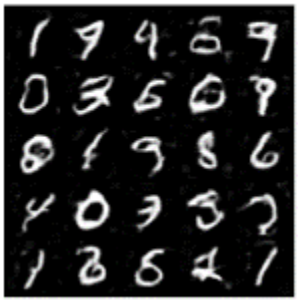


Epoch 1

Epoch 2

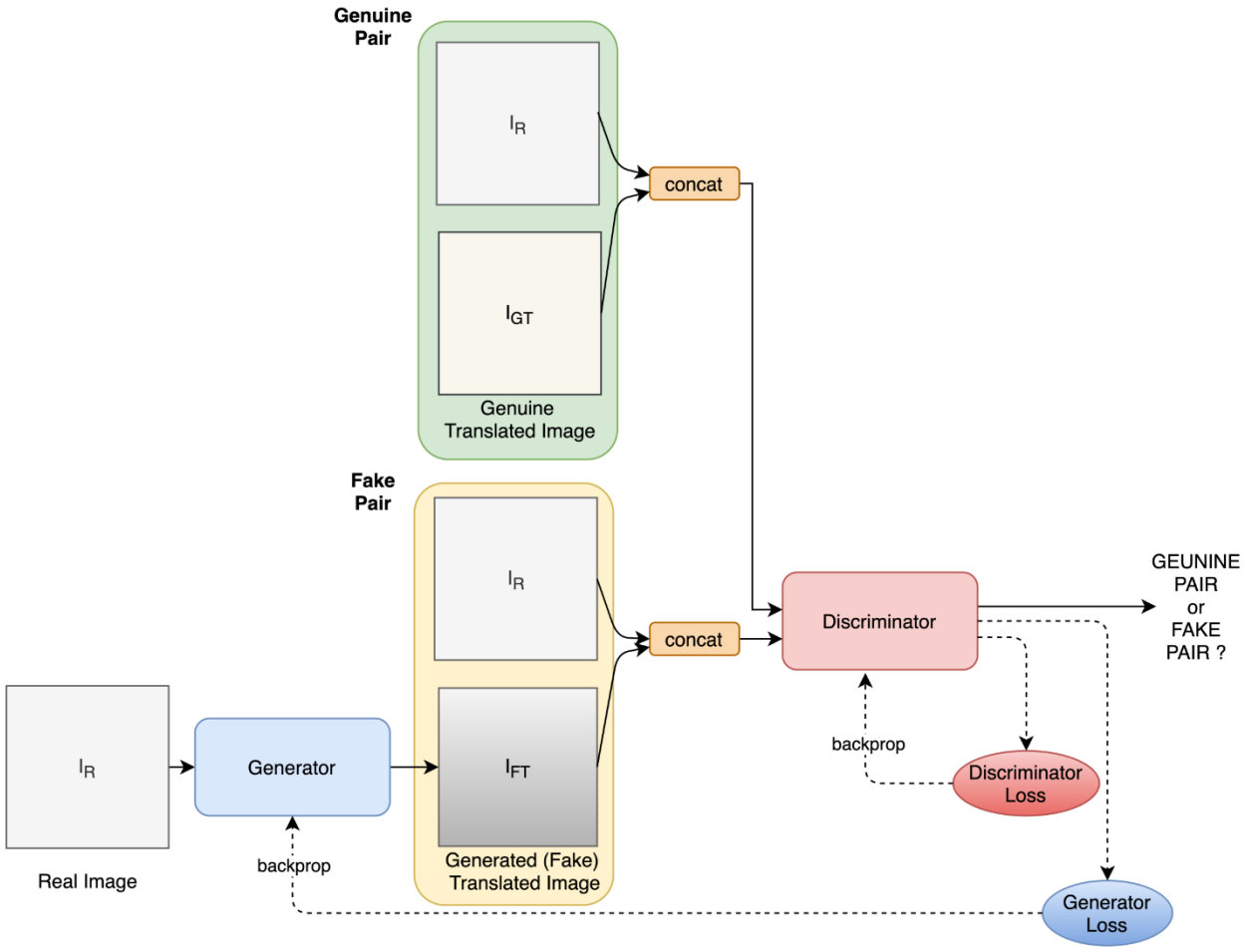


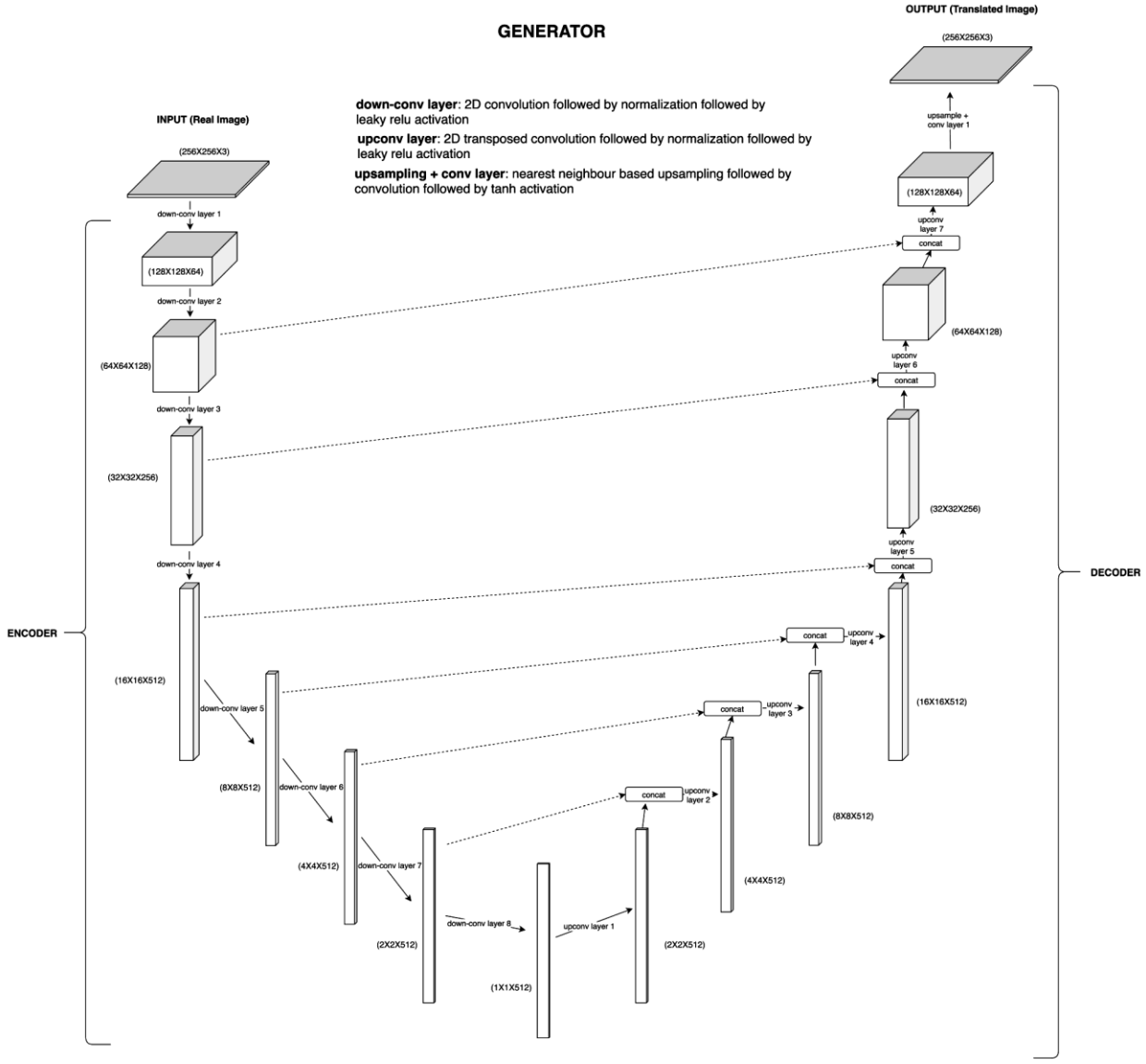
Epoch 3



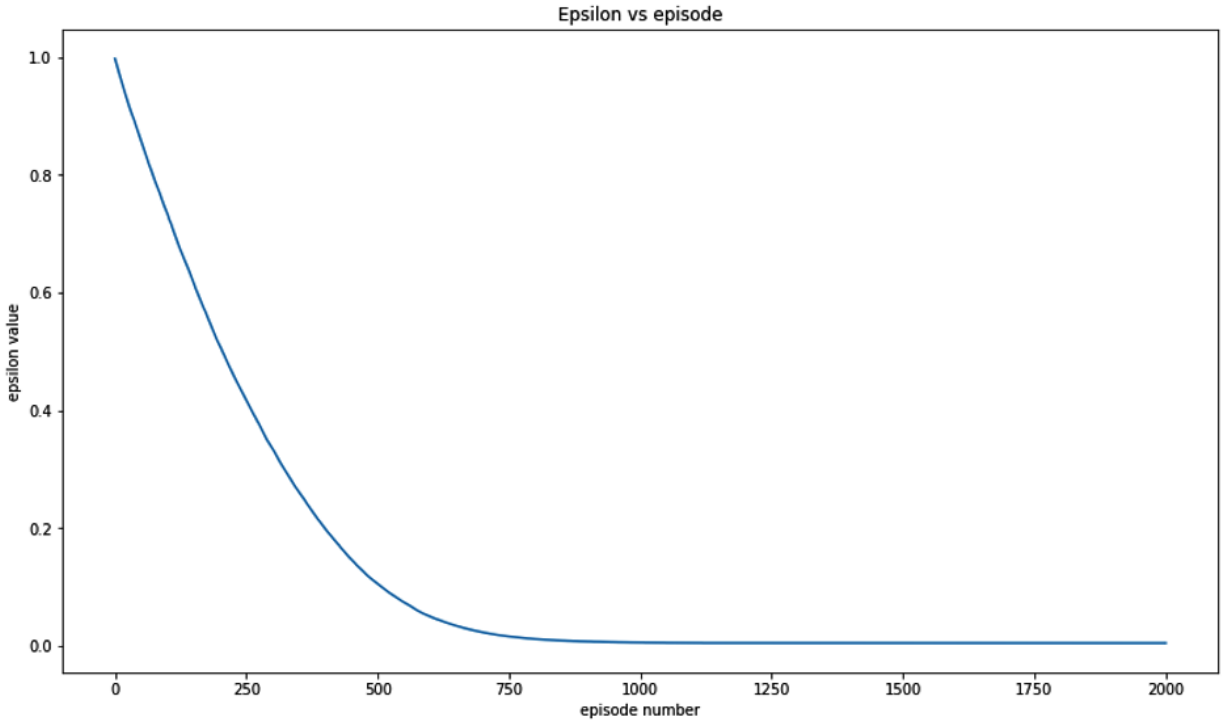
Epoch 4

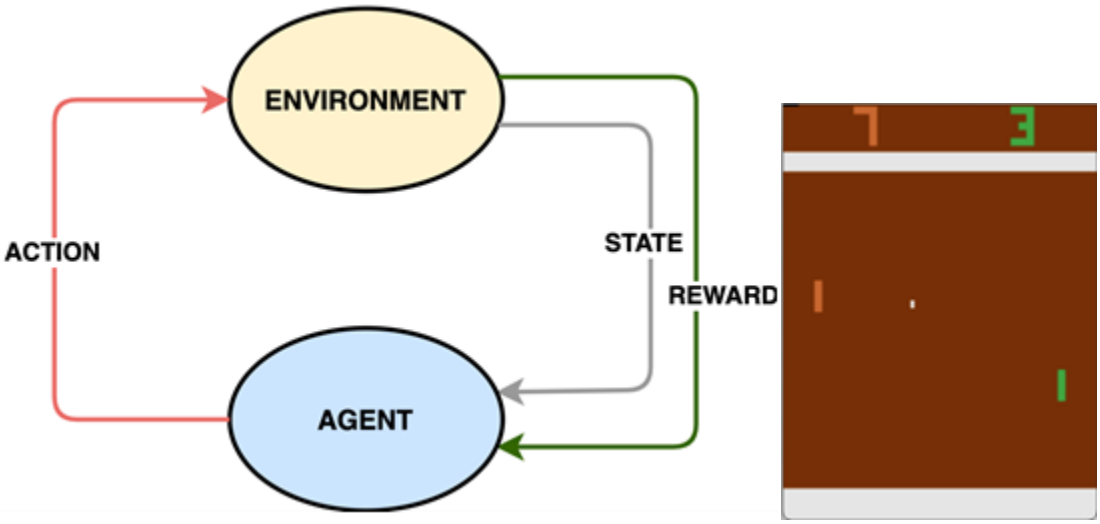
Epoch 5

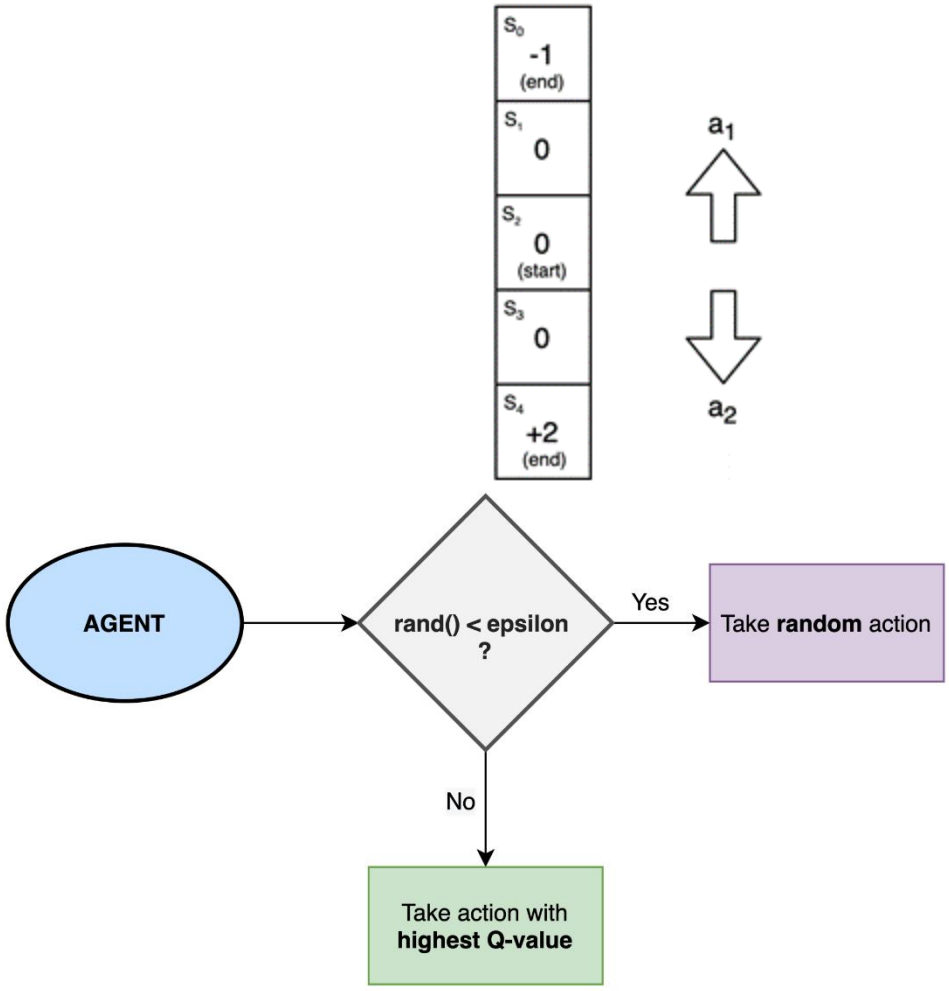

Genuine Pair
I_R
I_{GT}
Genuine Translated Image
concat
Fake Pair
I_R
I_{FT}
Generated (Fake) Translated Image
concat
Discriminator
GEUNINE PAIR or FAKE PAIR ?
backprop
Discriminator Loss
I_R
Real Image
Generator
backprop
Generator Loss


GENERATOR
down-conv layer: 2D convolution followed by normalization followed by leaky relu activation
upconv layer: 2D transposed convolution followed by normalization followed by leaky relu activation
upsampling + conv layer: nearest neighbour based upsampling followed by convolution followed by tanh activation
INPUT (Real Image)
(256X256X3)
down-conv layer 1
(128X128X64)
down-conv layer 2
(64X64X128)
down-conv layer 3
(32X32X256)
down-conv layer 4
(16X16X512)
down-conv layer 5
(8X8X512)
down-conv layer 6
(4X4X512)
down-conv layer 7
(2X2X512)
down-conv layer 8
(1X1X512)
ENCODER
upconv layer 1
(2X2X512)
concat
upconv layer 2
(4X4X512)
concat
upconv layer 3
(8X8X512)
concat
upconv layer 4
(16X16X512)
concat
upconv layer 5
(32X32X256)
concat
upconv layer 6
(64X64X128)
concat
upconv layer 7
(128X128X64)
upsample + conv layer 1
(256X256X3)
OUTPUT (Translated Image)
DECODER

Chapter 9: Deep Reinforcement Learning

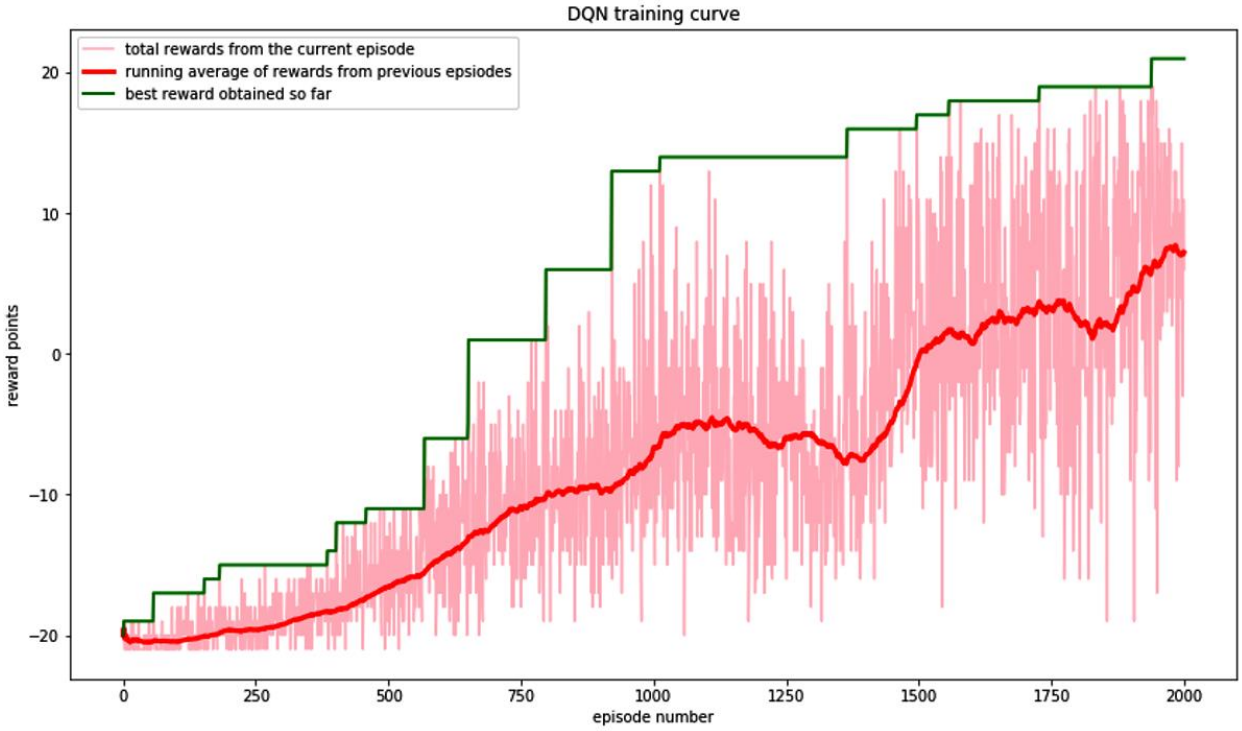






Episode 1: epsilon = 1.0
Episode 2: epsilon = 0.9
Episode 3: epsilon = 0.8
Episode 4: epsilon = 0.7
Episode 5: epsilon = 0.6
Episode 6: epsilon = 0.5
Episode 7: epsilon = 0.4
Episode 8: epsilon = 0.3
Episode 9: epsilon = 0.2
...

```
episode_num 0, curr_reward: -20.0, best_reward: -20.0, running_avg_reward: -20.0, curr_epsilon: 0.9971
checkpointing current model weights. highest running_average_reward of -19.5 achieved!
episode_num 1, curr_reward: -19.0, best_reward: -19.0, running_avg_reward: -19.5, curr_epsilon: 0.9937
episode_num 2, curr_reward: -21.0, best_reward: -19.0, running_avg_reward: -20.0, curr_epsilon: 0.991
episode_num 3, curr_reward: -21.0, best_reward: -19.0, running_avg_reward: -20.25, curr_epsilon: 0.9881
episode_num 4, curr_reward: -19.0, best_reward: -19.0, running_avg_reward: -20.0, curr_epsilon: 0.9846
episode_num 5, curr_reward: -20.0, best_reward: -19.0, running_avg_reward: -20.0, curr_epsilon: 0.9811
                                        .
                                        .
                                        .
episode_num 500, curr_reward: -13.0, best_reward: -11.0, running_avg_reward: -16.52, curr_epsilon: 0.1053
episode_num 501, curr_reward: -20.0, best_reward: -11.0, running_avg_reward: -16.52, curr_epsilon: 0.1049
episode_num 502, curr_reward: -19.0, best_reward: -11.0, running_avg_reward: -16.59, curr_epsilon: 0.1041
episode_num 503, curr_reward: -12.0, best_reward: -11.0, running_avg_reward: -16.53, curr_epsilon: 0.1034
checkpointing current model weights. highest running_average_reward of -16.51 achieved!
episode_num 504, curr_reward: -13.0, best_reward: -11.0, running_avg_reward: -16.51, curr_epsilon: 0.1026
checkpointing current model weights. highest running_average_reward of -16.5 achieved!
episode_num 505, curr_reward: -18.0, best_reward: -11.0, running_avg_reward: -16.5, curr_epsilon: 0.1019
checkpointing current model weights. highest running_average_reward of -16.46 achieved!
                                        .
                                        .
                                        .
episode_num 1000, curr_reward: -4.0, best_reward: 13.0, running_avg_reward: -6.64, curr_epsilon: 0.0059
checkpointing current model weights. highest running_average_reward of -6.61 achieved!
episode_num 1001, curr_reward: -9.0, best_reward: 13.0, running_avg_reward: -6.61, curr_epsilon: 0.0059

episode_num 1002, curr_reward: -15.0, best_reward: 13.0, running_avg_reward: -6.72, curr_epsilon: 0.0059
episode_num 1003, curr_reward: -3.0, best_reward: 13.0, running_avg_reward: -6.66, curr_epsilon: 0.0059
episode_num 1004, curr_reward: -7.0, best_reward: 13.0, running_avg_reward: -6.72, curr_epsilon: 0.0059
episode_num 1005, curr_reward: -12.0, best_reward: 13.0, running_avg_reward: -6.69, curr_epsilon: 0.0059
                                        .
                                        .
                                        .
episode_num 1500, curr_reward: 11.0, best_reward: 17.0, running_avg_reward: -0.22, curr_epsilon: 0.005
checkpointing current model weights. highest running_average_reward of -0.05 achieved!
episode_num 1501, curr_reward: 7.0, best_reward: 17.0, running_avg_reward: -0.05, curr_epsilon: 0.005
checkpointing current model weights. highest running_average_reward of 0.01 achieved!
episode_num 1502, curr_reward: -1.0, best_reward: 17.0, running_avg_reward: 0.01, curr_epsilon: 0.005

checkpointing current model weights. highest running_average_reward of 0.11 achieved!
episode_num 1503, curr_reward: 3.0, best_reward: 17.0, running_avg_reward: 0.11, curr_epsilon: 0.005
checkpointing current model weights. highest running_average_reward of 0.2 achieved!
episode_num 1504, curr_reward: 2.0, best_reward: 17.0, running_avg_reward: 0.2, curr_epsilon: 0.005
episode_num 1505, curr_reward: -8.0, best_reward: 17.0, running_avg_reward: 0.19, curr_epsilon: 0.005
                                        .
                                        .
                                        .
episode_num 1000, curr_reward: -4.0, best_reward: 13.0, running_avg_reward: -6.64, curr_epsilon: 0.0059
checkpointing current model weights. highest running_average_reward of -6.61 achieved!
episode_num 1001, curr_reward: -9.0, best_reward: 13.0, running_avg_reward: -6.61, curr_epsilon: 0.0059

episode_num 1002, curr_reward: -15.0, best_reward: 13.0, running_avg_reward: -6.72, curr_epsilon: 0.0059
episode_num 1003, curr_reward: -3.0, best_reward: 13.0, running_avg_reward: -6.66, curr_epsilon: 0.0059
episode_num 1004, curr_reward: -7.0, best_reward: 13.0, running_avg_reward: -6.72, curr_epsilon: 0.0059
episode_num 1005, curr_reward: -12.0, best_reward: 13.0, running_avg_reward: -6.69, curr_epsilon: 0.0059
```


DQN training curve
total rewards from the current episode
running average of rewards from previous epsiodes
best reward obtained so far
reward points
episode number
20
10
0
−10
−20
0
250
500
750
1000
1250
1500
1750
2000

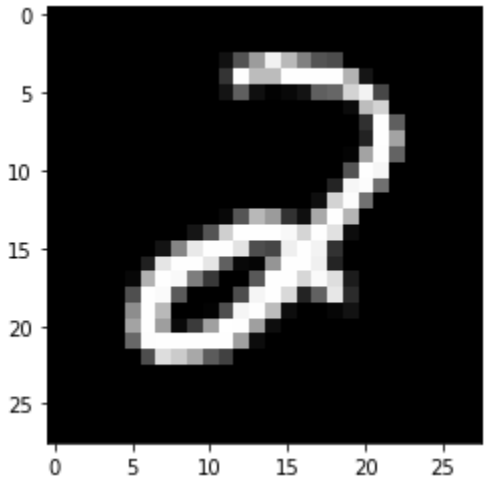
Chapter 10 : Operationalizing Pytorch Models into Production

```
[[-9.35050774e+00 -1.20893326e+01 -2.23922171e-03 -8.92477798e+00
  -9.81972313e+00 -1.33498535e+01 -9.04598618e+00 -1.44924192e+01
  -6.30233145e+00 -1.22827682e+01]]
```

Tracing	Scripting
• Dummy input is needed. • Records a fixed sequence of mathematical operations by passing the dummy input to the model. • Cannot handle multiple control flows (if-else) within the model forward pass. • Works even if the model has PyTorch functionalities that are not supported by TorchScript (`https://pytorch.org/docs/stable/jit_unsupported.html`).	• No need for dummy input. • Generates TorchScript code/graph by inspecting the `nn.Module` contents within the PyTorch code. • Useful in handling all types of control flows. • Scripting can work only if the PyTorch model does not contain any functionalities which are not supported by TorchScript.

```
<All keys matched successfully>
```

```
ConvNet(
  (cn1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
  (cn2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (dp1): Dropout2d(p=0.1, inplace=False)
  (dp2): Dropout2d(p=0.25, inplace=False)
  (fc1): Linear(in_features=4608, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```



```
output = run_model(input_tensor)
print(output)
print(type(output))
```

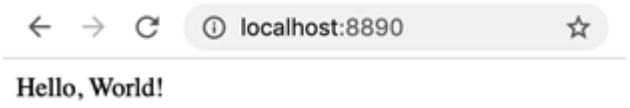
```
2
<class 'numpy.int64'>
```

```
[8.69212745e-05 5.61913612e-06 9.97763395e-01 1.33050999e-04
 5.43686365e-05 1.59305739e-06 1.17863165e-04 5.08185963e-07
 1.83202932e-03 4.63086781e-06]
```

```
final_output = post_process(output)
print(final_output)
print(type(final_output))
```

```
2
<class 'str'>
```

```
* Serving Flask app "example" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:8890/ (Press CTRL+C to quit)
```



Hello, World!

```
* Serving Flask app "server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8890/ (Press CTRL+C to quit)
```

Predicted digit : 2

```
Sending build context to Docker daemon  7.283MB
Step 1/9 : FROM python:3.8-slim
 ---> 62297c9f4e5c
Step 2/9 : RUN apt-get -q update && apt-get -q install -y wget
 ---> Using cache
 ---> e6142d540652
Step 3/9 : COPY ./server.py ./
 ---> Using cache
 ---> cb82fb5cb2e5
Step 4/9 : COPY ./requirements.txt ./
 ---> Using cache
 ---> faa39c98f044
Step 5/9 : RUN wget -q https://github.com/PacktPublishing/Mastering-PyTorch/raw/master/Chapter10/convnet.pth
 ---> Running in 198679553bac
Removing intermediate container 198679553bac
 ---> 8bddc82ccf1e
Step 6/9 : RUN wget -q https://github.com/PacktPublishing/Mastering-PyTorch/raw/master/Chapter10/digit_image.jpg
 ---> Running in 34836205c03d
Removing intermediate container 34836205c03d
 ---> 331c8447524a
Step 7/9 : RUN pip install -r requirements.txt
 ---> Running in e4bd0692812b
Collecting torch==1.5.0
  Downloading torch-1.5.0-cp38-cp38-manylinux1_x86_64.whl (752.0 MB)
Collecting torchvision==0.6.0
  Downloading torchvision-0.6.0-cp38-cp38-manylinux1_x86_64.whl (6.6 MB)
Collecting Pillow==6.2.2
  Downloading Pillow-6.2.2-cp38-cp38-manylinux1_x86_64.whl (2.1 MB)
Collecting Flask==1.1.1
  Downloading Flask-1.1.1-py2.py3-none-any.whl (94 kB)
Collecting numpy
  Downloading numpy-1.19.5-cp38-cp38-manylinux2010_x86_64.whl (14.9 MB)
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
Collecting Werkzeug>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux1_x86_64.whl (32 kB)
Building wheels for collected packages: future
  Building wheel for future (setup.py): started
  Building wheel for future (setup.py): finished with status 'done'
  Created wheel for future: filename=future-0.18.2-py3-none-any.whl size=491059 sha256=726027831b4159f39c497dee6280cf48539056b1b80d80fbf113330e7ea5af0d
  Stored in directory: /root/.cache/pip/wheels/8e/70/28/3d6ccd6e315f65f245da085482a2e1c7d14b90b30f239e2cf4
Successfully built future
Installing collected packages: numpy, future, torch, Pillow, torchvision, Werkzeug, itsdangerous, MarkupSafe, Jinja2, click, Flask
Successfully installed Flask-1.1.1 Jinja2-2.11.2 MarkupSafe-1.1.1 Pillow-6.2.2 Werkzeug-1.0.1 click-7.1.2 future-0.18.2 itsdangerous-1.1.0 numpy-1.19.5 torch-1.5.0 torchvision-0.6.0
WARNING: You are using pip version 20.2.3; however, version 20.3.3 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container e4bd0692812b
 ---> d7ecf4681868
Step 8/9 : USER root
 ---> Running in 90dac164d1bc
Removing intermediate container 90dac164d1bc
 ---> e4488fab0902
Step 9/9 : ENTRYPOINT ["python", "server.py"]
 ---> Running in a1d55f33a99f
Removing intermediate container a1d55f33a99f
 ---> 2de7d75aae1b
Successfully built 2de7d75aae1b
Successfully tagged digit_recognizer:latest
```

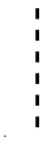
```
* Serving Flask app "server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8890/ (Press CTRL+C to quit)
```

Predicted digit : 2

```
Number of GPUs: 0
Number of CPUs: 8
Max heap size: 4096 M
Python executable: /Users/ashish.jha/opt/anaconda3/bin/python
Config file: N/A
Inference address: http://127.0.0.1:8080
Management address: http://127.0.0.1:8081
Metrics address: http://127.0.0.1:8082
```

```
{
  "models": [
    {
      "modelName": "convnet",
      "modelUrl": "convnet.mar"
    }
  ]
}
```

```
graph(%self.1 : __torch__.torch.nn.modules.module.___torch_mangle_6.Module,
      %input.1 : Float(1, 1, 28, 28)):
  %113 : __torch__.torch.nn.modules.module.___torch_mangle_5.Module = prim::GetAttr[name="fc2"](%self.1)
  %110 : __torch__.torch.nn.modules.module.___torch_mangle_3.Module = prim::GetAttr[name="dp2"](%self.1)
  %109 : __torch__.torch.nn.modules.module.___torch_mangle_4.Module = prim::GetAttr[name="fc1"](%self.1)
  %106 : __torch__.torch.nn.modules.module.___torch_mangle_2.Module = prim::GetAttr[name="dp1"](%self.1)
  %105 : __torch__.torch.nn.modules.module.___torch_mangle_1.Module = prim::GetAttr[name="cn2"](%self.1)
  %102 : __torch__.torch.nn.modules.module.Module = prim::GetAttr[name="cn1"](%self.1)
  %120 : Tensor = prim::CallMethod[name="forward"](%102, %input.1)
  %input.3 : Float(1, 16, 26, 26) = aten::relu(%120) # /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-packages/to
rch/nn/functional.py:914:0
  %121 : Tensor = prim::CallMethod[name="forward"](%105, %input.3)
  %input.5 : Float(1, 32, 24, 24) = aten::relu(%121) # /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-packages/to
rch/nn/functional.py:914:0
```



```
  %input.9 : Float(1, 64) = aten::relu(%123) # /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-packages/torch/nn/f
unctional.py:914:0
  %124 : Tensor = prim::CallMethod[name="forward"](%110, %input.9)
  %125 : Tensor = prim::CallMethod[name="forward"](%113, %124)
  %91 : int = prim::Constant[value=1]() # /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-packages/torch/nn/functi
onal.py:1317:0
  %92 : None = prim::Constant()
  %93 : Float(1, 10) = aten::log_softmax(%125, %91, %92) # /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-package
s/torch/nn/functional.py:1317:0
  return (%93)
```

```
  def forward(self,
      input: Tensor) -> Tensor:
    _0 = self.fc2
    _1 = self.dp2
    _2 = self.fc1
    _3 = self.dp1
    _4 = self.cn2
    input0 = torch.relu((self.cn1).forward(input, ))
    input1 = torch.relu((_4).forward(input0, ))
    input2 = torch.max_pool2d(input1, [2, 2], annotate(List[int], []), [0, 0], [1, 1], False)
    input3 = torch.flatten((_3).forward(input2, ), 1, -1)
    input4 = torch.relu((_2).forward(input3, ))
    _5 = (_0).forward((_1).forward(input4, ), )
    return torch.log_softmax(_5, 1, None)
```

```
tensor([[-9.3505e+00, -1.2089e+01, -2.2391e-03, -8.9248e+00, -9.8197e+00,
         -1.3350e+01, -9.0460e+00, -1.4492e+01, -6.3023e+00, -1.2283e+01]])
```

```
graph(%self : __torch__.ConvNet,
      %x.1 : Tensor):
  %51 : Function = prim::Constant[name="log_softmax"]()
  %49 : int = prim::Constant[value=3]()
  %33 : int = prim::Constant[value=-1]()
  %26 : Function = prim::Constant[name="_max_pool2d"]()
  %20 : int = prim::Constant[value=0]()
  %19 : None = prim::Constant()
  %7 : Function = prim::Constant[name="relu"]()
  %6 : bool = prim::Constant[value=0]()
                                                    :
                                                    :
                                                    :
  %x.19 : Tensor = prim::CallFunction(%7, %x.17, %6) # <ipython-input-3-936a1c5cab85>:20:12
  %42 : __torch__.torch.nn.modules.dropout.___torch_mangle_1.Dropout2d = prim::GetAttr[name
="dp2"](%self)
  %x.21 : Tensor = prim::CallMethod[name="forward"](%42, %x.19) # <ipython-input-3-936a1c5cab
85>:21:12
  %45 : __torch__.torch.nn.modules.linear.___torch_mangle_2.Linear = prim::GetAttr[name="fc
2"](%self)
  %x.23 : Tensor = prim::CallMethod[name="forward"](%45, %x.21) # <ipython-input-3-936a1c5cab
85>:22:12
  %op.1 : Tensor = prim::CallFunction(%51, %x.23, %32, %49, %19) # <ipython-input-3-936a1c5ca
b85>:23:13
  return (%op.1)
```

```
                  def forward(self,
                      x: Tensor) -> Tensor:
                    _0 = __torch__.torch.nn.functional.___torch_mangle_12.relu
                    _1 = __torch__.torch.nn.functional._max_pool2d
                    _2 = __torch__.torch.nn.functional.___torch_mangle_13.relu
                    _3 = __torch__.torch.nn.functional.log_softmax
                    x0 = (self.cn1).forward(x, )
                    x1 = __torch__.torch.nn.functional.relu(x0, False, )
                    x2 = (self.cn2).forward(x1, )
                    x3 = _0(x2, False, )
                    x4 = _1(x3, [2, 2], None, [0, 0], [1, 1], False, False, )
                    x5 = (self.dp1).forward(x4, )
                    x6 = torch.flatten(x5, 1, -1)
                    x7 = (self.fc1).forward(x6, )
                    x8 = _2(x7, False, )
                    x9 = (self.dp2).forward(x8, )
                    x10 = (self.fc2).forward(x9, )
                    return _3(x10, 1, 3, None, )
```

```
-- The C compiler identification is AppleClang 10.0.1.10010046
-- The CXX compiler identification is AppleClang 10.0.1.10010046
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Found Torch: /Users/ashish.jha/opt/anaconda3/lib/python3.7/site-packages/torch/lib/libtorch.dylib
-- Found OpenCV: /Users/ashish.jha/code/personal/Mastering-PyTorch/Chapter10/cpp_convnet/build_opencv (found version "4.5.0")
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/ashish.jha/code/personal/Mastering-PyTorch/Chapter10/cpp_convnet/build
```

```
Scanning dependencies of target cpp_convnet
[ 50%] Building CXX object CMakeFiles/cpp_convnet.dir/cpp_convnet.cpp.o
[100%] Linking CXX executable cpp_convnet
[100%] Built target cpp_convnet
```

```
 2
[ CPULongType{1} ]
```

```
(<tf.Tensor 'Const:0' shape=(16,) dtype=float32>,)
(<tf.Tensor 'Const_1:0' shape=(16, 1, 3, 3) dtype=float32>,)
(<tf.Tensor 'Const_2:0' shape=(32,) dtype=float32>,)
(<tf.Tensor 'Const_3:0' shape=(32, 16, 3, 3) dtype=float32>,)
(<tf.Tensor 'Const_4:0' shape=(64,) dtype=float32>,)
(<tf.Tensor 'Const_5:0' shape=(64, 4608) dtype=float32>,)
(<tf.Tensor 'Const_6:0' shape=(10,) dtype=float32>,)
(<tf.Tensor 'Const_7:0' shape=(10, 64) dtype=float32>,)
(<tf.Tensor 'input.1:0' shape=(1, 1, 28, 28) dtype=float32>,)
(<tf.Tensor 'transpose/perm:0' shape=(4,) dtype=int32>,)
(<tf.Tensor 'transpose:0' shape=(3, 3, 1, 16) dtype=float32>,)
                              :
                              :
                              :
                              :
(<tf.Tensor 'mul_2/x:0' shape=() dtype=float32>,)
(<tf.Tensor 'mul_2:0' shape=(1, 10) dtype=float32>,)
(<tf.Tensor 'mul_3/x:0' shape=() dtype=float32>,)
(<tf.Tensor 'mul_3:0' shape=(10,) dtype=float32>,)
(<tf.Tensor 'add_3:0' shape=(1, 10) dtype=float32>,)
(<tf.Tensor '18:0' shape=(1, 10) dtype=float32>,)
```

Chapter 11: Distributed Training

```
epoch: 0 [0/469 (0%)]    training loss: 2.314901
epoch: 0 [10/469 (2%)]   training loss: 1.642720
epoch: 0 [20/469 (4%)]   training loss: 0.802527
epoch: 0 [30/469 (6%)]   training loss: 0.679492
epoch: 0 [40/469 (9%)]   training loss: 0.300678
epoch: 0 [50/469 (11%)]  training loss: 1.030731
                         :
                         :
                         :
                         :
epoch: 0 [430/469 (92%)]         training loss: 0.100122
epoch: 0 [440/469 (94%)]         training loss: 0.253491
epoch: 0 [450/469 (96%)]         training loss: 0.027886
epoch: 0 [460/469 (98%)]         training loss: 0.120182
Finished training in 32.70223307609558 secs
```

```
epoch: 0 [0/469 (0%)]    training loss: 2.308408
epoch: 0 [10/469 (2%)]   training loss: 1.772532
epoch: 0 [20/469 (4%)]   training loss: 0.953913
epoch: 0 [30/469 (6%)]   training loss: 0.694977
epoch: 0 [40/469 (9%)]   training loss: 0.481864
epoch: 0 [50/469 (11%)]  training loss: 0.394739
epoch: 0 [60/469 (13%)]  training loss: 0.415441
                         :
                         :
                         :
                         :
epoch: 0 [430/469 (92%)]         training loss: 0.137537
epoch: 0 [440/469 (94%)]         training loss: 0.088957
epoch: 0 [450/469 (96%)]         training loss: 0.040298
epoch: 0 [460/469 (98%)]         training loss: 0.136536
Finished training in 50.57237482070923 secs
```

```
Hardware Overview:

  Model Name: MacBook Pro
  Model Identifier: MacBookPro15,2
  Processor Name: Intel Core i5
  Processor Speed: 2.4 GHz
  Number of Processors: 1
  Total Number of Cores: 4
  L2 Cache (per Core): 256 KB
  L3 Cache: 6 MB
  Hyper-Threading Technology: Enabled
  Memory: 16 GB
```

```
epoch: 0 [0/469 (0%)]    training loss: 2.310592
epoch: 0 [10/469 (2%)]   training loss: 1.276357
epoch: 0 [20/469 (4%)]   training loss: 0.693506
epoch: 0 [30/469 (6%)]   training loss: 0.666963
epoch: 0 [40/469 (9%)]   training loss: 0.318174
epoch: 0 [50/469 (11%)]  training loss: 0.567527
                         :
                         :
                         :
                         :

epoch: 0 [430/469 (92%)]         training loss: 0.084474
epoch: 0 [440/469 (94%)]         training loss: 0.140898
epoch: 0 [450/469 (96%)]         training loss: 0.154369
epoch: 0 [460/469 (98%)]         training loss: 0.110312
Finished training in 44.398102045059204 secs
```

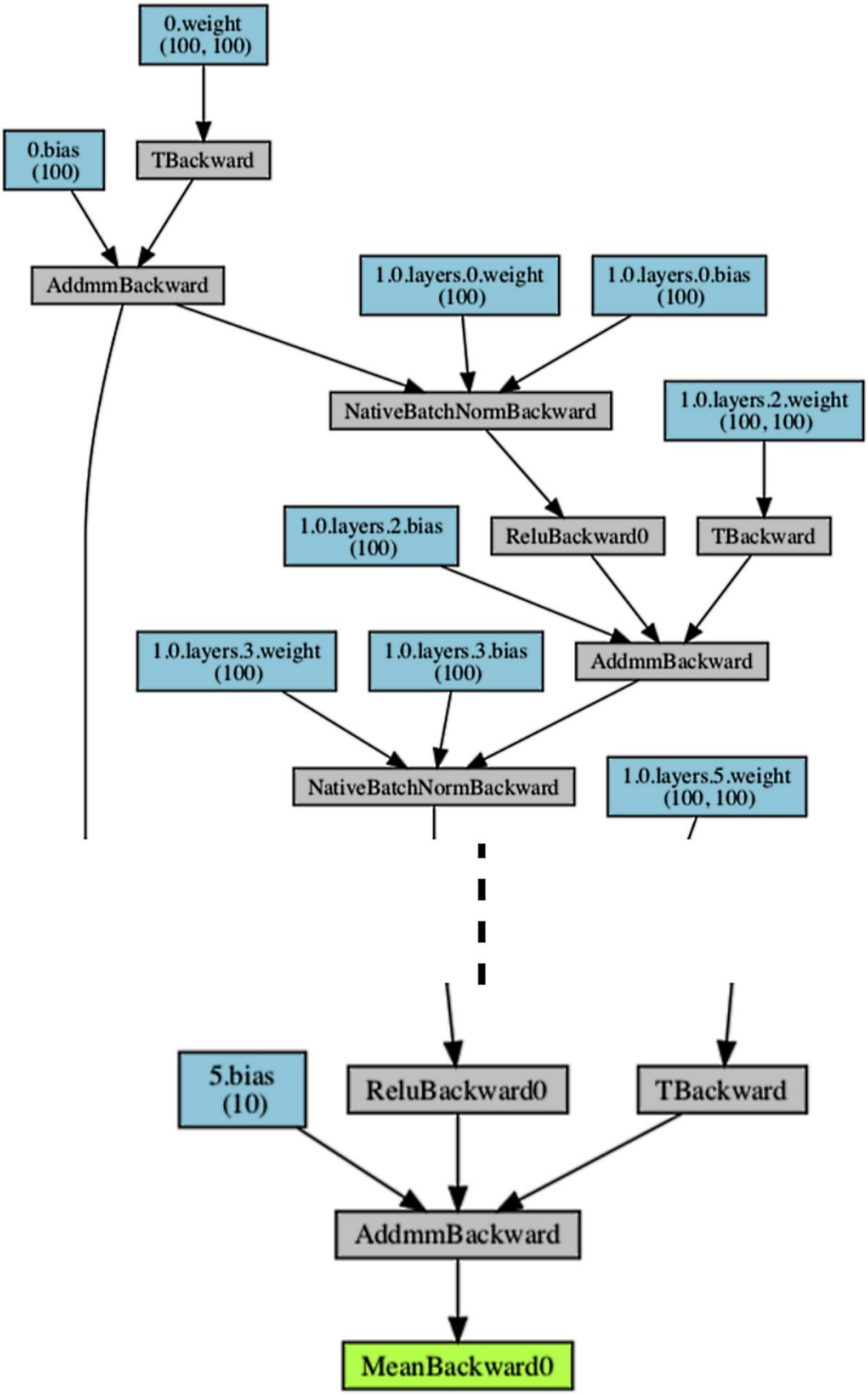
```
epoch: 0 [0/469 (0%)]    training loss: 2.309348
epoch: 0 [10/469 (2%)]   training loss: 1.524053
epoch: 0 [20/469 (4%)]   training loss: 0.993402
epoch: 0 [30/469 (6%)]   training loss: 0.777355
epoch: 0 [40/469 (9%)]   training loss: 0.407441
epoch: 0 [50/469 (11%)]  training loss: 0.655984
                         :
                         :
                         :
                         :

epoch: 0 [420/469 (90%)]         training loss: 0.179646
epoch: 0 [430/469 (92%)]         training loss: 0.059710
epoch: 0 [440/469 (94%)]         training loss: 0.052976
epoch: 0 [450/469 (96%)]         training loss: 0.039953
epoch: 0 [460/469 (98%)]         training loss: 0.181595
Finished training in 30.58652114868164 secs
```

```
{'optimized_hyperparameter_config': {'CreateDataLoader:batch_size': 125,
  'Imputation:strategy': 'median',
  'InitializationSelector:initialization_method': 'default',
  'InitializationSelector:initializer:initialize_bias': 'No',
  'LearningrateSchedulerSelector:lr_scheduler': 'cosine_annealing',
  'LossModuleSelector:loss_module': 'cross_entropy_weighted',
  'NetworkSelector:network': 'shapedresnet',
  'NormalizationStrategySelector:normalization_strategy': 'standardize',
  'OptimizerSelector:optimizer': 'sgd',
  'PreprocessorSelector:preprocessor': 'truncated_svd',
  'ResamplingStrategySelector:over_sampling_method': 'none',
  'ResamplingStrategySelector:target_size_strategy': 'none',
  'ResamplingStrategySelector:under_sampling_method': 'none',
  'TrainNode:batch_loss_computation_technique': 'standard',
  'LearningrateSchedulerSelector:cosine_annealing:T_max': 10,
  'LearningrateSchedulerSelector:cosine_annealing:eta_min': 2,
  'NetworkSelector:shapedresnet:activation': 'relu',
  'NetworkSelector:shapedresnet:blocks_per_group': 4,
  'NetworkSelector:shapedresnet:max_units': 13,
  'NetworkSelector:shapedresnet:num_groups': 2,
  'NetworkSelector:shapedresnet:resnet_shape': 'brick',
  'NetworkSelector:shapedresnet:use_dropout': 0,
  'NetworkSelector:shapedresnet:use_shake_drop': 0,
  'NetworkSelector:shapedresnet:use_shake_shake': 0,
  'OptimizerSelector:sgd:learning_rate': 0.06829146967649465,
  'OptimizerSelector:sgd:momentum': 0.9343847098348538,
  'OptimizerSelector:sgd:weight_decay': 0.0002425066735211845,
  'PreprocessorSelector:truncated_svd:target_dim': 100},
 'budget': 40.0,
 'loss': -96.45,
 'info': {'loss': 0.12337125303244502,
  'model_parameters': 176110.0,
  'train_accuracy': 96.28550185873605,
  'lr_scheduler_converged': 0.0,
  'lr': 0.06829146967649465,
  'val_accuracy': 96.45}}
```


Accuracy score 0.964

```
pytorch_model = autoPyTorch.get_pytorch_model()
print(pytorch_model)
```

```
Sequential(
  (0): Linear(in_features=100, out_features=100, bias=True)
  (1): Sequential(
    (0): ResBlock(
      (layers): Sequential(
        (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (1): ReLU()
        (2): Linear(in_features=100, out_features=100, bias=True)
        (3): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU()
        (5): Linear(in_features=100, out_features=100, bias=True)
      )
    )
    (1): ResBlock(
      (layers): Sequential(
        (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (1): ReLU()
        (2): Linear(in_features=100, out_features=100, bias=True)
        (3): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU()
        (5): Linear(in_features=100, out_features=100, bias=True)
      )
    )
    (2): ResBlock(
      (layers): Sequential(
        (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (1): ReLU()
        (2): Linear(in_features=100, out_features=100, bias=True)
        (3): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU()
        (5): Linear(in_features=100, out_features=100, bias=True)
      )
    )
    (3): ResBlock(
                                                  .
                                                  .
                                                  .
                                                  .
     )
     (3): ResBlock(
       (layers): Sequential(
         (0): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
         (1): ReLU()
         (2): Linear(in_features=100, out_features=100, bias=True)
         (3): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
         (4): ReLU()
         (5): Linear(in_features=100, out_features=100, bias=True)
       )
     )
   )
   (3): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (4): ReLU()
   (5): Linear(in_features=100, out_features=10, bias=True)
 )
```

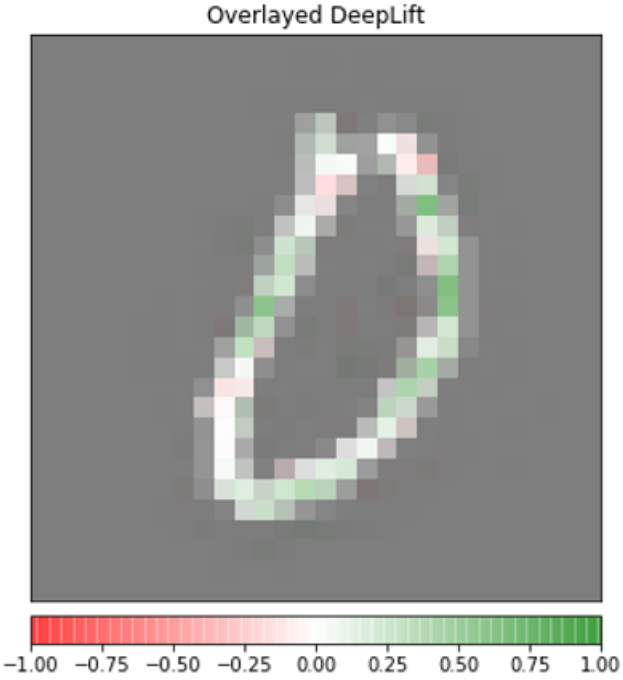

0.weight
(100, 100)
0.bias
(100)
TBackward
AddmmBackward
1.0.layers.0.weight
(100)
1.0.layers.0.bias
(100)
NativeBatchNormBackward
1.0.layers.2.weight
(100, 100)
1.0.layers.2.bias
(100)
ReluBackward0
TBackward
1.0.layers.3.weight
(100)
1.0.layers.3.bias
(100)
AddmmBackward
NativeBatchNormBackward
1.0.layers.5.weight
(100, 100)
5.bias
(10)
ReluBackward0
TBackward
AddmmBackward
MeanBackward0

```
Configuration space object:
  Hyperparameters:
    CreateDataLoader:batch_size, Type: Constant, Value: 125
    Imputation:strategy, Type: Categorical, Choices: {median}, Default: median
    InitializationSelector:initialization_method, Type: Categorical, Choices: {default}, Default: default
    InitializationSelector:initializer:initialize_bias, Type: Constant, Value: No
    LearningrateSchedulerSelector:cosine_annealing:T_max, Type: Constant, Value: 10
    LearningrateSchedulerSelector:cosine_annealing:eta_min, Type: Constant, Value: 2
    LearningrateSchedulerSelector:lr_scheduler, Type: Categorical, Choices: {cosine_annealing}, Default: cosine_annea
ling
    LossModuleSelector:loss_module, Type: Categorical, Choices: {cross_entropy_weighted}, Default: cross_entropy_weig
hted
    NetworkSelector:network, Type: Categorical, Choices: {shapedresnet}, Default: shapedresnet
    NetworkSelector:shapedresnet:activation, Type: Constant, Value: relu
    NetworkSelector:shapedresnet:blocks_per_group, Type: UniformInteger, Range: [1, 4], Default: 2
    NetworkSelector:shapedresnet:max_units, Type: UniformInteger, Range: [10, 1024], Default: 101, on log-scale
    NetworkSelector:shapedresnet:num_groups, Type: UniformInteger, Range: [1, 9], Default: 5
    NetworkSelector:shapedresnet:resnet_shape, Type: Constant, Value: brick
    NetworkSelector:shapedresnet:use_dropout, Type: Constant, Value: 0
    NetworkSelector:shapedresnet:use_shake_drop, Type: Constant, Value: 0
    NetworkSelector:shapedresnet:use_shake_shake, Type: Constant, Value: 0
    NormalizationStrategySelector:normalization_strategy, Type: Categorical, Choices: {standardize}, Default: standar
dize
    OptimizerSelector:optimizer, Type: Categorical, Choices: {sgd}, Default: sgd
    OptimizerSelector:sgd:learning_rate, Type: UniformFloat, Range: [0.0001, 0.1], Default: 0.0031622777, on log-scal
e
    OptimizerSelector:sgd:momentum, Type: UniformFloat, Range: [0.1, 0.999], Default: 0.5495
    OptimizerSelector:sgd:weight_decay, Type: UniformFloat, Range: [1e-05, 0.1], Default: 0.050005
    PreprocessorSelector:preprocessor, Type: Categorical, Choices: {truncated_svd}, Default: truncated_svd
    PreprocessorSelector:truncated_svd:target_dim, Type: Constant, Value: 100
    ResamplingStrategySelector:over_sampling_method, Type: Categorical, Choices: {none}, Default: none
    ResamplingStrategySelector:target_size_strategy, Type: Categorical, Choices: {none}, Default: none
    ResamplingStrategySelector:under_sampling_method, Type: Categorical, Choices: {none}, Default: none
    TrainNode:batch_loss_computation_technique, Type: Categorical, Choices: {standard}, Default: standard
  Conditions:
    LearningrateSchedulerSelector:cosine_annealing:T_max | LearningrateSchedulerSelector:lr_scheduler == 'cosine_anne
aling'
    LearningrateSchedulerSelector:cosine_annealing:eta_min | LearningrateSchedulerSelector:lr_scheduler == 'cosine_an
nealing'
    NetworkSelector:shapedresnet:activation | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:blocks_per_group | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:max_units | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:num_groups | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:resnet_shape | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:use_dropout | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:use_shake_drop | NetworkSelector:network == 'shapedresnet'
    NetworkSelector:shapedresnet:use_shake_shake | NetworkSelector:network == 'shapedresnet'
    OptimizerSelector:sgd:learning_rate | OptimizerSelector:optimizer == 'sgd'
    OptimizerSelector:sgd:momentum | OptimizerSelector:optimizer == 'sgd'
    OptimizerSelector:sgd:weight_decay | OptimizerSelector:optimizer == 'sgd'
    PreprocessorSelector:truncated_svd:target_dim | PreprocessorSelector:preprocessor == 'truncated_svd'
```

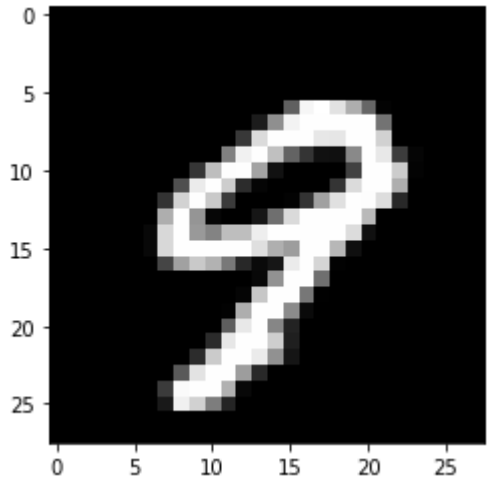
```
[I 2020-10-24 18:39:34,357] A new study created in memory with name: mastering_pytorch

epoch: 1 [0/60000 (0%)]  training loss: 2.314928
epoch: 1 [16000/60000 (27%)]     training loss: 2.339143
epoch: 1 [32000/60000 (53%)]     training loss: 2.554311
epoch: 1 [48000/60000 (80%)]     training loss: 2.392770

Test dataset: Overall Loss: 2.4598, Overall Accuracy: 974/10000 (10%)

epoch: 2 [0/60000 (0%)]  training loss: 2.352818
epoch: 2 [16000/60000 (27%)]     training loss: 2.425988
epoch: 2 [32000/60000 (53%)]     training loss: 2.432955
epoch: 2 [48000/60000 (80%)]     training loss: 2.497166

[I 2020-10-24 18:44:51,667] Trial 0 finished with value: 9.82 and parameters: {'num_conv_layers': 4, 'num_fc_layers':
2, 'conv_depth_0': 20, 'conv_depth_1': 18, 'conv_depth_2': 38, 'conv_depth_3': 27, 'conv_dropout_3': 0.18560304003563
005, 'fc_output_feat_0': 54, 'fc_dropout_0': 0.18233257074201586, 'fc_output_feat_1': 55, 'fc_dropout_1': 0.104182596
77735323, 'optimizer': 'RMSprop', 'lr': 0.49822431360836333}. Best is trial 0 with value: 9.82.

[I 2020-10-24 18:46:24,551] Trial 1 finished with value: 95.68 and parameters: {'num_conv_layers': 1, 'num_fc_layer
s': 2, 'conv_depth_0': 39, 'conv_dropout_0': 0.3950204757059781, 'fc_output_feat_0': 17, 'fc_dropout_0': 0.3760852329
345368, 'fc_output_feat_1': 40, 'fc_dropout_1': 0.29727560678671294, 'optimizer': 'Adadelta', 'lr': 0.254984294053231
25}. Best is trial 1 with value: 95.68.

[I 2020-10-24 18:51:37,575] Trial 2 finished with value: 98.77 and parameters: {'num_conv_layers': 3, 'num_fc_layer
s': 2, 'conv_depth_0': 27, 'conv_depth_1': 28, 'conv_depth_2': 46, 'conv_dropout_2': 0.3274565117338556, 'fc_output_f
eat_0': 57, 'fc_dropout_0': 0.12348496153785013, 'fc_output_feat_1': 54, 'fc_dropout_1': 0.36784682560478876, 'optimi
zer': 'Adadelta', 'lr': 0.4290610978292583}. Best is trial 2 with value: 98.77.

[I 2020-10-24 18:55:41,400] Trial 3 finished with value: 98.28 and parameters: {'num_conv_layers': 2, 'num_fc_layer
s': 1, 'conv_depth_0': 38, 'conv_depth_1': 40, 'conv_dropout_1': 0.3592746030824463, 'fc_output_feat_0': 20, 'fc_drop
out_0': 0.22476024022504099, 'optimizer': 'Adadelta', 'lr': 0.3167228174356792}. Best is trial 2 with value: 98.77.

[I 2020-10-24 18:59:54,755] Trial 4 finished with value: 10.28 and parameters: {'num_conv_layers': 2, 'num_fc_layer
s': 2, 'conv_depth_0': 26, 'conv_depth_1': 50, 'conv_dropout_1': 0.30220610162727457, 'fc_output_feat_0': 42, 'fc_dro
pout_0': 0.1561741472895425, 'fc_output_feat_1': 33, 'fc_dropout_1': 0.31642189637209367, 'optimizer': 'RMSprop', 'l
r': 0.45189990541514835}. Best is trial 2 with value: 98.77.

[I 2020-10-24 19:02:39,390] Trial 5 finished with value: 98.12 and parameters: {'num_conv_layers': 2, 'num_fc_layer
s': 1, 'conv_depth_0': 31, 'conv_depth_1': 22, 'conv_dropout_1': 0.3612944916702828, 'fc_output_feat_0': 25, 'fc_drop
out_0': 0.2839369529837842, 'optimizer': 'SGD', 'lr': 0.11490140528643872}. Best is trial 2 with value: 98.77.

[I 2020-10-24 19:06:33,825] Trial 6 finished with value: 98.29 and parameters: {'num_conv_layers': 2, 'num_fc_layer
s': 2, 'conv_depth_0': 24, 'conv_depth_1': 55, 'conv_dropout_1': 0.34239043023224586, 'fc_output_feat_0': 35, 'fc_dro
pout_0': 0.17065510224232447, 'fc_output_feat_1': 46, 'fc_dropout_1': 0.19804499857448277, 'optimizer': 'Adadelta',
'lr': 0.42138811722164293}. Best is trial 2 with value: 98.77.

[I 2020-10-24 19:09:33,855] Trial 7 pruned.

[I 2020-10-24 19:10:33,804] Trial 8 pruned.

[I 2020-10-24 19:15:36,906] Trial 9 pruned.
```

```
results:
num_trials_conducted:  10
num_trials_pruned:  3
num_trials_completed:  7
results from best trial:
accuracy:  98.77
hyperparameters:
num_conv_layers: 3
num_fc_layers: 2
conv_depth_0: 27
conv_depth_1: 28
conv_depth_2: 46
conv_dropout_2: 0.3274565117338556
fc_output_feat_0: 57
fc_dropout_0: 0.12348496153785013
fc_output_feat_1: 54
fc_dropout_1: 0.36784682560478876
optimizer: Adadelta
lr: 0.4290610978292583
```

Chapter 13: PyTorch and Explainable AI

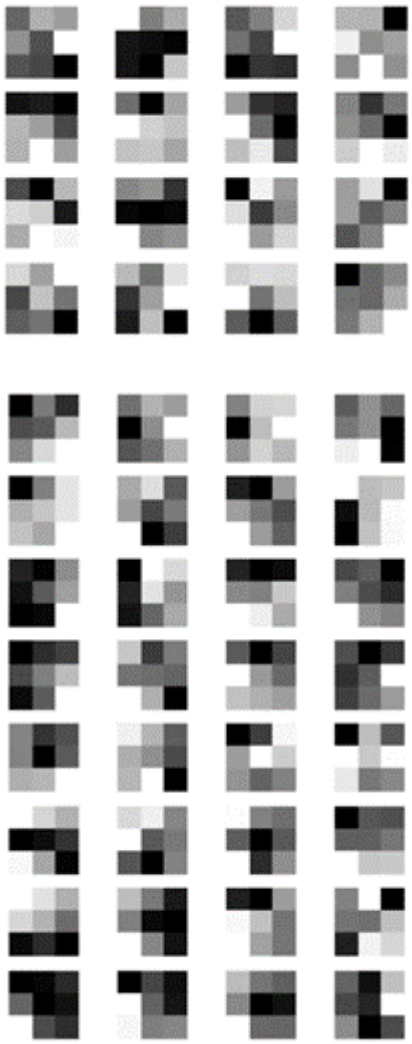



```
epoch: 1 [0/60000 (0%)]  training loss: 2.324445
epoch: 1 [320/60000 (1%)]        training loss: 1.727462
epoch: 1 [640/60000 (1%)]        training loss: 1.428922
epoch: 1 [960/60000 (2%)]        training loss: 0.717944
epoch: 1 [1280/60000 (2%)]       training loss: 0.572199
                                 :
                                 :
                                 :
epoch: 19 [58880/60000 (98%)]    training loss: 0.016509
epoch: 19 [59200/60000 (99%)]    training loss: 0.118218
epoch: 19 [59520/60000 (99%)]    training loss: 0.000097
epoch: 19 [59840/60000 (100%)]   training loss: 0.000271

Test dataset: Overall Loss: 0.0387, Overall Accuracy: 9910/10000 (99%)
```



```
Model prediction is : 9
Ground truth is : 9
```

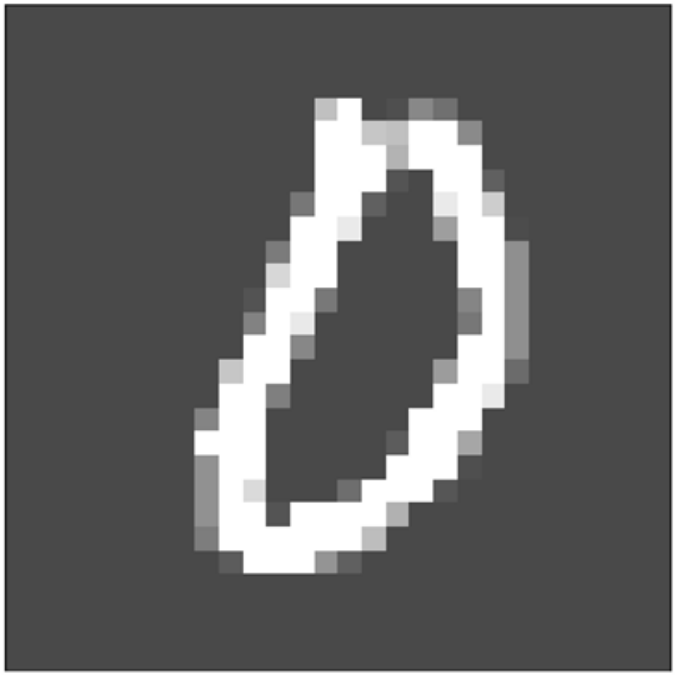
```
[Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1)),
 Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)),
 Dropout2d(p=0.1, inplace=False),
 Dropout2d(p=0.25, inplace=False),
 Linear(in_features=4608, out_features=64, bias=True),
 Linear(in_features=64, out_features=10, bias=True)]
```



```
torch.Size([16, 26, 26])
```

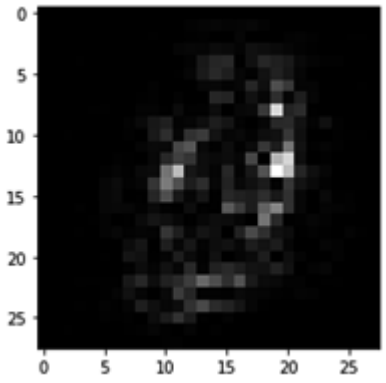
```
torch.Size([32, 24, 24])
```

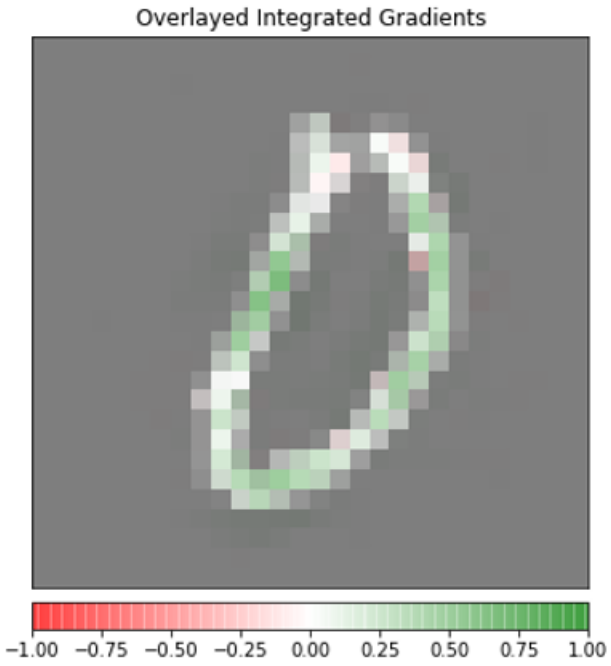
Original Image

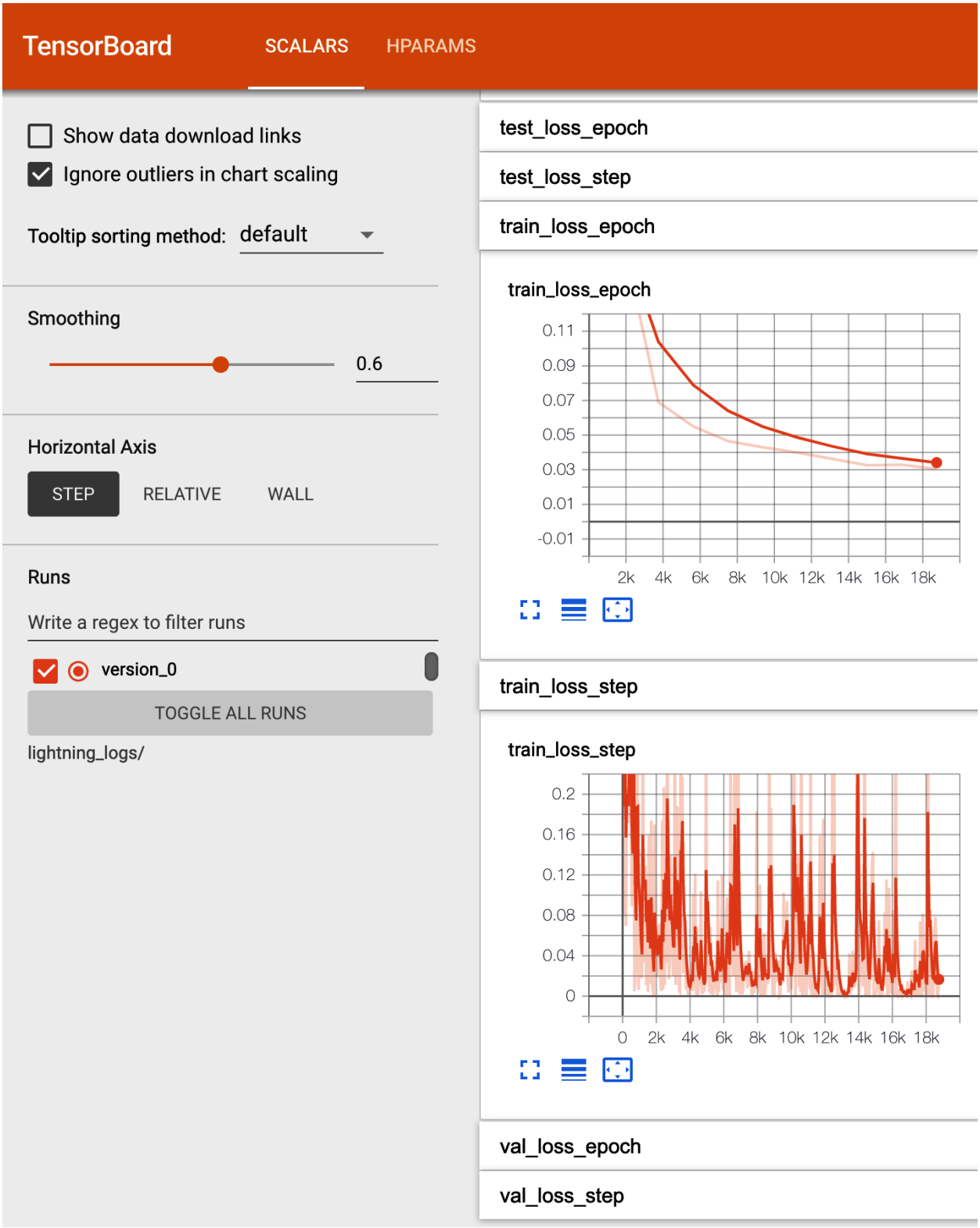


Original Image


Overlayed Gradients
0.0
0.2
0.4
0.6
0.8
1.0

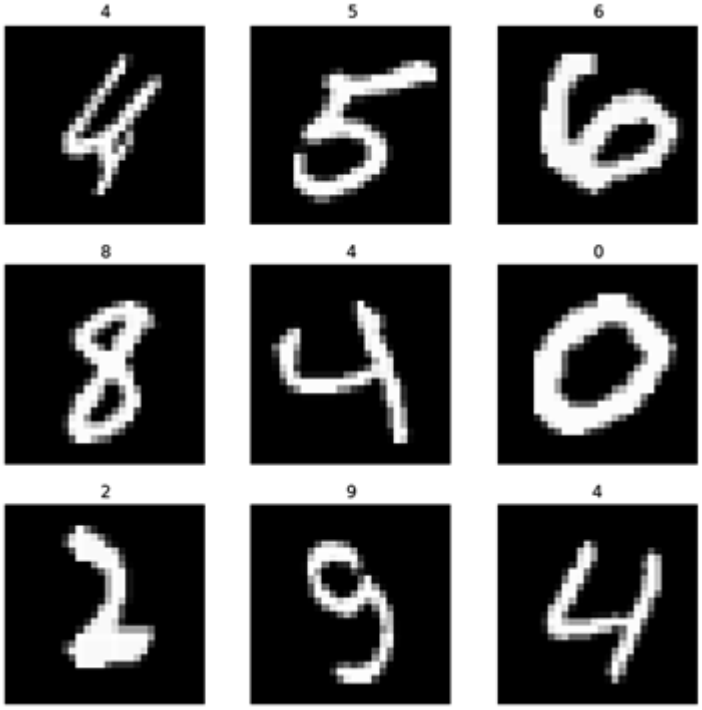

0
5
10
15
20
25
0
5
10
15
20
25


Overlayed Integrated Gradients
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

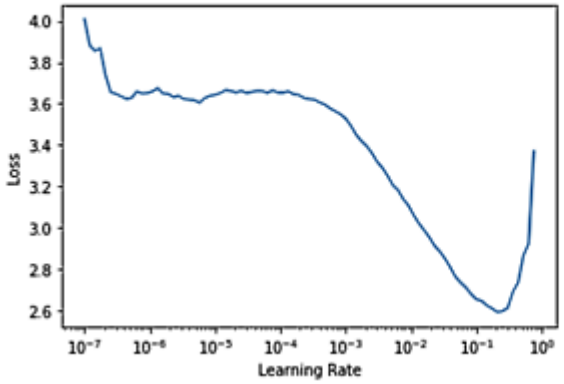


/Users/ashish.jha/.fastai/data/mnist_png

```
60000
/Users/ashish.jha/.fastai/data/mnist_png/training/9/36655.png
```

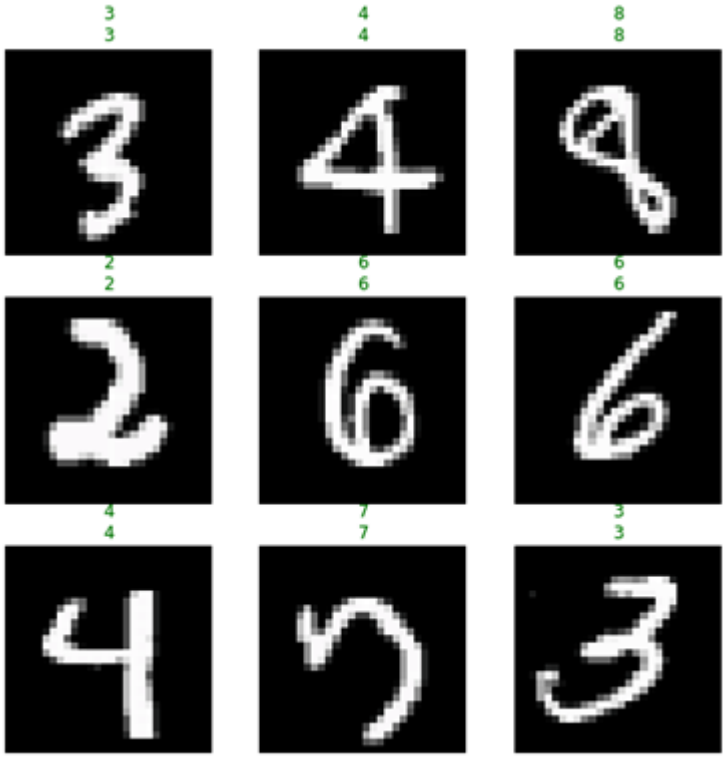



```
SuggestedLRs(lr_min=0.02089296132326126, lr_steep=0.009120108559727669)
```

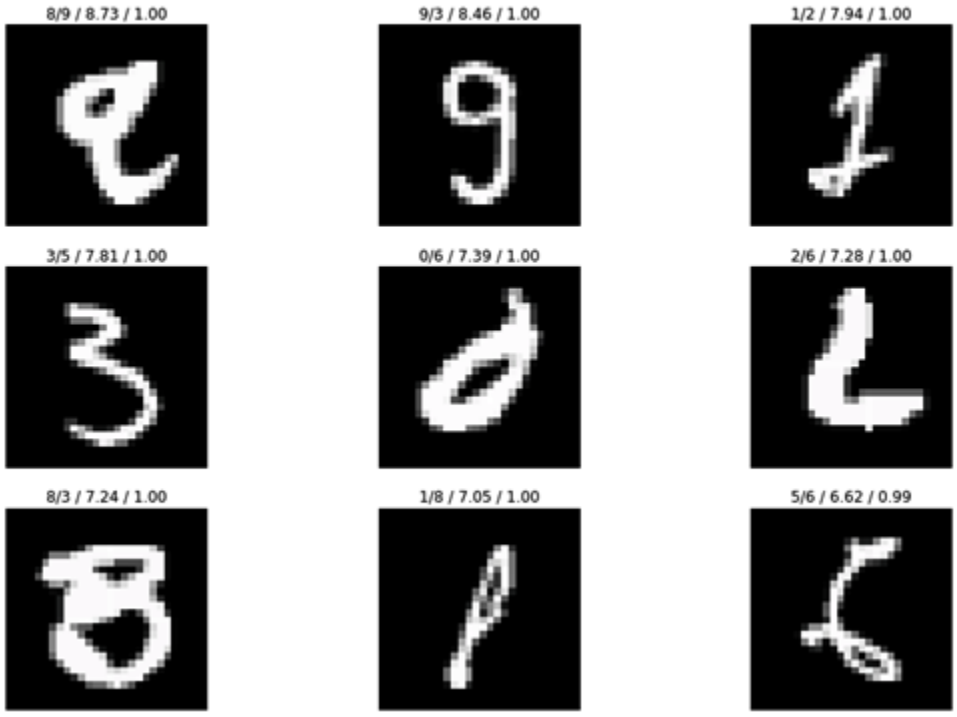


epoch	train_loss	valid_loss	accuracy	time
0	0.281835	0.199095	0.946417	08:30

epoch	train_loss	valid_loss	accuracy	time
0	0.122436	0.080322	0.982583	10:24
1	0.033702	0.027708	0.991833	08:32



Prediction/Actual/Loss/Probability



```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores

  | Name | Type      | Params
-----------------------------------
0 | cn1  | Conv2d    | 160
1 | cn2  | Conv2d    | 4 K
2 | dp1  | Dropout2d | 0
3 | dp2  | Dropout2d | 0
4 | fc1  | Linear    | 294 K
5 | fc2  | Linear    | 650
```

Validation sanity check: 0% 0/2 [01:04<?, ?it/s]

Epoch 9: 99%

3720/3750 [01:28<00:00, 42.13it/s, loss=0.066, v_num=0, val_loss_step=0.000419, train_loss_step=0.0105, val_loss_epoch=0.0109, train_loss_epoch=0.0329]

Testing: 96% 300/313 [00:03<00:00, 92.65it/s]

```
--------------------------------------------------------------------------------
DATALOADER:0 TEST RESULTS
{'test_loss': tensor(4.2241e-05),
 'test_loss_epoch': tensor(0.0361),
 'train_loss': tensor(0.0105),
 'train_loss_epoch': tensor(0.0304),
 'train_loss_step': tensor(0.0105),
 'val_loss': tensor(0.0004),
 'val_loss_epoch': tensor(0.0109)}
--------------------------------------------------------------------------------
```

Reusing TensorBoard on port 6007 (pid 21690), started 22:03:23 ago. (Use '!kill 21690' to kill it.)