Assignment 3

Assignment of
ELL 784: Introduction to Machine Learning

by

Lovlesh Roy

2021ESN2618

Under the Guidance of
Prof. Sumeet Agarwal

INDIAN INSTITUTE OF TECHNOLOGY DELHI

NEW DELHI - 110016

OCTOBER 2022

Table of Contents

1. **Part 1** 3
2. **Part 2** 7

Part 1

➢ Neural Networks

- **Given Dataset**

```
df.head(5)
```

	1	2	3	4	5	6	7	8	9	10	...	776	777	778	779	780	781	782	783	784	785
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

Fig 1: Given dataset

- **Printing the digit**

```
some_digit = np.array(X[2003:2004])
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap = mpl.cm.binary,
           interpolation="nearest")
plt.axis("off")
plt.savefig('6.png', bbox_inches='tight')
plt.show()
```

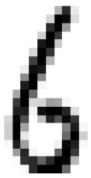

Fig 2: Printing of row in form of image

- **Building the model**

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dropout (Dropout)           (None, 784)               0

 dense (Dense)               (None, 900)               706500

 dropout_1 (Dropout)         (None, 900)               0

 dense_1 (Dense)             (None, 500)               450500

 dropout_2 (Dropout)         (None, 500)               0

 dense_2 (Dense)             (None, 400)               200400

 dropout_3 (Dropout)         (None, 400)               0

 dense_3 (Dense)             (None, 100)               40100

 dropout_4 (Dropout)         (None, 100)               0

 dense_4 (Dense)             (None, 10)                1010

=================================================================
Total params: 1,398,510
Trainable params: 1,398,510
Non-trainable params: 0
```

Fig 3: Six layer Neural networks

- **Training the model**

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0,2) # set the vertical range to [0-1]
plt.show()
```

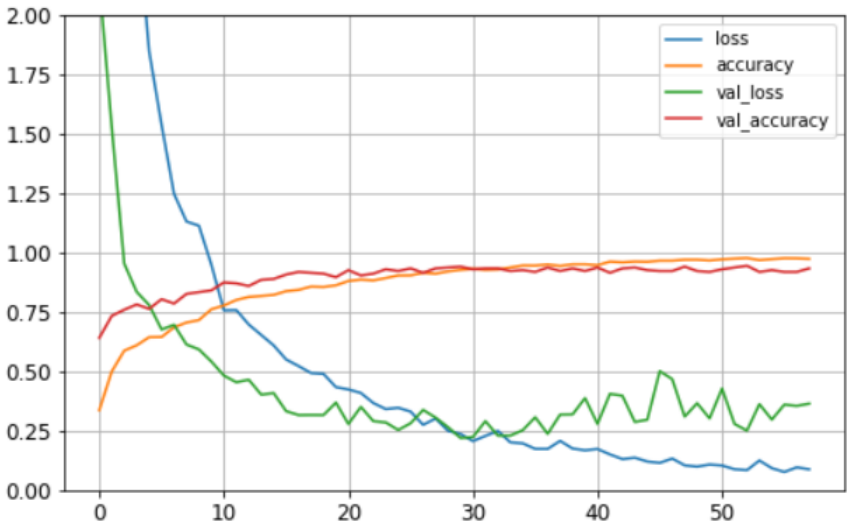

Fig 4: Training for 58 iterations, learning rate 0.0002

- **Model evaluation**

```
model.evaluate(X_test, y_test)
```

```
10/10 [==============================] - 0s 5ms/step - loss: 0.5042 - accurac
y: 0.9333

[0.5041508674621582, 0.9333333373069763]
```

Fig 5: Model test accuracy 0.9333

- **Plotting misclassified observations in test set**

Fig 6: Some of the misclassified images

- **Neural Networks with PCA data – Model training**

```
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_8 (Flatten)         (None, 25)                0

 dropout_40 (Dropout)        (None, 25)                0

 dense_40 (Dense)            (None, 200)               5200

 dropout_41 (Dropout)        (None, 200)               0

 dense_41 (Dense)            (None, 150)               30150

 dropout_42 (Dropout)        (None, 150)               0

 dense_42 (Dense)            (None, 100)               15100

 dropout_43 (Dropout)        (None, 100)               0

 dense_43 (Dense)            (None, 80)                8080

 dropout_44 (Dropout)        (None, 80)                0

 dense_44 (Dense)            (None, 10)                810

=================================================================
Total params: 59,340
Trainable params: 59,340
Non-trainable params: 0
```

Fig 7: Six layer neural network for PCA data

- **Model training**

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0,2) # set the vertical range to [0-1]
plt.show()
```

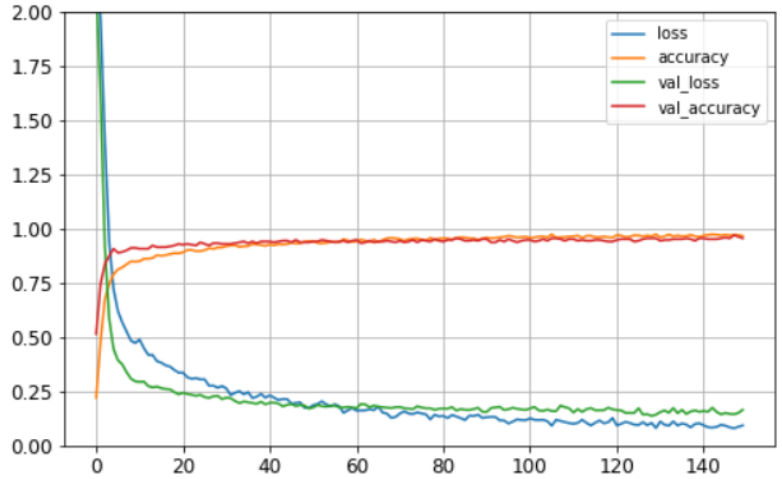

Fig 8: Model training with PCA data

- **Model Evaluation**

```
model.evaluate(X_test, y_test)
```

```
10/10 [==============================] - 0s 2ms/step - loss: 0.1670 - accurac
y: 0.9633

[0.16701318323612213, 0.9633333086967468]
```

Fig 9: NN Model PCA data with accuracy 0.9633

Part 2

- **CNN Model with MNIST Data**
- **Training model**

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        3200

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 128)       73856

 conv2d_2 (Conv2D)           (None, 14, 14, 128)       147584

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 conv2d_3 (Conv2D)           (None, 7, 7, 256)         295168

 conv2d_4 (Conv2D)           (None, 7, 7, 256)         590080

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 256)        0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 128)               295040

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 1,413,834
Trainable params: 1,413,834
Non-trainable params: 0
_________________________________________________________________
```

Fig 10: Model – Deep CNN Model

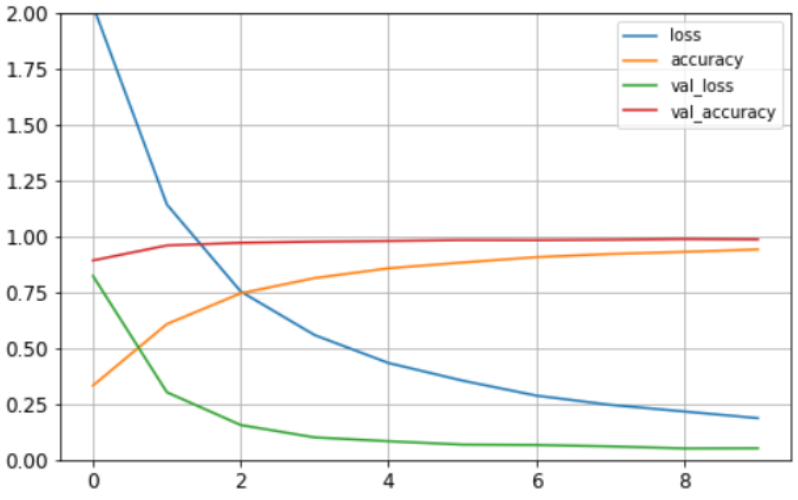

Fig 11: Training Deep CNN Model

```
model.evaluate(X_test, y_test)
```

```
219/219 [==============================] - 9s 39ms/step - loss: 0.0457 - accu
racy: 0.9889

[0.04568476229906082, 0.9888571500778198]
```

Fig 12: CNN Model , accuracy 0.9889

- **CNN Model with 3-4 layers**

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        3200

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 881,290
Trainable params: 881,290
Non-trainable params: 0
_________________________________________________________________
```

Fig 13: CNN Model

- **Model training**

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0,2) # set the vertical range to [0-1]
plt.show()
```

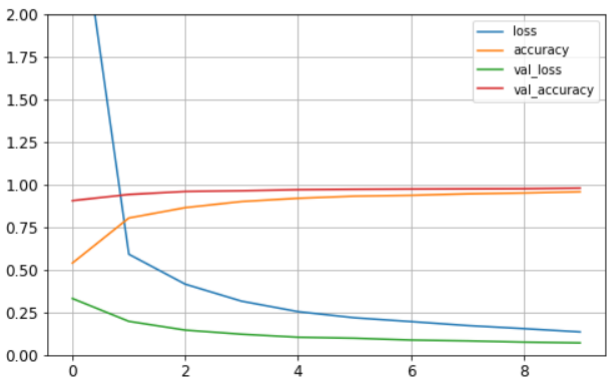

Fig 14: Model training

- **Model evaluation**

```
model.evaluate(X_test, y_test)
```

```
219/219 [==============================] - 3s 13ms/step - loss: 0.0665 - accu
racy: 0.9820

[0.06646185368299484, 0.9819999933242798]
```

Fig 15: Model evaluation, accuracy 0.9820