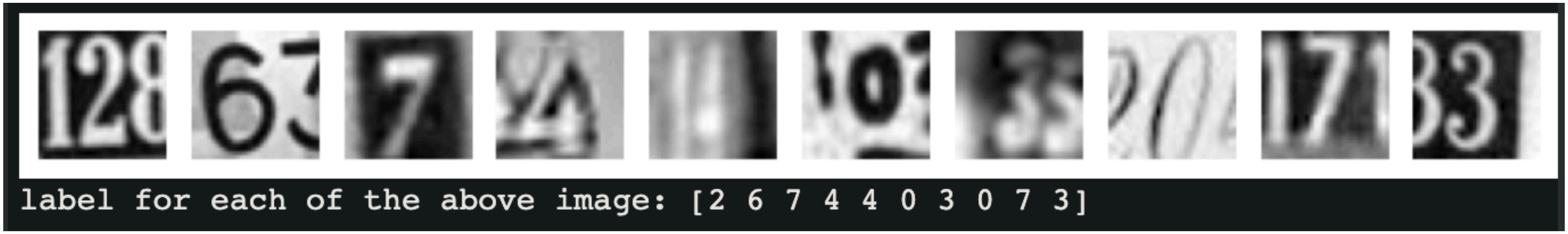
SVHN Digit Recognition

Neural Networks, Tensorflow, Keras

‘Femi Bolarinwa

Digit Image Snapshot



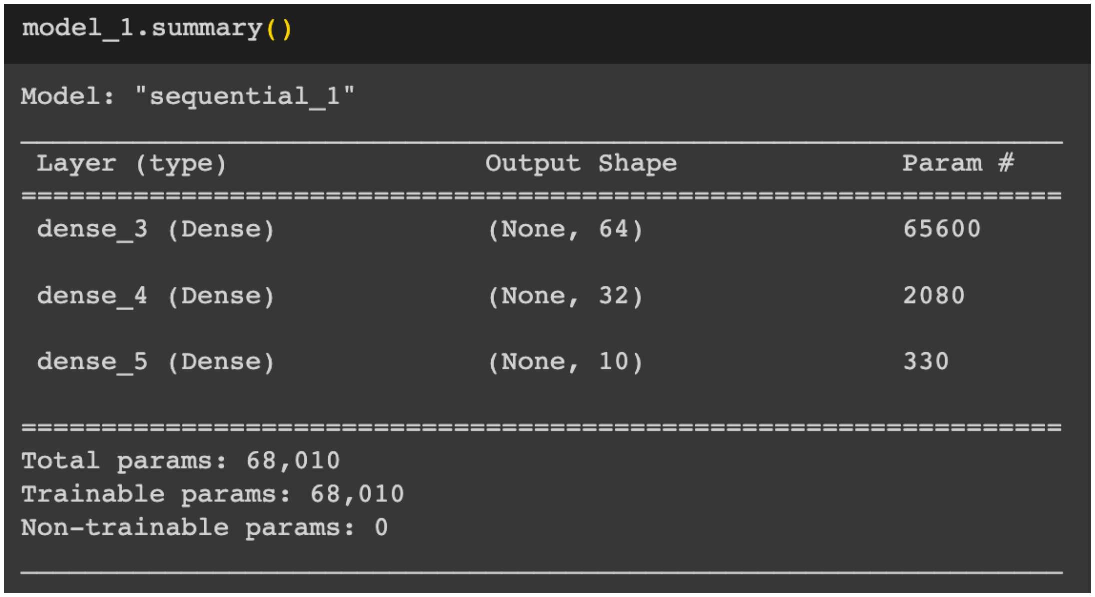
```
label for each of the above image: [2 6 7 4 4 0 3 0 7 3]
```

- Images available in grayscale stored in stack of 32 x 32 pixels
- 42,000 and 18,000 images used for training and testing respectively
- Pixels normalized to prevent exploding gradient
- One-hot encoding for target variables (image classifications)

Model 1

Artificial Neural Network

- First layer: 64 neurons, RELU activation
- Second layer: 32 neurons, RELU activation
- Output layer: 10 neurons, softmax activation
- 68,010 trainable parameters (weights & biases)



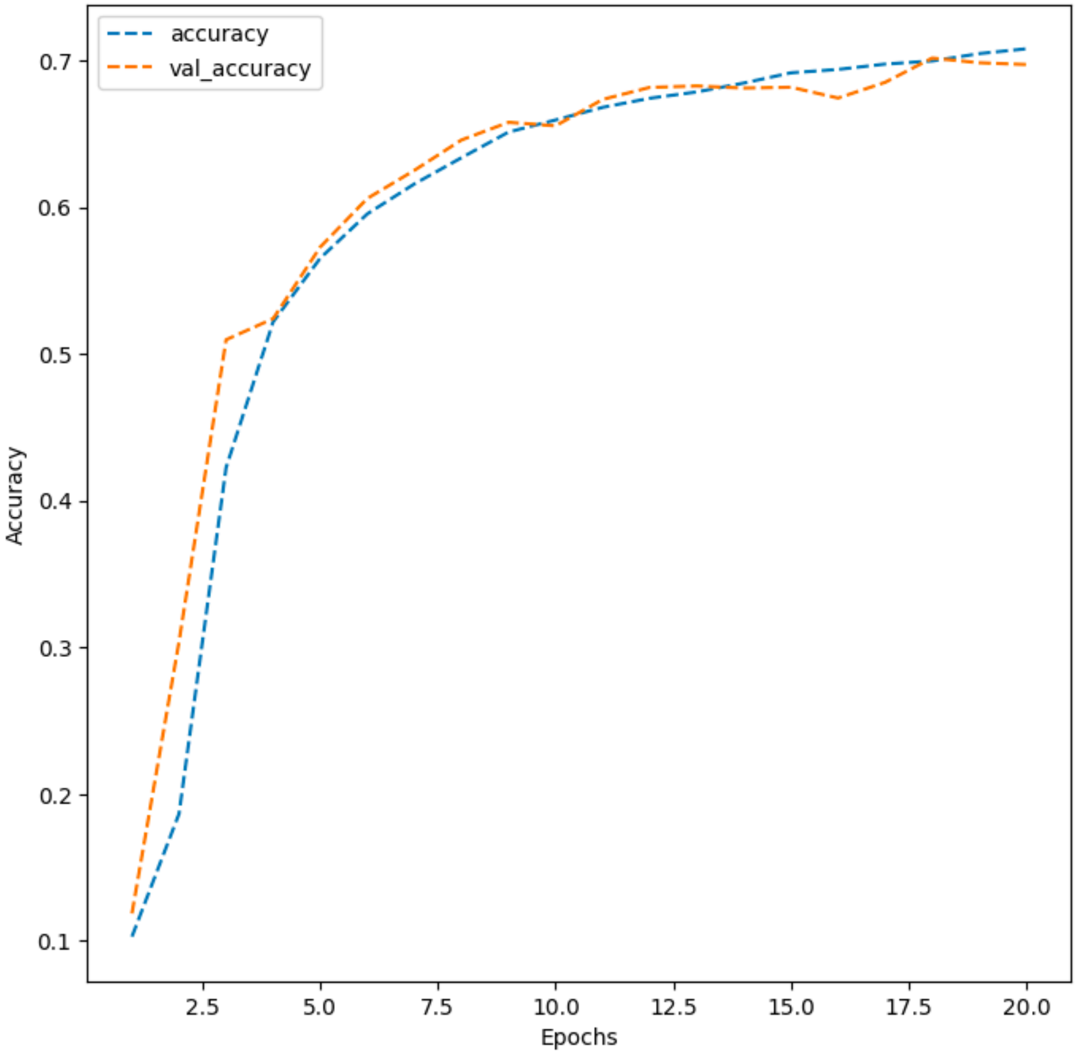
```
model_1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 64)                65600

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 10)                330

=================================================================
Total params: 68,010
Trainable params: 68,010
Non-trainable params: 0
_________________________________________________________________
```

Model 1 (ANN)

Model Training

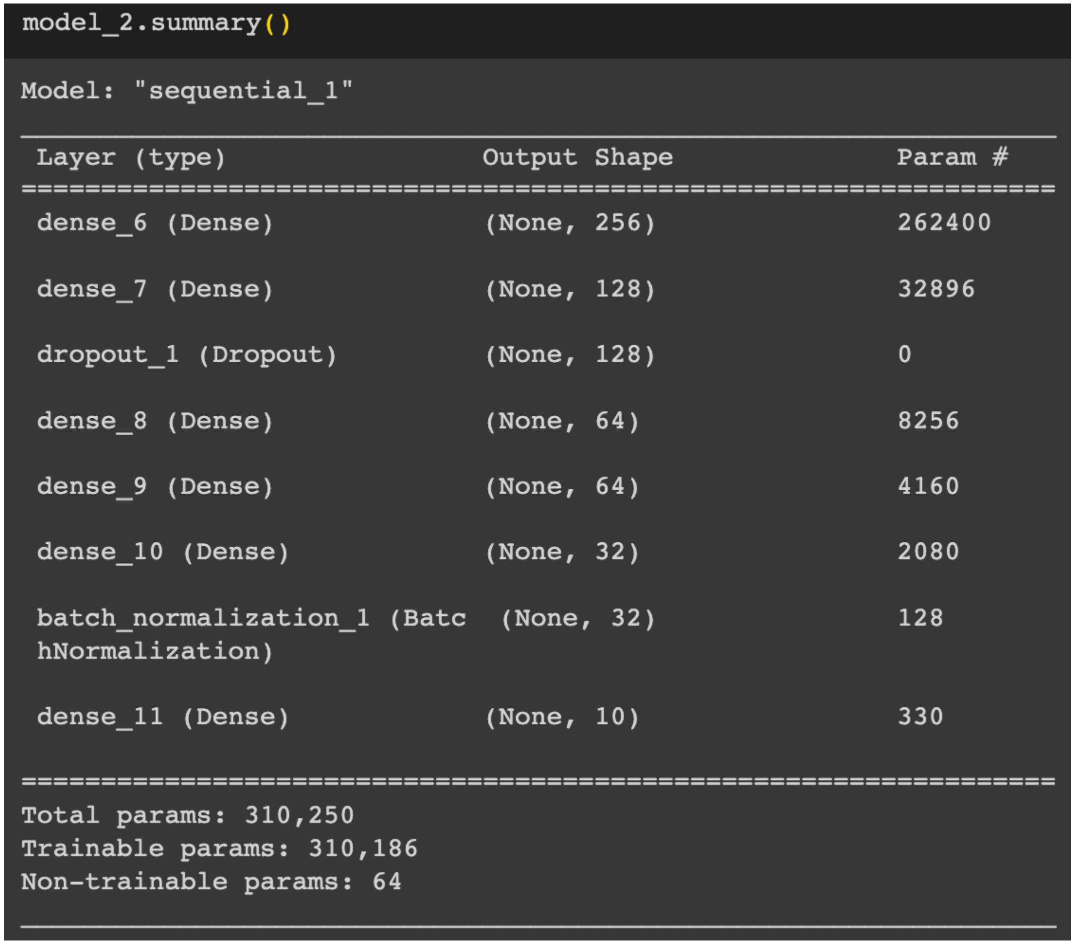
- 20 epochs
- ‘Adam’ optimizer, learning rate = 0.001
- Categorical-crossentropy loss function, accuracy metric
- ~70% accuracy on training and validation
- Decent accuracy



Model 2

Artificial Neural Network

- 6 layers, 554 neurons
- Dropout & BatchNormalization layers for regularization
- 310,186 trainable parameters (weights and biases)



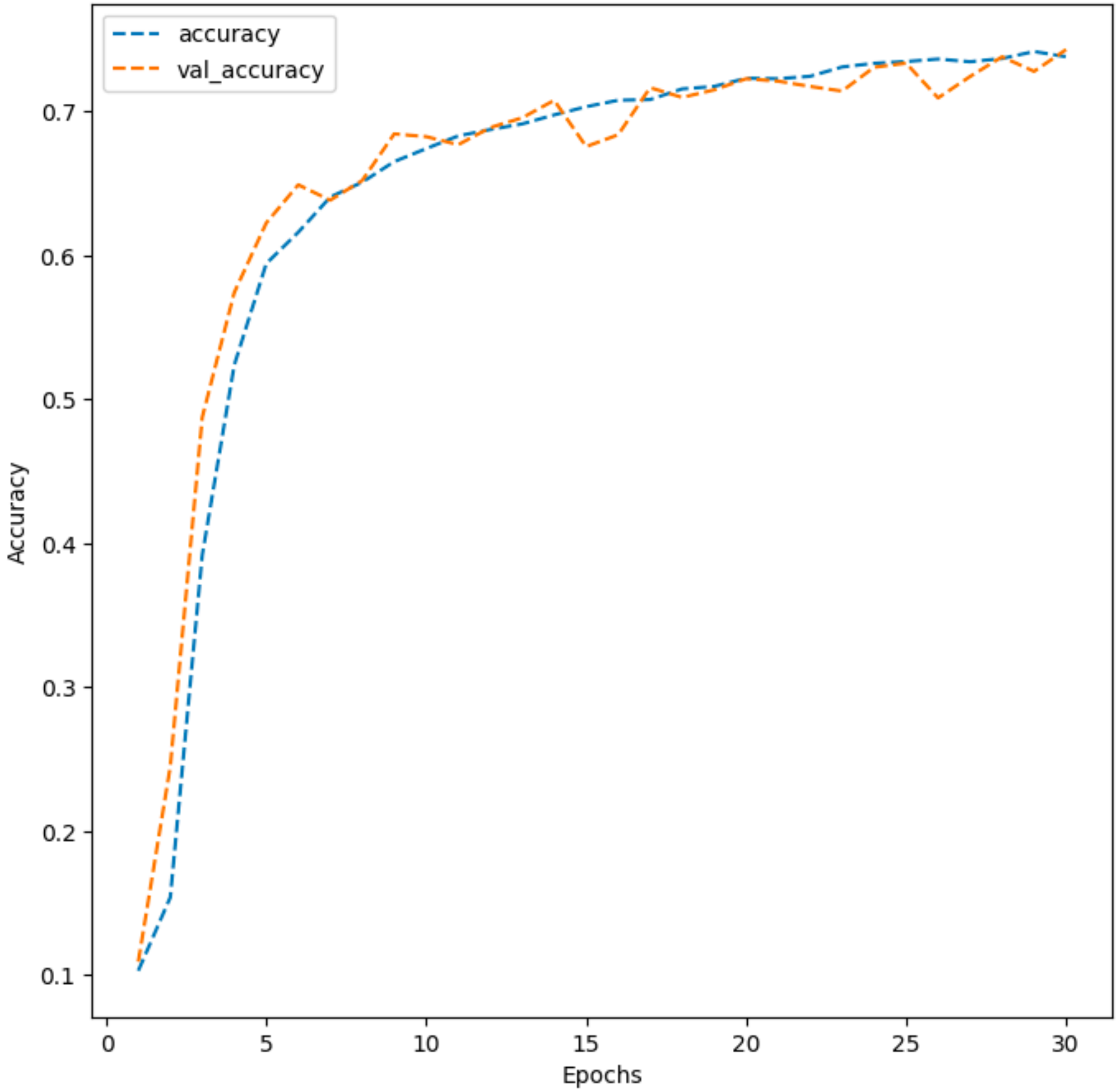
```
model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 256)               262400

 dense_7 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_8 (Dense)             (None, 64)                8256

 dense_9 (Dense)             (None, 64)                4160

 dense_10 (Dense)            (None, 32)                2080

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_11 (Dense)            (None, 10)                330

=================================================================
Total params: 310,250
Trainable params: 310,186
Non-trainable params: 64
_________________________________________________________________
```

Model 2 (ANN)

Model Training

- 30 epochs
- ‘Adam’ optimizer, learning rate = 0.0005
- Categorical-crossentropy loss function, accuracy metric
- ~74% accuracy on training and validation
- Bigger architecture, but no significant accuracy boost

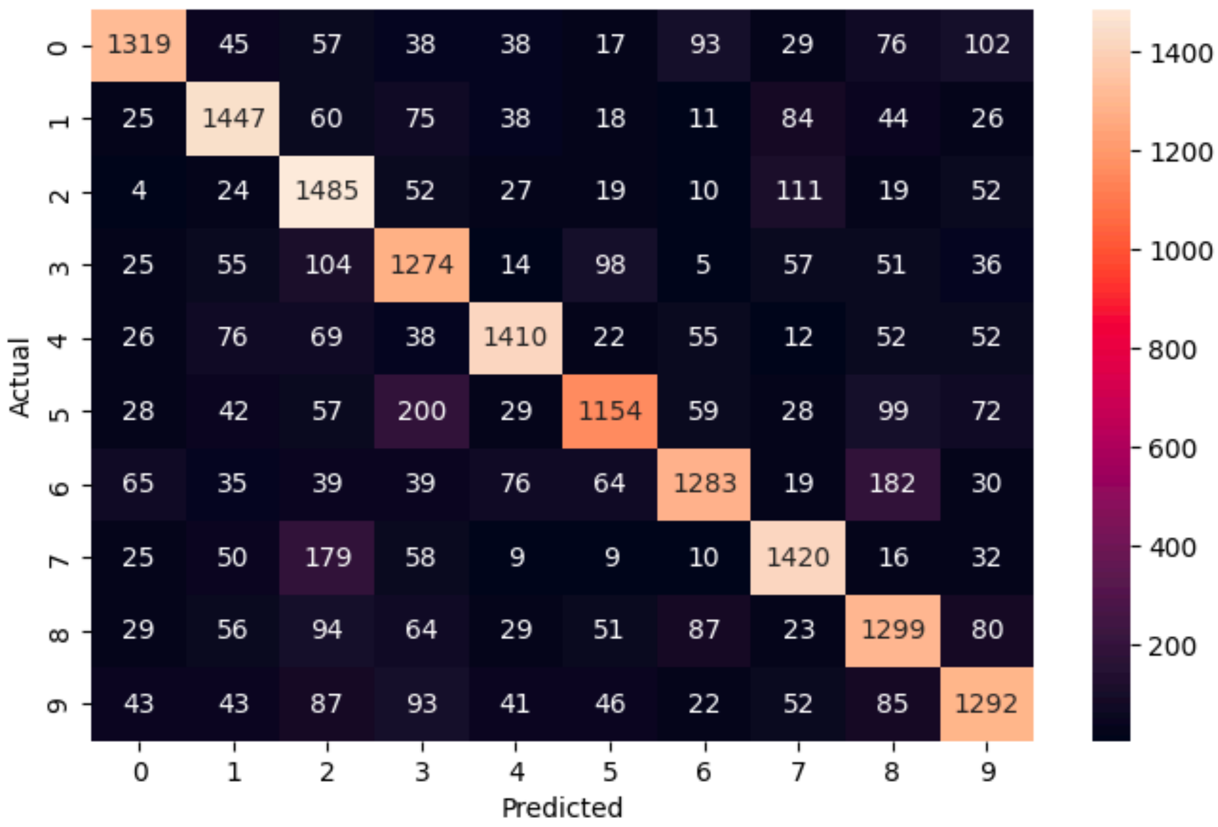


Model 2 (ANN)

Evaluation on Unseen Test data

- Accuracy: 74%
- Recall: 74%
- Precision: 75%
- f1-score: 74%
- Similar accuracy on training and test dataset.
- Decent model, not overfitted

Confusion Matrix

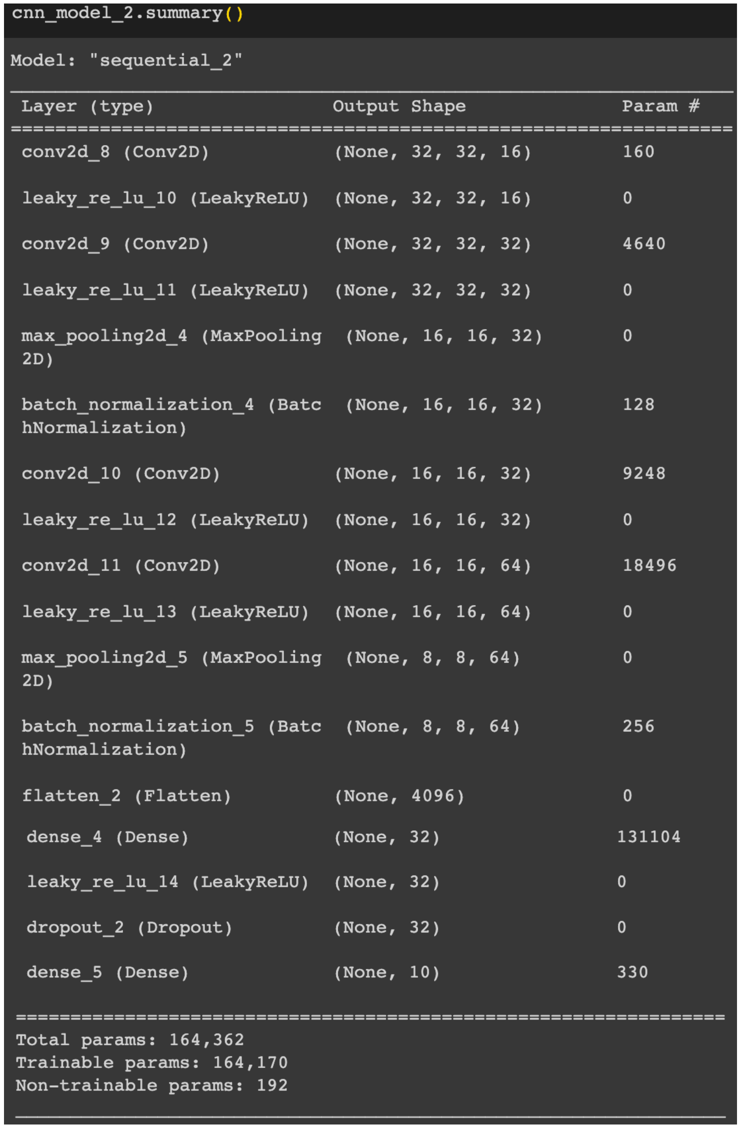


Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	1319	45	57	38	38	17	93	29	76	102
1	25	1447	60	75	38	18	11	84	44	26
2	4	24	1485	52	27	19	10	111	19	52
3	25	55	104	1274	14	98	5	57	51	36
4	26	76	69	38	1410	22	55	12	52	52
5	28	42	57	200	29	1154	59	28	99	72
6	65	35	39	39	76	64	1283	19	182	30
7	25	50	179	58	9	9	10	1420	16	32
8	29	56	94	64	29	51	87	23	1299	80
9	43	43	87	93	41	46	22	52	85	1292

Model 3

Convolutional Neural Network

- 6 layers
- 4 convolution filters for feature extraction
- Padding and MaxPooling
- Dropout & BatchNormalization layers for regularization
- 164,170 trainable parameters (weights and biases)

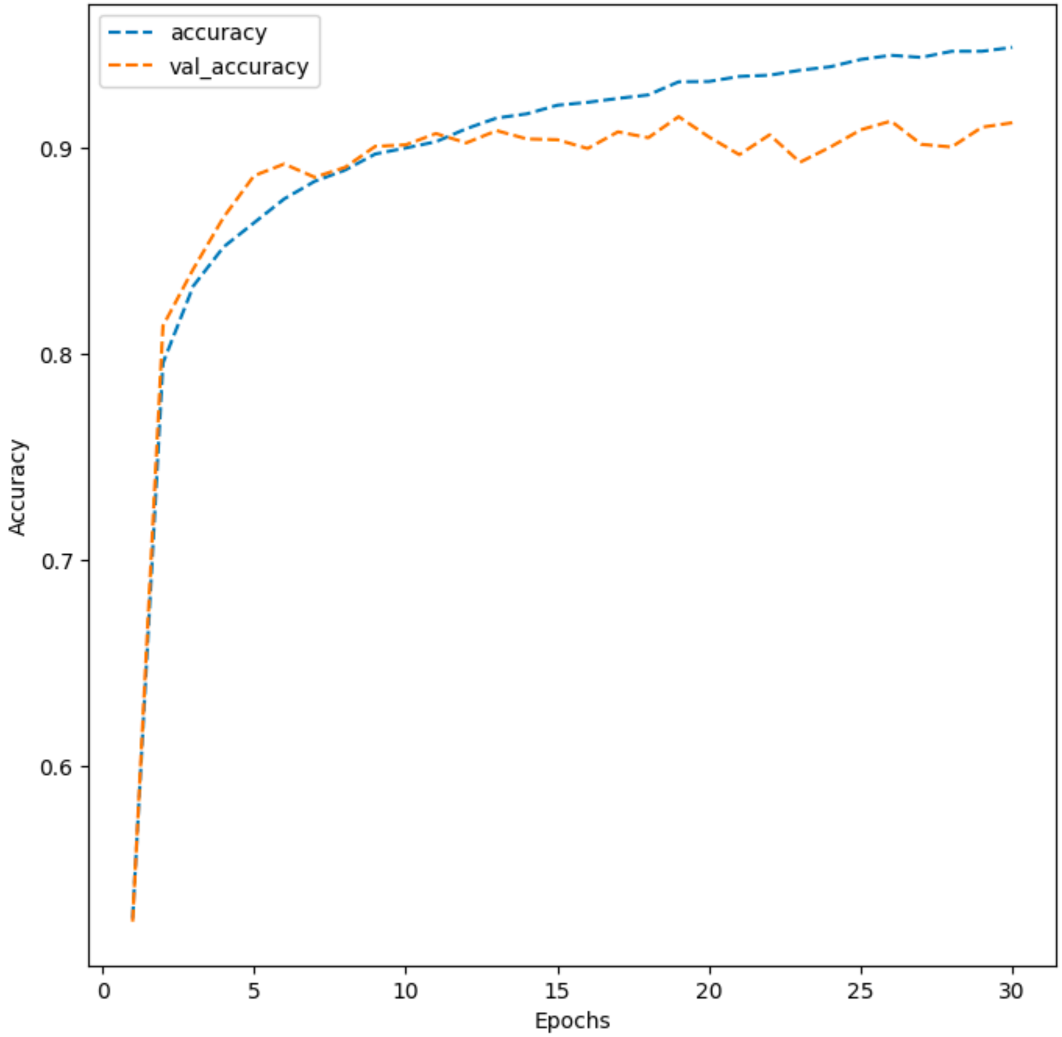


```
cnn_model_2.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 32, 32, 16)        160

 leaky_re_lu_10 (LeakyReLU)  (None, 32, 32, 16)        0

 conv2d_9 (Conv2D)           (None, 32, 32, 32)        4640

 leaky_re_lu_11 (LeakyReLU)  (None, 32, 32, 32)        0

 max_pooling2d_4 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 batch_normalization_4 (Batc  (None, 16, 16, 32)       128
 hNormalization)

 conv2d_10 (Conv2D)          (None, 16, 16, 32)        9248

 leaky_re_lu_12 (LeakyReLU)  (None, 16, 16, 32)        0

 conv2d_11 (Conv2D)          (None, 16, 16, 64)        18496

 leaky_re_lu_13 (LeakyReLU)  (None, 16, 16, 64)        0

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 batch_normalization_5 (Batc  (None, 8, 8, 64)         256
 hNormalization)

 flatten_2 (Flatten)         (None, 4096)              0
 dense_4 (Dense)             (None, 32)                131104

 leaky_re_lu_14 (LeakyReLU)  (None, 32)                0

 dropout_2 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 10)                330

=================================================================
Total params: 164,362
Trainable params: 164,170
Non-trainable params: 192
_________________________________________________________________
```

CNN Model

Model Training

- 20 epochs
- ‘Adam’ optimizer, learning_rate = 0.001
- Categorical-crossentropy loss function, accuracy metric
- >90% accuracy on training and validation

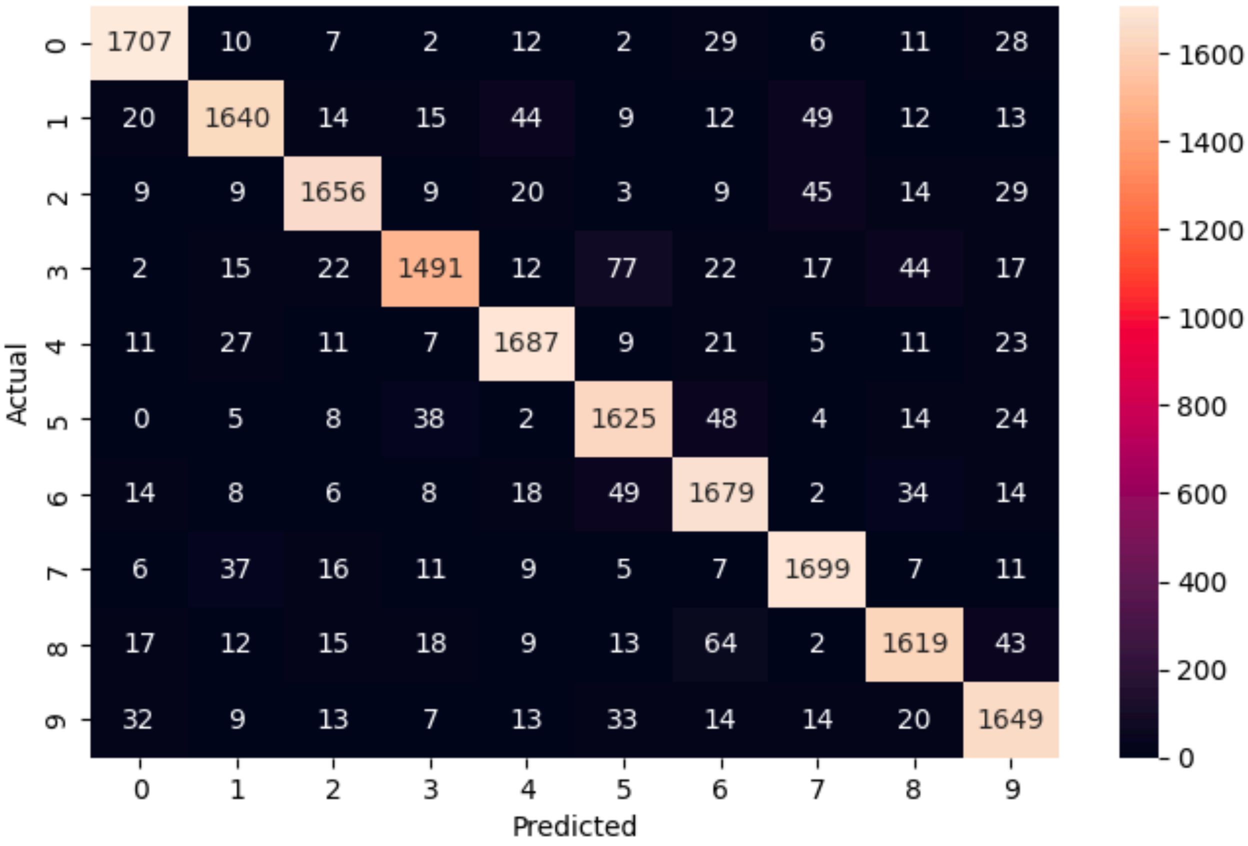


CNN Model

Evaluation on Unseen Test data

- Accuracy: 91%
- Recall: 91%
- Precision: 91%
- f1-score: 91%
- Similar accuracy on training and test dataset.
- Significantly less misclassification
- Great model, not overfitted

Confusion Matrix



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	1707	10	7	2	12	2	29	6	11	28
1	20	1640	14	15	44	9	12	49	12	13
2	9	9	1656	9	20	3	9	45	14	29
3	2	15	22	1491	12	77	22	17	44	17
4	11	27	11	7	1687	9	21	5	11	23
5	0	5	8	38	2	1625	48	4	14	24
6	14	8	6	8	18	49	1679	2	34	14
7	6	37	16	11	9	5	7	1699	7	11
8	17	12	15	18	9	13	64	2	1619	43
9	32	9	13	7	13	33	14	14	20	1649

Insight and Recommendation

- CNN appears to be better than ANN for feature extraction and hence higher accuracy
- However, this comes at the cost of more computational capacity requirement
- There is always a need to balance the need for higher accuracy and cost of complex architecture and hence computation time