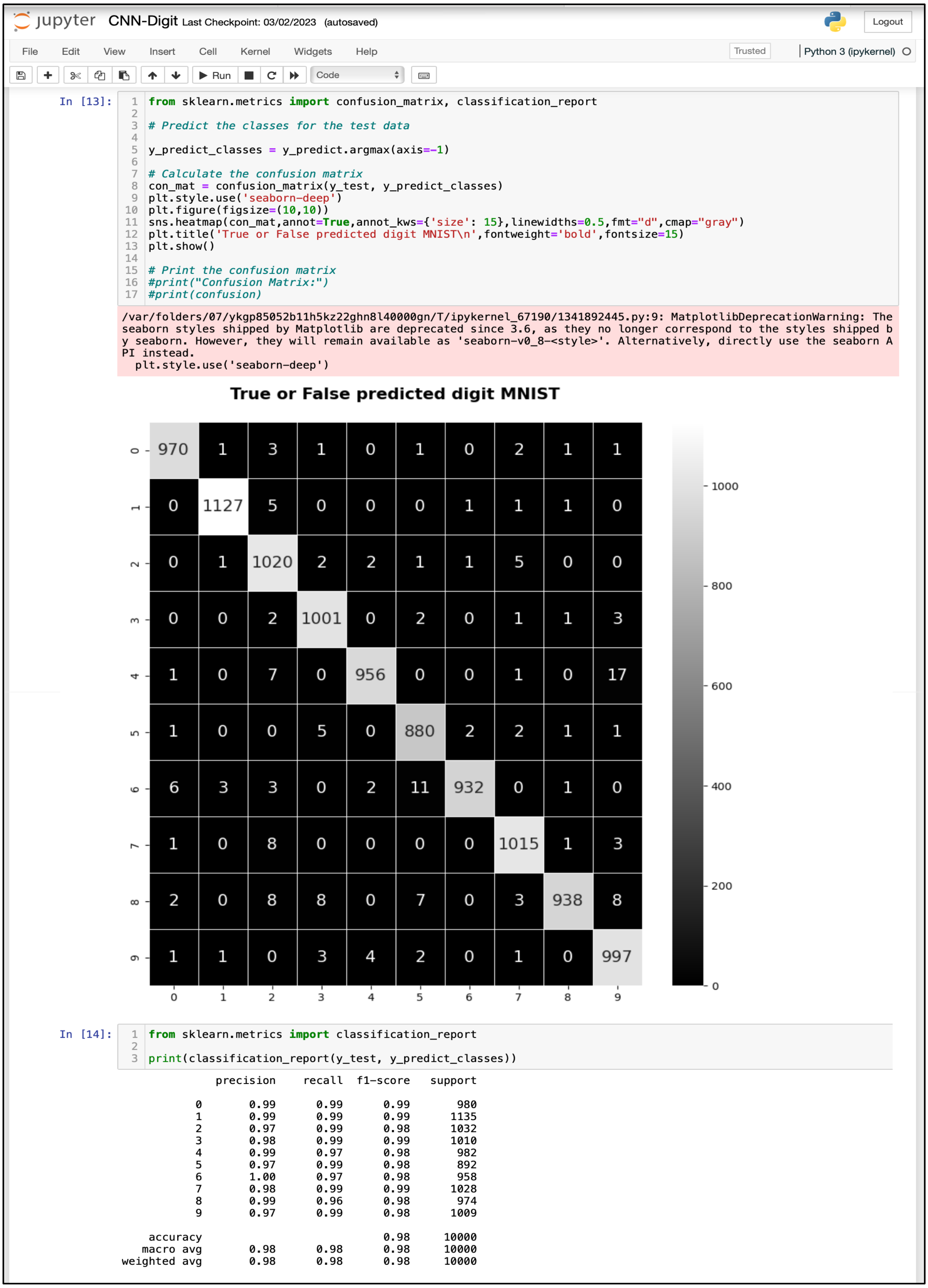
Deep Learning (Assignments From Video Lecture)

Table Of Contents (TOC)

1. CM & Classification Report for CNN

Proof:



In [13]:

```python
from sklearn.metrics import confusion_matrix, classification_report

# Predict the classes for the test data

y_predict_classes = y_predict.argmax(axis=-1)

# Calculate the confusion matrix
con_mat = confusion_matrix(y_test, y_predict_classes)
plt.style.use('seaborn-deep')
plt.figure(figsize=(10,10))
sns.heatmap(con_mat,annot=True,annot_kws={'size': 15},linewidths=0.5,fmt="d",cmap="gray")
plt.title('True or False predicted digit MNIST\n',fontweight='bold',fontsize=15)
plt.show()

# Print the confusion matrix
#print("Confusion Matrix:")
#print(confusion)
```

```
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_67190/1341892445.py:9: MatplotlibDeprecationWarning: The
seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped b
y seaborn. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn A
PI instead.
  plt.style.use('seaborn-deep')
```

True or False predicted digit MNIST

	0	1	2	3	4	5	6	7	8	9
0	970	1	3	1	0	1	0	2	1	1
1	0	1127	5	0	0	0	1	1	1	0
2	0	1	1020	2	2	1	1	5	0	0
3	0	0	2	1001	0	2	0	1	1	3
4	1	0	7	0	956	0	0	1	0	17
5	1	0	0	5	0	880	2	2	1	1
6	6	3	3	0	2	11	932	0	1	0
7	1	0	8	0	0	0	0	1015	1	3
8	2	0	8	8	0	7	0	3	938	8
9	1	1	0	3	4	2	0	1	0	997

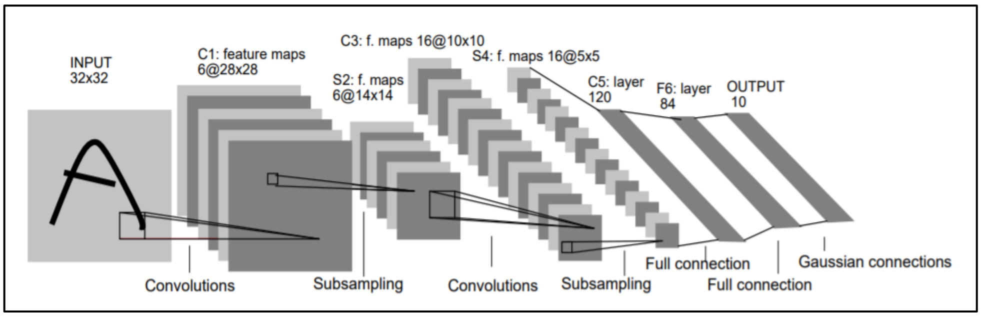
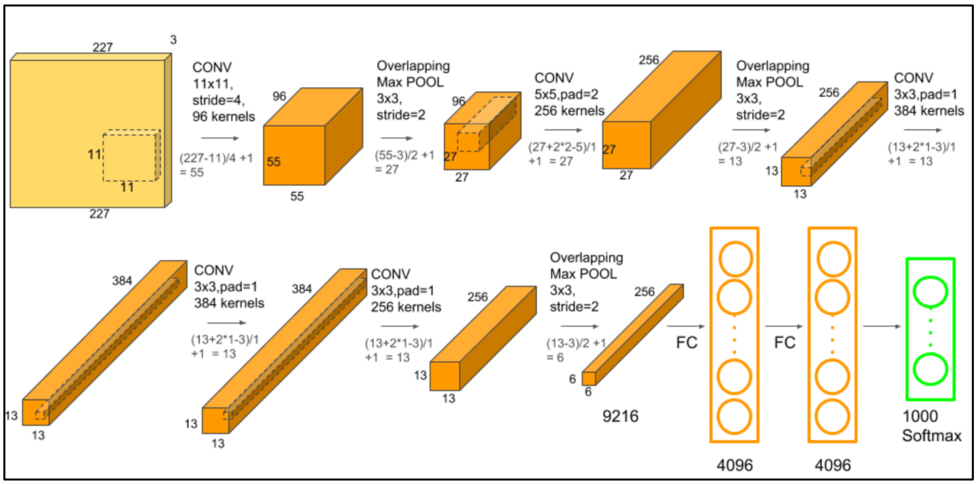
In [14]:

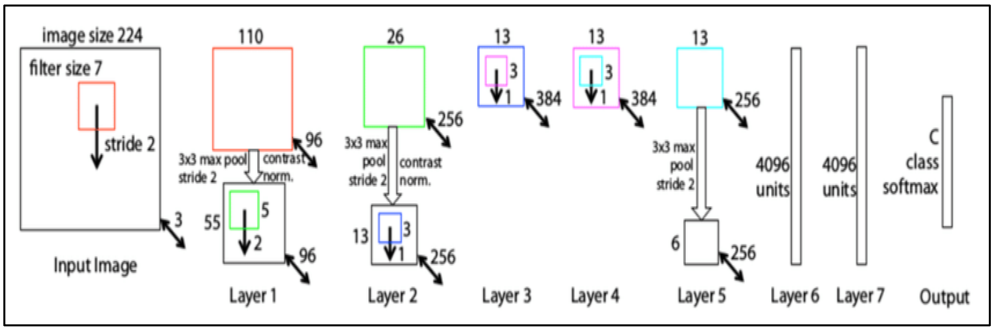
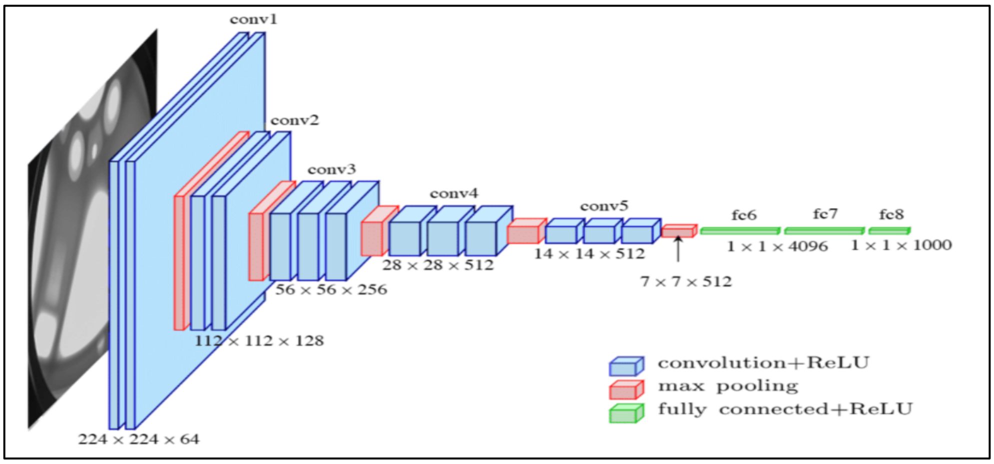
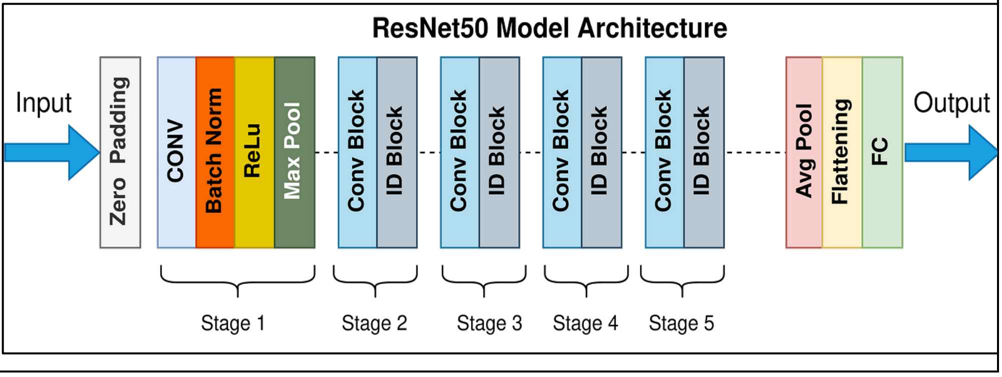
```python
from sklearn.metrics import classification_report

print(classification_report(y_test, y_predict_classes))
```

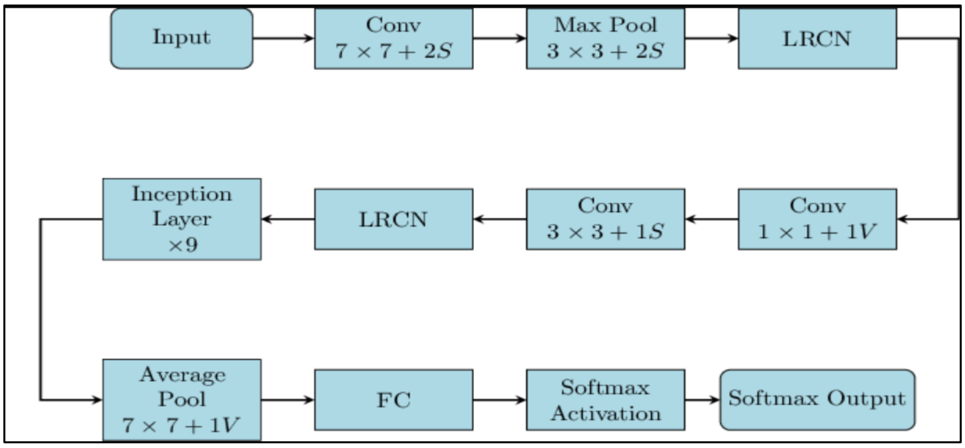
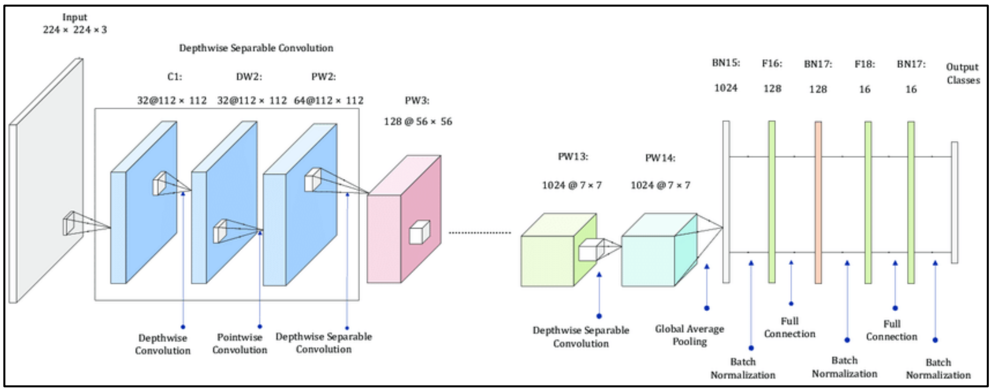
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.99      0.98      1032
           3       0.98      0.99      0.99      1010
           4       0.99      0.97      0.98       982
           5       0.97      0.99      0.98       892
           6       1.00      0.97      0.98       958
           7       0.98      0.99      0.99      1028
           8       0.99      0.96      0.98       974
           9       0.97      0.99      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

2. CNN Architecture Types

Summary of different types of CNN architectures:

Architecture	Year	Key Features	Use Case
LeNet	1998	First successful applications of CNNs, 5 layers (alternating between convolutional and pooling), Used tanh/sigmoid activation functions	Recognizing handwritten and machine-printed characters
Block Diagram	 INPUT 32x32 C1: feature maps 6@28x28 S2: f. maps 6@14x14 C3: f. maps 16@10x10 S4: f. maps 16@5x5 C5: layer 120 F6: layer 84 OUTPUT 10 Convolutions Subsampling Convolutions Subsampling Full connection Gaussian connections Full connection		
AlexNet	2012	Deeper and wider than LeNet, Used ReLU activation function, Implemented dropout layers, Used GPUs for training	Large-scale image recognition tasks
Block Diagram	 227 3 11 11 227 CONV 11x11, stride=4, 96 kernels (227-11)/4 +1 = 55 96 55 55 Overlapping Max POOL 3x3, stride=2 (55-3)/2 +1 = 27 96 27 27 CONV 5x5,pad=2 256 kernels (27+2*2-5)/1 +1 = 27 256 27 27 Overlapping Max POOL 3x3, stride=2 (27-3)/2 +1 = 13 256 13 13 CONV 3x3,pad=1 384 kernels (13+2*1-3)/1 +1 = 13 384 13 13 CONV 3x3,pad=1 384 kernels (13+2*1-3)/1 +1 = 13 384 13 13 CONV 3x3,pad=1 256 kernels (13+2*1-3)/1 +1 = 13 256 13 13 Overlapping Max POOL 3x3, stride=2 (13-3)/2 +1 = 6 256 6 6 9216 FC FC 4096 4096 1000 Softmax		
ZFNet	2013	Similar architecture to AlexNet, but with different filter sizes and numbers of filters, Visualization techniques for understanding the network	ImageNet classification

Architecture	**Year**	**Key Features**	**Use Case**
Block Diagram	 image size 224 filter size 7 stride 2 3 Input Image 110 3x3 max pool stride 2 contrast norm. 96 55 5 2 96 Layer 1 26 256 3x3 max pool stride 2 contrast norm. 13 3 1 256 Layer 2 13 3 1 384 Layer 3 13 3 1 384 Layer 4 13 256 3x3 max pool stride 2 6 256 Layer 5 4096 units Layer 6 4096 units Layer 7 C class softmax Output		
VGGNet	2014	Deeper networks with smaller filters (3×3), All convolutional layers have the same depth, Multiple configurations (VGG16, VGG19)	Large-scale image recognition
Block Diagram	 conv1 conv2 conv3 conv4 conv5 fc6 fc7 fc8 224 × 224 × 64 112 × 112 × 128 56 × 56 × 256 28 × 28 × 512 14 × 14 × 512 7 × 7 × 512 1 × 1 × 4096 1 × 1 × 1000 convolution+ReLU max pooling fully connected+ReLU		
ResNet	2015	Introduced "skip connections" or "shortcuts" to enable training of deeper networks, Multiple configurations (ResNet-50, ResNet-101, ResNet-152)	Large-scale image recognition, won 1st place in the ILSVRC 2015
Block Diagram	 ResNet50 Model Architecture Input Zero Padding CONV Batch Norm ReLu Max Pool Conv Block ID Block Conv Block ID Block Conv Block ID Block Conv Block ID Block Avg Pool Flattening FC Output Stage 1 Stage 2 Stage 3 Stage 4 Stage 5		

Architecture	Year	Key Features	Use Case
GoogleLeNet	2014	Introduced Inception module, which allows for more efficient computation and deeper networks, multiple versions (Inception v1, v2, v3, v4)	Large-scale image recognition, won 1st place in the ILSVRC 2014
Block Diagram	 Input Conv $7 \times 7 + 2S$ Max Pool $3 \times 3 + 2S$ LRCN Conv $1 \times 1 + 1V$ Conv $3 \times 3 + 1S$ LRCN Inception Layer $\times 9$ Average Pool $7 \times 7 + 1V$ FC Softmax Activation Softmax Output		
MobileNet	2017	Designed for mobile and embedded vision applications, Uses depth wise separable convolutions to reduce the model size and complexity	Mobile and embedded vision applications, real-time object detection
Block Diagram	 Input 224 × 224 × 3 Depthwise Separable Convolution C1: DW2: PW2: 32@112 × 112 32@112 × 112 64@112 × 112 PW3: 128 @ 56 × 56 PW13: 1024 @ 7 × 7 PW14: 1024 @ 7 × 7 BN15: 1024 F16: 128 BN17: 128 F18: 16 BN17: 16 Output Classes Depthwise Convolution Pointwise Convolution Depthwise Separable Convolution Depthwise Separable Convolution Global Average Pooling Full Connection Full Connection Batch Normalization Batch Normalization Batch Normalization		

3. Types of Pre-Trained Models Available

There are various types of pretrained models available in deep learning, which have been trained on large datasets for specific tasks. These pretrained models are often used as a starting point for transfer learning, where you fine-tune the model on a smaller dataset for a different or related task. Here are some common types of pretrained models:

Table:

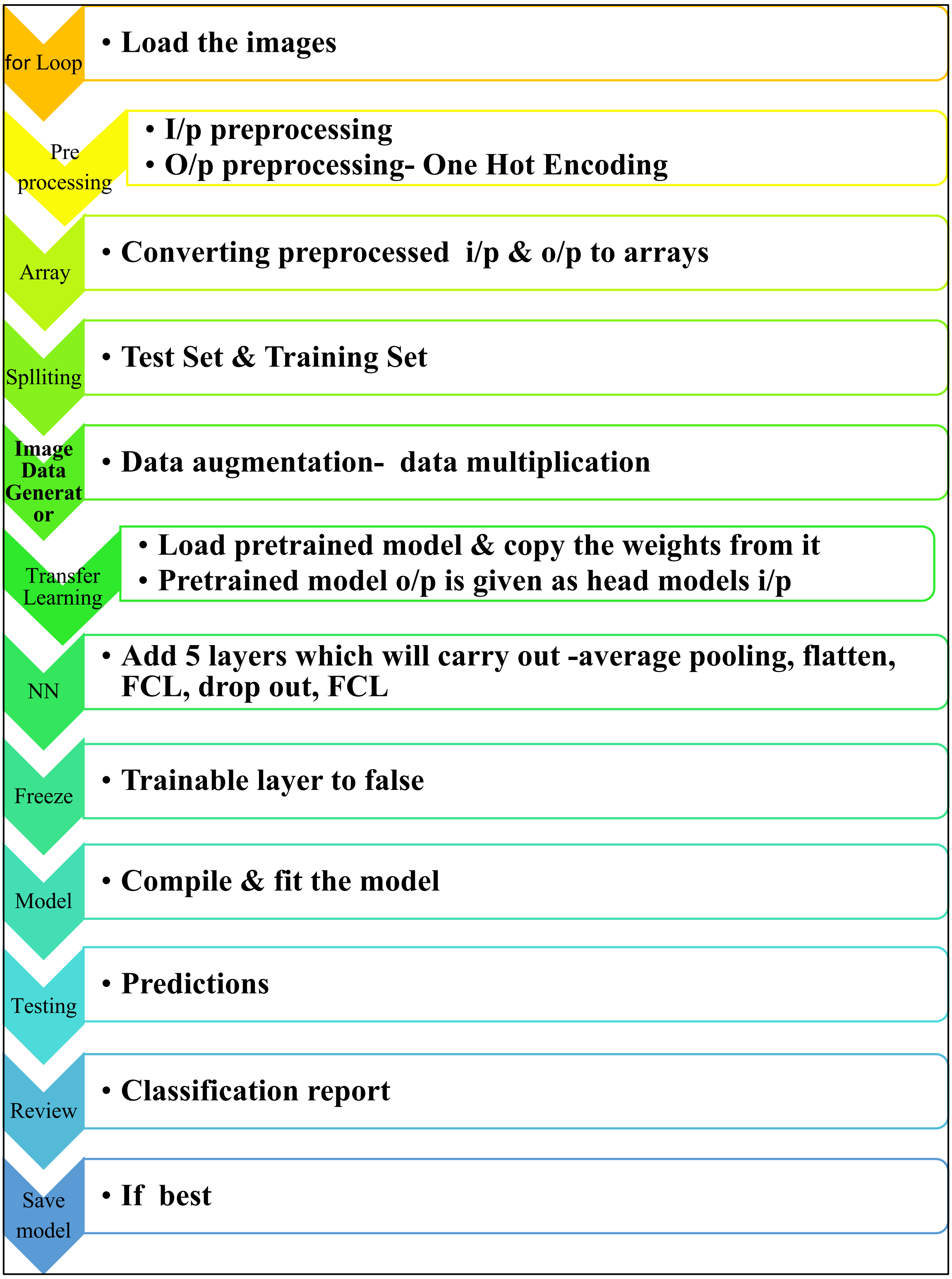
S.No.	Genre	Models Available
1.	Image Classification	○ VGG (e.g., VGG16, VGG19) ○ ResNet (e.g., ResNet50) ○ Inception (e.g., InceptionV3) ○ MobileNet ○ EfficientNet ○ DenseNet ○ AlexNet ○ Xception
2.	Object Detection	○ YOLO (You Only Look Once) ○ Faster R-CNN ○ Single Shot MultiBox Detector (SSD) ○ RetinaNet
3.	Semantic Segmentation	○ U-Net ○ FCN (Fully Convolutional Network) ○ Deeplab
4.	Natural Language Processing (NLP)	○ BERT (Bidirectional Encoder Representations from Transformers) ○ GPT (Generative Pretrained Transformer) ○ RoBERTa ○ XLNet ○ T5 (Text-to-Text Transfer Transformer) ○ ERNIE ○ ALBERT ○ TextCNN ○ DistilBERT ○ VADER (Valence Aware Dictionary & sEntiment Reasoner) ○ EmoReact
5.	Speech Recognition	○ DeepSpeech ○ Jasper ○ Wav2Vec

6.	Recommendation	○ Collaborative Filtering Models ○ Matrix Factorization Models ○ Neural Collaborative Filtering
7.	Time Series	○ LSTM (Long Short-Term Memory) networks ○ GRU (Gated Recurrent Unit) networks ○ TCN (Temporal Convolutional Network)
8.	Graph Neural Network	○ GCN (Graph Convolutional Network) ○ GAT (Graph Attention Network) ○ GraphSAGE
9.	Reinforcement Learning	○ DDPG (Deep Deterministic Policy Gradient) ○ PPO (Proximal Policy Optimization) ○ A3C (Asynchronous Advantage Actor-Critic)
10.	Generative Adversarial	○ DCGAN (Deep Convolutional GAN) ○ StyleGAN ○ CycleGAN ○ BigGAN

These pretrained models can save a lot of time and computational resources, allowing developers and researchers to build and fine-tune models for specific applications more effectively.

4. Face Mask Detection

4a. Flow Chart For Training Part (Face Mask Detection):



for Loop
- Load the images

Pre processing
- I/p preprocessing
- O/p preprocessing- One Hot Encoding

Array
- Converting preprocessed i/p & o/p to arrays

Splliting
- Test Set & Training Set

Image Data Generator
- Data augmentation- data multiplication

Transfer Learning
- Load pretrained model & copy the weights from it
- Pretrained model o/p is given as head models i/p

NN
- Add 5 layers which will carry out -average pooling, flatten, FCL, drop out, FCL

Freeze
- Trainable layer to false

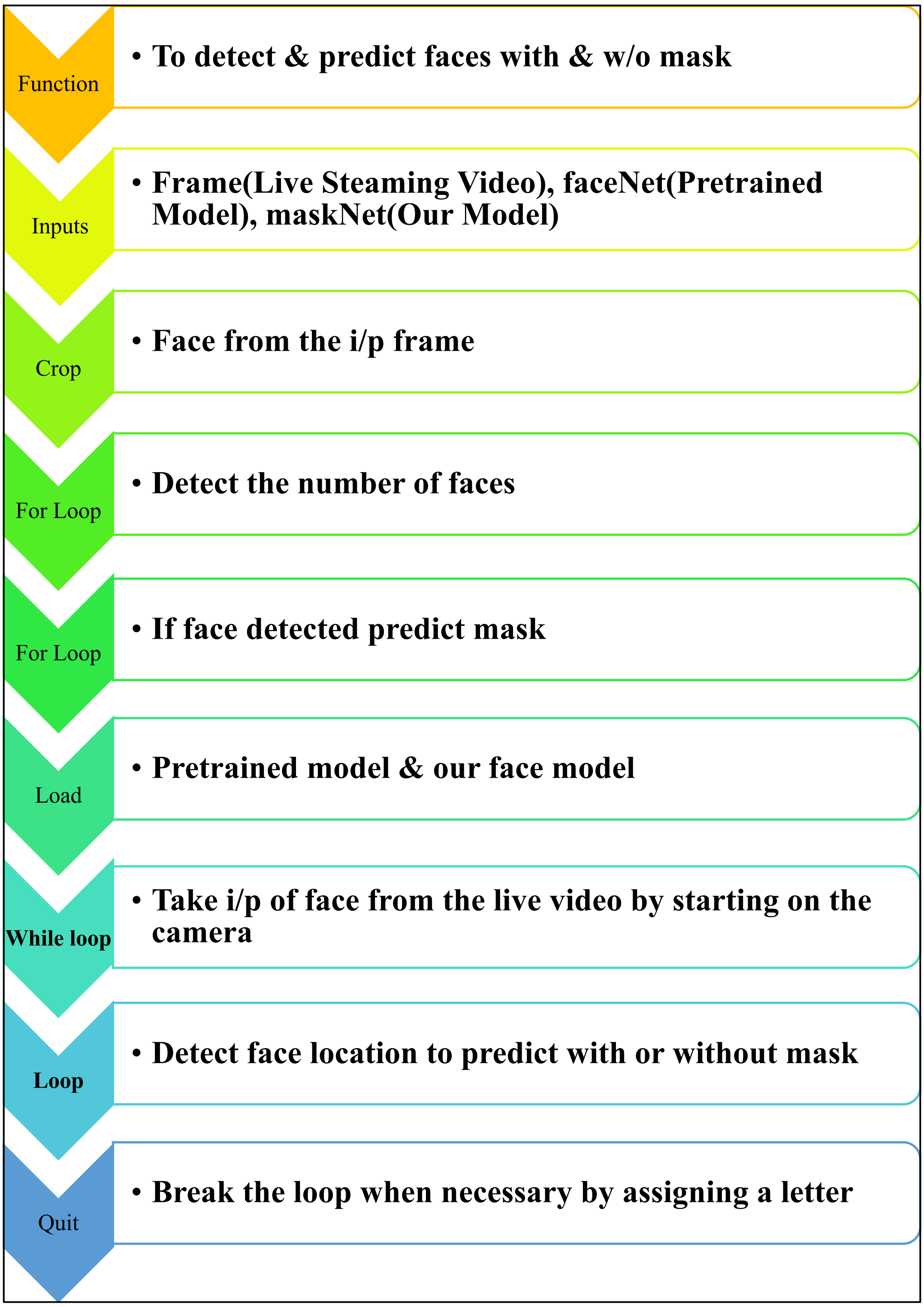
Model
- Compile & fit the model

Testing
- Predictions

Review
- Classification report

Save model
- If best

4b. Flow Chart For Application Part (Face Mask Detection):



Function
- To detect & predict faces with & w/o mask

Inputs
- Frame(Live Steaming Video), faceNet(Pretrained Model), maskNet(Our Model)

Crop
- Face from the i/p frame

For Loop
- Detect the number of faces

For Loop
- If face detected predict mask

Load
- Pretrained model & our face model

While loop
- Take i/p of face from the live video by starting on the camera

Loop
- Detect face location to predict with or without mask

Quit
- Break the loop when necessary by assigning a letter

----------END OF THE DOCUMENT---------