COMPARING TRANSFER LEARNING TECHNIQUES FOR DETECTION OF TRAFFIC SIGNS USING IMAGE RECOGNITION

MSc DATA ANALYTICS (JAN 2020)

BATCH-A TUTORIAL-2

Submitted to: **PROF. JOHN KELLY**

Hitesh Madhukar Patil

Vishal Shakya

DMML2 GROUP-F

Vikas Kishanrao Thamke

Amandeep Singh

WHAT IS THE AIM?

AMANDEEP SINGH

- Compare different types of Deep Learning algorithms for traffic sign detection and recognition

- CNN was used

- Transfer Learning models were included in the mix

WHAT IS THE MOTIVATION?

- 1.35 million deaths (each year, globally) [WHO Stats] in car-related accidents

- Primary cause: human error —> completely avoidable

- Need for automation to reduce deaths

- DL and TL techniques are used

SEQUENCE OF PRESENTATION

Data Description and Research Overview - Hitesh

Methodology - Vikas

Project Outline - Vishal

Results and Concluding Remarks - Aman

The Literature Review

DMML-2

Traffic Signposts Recognition System

- Identifying and classifying traffic signposts is a challenging job.
- CNN edge over other network.
- CNN has earned noteworthiness for its proved advancements over other networks.
- CNN has efficient learning skills, with many excellent features such as the translation invariance and local links.

General traffic signposts system:

- Detection of the signposts.
- Classification of the detected signposts.

Dataset:

- Kaggle repository (German Traffic Signposts Recognition Benchmark)

Related Work

- Reviewed total of 21 papers
- All papers are related traffic signposts recognition
- GTSRB – German Traffic Signposts Recognition Benchmark
- Artificial Neural Network
- Convolutional Neural Network
- Transfer learning
- InceptionV3
- VGG16 and VGG19
- Gradient Decent, Adam optimizer, SoftMax Function, ReLU activation function and techniques like Max-pooling, Blurring

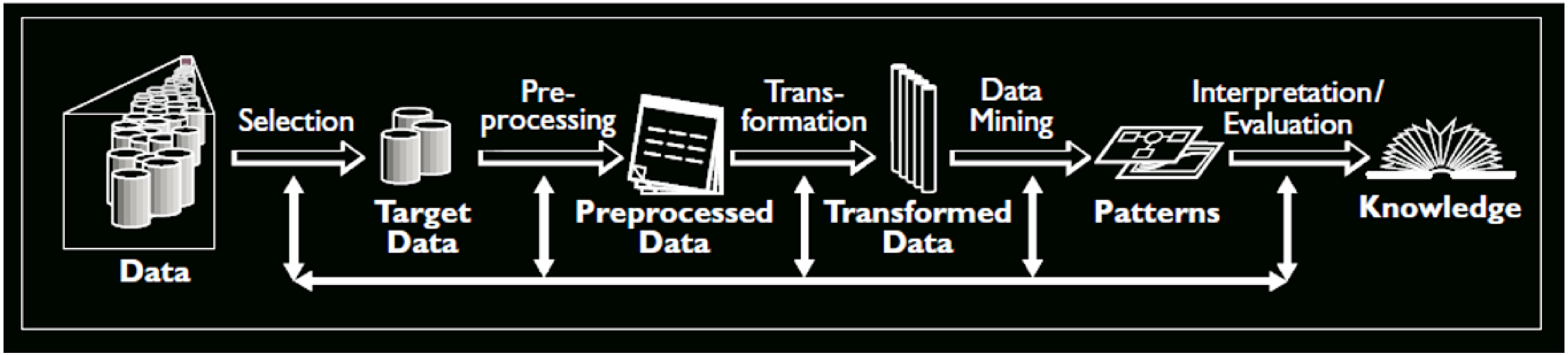
Why we selected VGG16, VGG19 and InceptionV3

- Because these models were the winner of "The ImageNet Challenge Competition" in 2015.
- And, they have better accuracy and less computational time.

Now, Vikas will explain about methodology

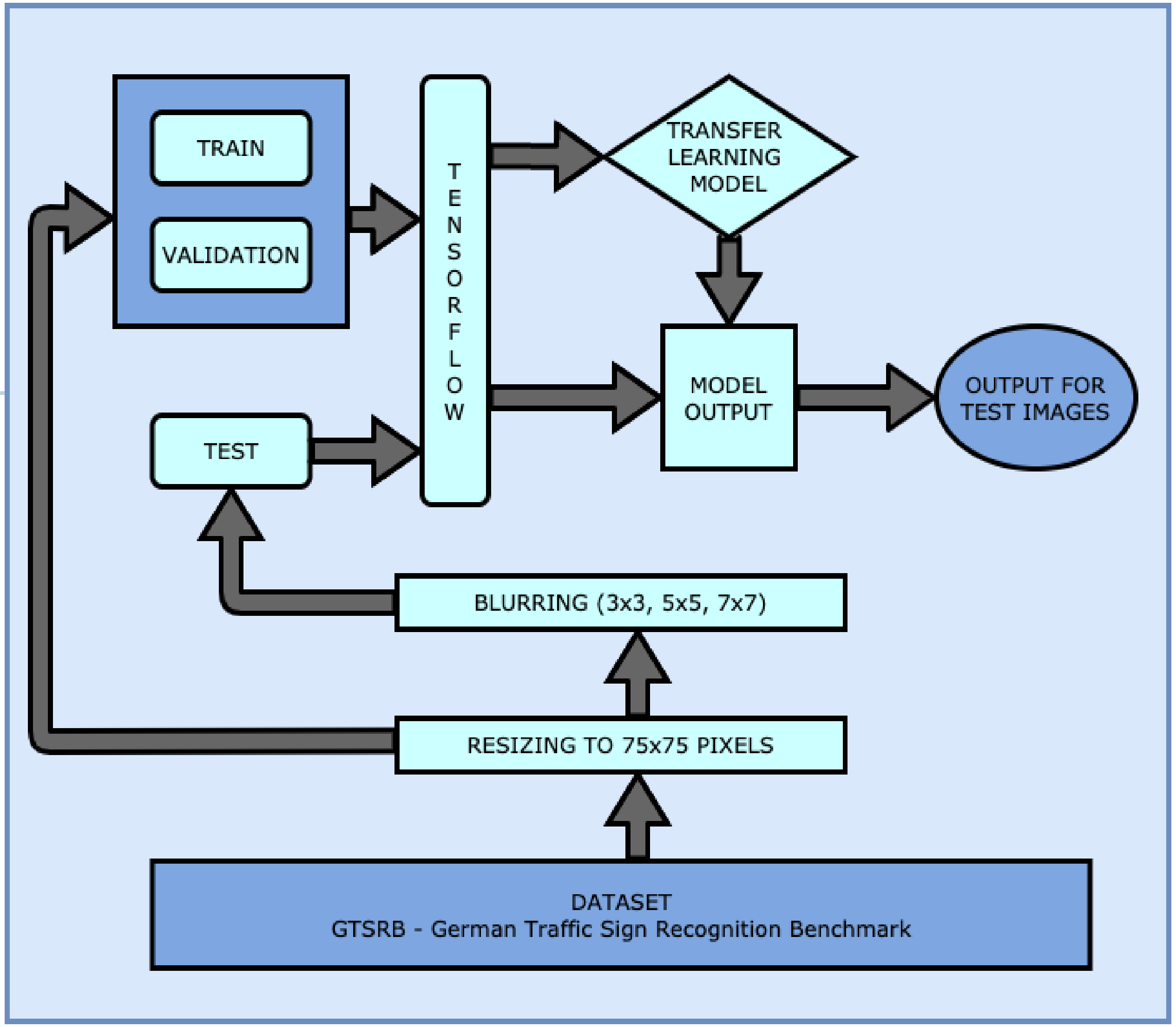
Methodology

Knowledge Discovery in Databases (KDD)



Process Design and Tasks:

Reading	Reading Images and Labels
Resizing	Resizing Images
Blurring	Blurring Images
Saving	Saving the Images in Arrays
Splitting	Splitting Data in Train, Test, and Validation Sets.

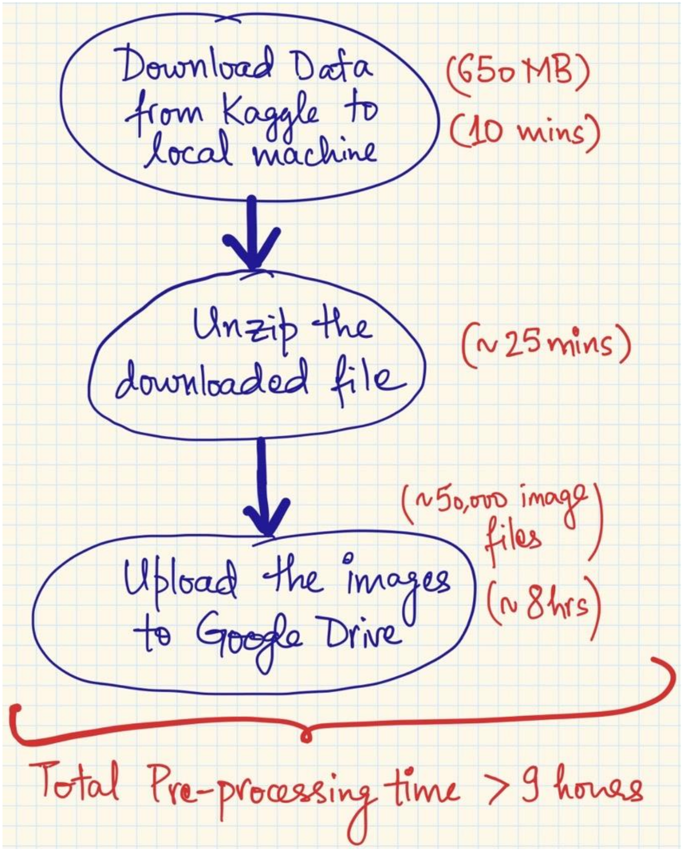


Reading the Images

- Library: CV2
- Three Approaches:
  - Approach 1
  - Approach 2
  - Approach 3
- Required time for pre-processing is improved from more than 9 hours to less than 1 minute.

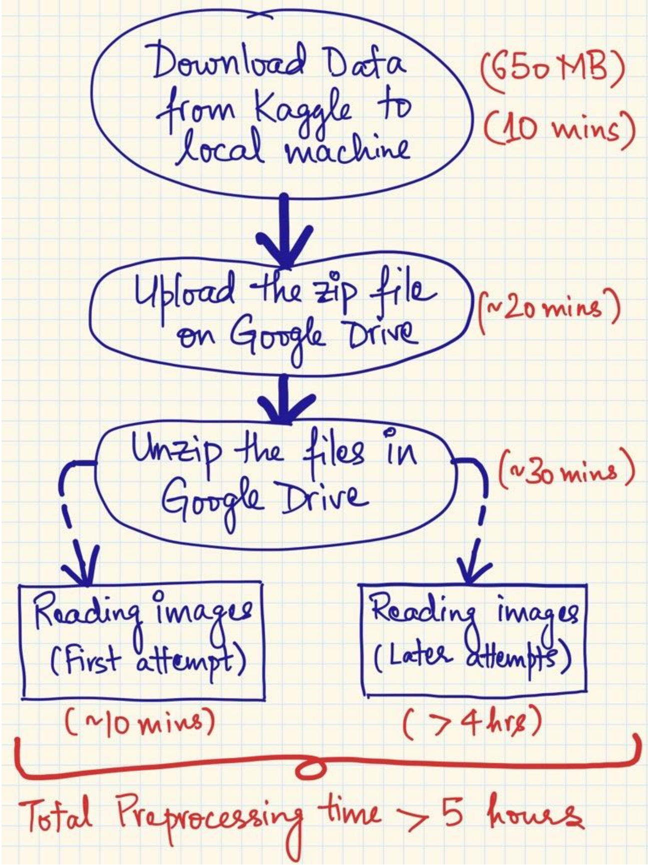
Approach 1 of Reading Images

- Download data from Kaggle to local machine.
  - Data Size: Around 650 MB
  - Time required: Around 10 minutes
- Unzip the downloaded file.
  - Time required: Around 25 minutes.
- Upload the images to Google Drive.
  - Number of files: 51,888
  - Time Estimated: >8 hours
- Total Pre-processing Time: >9 hours



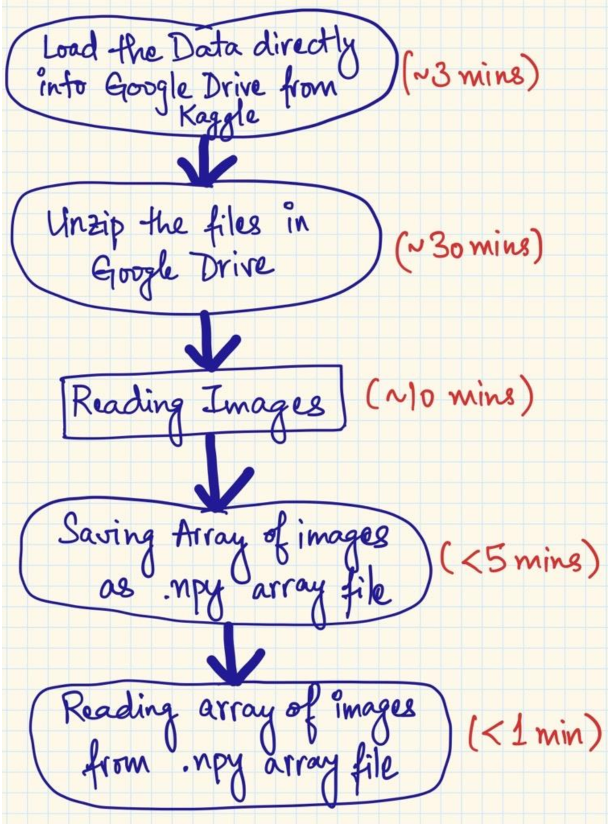
Approach 2 of Reading Images

- Challenge: Reduce the time required for uploading images to Google Drive
- Download data from Kaggle to local machine
  - Data Size: Around 650 MB
  - Time required: Around 10 minutes
- Upload the .zip data file to Google Drive
  - Time required: Around 20 minutes
- Unzip the files in Google Drive
  - Time required: Around 30 minutes
- Reading images (First Attempt):
  - Time required: Around 10 minutes
- Reading images (Second Attempt):
  - Estimated Time: >4 hours
- Total Pre-processing Time (Second Attempt): >5 hours



Approach 3 of Reading Images

- Challenge: Reduce the images reading time on second onward attempts
- Load data to Google Drive from Kaggle
  - Time required: Around 3 minutes
- Unzip the files in Google Drive
  - Time required: Around 30 minutes
- Reading images
  - Time required: Around 10 minutes
- Saving Array of Images as .npy file
  - Time required: <5 mins
- Reading array of Images from .npy file
  - Time required: < 1 min



Summary of Reading Images

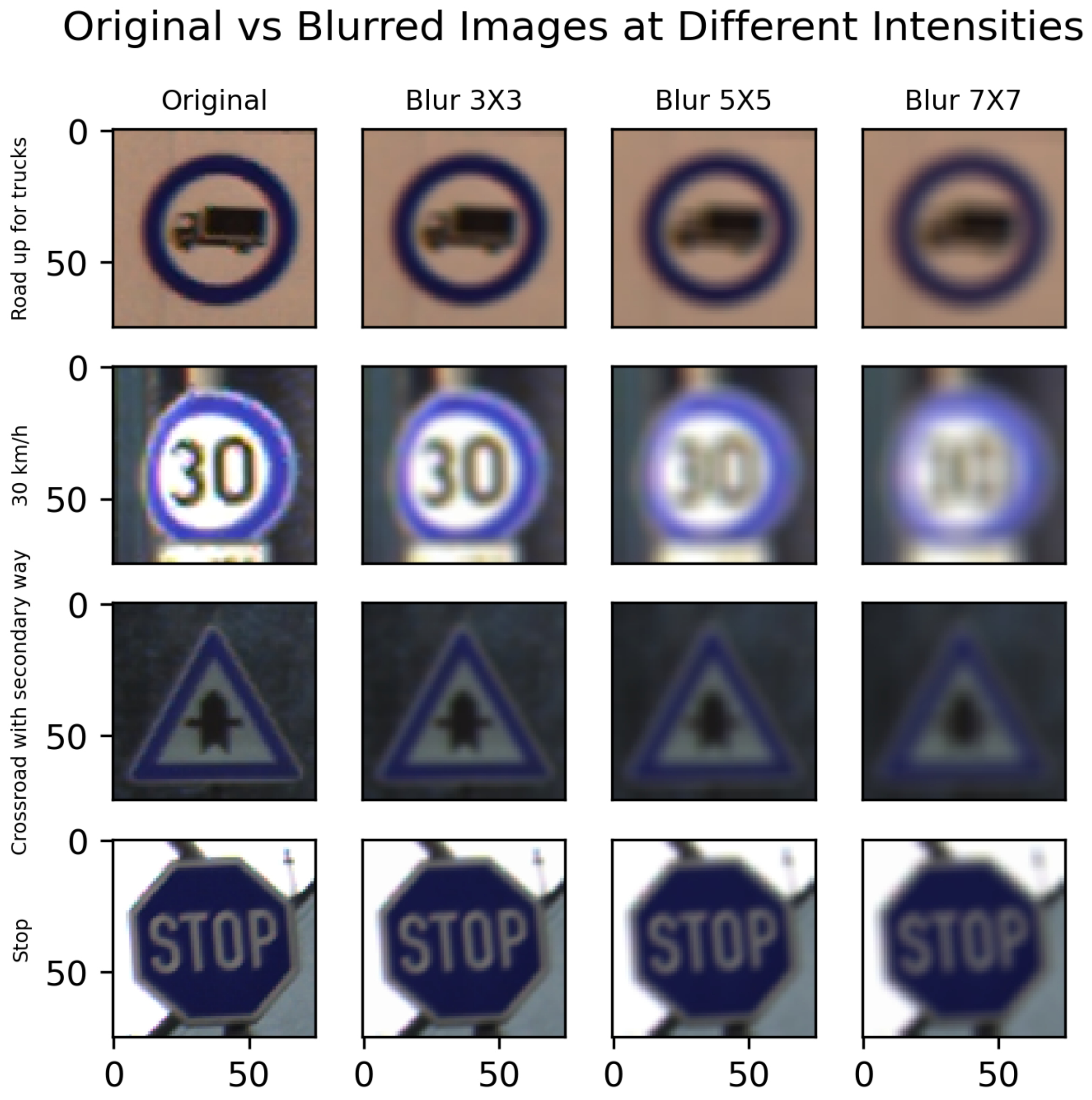
- First Attempt:
  - Load data to Google Drive from Kaggle: Around 3 minutes
  - Unzip the files in Google Drive: Around 30 minutes
  - Reading images: Around 10 minutes
  - Saving Array of Images as .npy file: Around 5 minutes
  - Total time: Around 50 minutes
- Subsequent Attempts:
  - Reading array of Images from .npy file: < 1 min

Resizing the Images

- Library: CV2
- Dataset Images has various sizes such as 28x29, 97x96, 125x136, 168x180, etc.
- Transfer Learning Models require minimum size of 32x32.
- Resized images to 75x75 pixels.

Blurring the Images

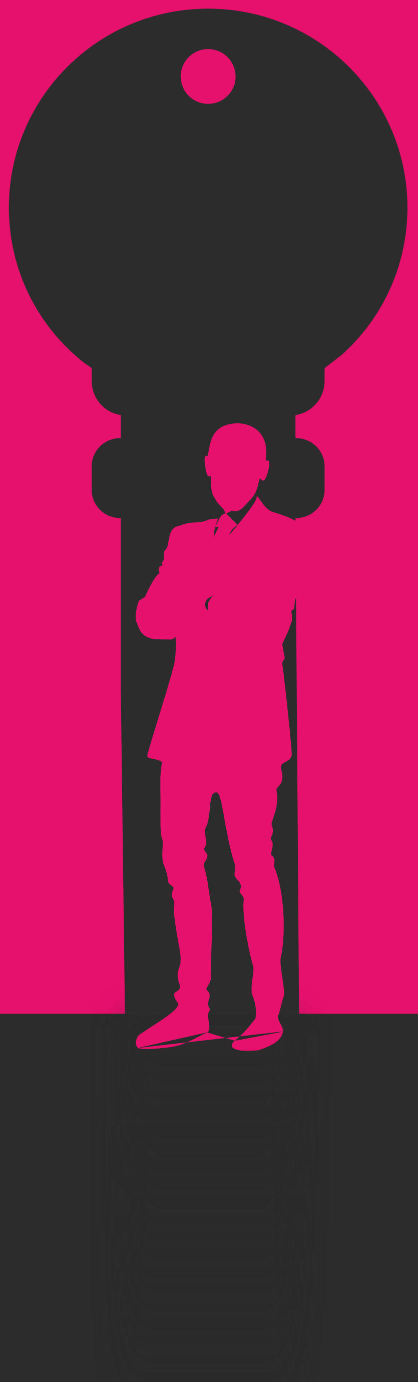
- Library: CV2
- Technique: Gaussian Blur
- Levels of Blurring:
  - 3x3
  - 5x5
  - 7x7



Splitting of Data

- Data available in Train and Test Subsets
- Train subset has 75% data
- Test subset has 25% data
- Train subset is further divided in two subsets
  - Train: 80% data
  - Validation: 20% data

PROJECT OUTLINE



VISHAL SHAKYA
x19182732@student.ncirl.ie

CONTENTS

CNN MODEL

```python
# Definition of the CNN model

self_model = Sequential()
self_model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
                      input_shape=X_train.shape[1:]))
self_model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
self_model.add(MaxPool2D(pool_size=(2, 2)))
self_model.add(Dropout(rate=0.25))
self_model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
self_model.add(MaxPool2D(pool_size=(2, 2)))
self_model.add(Dropout(rate=0.25))
self_model.add(Flatten())
self_model.add(Dense(256, activation='relu'))
self_model.add(Dropout(rate=0.5))
self_model.add(Dense(43, activation='softmax'))
```

Convolutional Layer

A fundamental layer which performs intensive processing.

Pooling Layer

Used to reduce the computational requirement.

Flattening Layer

Connector between feature and classifier layer.

CNN Model

Created by using 3 Convolutional Layers.

CLASSIFIER LAYER

```
# Classifier Layer.

model_classifier = layers.Flatten()(model_last_layer)
model_classifier = layers.Dense(1024, activation='relu')(model_classifier)
model_classifier = layers.Dropout(0.2)(model_classifier)
model_classifier = layers.Dense(classes, activation='softmax')(model_classifier)
```

Flatten Layer

Converts Multi-dimensional to one-dimensional.

Dense Layer

To reduce all features to 1024 using relu.

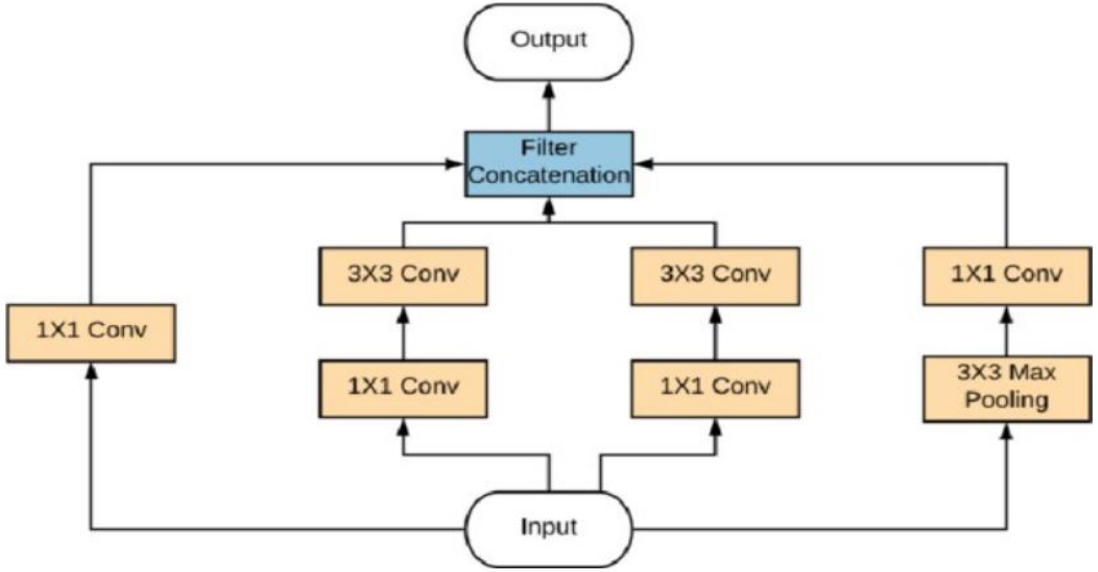
Dropout Layer

Barrier between dense layers..

Dense Layer

To predict all the 43 categories using Softmax.

INCEPTIONV3



History

Started by GoogLeNet in 2015

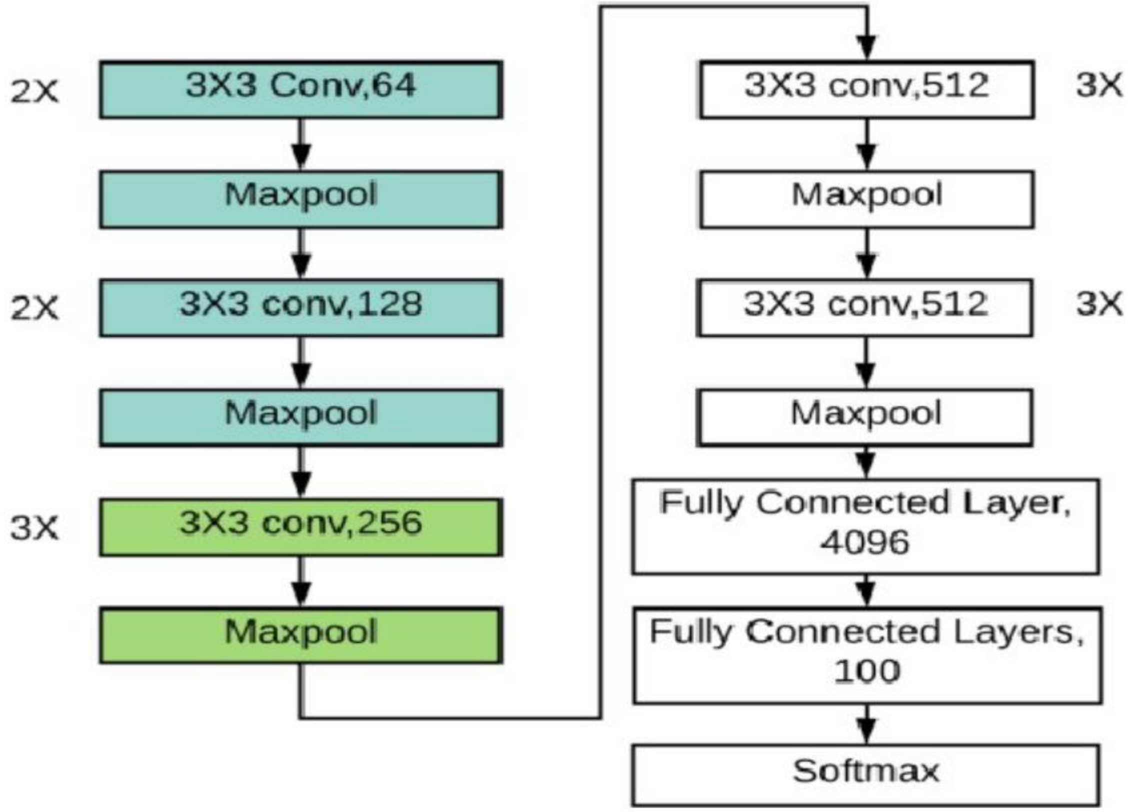
Model

It is a part of Transfer Learning model.

Layers

It is 48 layers deep.

VGG16



History

Started in 2014 by Oxford.

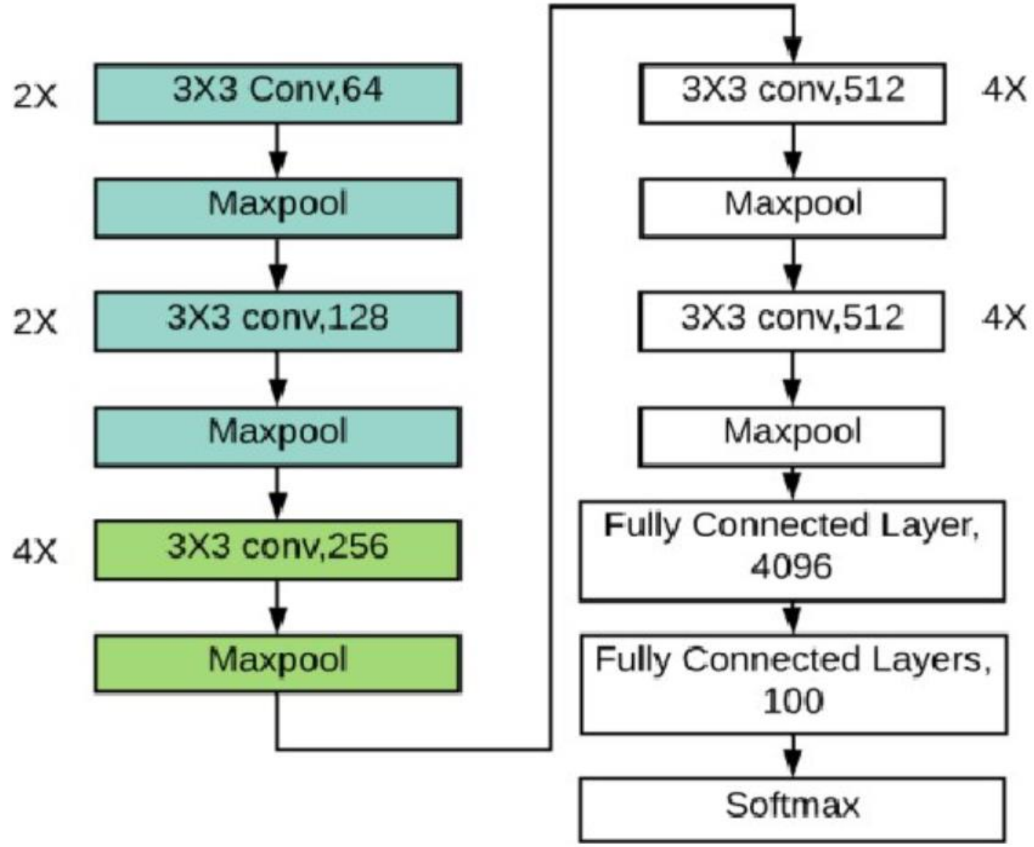
Model

It is a part of Transfer Learning model.

Layers

Shown in the diagram

VGG19



History

Started in 2012 by AlexNet.

Model

It is a part of Transfer Learning model.

Layers

Shown in the diagram

Thank You

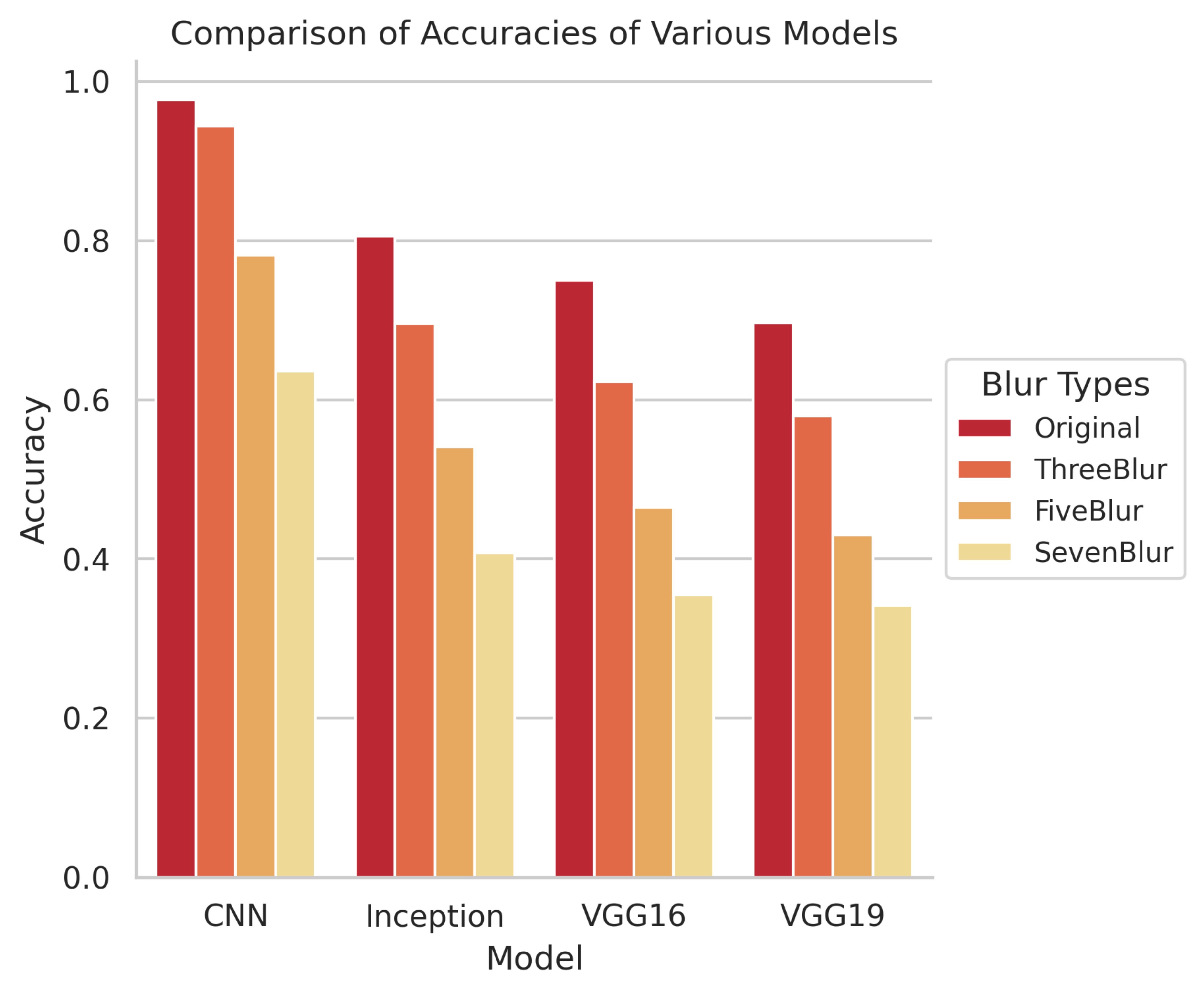
PROJECT RESULTS

AMANDEEP SINGH

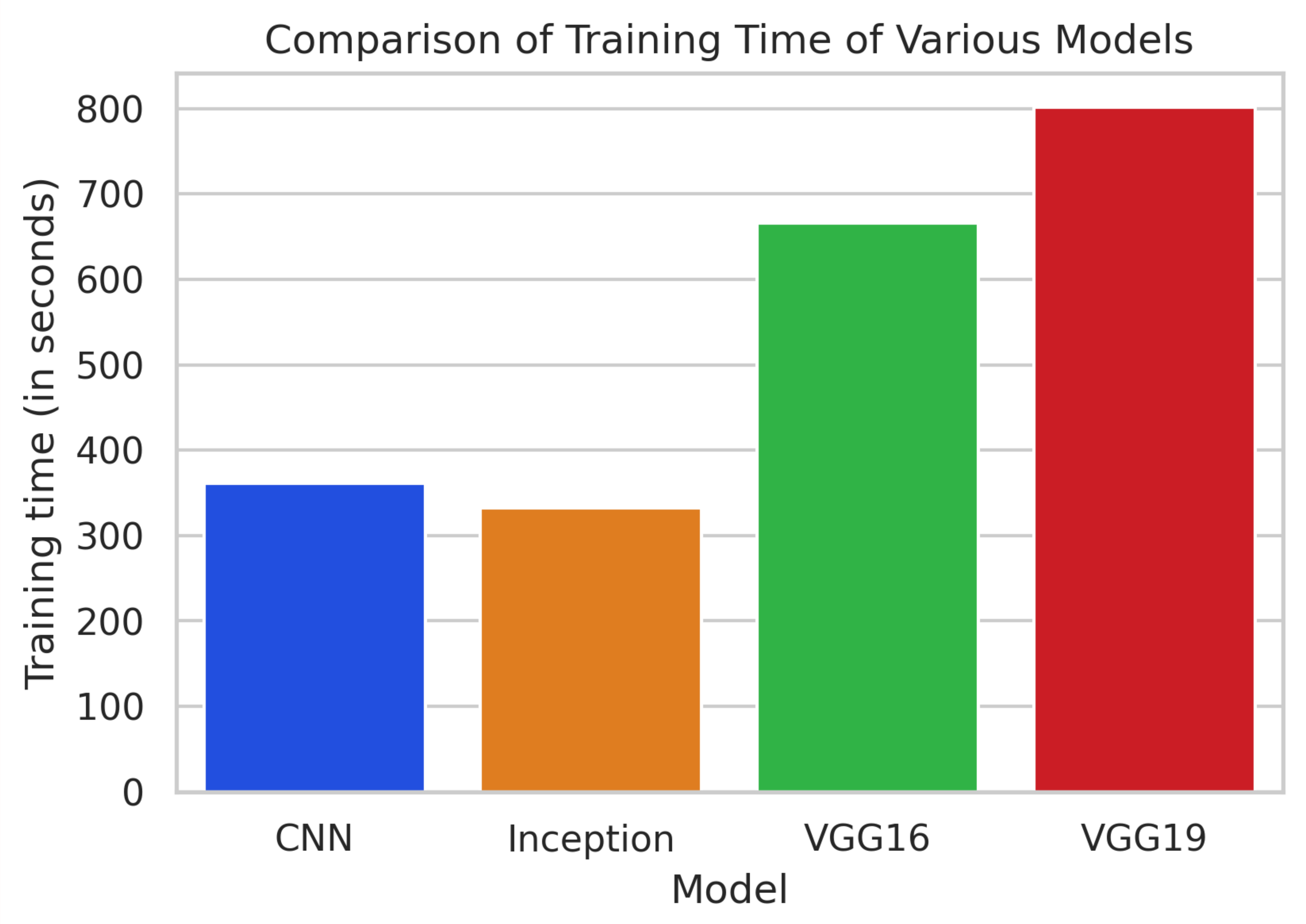
Four models were implemented:

1. Self-designed, simple, CNN
2. InceptionV3 (pre-trained TL model)
3. VGG16 (pre-trained TL model)
4. VGG19 (pre-trained TL model)

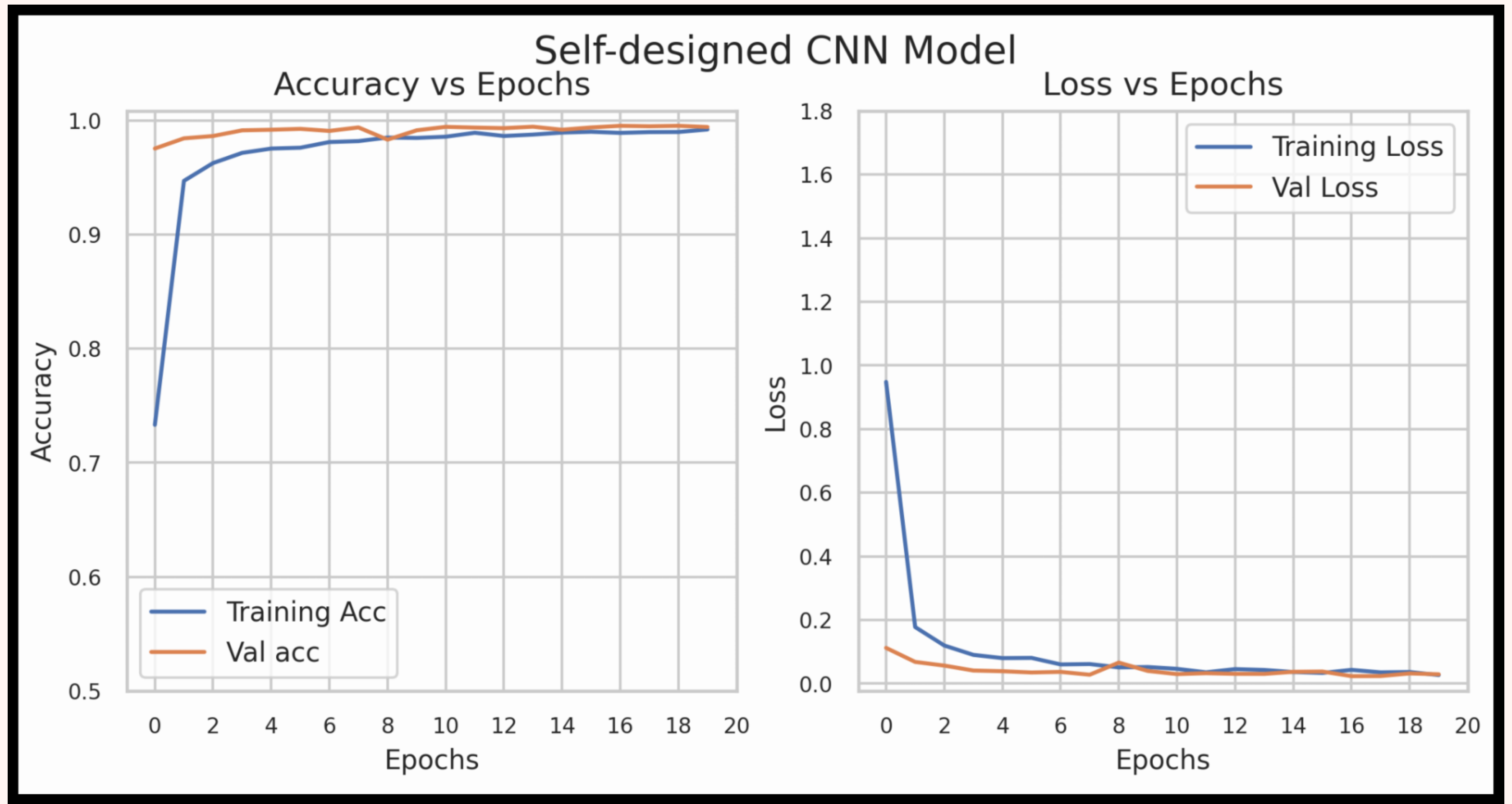
ACCURACY SCORES COMPARISON

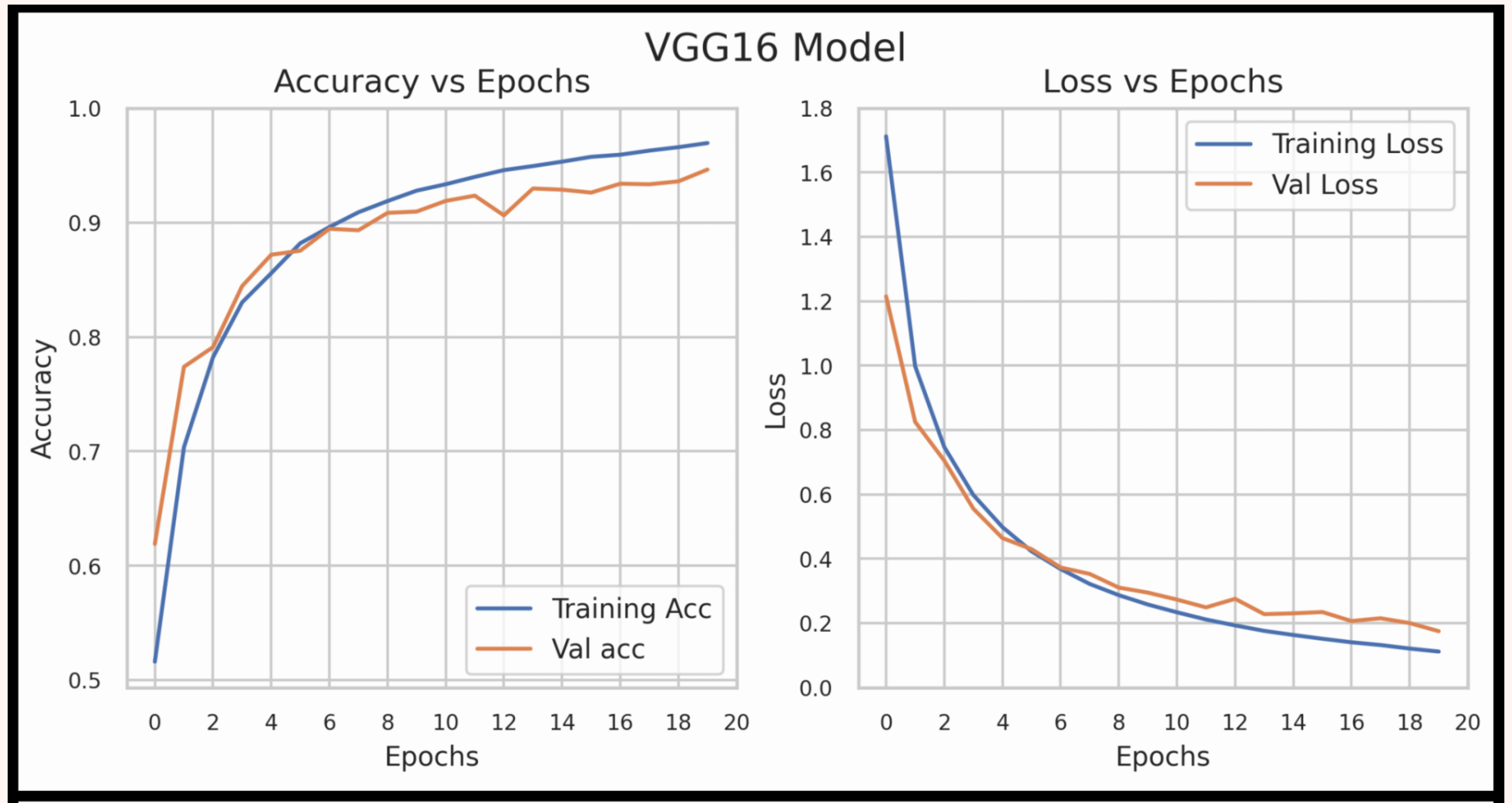


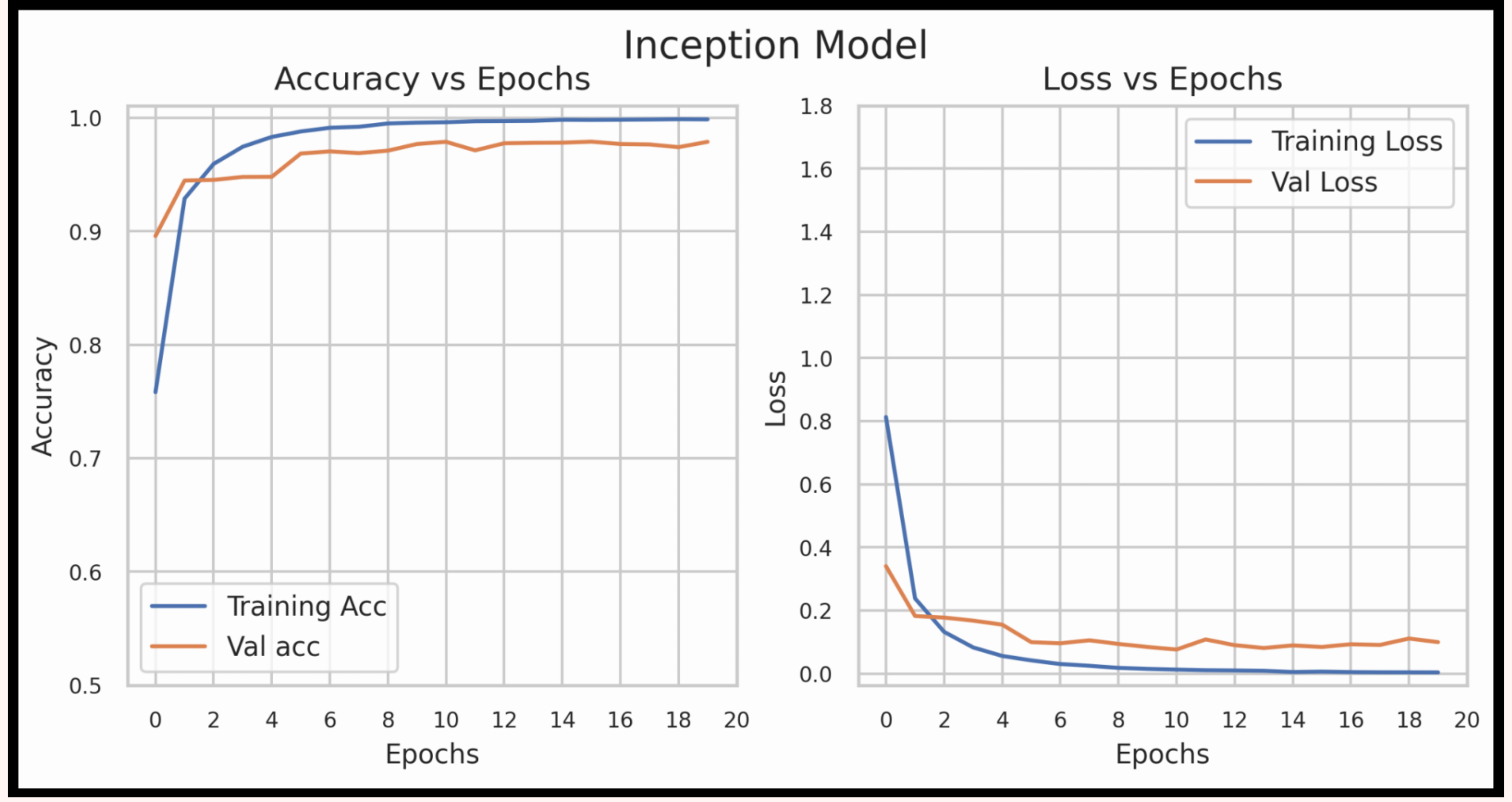
TRAINING TIME COMPARISON

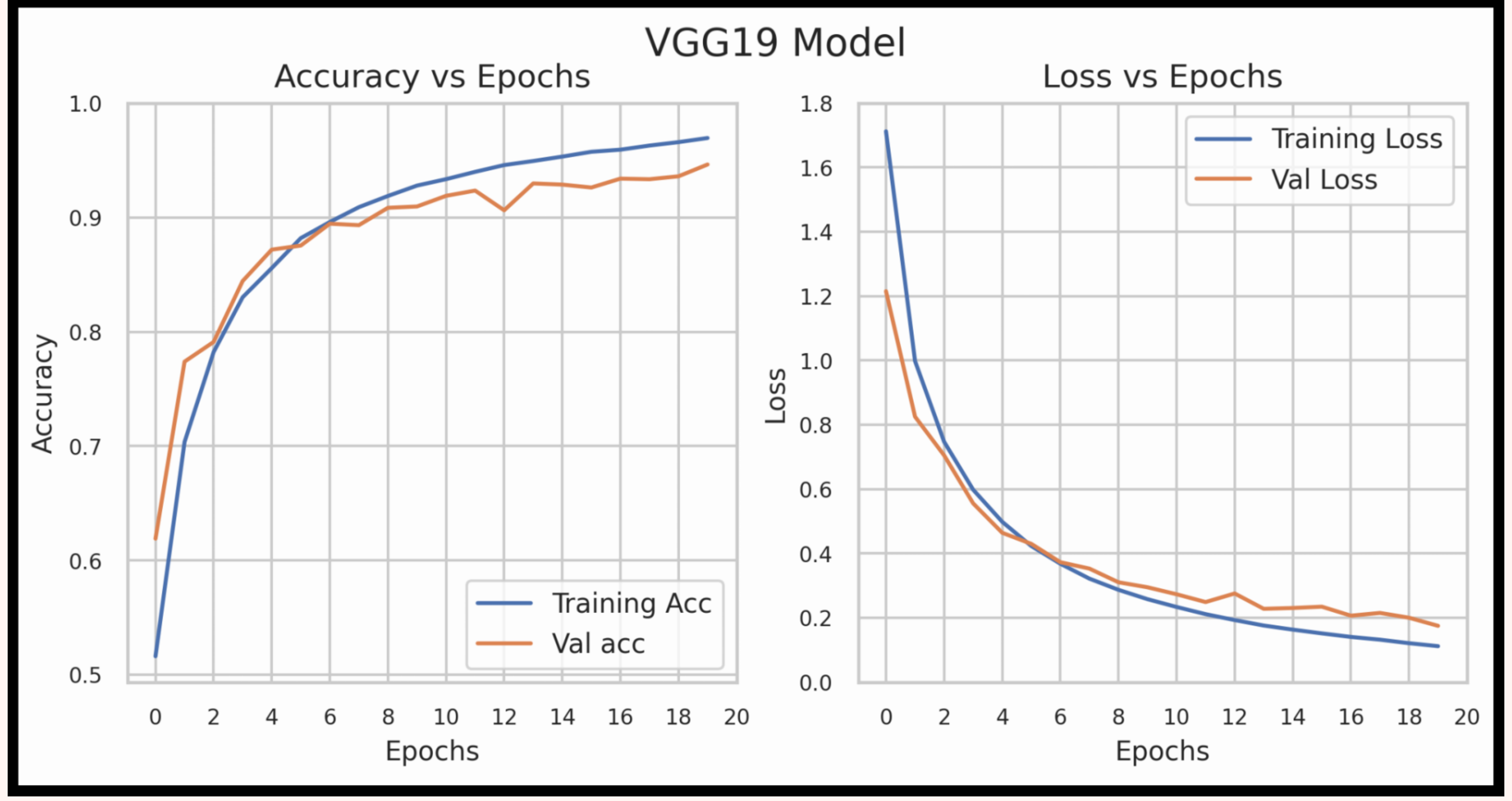


ACCURACY/LOSS VS EPOCHS COMPARISON









TOP/BOTTOM FIVE CLASS-WISE F1-SCORE COMPARISON

Models		Accuracy (%)
Self-Designed CNN	Original	97.71
	Blur 3x3	94.38
	Blur 5x5	78.21
	Blur 7x7	63.62
InceptionV3	Original	80.59
	Blur 3x3	69.54
	Blur 5x5	54.07
	Blur 7x7	40.79
VGG16	Original	75.02
	Blur 3x3	62.27
	Blur 5x5	46.51
	Blur 7x7	35.47
VGG19	Original	69.66
	Blur 3x3	57.97
	Blur 5x5	43.01
	Blur 7x7	34.15

Models	Training Time
CNN	361.44 s
InceptionV3	332.37 s
VGG16	665.41 s
VGG19	801.50 s