Capstone Project 3

Project Title

“Traffic Sign Recognition”

Project Proposed by

Anshul Bhardwaj

Project Mentor

Gritank Dhamija

Problem Statement

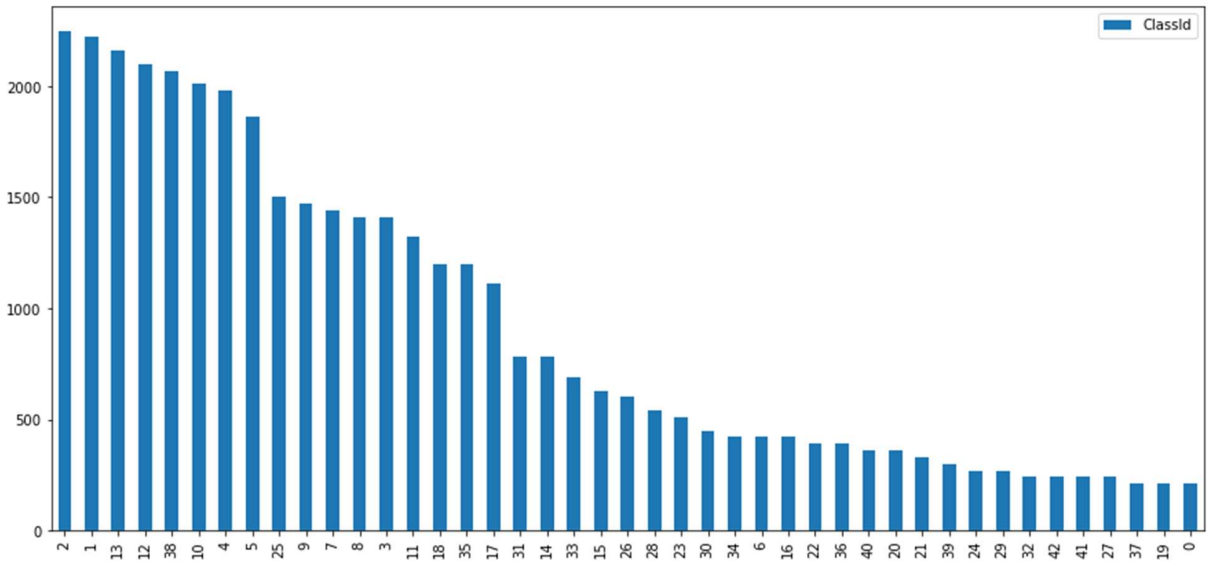
Automated Traffic Sign recognition is an important part of self-driving vehicles.

Traffic Signs can provide range of variations between classes in terms of colors and shape. In this project, I develop Deep Learning algorithms that will train on German Traffic Signs image dataset and them use these algorithms to classify unlabeled Traffic Signs images. The deep learning models will be built using Convolutional Neural Network and Transfer Learning.

Data Wrangling & EDA

The image dataset provided Kaggle contained 39029 train images and 12630 test images.

A total of 43 classes of images were there in dataset. The image dataset had the following distribution



- Class 2 images had the highest value count of 2250.
- Class 0 images had the lowest value count of 210.

Traffic Sign Meta Images

Augmenting the Images

```
start_time = time.time()

seq = iaa.Sequential([
    iaa.Affine(rotate=(-20, 20), random_state=42, shear = (10,-10))
])


X_aug = seq(images=X_train)
X_train = np.concatenate((X_train, X_aug))

print("Time taken to run this cell =  %s seconds" % (time.time() - start_time))
```

To double the number of images for training, I did Affine transformations on the images.

Examples of Image Augmentation:

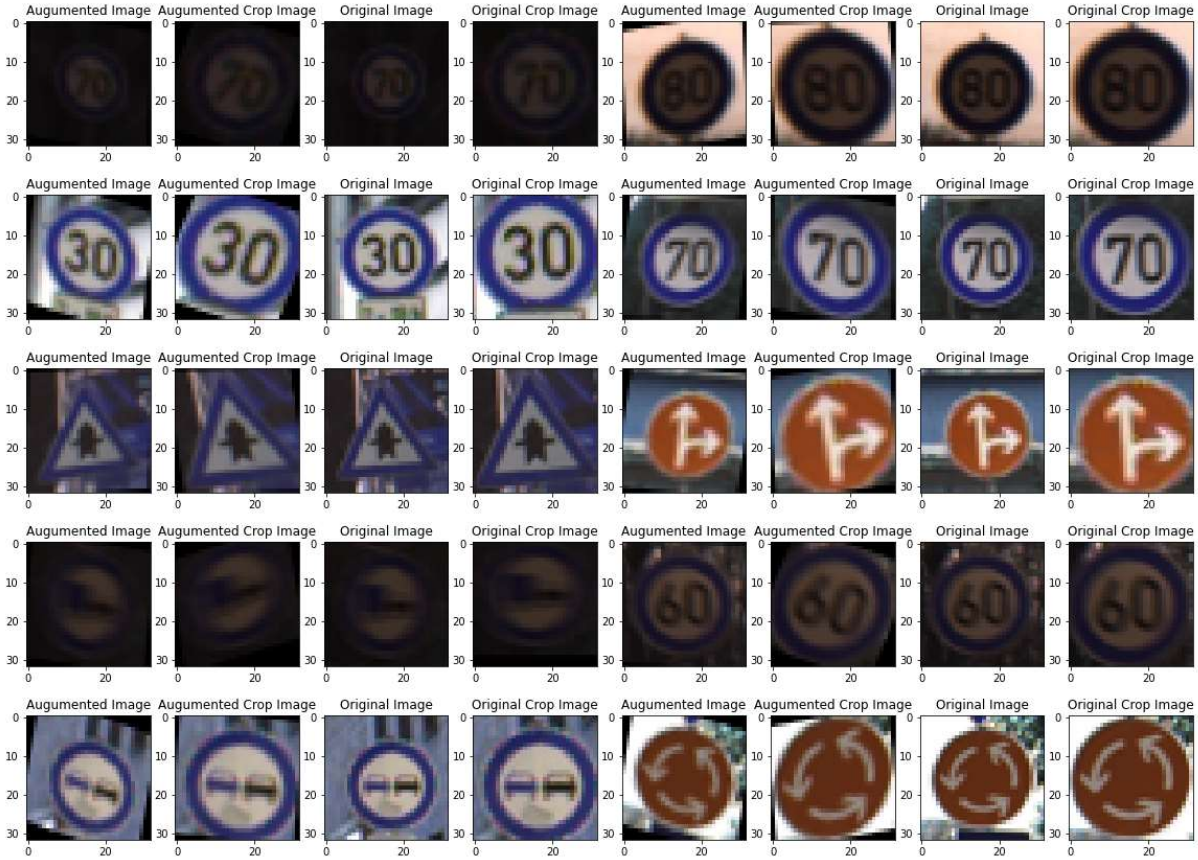


Image Generators

To reduce ram uses Image Generators were created using Keras Image generator function.

```
datagen_train = ImageDataGenerator(
    samplewise_center=True,
    samplewise_std_normalization=True
)

datagen_val = ImageDataGenerator(
    samplewise_center=True,
    samplewise_std_normalization=True)

datagen_train.fit(X_train, seed=42)
datagen_val.fit(X_val, seed=42)
```

Model Training

I created three Deep Learning models. Two of them were made with the help of transfer learning and the remaining was made with the help of Convolution Neural Networks (CNN).

- First Model was made with the help of Keras' ResNet50V2 Transfer Learning Model
- Second Model was made with the help of CNN.
- Third Model was made with the help of VGG19 Transfer Learning Model.

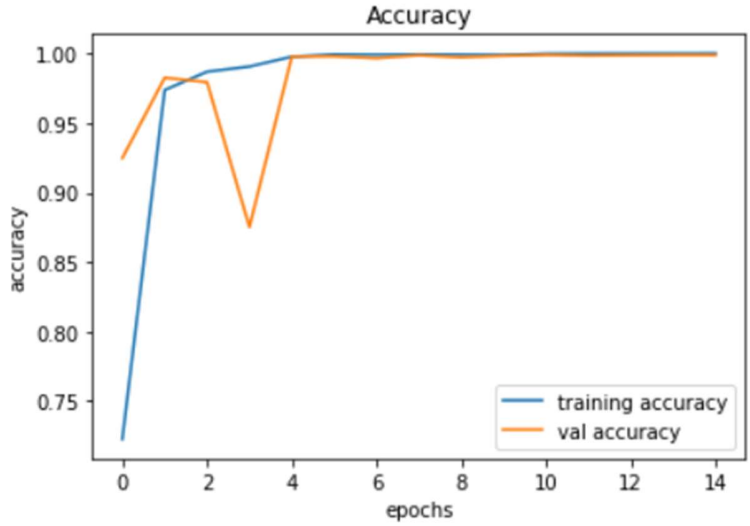
Model 1: Transfer Learning using ResNet152V2 Model

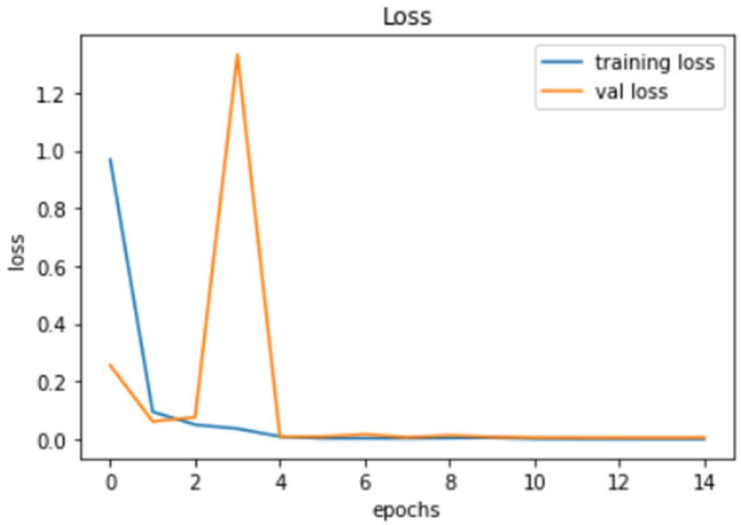
I used the following layers to train my model

```
from tensorflow.keras.applications import ResNet50V2
base_model_ResNet50V2 = ResNet50V2(include_top = False, weights= 'imagenet
', input_shape = (d, d, 3), classes = Y_train.shape[1])

#Adding the final layers to the above base models where the actual classif
ication is done in the dense layers
model1= Sequential()
model1.add(base_model_ResNet50V2)
#Adding the Dense layers along with activation and batch normalization
model1.add(Flatten())
model1.add(Dense(1024, activation='relu'))
model1.add(Dropout(rate=0.5, seed=42))
model1.add(Dense(43, activation='softmax'))

model1.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'],
)

monitoring1 = 'val_accuracy'
filepath="bestResNet50V2ModelWeights.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor=monitoring1, verbose=0, sav
e_best_only=True, mode='max')
earlystop = EarlyStopping(monitor = monitoring1, min_delta = 0, patience =
4, verbose = 1,restore_best_weights = True)
reduce_lr = ReduceLROnPlateau(monitor=monitoring1, factor=0.2, verbose = 1
, patience=2, min_lr=0.000001)
callbacks_list = [checkpoint, earlystop, reduce_lr]

epochs = 50

history1 = model1.fit(datagen_train.flow(X_train, Y_train, batch_size=256)
, epochs=epochs, verbose=1, validation_data=datagen_val.flow(X_val, Y_val)
,
                      shuffle=False, callbacks=callbacks_list, use_multipr
ocessing=True, workers=-1)
```

- Optimizer used – ADAM

Model History –





Model 2: Using CNN

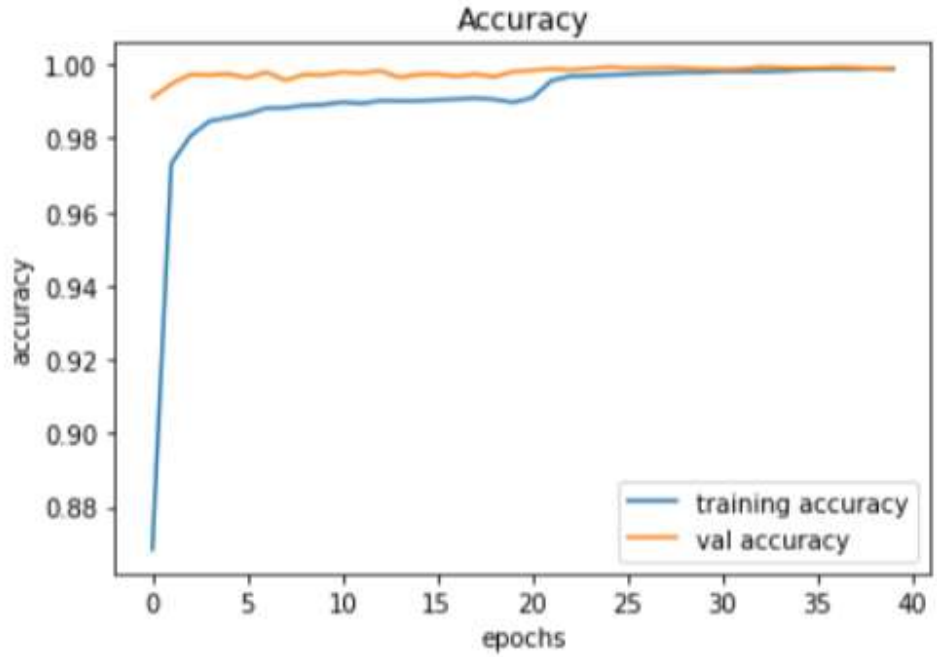
I used the following architecture to build my second model

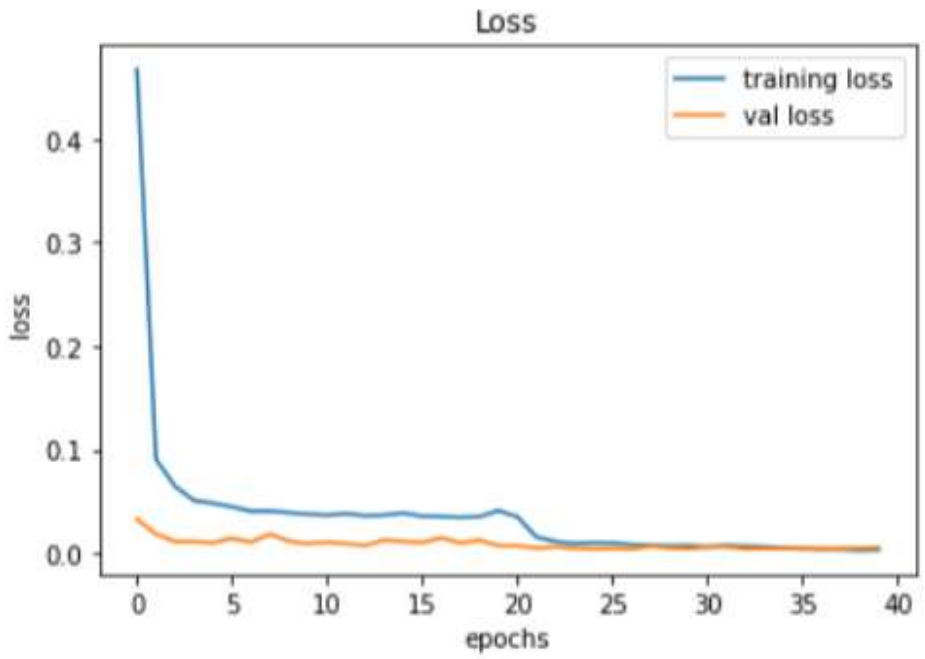
```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_s
hape=(d, d, 3)))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25, seed=42))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Dropout(rate=0.25, seed=42))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5, seed=42))
model.add(Dense(43, activation='softmax'))

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'],
)
```

```
monitoring2 = 'loss'
filepath="bestCNNModelWeights2.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor=monitoring2, verbose=0, sav
e_best_only=True, mode='min')
earlystop = EarlyStopping(monitor = monitoring2, min_delta = 0, patience =
4, verbose = 1,restore_best_weights = True)
reduce_lr = ReduceLROnPlateau(monitor=monitoring2, factor=0.2, patience=3,
min_lr=0.000001, verbose = 1)
callbacks_list = [checkpoint, earlystop, reduce_lr]

epochs = 40

history = model.fit(datagen_train.flow(X_train, Y_train, batch_size=128),
epochs=epochs, verbose=1,
                    callbacks=callbacks_list, shuffle=False, validation_da
ta = datagen_val.flow(X_val, Y_val))
```

Model 2 History



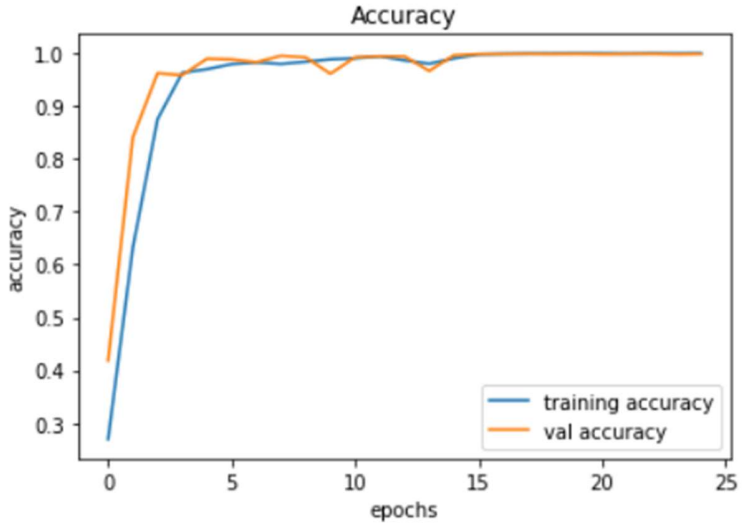


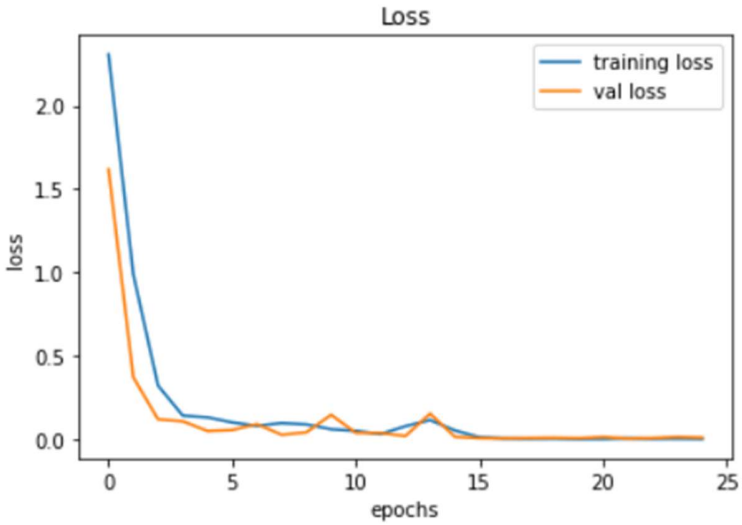
Model 3: Transfer Learning using VGG19

I used the following architecture to build my model 3.

```
from tensorflow.keras.applications import VGG19
base_model_VGG19 = VGG19(include_top = False, weights= 'imagenet', input_s
hape = (d, d, 3), classes = Y_train.shape[1])

model3= Sequential()
model3.add(base_model_VGG19)
#Adding the Dense layers along with activation and batch normalization
model3.add(Flatten())
model3.add(Dense(1024, activation='relu'))
model3.add(Dropout(rate=0.75, seed=42))
model3.add(Dense(43, activation='softmax'))

model3.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'],
)

monitoring3 = 'loss'
filepath="bestVGG19ModelWeights2.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor=monitoring3, verbose=0, sav
e_best_only=True, mode='min')
earlystop = EarlyStopping(monitor = monitoring3, min_delta = 0, patience =
4, verbose = 1,restore_best_weights = True)
reduce_lr = ReduceLROnPlateau(monitor=monitoring3, factor=0.2, patience=3,
min_lr=0.000001, verbose=1)
callbacks_list = [checkpoint, earlystop, reduce_lr]


history3 = model3.fit(datagen_train.flow(X_train, Y_train, batch_size=256)
, epochs=epochs, verbose=1, validation_data=datagen_val.flow(X_val, Y_val)
,
                      shuffle=True, callbacks=callbacks_list)
```

Model 3 History





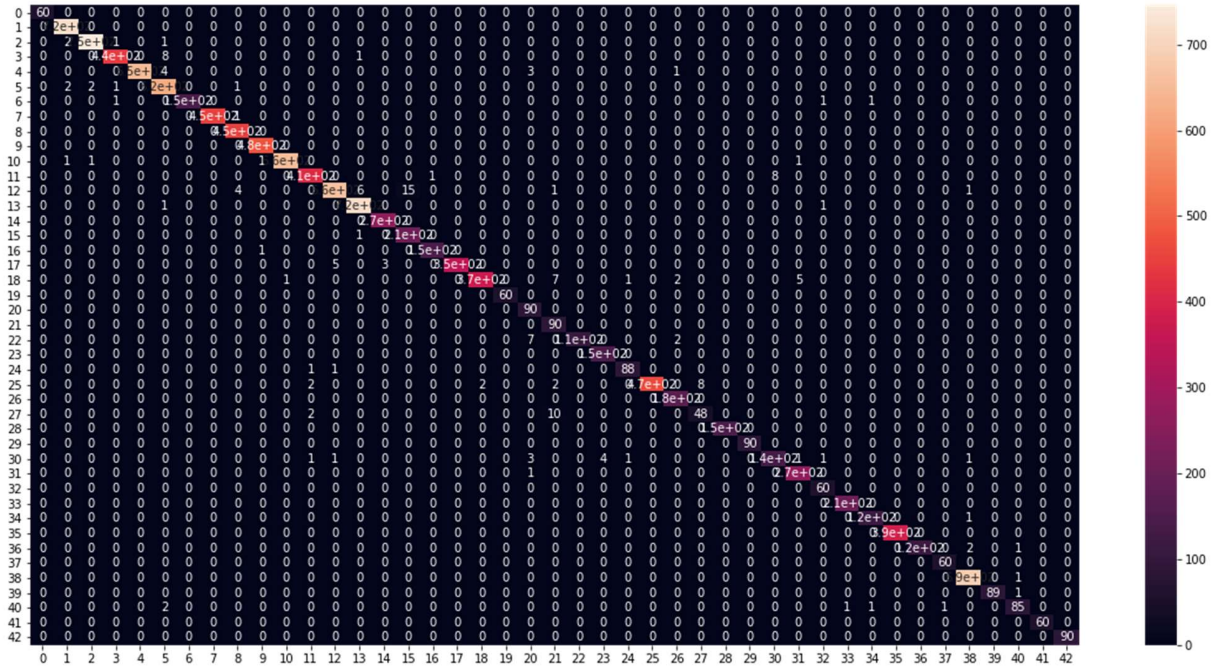
Testing the accuracies

The test accuracies of the three models were as follows –

Model Name	Accuracy
CNN Model	98.72%
ResNet50V2 Model	96.61%
VGG19 Model	98.33%

- CNN model performed the best

- Confusion Matrix Heat Map for CNN model -



Ensemble of the Models

I also create various ensembles of the above models to get more accurate model.

Weighted Ensemble model gave the accuracy of 99.1%

Mean Ensemble (Soft voting) model gave the accuracy of 99.07%

Mode Ensemble (Hard Voting) model gave the accuracy of 98.72%

Classification report for Mean Ensemble -

	precision	recall	f1-score	support
0	1.00	1.00	1.00	60
1	0.99	1.00	1.00	720
2	1.00	0.99	1.00	750
3	1.00	0.98	0.99	450
4	1.00	0.99	0.99	660
5	0.98	1.00	0.99	630
6	1.00	0.98	0.99	150

7	1.00	1.00	1.00	450
8	0.99	1.00	1.00	450
9	0.99	1.00	0.99	480
10	1.00	1.00	1.00	660
11	0.99	0.99	0.99	420
12	1.00	0.98	0.99	690
13	1.00	1.00	1.00	720
14	0.97	1.00	0.99	270
15	0.93	1.00	0.96	210
16	1.00	0.99	1.00	150
17	1.00	0.97	0.99	360
18	1.00	0.97	0.98	390
19	1.00	1.00	1.00	60
20	0.91	1.00	0.95	90
21	0.83	1.00	0.90	90
22	1.00	0.96	0.98	120
23	0.97	1.00	0.98	150
24	1.00	0.98	0.99	90
25	0.99	0.99	0.99	480
26	0.99	1.00	0.99	180
27	0.93	0.85	0.89	60
28	0.99	1.00	0.99	150
29	0.99	1.00	0.99	90
30	0.98	0.94	0.96	150
31	0.99	1.00	0.99	270
32	0.97	1.00	0.98	60
33	0.99	1.00	1.00	210
34	1.00	1.00	1.00	120
35	1.00	1.00	1.00	390
36	1.00	0.99	1.00	120
37	0.98	1.00	0.99	60
38	1.00	1.00	1.00	690
39	1.00	0.99	0.99	90
40	1.00	0.96	0.98	90
41	1.00	0.95	0.97	60
42	1.00	1.00	1.00	90
accuracy			0.99	12630
macro avg	0.98	0.99	0.99	12630
weighted avg	0.99	0.99	0.99	12630

Future Improvement Ideas:

- Do more image augmentations.
- Increase the image input dimension.
- Use more Image generators for better ram management.