Capstone Project Report: Computer Vision Model for Facial Emotion Recognition

Kaushal Nair

Executive Summary:

This project proposes a Convolutional Neural Network (CNN) to train a computer-vision model that can recognize the expression on a person's face and classify it as happy, sad, neutral and surprised. Additionally, the given project also explores the performance of three Transfer Learning Architectures i.e. VGG16, ResNet and EfficientNet on the data. The suggested model outperforms the pre-made Transfer Learning models, however, it comes with its own shortcomings such as the erratic validation accuracies during the model training as well as a tendency to misclassify the 'neutral' and 'sad' emotions as 'happy'. Thus, it is advisable that the CNN model should be trained with more layers or even replaced or combined with a Recurrent Neural Network (RNN) before implementation.

Problem Summary:

Deep Learning as a field of study has grown immensely over the last few years as it finds ways to not only solve existential problems such as fraud detection and election outcomes, but also finds itself being potentially useful for futuristic ones such as preventing car accidents via self-driving cars and reducing hospital mortality rates via robotic surgeries. Deep learning can also be used in facial emotional recognition software which has potential uses in diagnosing Down's syndrome in children, video games which change in real-time according to a player's facial expressions and surveillance purposes.

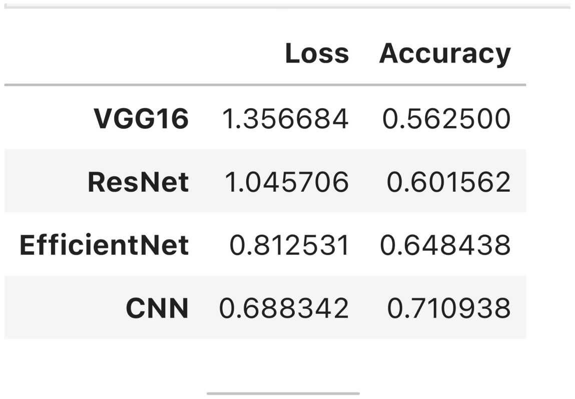
The main purpose of this project is to **design a computer vision model which can identify different emotions on a person's face**. Furthermore, this project compares and contrasts the different models such as VGG16, ResNet, EfficientNet (these are pre-trained models with the feature learning and classification layers having fixed inputs) and the CNN.

Solution Structure and Analysis:

The final proposed computer vision model is the **CNN** which has been trained using chosen parameters such as filters and type of activation function. Other models such as VGG16, ResNet and EfficientNet were explored When compared to VGG16, ResNet and EfficientNet, the CNN model had the least overfitting of the training and the validation accuracies. Additionally, the CNN had the highest test accuracy and the lowest test loss when compared to the other models. The test loss is the penalty for a misclassification.

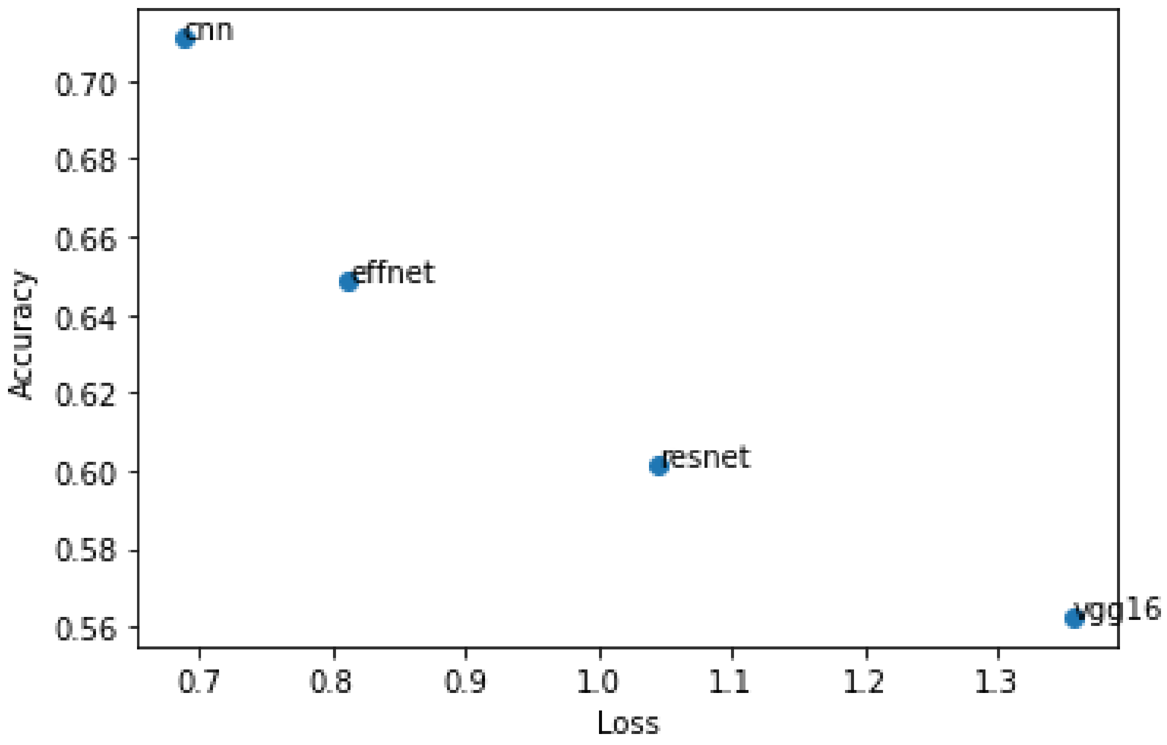
Figure 1 shows the test losses and accuracies for the different architectures that were tried for the facial emotion recognition model and **Figure 2** shows the same on a plot. From the figures, it is evident that the CNN model is able to train the data with the least amount of errors and the highest test accuracy indicates that the model is the best at predicting emotions correctly over the testing data.

Figure 1:



	Loss	Accuracy
VGG16	1.356684	0.562500
ResNet	1.045706	0.601562
EfficientNet	0.812531	0.648438
CNN	0.688342	0.710938

Figure 2:



Moreover, **Figure 3** shows the trend of the training and validation accuracy of the data when trained over 35 epochs of the CNN. The training and validation accuracies after 35 epochs of training are 0.8232 and 0.7458 respectively, which was the highest as compared to other models. Moreover, this model has the least difference between training and validation accuracies especially when compared to those of VGG16 (see **Figure 4**), ResNet (see **Figure 5**) and EfficientNet (see **Figure 6**) which shows that the validation data did not learn at a similar rate as the training data, thereby resulting in overfitting.

Figure 3 CNN:

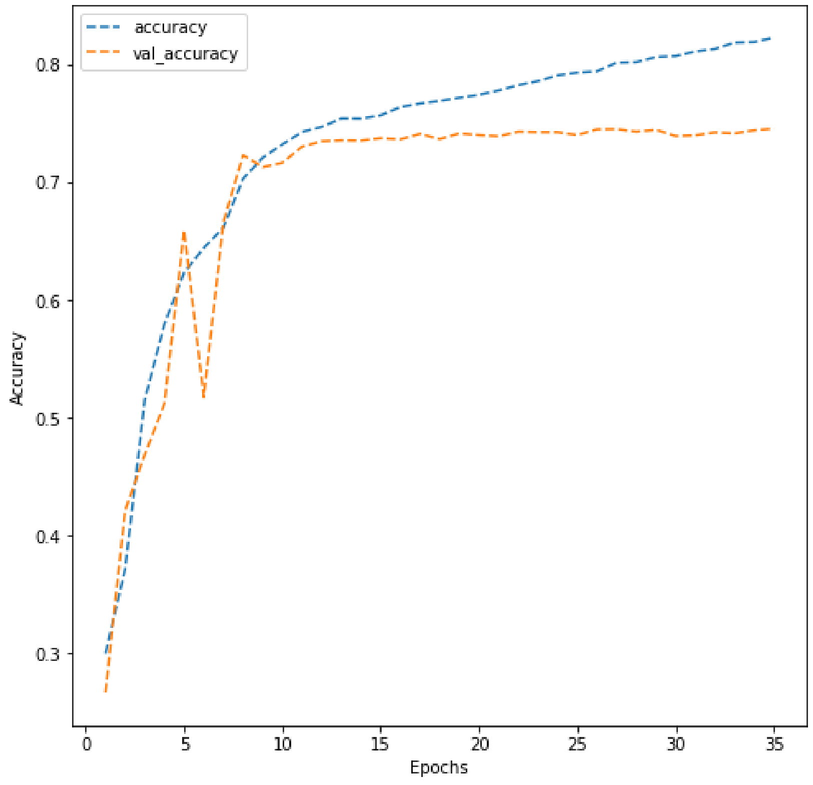


Figure 4 VGG16:

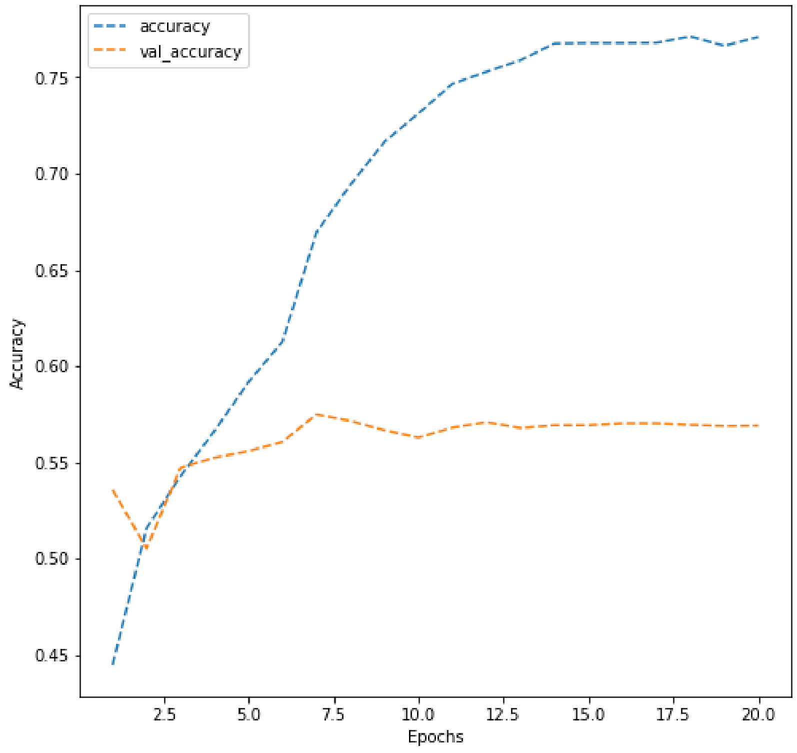


Figure 5 ResNet:

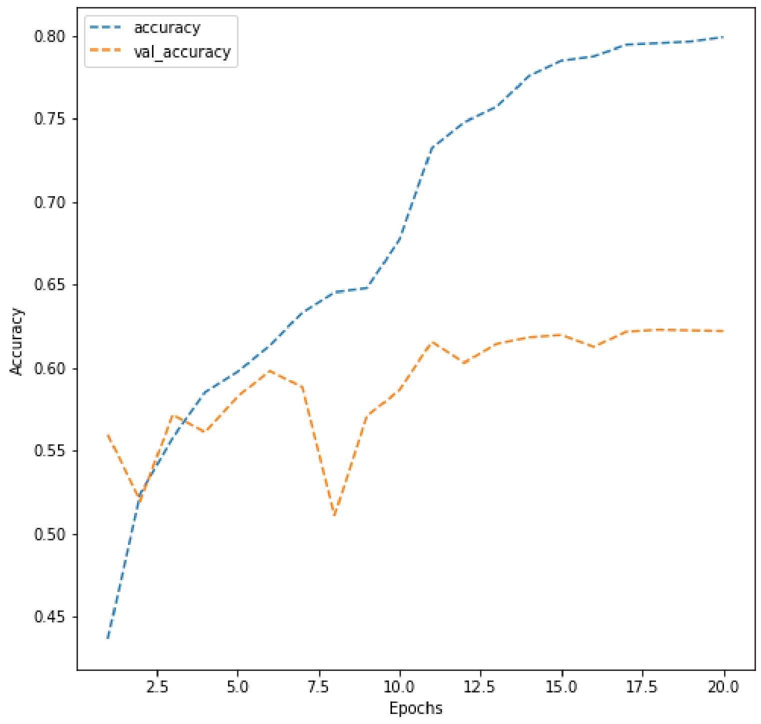
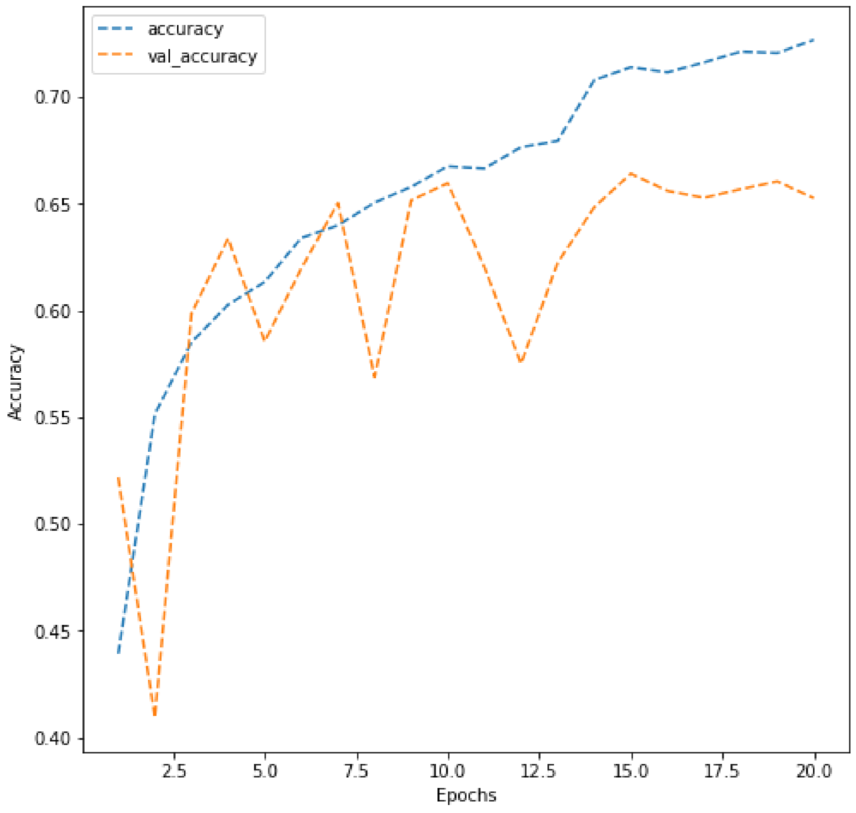
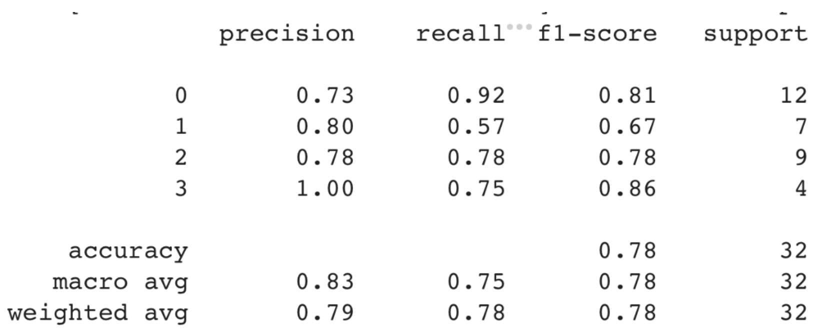


Figure 6 EfficientNet:



Additionally, we can generate more insights into the performance of the CNN model on the test data by not only checking its accuracy and loss but also generating a classification report and a confusion matrix as shown below in **Figure 7**.

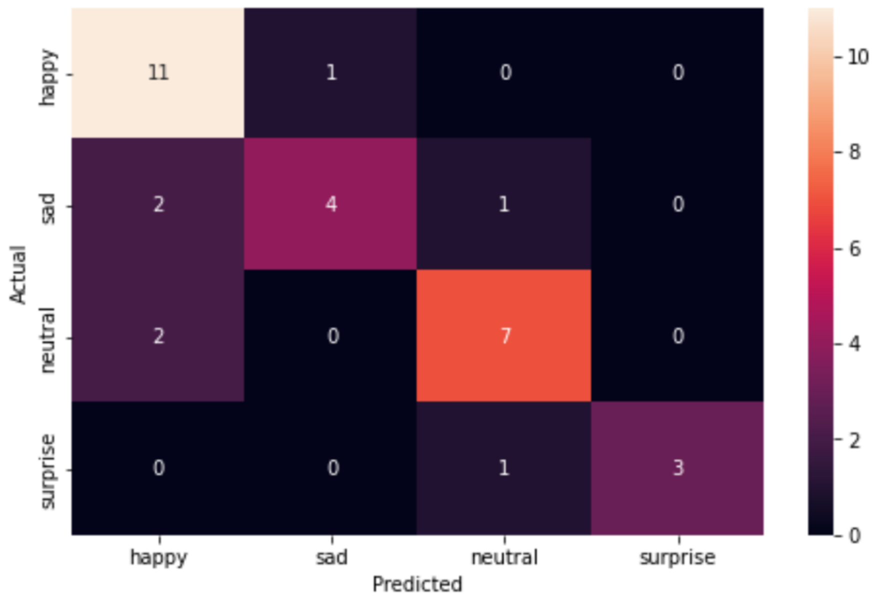
Figure 7 Classification Report and Confusion Matrix:



```
              precision    recall  f1-score   support

           0       0.73      0.92      0.81        12
           1       0.80      0.57      0.67         7
           2       0.78      0.78      0.78         9
           3       1.00      0.75      0.86         4

    accuracy                           0.78        32
   macro avg       0.83      0.75      0.78        32
weighted avg       0.79      0.78      0.78        32
```



The classification report shows that the precision values are high for all emotions (0 = Happy, 1 = Sad, 2 = Neutral and 3 = Surprise) which indicates that the CNN model correctly classifies most of the emotions for the 128 test data images.. However, the low recall value for the sad class of images indicates that the model cannot correctly classify many sad images within its own class. The recall values of the other 3 emotions is quite high, thereby indicating that within the images indicating a particular emotion, the number of correct classifications is high. Additionally, the confusion matrix indicates the percentage of instances where the actual emotion on the image is classified as the correct/incorrect one. For example, the numbers are high on the diagonal of the matrix especially for happy and neutral. Additionally, 2% of sad and neutral pictures on the test data are predicted as happy.

Implementation and Scope for Improvement:

The CNN models as well as the transfer learning architectures were initially implemented without using early stopping as a parameter in the model fit. One of the purposes of early stopping is to not only prevent overfitting on the data, but also reduce test loss. The images in **Appendix 1** to **Appendix 5** show models without early stopping and one can see the high test losses as well as overfitting.

While we used the early stopping parameter to improve our model in terms of its train and test accuracy while reducing the losses, we need to also consider a few things before this model is implemented by an industry/institution. For instance, in **Figure 3**, it is clear that there is a drop in the validation accuracy from the 5th epoch till the 6th one and this could be as a result of implementing the ReduceLROnPlateau with a patience od 2 (see **Appendix 6**) i.e. the model trains for 2 epochs before reducing the learning rate. If this parameter is modified, the validation accuracies will not fluctuate as much. Additionally, the low number of images in the validation and test sets could also contribute to the fluctuating accuracy for the Transfer learning architectures and hence, using more data could solve this problem.

Moreover, if we look at the f1 scores from **Figure 7,** we see that it is the lowest for the ‘sad’ emotion indicating that the CNN model is not very accurate when learned over the ‘sad’ images in the test data. Thus, this could be a potential risk a company has to consider before implementing this model. The model has a tendency to classify sad and neutral images as happy and, for example, if this computer vision model is being used to diagnose depression or anxiety by looking at a patient's facial emotion, then a misclassification could be a wrong diagnosis.

While CNN’s are useful for classification of images and videos it is important to note that CNN’s could be problematic especially when considering different orientations of objects. Some questions to consider would be: Would the CNN misclassify an image if a person has tilted their face especially, if the model is collecting real time data? Would an image be misclassified if a person has tattoos or nose rings on their face?

Furthermore, we could look at an RNN as they are able to deal with temporal changes in the images thereby reducing misclassification. Thus, RNN’s can be implemented to analyze video data and if our computer vision model could record a person in real time, we could potentially obtain a classification of the different emotions and hence, make better predictions. This could be a step towards building a model that can classify more emotions apart from sadness, happiness, surprise and a neutral emotion.

Bibliography:

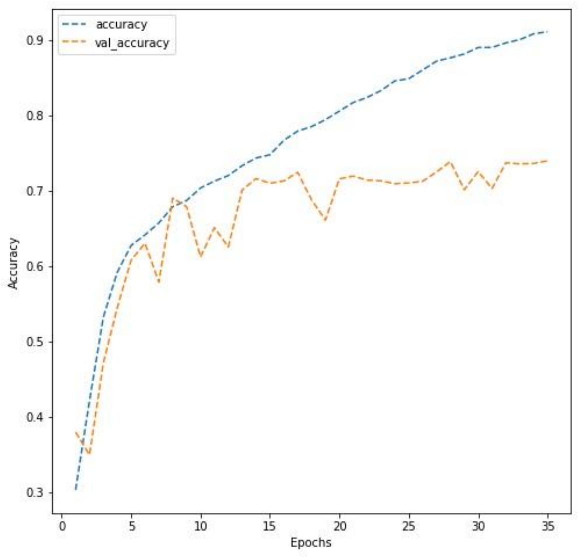
- “Difference between Ann, CNN and RNN.” GeeksforGeeks, 24 Aug. 2022, https://www.geeksforgeeks.org/difference-between-ann-cnn-and-rnn/.

Appendix:

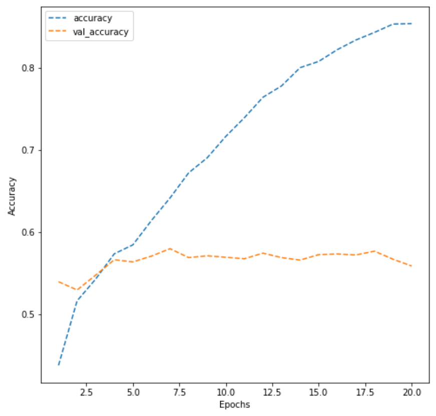
Test Loss vs Accuracy:

	Test Loss	Test Accuracy
VGG16	2.109382	0.515625
ResNet	1.057026	0.593750
EfficientNet	0.897941	0.648438
CNN	0.784626	0.710938

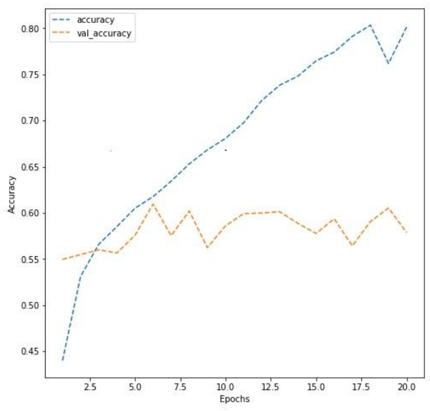
CNN:



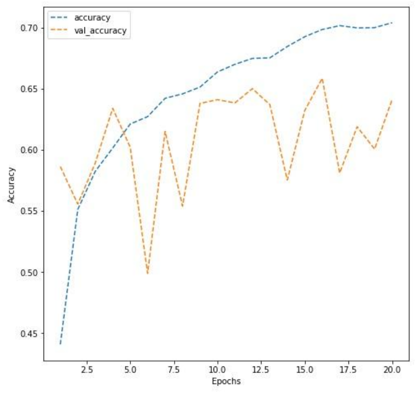
VGG 16:



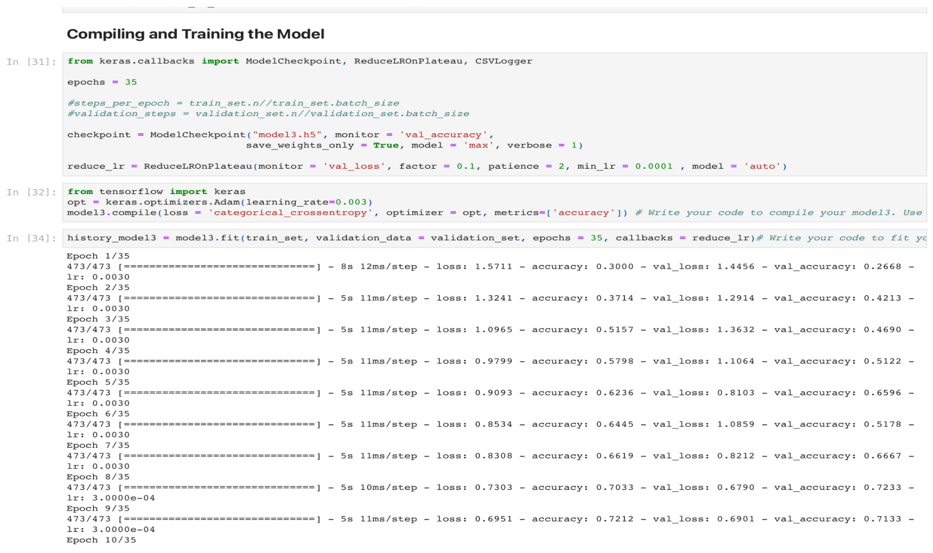
ResNet:



EfficientNet:



Code snippet:



Compiling and Training the Model

```
In [31]: from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger

         epochs = 35

         #steps_per_epoch = train_set.n//train_set.batch_size
         #validation_steps = validation_set.n//validation_set.batch_size

         checkpoint = ModelCheckpoint("model3.h5", monitor = 'val_accuracy',
                                     save_weights_only = True, model = 'max', verbose = 1)

         reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.1, patience = 2, min_lr = 0.0001 , model = 'auto')

In [32]: from tensorflow import keras
         opt = keras.optimizers.Adam(learning_rate=0.003)
         model3.compile(loss = 'categorical_crossentropy', optimizer = opt, metrics=['accuracy']) # Write your code to compile your model3. Use

In [34]: history_model3 = model3.fit(train_set, validation_data = validation_set, epochs = 35, callbacks = reduce_lr)# Write your code to fit yo
```

```
Epoch 1/35
473/473 [==============================] - 8s 12ms/step - loss: 1.5711 - accuracy: 0.3000 - val_loss: 1.4456 - val_accuracy: 0.2668 -
lr: 0.0030
Epoch 2/35
473/473 [==============================] - 5s 11ms/step - loss: 1.3241 - accuracy: 0.3714 - val_loss: 1.2914 - val_accuracy: 0.4213 -
lr: 0.0030
Epoch 3/35
473/473 [==============================] - 5s 11ms/step - loss: 1.0965 - accuracy: 0.5157 - val_loss: 1.3632 - val_accuracy: 0.4690 -
lr: 0.0030
Epoch 4/35
473/473 [==============================] - 5s 11ms/step - loss: 0.9799 - accuracy: 0.5798 - val_loss: 1.1064 - val_accuracy: 0.5122 -
lr: 0.0030
Epoch 5/35
473/473 [==============================] - 5s 11ms/step - loss: 0.9093 - accuracy: 0.6236 - val_loss: 0.8103 - val_accuracy: 0.6596 -
lr: 0.0030
Epoch 6/35
473/473 [==============================] - 5s 11ms/step - loss: 0.8534 - accuracy: 0.6445 - val_loss: 1.0859 - val_accuracy: 0.5178 -
lr: 0.0030
Epoch 7/35
473/473 [==============================] - 5s 11ms/step - loss: 0.8308 - accuracy: 0.6619 - val_loss: 0.8212 - val_accuracy: 0.6667 -
lr: 0.0030
Epoch 8/35
473/473 [==============================] - 5s 10ms/step - loss: 0.7303 - accuracy: 0.7033 - val_loss: 0.6790 - val_accuracy: 0.7233 -
lr: 3.0000e-04
Epoch 9/35
473/473 [==============================] - 5s 11ms/step - loss: 0.6951 - accuracy: 0.7212 - val_loss: 0.6901 - val_accuracy: 0.7133 -
lr: 3.0000e-04
Epoch 10/35
```