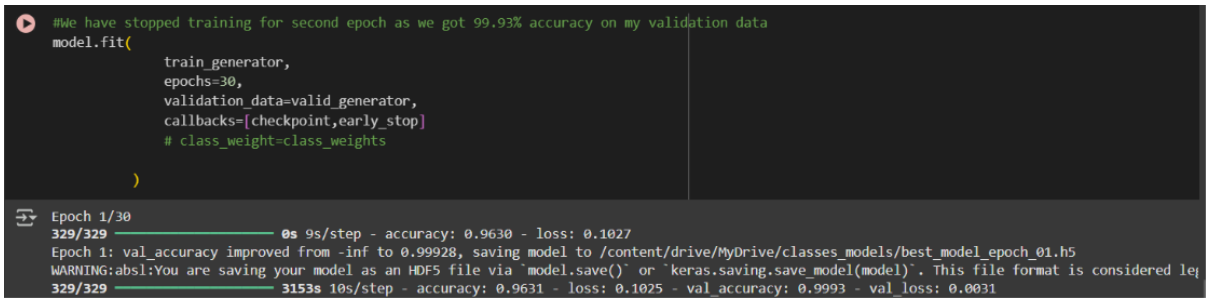
Deepfake Duel: Truth vs Trickery

=>99.93% on validation dataset in one epoch- Accuracy for labels-predicting classes: Animals, Human faces, vehicles.

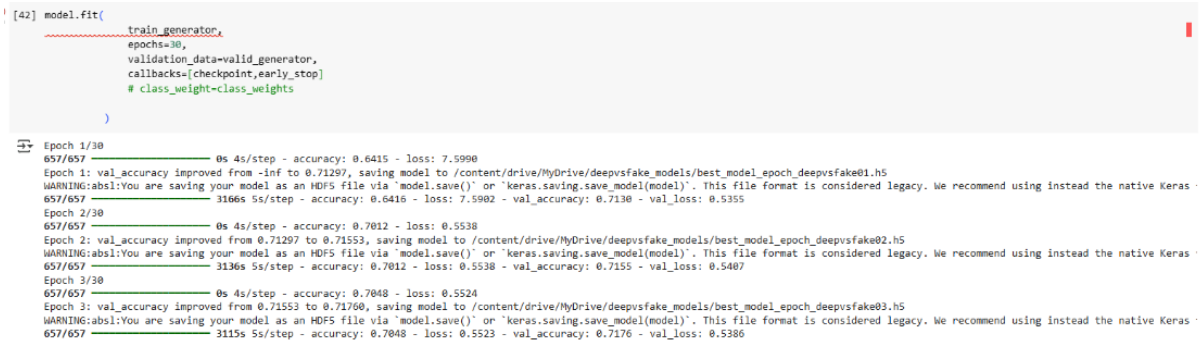


```
#We have stopped training for second epoch as we got 99.93% accuracy on my validation data
model.fit(
            train_generator,
            epochs=30,
            validation_data=valid_generator,
            callbacks=[checkpoint,early_stop]
            # class_weight=class_weights

        )
```

```
Epoch 1/30
329/329 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.9630 - loss: 0.1027
Epoch 1: val_accuracy improved from -inf to 0.99928, saving model to /content/drive/MyDrive/classes_models/best_model_epoch_01.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered leg
329/329 ━━━━━━━━━━━━━━━━━━━━ 3153s 10s/step - accuracy: 0.9631 - loss: 0.1025 - val_accuracy: 0.9993 - val_loss: 0.0031
```

=>**71.76%** on validation dataset in 3 epochs Accuracy for predicting fake vs Real



```
[42] model.fit(
            train_generator,
            epochs=30,
            validation_data=valid_generator,
            callbacks=[checkpoint,early_stop]
            # class_weight=class_weights

        )
```

```
Epoch 1/30
657/657 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.6415 - loss: 7.5990
Epoch 1: val_accuracy improved from -inf to 0.71297, saving model to /content/drive/MyDrive/deepvsfake_models/best_model_epoch_deepvsfake01.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras
657/657 ━━━━━━━━━━━━━━━━━━━━ 3166s 5s/step - accuracy: 0.6416 - loss: 7.5902 - val_accuracy: 0.7130 - val_loss: 0.5355
Epoch 2/30
657/657 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.7012 - loss: 0.5538
Epoch 2: val_accuracy improved from 0.71297 to 0.71553, saving model to /content/drive/MyDrive/deepvsfake_models/best_model_epoch_deepvsfake02.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras
657/657 ━━━━━━━━━━━━━━━━━━━━ 3136s 5s/step - accuracy: 0.7012 - loss: 0.5538 - val_accuracy: 0.7155 - val_loss: 0.5407
Epoch 3/30
657/657 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.7048 - loss: 0.5524
Epoch 3: val_accuracy improved from 0.71553 to 0.71760, saving model to /content/drive/MyDrive/deepvsfake_models/best_model_epoch_deepvsfake03.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras
657/657 ━━━━━━━━━━━━━━━━━━━━ 3115s 5s/step - accuracy: 0.7048 - loss: 0.5523 - val_accuracy: 0.7176 - val_loss: 0.5386
```

=> **Approach**

We used two separate models for training. One for training classes and another to train deep or fake.

1)We used the transfer learning approach for model training. We used the **Xception(for the classification of Animals, Human faces, vehicles)** and **VGG-16 (for classification of Deep vs Fake)** model from Keras.

2)Freeze the weights and keep model.trainable=False. We included top layers, which were the trainable layers

3)Preprocessed images. For this, we used preprocess_input from keras.Xception

4) Used **flow_from_directory** and **augmentation** for training from **keras** which creates the image on flow without requiring extra storage. This helped to generalize the model well. Because of this we get good accuracy on both train and validation.

Augmentation used

```
# Training data generator (With Augmentations). It generates data on fly
train_datagenerator = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=30,
    vertical_flip=True,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.3,
    brightness_range=[0.3, 1.7],
    **datagenerator_kwargs
)
```

5) Used callbacks to save the model after every epoch. But for the classification of labels(Animals, Human faces, vehicles) we just trained for 1 epoch with a batch size of 512.

=>Model Summary

-Total params: 21,912,107

-Trainable params: 1,050,627 (only in custom head)

-Non-trainable params: 20,861,480 (frozen Xception using learned weights from Imagenet).

=> Difficulties faced

The dataset was very large. It took hours to train the model. We also cannot keep the learning rate that much small(our learning rate was **0.001**). Even the batch size was **512**. Keeping the batch size large as 1024 was giving an error of OOM(Out of memory).

For predicting deep vs fake, it was very difficult as the images were indistinguishable even with the naked eye.

We trained on Google Colab because we were facing errors on Kaggle. Our loss function was getting to infinite during training.

On colab, the time is very limited, so we used two different accounts for two sub-tasks i.e, predicting fake vs real and predicting the three classes.

=> Learnings

We understood the working of CNN architecture and how to use the concept of transfer learning. We also learned about augmentation. This we learned during our Computer Vision class. It was challenging but we had a lot of fun doing this task.

=> **Future works**

To explore more CNN architecture for predicting deep vs fake images. Will also look into any preprocessing step helps or not.