APOD Image classification

Problem Statement Brief:

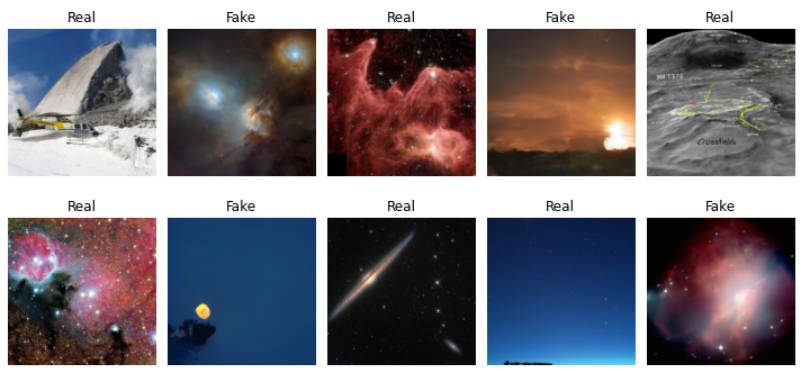
The Data science challenge deals with the astronomy picture of the day images where the task is to classify between the real and fake images using CNN.

The Data:

We are provided with a training dataset which contains images of 2 classes real (5000 images) and fake(2000 images). There is also a validation dataset which contains around ~1500 real images and ~700 fake images.

Data Loading:

The data is loaded from the directory using tf.keras generator functions, where the data is loaded in batches of specified batch\_size which in our case is 32. The image size is set at (112,112). Let us view a few training data samples, the RGB images are plotted after performing mix max scaling to bring the range to 0,1.



We can observe that the real image samples are a wide range of deep astro images both from space as well as terrestrial. The significant difference we can immediately notice wrt fake images is the vignette, contrast irregularities and improper shapes. Let us take a look at few random samples from the validation dataset



Pre processing and augmentation:

We are performing data augmentation by different methods. The images are flipped horizontally and vertically. They are also rotated by a random angle. The final image is then resized and scaled.

This processing is performed on the training dataset to increase the number of samples.

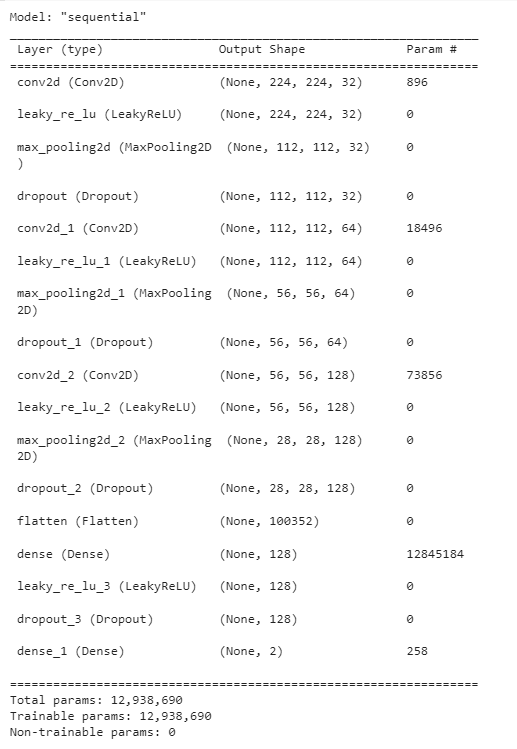
The snapshots of images post the processing are viewed below.

Data Pipelines:

The tensorflow data pipelines concept is used to run the processing function on each epoch. The pipeline contains a map function which maps the processing function on the input images, the map is done in parallel processes with the help of tf.Autotune. This helps in speeding the process. The mapped data is cached for the epoch which is stored in ram, which enables faster processing of data in next epochs. These steps are used to reduce computational bottlenecks when training data size is more or the epochs are high. Prefetch option with parallel workers is also used.

Model:

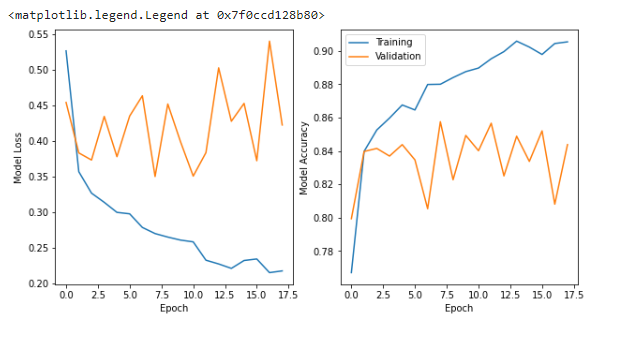
The model used here is CNN model with multiple layers as shown in the summary:

The model contains repeated series of convolution layers with ELU activation function followed by LeakyRelu, max pooling and a dropout layer. 

Upon trying multiple activation functions we observed ELU performed marginally better than the rest. The LeakyRelu function is added to accelerate convergence. The Max pooling after each conv2D layer mimics VGGnet architecture partially. Finally a dropout layer is added to avoid overfitting

The output layer contains a softmax activation which distributes probability between the classes.

Compilation and Optimisation:

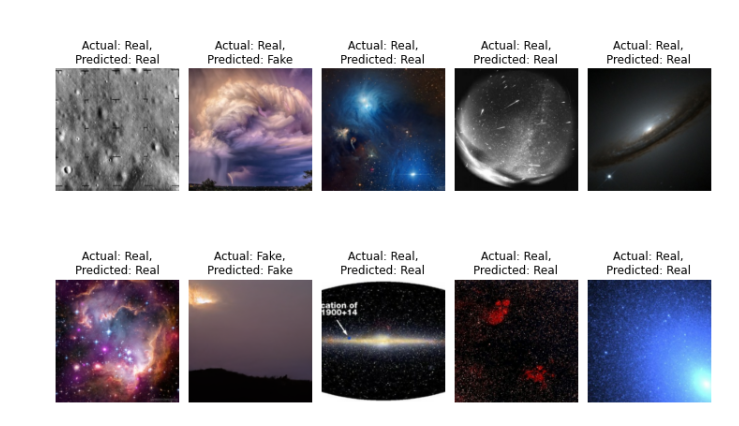
The ADAM optimiser is used over SGD here and sparse categorical cross entropy is used as the loss while compiling the model as our labels are one hot encoded i.e 0=Fake and 1=Real.

Model Training:

Trained the model for 50 epochs, but also added an early stopping callback with patience 20 to avoid overfitting of the model. The model is not running for ~15 epochs and the validation accuracy is around 84%.

Model Prediction and saving

One batch of the validation dataset is predicted on the trained model to observe patterns in the prediction capabilities of the model. The results are as follows:



The model is able to predict the samples reasonably well with a few mispredictions. The model is now saved and saved in drive as well as download as zip, which can be reloaded to predict on test data.

Brief of other options pursued which were unsuccessful:

* Built a model with similar architecture as VGGnet16, the validation and training accuracy had a huge gap.
* Tried changing the input image size starting from 32,32. Finally settled at 224,224 to where the model performed better maybe because it was able to identify patterns better. Also tried different processings
* Used pretrained VGGnet model with imagenet weights, by freezing all the layers. The performance was not good on the validation as well as training set.
* Tried augmenting the data along with the transfer learning concept, results were not good.
* Tried building a model from scratch, with conv2D layers and max pooling layers. After changing the optimiser, activation functions and other parameters. The final model (discussed in the above section) provided better results.
* Used caching concept, prefetch, rebatching and shuffling as well to vary the performance and improve the model training time.

Summary:

The final classification model is built on CNN with ELU activation, which is trained on images which are processed by rotating and flipping. The training lasted for ~17 epochs which resulted in an accuracy of 84% on the validation dataset.