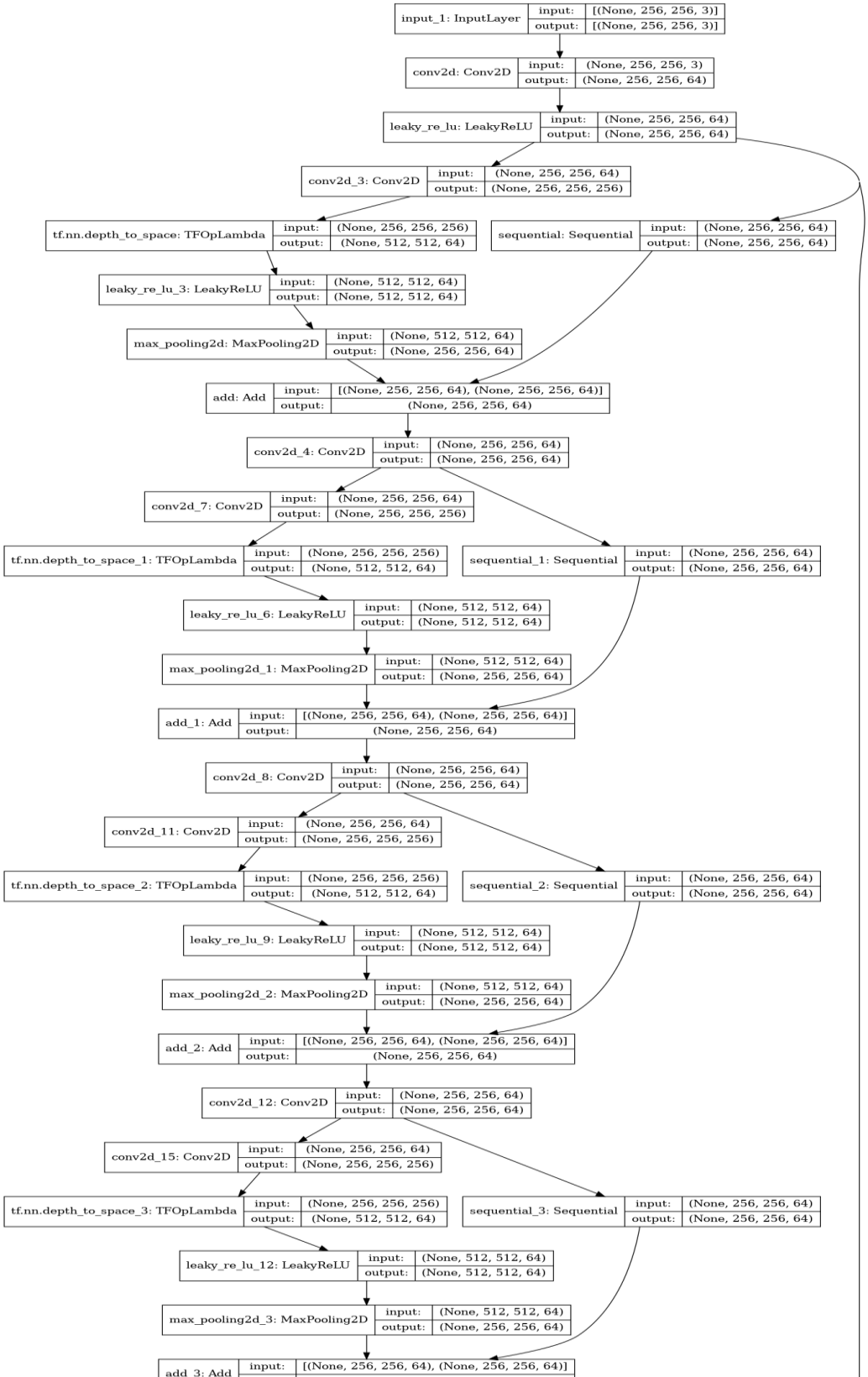
Model Implemented:

Enhanced Deep Super Resolution Network: A deep CNN which uses Residual Blocks , i.e ,skip connections, to preserve information from the initial input image.

Architecture:



HYPERPARAMETERS AND WHY THEY WERE CHOSEN:

ACTIVATION FUNCTION: Leaky ReLU – It works much better as an activation for prediction problems than counterparts like sigmoid and tanh, and also accounts for exploding gradient problem.

LAYERS:

Convolutional 2D – Reduces the no. of features than a classic fully connected NN, while preserving the important information about the image.

**CNNs work very well for image generation even though it is a lightweight model compared to more complex models like GANs

Max Pooling – Used to further reduce the no. Of features and from the feature map obtained from the CNN.

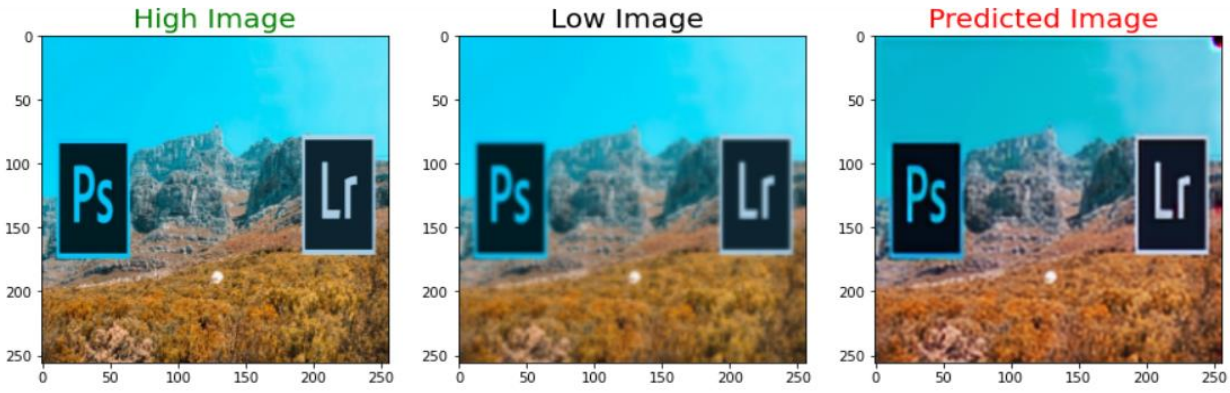
TfOpLambda layer - Resizes the image to a larger pixel dimension to get more features to train so that higher pixel quality can be obtained for the final output.

OPTIMIZERS:

Adam – It converges to the minima very quickly, as it has a momentum parameter(default 0.9), and also slows down learning rate when close to minima.

No. Of epochs = 10 - Model drastically slows learning beyond 10 epochs, and occasionally reduces accuracy and increases loss.

Results:

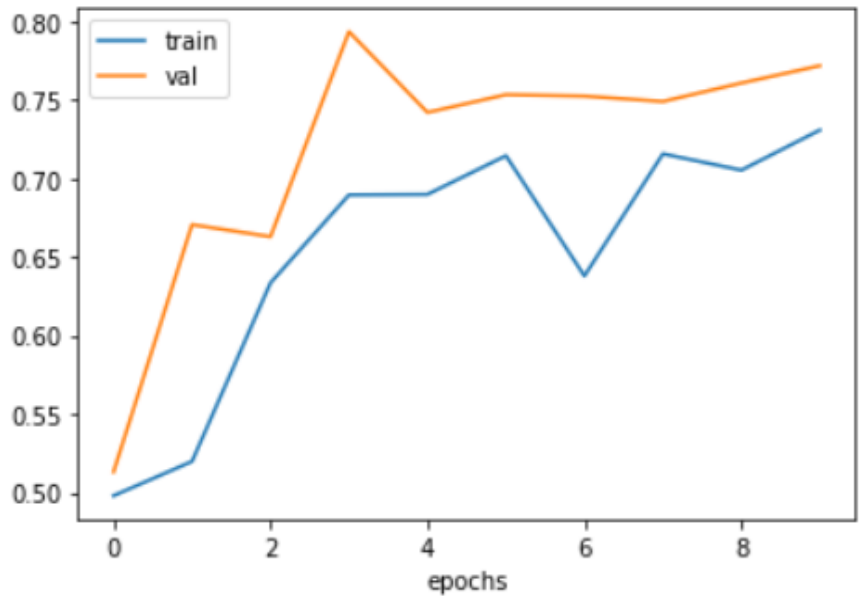


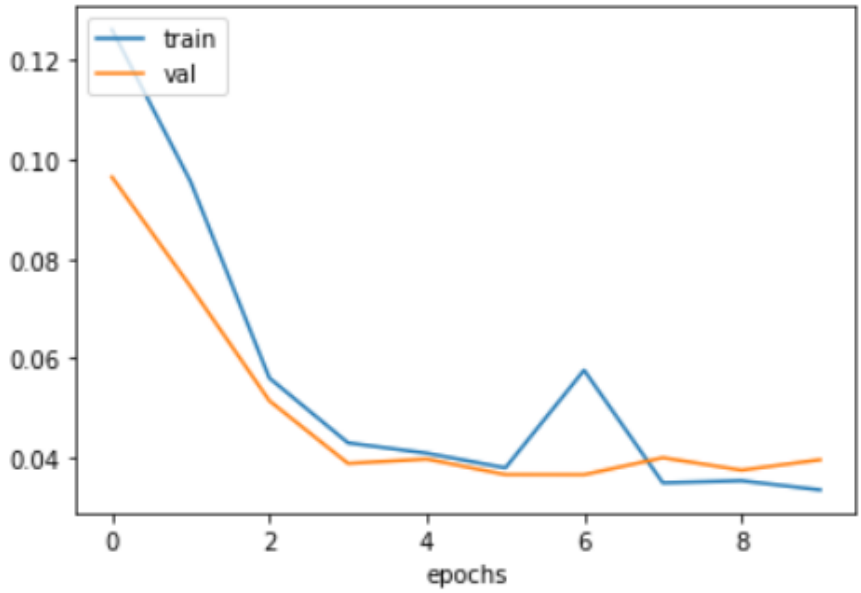
PSNR tf.Tensor(21.3858, shape=(), dtype=float32)

Metrics:

Loss: Mean Absolute Error

Metric: Accuracy (Mean squared error is a better metric, but due to a technical error could not implement it. However, accuracy tells us the no. Of pixel values in an image the model got right so it is useful.)





```
700/700 [==============================] - 167s 227ms/step - loss: 0.1262 - acc: 0.4981 - val_loss: 0.0965 - val_acc: 0.5132
Epoch 2/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0955 - acc: 0.5199 - val_loss: 0.0744 - val_acc: 0.6705
Epoch 3/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0561 - acc: 0.6335 - val_loss: 0.0515 - val_acc: 0.6628
Epoch 4/10
700/700 [==============================] - 158s 225ms/step - loss: 0.0430 - acc: 0.6894 - val_loss: 0.0389 - val_acc: 0.7932
Epoch 5/10
700/700 [==============================] - 158s 225ms/step - loss: 0.0409 - acc: 0.6897 - val_loss: 0.0398 - val_acc: 0.7419
Epoch 6/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0380 - acc: 0.7143 - val_loss: 0.0366 - val_acc: 0.7532
Epoch 7/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0576 - acc: 0.6379 - val_loss: 0.0366 - val_acc: 0.7523
Epoch 8/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0350 - acc: 0.7155 - val_loss: 0.0400 - val_acc: 0.7488
Epoch 9/10
700/700 [==============================] - 158s 225ms/step - loss: 0.0354 - acc: 0.7052 - val_loss: 0.0375 - val_acc: 0.7606
Epoch 10/10
700/700 [==============================] - 155s 222ms/step - loss: 0.0336 - acc: 0.7307 - val_loss: 0.0396 - val_acc: 0.7716
```

CONCLUSION:

Image Generation is an extremely computationally heavy task, due to which multiple models, datasets and hyperparameters had to be changed to produce results.

The current model shows that a relatively lightweight model can produce decent results, (validation accuracy is 77.16% on the last epoch). There is a upward trend in the accuracy, albeit a bit shaky(due to selecting accuracy as a metric), for both train and validation splits.

Similarly, there is a downward trend in the loss as well. On a model with the same architecture but double the number of epochs, the loss converges around 0.0325.

The dataset used to implement this models consists of 855 low resolution and high resolution images each, with train data having 700 images, validation having 130, and test having the remaining 25.

**Initially, the models used were very complex and deep. GANs produce better results but computationally are far too heavy to run on Google Colab, Kaggle or on the local GPU used for this model (NVIDIA GeForce 1650ti 4GB VRAM), due to which not only the model, but the datasets as well had to be changed.

However, according to the research papers in my previous report as well as the results from this model itself we can see that CNNs are far more lightweight, yet produce good results, even after hardware constraints.

Research paper referred for using CNN - https://www.researchgate.net/publication/270454670_Image_SuperResolution_Using_Deep_Convolutional_Networks