DL Assignment-2

Group Number: S21DL20
Korupoulu Saideepthi S20180010087
Varakala Sowmya S20180010187
Gendeti Manjju Shree Devy S20180010055
Swathi Kedarasetty S20180010172

Part-1 Object Detection

YOLO

You only look once It's an object detector that uses features learned by a deep convolutional neural network to detect an object , over the time many versions of yolo has been improved yolo v1,...v5 (yolo v1 is implemented for the assignment)

Code

- Data
  - PascalVoc Dataset is used
  - Download from link

- Configuration (config.py)
  - Configuration for yolo on pascal_voc datasets is done in this file

- Data Annotation (voc_labe.py)
  - All the images files need to be annotated in this format
    {object-class x y width height} which is done in this file

- Data Augmentation (datset.py)
  - Reads Images from the path
  - Gets the bounding boxes , class predictions
  - Random shift
  - Random scale
  - Random crop
  - Random flip
  - Subtract mean
  - Encoder
    - function takes as input bounding boxes and corresponding labels for a particular image
      sample and outputs a target tensor of size SxSx(5xB+C)
      where S,S are the spatial dimensions, B is the number of bounding box predictions, C is the number of classes in the dataset and 5 represents the bounding box coordinates
  - getitem
    - All the above functions are called in this function to get the final augmented images

- Model (restnet_yolo.py)
    - Used resnet model
- Loss functions (yolo_loss.py)
    - Compute_iou (Intersection over union)
      Compute the intersection over union of two set of boxes, each box is [x1,y1,x2,y2].
      Args:
    - box1: (tensor) bounding boxes, sized [N,4].
    - box2: (tensor) bounding boxes, sized [M,4].
    - Return:
    - (tensor) iou, sized [N,M]
    - forward
      Args:
      pred_tensor: (tensor) size(batchsize,S,S,Bx5+20=30)
    - where B - number of bounding boxes this grid cell is a part of = 2
    - 5 - number of bounding box values corresponding to [x, y, w, h, c]
    - where x - x_coord, y - y_coord, w - width, h - height, c - confidence of having an object
    - 20 - number of classes
    - target_tensor: (tensor) size(batchsize,S,S,30)
        - Compute loss for the cells with no object bbox
        - Compute loss for the cells with objects
        - Choose the predicted bbox having the highest IoU for each target bbox
        - BBox location/size and objectness loss for the response boxes
        - Class probability loss for the cells which contain objects
        - Total loss
          Returns : Total Loss
- Hyperparameters
    - learning_rate = 0.001
    - num_epochs = 50
    - batch_size = 24

      # Yolo loss component coefficients (as given in Yolo v1 paper)

    - lambda_coord = 5
    - lambda_noobj = 0.5

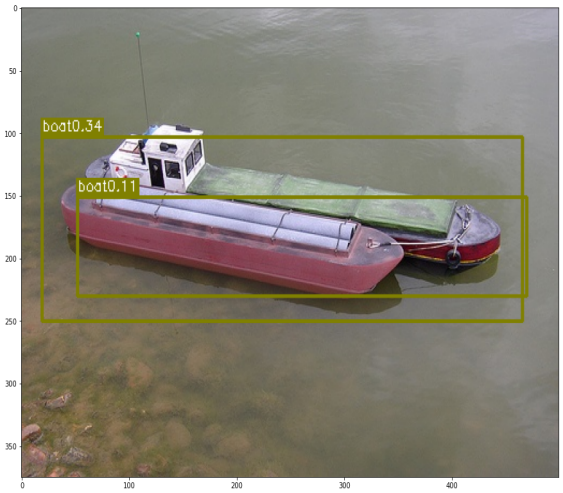
- Training (yolo.ipynb)
    - Model is trained with resnet model on the data augmented images with the above hyperparameters and the best detector is saved in **best_detector.pth**
    - **best_detector.pth** [link]

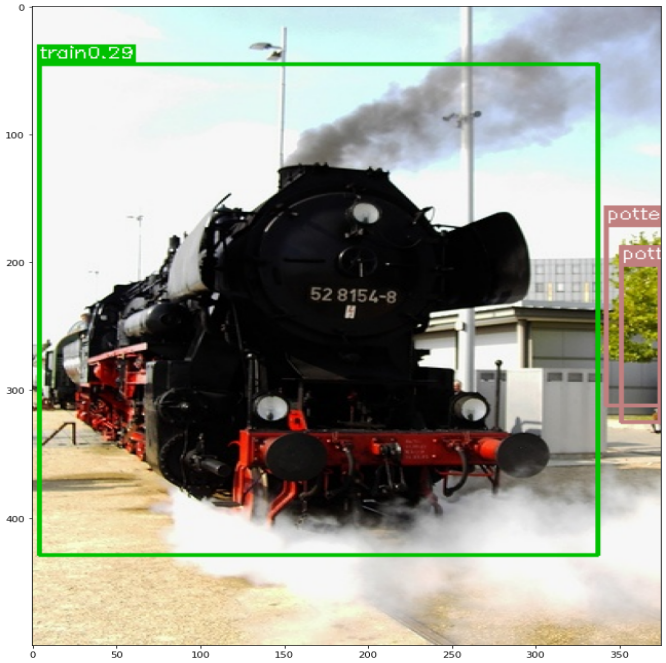
- Evaluating(eval_voc.py)
    - Using the best_detector.pth the test images are tested

- Results
    - Mean average precision over the test dataset : 0.49

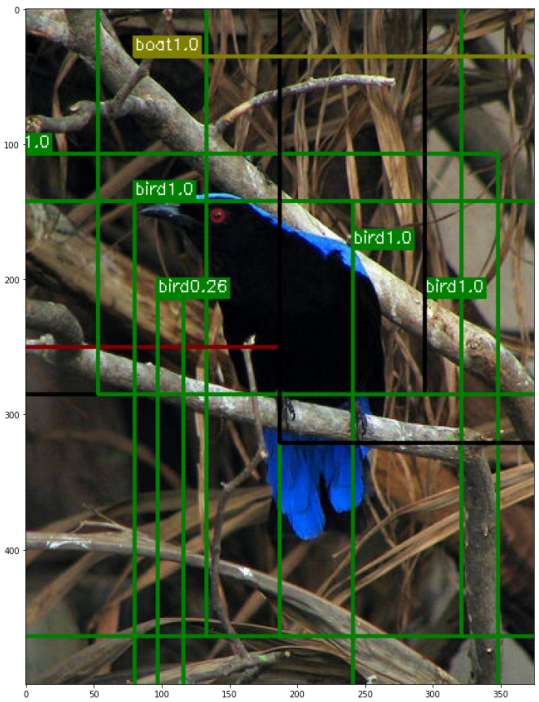
```
---Evaluate model on test samples---
100%|██████████| 4950/4950 [02:23<00:00, 34.43it/s
---class aeroplane ap 0.461997978943023---
---class bicycle ap 0.6359974440609961---
---class bird ap 0.47355089012157403---
---class boat ap 0.2862735768253211---
---class bottle ap 0.2091828800787261---
---class bus ap 0.6336731764541319---
---class car ap 0.6458709332098127---
---class cat ap 0.6858524443643779---
---class chair ap 0.315842207049645---
---class cow ap 0.5132370276353126---
---class diningtable ap 0.3388517367570423---
---class dog ap 0.6307587953970472---
---class horse ap 0.6760313169372795---
---class motorbike ap 0.5649322219448399---
---class person ap 0.5295174355939133---
---class pottedplant ap 0.1868921931047106---
---class sheep ap 0.4647626789384095---
---class sofa ap 0.4620061549727634---
---class train ap 0.6708275376684594---
---class tvmonitor ap 0.5022007930301944---
---map 0.4944129711543791---
```

- Testing on random images(predict.py)





Faster R-CNN



Faster r-cnn is one more approach for object detection

Code

- Data
  - PascalVoc Dataset is used
  - Download from link

- Data augmentation (dataset/base.py)
  - In this file the data is augmented
    - Transforming all the images
    - Padding
    - Scaling ...etc

- Data Annotations (dataset/voc_2007.py)
  - In this file dataset is annotated
  - And also all the classes are labeled

- Configuration(config)
  - In this folder there will three .py files train_config.py, eval_config.py, config.py
  - In config.py basic configurations are given
    - Image sizes
    - Anchor sizes and ratios
  - In train_config.py all the necessary parameters needed for training are given
    - Batch_size
    - Learning_rate
    - Weight_decay
    - Steps(iterations)
    - RPN and NMS parameters
  - In eval_config.py all the necessary parameters needed for testing are given
    - RPN and NMS parameters

- Region Proposal Network (rpn/region_proposal_network.py)
  - Region proposal network (**RPN**) is used for generating region proposals and a network using these proposals to detect objects.

- Region of Interest (roi/pooler.py)
  - Region of Interest (**ROI**) pooling is used for utilising single feature map for all the proposals generated by RPN in a single pass

- bbox.py
  - Anchor boxes are defined in this file

- Non Maximum Suppression (support/layer/nms.py)
  - **Non-maximum suppression** (NMS) is used to get rid of overlapping boxes,

- Backbone (backbone)
    - resnet cnn is written in resnet101.py

- Model (model.py)
    - This file contains code for Resnet model , loss functions and for detections

- Training (train.py)
    - Dataset is loaded
    - All the above files mode,rpn,roi,hyperparameters are imported and used for training

- Evaluation (eval.py)
    - This file is used for evaluating

- Faster-rcnn.ipynb
    - Train data is trained and model is saved link
    - Using the model the test data is evaluated

- Results
    - Mean Average Precision : 0.43

```
2021-04-14 08:48:36 INFO     Start evaluating with 1 GPU (1 batch per GPU)
100% 2510/2510 [05:15<00:00,  7.97it/s]
2021-04-14 08:53:59 INFO     Done
2021-04-14 08:53:59 INFO     mean AP = :0.4316
1: aeroplane AP = 0.5056
2: bicycle AP = 0.0462
3: bird AP = 0.6708
4: boat AP = 0.4821
5: bottle AP = 0.2817
6: bus AP = 0.5478
7: car AP = 0.1867
8: cat AP = 0.4620
9: chair AP = 0.5142
10: cow AP = 0.6853
11: diningtable AP = 0.3388
12: dog AP = 0.5429
13: horse AP = 0.6760
14: motorbike AP = 0.3278
15: person AP = 0.5022
16: pottedplant AP = 0.6490
17: sheep AP = 0.4285
18: sofa AP = 0.5343
19: train AP = 0.3158
20: tvmonitor AP = 0.4523
```

Part 2 - Self Supervised learning

Dataset

CIFAR100 dataset

Model

CIFAR100Resnet → Built by using resnet34 as the base model

```
class CIFAR100Resnet(ImageClassificationBase):
        def __init__(self):
                super().__init__()
                self.network = torchvision.models.resnet34()
                num_ftrs = self.network.fc.in_features
                self.network.fc = nn.Linear(num_ftrs, 100)

        def forward(self, xb):
                return self.network(xb)

model = CIFAR100Resnet()
```

Training

For each epoch,

- Calculating the loss of the model (Note that, we used regularization also to prevent the weights from becoming too large) with training data
- Clipping the gradients (to limit the gradients to a small range, preventing undesirable changes in the parameters).
- Optimizing the model
- Repeating the above steps

Optimizer function chosen : Adam

Weight initialization :

- He's initialization is used
- The weights are initialized keeping in mind the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently.
- The weights are still random but differ in range depending on the size of the previous layer of neurons.

Hypterparameters:

- For weight decay : 5e-4
- For gradient clipping : 0.1

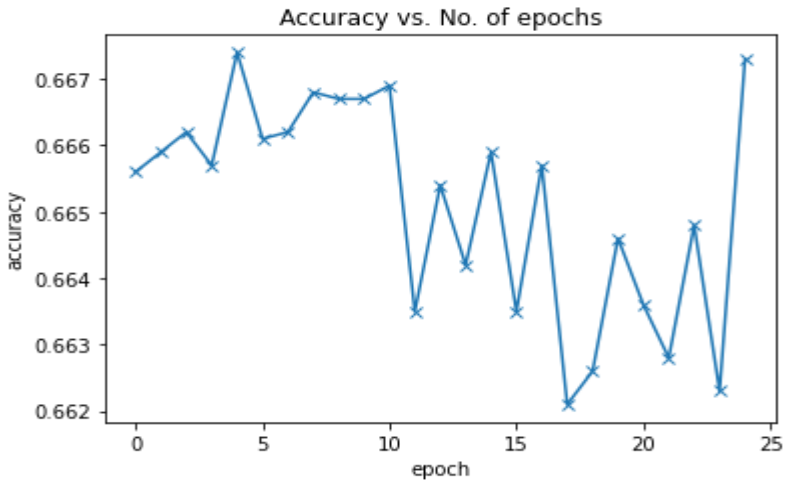
Number of epochs	Learning rate used

8	1e-3
10	1e-5
5	1e-4
15	1e-4
25	1e-5

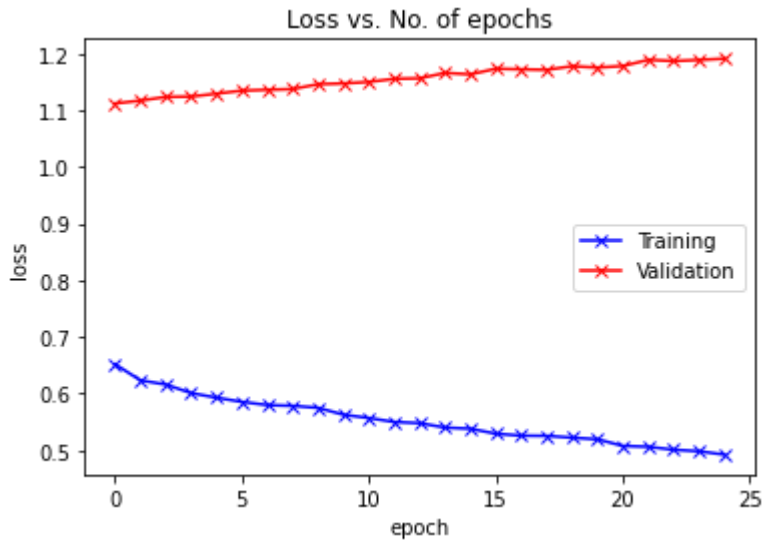
Results obtained

Accuracy : 66%
Training loss : 0.4921
Validation loss : 1.1913

Plot between Accuracy vs number of epochs



Plot between Loss vs number of epochs



Self supervised Learning

- Creating an augmented version of the cifar 100 dataset by rotating all the images by 0,90,180 or 270 degrees.
- Training the Resnet based neural network with this rotated dataset to classify images into their appropriate levels of rotation.
- Then, training the ResNet model with and without the self-supervised feature extractor on different portions of the data, to see how potential performance improvements depend on the amount of labeled data available.

Conclusion

Self supervised learning learns general features from a large distribution of data that cannot be obtained from a small labeled but non-representative subset of the data.

As that subset gets larger and larger, it becomes more and more representative and thus more conducive to learn general features that boost the performance on unseen data.

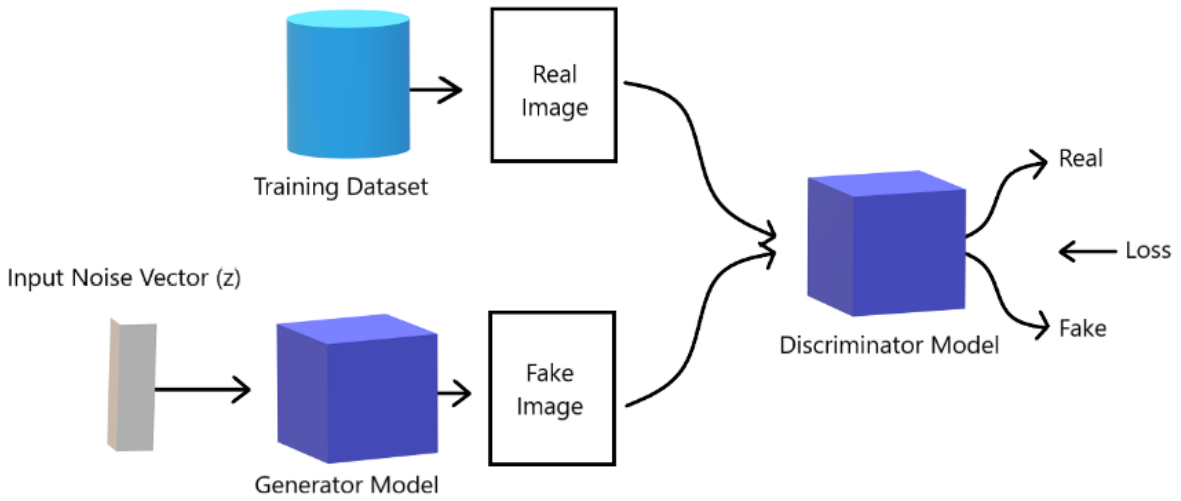
Part - 3 Image-to-Image Translation using Conditional GAN

Conditional GAN :

Conditional generative adversarial network is a type of GAN that involves the conditional generation of images by a generator model. GANs rely on a generator that learns to generate new images, and a discriminator that learns to distinguish synthetic images from real images.

Gans consists of two models, namely:

- Generator : It is a function that takes an input noise vector(z) and maps to an image that hopefully resembles the images in the training dataset.
- Discriminator : The primary purpose of the discriminator model is to find out which image is from the actual training dataset and which is an output from the generator model.

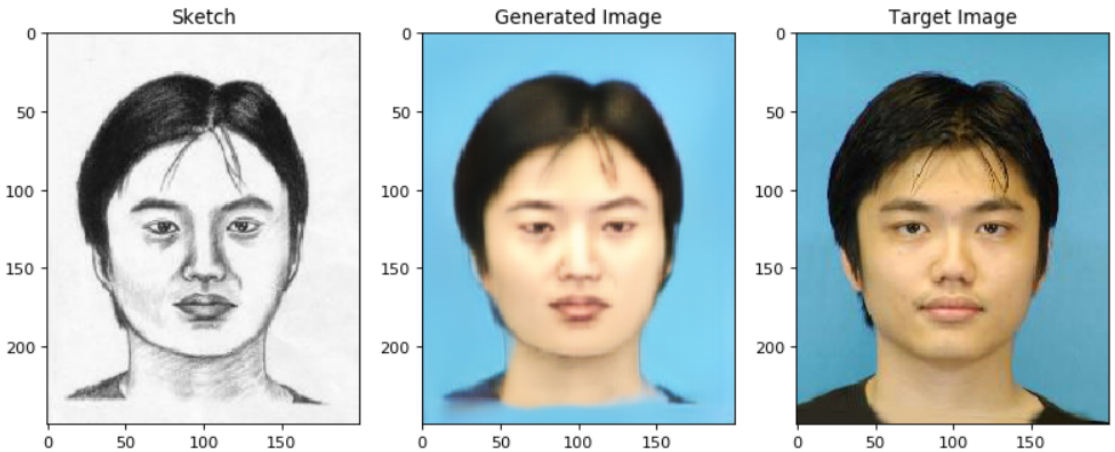


Source: https://towardsdatascience.com/generative-adversarial-networks-gans-8fc303ad5fa1

Code

- Dataset :
  - CUHK student dataset is used.
  - 88 training faces.
  - 100 testing faces.
- Data augmentation(dataPreprocessing.ipynb)
  - Reads Images from the path
  - All the training images and sketches are rotated into different angles to generate different parameters for transformation.
  - Random shift
  - Random scale

    - Random crop
    - Random flip.
- Conditional Gan (model) notebook : ConditionalGAN.ipynb
    - Need to install keras as pip install git+https://www.github.com/keras-team/keras-contrib.git
    - Functions ‘load_filename’ and ‘load_images’ reads the input images.
    - Function ‘generate_real_samples’ selects a batch of random samples, returns real images and target.
    - Function ‘generate_fake_samples’ generates a batch of images, returns images and targets.
    - In the function generator we have 2 functions ‘conv2d’ and ‘deconv2d’.
    - Function conv2d is used to downsample the image.
    - Function deconv2d is used to upsample the image.
    - Function ‘discriminator’ used to differentiate target and predicted images.
    - Function ‘train’ is used train the model
    - Function ‘summarize_performance’ is used to save the models at target location.
    - Both models are too large in size so the models are uploaded in google drive.
    - Link for d_model - link
    - Link for g_model - link
- Testing and predicting(testingImage.ipynb)
    - Loads the generator model from the Models folder.
    - Load the sketch and target images from the testing folder.
    - The model ‘g_model’ generates the photo.
    - Results :



    - Functions ‘load_filename’ and ‘load_images’ reads the input images.
    - Function ‘pred_images’ is used to generate the images.
    - The generated images are saved in GeneratedImages/Generated_Pixel[02]_Context[08] folder.

PART - 4 - Sentiment analysis on IMDB reviews

LSTM(Long Short-term Memory)

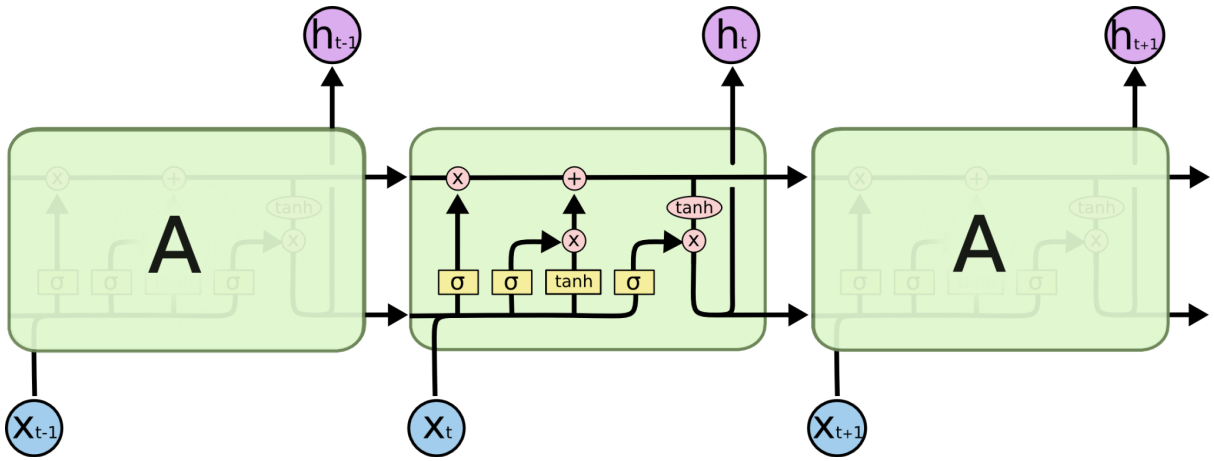
Dataset - IMDB Dataset of 50K Movie Reviews
https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

About Dataset

- Samples - 50K rows
- 25K positive and 25K negative examples

LSTM

- Long short-term memory is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

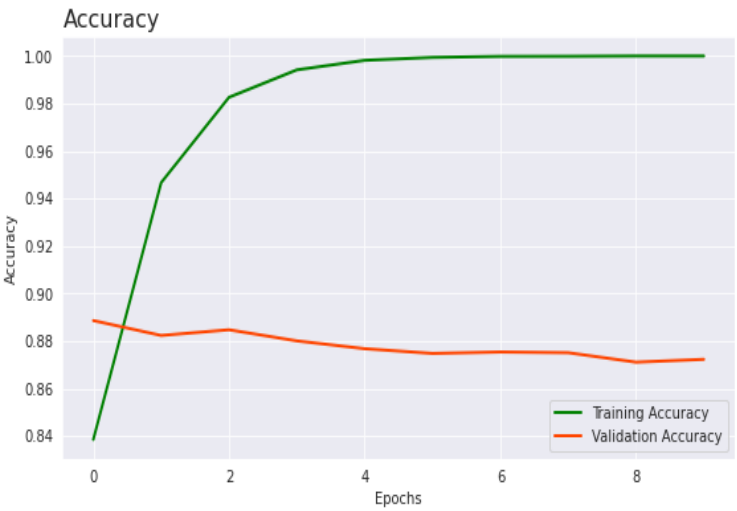


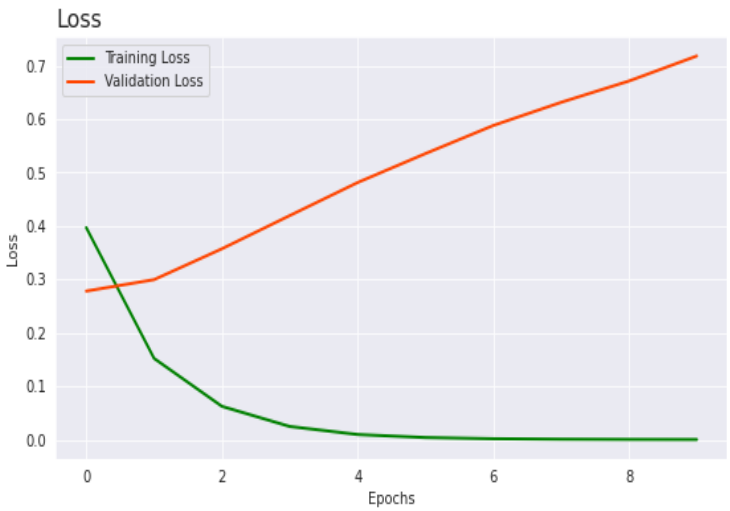
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Code

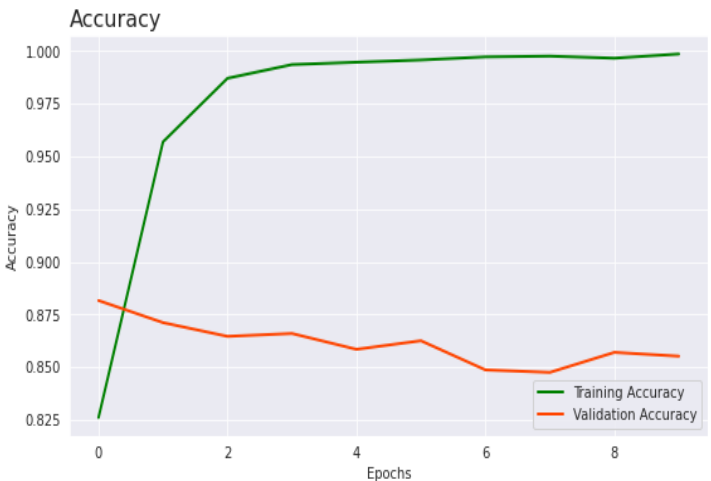
- Preprocessing
  - Removed stop words
  - Removed non alphanumeric characters
  - Removed URLs
  - Removed Html tags
  - Using Labelencoder encoded target labels with values between 0 and 1.
  - Splitted the data into train and test where 33500 training set and 16500 test set.

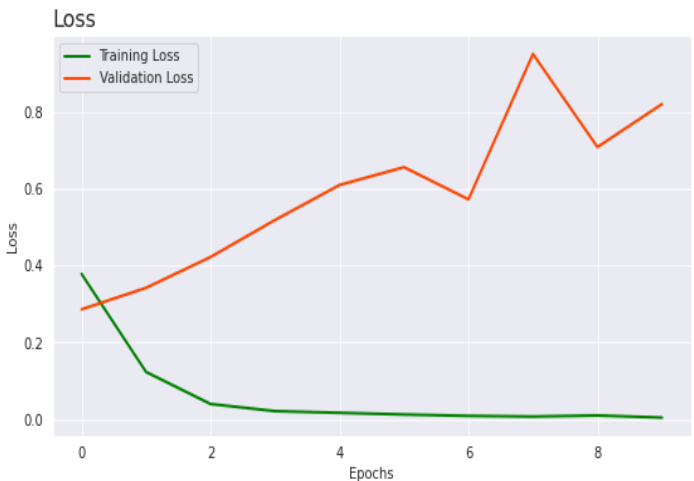
- Word Embedding
    - Embedding Layer
    - Pooling Layer - used GlobalAveragePooling1D
    - Dense Layer - with activation function as relu
    - Dense Layer - with activation function as sigmoid
    - Used binary cross entropy loss function and adam optimizer.
    - Used 10 epochs.
- Results
    - Accuracy - train 100 and test 87.23
- Plots



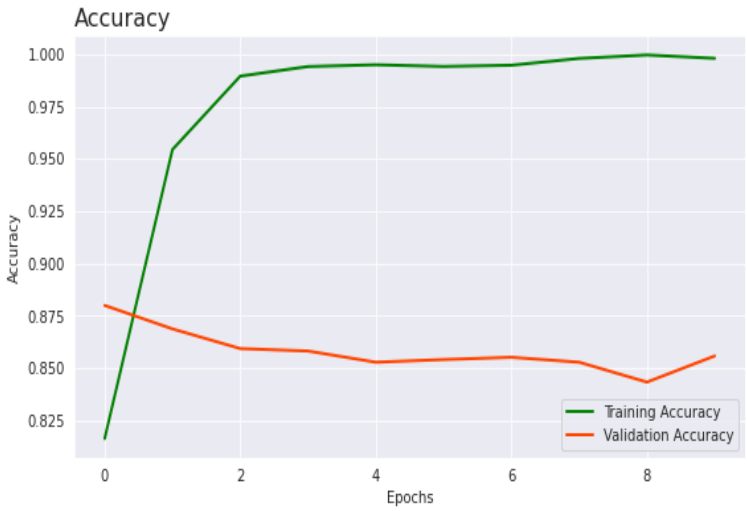


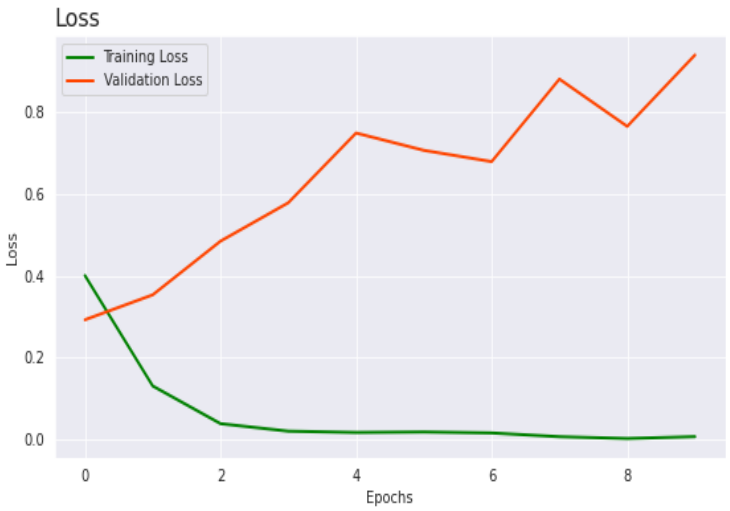
- LSTM
    - Embedding Layer
    - Bidirectional lstm - used lstm(64)
    - Dense Layer - with activation function as relu
    - Dense Layer - with activation function as sigmoid
    - Used binary cross entropy loss function and adam optimizer.
    - Used 10 epochs.
- Results
    - accuracy - train 99.8 and test 85.53
- Plots



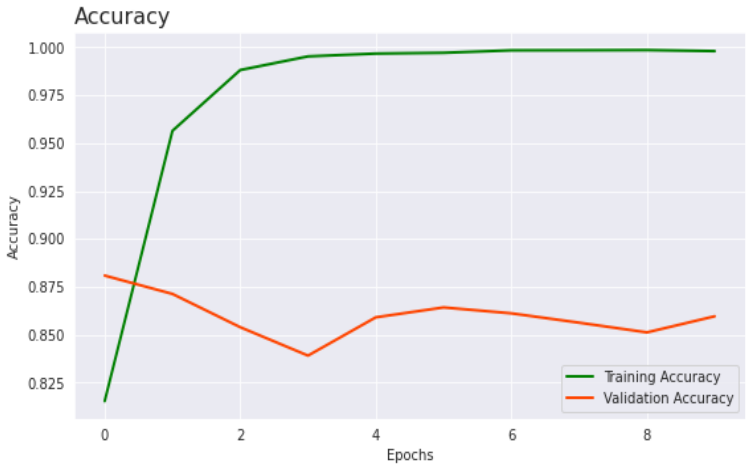


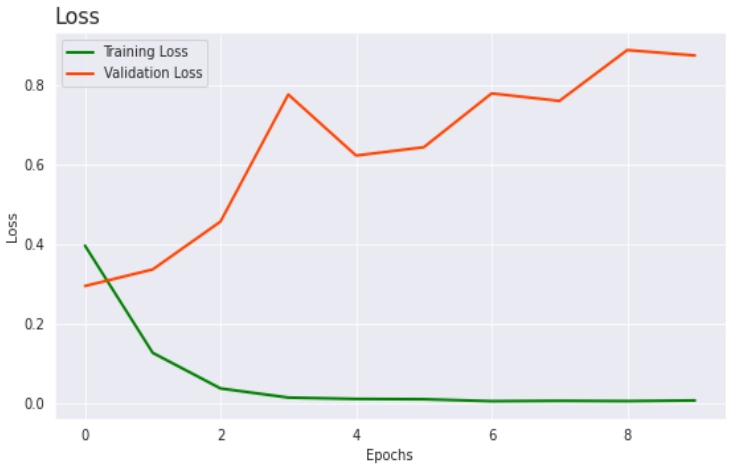
- LSTM 2 layer
    - Embedding Layer
    - Bidirectional lstm - used LSTM(64)
    - Bidirectional lstm - used LSTM(32)
    - Dense Layer - with activation function as relu
    - Dense Layer - with activation function as sigmoid
    - Used binary cross entropy loss function and adam optimizer.
    - Used 10 epochs.
- Results
    - accuracy - train 99.8 and test 85.58
- Plots



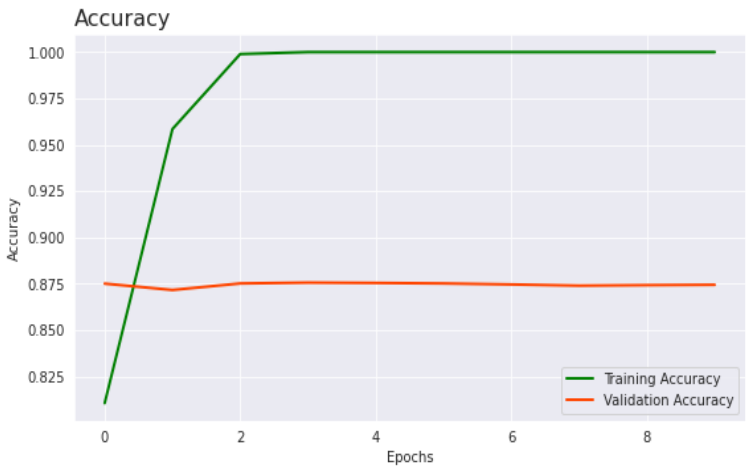


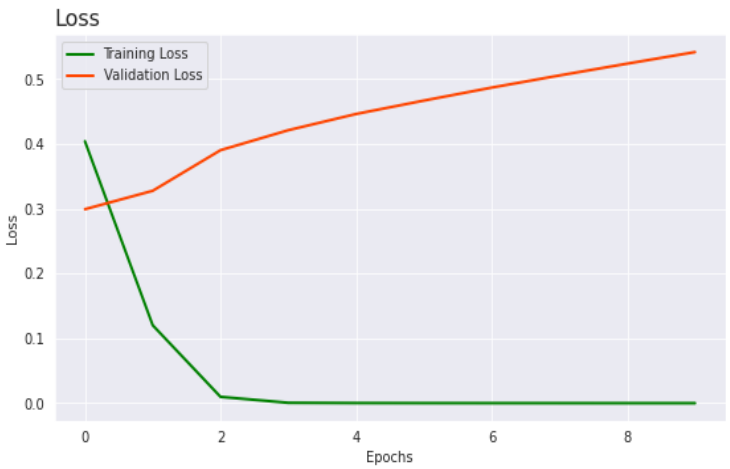
- GRU
    - Embedding Layer
    - Bidirectional GRU - used GRU(64)
    - Dense Layer - with activation function as relu
    - Dense Layer - with activation function as sigmoid
    - Used binary cross entropy loss function and adam optimizer.
    - Used 10 epochs.
- Results
    - accuracy - train 99.8 and test 85.96
- Plots





- Convolution
    - Embedding Layer
    - Conv Layer - Conv1D, activation is relu
    - Pooling Layer - GlobalMaxPooling1D
    - Dense Layer - with activation function as relu
    - Dense Layer - with activation function as sigmoid
    - Used binary cross entropy loss function and adam optimizer.
    - Used 10 epochs.
- Results
    - accuracy - train 100 and test 87.45
- Plots





Contributions

- Korupoulu Saideepthi S20180010087
    - Part-1 Object Detection Yolo ,Faster R-CNN

- Swathi Kedarasetty S20180010172
    - Part-2 Self- Supervised Learning

- Varakala Sowmya S20180010187
    - Part-3 Image-to-Image Translation using Conditional GAN

- Gendeti Manjju Shree Devy S20180010055
    - Part-4 Sentiment analysis on IMDB reviews

Note :

After combining all the files of 4 parts of the assignment the size of the submission folder exceed the limit and we are not able to mail the folder hence we are not submitting the models.pth files of part-1,3 instead we attached the models.pth of our google drive links (which are accessible) in the report

----------CODE----------

Step 1:

```
Step 1:
Step 2:
Step 3:
Step 4:
Step 5:
Step 6:
Step 7:
Step 8:
Step 9:
Step 10:
Step 11:
Step 12:
Step 13:
Step 14:
Step 15:
Step 16:
Step 17:
Step 18:
Step 19:
Step 20:
Step
```

Part 1

- Mounting to drive
- Installing necessary libraries
- Training
- Inference
- Fine-tuning Llama3-instruct
- Fine-tuning Llama3-base
- Fine-tuning Llama3-base-instruct
- Fine-tuning Llama3-base-instruct-chat

Mounting to drive

```
!pip install -qU gdown
!mkdir -p /content/drive/MyDrive
```

```
!wc -l /content/drive/MyDrive/*.txt
```

```
!head -n 1 /content/drive/MyDrive/*.txt
```

```
!gdown --id 1-ZeZ7ERfQ8JCCEbFqqqqqqqqqqqqqqqq
```

```
!ls -lh
```

Installing necessary libraries

```
!pip install -qU trl
!pip install -qU peft
!pip install -qU bitsandbytes
!pip install -qU transformers
```

```
[K     [91m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [0/0] [ultralytics]
```

Part-2

- Supervised learning
- Image classification
- Tools: tensorflow
- Dataset: Kaggle Dog breed identification
- Model: Tensorflow Hub mobilenet_v2
- Model: Tensorflow Hub EfficientNet
- Google Colab: Tensorflow 2.0 GPU runtime

GETTING THE NECESSARY MODULES

```
!pip install -U -q
!apt install tree
import os, datetime
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.image as mpimg
%matplotlib inline
%load_ext tensorboard
import sklearn, sklearn.metrics, sklearn.model_selection, sklearn.utils
import tensorflow as tf, tensorflow_hub as hub
print("TensorFlow version: {}".format(tf.__version__))
print("Hub version: {}".format(hub.__version__))
print("GPU {} available".format(tf.test.gpu_device_name() and "is" or "is not"))
```

EXPLORING THE DATA

```
!ln -s /gdrive/My\ Drive/kaggle/dog-breed-identification/{*,train,test} .
!ln -s /gdrive/My\ Drive/kaggle/dog-breed-identification/models models
!ls -lhtrd *
```

The dataset is available to the directory data/breed-identification. It contains 2 folders (train and test), containing files having id=UNIQUE_HASH.jpg and labels.csv (10350 images) and test (10357 images) respectively. Each of them contains 10 folders, one for each class of breed. We are using only a subset of the data.

```
!tree -d -L 1 -lF --dirsfirst --du --noreport data/breed-identification
```

```
!find train | wc -l
!find test | wc -l
!wc -l data/breed-identification/labels.csv
!head -5 data/breed-identification/labels.csv
!tail -5 data/breed-identification/labels.csv
```

```
!df -Hx squashfs -x iso9660 -x tmpfs -x devtmpfs /content/.
```

CHECKING AN IMAGE OF THE DATASET

```
def show_random_image(target, classes):
  random_index = np.random.choice(os.listdir(target))
  img = mpimg.imread(target + '/' + random_index)
  print("Shape of image: ", img.shape)
  plt.imshow(img)
  plt.title("Class: " + classes[classes['id'] == random_index[:-4]]['breed'].values[0])
  plt.axis("off")
  return random_index, img.shape
```

```
show_random_image('train', pd.read_csv('data/breed-identification/labels.csv'))
```

Training and Validation Datasets

Could contribute a lot using the tensorflow/keras/fastai/pytorch.

Dumb-ass(ly simple) chosen train/validation split: taking images from the training dataset.

Don't use the validation set from the dog breed challenge, it's not labelled and we don't have access to evaluate against the leaderboard.

```
# data = pd.read_csv('data/breed-identification/labels.csv')
# data['breed'].value_counts().plot.bar();
# data_train = data[(data['id'] != '000bec180eb18129602dd2a383899bee')]
# data_val = data[(data['id'] == '000bec180eb18129602dd2a383899bee')]
# print("Training: ", data_train.shape[0])
# print("Validation: ", data_val.shape[0])
# !rm -rf data/breed-identification/{train,test}
# !ln -s /dev/shm data/breed-identification/{train,test}
# !cp -r data/breed-identification/{train,test} /dev/shm/.
```

Let's create class labels for converting images to tensors as using the entire dataset ran into difficulties in terms of memory and ended up with shape batches of data from the resulting dataset and sorts of mess.

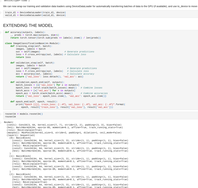
```
labels_csv = '/dev/shm/labels.csv'
!head -5 $labels_csv
!tail -5 $labels_csv
print("Number of lines: ", len(open(labels_csv).readlines()), "in $labels_csv")
unique_breeds = np.unique(pd.read_csv(labels_csv)['breed'])
print("Unique breeds: ", len(unique_breeds))
```

Let's take a look at some sample images from the training dataset.

Batch size: 32 is a good default, but we can experiment with larger batch sizes. Since the data consists of 10,222 of a color images with 3 channels (RGB), each image stores has the shape [1, 224, 224, 3].

```
TRAIN_DIRECTORY = '/dev/shm/train/'
BATCH_SIZE = 32
IMG_SIZE = 224
IMG_EXTENSION = '.jpg'
print("Training images: ", len(os.listdir(TRAIN_DIRECTORY)), " in $TRAIN_DIRECTORY")
print("Image size: ", IMG_SIZE, " Batch size: ", BATCH_SIZE)
```

```
!ls -lhtrd /dev/shm/train/*{0,1,2,3,4,5,6,7,8,9}*jpg | wc -l
!ls -lhtrd /dev/shm/train/*{0,1,2,3,4,5,6,7,8,9}*jpg | head -5
!ls -lhtrd /dev/shm/train/*{0,1,2,3,4,5,6,7,8,9}*jpg | tail -5
```



lets view our batch of images

```
def show_25_images(images, labels):
  plt.figure(figure=(10,10))
  for i in range(25):
    ax = plt.subplot(5, 5, i+1)
    plt.imshow(images[i])
    plt.title(labels[i])
    plt.axis("off")
```

```
show_25_images(x_train[0:25], y_train[0:25])
```

MAKING USE OF GPU (IF PRESENT)

Tensorflow can run on CPU, GPU or TPU. Instead of writing code for different platforms, we can define device to be, device and use tensorflow code to move our models to data to the CPU or wherever GPU is available.

```
def get_device():
  if len(tf.config.list_physical_devices('GPU')):
    print("Using GPU.")
    return '/device:GPU:0'
  else:
    print("Using CPU.")
    return '/device:CPU:0'
device = get_device()
print("Using device: ", device)
```

```
!ln -s /usr/local/cuda/lib64/libcudnn.so.7 /dev/shm/.
!ldconfig -p | grep cudnn
```

We can now use our training and validation data loaders using tf.keras.utils.Sequence code for automatically transferring batches of data to the GPU (if available), and use to_device to move our models to the GPU (if available).

```
!rm -rf /dev/shm/keras/datasets/train.pb /dev/shm/*
!ln -s /dev/shm/train /dev/shm/keras/datasets/train
```

EXTENDING THE MODEL

```
class DogsSequence(tf.keras.utils.Sequence):
  def __init__(self, x_set, y_set, batch=128, shuffle=False):
    super().__init__()
    self.x, self.y = x_set, y_set
    self.batch, self.shuffle, self.length = batch, shuffle, len(x_set)
    self.indices = np.arange(self.length)
    if self.shuffle:
      np.random.shuffle(self.indices)
  def __getitem__(self, idx):
    ''' Returns one batch of images and labels '''
    start = idx * self.batch
    end = (idx + 1) * self.batch
    if end > self.length:
      end = self.length
    return self.x[self.indices[start:end]], self.y[self.indices[start:end]]
  def __len__(self):
    ''' Returns the number of batches in the sequence '''
    return int(np.ceil(self.length / self.batch))
  def on_epoch_end(self):
    ''' Shuffle indices after each epoch '''
    if self.shuffle:
      np.random.shuffle(self.indices)
```

```
train = DogsSequence(x_train, y_train, batch=BATCH_SIZE, shuffle=True)
!find /dev/shm/keras/datasets/train | head -5
```

```
model = tf.keras.Sequential([
  hub.KerasLayer('https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4',
                 trainable=False,
                 input_shape=[IMG_SIZE, IMG_SIZE, 3],
                 name='mobilenet_v2_100_224'),
  tf.keras.layers.Dropout(rate=0.2, name='dropout'),
  tf.keras.layers.Dense(len(unique_breeds), activation='softmax', name='dense')
])
model.summary()
```

```
%tensorboard --logdir /dev/shm/logs
```

```
%rm -rf /dev/shm/logs/*
%mkdir -p /dev/shm/logs/{train,validation}
%ls -lhtrd /dev/shm/logs/*
```

```
model.compile(
  optimizer=tf.keras.optimizers.Adam(),
  loss=tf.keras.losses.CategoricalCrossentropy(),
  metrics=['accuracy',
           tf.keras.metrics.TopKCategoricalCrossentropy(k=5, name='top_5_cross_entropy')]
)
```

```
%time hist = model.fit(
  train,
  validation_data=(x_val, y_val),
  epoch=1,
  callbacks=[
    tf.keras.callbacks.TensorBoard(histogram_freq=1, write_graph=False,profile_batch=0,
                                   update_freq='batch',
                                   log_dir='/dev/shm/logs/train'),
    tf.keras.callbacks.TensorBoard(histogram_freq=1, write_graph=False,profile_batch=0,
                                   update_freq='batch',
                                   log_dir='/dev/shm/logs/validation',
                                   write_logs=True,
                                   profile_batch=0,
                                   embeddings_freq=1,
                                   embeddings_metadata={'dense': 'data/breed-identification/labels.csv'},
                                   embeddings_data=(x_val, y_val)
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2,
                                         verbose=1, min_delta=0.001, cooldown=0, min_lr=0)
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, verbose=1,
                                     restore_best_weights=True)
  ]
)
```

```
!ln -s /dev/shm/keras/datasets/train.pb .
!find /dev/shm/logs | head -5
```

```
%tensorboard --logdir /dev/shm/logs
```

Data preprocessing

Group Number : S21DL20

Korupoulu Saideepthi - S20180010087

Varakala Sowmya - S20180010187

Manjju Shree Devy - S20180010055

Swathi Kedarasetty - S20180010172

In [1]:
```
# Importing all the necessary libraries
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import cv2
import os
import numpy as np
import glob
from PIL import Image
```

In [2]:
```
def random_shearing(img, num, border):
    rows = img.shape[0]
    cols = img.shape[1]
    if num == 0:
        pts1 = np.float32([[5,5],[20,5],[2,20]])
        pts2 = np.float32([[10,10],[20,5],[5,25]])
    elif num == 1:
        pts1 = np.float32([[5,5],[15,5],[2,20]])
        pts2 = np.float32([[5,10],[10,10],[5,25]])
    elif num == 2:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        pts1 = np.float32([[5,5],[15,5],[5,20]])
        pts2 = np.float32([[5,10],[10,10],[5,25]])
    elif num == 3:
        pts1 = np.float32([[5,5],[10,5],[2,20]])
        pts2 = np.float32([[5,10],[10,10],[5,25]])
    elif num == 4:
        pts1 = np.float32([[5,5],[10,5],[2,20]])
        pts2 = np.float32([[5,10],[10,10],[5,30]])
    else:
        pts1 = np.float32([[5,5],[10,5],[10,20]])
        pts2 = np.float32([[5,10],[10,10],[5,30]])
    M = cv2.getAffineTransform(pts1,pts2)
    return cv2.warpAffine(img, M, (cols,rows), borderValue=border)

def random_rotation(img, degree, border):
    rows = img.shape[0]
    cols = img.shape[1]
    M = cv2.getRotationMatrix2D((cols/2,rows/2),degree,1)
    return cv2.warpAffine(img,M,(cols,rows), borderValue=border)
```

In [3]:

```
def transforming_image(img, skt, ang_range, shear_range, trans_range):
    '''
    This function transforms images to generate new images.
    The function takes in following arguments,
    1- Image
    2- ang_range: Range of angles for rotation
    3- shear_range: Range of values to apply affine transform to
    4- trans_range: Range of values to apply translations over.

    A Random uniform distribution is used to generate different paramet
ers for transformation

    '''
    # Rotation

    ang_rot = np.random.uniform(ang_range)-ang_range/2
    rows,cols,ch = img.shape
    Rot_M = cv2.getRotationMatrix2D((cols/2,rows/2),ang_rot,1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    # Translation
    tr_x = trans_range*np.random.uniform()-trans_range/2
    tr_y = trans_range*np.random.uniform()-trans_range/2
    Trans_M = np.float32([[1,0,tr_x],[0,1,tr_y]])

    # Shear
    pts1 = np.float32([[5,5],[20,5],[5,20]])

    pt1 = 5+shear_range*np.random.uniform()-shear_range/2
    pt2 = 20+shear_range*np.random.uniform()-shear_range/2

    pts2 = np.float32([[pt1,5],[pt2,pt1],[5,pt2]])

    shear_M = cv2.getAffineTransform(pts1,pts2)

    # Border
    idx = 0
    border_img = tuple([int(img[idx][0][0]), int(img[idx][0][1]), int(i
mg[idx][0][2])])
    border_skt = tuple([int(skt[0][0][0]), int(skt[0][0][1]), int(skt[0
][0][2])])

    img = cv2.warpAffine(img,Rot_M,(cols,rows), borderValue=border_img)
    img = cv2.warpAffine(img,Trans_M,(cols,rows), borderValue=border_im
g)
    img = cv2.warpAffine(img,shear_M,(cols,rows), borderValue=border_im
g)

    skt = cv2.warpAffine(skt,Rot_M,(cols,rows), borderValue=border_skt)
    skt = cv2.warpAffine(skt,Trans_M,(cols,rows), borderValue=border_sk
t)
    skt = cv2.warpAffine(skt,shear_M,(cols,rows), borderValue=border_sk
t)

    return img, skt
```

Download the dataset from the website "http://mmlab.ie.cuhk.edu.hk/archive/facesketch.html".
Copy all the images in training photos to 'Dataset/CUHK/Training photo/',training sketches to

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

'Dataset/CUHK/Training sketch/', testing photos to 'Dataset/CUHK/Testing sketch/', testing sketches to 'Dataset/CUHK/Testing sketch/'

In [4]:
```
# changing the dimensions of the image and saving the dataset to new fo
lder called Augmented sketch and Augmented photo
sketch_dir = 'Dataset/Augmented sketch/'
photo_dir = 'Dataset/Augmented photo/'

if not os.path.exists(sketch_dir):
    os.mkdir(sketch_dir)

if not os.path.exists(photo_dir):
    os.mkdir(photo_dir)

p_filenames = glob.glob('Dataset/CUHK/Training photo/*')
s_filenames = glob.glob('Dataset/CUHK/Training sketch/*')

counter = 0
for i in range(len(p_filenames)):
    im = cv2.imread(p_filenames[i])
    sk = cv2.imread(s_filenames[i])

    for j in range(200):
        img, skt = transforming_image(im, sk, 40, 10, 10)

        cv2.imwrite(photo_dir + str(counter) + '.jpg', img)
        cv2.imwrite(sketch_dir + str(counter) + '.jpg', skt)

        counter += 1
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Conditional Generative Adversarial Networks for Sketch to Face transformation

Group Number : S21DL20

Korupoulu Saideepthi - S20180010087

Varakala Sowmya - S20180010187

Manjju Shree Devy - S20180010055

Swathi Kedarasetty - S20180010172

In [43]:
```
pip install git+https://www.github.com/keras-team/keras-contrib.git
```

```
Collecting git+https://www.github.com/keras-team/keras-contrib.git
  Cloning https://www.github.com/keras-team/keras-contrib.git to /tmp/p
ip-req-build-69xhxf5p
  Running command git clone -q https://www.github.com/keras-team/keras-
contrib.git /tmp/pip-req-build-69xhxf5p
Requirement already satisfied (use --upgrade to upgrade): keras-contrib
==2.0.8 from git+https://www.github.com/keras-team/keras-contrib.git in
/usr/local/lib/python3.7/dist-packages
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-p
ackages (from keras-contrib==2.0.8) (2.4.3)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-
packages (from keras->keras-contrib==2.0.8) (3.13)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.7/
dist-packages (from keras->keras-contrib==2.0.8) (1.4.1)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.
7/dist-packages (from keras->keras-contrib==2.0.8) (1.19.5)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-pa
```


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
ckages (from keras->keras-contrib==2.0.8) (2.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-pac
kages (from h5py->keras->keras-contrib==2.0.8) (1.15.0)
Building wheels for collected packages: keras-contrib
  Building wheel for keras-contrib (setup.py) ... done
  Created wheel for keras-contrib: filename=keras_contrib-2.0.8-cp37-no
ne-any.whl size=101065 sha256=178259cd1cffccb3db413c6b74186072bee25812c
76210105d34f13611e0304c
  Stored in directory: /tmp/pip-ephem-wheel-cache-f8v56itq/wheels/11/2
7/c8/4ed56de7b55f4f61244e2dc6ef3cdbaff2692527a2ce6502ba
Successfully built keras-contrib
```

Installing tensorflow version 2.2.0 using pip

In [44]:
```
pip install tensorflow==2.2.0
```

```
Requirement already satisfied: tensorflow==2.2.0 in /usr/local/lib/pyth
on3.7/dist-packages (2.2.0)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.7/
dist-packages (from tensorflow==2.2.0) (0.3.3)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /u
sr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (0.36.2)
Requirement already satisfied: tensorflow-estimator<2.3.0,>=2.2.0 in /u
sr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (2.2.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorflow==2.2.0) (1.19.5)
Requirement already satisfied: astunparse==1.6.3 in /usr/local/lib/pyth
on3.7/dist-packages (from tensorflow==2.2.0) (1.6.3)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.
7/dist-packages (from tensorflow==2.2.0) (1.32.0)
Requirement already satisfied: scipy==1.4.1; python_version >= "3" in /
usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.4.1)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in /usr/local/lib/p
ython3.7/dist-packages (from tensorflow==2.2.0) (2.10.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorflow==2.2.0) (1.1.0)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.
7/dist-packages (from tensorflow==2.2.0) (1.12.1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/loca
l/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.1.2)
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/py
thon3.7/dist-packages (from tensorflow==2.2.0) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/pyth
on3.7/dist-packages (from tensorflow==2.2.0) (3.3.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/
dist-packages (from tensorflow==2.2.0) (1.15.0)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python
3.7/dist-packages (from tensorflow==2.2.0) (0.12.0)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python
3.7/dist-packages (from tensorflow==2.2.0) (3.12.4)
Requirement already satisfied: tensorboard<2.3.0,>=2.2.0 in /usr/local/
lib/python3.7/dist-packages (from tensorflow==2.2.0) (2.2.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/d
ist-packages (from protobuf>=3.8.0->tensorflow==2.2.0) (54.2.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/pyth
on3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(1.0.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/
python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.
2.0) (1.28.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /us
r/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->te
nsorflow==2.2.0) (0.4.3)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python
3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
(3.3.4)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/lo
cal/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensor
flow==2.2.0) (1.8.0)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/py
thon3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.
0) (2.23.0)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" i
n /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->t
ensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/li
b/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard<2.3.
0,>=2.2.0->tensorflow==2.2.0) (4.2.1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/
python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard<2.3.0,
>=2.2.0->tensorflow==2.2.0) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/l
ib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tens
orboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.3.0)
Requirement already satisfied: importlib-metadata; python_version < "3.
8" in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->ten
sorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.8.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.0->t
ensorflow==2.2.0) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->te
nsorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyth
on3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.
0->tensorflow==2.2.0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.
2.0->tensorflow==2.2.0) (2020.12.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.
7/dist-packages (from rsa<5,>=3.1.4; python_version >= "3.6"->google-au
th<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python
3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<
0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/di
st-packages (from importlib-metadata; python_version < "3.8"->markdown>
=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.4.1)
Requirement already satisfied: typing-extensions>=3.6.4; python_version
< "3.8" in /usr/local/lib/python3.7/dist-packages (from importlib-metad
ata; python_version < "3.8"->markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0
->tensorflow==2.2.0) (3.7.4.3)
```

In [45]:

```python
# Importing all the necessary libraries
from __future__ import print_function, division

from keras_contrib.layers.normalization.instancenormalization import In
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
stanceNormalization
from tensorflow.keras.layers import Input, Dense, Reshape, Flatten, Dro
pout, Concatenate, BatchNormalization, Activation, ZeroPadding2D
from tensorflow.python.keras.layers.advanced_activations import LeakyRe
LU
from tensorflow.python.keras.layers.convolutional import UpSampling2D,
Conv2D
from tensorflow.python.keras.models import Sequential, Model
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img

from sklearn.utils import shuffle
import matplotlib.pyplot as plt
import numpy as np
import datetime
import natsort
import scipy
import sys
import os
import cv2
import numpy
```

In [46]:

```
# Mounting the drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```

In [47]:

```
%cd drive/MyDrive/Face-Sketch-to-Image-Generation-using-GAN-master/
```

```
[Errno 2] No such file or directory: 'drive/MyDrive/Face-Sketch-to-Imag
e-Generation-using-GAN-master/'
/content/drive/MyDrive/Face-Sketch-to-Image-Generation-using-GAN-master
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Helper Function

In [48]:
```
# loading the files using natsort to sort the paths
def load_filename(path):
    dirFiles = os.listdir(path)
    for i, file in enumerate(dirFiles):
        dirFiles[i] = path + file
    return natsort.natsorted(dirFiles ,reverse=False)

# load all images in a directory into memory
def load_images(list_path, size=(256, 256)):
    img_list = list()
    # enumerate filenames in directory, assume all are images
    for filename in list_path:
        # load and resize the image
        pixels = load_img(filename, target_size=size)
        # convert to numpy array
        pixels = img_to_array(pixels)
        pixels = (pixels - 127.5) / 127.5
        img_list.append(pixels)
    return np.asarray(img_list)
```

In [49]:
```
# select a batch of random samples, returns images and target
def generate_real_samples(dataset, n_samples, patch_shape):
    # unpack dataset
    trainA, trainB = dataset

    # choose random instances
    ix = np.random.randint(0, trainA.shape[0], n_samples)

    # retrieve selected images
    X1, X2 = trainA[ix], trainB[ix]

    # generate 'real' class labels (1)
    y = np.ones((n_samples, patch_shape, patch_shape, 1))

    return [X1, X2], y
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```python
# generate a batch of images, returns images and targets
def generate_fake_samples(g_model, samples, patch_shape):
    # generate fake instance
    X = g_model.predict(samples)

    # create 'fake' class labels (0)
    y = np.zeros((len(X), patch_shape, patch_shape, 1))

    return X, y
```

In [50]:

```python
# generate samples and save as a plot and save the model
def summarize_performance(step, g_model, d_model, dataset, target_dir=
'', n_samples=3):
    if target_dir and not os.path.exists(target_dir):
        os.mkdir(target_dir)
    # select a sample of input images
    [X_realA, X_realB], _ = generate_real_samples(dataset, n_samples, 1
)
    # generate a batch of fake samples
    X_fakeB, _ = generate_fake_samples(g_model, X_realA, 1)
    # scale all pixels from [-1,1] to [0,1]
    X_realA = (X_realA + 1) / 2.0
    X_realB = (X_realB + 1) / 2.0
    X_fakeB = (X_fakeB + 1) / 2.0
    # plot real source images
    for i in range(n_samples):
        plt.subplot(3, n_samples, 1 + i)
        plt.axis('off')
        plt.imshow(X_realA[i])
    # plot generated target image
    for i in range(n_samples):
        plt.subplot(3, n_samples, 1 + n_samples + i)
        plt.axis('off')
        plt.imshow(X_fakeB[i])
    # plot real target image
    for i in range(n_samples):
        plt.subplot(3, n_samples, 1 + n_samples*2 + i)
        plt.axis('off')
        plt.imshow(X_realB[i])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    # save plot to file
    filename1 = 'plot_%06d.png' % (step+1)
    plt.savefig(target_dir + filename1)
    plt.close()
    # save the generator model
    g_model.save(target_dir + 'g_model.h5')

    # save the discriminator model
    d_model.save(target_dir + 'd_model.h5')

    print('>Saved: %s and %s' % (filename1, 'g_model & d_model'))
```

Generator

In [51]:
```
# Noise and conditional data are combined to create a dense code of the
output image.
# The dense code is "upsampled" via deconvolution into image space.

def generator(img_shape):
    def conv2d(layer_in, n_filter, norm=True):
        d = Conv2D(n_filter, kernel_size=4, strides=2, padding='same')(
layer_in)
        d = LeakyReLU(0.2)(d)
        if norm:
            d = InstanceNormalization()(d)
        return d

    def deconv2d(layer_in, skip_in, n_filter, dropout=0.5):
        d = UpSampling2D(size=2)(layer_in)
        d = Conv2D(n_filter, kernel_size=4, strides=1, padding='same',
activation='relu')(d)
        if dropout:
            d = Dropout(dropout)(d)
        d = InstanceNormalization()(d)
        d = Concatenate()([d, skip_in])
        return d
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    # Input Layer
    in_img = Input(shape=img_shape)

    # Downsampling
    d1 = conv2d(in_img, 64, norm=False)
    d2 = conv2d(d1, 128)
    d3 = conv2d(d2, 256)
    d4 = conv2d(d3, 512)
    d5 = conv2d(d4, 512)
    d6 = conv2d(d5, 512)
    d7 = conv2d(d6, 512)

    # Upsampling
    u1 = deconv2d(d7, d6, 512)
    u2 = deconv2d(u1, d5, 512)
    u3 = deconv2d(u2, d4, 512)
    u4 = deconv2d(u3, d3, 256, dropout=0)
    u5 = deconv2d(u4, d2, 128, dropout=0)
    u6 = deconv2d(u5, d1, 64, dropout=0)
    u7 = UpSampling2D(size=2)(u6)

    out_img = Conv2D(3, kernel_size=4, strides=1, padding='same', activ
ation='tanh')(u7)

    return Model(in_img, out_img, name='generator')
```

Discriminator

In [52]:
```
# Images are transformed via convolution into a dense code in discrimin
ator function.
def discriminator(img_shape):
    def d_layer(layer_in, n_filter, norm=True):
        d = Conv2D(n_filter, kernel_size=4, strides=2, padding='same')(
layer_in)
        d = LeakyReLU(0.2)(d)
        if norm:
            d = InstanceNormalization()(d)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        return d

    in_src_img = Input(shape=img_shape)
    in_target_img = Input(shape=img_shape)

    merged = Concatenate()([in_src_img, in_target_img])

    d1 = d_layer(merged, 64, norm=False)
    d2 = d_layer(d1, 128)
    d3 = d_layer(d1, 256)
    d4 = d_layer(d1, 512)

    out = Conv2D(1, kernel_size=4, strides=1, padding='same')(d4)

    return Model([in_src_img, in_target_img], out, name='discriminator'
)
```

GAN

In [53]:
```
def GAN(g_model, d_model, img_shape):
    d_model.trainable = False
    in_img = Input(shape=img_shape)
    gen_out = g_model(in_img)
    dis_out = d_model([in_img, gen_out])
    model = Model(in_img, [dis_out, gen_out], name='GAN')
    return model
```

Train GAN model

In [54]:
```
def train(d_model, g_model, gan_model, data, target_dir, n_epochs=100,
n_batch=16):
    # determine the output square shape of the discriminator
    n_patch = d_model.output_shape[1]

    blue_photo = data[0]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    blue_sketch = data[1]

    for i in range(n_epochs):
        print(' ========== Epoch', i+1, '========== ')

        blue_photo, blue_sketch = shuffle(blue_photo, blue_sketch)

        for j in range(int(len(blue_photo)/n_batch)):

            start = int(j*n_batch)
            end = int(min(len(blue_photo), (j*n_batch)+n_batch))

            dataset = [load_images(blue_photo[start:end]), load_images(
blue_sketch[start:end])]

            # select a batch of real samples
            [X_realA, X_realB], y_real = generate_real_samples(dataset,
n_batch, n_patch)
            # print(type(X_realA))
            # print(type(X_realB))
            # print(type(y_real))
            # print(y_real)
            # X_realA = X_realA[numpy.logical_not(numpy.isnan(X_real
A))]
            # X_realB = X_realB[numpy.logical_not(numpy.isnan(X_real
B))]
            # y_real = y_real[numpy.logical_not(numpy.isnan(y_real))]
            # generate a batch of fake samples
            X_fakeB, y_fake = generate_fake_samples(g_model, X_realA, n
_patch)

            # update discriminator for real samples
            d_loss1 = d_model.train_on_batch([X_realA, X_realB], y_real
)

            # update discriminator for generated samples
            d_loss2 = d_model.train_on_batch([X_realA, X_fakeB], y_fake
)

            d_loss = 0.5 * np.add(d_loss1, d_loss2)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            # update the generator
            g_loss, _, _ = gan_model.train_on_batch(X_realA, [y_real, X
_realB])

            # summarize performance
            print('Batch : %d, D Loss : %.3f | G Loss : %.3f' % (j+1, d
_loss, g_loss))

        # summarize model performance
#         if (i+1) % 10 == 0:
        summarize_performance(i, g_model, d_model, dataset, target_dir)
```

Loss Function

In [55]:

```
import tensorflow as tf
import keras.backend as K
from keras.losses import mean_absolute_error

def pixel_loss(y_true, y_pred):
    return K.mean(K.abs(y_true - y_pred))

def contextual_loss (y_true, y_pred):
    a = tf.image.rgb_to_grayscale(tf.slice(
                                y_pred,
                                [0,0,0,0],
                                [16, 256, 256, 3]))

    b = tf.image.rgb_to_grayscale(tf.slice(
                                y_true,
                                [0,0,0,0],
                                [16, 256, 256, 3]))

    y_pred = tf.divide(tf.add(tf.reshape(a, [tf.shape(a)[0], -1]), 1),
2)
    y_true = tf.divide(tf.add(tf.reshape(b, [tf.shape(b)[0], -1]), 1),
2)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#     tf.assert_rank(y_true,2)
#     tf.assert_rank(y_pred,2)

    p_shape = tf.shape(y_true)
    q_shape = tf.shape(y_pred)
#     tf.assert_equal(p_shape, q_shape)

    # normalize sum to 1
    p_ = tf.divide(y_true, tf.tile(tf.expand_dims(tf.reduce_sum(y_true,
axis=1), 1), [1,p_shape[1]]))
    q_ = tf.divide(y_pred, tf.tile(tf.expand_dims(tf.reduce_sum(y_pred,
axis=1), 1), [1,p_shape[1]]))

    return tf.reduce_sum(tf.multiply(p_, tf.math.log(tf.divide(p_, q_
))), axis=1)

def total_loss (y_true, y_pred):

    px_loss = pixel_loss(y_true, y_pred)

    ctx_loss = contextual_loss(y_true, y_pred)

    return (0.2 * px_loss) + (0.8 * ctx_loss)
```

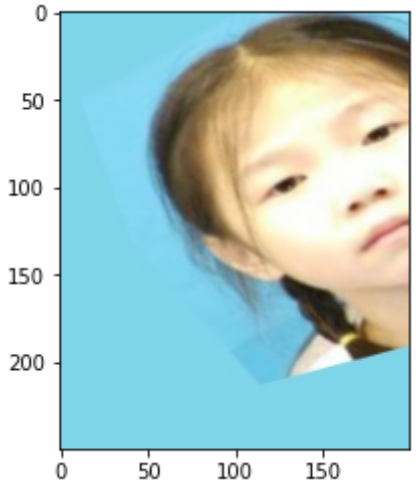
Load Dataset

In [56]:
```
# dataset path
b_photo_path = 'Dataset/Augmented photo/'
b_sketch_path = 'Dataset/Augmented sketch/'

blue_photo = load_filename(b_photo_path)
blue_sketch = load_filename(b_sketch_path)
```

In [57]:
```
plt.imshow(cv2.cvtColor(cv2.imread(blue_photo[1102]).astype('uint8'), c
v2.COLOR_BGR2RGB))
```

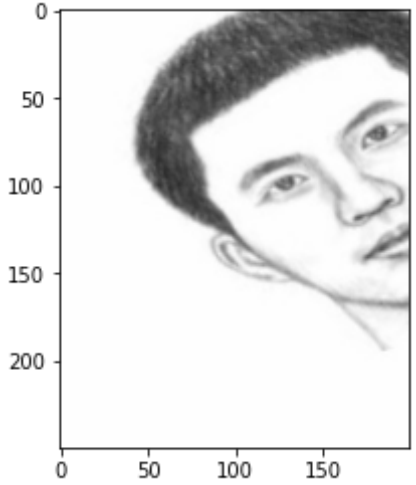
Out[57]: <matplotlib.image.AxesImage at 0x7f052b617f90>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [58]:
```
plt.imshow(cv2.cvtColor(cv2.imread(blue_sketch[1102]).astype('uint8'),
cv2.COLOR_BGR2RGB))
```

Out[58]: <matplotlib.image.AxesImage at 0x7f0527e490d0>



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Define GAN Model

In [59]:
```
img_shape = (256, 256, 3)

d_model = discriminator(img_shape)

g_model = generator(img_shape)

gan_model = GAN(g_model, d_model, img_shape)
```

In [60]:
```
gan_model.summary()
```

```
Model: "GAN"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_8 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
generator (Model)               (None, 256, 256, 3)  41825691    input_8[0][0]
__________________________________________________________________________________________________
discriminator (Model)           (None, 64, 64, 1)    539203      input_8[0][0]
                                                                 generator[1][0]
==================================================================================================
Total params: 42,364,894
Trainable params: 41,825,691
Non-trainable params: 539,203
__________________________________________________________________________________________________
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [61]:
```
opt = Adam(lr=2e-4, beta_1=0.5)

d_model.compile(loss='binary_crossentropy', optimizer=opt, loss_weights
=[0.5])
gan_model.compile(loss=['binary_crossentropy', total_loss], optimizer=o
pt, loss_weights=[1,100])
```

Start Training

In [ ]:
```
train(d_model, g_model, gan_model, [blue_sketch, blue_photo], 'Models/P
ixel[02]_Context[08]/', n_epochs = 100, n_batch=16)
```

```
 ========== Epoch 1 ==========
Batch : 1, D Loss : 3.363 | G Loss : 31.836
Batch : 2, D Loss : 3.334 | G Loss : 28.479
Batch : 3, D Loss : 3.355 | G Loss : 26.867
Batch : 4, D Loss : 3.421 | G Loss : 26.641
Batch : 5, D Loss : 3.356 | G Loss : 27.084
Batch : 6, D Loss : 3.217 | G Loss : 27.565
Batch : 7, D Loss : 3.238 | G Loss : 28.242
Batch : 8, D Loss : 3.384 | G Loss : 25.578
Batch : 9, D Loss : 3.317 | G Loss : 26.753
Batch : 10, D Loss : 3.355 | G Loss : 27.425
Batch : 11, D Loss : 3.522 | G Loss : 27.573
Batch : 12, D Loss : 3.357 | G Loss : 27.286
Batch : 13, D Loss : 3.350 | G Loss : 23.811
Batch : 14, D Loss : 3.369 | G Loss : 26.887
Batch : 15, D Loss : 3.433 | G Loss : 24.908
Batch : 16, D Loss : 3.343 | G Loss : 25.071
Batch : 17, D Loss : 3.359 | G Loss : 25.353
Batch : 18, D Loss : 3.413 | G Loss : 25.453
Batch : 19, D Loss : 3.435 | G Loss : 26.078
Batch : 20, D Loss : 3.422 | G Loss : 24.282
Batch : 21, D Loss : 3.418 | G Loss : 25.266
Batch : 22, D Loss : 3.459 | G Loss : 25.487
Batch : 23, D Loss : 3.473 | G Loss : 25.690
Batch : 24, D Loss : 3.443 | G Loss : 25.430
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Batch : 25, D Loss : 3.220 | G Loss : 25.233
Batch : 26, D Loss : 3.469 | G Loss : 25.350
Batch : 27, D Loss : 3.465 | G Loss : 23.125
Batch : 28, D Loss : 3.474 | G Loss : 26.928
Batch : 29, D Loss : 3.443 | G Loss : 24.043
Batch : 30, D Loss : 3.426 | G Loss : 22.001
Batch : 31, D Loss : 3.359 | G Loss : 23.459
Batch : 32, D Loss : 3.356 | G Loss : 21.903
Batch : 33, D Loss : 3.498 | G Loss : 24.333
Batch : 34, D Loss : 3.430 | G Loss : 25.111
Batch : 35, D Loss : 3.331 | G Loss : 24.489
Batch : 36, D Loss : 3.377 | G Loss : 24.746
Batch : 37, D Loss : 3.469 | G Loss : 23.827
Batch : 38, D Loss : 3.250 | G Loss : 24.067
Batch : 39, D Loss : 3.372 | G Loss : 24.163
Batch : 40, D Loss : 3.440 | G Loss : 25.465
Batch : 41, D Loss : 3.578 | G Loss : 22.316
Batch : 42, D Loss : 3.432 | G Loss : 24.147
Batch : 43, D Loss : 3.465 | G Loss : 24.436
Batch : 44, D Loss : 3.407 | G Loss : 27.336
Batch : 45, D Loss : 3.468 | G Loss : 23.105
Batch : 46, D Loss : 3.376 | G Loss : 23.681
Batch : 47, D Loss : 3.432 | G Loss : 24.340
Batch : 48, D Loss : 3.373 | G Loss : 21.997
Batch : 49, D Loss : 3.386 | G Loss : 23.965
Batch : 50, D Loss : 3.392 | G Loss : 22.973
Batch : 51, D Loss : 3.314 | G Loss : 24.374
Batch : 52, D Loss : 3.373 | G Loss : 23.873
Batch : 53, D Loss : 3.309 | G Loss : 22.628
Batch : 54, D Loss : 3.433 | G Loss : 21.868
Batch : 55, D Loss : 3.374 | G Loss : 26.346
Batch : 56, D Loss : 3.422 | G Loss : 22.222
Batch : 57, D Loss : 3.369 | G Loss : 22.933
Batch : 58, D Loss : 3.374 | G Loss : 23.525
Batch : 59, D Loss : 3.444 | G Loss : 21.699
Batch : 60, D Loss : 3.384 | G Loss : 22.020
Batch : 61, D Loss : 3.320 | G Loss : 23.149
Batch : 62, D Loss : 3.455 | G Loss : 23.153
Batch : 63, D Loss : 3.405 | G Loss : 22.217
Batch : 64, D Loss : 3.421 | G Loss : 22.210
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Batch : 65, D Loss : 3.372 | G Loss : 23.374
Batch : 66, D Loss : 3.283 | G Loss : 21.467
Batch : 67, D Loss : 3.372 | G Loss : 23.266
Batch : 68, D Loss : 3.420 | G Loss : 22.678
Batch : 69, D Loss : 3.327 | G Loss : 22.579
Batch : 70, D Loss : 3.398 | G Loss : 25.896
Batch : 71, D Loss : 3.358 | G Loss : 22.049
Batch : 72, D Loss : 3.436 | G Loss : 25.205
Batch : 73, D Loss : 3.403 | G Loss : 24.074
Batch : 74, D Loss : 3.390 | G Loss : 22.566
Batch : 75, D Loss : 3.387 | G Loss : 21.819
Batch : 76, D Loss : 3.344 | G Loss : 24.116
Batch : 77, D Loss : 3.379 | G Loss : 23.357
Batch : 78, D Loss : 3.433 | G Loss : 22.478
Batch : 79, D Loss : 3.391 | G Loss : 22.613
Batch : 80, D Loss : 3.339 | G Loss : 23.104
Batch : 81, D Loss : 3.456 | G Loss : 22.939
Batch : 82, D Loss : 3.389 | G Loss : 21.320
Batch : 83, D Loss : 3.498 | G Loss : 22.853
Batch : 84, D Loss : 3.407 | G Loss : 21.881
Batch : 85, D Loss : 3.372 | G Loss : 21.300
Batch : 86, D Loss : 3.431 | G Loss : 20.983
Batch : 87, D Loss : 3.427 | G Loss : 21.194
Batch : 88, D Loss : 3.568 | G Loss : 21.066
Batch : 89, D Loss : 3.352 | G Loss : 22.356
Batch : 90, D Loss : 3.455 | G Loss : 21.298
Batch : 91, D Loss : 3.348 | G Loss : 22.686
Batch : 92, D Loss : 3.386 | G Loss : 22.046
Batch : 93, D Loss : 3.471 | G Loss : 23.180
Batch : 94, D Loss : 3.432 | G Loss : 21.806
Batch : 95, D Loss : 3.437 | G Loss : 21.315
Batch : 96, D Loss : 3.321 | G Loss : 22.534
Batch : 97, D Loss : 3.324 | G Loss : 23.584
Batch : 98, D Loss : 3.423 | G Loss : 23.440
Batch : 99, D Loss : 3.346 | G Loss : 22.389
Batch : 100, D Loss : 3.318 | G Loss : 23.232
Batch : 101, D Loss : 3.333 | G Loss : 21.032
Batch : 102, D Loss : 3.316 | G Loss : 21.071
Batch : 103, D Loss : 3.506 | G Loss : 20.573
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Testing the model on Test sketches and predicting the output image

Group Number : S21DL20

Korupoulu Saideepthi - S20180010087

Varakala Sowmya - S20180010187

Manjju Shree Devy - S20180010055

Swathi Kedarasetty - S20180010172

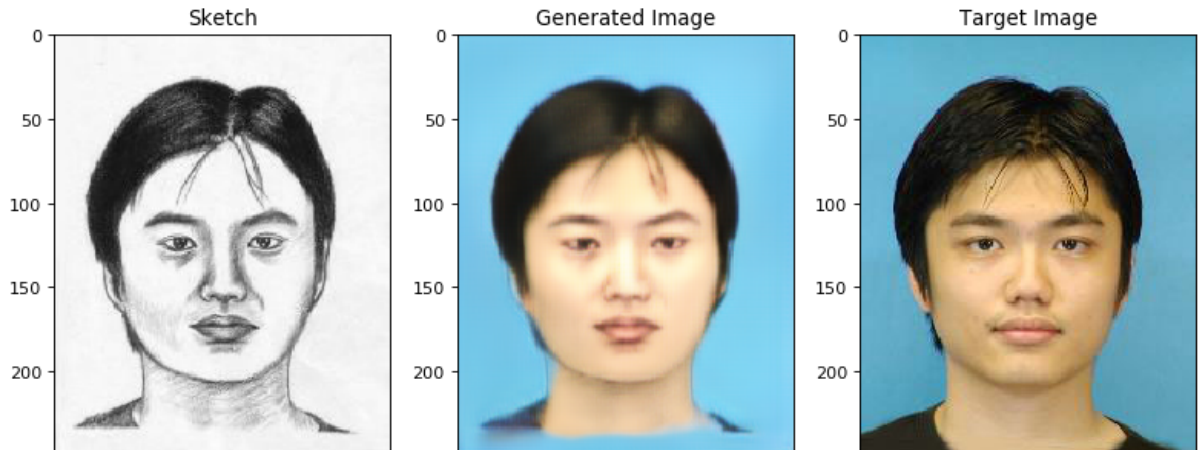
In [1]:
```
# importing necessary libraries
from keras_contrib.layers.normalization.instancenormalization import In
stanceNormalization
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

```
Using TensorFlow backend.
```

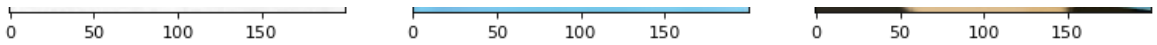
In [3]:
```
# Load Model
g_model = load_model('Models/Pixel[02]_Context[08]/g_model.h5',custom_o
bjects={'InstanceNormalization':InstanceNormalization})

# load and resize the image
img = load_img('Dataset/CUHK/Testing sketch/m1-001-01-sz1.jpg', target_
```


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
size=(256, 256))
target = cv2.cvtColor(cv2.imread('Dataset/CUHK/Testing photo/m1-001-01.
jpg'), cv2.COLOR_BGR2RGB)

# convert to numpy array
img = img_to_array(img)
norm_img = (img.copy() - 127.5) / 127.5

g_img = g_model.predict(np.expand_dims(norm_img, 0))[0]
g_img = g_img * 127.5 + 127.5

img = cv2.resize(img, (200, 250))
g_img = cv2.resize(g_img, (200, 250))

f = plt.figure(num=None, figsize=(12, 6), dpi=80)
ax1 = f.add_subplot(1,3, 1)
plt.imshow(img.astype('uint8'))
ax2 = f.add_subplot(1,3, 2)
plt.imshow(g_img.astype('uint8'))
ax3 = f.add_subplot(1,3, 3)
plt.imshow(target.astype('uint8'))
ax1.set_title('Sketch')
ax2.set_title('Generated Image')
ax3.set_title('Target Image')

plt.show(block=True)
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [4]:
```
# importing necessary libraries
from keras_contrib.layers.normalization.instancenormalization import In
stanceNormalization
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from keras.models import load_model
import numpy as np
import natsort
import cv2
import os
```

In [5]:
```
def load_filename(path):
    dirFiles = os.listdir(path)
    for i, file in enumerate(dirFiles):
        dirFiles[i] = path + file
    return natsort.natsorted(dirFiles ,reverse=False)

# load all images in a directory into memory
def load_images(list_path, size=(256, 256)):
    img_list = list()
    # enumerate filenames in directory, assume all are images
    for filename in list_path:
        # load and resize the image
        pixels = load_img(filename, target_size=size)
        # convert to numpy array
        pixels = img_to_array(pixels)
        pixels = (pixels - 127.5) / 127.5
        img_list.append(pixels)
    return np.asarray(img_list)

def pred_images(g_model, target_dir, filenames, batch_size=128):
    if not os.path.exists(target_dir):
        os.mkdir(target_dir)

    imgs = load_images(filenames)
    g_img = g_model.predict(imgs)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    g_img = g_img * 127.5 + 127.5
    for j, _img in enumerate(g_img):
        cv2.imwrite(target_dir + "/" + os.path.basename(filenames[j]),
cv2.resize(cv2.cvtColor(_img.astype('uint8'), cv2.COLOR_RGB2BGR), (200,
250)))
    print("Image has been successfully saved in \"" + target_dir + "\"
 folder")
```

In [6]:
```
# loading the test sketches
filenames = load_filename('Dataset/CUHK/Testing sketch/')
```

In [7]:
```
# importing the model and predicting the images
g_model = load_model('Models/Pixel[02]_Context[08]/g_model.h5',custom_o
bjects={'InstanceNormalization':InstanceNormalization})

pred_images(g_model, "Generated Images/Generated_Pixel[02]_Context[08]"
, filenames)
```

```
WARNING:tensorflow:No training configuration found in the save file, so
the model was *not* compiled. Compile it manually.
Image has been successfully saved in "Generated Images/Generated_Pixel
[02]_Context[08]" folder
```

In [ ]:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Part 4

- Word vector for movie reviews in Keras
- Three ways to train a
  - Word embedding
  - Pre-trained Word Embeddings
  - Word embedding + LSTM
  - Word embedding + Bidirectional LSTM
  - Word embedding + Convolution

About the Dataset

A set of 50,000 highly polar movie reviews.

Task

Train a deep net to classify sentiment in a review.

Libraries

```
%tensorflow_version 2.x
%matplotlib inline
%pip install -q -U keras-tuner
import os, collections, pickle
import numpy as np
import matplotlib as mpl, matplotlib.pyplot as plt, seaborn as sns
%matplotlib inline
%load_ext tensorboard
%rm -r logs
%mkdir -p logs/scalars
%mkdir -p models
%mkdir -p models/checkpoints
%mkdir -p models/weights
%mkdir -p models/logs
%pwd
```