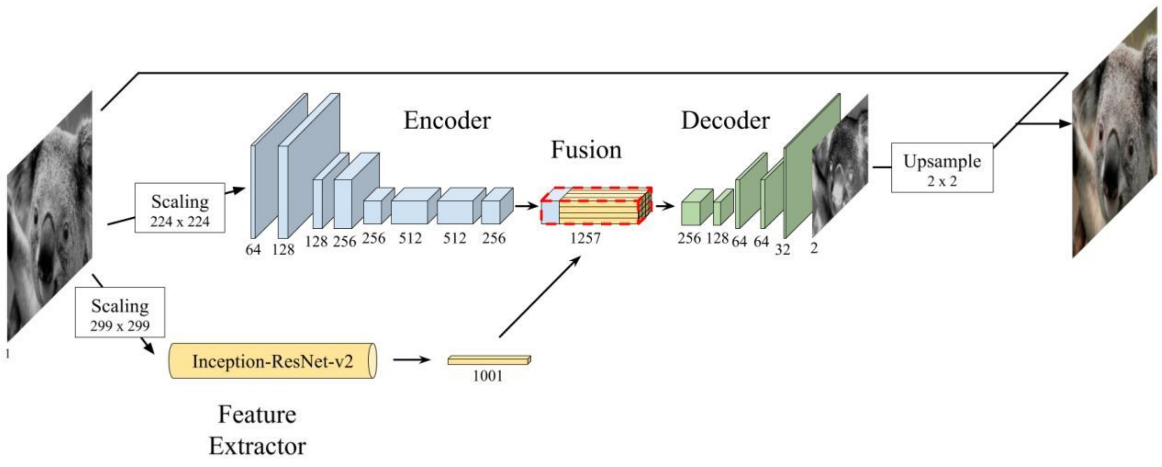
Image Colourization

Saurabh Bazari 160101061 *Namit Kumar 160101039* *Abhinav Mishra 160101005,* *Ritik Agrawal 160101055* *G Sharath Kumar 150101023*

Image Colorization using CNNs and Inception-Resnet-v2:

Implementation:
The paper describes a novel architecture using a Convolutional Neural Network(CNN) model which was trained from scratch with a pre-trained Inception-ResNet-v2 model for high-level feature extraction.



The implementation can be divided into 4 parts:

Encoder:
- Input to the encoder was required to be in 224 X 224 size, therefore dataset image was transformed from their original to size to 224 X 224.
- The encoder consisted of convolutional layer size of 64, 128, 128, 256, 256, 512, 512, 256
- Downsampling is applied at first, third and fifth layers with stride 2 to reduce the number of computations required.
- The final output obtained after all layers of the encoder is of 256 X 28 X 28

Feature Extractor:
- Input to the encoder was required to be in 299 X 299 size, therefore dataset image was transformed from their original to size to 299 X 299.
- Pre-trained model of Inception-ResNet-v2 is used for feature extraction
- The final output obtained is of size 1000 X 1 X 1

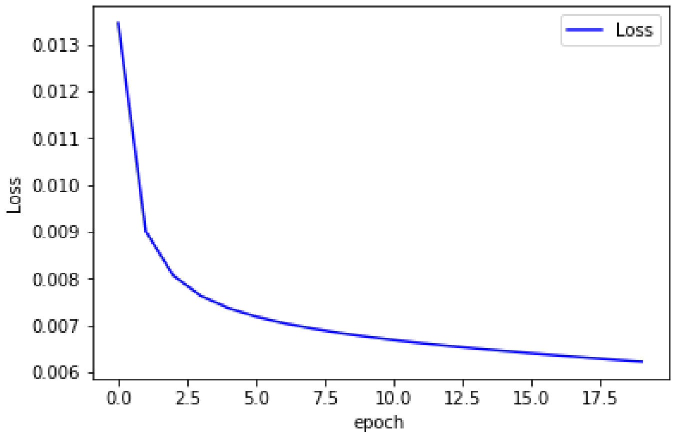
Fusion :
- This part of architecture combines the outputs of the encoder and feature extractor.

- Due to the difference in the output size of both the components the output of Feature Extractor is copied 28 X 28 times.
- The output of both the components is now combined to create an input for the decoder component of size 1256 X 28 X 28.

Decoder:

- Input to the decoder is of size 1256 X 28 X 28.
- The decoder consisted of convolutional layer size of 256,128,64,64,32.
- with upsampling at first, third and fifth layers.
- The final output of the decoder is size 224 X 224 X 2, the output is a representation of the image's a and b component in the Lab format of image representation.
- Combining the a and b value obtained from the model with the already present L value the final image is produced.

Results:
For training and testing our model, we used the "Cats and Dogs dataset to train a DL model" (https://www.kaggle.com/tongpython/cat-and-dog/download) which consisted of 2000 images of cats and dogs.
The batch size used for training purposes was 20. The number of epochs was 20.



Graph of loss vs epoch

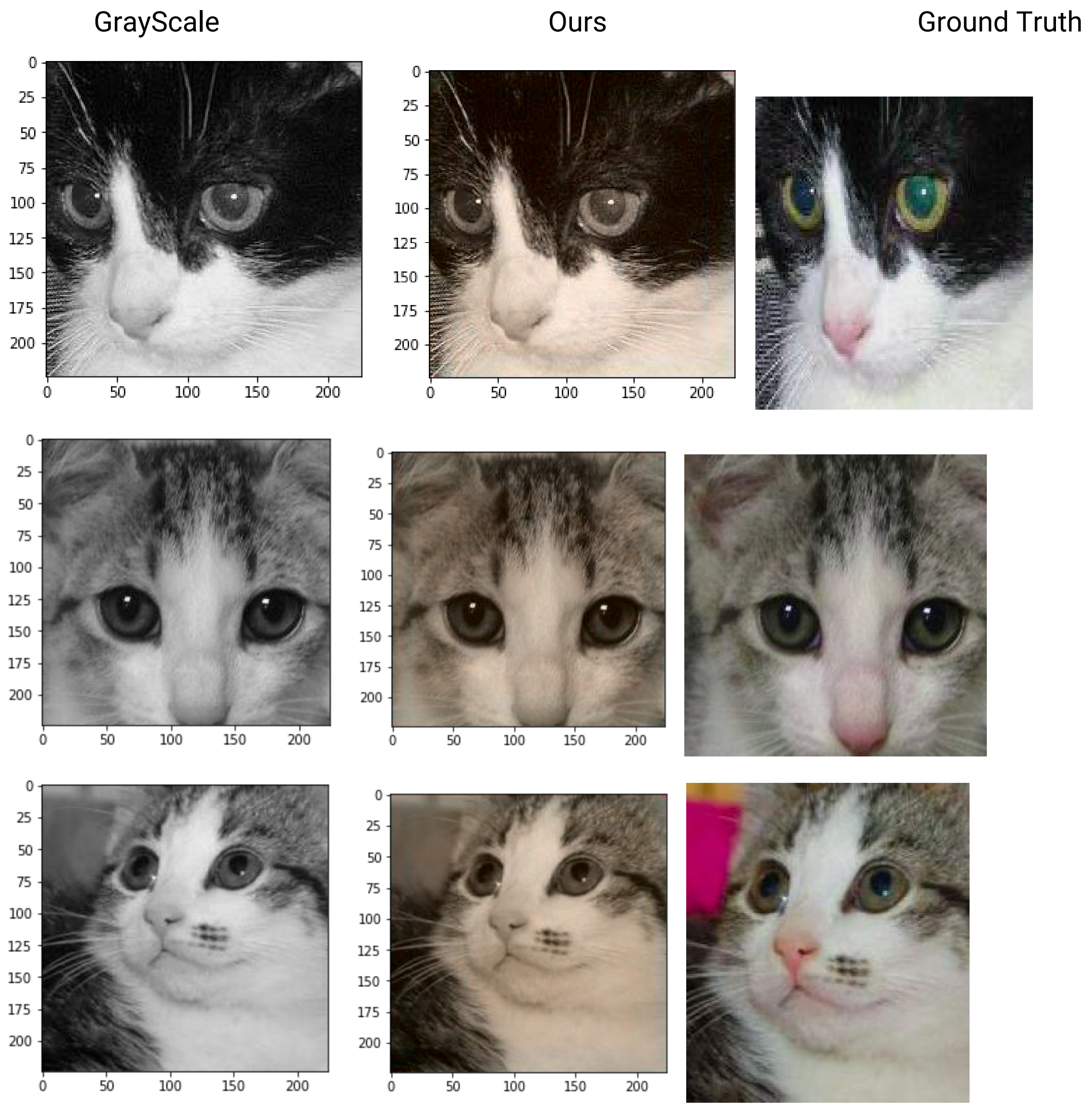
Limitations:

- The implementation is able to color out high-level components of the image such as objects like water bodies, trees or sky. But for the small level components, the performance is not that satisfactory.

- As we only used a reduced subset of ImageNet, only a small portion of the spectrum of possible subjects is represented, therefore, the performance on unseen images highly depends on their specific contents.

- The implementation does factors in any given context of the image provided.
- The paper for finding accuracy depended on the survey conducted for discriminating between generated output and what should be the ideal ground truth image.

Following is some of the output generated by the model on given test cases.



Suggested Improvements:

- Training the model on a larger dataset should potentially help the model to learn variety while coloring the provided image.
- Output can be improved by following an approach similar to variational autoencoders, allowing for image generation by sampling from a probability distribution as done in the paper (https://arxiv.org/pdf/1603.08511.pdf).
- Discriminator can be applied for discriminating between generated output and ground truth image.
- For factors in the context provided with the image we can use the FILM model as done in the paper (https://arxiv.org/pdf/1804.06026v1.pdf) which uses captions to

generate the output.

- The reference image can be used to improve the generated output as done in the paper (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8781608) .

Image Colorization using Generative Adversarial Network:

Implementation:

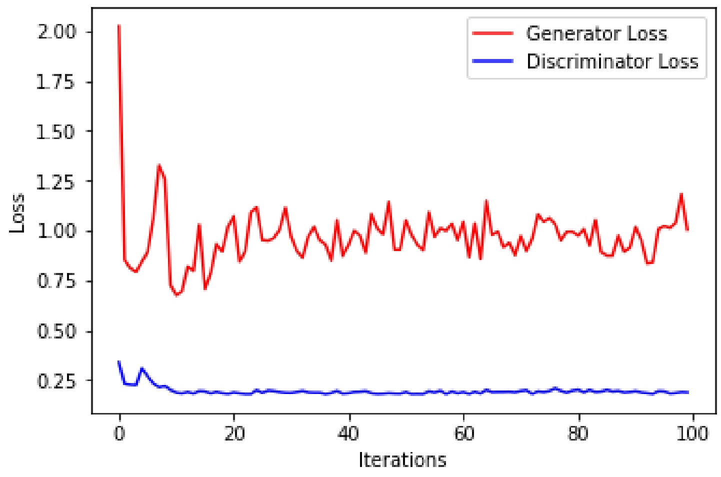
Generative Adversarial Network (GAN) is a generative model used to get colorized images from the grayscale images. It mainly consists of 2 parts: Generator and Discriminator. The paper proposes models for both parts. The generator tries to create colorized images that are similar to the ground truth image whereas the discriminator tries to differentiate between the ground truth and the image generated by the generator. This way both the parts are trained simultaneously. Finally, the output of the generator is the required colorized image.

Both the generator and the discriminator are Convolutional Neural Networks and follow the multilayer perceptron model. The architecture of the models are discussed below -

Generator :
- Input to the generator was required to be in 32 X 32 size. The baseline model is UNet.
- Downsampling is applied at second, third, fourth and fifth layers with stride 2 to reduce the number of computations required. The 32 X 32 size input is reduced to 512 X 2 X 2. It uses the ReLU activation function in the process.
- Upsampling with stride 2 at each layer is used to get the final colorized image. Except for the last layer, which uses the Tanh activation function, other layers use the ReLU activation function.
- Batch Normalization is used to prevent the generator from generating similar outputs for all inputs which are able to fool the discriminator.

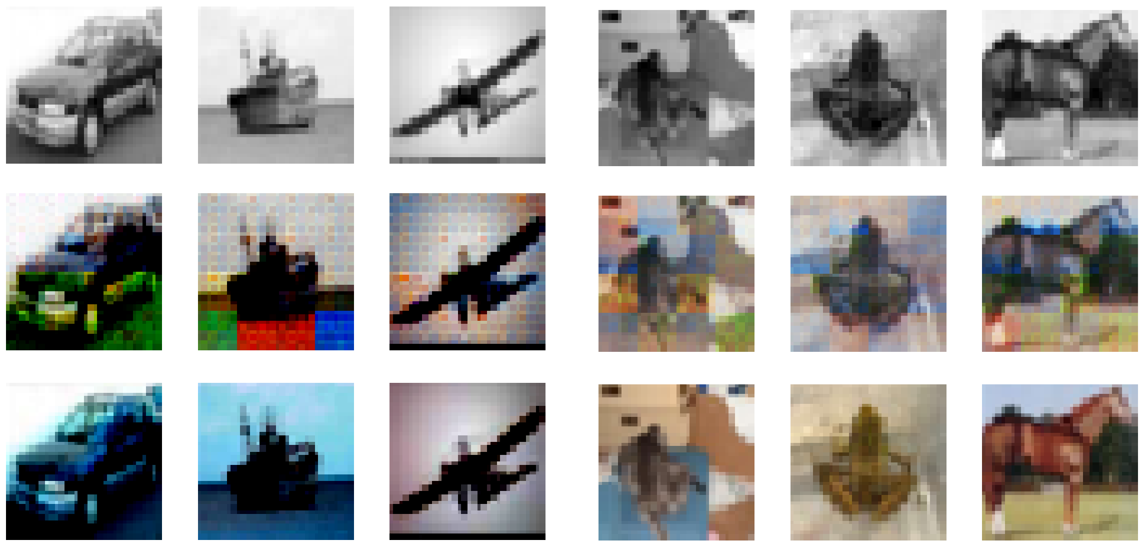
Discriminator:
- The input of the discriminator is 2 colorized images of 32 X 32 X 3 size.
- Downsampling is applied on each layer with stride 2 which finally gives the probability value of the input being real or fake. Except for the last layer, which uses the Sigmoid activation function, other layers use the ReLU activation function.

Results :
For training and testing our model, we used the “Cifar10” dataset which consists of 50000 training and 10000 testing images. But we use only 5000 training images and 1000 for testing.
Number of Epochs = 100, Batch size = 50, Number of input images fo training = 5000.
Loss(MSE) = 0.00622.



Graph: Loss vs epoch

Following is some of the output generated by the model on given test cases.



First row: Grayscale Image, Second row: our output Image, Third row: Gound Truth Image

Limitations:

- GAN colorizes images using the colors that occur more frequently in the dataset. Example - Many car images were colored red by the GAN since most of the images had red cars.

- Areas of images with a lot of features were colored green since the dataset contained images of green fields which has a lot of fluctuations in pixel intensity values.

- “Sepia Effect” was seen in some images with the clear sky where the color gradient is very wide, ranging from blue to yellow. Insufficient training could be a possible reason for this.

- Color leaks were seen in the colorized images. Better object detection is required to overcome this issue.

Suggested Improvements :

- We can add ResNet block in the generator which helps us to identify the high-level features of the image. This will help in better isolation of objects which finally leads to more accurate colorization.

- To counter color leaking we can use the EdgeNet model on Generator’s output and its output will reflect edges more clearly as done in this paper (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8451230&tag=1)

- Output can be improved by following an approach similar to variational autoencoders, allowing for image generation by sampling from a probability distribution as done in the paper (https://arxiv.org/pdf/1603.08511.pdf).

References:

1. Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2, Federico Baldassarre, Diego Gonz ́alez Mor ́ın, Lucas Rod ́es-Guirao, KTH Royal Institute of Technology.
   https://arxiv.org/pdf/1712.03400v1.pdf
2. Image Colorization using Generative Adversarial Networks Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi Faculty of Science, University of Ontario Institute of Technology 2000 Simcoe Street North, Oshawa, Ontario, Canada L1H 7K4.
   https://arxiv.org/pdf/1803.05400v5.pdf