A Comparative Study of Different Models on Image Colorization using Deep Learning

Sk. A. Sameer Janu[1], V. Lohit Ram[2], V. Naveen[3], S. Kalpana[4], Dr. D. Eswara Chaitanya[5]

[1,2,3,4] Students, Dept. of Electronics & Communication Engineering, R.V.R & J.C College of Engineering, Guntur, Andhra Pradesh, India
[5]Associate Professor, Dept. of Electronics & Communication Engineering, R.V.R & J.C College of Engineering, Guntur, Andhra Pradesh, India

---***---

Abstract - *Black-and-white image colorization by humans is a difficult and futile task. Photoshop editing has been tried, but it turns out to be delicate because it necessitates extensive research and takes up to a month to colorize an image. The work can be accomplished well by using sophisticated image colorization techniques. The literature on image colorization has drawn attention during the past ten years because it sits at the intersection of two obscure fields, deep learning, and digital image processing. This project aims to influence the advantages of transfer learning by utilizing the increasing availability of end-to-end deep learning models. Deep learning algorithms like convolutional neural networks can be used to automatically uproot image features from the training data (CNN). Deep neural networks and human involvement can both be used to achieve this. The performance of these models will be compared as image colorization is enforced in this design utilizing colorful CNN models.*

***Key words*: Deep learning, Colorization, CNNs, pre-trained model, Transfer learning.**

1. INTRODUCTION

Generally, the idea of coloring a grayscale image is a task that's simple for the natural mind, we learn from an early age to fill in missing colors in coloring books, by flashing back that lawn is green, the sky is blue with white clouds or that an apple can be red or green.

In some disciplines, automatic colorization can be truly useful indeed without the semantic understanding of the image, the simple act of adding color can increase the quantity of information that we gather from an image. For illustration, it's generally used in medical imaging to enrich visual quality when viewed by the natural eye. The majority of kits used for medical imaging captures grayscale images, and these images may contain a region that is delicate to interpret, due to the incapability of the eye of an average person to distinguish other than some shades of grayish [1].

As a computer vision task, several user-supported techniques have been proposed for colorization, be it for natural or hand-drawn images, and anticipate supplied localized color hints, or handed reference images that are semantically analogous to the target image that we can transfer color from, or indeed just keywords describing the image, to search the web for the reference images automatically.

Though, the high-ranking understanding of scene composition and object relations needed for the colorization of further complex images remains the reason developing new, completely automated results is problematic.

Newly, the approach to this task has shifted significantly from user-supported techniques to completely automated results, enforced generally by convolutional neural networks. Research in this area seeks to make automated colorization cheaper and lower time-consuming, and by that, allowing its operation on a larger scale.

With the arrival of deep convolutional neural networks, the task has been getting increased quantities of attention as a representative issue for complete visual understanding of artificial intelligence, analogous to what numerous allowed object recognition to be preliminary, since to truly convincingly colorize a target image, the technique needs to be suitable to rightly answer several subtasks analogous to segmentation, classification, and localization.

Our intention isn't necessarily to recover the actual ground trueness color, but rather to produce a probable colorization that could potentially wisecrack a natural viewer. Thus, our task becomes much further attainable to model enough of the statistical dependences between the semantics and the textures of grayscale images and their color interpretations to produce visually compelling results.

Given the lightness channel L, our system predicts the corresponding a and b color channels of the image in the CIE Lab colorspace. To break this problem, we use large-scale data. Predicting color has the nice property that training data is virtually free any color photograph can be used as a training illustration, simply by taking the image's L channel as an input and its ab channels as the administrative signal. Others have noted the easy openness of training data, and former workshops have trained convolutional neural networks (CNNs) to predict the color of the images on large datasets.

In this paper, we compare the three models which are built by CNNs. One of the models is baseline CNN, Inception-resnet-v2, and the last model is based on the Caffe framework. The first two models are based on Tensorflow and Keras frameworks. Here we are using pre-trained networks: Inception resnet v2 and the Caffe framework model.

2. RELATED WORKS

The several ways for adding chrominance values to a Grayscale image are talked over in this section. Indeed, though there are numerous ways, there's no optimum result for this problem. Some of the colorization ways are luminance keying, scribble-grounded colorization, and colorization by meager representation. The problem with these ways is that they demand considerable trouble from the user.

2.1 Luminance Keying

The previous colorization technique is called luminance keying. It was proposed in 1987 by R.C. Gonzalez [2]. Here colorization is achieved by referring to a look-up table in which each grayscale level is mapped to a specific hue, saturation, and brightness. The lookup table is defined by the user. It reduces user work but produces low-quality outputs and also it often fails to produce natural-looking color outputs. Luminance fluctuation is one of the problems which is common In old movies. However, this technique was used extensively in these times.

2.2 Colorization by Scribbling

A scribble-predicated colorization method was introduced by Levin et al[3]. This semi-automatic method requires the user to mark certain color scribbles on the input grayish image. Automatic propagation algorithms propagate these scribbled colors over the image to produce the final yield. These algorithms are grounded on the idea that neighboring pixels that have analogous intensities can be marked by analogous colors. Poor quality colorized outputs are produced with color bleeding effects at edges since the color scratches are noticeable roughly without considering boundaries and edges. Some other methods similar to scribbling were introduced by H. Noda in 2006 and D. Nie in 2007 [4] [5]. However, scribble marking remained a burden in the case of novice users. X. Ding, Y. Xu, L. Deng, and X. Yang proposed an automatic scribble-generation technique in 2012 [6] but still the color of each scribble needs to be specified manually. This technique was used by graphic artists to colorize old flicks with limited user work.

2.3 Colorization by Sparse Representation

An illustration-based colorization method was introduced by Bo Li, Fuchen Zhao, Zhuo Su, Xiangguo Liang, Yu- Kun Lai, and PaulL. Rosin [7]. The algorithm takes a color image called a reference image along with the input grayscale image. This technique operates at the superpixel position and hence it's an effective way. High, medial, and low position features are pulled from each superpixel, and a descriptor is formed by concatenating them. The feature vectors corresponding to the reference image constitute a wordbook. The intensity values of the corresponding target and reference image superpixels are rooted and compared. polychromatic values are transferred from reference to target superpixel whenever a match is set up. Eventually, an edge-conserving filter is developed for chrominance channels to conserve edges and boundaries. This system produces comparatively good results but the computational complexity is too high and also, and it's a time-consuming process hence operations are carried out at the superpixel position used by graphic artists to colorize old flicks by limited user work.

3. MODELS

3.1 Model 1

The general system channel is talked over in Fig 1[12]. Fig 2 represents the total number of trainable and non-trainable parameters. The dataset collection is the first phase which deals with collecting applicable data demanded for training purposes. Since it's veritably delicate to deal with a general dataset due to the limited memory and hardware specifications, the proposed system dataset is limited to natural face images in RGB color space. The test data set is used for assessing the effectiveness of our neural network. Several operations are performed on the training data to find the characteristics that link input to output data. The training dataset is subordinated to pre-processing and segmentation first. Pre-processing is carried off for removing noise from the raw data. Then scaling and normalization are the pre-processing ways applied to achieve uniformity over the dataset. Images in the dataset are scaled to confines of 256X256X3 during this step. The segmentation step deals with partitioning each image into different parts to pull features from it.

During the feature extraction phase, RGB values are pulled from each pixel by reducing the image size in three ways. The 256X256 image is reduced to 128X128 and lower position patterns are uprooted. To gain detailed patterns, the size is again reduced to half which is 64X64.

More complex and minute details can be uprooted by reducing the size by 32X32. After this step, each pixel intensity value is attained. Based on the uprooted features, a model is generated.

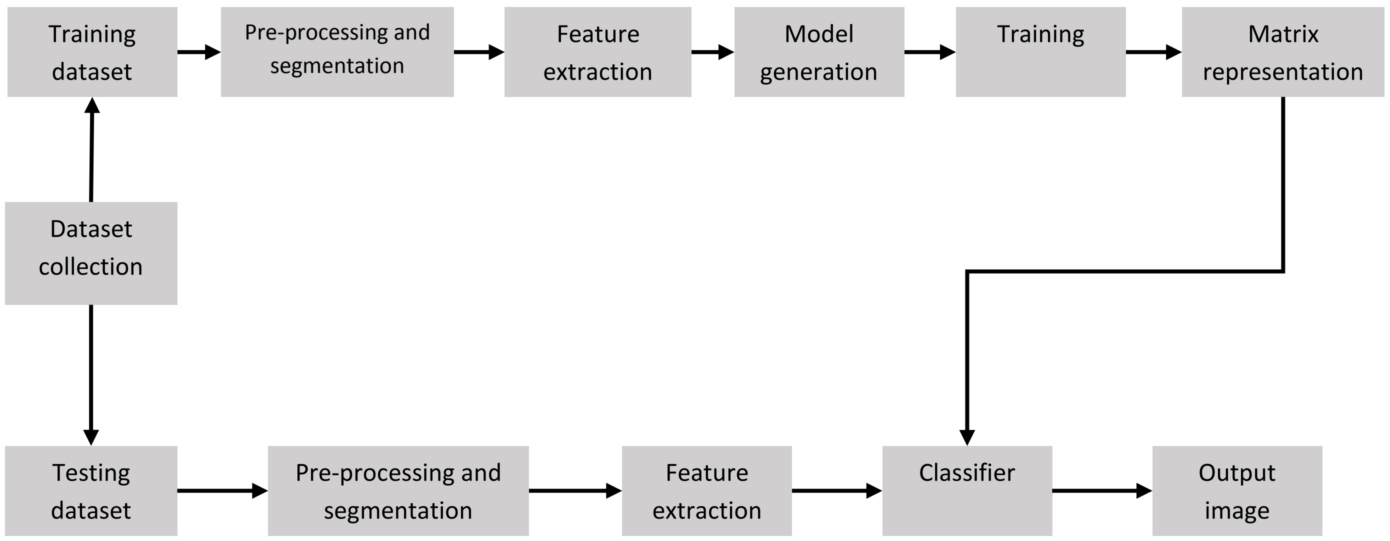
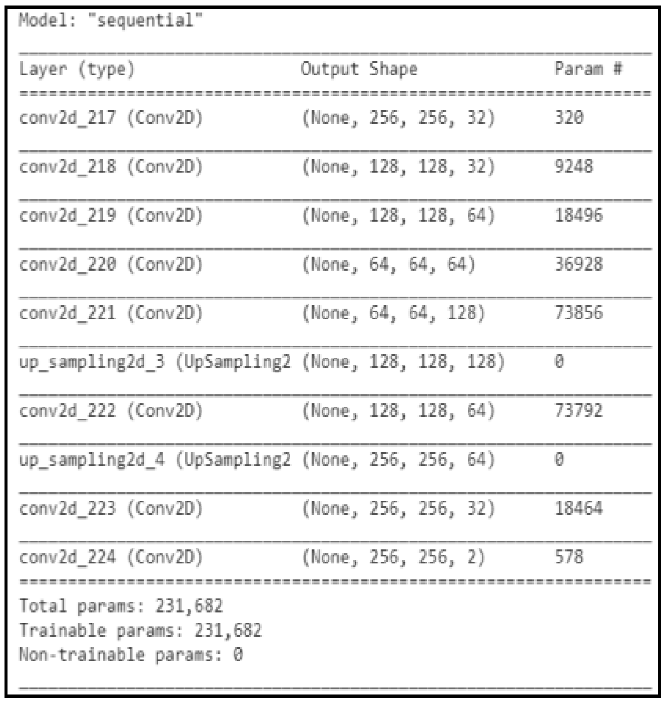


Fig 1: System Framework



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_217 (Conv2D)          (None, 256, 256, 32)      320
_________________________________________________________________
conv2d_218 (Conv2D)          (None, 128, 128, 32)      9248
_________________________________________________________________
conv2d_219 (Conv2D)          (None, 128, 128, 64)      18496
_________________________________________________________________
conv2d_220 (Conv2D)          (None, 64, 64, 64)        36928
_________________________________________________________________
conv2d_221 (Conv2D)          (None, 64, 64, 128)       73856
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 128, 128, 128)     0
_________________________________________________________________
conv2d_222 (Conv2D)          (None, 128, 128, 64)      73792
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_223 (Conv2D)          (None, 256, 256, 32)      18464
_________________________________________________________________
conv2d_224 (Conv2D)          (None, 256, 256, 2)       578
=================================================================
Total params: 231,682
Trainable params: 231,682
Non-trainable params: 0
_________________________________________________________________
```

Fig 2: Custom CNN Architecture

This model may have errors and therefore cannot be finalized as an effective model. To make it accurate and effective by avoiding errors, it passes through the training phase. The training includes a backpropagation technique which works in a trial-and-error manner. The errors are reduced and a more effective learned network is attained during this phase. It has mainly three layers. The input layer, several hidden layers, and the last is the output layer. The trained model is given away to a neural network classifier to achieve the colorization of new inputs in the future. The classifier identifies the orders of inputs grounded on the Information from the trained data. Logistic regression, Bayesian classifier, and SVM (Support Vector Machine) are illustrations of classifiers. After all these phases, the neural network can accept new inputs and produce colorized output images.

3.2 Model 2

This model owes to [8]. Given the luminance element of an image, the model estimates its a * b * factors and combines them with the input to gain the final estimate of the colored image. Rather than training an attribute extraction branch from scratch, we make use of an Inception- ResNetv2 network shown in Fig 3 (appertained to as Inception henceforth) and get back an embedding of the gray-scale image from its last layer. Here we make use of transfer learning which contains mainly two phases. One is to extract the features and second is the to fine-tune of model parameters of the pre-trained model. The network is logically divided into four main factors. The encoding and the attribute extraction components gain mid and high-position features independently which are also intermingled in the fusion layer. Eventually, the decoder uses these features to estimate the output.

Preprocessing: To assure correct learning, the pixel values of all three image elements are centered and scaled to gain values within the interval of (-1, 1).

Encoder: The Encoder processes H × W gray-scale images and outputs an H/ 8 × W/ 8 × 512 attribute representation. To this end, it uses 8 convolutional layers with 3 × 3 kernels. Padding is used to save the layer's input size. Likewise, the first, third, and fifth layers apply a stride of 2, consequentially halving the dimension of their output and hence reducing the number of calculations needed.

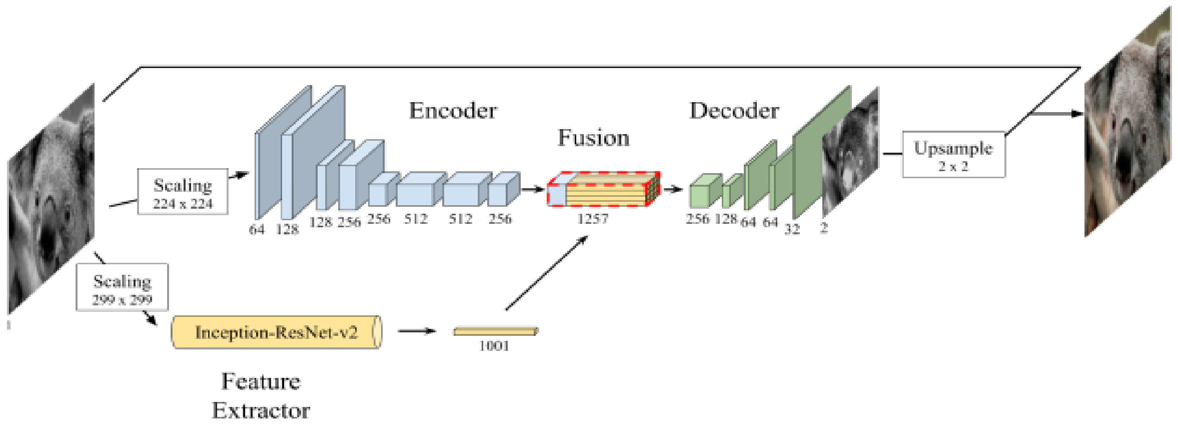


Fig 3: Overview of model architecture

Feature Extractor: High-level features "underwater" or "inner scene", convey image information that can be used in the colorization process. To pull an image embedding we used a pre-trained Inception model. First, we scale the input image to 299 × 299. Next, we pile the image with itself to gain a three-channel image to satisfy Inception's dimension conditions. Next, we feed the performing image to the network and pull the output of the last layer before the softmax function. This results in a 1001 × 1 × 1 embedding.

Fusion: The fusion layer takes the attribute vector from Inception, replicates it HW/ 8^2 times, and attaches it to the feature volume outputted by the encoder along the depth axis. This approach obtains a single volume with the decoded image and the mid-level features of shape H/ 8 × H/ 8 × 1257. By mirroring the point vector and compounding it several times we assure that the semantic data given by the feature vector is slightly assigned among all spatial regions of the image. also, this result is robust to arbitrary input image sizes, adding the model adaptability. Eventually, we apply 256 convolutional kernels of size 1 × 1, eventually generating a feature volume of dimension H/ 8 × W/ 8 × 256.

Decoder: Eventually, the decoder takes this H/ 8 × W/ 8 × 256 volume and applies a series of convolutional and up-sampling layers to gain a last layer with dimension H × W × 2. Up-sample is performed using the elemental nearest neighbor approach so that the output's height and width are double the inputs.

3.3 Model 3

As in this technique, we're using a new framework called Caffe [10]. Caffe is a deep learning framework that has been used considerably in computer vision tasks, including image colorization. One popular Caffe model used for image colorization is grounded on the VGG16 architecture. The architecture is shown in Fig 4.

The VGG16 armature was first developed for object recognition and classification tasks, but it has been acclimated for image colorization by removing the pooling layers that are generally used in the architecture. The disposition of the pooling layers is because pooling layers reduce the resolution of the feature maps, which can lead to information loss and poorer colorization results. By keeping the original resolution of the feature maps, the model is better suitable to capture fine-granulated details and textures in the image.

The architecture of the VGG16 model consists of 13 convolutional layers, followed by three completely connected layers. Each convolutional layer is followed by a rectified linear unit (ReLU) activation function, which introduces non-linearity into the model and helps it learn more complex representations of the image features. The completely connected layers are used to produce the final output of the model. To adjust the VGG16 architecture for image colorization, the model is trained using a process called" regression" in which the model takes the grayscale input image and produces a color output image as the prediction. The model is trained on a large dataset of color images and their corresponding grayscale performances, learning to associate different grayscale values with specific color values. The execution of the VGG16 model for image colorization involves several ways. First, the model structure is defined in Caffe, specifying the layers and their parameters. Next, the model is trained on a large dataset of color images and their corresponding grayscale performances, using an optimization algorithm to edit the model parameters and minimize the difference between the predicted colorization and the ground truth color image. During training, the model is estimated on a confirmation set to cover its performance and help to overfit. Once training is complete, the model can be used to colorize new grayscale images.

This involves transferring the grayscale image through the trained model to produce a colorized output image. Overall,

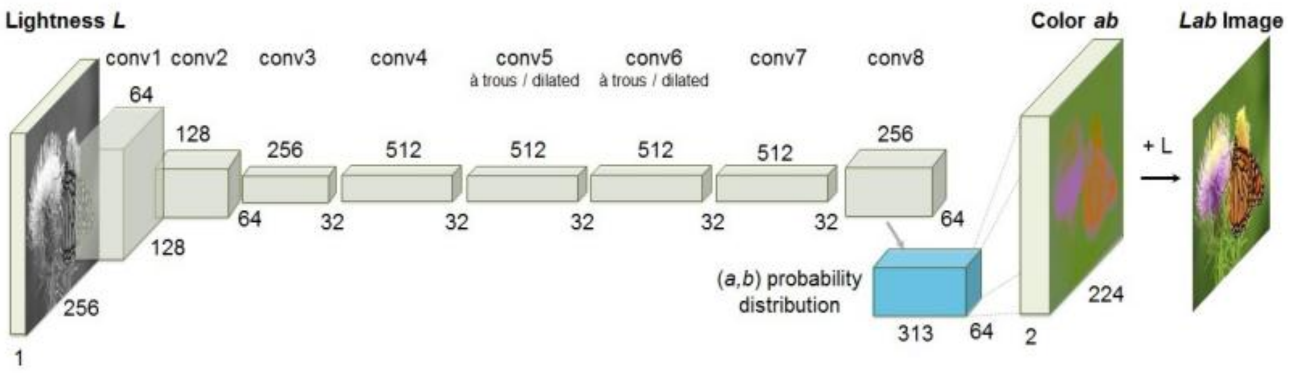


Fig 4: Network architecture

the VGG16 structure without pooling layers is an important tool for image colorization using deep learning ways. Its capability to capture fine-granulated details and textures in the image, along with its adaptability and flexibility, make it a precious tool for a range of computer vision tasks. The execution of the model in Caffe is straightforward, but it requires a large dataset and significant computational resources for training and optimization.

4. EXPERIMENTAL RESULTS

4.1 Model 1

Emil Wallner's public dataset from Floydhub was used to train the neural network. Images of dimension 256X256X3 are used in the training data Wallner's training dataset consists of 200 images of natural faces and the test consists of 50 images with a dimension of 256X256X1. Fig 5 shows the selected images in the testing datasets. After training and learning, the neural network will find traits that link each grayscale value to corresponding color values. The figure shows the sample images in the test dataset. During the testing aspect, some test inputs are fed to the neural network to check the performance. The testing inputs also pass through the preprocessing, segmentation, and feature extraction steps. The input and output images are shown in Fig 6. Good quality results can be attained within a reasonable time without any natural help.

Fig 5: Model-1 Test Images

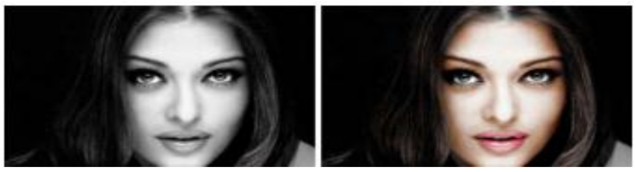


Fig 6: Model-1 Input and Output Images

4.2 Model 2

Once trained, we fed our network with some images. The results turned out to be relatively reasonable for some of the images, generating - photorealistic pictures. Still, due to the small size of our training set our network performs better when certain image features appear. For case, natural essentials like the ocean or greenery seem to be well honored. Still, specific objects aren't always well-colored. Fig 7 illustrates results for some cases where our network produces alternative-colored estimates. In the first row, our approach is able of honoring the green vegetation. Still, the butterfly wasn't colored. Likewise, in the illustration in the alternate row, we observe that the network changes the color of the rowers' clothes from green-yellow to red-blue.

The last row shows a geography illustration where our model handed a photo-realistic image.

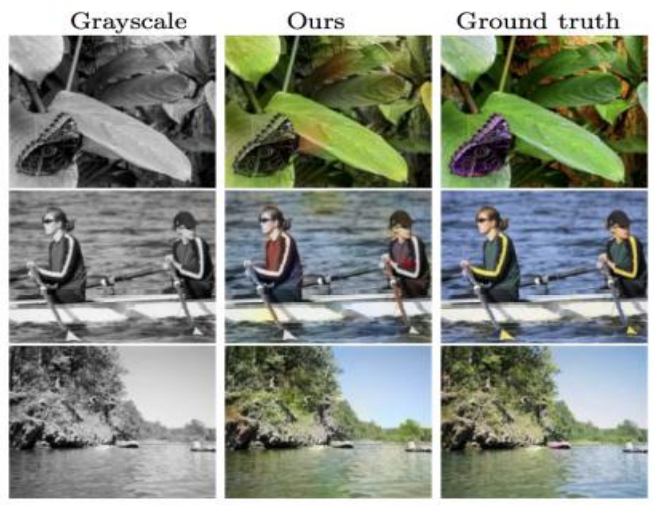


Fig 7: Model-2 Input, Output, and Ground truth images

4.3 Model 3

We train this network on the 1.3M images from the ImageNet training set [9], validate on the first 10k images in the ImageNet validation set, and test on separate 10k images in the validation set.

In this method [10], with classification loss and class rebalancing the network was trained from scratch with k-means initialization, using the ADAM solver for approximately 450k iterations. The input, output, and ground-truth images can be seen in Fig 8.

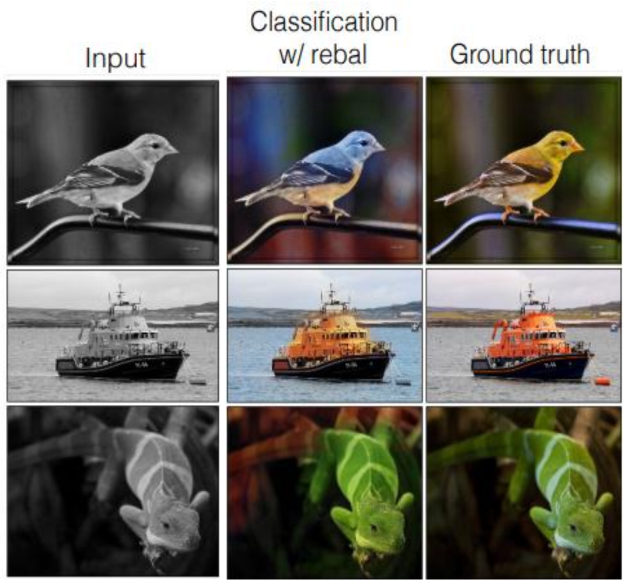


Fig 8: Model-3 Input, Output with color rebalancing and groundtruth images

5. CONCLUSION

Image colorization is a challenging computer vision task that has received considerable attention in recent years. Many deep learning models have been proposed to address this problem, each with its strengths and weaknesses. In this comparison, we will examine three popular models for image colorization: Baseline CNN, Inception-ResNet v2, and the Caffe model.

Baseline CNN is a simple convolutional neural network that is commonly used as a starting point for image colorization. It typically consists of several convolutional layers followed by one or more fully connected layers. The architecture can be adjusted based on the specific requirements of the task, such as the size of the input images and the number of color channels.

Inception-ResNet v2 is a more advanced model that combines the Inception architecture with residual connections. The Inception architecture is known for its ability to extract features at multiple scales, while residual connections help to prevent vanishing gradients during training. This model is very deep and complex, with over 100 layers, which allows it to capture highly detailed features in the image.

The Caffe model is based on the VGG16 architecture but without any pooling layers. It is a regression model that takes a grayscale image as input and produces a colorized output image as the prediction. The model is trained on a large dataset of color images and their corresponding grayscale versions, learning to associate different grayscale values with specific color values.This model tends to have a MSE of 0.0039,PSNR of 24.761db,AIT of 0.58s.This values obtained are more than the two models except AIT.But we can't relied on this.Because our objective is to produce plausible colors which looks like a real image.

To compare these models, we can consider several factors, including their accuracy, speed, and memory usage. In terms of accuracy, all three models have shown impressive results in image colorization tasks, with each model able to produce realistic and vibrant colorizations. However, Inception-ResNet v2 tends to outperform the other models in terms of accuracy, thanks to its ability to capture highly detailed features in the image.

In terms of speed, Baseline CNN is the fastest model of the three due to its simpler architecture. However, Inception-ResNet v2 and the Caffe model are also able to produce colorizations relatively quickly, and their speed is generally considered acceptable for most applications.

In terms of memory usage, the Caffe model tends to require the most memory due to its large dataset and complex architecture. However, this is not always a significant issue, as modern hardware can often handle the memory requirements of deep learning models.

Overall, it is difficult to declare one of these models as definitively better than the others, as each has its strengths and weaknesses. Baseline CNN is a good starting point for image colorization tasks and can produce good results with less computational complexity. Inception-ResNet v2 offers high accuracy and detail, making it a good choice for applications that require the most precise colorizations. The Caffe model is a powerful tool for image colorization, particularly in cases where detailed color distributions are critical.

In addition to these models, many other deep-learning models have been proposed for image colorization. For example, GAN-based models such as Pix2Pix and CycleGAN have shown promising results, while other models such as U-Net and MobileNet have also been used for image colorization. The performance of different models of pix2pix is shown in terms of MSE, PSNR, and average inference time[11] in table 1. The choice of the best model ultimately depends on the specific requirements of the task, such as the available resources, the desired accuracy, and the application context.

In conclusion, image colorization is a complex task that requires a deep understanding of the image data and the best approach to extract the most relevant features. Baseline CNN, Inception-ResNet v2, and the Caffe model are three popular models for image colorization that have their strengths and weaknesses. Other models like GAN-based models, U-Net, and MobileNet have also shown promise in this area. The choice of the best model depends on the specific needs of the task at hand.

Table 1: Performance of different Models in terms of MSE, PSNR,and Average inference time(AIT).

Model	MSE	PSNR (in dB)	Average inference time (in ms)
Baseline CNN	0.0120	26.443	~56
Inception-resnetv2 based CNN	0.0121	26.630	~159
Pix2pix-Mobilenetv2 (LAB)	0.0107	26.870	~61
Pix2pix-Densenet121 (LAB)	0.0108	26.872	~117
Pix2pix-Mobilenetv2 (RGB)	0.0289	22.59	~53
Pix2pix-Densenet121 (RGB)	0.0237	23.725	~103

REFERENCES

[1] V. Bochko, P. V¨alisuc, T. Alho, S. Sutlnen, J. Parkkinen, and J. Alander, “Medical image colorization using learning”, pp. 70–74, Jan. 2010.

[2] R.C. Gonzalez, R.E. Woods, “Digital Image Processing” second ed., Addison–Wesley Publishing, Reading, MA, 1987.

[3] A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” ACM Transactions on Graphics, vol. 23, no. 3, pp. 689– 694, 2004.

[4] H. Noda, M. Niimi, and J. Korekuni, “Simple and efficient colorization in YCbCr color space,” in International Conference on Pattern Recognition, 2006, pp. 685–688.

[5] D. Nie, Q. Ma, L. Ma, and S. Xiao, “Optimization-based grayscale image colorization,” Pattern Recognition Letters, vol. 28, pp. 1445– 1451, 2007.

[6] X. Ding, Y. Xu, L. Deng, and X. Yang “Colorization Using Quaternion Algebra with Automatic Scribble Generation” Springer-Verlag Berlin Heidelberg 2012.

[7] Bo Li, Fuchen Zhao, Zhuo Su, Xiangguo Liang, Yu-Kun Lai, Paul L. Rosin “Example-based Image Colorization using Locality Consistent Sparse Representation” Journal of latex class files, Vol. 13, NO. 9, September 2014.

[8] Iizuka, S., Simo-Serra, E., Ishikawa, H.: Let there be color! : joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics (TOG) 35(4) (2016) 110.

[9] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115(3) (2015) 211–252.

[10] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in ECCV, 2016.

[11] Abhishek Kumbhar, Sagar Gowda, Ruchir Attri, Anjaneya Ketkar, Prof. Ankit Khivasara, "Colorization of Black and White Images Using Deep Learning" in IRJET vol 08., Issue: Oct 2021.

[12] S. Titus and J. R. N.M., "Fast Colorization of Grayscale Images by Convolutional Neural Network," *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, Kottayam, India, 2018, pp. 1-5, doi: 10.1109/ICCSDET.2018.8821180.