International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 10 | Oct 2021 www.irjet.net p-ISSN: 2395-0072

Colorization of Black and White Images Using Deep Learning

Abhishek Kumbhar[1], Sagar Gowda[2], Ruchir Attri[3], Anjaneya Ketkar[4], Prof. Ankit Khivasara[5]

[1,2,3,4]Student, Department of Electronics and Telecommunication Engineering, K.J. Somaiya College of Engineering, Mumbai, Maharashtra, India
[5]Professor, Department of Electronics and Telecommunication Engineering, K.J. Somaiya College of Engineering, Mumbai, Maharashtra, India

---

Abstract - *Manual colorization of black and white images is a laborious task and inefficient. It has been attempted using Photoshop editing, but it proves to be difficult as it requires extensive research and a picture can take up to one month to colorize. A pragmatic approach to the task is to implement sophisticated image colorization techniques. The literature on image colorization has been an area of interest in the last decade, as it stands at the confluence of two arcane disciplines, digital image processing and deep learning. Efforts have been made to use the ever-increasing accessibility of end-to-end deep learning models and leverage the benefits of transfer learning. Image features can be automatically extracted from the training data using deep learning models such as Convolutional Neural Networks (CNN). This can be expedited by human intervention and by using recently developed Generative Adversarial Networks (GAN). We implement image colorization using various CNN and GAN models while leveraging pre-trained models for better feature extraction and compare the performance of these models.*

Key Words: **Deep learning, Pre-trained model, CNN, GAN, image colorization, Pix2pix**

1.INTRODUCTION

In old times when photography was starting out, most images were in black and white(B&W). Hence efforts to colorize old B&W images started taking place to give the image a different perspective and a beautiful insight into the captured moments. Colorization efforts started to appear in the early 1900's, using paints and brushes. This painstaking process took days, sometimes weeks to generate a rough recreation of reality.

The trend has shifted to use of computer software such as Adobe Photoshop, GIMP etc but the procedure remains the same. The present-day techniques for manual colorization of these B&W images are: Cleaning the Image, Adjusting the image tones and contrast, converting the image to CMYK and finally adding solid color to specific entities in the image. But even digital colorization using softwares like photoshop, that helped in colorization, pixel by pixel, along with improvements handling color bleeding and color continuity, and reduced manual work to some extent, now seems to be a tedious task.

In the present times, a massive gallery of photographs is available, thanks to the color cameras. It offers an abundance of information to determine the color schemes of common objects, scenes, lighting conditions and color intensities. With advancement in technology and extensive research in the field of deep learning, models can be trained to learn the knowledge and then apply it to colorize old photographs. Using such techniques to colorize the photographs can modernize and automate the way how things are done and relieve the pressure on the colorizing artists to some extent.

We implemented different CNN and GAN models used for image colorizing and provided their quantitative and qualitative comparison. We also show that incorporating pre-trained models can improve the performance greatly while making the training and hyperparameter tuning process less cumbersome. We created a custom dataset of high-resolution images; categorized according to scenery, background and artistic theme since the colors involved in such images are generic and are not complex.

2. RELATED WORK

Iizuka et al. [1] In this paper he combined two networks, one to predict the global features of the input image and the other to specialize in local features of input images. The global features network is trained for image classification and directly concatenated to the local features network which are then trained for colorization of images using L2 Euclidean loss function.

Richard Zhang [2] In this paper he introduced an optimized solution by taking a huge data-set and single feed-forward pass in CNN. They used a custom multinomial cross entropy loss with class rebalancing and by using humans as subjects they were able to fool 32% of them by their results. They used prior color distribution obtained from the training set to predict a distribution for each output pixel.

Baldassarre et al. [3] In this paper he made a network model that combines a deep CNN architecture, that is trained from scratch, with a pre-trained model Inception-Resnetv2 for high level feature extraction. They train this network on a small subset of 60,000 images from ImageNet. This architecture is similar to that used by Iizuka et al. [1] and it also uses Euclidean (L2) loss function.

In [4] deep convolutional neural network architectures used are inherited from the VGG16 network. They implemented two models: one as a regression model and other as a classification model. They use the CIE LUV colorspace for input and output. They posed it as a classification task that can produce colorized images which are much better than those generated by a regression-based model.

David Futschik[5] In this paper he made use of several variants of CNNs and compared their performance on the data, using two different NN architectures; one traditional, plain CNN, and other being inspired by residual CNNs, which had not been used for colorization previously. Despite the smaller fewer parameters, this model was able to generate results that surpass plain CNN in generalization to unseen/test data.

In [6], automatic image colorization with two different CNN models is proposed. They train a classification and regression model on CIFAR-10 dataset using Lab colorspace. They train the classification model from scratch and also by transfer learning from a pretrained VGG16 network. They also use the Annealed-mean technique with the model to map prediction distribution to single output prediction and show that it can produce vibrant and spatially more consistent results.

Larsson et al. [7] In this paper They train their model on ImageNet dataset to predict per-pixel color histograms and made use of convolutional layers from VGG16 network layer to predict pixels' values, which are pre trained on the image classification task and fine-tuned for colorization.

Phillip Isola et al. [8] In this paper they defined a conditional GAN for image-to-image translational problem by putting a condition on the GAN to produce corresponding output data. This network learns a mapping between the input image and the output image and by using loss function during the training procedure of the mapping, great results were achieved. In this paper the generator model used “U-net" architecture and convolutional "PatchGAN" classifier for discriminator model.

In this paper [9], In this paper they used GANs to train on CIFAR-10 and Places365 dataset and its results were compared with those obtained using existing CNN. They used L*a*b color space instead of RGB depicting the brightness of the image L and the color information are fully encoded in the remaining two channels a & b. This model generated great results with the CIFAR-10 dataset rather than U-net CNN, but moderate results were obtained with Places365 dataset compared to U-Net results that had the “Sepia effect” (a shade of brown in the foreground) due to the L2 regularization that caused a blurring in U-Net results.

3. METHODOLOGY

3.1 Brief Approach

Our goal is to take an input grayscale image, a single channel of image data, and transform it into a standard RGB image, an image with three channels of data. The CIE Lab colorspace is used to represent the input and output images of the model, since it separates the lightness(intensity) and color components of an image.

We train CNN models to map the input grayscale image to colorized Lab space image, which is then converted to RGB image. Specifically, given an input L channel(grayscale) image $X \in R^{H \times W \times 1}$, the objective is to learn a mapping $\hat{Y} = F(X)$ to the two associated color channels (a, b channels) $Y \in R^{H \times W \times 2}$, where H, W are image dimensions. The predicted color channels a and b are then combined with the input L channel image to give the predicted colorized Lab space image. We use the Euclidean loss L2 function, also known as mean squared error loss, between predicted and ground truth Lab images as the objective function. The architecture of the CNN models used is described in section 3.2.

For GAN, the generator network maps the input L channel(grayscale) image to a, b channels just like for the CNN model explained above. The discriminator network takes in these generated images as well as real images and detects whether they are real or fake (generated). Both these networks are trained simultaneously until the generator produces images that are close enough to real data to fool the discriminator.

3.2 Architecture

3.2.1 Baseline CNN Model:

```
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 1: Custom CNN Architecture

The input CIE Lab color space is split into two components, Lightness and a, b color values. The lightness(L) channel is fed into the Encoder which contains convolutional layers that extract features while downsampling to reduce the amount of computation. We get a 128x128x128 feature representation as output. The decoder part takes this as input and applies upsampling and convolutional layers again to finally output 256 x 256 x2 features which are the *a* and *b* channels. To map the output values between -1 and 1, we use the *tanh* activation function and the ground truth values lie in range -128 to 127 which are also normalised to -1 to 1. The ground truth is compared to the output and the MSE loss function updates the parameters.

3.2.2 Inception-resnetv2 based CNN:

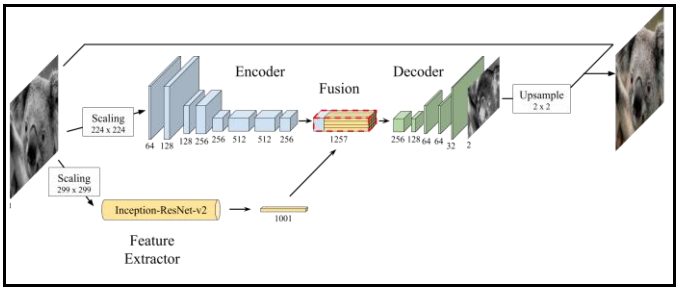

Fig 2: Inception-resnetv2 based Architecture from [3]

We implement this model exactly as defined in the original paper [3].

The **Encoder** processes H x W gray-scale images, learns features using convolutional layers while downsampling and outputs a H/8 x W/8 x 512 feature representation. This part is exactly like our previous CNN model.

The **Inception-Resnetv2** model is a pre-trained model trained on the large ImageNet dataset. It is used here as a high-level feature extractor. We feed the input grayscale image and extract the output of the last layer before the softmax function which results in a 1001x1x1 embedding.

The **fusion** layer this embedding, replicates it H/8 x W/8 times and combines it with the feature representation obtained from the encoder. Then we apply 256 convolutional kernels of size 1x1, ultimately generating a feature volume of dimension H/8 x W/8 x 256.

Finally, the **decoder** takes this H/8 x W/8 x 256 volume and applies a series of convolutional and up-sampling layers in order to obtain a final layer with dimension HxWx2. Up-sampling is performed using the basic nearest neighbor approach so that the output's height and width are twice that of input. To map the output values between -1 and 1, we use the *tanh* activation function and the ground truth values lie in range -128 to 127 which are also normalized to -1 to 1. The loss function used here is also Mean squared error (MSE) loss.

3.2.3 Pix2pix GAN:

Pix2pix GAN model is a type of conditional-GAN that maps an input image and noise to output image, unlike traditional GANs that map an input noise vector to output. The generator network is similar to that used in [8] and [9], it is a modified U-net, an encoder-decoder architecture with skip-connections. Further we leverage the power of transfer learning by replacing the encoder part with pretrained MobilenetV2 and Densenet121 models. The number of upsampling and downsampling layers is reduced from 8 in [8] to 5, which reduces the number of parameters, memory and computation time.

The lightness(L) channel is fed into the pretrained encoder which outputs features of various sizes that are later concatenated with corresponding upsampled layers. The decoder part takes the output features from the last layer as input. The upsampling blocks each containing a transposed convolution layer, batch normalization, dropout & relu activation layers, are applied while concatenating correspondingly sized features from encoder using skip connections to finally output 256 x 256 x2 features which are the *a* and *b* channels. To map the output values between -1 and 1, we use the *tanh* activation function and the ground truth values lie in range -128 to 127 which are also normalised to -1 to 1.

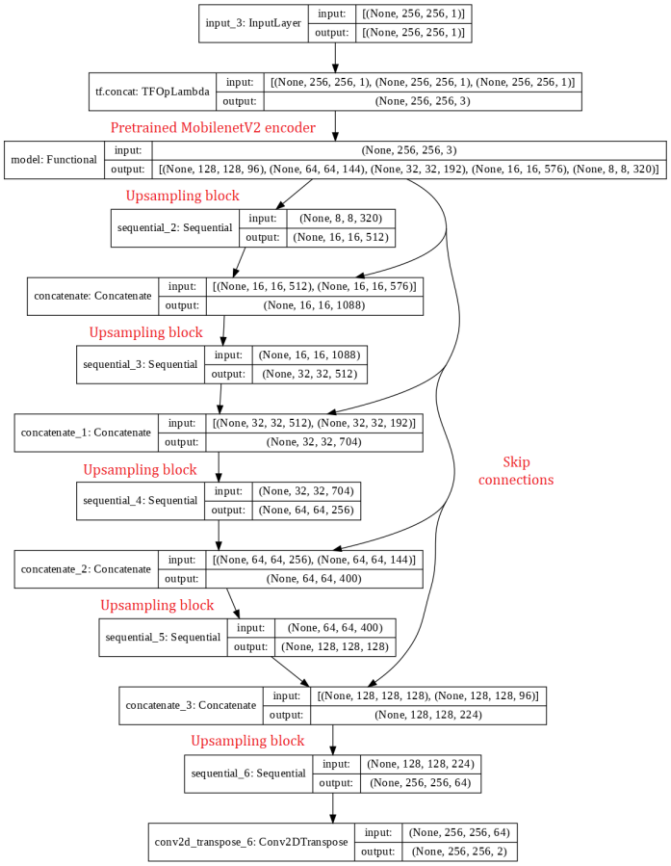

Fig 3: Pretrained MobilenetV2 based generator

The discriminator network is adopted from the original Pix2pix paper [8]. The downsampling block contains convolution, batch normalization and leaky relu activation layers.

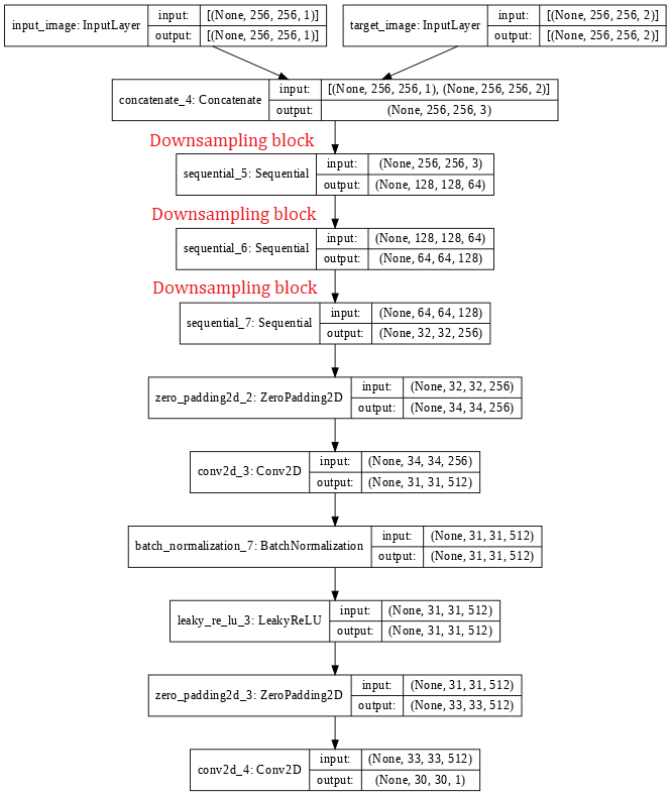

Fig 4: Discriminator network, adopted from the original Pix2pix paper [8]

The loss function for a conditional-GAN is as follows:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],$$

The generator tries to minimize this loss while discriminator tries maximize it and so is also known as min-max loss. It is essentially modified binary cross entropy loss.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

Additionally, the generator also minimises the L1-loss or Mean Absolute Error(MAE) as shown above, so that it generates images that are similar to the real target (color) images.

The final loss function for Pix2pix GAN is a weighted sum of both these losses with λ = 100, as given in [8].

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

3.3 Dataset & Preprocessing:

The dataset was created by collecting nature and landscape images from different stock photo sites. The images were mostly restricted to scenes like beaches, forests, fields, lakes, waterfalls and mountains, but some images from others scenes are also included. Learning color patterns for a wide variety of scenes requires large datasets, in order of millions, and large models, training of which requires higher computational power. A smaller dataset with images from fewer scenes allowed fast experimentation and lesser computational power. The images were filtered to remove grayscale(B/W) images, too faint images, too vibrant with artificial colors, and images with too many details as all these could negatively affect the training process making convergence difficult. These simplifications were performed to make the process of experimentation and learning patterns on small dataset easier.

After this process, the final dataset is composed of 7200 RGB images of varying sizes. The dataset was split into a training set(90%) and test set(10%).

During pre-processing, the images are first resized to 256x256 and normalized. Then it is converted from RGB colorspace to *Lab* colorspace. The 'L' channel ranges in values from 0 to 100 and 'a,b' channels from -128 to 127, so they are normalized before training. The 'L' channel is the input of the model and 'a,b' channels are the target output.

3.4 Training details

All the models were trained on 90% of the dataset and tested on 10% of the dataset. We used the TensorFlow framework for implementing deep learning models and Google Colab platform for training and experimentation

For CNN models, we used the Adam optimizer with an initial learning rate = 0.001, beta1 = 0.9, beta2 = 0.999 and epsilon = 1e-8. We trained both the models for 150 epochs and after 100 epochs, the learning rate was reduced to 1e-4. A batch size of 32 was used. These models are trained only in Lab colorspace.

For Pix2pix models, we used the Adam optimizer with an initial learning rate = 0.0002, beta1 = 0.5, beta2 = 0.999 and epsilon = 1e-8 for generator as well as discriminator network. We trained both the models for 100 epochs with a batch size of 16. We also train these models in both RGB and Lab colorspace, i.e., mapping L-channel to a, b channels as well as mapping grayscale to RGB channels

4. RESULTS AND DISCUSSION

For quantitative results, we used Mean Squared Error (MSE) to measure the error between colorized image and ground truth image and Peak-Signal-to-Noise-Ratio (PSNR) to measure the output image quality on the test set (10% of total).

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

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

The Pix2pix models trained in LAB colorspace give the lowest MSE and highest PSNR values. Both the mobilenetv2 and densenet121 perform equally well except the former has inference time half of that of the latter. CNN based models have next best performance, both baseline and inception-resnetv2 based performing nearly the same on the metrics but inference time of inception-resnetv2 is 3x the baseline model. The worst performance is shown by Pix2pix models trained in RGB colorspace with high MSE and low PSNR values.

However, both MSE and PSNR are basically metrics for image restoration tasks and hence cannot be relied on completely. Image colorization differs from such tasks as the objective is not exactly restore colors but produce plausible colors which may look realistic. As a result, a visually appealing output image with realistic colors may perform worse on these metrics if the colors are not exactly the same or similar to the target image. Figures 5 and 6 will demonstrate this fact.

In Fig 5, it can be clearly seen that the images are simpler to colorize as there aren't many plausible colors. A grass field can be yellow-green, forests are usually green and skies blue and beaches are also typically blue. Consequently, low MSE and high PSNR values are obtained in all of these images, thus both metrics are representative of the performance.

In Fig 6, the first image contains many minor details with different colors which makes colorization difficult, but still the model manages to produce acceptable results. In the 2nd image too, the model produces plausible colors which do not

Fig 5: Metric results of colorized images example 1

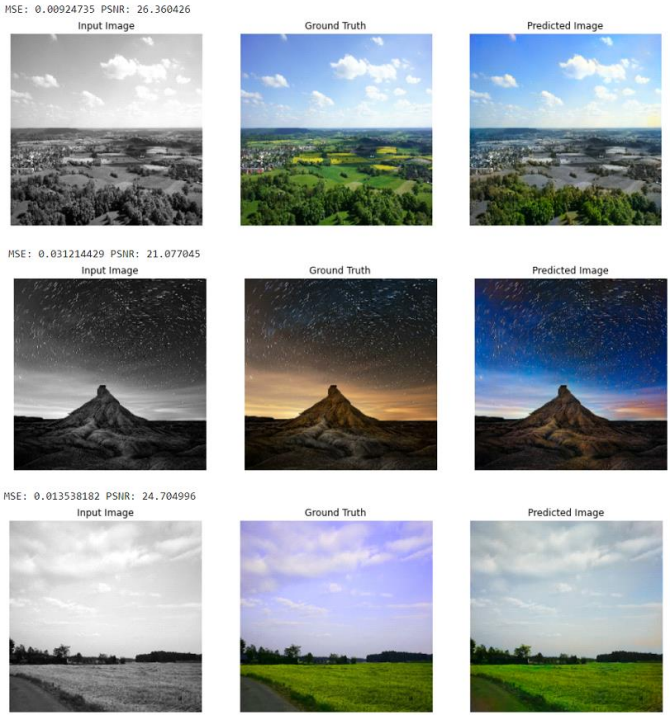

Fig 6: Metric results of colorized images example 2

feel artificial. For the last image, the output seems very natural as sky color can vary from blue to purple. Yet in all these cases, higher MSE values as well as lower PSNR values are obtained thereby proving these metrics less useful. So, it is necessary to couple quantitative results with qualitative evaluation to thoroughly assess the performance.

For qualitative results, we visually inspected some of the examples from the test set as quantitative error is based on

comparison with ground truth. And we do not expect the models to exactly reproduce the colors but rather predict plausible colors that look natural. Figures 7 and 8 depict the colorized outputs of our four models along with the grayscale images and ground truth images on the left and right ends, respectively. Note that Pix2pix models trained in RGB colorspace are not included in these due some issues which are discussed later.

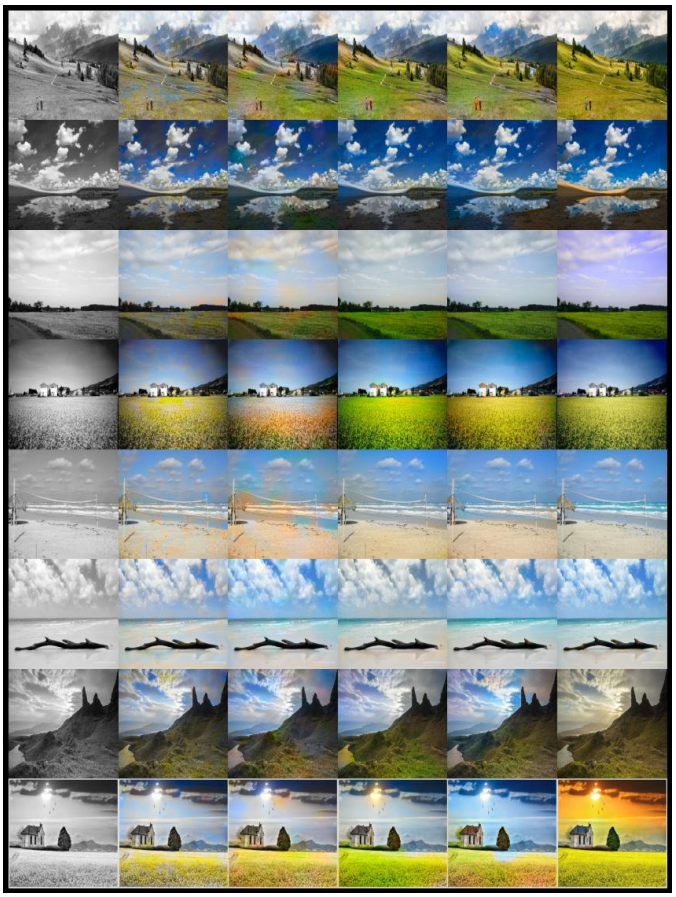

Fig 7: Colorized output of the four models on test set examples-1. Order: (Left)Grayscale, Our Custom CNN, Inception- Resnetv2 based, Pix2pix with MobilenetV2, Pix2pix with Densenet121, Ground Truth (Right)

In figure 7, we see that both the Pix2Pix models worked well on green landscapes and blue skies. Our custom CNN model produced more realistic colors and lesser random patches, than the Inception-resnetv2 based model. The Inception-resnetv2 based model showed inconsistent color patterns. Pix2Pix with Densenet121 showed better prediction of brightness and contrast, as compared to other models.

In figure 8, we see that our Pix2pix with MobilenetV2 model colorized the blue skies, clouds and white sand quite accurately though in the third row we do have some random color patches. The inception-resnetv2 based model performed poorly with many yellow-orange patches. Pix2Pix with Densenet121 performed the best on blue skies and white clouds.

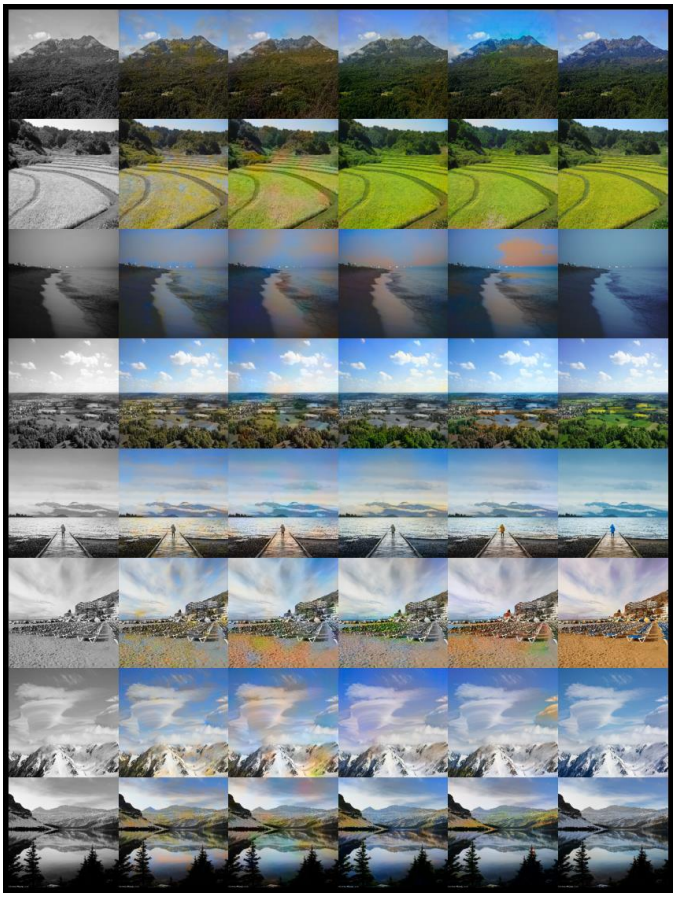

Fig 8: Colorized output of the four models on test set examples-2. Order: (Left)Grayscale, Our Custom CNN, Inception- Resnetv2 based, Pix2pix with MobilenetV2, Pix2pix with Densenet121, Ground Truth (Right)

The inception-resnetv2 based model produces images with random color patches most of the time and sometimes with inconsistent colors. This may be due to the fact that it is a much larger and complex model (about 6.5M parameters) compared to our light custom CNN model with lesser parameters (approx. 231k). So requires much more data to learn patterns.

On the other hand, the Pix2pix GAN models clearly outperform CNN models on all images with reduced inconsistencies, negligible random patches, more vibrant colors and close to actual targets. Pix2pix-densenet121 model's performance is slightly better than mobilenetv2. Overall, the colors tend towards green and blue more as majority landscape images contain blue sky, green trees, beaches, etc. which lie in green – blue color range. Most of the models fail in the orange-brown color range due underrepresentation of images in that range. So, choice of dataset affects the model output as is the case with most deep learning models.

We also experimented with training Pix2pix models in RGB colorspace, i.e., mapping grayscale to R, G, B channels. Compared to those trained in LAB colorspace, these required extensive tuning of hyperparameters like batch size and

learning rate to even achieve satisfactory results. Also, these models required twice the time of Pix2pix LAB colorspace models to achieve similar quality of outputs.

Pix2pix-mobilenetv2-RGB Pix2pix-Densenet121-RGB

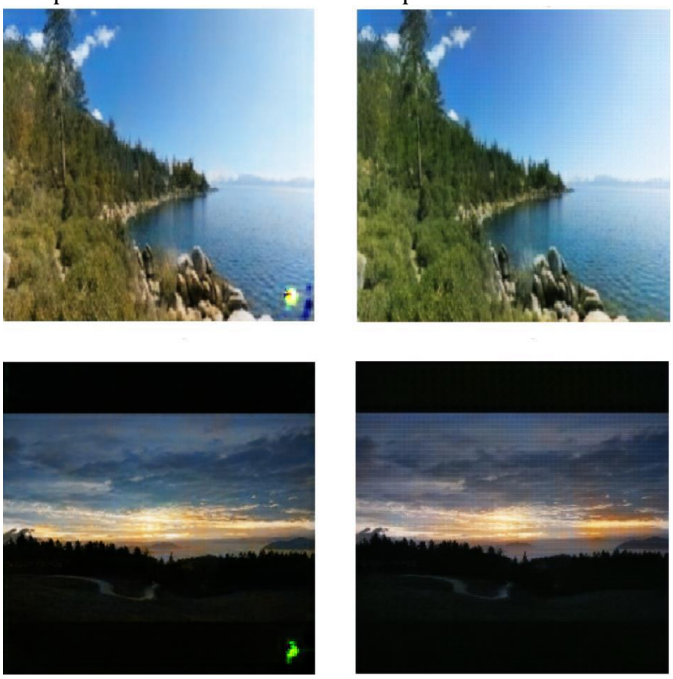

Fig 9: Pix2pix RGB models

In the 1st column, output of the mobilenetv2 version can be seen to contain a large spot at the bottom right corner which was consistent across all the images. Despite extensive training and experimentation, such artifacts were inevitable. This explains the high MSE value of 0.0289 and low PSNR value of 22.59 dB. Similarly, with the densenet121 version grid artifact can be observed in all the images that too were inevitable. Though the predicted colors were accurate, vibrant and realistic, the quality of image is degraded which is rightly represented by the high MSE value of 0.0237 and low PSNR value of 23.725 dB. Due to the presented issues and longer training times, RGB colorspace was not considered for further experiments.

5. CONCLUSIONS

We implemented four different Deep Learning models for automatic colorization of grayscale images, two based on CNN and two based on GAN. We have shown that a simple CNN model outperforms large and complex Inception-resnetv2 based model on a small dataset. It was easily able to produce plausible colors for the high-level components in images like sky, mountains and forests but did not focus on smaller details. The inception-resnetv2 based model was too sophisticated for the simple dataset we collected which was one of the reasons for its poor performance, though [3] had shown this model works well.

However, the Pix2Pix GAN models outperformed both CNN models providing high quality, nearly artifact free and vibrant output. Leveraging pre-trained Mobilenetv2 and Densenet121 allowed us to train high performance models with limited computational resources. Pix2Pix with MobileNetV2 was the most optimum in performance and compact enough to be deployed in a mobile device.

The generalisation power of the models is low due to the restricted data. Also, all these models perform poorly in complex scenes with fine details, though GAN models are still better than CNN based. A larger and diverse dataset would enable the model to learn a broader range of colour schemes increasing the generalisation power. Further, more complex models can be explored to mitigate the issues of colorizing small objects and fine details.

REFERENCES

[1] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification". ACM Transaction on Graphics (Proc. of SIGGRAPH), 35(4):110, 2016.

[2] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in ECCV, 2016.

[3] Baldassarre, F., Morín, D. G., & Rodés-Guirao, L. (2017). Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2. CoRR, abs/1712.03400. Retrieved from http://arxiv.org/abs/1712.03400.

[4] Jeff Hwang and You Zhou, "Image Colorization with Deep Convolutional Neural Networks", Stanford University.

[5] David Futschik, "Colorization of black-and-white images using deep neural networks", January 2018.

[6] Alex Avery and Dhruv Amin, "Image Colorization".CS230 – Winter 2018.

[7] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization", CoRR, vol. abs/1603.06668, 2016.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proceedings of the IEEE Conference on computer vision and pattern recognition, 2017, pp. 1125–1134

[9] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in International Conference on Articulated Motion and Deformable Objects. Springer, 2018, pp. 85–94