AutoColorization of Monochrome Images

Kushaagra Goyal
Stanford University
goyalk@stanford.edu

Bhuvneshwar LNU
Stanford University
bhuvnesh@stanford.edu

Yash Malviya
Stanford University
ymalviya@stanford.edu

Abstract

Automatic colorization of grayscale images is inherently a multi-modal problem. We target hallucinating a plausible colorization of a given grayscale image. We train different convolutional neural networks on CIFAR-10 dataset and compare the effect of varying the loss functions. We compare the effect of training the network using a regression loss, a classification loss and a generative adverserial network. Regression losses tend to give desaturated colorizations whereas GAN and classification loss generate more realistic and vibrant colors.

1. Introduction

Autocolorization of grayscale images is a powerful pretext task for self-supervised feature learning and has useful applications in image/video compression and information-extraction from historic image/video data. Its an ideal problem for automation owing to the ease of generating a large training set. In addition, its a fascinating idea to enable automatic yet realistic colorization of grayscale images.

Given a grayscale image, many times the semantics of a scene and the surface texture provide ample information to color a region of the image accurately. For instance, sky is mostly blue, grass is always green, however in many other situations the same grayscale image can map to multiple colored images. As seen in Figure 1 multiple colored dresses can have identical grayscale version. Accounting for this multi-modal nature we formulate our problem as hallucinating a plausible colorization of a grayscale image.

The sheer possibility of different colors being equally likely to be present in a picture for the same object explains the difficulty of evaluating the network fairly. We use two evaluation metrics namely Area Under the Curve (AUC) and classification performance on a pre-trained model. We believe though that the best metric would be to have humans distinguish between fake and real images.

We try out three different approaches to achieve the colorization task. The first approach is using a variety of regression losses, which include l2 loss, l1 loss and Huber loss (including smooth l1) for training our colorization network. The second approach involves using a classification loss with class-rebalancing as proposed in [3]. The third approach involves training a GAN (generative adversarial network) to achieve automatic colorization. We train convolutional neural networks implementing the approaches discussed above and compare their results.

The rest of the report is organized as follows, Section II discusses related work on autocolorization, Section III presents a description of implementation of the three approaches discussed above. Section IV presents a discussion on the results of our experiments and a comparison of the three approaches. Section V concludes the report and discusses the future work.

2. Related Work

Historical approaches of tackling image colorization [1] [2] required human intervention to specify colors in different regions of image (scribbling). Scribble based methods are time consuming and are limited by the skill set of person performing the image. Recent efforts have been focused on automated colorization methods.

In this project we focused on automated colorization of grayscale images. Specifically we explored parametrization of colorization models using CNN architecture. Parametric methods treat the colorization of grayscale images as either regression problem in continuous color space, or classification problem in discretized color space.

[5] treated colorization as regression problem with l2 loss. It consists of four main components, a low level features network, a mid-level features network, a global features network, and a colorization network. It concatenates Global and local features, which allows this model to run on image of any size which has been a drawback of models based on CNN's. Regression approach tends to be conservative in nature providing desaturated images as output.

[4] defined colorization of grayscale images as a classification problem. It uses pre-trained models (VGG with some modifications for grayscale images) to obtain spatially localized multilayer slices (hyper columns) as per pixel descriptors. It then uses these hypercolumns to predicts hue

Figure 1: Consider these differently colored dresses, all map to the same grayscale version which is the rightmost dress.

and chroma distributions for each pixel p. [3] also formulated this problem as classification problem but with class re-balancing for rare classes. The architecture is similar to VGG style network with added depth and is trained on Image-net data.

[6] took a different approach of using conditional generative adversarial networks to model the distribution. They used LSUN bedroom dataset and produced multiple colored images for a single grayscale image, which performed very close to ground truth images in terms of depicting reality

3. Methodology

3.1. Dataset

We used CIFAR-10 dataset - 50,000 training and 10,000 validation images of size 32*32*3. We chose this dataset since, low resolution images are computationally cheaper to train and hence enable faster prototyping. But having only 50,000 training images limited us to shallower models and caused issues in training very good models. The images were converted to YUV space and Y component was used as the grayscale component of the image. YUV space separates out luminance information from the color information. We train our model to only predict UV components from the Y component and append it to the existing Y component. We also tried directly generating RGB images but in our observation those models were harder to train in comparison.

3.2. Regression Loss

We trained a CNN model using L2, L1 and Huber losses which are given by following equations:

$$L2 = 1/2 * \sum_{hw} [(Y_{hw} - \hat{Y}_{hw})^2] \tag{1}$$

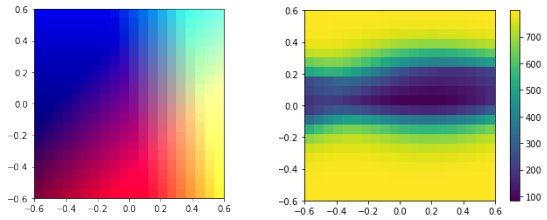
$$L1 = \sum_{hw} |(Y_{hw} - \hat{Y}_{hw})| \tag{2}$$

$$Huber = \begin{cases} 1/2 * a^2, & \text{if } a \leq \delta \\ \delta * (|a| - 1/2 * \delta), & \text{otherwise} \end{cases}$$

The regression network had an architecture similar to the classification network shown in figure 3, with the caveat that the regression network generates UV values directly and thus had only 2 outputs per pixel instead of a probability distribution.

3.3. Classification Loss

In this approach we modeled colorization as a classification problem. The UV color space is split into 400 evenly spaced bins, each bin represents an output class, figure 2a shows the different bins and their colors. Each pixel's class is predicted. Figure 3 shows the architecture of the CNN that takes as input a gray scale image and generates scores for all classes at each pixel.



(a) Bins in UV space,Y=0.5 (b) Weightage for rebalancing

Figure 2: 2a shows the color bins in UV space, classification model classifies pixels in these bins. 2b shows the weight assigned to bins for rebalancing

At training time we train the network using softmax loss function for which the expected distribution is generated from the ground truth. For each pixel, its actual UV bin is calculated, then a block of 9 bins with the actual value at center is assigned a Gaussian probability distribution with mean at the center bin and σ = 5

Further, its observed in [3] that majority of UV values in natural images tend to be concentrated in the center of the ab-space. Hence, in order to prevent most images from being biased to those values we perform class-rebalancing.

Class rebalancing involves weighting the losses corresponding to different ground-truth UV values differently such that the model learns to use all colors. The weights

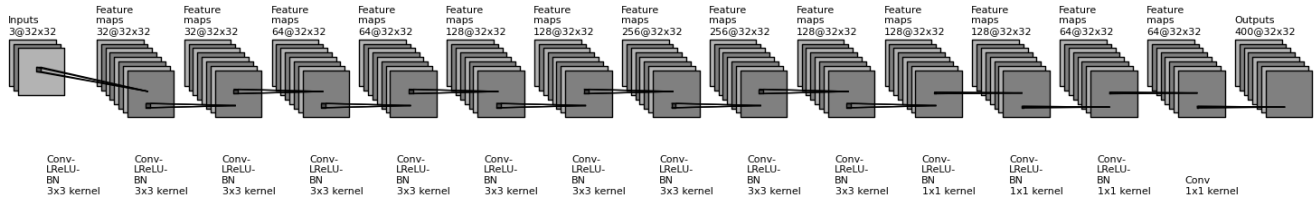


Figure 3: Architecture of our CNN model using classification approach, starting from 32x32x3 we generate output of size 32x32x400. BN stands for batch-normalization and LReLU represents leaky ReLU non-linearity

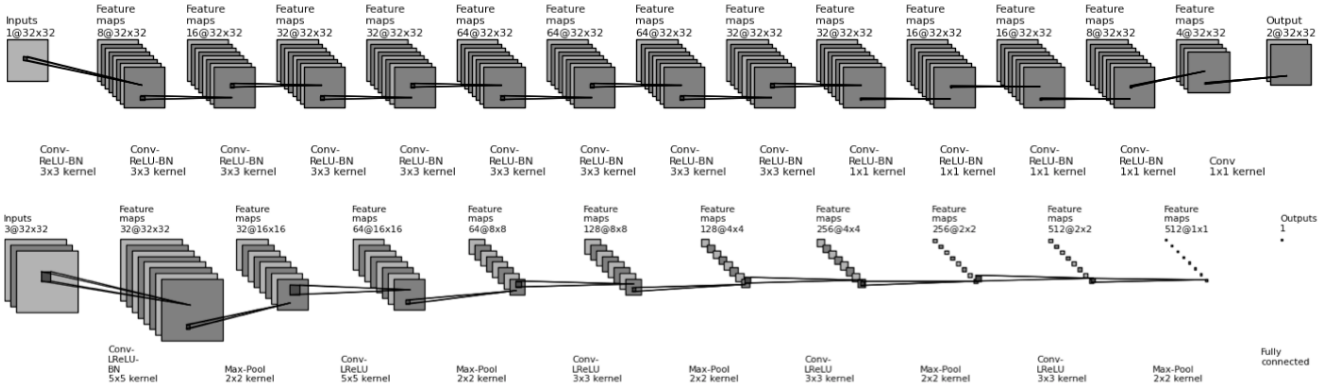


Figure 4: GAN model architecture, top and bottom panel represent generator and discriminator architecture respectively. BN stands for batch-normalization and LReLU stands for leaky ReLU non-linearity

used in rebalancing are calculated as follows

$$w \propto ((1-\lambda)p + \lambda/Q)^{-1} \tag{3}$$

where p is obtained by first calculating the empirical probability distribution of all the color bins in the CIFAR-10 dataset, following which we applied a Gaussian filer to it for smoothening (σ=5). Thereafter we mix this distribution (p) with a uniform distribution with weight λ (λ=0.5, Q=400 no. of bins). Lastly we take inverse of these weights and normalize such that expectation of weighting factor is 1. Figure 2b shows the final weights for CIFAR-10 dataset.

One last step to colorizing images in this approach is to map the output scores to UV values. Inspired from [3, 9] we used annealed mean of output scores to calculate final UV values. The formula for annealed mean is as

$$Annealed_{mean} = \mathbb{E}[f_T(z)] \tag{4}$$

$$f_T(z) = \frac{exp(log(z)/T)}{\sum_q exp(log(z_q)/T)} \tag{5}$$

setting T=1, leaves the distribution unchanged, however taking mean over a large set of values results in bias towards one mean value (resulting in purple colored images), contrarily setting smaller values to temperature results in strongly peaked distribution. As $\lim T \to 0$ we approximate one-hot encoding which results in sharp color changes in adjacent pixels, giving it a patchy look. For our model and dataset, T=0.32 gives the best results.

3.4. Generative Adversarial Networks

GANs are powerful generative models that cast generative modeling as a game between two networks : A generator trying to produce synthetic data and a discriminator/critic trying to distinguish between synthetic and real data. They can generate visually appealing samples but are generally hard to train and lots of research has gone into training them. We have a generator(G) that takes in a grayscale image and outputs a RGB version of the image which is fed to the discriminator. Our first attempt at training GANs involved a model architecture inspired from the DCGAN schema . The architecture is shown in figure 4. The corresponding loss functions are given as :

$$G = -1 * E_{z\sim P(Z)}[log(D(G(z)))] \tag{6}$$

$$D = -E_{x\sim P(r)}[log(D(x)] - E_{z\sim P(z)}[log(1 - D(G(z))] \tag{7}$$

We observed that the images produced by DCGAN had vibrant colors but were not quite crisp and had artifacts (Fig-

(a) Gray Image (b) L1 Loss (c) Huber Loss (d) L2 Loss (e) Ground Truth

Figure 5: Random sample of 16 images from their test set. Figure 5a shows the grayscale version, the next three figures show colorization produced by CNN's trained with different regression losses and figure 5e shows ground-truth

ure 8). We attributed this behavior to the finicky nature of GAN training and lack of training data but still results looked more pleasing visually as compared to the regression model.

In order to generate better results, we decided to experiment with the recently proposed Wasserstein GAN with gradient penalty. WGAN is supposed to be more stable to train and the value function has better theoretical properties than the original [7]. The modified loss function for the critic is given as:

$$L = E_{z\sim P(z)}[D(z)] - E_{x\sim P(r)}[D(x)] + \lambda E_{\hat{x}\sim P(\hat{x})}[||\nabla D(\hat{x})||_2 - 1]^2$$

For implementation of this loss function help was taken from the source code at [8].

The results from this were definitely better than the previous results and the images generated were much better. We believe that the size of the CIFAR 10 dataset may not be enough to train a good GAN colorization network. We also tried to integrate L2 loss and the GAN model by adding a weighted L2 term to the generator loss. This did not have a very noticeable improvement in the generated images but by visual perception, it looked like it helped a little.

4. Results

4.1. Evaluation Metrics

- AUC (Area Under Curve) : As an indirect test, we compute the percentage of predicted pixel colors within a thresholded l2 distance of the original RGB colors. The thresholds are swept to generate a cumulative mass function and the area under curve is normalized and taken as AUC. Note that it measures raw accuracy and not the plausibility, so its not a great metric to compare the performance. L2 loss should give near the best results for this metric.

- Classification on Pre-Trained Model : In this approach, we trained a classification model on CIFAR 10 dataset, which gives a validation accuracy of around 77 percent. Now to compare the performance of different models, we test the classification accuracy of the generated images from these models. If the classifier performs well, it shows that colorizations are accurate enough to be informative of object class.

- Human Perception : The ultimate test of the colorization is how compelling the images look to the human observer. Unfortunately, we did not use this metric to evaluate the performance.

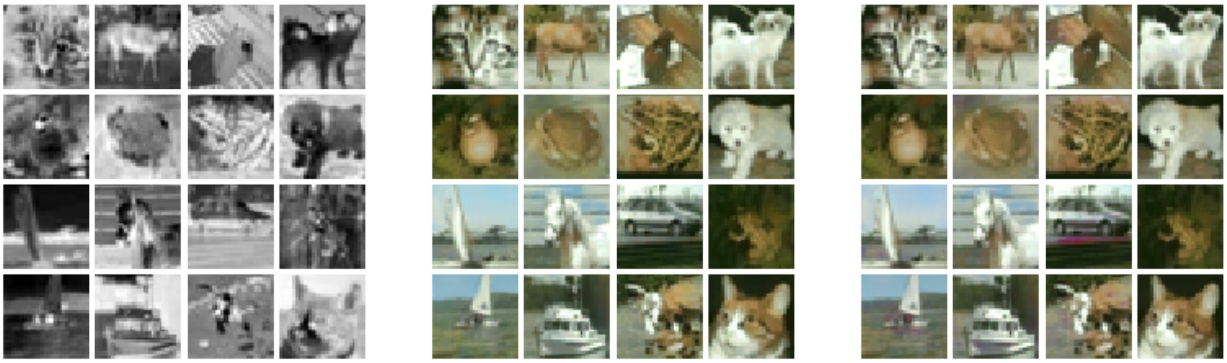
We evaluated the different approaches on the first two metrics. Though these metrics are not the real test of the model's performance but they give an estimate. The best test would be to visually see the results, hence we give visual results for the images generated from these models.

4.2. Analysis and Generated Images

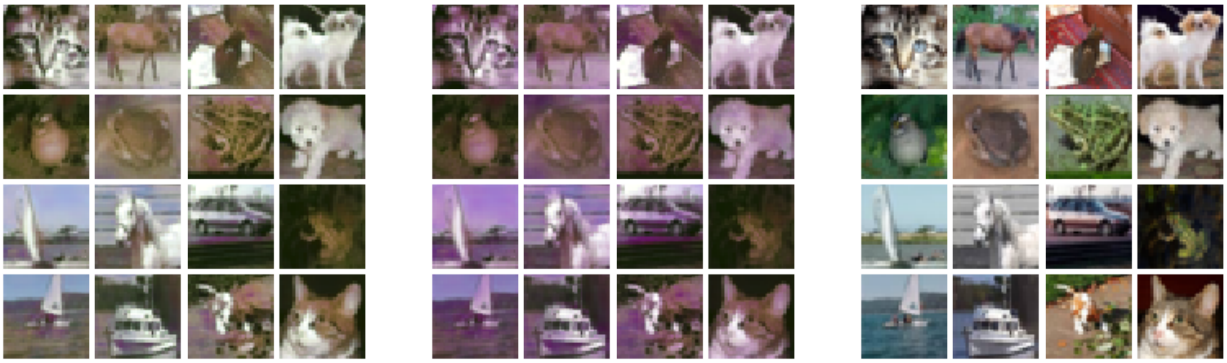
4.2.1 Regression Approach

As can be seen in figure 5 the images generated through regression approach show sharp granularity. At the same time, since the loss function is minimized by the mean/median of plausible colors for each pixel, the end result is desaturated images with little color variation. So the regression losses do a pretty good job in colorization, except for the fact that the results are desaturated. We explore classification and GAN approach for colorization to generate more vibrant and colorful images.

An interesting observation we made was that these regression losses took different number of iterations to start generating their best colorization. L2 loss was the slowest, followed by Huber loss and L1 loss network. This can be explained considering that all three networks are trying to generate a UV value pair that $\epsilon(-1, 1)^2$. Hence all losses are limited to that range. Since, L2 loss squares the error margin, hence it takes a longer time and more iterations to generate the same quality of colorized images. Figure 7 shows the loss curve for our L2 regression model, as it can



(a) Gray Image (b) T=0.2 (c) **T=0.32**



(d) T=0.4 (e) T=0.6 (f) Ground Truth

Figure 6: Random sample of 16 images from their test set. Figure 6a shows the grayscale version, the next three figures show colorization produced by classification model for different values of T, figure 6f shows ground-truth

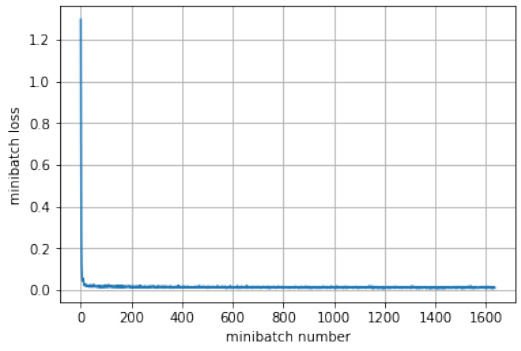


Figure 7: Figure shows the loss curve for L2 regression.

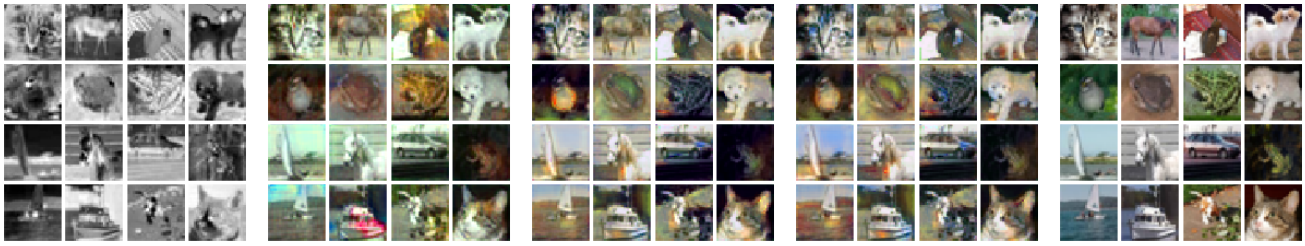
be seen the model converges well. As compared to classification network and GAN models L2 regression model converges the fastest.

4.2.2 Classification Approach

The classification approach was expected to capture the multi-modal nature of automatic colorization the best, however we observed that a vanilla classification network tends to give desaturated images, similar to regression approach.

As pointed out by [3] the distribution of colors in natural images is heavily biased towards near 0 UV values, for example - white background in many images. This causes the generated images also to have a bias towards these values, hence the desaturated image generation. To resolve this we apply class-rebalancing as described in Section 3.4.

Adding rebalancing improves the color variation in images by weighting rare color variations higher than common ones, as can be seen in figure 2b. Finally converting the output color distribution generated by the classification network with appropriate annealed averaging (T=0.32) we get good colorized images. Figure 6c shows a sample of the colored images generated by the classification network. Values of T below 0.32 result in yellowish desaturated images, whereas higher values of T slowly turn images into a



(a) Gray Image (b) DCGAN (c) WGAN (d) WGAN with L2 (e) Ground Truth

Figure 8: Random sample of 16 images from their test set. Figure 8a shows the grayscale version, the next three figures show colorization produced by GAN networks and figure 8e shows ground-truth

purple shade. 0.32 achieve an optimal balance and generates best colorizations.

We believe the performance of the classification approach can be further improvised by performing non-uniform binning of the UV space such that more bins are available at sub-zero UV values enabling higher variation in that highly frequent region. However that is a part of future work. Figure 9 shows a plot of training loss for our classification model. We observe that the loss function decreases initially as the model learns to identify different objects in the image, thereafter the loss remains steady as the model learns to color objects aptly. The loss remains steady even as we increase the number of epochs.

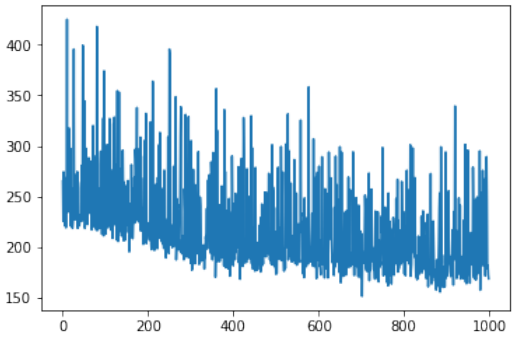


Figure 9: Loss curve for Classification model

4.2.3 Generative Adversarial Networks Approach

In figure 8, we can see the results generated by the Generative Adversarial networks. The first set of images (corresponding to DCGAN) show that GANs can color the images pretty good. They tend to throw bright and vibrant colors at the images. But the results tend to have some shortcomings, there are artifacts in some of the images and the color boundary are generally very sluggish and the images are not sharp. We believe that the potential reasons for this could be either not having a proper training regime, the poor resolution and the quantity of the data (only 50000 images).

So to achieve slightly better results we decided to try out WGAN with gradient penalty as referred in [8]. This approach tends to have better training properties as compared to the the DCGAN. We saw a little improvement in the results using WGAN as is evident from the figure. The generated images are colorful but still they suffer from artifacts. Since the regression loss images are generally pretty stable and sharp, we decided to try a new approach where we add a weighted L2 term to the generator loss. But the results did not improve much, although they had slightly less bright spots now.

Its evident that the GANs have the potential to generate pretty decent images if trained properly. They tend to produce vibrant colors. If the finicky and the unreliability of GA's could be solved they could be a very powerful colorization tool. Figures 11 and 12 show the training losses for the generator and discriminator network of our GAN model respectively. On comparison with [8] we observe that our Loss curve closely resemble those observed in previous work, establishing that our WGAN network was trained decently well.

4.2.4 Comparative Analysis

Now we give a qualitative and quantitative comparison of the results generated by the three models. Figure 13 and 14 give a set of 16 images each with the outputs produced by the three different class of models. The table 1 gives the AUC and classification accuracy of the three types of models. It can be seen that L2 loss has the highest AUC which was expected because L2 loss inherently minimizes the squared distance between target and actual image. L2 regression loss also fares best in terms of classification accuracy, despite its desaturated colorization.

GANs have the lowest classification accuracy as compared to the L2 and classification loss. We can attribute this to the patches we observe in the output of GAN.

(a) Black and White (b) Classification Model (c) GAN model (d) L2 model

Figure 10: 16 Legacy BW photos were taken from the internet and resized to 32*32 and colorized using our three different models

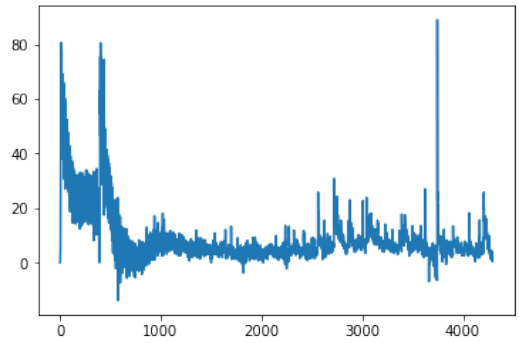


Figure 11: Loss curve for WGAN generator.

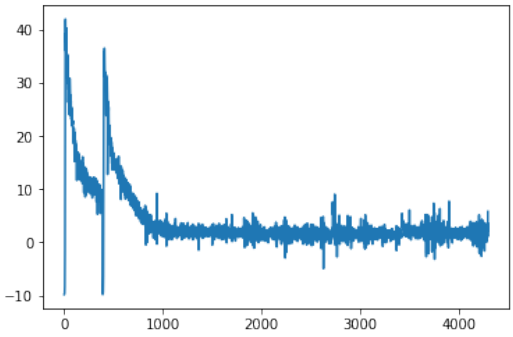


Figure 12: Loss curve for WGAN discriminator.

Note that even though our observation indicates that GANs and classification model generated better colored images our metrics indicate otherwise. However these metrics are an indirect measure of performance of autocolorization. Human perception is the real test and we hope to do that as a part of our future work.

Comparing the three models, figure 13 and figure 14 from our visual perception, the classification model seemed to be the best model with the most realistic colorization.

	AUC (%)	**Classification T (%)**
Grayscale	80.33	22.19
Regression, L2 loss	98.37	67.75
Classification	98.28	66.6
DCGAN	97.26	61.24
WGAN	97.54	64.07
Ground Truth	100	77.76

Table 1: Evaluation results for the best networks from our three approaches, AUC stands for Area under the curve and Classification T denotes classification accuracy on pre-trained model as defined in section 4.1

The best results observed in [3] seem more vibrant and realistic as compared to some of our best results, this clearly indicates that CIFAR-10 with its small image size (32x32x3) and smaller number of images (50000-train-10000-test) is good for prototyping only, however for more vibrant and realistic colorization training on a larger image-set like IMAGENET might be a better choice. Due to limited resources and time we could not use this dataset.

4.3. Legacy Photos Colorization

Our model was trained from 'fake grayscale' images generated from RGB images from the CIFAR dataset. In order to check performance against real black and white photos, we took legacy black and white photos from the Internet, resized them to 32*32 and passed through our model. In figure 10, it can be seen that all the models give decent results for the task. The classification model gives pretty good images except a few purplish patches. The L2 model gives desaturated results, whereas GAN model gives patchy results but with vibrant colors. On a whole our models are able to achieve good colorizations on legacy photographs.

(a) Gray Image (b) WGAN (c) L2 Loss (d) Classification (e) Ground Truth

Figure 13: Random sample of 16 images from their test set. Figure 13a shows the grayscale version, the next three figures show colorization produced by the best models in regression, classification, GAN approaches and figure 13e shows ground-truth

(a) Gray Image (b) WGAN (c) L2 Loss (d) Classification (e) Ground Truth

Figure 14: Random sample of 16 images from the test set. Figure 14a shows the grayscale version, the next three figures show colorization produced by the different models and figure 14e shows ground-truth

5. Conclusion and Future Work

Automatic colorization of grayscale images has interesting applications like image/video compression. Another useful application is coloring the legacy photographs which we demonstrated using a sample of 16 legacy photographs. Here we also presented a comparative analysis of different deep CNN architectures trained to generate plausible colorizations for grayscale images and observed some unique traits of different approaches.

Regression losses generate images with sharp granularity however they are mostly desaturated. On the other hand, GAN generates images full of vibrant and realistic colors, however images colored by GAN are patchy and lack sharpness. Classification loss network presents middle ground between the previous two generating images which have sharp boundaries and vibrant colors. A sample of good colorizations from the classification model are plausible enough to deceive human perception.

For the future work, we would like to train our models on larger dataset like Image Net and high resolution images. Another thing that can be explored is the non-uniform binning around the highly frequent portions of the UV space. Moreover, we can also have a study regarding the effects of different color spaces like HSV, YCrCb and so on. Lastly, a human survey test should be done to effectively evaluate the performance of the colorization model.

Code for this project can be found at *https://github.com/bhuvnesh2259/AutoColorization*. It contains source code in src folder and model binaries in model folder (regressions, classification, GANs).

References

[1] Levin A., Lischinski D., Weiss Y. *Colorization using Optimization* ACM Trans.Graph., 23(3):689694, 2004. 1

[2] Huang, Y.C. Tung, Y.S., Chen, J.C., Wang, S.W., Wu, J.L. *An adaptive edge detection based colorization algorithm and its applications* ACM international conference on Multimedia (2005). 1

[3] Zhang R., Isola P., Efros A.A. *Colorful Image Colorization* European Conference on Computer Vision (2016). 1, 2, 3, 5, 7

[4] Larsson G., Maire M., Shakhnarovich G. *Learning Representations for Automatic Colorization* European Conference on Computer Vision (2016). 1

[5] Iizuka S., Simo-Serra, Ishikawa H. *Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors*

for Automatic Image Colorization with Simultaneous Classification Computer Vision and Pattern Recognition (cs.CV) (2017). 1

[6] Cao Y., Zhou Z., Zhang W., Yu Y. *Unsupervised Diverse Colorization via Generative Adversarial Networks* Computer Vision and Pattern Recognition (cs.CV) (2017) 2

[7] Arjovsky M., Chintala S., Bottou L. *Wasserstein GAN https://arxiv.org/pdf/1701.07875.pdf*. 4

[8] Gulrajani I., Ahmed F., Arjovsky M., Dumoulin V., Courville A. *https://arxiv.org/pdf/1704.00028.pdf:* Improved Training of Wasserstein GANs (2017). 4, 6

[9] Kirkpatrick, S., Vecchi, M.P., et al. *Optimization by simmulated annealing.* Science (1983). 3

[10] Deshpande, A., Rock, J., Forsyth, D., *Learning large-scale automatic image colorization.* Proceedings of the IEEE International Conference on Computer Vision (2015).

[11] Farabet, C., Couprie, C., Najman, L., LeCun, Y., *Learning hierarchical features for scene labeling.* Pattern Analysis and Machine Intelligence, IEEE Transactions (2013).

[12] Ramanarayanan, G., Ferwerda, J., Walter, B., Bala, K.*Visual equivalence: towards a new standard for image fidelity.* ACM Transactions on Graphics (2007).

[13] Deng, Jia, et al. *Imagenet: A large-scale hierarchical image database.* Computer Vision and Pattern Recognition, 2009.

[14] LeCun, Yann, and Yoshua Bengio. *Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks(1995).

[15] Goodfellow, Ian, et al. *Generative adversarial nets.* Advances in neural information processing systems (2014).

[16] *https://github.com/richzhang/colorization* .

[17] *https://github.com/igul222/improved_wgan_training*

[18] *https://github.com/cameronfabbri/Colorful-Image-Colorization*

[19] *https://github.com/aleju/colorizer*.

[20] *https://github.com/gwding/draw_convnet*.

[21] *http://cs231n.github.io/assignments2017/assignment1/*

[22] *http://cs231n.github.io/assignments2017/assignment2/*

[23] *http://cs231n.github.io/assignments2017/assignment3/*

[24] *https://gist.github.com/omoindrot/dedc857cdc0e680dfb1be99762990c9c*

[1]

[1] Code-base: *https://github.com/bhuvnesh2259/AutoColorization*