Artificial Image Colorization Using Transfer Learning and EfficientNet

Troy Cope

Abstract—Automatic Image Colorization is the procedure of transforming a gray-scale image into a colored image without any human intervention. This field is highly researched and strongly applicable for the real world due to: historic importance, data generation/augmentation, and human satisfaction. The main objective of this research is to develop an artificial intelligence feature extraction method which implements color into a gray-scale image with a more efficient or more accurate model than the baseline [1]. To solve this problem, this paper relied on transfer learning [2] through the EfficientNet [3] model using the Places [4] dataset. The problem was treated in multiple parts, those being: processing of images into features, feature extraction using the model, and then colorization via the luminosity channel. By implementing repetitive training via history, early stopping and dropout layers (final with a singular dropout layer), the new model preformed significantly faster and had minor improvements in accuracy.

Index Terms—Artificial Neural Network, Convolution Neural Networks, Multi-layered CNN, Neural Networks, Transfer learning

1 Introduction

Automatic Image Colorization is the procedure of transforming a gray-scale image into a colored image without any human intervention. The reason for automatic colorization is to solve human intensive problems such as historical image restoration, colorizing of movies and older family artifacts. These problems also extent to a data perspective, where a programmer may want to exploit color from a black and white image/pixel. The topic of automated image colorization has been explored with plenty of models. This problem is particularly challenging due to the ill-posed (due to many degrees of freedom for colorization) nature of turning a gray-scale image into a fully colored one. It has also been explored over a multitude of cases deep learning techniques, specifically CNNs [4]–[8] and GANs [9]–[11], have shown successful results in problems concerning machine vision. In previous cases there have been results similar to this project, as cited previously. The goal of this paper is to present the Stanford paper's colorization algorithm based on Convolution Neural Network (CNN) through an expanded version of EfficinetNetB7 [3]. I formulated image colorization as a regression problem and CNNs are used to solve this problem. This research will shine further light onto methods for solving automated colorization as well as gauge the efficiency of the three (3) models used in this project.

2 Related works

The first related work that should be mention is the baseline paper for this research addition. This paper was proposed by Hanzhao and Rafael in 2021 from Stanford University [1]. Hanzhao and Rafeal implemented a VGG-16 Model [6], and EfficientNetB7 [3] model with CNN back ends to evaluate colorization on images. Their model(s) lacked complexity and time efficiency, thus this paper tried to improve one of those areas.

The paper proposed by Xiao, et al. [10] was an extensive study on image colorization using a CycleGAN solution. This solution was much more appropriate to the condition of image colorization, having a more complex architecture, and using grayscale SUN data. Ideally, my model would be able to implement some of these GAN pieces in order to build a more reliable color image prediction or stack onto the architecture.

The paper authored by Nazeri, et al. [11] is another implementation of a GAN, specifically a DCGAN (Deep Convolution Generative Adversarial Network). GANS see much more suited for the implementation of gray-scale to color imaging, and as such they are very complex networks. It can be justifiably said that this paper proposed a much less resource strenuous solution for mediocre results.

The paper written by Li, et al. [12] was over automatic feature selection and fusion to colorize images. It is noted that the colorization is affected by the local image region. This current paper's implementation does not have locality issues, but could implement more complexity such as [12]'s Markov Random Field model.

Probabilistic image colorization, written by Royler, et al. [13] is a framework proposition which can display multiple plausible colorization options and gives a proper stochastic sampling scheme. Royler's paper uses a much different dataset, those being CIFAR-10 and ILSVRC 2012 which are arguable more varied that places but with less data. This paper is focused on optimizing a practical, current model, but this paper provides insight to future attempts.

Another probabilistic model is shown with Charpait, et al. [14], where the technique used is Support Vector Regression from a framework. It uses the grouping of pixel predictions to drive the colorization which are applied on a larger scale to color the whole image. Charpait's solution is very efficient, so much so that it is work noting that the

Troy Cope is with the College of Computing and Software Engineering, Kennesaw State University, Marietta, GA, 30060 USA. Email: tcope2@students.kennesaw.edu

complexity of this current paper's model overcomes it.

The next example is a new form of colorization, using similar images [15]. Gupta, et al. explains that the user has to supply an image close to the gray-scaled image in order to operate. For this reason it does not fully apply to this paper since this paper is concerned with automatic colorization. The concept, however, would be very useful to help verify results.

A similar application to the previous few is patch-based image colorization by Bungeau, et al. [16] The second part of this paper is much more applicable due to it's automatic nature, but the first part shows the effective use of predefined color inputs. This current paper's model is comparatively state of the art, and therefore a much stronger baseline for application.

Image colorization using histogram regression is a common topic, as shown in [17] from Lui, et al. The approach described is a much more dated approach, and less efficient than most current methods. Due to the heavy processing necessary this research is a good reference for how to go about research and report of an ill-posed task.

The last paper used as reference was one comparing image colorization in general, from Grgic, et al. [18] Grgic's paper is a very good example of a theoretical research paper that has a wide application. It shows off the multitude of methodologies for approaching this current paper's predicament. This shows some of the most efficient are user-guided networks, but for this paper's hypothesis that was not applicable. For this reason this paper pursued the solely neural network application.

3 Approach (and technical correctness)

The project build was started using a multitude of datasets that could fit the purpose of image colorization. For this case, the optimal choices were datasets with numerous images due to no dedicated image colorization dataset. The datasets chosen were Places [4], ImageNet [19], and Food-101 [20]. These datasets have been used in similar cases, such as image classification, segmentation, and other colorization projects. Due to the nature of the images being fully colored, it was simple to convert them into gray-scale for the models to process as inputs. The goal of the model was then to find a colorized version of the image as close to the ground truth as possible, evaluated by the cost functions.

Fig. 1: A sample of images from the Places dataset.

The majority of this research is centered around the small version of the Places dataset. This dataset is roughly 2 million images of 256 x 256 resolution with roughly 400 scene categories. To give context, Food-101 only uses 101,000 food photos which lacked the data necessary to train the models for all categories of generic real-world application. On the other hand, ImageNet, which contains 14 million images, was unable to be processed within this research's resources. Therefore, Places was the ideal dataset for size and generalization.

Fig. 2: A sample of images from the Food101 dataset.

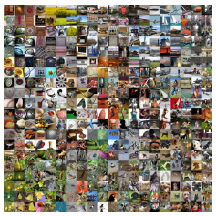

Fig. 3: A sample of images from the Imagenet dataset.

Most of the issues with data stemmed from the author's laptop hardware restrictions. All of the data presented was run on a Windows 10 G-3 Dell Laptop with an Intel I5-83000H CPU @ 2.30GHz, and 8GB of ram, and 64-bit system type. Due to these constraints, the data set was trimmed down to 3% of their initial portions.

Data preparation was implemented on the Places dataset following the recommended split. This split was: 1,803,460 images for training, 36,500 images for validation, and 328,500 images for testing. Due to hardware restrictions, these numbers were applied at 3% as: 54,104, 1,095, and 9,855, respectively. These images were ideal for integration due to the fact that they were already 224 x 224 - the same as the batch size. These 224 x 224 images were then augmented with randomized rotations and horizontal flipping of the 1,803,460 images in the training set.

In order to build the data the models will be trained on, the images from the training dataset need to be transformed into grayscale images. This was done by taking the RGB color space and filtering it through CIE Lab [21] color space to generate the L* channel. This luminance channel is the human-perceived lightness value which are normalized to floating point values from 0-1. Thus, the process is summarized as taking a 224 x 224 color image and resulting as a grayscale 224 x 224 x 1 floating point matrix with values between 0-1 in post.

Fig. 4: Sample image after preprocessing was completed.

The objective of the code was to minimize the mean squared error (MSE) from the 2 output color channels in CIE Lab color space (a* and b*) in respect to the ground truth of the image and the model's predicted image.

Equation 1 is used to represent the Mean Squared Error [22] loss of a singular point in the training dataset using the ground truth (Y) and the prediction (Y$\prime$) :

$$L(Y^i, Y\prime^i) = \frac{1}{2hw} \sum_{h,w} ||Y^{(i)}_{h,w} - Y\prime^{(i)}_{h,w}||^2_2) \tag{1}$$

Due to the model's tendency to choose desaturated colors, Equation 2 is used as a way to penalize common colors in order to encourage much more varied colorization. The inspiration was shown in [1] and [8]. This led to the following modified cost function:

$$L(Y^i, Y\prime^i) = \frac{1}{2hw} \sum_{h,w} v(Y^{(i)}_{h,w}) ||Y^{(i)}_{h,w} - Y\prime^{(i)}_{h,w}||^2_2 \tag{2}$$

This then leads to a color re-balancing function, represented by v, which related weight to a rarity of color. Therefore, we can then represent the weight of each color as:

$$v(a,b) \propto \frac{1}{(1-\lambda) \times p\prime(a,b) + \lambda \div 4}$$
$$s.t.\ \mathbb{E}[v(a,b)] = \sum_{a,b} p\prime(a,b) v(a,b) = 1 \tag{3}$$

Equation 3 is inspired by [1] and [8], and obtains an empirical probability p prime, via discretized (a*,b*) pairs in the full training set and then smoothed with the distribution of a Gaussian kernel. Lastly, a hyperparameter to mix the probability with a uniform distribution is added, lamda bound to the domain of [0,1]. These all become normalized to create a final weight factor of 1.

4 Experimental results (and technical correctness)

This approach, much like the base paper [1], was implemented on Google Colab. Through vigorous literature review and a constraint of computing resources, transfer learning [2] was seen as the best approach to maximize accuracy and minimize time dedication. In order to achieve this, the models from Dahl [6] were used as a starting point due to their utilization of the Places dataset. This helped with performance tremendously for customizing objective functions as well as modifying NN models which served as important variations. The EfficientNet Model [3] parameters were pulled from the default pretrained weights on the ImageNet dataset and frozen to ensure that they maintained their training.

The main model is a VGG-16 Base, with 5 intermediate layers swapped for EfficientNetB7 layers. This maintains the pretrained nature of the VGG-16 network while allowing for the EfficientNet Feature extractor. Those layers are then connected to multiple Conv2D blocks, first one being a 1x1 block to build a 7x7x1334 output matrix. All the following blocks have the build: 3x3 Conv2D layer, Batch Normalization Layer and ReLU activation layer in that order. This output is finalized as a 224x224x2 setup, which is filtered through a tanh activation layer (between [-1,1]) to reflect the ground truth's values for a* and b* in CIE Lab. This strategy and architecture was suggested by: [1], [7] in order to match the most updated models.

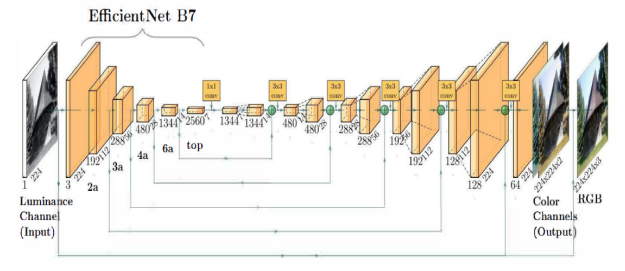

Fig. 5: The baseline model described in the last paragraph.

Hyper-parameters of the following were implemented: Adam optimizer [23] with transfer learning rate [2] of 0.001, B_1 as 0.9 and B_2 as 0.999 throughout training, 20 epochs per model, early stopping [24] with a patience of 2.

To find the empirical probability (p with squiggle) for rebalance from the given model, a* and b* are pushed into 256 x 256 grids, with a 10x10 Gausian [25] kernel over sigma = 5 smoothing distribution. Next Lamda=0.9 is mixed with the uniform distribution in Equation 3.

All results were shown using MatLab's plotting functions on the completed history of each run model. The accuracy and loss, for both validation and standard, were taken from the fitting of the model.

All of these metrics are used to predict and quantify the results of the CNN-Based architectures, and each apply their own benefits and costs to the networks. They are used in tandem in order to achieve the best results. Multiple cost functions as well as other architectures were tried as well, but due to varying structures it was difficult to see how they applied directly to the scope of this project. Therefore, my personal application was on a modified Baseline Model (presented in [1] and based off [8]), which had EfficientNetB7 layers replace the intermediate of the pretrained VGG-16 Model's. This paper's addition was this model, but one that employed dropout [26] and early stopping [24] to optimize the baseline and expand the goal of the project. These models were evaluated via cost and accuracy.

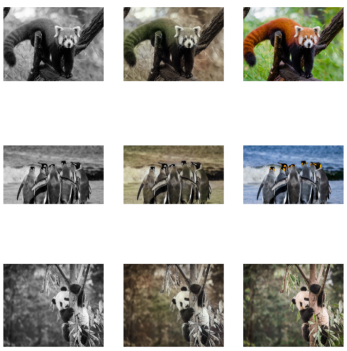

Fig. 6: The results of the adjusted baseline model are shown above.

Generally, no matter which color space is being used, models will predict 2 missing color channels for each pixel, thus the output image is represented as 224 x 224 x 2 floating number matrix. To display the model output, the input luminance channel and two output color channels need to be unnormalized, combined and transformed back into RGB color space. This is completed after the model to show the following image, leftmost being the grayscale image, the middle being the predicted image, and the rightmost being the ground truth.

Despite the baseline model and this paper's model being almost identical, there are other empirical statistics that show the paper's model to be superior. The metrics evaluated are shown below:

TABLE 1: This table shows the comparison between the three major models, this paper's presented model, the baseline, and the baseline after being trained color augmentation of Equation 3:

	New Model	BaseLine	BaseLine (3)
Execution (hours):	4.668333	15.56111	15.56111
Val Loss:	0.008	0.0086	0.0057
Val Accuracy:	0.7523	0.7139	0.6766
Loss:	0.007	0.0082	0.0047
Accuracy:	0.7662	0.7225	0.6886

As shown in the table, the new proposed model is much more time efficient, with a 333% increase in run time, and a 0.0384 increase in the model's accuracy in comparison to the baseline without color reblancing. Thus, it can easily be said that this new proposed model is a better augmentation of the baseline model.

To analyze these results in comparison to the baseline with color rebalancing it is important to note a few things. The baseline noted that using this objective function (1) led to a tendency of desaturated colors. Inspired by [8], the authors of this paper's baseline [1] analyzed the training set and confirmed that desaturated colors appear much more frequently than vivid colors in the real world. They chose to apply a penalization to common colors to encourage output of more vivid colors. The cost function was then modified to be [-1, 1]. This is the inspiration for BaseLine (3) as shown in the large deviation of accuracy above. The model is trained on these new equations (2 and 3) and built to create more colorful images. Due to MSE being the evaluation at the end of all these models, the change in vibrancy results in a worse objective comparison. That is why the baseline without color balancing is so much closer to the new proposed model despite being a direct augmentation.

The trend of the color set to become dark yellow and light brown in nature is due to the simple approach of the baseline models. This deterring issue was attempted to be mitigated with through more complicated model architecture and more indirect cost functions. The effects of this are shown through EfficientNet and the color rebalancing function providing more colorful images. The issues with this approach is that it can be seen to preform as well as or worse than the baseline model in multiple cases. This is due to the fact that the metric used to measure a color's accuracy (in this case MSE) favoring an average of distances over getting data points with high variable precision and low accuracy. This can be concluded that the key metric disconnects human preference from the achievable result on this colorization task. Even in this case, however, this paper has shown that by optimizing these models it is still possible to increase the accuracy of the model's intended use. The problem lies in the application of these well-fitted models to a linear cost function for an ill-posed problem. The problem is ill-posed because it is not easily traced back to luminance in nature, and there are multiple variations which the model prefers to average the tonal colors of.

5 Conclusion

In conclusion, Automatic Image Coloriation, or the procedural transformation of a gray-scale image into a colored image wihotut human interaction, is a very ill-posed problem due to the many different routes of color from a singular luminance value. Therefore, unless forced to do otherwise, the models will opt to find the average of the color set and learn related patterns. This can be observed in multiple places due to organic material being predictable, whereas the materials of clothes and other inorganic matter are often incorrectly colored. In the case where something has the possibility of being incorrectly colored, the model takes on the yellowish brown hue seen in the test images. This is because the baseline paper [1] proved that the average of the places dataset match to these colors. In regards to the models, tuning of epochs as well as training each model from its past history gives the model more accuracy. Whether this is due to the model finding out textural components in images or predicting luminance better is unknown, but the most likely case is that the models are getting closer to the precise average of all colors in the training set. Dropout and early stopping caused significant changes to time dedication and stopped the model from declining, so those were very good additions to the base model. Therefore, it can be concluded that these should be included in future models of the same type. Going forward, it would always be better to change the metric of evaluation, or introduce other components to deter the model from picking the average. Perhaps a form of reinforcement learning would assist with this on a pixel by pixel basis, or a stronger model in general like a GAN could be taught to avoid these cases.

References

[1] Hanzhao and Rafael, "Automated image colorization using deep learning." Stanford University, 2021.

[2] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[3] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[4] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*. Ieee, 2017, pp. 1–6.

[6] R. Dahl. Automatic colorization, tinyclouds.org.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[9] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[10] Y. Xiao, A. Jiang, C. Liu, and M. Wang, "Single image colorization via modified cyclegan," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3247–3251.

[11] 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.

[12] B. Li, Y.-K. Lai, and P. L. Rosin, "Example-based image colorization via automatic feature selection and fusion," *Neurocomputing*, vol. 266, pp. 687–698, 2017.

[13] A. Royer, A. Kolesnikov, and C. H. Lampert, "Probabilistic image colorization," *arXiv preprint arXiv:1705.04258*, 2017.

[14] S. Liu and X. Zhang, "Automatic grayscale image colorization using histogram regression," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1673–1681, 2012.

[15] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong, "Image colorization using similar images," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 369–378.

[16] A. Bugeau and V.-T. Ta, "Patch-based image colorization," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 3058–3061.

[17] S. Liu and X. Zhang, "Automatic grayscale image colorization using histogram regression," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1673–1681, 2012.

[18] I. Žeger, S. Grgic, J. Vuković, and G. Šišul, "Grayscale image colorization methods: Overview and evaluation," *IEEE Access*, 2021.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[20] L. Bossard, M. Guillaumin, and L. V. Gool, "Food-101–mining discriminative components with random forests," in *European conference on computer vision*. Springer, 2014, pp. 446–461.

[21] X. Zhang, B. A. Wandell *et al.*, "A spatial extension of cielab for digital color image reproduction," in *SID international symposium digest of technical papers*, vol. 27. Citeseer, 1996, pp. 731–734.

[22] K. Das, J. Jiang, and J. Rao, "Mean squared error of empirical predictor," *The Annals of Statistics*, vol. 32, no. 2, pp. 818–840, 2004.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[25] G. Deng and L. Cahill, "An adaptive gaussian filter for noise reduction and edge detection," in *1993 IEEE conference record nuclear science symposium and medical imaging conference*. IEEE, 1993, pp. 1615–1619.

[26] P. Baldi and P. J. Sadowski, "Understanding dropout," *Advances in neural information processing systems*, vol. 26, 2013.