Image Colorization Enhancement using Transfer Learning

Aayushi Beniwal, Sai Anurag Neelisetty, and Uttej Reddy Pakanati
abeniwal@uwaterloo.ca, saneelisetty@uwaterloo.ca, urpakanati@uwaterloo.ca

***Abstract*—Image colorization is an active area of research, and many new strategies are introduced every day wherein deep learning is extensively used to enhance the process. In this paper, we propose a transfer learning method that uses EfficientNet as a feature extractor to aid the process of fully automatic image colorization. Later, the results of the suggested method have been compared with three existing techniques which aim to solve the same problem.**

***Keywords*—Convolution neural networks, EfficientNet, Auto-encoders, Image Colorization, Transfer learning, structural similarity index measure**

I. Introduction

Image colorization is an image-to-image translation problem. It can be considered a pixel-wise regression problem where structures in inputs and outputs are highly aligned. Colorization of images is a tricky problem due to the changing conditions of imaging that need to be dealt with by a particular algorithm. It is a complex problem because two out of the three image dimensions are not present; although the scene semantics may be beneficial in many cases, for example, the grass is generally green, clouds are primarily white, and the sky is blue. However, such semantic priors are rare for many artificial and natural objects, e.g., shirts, cars, flowers, etc. With the swift progress of deep learning techniques, various image colorization approaches have been proposed, and state-of-the-art performance on existing datasets has been reported.

Transfer learning is a subfield of artificial intelligence and machine learning which intends to apply the knowledge gained from the source task to a different but similar task. A pre-trained model is designed and trained by someone else to solve a problem like ours. The researchers usually choose a huge dataset such as ImageNet or the Wikipedia Corpus as their base dataset. Then, they create an extensive neural network (e.g., VGG19 has 143,667,240 parameters) to solve a particular problem. This pre-trained model is then made public so that the models can be repurposed. The pre-trained model is loaded and then fine-tuned to achieve higher accuracy and to generate the output in the correct format [1]. The pre-trained model considered in this paper is EfficientNet-B4.

This paper first introduces our proposed method of improving image colorization using EfficientNet. Then, the architecture that we employed, and the training and test datasets considered for our tests are explained. Next, the results of the proposed method and existing methods of image colorization enhancement are compared and evaluated in detail. Lastly, a conclusion is given.

II. EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that evenly scales width/depth/resolution dimensions using a compound coefficient. Unlike traditional practice that arbitrarily scales these factors, the EfficientNet scaling method evenly scales network width, depth, and resolution with fixed scaling coefficients. For example, suppose we want to use 2^N times more computational resources, then we can increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, γ are constant coefficients determined by a small grid search on the original miniature model. EfficientNet uses a compound coefficient Ø to uniformly scale network width, depth, and resolution in a principled way [2]. The intuition that justifies the compound scaling method is that if the input image is bigger, the network needs further layers to increase the receptive field and additional channels to capture more fine-grained patterns on the bigger image. Fig. 1 illustrates the difference between the compound scaling method and conventional methods.

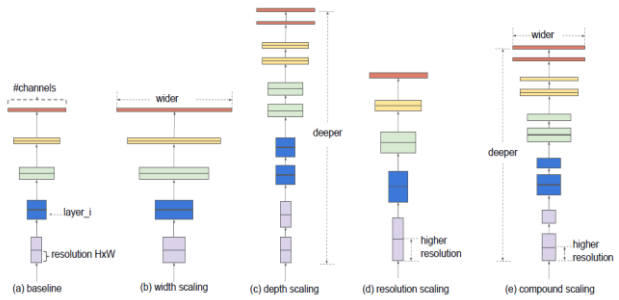

Fig 1. Model Scaling [2]

The base EfficientNet-B0 network is centered on the inverted bottleneck residual blocks of MobileNetV2, and squeeze-and-excitation blocks. EfficientNets also transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and three other transfer learning datasets with fewer parameters [2]. Fig. 2 summarizes the ImageNet performance, where EfficientNets significantly outperform other ConvNets.

Our proposed method uses EfficientNet-B4 as the feature extractor, and Fig. 3 shows the architecture for the same. The reason behind choosing EfficientNet-B4 is that its top-1 accuracy is higher than the models Xception, Inception-ResNet-v2 & ResNeXt-10 and it also has a considerable number of parameters to train as seen from the Fig. 2. On ImageNet dataset, EfficientNet-B4 performed better when compared to widely used ResNet-50 improving the top-1 accuracy from 76.3% of ResNet-50 to 82.6% [2].

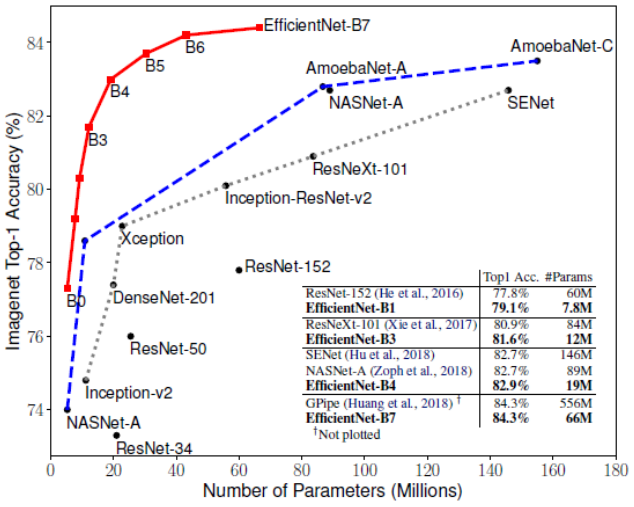

Fig. 2. Model Size vs. ImageNet Accuracy [2]

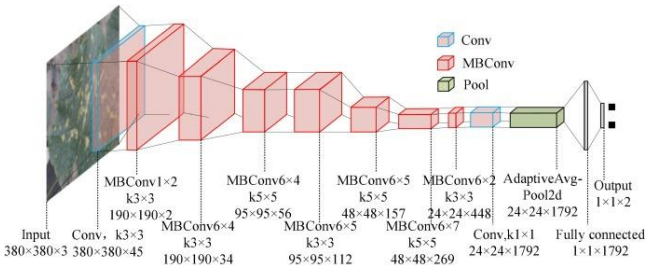

Fig. 3. EfficientNet-B4 Architecture [3]

III. THE APPROACH

A colored image is composed of three channels red, green, and blue. A grayscale image has only one channel. In both the cases, any pixel value range between 0 and 255. We know that a neural network creates a relationship between the input and the output. For the colorization problem, if we choose RGB color space, input to the neural network will be a grayscale image, and the network must predict three channels. Considering LAB color space, where L is the grayscale image same as the network input and a, b are the color spectra green-red, blue-yellow respectively. In this case, the network has to predict only two channels, i.e., a and b. Hence, we have considered LAB color space for the image colorization problem [4].

A. Architecture

Transfer learning has become very popular and is extensively used for various problems. We used transfer learning in our approach, where the pre-trained network acts as a feature extractor and helps the actual network understand what it is coloring instead of blindly predicting values for color channels.

We have considered an encoder-decoder network architecture for the proposed method [5]. The encoder and decoder together perform image reconstruction. The encoder extracts features of the input grayscale image whereas the decoder uses these features to reconstruct the image but a colorized one as we target the decoder to do so. Our network consists of two paths, one path has the EfficientNet-B4 pre-trained network, and the second path has an encoder-decoder network.

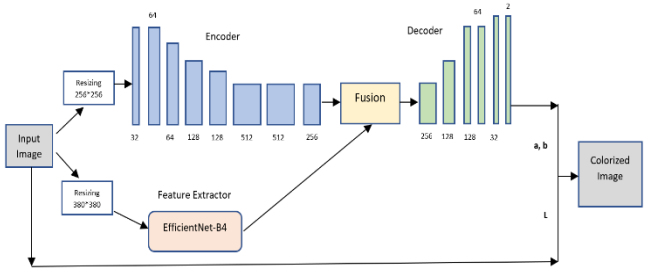

Fig. 4. Architecture of proposed method

EfficientNet-B4 is pre-trained on the ImageNet dataset. The input to the network is a grayscale image. High-level features of the input are initially extracted. The input features are then passed through an encoder network with pooling layers to reduce the dimensions of the feature vector produced. Inputs to both the pre-trained network and encoder are the same grayscale images. The classification layer of the pre-trained network is then merged with the output of the encoder. The decoder network consists of a combination of convolutional layers and upsampling layers. Upsampling in the decoder network reconstructs the image with original dimensions and two channels.

B. Datasets and Pre-Processing

The paintings dataset and Common Objects in Context (COCO) dataset are used to train the model. The paintings dataset is a small and compact dataset that is a part of the Art Images dataset from Kaggle [6]. COCO is a large-scale object detection, segmentation, and captioning dataset having several features [7]. A sample of 5000 images from this large dataset has been taken. The reason for choosing COCO is because of its diverse categories. The test dataset chosen has a mixture of paintings, COCO and out-of-dataset images.

Input images to the encoder are resized to 256×256 to have equal dimensions. The same input images to the EfficientNet-B4 are resized as it accepts images of size 380×380. Input to a neural network is generally normalized, which speeds up the learning and helps converge faster. Therefore, we normalized the image's pixel values.

The dataset is divided into train and validation sets. Images are first converted from RGB to LAB color space, and channel 'L' is extracted, which is the grayscale input image. Batches of these images are created and fed to an image data generator. This data generator takes in an image and applies specified transformations like flip, zoom, etc., thus producing more images from a single image. This helps in regularizing the network and prevents overfitting. Feature vectors are extracted from the pre-trained EfficientNet-B4 model for every image.

C. Loss and Performance Metrics

Mean Squared Error is considered as the loss parameter. Checkpoints are created to monitor the loss and save the best model. The learning rate is reduced whenever there is no change or when there is an increase in the loss.

$$C(X,\theta) = \frac{1}{2HW} \sum_{k\in\{a,b\}} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(X_{k_{i,j}} - \tilde{X}_{k_{i,j}}\right)^2 \quad (1)$$

The above equation is the mean squared error between actual and predicted pixel values where X is the image, H is the height and W is the width of the image, θ is the model parameters, $X_{k_{i,j}}$ and $\tilde{X}_{k_{i,j}}$ are pixel values of actual and predicted images, respectively.

The quality of colorization output can be perceived visually, but a quantitative measure is necessary. We considered peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) as the performance metrics to measure the quality of colorized images. PSNR is used to measure the quality of the reconstructed image. The signal in this case is the original data, and the noise is the error introduced by the colorization process. SSIM is a technique for predicting the perceived quality of digital television, cinematic pictures, and other digital images and videos. It is primarily used for measuring the similarity between two images.

The model developed is trained on both the datasets and is tested on the chosen test dataset. The PSNR and SSIM values are tabulated. We have chosen three other existing algorithms for comparing our results with them. The selected algorithms are 'Let there be Color' [8], 'Colorful Image Colorization' [9], and 'Image Colorization using Generative Adversarial Networks' [10]. These algorithms are trained on the paintings dataset, and the model is tested on the chosen test dataset.

IV. Experiments and Results

The model has been trained, and the parameters are tuned to perform better and prevent overfitting. Final model parameters, activation functions and optimizer used, loss parameter, and number of epochs are tabulated in Table 1.

Table 1. Final model parameters.

Parameter	Value
Internal activation function	ReLU
Output activation function	Tanh
Optimizer	Adam
Loss	Mean Squared Error
Batch size	20
Number of epochs	100

The proposed model was first trained with the paintings dataset and was tested on the test dataset, which is a mixture of paintings, COCO and out-of-dataset images. The validation accuracy and loss are shown in Table 2.

Table 2. Training results of the proposed model.

Training Dataset	Runtime (Hrs)	MSE	Accuracy
Paintings	3.57	0.0035	88.5
COCO	5.33	0.119	64.89

Model trained on the paintings dataset had better accuracy of 88.5% compared to the accuracy of the model trained on the COCO dataset. The result of colorization is shown in Fig. 5. The resulting images, when trained on the paintings dataset, produced better colors and looked like the original images, whereas other test images looked mostly brown. This might be because of the dataset size used. The model trained on COCO dataset also produced images that are brownish. This is because the dataset contains diverse categories of images. In the case of the paintings dataset, the images belonged to a single category and the model was able to learn the underlying pattern better.

Table 3. PSNR and SSIM values of 8 images.

Training Dataset	PSNR (dB)	SSIM (%)
Paintings	22, 20, 23, 23, 10, 15, 10, 7	98, 97, 97, 99, 88, 76, 65, 82
COCO	8, 7, 10, 11, 6, 13, 16, 15	75, 68, 77, 83, 81, 73, 69, 65

Performance metrics like PSNR and SSIM can help measure the quality of model output quantitatively. The values of these two metrics for eight images shown in Fig. 5 are tabulated in Table 3. The PSNR and SSIM values on the resulting test images also show that the model performed well on test images from the paintings dataset.

Table 4. Comparison of EfficientNet-B4 with other chosen algorithms.

Method	SSIM (%)
EfficientNet-B4 (Proposed method)	98, 97, 97, 99, 88, 76, 65, 82
Let there be Color	96, 96, 96, 99, 92, 89, 89, 91
Colorful Image Colorization	93, 92, 87, 96, 69, 94, 82, 89
Image Colorization using Generative Adversarial Networks	92, 89, 89, 94, 69, 90, 75, 77

The three existing models have been trained on the paintings dataset and the performance is compared with the proposed model. All four methods, including EfficientNet-B4, produced good SSIM on the paintings test images as shown in Table 4. 'Let there be color' and 'Colorful image colorization' produced good results on out-of-dataset test images. 'Image colorization using GANs' and EfficientNet-B4 did not produce better results when compared to the other two methods.

V. Conclusion

Our proposed method, which uses EfficientNet-B4 pre-trained model for feature extraction, was able to colorize the images from the painting dataset well. However, it produced brown images that are out of this dataset. When trained on the COCO dataset, the model did not perform well as the dataset is diverse with images from several categories. To make the model more generalized, the training dataset size has to be increased, model complexity must be enhanced and should be trained on a greater number of epochs.

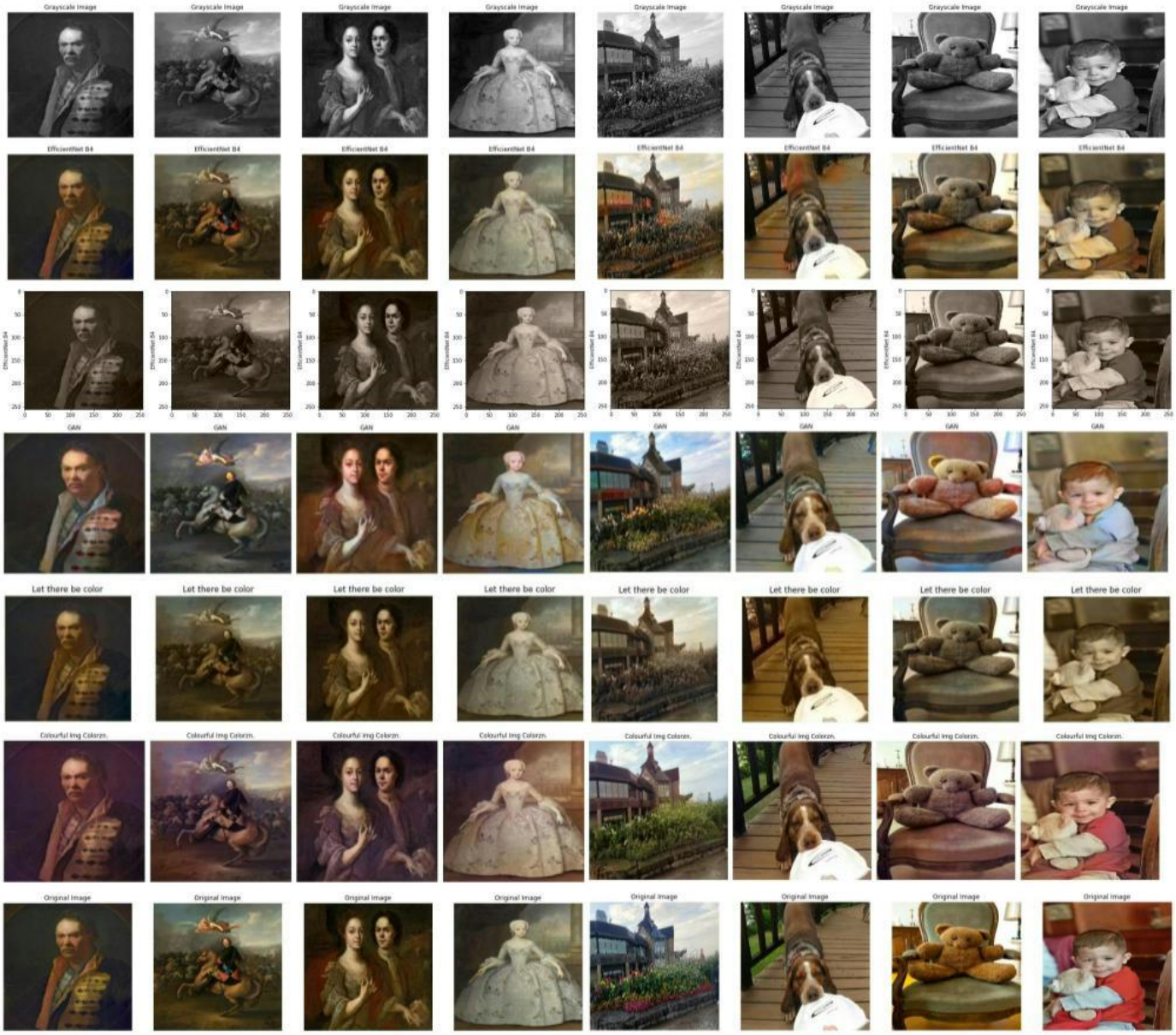

Fig. 5 Colorization Results

The performance metrics chosen finds similarity between the original and colorized image, but this is sometimes not ideal. For example, the eighth colorized image produced by GAN model in Fig. 5, produced colors that are visually appealing however, the colors are different from the original image which led to a low SSIM score.

REFERENCES

[1] O. G. Yalcin, "Towards Data Science," 23 September 2020. [Online]. Available: https://towardsdatascience.com/4-pre-trained-cnn-models-to-use-for-computer-vision-with-transfer-learning-885cb1b2dfc.

[2] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *CoRR*, vol. abs/1905.11946, 2019.

[3] P. Zhang, L. Yang and D. Li, "EfficientNet-B4-Ranger: A novel method for greenhouse cucumber disease recognition under natural complex environment," *Computers and Electronics in Agriculture*, vol. 176, 2020.

[4] E. Wallner, "Medium," 29 October 2017. [Online]. Available: https://emilwallner.medium.com/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d.

[5] F. Baldassarre, D. Gonz and L. Rod, "Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2," *CoRR*, vol. abs/1712.03400, 2017.

[6] Danil, "Kaggle," 2017. [Online]. Available: https://www.kaggle.com/datasets/thedownhill/art-images-drawings-painting-sculpture-engraving.

[7] T. Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollar, "Microsoft {COCO:} Common Objects in Context," *CoRR*, vol. abs/1405.0312, 2014.

[8] S. Iizuka, E. Simo-Serra and H. Ishikawa, "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 (Proc. of SIGGRAPH 2016)*, vol. 35, pp. 110:1--110:11, 2016.

[9] R. Zhang, P. Isola and A. A. Efros, "Colorful Image Colorization," *CoRR*, vol. abs/1603.08511, 2016.

[10] K. Nazeri and E. Ng, "Image Colorization with Generative Adversarial Networks," *CoRR*, vol. abs/1803.05400, 2018.