KMHCNet: A Gray-scale Image Colorization Approach Using ResNet Encoder based UNET++ Convolutional Autoencoder

Khandaker Mobashyer Hossain *
Department of Electrical and Computer Engineering
North South University
Dhaka-1229, Bangladesh
hossain.mobashyer007@gmail.com

Abstract

We propose a novel approach to solving the colorization problem which aims to colorize a gray-scale image by using UNet++ Auto-Encoder where the Encoder is compiled using ResNet. This article also shows the effect of each variant of ResNet as well as InceptionResnetV2 and EfficientNet as potential encoders in the network. To show the advantages of using UNet++, we also created some variants replacing the UNet++ with traditional UNet. Thus our article proposes a deep convolutional network framework called KMHCNet which contains multiple variants of the network each having different combination of architecture, encoder, color-space in which to calculate loss on, embedding used before calculating loss and loss function used. Each variants provides us a different overall network with their own advantages and flaws. In this article we showed the results of 16 variants of the KMHCNet after training them on the same Dataset for differing number of epochs. Experiments have shown that most variants of our proposed network can provide pretty good result. To our best apprehension, no prevailing papers or research studies have mentioned this approach of using a segmentation model with variant encoders and using MS-SSIM as a potential loss function to colorize gray-scale images.

Keywords Colorization · Deep Learning · CNN · FCN · UNet · UNet++ · ResNet · InceptionResnetV2 · MS-SSIM · More

1 Introduction

Before smart phones, digital cameras and the invention of the first camera capable of taking colored picture, people used to preserve their memory in the form of black and white images like the ones in Figure 1. But as time passed and the era of preservation of memory in colorful form approached we started to forget the beauty of these past memories.

So, to make these memories relevant to current generation and bring lost memories back for the older generation, colorization of black & white or gray-scale image has become more than essential. It's not just a cultural need but also a way of preserving history for the future generation.

We acknowledge this fact and thus we tried to bring color to the old era through colorization of gray-scale images. For this purpose we propose a solution of colorization by using a segmentation model. Our model is a Convolutions Auto-Encoder having a CNN as the encoder. We know that CNN have a strong capacity for Learning[2].

Inspired by all the work being done in the field of Colorization[3, 4, 5, 6, 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] we decided to try something new with what we already have. So, we tried using a segmentation auto-encoder like UNet[17] or UNet++[18] to build, Multi-scale Structured Similarity Index Measurement(MS-SSIM)[19] as a loss function to train and Adam optimizer to Optimize our model.

UNet++ is already a very robust architecture capable[20, 21, 22] a lot of task related to image segmentation for example, white blood cell segmentation[23], congestive heart failure diagnosis[24], breast cancer nuclei segmentation[25] etc. In short, UNet++ was the perfect architecture for what we wished to archive.

*https://github.com/clownprincejoker/KMHColorNet

Figure 1: Legacy Black & White Images

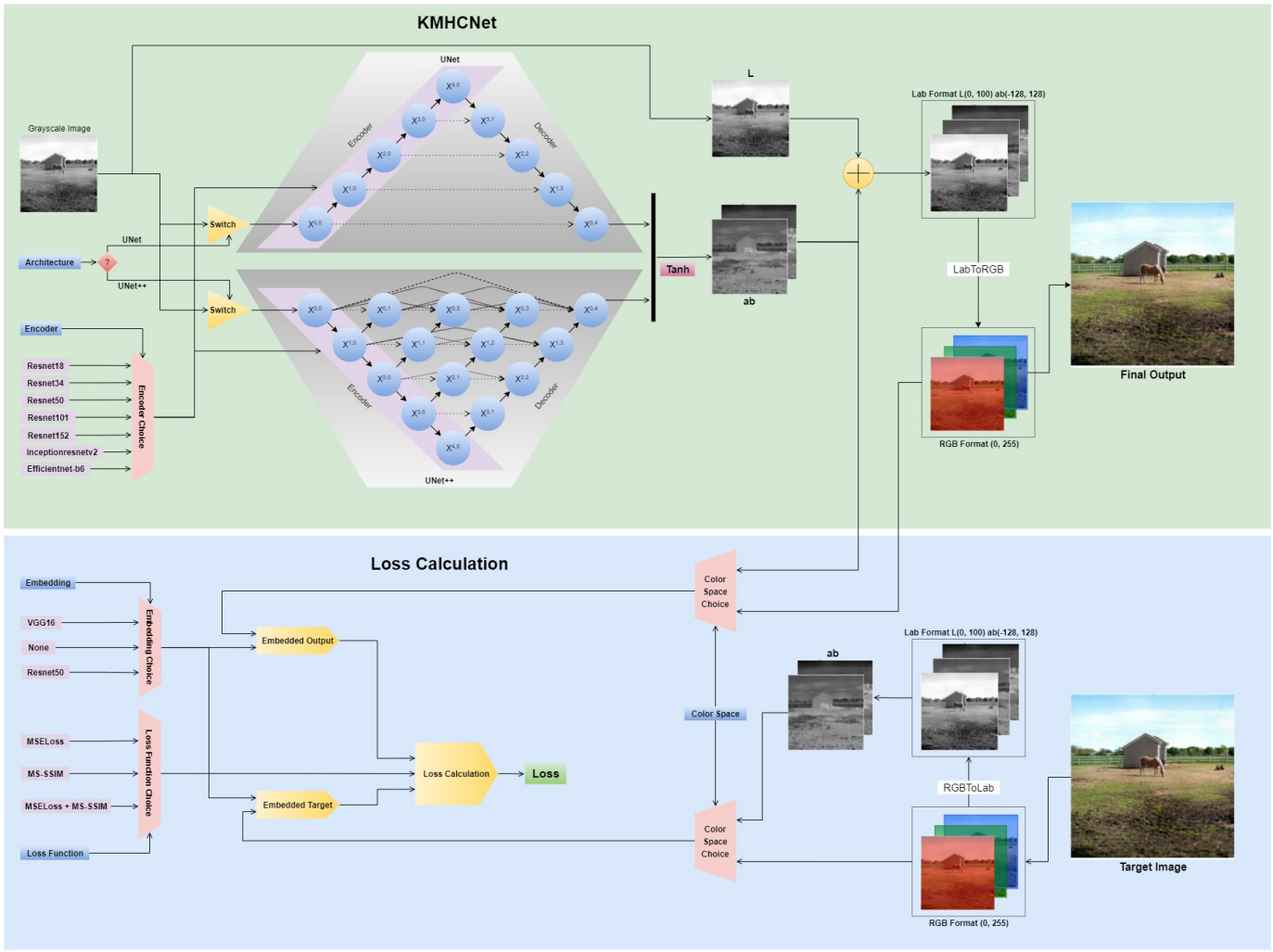

Figure 2: Our proposed KMHCNet showing all possible variants of the model with Loss Calculation Process Diagram

As for our choice of encoder, we tried multiple classification model like VGG16[26], InceptionResnetV2[27], EfficientNet[28] and finally ResNet[29] which turned out to be the best choice for our intended task.

Thus, our final model provides not one but multiple strategies to produce colorize images from given black & white or gray-scale images, most of which gives pretty good result based on the resultant images shown in Figure 4 and Figure 5.

Figure 3: The 1st Image shows the ***L*** channel as input tensor and the 2nd and 3rd image shows the ***ab*** channel as target tensor

Variant Name	Architecture	Encoder	Colo Space	Loss Function	Embedding
$KMHCNet_1$	UNet++	Resnet152	LAB	MS-SSIM	None
$KMHCNet_2$	UNet++	Resnet50	LAB	MS-SSIM	None
$KMHCNet_3$	UNet++	Resnet152	RGB	MS-SSIM	None
$KMHCNet_4$	UNet++	Resnet34	LAB	MS-SSIM	None
$KMHCNet_5$	UNet++	Inceptionresnetv2	LAB	MS-SSIM	None
$KMHCNet_6$	UNet++	Resnet18	LAB	MS-SSIM	None
$KMHCNet_7$	UNet++	Efficientnet-b6	LAB	MS-SSIM	None
$KMHCNet_8$	UNet++	Resnet101	LAB	MS-SSIM	None
$KMHCNet_9$	UNet	Inceptionresnetv2	LAB	MS-SSIM	None
$KMHCNet_{10}$	UNet	Resnet152	LAB	MS-SSIM	None
$KMHCNet_{11}$	UNet++	Resnet152	LAB	MSELoss	None
$KMHCNet_{12}$	UNet++	Resnet152	LAB	MSELoss+MS-SSIM	Resnet50
$KMHCNet_{13}$	UNet++	Resnet50	RGB	MS-SSIM	None
$KMHCNet_{14}$	UNet++	Resnet152	LAB	MSELoss+MS-SSIM	Vgg16
$KMHCNet_{15}$	UNet++	Resnet152	LAB	MSELoss	Vgg16
$KMHCNet_{16}$	UNet++	Resnet152	LAB	MSELoss	Resnet50

Table 1: Specifications of the 16 Variants of our KMHCNet. Every variant of our KMHCNet model is created with the a Architecture based the a given Encoder which provides us with the ***ab*** channel of an Lab Image where ***L*** channel is given as an input. The Color Space decides if the loss calculation should be performed on the ***ab*** channel of the Lab Image or on the ***rgb*** channel after converting the LAB Image to RGB format, The Loss Function is used to calculate the loss on the raw output or an embedding of the output given the Embedding used

2 Related Work

Chen et al.[6] tried to bring old Chinese black and white movies to life by using a colorization model which combines two Convolutional Neural Networks and uses multi-scale convolution kernels to get better spatial consistency. They explained how current datasets used in the colorization networks are not applicable to colorizing images from Chinese black and white films. The main reason being that the objects in those films are very different from today's. To address this, Chen et al. extracted a large number of images from Chinese color films of the last century as a training dataset. The experiments conducted by them demonstrated that their model can obtain pretty good results of colorizing images from Chinese black and white films.

Another traditional approache to image colorization consists of propagating colored user-specific scribbles to the whole image. Levin et al.[3] proposed an optimization-based framework for colorization. He did that by solving a quadratic cost function which he derived from differences of intensities between a pixel with its neighboring pixels.

Welsh et al.[30] proposed a colorization approach which transfers the color from a reference image to the target image. Basically, local descriptors[31] are used to create mapping functions between reference images and target images. Manual intervention is added in later researches.

Figure 4: Colorized Output of 5 Test Images from test-set for every variants of KMHCNet

During past few years, a lot of automatic methods [2, 6, 16, 32] have been proposed. Desphande et al.[32] proposed colorizing an image by minimizing a quadratic objective function trained with a LEARCH framework. Cheng Z. et al.[8] present a deep neural network that leverages variety of features, including DAISY feature[33], semantic feature, and the image patch feature.

Iizuka, Serra et al.[2] present a network which combines both global priors (the classification of images) and local image features. Larsson et al.[34] develop a network based on VGG-16 with the classification layer discarded and predict a color probability distribution for each pixel.

UNet was created for the purpose of image segmentation, which was further improvised in the form of UNet++ by making all of its nodes densely connected. Zongwei et al.[18] have used UNet++ architecture for Medical Image Segmentation. Yan et al.[23] build a white

Figure 5: Colorized Output of 5 Old Black and White Images for every variants of KMHCNet

blood cell segmentation network based on UNet++ and ResNet. Lei et al. [24] improved the already robust UNet++ to detect Congestive Heart Failure Using Short-Term RR Intervals. Wang et al.[25] used UNet++ the improve a method of Breast Cancer Nuclei Segmentation.

The KMHCNet model we proposed is partially motivated by the work done using UNet and UNet++ architecture. Since. these autoencoders can be used for other tasks such as colorization. For example, Yide et al.[35] proposed such a approach where they designed their model in YUV instead of RGB color space, with an objective function that is appropriate to the coloring problem and can capture a wide range of colors. A large number of experimental results on LFW and LSUN datasets confirm the method's superiority. We expended on their idea of using UNet++ for colorization by adding MS-SSIM[19] as a loss function and using pretrained encoders. Our model also uses Lab instead of their proposes YUV color Space.

Variant Name	Test Accuracy	Test Loss	Epochs Trained	Trainable Parameters
$KMHCNet_1$	94.39%	18.40%	43	83.62 M
$KMHCNet_2$	94.69%	16.76%	39	48.98 M
$KMHCNet_3$	94.07%	5.93%	38	83.62 M
$KMHCNet_4$	94.04%	19.07%	33	26.07 M
$KMHCNet_5$	95.38%	14.16%	27	71.08 M
$KMHCNet_6$	94.93%	15.69%	26	15.96 M
$KMHCNet_7$	94.44%	17.73%	25	43.33 M
$KMHCNet_8$	95.51%	13.84%	23	67.97 M
$KMHCNet_9$	95.17%	14.60%	20	62.03 M
$KMHCNet_{10}$	94.75%	16.06%	20	67.15 M
$KMHCNet_{11}$	93.03%	73.45%	17	83.62 M
$KMHCNet_{12}$	93.39%	17.24%	13	83.62 M
$KMHCNet_{13}$	94.32%	5.68%	12	48.98 M
$KMHCNet_{14}$	93.73%	11.12%	10	83.62 M
$KMHCNet_{15}$	92.85%	1.47%	10	83.62 M
$KMHCNet_{16}$	92.68%	12.54%	9	83.62 M

Table 2: Results and and parameters of the 16 variants of our KMHCNet. We can observe that not all variants were trained for the same No. of epoch. Some variants converged faster so we did not need to train any further which can be seen from Figure 7. Some variants didn't show any sign of improving so we stopped training it further. We can also see the Test Loss doesn't show the correct accuracy of the model variant

3 Methodology

3.1 Dataset

For this research we used the images from Image Colorization[36] Dataset created by Mario Matos. But we didn't us it directly in our model training. First we had to customize the Dataset content to fit our model.

We discarded all the grayscale image already pressent in the dataset so that it doesn't effect the model. Than we converted all the RGB format images into Lab images. The ***L*** channel of a Lab image is mainly the grayscale version of that image. So, we didn't have to transform any images to grayscale. Thus from this Lab Image the ***L*** channel can be considered as the input and the ***ab*** channel to be the target of the model output. The ***L*** channel pixel values and the ***ab*** channel pixel values are kept between (-1 to 1).

So, our final Dataset consists of an input tensor having 1 channel and a target tensor having 2 channels, shown as in Figure 3.

3.2 Proposed Network

The core of our model comes from Segmentation Models Pytorch (SMP)[1] a python module. We use SMP to create the 1st part of our network with a particular model architecture and encoder from over 9 available model architectures (for binary and multi class segmentation) and 104 available encoder. All of the encoders have been pretrained using Imagenet[37] Dataset which gives us a higher accuracy and faster convergence. All 16 variants of our KMHCNet model uses pretrained encoder which reduces the convergence time which can observed from Figure 6 and Figure 7

Our model uses certain keywords when deciding which variant of the model to use. From Table 1 we can see all 16 variant of the model has an Architecture, an Encoder, a Color Space, an Embedding and a Loss Function. Depending on which combination is picked more than 16 variants can be created. But because of time constrains and getting good results form already tried variants we stopped at 16 variant of the model. But not all of these variants perform well in Test images in contrast with Test Accuracy shown in Table 2. From Figure 4 we can observe $KMHCNet_{12}$, $KMHCNet_{14}$, $KMHCNet_{15}$, $KMHCNet_{16}$ has some color temperature issue and $KMHCNet_9$ has some spatial distortion issue. The same issues can be observed in old black & white real life images shown in Figure 5.

$$\hat{y} = f(x), \qquad f(x) = \begin{cases} Tanh(UNet(x)) & arch = Unet \\ Tanh(UNet++(x)) & arch = Unet++ \end{cases} \tag{1}$$

$$T(p) = [((x * 100) + 50), ((p * 128) + 128)] \tag{2}$$

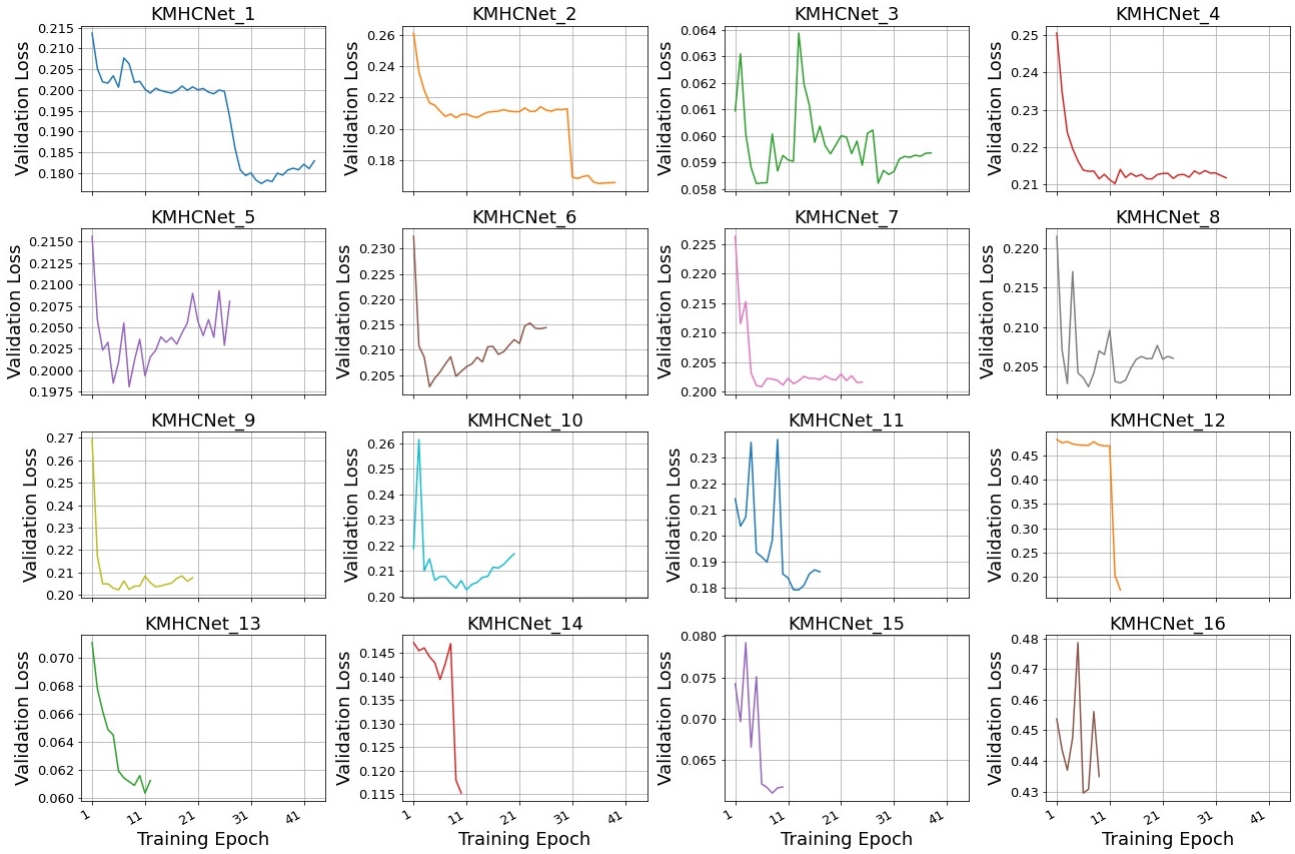

Figure 6: Validation Loss plotted for every variants of KMHCNet shows that some variants converge faster than the others. Some fall in the curve can also be observed which occurs when the hyper parameters are updated in favour of the model improvement after some period of stagnancy in the loss reduction.

$$K_{LTR}(a) = LTR(T(a)) \tag{3}$$

As illustrated in Figure 2 our KMHCNet model starts by picking between UNet or UNet++[35] as the model architecture and an Encoder for that architecture from among ResNet18, ResNet34, ResNet50, ResNet101, ResNet152[29, 23], InceptionResnetV2[27] or EfficientNet-B6[28]. The output we get from this network is than brought down between -1 and 1 by using Tanh activation function. This step is done because the ***ab*** channel of a Lab image has value restriction of (-128, 128). So, the activation output can be easily scaled up to fit that restriction. This process is shown is Equation 1.

The output from the activation function is than concatenated with the given input which is considered to be ***L*** channel of Lab image Using Equation 2 to form the complete Lab image. From this point can get the final colored image output as a RGB image format after we convert it using Kornia[38] which is a differentiable computer vision library for PyTorch. The conversion function will be similar to Equation 3.

The color space1 keyword decides in which format of the image the the loss function will be calculated in. Equation 4 shows how the output of choosing a color space might look like.

Thus, we can get the final output of our model as a Lab Image or a RGB Image. But for simplicity's sake we decided to keep the final output as RGB image as shown in Figure 2.

3.3 Loss Calculation

In our model we have used two types loss functions MSELoss and MS-SSIM loss. We also showed some variants which used the average of both loss functions.

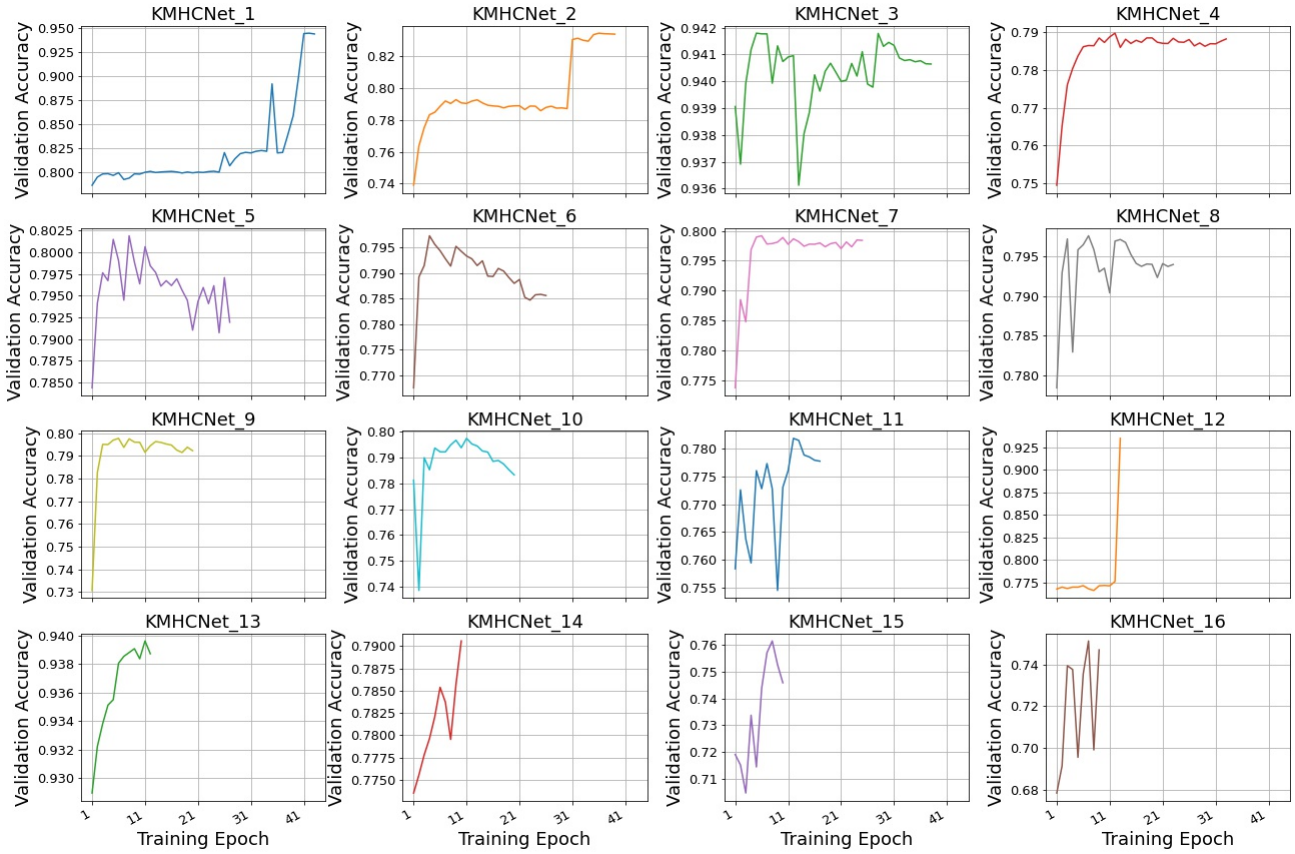

Figure 7: Validation Accuracy plotted for every variants of KMHCNet shows that some variants perform batter and gives higher accuracy than others. Some spike in the curve can also be observed which occurs when the hyper parameters are updated in favour of the model improvement since the accuracy becomes stagnant.

But before we can calculate the loss our model must do some additional tinkering based on the color space and embedding status of a variant. Currently our model only supports VGG16[26] and ResNet50[29] as Embedding. Figure 2 shows how the status of this two keywords can change the how the final loss of that variant is calculated. Equation 4 explains how the color space decides the output of G. Same way the Equation 5 explains how the choice of embedding effects the output of E.

As for the loss calculation, Equation 8 shows the effect of loss keyword on the choice of loss function. Final loss can be calculated using Equation 6 which is the MSELoss or L_2 Loss or using Equation 7 which represent the MS-SSIM[19] (Multi-Scale Structured Similarity Index Measurement) loss. The final loss can also be taken as the average of both of these losses.

$$G(u) = \begin{cases} T(u) & colorSpace = Lab \\ K_{LTR}(u) & colorSpace = RGB \end{cases} \tag{4}$$

$$E(j) = \begin{cases} ResNet50(G(j)) & embedding = resnet50 \\ VGG16(G(j)) & embedding = vgg16 \\ T(j) & embedding = None \end{cases} \tag{5}$$

3.3.1 Mean Squared Error or L_2 Loss

In statistics, the mean squared error(MSE) or mean squared deviation(MSD) of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to

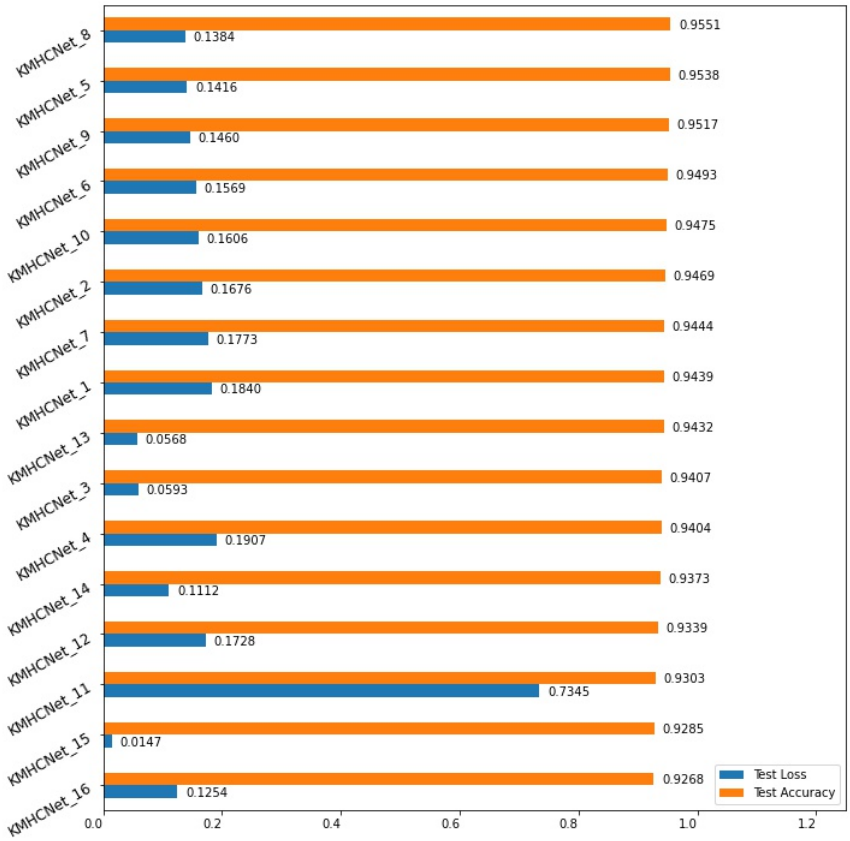

Figure 8: Test Accuracy and Loss for every variants of KMHCNet plotted in order of Accuracy. From the bar chart we can observe how every variant differs little in term of Accuracy mostly between *0.92* to *0.96*. But the difference becomes significant in terms of loss which occurs because the Accuracy for every variant is Calculated using the same metric which is MS-SSIM[1]. But the loss function differs for some variants having one of MSELoss, MS-SSIM Loss or both.

the expected value of the squared error loss. The fact that MSE is almost always strictly positive is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.

$$L_2\left(E\left(y\right), E\left(\hat{y}\right)\right) = \frac{1}{n}\sum_{i=0}^{n} ||E\left(y\right)_i - E\left(\hat{y}\right)_i||^2 \tag{6}$$

The MSE is a measure of the quality of an estimator. As it is derived from the square of Euclidean distance, it is always a positive value with the error decreasing as the error approaches zero.

3.3.2 MS-SSIM as a Loss Function

The structural similarity index measure (SSIM) is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference.

$$L_{MS-SSIM}\left(E\left(y\right), E\left(\hat{y}\right)\right) = 1 - \left[l_M\left(E\left(y\right), E\left(\hat{y}\right)\right)^{\alpha_M} \cdot \prod_{j=1}^{M}\left[c_j(E\left(y\right), E\left(\hat{y}\right))\right]^{\beta_j}\left[s_j(E\left(y\right), E\left(\hat{y}\right))\right]^{\gamma_j}\right] \tag{7}$$

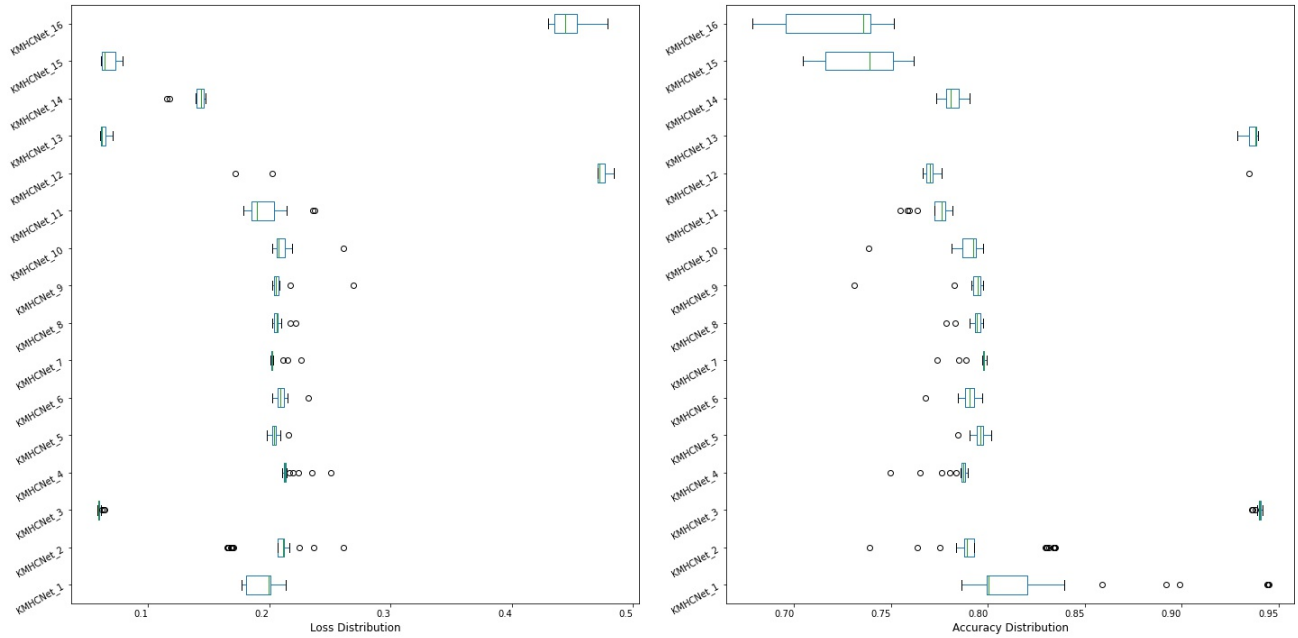

Figure 9: The Box plots shows that both in terms of Accuracy and Loss calculation some KMHCNet variant displays outliers which can occur for either of two reasons. *a.* The hyper parameters or *b.* The Loss and Accuracy function were updated mid-training in an effort to increase model performance.

SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. The difference with other techniques such as MSE or PSNR is that these approaches estimate absolute errors. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or "texture" in the image.

A more advanced form of SSIM, called Multi-Scale SSIM (MS-SSIM)[19] is conducted over multiple scales through a process of multiple stages of sub-sampling, reminiscent of Multi-Scale processing in the early vision system. It has been shown to perform equally well or better than SSIM on different subjective image and video databases.

Since MS-SSIM gives us the similarity index between two images, we can easily get the dissimilarity index by subtracting MS-SSIM from 1. which gives us the $L_{\text{MS-SSIM}}$ from Equation 7. This represent the difference between two images.

$$Loss\left(E\left(y\right),E\left(\hat{y}\right)\right)=\begin{cases}L_2\left(E\left(y\right),E\left(\hat{y}\right)\right) & loss=MSELoss\\ L_{MS-SSIM}\left(E\left(y\right),E\left(\hat{y}\right)\right) & loss=MS-SSIM\\ \frac{L_2(E(y),E(\hat{y}))+L_{MS-SSIM}(E(y),E(\hat{y}))}{2} & loss=MSELoss+MS-SSIM\end{cases} \tag{8}$$

3.4 Optimization and Learning

As the optimizer to optimize our model and improve it we used Adam optimizer[39]. Adam is a great choice for optimization when we have large amount of data. It also works very well with Convolutional Neural Networks.

Adam is a Algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal re-scaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods.

4 Experimental Results and Discussion

we assess the graphics aspect of the every variant of our model, evaluating the perceptual realism of our colorization effect, along with other measures of accuracy. We compare every variants of our model in terms of Loss and Accuracy as well as Artistic integrity. Finally, in Section 4.3, we show qualitative examples on legacy black and white images.

4.1 Test Result

From Table 2 we can see the Accuracy and Loss found on the Test Set which consists over 1000 images. We see how most of the KMHCNet variants gives us a accuracy of over 90%. The accuracy metric we used is the MS-SSIM[19]. This metric gives us a very accurate indication when two images are compared since it considers not only the pixel value but also luminance, contrast and structure of an image.

We can also see how the Loss value becomes quite divergent for every variant pertaining to the fact that we used different Loss function depending on the variant. We can Also see the No. of trainable parameters for each variant. From Figure 6 we can see the validation loss of each variant and how and when the loss becomes the minimum. some variant also produce unrealistic loss which in turn affects the Accuracy of the model.

From Figure 7 we can see how each variant of produces validation accuracy. How some variant takes longer to train and some variants can easily achieve high accuracy with low training epochs. We can get a better understanding of Loss and Accuracy on the Test-Set by looking at Figure 8. From this we get a better understanding of the Test Score other than looking at Test Image results. Figure 9 shows how validation Loss and Accuracy are changing and some of the discrepancies and outliers during each epochs.

4.2 Test Image Accuracy

From Figure 4 we can see how well each variant of our KMHCNet model performs on modern Images. In the beginning of the training phase we encountered problems with color temperature, but as we improved the hyper parameters and overall design structure this problem was easily solved. Currently different variants of our KMHCNet model is capable producing bright and aesthetically pleasing Colorful Images as can be seen from Figure 4.

But, the main question still remains how well can it perform on real black and white images. Thus, we used every variant of our Model to colorize real life old images. The result for this can be seen in Figure 5.

4.3 Legacy Black and White Image

Since our model was trained using "fake" grayscale images generated by stripping ab channels from color photos, we also ran our method on real legacy black and white photographs, as shown in Figure 5. One can see that our model is still able to produce good Colorful Image even from old real black and white Images. Still the difference can be observed where the color quality fails to meet the modern expectations.

To bring out brighter colors from these Legacy images we can try including some Legacy colored photos taken just after the creation of color cameras to our Dataset. Even though unproven until now, we believe our model can detect the difference between old images taken with analog cameras from modern high resolution images.

5 Conclusion

While image colorization is an interesting task when it comes to computer graphics, it can also become a difficult pixel prediction problem. Here we have shown that colorization with UNet++ auto-encoder having ResNet as the base encoder can come closer to producing results indistinguishable from real color photos. Our approach not only shows the remarkable affects of ResNet but it also show InceptionResnetV2 and EfficientNet-B6 as potential encoders for future studies using KMHCNet. Although only trained for the purpose of colorization, we believe our model is robust enough to be useful for object classification, detection, and segmentation.

References

[1] Pavel Yakubovskiy. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2020.

[2] 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 Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4):110:1–110:11, 2016.

[3] Levin Anat, Lischinski Dani, and Weiss Yair. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 689–694, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 9781450378239. doi:10.1145/1186562.1015780. URL https://doi.org/10.1145/1186562.1015780.

[4] Bin Bao and Hongbo Fu. Scribble-based colorization for creating smooth-shaded vector graphics. *Computers Graphics*, 81:73–81, 2019. ISSN 0097-8493. doi:https://doi.org/10.1016/j.cag.2019.04.003. URL https://www.sciencedirect.com/science/article/pii/S0097849319300445.

[5] Vincent Billaut, Matthieu de Rochemonteix, and Marc Thibault. Colorunet: A convolutional classification approach to colorization. *CoRR*, abs/1811.03120, 2018.

[6] Y. Chen, Y. Luo, Youdong Ding, and Ting Yu. Automatic colorization of images from chinese black and white films based on cnn. *2018 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 97–102, 2018.

[7] Zhang Richard, Isola Phillip, and Efros Alexei A. Colorful image colorization. In Leibe Bastian, Matas Jiri, Sebe Nicu, and Welling Max, editors, *Computer Vision – ECCV 2016*, pages 649–666, Cham, 2016. Springer International Publishing.

[8] Cheng Zezhou, Yang Qingxiong, and Sheng Bin. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[9] Qu Yingge, Wong Tien-Tsin, and Heng Pheng-Ann. Manga colorization. *ACM Trans. Graph.*, 25(3):1214–1220, July 2006. ISSN 0730-0301. doi:10.1145/1141911.1142017. URL https://doi.org/10.1145/1141911.1142017.

[10] Wan Shaohua, Xia Yu, Qi Lianyong, Yang Yee-Hong, and Atiquzzaman Mohammed. Automated colorization of a grayscale image with seed points propagation. *IEEE Transactions on Multimedia*, 22(7):1756–1768, 2020. doi:10.1109/TMM.2020.2976573.

[11] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. Colorization transformer. *CoRR*, abs/2102.04432, 2021.

[12] Xu Zhongyou, Wang Tingting, Fang Faming, Sheng Yun, and Zhang Guixu. Stylization-based architecture for fast deep exemplar colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] Shamsabadi Ali Shahin, Sanchez-Matilla Ricardo, and Cavallaro Andrea. Colorfool: Semantic adversarial colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[14] Vitoria Patricia, Raad Lara, and Ballester Coloma. Chromagan: Adversarial picture colorization with semantic class distribution. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[15] Lee Junsoo, Kim Eungyeup, Lee Yunsung, Kim Dongjun, Chang Jaehyuk, and Choo Jaegul. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Poterek Quentin, Herrault Pierre-Alexis, Skupinski Grzegorz, and Sheeren David. Deep learning for automatic colorization of legacy grayscale aerial photographs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:2899–2915, 2020. doi:10.1109/JSTARS.2020.2992082.

[17] Zyuzin Vasily and Chumarnaya Tatiana. Comparison of unet architectures for segmentation of the left ventricle endocardial border on two-dimensional ultrasound images. In *2019 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 110–113, 2019. doi:10.1109/USBEREIT.2019.8736616.

[18] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018.

[19] Wang Z., Simoncelli E.P., and Bovik A.C. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi:10.1109/ACSSC.2003.1292216.

[20] Jia Yutong, Liu Lei, and Zhang Chenyang. Moon impact crater detection using nested attention mechanism based unet++. *IEEE Access*, 9:44107–44116, 2021. doi:10.1109/ACCESS.2021.3066445.

[21] Zhang Xiuwei, Yue Yuanzeng, Gao Wenxiang, Yun Shuai, Su Qian, Yin Hanlin, and Zhang Yanning. Difunet++: A satellite images change detection network based on unet++ and differential pyramid. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2021. doi:10.1109/LGRS.2021.3049370.

[22] Zhao Tuo, Zhao Yunxin, Wang Shaojun, and Han Mei. Unet++-based multi-channel speech dereverberation and distant speech recognition. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5, 2021. doi:10.1109/ISCSLP49672.2021.9362064.

[23] Yan Lu, Xuejun Qin, Haoyi Fan, Taotao Lai, and Zuoyong Li. Wbc-net: A white blood cell segmentation network based on unet++ and resnet. *Applied Soft Computing*, 101:107006, 2021. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2020.107006. URL https://www.sciencedirect.com/science/article/pii/S1568494620309455.

[24] Lei Meng, Li Jia, Li Ming, Zou Liang, and Yu Han. An improved unet++ model for congestive heart failure diagnosis using short-term rr intervals. *Diagnostics*, 11(3), 2021. ISSN 2075-4418. URL https://www.mdpi.com/2075-4418/11/3/534.

[25] Wang Haonan, Li Yinhan, and Luo Zhiyi. An improved breast cancer nuclei segmentation method based on unet++. In *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, ICCAI '20, page 193–197, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450377089. doi:10.1145/3404555.3404577. URL https://doi.org/10.1145/3404555.3404577.

[26] Qassim Hussam, Verma Abhishek, and Feinzimer David. Compressed residual-vgg16 cnn model for big data places image recognition. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 169–175, 2018. doi:10.1109/CCWC.2018.8301729.

[27] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2. *CoRR*, abs/1712.03400, 2017.

[28] Tan Mingxing and Le Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/tan19a.html.

[29] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *CoRR*, abs/1603.08029, 2016.

[30] Welsh Tomihisa, Ashikhmin Michael, and Mueller Klaus. Transferring color to greyscale images. *ACM Trans. Graph.*, 21:277–280, 07 2002. doi:10.1145/566570.566576.

[31] Charpiat Guillaume, Hofmann Matthias, and Schölkopf Bernhard. Automatic image colorization via multimodal predictions. In Forsyth David, Torr Philip, and Zisserman Andrew, editors, *Computer Vision – ECCV 2008*, pages 126–139, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[32] Deshpande Aditya, Rock Jason, and Forsyth David. Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[33] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi:10.1109/CVPR.2008.4587673.

[34] Larsson Gustav, Maire Michael, and Shakhnarovich Gregory. Learning representations for automatic colorization. In Leibe Bastian, Matas Jiri, Sebe Nicu, and Welling Max, editors, *Computer Vision – ECCV 2016*, pages 577–593, Cham, 2016. Springer International Publishing.

[35] Di Yide, Zhu Xiaoke, Xin Jin, Dou Qiwei, Zhou Wei, and Duan Qing. Color-unet++: A resolution for colorization of grayscale images using improved unet++. *Multimedia Tools and Applications*, pages 1–20, 03 2021. doi:10.1007/s11042-021-10830-2.

[36] Mario Matos. Image colorization, 2020. URL https://www.kaggle.com/mariomatos/image-colorization.

[37] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR09*, 2009.

[38] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL https://arxiv.org/pdf/1910.02190.pdf.

[39] Kingma, Diederik P., and Ba. Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.