DEEP LEARNING IMAGE COLORIZATION

Nikiforos Efthymiadis(s202430), Abdulstar Shihada Kousa(s174360),
Malthe Andreas Lejbølle Jelstrup(s184291), Sixian Zuo(s210365) and Renjue Sun(s181294)

DTU Compute

ABSTRACT

The process of automatically converting black and white images to coloured ones is a multi-modal problem where various colorized outputs are valid based on a single grayscale image. In this report we investigated different architectures to tackle this problem where the architectures researched range from simple Convolutional Neural Networks to conditional Generator-Discriminator pipelines. Lastly we compared the pros and cons of each architecure and provided results based on well established metrics in the computer vision field, namely PSNR, SSIM and LPIPS.

***Index Terms*—** Grayscale Images, Colorization, L·a·b color space, CNN, GAN, CGAN, Attention U-Net

1. INTRODUCTION

Colorizing black and white images has many applications in today's world, some of them are restoring old historic photographs, helping artists colorize their sketches or even helping the film industry in various tasks. The colorization of a grayscale image is challenging as it is a multi-modal and undetermined problem. Multi-modal refers to the fact that many plausible colorization might occur from a single input of a grayscale image. Undetermined refers to the fact that having as input only one channel we are trying to predict multiple channels where there is no unique solution. Due to the nature of the problem researchers are still coming up with new solutions each with different pros and cons. The first attempts in this field required user interaction to achieve satisfying results [1][2]. As deep learning techniques are becoming more and more dominant in computer vision new deep learning architectures were created for the process of image colorization [3][4][5]. Typically, when handling images, one works in the RGB color space where each pixel is comprised of 3 values representing a color of Red, Green and Blue respectively. In order to avoid trying to predict 3 channels from the grayscale image we switched to the L·a·b color space. In L·a·b color space, we have again three numbers for each pixel but these numbers have different meanings. The first number (channel), L, encodes the Lightness of each pixel and when we visualize this channel it appears as a black and white image. The a and b channels encode how much green-red and yellow-blue each pixel is, respectively. When using L·a·b, we can give the L channel to the model (which is the grayscale image) and want it to predict the other two channels (a and b) and after its prediction, we concatenate all the channels and get our colorful image. But if you use RGB, you have to first convert your image to grayscale, feed the grayscale image to the model and hope it will predict 3 numbers for you which is a way more difficult and unstable task due to the many more possible combinations of 3 numbers compared to two numbers.

2. RELATED WORK

A traditional way to colorize grayscale images is to use the Image Analogies framework which is a non-parametric approach. Image analogies in Computer Graphics is to attach the same colorization characteristic to images with the same content but different textures [6]. Inspired from this, researchers found that extracting notable features from similar colored images and then conducting the colorization accordingly can produce reliable outcome [7][8]. However, the orientation of 'similar colored images' or 'related colored images' involves user interactions or some automatically retrieved algorithms. This is inconvenient and problematic.

Parametric methods constantly optimize prediction functions from large data-sets of colored images at training time, transferring the problem into either a regression problem onto continuous color space [9] or a classification problem of quantized color values [10][5]. Our CNN with classifier model also learns to classify colors but the model is sophisticated and we have a rather large data-set. Therefore the model output lies in continuous color space.

U-Net architecture is widely used in image segmentation tasks [11][12][13]. Attention U-Net is a development of the original U-Net architecture, which introduces Attention Gates(AGs) in the up-sampling process. Models trained with AGs implicitly learn to suppress irrelevant regions in an input image while highlighting salient features useful for a specific task [14]. U-Net architecture itself is capable of doing colorization tasks since it performs pixel-wise transformation and can attach colors to all pixels of a image. However, the sole U-Net architecture acts conservatively, which produces

gray-ish result even with bolder L1 loss rather than conventional L2 loss [15][16]. One of our proposed models adds a PatchGAN as a discriminator while taking Attention U-Net as a generator, which also introduces conditional GAN loss to the generator loss. This helps the model make more vibrant and qualified colorization.

3. PROPOSED METHODS

3.1. CNN Baseline

3.1.1. Architecture

Our Baseline model is a fully-convolutional network based on the ResNet-18 classifier — an image classification network with 18 layers and residual connections. The baseline outputs a colored image from an input grayscale image.

We worked with ImageNet data-set in the L·a·b color space using the following:

- inputs are 256 x 256 x 1 (the lightness channel).
- outputs are 256 x 256 x 2 (the other two channels).

ResNet-18, is the starting point for our model to extract features with a number of convolutional layers. We changed the network's initial layer so that grayscale input was accepted. We observed that incorporating global features into our network did not increase our colorization performance while using ResNet-18. Thus, we cut the ResNet-18 off after the sixth set of layers. Then we upscaled our features with deconvolutional layers. **Fig. 1** below shows our baseline colorization network architecture.

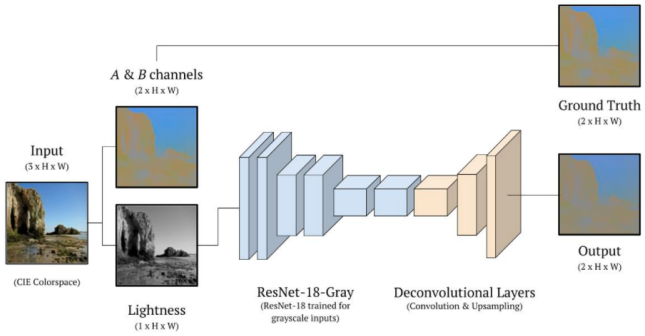

Fig. 1. A diagram of the baseline network architecture [17]

3.1.2. Loss function

Because of the problem's multi-modality — multiple plausible colored images can be derived from a single grayscale image, the choice of loss function was a little challenging. However, we trained with both mean absolute error (L1) and mean squared error (L2) loss functions, and found that mean absolute error (L1) created more qualified colorizations due to its resistance to outliers in the data-set.

3.1.3. Training

First we converted images to grayscale. Then we defined transformations for our training data, where we rescaled the training images to 224x224 before input to the initial layer of the ResNet-18. The final baseline network is trained with mean absolute error (L1) loss function as mentioned before. We optimized our loss function with the Adam optimizer using 0.0001 learning rate. We trained for 100 epochs on a NVIDIA-SMI GPU on the Google Colab Pro.

As a result, we discovered that our model prefers desaturated colors over bright colors because they're less likely to be wrong for the loss function (L1).

The progression of the final baseline model over 100 training epochs can be found in **A. APPENDIX** at **A.1**.

3.2. CNN + Classifier

3.2.1. Architecture

The implementation of this model was an attempt to improve upon the performance of the pure CNN model mentioned above, and is a model inspired by 'Let there be color' [4] [18]. A classifier is incorporated into the CNN architecture to attempt to learn global class-wide image features through classification.

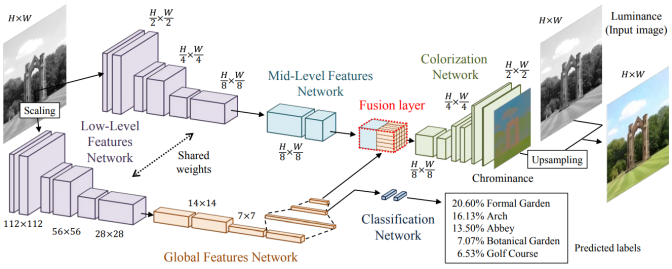

Fig. 2. CNN with classifier [4]

It can be seen from **Fig. 2** above that the model's architecture includes six different sub-sections, two of which being of a smaller size. The model takes the L channel of a 224 by 224 L·a·b colorspace image as input and outputs the predicted corresponding a and b channels. The first network, denoted 'Low-level Features Network' in the figure, down-samples the image through six convolutional layers. This is not achieved through traditional max-pooling down-sampling, but by increasing the stride of three of the convolution layers. The kernel for all layers is of size 3x3, a padding of 1x1 is added and a stride of either 1 or 2 is used respectively, 2 in order to halve the dimensions of the image. This method of adding a padding of 1x1 and increasing stride has been found to be able to replace max-pooling layers without loss in accuracy [19]. The output of this network is a 512 channel 28x28 feature tensor.

This output flows to the 'Mid-Level Features Network'. A more conventional CNN block that bottlenecks the original 521 channels down to 256 over two layers, while keeping the dimensions the same 28x28. This effectively reduces the information points per pixel and thereby combining and refining the features kept to mid-level features.

The 'Global Features Network' also takes as input the output from the 'Low-Level Features Network' and consists of four convolution layers with identical parameters to the ones mentioned under said network description, also alternating between a stride of 2 and 1. This is followed first by two fully connected layers for an output tensor of size 512. This output is fed straight to the 'Classification Network' and also through a final third fully connected layer reducing the dimensions to size 256 representation of the image before being fed to the 'Fusion layer'.

In order to classify the image the 'Classification Network' takes the aforementioned input and feeds it through two fully connected layers, resulting in a final tensor of size equal to the number of classes in the dataset 27. From the other output the 'Fusion layer' effectively incorporate the global features into the mid-level features, by combining the tensor representation of the image of size 256 from the 'Global Features Network' with the 256x28x28 outputs taken from the 'Mid-Level Feature Network' into a 512x28x28 tensor. This is then passed through a single convolution layer with 1x1 kernel size, zero padding and a stride of one to create the fused representation of the image of size 256x28x28.

Lastly this output is fed to the 'Colorization Network', that utilizes a set of convolution- and upsampling-layers. The convolution layers use a 3x3 kernel, zero padding and a stride of one, this results in the image dimensions of the in- and output from the convolution layers remaining the same, while keeping the channels flexible. The upsampling-layers increases the image dimensions of both height and width by a factor of 2, through utilizing the nearest neighbour technique [4]. These layers are ordered in a particular pattern to have the resulting output be a 2 channel 224x224 image. The final convolution layer before the last upsamling-layer is sigmoid-activated, while the remaining convolution layers for the entire model is relu-activated.

3.2.2. *Loss function*

This model utilizes two different loss functions. The colorization itself is evaluated with L1 loss and the classification with Cross-Entropy Loss. L1 loss was chosen for the colorization for the same reasons as mentioned under the regular CNN model and Cross-Entropy Loss was chosen as it is a common and good loss function for classification problems, because it calculates the loss between two probability vectors, which is exactly what is needed. Equation 1 illustrates the L1 loss function. Col denotes the colorization aspect of the model, x is considered as the grayscale image, y represents the 2 color channels of the real image.

$$\mathcal{L}_{L1}(Col) = \mathbb{E}_{x,y}[\|y - Col(x)\|] \tag{1}$$

Equation 2 illustrates the Cross-Entropy Loss function (CE). Class denotes the classification aspect of the model, label denotes the true image label and $(\cdot)$ is the inner product.

$$\mathcal{L}_{CE}(Class) = \mathbb{E}_{x,label}[-label \cdot \log\left(Class(x)\right)] \tag{2}$$

The combined global loss function for the network is illustrated by Equation 3, where σ is a parameter controlling the weight of the importance of the classification error in the resulting global loss. For this model a σ value of $\frac{1}{300}$ is used resulting in a relative weight of roughly 1/5. M_{Col} denotes the colorization aspect of the model and M_{Class} denotes the classification aspect of the model.

$$\begin{aligned}\mathcal{L}_{Global}(M) =& \mathbb{E}_{x,y,label}[\|y - M_{Col}(x)\| \\ &+ \sigma(-label \cdot \log\left(M_{Class}(x)\right))]\end{aligned} \tag{3}$$

3.2.3. *Training*

In order to train the network all images had to be scaled to exactly 224x224 before being input into the network, this is due to the implementation of the 'Colorization Network' and how its output is fused with the mid-level features. The image class is also used during training to validate the accuracy of the 'Classification Network'. The network is trained using the aforementioned combined loss function. It is important to note that when performing back-propagation, the L1 loss from colorization affects the entire network, however the Cross-Entropy Loss from classification only affects the 'Classification Network', 'Global Features Network' and the 'Low-Level Features Network', it has no effect on weights and biases in either the 'Mid-Level Features Network', 'Fusion layer' nor the 'Colorization Network'. The optimizer chosen for the model is Adam with a learning rate of 5×10^{-5} and the model has trained for 200 epochs reaching its lowest global loss at the 110th epoch.

3.3. Attention U-Net + PatchGAN

3.3.1. *Architecture*

The implemented model is composed of two major parts – a generator and a discriminator. Attention U-Net is taken as the generator and PatchGAN is taken as the discriminator. In order to obtain grayscale images easier, L·a·b colorspace was used instead of normal RGB colorspace.

Fig. 3 illustrates Attention U-Net Architecture with a deep dive into Attention Gate(AG). Attention U-Net performs pixel-wise transformation which does not change the spatial resolution of images but more channels can be output.

Our input images are with size 256×256×1 as only L Channel(grayscale) is taken and the output size is 256×256×2 representing a Channel and b Channel. If the output is combined with the input, colorized images are acquired. The down-sampling process of Attention U-Net is quite similar to original U-Net [11], which continuously reduces the spatial resolution of images by Max-pooling and introduces more channels by convolutional layers. However, the up-sampling process of Attention U-Net acts different from the original U-Net, which introduces AGs. At the beginning, two different signals are input to AGs, the first signal is the up-sampled previous layer output g and the second signal is the parallel down-sampling layer output x^l. They are both processed with convolutional layers with kernel size 1×1, no padding and 1 stride and then added up. After a series of processing steps described in **Fig. 3**, Attention Coefficient α is calculated. Attention Coefficient can identify salient image regions and prune feature responses to preserve only the activations relevant to the specific task [14].

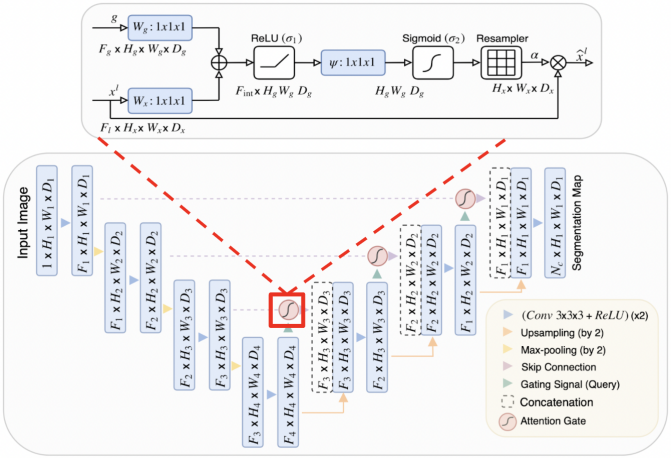

Fig. 3. Attention U-Net with Attention Gate deep dive [14]

Equation 4 and 5 describe how to calculate Attention Coefficient ${\alpha_i}^l$ in detail [14].

$$ {q_{att}}^l = \psi^T(\sigma_1({W_x}^T {x_i}^l + {W_g}^T g_i + b_g)) + b_\psi \tag{4} $$

$$ {\alpha_i}^l = \sigma_2({q_{att}}^l({x_i}^l, g_i; \Theta_{att})) \tag{5} $$

For Equation 4, ${x_i}^l$ is the signal represents the output from the down-sampling layer which is parallel to the current layer and g_i is the signal represents the up-sampled output of the previous layer. $W_x \in \mathbb{R}^{F_l \times F_{int}}$, $W_g \in \mathbb{R}^{F_g \times F_{int}}$ and $\psi \in \mathbb{R}^{F_{int} \times 1}$ are linear transformations. By applying W_x, the number of channels of ${x_i}^l$ will be changed from F_l to F_{int}. Similarly, by applying W_g, the number of channels of g_i will be changed from F_g to F_{int}. By applying ψ, the number of channels will be changed from F_{int} to 1, which results in a coefficient and the spatial resolution keeps the same as ${x_i}^l$. $b_\psi \in \mathbb{R}$ and $b_g \in \mathbb{R}^{F_{int}}$ are bias terms. σ_1 denotes ReLU activation function and σ_2 corresponds to Sigmoid activation function.

The output $\hat{x}^l$ of AGs is calculated by element-wise multiplication of α and x^l. $\hat{x}^l$ is then concatenated with g to double the number of channels. The concatenation is then followed by two convolutional layers.

PatchGAN is a type of discriminator for generative adversarial networks(GANs) which only penalizes structure at the scale of local image patches [15]. **Fig. 4** describes the architecture of PatchGAN in our case. The input is the combination of L Channel and 2 channels generated by Attention U-Net, which results in the size of 256×256×3. After a sequence of stacking blocks of Conv-BatchNorm-LeackyReLU, the output with size 30×30×1 can decide whether the input image is fake or real [16].

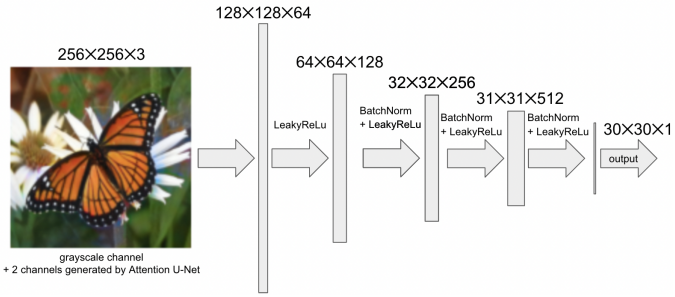

Fig. 4. PatchGAN for this specific case

3.3.2. *Loss function*

Equation 6 illustrates the loss function of PatchGAN. G and D denote the generator and the discriminator respectively. x is considered as the grayscale image, y represents 2 color channels of the real image and z is the input noise for the generator. As we trained the discriminator with both fake images generated by Attention U-Net and real images, the PatchGAN loss consists of two types – the loss for judging fake images and the loss for judging real images.

$$ \begin{aligned} \mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] \\ &+ \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))] \end{aligned} \tag{6} $$

Equation 7 illustrates the $L1$ loss of the generator. $L1$ loss is used rather than traditional $L2$ loss for it encourages less blurring [15]. $L1$ loss also acts bolder than $L2$ loss, which results in quite vibrant colorization. Meanwhile, $L2$ loss tends to perform conservatively, which results in gray-ish colorization [16].

$$ \mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1] \tag{7} $$

Equation 8 describes the complete loss of the generator, which is composed of conditional PatchGAN fake loss where

Fig. 5. Qualitative comparisons between models

G tries to minimize this objective against an adversarial D that tries to maximize it and balanced $L1$ loss. λ is a coefficient to balance the contribution of two losses to the final loss.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (8)$$

3.3.3. Training

The training from the scratch involves 100 epochs (see **A.2** in **A. APPENDIX** for training curves of both the generator and the discriminator). After a few epochs, the discriminator is fooled by fake images generated by Attention U-Net. Therefore $L1$ loss becomes the main contributor of the generator loss. In order to suppress the fluctuation of the discriminator loss, we reduced the learning rate of the discriminator to a quite small value.

Apart from training from the scratch, we also trained the model on COCO data-set [20] and extracted parameters first. The model is trained on selected images from ImageNet data-set [21] with pre-trained parameters later and the training takes 20 epochs. We observed that conditional PatchGAN fake loss has little contribution to the generator loss during the whole training process and $L1$ loss always takes the lead because the discriminator is fooled at the beginning of the training after the completion of pre-training process.

4. RESULTS AND DISCUSSION

The premier goal of the colorization task is to produce images that look 'good'. The term 'good' refers to numerous benchmarks – If the image is colorized at all; If the colors are reasonably chosen under the multi-modal premise; If the colors are assigned to pixels to generate reasonable geometric structures. Despite all the subjects to consider, the ultimate theme is always to evaluate in accordance with human perception. Therefore, examining colorization results by eyes is always an appropriate start. Examples of both bad (bird images) and good (butterfly images) colorization results can be seen in **Fig. 5**. All the four networks have successfully colorized the butterflies with the red-orange patterns and black rims, but for the bird the first two networks show in general not very vibrant colors while the latter two generate quite bright colors.

There lie drawbacks with this intuitive approach, though. Human perception and judgement are under the influence of irrelevant conditions such as display techniques and the surroundings of viewers. It also lacks a consistency due to individual biological differences and has subjective preferences. Still under the same theme of simulating human perception, various quantified metrics have been trialed out and proven to be representative of image quality.

In our project, classical image similarity metrics were adopted to compare different models. We applied PSNR, SSIM and LPIPS to discuss how much the generated images resemble the authentic ones. The average scores of each image in validation set are illustrated in **Table 1**.

PSNR, as a 'baseline' metric, only involves with MSE between two images and no further modification to simulate human perception. SSIM improves this by including the covariance between images to reveal structures, assigning higher scores when two images are not only similar on a pixel-basis but also have a similar pattern in an area. LPIPS utilizes the fact that 'the internal activations of deep convolutional networks, though trained on a high-level image classification task, are often surprisingly useful as a representational space for a much wider variety of tasks' [22], since the neurons in the hidden layers can capture features that are also important during the cognitive process. In our setting we used AlexNet as suggested in the paper.

Table 1. Quantitative comparisons between models

	PSNR↑	SSIM↑	LPIPS↓
CNN	21.239301	0.895211	0.168267
CNN + Classifier	**21.259139**	**0.907357**	0.161973
Attention UGAN	18.476772	0.873606	0.188972
Baseline UGAN [16]	17.939683	0.852737	0.199551
Pretrained UGAN	19.658543	0.898401	0.150918
Pretrianed Attention UGAN	19.997733	0.901787	**0.147607**

For PSNR and SSIM, the two CNN based networks have higher scores, but there is an upward trend from Baseline UGAN to Pretrained Attention UGAN which ends near to the highest score. For LPIPS metric, Pretrained Attention UGAN gives the best score, and the two pretrained networks are on a lower dissimilarity level than UGAN networks without pre-training. All of the numbers indicate that Pretrained Attention UGAN works the best on our dataset.

The overall performance is fairly consistent within the three metrics, nonetheless many interesting findings are exposed after comparing quantitative with qualitative results. The simple CNN network in general does not generate enough colors with a brown and grey tone, but it ends with a quite good numerical result, proving underlying flaws with the metrics to genuinely incorporate human perception. This also resonates with CNN's intrinsic tendency to prefer neutral colors over bright colors in order to avoid punishment during training. CNN + classifiers has more qualified colorization results and more accurate butterfly colors compared to the baseline, but does not receive significant boosting numerically, again questioning the metrics' credibility. On the other hand, recalling that Attention Unet emphasizes the important features, both the butterfly and bird become therefore more prominent from the background, and the grass in butterfly images is greener and similar to real images compared to CNN based networks. This difference is partly distinguished by the LPIPS results when Pretrained Attention UGAN receives the lowest dissimilarity score.

There left some aspects worth discussion. In our optimal model, Pretrained Attention UGAN, there are objects not learned, and parts of wrong decision. After re-examining our dataset, the multi-modal phenomenon can be further explained. For the bird image, the red head does not come from nowhere – We have another kind of bird classified as *downy woodpecker* shown in **Fig. 6** particularly characterized by the small red area in the back of its head. Besides, the background of the bird is colored blue is quite intuitive, since most scenery images contain vast sky background such as the second image labeled as *tower*. For the butterfly image, the center of the flower is green instead of yellow. It is hard to tell whether this choice is the consequence of under-fitting or from the green stamen of *Welsh poppy*. Possible solutions or improvements are applying more epochs of training, and larger and more representative datasets. Deeper networks may improve, too, but risk the uncertainty of emphasizing some trivial features. The overall performance we gained implies that we are likely to have been around the global optimum, so altering loss function and learning rate may not have significant improvements on top of the current results.

Fig. 6. Other examples from dataset [21]

5. CONCLUSION

We successfully implemented three models with different architectures for the colorization task – CNN, CNN with classifier and Attention U-Net(generator) + PatchGAN(discriminator). The common thing for all models is that the quality of colorization is varied between different images in validation set. This can be traced back to the intrinsic characteristic of the chosen dataset. The uniform quality of colorization is most likely to be achieved by larger dataset with more categories of features. Deeper networks and more training epochs might help as well.

In comparison, Unet architecture doesn't necessarily outperform simple CNN, according to the quantified results, but combined with the Attention Gate to reinforce important features and pre-trained parameters for better initialization for training optimization, the output is significantly enhanced to human eyes, which in the end brings Pretrained Attention UGAN to be the optimal model in our setting.

On top of the results themselves, we also discussed the credibility of ways of image quality evaluation. Using human eyes is essential, but it faces the same problems as most qualitative metrics would do. Quantitative metrics, PSNR, SSIM and LPIPS have different balance between pixel-wise difference with representation of human perception. They all managed to reflect the qualitative performance to an acceptable extent, but still with further space for improvement.

Code for all proposed models can be found in this GitHub repository. Evaluation of all proposed models can be found here.

6. REFERENCES

[1] Daniel Sỳkora, John Dingliana, and Steven Collins, "Lazybrush: Flexible painting tool for hand-drawn cartoons," in *Computer Graphics Forum*. Wiley Online Library, 2009, vol. 28, pp. 599–608.

[2] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum, "Natural image colorization," in *Proceedings of the 18th Eurographics conference on Rendering Techniques*, 2007, pp. 309–320.

[3] Zezhou Cheng, Qingxiong Yang, and Bin Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 415–423.

[4] 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 Trans. Graph.*, vol. 35, no. 4, jul 2016.

[5] Richard Zhang, Phillip Isola, and Alexei A. Efros, "Colorful image colorization," 2016.

[6] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin, "Image analogies," in *Proceedings of 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001*, 2001, pp. 327–340.

[7] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng, "Intrinsic colorization," in *Proceedings of ACM SIGGRAPH Asia 2008 Papers, SIGGRAPH Asia 2008*, 2008, p. 152.

[8] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong, "Image colorization using similar images," in *Proceedings of 20th ACM International Conference on Multimedia, MM 2012*, 2012, pp. 369–378.

[9] Aditya Deshpande, Jason Rock, and David Forsyth, "Learning large-scale automatic image colorization," in *Proceedings of 5th IEEE International Conference on Computer Vision, ICCV 2015*, 2015, pp. 567–575.

[10] Adrian Pipirigeanu, Vladimir Bochko, and Jussi Parkkinen, "A computationally efficient technique for image colorization," in *Proceedings of 2nd International Workshop on Computational Color Imaging, CCIW 2009*, 2009, pp. 120–129.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Springer Verlag, 2015.

[12] Huimin Huang, Lanfen Lin, Ruofeng Tong, Hongjie Hu, Qiaowei Zhang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, and Jian Wu, "Unet 3+: A full-scale connected unet for medical image segmentation," in *Proceedings of 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2020*, 2020, pp. 1055–1059.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, 2018.

[14] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert, "Attention u-net: Learning where to look for the pancreas," vol. 10, 2018.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Berkeley AI Research (BAIR) Laboratory, USA, 2017, pp. 5967–5976.

[16] Moein Shariatnia, "Image colorization tutorial," .

[17] Luke Melas-Kyriazi, "Image colorization with convolutional neural networks," 2018.

[18] Przemyslaw K. Joniak, "Automatic image colorization with simultaneous classification based on let there be color!," .

[19] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar, *Microsoft COCO: Common Objects in Context*, Springer Verlag, 2014.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[22] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

A. APPENDIX

A.1. Baseline model progression over training epochs

Fig. 7 shows our CNN baseline model's evolution over 100 training epochs. The network starts by anticipating nearly all pixels in the image to be the same hue, gradually learning to assign red colours to the butterfly wings.

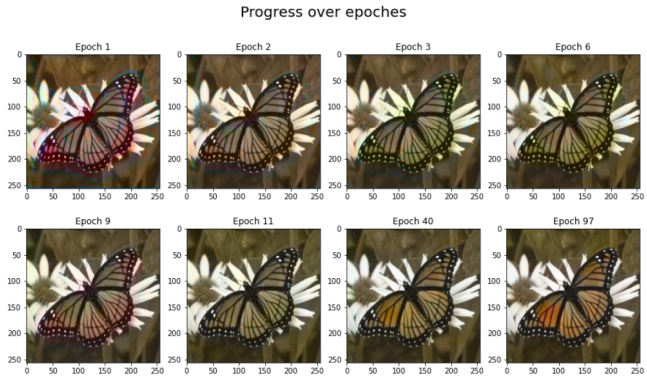

Fig. 7. Sample of the Baseline model (CNN) progression over 100 training epochs.

A.2. Training curves for Attention U-Net + PatchGAN

Fig. 8 shows training curves of the generator. The $L1$ loss is the main contributor of the generator loss. The generator loss continuously decreases during the entire training process.

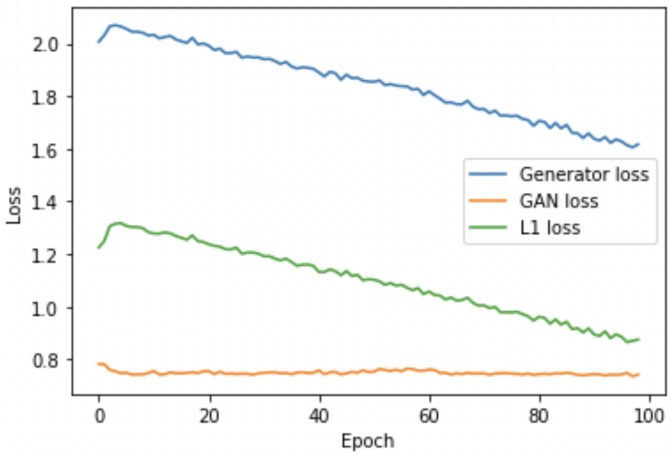

Fig. 8. Attention U-Net(generator) training loss

Fig. 9 shows training curves of the discriminator. After a few epochs, both the real loss and the fake loss stop decreasing and start to fluctuate, which indicates that the discriminator is fooled and hard to tell whether the image is real or fake. This also means that the generator is good enough for producing colorized images. The conditional PatchGAN fake loss contributes to the generator loss much more at the beginning of the training than other periods of the training because the discriminator cannot make valid judgement afterwards.

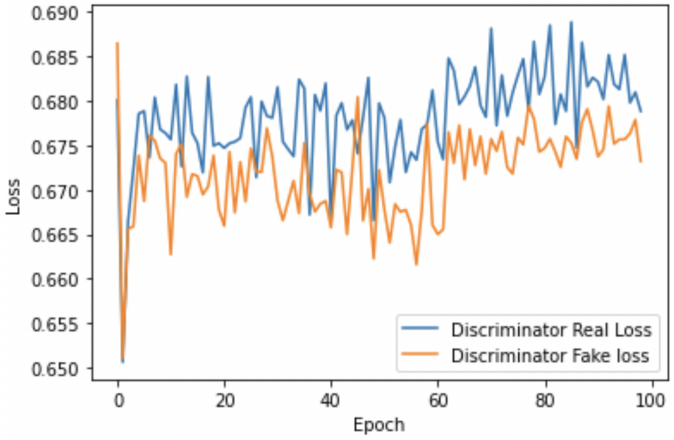

Fig. 9. PatchGAN(discriminator) training loss