How is the impact of training separate chrominance channels on the accuracy of image colorization?

Luc Lacombe (lla216), Ali Muftu (amu216), Andy Popov (apo221) and Alp Sarici (asa525)

1 Abstract

The task of recreating black and white images with color has been a long awaited and anticipated task in the field of computer vision, requiring either manual labor of someone coloring the image in or by the use of intelligent computer systems using machine learning. By using modern techniques we aim to colorize grayscale images to enhance and restore them. This mainly comes from the idea of historical and old family photographs that don't have any color. The given challenge is guessing the missing color accurately.

Our hypothesis was that by using the LAB color space and separating the A and B channels we could enhance accuracy and precision of the color precision as each model could focus on learning the predictions of its own respective color axis without the interference of another. To assess this approach we used a modified U-Net architecture trained on the LAB color space. There were a total of 4 models tested: LAB, LA, LB, and LA + LB. We preprocessed 20 thousand images from the larger dataset and performed histogram matching to improve consistency. The models were trained individually on CPU environments.

Our findings show that even though the split channels performed well (LA and LB) the unified LAB model still performed the best out of all the evaluated models. However, the LA + LB model came close, suggesting that there is some worth to training separated channels. The training separated channels were more complex and harder to work with. We conclude that even though training A+B introduces overhead, it can still show better results in special cases. Future work could work on the post-merging processes to fully benefit from the approach of splitting channels.

2 Introduction

Old black and white photos hold incredible value to us, and as a gift to our families, we were motivated to colorize them ourselves, which would be a strenuous task to do manually. The aim of our algorithm is to accurately colorize photos given any grayscale image. We chose the Flickr-Faces-HQ (FFHQ) dataset, which consists of 52, 003 face images from a variety of different ethnicities, races, and ages. To create the best possible colorized photos we wanted to investigate a new approach to colorization processes.

This approach stems from the idea of getting the most accurate colorization of the images. To do this, we work with the *LAB* color space where *L* is lightness *A* is green to purple and *B* is blue to yellow. [1] What this achieves is the separation of luminance from color. Therefore, when the images are

black and white or colored, the L channel stays the same, and we would only need to predict A and B values. This allows us to reconstruct the full image with precision. (Citation)

By splitting the training into separate channels, A and B, to reduce the complexity of tasks needed to be learnt, we envisioned the product would be a better outcome than if we trained the channels together. The approach is aimed at increasing the accuracy of the colorization process by allowing better color predictions without the interference of luminescence or other channels.

This brought us to our final question: How does the performance of independently trained models on the LA and LB channels, combined post-training, compare to a single model trained on the complete LAB color space regarding colorization accuracy?

---

3 Methodology

3.1 Data Preparation and Visualization

The dataset we chose consists of $52,003$ pictures with a 512×512 resolution. After some brief testing, we found that our computers did not have the processing power to train on all the images. To fix this, we lowered the resolution to 256×256 and additionally reduced the number of images to 20,000. This was done by writing an algorithm which took the photos and resized them to our preferred resolution, after which another algorithm was then used to make grayscale versions of all the images.

The data visualisations were made to assess the quality and diversity of features of the dataset. To get into the specifics of what we have done, we analyzed facial variety, most importantly skin color. We analyzed background types, and edge density to ensure a fair distribution, representation, and connectivity throughout the data. This let us see if the data encompassed a wide range of features.

We can see in figure 1 that there is a good variety of dominant skin colors among the data set. This means that the dataset is diverse and ensures that the model is less biased and more inclusive. Additionally, the consideration of different ethnic backgrounds and races are done fairly.

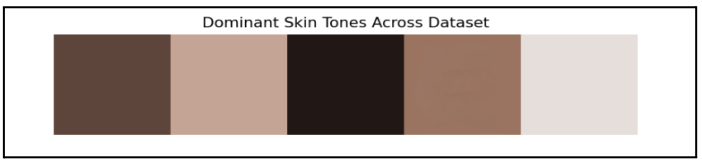

Fig. 1. Test for skin tones in dataset

Edge density is the number of edges in a picture relative to its size, basically where there is a significant change in colors or brightness. [2] This generally links to detail in an image. In figure 2, we can see that most images have a density between fifteen and thirty. This indicates a moderate level of detail in the images; they aren't overly complex, and they aren't simple. There are exceptions of overly complex and simpler images but that could be due to background and other factors such as noise.

However, the highest edge density of the images appears to be around 18 which is a good value to work with.

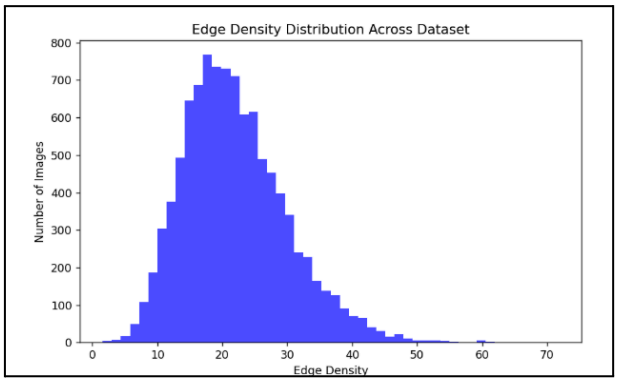

Fig. 2. Edge density histogram of dataset

The RGB distribution tests for the red, green, and blue pixels and their frequencies in the given images. [3] Figure 3 looks like it has a fairly balanced distribution among the channels alongside some irregularities. We can see that blue spikes around low, green at medium, and red at high intensity. This suggests that there is a wide variety of color, there will be lighter and darker spots in images. However, there looks to be that there could be some bias towards red tones towards high pixel intensities.

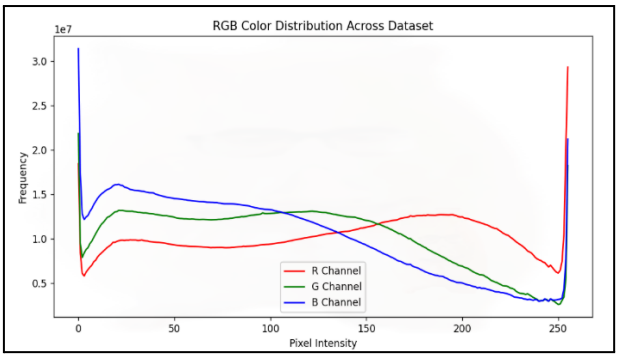

Fig. 3. RGB color distribution across dataset

From the graphs we were able to conclude that the dataset has a good variety of skin colors meaning it shows inclusive representation. The edge density shows us that the images are of overall good quality so we can extract features accurately. The RGB color distribution shows us that the images are not too dark or too light, they have slight irregularities but are overall diverse in color and are a good set of images to work with. Altogether, the dataset looks to be a good match for training models and researching the question.

3.2 Prediction Models

Multiple models were tested and tried for image colorization and we ended up choosing a lightweight U-net model. Below we evaluate three candidates: U-Net, Pix2Pix GAN, and a Residual Autoencoder.

3.2.1 U-Net

Our first option was U-Net: a widely-used convolutional architecture for dense prediction tasks, known for its strong performance in segmentation and restoration problems [4]. Since we were colorizing images we value the property of UNET that skips connections between the encoder and decoder layers. These layers are responsible for reconstructing fine image details as they contain spatial features that may be lost during downsampling.

Additionally, it is quite easy to implement and can withstand noisy data, which works in our favour since we don't have labels in our dataset. It can also be trained effectively on relatively small datasets, and also with unlabeled data by treating the problem as self-supervised. [5]

However, it can also lead to the production of slightly blurry or average-toned results, since it aims to reduce the mean squared error (MSE) loss per pixel without a focus on perceptual quality. We include and work on the output of activation of the tanh function and rescaling to fit LAB color ranges. We normalized the LAB channels to the range [-1, 1] during training and used a tanh activation at the final layer. We chose the hyperbolic tangent function since it fits our normalization of -1 to 1 while centering the data. The alternatives of using the sigmoid function or ReLU, which we analyzed from the course, did not fit our U-Net model since the sigmoid function does not center the data, and ReLU is used for positive values and is not ideal for output layers in tasks like image generation.

We used a lightweight architecture with lesser filters so that we could have a smaller training time. Our lightweight U-Net variant made our model faster, more memory efficient and with good accuracy. We also took advantage of the memory by dynamic data training, we used a tf.data.dataset generator to dynamically load the data into memory. We also preserved spatial details such as edges, textures, shapes, and patterns by skipping connections from encoder to decoder layers via concatenating between them. Lastly, during post-processing, we matched the predicted values to the histogram. Once the color channel prediction terminated, we matched these values to a CDF-based histogram we generated from the dataset. This aligns the predicted values with those in the training set to make it color realistic and consistent.

3.2.2 Pix2Pix GAN

Regarding the Pix2Pix GAN, in general, the training complexity and instability made it unsuitable given our time frame and hardware constraints. We mainly came up with cons to this model such as: the training being unstable, requiring loss weight tuning and a discriminator architecture. It would also generate artifacts in the outputs such as checkerboard patterns or color bleeding. [6] Most importantly, a higher computational cost and sensitivity to data imbalance.

3.2.3 Residual Autoencoder

Regarding the residual autoencoder, we did not favour this model, because its performance was inferior to U-Net in both quantitative metrics (PSNR, SSIM) and visual evaluation. [7] It was also harder to debug and integrate with histogram matching. However, the pros if we were to redo this with more computational power, would be: having an Encoder-decoder model with residual connections in

the latent space to improve reconstruction quality. Also, the model Can be trained for self-supervision and reconstructs residual color values instead of full color prediction.

3.3 Model Architecture

As mentioned before, we are using a modified lightweight U-Net convolutional neural network architecture. Our model has an encoder that has convolutional layers (Conv2D) and is capable of max-pooling operations(MaxPooling2D). These allow the extraction of contextual features. The decoder in our model has up-sampling (UpSampling2D) and skip connections(concatenate) capabilities. These allow the preservation of details crucial for accurate and precise color prediction.

To answer our research question "How is the impact of training separate chrominance channels on the accuracy of image colorization?" Our independent variable is set to be the LA, LB, and LAB channels separately. To achieve this, we set the mode of training output before training. Then the output of the network is a channel in the tanh range of [-1, 1], suitable for the normalized AB channels in LAB color space.

3.4 Feature Design

We focused on the LAB color space for its separation of luminance (L) from color information (AB). This allows leveraging inherent brightness (L) from grayscale images, which stays consistent as our control variable for all models. This way, the network could focus exclusively on predicting chromaticity components (A and B). We furthered this exploration with three distinct prediction strategies: A Separate Model: Train independent models for predicting either the A-channel (green-magenta) or the B-channel (blue-yellow). Joint AB Prediction having a single network predicting both A and B channels simultaneously. Planar (Vectorized) Prediction: The network outputs a single vector that is reshaped to AB channels. Which captures all the inter-channel dependencies. The final model further incorporates partial histogram matching post-processing to refine predicted color distributions. [8]

3.5 Data Preparation and Scaling

Our dataset consisted of grayscale and corresponding ground-truth color images. The preprocessing steps included: Resizing all images to a consistent size of 256×256 pixels. Then we convert the RGB images to the LAB color space. This extracts the luminance (L) channel from grayscale images and chromatic channels (AB) from color images. Furthermore, we normalize the L channel to [0, 1] and AB channels approximately to [-1, 1]. Lastly, we split the dataset into training (90%) and validation (10%) subsets. Because of the limited computing power, we trained on a subset of 20,000 images from the original dataset of 52,000 images, which proved sufficient to achieve reliable performance.

3.6 Creating Sequences and Data Generators

To efficiently manage our use of memory, especially with our large dataset, we implemented TensorFlows data pipeline with a custom Python generator. These generators dynamically loaded,

processed, and batched images as needed during training. This ensured a minimized memory overhead as well as a smooth use of GPU or CPU. Our data pipeline is randomized batching and shuffling of training data per epoch. It also included prefetching of training data using tensorflow tf.data.AUTOTUNE. Essentially, batching makes sure the data is in a different batch each time, and autotune makes it so that the data gets preloaded in these batches.

3.7 Hyperparameters

We selected our hyperparameters by extensive trial-and-error and model analysis such as looking at our training and validation loss plots. We prioritised both model performance and computational efficiency but concluded that after 20 epochs the gradient remained the same. Therefore, we settled for 20 epochs with a batch size that our CPU's could handle of 64. We set an Adam default optimizer learning rate of 0.001. A value widely recognized as stable and effective for most neural network tasks (Kingma & Ba, 2014). This value gave us a good balance between mix of speed and training stability, and lowered the divergence and oscillations during training. We also set a filter configuration with 16, 32, and 64 filters across convolutional layers. We did this because based on our trials, a simplified filter configuration still captured enough spatial and contextual features necessary for accurate colorization. Lastly, we normalized to [-1, 1] via tanh activation, before rescaling to LAB range. This was because the LAB color channels naturally converge around zero and this normalization helps gradients propagate more effectively.

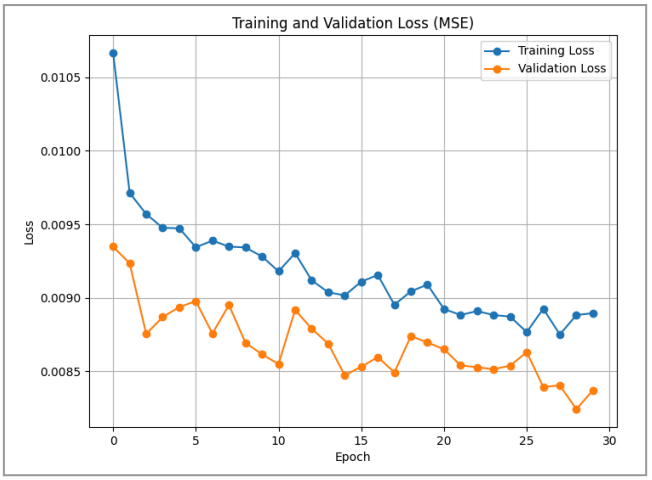

Fig. 4. Training and validation loss graph

3.8 Training Procedure

Training was done on CPU enabled environments. Model training involved MSE, monitoring, and a saveable model for reproducibility and analysis. The Mean Squared Error (MSE) loss as seen in the figure was between the predicted and ground truth AB channels. This helped us observe the training and validation losses to detect overfitting which as seen in figure 4 is successful until our selected epoch.

3.9 Innovative Character

Our adaptation of U-Net goes beyond standard implementations of this model. We use post processing by matching the output to our data's frequency distributions. We matched the AB-channel distributions with the predetermined cumulative distribution functions (CDFs) from training data, blended at 50% (alpha=0.5) enhanced the color realism and as mentioned in the results below improved the accuracy. By allowing separate or combined AB channel predictions we optimized the network to learn for specific color channels. The results sections go over how recombining the channels isn't the most successful form of colorization yet, but this has great potential.

Our customized U-Net required deep and extensive knowledge over convolutional neural networks, gradient-based optimization, and data preprocessing techniques. The mathematical complexity involved understanding: Convolution and pooling operations which extracted the most important features while reducing spatial dimensions which improved efficiency. Additionally, normalization and activations which scaled data for stable training; use tanh activation to handle LAB color ranges. Also, Gradient descent (Adam) which traversed our network reducing prediction errors. Lastly, histogram matching (CDF) improved overall realism when colorizing.

---

4 Results & Discussion

Evaluating the success of each model's image colorization is a challenging task. For image colorization humans are the final judges of success, but manually assessing thousands of images across different models is nearly impossible and subject to errors and inconsistencies. Our goal was to find a quantitative metric that closely mirrored human judgment in order to effectively evaluate the models at scale. Ultimately, we identified two broad categories of evaluation methods: Low-level color difference metrics and human-tuned perceptual metrics.

Mean absolute error (MAE) and earth mover's distance (EMD) are examples of low-level color difference metrics. These two metrics are relatively simple and intuitive and broadly speaking calculate the color differences between two images. MAE measures the average absolute difference between RGB pixel values and EMD evaluates the distance between RGB color distributions in an image. While these metrics are objective and computationally efficient, they fail to mimic human evaluation. A reason for why these metrics don't correlate to proper evaluations can be explained by the equal evaluation of all colors/pixels. Our machine learning model disproportionately recolors faces over backgrounds, many backgrounds remain black and white making these low-level difference metrics unsuccessful. We did small sample subjective testing and found that these metrics didn't seem to agree with our evaluations. While this is subjective we generally saw these metrics as too simple and ineffective.

Delta E 2000 and learned Perceptual Image Patch Similarity (LPIPS) are examples of human-tuned perceptual metrics. These metrics are specifically designed to align with human color perception. Delta E 2000 recognizes that the human eye perceives different colors with varying sensitivities due to the uneven distribution of cone photoreceptors' wavelength. [9] It uses our understanding of these differences to compute a more accurate difference value similar to the low-level color difference metrics. Where it differs from the low level metrics is that it uses the LAB color space

and doesn't use an equal weight for each wavelength of color. To put it simply, it uses complex weighting functions and correction terms to output a metric closest to human perception. This metric initially seemed like the perfect metric until we found LPIPS which uses a deep learning approach trained on human perceptual similarity. Instead of comparing raw pixel values, it extracts high-level features from multiple layers of a convolutional neural network, to better reflect human perception. The network is trained on human-labeled image similarity datasets, allowing it to predict how different an image appears to the human eye rather than just computing a numerical color difference. [10] By computing the distance between these extracted feature representations, LPIPS can effectively capture differences in texture, structure, color, and overall image quality. Evaluating the recolored image against the ground truth LPIPS quantifies the similarity between the two and therefore gives a metric to the success of colorization. We found this to be the most fitting metric to use over the low-level color difference metrics and Delta E 2000 as doing subjective tests on samples it generally did better and we found it fun to use a deep learning metric on this machine learning project.

LPIPS has three different network types to select: 'vgg', 'alex', and 'squeeze'. We opted to use 'vgg' as it is the VGG-16 deep neural network which is the most complex and accurate out of the three. For the 2000 colorized images in LA, LB, LA+LB, and LAB, we evaluated them against the ground truth image using. This allowed us to determine the LPIPS score for each image across the four models, providing a comparative measure of their performance. We originally had not intended to evaluate the individual LA and LB image, however, after seeing the results of the LA+LB model we decided it would be interesting to add these to the results. To help understand the colorization accuracy of these models using the LPIPS metric we produced box-and-whisker plots, kernel density estimation plots, and also displayed some of the key colored images.

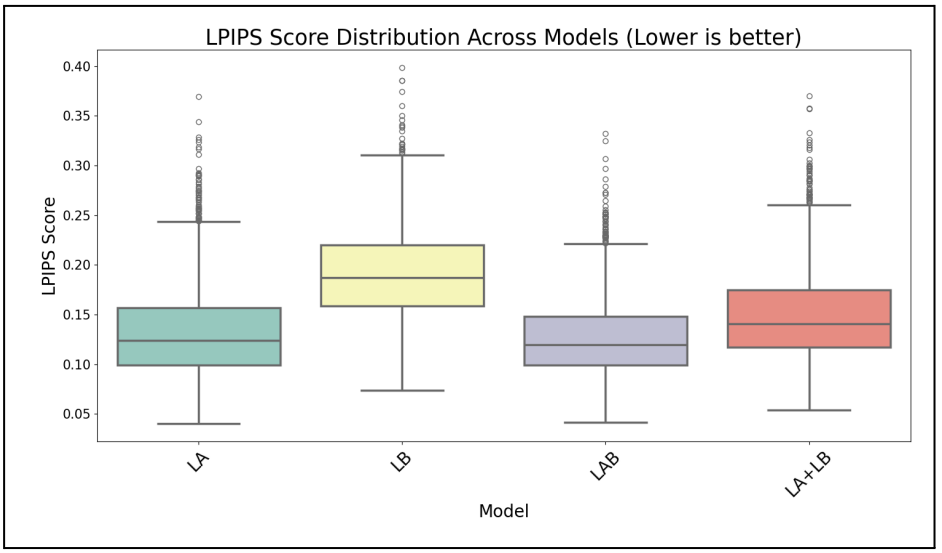

Fig. 5. LPIPS Score Distribution Across Models (Lower is better)

The box plot in Figure 5 compares the LPIPS score distributions for the LA, LB, LAB, and LA+LB models, note for LPIPS scores, lower is better. Overall the LAB model shows the best results in terms of both accuracy and consistency. It has the lowest median score and the thinnest interquartile range. It additionally has the lowest average score of 0.127 and lowest standard deviation of 0.040 [See

Table A, Appendix], signifying consistent performance across the test set. Furthermore, its worst-case score was better than others with a maximum of 0.332 [See Table A, Appendix]. Comparing these results with the LA+LB scores it is clear training on separate chrominance channels has a negative impact on the accuracy of colorization. The LA+LB images are far worse, and surprisingly even worse than the single LA model images. These findings are further supported by Figure 6 which shows the Kernel Density Estimate (KDE) plots of the models' scores.

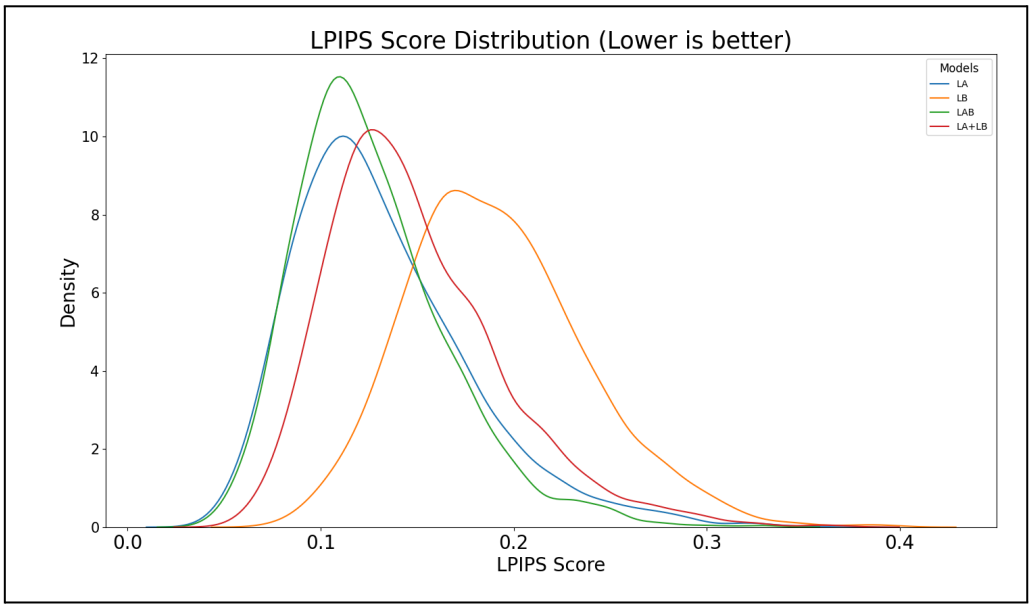

Fig. 6. LPIPS Score Density Distribution (Lower is better)

This plot illustrates the probability density of LPIPS scores for each model and again the LAB model's distribution (green curve) is clearly advantageous, featuring the leftmost peak, which is also the sharpest. This indicates that the majority of LAB's scores are tightly clustered around low values which is consistent performance identified in the box plot. Comparing this to the LA+LB model's (red curve) peak, it is shifted significantly to the right, and has fatter right sided tails. This indicates a lower concentration of good scores and more less desirable scores. This visual evidence strongly corroborates that splitting training into A and B channels and then recombining had a negative effect on colorization accuracy over LAB, despite twice the training time. What is once again interesting is the distribution of LA (blue curve) and LB images (yellow curve) with LA outperforming LA+LB, and LB being the worst performing by far. This potentially indicates that the A channel colors are more important to colorizing facial images.

It is clear from these plots that the non channel splitting LAB model outperformed the LA+LB model and channel splitting has a negative effect on colorization but it is important to see the difference beyond plots. In Table 1 we can see samples from each model's output, these images were randomly selected from the subset of outputs with the models average score. We also added the accompanied black and white (BW), and ground truth (GT) images. From this table once again we can subjectively evaluate that the LAB has the best colorization with the colored image being very close to the ground truth. The LA+LB colored image has a purple tint to it, and especially in areas where you'd expect some red (cheeks, lips, ears) there is purple. The LA average scored image is surprisingly and by pure coincidence Vince Gilligan, the creator of the show Breaking Bad. It was surprising to us that LA scored so high

given that we found this sample not very impressive. The LB image shows a very bland monotone colorization which looks like a simple filter which aligns with the previously seen performance.

SAMPLE MODEL OUTPUTS WITH AVERAGE LPIPS SCORE			
Model	**BW**	**GT**	**Colored**
LAB			
LA+LB			
LA			
LB			

Table. 1. Colored Output with Average LPIPS score from each Model

In Table 2 seen below we wanted to show the interesting interaction between the A channel model image, the B channel model image, and the combined LA+LB image. This image has an average score for the LA+LB and also has the BW and GT above it to compare. We can see that in this case the LA image does have significantly better colorization, while the LB image again has poor monotone colorization. The resultant LA+LB image is worse than the LA image which indicates that the A channel is much more important to the colorization and the B channel hinders the colorization when trained and added at equal weights.

LA, LB, LA+LB		
 BW  GT		
 LA	 LB	 LA + LB

Table. 2. Average Scoring LA+LB Image with LA and LB model images.

Finally, in Table 3 below we show the worst images for each model to highlight the limitations of the colorization. Interestingly we see that LA+LB, LA, and LB all have the same worst scoring image. It is clear from this where these models struggle. Whenever there is an image with bright or saturated colors they simply apply the average skin tone. It is most clear for the LAB model which colors the woman with an average white skin tone, completely failing to recognize any of the green. Another interesting observation from these images can be seen in the LA+LB, LA, and LB images. We see this strange square across all three which highlights where the model has been trained to add color. Given the background of the input BW image is monotone in texture, this training area becomes clear. Another limitation we have seen which is not apparent in these images are ears. These sometimes appear to stay black and white. This can be attributed to many of the training images having their ears covered or having one or both not being present due to the angle of the image.

SAMPLE MODEL OUTPUTS WITH WORST LPIPS SCORE			
Model	BW	GT	Colored
LAB			
LA+LB			

LA			
LB			

Table. 3. Colored Output with Worse LPIPS score from each Model

5 Future Work & Limitations

Despite the fascinating results we produced, our approach was limited by the hardware we were running our algorithms on. We did not have access to computers with GPUs so the entire algorithm ran on our CPUs instead which in turn made the process of training an individual model extremely slow, with the average time being 15 hours when running 20,000 images. An alternative we tried was running it on Google Colab, however when we ran it with more than 1000 images it would eventually time out because it was using too many resources.

Apart from running our algorithm with a GPU enabled, an improvement we can do to repeat the experiment would be to create a better merging algorithm for the LAB channels. Currently we don't know exactly what is wrong with the merging algorithm, since our heuristics suggest that it should be merging them to create better images than they are currently. When we investigated, we found that the individual LA and LB models were producing images that had the missing channel at disproportionate rates, which should not have been occurring. This might have led to an "overload" in data when merging the images, as they both contained data from the opposite channel which might have affected the output image's accuracy. This could be fixed with post-processing or color theory, however this wasn't within the scope of our experiment.

An alternative project within the same context would be to do the same investigation but with RGB rather than LAB. This would allow us to compare the accuracy of the colorization as LAB is based on perceptual lightness and split chromatic channels, while RGB only splits into the respective chromatic channels. LAB in theory should perform better due to the separation of lightness and color which allows for the algorithm to focus on more accurate colorization.

U-Net is primarily used in segmentation; however, in our case we extended it for image colorization. This use-case was not explicitly explored in our coursework. Additionally, adding

data-driven histogram matching for color refinement introduced brand new challenges such as red/green spots on faces and required extensive research about solving these issues.

As mentioned in section 3.2, we could also have used a different model instead of U-Net, such as Pix2Pix GAN, or a residual autoencoder. We could expand the scope of the project to not only include the separate training of different channels, but to also compare it with different models and see which ones have the shortest training time and output accuracy.

---

6 References

[1] HunterLab, "What is CIELAB Color Space?," HunterLab Blog, Oct. 27, 2015. [Online]. Available: https://www.hunterlab.com/blog/what-is-cielab-color-space/

[2] FRAGSTATS, "Area and Edge Metrics," *FRAGSTATS*, [Online]. Available: https://fragstats.org/index.php/fragstats-metrics/patch-based-metrics/area-and-edge-metrics.

[3] Zilliz, "Demystifying Color Histograms: A Guide to Image Processing and Analysis," *Zilliz Learn*, 2023. [Online]. Available: https://zilliz.com/learn/demystifying-color-histograms.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv preprint arXiv:1505.04597, 2015. [Online]

[5] O. Kapon, "Image Colorization Using UNET," Kaggle, Aug. 2021. [Online]. Available: https://www.kaggle.com/code/omreekapon/image-colorization-using-unet

[6] J. Brownlee, "A Gentle Introduction to Pix2Pix Generative Adversarial Network," *Machine Learning Mastery*, Dec. 2019. [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarial-network/

[7] M. Tripath, "Facial Image Denoising Using AutoEncoder and UNET," *Humanitarian and Social Development*, vol. 2, no. 2, pp. 89–96, Jul. 2021. [Online]. Available: https://hsd.ardascience.com/index.php/journal/article/download/71/57

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proc. IEEE (CVPR), 2017

[9] Techkon USA, "Demystifying the CIE DE 2000 formula," Techkon USA. [Online]. Available: https://techkonusa.com/demystifying-the-cie-de-2000-formula/.

[10] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep

Features as a Perceptual Metric," arXiv preprint arXiv:1801.03924, 2018. [Online]. Available: https://arxiv.org/pdf/1801.03924

---

7 Information Sheet

MLVU final report information sheet

Please include this page in your report either at the start or at the end, before the appendix. Do not change the formatting.

Group number 125

Authors

name	student number
Ali Muftu	2770908
Luc Lacombe	2769886
Andy Popov	2772025
Alp Sarici	2766355
Jaehyuk Choi	

Software used
VScode, Pycharm

Python

Pillow (PIL)
OpenCV(cv2)
Scikit-image
PyTorch
TensorFlow/Keras
NumPy
Pandas
Scikit-learn
Matplotlib

U-Net Architecture: Implementation was done from scratch using PyTorch and TensorFlow/Keras

Use of AI tools

We used ChatGPT to give us an idea of how and what our project should be. We asked for a schedule and how the outline of the report should be (points that are essential and shouldn't be left out, introduction & methodology for example).

Group disagreements (optional)

Jaehyuk had no attendance. We met multiple times and we've never seen the guy. Despite multiple attempts of communication we were unable to get in reach. He didn't even get on the Google document to put his ID on it.

---

8 Appendix

MODEL	Average Score	Std Dev	Minimum	Maximum
LAB	0.127	0.040	0.042	0.332
LA+LB	0.149	0.045	0.054	0.370
LA	0.132	0.046	0.040	0.369
LB	0.191	0.047	0.074	0.398

Table. A. Model LPIPS Score statistics