PROJECT PROGRESS REPORT

Huda Imran
Student# 1009910606
huda.imran@mail.utoronto.ca

Claire He
Student# 1010255698
claire.he@mail.utoronto.ca

Manha Siddiqua
Student# 1010315375
manha.siddiqua@mail.utoronto.ca

Jihoon You
Student# 1009796801
jihoon.you@mail.utoronto.ca

ABSTRACT

This project utilizes deep learning for automatic image colorization, transforming greyscale images into color versions using a convolutional neural network (CNN). The L channels of the greyscale image will be inputted to the neural network to output predicted ab channels which will then be combined with the L channel to create the colour images. —-Total Pages: 9

1 BRIEF PROJECT DESCRIPTION

The goal of this project is to develop a model that colourizes greyscale images by predicting their appropriate colour schemes. The motivation for this project lies in the fact that there is a visual disconnect between the past and present, and colorization helps people better understand and connect with history. While hand-painted colorization techniques have existed for years, deep learning offers a faster, more cost-effective solution. Convolutional Neural Networks (CNNs) are particularly appropriate for this task due to their ability for image detection and breaking up an image in layers that detect different features, building a deeper understanding of the image to choose appropriate colouring Mandić et al. (2024). Our model takes the L channel of a greyscale image and predicts the ab (colour) channels. These are then combined and converted back into RGB to generate the final colourized image as seen in Figure 1 - a process that used to take hours, now done in mere seconds.

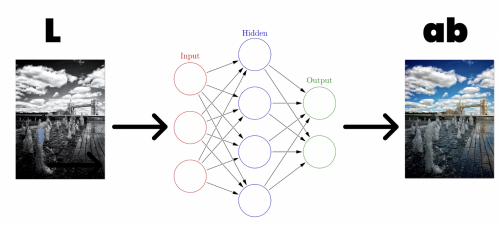
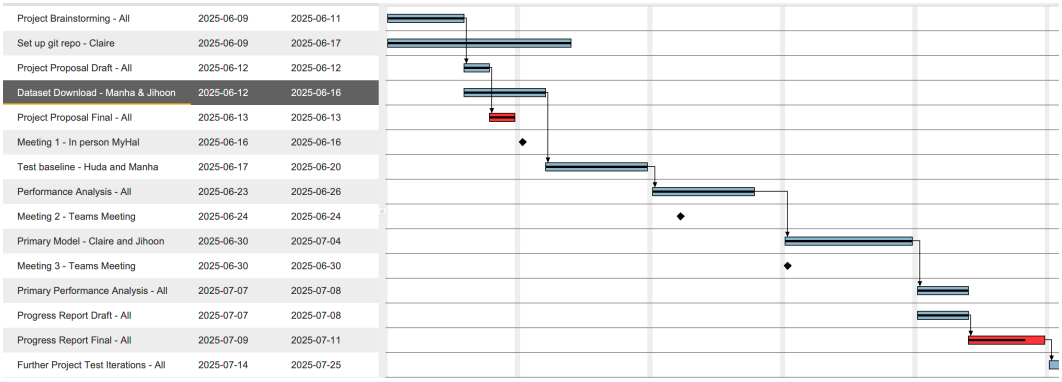


Figure 1: L channel as input to the CNN and outputting the predicted ab channels

2 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

Our team is consistent and collaboratively creating a deep learning model of image colorization by maintaining communication through group chats, in-person meetings, and Microsoft Team meetings throughout the weeks. In order to keep track of the progress, we use GanttProject as seen in Figure 2 to distribute our work, set deadlines, and update it for all members to see. Google Colab is used to manage, share, and update code among members.



Project Brainstorming - All	2025-06-09	2025-06-11
Set up git repo - Claire	2025-06-09	2025-06-17
Project Proposal Draft - All	2025-06-12	2025-06-12
Dataset Download - Manha & Jihoon	2025-06-12	2025-06-16
Project Proposal Final - All	2025-06-13	2025-06-13
Meeting 1 - In person MyHal	2025-06-16	2025-06-16
Test baseline - Huda and Manha	2025-06-17	2025-06-20
Performance Analysis - All	2025-06-23	2025-06-26
Meeting 2 - Teams Meeting	2025-06-24	2025-06-24
Primary Model - Claire and Jihoon	2025-06-30	2025-07-04
Meeting 3 - Teams Meeting	2025-06-30	2025-06-30
Primary Performance Analysis - All	2025-07-07	2025-07-08
Progress Report Draft - All	2025-07-07	2025-07-08
Progress Report Final - All	2025-07-09	2025-07-11
Further Project Test Iterations - All	2025-07-14	2025-07-25

Figure 2: Progress Report Distribution

The members divided work evenly among the team, and fairly in terms of the project and it's deliverables, as we can see in detail in table 1.

Table 1: Distribution of Image Colorization Project

Name	Responsibility	Accomplished
Claire He	Working on the Primary Model using the U-NET architecture	Successfully created a U-Net that uses encoders and decoders with skip connections for better training. Helped split the data from RGB to LAB and visualized the comparison for the CNN architecture
Huda Imran	Working on the Baseline Model using the CNN architecture and training it	Successfully created a CNN that includes encoder and decoders to capture features allowed for better training of the model. Due to the size of the actual Sharma Kaggle dataset created a smaller dataset and used it to train the model. Trained the model to find the MSE training loss, and the validation loss, Visualized the images and their comparison to the ground truth label.
Jihoon You	Splitting up Data for Data Processing Working on Training the Primary Model	Successfully split up the data from the Sharma Kaggle dataset into colored and grayscale and formatting all images to one size. Converted RGB images data to LAB color spectrum Trained the U-NET model to find the error and validation loss, and accuracy. Visualized image dataset for UNET and their comparison to the ground truth label
Manha Siddiqua	Working on the Baseline Model using the CNN architecture and training it Split up data processing	Using the processed data from RGB to LAB, split the images into Training, Validation, and Testing, with their ground truth label. Successfully created the CNN architecture to train the model using encoders and decoders. Due to the large dataset, created a smaller LAB dataset for training Trained the model to find the accuracy using a threshold comparison using the Euclidean distance from the predicted to ground truth label.

The next following weeks will be used to work on improving the primary model, and work towards the tasks set out for us, and completing them in a timely manner. Using GanttChart as our collaborative progress check and our task report as seen in Figure 3, shows what changes and improvements we will make to our project.
From Figure 3 we can see it is possible that there might possibly be issues with completing the project on time, as there are many tasks depending on the previous ones. One major task that could set us behind our deadlines is possibly the 'Different Iterations of Model Depth' and 'Add DropOut or Regularization to prevent overfitting' as they may require more time to work on and implement in the primary model. Knowing that these are heavier tasks we have assigned 2 people on each task, and have the other two members caught up and ready to help out. We also have a meeting before 'Different Iterations of the Model Depth' for any help from the previous tasks. We also have about a weeks time before our Final Presentation where the team can also work on the project and areas where they believe might need more work and improvements.

Further Project Test Iterations - All	2025-07-14	2025-07-14
Hyper Parameters Tuning - Claire and Huda	2025-07-14	2025-07-15
Add DropOut or Regularization, prevent overfittin...	2025-07-15	2025-07-17
Microsoft Team Meeting	2025-07-16	2025-07-16
Different Iterations of Model Depth - Claire and J...	2025-07-18	2025-07-22
Train on Full Dataset - Manha and Huda	2025-07-23	2025-07-24
Demo Preparation - All	2025-07-25	2025-07-25
Generate never seen before data - All	2025-07-25	2025-07-29
In Person Meeting	2025-07-28	2025-07-28
Qualitative and Quantitative Analysis - All	2025-07-30	2025-07-30
Final Performance Analysis - All	2025-07-31	2025-07-31
Project Iterations after Final Project Draft	2025-08-01	2025-08-06
Microsoft Teams Meeting	2025-08-04	2025-08-04
Presentation Recording - All	2025-08-08	2025-08-11
Final Report - All	2025-08-12	2025-08-15

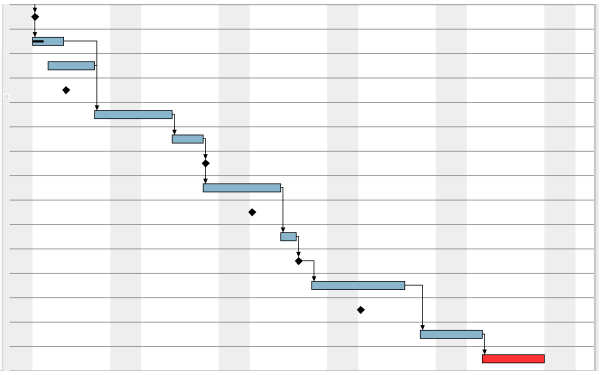


Figure 3: The teams tasks report and progress checks for the project

3 Notable Contribution

3.1 Data Processing

In our Project Proposal, we initially planned to use a 50,000 image subset of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset. However, due to the storage and GPU usage limitations we encountered on Google Colab, we opted to use a smaller dataset of 5739 images, found on Kaggle (Sharma). These images were downloaded and pooled into a single folder, where they were subsequently resized. While we initially proposed to resize images to 572x572; however, due to the aforementioned limitations encountered with Colab, we resized the images to 256x256 instead. The resized images were then converted into the CIELAB colour space, where the L value was normalized to [0, 1] and the ab values were normalized to [-127, 128]. These LAB values could not be saved as typical image formats, such as .jpg or .png, and had to be stored as .npz (NumPy Array) files. In the final step for data processing, the images were split into train-validate-test subsets using an 80-10-10 split.

For final testing of our model, we plan to introduce additional data by randomly selecting images from the aforementioned ILSVRC dataset. In addition, we plan to incorporate data augmentation (such as crops, rotations, and Gaussian Noise) in order to further examine the model's generalizability to "never before seen data."

3.1.1 Baseline Model

The baseline model was created as an encoder-decoder simple structure to provide a benchmark for the image colorization task. We did this because autoencoders are a generative learning model that can efficiently learn compressed representations of data (encodings) and then reconstruct the original input Kamoutsis (2020). Figure 4 shows how the model implements a simple Convolutional Neural Network (CNN) with a symmetrical encoder-decoder structure using transpose convolutions. Specifically, the encoder applies three convolutional layers with ReLU activations and max-pooling to progressively extract features from the greyscale image while reducing its dimensions. The three encoder layers expand the channel depth ($1 \rightarrow 64 \rightarrow 128 \rightarrow 256$) and downsamples the input (256×256 to 112×112 to 56×56 to 28×28). This compressed representation captures complex features such as edges and textures. The decoder then upsamples these features using three transposed convolutional layers: the layers upsamples (28×28 to 56×56 to 112×112 to 256×256) and reduces channels ($256 \rightarrow 128 \rightarrow 64 \rightarrow 2$) to output 2 ab channels at 224×224 resolution. A final tanh activation maps the output to the normalized [-1,1] range of Lab colour space.

We created a smaller dataset of 1,000 paired greyscale and colour images. The data set was divided into 60-20-20 splits for training, validation, and testing. Figure 5 and 6 show the training and validation loss (Figure 5) and accuracy (Figure 6) across 20 epochs. As seen, both training and validation loss decreases with the increase in epochs, indicating consistent learning, while accuracy increases, peaking around 45 percent. Accuracy was measured using a deltaE-based threshold that computes how close the predicted ab channels were to the ground truth values, offering a perceptually meaningful measure of colour prediction accuracy [].

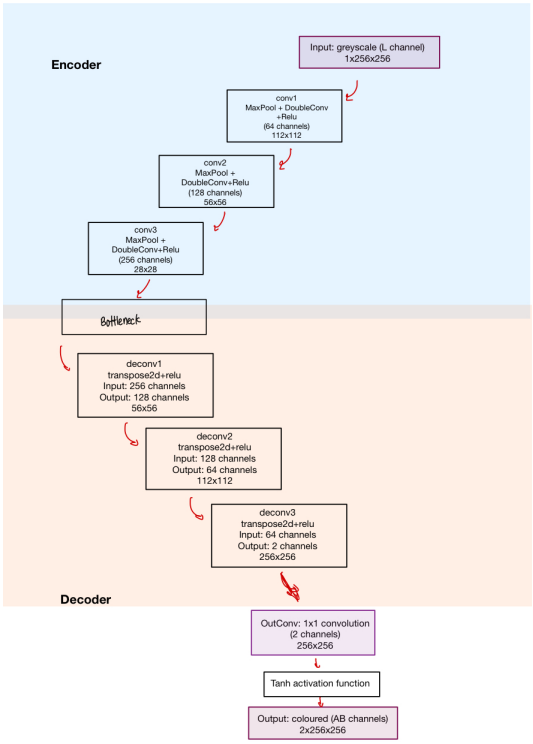


Figure 4: diagram of baseline model

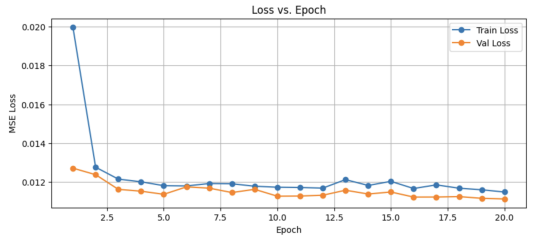


Figure 5: The baseline model train and validation loss for each epoch

Our qualitative data from our baseline model is shown in Figure 7, which displays a side-by-side comparison of 2 original images, the greyscale input (L channel), and the colourized outputs. The model was able to distinguish between different regions of the image and apply colour in a spatially coherent manner. For example, it correctly identified the airplane body and the sky as distinct regions, as well as finds the body of the giraffes compared to the sky and grass, applying some colour variations. The fence, ground, and background are also segmented and have some colour, demonstrating the model's ability to extract structure from the L channel data. However, the model struggled to create vibrant colours, particularly in the blue and green regions. The colourized outputs of the images appear more muted and brownish. This tells us that although the model learned the general placement and shading of colours, it had difficulty predicting saturation. This is likely due to the limited dataset size of or training. However, the result indicates that we are on the right track and that the baseline model can generate meaningful colorizations, even if it lacks vividness and fine detail. It can be refined and finetuned more for our primary model.

One of the main challenges we faced was due to computational constraints as dueto our large dataset and use of the CPU, training time was very high and limited our ability to experiment with deeper architectures. To solve this issue, we created a small dataset with the images to ensure that the baseline architecture could still learn meaningful colour mappings within our constraints. Additionally,

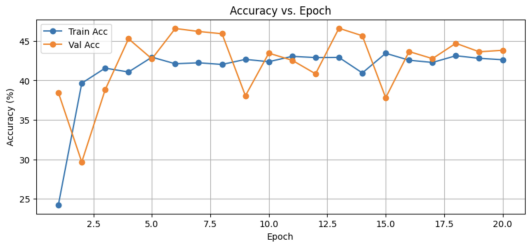


Figure 6: The baseline model train and validation accuracy for each epoch

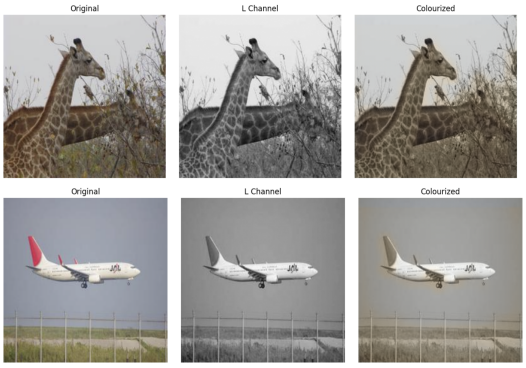


Figure 7: Enter Caption

computing the accuracy of the model was hard as it is a regression problem, and it was challenging to define a clear accuracy that could reflect how close the predicted colours are to the ground truth. Unlike classification tasks, colourization involves predicting continuous values for each pixel's ab channels. We solved this by implementing an accuracy metric based on deltaE which is the Euclidean distance between the predicted and true ab channels in the LAB colour space.

3.1.2 Primary Model

Our primary model for image colorization employs a U-Net architecture, which is well-suited for this task due to its encoder-decoder structure with skip connections that help preserve spatial information important for accurate color prediction. We decided on this architecture because in tasks like colorization, skip connections are particularly useful in UNET, where they bridge the encoder and decoder layers. This way, the encoder compresses the input image, while the decoder reconstructs it. Furthermore, skip connections help the decoder access high-resolution features from the encoder, allowing for better preservation of details like edges and textures Sayeed et al. (2025). Our U-Net model consists of a contracting path (the encoder) with 5 downsampling blocks that progressively reduce spatial dimensions while increasing channel depth (from 64→128→256→512→1024 channels), followed by an expanding path (the decoder) with 4 upsampling blocks that reconstruct the full resolution output. This neural network implementation contains many different layers, including convolutional layers, batch normalization, and ReLU activations. Furthermore, our architecture takes grayscale L-channel images (1×256×256) as input and outputs AB color channels (2×256×256) with tanh activation to constrain values to the appropriate [-1,1] range for LAB color space.

Since our UNET structure scales up to 1024 channels in the bottleneck and we are also using double convolutional blocks, parameters reach up to the millions. Around 17 million based on the output of our model currently. Expanding more, we used double convolutional blocks because it helps us understand larger spatial contexts. Basically, double convolutions help ensure neighboring pixels with similar textures get similar colors. The double convolution blocks in our architecture follow the regular U-Net design pattern where each block consists of two consecutive 3×3 convolutions, each followed by batch normalization and ReLU activation as seen in Figure 8. This block is repeated and,

after each down-sampling, the number of filters in the convolution is doubled Ange Lou (2025). This is important because it allows the network to capture increasingly abstract and semantic features at deeper levels. For our colorization project specifically, this design choice is beneficial because they enable the network to learn complex mappings from grayscale to the appropriate colour predictions. The shallow layers with fewer channels will focus more on local texture patterns and boundaries, while the deeper layers with more channels develop more sophisticated understandings of objects and scenes that are essential for predicting realistic colours.

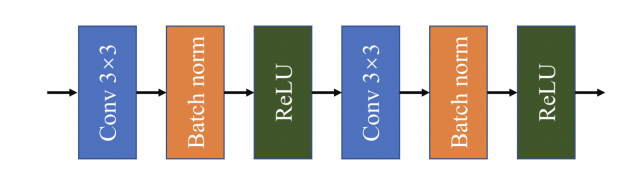


Figure 8: Diagram of double convolutional blocks Liu et al. (2020)

As you can see, figure 9 illustrates our U-Net architecture for image colorization, which follows a symmetric encoder-decoder design with skip connections that help preserve spatial details. The encoder progressively downsamples the input grayscale image (1×256×256) to a bottleneck, while the decoder upsamples back to the original size to output 2 AB color channels.

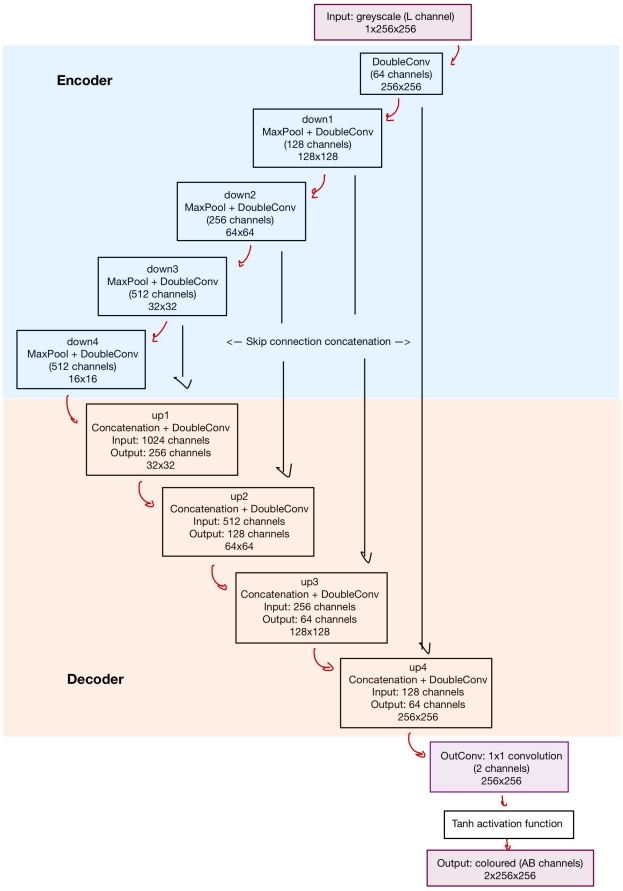


Figure 9: Diagram describing our model's architecture

Figure 10 shows the training results show strong convergence and learning over the 20 epochs, with both training and validation MSE losses stabilizing around 0.010-0.011. The training loss (blue) shows a steady decrease, while the validation loss (red) starts higher and converges to similar levels as the training loss, indicating good generalization without overfitting. The close alignment between training and validation curves throughout the training process indicates that the model is generalizing well to unseen data, and the lack of divergence between the two curves means that the model's architecture and training parameters are good for our task of colourization.

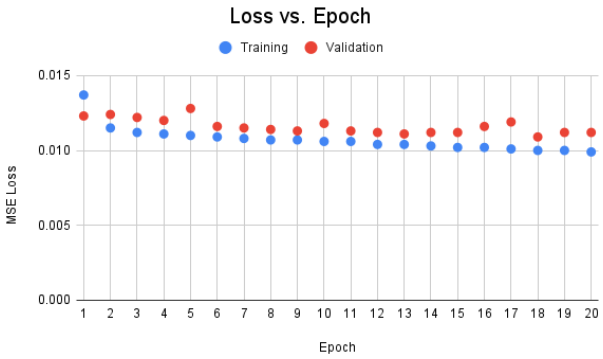


Figure 10: MSE loss per epoch

In addition, in figure 11, the qualitative comparison shows some interesting patterns in our U-Net model's colorization output, which demonstrates both its strengths and limitations. The successful colorization example (left) demonstrates the model's ability to colorize outdoor nature scenes, where it accurately colourized earth tones for the ground, realistic colors for the horses, and appropriate sky coloring. Even in the unsuccessful colorization example (right), it was able to correctly colourize the more natural aspects of it, like the sky. However, the right example also shows the model's difficulty with more artificial, man-created environments, particularly the tennis court scene where it failed to restore the blue court's surface. This suggests that our model performs better on natural scenes, likely because these types of images were more prevalent in the training dataset, while it struggles with scenes containing artificial materials that require more precise color prediction. The comparison indicates that the model has learned to associate certain grayscale patterns with natural color schemes but it may need additional training and improvements to handle more diverse scenes effectively.

One of the most challenging aspects of implementing the U-Net for our project was figuring out the correct channel dimensions throughout the network, especially in the decoder path. For the encoder it was straightforward going from $1 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 512$ (due to the factor = 2). But the decoder gets tricky due to the skip connections which significantly alter the expected input channels. The decoder doesn't just upsample the previous layer's output, it concatenates it with the corresponding encoder feature map, doubling the channel count at each skip connection. For example, while the encoder bottleneck outputs 512 channels, the first decoder block must handle 1024 channels (512 from the encoder + 512 from the skip connection) before reducing them back down to 256 channels for the next stage. This meant carefully calculating the input channel dimensions for each Up module to properly handle the concatenated features. In addition, due to pooling and upsampling operations, the spatial dimensions didn't always match perfectly. For example, encoder feature maps might be 129×129 while the decoder expects 128×128. To fix this, we needed to add dynamic padding calculations during our forward function.

4 Link to Repository

Colab Notebook Link:
Click here to view Colab page

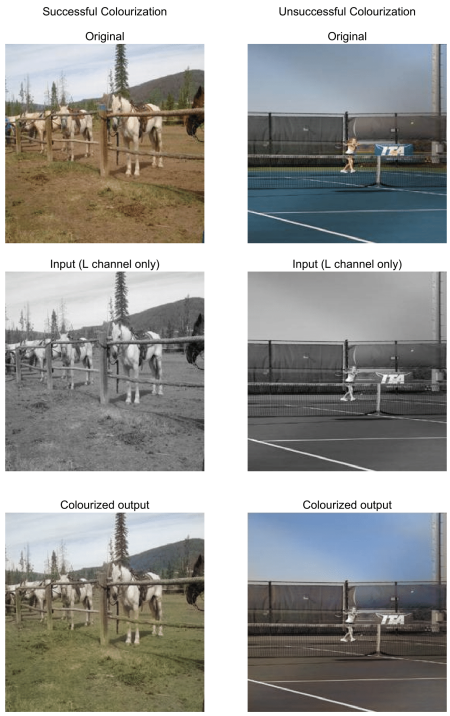


Figure 11: Qualitative comparison of successful and unsuccessful colourizations

REFERENCES

Murray Loew Ange Lou, Shuyue Guan. Dc-unet: Rethinking the u-net architecture with dual channel efficient cnn for medical images segmentation. *Department of Electrical and Computer Engineering, 2Department of Biomedical Engineering*, 2025. URL https://arxiv.org/pdf/2006.00414.

George Kamoutsis. Building an image colorization neural network - part 1: Generative models and autoencoders. https://medium.com/@geokam/building-an-image-colorization-neural-network-part-1-generative-models-and-autoenc 2020. Accessed: 2025-07-11.

Wei Liu, Fulin Su, Xinfei Jin, Hongxu Li, and Rongjun Qin. Bispace domain adaptation network for remotely sensed semantic segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–11, 11 2020. doi: 10.1109/TGRS.2020.3035561.

Lidija Mandić, Jesenka Pibernik, Maja Kurečić, and Ana Cmrk. Artificial intelligence or man: Colorization of black and white photographs. 11 2024. doi: 10.24867/GRID-2024-p34.

Khondker Salman Sayeed, Haz Sameen Shahgir, Tamzeed Mahfuz, Satak Kumar Dey, and M Saifur Rahman. Efficient real-time video colorization on low-end cpus via pruning and quantization. In *Proceedings of the 11th International Conference on Networking, Systems, and Security*, pp. 146–153, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711589. URL https://doi.org/10.1145/3704522.3704536.

Aayush Sharma. Image colorization dataset. URL https://www.kaggle.com/datasets/aayush9753/image-colorization-dataset.