CS 464: Introduction to Machine Learning

Final Presentation

Group 2

Beril Bayram – ID Numbers

Cem Daloğlu –

Kenan Korhan Odabaş – ID Number

Damla Kartal – ID Numbers

Tuna Özcan –

1. Introduction

- For this project, grayscale images were converted into their colored versions using 3 different Convolutional Neural Network (CNN) algorithms:
  - Vanilla (Chosen as baseline)
  - ResNet
  - U-Net
- **Results:** ResNet > U-Net > Vanilla CNN

2. Problem Description

- The aim is to see whether grayscale images can be colored fully using Machine Learning.

- This project also aims to test numerous algorithms and to see which algorithm is the best for the colorization of images.

3. Methods

- The dataset
  - Contains grayscale and color versions of 25.000 224 x 224 pixel images [1].
  - Includes the colored images in LAB color space.
    - The L channel encodes the strength of light from black (0) to white (100).
    - The A channel encodes green (-) to red (+)
    - The B channel encodes blue (-) to yellow (+)

3. Methods

- The dataset categorized into:
  - Training set
  - Validation set
  - Test set
- Preprocessing and normalization is applied.
- During the train process,
  - The LAB color and gray values of each image are mapped by the algorithm.
- During the test process,
  - An unseen gray image is used as an input while the color image is predicted by the model as output.

3. Methods

- Vanilla CNN
  - Various characteristics of images can be detected and learned by CNN algorithms that can have tens to hundreds of layers. Each layer learns to recognize various features of an image.
- Implementation of Vanilla CNN
  - The Keras API which runs on TensorFlow was used.
  - Two different architectures were applied for Vanilla CNN.
    - These architectures were compared to get better results.

3. Methods

- Vanilla CNN Architecture 1
  - The ReLU activation function was used
    - 2D convolutional/transpose layers were activated and these layers were applied with different kernel and filter sizes.

Model: “sequential”	
Layer (type)	**Output Shape**
conv2d (Conv2D)	(None, 222, 222, 32)
conv2d_1 (Conv2D)	(None, 222, 222, 16)
conv2d_transpose (Conv2DTran)	(None, 224, 224, 2)

Table 1: First Model Architecture

3. Methods

- Vanilla CNN Architecture 2
  - The difference between Architecture 1 and 2
    - A 2D max pooling layer was also applied in addition to 2D convolutional/transpose layers
      - To increase the complexity of the model.

Model: “sequential”	
Layer (type)	**Output Shape**
conv2d (Conv2D)	(None, 222, 222, 32)
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)
conv2d_1 (Conv2D)	(None, 109, 109, 16)
conv2d_transpose (Conv2DTran)	(None, 224, 224, 2)

Table 2: Second Model Architecture

3. Methods

- ResNet
  - First half is the Resnet18, the second half has 14 layers for upsampling [2].
  - Training set contains 4000 preprocessed data, 800 of the data is splitted as validation set.
  - Tested with 100 unseen test data.
  - Training is applied with batch size equal to 32 and 250 epochs.

3. Methods

- U-NET
  - Contraction path (encoder path) and symmetric expanding path (decoder path) have both 16 layers [3].
  - Training set contains 4000 preprocessed data, 800 of the data is splitted as validation set.
  - Tested with 100 unseen test data.
  - Training is applied with batch size equal to 16 and 250 epochs.

4. Results

- **Vanilla CNN**

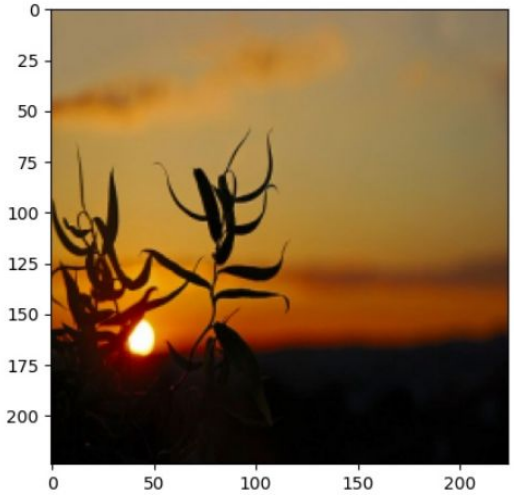

Figure 1: Actual Image

SSIM: 0.573

PSNR: 9.548

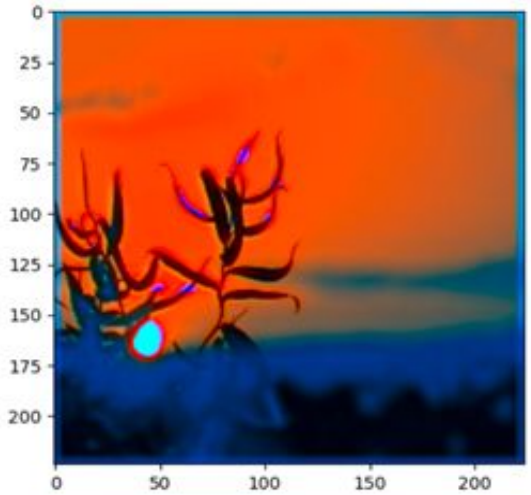

Figure 2: Predicted Colored Image

4. Results

- **Vanilla CNN Train and Test MSE Loss**

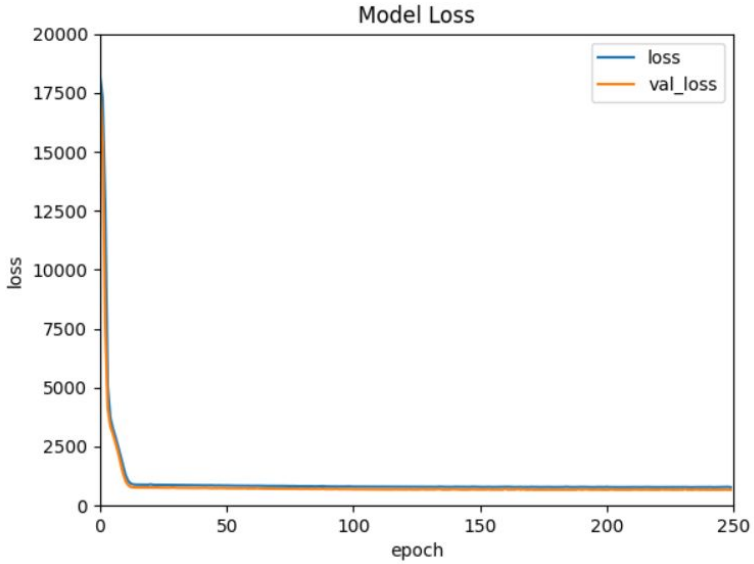

Figure 3: Vanilla CNN Train and Test MSE Loss

4. Results

- **ResNet**

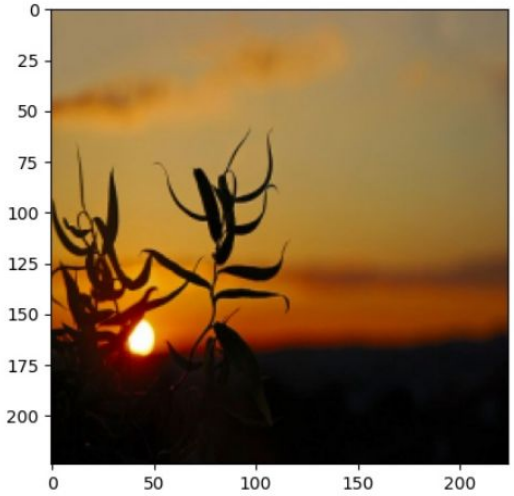

Figure 4: Actual Image

SSIM: 0.573

PSNR: 9.548

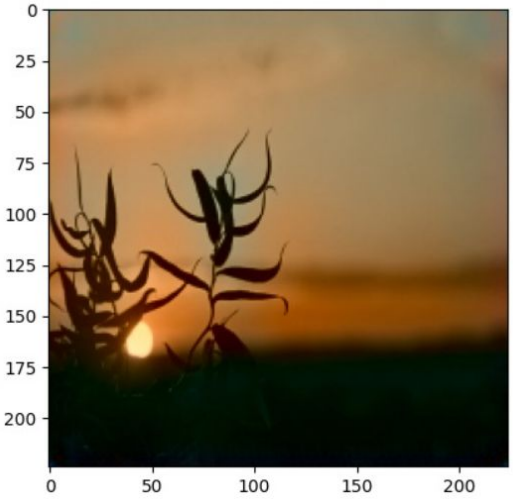

Figure 5: Predicted Colored Image

4. Results

- **ResNet**

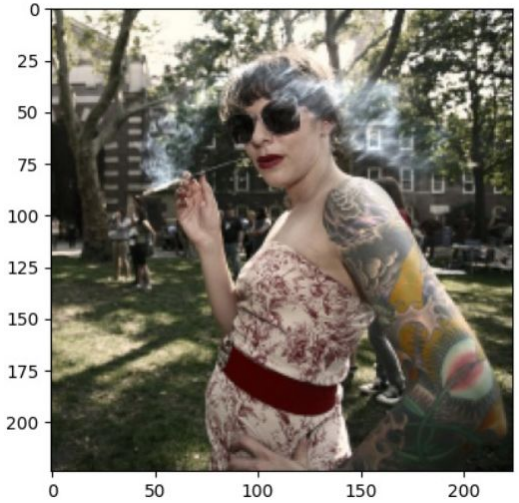

Figure 6: Actual Image

SSIM: 0.4757

PSNR: 11.659

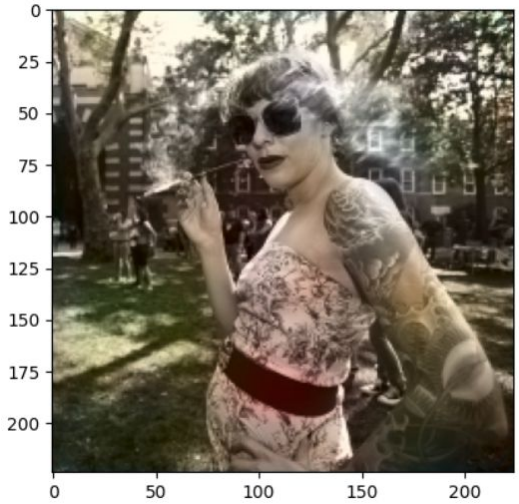

Figure 7: Predicted Colored Image

4. Results

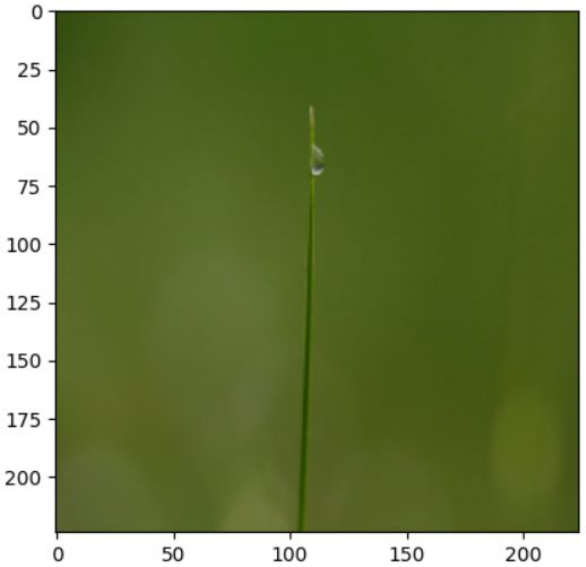

Figure 8: Actual Image

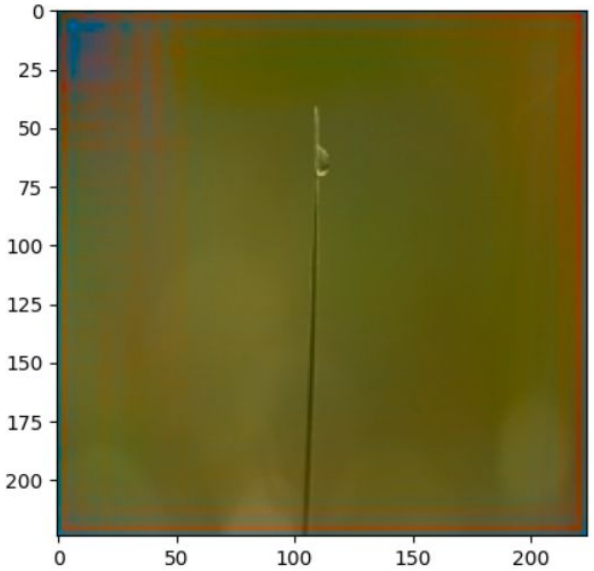

Figure 9: Predicted Colored Image

4. Results

- **ResNet Train and Test MSE Loss**

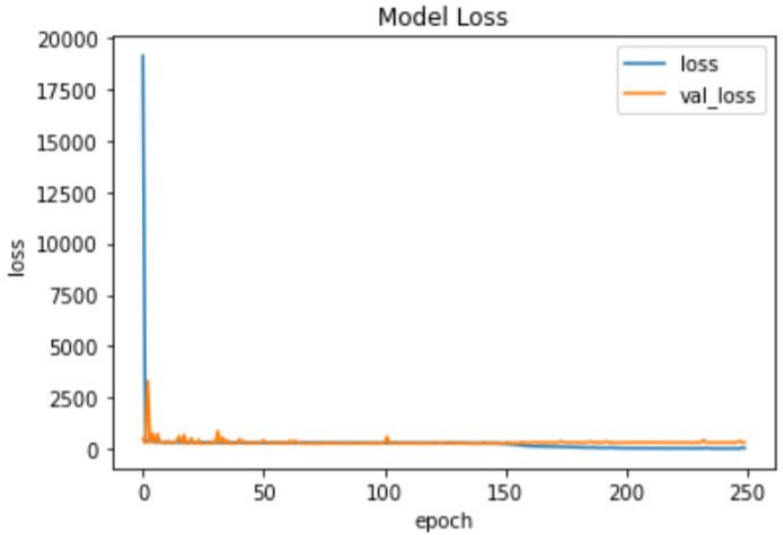

Figure 10: ResNet Train and Test MSE Loss

4. Results

- **U-Net**

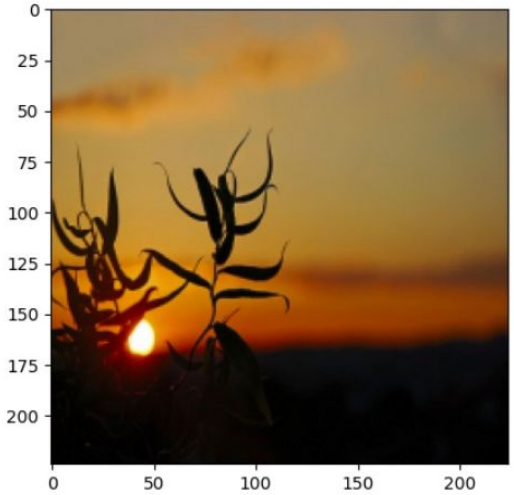

Figure 11: Actual Image

SSIM: 0.573

PSNR: 9.548

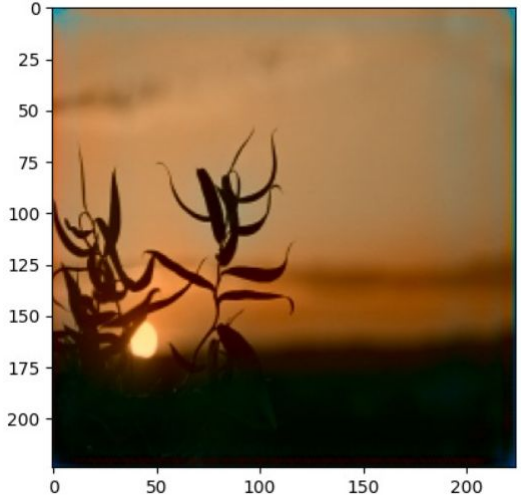

Figure 12: Predicted Colored Image

4. Results

- **U-Net**

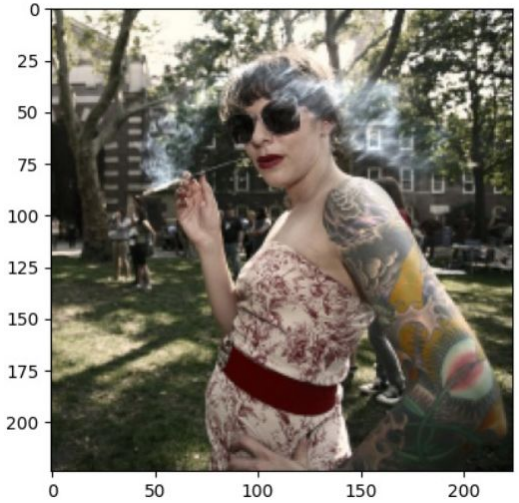

Figure 13: Actual Image

SSIM: 0.4757

PSNR: 11.659

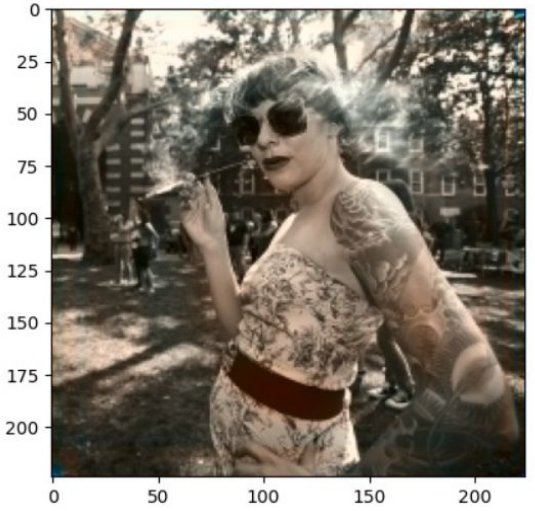

Figure 14: Predicted Colored Image

4. Results

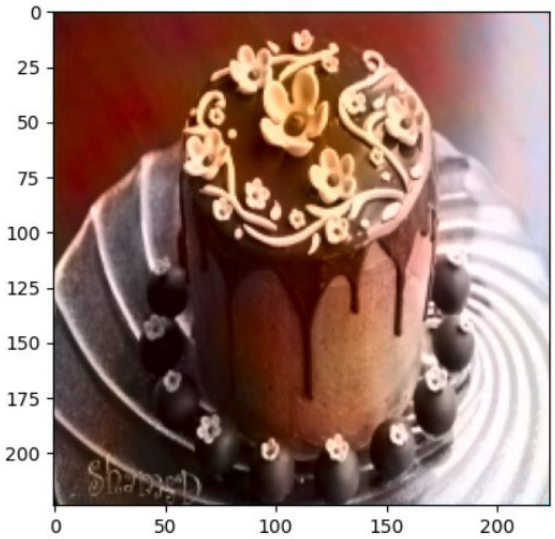

Figure 15: Actual Image

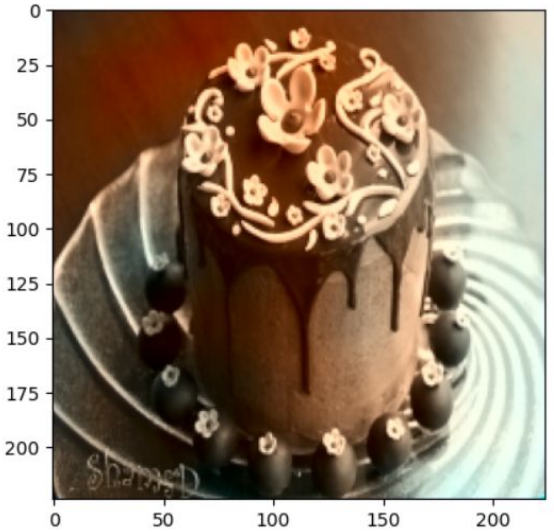

Figure 16: Predicted Colored Image

4. Results

- **U-Net Train and Test MSE Loss**

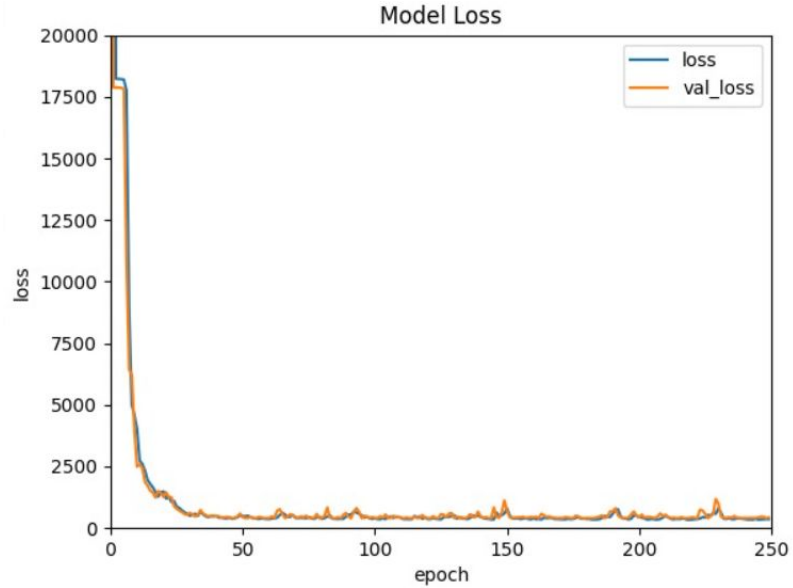

Figure 17: U-Net Train and Test MSE Loss

5. Discussion

- ResNet model performs better than the U-Net and Vanilla CNN models.
  - The subjective results are also supported by mathematical accuracy metrics.

- ResNet algorithm performed better colorization operation compared to the U-Net model in terms of predicting the original image color spaces.

- SSIM and PSNR values are the same for all models.
  - This means that, none of the models add any noise or blurriness to the predictions.

5. Discussion

- Some of the output images were grayscale.
- Some of the colored images were included in gray scale on the colorized image set.
- After preprocessing, gray tone was eliminated in the predictions.
- Yellow and brown tones were dominantly observed on the colored images.

5. Discussion

- Dataset contains mostly yellow and orange colors
  - Best results for images with sun or sun light.
  - All predictions have a general yellow tone which causes high prediction errors.

- The test set and train set should contain similar colored pictures for better predictions.

6. Conclusion

- Our aim was to colorize grayscale images and with described methods we achieved our goal.
- Grayscale images can be colored mostly but not fully.
- ResNet model has better accuracy overall images than U-Net and Vanilla models.
- U-Net model is also applicable for colorization for some grayscale images
  - **Successful:** Images with sunlight.
  - **Unsuccessful:** Images with green color.
- The Vanilla model is not applicable
  - Could not colorize the grayscale images detailly.

7. References

[1] "Image Colorization." https://kaggle.com/shravankumar9892/image-colorization (accessed Nov. 15, 2020).

[2] Lukemelas. (n.d.). Lukemelas/Automatic-Image-Colorization. Retrieved December 19, 2020, from https://github.com/lukemelas/Automatic-Image-Colorization/

[3] Rahuldshetty. (2019, October 23). Image Colorization with UNET-Auto Encoders. Retrieved December 19, 2020, from https://www.kaggle.com/rahuldshetty/image-colorization-with-unet-auto-encoders/notebook