Analyzing Bias in Human Portrait Colorization Using Conditional Wasserstein GANs

Edward Deleu
Department of Electrical and Computer Engineering
Princeton University
Princeton, NJ 08544
edeleu@princeton.edu

Vishva Ilavelan
Department of Computer Science
Princeton University
Princeton, NJ 08544
vi6908@princeton.edu

***Abstract*—Colorizing grayscale human portraits is an important task well suited for generative modeling, allowing for the realistic generation of new image data. The deep biases associated with many conventional machine learning datasets often come into view, considering the importance of fairly generating image data for individuals from diverse backgrounds. This paper implements a conditional Wasserstein Generative Adversarial Network (cWGAN) to realistically colorize human portraits, trained on the `AISegment.cn` human portrait dataset. This work demonstrates that, despite the underlying biases in the training data, the network can be implemented and designed so that it performs well on diverse demographics of race, age, and gender, generating convincing output images. An extensive analysis is performed on images colorized through this network from a variety of human portrait datasets.**

I. Introduction

Image colorization, or the task of colorizing grayscale images, is an example of a generative image-to-image translation task. More specifically, models are tasked with transforming or translating given grayscale images to their colorized counterparts [1].

A. Generative Computer Vision Models: GAN and WGAN

As an image generation sub-problem, approaching the task of image colorization involves leveraging generative computer vision models. Generative Adversarial Networks (GANs) and Wasserstein Generative Adversarial Networks (WGANs) form the backbone of the proposed image colorization system [2]–[4].

The GAN is a generative computer vision framework proposed by [2] that tackles image generation through the training of two adversarial models, called a generator and discriminator. The generator is tasked with generating synthetic images from a learned distribution of image data, whilst the discriminator aims to distinguish real images (present within the training dataset) from synthetic images created by the generator. During training, these two models are placed in direct opposition. While the discriminator aims to distinguish generated and real images, the generator works to fool the discriminator by generating images that are in line with the training data distribution. Thus, the training objective for a standard GAN can be expressed as follows [2]–[4]:

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_R}[\log(D(\boldsymbol{x}))] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_G}[\log(1-D(\tilde{\boldsymbol{x}}))] \quad (1)$$

where G is the generator, D is the discriminator, $\mathbb{P}_G$ is the generator's learned data distribution, and $\mathbb{P}_R$ is the distribution of the training data. Note that the output of the discriminator corresponds to a scalar value and that the standard GAN objective uses Binary Cross Entropy loss over two image classes: real and fake [2], [4].

A key problem with the standard GAN hitherto described is the presence of training instability which arises due to the standard GAN objective's discontinuity during optimization. The Wasserstein Generative Adversarial Network (WGAN) provides a solution through a different loss objective that is built on the Wasserstein or Earth-Mover distance which provides a characterization of the cost of moving between two distributions. This new objective is fully continuous and mostly differentiable, thus leading to better optimization [3], [4]:

$$\min_G \max_{D\in\mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_R}[D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_G}[D(\tilde{\boldsymbol{x}})] \quad (2)$$

Note that the discriminator aims to maximize this cost, maximizing the distance of the output between real and generated images, hence confidently differentiating between the two classes.

The newly introduced symbol $\mathcal{D}$ corresponds to the set of all 1-Lipschitz functions. The WGAN discriminator D objective is constrained to the set of 1-Lipschitz functions, which is enforced during training through weight clipping in the discriminator. Although an imperfect method, this constraint is necessary to maintain the continuous and differentiable properties of the WGAN objective [3], [4]. For the purpose of this image colorization system, a modified version of this objective is implemented, primarily for its useful optimization properties when compared with the standard GAN objective. More advanced methods of constraining the discriminator exist with improved characteristics, as proposed in [4], by using gradient penalty. However, this method adds significant complexity and was not required in this paper.

B. Image Colorization: Prior Approaches Utilizing GANs

The usage of GANs in the context of image colorization has been pursued before. The work `pix2pix` [1] presents the idea of applying conditional GANs for broader image-to-image translation tasks, including image colorization.

The conditional GANs utilized by Isola et al. [1] work to learn an image-to-image mapping. While traditional GANs utilize a sampled noise vector to generate output images, conditional GANs utilize a reference input image when generating output. In this image colorization system, for instance, this corresponds to providing the generator with a grayscale image which is utilized to generate a colorized output image. Moreover, Isola et al. also modify the traditional GAN objective to better suit image-to-image translation tasks, mainly by incorporating reconstruction error in the form of $L1$ loss for generator training [1]. These modifications and the basic architectural topology are incorporated in this paper in combination with the WGAN loss objective.

C. Lab Color space

The *Lab* color space is utilized in the proposed colorization system. Similar to the RGB color space, *Lab* consists of three distinct image channels L, a, b, where each channel provides distinct information about a given image. The L channel, encodes the lightness or luminosity information of a given image and takes values from 0 to 100, with 0 representing black and 100 representing white. In contrast, the a and b channels encode the non-grayscale color information and range from -128 to 127 [6], [7].

When performing colorization, this system's generator is conditioned on a given image's L channel and works to generate its corresponding a and b channel information. These three channels are concatenated to get the final generated colorized image which may either be fed into the discriminator or converted and saved in the standard RGB format [6].

II. cWGAN Architecture

This paper implements a conditional variant of the WGAN network, abbreviated cWGAN, in order to support the colorization of specific grayscale images. Both a generator and discriminator are utilized to be trained in the typical adversarial manner with the Wasserstein methodology. The architecture is implemented in Python with the PyTorch machine-learning library and is most heavily inspired by `pix2pix` [1]. Before passing data to the generator or discriminator, the L channel is normalized to the range -1 to 1 from the original 0 to 100 range, to improve stability and network properties. The a and b channels are similarly normalized from -1 to 1 from the original -128 to 127 range.

A. Generator

The objective of the generator is to generate a and b color channels, conditioned on an image's L channel which is simply the grayscale image. By using this implementation, the generator is not required to reproduce the structure from the original image and may simply focus on the color channels. Past implementations that attempt to reproduce the entire color

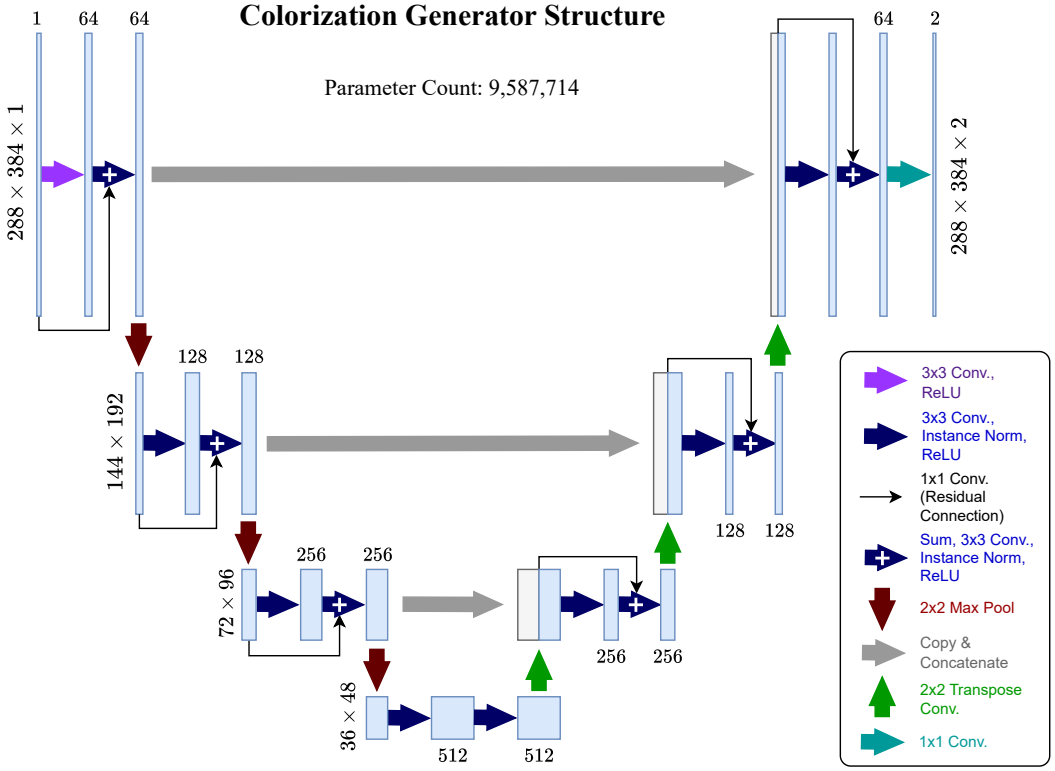


Fig. 1. The structure of the conditional Generator implemented with a residual U-Net structure. The generator accepts an L image channel and generates the corresponding a and b color channels. The diagram structure is adapted from [5].

output image struggle and often produce blurry outputs [8]. The generator follows a residual U-Net structure, depicted in Fig. 1, which is typical and well suited for image-to-image translation tasks [1]. The diagram displays how the dimensionality of the data changes throughout the network, assuming a $288 \times 384 \times 1$ ($W \times H \times C$) input map. However, the generator supports input images of arbitrary heights and widths divisible by 8 with a single L channel. The generator will still precisely produce the correct dimensionality for the a and b color maps for the arbitrarily sized input maps. The left half of the generator is the encoder and the right half of the generator is the decoder. These sections are connected with a "bottleneck" region at the bottom with reduced dimensions compared to the input map. The encoder serves to increase the number of channels of the data while distilling information and decreasing the dimensionality, keeping the overall volume of data constant. The decoder serves to expand the dimensionality to construct the desired output. Besides the residual connections between the encoder and decoder, each level contains an additional residual connection which is displayed with thin black arrows. This connection is implemented with a 1×1 conv. which effectively expands the number of channels in a level's input to match the number of output channels before the sum operation is performed. For example, in the first residual block, the 1×1 conv. transforms an input map of dimension $288 \times 384 \times 1$ to a volume of $288 \times 384 \times 64$.

Each level of the encoder and decoder can be considered a "residual block" which includes two convolution operations, with padding utilized to maintain dimensionality. The first convolution expands the channel dimension. Then, the residual map referenced above created with the 1×1 conv. is added through element-wise addition, and the second convolution is performed. More specifically, after each convolution operation, there is instance normalization which normalizes the data associated with each batch item passed through the network. This is followed by applying the ReLU activation function. Instance normalization is not performed after the network's first convolution, as indicated by the magenta arrow, so that the structure and scale of the input data is better maintained at the early stage of the network.

Between residual blocks in the encoder, a 2×2 max pooling operation is performed to precisely cut in half the dimension of the latent data. From the network's input to the bottleneck section, the channel dimension is expanded from 1 to 512. The decoder section then expands the dimensionality through the transposed-convolution operation, exactly doubling the data dimensions at each successive level and cutting in half the number of channels. The decoder blocks are residually connected to the encoder: after each transposed-convolution, the residual data is concatenated with the upscaled data before the convolution is performed. Then, the convolution effectively compresses the number of channels, for example from $256 + 512 \rightarrow 256$ at the first (bottom) residual block in the decoder. Once the data has returned to its input dimensionality and two convolutions are performed in the final residual block, a mapping is performed through another 1×1 conv. to reduce the channels from 64 to 2, as indicated with the cyan arrow.

Unlike more traditional GAN generators, this generator does not take in any random noise as input. The authors of the `pix2pix` methodology [1] found that the adversarial training of the U-Net structure caused the noise to simply be averaged out, making no impact on the architecture's output. Thus, as in the `pix2pix` paper, randomness in the training of the network is implemented in the form of dropout between each of the residual blocks in the network. Dropout is turned off during inference, since it sometimes causes color artifacts and decreases the quality of the colorizations.

The PyTorch implementation for the cWGAN generator structure is adapted from [6] to apply to this colorization task with some further architectural modifications such as adding instance normalization, utilizing transpose convolutions, and handling the input and output data differently.

B. Discriminator

The discriminator accepts an *Lab* color image as input and outputs a single scalar value, with the goal of maximizing the distance between the real and fake image distributions.

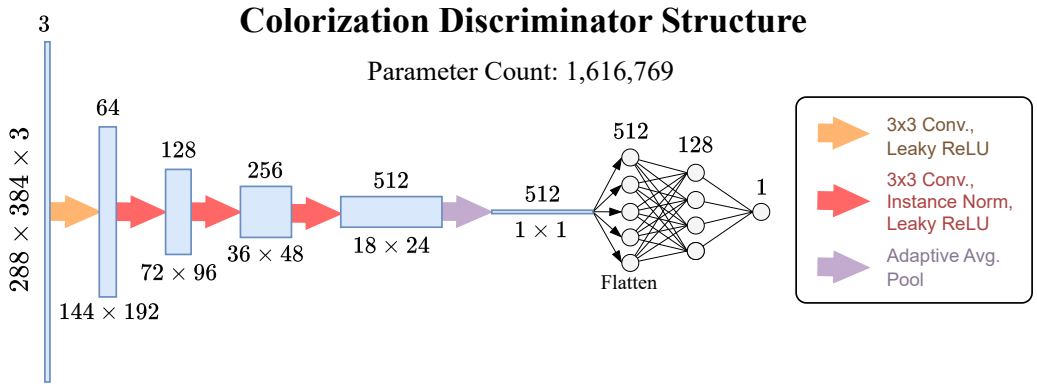


Fig. 2. The structure of the cWGAN Discriminator implemented with a convolutional structure. The discriminator accepts an L, a, b image channels and outputs a single scalar value.

Although it is not constrained with a sigmoid function as in standard GAN setups, it will still be referred to as the discriminator. The discriminator is implemented with a convolutional structure, as represented in Fig. 2. The convolutional nature of the discriminator allows it to focus on patches of the images, distilling important properties of the image while reducing the dimensionality and increasing the number of channels. Each convolution operation is followed by instance normalization and then the Leaky ReLU activation function. This activation is used in place of ReLU as done in the `pix2pix` paper [1]. One note that, as done in the generator, normalization is not performed in the convolution directly after the input layer. At the end of the convolutional layers, the dimensionality is $18 \times 24 \times 512$ which is reduced to $1 \times 1 \times 512$ through an average pooling operation. Since the PyTorch operation is adaptive, it supports arbitrary input dimensions. Thus, the entire discriminator supports images of arbitrary width and height. The pooled vector is then flattened and passed on as input to a 2-layer fully connected neural network. This neural network then outputs the single scalar value.

The PyTorch implementation for the discriminator is adapted from [6] with modifications to the neural network, normalization, network dimensionality, and data-handling.



Fig. 3. Sixteen randomly selected images from the `AISegment.cn` portrait dataset.

III. Dataset and Data Processing

The `AISegment.cn` Matting Human dataset contains 34,427x 600×800 RGB images and corresponding "matted" images which have the backgrounds removed [9]. The creators of this dataset, Beijing Play Star Convergence Technology Company, retrieved these images from the Chinese marketplace platform TaoBao, the Chinese search engine Baidu, and the Flickr image hosting site. Fig. 3 displays randomly selected original images from this dataset. The dataset's photos are very consistent and high quality; they depict individuals from the shoulders up, with the faces generally in the center of the frame. There are no occlusions and faces are typically angled directly toward the camera. Of course, there are some exceptions, and a small fraction of the original data is of lower quality or in grayscale. It is worth noting that the dataset is heavily biased towards White and East Asian individuals: there is significantly less representation of Black, Latino, Middle Eastern, and Indian individuals. While other portrait datasets exist with more balanced distributions of data, such as FairFace or UTKFace, the quality of image data is often significantly lower which would result in a much more challenging training process [9]. There are also portrait datasets that only contain data for a certain race, such as the Asian face dataset [10], but these datasets are also not well-suited for the general portrait colorization task.

For this colorization task, only the original portrait images are utilized from the `AISegment.cn` dataset. To train the architecture, these images were resized to 288×384 in order to maintain the same aspect ratio but shrink the amount of data. This serves to reduce the amount of data flowing through the model, which consequently requires less memory during training. Training models with image datasets in batches inherently requires a lot of memory, so this is a crucial step. This step also speeds up training since it significantly reduces the volume of data flowing through the network.

Minimal data transformations were applied to the training images since the dataset was sufficiently large. To ensure model robustness, images were randomly mirrored in the horizontal direction.

IV. Framework Training

The entire architecture has undergone several iterations before arriving at the proposed method. In total, over 160 hours of training has been completed to improve the architecture and find optimal hyperparameters using Princeton's Della and Adroit clusters with A100 GPUs. Model architectures aside, the training methodology remained relatively constant. The Wasserstein minimax objective given in Eq. 2 can directly be applied to the colorization task with the *Lab* color space for training the architecture. The generator's Wasserstein loss can be defined as:

$$\mathcal{L}_{WGAN,Gen} = \frac{1}{N}\sum_{i=1}^{N} -D(L_i, G(L_i)) \tag{3}$$

where the sum is finding the average over each item in the batch. The L1 loss is also computed between the generated

and reference a and b color channels. The overall generator loss is a linear combination of the Wasserstein loss and the L1 reconstruction loss, to ensure that the architecture produces realistic colors that are authentic to the ground-truths. The overall generator loss can be written as:

$$\mathcal{L}_{Total,Gen} = \mathcal{L}_{WGAN,Gen} + \lambda \mathcal{L}_{L1} \tag{4}$$

where λ is a hyperparameter that defines the relative importance of the L1 loss. The discriminator has the opposite objective, to maximize the distance between discriminator outputs on real and fake images:

$$\mathcal{L}_{WGAN,Discrim} = \frac{1}{N}\sum_{i=1}^{N} -D(L_i, a_i, b_i) + D(L_i, G(L_i))) \tag{5}$$

The model architectures are trained using mini-batches of data randomly extracted from the test dataset. As in the original WGAN paper [3], the discriminator is trained for $n_{discrim}$ iterations before the generator is trained for one iteration. After the discriminator parameters are updated, the weights are clipped within $[-0.01, 0.01]$ as proposed in [3] for the 1-Lipschitz constraint. The Adam optimizer is used with $\beta_1 = 0, \beta_2 = 0.9$ as suggested in [4]. The β_1 term is the weighting for momentum which typically destabilizes the WGAN learning process so it is not used. The most significant hyperparameters tuned during the experimentation phase include learning rate, batch size, $n_{discrim}$, λ, and the proportion of the `AISegment.cn` dataset used as the training dataset. Python code for the WGAN training is roughly inspired by [11] and [12], although the training setup is mostly custom.

A. Initial Framework Variations

Before arriving at the proposed model architecture, simpler variations were experimented with. Namely, the discriminator did not contain instance normalization and the generator contained batch normalization at all layers. The initial generator setup contained approximately 2-million network parameters. However, this yielded quite poor results, so the model capacity was increased to the currently proposed layer sizes as shown in Fig. 1. Hyperparameter optimization was manually performed with this setup, yielding optimal results with $LR = 5 * 10^{-5}, \lambda = 10, n_{discrim} = 5$, and leaving the weight-clipping constrained to $[-0.01, 0.01]$. If this clipping range was at all increased, the losses would continuously grow and the model would fail to learn. Using smaller λ values would result in more vibrant, yet less realistically colorized images by giving more weight to the GAN loss instead of the reconstruction loss. Reasonable results were achieved using 15% to 35% of the `AISegment.cn` dataset as the training set. The batch size made little impact on the training characteristics, so a batch size of 8 images was used to prevent the training systems from running out of memory. This batch size kept the peak GPU memory usage under 13GB.



Fig. 4. Randomly selected colorized images from the intermediate model architecture using transposed-convolutions in the generator's encoder.

Inference images from the best-achieved model with this setup are shown in Fig. 4. While the generator produces quite realistic colorizations, there are extreme red artifacts prevalent on many of the output images. This is extremely distracting, and it indicates that something is not quite right in the training setup. The colorization results seem to plateau after around 100 epochs: training until 200 epochs does not reduce the red artifacts.

In an attempt to correct these artifacts, the transposed-convolution operation in the generator's decoder section was swapped for Bilinear upsampling. A common problem with the transposed-convolution operation is that it can produce visually distracting artifacts and checkerboard patterns [13]. These checkerboard artifacts can actually be seen in some of the most prominent red patch artifacts, as seen in Fig. 5 which was taken from Epoch 95 of the model's training process.



Fig. 5. A colorized image from Epoch 95 of model training, displaying the significant checkerboard artifacts produced by the transposed-convolution upscaling method.

After making the swap to Bilinear upsampling, the checkerboard pattern did indeed disappear. The new inference images seem more convincing, but the patchy red artifacts still remain in the inference images. Fig. 6 shows how the new upsampling



Fig. 6. Randomly selected colorized images from the intermediate model architecture using Bilinear-upsampling in the generator's encoder.

method yields better images, and some images have very sparse artifacts. While the model is correctly learning the colorization task, and doing quite well in the process, there appears to be some more fundamental issue. Looking into the loss plot, it becomes evident that the WGAN training process is completely unstable for this model architecture, despite the fact that it yields somewhat reasonable results in the end.

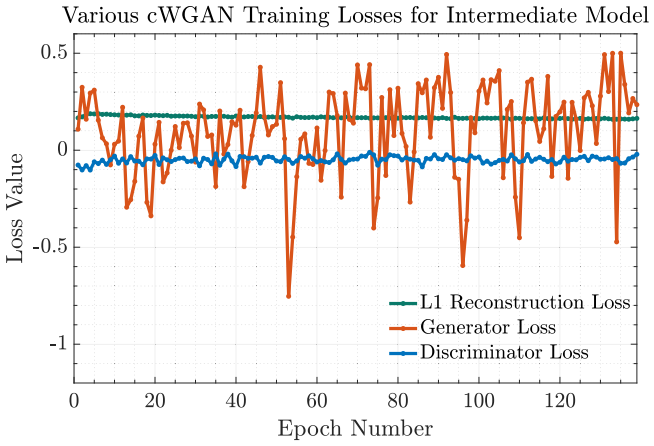


Fig. 7. Average generator, discriminator, and L1 training losses per epoch for the intermediate architecture.

Fig. 7 displays the average losses per epoch and confirms that the WGAN losses are completely unstable with this model architecture. Note that the generator loss plotted is only the WGAN contribution to the generator loss. The sign of the WGAN generator loss continuously flips between positive and negative. The discriminator loss remains steady over the entire training duration, indicating that the discriminator fails to successfully learn, causing the WGAN generator loss to act chaotically since it relies on the discriminator outputs. The L1 contribution actually seems to improve over time, which could explain the somewhat acceptable colorizations that the model began to produce. From this point, more significant updates were made to the architecture in order to resolve these issues and achieve successful training with the WGAN losses.

B. Final Framework Training

Once the discriminator was identified as the issue, instance normalization was added after each convolution operation except for in the first layer. This was implemented in order to standardize the data and try to minimize any unwanted behavior. Since grayscale images in each mini-batch could also contain quite significant variations, batch normalization was swapped for instance normalization in the generator. The model was also swapped back to using transposed-convolutions with no negative side-effects and slightly added capacity. With these changes, the architecture was now exactly as described in Section II.

Further hyperparameter tuning was executed, although the optimal hyperparameters remained the same as previously. Thus, for training the final version of the model, the hyperparameters were as follows: $LR = 5 \times 10^{-5}, n_{discrim} = 5, \lambda = 10$ with a batch size of 8 images, a training length of 155 Epochs, and a 25% training set size in relation to the full dataset. Thus, the model was trained with 8,607 images from the `AISegment.cn` dataset. This model trained for 20 total hours on Princeton's Adroit cluster with 6 CPU-cores, 24GB memory, and an NVIDIA A100 GPU. The generator has $9,587,714$ total parameters and the discriminator has $1,616,769$ total parameters. Thus, the architecture has a total trainable parameter count of $11,204,483$.

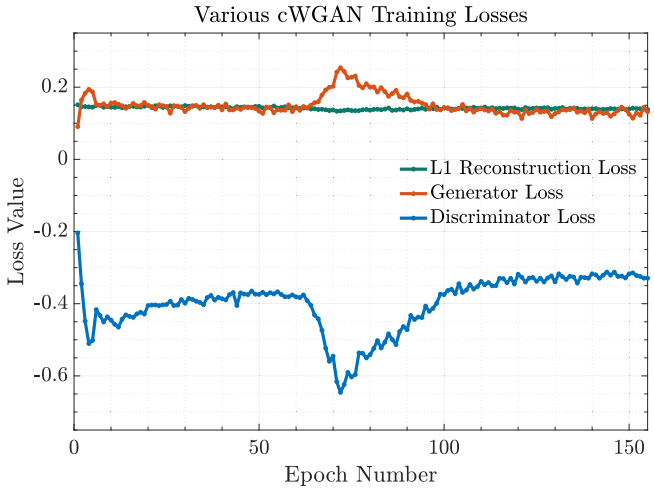


Fig. 8. Average generator, discriminator, and L1 training losses per epoch for the final architecture.

These model changes yield a healthy loss plot, as shown in Fig. 8. Note that the WGAN generator and discriminator losses have opposite signs, but this is to be expected due to the opposing objectives; both optimize towards zero. The L1 loss stays relatively constant across the plot, slightly decreasing from epochs 50 to 100 and then marginally increasing at the end. However, despite the seemingly constant and potentially unimportant L1 loss, it was still a critical training constraint that allowed for realistic image colorizations. Without this term and only using the WGAN losses, the model colorizations were unrealistic and training was stopped early. Both the gen-

erator and discriminator loss curves see healthy optimization and steady improvements up until approximately epoch 70. Afterwards, there is a slight divergence and jump in losses. This unexpected behavior acts as an effective restart in the training process, similar to the stochastic gradient descent with warm restarts methodology as proposed in [14]. However, both losses recover from this jump and steadily decrease until the end of the training process. By 150 epochs, the loss has essentially converged and produced optimal results for this model architecture and training setup. Several images from the `AISegment.cn` dataset displayed in grayscale and the corresponding generator colorized images are shown in Fig. 9. Further evaluation and analysis of model results will be performed in the following sections.



Fig. 9. Examples of colorizing images from the `AISegment.cn` dataset. (Top): Grayscale images fed to the generator. (Bottom): Colorized images produced by the generator.

V. Framework Evaluation

Qualitative and quantitative framework evaluation is performed over three datasets, analyzing patterns and errors in colorization:

- The `AISegment.cn` dataset on which the model was trained [15].
- The FairFace dataset which contains portrait images categorized by racial demographics [9].
- The SF-HQ dataset containing high resolution synthetically generated human portrait images [16].

Since the architecture supports input images of arbitrary resolutions that are divisible by 8, the image resolutions utilized will also be mentioned. For quantitative analysis, colorized images from each of the three datasets were converted from the *Lab* color space to RGB before computing reconstruction error. The metrics utilized included Mean-Absolute error (MAE) for each of the three RGB channels as well as Mean L_2 RGB reconstruction error.

Mean pixel L_2 RGB reconstruction error:

$$\frac{1}{N}\sum_{i=1}^{N}\sqrt{(R_{i,g}-R_{i,r})^2+(G_{i,g}-G_{i,r})^2+(B_{i,g}-B_{i,r})^2} \tag{6}$$

Mean pixel absolute error (for any channel $C \in (R, G, B)$):

$$\frac{1}{N}\sum_{i=1}^{N}|C_{i,g}-C_{i,r}| \tag{7}$$

where R, G, and B correspond to pixel values in each of their respective RGB values. The symbols g and r denote generated and real pixel values, while N refers to the total number of pixels in any number of photographs.

Quantitative analysis on individual images is also performed by creating normalized heat-maps for visualizing localized L_2 RGB reconstruction error.

A. *Inference on the `AISegment.cn` Dataset*

A held out set of 300 randomly selected images from the `AISegment.cn` dataset is utilized for inference and initial model evaluation. Unlike the training dataset images, these images are not resized to 288×384 pixels, for sake of maintaining image quality while performing analysis.

As discussed earlier, the `AISegment.cn` dataset mainly consists of front-facing human portraits. Although the dataset lacks image metadata, these front-facing portraits can roughly be classified into two different classes, namely professional portrait photographs (e.g. red-carpet celebrity photos) and amateur portrait photographs (e.g. images taken in candid settings).

Key differences between these two classes of photographs correspond to backgrounds and lighting, which likely play a significant role in the way images from each class are colorized. More specifically, professional photographs are generally well-exposed and illuminate a person in soft light. In contrast, amateur portrait photographs often have uneven lighting, resulting in harsh shadows and poor image exposure. In terms of backgrounds, professional photographs are highly likely to have monotone, neutral colored backgrounds associated with prepared photo shoots, while amateur portrait photographs are often taken in a variety of diverse settings that have cluttered or disorganized background structures, potentially being outdoors.

Fig. 10 juxtaposes four ground-truth professional photographs with their colorized counterparts and presents a heat map that depicts the locations of highest L_2 RGB reconstruction error (depicted in intense orange). Direct comparison of colorized and ground-truth images utilizing these heat-maps demonstrates highly accurate colorization. Image regions associated with skin, dark clothes, and neutral backgrounds are relatively well colorized, demonstrating little RGB reconstruction error. Areas of high colorization error within these images are limited. More specifically, areas of poor colorization are limited to patches of vibrant, non-neutral colors and areas



Fig. 10. Colorization of well-exposed professional photographs from the held-out evaluation subset of the `AISegment.cn` dataset. (Top): Colorized images produced by the generator. (Middle): Original color images from the dataset. (Bottom): Heat maps of normalized L_2 RGB reconstruction error, where more orange values indicate larger margins of error. Images from left to right are labeled (a, b, c, d) respectively.



Fig. 11. Colorization of under-exposed amateur photographs from the held-out evaluation subset of the `AISegment.cn` dataset. (Top): Colorized images produced by generator. (Middle): Original color images from the dataset. (Bottom): Heat maps of normalized L_2 RGB reconstruction error, where more orange values indicate larger margins of error. Images from left to right are labeled (a, b, c, d) respectively.

where rare colors exist (relative to the given photograph). Fig. 10a demonstrates this, showing the colorization of red lipstick as pale gray with patches of poorly-colored red. Across almost all evaluated images, vibrant colors such as reds, greens and blues are poorly colorized, likely due to the relatively neutral color palette of the `AISegment.cn` dataset.

A key characteristic of these images also corresponds to low color variance within the ground-truth images. Due to relatively consistent image exposure and monotone backgrounds, most regions of the image can visually be segmented into distinct colors that have very little variance in shade or intensity. This is further reflected in the colorized images, where both correct and incorrect regions of the images are also colored consistently, leading to regions that have distinct and consistent levels of low or high RGB reconstruction error. This can be seen in Fig. 10c where there is a slight skin-tone patch of color above each of the shoulders in the colorized image.

In contrast, images characterized by cluttered backgrounds and harsh, uneven lighting lack this relatively consistent colorization. Fig. 11 provides examples of these kinds of images. The colorization heat-maps within this figure demonstrate large, patchy regions of high and low reconstruction error, which is often localized in and around the human body, clothing, and face. This observation suggests that the current system struggles when an images contains high color variance, caused by shadows, cluttered backgrounds, or uneven lighting. In contrast, when an image is well lit and a person is photographed with a clean background, as demonstrated in Fig. 10, the model appears to perform significantly better at creating more accurate colorizations.

Visual comparison of each generated image in Fig. 11 also demonstrates that the prominent color palette of each image is almost identical. More specifically, each image is colorized with a pallette of neutral browns. The presence of this phenomenon, which is consistently reflected among other images that are either poorly exposed or contain cluttered backgrounds, can possibly be explained by the incorporation of L_1 loss during generator training, constraining the range of color tones generated and limiting the vibrancy.

During training, L_1 error is computed for each individual pixel across its a and b channels in the *Lab* color space. Afterwards, the L_1 reconstruction error for the whole image is obtained by averaging over each pixel's corresponding L_1 error. In turn, it appears that when there exists high amounts of color variance in a given image, the model attempts to average out the large variance in color, resulting in pervasive neutral tones that dominate large image regions. A larger set of inference images from the `AISegment.cn` dataset is presented in Appendix A.

B. Inference on Background-Removed Images

Recall that the `AISegment.cn` dataset also contains images with their backgrounds removed for all images in the dataset. A range of images with removed backgrounds is also passed through the model for colorization. As a point



Fig. 12. Colorization of background-removed and original images from the held-out evaluation subset of the `AISegment.cn` dataset. (Top): Colorized background-removed images produced by generator. (Middle): Colorized original images produced by the generator. (Bottom): Original images from the dataset. Images from left to right are labeled (a, b, c, d, e, f, g, h, i) respectively.

of comparison, the unmatted images are also colorized for comparison in Fig. 12, to test if the backgrounds impact the colorization of the faces and clothing.

Overall, the colorization results are similar with and without the image backgrounds. However, there are some subtle nuances. In Fig. 12a and Fig. 12g, the images have a consistent backdrop and are professionally shot. As such, the colorizations with and without the background are essentially the same. In Fig. 12c, the man's colorization without the background is quite realistic and the skin-tone appears consistently colorized. However, the colorized image with the background is significantly worse. While the shirt is colorized as a different color, the skin lacks color and appears extremely gray. The same phenomenon can be seen in Fig. 12h where the man's face has significantly more color in the background-removed version, and the face lacks color when the background is present. Similarly, in Fig. 12d, the neck region has a gray spot in the colorized version with the background, whereas the skin is completely filled in with the background removed. Thus, on balance, the generator produces stronger colorizations when no distracting backgrounds are present. This is an interesting point for future study and improvement, and this supports the conclusion that cluttered backgrounds negatively impact the results and produce duller photos. Only in one case, in Fig. 12i, does the colorized image with the background appear as more realistic than the background-removed counterpart. This occurs since the version without the background has a slight brown cast over the entire region. It is possible that by adding further capacity to the model, the architecture would be able to better colorize and understand background regions, thereby producing more realistic colorizations and not detracting from the face regions.



Fig. 13. Randomly selected images from the FairFace dataset.

C. Inference on the FairFace Dataset

The FairFace dataset is a specialized dataset consisting of roughly 108,000 human portrait images categorized by racial demographic, age, and gender [9]. The images are retrieved from various sources, including Flickr, Twitter, Google Images, and online newspapers. The demographic annotations are constructed by annotators from the Amazon Mechanical Turk platform. Its images are relatively low quality when compared with the `AISegment.cn` dataset. Notably, this



Fig. 14. Example of a FairFace image that has been transformed, scaled, and rotated to center the face in a 448×448 image, performed by [17].

dataset contains images "In-the-Wild" which have variable lighting, focus, positioning, and potential occlusions. Example images from this dataset are shown in Fig. 13. It is also common for images in this dataset to contain multiple people close together in view.

This makes the dataset much more varied than the highly standardized `AISegment.cn` portrait dataset, which thus makes it a good test in the generalization capabilities of the colorization network. Images from this dataset are retrieved from Kaggle [17], where images have been standardized to 448×448. Furthermore, this dataset has already been sorted by racial demographic and images were previously transformed, rotated, and cropped such that faces are in center view at a consistent scale. This sometimes produces images that look like Fig. 14 where the original image does not span the entire 448×448 image. However, this is not an issue for the colorization task, and the colorization model simply ignores these black background regions.

The racial demographics present in the dataset are categorized into the following groups in a highly generalized manner: White, Black, East Asian, Southeast Asian, Middle Eastern, Latino, and Indian.

Across racial groups, the model's performance seems to be relatively consistent. This is shown in Fig. 15, which displays colorization examples corresponding to the seven racial demographics in the FairFace dataset. Visual examination of the heat maps in the bottom row demonstrate consistent, high levels of skin colorization accuracy among all seven different demographics. Notice that the heatmaps often show high error in clothing regions: this occurs since the model colorizes clothing in neutral tones, suppressing the originally vibrant blue, orange, yellow, and green clothing.

Table I presents reconstruction error metrics for each of the different racial demographics discussed prior, including Mean-Absolute Error across each of the RGB channels and overall L_2 RGB Error (computed over all image pixels for each demographic), which further supports this argument.

Demographic	Red MAE	Blue MAE	Green MAE	L_2 RGB Error
Black	14.4832	5.1362	16.9982	25.1666
East Asian	14.1309	5.4172	15.7837	24.0005
Latino	13.4782	5.3508	15.5070	23.2841
Indian	14.4709	5.4780	16.1638	24.4988
Middle Eastern	12.9918	4.5700	15.5266	22.4669
Southeast Asian	14.6744	5.5838	17.0340	25.2815
White	13.4165	5.0175	15.9579	23.4175

TABLE I
MEAN PIXEL ABSOLUTE ERROR AND MEAN PIXEL L_2 RGB RECONSTRUCTION ERROR FOR DIFFERENT RACIAL DEMOGRAPHICS IN THE FAIRFACE DATASET [17]

Looking at these error metrics, reconstruction error among different demographics is relatively consistent, both for chan-

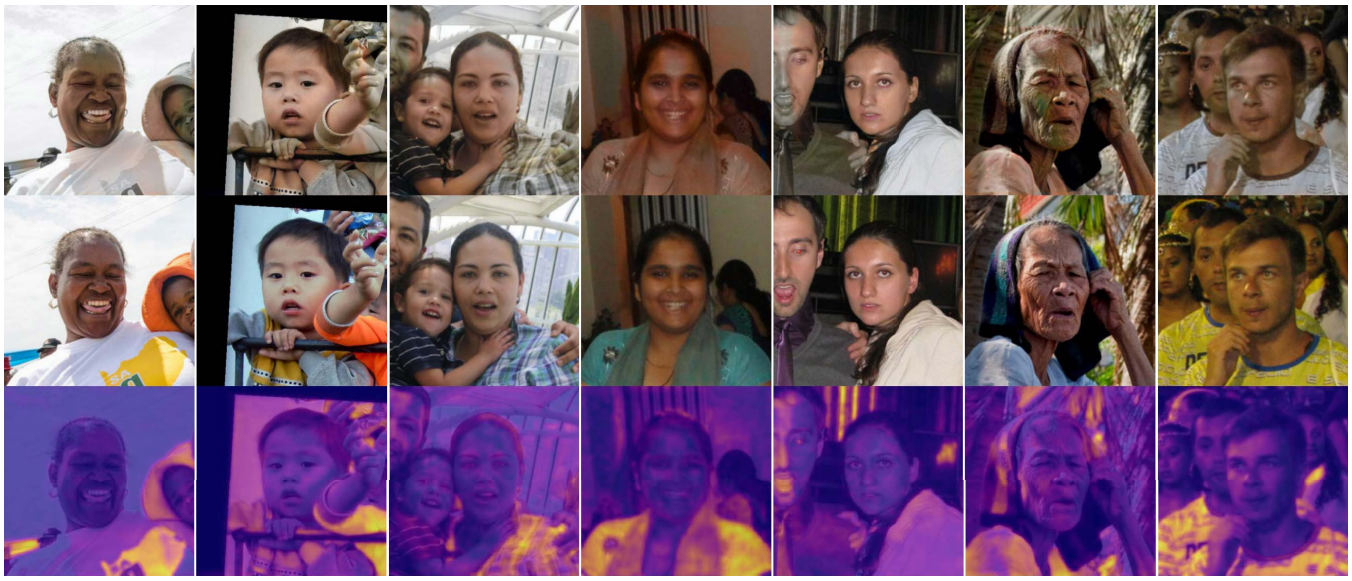


Fig. 15. Colorization of images from different racial demographic classes in the FairFace dataset (From left to right: (a) Black, (b) East Asian, (c) Latino, (d) Indian, (e) Middle Eastern, (f) Southeast Asian, (g) White). (Top:) Colorized racial demographic images (Middle:) Ground-truth racial demographic images from the FairFace dataset (Bottom:) Heat maps of normalized L_2 RGB reconstruction error, where more orange values indicate larger margins of error.

nel specific MAE and overall mean pixel L_2 RGB reconstruction error. Among all demographics, Red and Green channels were the most error prone, while Blue error was comparatively low.

This phenomenon likely arises due to the system's usage of the *Lab* color space. Recall that this system's generator only reconstructs the a and b channels in the *Lab* color space, with the L channel from the original input image being unchanged. In turn, when converted to the RGB color space, it is likely that the generated image's blue channel values are mostly derived from the original gray-scale image, which is preserved throughout the generation process.

This preservation of information is likely also a driver of the model's relatively consistent colorization across different racial demographics. Grayscale images directly encode information about how dark a given region within an image is. Thus, as most skin-tones are relatively neutral colors that range from light to dark, the model only has to learn how to generate a relatively small range of colors in order to effectively colorize most human portraits. When these colors are overlaid onto preserved gray-scale information, most skin colors are likely easily replicated in a consistent manner.

It is also interesting to note that although the `AISegment.cn` training dataset is heavily biased towards White and East Asian individuals, their reconstruction error metrics are not the lowest. Based on analysis from the prior section, this could possibly be attributed to the model's neutral, brown pallette bias when colorizing images with large color variance. Most images within the FairFace dataset are taken by amateur photographers in an uncontrolled setting. In turn, many images within this dataset, irrespective of demographic, can be subject to high color variance due to both lighting and background structure, making them susceptible to colorization with a neutral, brown pallette.

This is evidenced through visual inspection, and can be observed in the sample images from Fig. 15. Fig. 15g, Fig. 15f, Fig. 15d, and Fig. 15c demonstrate this, with relatively accurate colorizations performed with a neutral, brown pallette. In turn, the slightly lower RGB reconstruction error values for the Latino and Middle Eastern demographics can possibly be attributed to pervasive colorization using a neutral color pallette, which may mirror the natural skin tones of these demographics with greater accuracy.

D. Inference on the SF-HQ Dataset

The SF-HQ dataset consists of 425,000 synthetically generated 1024×1024 facial images [16]. Specifically, synthetic images from part 2 of this dataset are chosen for inference.

The results of colorizing 5 random images from this dataset at full resolution are displayed in Fig. 16. The colorized images are a significant step up from the grayscale versions, but they are still quite far off from the ground truths. While the colors are mostly realistic, the colorized images lack vibrancy and are very desaturated, leading to an unnatural and strange quality. Several individuals appear washed-out, despite the original images having reasonable lighting. All 5 generated images appear to have an almost identical color palette, despite the variations in skin-tones in the original images. This causes an overall strange appearance, and these colorization are definitely less realistic than colorization of real images. Based on previous results, one would expect these colorizations to be most realistic due to the high-quality of the images and the clarity of facial information.



Fig. 16. Examples of colorizing synthetic portrait images. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.

As a further experiment, these same images are downsized to 384×384 and then colorized by the generator to determine if its the larger image dimensions and potentially over-sized features that the network is struggling to colorize. These results are shown in Fig. 17.



Fig. 17. Examples of colorizing downscaled synthetic portrait images corresponding to the images in Fig. 16.

The newly colorized images are more realistic than the previous colorizations, while the color palette still appears to be quite limited. The differences are not extreme, however, there is more tonal variation and the faces do not get washed out as previously. More specifically, it appears that the faces have more prominent red tones.

This demonstrates that although the network may successfully colorize images at any resolution, it still seems to work optimally near the resolution it was trained at, around 288×384. This is logical, since the convolutional kernels of the generator likely work best at a certain scale. For further reference, more synthetically colorized images at the downscaled resolution are shown in Fig. 18

E. Quantitative Comparison of Different Datasets

Mean pixel RGB reconstruction error was computed on evaluation subsets of each of the dataset hitherto described, as listed in Table II. Overall, the dataset with the lowest reconstruction error corresponded to the Synthetic (SF-HQ) dataset.

Dataset	Red MAE	Blue MAE	Green MAE	L_2 RGB Error
FairFace (384x384)	13.9494	5.2219	15.9816	24.0166
SF-HQ Synthetic (384x384)	10.7339	3.4714	12.9593	18.4927
SF-HQ Synthetic (1024x1024)	12.9311	4.1492	13.7428	20.6981
AISegment (600x800)	14.7615	5.3746	17.5922	25.6685
AISegment (288x384)	13.8226	5.2033	17.0633	24.5728
Matted AISegment (600x800)	14.5163	5.6009	17.1618	25.3673
Matted AISegment (288x384)	13.1800	5.2386	15.9608	23.4633

TABLE II

MEAN PIXEL ABSOLUTE ERRORS AND MEAN PIXEL L_2 RGB RECONSTRUCTION ERROR FOR DIFFERENT DATASETS AT DIFFERENT RESOLUTIONS. ROUGHLY 2100 EVALUATION IMAGES WERE UTILIZED FOR THE FAIRFACE [17] DATASET (TO EFFECTIVELY CAPTURE EACH RACIAL DEMOGRAPHIC) AND 300 IMAGES ARE UTILIZED FOR ALL OTHER DATASETS. THE MATTED AISEGMENT [15] DATASET REFERS TO PORTRAIT IMAGES WITH BACKGROUNDS REMOVED.

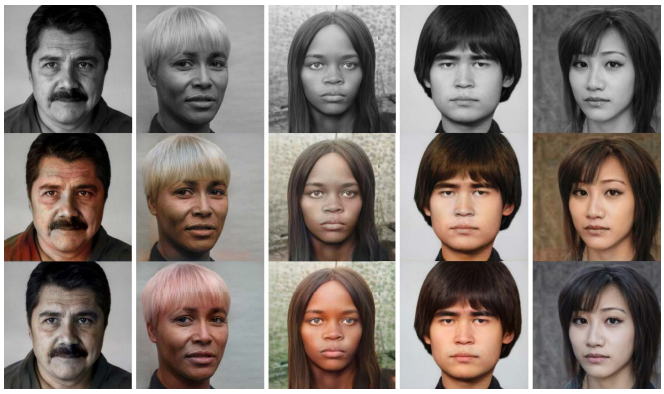


Fig. 18. Examples of colorizing downscaled synthetic portrait images. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.

This is likely due to the presence of pervasive neutral colors within generated images and smaller, cleaner background areas that are relatively easy to colorize.

The dataset with the second-best reconstruction error corresponded to the FairFace dataset, followed by the background-removed (matted) AISegment dataset. While somewhat counterintuitive, this result also makes logical sense. Analysis from previous sections shows that the model struggles with large cluttered backgrounds. In the FairFace dataset, most images are cropped such that human faces fill up most of the frame. As a result, colorful clothing and cluttered backgrounds, image regions that inhibit colorization accuracy, are comparatively rare in FairFace, providing an explanation for this result.

The system was also evaluated on the segmented (matted) portrait images that are contained within the AISegment dataset. While calculating reconstruction errors on these matted images, only non-black pixels (i.e pixels where at least one RGB channel is non-zero) were included, to avoid incorporating matting into the reconstruction metrics. This experiment showed a slight but noticeable decrease in reconstruction error, further suggesting that simpler backgrounds have an impact on reducing reconstruction error in the context of this system.

Lastly, evaluation at different resolutions showed that reconstruction error was minimized when the system was run with input images that were closer to the image resolution utilized for training, 288×384. This fundamentally makes sense, as model parameters were optimized during training to deal with human portraits and features represented at lower resolutions.

Overall, RGB reconstruction error metrics across the different datasets do not significantly differ, demonstrating the generator's consistency and capability to generalize well.

F. Takeaways from System Evaluation

One may expect these results, since this is a generative modeling task. Regardless of how good the colorization model is, it will never be able to perfectly match the original colors for a wide spectrum of images. While models may learn quite well how to colorize skin tones, and what constitutes a realistic image, they will never be able to identically colorize all aspects of grayscale images compared to the ground truths. For example, in the examples presented, the generator often colorized clothing with neutral tones (grays, browns, and black), which is logical considering the L1 loss constraint. Still, a perfect colorization model will not know if an article of clothing should be blue, or perhaps bright green. Thus, the model learns a "safe" color assumption which may simply be the average of all plausible clothing colors. A similar argument can be made about hair color. While individuals in the inference data occasionally had dyed hair, such as in Fig. 18, the model colorizes hair with more plausible and natural tones. It would be illogical for a colorization model to frequently generate brightly dyed hair, since the probability of this in the real world is indeed quite low.

VI. CONCLUSIONS

This work successfully implements and analyzes an architecture and training setup for colorizing grayscale images. A residual U-Net structure is implemented for the generator and a convolutional network is implemented for the discriminator. This setup is trained in a traditional adversarial fashion, using the WGAN objective to improve training characteristics.

The generator is able to produce realistic colorizations, the best of which could certainly be mistaken as real images. Despite the biases present in the training data, the model achieves consistent colorizations across racial demographics which may be a concern for the colorization task. This likely stems from the fact that the grayscale L channel is always

used as a base for output images. The model achieves most convincing results on well-exposed professional photographs, while the colorizations for poorly exposed candid photographs sometimes struggle. The model excels at colorizing people, their faces, hair, and clothing. Colorizations produced by this model sometimes lack vibrancy and can be quite dull, typically coming from a limited palette of neutral tones. This could potentially be ameliorated by reducing the weight of the L1 loss constraint, or individually weighting the losses for each color channel during training. Still, the methodology is very promising. With further study, additional model capacity, and increased diversity of training data, improved results could almost certainly be attained.

ACKNOWLEDGEMENTS

The authors are pleased to acknowledge that the work reported on in this paper was substantially performed using the Princeton Research Computing resources at Princeton University which is a consortium of groups led by the Princeton Institute for Computational Science and Engineering (PICSciE) and Research Computing. The authors also thank Professors Olga Russakovsky and Vikram Ramaswamy for their guidance and support.

REFERENCES

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018. [Online]. Available: https://arxiv.org/abs/1611.07004

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[3] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs," 2017. [Online]. Available: https://arxiv.org/abs/1704.00028

[5] K. Leung, "Neural network architecture diagrams," 2024. [Online]. Available: https://github.com/kennethleungty/Neural-Network-Architecture-Diagrams/

[6] S. Benhamadi, A. Hossary, and Y. Abdennadher, "Pix2pix image colorization with conditional WGAN," Kaggle, Notebook, 2022. [Online]. Available: https://www.kaggle.com/code/salimhammadi07/pix2pix-image-colorization-with-conditional-wgan/notebook

[7] V. Nagpal, "pix2pix is all you need," Kaggle, Notebook, 2022. [Online]. Available: https://www.kaggle.com/code/varunnagpalspyz/pix2pix-is-all-you-need/notebook

[8] A. Guilbert, "Black and white image colorization with deep learning," 2021. [Online]. Available: https://anne-guilbert.medium.com/black-and-white-image-colorization-with-deep-learning-53855922cda6/

[9] K. Karkkainen and J. Joo, "FairFace: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1548–1558.

[10] Z. Xiong, Z. Wang, C. Du, R. Zhu, J. Xiao, and T. Lu, "An Asian face dataset and how race influences face recognition," in *Advances in Multimedia Information Processing – PCM 2018*, R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, and C.-W. Ngo, Eds. Cham: Springer International Publishing, 2018, pp. 372–383.

[11] N. Hung, "Improved WGAN in pytorch," 2023. [Online]. Available: https://github.com/jalola/improved-wgan-pytorch/

[12] E. Linder-Norén, "Pytorch-gan," 2020. [Online]. Available: https://github.com/eriklindernoren/PyTorch-GAN/

[13] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[14] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2017. [Online]. Available: https://arxiv.org/abs/1608.03983

[15] Beijing Play Star Convergence Technology Co., Ltd, "AISegment.com Matting Human Datasets," Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/laurentmih/aisegmentcom-matting-human-datasets

[16] D. Beniaguev, "Synthetic faces high quality (sfhq) dataset," 2022. [Online]. Available: https://github.com/SelfishGene/SFHQ-dataset

[17] Beijing Play Star Convergence Technology Co., Ltd, "FairFace Race Dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/abdulwasay551/fairface-race

APPENDIX A
ADDITIONAL IMAGES



Fig. 19. Examples of colorizing images from the held-out evaluation subset of the `AISegment.cn` dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 20. Examples of colorizing images from the held-out evaluation subset of the `AISegment.cn` dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 21. Examples of colorizing images with the 'Black' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 22. Examples of colorizing images with the 'White' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 23. Examples of colorizing images with the 'East Asian' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 24. Examples of colorizing images with the 'Southeast Asian' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 25. Examples of colorizing images with the 'Middle Eastern' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 26. Examples of colorizing images with the 'Latino' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.



Fig. 27. Examples of colorizing images with the 'Indian' race category from the FairFace dataset. (Top): Grayscale images fed to the generator. (Middle): Colorized images produced by the generator. (Bottom): Original color images from the dataset.