The Wizard of GANs: From Grayscale to Somewhere Over the Rainbow

EENG 439 Final Report, Fall 2022

Annie Lin Adam Zapatka David Peng Maggy Lambo

Fig. 1: Judy Garland in *The Wizard of Oz (1939)*. The left is the original, the right is our colorized frame.

***Abstract*—Colorization is the process of generating color images from black-and-white images or frames. In machine learning applications, this involves predicting the colors to assign to portions of black and white images, often to restore or modernize them. A popular architecture for image colorization is the conditional generative adversarial network (cGAN), consisting of a generative model that learns to produce a colored image, and a discriminative model to classify the generated image as real or fake. Both models take the original grayscale image as a "condition", or additional input. Pairing the two models as adversaries removes the need to design one's own loss function. Previous work has implemented the generative model with a U-Net architecture that downsamples the grayscale image into a vector and then upsamples it as a colored image. Another architecture, deep convolutional GAN (DCGAN), uses convolutional neural networks (CNN) for the generator and is designed to optimize for stable training. We propose that a modified DCGAN-based image colorization model will outperform a U-Net-based model given the following modifications: the discriminator of the DCGAN model is weakened and changed to a patch discriminator to allow it to improve at a similar rate to the generator, and the input data is normalized with batch normalization. By training and testing our DCGAN model with these modifications, we observed that the loss graphs were more stable, and the model's qualitative results outperformed those of the U-Net-based model.**

I. Introduction

Despite the massive impact of photography and motion pictures on the way we perceive the world, much of this media from decades past is stuck in black-and-white, limited by the technology of its time. For almost half a century, people have been working to bring these photos and videos to a modern light with colorization, a painstaking process involving the approximation and manipulation of colors within an image [1]. Even with modern digital editing tools, this process could be tedious and time-consuming. Recent algorithms and machine learning methods seek to automate the task.

One such machine-learning architecture is a generative adversarial network (GAN), which pits two models against each other to create output similar to data from the desired distribution [2]. The generative model learns to produce a colored image, and the discriminative model learns to classify the generated image as real or fake. Pairing the two models as adversaries removes the need to design one's own loss function.

Conditional generative adversarial networks (cGANs) allow both models to be conditioned on additional input [3]. The purpose of the cGANs model is thus to modify the conditioned input data, instead of generating a completely independent output from random noise. For the colorization task, cGAN makes a natural fit since the generative model now has access to the original greyscale image. Prior works using cGANs to colorize images [3], [4] implement their generator with a U-Net [5], an encoder-decoder-based architecture with skip connections. In this context, the U-Net downsamples the grayscale image into a vector and then upsamples into a colored image.

A different modification to GANs, Deep Convolutional GANs (DCGANs) [6], has also been applied to colorization. This network was originally designed as a fully convolutional version of GANs, but is modified in [7] to take in an image as conditional input. Similar to the cGAN used in [3], this conditional input is a black and white image that the model colorizes.

The goal of the rest of our paper is to demonstrate that this

style of DCGANs, along with a few modifications to its discriminator and input data, can outperform the aforementioned cGANs with a U-Net generator.

II. Method

We run four experiments. For each experiment, we train our own model on a dataset of images. The black-and-white version of these images are input into the model, and their colored counterparts are used to compare to the output in the discriminator. We then observe the loss, and evaluate the results on a validation dataset and sample black-and-white movies. Figure 1 shows a frame from our demo video of a colorized *The Wizard of Oz (1939)*. The experiments train and evaluate the following models:

1) U-Net with Pre-Trained Downsampling
2) DCGAN with Patch Discriminator
3) DCGAN with Weakened Discriminator
4) DCGAN with Batch Norm and Training Offset

A. Dataset

We train our colorization models on Common Objects in Context (COCO) [8], which is the same dataset used in [4]. COCO contains over 330,000 images and is commonly used for object classification and other image tasks. COCO is an effective dataset for training our model to recolor black-and-white films due to the breadth of objects represented and the frequency at which people appear in its images. In our training, we only used 8,000 randomly chosen images as the training dataset and 2,000 as the validation dataset. Images were resized to 256x256 to input into the model. Between epochs, data augmentation tools are used on images (such as mirroring images) to reduce the overfitting of the model.

B. Output Channels

There are different ways to separate color into channels. One popular method is RGB, which determines the weight of red, blue, and green on a pixel on a scale of 0 to 255. The method we use in our experiments has *L*a*b* channels, where *L* is lightness, *a* defines green and red colors, and *b* defines blue and yellow colors. This color breakdown makes more sense for the task at hand; since the lightness channel is already available through the black-and-white image, this reduces the task of training an additional channel. [4]. This also limits the number of predictions that the model has to make and reduces the number of computations.

C. Image to Video Conversion

Since the model was trained on COCO data to convert individual black-and-white images to color, colorized videos had to be produced on a frame-by-frame basis. As such, we made a custom Python script to turn the black-and-white videos we wanted to colorize into a folder of image frames. After model evaluation, we input the colorized image frames into a second Python script to concatenate them back into one video. To synchronize the videos for a side-by-side comparison, we used iMovie and adjusted the starting time of the two clips.

D. Loss Functions

We run a min-max game between the discriminator and the generator with the following cGAN loss function [3]:

$$L_{cGAN}(G,D) = E_{x,y}[\log D(x,y)] + E_{x,z}[\log(1 - D(x,G(x,z)))]$$

The goal of the discriminator is to maximize the GAN loss, whereas the goal of the generator is to minimize the GAN loss. This loss function calculates the gradient ascent to apply to the discriminator and the gradient descent to apply to the generator. We also use L1 loss [3], or mean absolute error, which attempts to introduce supervised learning to the colorization problem by taking the difference between the generated color and the actual colors:

$$L_{L1}(G) = E_{x,y,z}[||y - G(x,z)||_1]$$

This is similar to the mean squared error (MSE) of a classification problem. The existence of a L1 loss is possible because unlike other generative problems, colorization has a "correct" colorization during training, which can be utilized as part of the loss function. The L1 loss combined with the GANs loss allows the model to generate realistic looking images. Thus, the combined loss function is as follows [3]:

$$G^* = \arg\min_G \max_D [L_{cGAN}(G,D) + \lambda L_{L1}(G)]$$

Where λ determines the contribution from the two loss functions.

E. Network Architectures

1) U-Net: U-Nets [5] consist of contracting layers followed by expansive layers. Together, these two downsampling and upsampling sections form the shape of a symmetric "U", as in Figure 2.

The encoder portion of the U-Net is a series of convolutional layers using 3x3 filters. Each layer doubles the feature size and is followed by rectified linear unit (ReLU) activation and a 2x2 max pooling operation. The output of each layer also has a skip connection to its counterpart in the decoder.

The decoder portion of the model uses transposed convolutional layers with 3x3 filters to decode input features, halving the number of feature channels with each layer. Each upsampling step also consists of a ReLU activation, batch normalization, and dropout.

Building on the U-Net model, we initialize the U-Net model with a ResNet18 [9] model that we pre-train on image colorization as the downsampling component, keeping all other parts of the U-Net model the same. This is to avoid the possible problem in GANs of "blind leading the blind." [4]

2) DCGAN: DCGANs [6] are an implementation of GANs using CNNs and transposed CNNs. Its encoder-decoder structure exists to downsize and then upsize the image, as in Figure 2.

The encoder portion of the DCGAN consists of a convolutional layer for each output, each of which uses a 4x4 filter with stride 2. The number of feature channels is doubled with

each convolutional layer, and the model uses no max pooling layers as to preserve the details of the images. Each strided convolution is followed by batch normalization and Leaky ReLU.

In the decoding portion of the model, transposed convolutional layers with filter size of 4x4 and stride of 2 are used to decode the feature channels. Each transposed convolution is followed by batch normalization and ReLU, except for the last layer which uses Tanh activation.

The generator we use is very similar to the DCGANs from the paper by Nazeri et al. [7] with several important modifications. First, we eliminate the dropout layers as done in [4]. By eliminating the dropout layers, we can save computing power and time during training. Second, we initially experiment with no batch normalization layers, but later add them to yield faster training time. Since our model is larger, we explore different ways of better integrating the generator with the discriminator, so one does not overpower the other.

III. Experimental Results

For each of the four architectures, we trained our own model, tracking the loss and qualitiative performance over at least 20 epochs.

We display the qualitative performances on the COCO dataset in Figure 3, and we show the loss for each model in Figure 4.

A. Experiment 1: U-Net with Pre-Trained Downsampling

We wanted to recreate results from an implementation with U-Nets and a pre-trained downsampling component. We first pre-trained the ResNet18 model [9], optimizing for L1 loss, on 20 epochs. Then, we initialized the U-Net model with the ResNet18 model as the downsampling component, keeping all other parts of the U-Net model the same, and trained with L1 loss and GAN loss for 20 epochs. The generator's loss decreased over epochs while the discriminator's loss increased.

B. Experiment 2: DCGAN with Patch Discriminator

In our experiment, we changed the generator model to the DCGAN used in the paper by Nazeri et al. [7] instead of the U-Net generator, keeping all other parts of the architecture–parameters, hyperparameters, etc. the same. We kept the same patch discriminator used by the U-Net model and trained it for 100 epochs. The results were used to generate colored images for the chosen black-and-white film. From observing the loss function of both the discriminator and the generator the generator's loss fell only at the beginning of training, whereas the discriminator's loss more or less continued to fall.

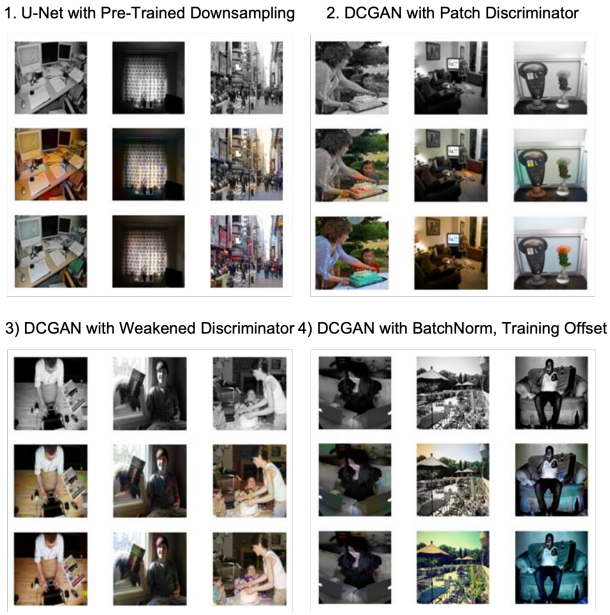

Fig. 3: Results from running the trained models on validation splits of the COCO dataset.

C. Experiment 3: DCGAN with Weakened Discriminator

Since we observed that the original patch discriminator easily outperformed the generator, for the next experiment we weakened the discriminator. We modified the patch discriminator by changing the activation functions from Leaky ReLUs to ReLUs and by removing the batch normalization layers, resulting in faster training and a steeper slope of the loss.

D. Experiment 4: DCGAN with Batch Normalization and Discriminator Training Offset

In this experiment, we first added batch normalization layers between each convolutional and transposed convolutional

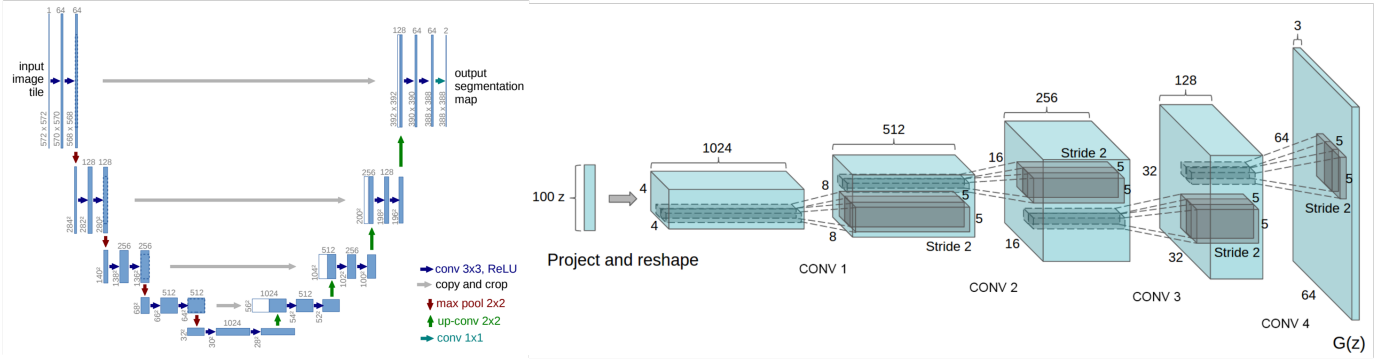

Fig. 2: U-Net model architecture (left) [5] and DCGAN model architecture (right) [6].

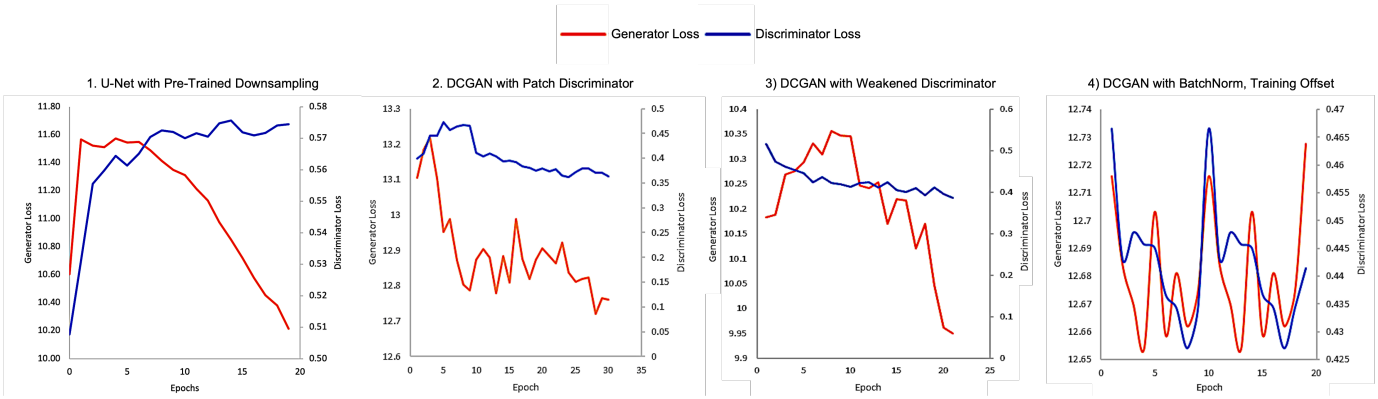

Fig. 4: Comparison of generator and discriminator loss for each of the four experiments. Experiment 4 is the only model that doesn't have a distinct decrease in the generator loss and instead shows a competitive relationships between the generator and discriminator.

layers. Batch normalization layers add a level of regularization to the generator so that it does not simply memorize the input training data, and it can make training faster. Additionally, since the generator contained more layers and parameters than the discriminator, we only trained the discriminator every other time the generator trained. This ensured that the discriminator did not overpower the generator as it initially had.

The result was promising. We see how the model produced similar visual results after training for half of the epochs as the previous models in Figure 3. While the previous experiments had a sharp decrease in the generator's loss, Figure 4 shows that this model led to fluctuations in both generator and discriminator loss.

As shown by Figure 4, the models using DCGANs with the weakened discriminator or the original patch discriminator had a very distinct decrease in the generator's loss, whereas the loss for the model using a DCGAN with batch norm fluctuated. In this respect, one could expect the DCGAN with a weakened discriminator would perform the best out of the three models, but the model using the DCGAN with batch norm actually yielded more aesthetically pleasing results. The three models are relatively good at picking up live objects (humans, dogs, cats etc.) compared to inanimate ones. This might be due to the inherent biases within the COCO dataset, as humans are its most reoccurring subject.

In terms of training time, each epoch of the DCGANs with batch normalization took up around 1.5 minutes more than the other two DCGANs models, though it matched or even visually outperformed the other two models with half the number of epochs. Thus, batch normalization does speed up the training time in terms of reducing the number of epochs of training.

E. U-Net and DCGANs Visual Comparison

The images in Figure 5 show screenshots of the colorized film from both DCGANs and U-Net. As seen in the figure, DCGANs had more consistent and certain coloration than U-Net, whose uncertainty is evidenced by the large gray areas in its output. Both models were susceptible to uncertainty when the objects in the frame were unclear. For example in row 3, the colors of the man's face were less accurate for both models when the hand was blocking it, but became sharper when the hand moved away in row 4.

Fig. 5: Comparison of DCGANs and U-Net image colorization outputs for different frames of *Pleasantville* (1998)

TABLE I: Comparison of training efficiency between the U-Net-based model (Experiment 1) and the DCGAN-based model (Experiment 4).

	Avg Time to Train per Epoch (m:s)	Params (M)	MACs (G)
U-Net (Experiment 1)	5:42	31.1	455.14
DCGAN (Experiment 4)	6:55	16.0	496.12

F. Efficiency Analysis

Noting that the DCGAN-based models took significantly longer to train, we wanted to compare the model with the U-Net-based model to see if there were tradeoffs in time complexity. Table I contains the average time to train per epoch, the number of parameters in the generative models, and the number of multiply–accumulate (MAC) operations. We calculated the params and MACs for just the generative models in each GAN model because both use the same patch discriminator architecture.

While the U-Net-based model has more parameters, compared to the DCGAN-based model, the U-Net-based model saves more than 1 minute per training epoch and 40 billion MACs.

Comparing DCGAN models with no batch normalization to the one with batch normalization, each epoch of the DCGANs with batch normalization took up around 1.5 minutes more than the other two DCGANs models. However, it was able to visually match or outperform the other two models with half the number of training epochs. Thus, while U-Nets took less time per epoch, batch normalization does speed up the training time in terms of reducing the number of epochs of training required to reach visually pleasing results.

IV. Conclusion

While the cGANs and U-Net-based architecture tends to take less time during training and uses fewer MAC operations, the DCGANs-based colorizer leads to both more stable training and qualitatively more convincing results. These qualitative differences are especially evident in movie colorization, where the U-Net-based model often leads to large patches of incorrect color, while the DCGANs model avoids such artifacts. For the movie colorization application, future experiments are needed to determine if an RNN-based architecture, which could gain better insight on colorization due to access to past frames, may outperform the DCGANs model.

Acknowledgment

We'd like to thank Professor Panda for the feedback on our project and for a wonderful semester learning about and engaging with neural networks!

References

[1] Vox, "How obsessive artists colorize old photos," *Youtube*, May 30, 2017. Available: https://www.youtube.com/watch?v=vubuBrcAwtY

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks." arXiv, Jun. 10, 2014. Accessed: Dec. 13, 2022. [Online]. Available: http://arxiv.org/abs/1406.2661

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros,"Image-to-Image Translation with Conditional Adversarial Networks." arXiv, Nov. 26, 2018. Available: http://arxiv.org/abs/1611.07004

[4] M. Shariatnia, "Colorizing black & white images with U-Net and conditional GAN — A Tutorial," Medium, Nov. 18, 2020. Available: https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation." arXiv, May 18, 2015. Available: http://arxiv.org/abs/1505.04597

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." arXiv, Jan. 07, 2016. Available: http://arxiv.org/abs/1511.06434

[7] K. Nazeri, E. Ng, and M. Ebrahimi, "Image Colorization with Generative Adversarial Networks," vol. 10945, 2018, pp. 85–94. Available: http://arxiv.org/abs/1803.05400

[8] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context." arXiv, Feb. 20, 2015. Available: http://arxiv.org/abs/1405.0312

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." arXiv, Dec. 10, 2015. Available: http://arxiv.org/abs/1512.03385