GAN Video Colourization

Hanzalah Firdausi
MT21027
hanzalah21027@iiitd.ac.in

Harsh Vardhan Bhadauriya
MT21122
harsh21122@iiitd.ac.in

Shreyansh Jain
MT21089
shreyansh21089@iiitd.ac.in

Abstract

With the recent advances in deep learning it is now possible to colourize an image or even a video without any human intervention. The intention behind this project is to develop an end to end trainable and fully automatic video colourization network such that temporal consistency is maintained throughout the processed colored video. The aim would be to strike a good balance between the vibrant colors and color consistency throughout the video.

1. Introduction

Classical Hindi and English movies inspire the motivation for this project. There are many classical legacy movies in black-and-white format due to the limitation of technology at that time. We aim to reinvent the old classic film and bring forth the beauty and the vividness of the old classics using deep learning methods and techniques to color the black-and-white movies realistically.

1.1. Problem Statement

The problem in our hand is different from simple image colourization because video colourization has two primary issues: temporal and spatial consistency.
The input of our problem is a black-and-white video, and the output is a spatiotemporal consistent coloured video.
We propose leveraging COCO and ImageNet dataset beacuse these dataset contains images of varying shapes and vivid colors.

Our inference is based on a classic Charlie Chaplin clip for which there are no ground truth available. The metric for our work will be consistent and vibrant colors of the generated video.

1.2. Challenges Involved

There is a lot of work done in the field of Image Colorization and while it seems like an easy task to convert a video into frames and colorize each image using any of the Image colorization techniques. However, there are additional challenges faced while attempting to color a grayscale video. Those challenges are-

- `Spatial Consistency` - For any object in a video frame, it is not possible to surely say what color it should be but it is important to maintain that color consistency does not change across the spatial region of an object in the frame. For example, a colored human face should have a consistent color.

- `Temporal Consistency` - In an image we don't have to care about how the next image will be colored but in the case of a video, frames should have consistency when it comes to similar objects across frames throughout the video.

1.3. Motivation

We have many legacy videos of historical and cultural importance that if coloured can reach to a wider and younger audience. The idea that we propose is based on the recent works and presents an architecture that tackles the wider issues with video colourization such as temporal consistency and color diversity. Our aim is to produce results which are natural, consistent across frames and rich in information.

1.4. Contribution

The idea of this project is to go through the recent advancements in the field of video colourization and analyze the better available architectures that are able to tackle the consistency issues. After that we zoom into parts which can be improved such that replacing feature extracting network with a recently developed better performing network or mix the moving parts of different architectures to come up with a new architecture.

Our evaluation metric would be to generate consistent natural color throughout the video.

2. Literature Survey

In this section, we review latest related works in the field of the image and video colourization. Starting with

early scribble based methods to deep learning based image colourization to reviewing video colourization architecture.

Levin et al.[2004] [10] proposed image colourization using user-specific scribbles to color the grayscale image. The method was improved by Huang et al. [2005] [5] to correctly color the image by preventing the color from bleeding over the object boundaries. Yatziv et al. [2004] [14] proposed a technique for fast colorizing images using chrominance blending based on weighted geodesic distances. Luan et al. [2007] [11] employed the texture similarity for effective color propagation. All of the methods, including the method proposed by Xu et al. [2013] [13], relied heavily on the user input and required trial and error to obtain acceptable results. In contrast, color transfer techniques have also been used, which uses mapping between the input image and a reference image to establish the transfer of the color from reference to the input image. The issue with the color transfer method is that the user needs to provide the reference image, which was a time-consuming task.

Cheng et al. [2015] [2] proposed a fully automatic approach in which various features are extracted and the different patches of the image are colorized using a small neural network. They used very little training data, limiting the types of images, and the performance depended on the image segments. Due to this limitation, results were poor for images in which none of the segmentation classes were found, and the application got restricted to only outdoor images.

Satoshi et al. [2016] [6] proposed a novel methodology that does not rely on hand-crafted or pre-trained models. The model learns everything in end-to-end fashion from a huge dataset which allows the model to generalize many types of images. The method was developed to color the grayscale images automatically by combining the global priors and the local image features. The global priors deal with the details of the complete image capturing the surrounding whether the image belongs to an indoor or outdoor setup. At the same time, local features represent the local object or texture in the given location of the picture. Combining both image features allows semantic information to color the image without any manual intervention. The model is based on the Convolutional Neural Network using a fusion layer to efficiently extract and merge the local details by extracting the features using the small image patches with the global priors computed using the full image. The architecture proposed was flexible enough to process images of any resolution unconditionally to attain realistic colorization.

The broadcast of the Image colorization technique on video suffers greatly because of inconsistency between frames. We went through multiple papers that tackle these issues.

Lai et al (2018) [8] proposed a methodology to decrease temporal issues but this was done as a post-processing step after applying Image-based colorization. The issue with the approach is that it does not take the temporal issues into consideration while coloring the frames.

Lei and Chen(2018) [9] proposed a model that can combat spatial, temporal consistency(which was missing in Image Colorization) and diverse color without tapping into the Generative Adversarial Networks. The idea presented is to divide the architecture into two parts- the Colorization network and the Refinement network. The colorization network will take care of coloring a frame based on a feature vector consisting of (R, G, B, x, y) for each pixel. To tackle color diversity, it would also generate a number of differently colored frames. The refinement network would solve the temporal issues by taking into consideration the nearby frames and generating a confidence score in range [0, 1] based on how confident we are that the colored frame is consistent with nearby frame pixel values.

The idea here is to tackle the main challenges of video colorization.

The above-proposed methods solved most of the problems faced in the video colourization, but to improve the results further, we can explore how to leverage the power of GANs. GAN was first proposed by Goodfellow et al. [3] which includes two neural networks (i.e., generator and discriminator) that compete against each other.
A GAN's core concept is based on "indirect" training via the discriminator, another neural network that can tell how much the input is "realistic", and the discriminator is also dynamically updated. The above statement means that the generator is not trained to reduce the distance to a particular picture but instead to fool the discriminator. This allows the framework to learn in an unsupervised way.
GAN has been used in colourization to improve the vividness of colourized images or to produce a range of colored image outcomes. Cao et al. [1] directly added noise to the first three layers of the encoder to achieve diverse colourization.
Zhao et al. [15] proposed using an end-to-end video colourization generative adversarial network to combine image and video colourization into a hybrid architecture (VCGAN).
The hybrid model has two significant advantages:

- A single model can be used for both image and video colourization
- A reference frame is provided for initialization to ensure no frames are lost.

VCGAN is proposed as a recurrent architecture that processes input frames sequentially during forward propagation.
The VCGAN architecture includes a mainstream encoder-decoder as well as two feature extractors. The mainstream

uses the U-Net structure. ResNet-50 is used to load the weights of two feature extractors. The first global feature extractor extracts the semantics of the input grayscale frame of the current time, providing high-level information for the network to improve the colour schemes of objects. The second placeholder feature extractor receives the most recently coloured frame in order to improve the spatiotemporal continuity provided by the recurrent connection. The output features of both extractors are concatenated and fed into the mainstream encoder. A patch-based structure is used in the VCGAN discriminator.

3. Methodology

We have focused on training Generative Adversarial Networks to colorize the video. The generator takes the gray image as input and outputs a colored image while the discriminator takes the generator's colored image and true colored image. The discriminator will then give a loss value based on the loss function used. The Discriminator total loss(loss generated by fake colored image and true colored image) will be used to update the discriminator weights and loss generated by just the generator will be used to update the generator weights.

Although RGB is the most widely used method of expressing images. However, there are a variety of options available, one of which is the Lab colour space (also known as CIELAB).We have used Lab color space to train our model. The Lab color space expresses colors as three values:

- L: the lightness on a scale of 0 to 100, which is a grayscale image – it's not quite the same as converting RGB to grayscale, but it's near enough.
- a: colour spectrum spanning from -128 (green) to 127 (red)
- b: colour spectrum spanning from -128 (blue) to 127 (yellow) The obvious conclusion from the preceding description is that by using the Lab colour space, we solve two problems at once: we have the grayscale image as input, and the complexity of the colour channels is reduced from three to two (in comparison to RGB).

We have proposed various architectures and below are their configurations:

3.1. Colorization Net

The idea here is to have to generate colored frames by passing gray frames as input to the encoder(generator) and then use the loss generated by the discriminator to update the generator and make it produce better-colored frames. In the meantime, the discriminator will also learn how to get better at spotting fake image. In (Figure 1), we have shared outline of the architecture of our proposed model.

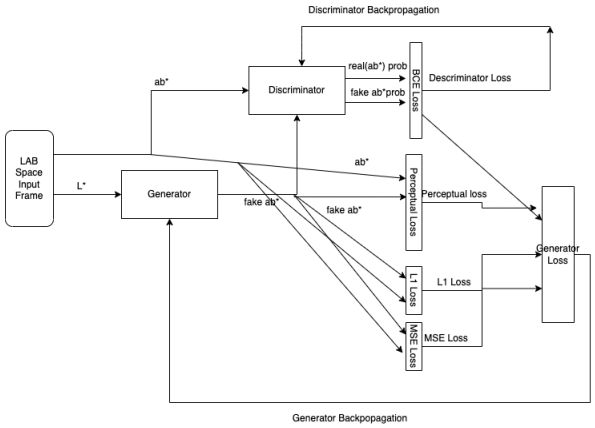

Figure 1. This is the architecture of the Colorization Net proposed.

For the generator:

- We have utilized a pre-trained resnet18 and used its first 6 layers to produce mid-level features of each frame.
- The mid-level features are then passed through a series of convolutional layers and batch normalization and finally, we keep 3 channels as output.
- Finally upsampling is done by a scale factor of 2 to get the original input size image back but with colors. We will keep the resnet layers trainable as well

For the discriminator:

- We have used the PatchGAN discriminator to train our model.
- Kernel size used are [64, 128, 256, 512]. Finally, we output the patch of channel 1.
- We get an output probability of the image being fake on an NxN patch size which depends on the input size of the batch loader. If we are using a 256x256 frame, we get a 70x70 patch gan.

Loss Used: We have used four different loss functions to train the model. The idea to train on multiple loss is taken from Thasarathan et al [12].

- Discriminator uses only Binary Cross-Entropy loss as its function is to output the probability between 0 and 1 of the frame being real. Both real and fake colored frames are passed through the discriminator and their average is taken as the final discriminator loss.

- The generator uses Binary Cross-Entropy, MSE, L1 Loss, and Perceptual loss. The sum of all four losses acts as the final generator loss

In Perceptual Loss proposed by Johnsonet al. [7], we extract high-level features from the pre-trained network which in this case is VGG. We extract high-level features at five stages which are activation outputs after relu1 1, relu2 1, relu3 1, relu4 1, and relu5. L1 Loss is calculated at all five stages and combined to give the final perceptual loss. We have used the implementation of perceptual loss in https://github.com/Harry-Thasarathan/TCVC/blob/master/src/loss.py.

Training:

- The model was trained on 150 epochs.
- We found out that the discriminator is able to learn easily in comparison to the generator and to stop the discriminator from becoming too smart, the discriminator was updated after every four iterations during the first 50 epochs.
- This gave the generator time to learn and adapt to the real colorization.
- For the next 100 epochs, the discriminator was trained after every 2 epochs.
- In both cases, we trained the generator at every epoch.
- Discriminator loss is the average of BCE loss generated by both real and fake colored frames.
- Generator loss is the summation of the BCE Loss, MSE Loss, L1 Loss, and Perceptual Loss. All the 4 loss functions are summed up with a ratio of 1:1.
- The generator was trained with a learning rate of 3e-5 while the discriminator was trained with earning rate of 1e-5. A smaller learning rate for the discriminator was used so that the discriminator learning was slower compared to the generator.

Dataset:

A dataset of 8.2 k images was used which had around 5k images from ImageNet and other images scraped from the Flickr using flickrapi. Images with keywords buildings, scenery, movie scenes, animals, etc were scraped.

3.2. Colorization with U-Net

The architecture of this model (Figure 2) is very similar to the above model but there are some differences that help it to achieve better results.

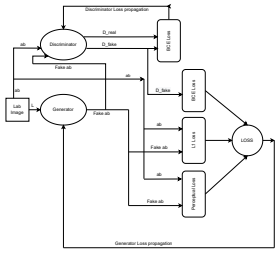

Figure 2. Architecture of Colorization with U-Net and GAN

- The Generator is an encoder-decoder model with Resnet-18 pre-trained on the ImageNet in the encoder part and U-Net in the decoder part.
- The discriminator is similar to the above model.
- We have used a new form of training which we call the No Gan training. In the No Gan training, we first pre-train the generator for some epochs and then do the adversarial training to better the performance of the generator. The idea behind No Gan training is that when both Generator and Discriminator are not pre-trained it leads to a problem of "blind leading the blind".
- We don't pre-train the Discriminator because the job of the discriminator is easy as compared to Generator, hence it learns quicker than Generator, so it does not need pretraining.
- We have used L1 Loss to pre-train the Generator.
- We have used three losses to train our Generator during the adversarial training phase namely the Perceptual Loss, L1 Loss, Binary Cross-Entropy, and only Binary Cross-Entropy Loss for Discriminator.

Training:

- The generator was pre-trained for 70 epochs with the help of Adam Optimiser with default beta values and a learning rate of 1e-4 using L1 Loss.
- We found out that the discriminator is able to learn easily in comparison to the generator and to stop the discriminator from becoming too smart, the discriminator was not pre-trained.
- This gave the Generator time to learn and adapt to the real colorization.
- The Generator and Discriminator were trained simultaneously.
- Discriminator loss is the average of BCE loss generated by both real and fake colored frames.

- Generator loss is the summation of the BCE Loss, L1 Loss, and Perceptual Loss. The weightage of all the loss functions are same.
- The generator was trained with a learning rate of 2e-4 with the help of Adam optimizer with beta1 set to 0.5 and beta2 set to 0.999.
- The discriminator was also trained on the same configuration with the help of Adam optimizer with the same beta values as above.

Dataset:
A dataloader of 8k sample of COCO dataset is used to train the model. 2k sample from the COCO dataset is used to validate the model. The reason behind selecting this dataset is that it contains images which are varying in shapes and vivid in colors.

3.3. Colorization with U-Net and Wasserstein loss

The architecture of this model is same as the above model but there is addition of the Wasserstein loss for the training which enables the generator model to better approximate the data distribution of the train set based on the argument that training the generator should minimize the distance between the distribution of the data observed in the training dataset and the distribution observed in generated example.

- We have used L1 Loss to pre-train the Generator.
- We have used Wasserstein loss with gradient penalty for Discriminator.

Training:

- The generator was pretrained for 10 epochs with the help of Adam Optimiser with default beta values and learning rate of 1e-4 using L1 Loss.
- We found out that the discriminator is able to learn easily in comparison to the generator and to stop the discriminator from becoming too smart, the discriminator was not pre-trained.
- This gave the Generator time to learn and adapt to the real colorization.
- The Generator and Discriminator were trained simultaneously with discriminator trained for for four iteration for every iteration of Generator as described in [4]
- Discriminator loss is the Wasserstein loss generated by both real and fake colored frames with gradient penalty.
- Generator loss is L1 Loss.
- The generator was trained with a learning rate of 2e-4 with the help of Adam optimiser with beta1 set to 0.1 and beta2 set to 0.9.
- The discriminator was also trained on the same configuration with the help of Adam optimiser with same beta values as above.

Dataset:
A dataloader of 8k sample of COCO dataset is used to train the model. 2k sample from the COCO dataset is used to validate the model. The reason behind selecting this dataset is that it contains images which are varying in shapes and vivid in colors.

4. Experiments

In this section, we look into the results of our architecture and analyse them.

4.1. Colorization Net

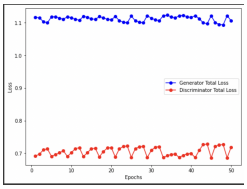

Figure 3. Loss function of Colorization Net for first 50 epochs

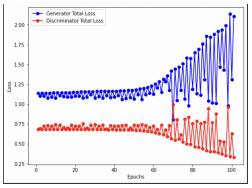

Figure 4. Loss function of Colorization Net for next 100 epochs

Analysis of Loss Graph: As we can see in (Figure 3), in the first 50 epochs we have updated the discriminator on the fourth epoch. This is why discriminator loss increases for the next 3 epochs as the generator becomes better at creating fake color frames. At the fourth epoch, the discriminator updates itself and brings the loss down. This is the trend observed over 50 epochs. Similarly, with GAN loss, it decreases for 3 epochs and when the discriminator is updated the GAN loss rises again. We are able to keep the loss almost constant this way.

For the next 100 epochs in the (Figure 4), loss starts to increase after the 60th epoch which can also be seen from the results obtained on gray images. The model trained after 90 epochs produce worse results in comparison to

models before 90 epochs.

Analysis of Colorized Output:

In (Figure 5), we can see that the colors produced by model after 90 epochs are much more natural and vibrant. The results produced by the model after 150 epochs are unnatural with a spill of red around the frames. This also supports the evidence given by Loss function in (Figure 4). The increase in the generator and discriminator loss can also be seen in the output as it deteriorates.

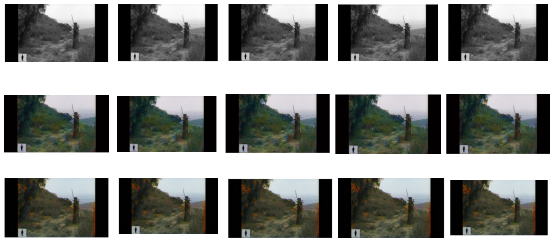

Figure 5. Row 1 describes the input gray frames, Row 2 shows the output produced after 90 epochs of Colorization Net, and Row 3 shows the output produced after 150 epochs of Colorization Net

4.2. Colorization with U-Net

Analysis of Loss Graph:

As we can see in (Figure 6), the loss of Generator keeps on decreasing which shows that the Generator is continuously learning and the Discriminator's loss is also steady which shows that the Generator is producing more better results with every epoch.

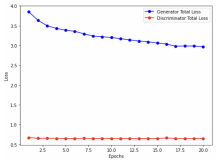

Figure 6. Loss function of Colorization with U-Net and GAN

Analysis of Colorized Output:

In (Figure 7), we can see that the colors produced by model after 20 epochs are much more natural and vibrant.This also supports the evidence given by Loss function in (Figure 6). The decrease in the generator's loss can also be seen in the output as it produces better results.

4.3. Colorization with U-Net and WGAN

Analysis of Loss Graph:

As we can see in (Figure 8), the loss of Discriminator keeps on decreasing while the loss of Generator remains almost constant for the training on the frames which shows that Generator is unable to learn how to color the frames and same can be seen by observing the output.

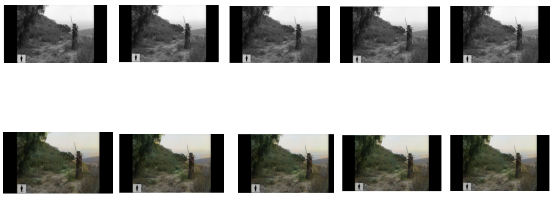

Figure 7. Row 1 describes the input gray frames, Row 2 shows the output produced after 20 epochs of Colorization with U-Net

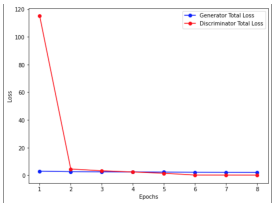

Figure 8. Loss function of Colorization with U-Net and WGAN

5. Conclusion

We have implemented the task of video colorization using the GAN architecture. Our model of Colorization net could color the frame satisfactorily, but we can observe a hint of temporal inconsistency. However, the Colorization model with U-net produced much more vibrant and natural color and produced the best output out of all the models trained. We could have performed better if we had more resources with us as it was a computationally challenging task to train the model.

References

[1] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. *CoRR*, 2017. 2

[2] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE international conference on computer vision*, pages 415–423, 2015. 2

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017. 5

[5] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 351–354, 2005. 2

[6] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, 35(4):1–11, 2016. 2

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 4

[8] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *Proceedings of the European conference on computer vision (ECCV)*, pages 170–185, 2018. 2

[9] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[10] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004. 2

[11] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 309–320, 2007. 2

[12] Harrish Thasarathan, Kamyar Nazeri, and Mehran Ebrahimi. Automatic temporally coherent video colorization. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 189–194. IEEE, 2019. 3

[13] Li Xu, Qiong Yan, and Jiaya Jia. A sparse control model for image and video editing. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013. 2

[14] Liron Yatziv and Guillermo Sapiro. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, 15(5):1120–1129, 2006. 2

[15] Yuzhi Zhao, Lai-Man Po, Wing Yin Yu, Yasar Abbas Ur Rehman, Mengyang Liu, Yujia Zhang, and Weifeng Ou. VCGAN: video colorization with hybrid generative adversarial network. *CoRR*, abs/2104.12357, 2021. 2