CS230

Video Colorization

Vedi Chaudhri
Stanford University
Department of Computer Science
vchaudhr@stanford.edu

Yang Fang
Stanford University
Department of Computer Science
yangfang@stanford.edu

Abstract

This project attempts to apply deep learning to grayscale video colorization. One current video colorization technique is to independently color individual frames using an image colorization neural network. This method, however, leads to temporal inconsistencies in the coloring of consecutive frames. This project proposes additionally encoding the last previously colorized frame as input to help colorize the current frame in a more positionally consistent manner.

1 Introduction

This project explores deep learning methods for the colorization of grayscale videos. Video colorization is an interesting problem with potential applications for many areas, with the one that comes to mind immediately being the automatic colorization of old black-and-white movies. Video colorization could also prove useful for improving or replacing video compression algorithms, since RGB colored videos use three channels to store information whereas grayscale videos only require storing a single channel. So if a platform needs to store information in color video format, it could potentially store it more compactly as a grayscale video and then later convert it to a colored video when it is retrieved.

In order for this to be possible, however, algorithms for video colorization need to be improved. Currently, there exist neural networks that can color black and white images with relatively realistic results. This is primarily done through the use of convolutional neural networks due to their ability of sharing parameters and having sparse connections. Each frame of the video can be seen as a black and white image that needs to be colored. But the task of realistically colorizing a video is more complex because it requires that the relative position and context of each frame be taken into account. When coloring independent frames using a pretrained image colorization network, certain objects might be colored differently in different frames even though it is the same object. For this reason, to more realistically colorize a video, frames must be colored relative to one another. Instead of independently processing each frame, this project experiments with methods of recolorization that consider the relation of different frames and the relative context of each video frame when performing image colorization.

2 Related Work

The problem of grayscale video colorization has been studied by several other researchers. The first part of video colorization is independently coloring each frame. This common approach to solving this has been utilizing a CNN and posing this as a classification task using class-rebalancing at training time to increase the diversity of colors in the result [4]. The image colorization task is extended to colorize videos as seen in the paper "Consistent Video Colorization" [2]. For their project workflow, they examined N previous frames of a video in order to better recreate a relational basis for coloring the current frame. Their framework explored a hybrid supervised/conditional generative adversarial networks. When they trained their model on consecutive frames, especially in the case when conditioning on the previous ground-truth colorized frame, the consecutive GAN model achieved high pixel accuracy, but when color propagation was applied throughout a video, colorization errors occurred on fast moving objects. Another problem that they encountered was exponential error propagation caused by the distribution difference between the distribution difference between training examples and what GAN has to work with for color propagation. To solve the problem of coloring fast moving objects, the first step is to recognize motion and continuity of objects in the video [6]. Furthermore, researchers used a temporal propagation network architecture to classify images on each frame and color the objects consistently using bi-directional training on pairs of frames [5].

This problem was also explored in "ColorNN Book: A Recurrent-Inspired Deep Learning Approach to Consistent Video Colorization" [1]. They constructed an architecture that inherits layers from a VGG-16 model trained on ImageNet and employed transfer learning to create a baseline coloring model. Then, to look at frames in relation to one another, they used a recurrent neural network since the neurons in a recurrent network preserve a selective temporal memory, and output different things based on the input ordering of examples. This model proved to be too computationally costly though so instead of using recurrent nodes, they ran the Y-channel input of a frame t through the color model, and takes in the estimated coloring of the previous frame as additional input.

The approach of this project hopes to build upon the findings of these research papers and improve upon the problems of exponential error propagation and aim to decrease computational costs.

3 Data Set

This project uses the open source Moments in Time dataset developed by the MIT-IBM Watson AI Lab, which includes a collection of one million labeled 3-second videos. The 500-video hiking subdataset was specifically selected and converted to grayscale. Then, the grayscale videos were colorized using a pre-trained image colorization network and center-cropped to produce 128 x 128 colorized videos for training. We then shuffled and split the videos into 80% train / 10% dev / 10% test, and constructed a dataset consisting of labeled examples of (X=[previous colorized frame, current colorized frame], Y=[current true frame]).

4 Method

4.1 Baseline

A simple baseline implementation simply leverages the pretrained image colorization model presented in Zhang, Isola, and Efros's "Colorful Image Colorization" and applies it to independently colorize each frame [4]. While this image colorization model generates relatively realistic recolorizations on an image-by-image basis, given that the positional context of each frame is not taken into account, this baseline algorithm produces videos with large color-shift variations between frames. This is especially apparent during inspection by human eye -- large color oscillations occur throughout the recolorized video.

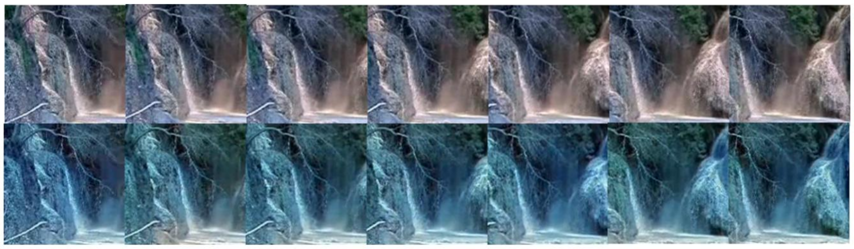

Figure 1: Baseline Colorization Algorithm (frames sampled 10 apart). Top: true colorization, Bottom: baseline colorization

Using the baseline algorithm, we found that frames that are moving quickly seem to do worse than frames that are moving more slowly. The pixel match percentage accuracy (binary pixel difference averaged across all frames) of videos selected to encompass a variety of environments, such as the forest, desert, ocean, vary from 23% to 40% in similarity to the ground truth videos. Pixel-match accuracy in itself, however, can be a misleading metric since pixel match accuracy is a binary metric that doesn't account for the degree of variation between pixel values.

4.2 Loss Function

As mentioned in the previous section, exact pixel match accuracy is a relatively simple and non-holistic metric for good video colorization, and leads to inconsistent colorization of each individual frame. For our loss function, we wanted to reward colorizations for being closer in pixel value to the true video, and thus we used a MSE loss to penalize large pixel value differences. Additionally, we wanted to reward colorizations for consistently coloring consecutive frames, and so we also calculate the MSE loss between the current predicted frame prediction and the previous predicted frame. We then take a weighted sum of the two losses to complete our loss function, which is defined as:

$$\beta * \tfrac{1}{N} \textstyle\sum_{i=1}^{N} (\hat{y}_i - y_i)^2 + (1 - \beta) * \tfrac{1}{N} \textstyle\sum_{i=2}^{N} (\hat{y}_i - \hat{y}_{i-1})^2$$

If β, a tunable parameter, is 1, then we only care about the true frame (independent frame colorization); if β=0, then we only care about the previous frame (results in duplication of previous frame). In practice, β=0.8 was selected, with the intuition that it is more important to first color the frame to closely resemble the true frame color and then adjust the colorization to be similar to that of the previous frame's. Finally, it's important to note that this loss function, like exact pixel match, is simply a heuristic for approximating a measure of video colorization. The final performance evaluation of the algorithm should be conducted by human inspection, as the purpose of the project is to produce a realistic video recolorization, rather than an exact one. For example, we are satisfied if a grayscale shirt is plausibly recolorized as red when, in reality, it was actually blue.

4.3 AlexNet CNN

The first model architecture we trained was an AlexNet-based CNN. We used Keras to implement an AlexNet model that accepted a concatenation of the current and previous 128 x 128 x 3 colorized frames as input, and then replaced the FC output layers with deconvolutional layers to output a 128 x 128 x 3 colorization for the current frame. The model resulted in ~9 million trainable parameters and is shown below:

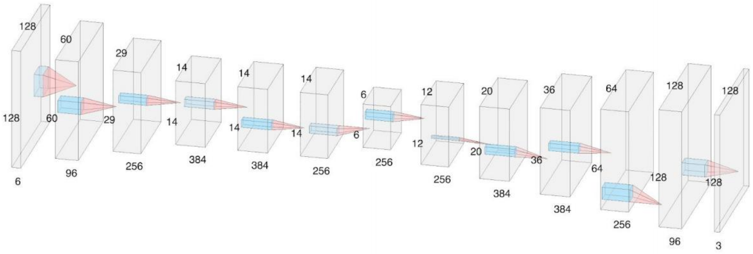

Figure 2: AlexNet-based CNN

4.4 VGG16 ImageNet CNN

After training and analyzing the results of the AlexNet CNN, it became apparent that there was an insufficient amount of data to train a model to both recognize and preserve object shapes as well as perform video colorization. Thus, instead of training a CNN from scratch, it was proposed that we instead fine tune the Keras pretrained VGG16 ImageNet model. We first froze all of the existing VGG16 layers, then prepended a concatenation and convolutional layer to allow the acceptance of the current and previous 128 x 128 x 3 colorized frames as input. Then, as in the AlexNet CNN, we appended a series of deconvolutional layers to output a 128 x 128 x 3 colorization for the current frame. The model resulted in approximately 7 million trainable parameters and 15 million non-trainable parameters.

5 Results and Analysis

When training our initial models, we noticed that the models would quickly learn to colorize each frame as black to minimize loss, converging within the first training epoch. After trying a series of gradient dampening techniques and other optimizations, it was found that the resizing of the videos during preprocessing caused many to have significant amounts of black padding, thus causing the models to optimize for producing black frames. The implemented solution, then, was simply to take a 128 x 128 center crop of each frame to eliminate the black regions. We then trained the AlexNet and VGG16 models described above, and the results of training for 100 epochs are visualized in Figure 3 below:

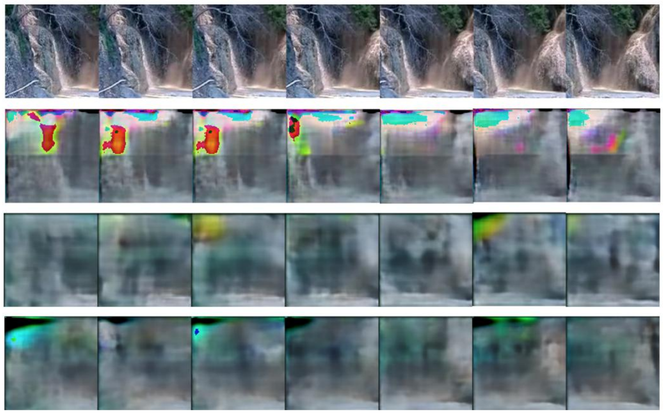

Figure 3: Video Colorizations. Top to bottom: (1) True colorization, (2) AlexNet CNN, β=0.8, (3) VGG16 ImageNet CNN, β=1, (4) VGG16 ImageNet CNN, β=0.8

As seen in Figure 3.2, when first training an AlexNet model, it became quite apparent that while the model was learning to minimize the loss function relatively well, it lacked the data and complexity to fully learn the contents of each image. As a next step, we hypothesized that the results could be improved if we incorporated a pretrained

VGG16 ImageNet model and fine tuned it to generate consistent video colorizations while preserving image contents. The trained VGG16 ImageNet model seemed to produce colors that were closer to the true color than the model using the AlexNet CNN (Figure 3.3), likely due to its use of a pretrained image network as well as the fact that it simply has more parameters and higher complexity. While the models unfortunately generated blurry images, they did noticeably reduce the amount of error propagation (a common issue for video colorization algorithms) since an independant colorization of the current frame was used to "checkpoint" the colorization of the current frame, given the previous frame.

An examination of the training loss over time also yielded some interesting observations:

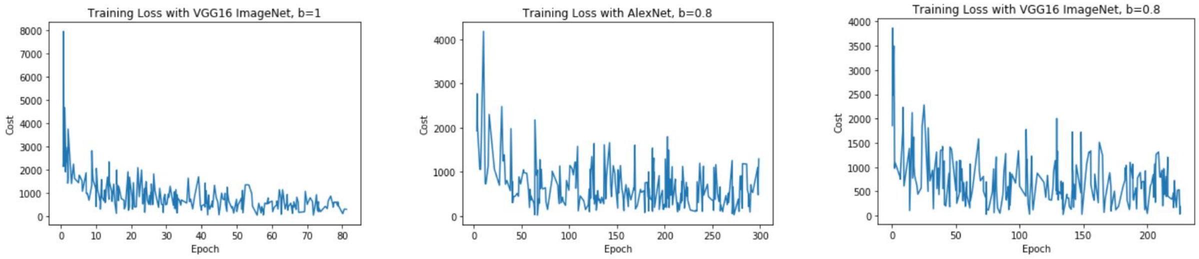

Figure 4: Training Loss vs. Epoch.

The VGG16 ImageNet with $\beta = 1$ model converges most quickly, is less noisy, and has a smaller stable cost, further confirming that its generations are closer in color to the ground truth when compared to other models (see Figure 3). In all cases, there were likely insufficient training examples for the models to learn both to recognize object shapes as well as to optimize for a similar colorization between frames, resulting in blurry frames. The VGG16 likely performed better because it fine tunes a pre-trained ImageNet network, and thus performs better given few training examples. Additionally, a model might perform better when $\beta = 1$ compared to when $\beta = 0.8$ because there's a slight mismatch between the input and the loss function since the input is using the previously colorized frame, whereas the loss is using the generated previous frame.

6 Conclusion and Future Work

From our results, we can conclude that having β values closer to 1 results in faster model training and leads to video colorizations that are closer to the ground truth video when compared in terms of pixel difference, but more experimentation and human agreement would be necessary to determine the exact optimum β value. The trained models all seemed to suffer from a lack of training data, as supported by the resulting blurry frame colorizations as well as the dramatic improvement in color consistency when incorporating a pretrained ImageNet network. Overall, this project was a rewarding deep learning experience, through which we were able to explore methods of improving pre-existing video colorization techniques, gain real-world experience working with CNNs, and better understand the importance of the iterative deep learning process.

If given more time and resources, it would've likely been rewarding to continue exploring different, more complex models and optimization techniques, as well as attempting to gather and train with a larger dataset (video training is quite memory-intensive, and we were limited in terms of storage / computing power). For example, given enough data, it would be interesting to see the results of unsupervised training techniques on video colorization. Additionally, instead of only considering the previous generated frame, another interesting feature to explore would be conditioning on a sliding window of the previous *n* generated frames (or even all previously generated frames), perhaps using a technique such as keeping a running average of each pixel value.

7 Contributions

Both members of the team were in charge of conducting initial research about the topic and understanding the state-of-the-art video colorization techniques. Afterwards, we worked together to define a loss function and design an architecture that would improve upon the currently existing ones. Yang set up the GitHub repository and built the model and training architecture for the video colorization model. Vedi set up an EC2 instance on AWS and conducted experiments by training a series of models with different hyperparameters. Both members worked together in analyzing the results and creating the project poster and report.

9 Acknowledgements

We would like to give a special thanks to Gael Colas, our project TA, for always being a resource to answer questions that we have. We would also like to thank Kian Katanforoosh for initially helping us think through our project problem statement.

10 Codebase

This code for this project can be found at: https://github.com/yangfangk/cs230-video-colorization

11 References

[1] Divyahans Gupta, Sanjay Kannan. *ColorNN Book: A Recurrent-Inspired Deep Learning Approach to Consistent Video Colorization*. 2016, http://cs229.stanford.edu/proj2016/report/KannanGupta-ColorNNBook-report.pdf.

[2] Gael Colas, Kevin Lee, Rafael Rafailov. *Consistent Video Colorization*. 2018, cs230.stanford.edu/projects_fall_2018/reports/12444081.pdf.

[3] MIT-IBM Watson AI Lab. “moments in time” dataset. http://moments.csail.mit.edu

[4] Richard ZHang, Phillip Isola, and Alexei Efros. Colorful image colorization. 9907:649-666, 10 2016. https://github.com/richzhang/colorization.

[5] Liu, Sifei, et al. *Switchable Temporal Propagation Network*. NVIDIA, UC Merced, Dalian University of Technology, 4 May 2018, arxiv.org/pdf/1804.08758.pdf.

[6] Bashkirova, Dina, et al. *Unsupervised Video-to-Video Translation*. Boston University, 10 June 2018, arxiv.org/pdf/1806.03698.pdf.

[7] He, Mingming, et al. *Deep Exemplar-Based Colorization*. Hong Kong UST, University of Science and Technology of China, Microsoft Research, 21 July 2018, arxiv.org/pdf/1807.06587.pdf.