Colorization of video games' images using Convolutional Neural Networks

Capstone Project Proposal

Domain Background

For the capstone project, I would like to explore a domain of computer vision, namely image colorization. From as long as I can remember, I was a video games fan. That is why I want to combine my two passions, the other one obviously being data science :).

Recently, I saw a few posts on the Internet showing that using Deep Learning it is possible to enhance the quality of emulated video games, for example by upscaling the resolution to 4K or increasing the quality of the textures. What is really amazing is that these solutions work out of the box for all games, not only for one or two.

Of course, doing a project on such a scale is much more complex than what I intend to do for this capstone project. When I was starting my adventure with video games, I was playing on GameBoy Color and saw both grayscale and color games working on the same machine. In line with those observations, I will try to use Deep Learning to colorize grayscale images.

I will base my project on the following sources:

[1] https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/
[2] Baldassarre, F., Morín, D. G., & Rodés-Guirao, L. (2017). Deep koalarization: Image colorization using cnns and inception-resnet-v2. arXiv preprint arXiv:1712.03400.
[3] https://lukemelas.github.io/image-colorization.html

In case I use any other sources, I will add them to the references section of the final report.

Problem Statement

The problem I would like to tackle in this project is quite simple – to colorize grayscale images (to a satisfactory degree) coming from old-school video games. To do so, I would like to use Deep Learning, in particular an Auto-Encoder based on Convolutional Neural Networks.

Datasets and Inputs
For the problem at hand, I would like to use frames captured from a YouTube video showing a complete playthrough of one video game in color. The video can be found here: https://www.youtube.com/watch?v=btnFUbexxEE.

The video has 24 frames per second, and I will extract every 58th frame. Doing so results in 7640 images. I will use 10% of that for the validation sample (not seen during training) and 90% for training.

All the pre-processing will be done within a PyTorch DataLoader, however, I can describe all the steps:

- I will use the CIELAB color space, which consists of a lightness layer L (the grayscale image) and two color channels (a and b respectively). This way, I actually only need to convert the grayscale into 2 layers not 3 as in the case of RGB. In other words, the grayscale layer will be the input to the model (one channel image), and the output will be the two color layers of the LAB color space.
- As additional preprocessing steps, I will resize/crop (random or center) the images to either 256x256 or 224x244 (in line with the input for ResNet/VGG type of networks). The transformation will be the same for all the images, to ensure the comparability of the results.
- For evaluating the results, I will be comparing the original color layers to the ones obtained from the Auto-Encoder network by the means of MSE. So as this is a regression problem, there will be no explicit labels (even though the structure of the images in folders is the same as in the case of classification tasks).

All the data-splitting and preprocessing will be done with prespecified seeds, to ensure the reproducibility of the research.

Ideally, the project could be expanded to capture more than 1 video game, potentially from the same genre (for example, platformer), to increase the performance and cover for potential overfitting. For simplicity though, I will start with one game.

In the attached zip file, I am including a small sample of the dataset, two directories with 100 JPG images in both training and validation sets.

Solution Statement
In this project, I will train a custom CNN defined in PyTorch. The model will use a combination of the convolutional and upsampling layers to extract important

information about the image and map the grayscale representation into a colored one.

In the end, we can create an Endpoint, which will return a colored image provided a grayscale input. However, as the domain of the project is quite limited (one game only), I will end the project with a trained and optimized network and creating an estimator within SageMaker for predictions.

Benchmark Model

For the benchmark, I will use the Auto-Encoder presented in this blog post [1], particularly the alpha variant. The reason for that is that it is the simplest model I found during my literature study. Then, I will try to outperform it using some different, more complex architectures.

Evaluation Metrics

It is quite hard to select a good evaluation metric, as the coloring is to a great extent subjective and requires human control. However, a popular solution is to use the Mean Squared Error (MSE) as a loss metric. If there is enough time, I will experiment with more sophisticated pixel-level metrics.

In the end report, I will present the MSE of different models at different epochs (for example, 25th and 50th, depending on the training time) and print out the original pictures vs. the colored one for manual inspection.

Project Design

1. Downloading the video with a complete playthrough of a selected game from YouTube
2. Extracting frames from the video and splitting them into training and validation sets
3. Building custom DataLoaders with required transformations of the images (converting RGB to Lab, treating the *L* layer as input, the *a* and *b* as targets.
4. Defining a few CNN-based Auto-Encoders (at least the ones from the two blog posts mentioned in the literature review) to colorize the grayscale images. Potentially, I would like to see the network from [3], while replacing the ResNet with a VGG-16 network.
5. Training the model using SageMaker's Pytorch wrapper in order to utilize GPUs
6. Creating an estimator based on the trained network and evaluating the results of the colorization