Deep Learning: Colorization of Black and White Videos with CNN and LSTM

Anders Emil Pedersen[1] - s144771, Victor Ibsen[2] - s163928

1 Autonomous Systems, Technical University of Denmark; 2 Human-Centered Artificial Intelligence, Technical University of Denmark

Introduction

Adding colors to black and white images is a very time-consuming process and keeping details such as wrinkles in clothes and foam on top of a wave, is almost impossible for humans to achieve. In the recent years the power of AI and Deep Learning have been used to achieve this colorization of images task from end-to-end, very fast and with great results.
An image in RGB color space have 3 channels that each can have a value between 0 and 255. That is in total $\mathbf{256^3}$ different possibilities. To make it easier for a deep learning model to colorize images, we can transfer the image to LAB color space. This color space consists of a light channel (L) and two color channels (*a and *b). This reduces the amount of guesses the model have to make to $\mathbf{256^2}$ which is millions of possibilities less. Two examples of an image in RGB color space and L*a*b color space can be seen on figure 1.
In this project we will construct a deep learning model with and without a LSTM layer to colorize videos specifically. The hypotheses is that videos have images that only change slightly each frame and that a LSTM layer therefore can be used to colorize more effectively.

Figure 1: Original RGB images to LAB channels, here shown separately

Key Points

- We construct a PyTorch DataLoader with images preconverted from small video files. In total 20 different videos are used for training of the model, 10 for validation. The order of which a video is loaded into the DataLoader is random where as the images within that video stays sequential.
- A U-Net is build to act as the generator of our GAN. This U-Net is composed of an encoder and a decoder. The encoder consists of 7 layers, which each have the following structure: LeackyReLU → Convolution → BatchNorm. The decoder consists of 7 layers as well, having the following structure: ReLU → ConvTranspose → BatchNorm. In the middle of the encoder and the decoder, lies the LSTM layer.
- A PatchDiscriminator is being generated to act as the discriminator of our GAN. It consists of 4 layers where 3 of them follow the structure: Convolution → BatchNorm → LeackyReLU. The last layer is just a convolution.
- Training of the model is done with 50 epochs because of limited hardware and time.

CNN and LSTM

CNN (Convolutional Neural Network) is commonly used when working with images. Traditional neural networks are not ideal for image processing because an image normally consists of a lot of pixels. For a 1920x1080 resolution image there is more than 2 million pixels. The reason that the CNN is a nice tool for image processing is that it enables down and up scaling of the image resolution without loosing to much information. LSTM (Long-Short Term Memory) on the other hand, is often used for sequential inputs and to predict results. It is a type of RNN (Recurrent Neural Network) but with extra memory implemented to battle the vanishing gradient problem, for longer sequences. A video could be seen as a sequential set of images, and therefore a LSTM layer should be able to predict the next color of a pixel after a set of similar images.

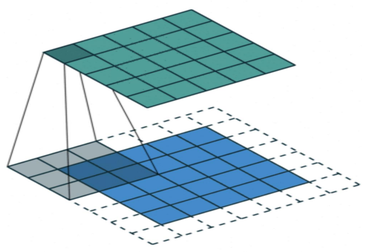

Figure 2: CNN convolution example with padding

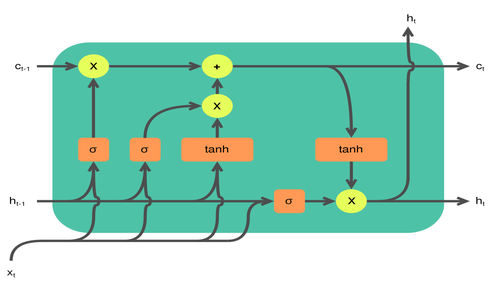

Figure 3: LSTM Cell with all gates modelled

GAN

GAN (Generative Adversarial Networks) is an approach to generative modeling using deep learning methods. It is considered a clever way of training a model because it consists of two sub-models. A generator and a discriminator. The goal of the generator is to create something "fake" that the discriminator mislabels as "real". The discriminator tries to estimate if the input is fake or real. The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples. In this project it means that the generator is applying colors to the training image that is nearly perfectly colorized.

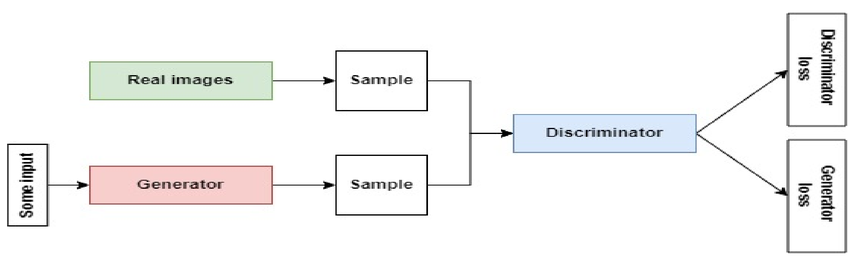

Figure 4: Typical GAN Pipeline, with Generator and Discriminator in sequence

Model

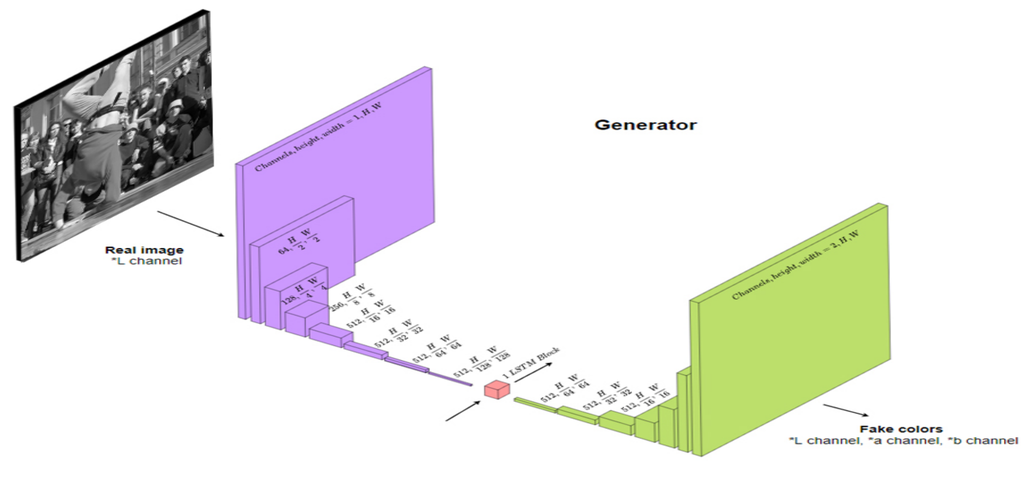

Figure 5: Visual model of the generator

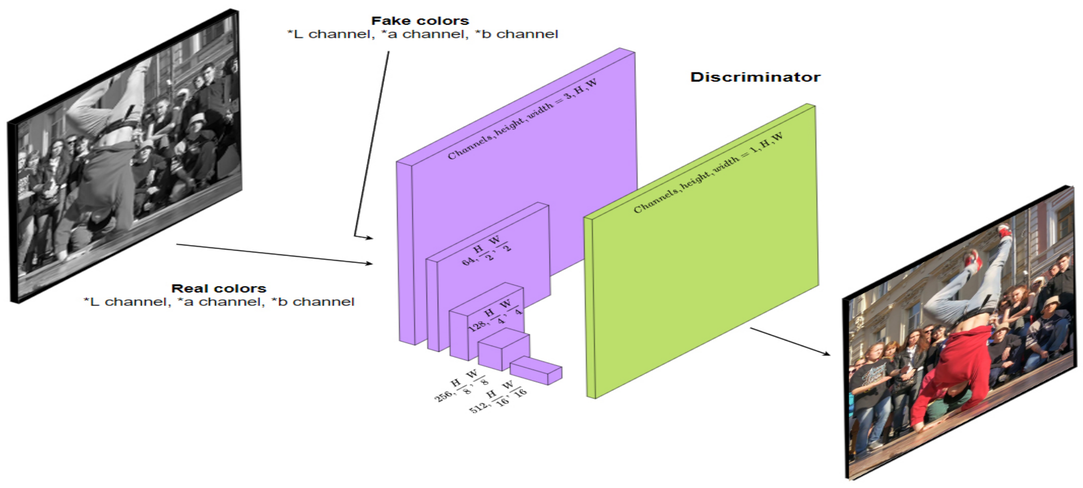

Figure 6: Visual model of the discriminator

Model Losses

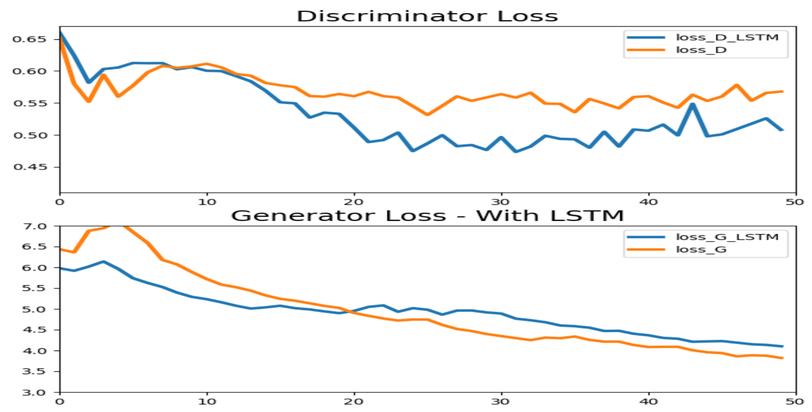

Figure 7: Generator and Discriminator losses. LSTM NN vs Standard NN

Results

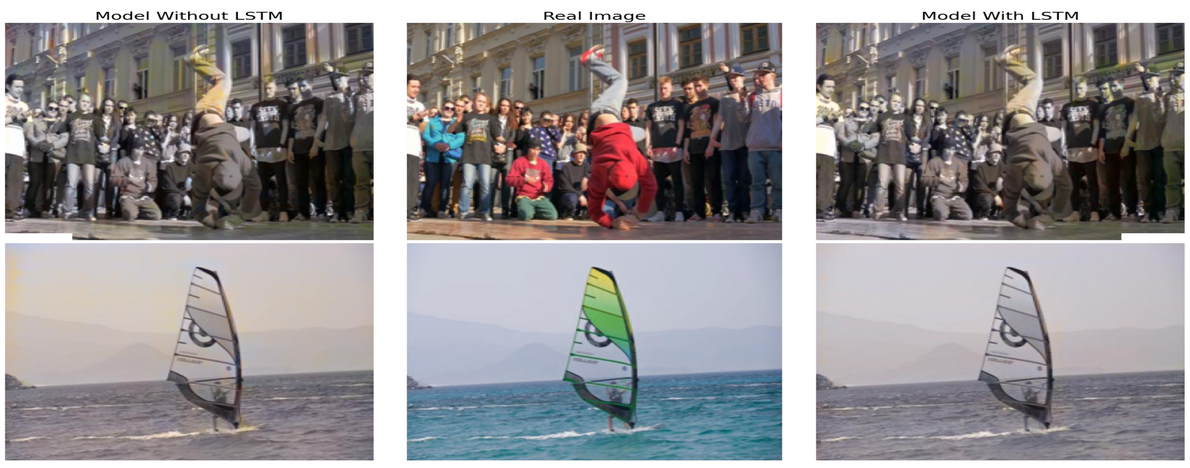

Figure 8: Standard NN (left), True image (middle), LSTM NN (right)

Conclusion

From our results it is clear to see that our trained model is far from perfect. Both the model with and without the LSTM layer lacks colors and in general doesn't look good. One of the reasons for these poor results are the little amount of epochs that we have trained the model with. We would have preferred to train the model with 100 epochs but due to limited hardware, we only trained the model for 50 epochs. For the same reason only 20 videos are used for training, instead of the 40 available or more. Furthermore, tweaking with the learning rate, number of hidden layers, ect. could lead to better results.
One future improvement is to add "feature detection" to our model. This has been implemented by others in video colorization projects, e.g. in the project [1] and [2] which both have astonishing results with the state of the art "ResNet".

References

[1] J. Antic. deoldify. https://github.com/jantic/DeOldify, 2018.
[2] T. Wijesinghe and J. Lahiru. Flowchroma. https://github.com/ThejanW/FlowChroma, 2019.