Video Frame Interpolation - Report

Marek Nevole
CTU FIT
nevolmar@fit.cvut.cz

December 25, 2021

1 Introduction

The task of this semester assignment was to research, implement and evaluate neural network models that are capable of video frame interpolation which results in frame rate upscaling. Link to repository: `https://gitlab.fit.cvut.cz/nevolmar/mvi-sp`.

The problem of video frame interpolation is a classic problem in image and video processing. There are many methods that achieve the wanted results. Popular approaches are based either on finding optical flow for perfect motion description or using kernel based methods to keep structural alignment of the frames. However, in recent years, with the rise of deep learning, in popularity, many new approaches have been proposed. The most suited neural networks for this problem are convolutional neural networks and generative adversarial networks.

1.1 CNN model

Convolutional neural networks are very popular choice when it comes to image and video processing. These networks are based on convolutional layers, which apply filters to input to create feature maps.

I used convolutional network based on U-Net architecture[2]. U-Net consists of 2 basic blocks. Pooling (downscaling) block and Upscaling (concatenate) block. Pooling block usually consists of 2 convolution layers followed by pooling layer. As activation functions I used ReLU and after each convolution there is batch normalization layer. Upscaling block is built from Upscaling layer that concatenates inputs from last layer of previous block and last layer of previous block in same level. After that there are 2 convolutional layers with ReLU as activation followed by batch normalization layers. Used output layer is convolutional layer of 3 filters of size 1 with sigmoid as activation. The shape of input I used was 144, 256, 3 as height, width, number of channels.

1.2 GAN model

GANs consist of 2 neural networks, generator and discriminator. Generator generates desired output usually from high dimensional random noise data and discriminator classifies whether the output is fake from generator or real from training samples. These 2 networks are trained simultaneously in basicly zero sum game (ones gain is others loss).[1]

For video frame interpolation GAN is often made from 2 convolutional neural networks where generator is fully convolutional network and discriminator is again convolutional network used as binary classifier. These GANs are often called deep convolutional GANs (DCGAN).[3, 4]

Discriminator I used consists of 5 blocks built from 2 convolutional layers followed by LeakyReLU activation with $\alpha = 0.2$ and batch normalization. After last block there are 2 fully connected dense layers of 64 a 1 units. Output activation used was sigmoid.

For generator I used modified and slightly smaller network used earlier as just CNN.

2 Data

One training sample is created from 3 consecutive images, where first and last image is used to generate image that should be as close as possible to middle image. $x_i = (first, last)$, $y_i = middle$. All individual frames were downscaled to 144p resolution using Lanczos method for faster training. The idea was to use 2 hours of footage to train the networks. That conluded in to about 180 000 training samples. Training for just one epoch of used CNN would take about 200 hours on my computer CPU which I could not afford. After moving all code to google drive and using colab I ran into some issues with memory and my data generators, that after many hours I did not resolve, thus I had to reduce samples that they all could fit in memory at the same time. Final number of samples used for training and validation was around 1800. Images were normalized to [0, 1] for CNN and [-1, 1] for DCGAN.

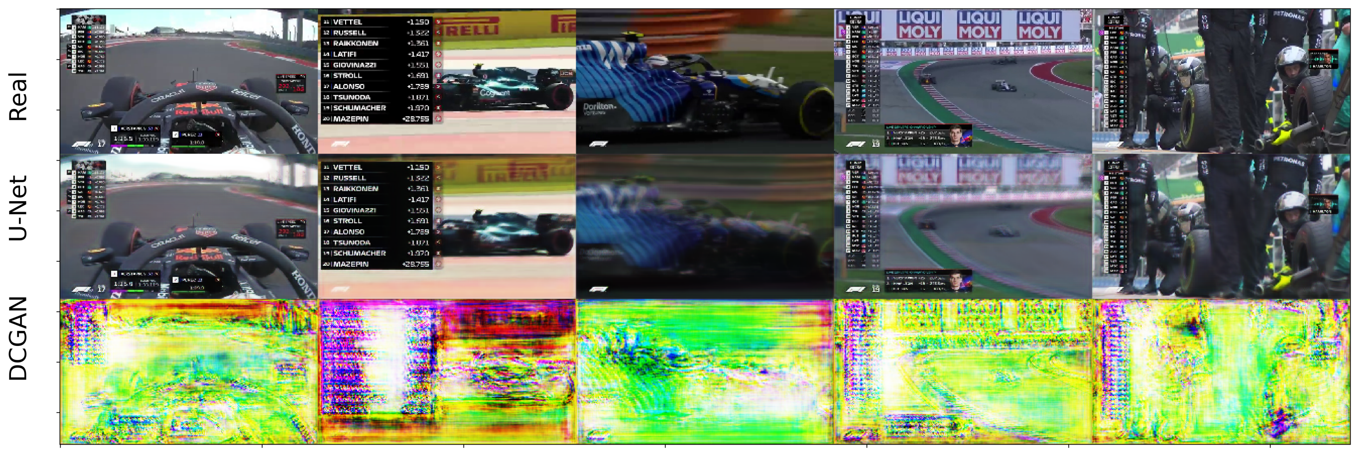

Figure 1: Generated frames of CNN and GAN models in comparison to real frame.

3 Results

To train CNN model Adam optimizer was used with $lr = 0.0001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e{-}07$. Mean squared error was used to calculate loss and accuracy as a metric. Model converged fairly quickly just after few epochs. Final validation loss was 0.0065.

Training of GANs is way harder. As it is basicly about fine tuning of the hyperparameters and just slight change leads to unstable training and divergence. Unfortunately I was unable to train the network to produce any sensible outputs. After every change the generator seemed to be overfitted just after first epoch and the results just got worse with each epoch. Here I list all changes I found and tried that were supposed to help in GAN training:

- Different learning rates for generator and discriminator. Generator had same or slighly lower learning rate than discriminator.
- Replaced all ReLUs with LeakyReLUs.
- Replaced MaxPooling with AveragePooling layers.
- Tried using glorot normal and uniform initiliazers of weights.
- Changed normalization of training samples to [-1,1] and output activation of generator to tanh instead of sigmoid.
- Swapped labels of real and fake images.
- Added small random noise to labels from uniform distr.
- Trained discriminator on fake and real images separately.

The predicted images in comparison to real ones can be seen in figure 1.

4 Conclusion

CNN model based on U-Net architecture seems to generate quite accurate images. Training was easy, quick and straight forward. There is not much to change about this architecture. One could try to add more pooling and upscaling blocks if equipped with enough computational power. Another possible improvment of this NN would be loss function. After quick search SSIM or Huber loss could produce better results.

For GAN model there is a lot to improve as my training of this network failed. To my limited knowledge I find GAN to be more powerful, but way harded to train. If I had more time I would try to implement Wasserstein GAN (WGAN) as its supposedly more stable and faster in training. WGANs replace discriminator classifiers with critics.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[3] Quang Tran. Efficient video frame interpolation using generative adversarial networks. *Applied Sciences*, 10, 09 2020.

[4] T. Xiao. Frame rate upscaling with deep neural networks. 2016.