Mateusz Żak

Bartek Podgórski

**1.Introduction**

**Problem description:**

Video processing is one of the biggest problems in the world of videography. We

decided to try to solve the problem of video quality by interpolating video and creating an intermediate frame from two adjacent video frames. The collections that we decided to use in our project are Vimeo90k from the website (http://toflow.csail.mit.edu/), which is a large video collection consisting of about 90k, highly scalable, high quality clips. It is already properly prepared and divided into a set for training (17GB) and a high resolution test set (16GB) to objectively assess the work of our algorithm. The data set (about 73k elements) are in the form of a triplet which consists of 3-frame sequences with a fixed resolution of 448 x 256, extracted from 15K selected video clips.

**Available solutions:**

There exist many solutions to the proposed problem. The traditional approach is to use optical flow to establish the motion between frames and generate the intermediate frame at any given time between the frames. This method however has a hard time with generating those parts of the frames where an object is occluded. More recently there have been numerous approaches using deep learning. One of the solutions proposed by Niklaus et al.1 is to generate 2D convolution kernels to calculate the value of a pixel in the intermediate frame based on the pixels around the pixels in the original frame. This approach however is computationally expensive as it is required to generate a different kernel for every pixel of the frame. Another approach proposed by Bao et al.2 addresses the occlusion problem by exploring the depth information of the frames. It achieved the best scores on UCFI101, Vimeo90k, Middlebury and HD data sets. Despite this results when the depth maps are poorly estimated the method generates blurred images and less clear boundaries.

**Our solution:**

We are going to generate an intermediate frame based on the two adjacent frames. To do that we will employ a deep convolutional neural network. The network is going to directly generate a frame at the half time between the two frames. Difference between our solution and another is that we train our program to get something like rules to generate intermediate frames , but for example in optical flow frames are generated based on the motion information of objects in video and in this situation we have a problem with big and quick elements.To train our network we will use the Vimeo90k data set, removing every second frame to be used as a target for our network. To test our model we are going to use videos from the same data set. In order to assess the quality of produced frames in the testing phase we will use the peak signal-to-noise ratio and structural similarity index.

1 https://arxiv.org/pdf/1708.01692v1.pdf

2 <https://arxiv.org/pdf/1904.00830v1.pdf>

**Metrics:**

Peak signal-to-noise ratio:

Given a monochrome mˣn image I and its approximation K MSE is defined as:



The PSNR is defined as:

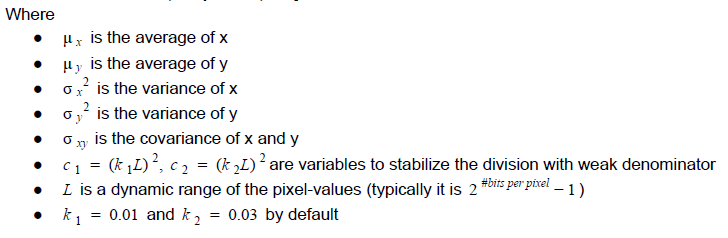


where MAX I is the maximum possible pixel value of the image.

**Structural similarity index:**

Given two images x and y:





**2.Evaluation part**

To interpolate the intermediate frames we used a deep convolutional neural network similar to U-Net. The network consists of two parts, the first part being the downsampling part and the second being the upsampling part. Both parts were composed of “packs” of convolutional layers. These packs were structured in the following way: the last layer of downsampling packs is a strided convolutional layer and the first layer of the upsampling packs is an upsampling layer. Preceding the strided convolution and following the upsampling layer were “flat” convolutional layers that didn’t change the size of the image. We also employed skip connections that connect output and input layers of packs with the same number of channels. Each layer is followed by a ReLU layer.

We had some problems with transporting our data to google drive, because we trained our neural network in google colab. At the beginning

we downloaded films from vimeo90 in mp4 format. After this we sent it to our google drive and there cut them into frames and divided into three elementary groups. We did it in this way, because when we sent thousands of frames from our computer it was very uneffective.

We had to use google colab, because google cloud computing didn’t grant us permissions to use high quality graphic cards which are needed in our project. We decided to use google colab because you can get here high quality graphic cards like P-100. When we have it we can train our neural network effectively by average 5 epochs.

Finally we decided to show our results visually with the package OpenCv. We made a script which allowed us in two windows side by side at the same time to show the original video and our video. In this way we can compare them without big effort.

The following changes were the most influential when it comes to the achieved results:

* skip connections - after adding the skip connections the resulting images became visibly sharper and retained the edges much better.
* he initialization - a special type of weight initialization for convolutional layers that allows really deep networks to converge.
* combination loss function - we used a combination of multi-scale structural similarity index and L1 which improved the sharpness of the images.

Achieved scores:

|  |  |
| --- | --- |
| SSIM | PSNR |
| 0.881 | 99.920 |