Bilinear Interpolation Upsampling on CUDA

Project Phase-I

Bilgin AKSOY

Informatics Enstitute

12^{th} Nov 2017

Problem

- A deep network for object segmentation.
- This network has 5 two-by-two max-pooling layer, each of which down-samples its input by factor two.

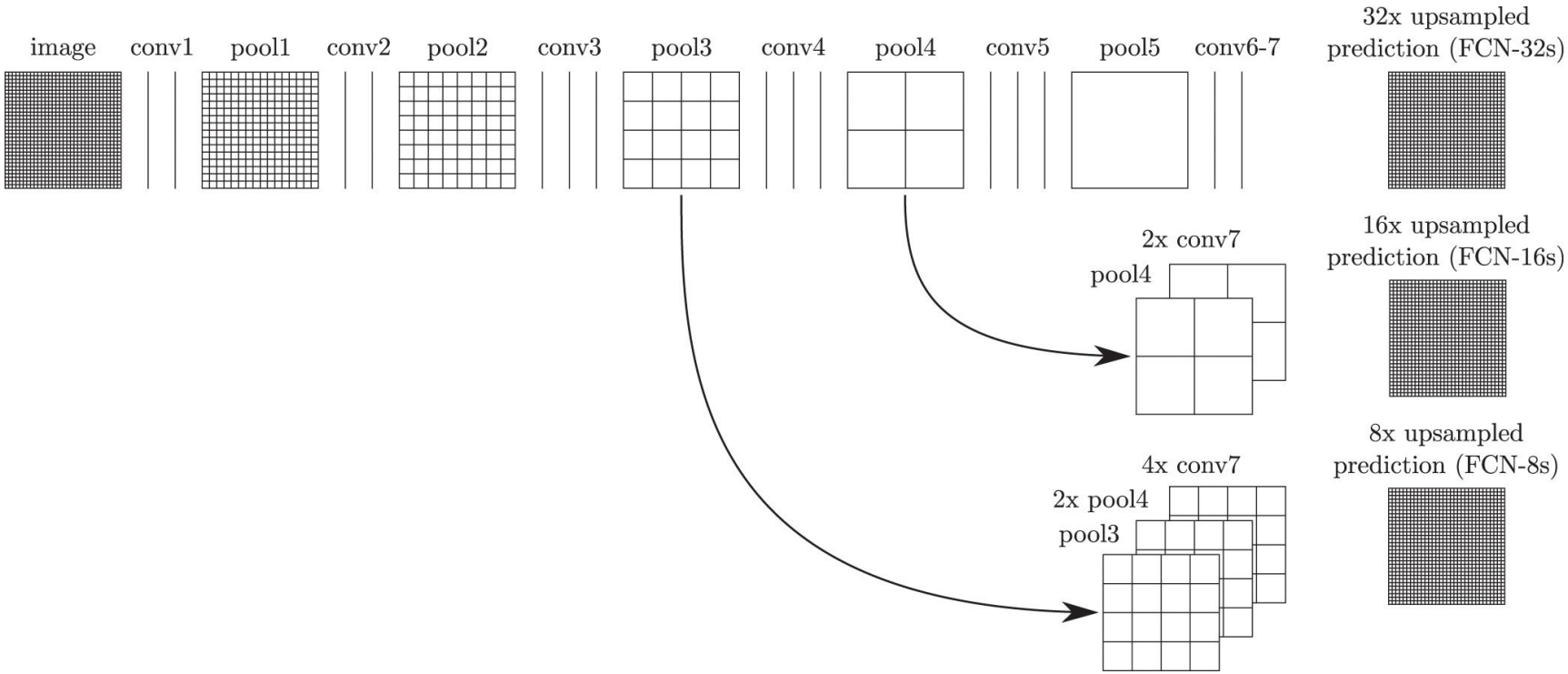

Figure 1 : Network Architecture-Long et al.[1]

Problem (cont'd)

- After pool3, the activation size have been reduced by factor $8 = 2^3$.
- After pool4, the activation size have been reduced by factor $16 = 2^4$.
- After pool5, the activation size have been reduced by factor $32 = 2^5$.

Problem (cont'd)

- Sum three activations and calculate the error(loss) and back-propagate the error at the end of the final layer.

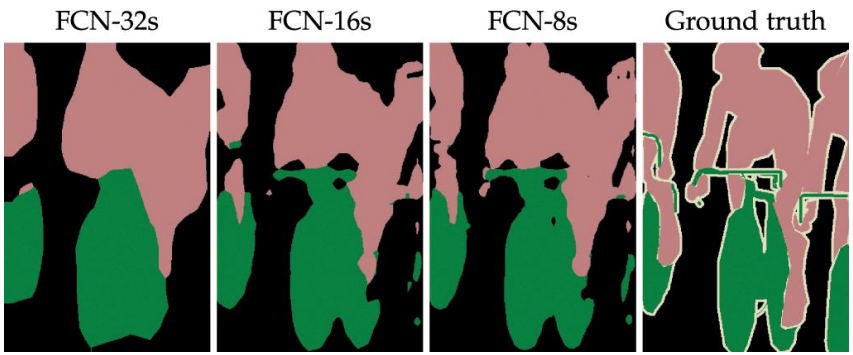

Figure 2 : Activations of Pooling Layers-Long et al.[1]

Problem (cont'd)

- I will implement a bilinear interpolation upsampling kernel and my kernel will not only upsample the activations but also produce the final result by summing the activations.

Literature Survey

- **Bilinear Interpolation:** Bilinear interpolation is used to know values at random position from the weighted average of the four closest pixels to the specified input coordinates, and assigns that value to the output coordinates.[2] (1)

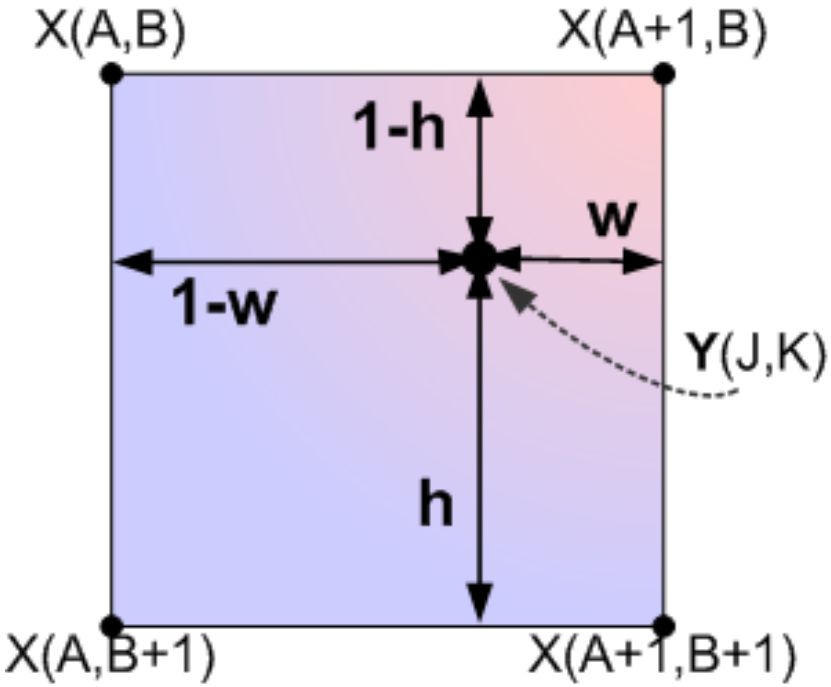

Figure 3 : Image from `https://www.giassa.net/?page_id=240`[3]

$$Y(J,K) = (1-W)(1-H)X(A,B) + (W)(1-H)X(A+1,B) + (1-W)(H)X(A,B+1) + (W)(H)X(A+1,B+1) \quad (1)$$

- dfsdfs[4]
- Akgün and Gevrekci have been proposed "a massively multi-threaded implementation of super-resolution image formation on the NVIDIA CUDA architecture." [5] Super-resolution reconstruction begins with an initial estimate image which is typically chosen as a bilinearly upscaled version of the original low resolution frame

OpenCV CPU

```
void  cv::resize (
InputArray src, OutputArray dst, Size dsize,
    ↪ double fx=0, double fy=0, int interpolation
    ↪ =INTER_LINEAR
)
```

OpenCV GPU

```
void cv::cuda::resize (
InputArray src, OutputArray dst, Size dsize,
    ↪ double fx = 0, double fy = 0, int
    ↪ interpolation = INTER_LINEAR, Stream &
    ↪ stream=Stream::Null()
)
```

Intel Integrated Performance Primitives(CPU Optimizations)

```
IppStatus ippiResizeLinear_<mod>(const Ipp<
    ↪ datatype>* pSrc, Ipp32s srcStep, Ipp<
    ↪ datatype>* pDst, Ipp32s dstStep, IppiPoint
    ↪ dstOffset, IppiSize dstSize, IppiBorderType
    ↪  border, const Ipp<datatype>* pBorderValue,
    ↪  const IppiResizeSpec_32f* pSpec, Ipp8u*
    ↪ pBuffer)
```

- Just for Intel CPU. Not free.(249$for Academic Licence)
- OpenCV 3.x has native support for IPP-ICV library which is a special subset of Intel IPP functions for image processing and computer vision.

Linux Command Line Tool - Imagemagick (CPU Only, OPENCL For Some Algorithms)

```
$ convert input.jpg output jpg -interpolate
    ↪ bilinear
```

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[2] S. Fadnavis, "Image interpolation techniques in digital image processing: An overview," *International Journal of Engineering Research and Applications*, vol. 4, no. 10, pp. 70–73, 2014.

[3] "Bilinear interpolation." `https://www.giassa.net/?page_id=240`.

[4] W. QINGSHUANG, W. QIANG, and C. XIANGFU, "Parallel bilinear spatial interpolation algorithm based on gpgpu.," *Journal of Theoretical & Applied Information Technology*, vol. 48, no. 3, 2013.

[5] T. Akgün and M. Gevrekci, "Accelerating super-resolution reconstruction using gpu by cuda," in *Advances in Computational Science, Engineering and Information Technology*, pp. 47–58, Springer, 2013.