Rendering Natural Camera Bokeh Effect

Using the PyNET Convolutional Neural Network

Harsh Poonia
2nd Year UG, CSE

Report for the project under Winter in Data Science,
Analytics Club, IIT Bombay
Mentored by Vinayak Srivastava

January 2023

Rendering Natural Camera Bokeh Effect

Using the PyNET Convolutional Neiral Network

Harsh Poonia

Abstract

Bokeh is an important artistic effect used to highlight the main object of interest on the photo by blurring all out-of-focus areas. While DSLR and system camera lenses can render this effect naturally, mobile cameras are unable to produce shallow depth-of-field photos due to a very small aperture diameter of their optics. Unlike the current solutions simulating bokeh by applying Gaussian blur to image background, in this implementation of the paper conceptualised by the Andrey Ignatov and team at ETH Zurich Vision Lab, we try to learn a realistic shallow focus technique directly from the photos produced by DSLR cameras.

The architecture of the model has been taken from `github.com/aiff22/PyNET-bokeh`, but the code served only to guide us in the journey, for it was written in Tensorflow 1.x. The model backend has been written in Tensorflow 2.x and Keras, and the data input pipeline was created from scratch in tensorflow. A subset, more than one fifth of the original *Everything is Better with Bokeh!* dataset was used for training, for some hours on a cloud based NVIDIA Tesla P100 GPU.

While the model is able to reasonably predict and apply the bokeh effect, limitations in training resources did not allow the model to learn colors in the dataset, so the output spectrum has little color contrast.

The code for the repository can be found at my github below.

harpoonix

PyNET Architecture

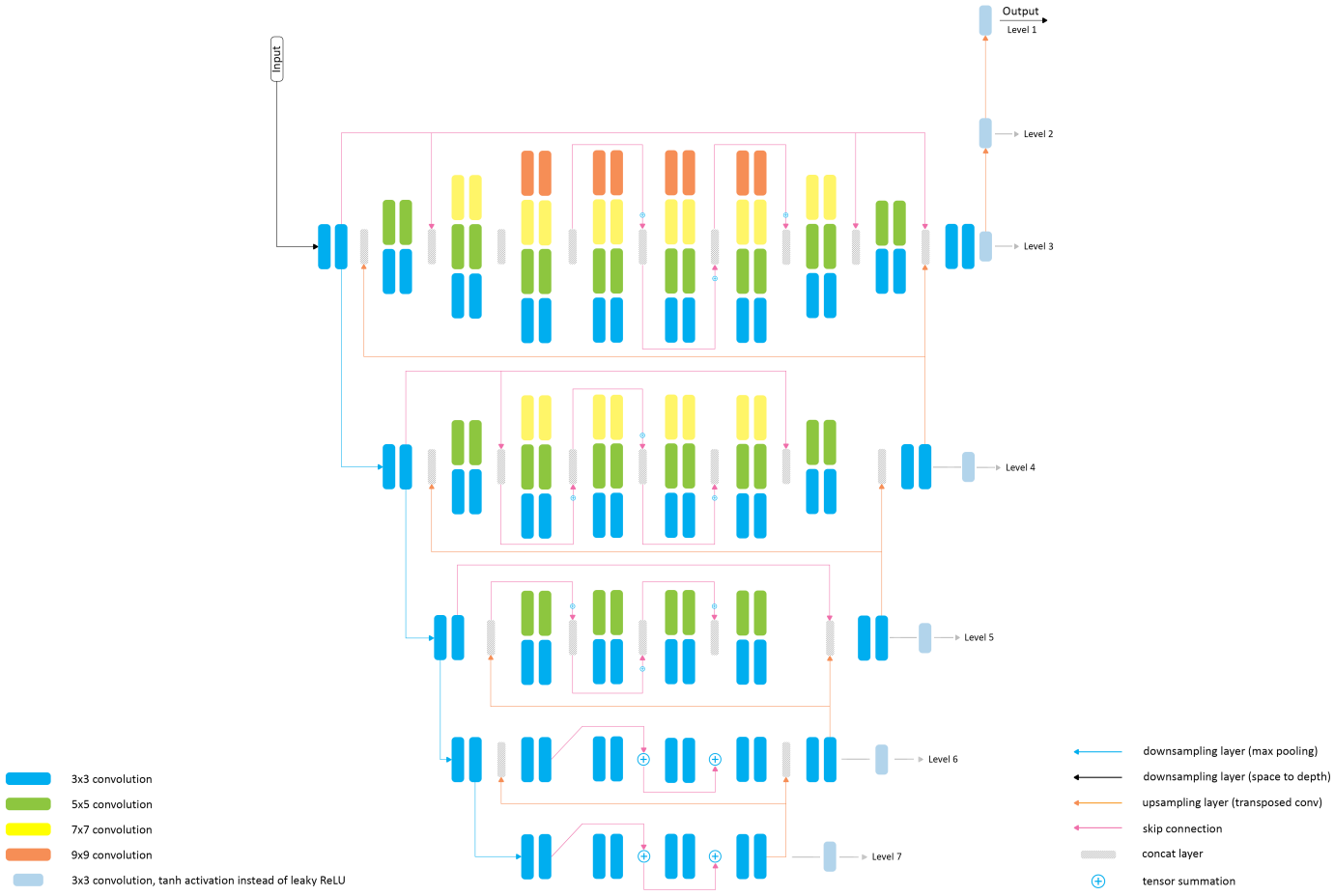

Figure 1: The PyNET CNN

The model has an inverted pyramidal shape and is processing the images at seven different scales. The model is trained from the bottom up - starting from level 7 and moving our way upwards. This allows us to obtain good recontruction results at lower layers that work with low resolution images, and each higher level gets upsclaced high quality feautres from lower levels.
At each convolutional layer, we perform instance normalisation on each of the channels to prevent training from diverging. Each color channel is normalised across its height and width while training. Skip connections and upsampling through transpose convolutional layers allow better learning of low level and high resolution features.

Data Input Pipeline