**Image Super Resolution**

This file documents the instructions on how to use the various files related to this project “Super Resolution”.

**Overview :**

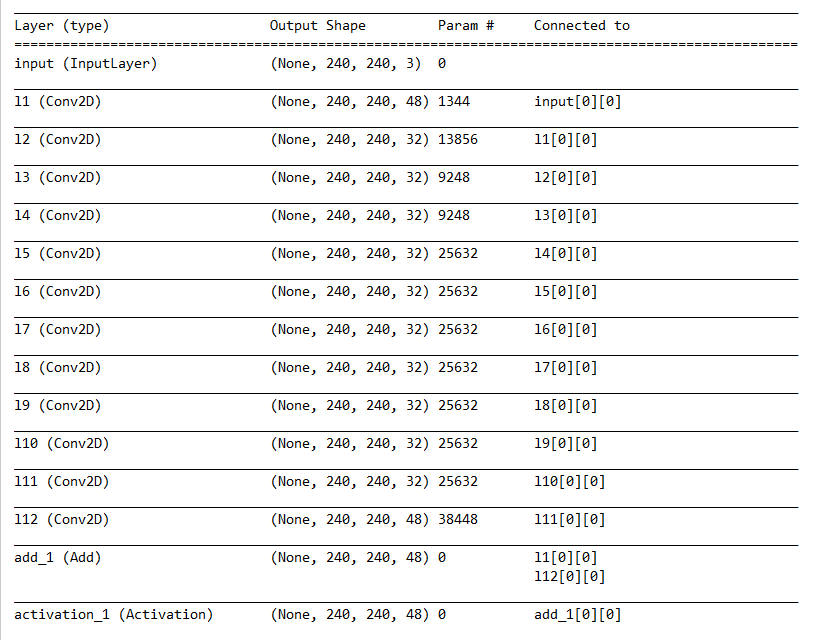
* We do single image super resolution in this project.
* We follow the GL-Net architecture from the paper “Real-Time Deep Video SpaTial Resolution UpConversion SysTem (STRUCT++ Demo)” to do so.
* We modify GL-Net architecture by adding some more convolutional layers after the final deconvolution layer in GL-Net, everything else is same.
* We train and test on images in patches of specific size (in our case it is 240 x 240), which allows us to use images of variable size during training and testing. (NOTE : The patch size should be same/constant for the model, i.e. patch size is what characterises the model).
* No python file takes runtime or command line arguments. You have to open the file and edit at the necessary places if ever required. But, I also tried to keep this task easy by simplifying and organising my code as much as I could.

**Files used in this Project :**

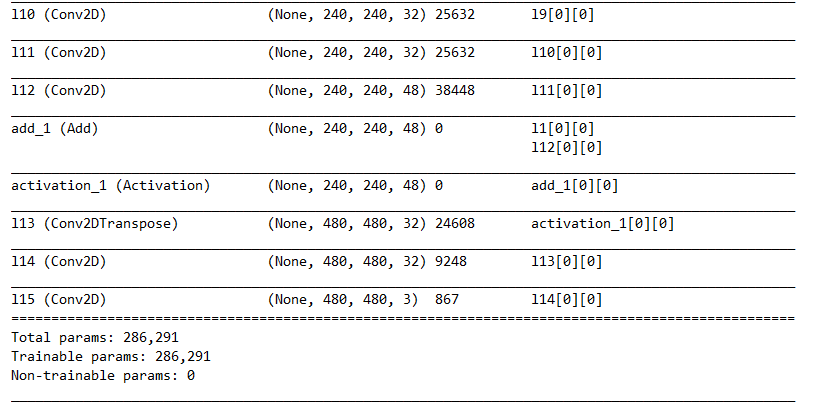
* vid2frame.py
* download.py
* downsample.py
* preprocess.py
* sr\_main\_keras.py
* sr\_gui\_browse.py

**Functions of various files :**

* **vid2frame.py** **:**
  + This file is used to generate frames from a given input video.
  + It takes the path to video and the path to save images as input.
  + It also takes the maximum number of frames to be generated and also the number of frames to skip between two consequent frames as input.
  + It also takes the extension in which you would like to save the generated frames.
* **download.py :**
* This file is used to download large files from google drive.
* It takes google file id (can be found in the shareable link).
* It also takes the path with filename and extension to save the downloaded file.
* There is also code to unzip files but is commented out since optional. It takes the path to save the unzipped file.
* **downsample.py :**
* We need High and Low resolution images for training a super resolution model. So to generate low resolution images from a given set of high resolution images this file can be used.
* This takes the path to input high resolution images and path to save the low resolution images.
* This also takes the factor (new size = what fraction of input image) by which to resize the HR images. (i.e. you input 0.5 to resize the image to half its original size)
* We can also resize images by specific size if needed. ( i.e. you specify the size of the images like (200, 100)).
* **preprocess.py :**
* This file is used to pre-process the image files to make them suitable for training.
* *Pre-process* : extracts overlapping patches of size 240 (i.e. p\_height and p\_width) from the Low resolution images and corresponding patches of size 480 (i.e. sf x p\_height and sf x p\_width) from the high resolution images. The overlapping is 25% and is in both x and y direction.
* We first pad the images with appropriate padding of a colour and then divide the images into overlapping patches.
* This file is able to handle images of various sizes to extract patches from them, but for a run the images should be of same size. i.e when you run the file all the input images should be of same size. When you run the file next time, the size of input images must be same but can be different from the previous run.
* It takes path to HR images, LR images and also paths to save the processed HR and LR patches.
* Patches from the LR images is 240 x 240 while the patches from the HR images is of 480 x 480 (i.e. double the size, since we are super resolving images by a factor of 2).
* **sr\_main\_keras.py :**
* This is the main file used for training, written in keras.
* Consists of class Super\_Resolution which contains all relevant methods for training, testing, prediction etc.
* The code is commented as required for better understanding of the user. The name of the methods are self-explanatory too.
* The model summary can be seen in the pictures below :



The architecture is split into two images since, the whole couldn’t fit into one.



* **sr\_gui \_browse.py :**
  + This is the GUI for the Super Resolution model.
  + This version of GUI uses browsing to select image files.
  + All you need to do is run this file, a GUI window will open which will guide you through further instructions.
  + It consists of two important functions, show\_sr and show\_lr which are to show the Super resolved and the original low resolution images respectively.
  + Everything in the code sequentially, as shown/displayed in the GUI. Hence, it is not too difficult to understand the code. It gets much easier to understand if the GUI is open and is matched with the corresponding code written.
* **WEIGHTS folder :**
* The folder consists of weights as the model is trained on different datasets, with previous weights as initializer.
* The weights in the highest numbered folder are latest.
* The .json file contains the model.

**NOTE :**

* To use the GUI simply run the sr\_gui\_browse.py file.
* Our code is invariant to size of images, which is very useful esp. at the time of prediction/inference. This is so because we divide the image into overlapping patches (240 x 240), super resolve the patches and then again combine the patches to an image.
* The reason to take overlapping patches is to avoid distortion at the edges.