Why UNET?

* UNET is really versatile and can be used for any reasonable image masking task.
* Can be easily scaled to have multiple classes.
* Smaller model weight size (for 256x256 U-NET - ca. 118MB).
* No dense layer, so images of different sizes can be used as input

DOWNSAMPLING:

CONVOLUTIONS + MAXPOOLING:

* Reduced resolution, increased depth.
* The idea behind max pooling is to retain only the important features (max valued pixels) from each region and throw away the information which is not important.
* Note that the size of the filter and strides are two important hyper-parameters in the max pooling operation.

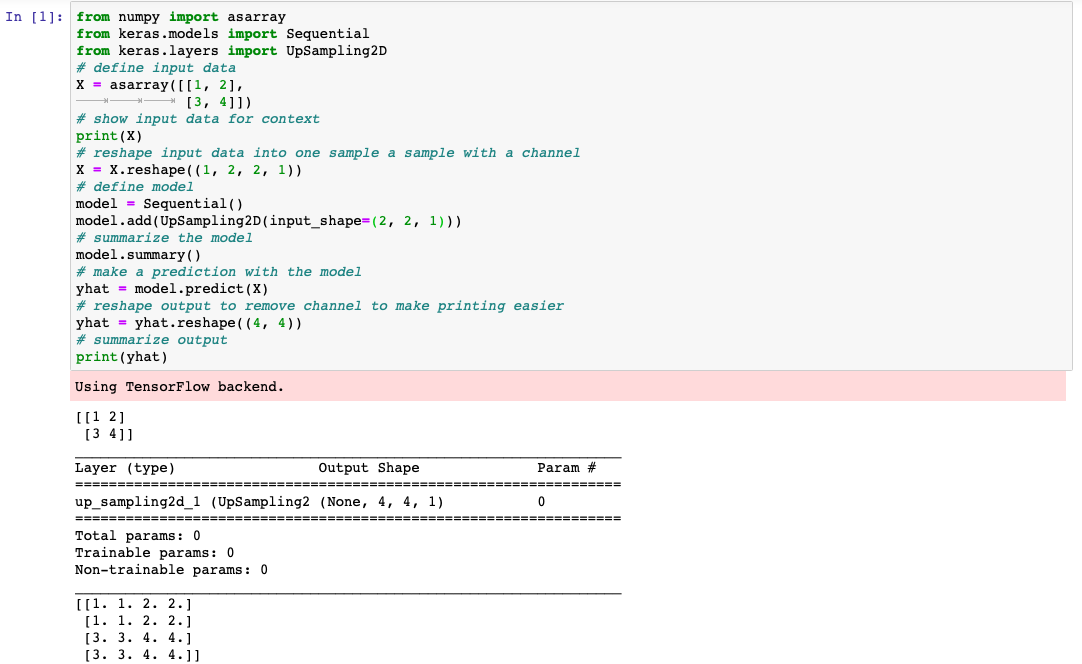
By down sampling, the model better understands “WHAT” is present in the image, but it loses the information of “WHERE” it is present.

UPSAMPLING:

Why do we need upsampling?

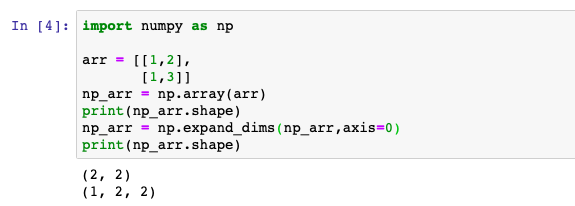
* As stated previously, the output of shadow detection AI model is not just a class label or some bounding box parameters.
* In-fact the output is a complete high resolution image in which all the pixels are classified as shadow and non-shadow.
* Thus if we use a regular convolutional network with pooling layers and dense layers, we will lose the “WHERE” information and only retain the “WHAT” information which is not what we want.
* In case of our problem we need both “WHAT” as well as “WHERE” information.
* Hence there is a need to up sample the image, i.e. convert a low resolution image to a high resolution image to recover the “WHERE” information.

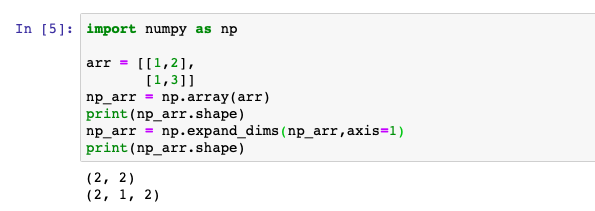
In the literature, there are many techniques to up sample an image. Some of them are bi-linear interpolation, cubic interpolation, nearest neighbor interpolation, transposed convolution, etc.

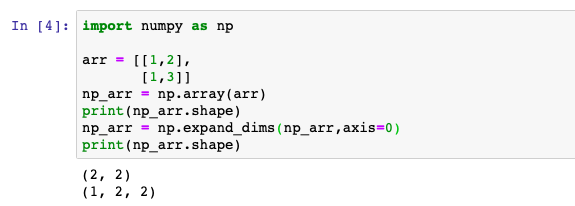
Our current model uses nearest neighbour interpolation, which can be understood by the following code snippet,

Another implementation of U-NET architecture using Transposed Convolutions is also attached in the current folder for future training and experimentation. It can be found here.

np.expand\_dims:

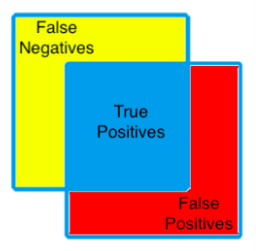
Another NumPy function used in the code is np.expand\_dims whose working can be understood by the following three code snippets:





Metric

Dice Coefficient:

* Dice similarity coefficient (a.k.a Dice score) is used to to quantify how closely our generated mask matched the training dataset’s hand annotated ground truth mask.
* It is used for calculating pixel-level image segmentation performance.
* The area of overlap between human and AI results is the blue square. This is the region where an image segmentationImage algorithm identifies pixels that exactly match the annotated ground truth segmentation. These pixels are known as true positives (TP).
* The pixels in the red region were erroneously segmented by the CNN and are known as false positives (FP).
* The pixels in the yellow region should have been segmented by the CNN but were missed. These missed pixels are known as false negatives (FN).