SHADOW DETECTION USING NEURAL NETWORKS

-HIMANSHU AGGARWAL

PROBLEM STATEMENT:

OBJECT DETECTION

Object Recognition: In a given image you have to detect all objects (a restricted class of objects depend on your dataset), Localized them with a bounding box and label that bounding box with a label. In below image you will see a simple output of a state of the art object recognition.

Object Detection: it's like Object recognition but in this task you have only two class of object classification which means object bounding boxes and non-object bounding boxes. For example Car detection: you have to Detect all cars in a given image with their bounding boxes.

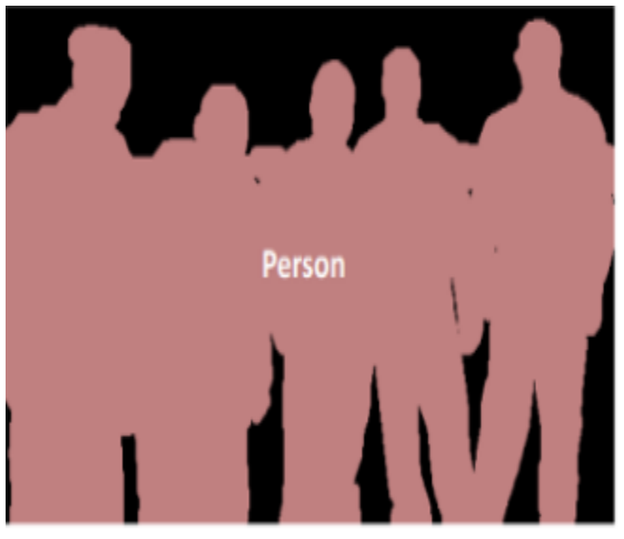
Object Segmentation: Like object recognition you will recognize all objects in an image but your output should show this object classifying pixels of the image.

IMAGE SEGMENTATION

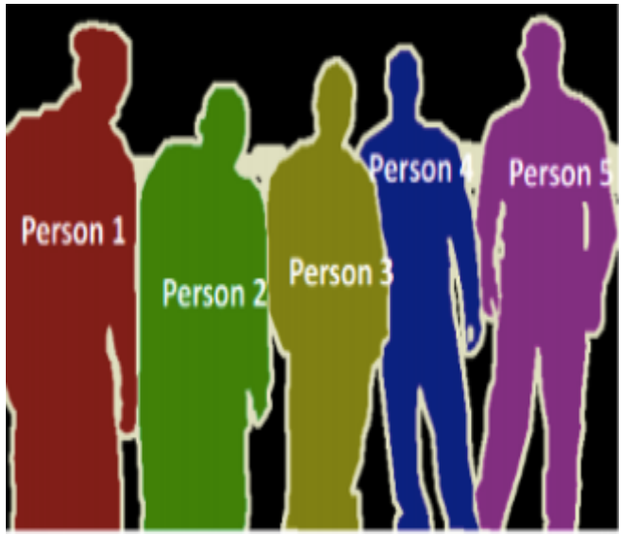
Image Segmentation: In image segmentation you will segment regions of the image. your output will not label segments and region of an image that consistent with each other should be in same segment. Extracting super pixels from an image is an example of this task or foreground-background segmentation.

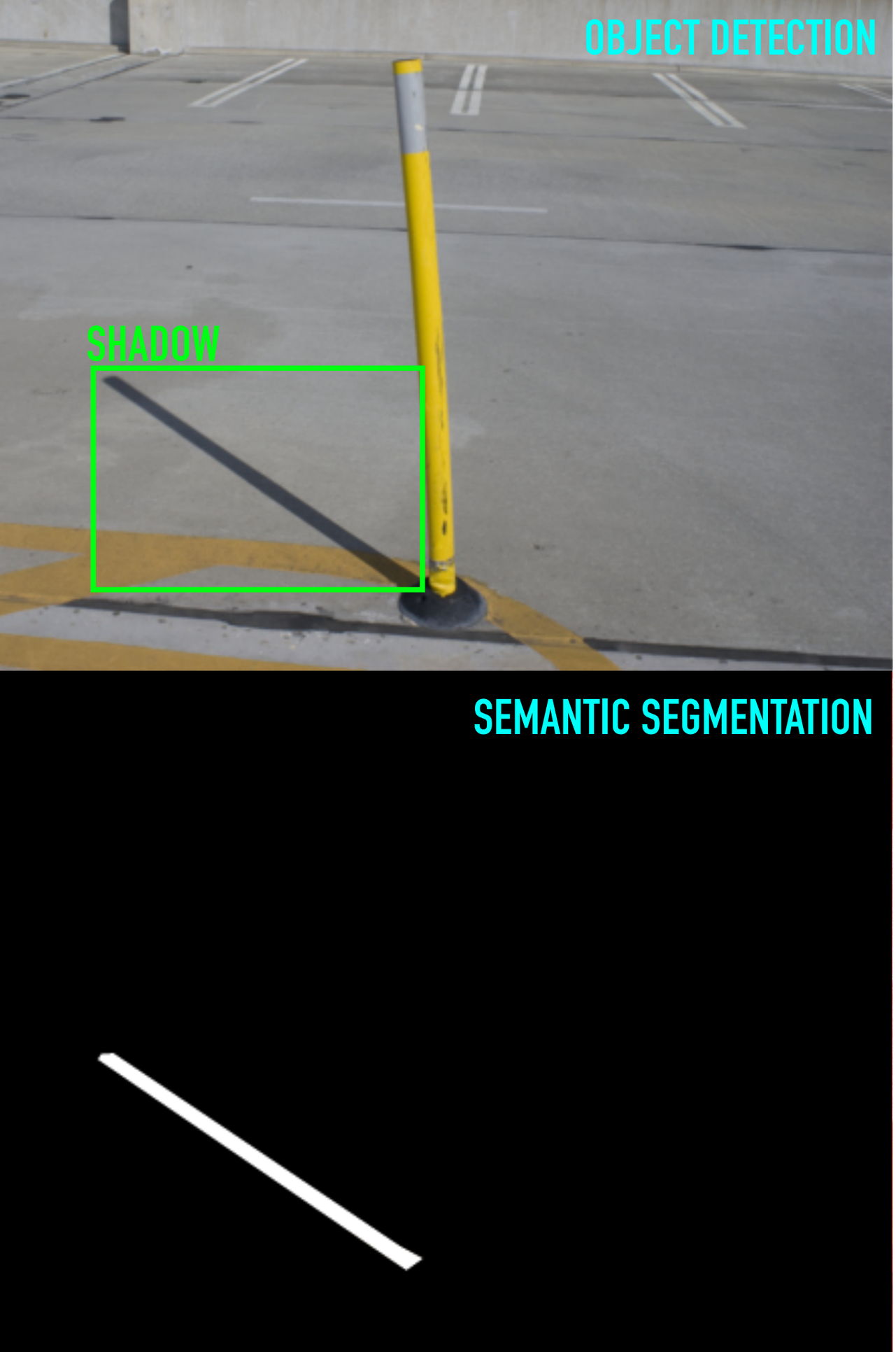
Semantic Segmentation: In semantic segmentation you have to label each pixel with a class of objects (Car, Person, Dog, ...) and non-objects (Water, Sky, Road, ...). I other words in Semantic Segmentation you will label each region of image.

INSTANCE SEGMENTATION



SEMANTIC SEGMENTATION





WHY OBJECT DETECTION WON'T WORK FOR OUR PROBLEM (SHADOW DETECTION)?

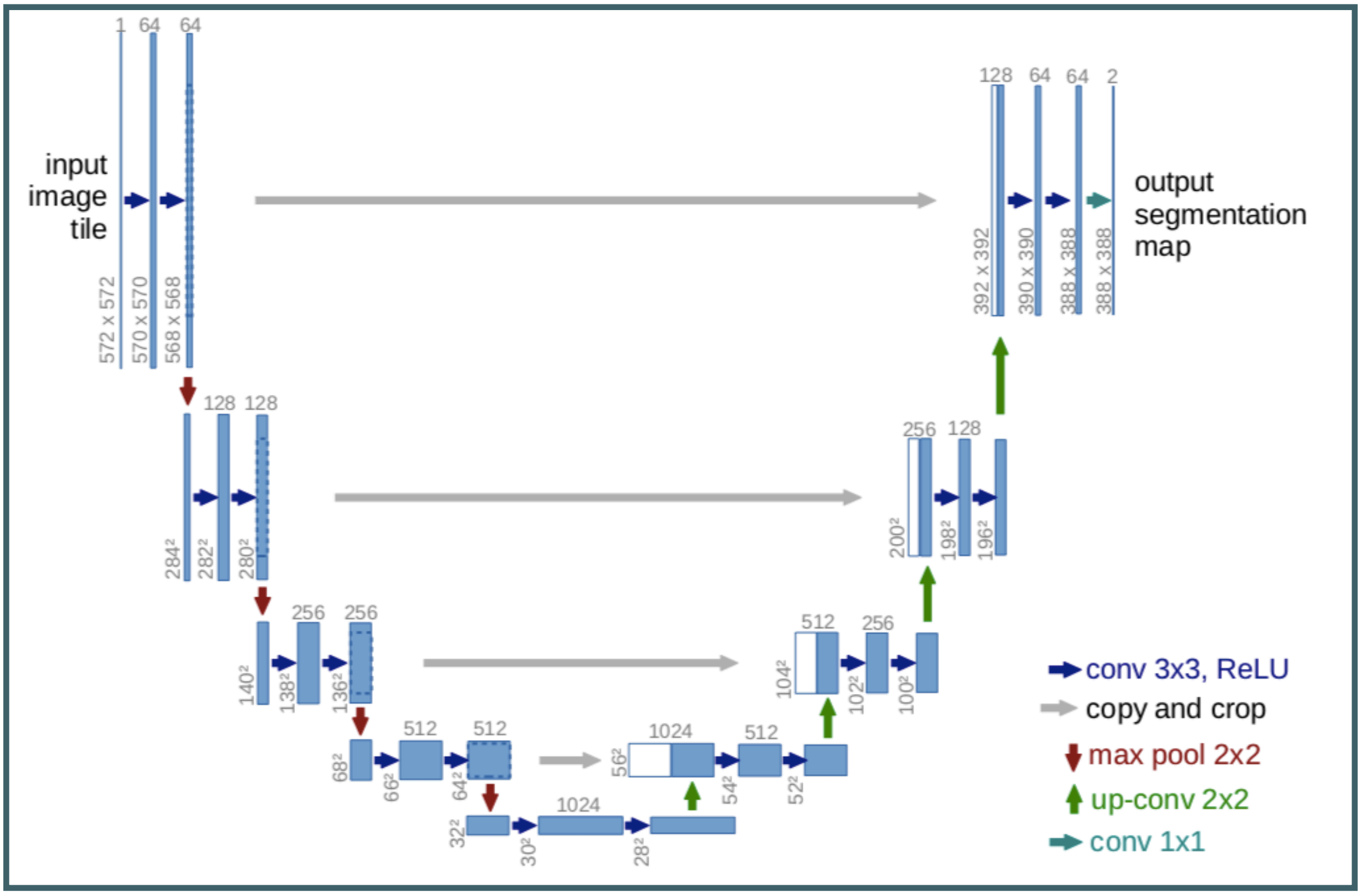
- Using Object Detection, we only get the set of bounding box coordinates. Bounding box doesn't tell anything about the shape of the detected shadow. This information is not sufficient for any further work to be done regarding the detected shadow (for eg: removal of shadow).
- We need pixel-level accuracy to distinguish between different classes (shadow and non-shadow).
- Each pixel on the input must correspond to a pixel in the exact same location on the output.

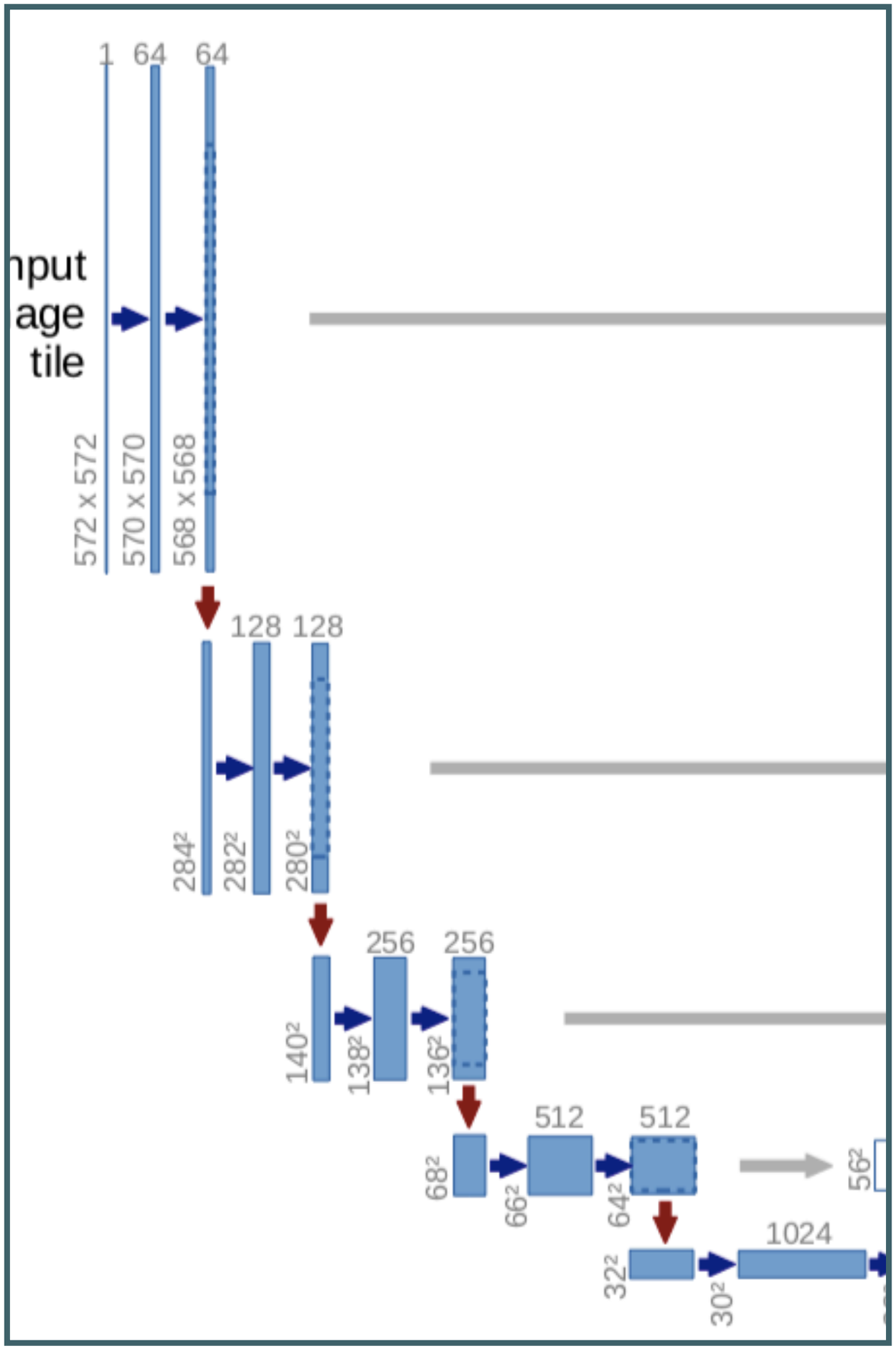
U-NET

WHY UNET?

- UNET is really versatile and can be used for any reasonable image masking task.
- Can be easily scaled to have multiple classes.
- This architecture is input image size agnostic since it does not contain fully connected layers.
- This also leads to smaller model weight size (for **512x512 U-NET - ca. 89mb**).
- No dense layer, so images of different sizes can be used as input (since the only parameters to learn on convolution layers are the kernel, and the size of the kernel is independent from input image' size).

U-NET ARCHITECTURE:



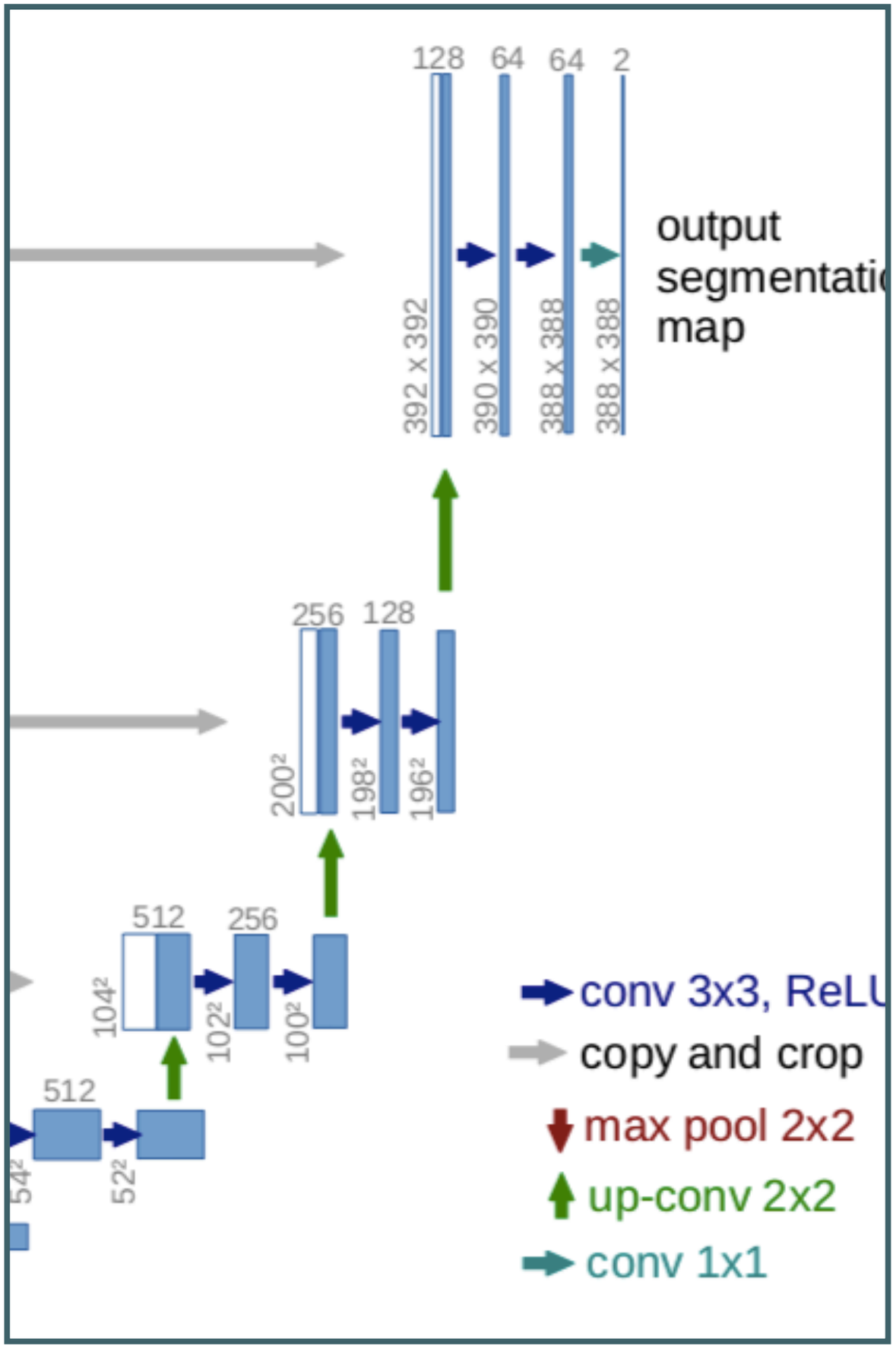


DOWNSAMPLING PATH:

- 4 Convolution blocks(2 Conv2D Layers each) each followed by a MaxPooling layer 2×2 with stride 2 for downsampling.
- 5th Convolution block without MaxPooling (connection to upsampling path)
- First Convolution block with 64 filters on each Conv2D Layer.
- Number of filters doubled with each consecutive convolution block.
- 3×3 Filters, with ReLU activation.
- Reduced resolution, increased depth(number of layers)

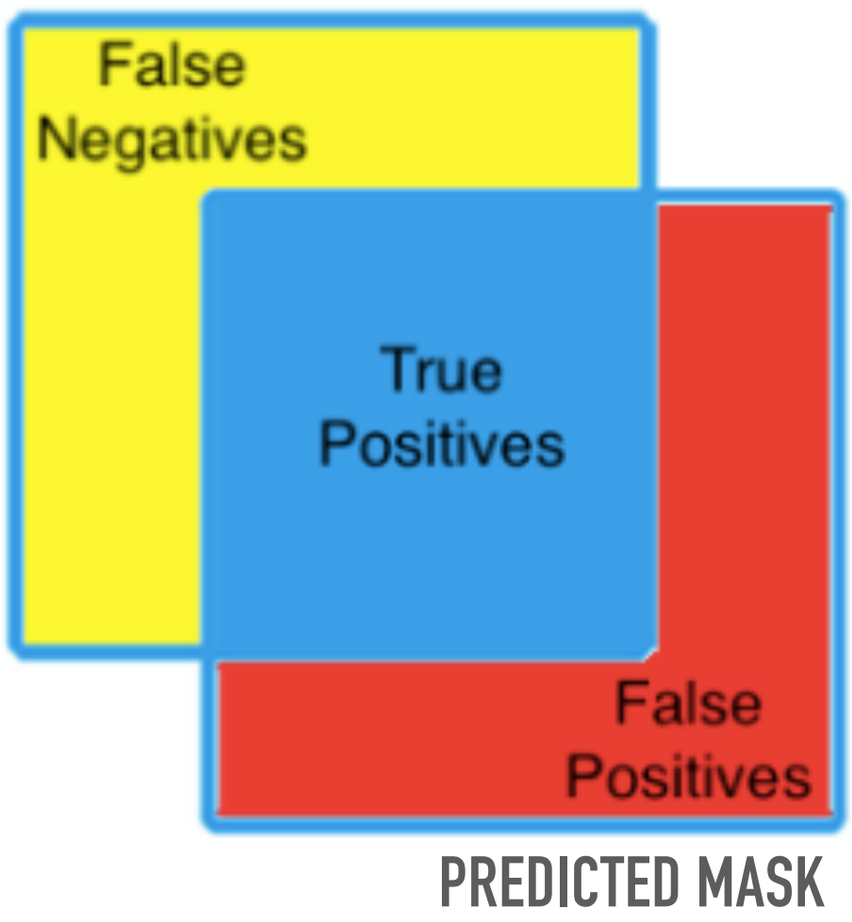
UPSAMPLING PATH:

- Symmetric to the downsampling path.
- 4 Convolution blocks(2 Conv2D Layers each) each followed by a UpSampling layer(Nearest neighbour algorithm is used for upsampling).
- Number of filters halved with each consecutive convolution block.
- 3×3 Filters, with ReLU activation.
- Increased resolution, reduced depth(number of layers)
- Feature maps from corresponding downsampling layers are concatenated for more precise localization.
- Final layer is a 1×1 Convolution, used to map each 64 component feature vector to the desired number of classes(in this case shadow and non-shadow).



METRIC:

(DICE COEFFICIENT)



$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

- **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 segmentation 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).**

RESULTS:

CHALLENGES:

- Less dataset available
- rgb vs grayscale

FURTHER PLANS:

- Use Transposed Convolutions instead of UpSampling, so that neural network can learn weights for upsampling.
- Use IOU, as a metric for measuring accuracy.
- Use advanced versions of U-NET, for the task in hand.