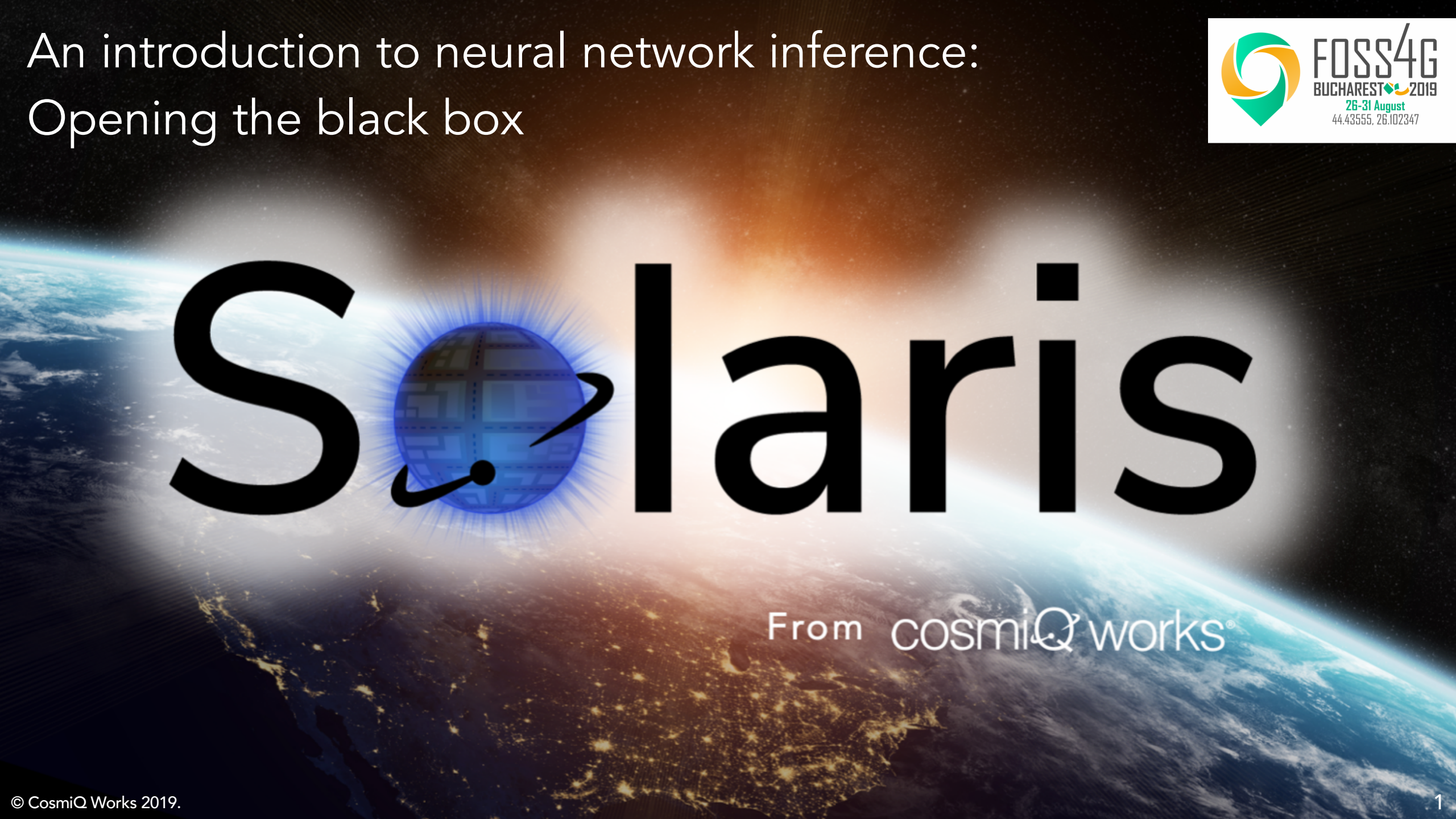
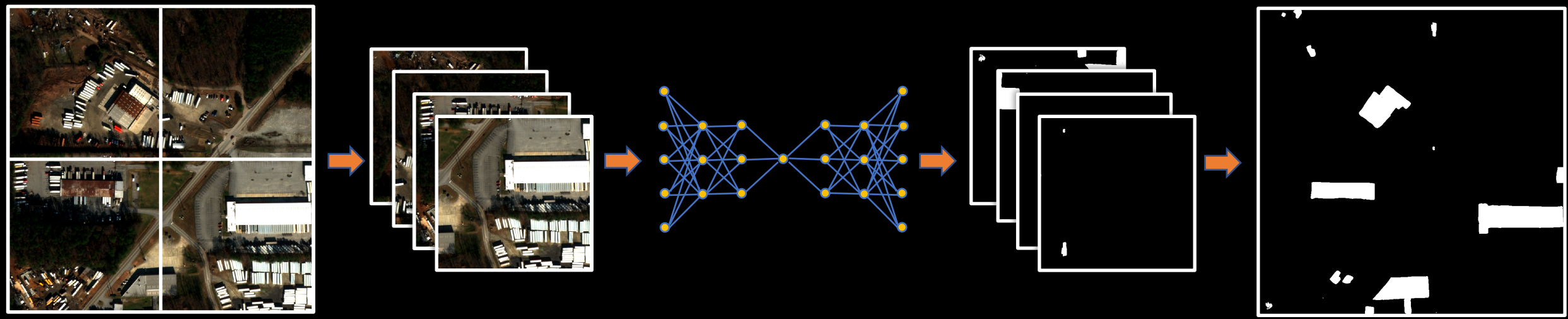
An introduction to neural network inference: Opening the black box



FOSS4G BUCHAREST 2019
26-31 August
44.43555, 26.102347

A schematic view of inference



What the neural net sees

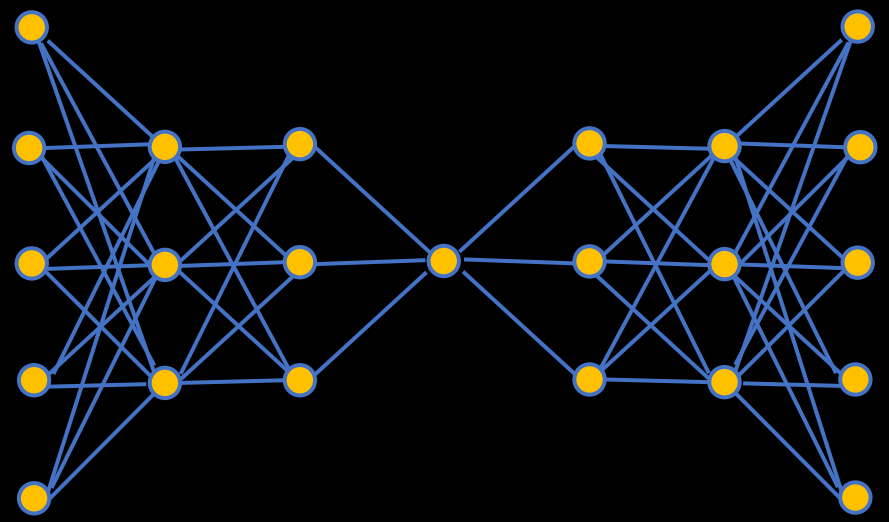
=

```
[[0.1,  0.2,  0, -0.1, …],
 [  0, 0.12, -1,  1.5, …],
 [Etc. Etc. Etc. Etc……]]
```

What actually *is* a neural net?

In this case…

A series of distortions, shape changes, and other mathematical operations on the input image to convert it into *p(building)* for each pixel.



= Mathematical operation
(usually a combination of multiplication, addition, non-linearity)

= Values pass between these sets of operations

Example: one “neuron” in the first layer of a NN

```
[[0.1,  0.2,  0, -0.1, …],
 [  0, 0.12, -1,  1.5, …],
 [Etc. Etc. Etc. Etc……]]
```

→

1st: × [1, 3, 0.25, …], “weights”

2nd: + [-0.3, 1, 0.25, …], “biases”

3rd: non-linearity

→

```
[  0, 1, 0.33,    0, …]
```

Output vector
(or array for convolutions)

Results from input neurons (generally) get concatenated on a new axis when they’re passed to the next “layer” of neurons.

What is the non-linear operation in the neuron?

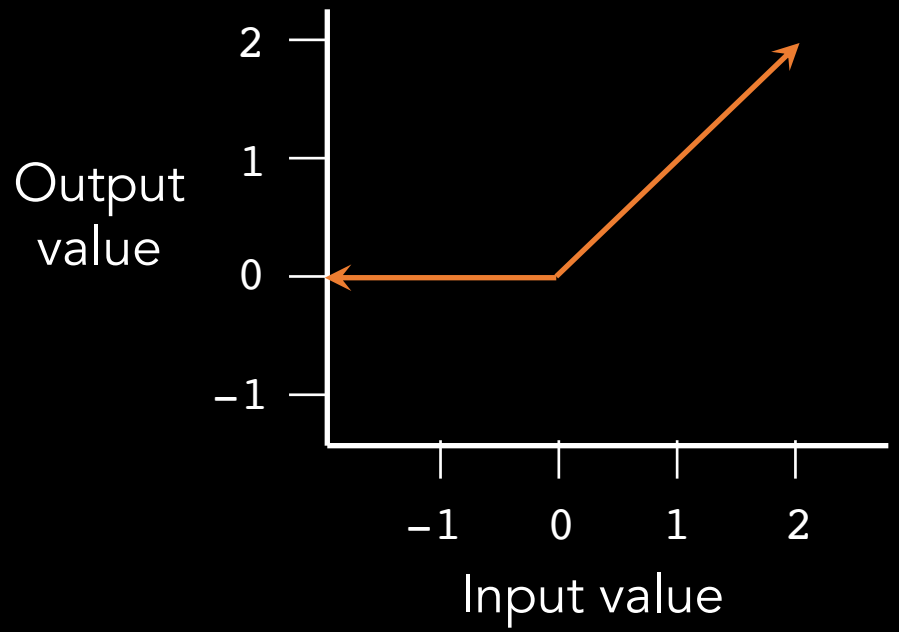
Most commonly Rectified Linear Unit (ReLU):

```
f(x) = max(0, x)
```

ReLU written as an if-then:

```
If x < 0: f(x) = 0
Else: f(x) = x
```

Graphical display of ReLU



ReLU example:

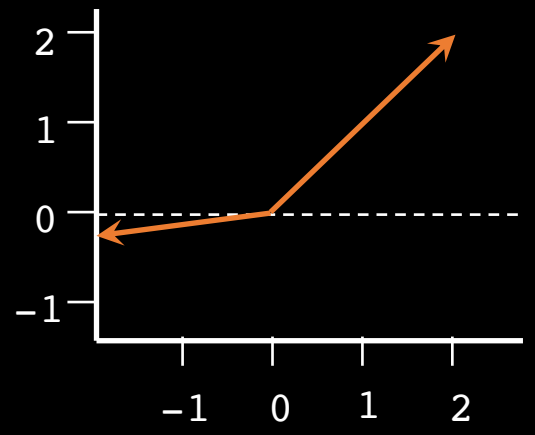
```
x: [-0.3, 1, 0.33, -2, …]

ReLU(x): [   0, 1, 0.33,  0, …]
```

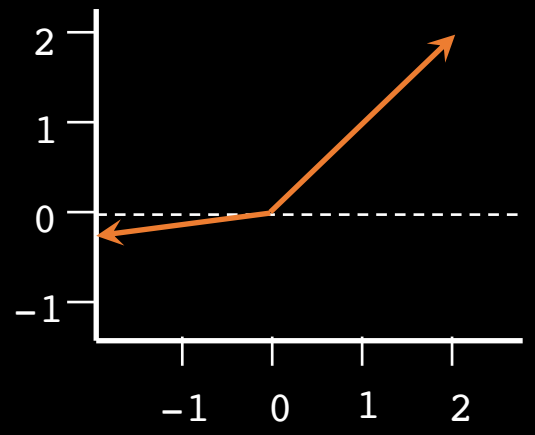
Other common non-linearities (AKA “activations”):

Leaky ReLU:

$f(x) = \max(\alpha x, x)$

α is usually in [0.01, 0.1]





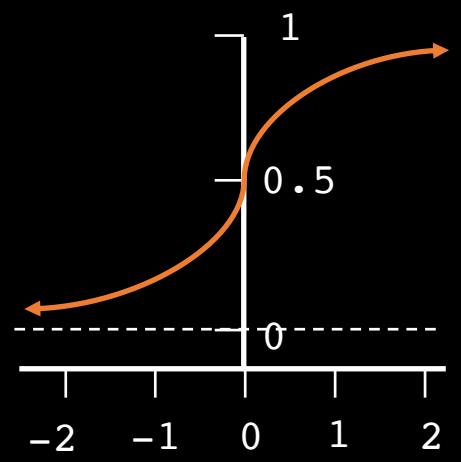
PReLU:

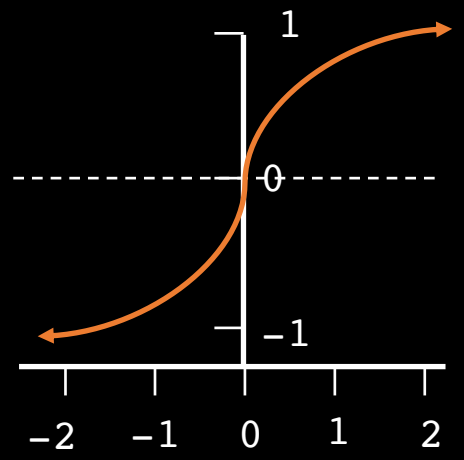
$f(x) = \max(\alpha x, x)$

model learns α

Sigmoid (logistic):

$$f(x) = \frac{1}{1+e^{-x}}$$





tanh:

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

What do most segmentation networks look like?

Convolutional neural networks

Layers composed of convolutions rather than simple vector multiplications

Layer values correspond to convolutional filters, usually many filters per layer

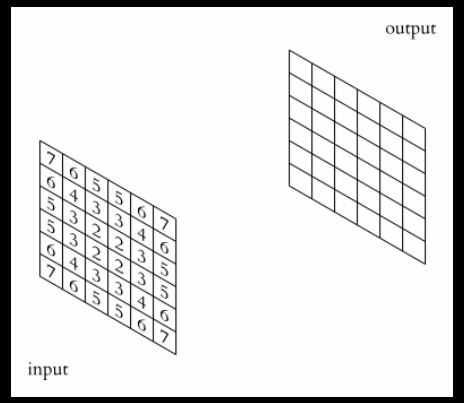
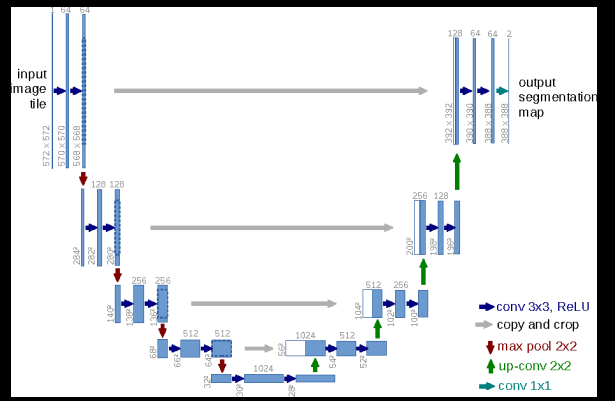


Image credit: Michael Plotke

U-Nets

Images are compressed in x-y space to generate more compact representations, then re-grown to the original dimensions



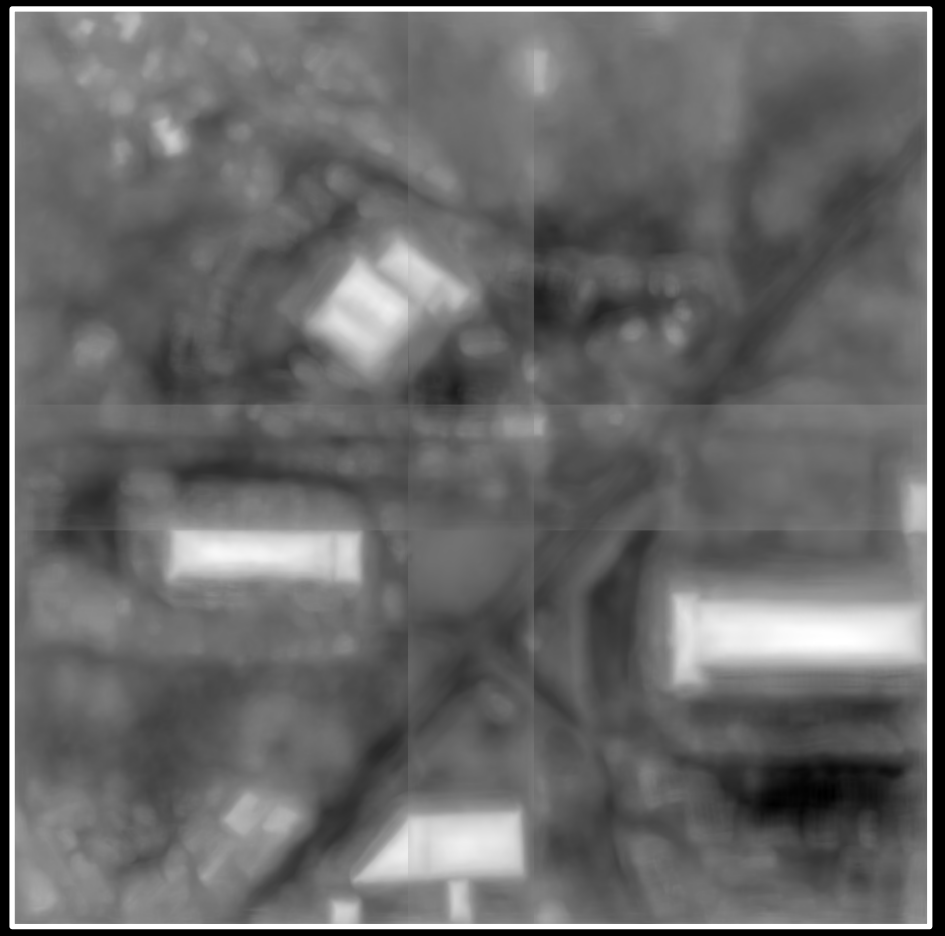
https://arxiv.org/abs/1505.04597

What pre-trained neural nets are currently available in Solaris?

SpaceNet challenge model name	General Architecture	Encoder	Training set	Input shape	# Params
XD_XD	U-Net	VGG16	MVOI PS RGB	3 × 512 × 512	29.3M
selim_sef resnet	U-Net	ResNet-34	MVOI PS RGB+NIR	4 × 416 × 416	30.0M
selim_sef densenet121	U-Net	DenseNet-121	MVOI PS RGB	3 × 384 × 384	15.6M
selim_sef densenet161	U-Net	DenseNet-161	MVOI PS RGB	3 × 384 × 384	41.1M

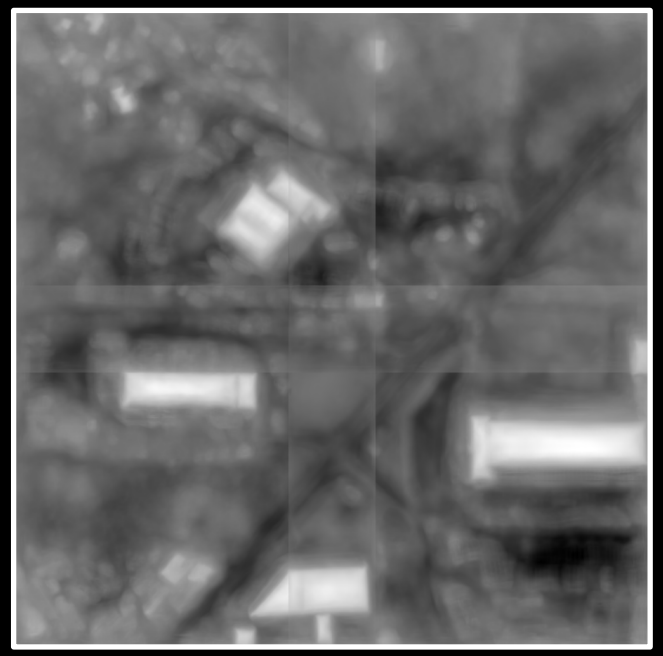
What comes out of the neural net?

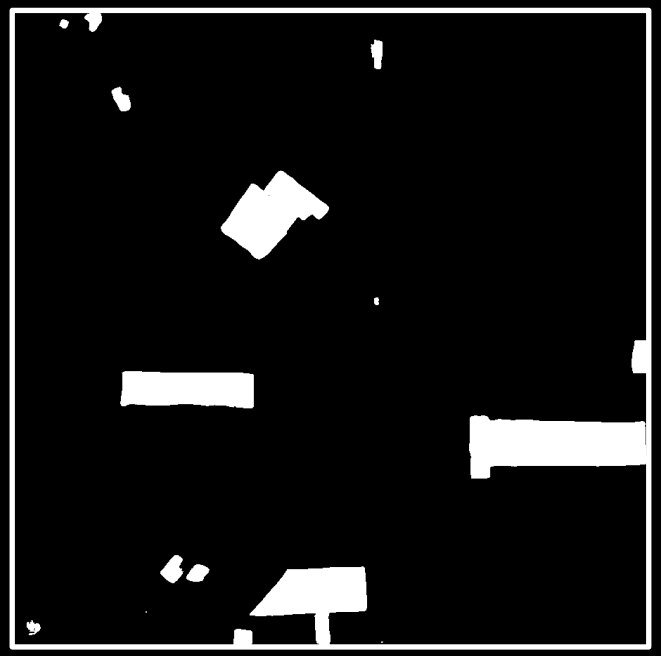
Input image



Raw output from XD_XD's model in the intro notebook

Solaris's automated prediction post-processing pipeline





POLYGON ((745804.5 3726489, 745812.5 3726489, ...
POLYGON ((746007 3726470, 746008.5 3726470, ...
POLYGON ((745825.5 3726436.5, 745826.5 3726436, ...
POLYGON ((745940.5 3726377, 745941 3726377, ...
POLYGON ((746192.5 3726256, 746201 3726256, ...
POLYGON ((745829.5 3726234, 745842 3726234, ...
POLYGON ((746078 3726202.5, 746080.5 3726202.5, ...
POLYGON ((745865 3726103, 745868 3726103, ...
POLYGON ((745880.5 3726096.5, 745883.5 3726096, ...
POLYGON ((745763 3726056.5, 745764.5 3726056.5...

Raw output from XD_XD's model in the intro notebook

Binarized output

Georegistered vector labels

Back to the notebook for a post-processing tutorial!