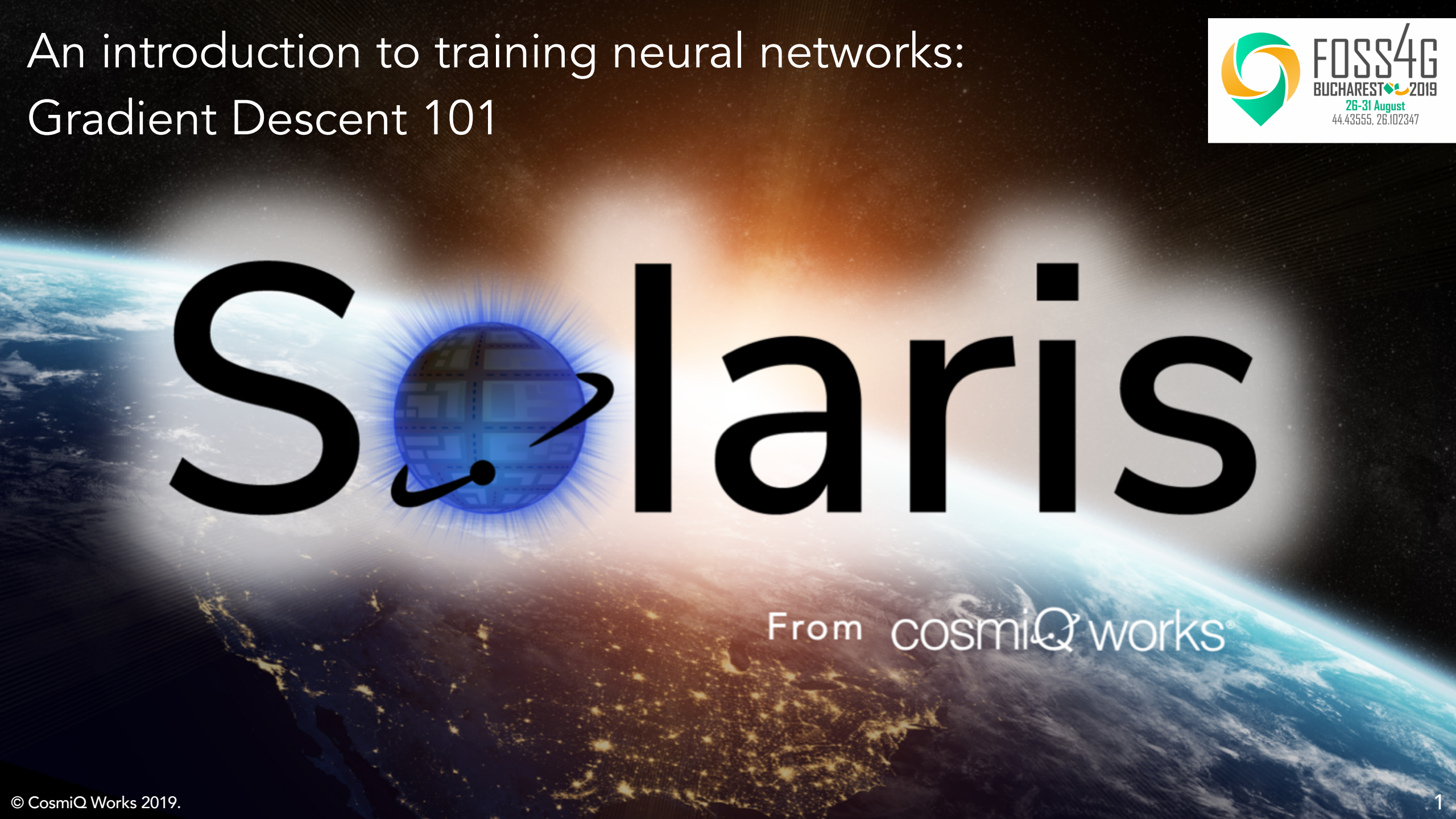
An introduction to training neural networks: Gradient Descent 101



FOSS4G BUCHAREST 2019
26-31 August
44.43555, 26.102347

A reminder of how NNs do math

```
[[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

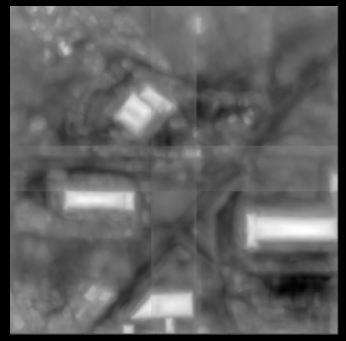
→

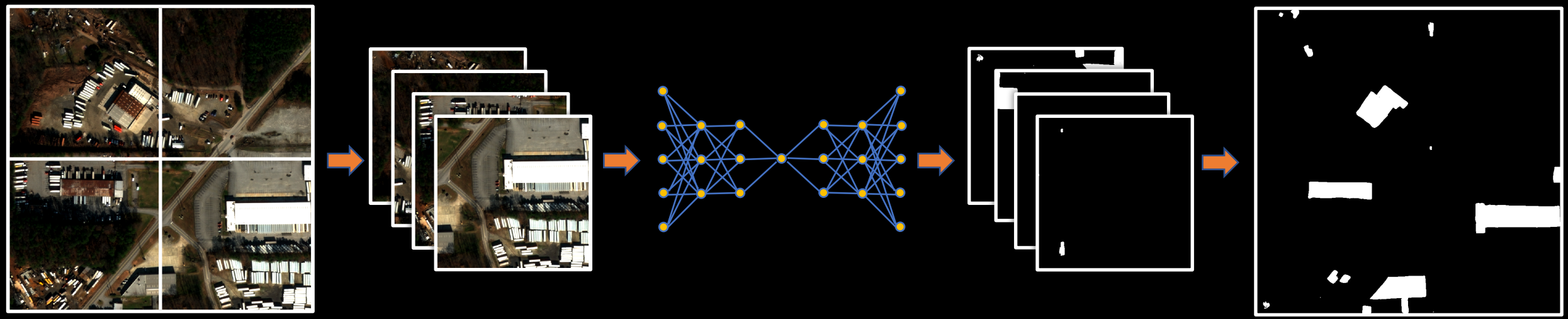
```
[  0, 1, 0.33,    0, …]
```

Output vector
(or array for convolutions)

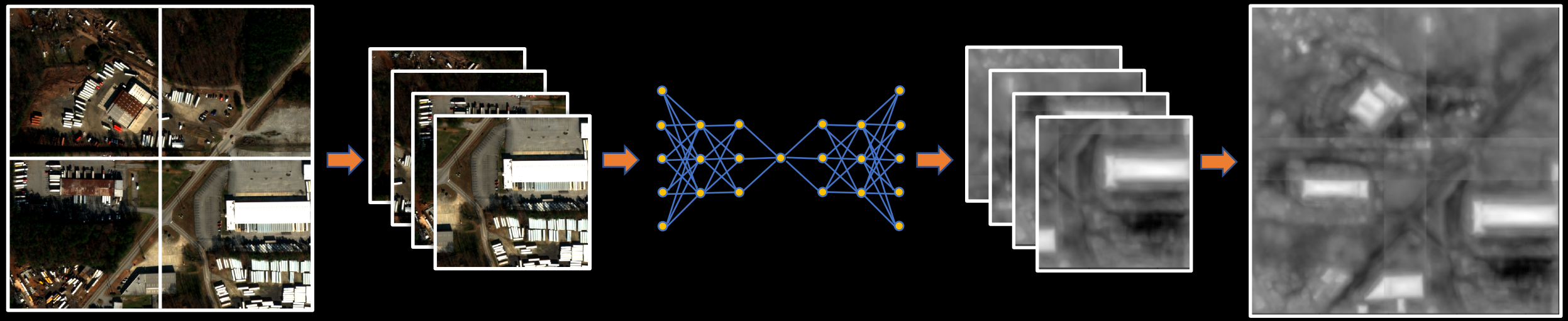
Results from input neurons (generally) get concatenated on a new axis when they're passed to the next "layer" of neurons.

A schematic view of inference

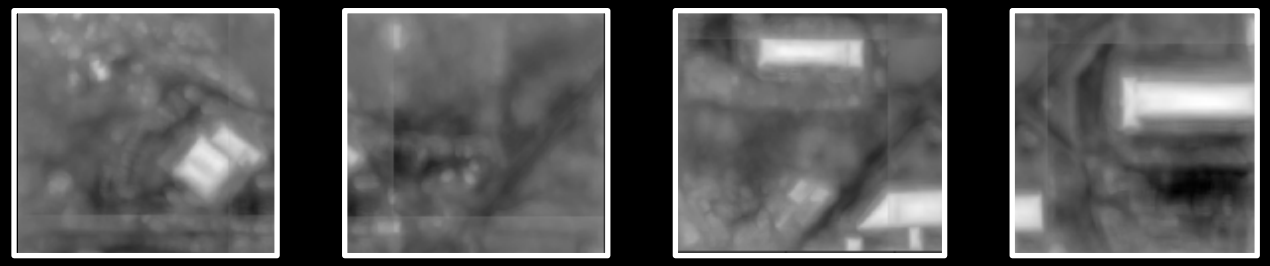




A more realistic view of neural net outputs

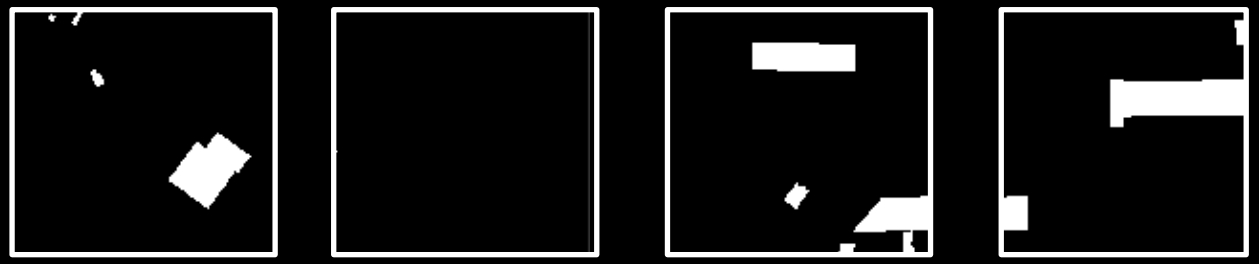


Model training goal



prediction outputs

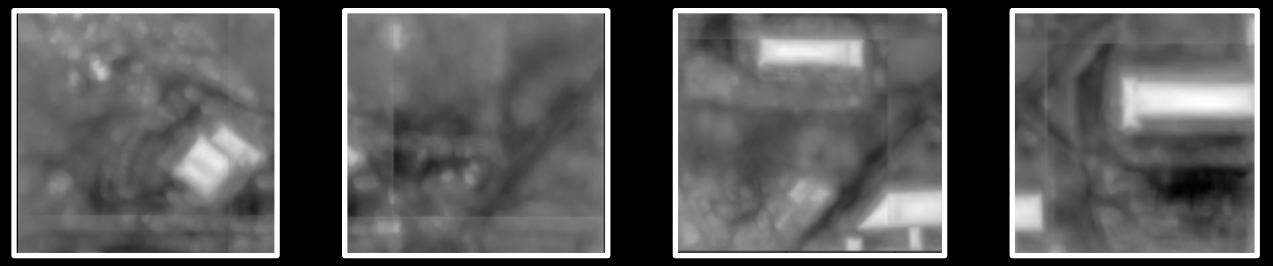
Make these…



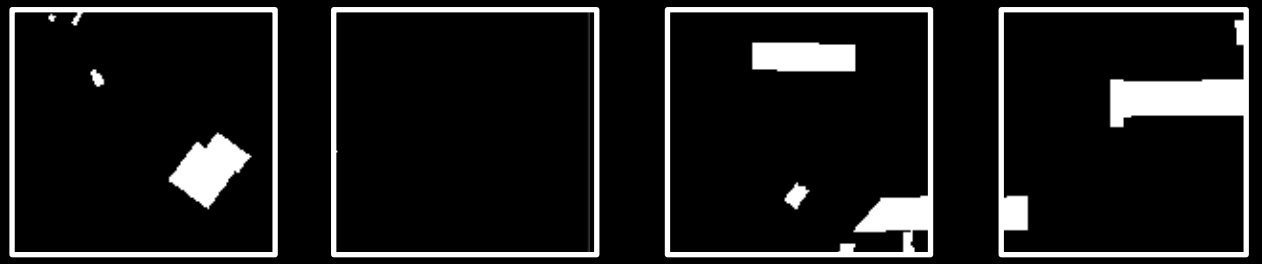
ground truth

Look more like these!

How can we tell (quantitatively) how similar or different these are?



prediction outputs



ground truth

Use a *loss function*:

- A quantitative measure of how bad the model's predictions are (higher score = worse)
- In this case, a measure of how different the prediction outputs are from ground truth

Using the loss to improve models

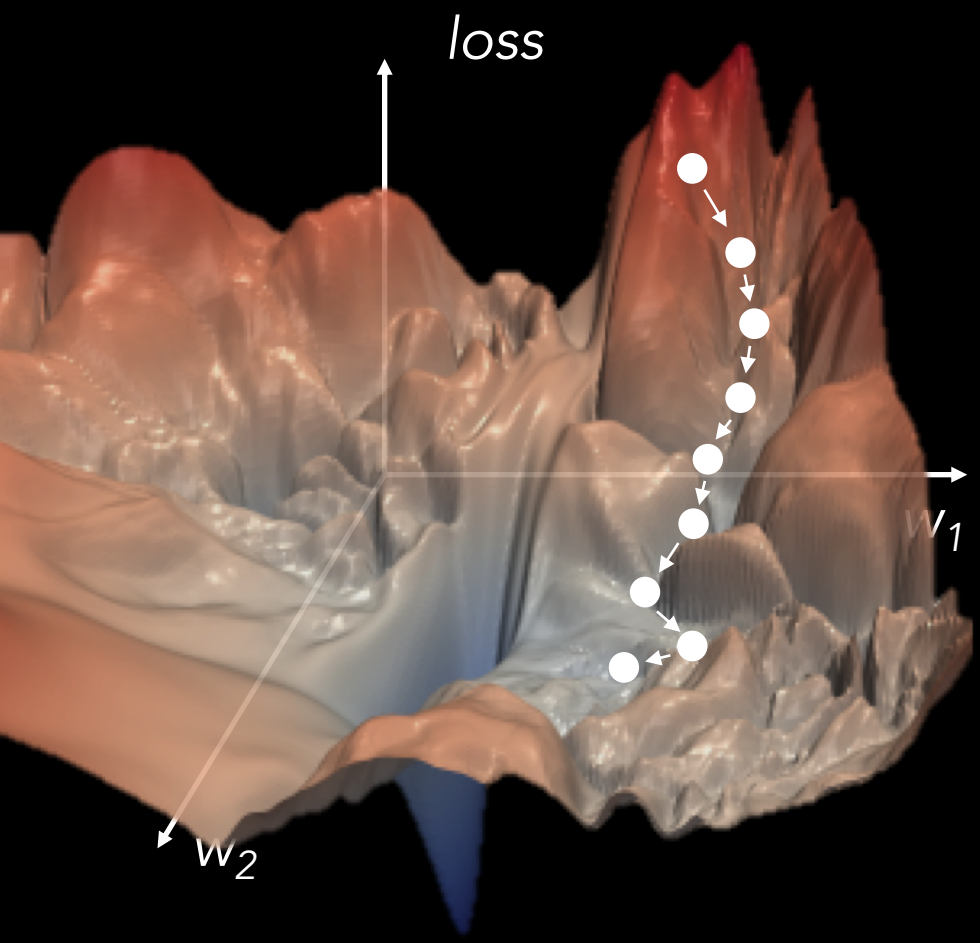


Image modified from Li et al. NeurIPS 2018, https://arxiv.org/abs/1712.09913

1. Calculate derivative of loss w.r.t weights, biases at current *w*, *b* values
2. Take a small step down the gradient
3. Repeat the above process *many* times

Let's try it!

Start up notebook 4 to "fine-tune" (minimally re-train) a model

So instead of predicting buildings in this…

It can predict buildings in this!