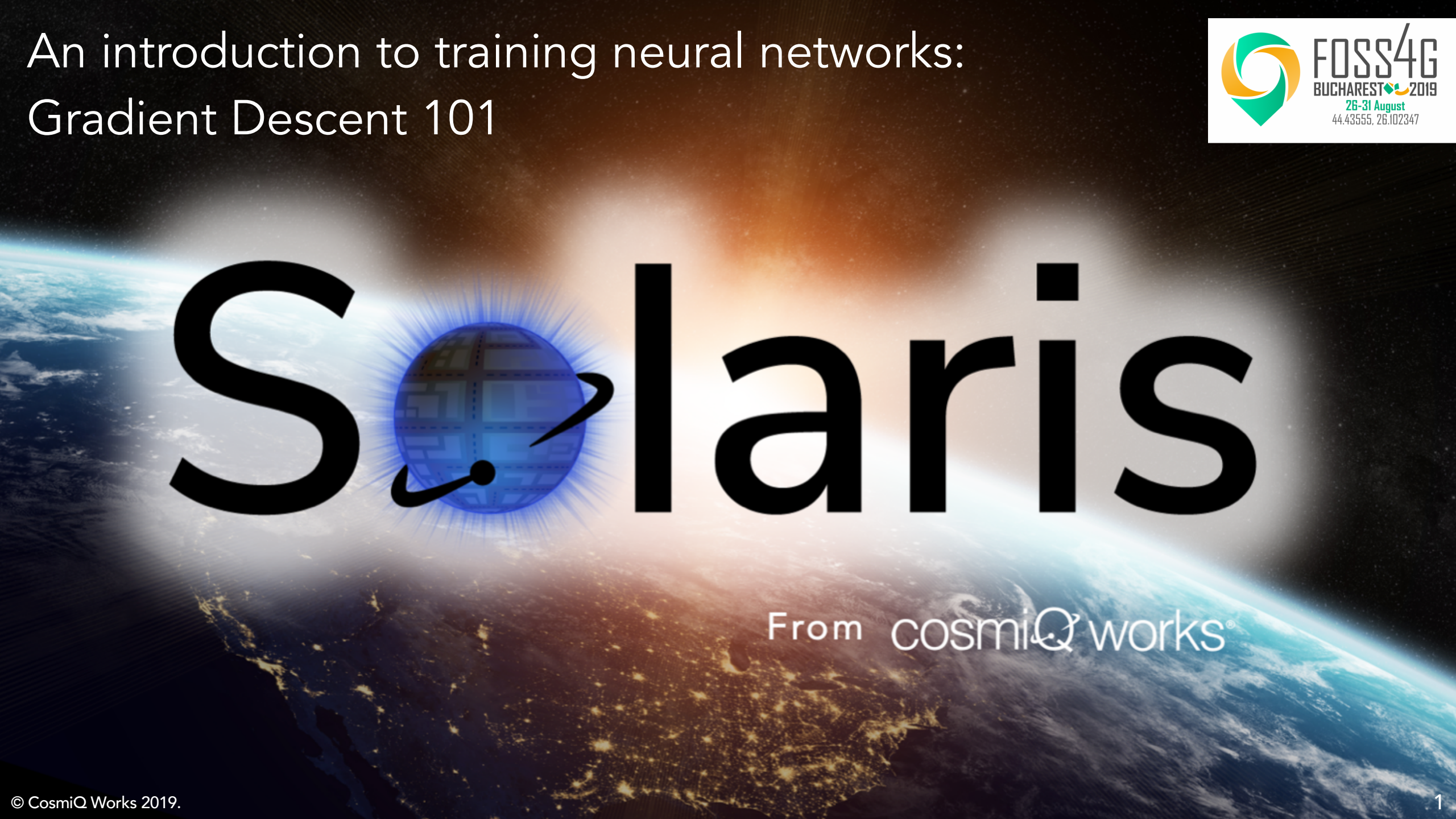
An introduction to training neural networks: Gradient Descent 101



FOSS4G BUCHAREST 2019
26-31 August
44.43555, 26.102347

A reminder of how neural nets 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"

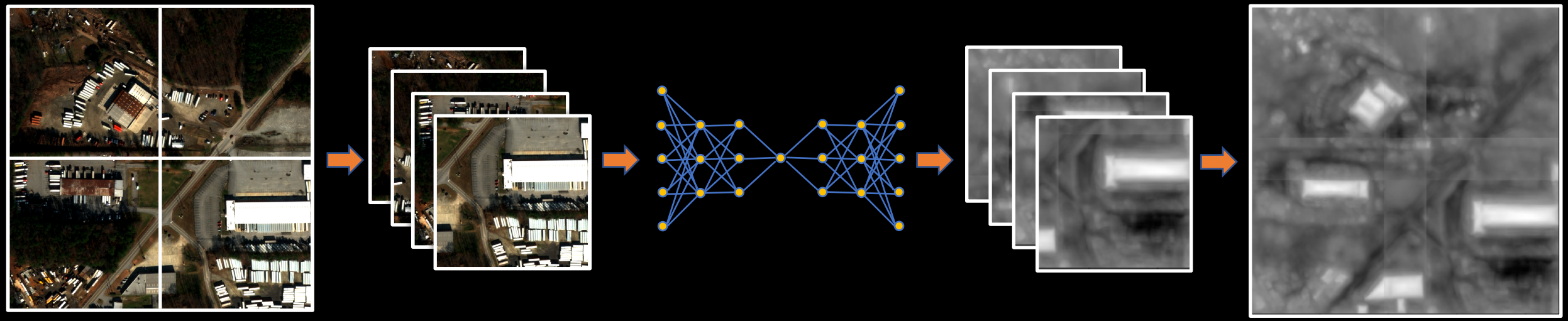
3rd: non-linearity

→

```
[  0, 1, 0.33,    0, …]
```

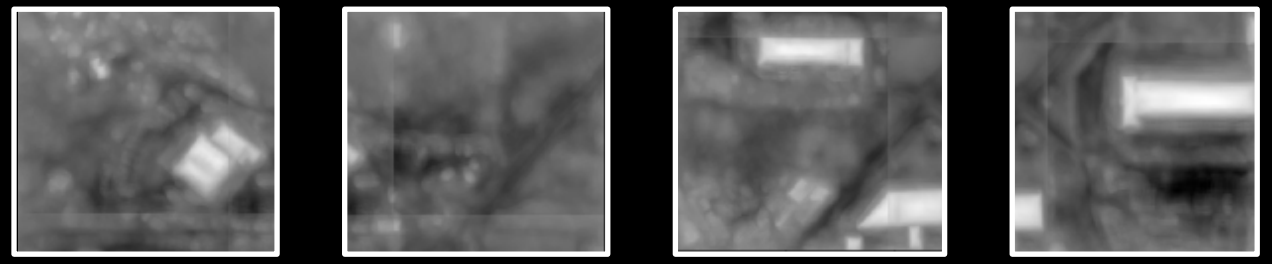
Output vector
(or array for convolutions)

A more realistic view of neural net outputs



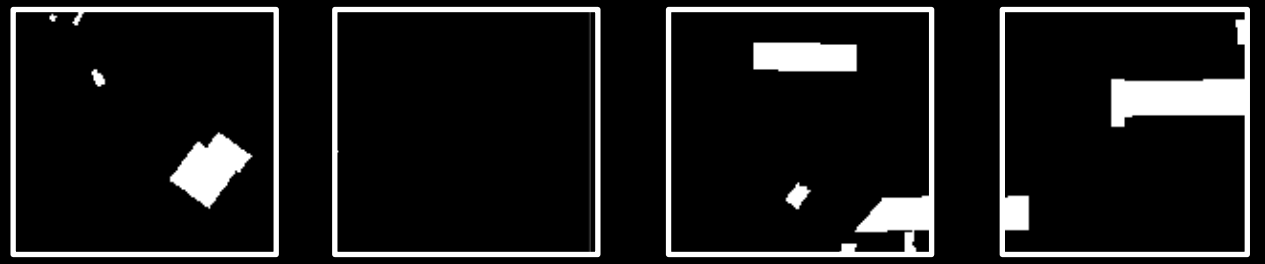
Training neural nets: the goal

Change the *weights* and *biases* so that these…



prediction outputs

Look more like these!

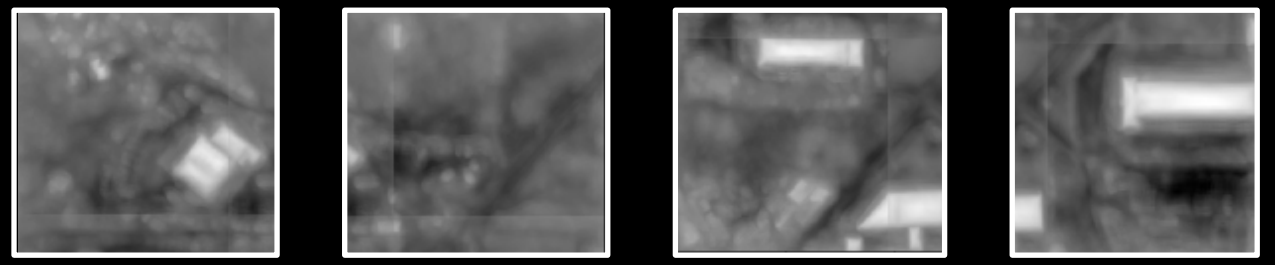


ground truth

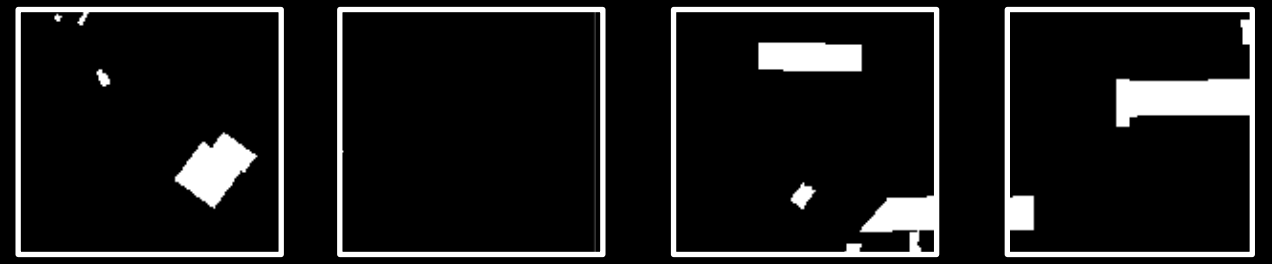
(Also, we need to use Solaris to make these ground truth masks from vector labels!)

How different are the predictions from the ground truth?

We need a way to quantitatively say how "far apart" these are!



prediction outputs

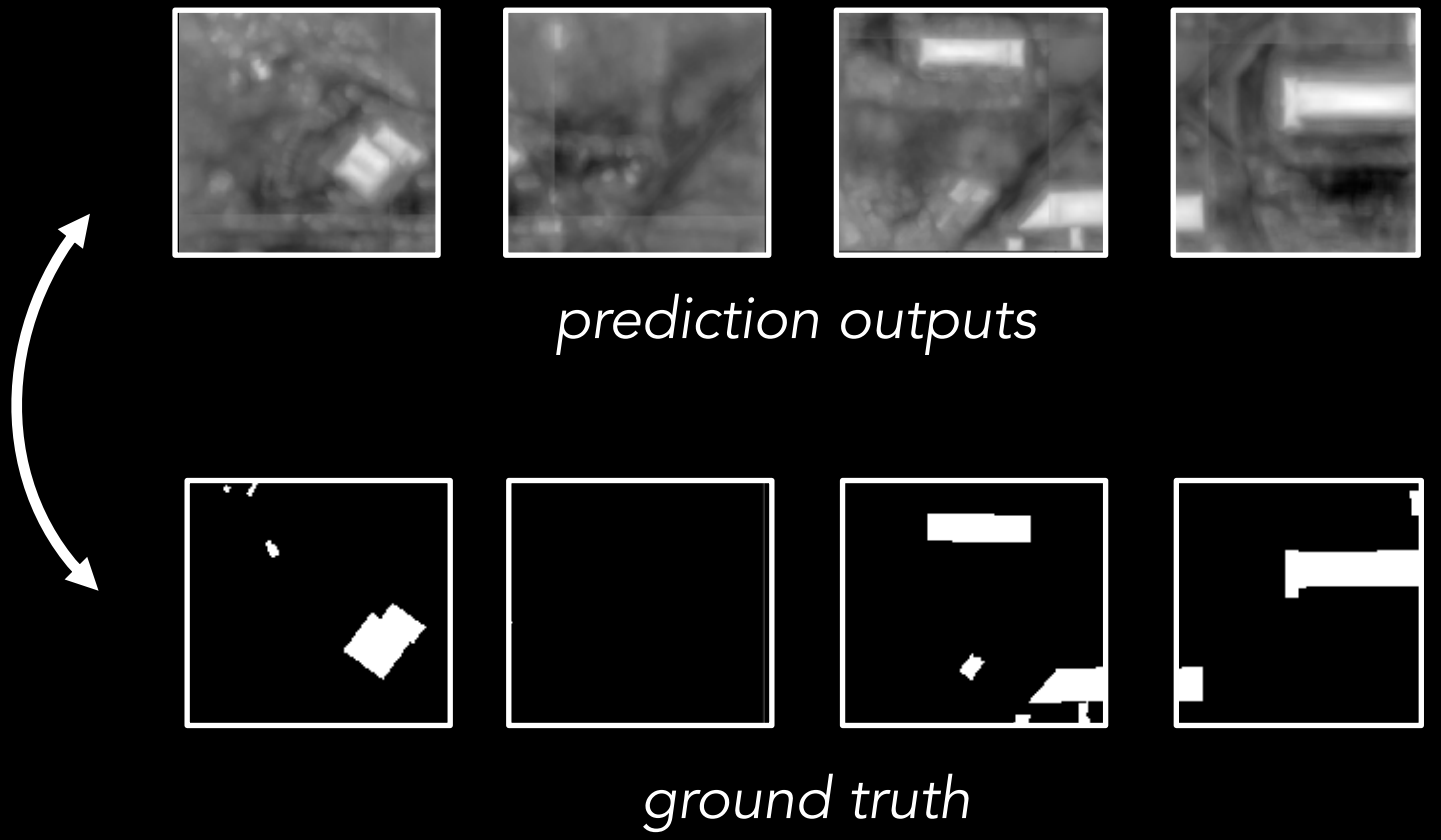


ground truth

How different are the predictions from the 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



prediction outputs

ground truth

Using the loss to improve models

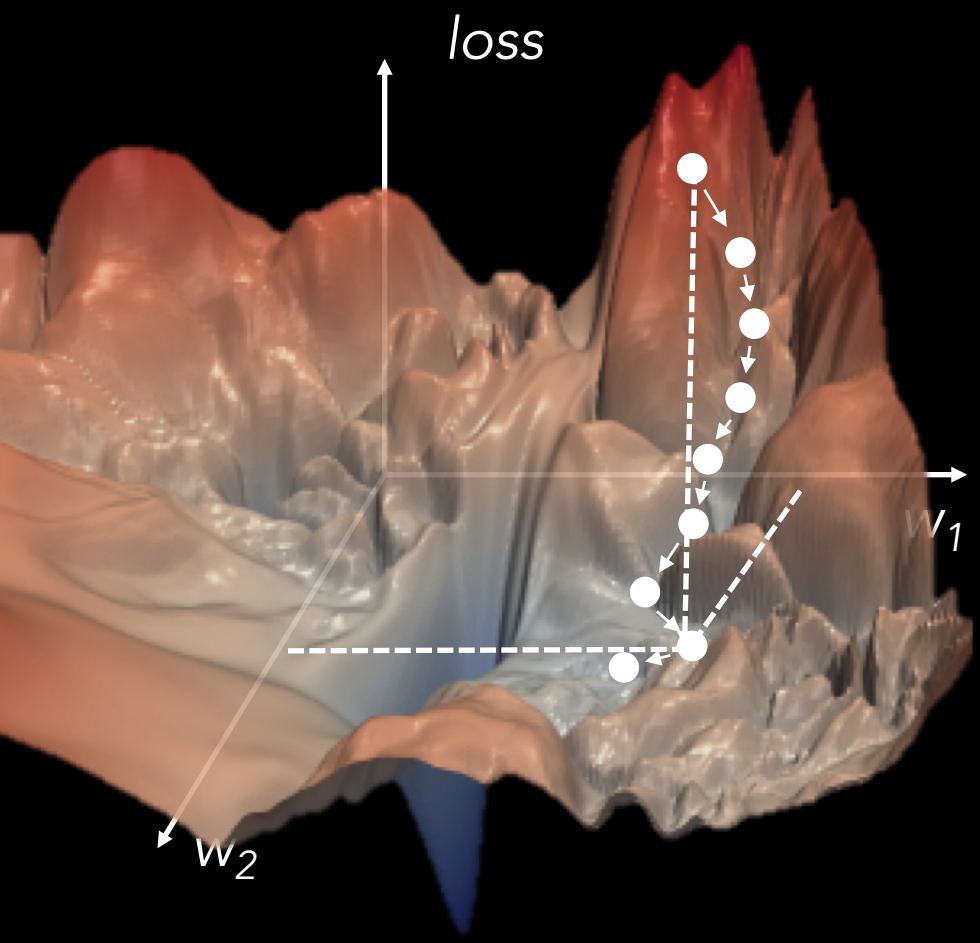


Image modified from Li et al. NeurIPS 2018, https://arxiv.org/abs/1712.09913

1. Calculate difference between *ground truth* and *predictions* (the ***loss***) given your weights w_1 and w_2
2. Calculate derivative of loss w.r.t weights, biases at current *w*, *b* values
3. Take a small step down the derivative gradient and re-measure loss with new w_1, w_2
4. Repeat the above process *many* times

Note: model training is very computationally intensive and is very slow without deep learning-specific hardware (GPUs, TPUs, etc.)

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!