What excites a neuron ?

The inversion process that we described by Deconvolution and Saliency Maps

- Is **input dependent** (depends on an example in a dataset)
- The Saliency Map for a single (or summary) location at feature map k of layer l
- Depends on a particular input $\mathbf{x}^{(\mathbf{i})}$ being feed to layer 0

By finding the input examples that "most excite" the feature map, we were indirectly able to guess at the feature being recognized by the feature map.

We now demonstrate a more direct **input independent** approach

- Determine the input value (not necessarily an example in a dataset)
- That excites (causes large values)
- A single location/neuron (or summary) of feature map k of layer l

By finding the *single input* that most excites a feature map, we may interpret the feature map as attempting to recognize similar inputs.

Gradient Ascent: Inverting a Neural Network

We have already introduced the notion of computing the *sensitivity* of a feature

- At spatial location idx of feature map k of layer l
- To a change in the feature at spatial location idx' feature map k' of layer 0

$$\boldsymbol{s}_{(l),\text{idx},k,(0),\text{idx}',k'} = \frac{\partial \mathbf{y}_{(l),\text{idx},k}}{\partial \mathbf{y}_{(0),\text{idx}',k'}}$$

We used this to define Saliency Maps

- Which indicate how much more "excited" $\mathbf{y}_{(l),\text{idx},k}$ becomes
- When we increase the stimulus at layer $0: \mathbf{y}_{(0),\text{idx}',k'}$
- For a particular input $\mathbf{y}_{(0)} = \mathbf{x}^{(\mathbf{i})}$

We also know that Gradient Descent is used

- To find the optimal value $\mathbf{W}^*$ for the weights $\mathbf{W}$ that parameterize the layers of a Neural Network
- By optimizing (find the minimum) a Loss Function
- Using derivatives of the Loss with respect to the weights

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} L(\hat{\mathbf{y}}, \mathbf{y}; \mathbf{W})$$

What happens if we *combine* these two ideas:

- Find the optimal value for input $\mathbf{x}^*$
- By optimizing (maximizing) the value $\mathbf{y}_{(l),\text{idx},k}$
- Using derivatives of $\mathbf{y}_{(l),\text{idx},k}$ with respect to $\mathbf{x}$?

$$\mathbf{x}^* = \underset{\mathbf{y}_{(0)}=\mathbf{x}}{\operatorname{argmax}} \mathbf{y}_{(l),\text{idx},k}$$

(Remember that the value of $\mathbf{y}_{(l),\text{idx},k}$ is a function of input $\mathbf{x}$)

That is:

- We can use Gradient Descent
- To find the value $\mathbf{x}^* = \mathbf{y}_{(0)}$
- That, when used as input to the Neural Network
- Maximizes the value of a particular neuron $\mathbf{y}_{(l),\text{idx},k}$
- Using derivatives

$$\frac{\partial \mathbf{y}_{(l),\text{idx},k}}{\partial \mathbf{y}_{(0),\text{idx}',k'}}$$

We start off by initializing $\mathbf{y}_{(0)}$ to random noise.

- Compute $\mathbf{y}_{(l),\text{idx},k}$ on the Forward Pass
- Compute $\frac{\partial \mathbf{y}_{(l),\text{idx},k}}{\partial \mathbf{y}_{(0),\text{idx}',k'}}$ given the current $\mathbf{y}_{(0)}$, on the Backward Pass
- Move $\mathbf{y}_{(0)}$ in the direction of the derivative

After some number of epochs, we obtain an $\mathbf{x}^* = \mathbf{y}_{(0)}$ that maximizes $\mathbf{y}_{(l),\text{idx},k}$.

That is: we find the input $\mathbf{x}^*$ that maximally excites $\mathbf{y}_{(l),\text{idx},k}$.

We can then interpret $\mathbf{y}_{(l),\text{idx},k}$ as looking for the feature

> *"Is like $\boldsymbol{x}^*$"*

Since we are maximizing a value ($\mathbf{y}_{(l),\text{idx},k}$) rather than minimizing one (the Loss)

- This method is called *Gradient Ascent*
- My multiplying the objective $\mathbf{y}_{(l),\text{idx},k}$ by -1 we can trivially turns this into a minimization problem

Conclusion

Gradient Ascent is a technique for find the input $\mathbf{x}^*$ that is the "paradigmatic" value for a feature at layer l

It is a simple combination of techniques that we have already learned.

You can do many more interesting things with Gradient Ascent

- What if your initial guess is not random noise ?
- What if you add a constraint on $\mathbf{x}^*$?

We will explore these ideas in another lecture.

```
In [4]: print("Done")
```

```
Done
```