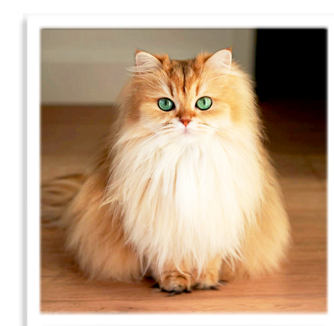
Optimization of deep networks

Data

- Input: $\{\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}\}$
- Label: $\{\mathbf{y}_0, \ldots, \mathbf{y}_{N-1}\}$
- Dataset: $D = \{(\mathbf{x}_0, \mathbf{y}_0), \ldots, (\mathbf{x}_{N-1}, \mathbf{y}_{N-1})\}$

( , dog)

( , dog)

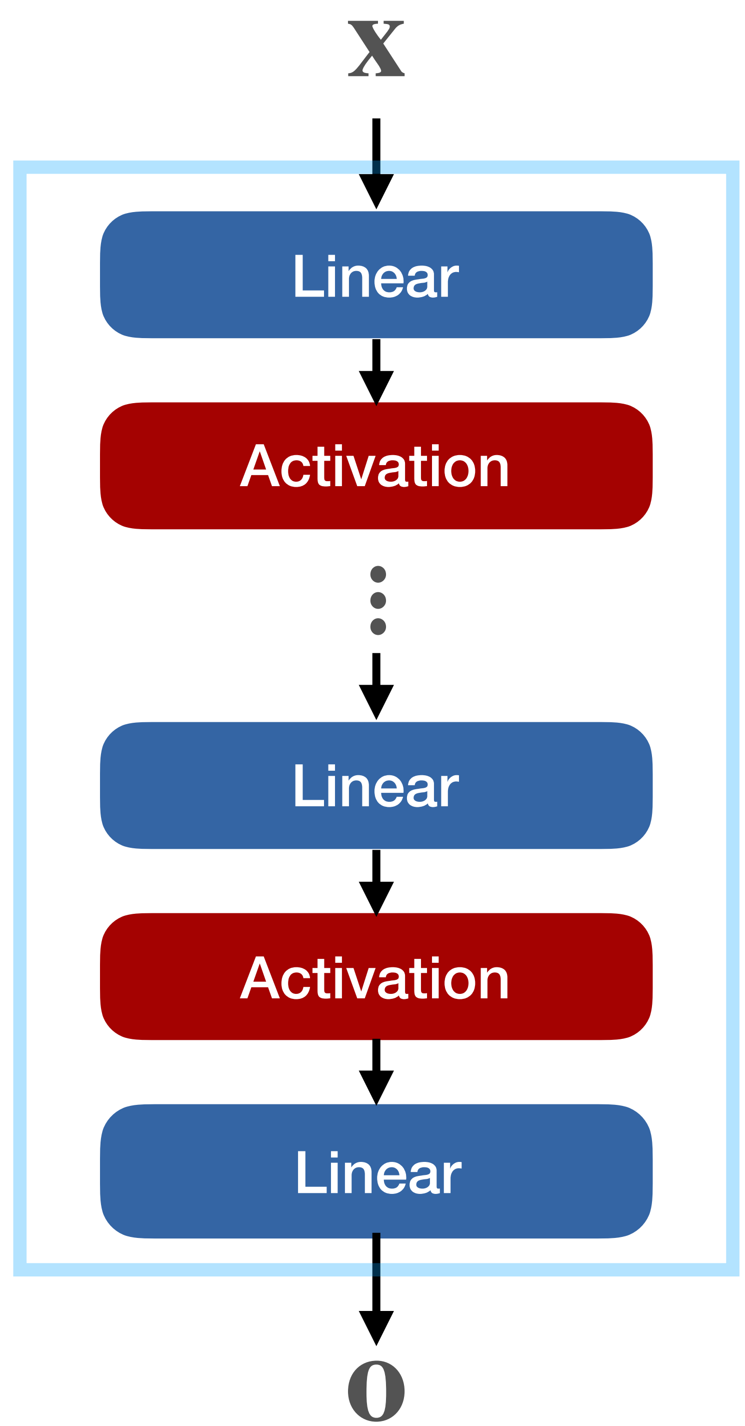
( , cat)

( , cat)

⋮

Model

- Deep network $f : (\mathbf{x}, \theta) \rightarrow \mathbf{o}$
  - Layers of computation
  - Parameters θ
  - Differentiable computation graph

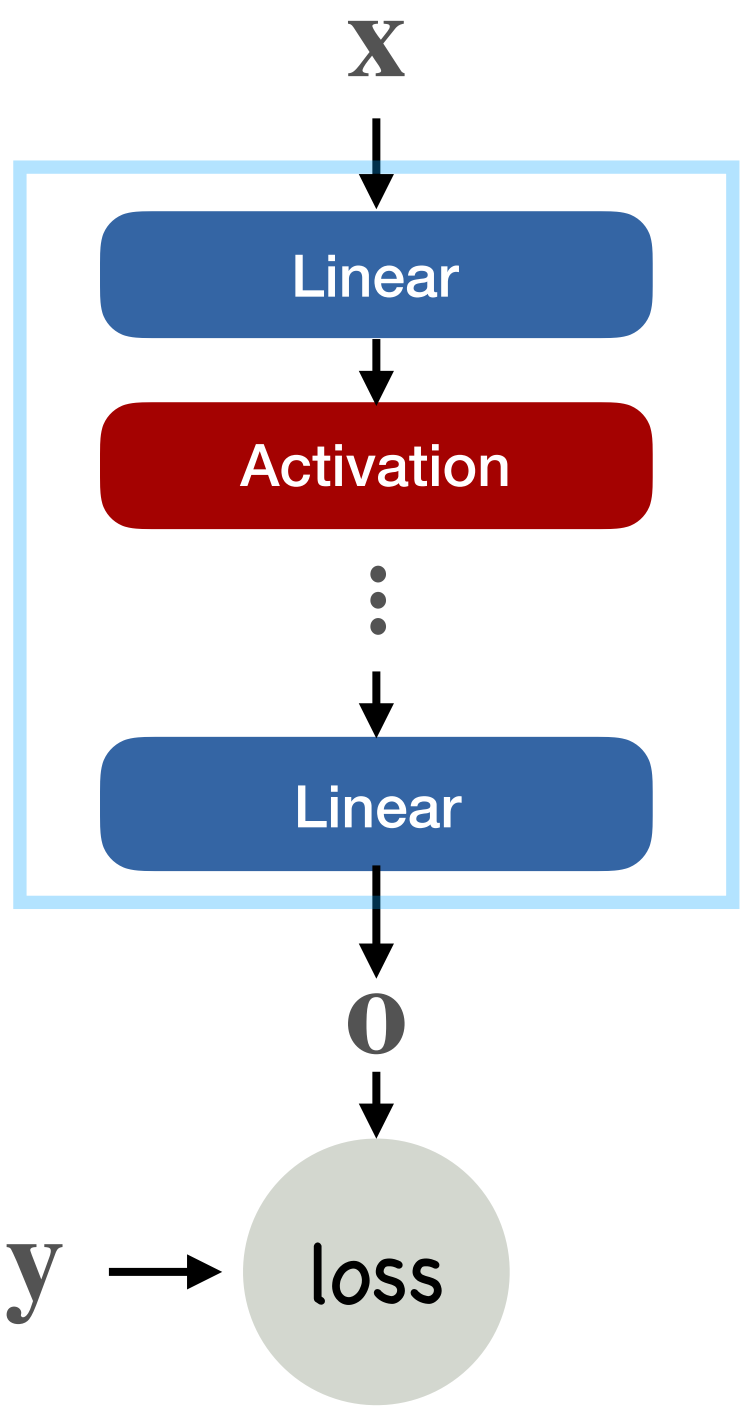


Loss

- Differentiable $\ell(\mathbf{o}, \mathbf{y})$
- Regression
  - Distance norm

    $$\ell(\mathbf{o}, \mathbf{y}) = \|\mathbf{o} - \mathbf{y}\|$$
- Classification
  - Cross Entropy

    $$\ell(\mathbf{o}, y) = -\log p(y)$$
- Over training dataset
  - $L(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim D}[\ell(f(\mathbf{x}, \theta), \mathbf{y})]$

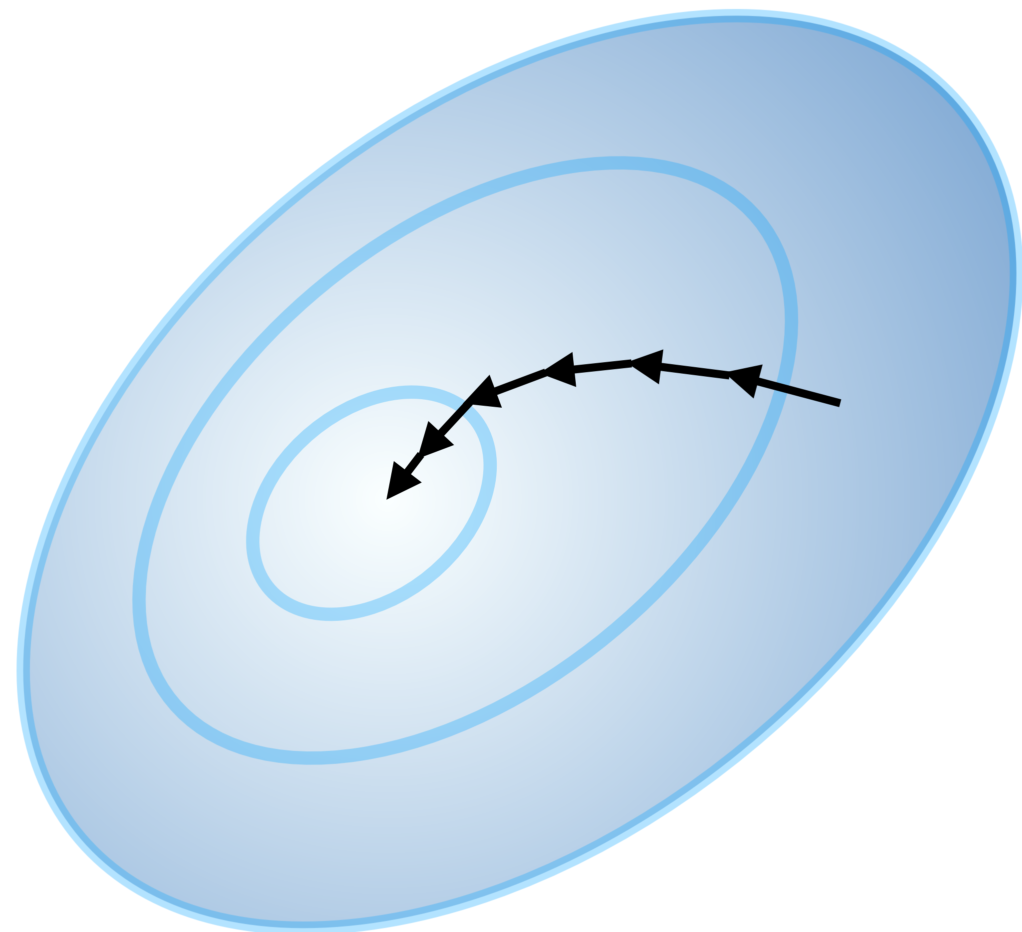


Optimization

- Minimize $L(\theta)$

Gradient Descent

- Repeat until convergence:
  - $\theta := \theta - \epsilon \frac{dL(\theta)}{d\theta}$



Issue with Gradient Descent

Now if you look at larger models
such as deep networks,
gradient descent is actually going to break.
And it's going to break mainly because it's too slow
in updating the parameters.

So computing a gradient for gradient descent
for deep networks, involves computing the gradient
of all of your data off the loss function.
Off the output of your network.
So you have to loop over your entire data set
every time you want to change the parameters
just a little bit.

- Slow to compute gradient
  - $$\frac{dL(\theta)}{d\theta} = \mathbb{E}_{\mathbf{x},\mathbf{y}\in D}[\frac{d\ell(f(\mathbf{x},\theta),\mathbf{y})}{d\theta}]$$