Previously on…
Deep Learning

Romain Tavenard (Université de Rennes)
A course @UR2

Stochastic gradient descent

- Given a loss function to minimize:

$$\mathcal{L}(\text{dataset}, \theta) = \frac{1}{|\text{dataset}|} \sum_{(x,y)\in\text{dataset}} \ell(x, y, \theta)$$

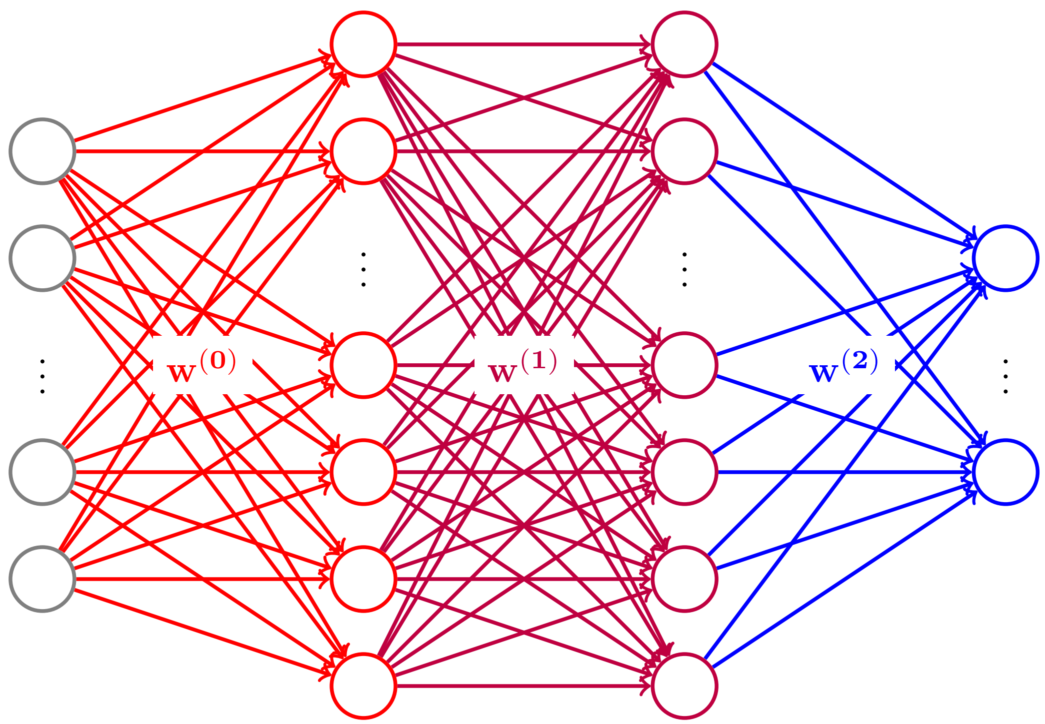
- Gradient Descent update rule:

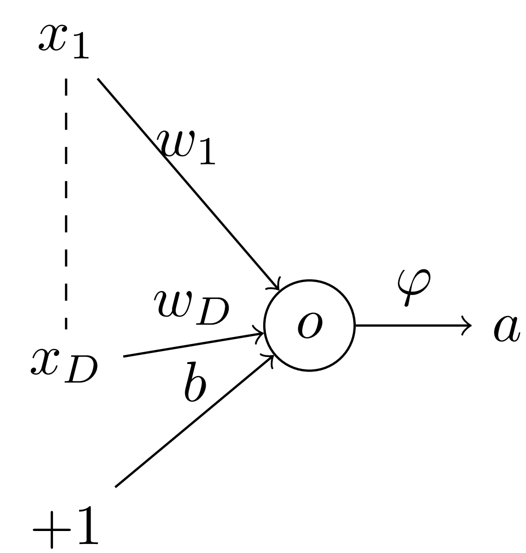
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\text{dataset}, \theta^{(t)})$$

- (Mini-batch) Stochatic Gradient Descent update rule:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \mathcal{L}({\color{red}\text{minibatch}}, \theta^{(t)})$$

Neural networks and back-propagation





$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial w^{(2)}}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial w^{(1)}}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(0)}} = \frac{\partial \mathcal{L}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial o^{(1)}} \frac{\partial o^{(1)}}{\partial w^{(0)}}$$

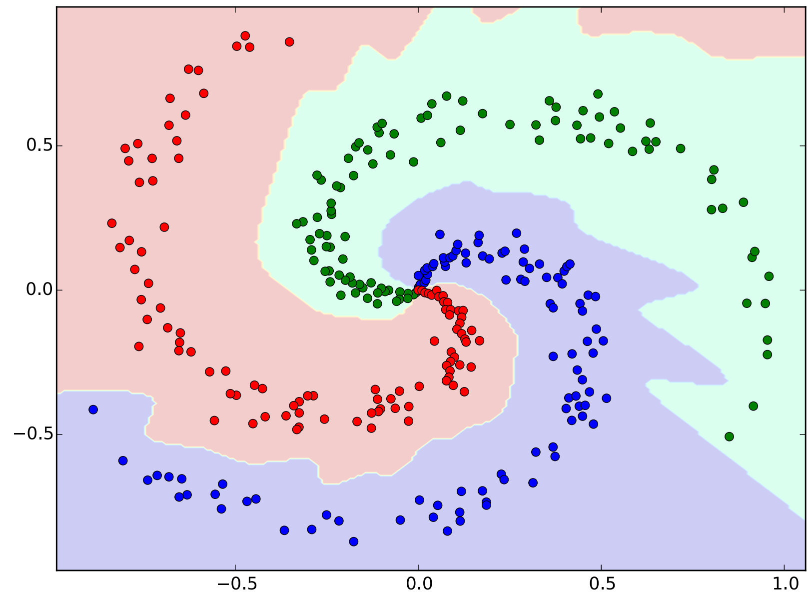
$$\frac{\partial a^{(l)}}{\partial o^{(l)}} = \varphi'(o^{(l)})$$

$$\frac{\partial o^{(l)}}{\partial a^{(l-1)}} = w^{(l-1)}$$

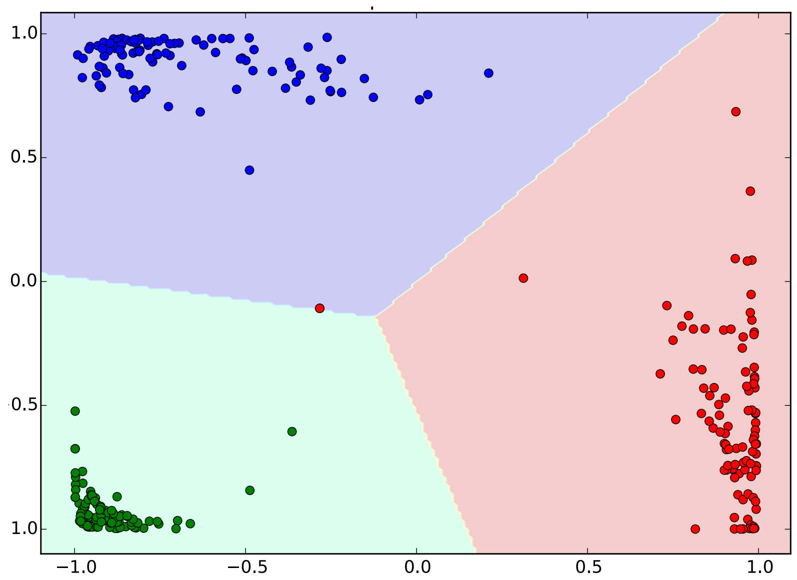
End-to-end learning

- Classification using MLP
  - Hidden layers: non-linear transformations
  - Last layer: logistic regression

- Example with a 3-hidden-layer net (last layer with 2 units)



Input space



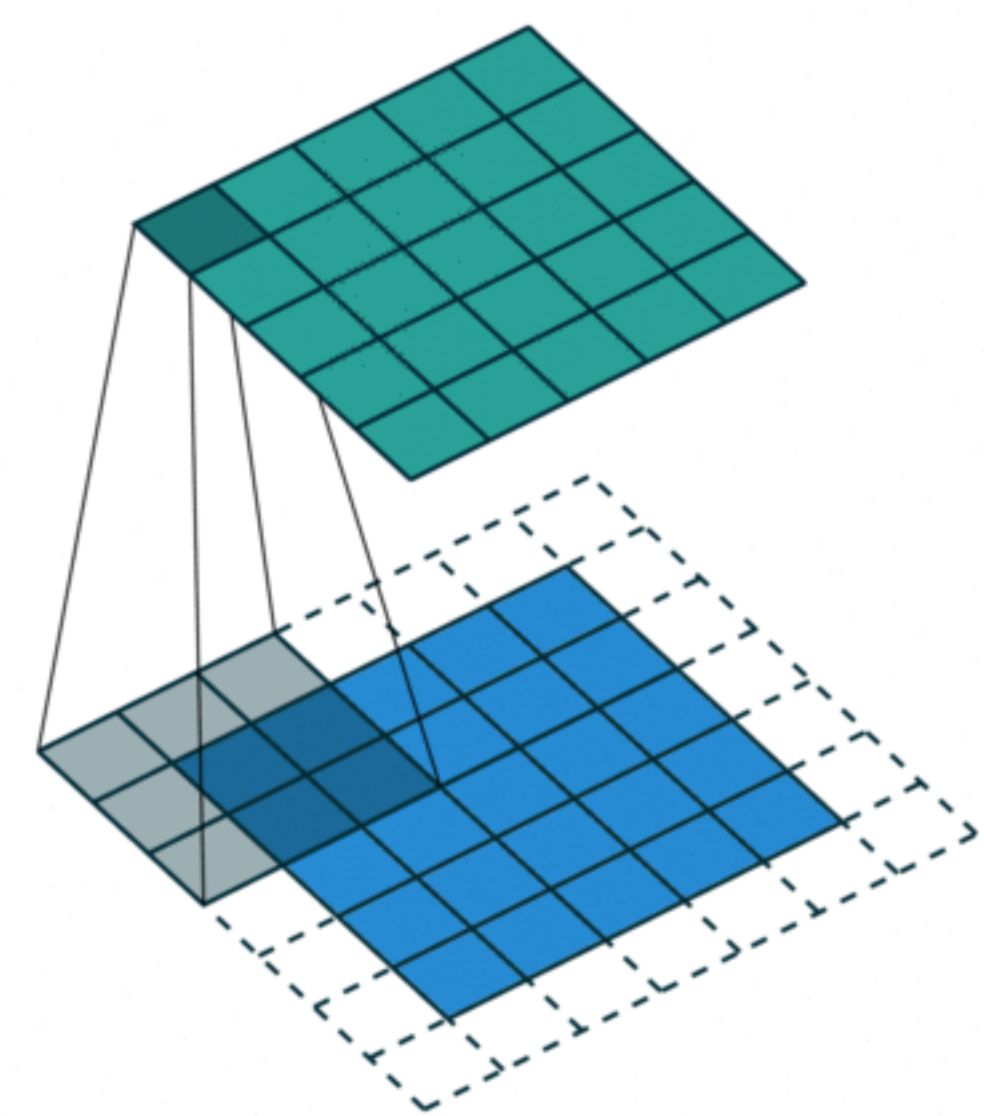
Last hidden layer

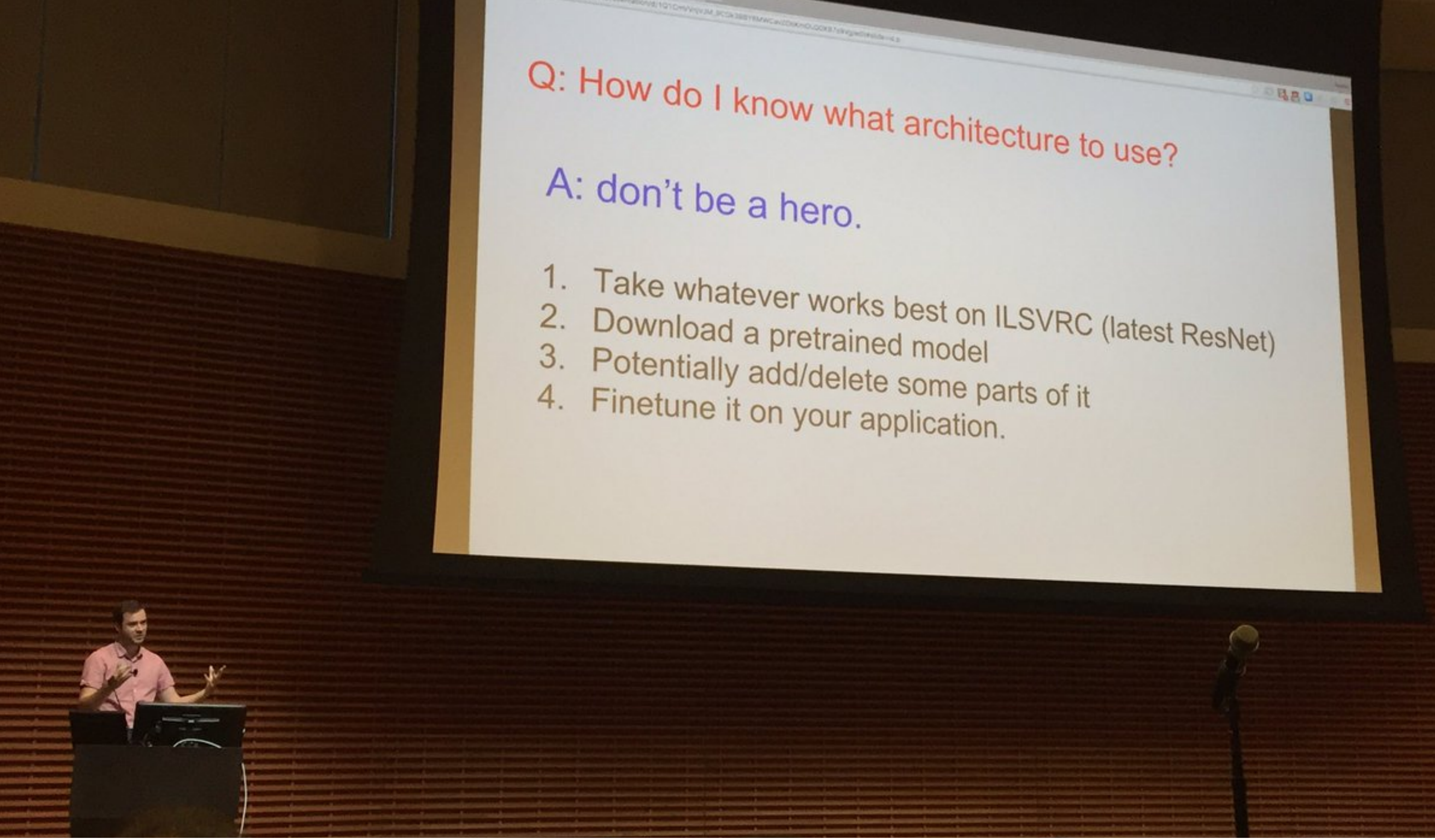
Why such sudden changes?

- Big data (ImageNet & co)
- Big infrastructures (GPU)
- Optimization
  - Algorithms
  - *Tricks* (initialization, regularization, fighting vanishing gradients)
- Automatic differentiation libraries (tensorflow, pytorch, …)

The convolution operator

- 2D convolution
  - Blue: input image
  - Gray: convolution kernel
  - Cyan: activation map
- Convolution operation = Dot product between
  - convolution kernel (aka filter)
  - subpart of the input





Andrej Karpathy,
Deep Learning Summer School,
2016