Machine Learning - Deep Learning fundamentals (69152)

Master in Robotics, Graphics and Computer Vision
Ana C. Murillo

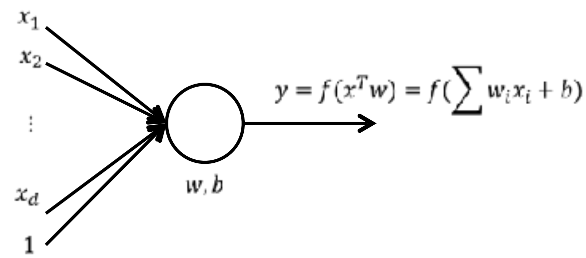
Universidad Zaragoza

Next

- What's deep learning?
  - Related topics
  - DL pipeline
- Fundamentals of DL
  - Review basic concepts
  - **NN and DNN**

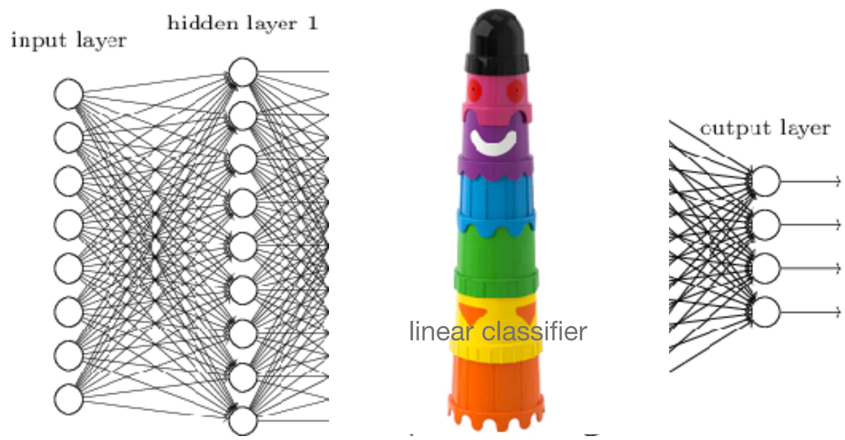
Neural Networks and Deep Neural Networks

- **Neural Networks**
  - **Neuron**: atomic computational unit in NN.
    Params: w, b and f.



  - Activation function (f) usually non linear
  - Neural Network: connect several layers, neurons, …
  - *Perceptron:* simplest NN. 1 layer - binary linear classifier

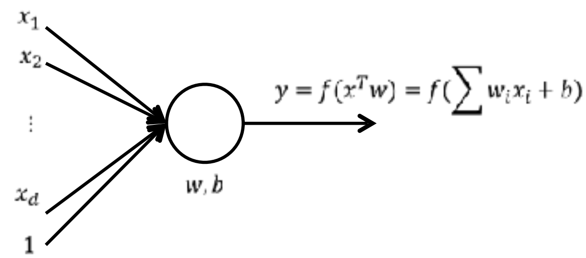
data/features



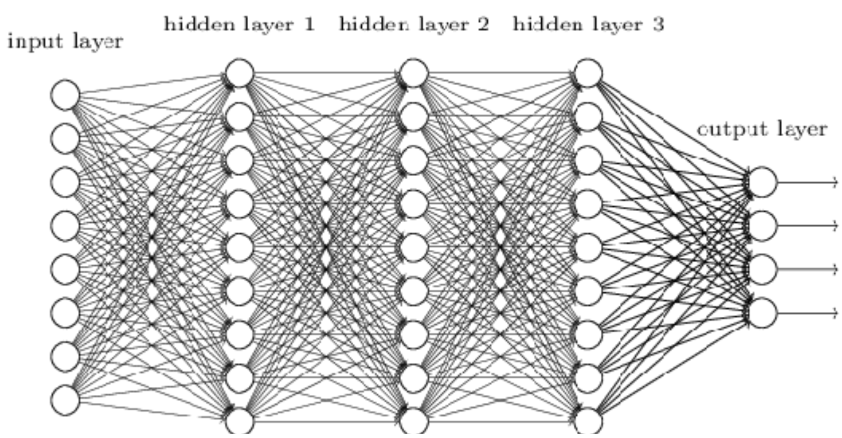
output

Neural Networks and Deep Neural Networks

- **Neural Networks**
  - Neuron: atomic computational unit in NN.
    Params: w, b and f.
  - Activation function (f) usually non linear
  - Neural Network: connect several layers, neurons, …
  - **Deep** Neural Network: many hidden layers



data/features



output

Deep Neural Network

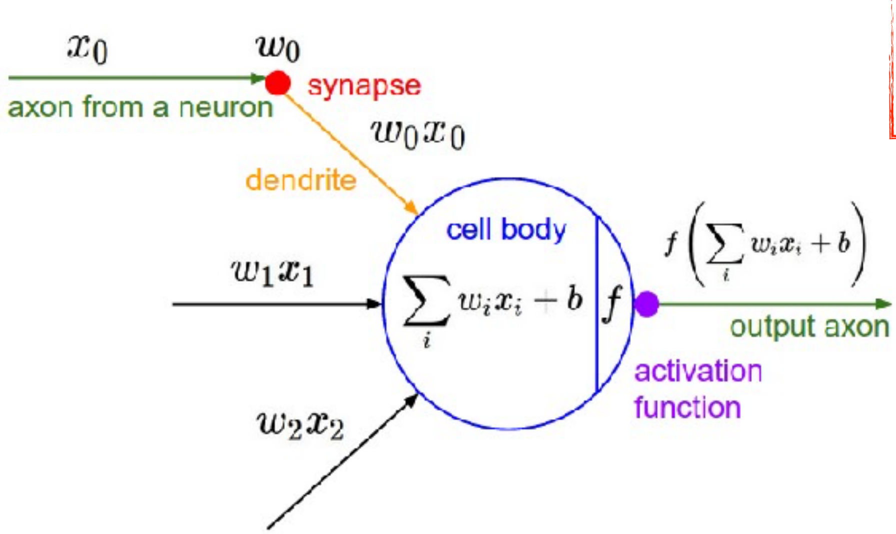
- Deep **feed-forward** Neural Networks:
  data propagated through the network to predict the output

- **Some important DNN *ideas* and *ingredients***
  - forward-backward pass
  - activation, optimization, gradient descent, backpropagation
  - parameters (model) and hyperparameters (config.)

Deep Neural Network

Forward pass (prediction, inference, …)

of a fully-connected layer: *one matrix multiplication followed by a bias offset and activation function*



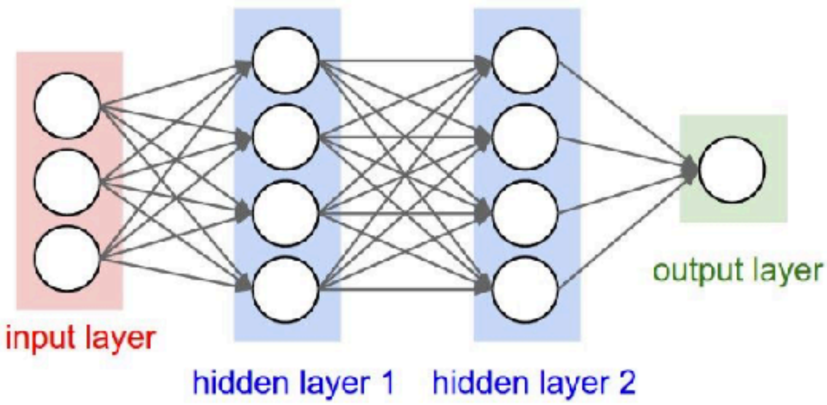
What do we compute here? Score, Loss, Optimization?

source: http://cs231n.github.io

Deep Neural Network

Forward pass. Feed-forward computation on a 3 layer NN

```
f(W1, W2, W3) = a(W1, b(W2, c(W3)))
```




```
fc1 = X.dot(W1) + b1     # A?
X2 = f(fc1)  # B?
fc2 = X2.dot(W2) + b2 # C?
X3 = f(fc2)  # D?
scores = X3.dot(W3) + b3 # E?
```

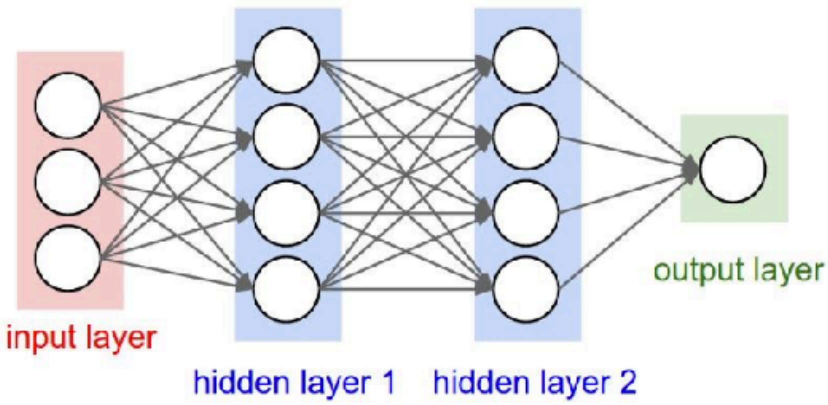
What's A, B, C, D, E?

source: http://cs231n.github.io

Deep Neural Network

Forward pass. Feed-forward computation on a 3 layer NN

```
f(W1, W2, W3) = a(W1, b(W2, c(W3)))
```

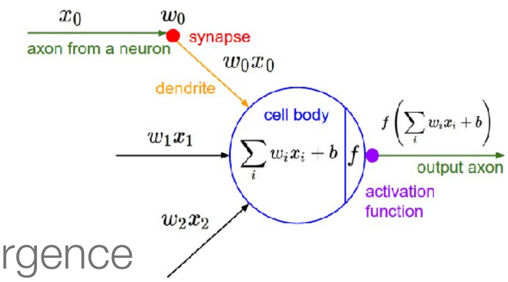



```
fc1 = X.dot(W1) + b1     # fully connected
X2 = f(fc1)  # Activation Function from hidden layer1
fc2 = X2.dot(W2) + b2 # fully connected
X3 = f(fc2)  # Activation Function from hidden layer2
scores = X3.dot(W3) + b3 # output layer
```

source: http://cs231n.github.io

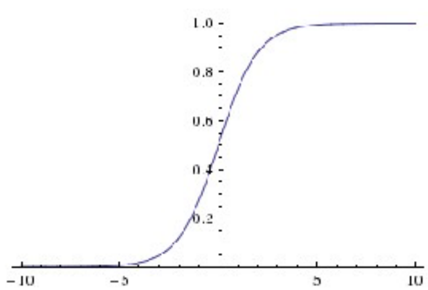
Deep Neural Network

- **Activation** of a Neuron - **non linearity!**

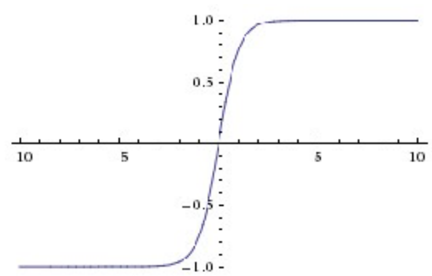


- ReLU very popular `f(x) = max(0,x)`
  - fast to compute and shown to accelerate convergence
  - weak. high learning rate can “kill” many of the neurons (never activated)
  - generalisation —> MaxOut unit $\max(w_1^T x + b_1, w_2^T x + b_2)$.

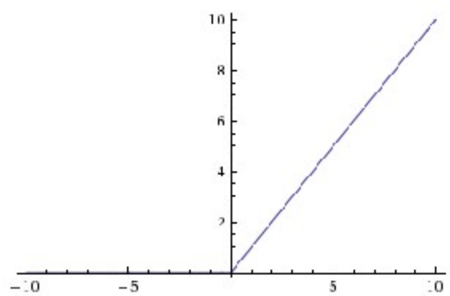
Sigmoid



Tanh



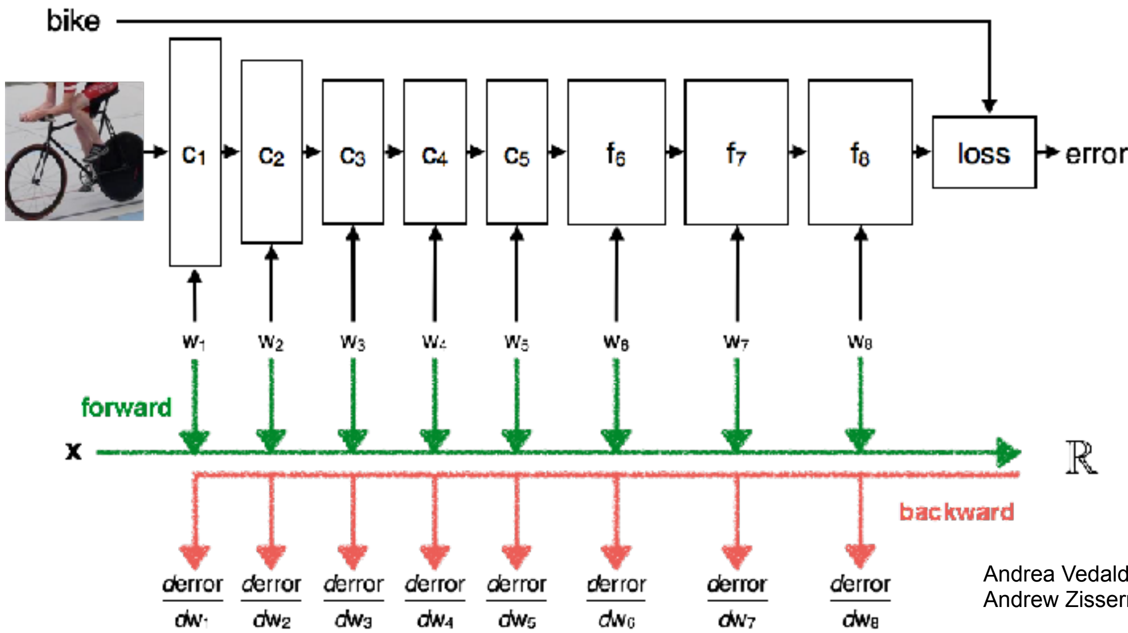
ReLU (rectified Linear Unit)



source: http://cs231n.github.io

Deep Neural Network

Backward pass …



Andrea Vedaldi and Andrew Zisserman

Score, Loss, Optimization?

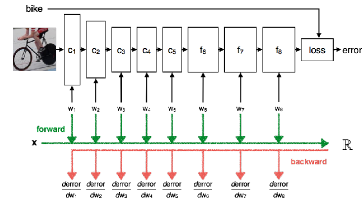
Deep Neural Network

Optimisation/Training:
start with random weights & iteratively refine to get lower loss

```
while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

- **miniB SGD**: gradient over batches. improve performance
- ***backpropagation***: gradient analytically using *chain rule*
  - gates communicate to each other what they need in order to increase the final output (score).
  - allows to optimize relatively arbitrary loss functions (to define all kinds of NN, e.g. CNN)

Deep Neural Network



Optimisation/Training in practice:

- Network = operations *chained* together, each one has a simple derivative
- Graph structure. The nodes implement the **forward() / backward() API**
  - ***forward pass function*** (local score from its *forward* input)
    = **score** (using layer weigths)

$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x)$$

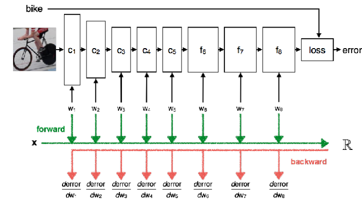
`f = q z`

  - ***backward pass function*** (local gradient from its *backwards* input)
    = *d_error / d_layer-weigths*

$$\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}$$

`df/dz = q`

  - **update** $W_1 = W_1 - lr \dfrac{\partial L}{\partial W_1}$; …



Deep Neural Network

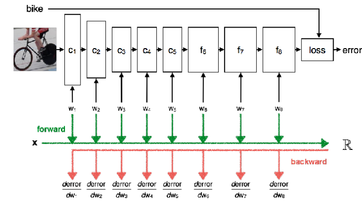
Optimisation/Training - Toy example of back propagation

$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x)$$

$$\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}$$

How do we do this?
Analytically would be just too long for "deep" sizes …
L is not directly a function of all W, but is a function of a function of a function, etc
—> **CHAIN RULE**

Deep Neural Network



Optimisation/Training - Toy example of back propagation

How do we do this?
Analytically would be just too long for "deep" sizes …

$$f(x, y, z) = (x + y)z$$

Chain Rule

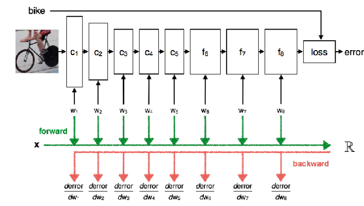
`f = q z`

`df/dq = z`

`...`

`df/dx = df/dq * dq/dx`

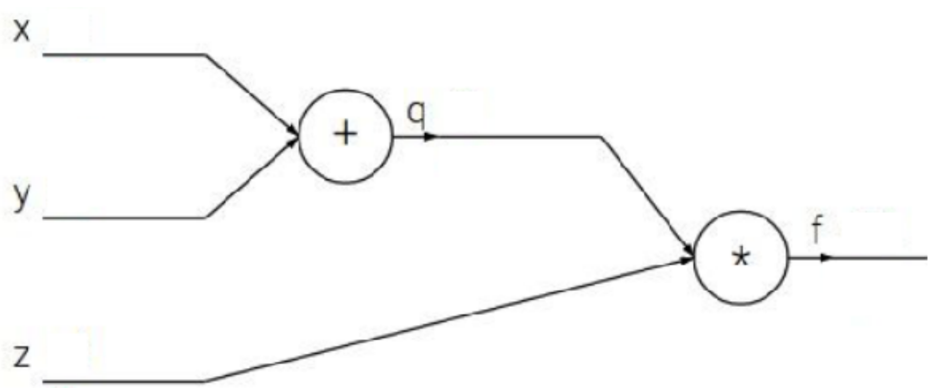
Deep Neural Network



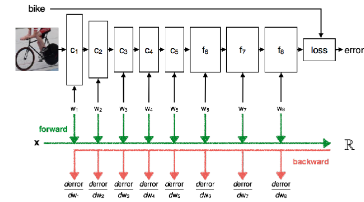
Optimisation/Training - Toy example of back propagation

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



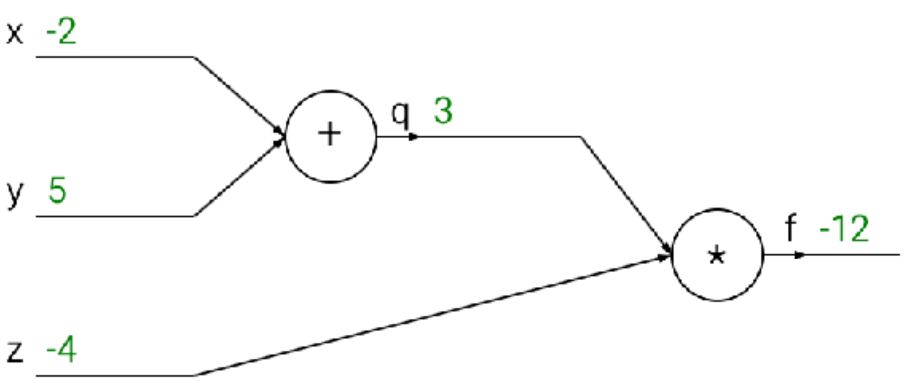
source: http://cs231n.github.io



Optimisation/Training - Toy example of back propagation

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

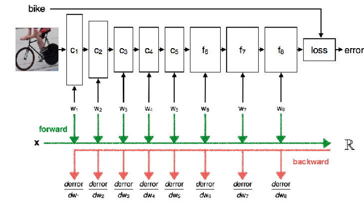
Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$

Upstream gradient

Local gradient

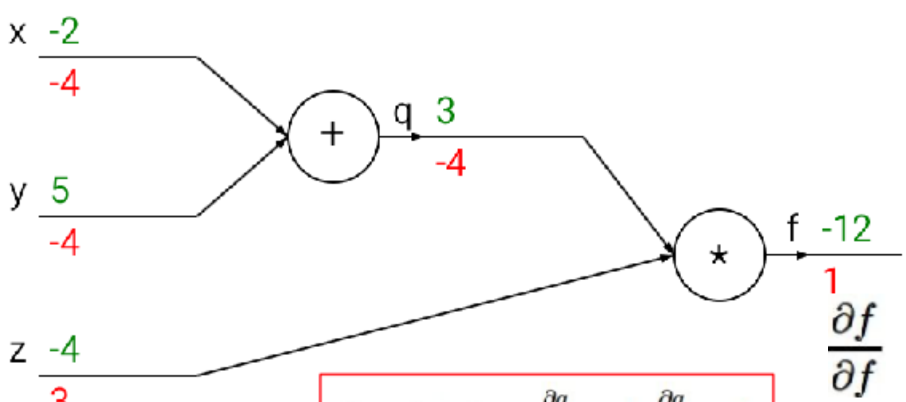
source: http://cs231n.github.io



Optimisation/Training - Toy example of back propagation

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain rule:

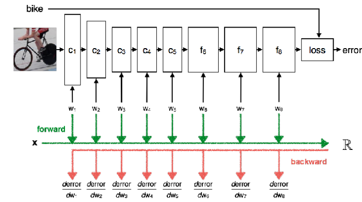
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

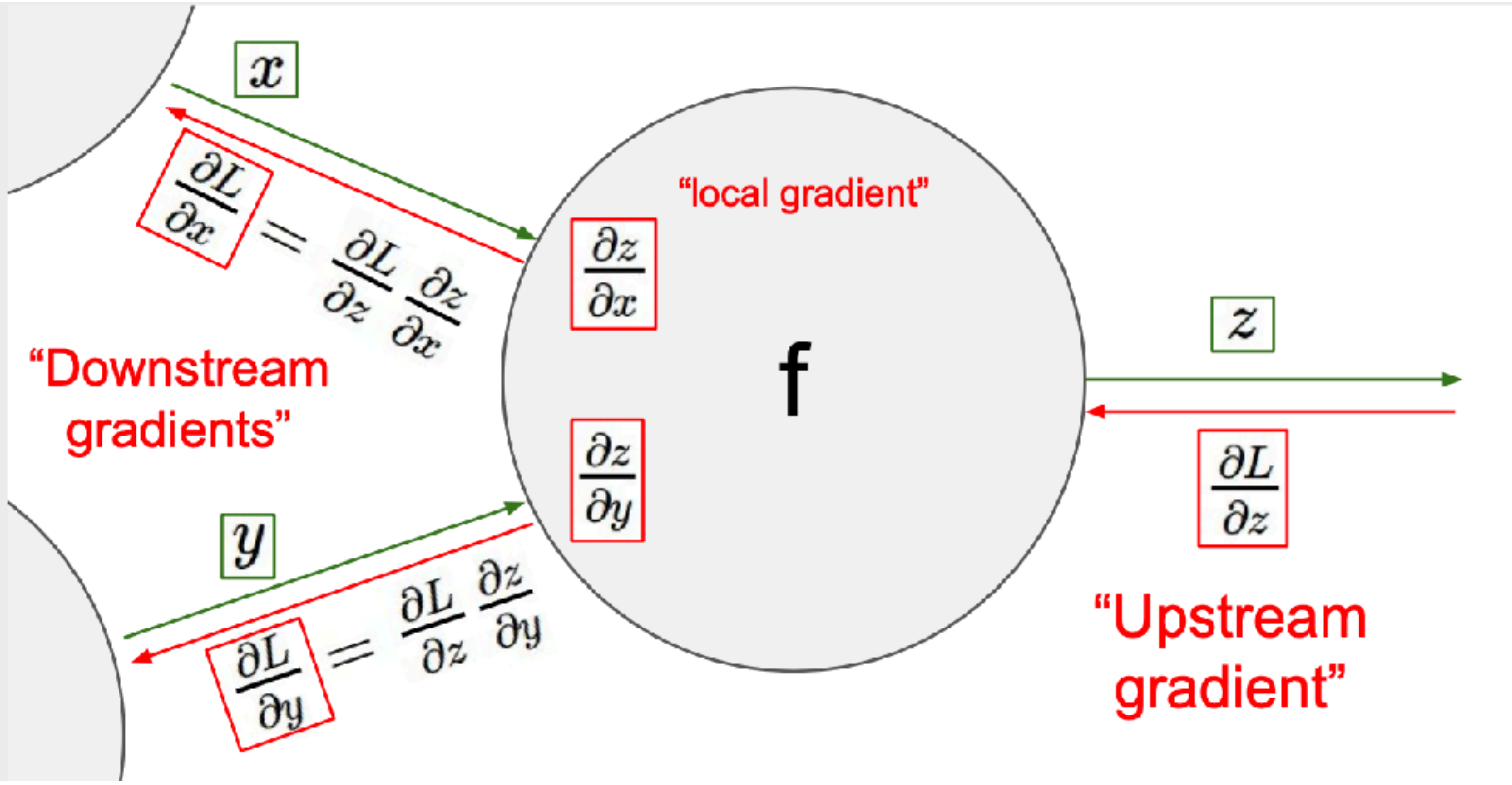
Upstream gradient

Local gradient

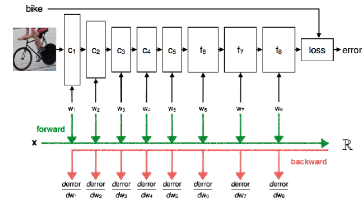
source: http://cs231n.github.io

Deep Neural Network





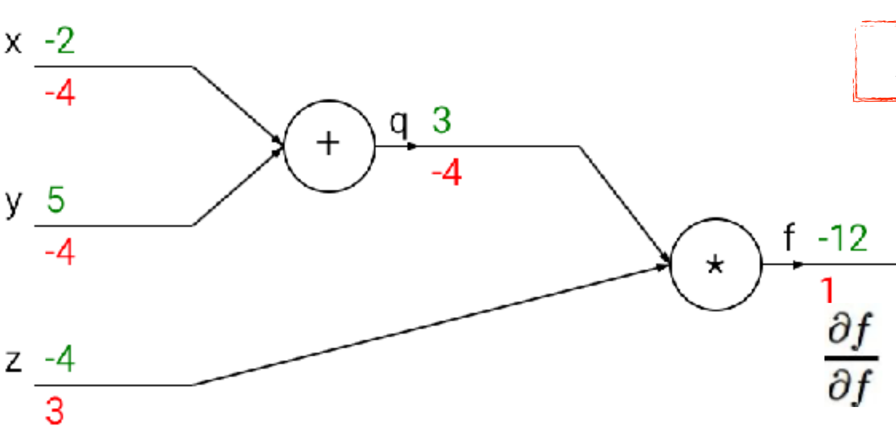
source: http://cs231n.github.io



Optimisation/Training - Toy example of back propagation

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



And what does this mean?

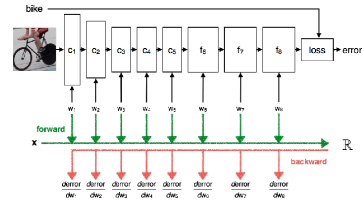
Weigths update:

```
w = w - lr * dw
```

$$W_1 = W_1 - lr \frac{\partial L}{\partial W_1} \quad ; \ldots$$

source: http://cs231n.github.io

Deep Neural Network



Optimisation/Training. Summary of simple common practices:

- **Normalize** input data
- **Initialize** weights. Not zero but small random numbers.
- **Batch normalisation**: forcing the activations throughout a network to take on a unit gaussian distribution at the beginning of the training
- Strong **regularization** - avoid overfitting (regularization strenght, dropout, augmentation, …)
- Different **loss functions** depending on the task
- **Hyper-parameters** tunning

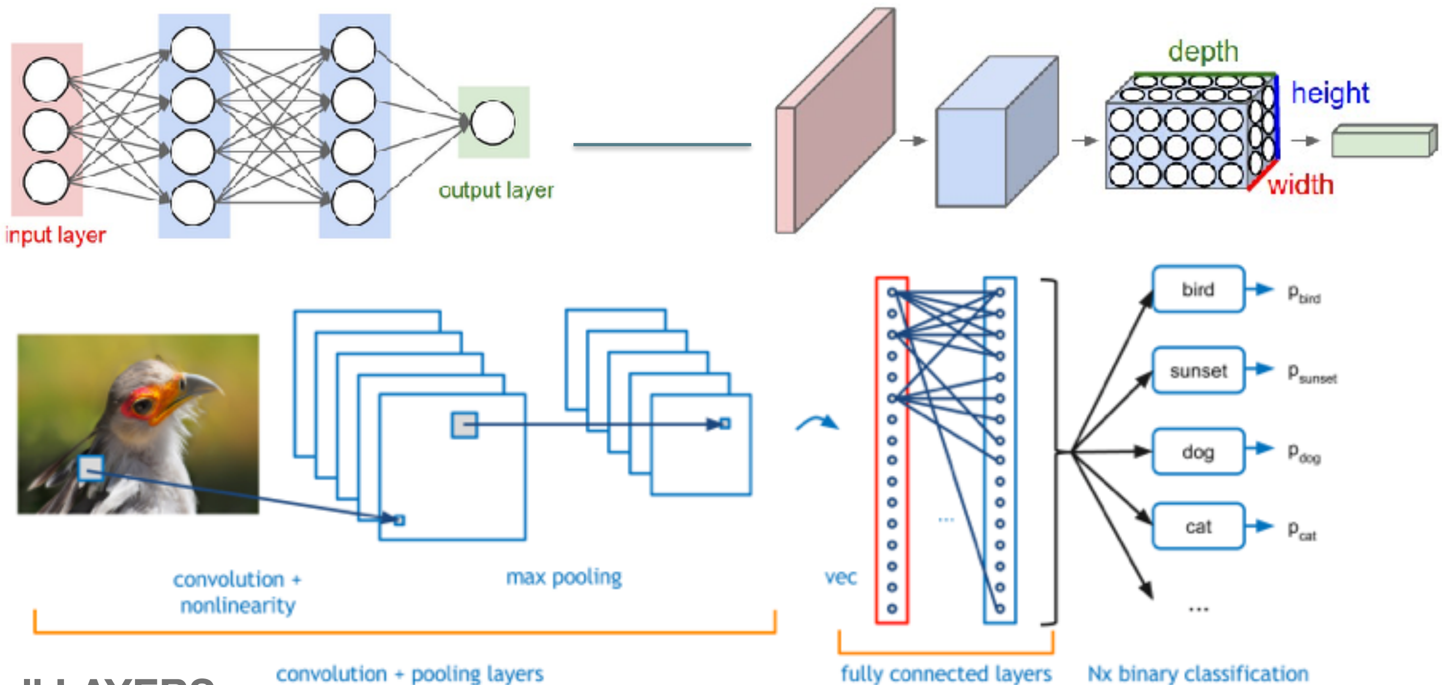
Next

- ~~What's deep learning?~~
- ~~Fundamentals of DL~~
  - ~~Review basic concepts~~
  - ~~NN and DNN~~
- **CNNs**

CNNs

- Convolutional Neural Networks (CNN)

DEMO online, CNN



NOT all LAYERS are the same

Fei-Fei, Karpathy, Johnson. Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu)
Evan Shelhamer, Jeff Donahue, Jon Long, Yangqing Jia, and Ross Girshick. *Deep Learning for Vision:* a Hands-On Tutorial

Bibliography - Resources for materials in this block

- Stanford online materials on
  Deep learning for Computer Vision (http://cs231n.stanford.edu)
  and Deep Learning (https://cs230.stanford.edu/)

- Ian Goodfellow, Yoshua Bengio, Aaron Courville,
  Deep Learning, MIT Press, 2016.
  http://www.deeplearningbook.org

TO-DO …

Have you all used COLAB?

- Lab 2 - **THIS WEEK (18 OCT , 20 OCT) —> AT A07 unless stated otherwise**
- PLEASE **have your computer ready with Tensorflow2+Keras (ideally with GPU available) AND/OR we will use Google COLAB**
- Recommended COLAB tutorial if you have not used it much before:
  https://colab.research.google.com/notebooks/intro.ipynb
  https://colab.research.google.com/notebooks/basic_features_overview.ipynb
  Loading data: Drive, Sheets, and Google Cloud Storage

***ASSIGNMENT* BEFORE YOUR LAB:**

1. Pick 5 to 10 classes from one of these datasets (do not take all images from each class if you don't have space)
   - https://www.kaggle.com/kmader/food41/version/5#
   - http://www.robots.ox.ac.uk/~vgg/data/pets/
   - http://www.robots.ox.ac.uk/~vgg/data/flowers/
   - **any other dataset you have?**
2. Put them in folders like ————————————————>
3. Upload to Google Drive if you plan to use COLAB

```
data/
    dogs/
        dog001.jpg
        dog002.jpg
        ...
    cats/
        cat001.jpg
        cat002.jpg
        ...
```