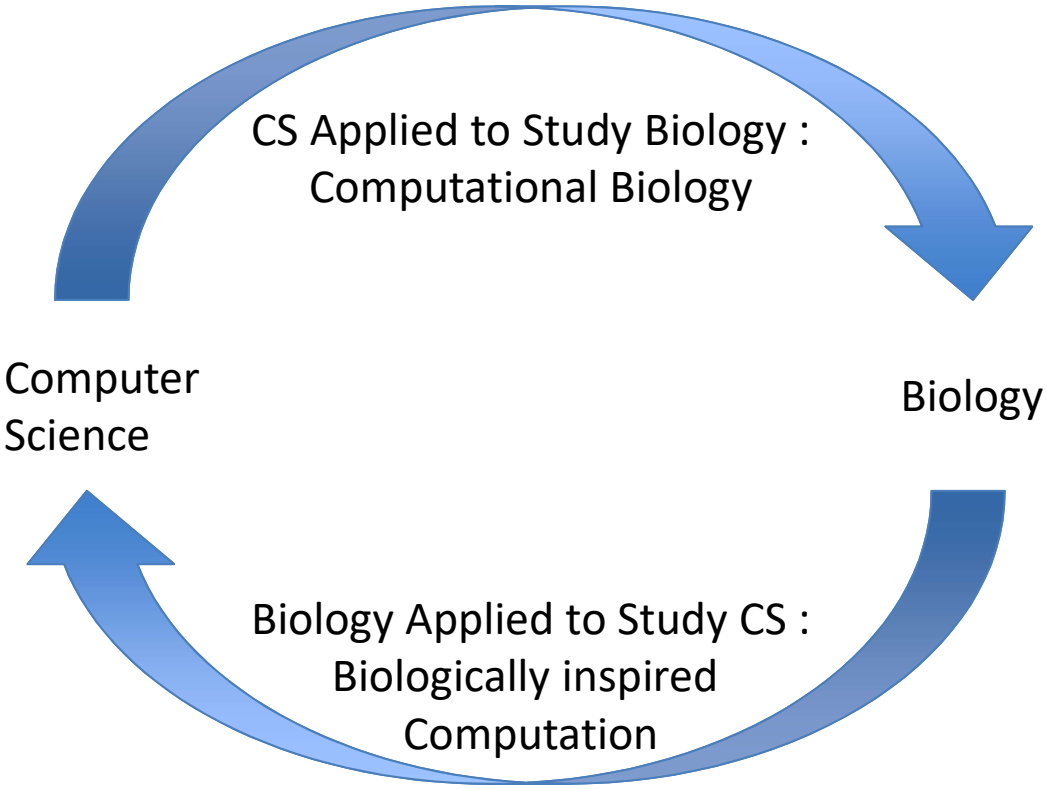
Deep learning in bioinformatics

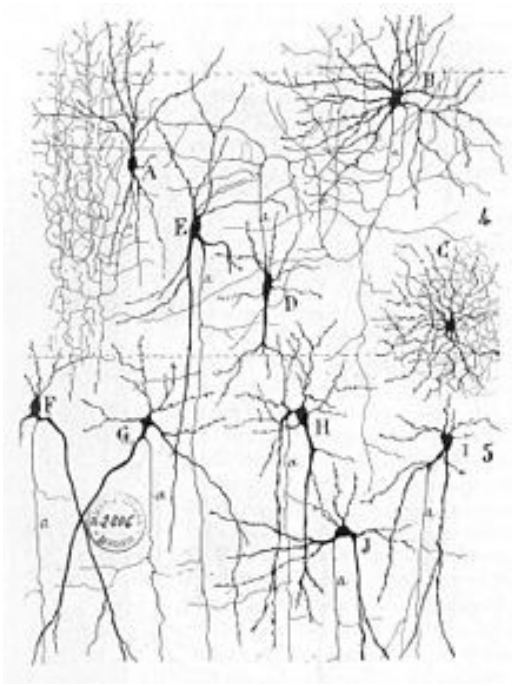
Pradipta Ray
BIOL 6385 / BIOL 6389
(pedagogical structure based on
Vincent Vanhoucke's course)

Transfer learning : Computational biology vs Biologically inspired computation



Reticular vs cellular - connectionist models of brain

Joseph von Gerlach



Wikipedia

The brain is composed of many cells (neurons) each with multiple connections.

Santiago Ramon y Cajal

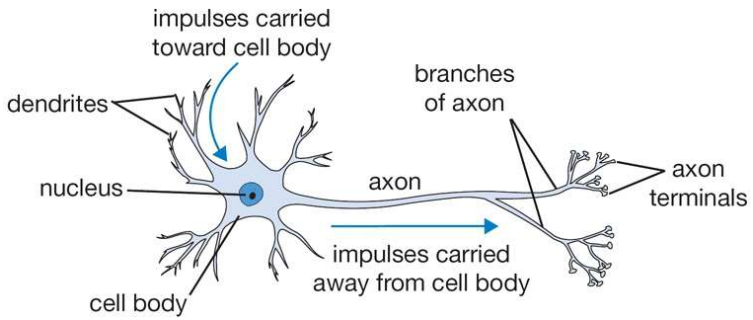
Camillo Golgi

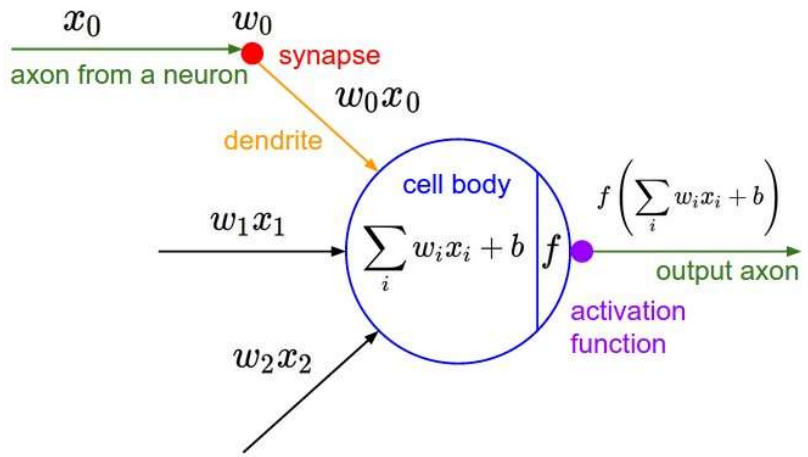
The brain is a single, continuous network flow, with branching connections between regions

Neurons that fire together wire together : electrical conductivity identified by Berger is achieved by dedicated circuits.

Donald Hebb

Perceptron





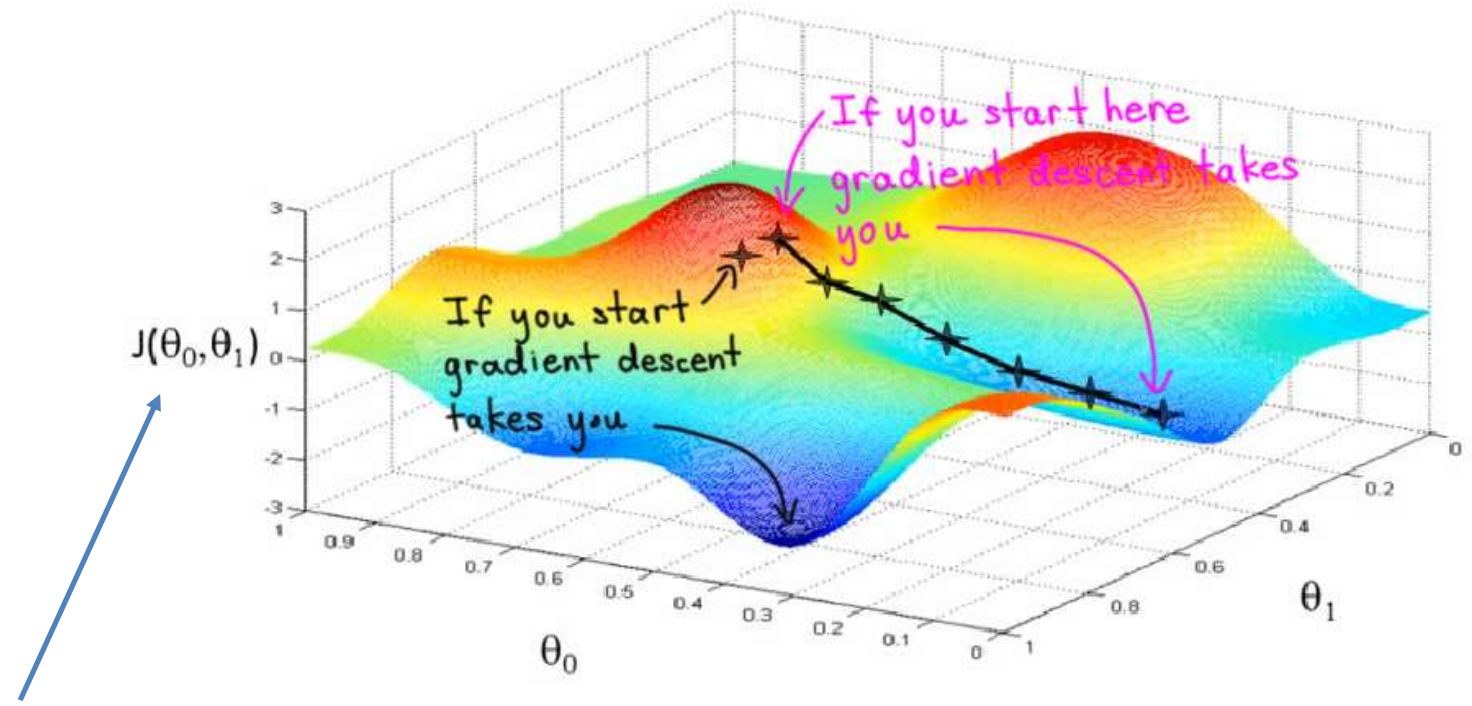
- Thresholded, weighted sum of inputs : output 0 or 1 based on superiority test

- Predictor / input variables : Many, real-valued
- Response variable : discrete (binary)

What kind of a learning problem is this ?

Estimating perceptron parameters

- Gradient descent



What function should we optimize ?

quinnliu/machineLearning

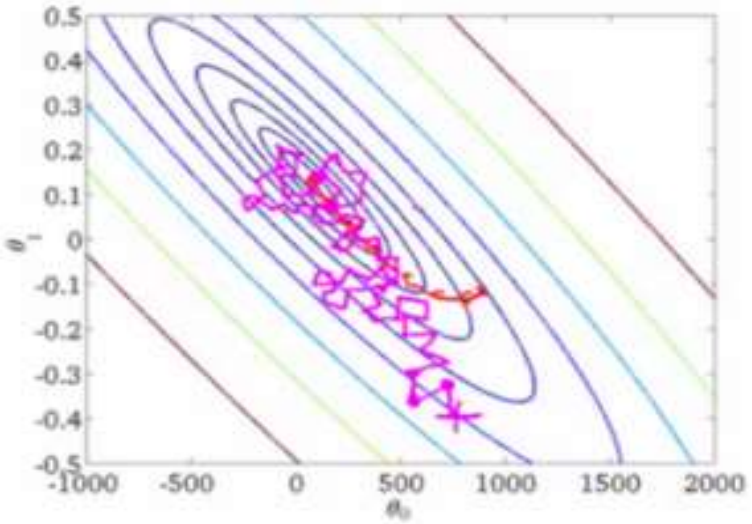
Stochastic gradient descent

Offline, batch and online learning

Adaptive learning rate

$$\text{For } j = 1, 2, \ldots$$

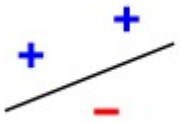
$$\mathbf{x}^{(\mathbf{j}+\mathbf{1})} = \mathbf{x}^{(\mathbf{j})} - \gamma \nabla f_i(\mathbf{x}^{(\mathbf{j})})$$



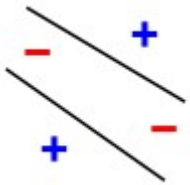
Instead of using entire dataset , uses batches of data for each step

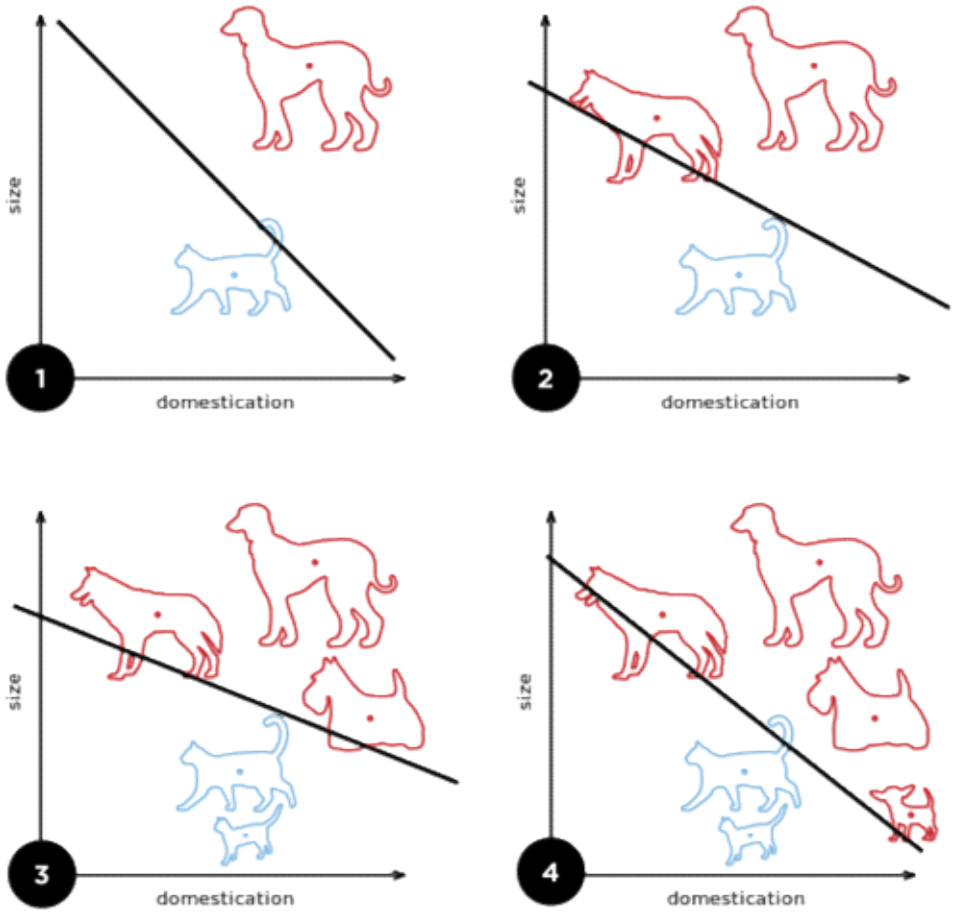
Problems with perceptrons

- Can only classify linearly separable datasets



versus

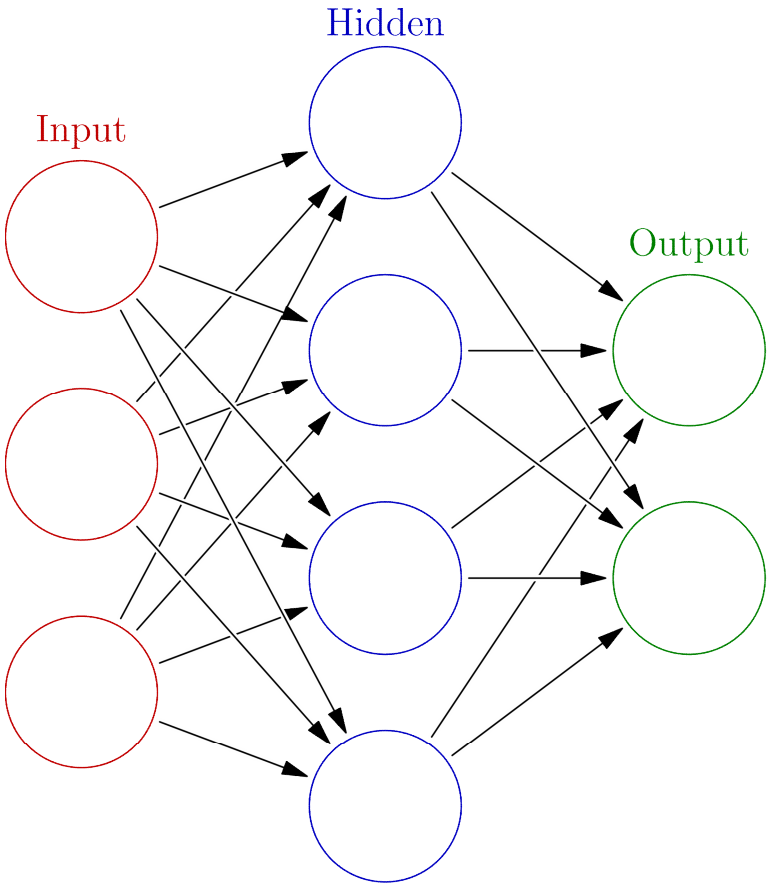




Wikipedia

Neural Networks

- Layered network of perceptrons
- Able to perform non-linear separation for classification
- Question : how to train it ?



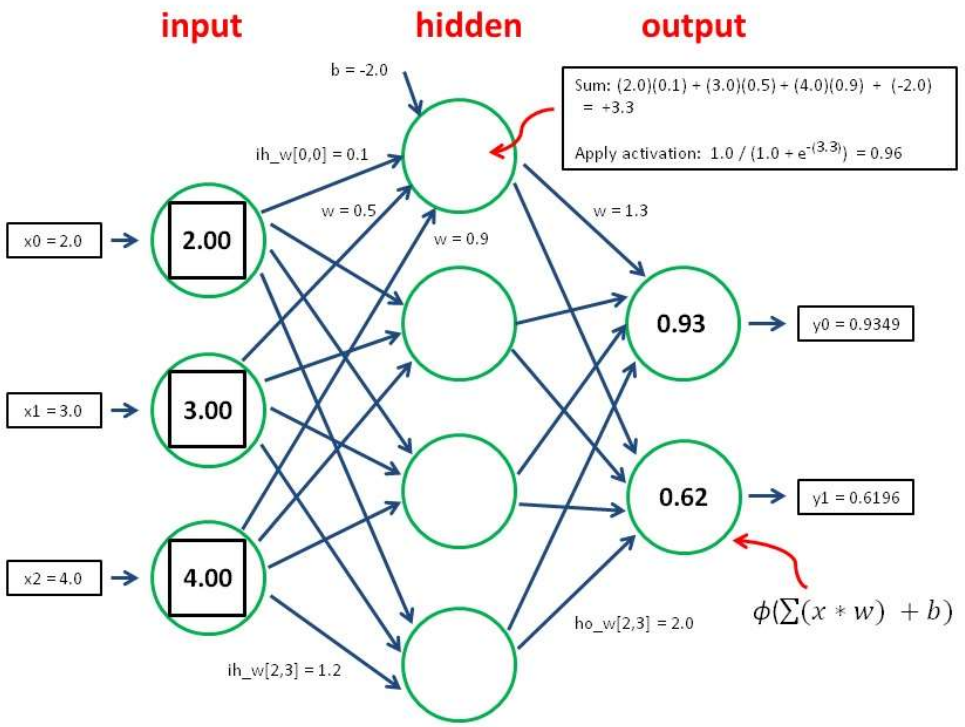
Wikipedia

A case in action

NB

- There can be multiple outputs
- The output layer need not be thresholded
- Topology :

  feedforward vs recurrent



James McCaffrey

Backpropagation

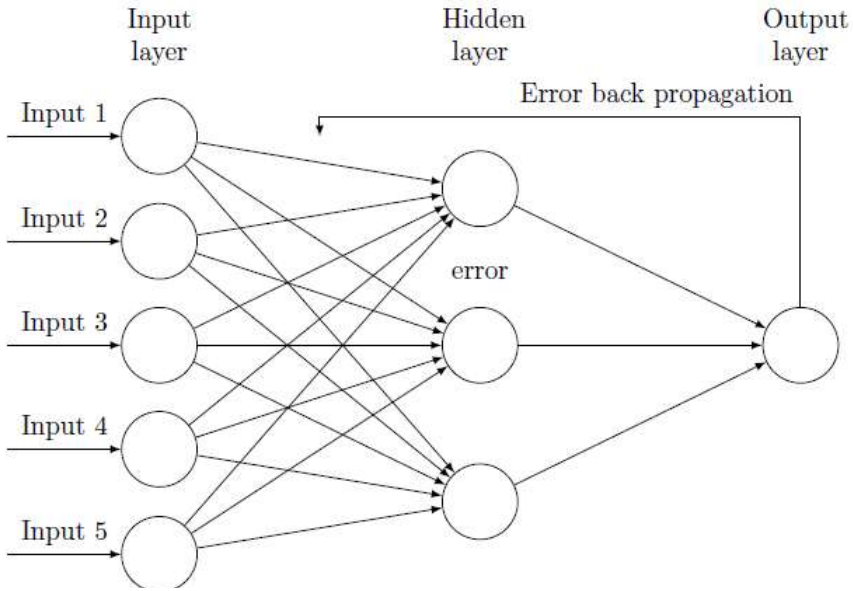
- SGD through layers : how well does it work ?

Wikipedia

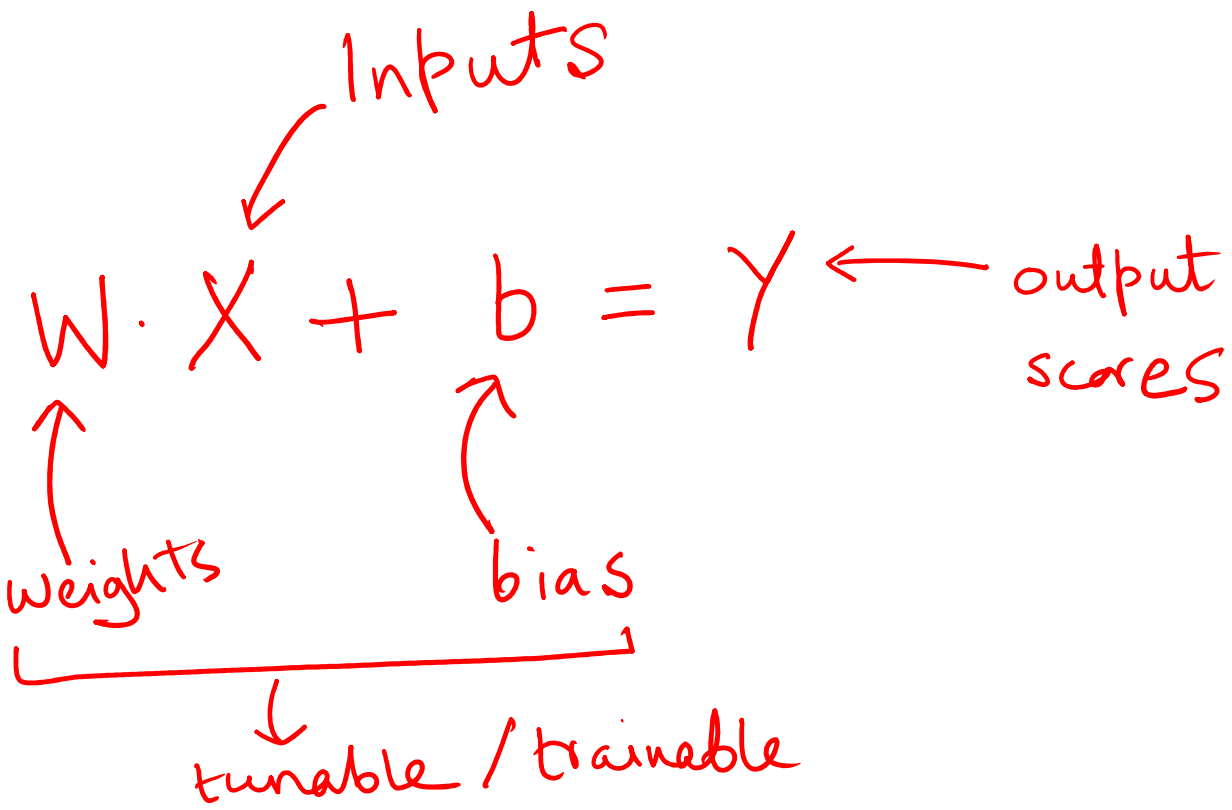
```
initialize network weights (often small random values)
do
   forEach training example named ex
      prediction = neural-net-output(network, ex)  // forward pass
      actual = teacher-output(ex)
      compute error (prediction - actual) at the output units
      compute Δw_h for all weights from hidden layer to output layer  // backward pass
      compute Δw_i for all weights from input layer to hidden layer   // backward pass continued
      update network weights // input layer not modified by error estimate
until all examples classified correctly or another stopping criterion satisfied
return the network
```

- Works well in theory & practice
  - "Ripple" backwards

C M Hughes

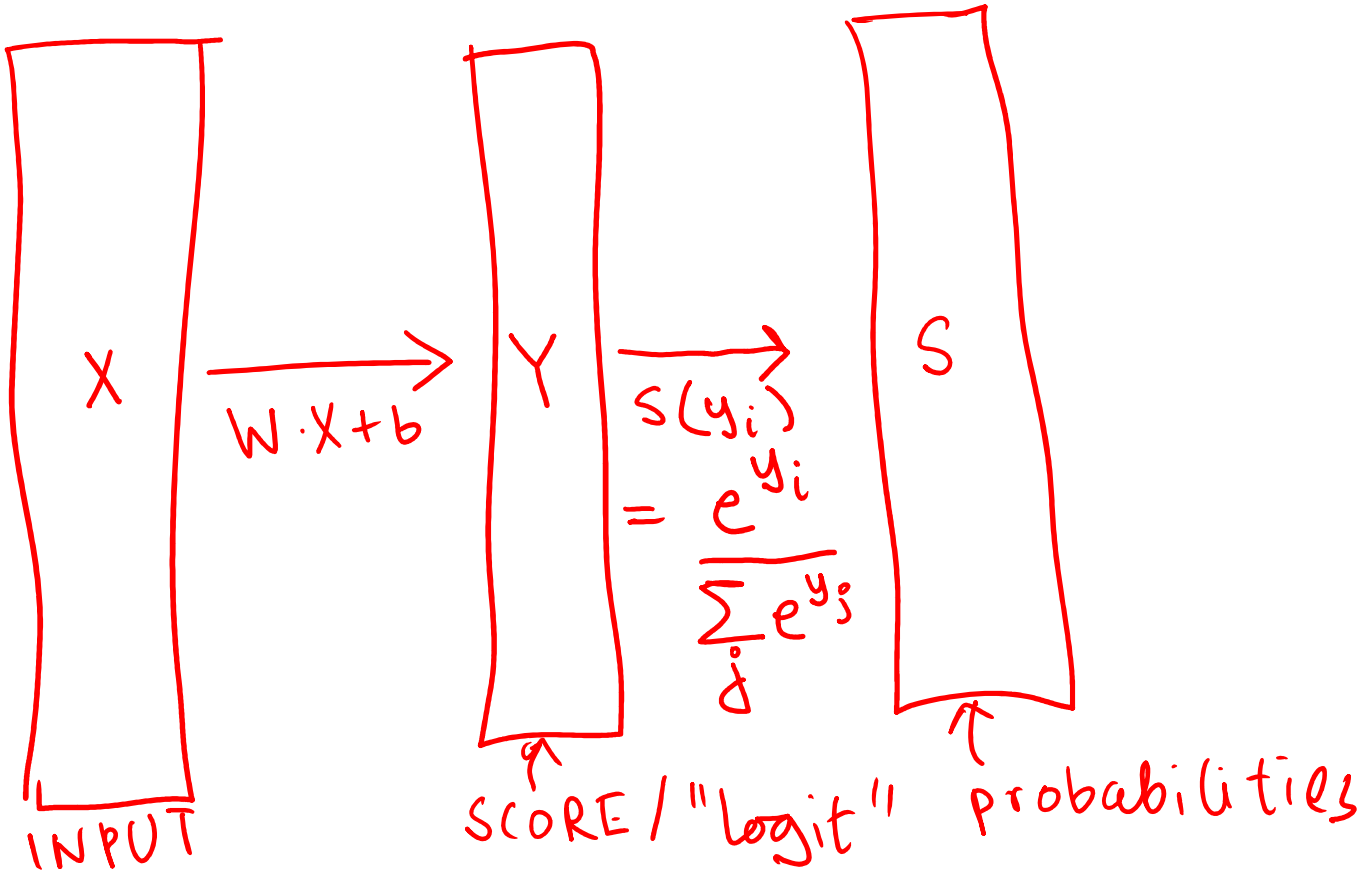


Logistic classifier



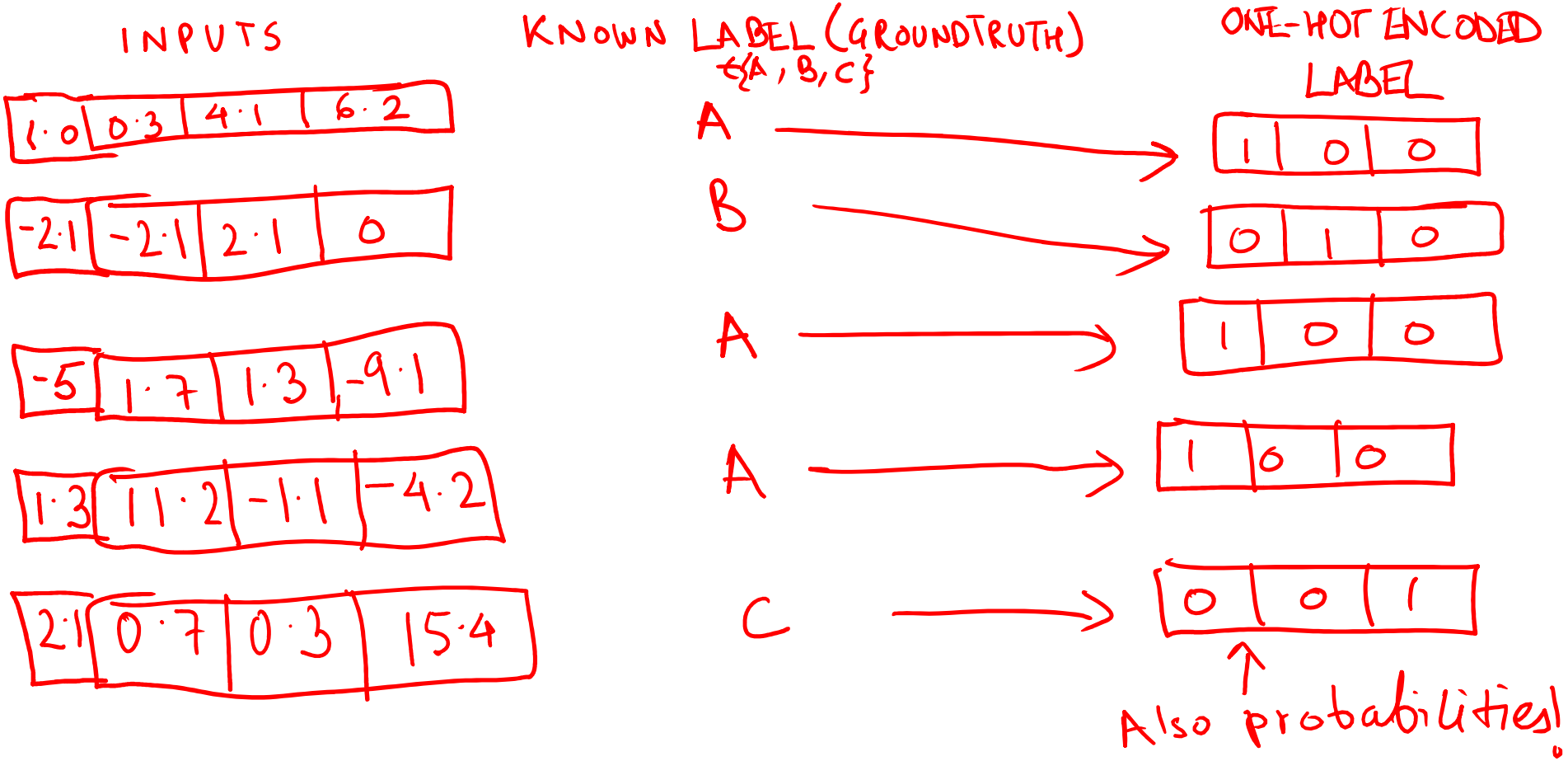
Logits to probabilities

- Converting scores to probabilities



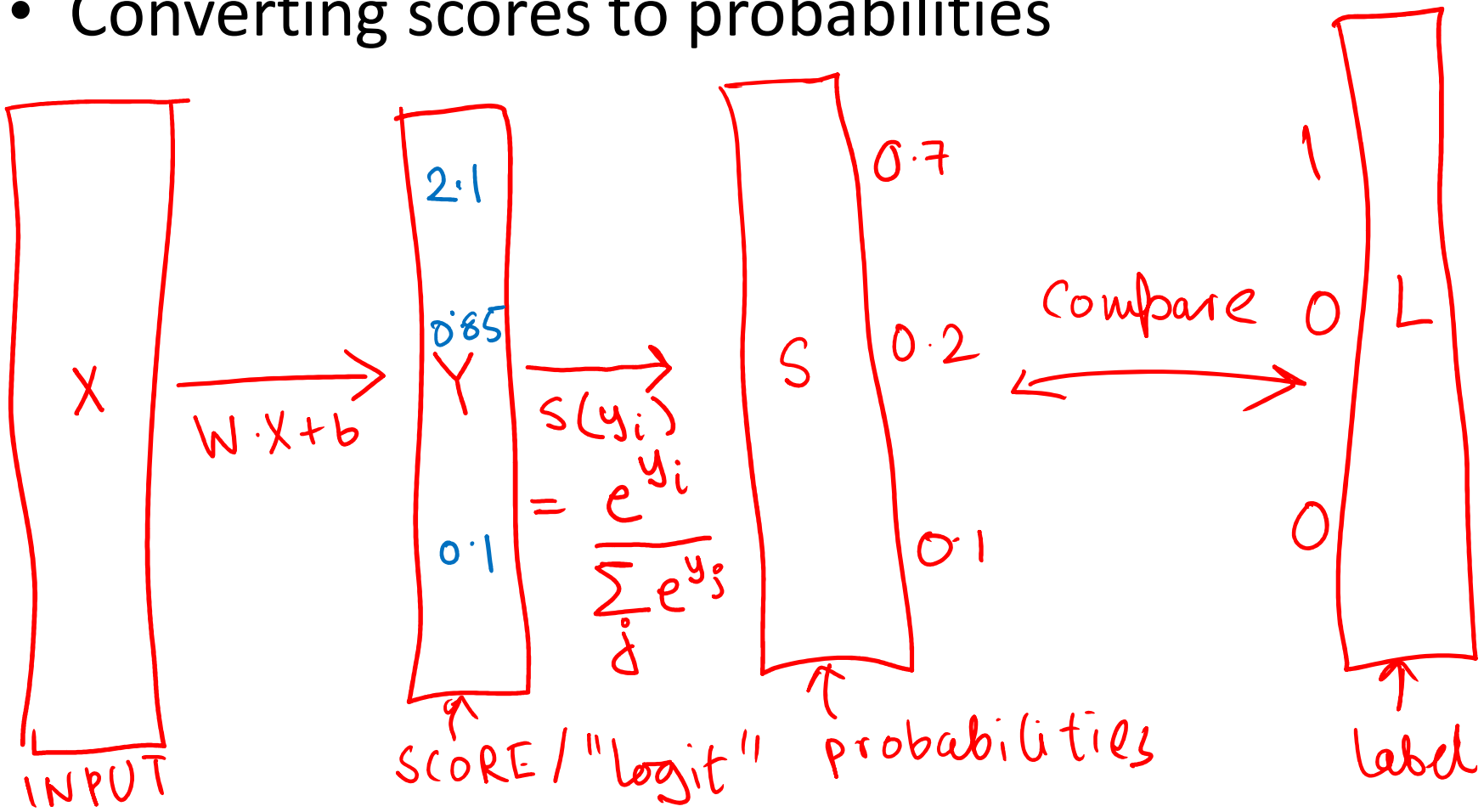
One-hot encoding

- A useful representation of the class label



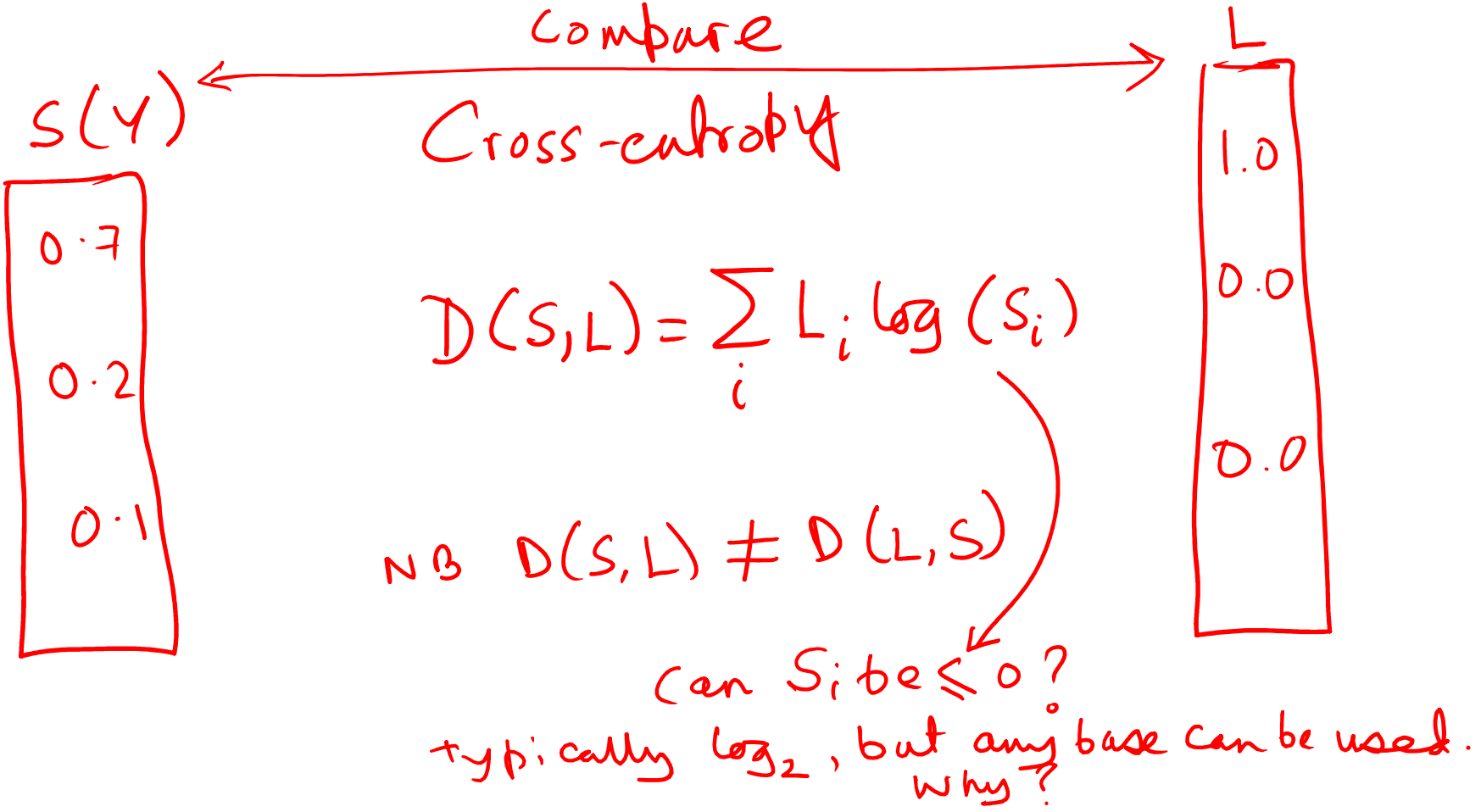
How good is the prediction ?

- Converting scores to probabilities



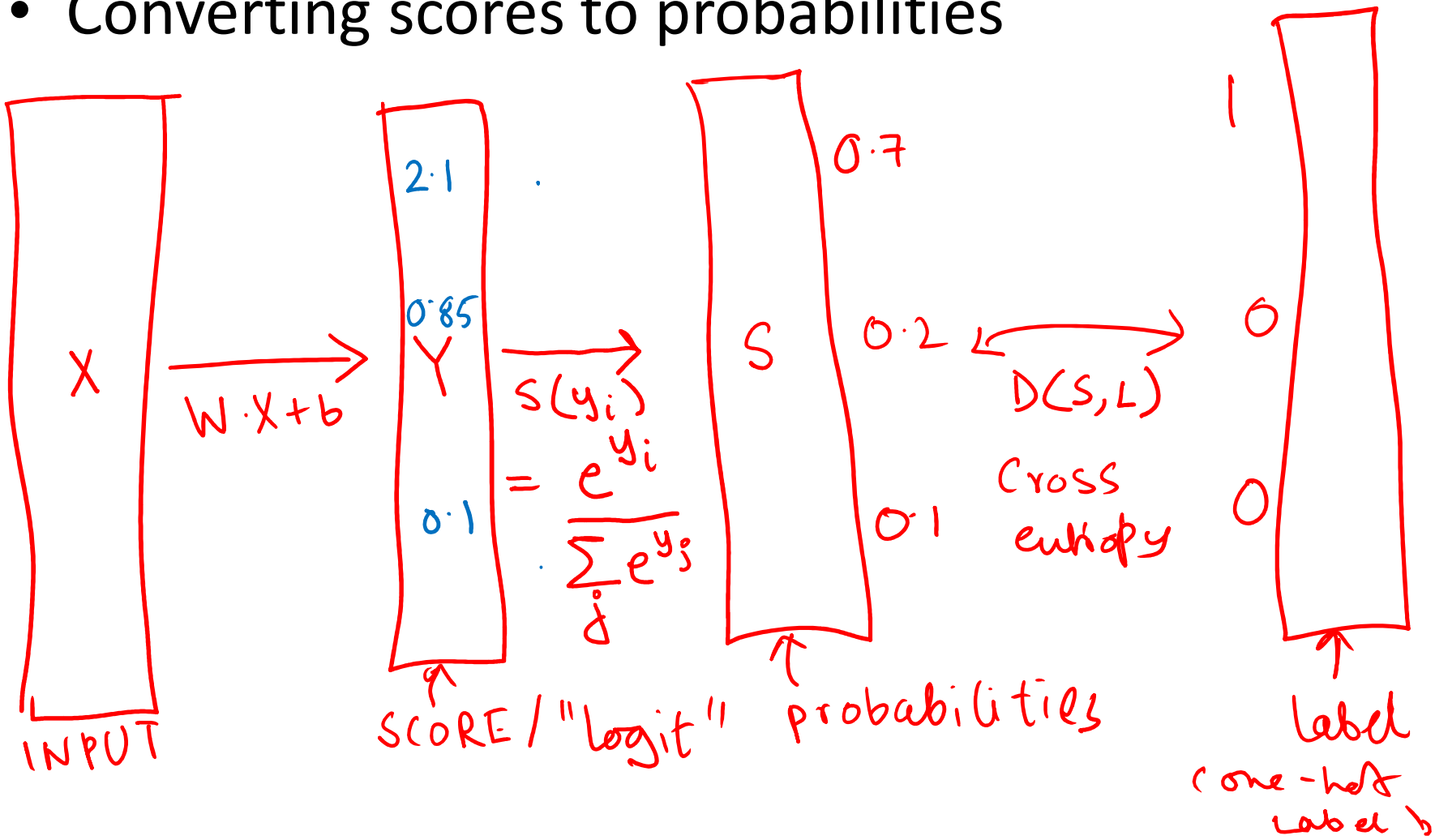
How good is my prediction ?

- Asymmetric : not a proper distance function



Putting it together : Multinomial logistic classifier

- Converting scores to probabilities



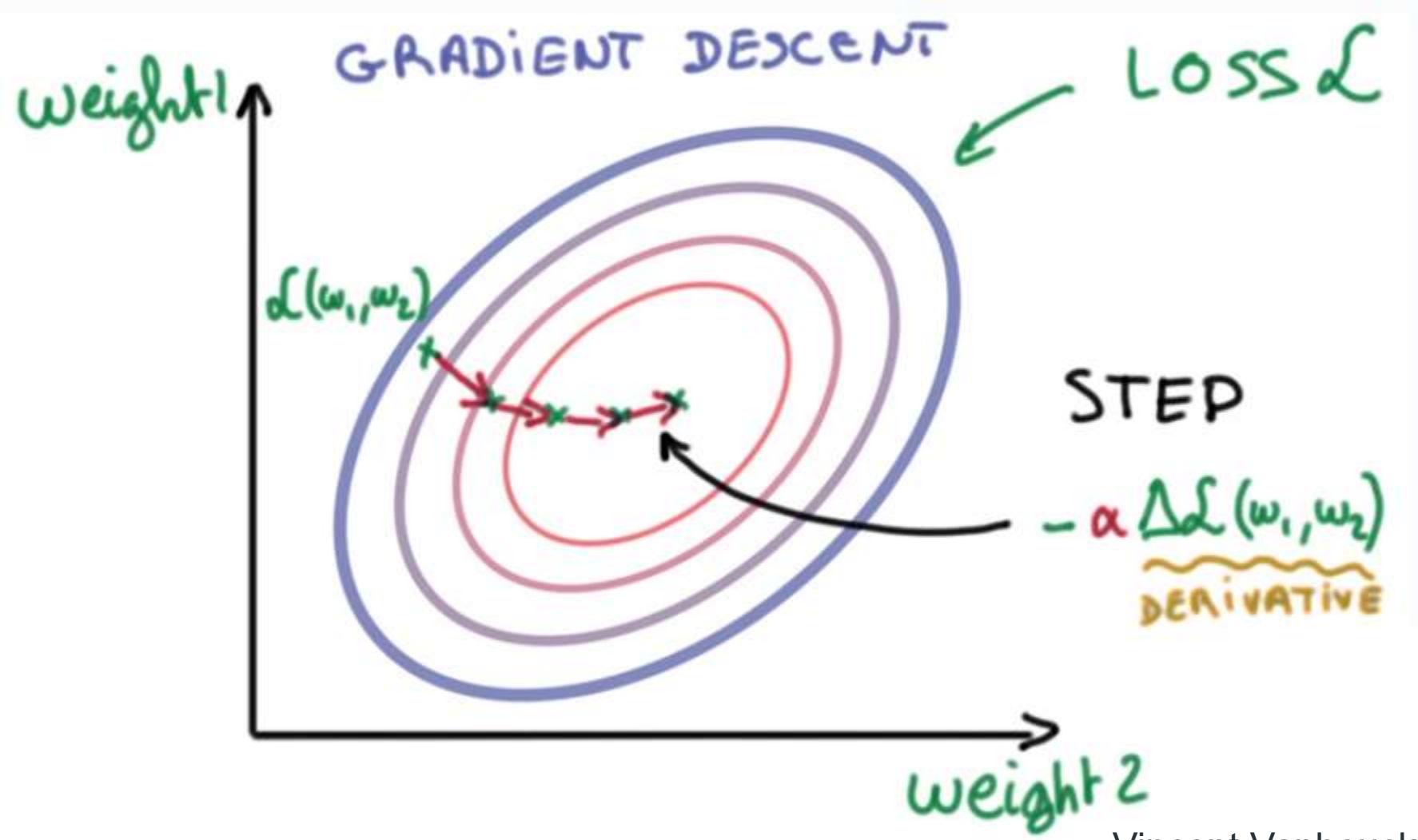
Loss function

- How well are we doing on the entire training data ?

$$L = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$$

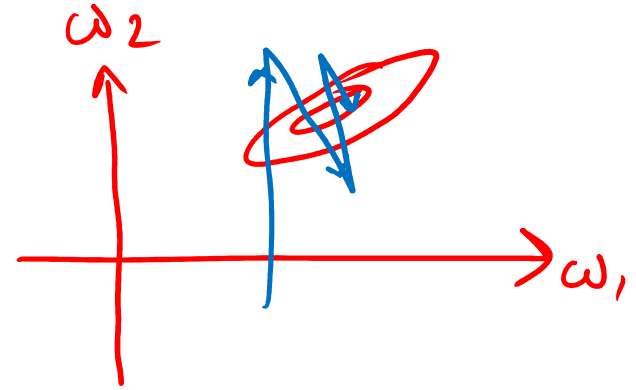
cross entropy averaged over entire training data

Gradient descent

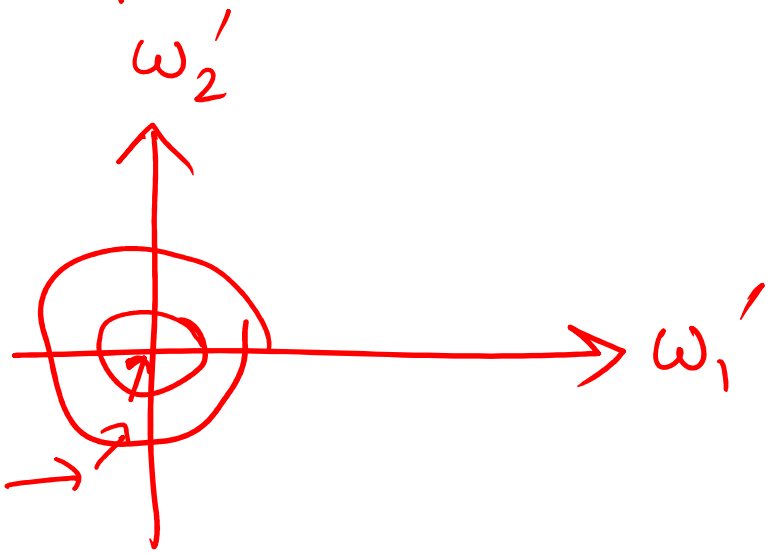


Vincent Vanhoucke

Transform your variables



BADLY CONDITIONED WEIGHTS



WELL CONDITIONED WEIGHTS

$\omega_i \rightarrow \omega_i'$

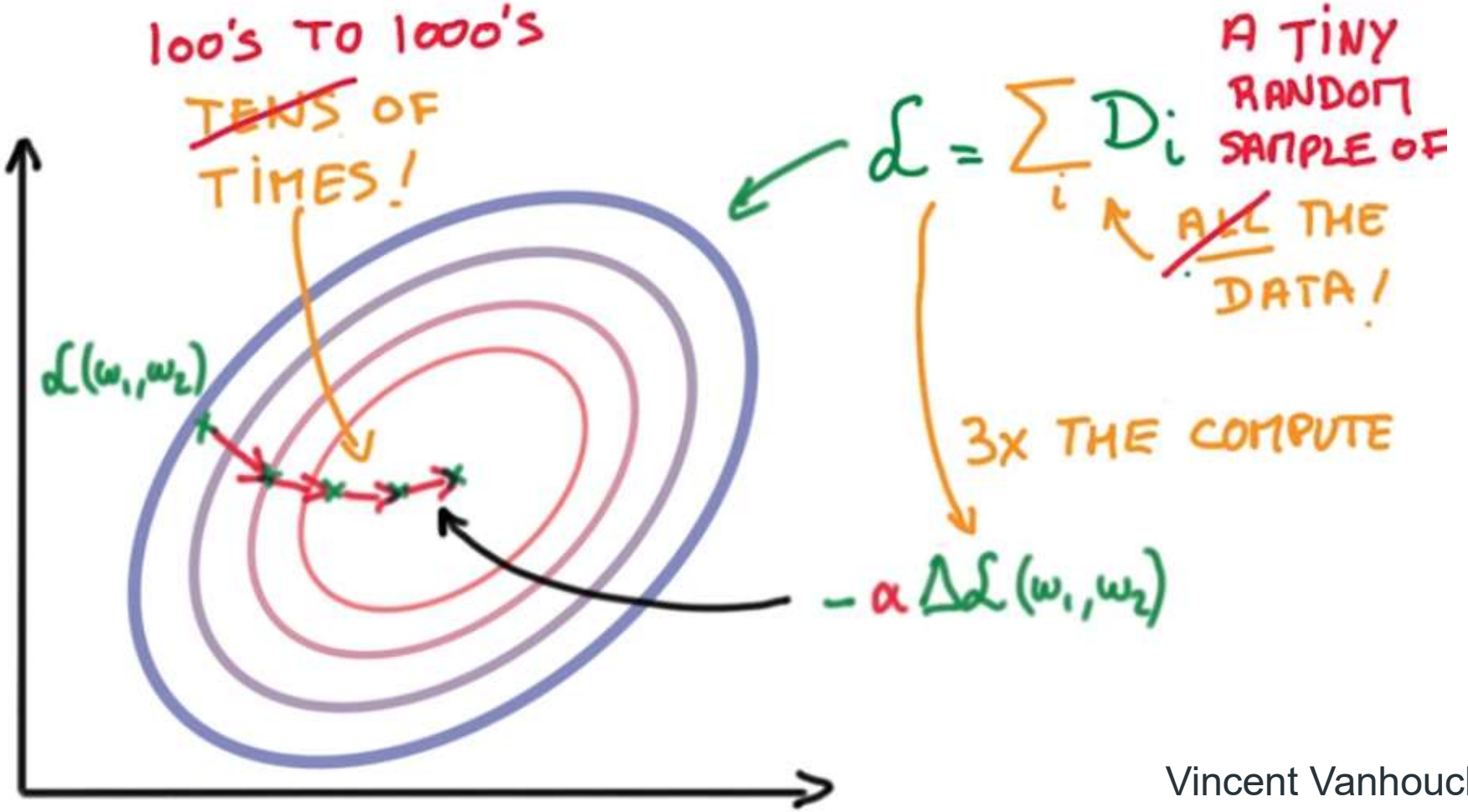
s.t.

$\overline{\omega_i} = 0$

$V(\omega_i)$'s are the same

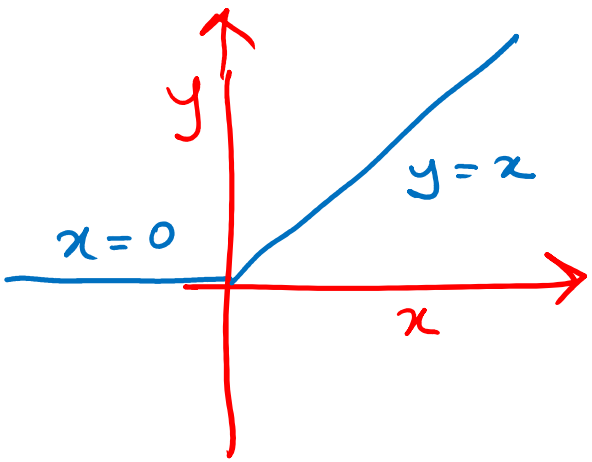
Still doesn't scale with data

- Stochastic gradient descent to the rescue

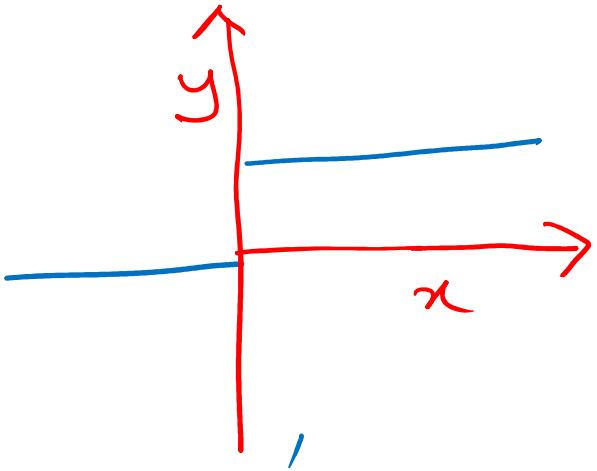


A simple way to introduce nonlinearity

- Piecewise linear function, well behaved derivative (catch : not defined at x = 0)



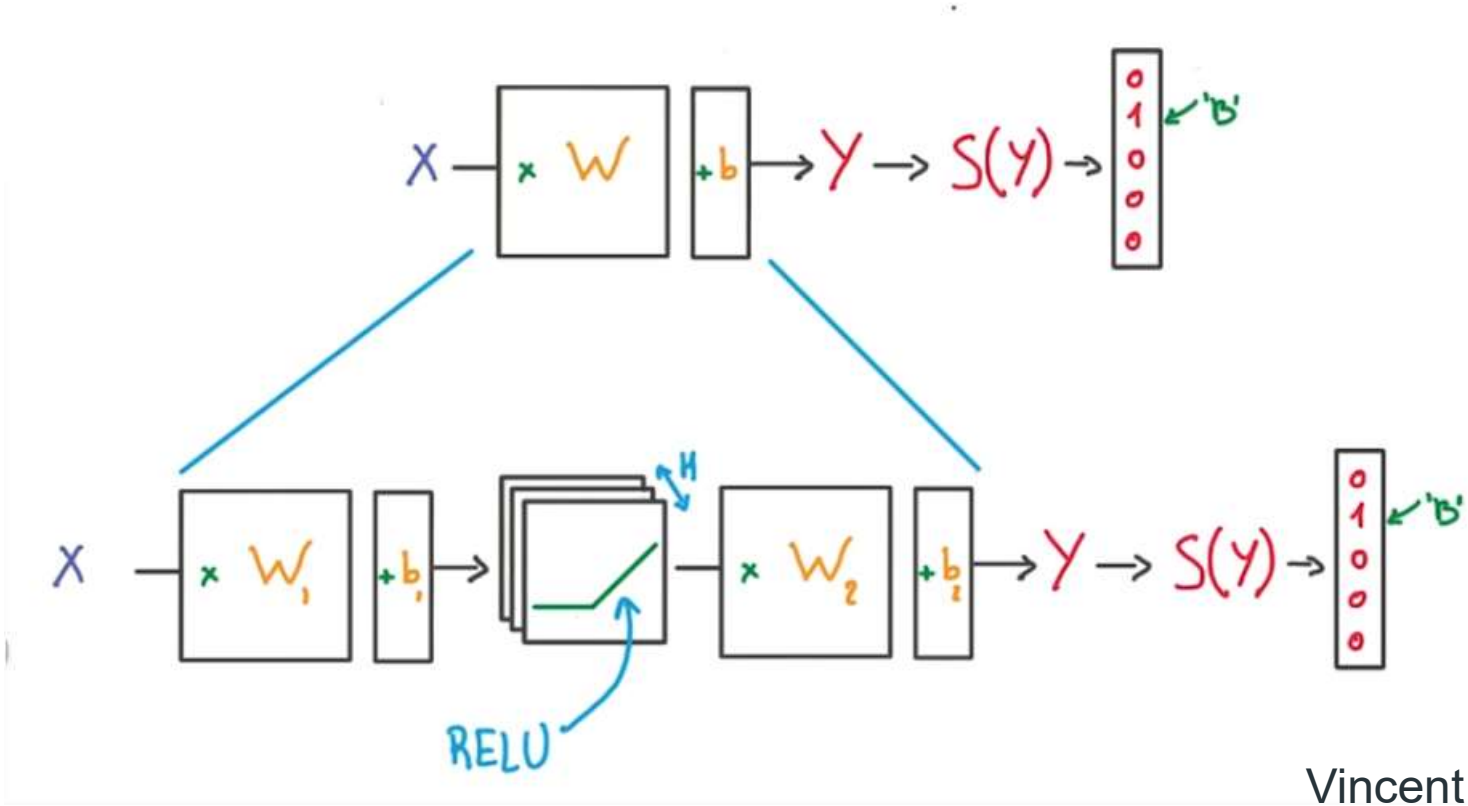
f(x)



f'(x)

Adding nonlinearity to the system

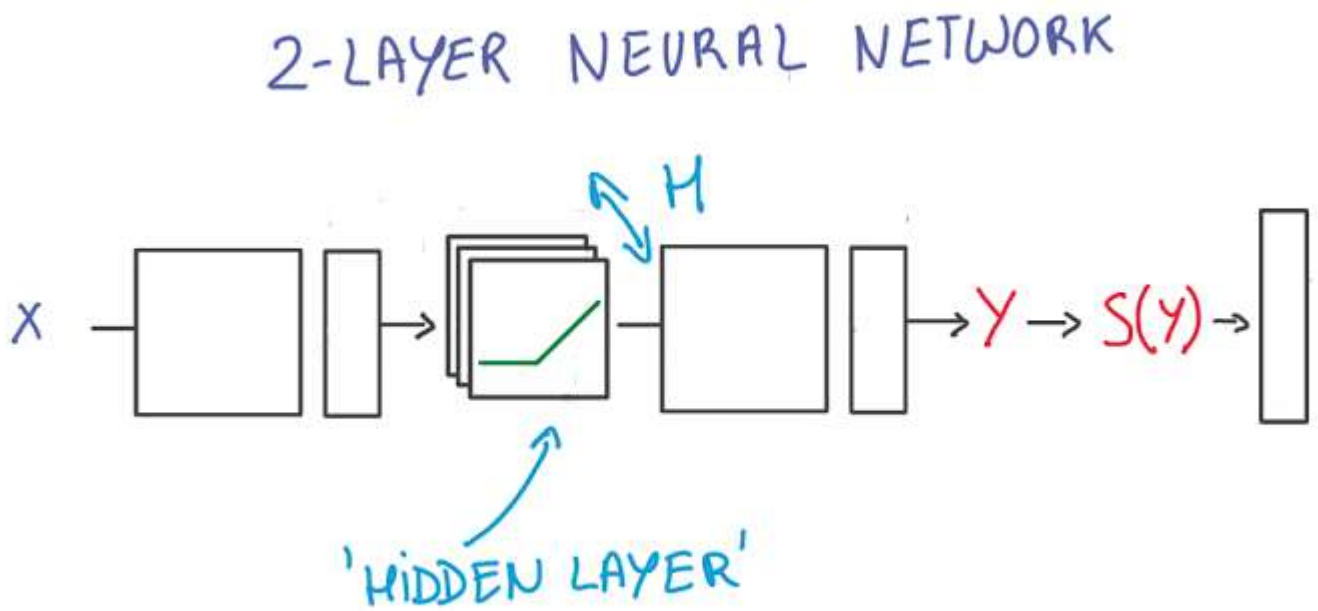
- Introduce RELUs as a layer before calculating the output : we have a neural network



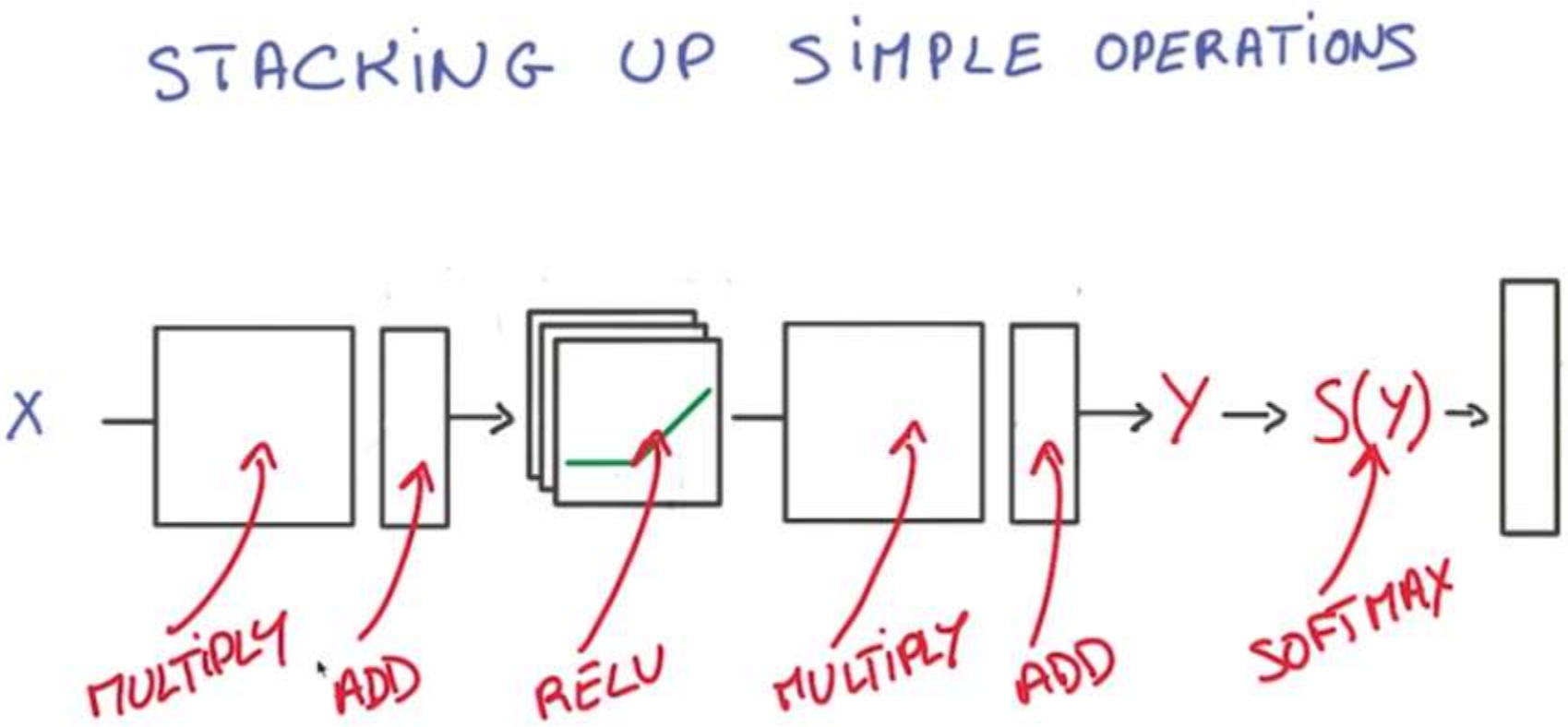
Vincent Vanhoucke

Traditional neural network

- One hidden layer



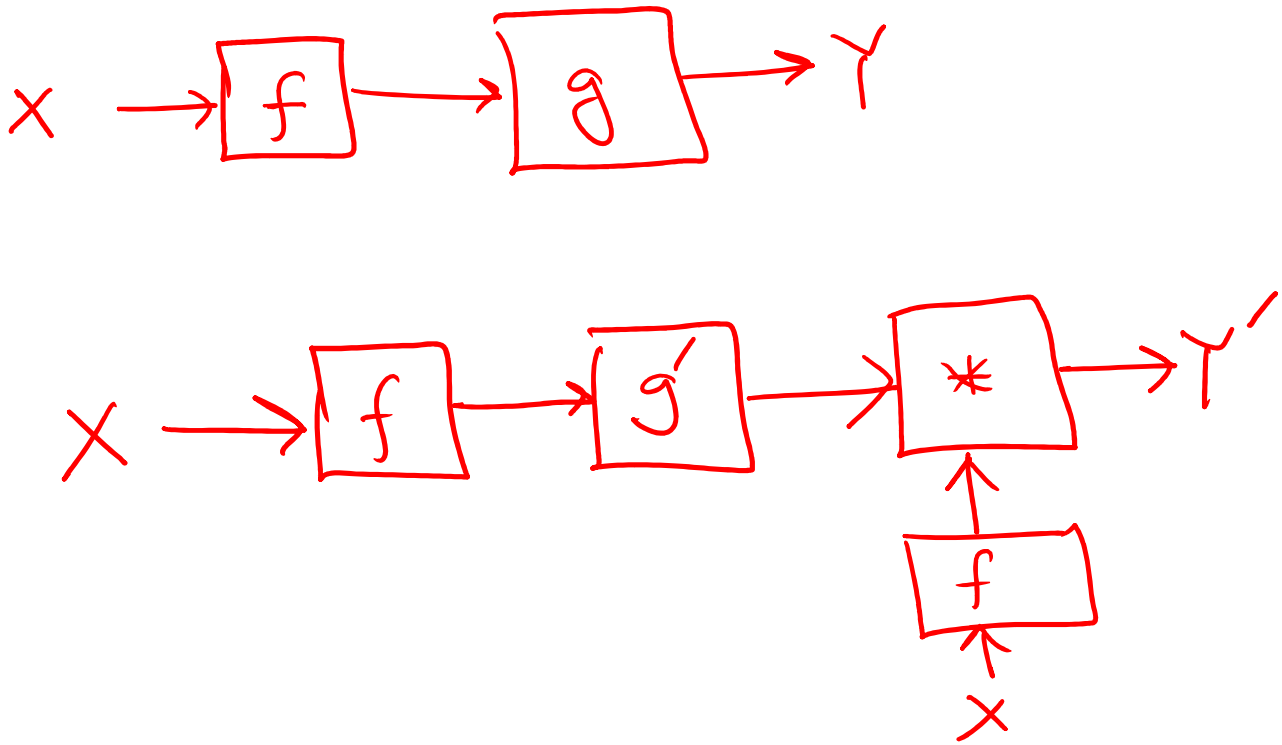
Step back



Vincent Vanhoucke

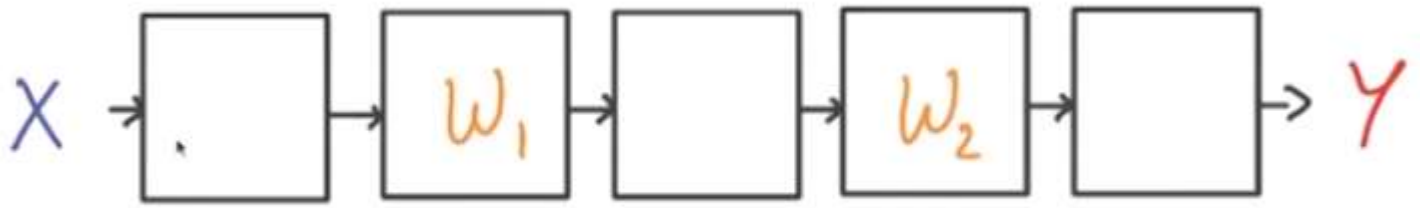
Chain rule

$$[g(f(x))]' = g'(f(x)) \times f'(x)$$



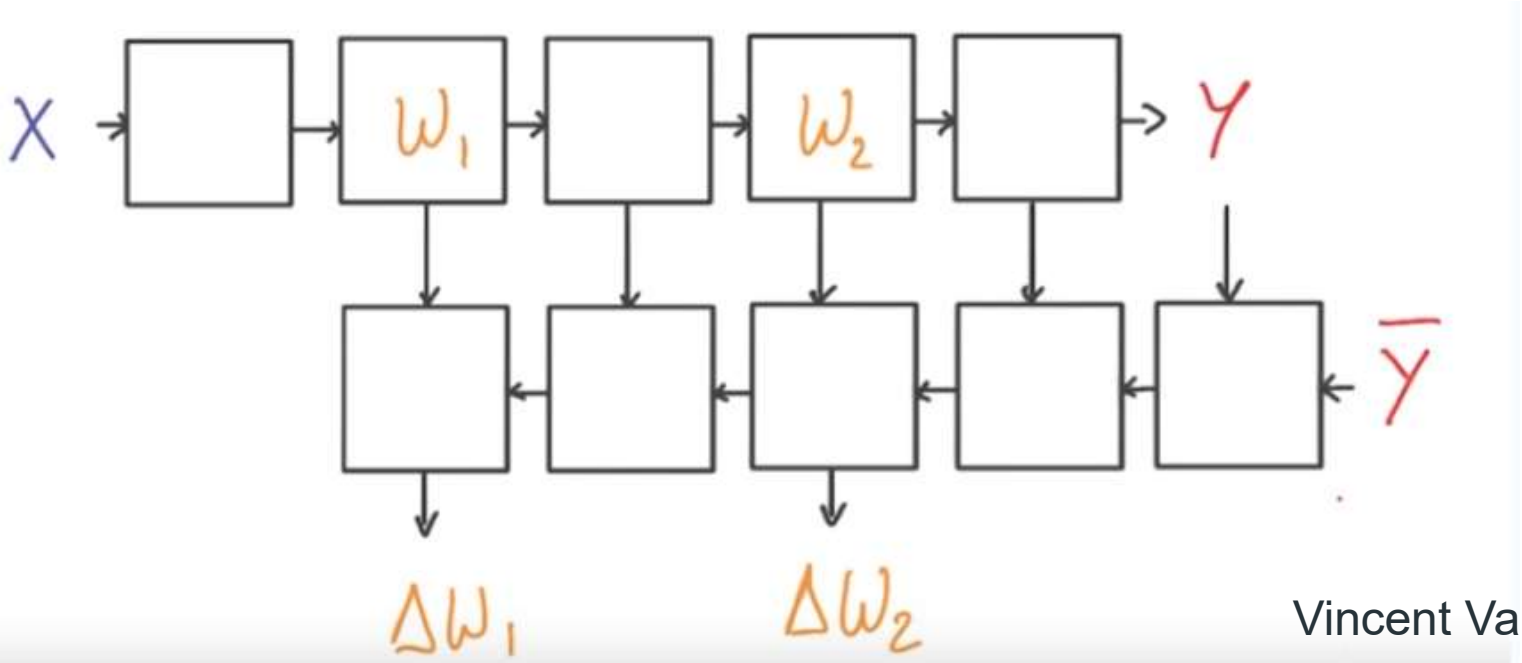
How would it work in practice ?

- Consider your framework as a pipeline of transformations transforming input X into output Y
- Some transformations are parameterized (eg. matrix multiplications) and some are not (eg. ReLu)

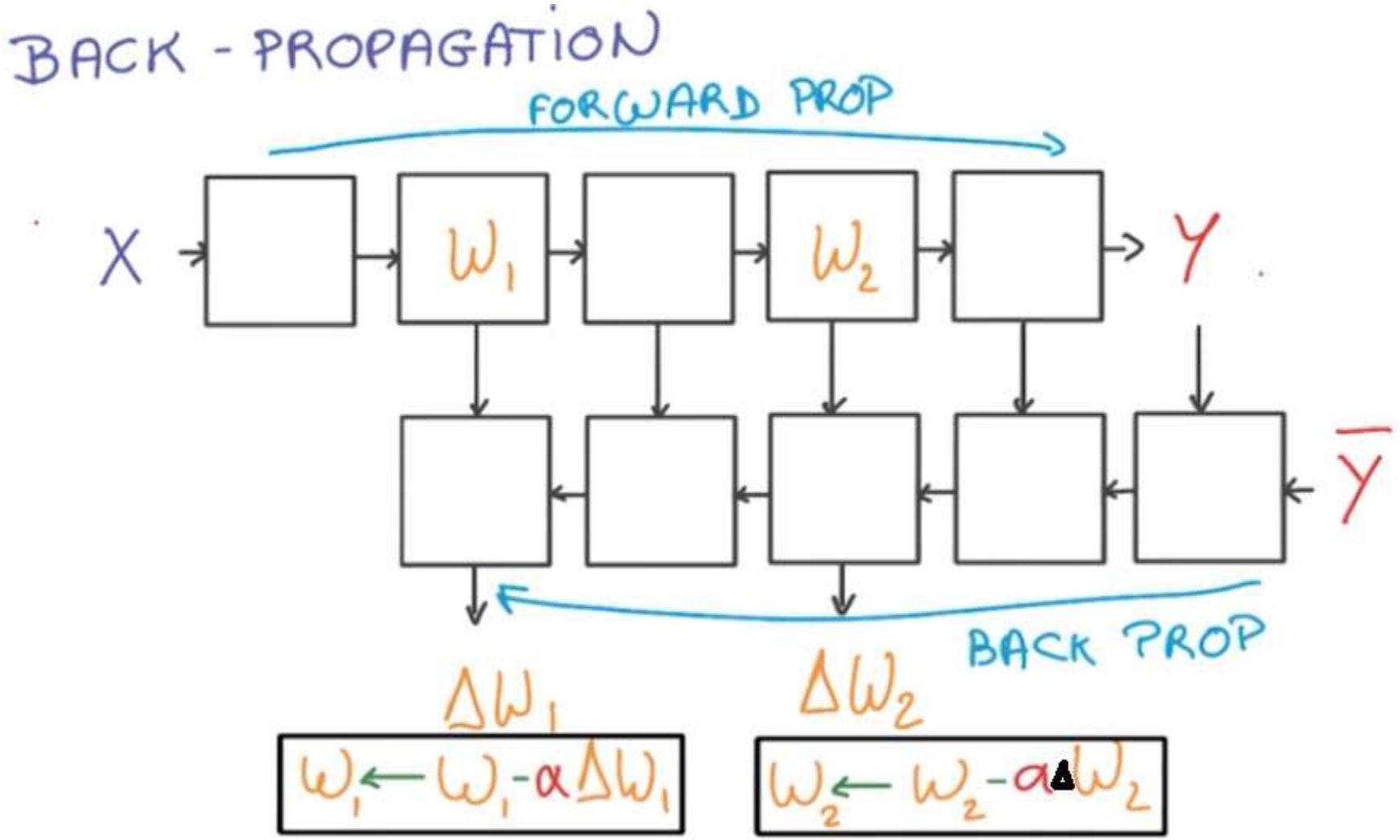


Utilizing chain rule

- Copy and shift the whole pipeline one block over
- Reverse flow of information on copied pipeline and couple the 2 pipelines as follows

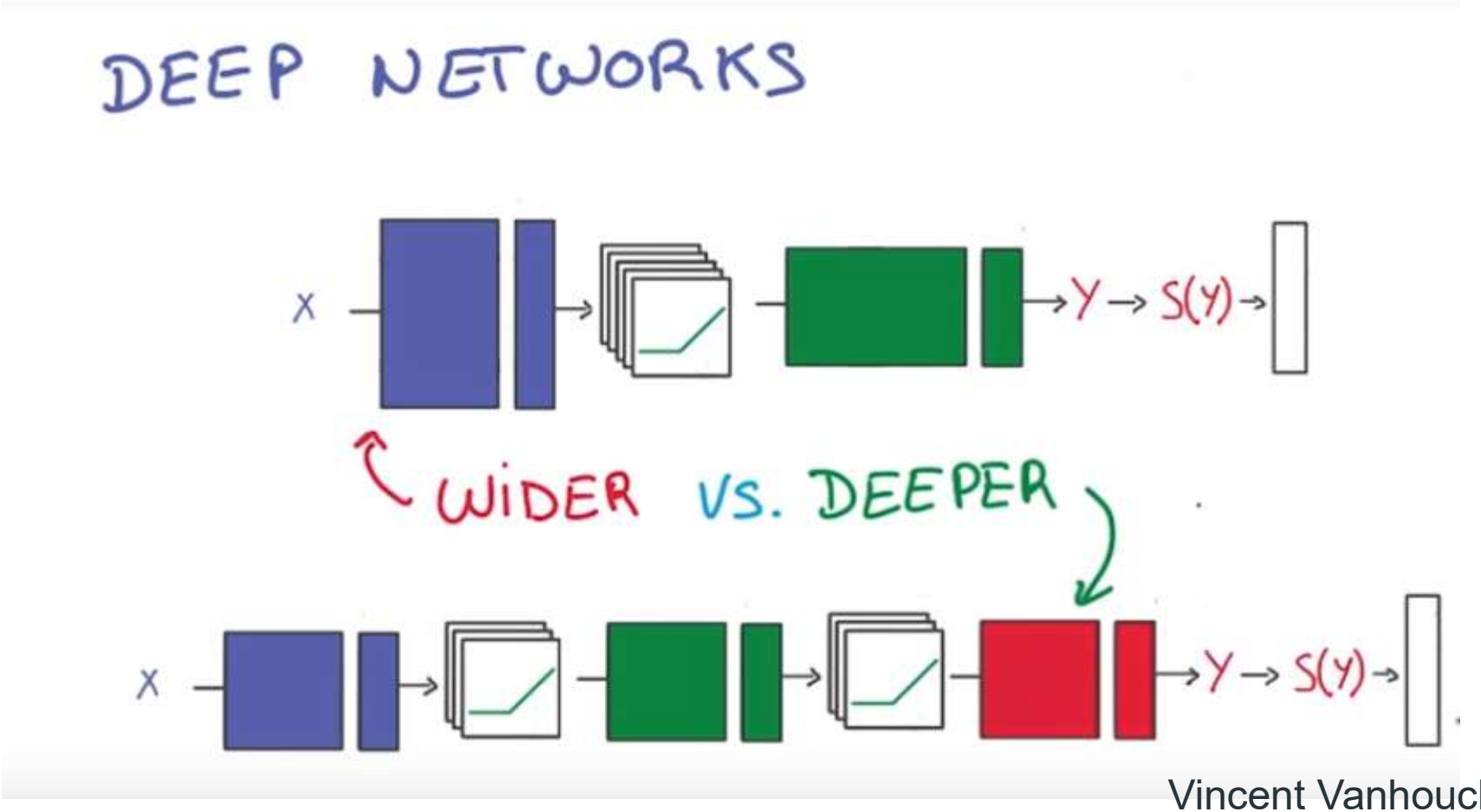


Backpropagation



Scaleability : wider or deeper ?

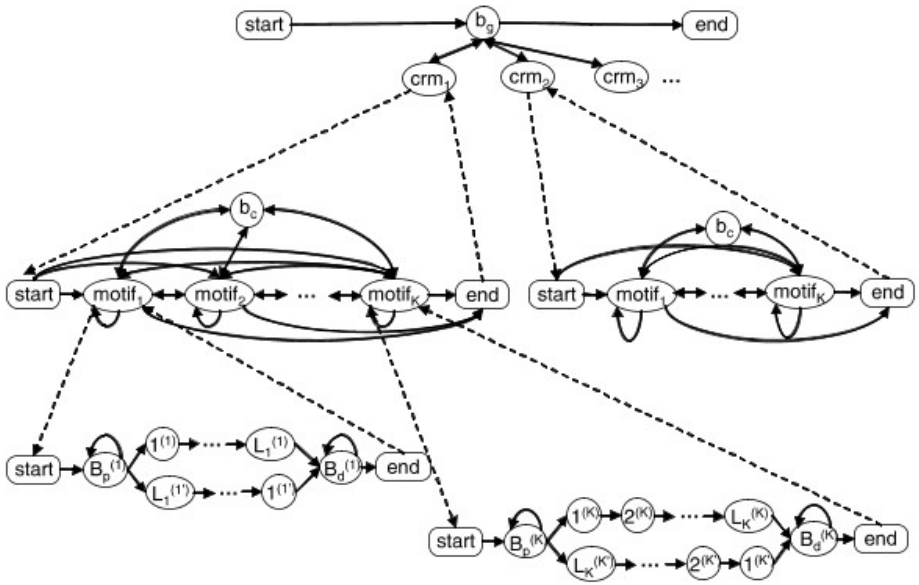
- Which is more tractable ?



Vincent Vanhoucke

Why should multi-layered approaches work ?

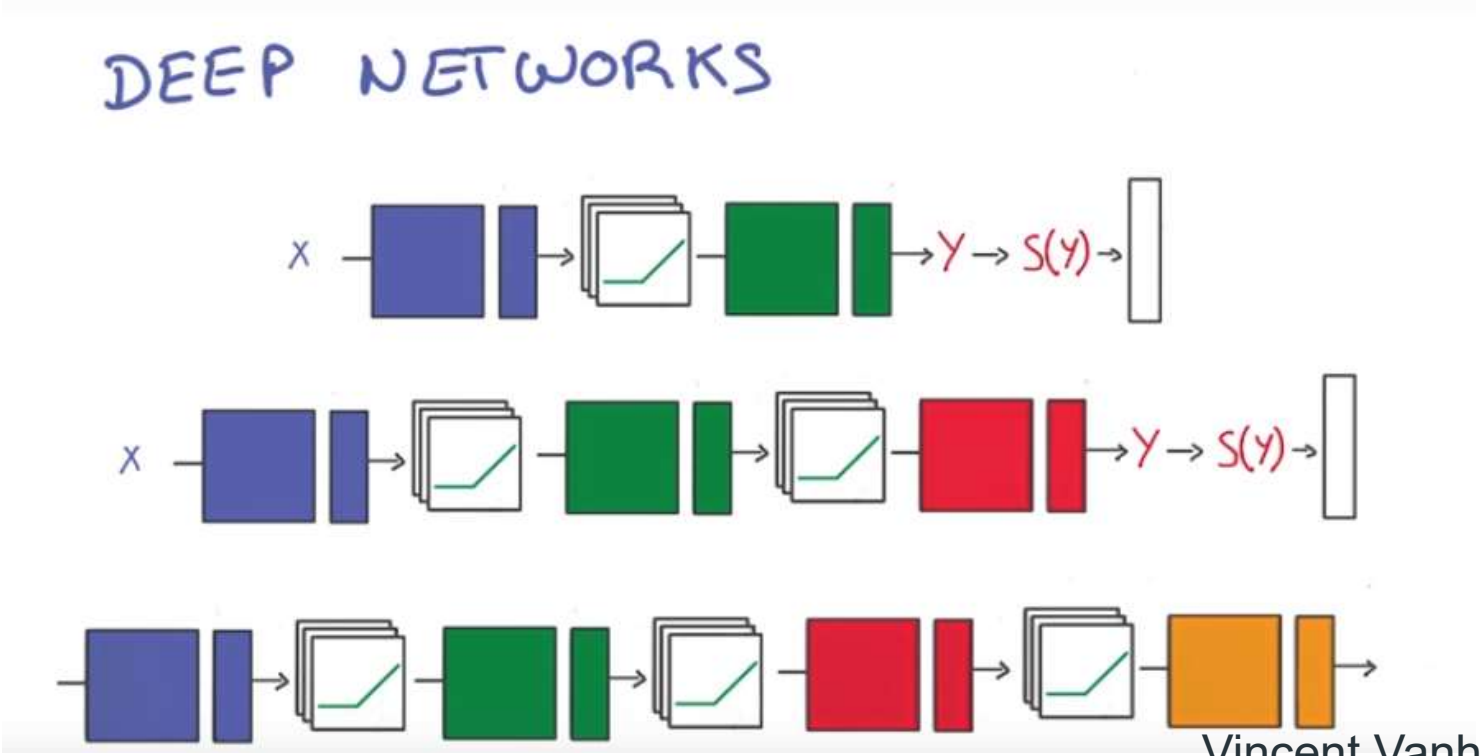
- Typically, many learning tasks are hierarchical
  - Identifying gene regulatory regions in the genome



BayCis motif finding tool

Why were they not discovered before ?

- Lots of parameters to train : computational power and training data only recently available



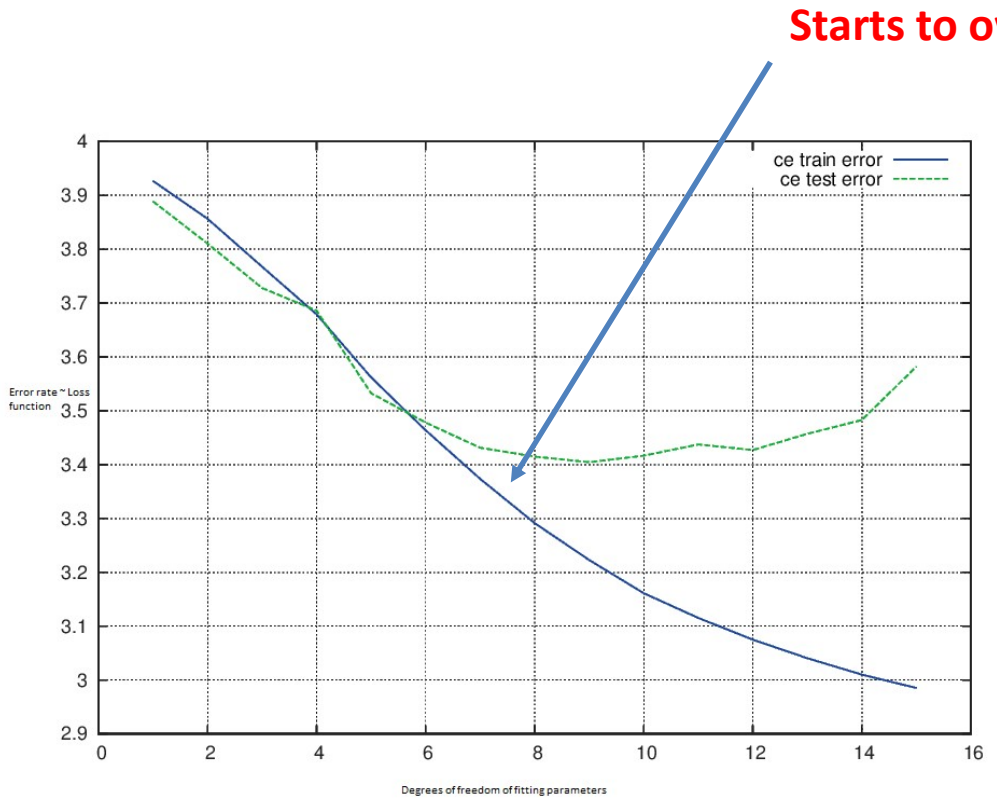
Vincent Vanhoucke

High dimensional models require Big Data

- Take an extreme case
  - More model parameters than training data points
- Perfectly mimic the response variable on the training set
- Fit to the idiosyncracies (noise) of the training set, and not to the underlying broad correlative patterns (signal) : overfitting

How can we recognize overfitting ?

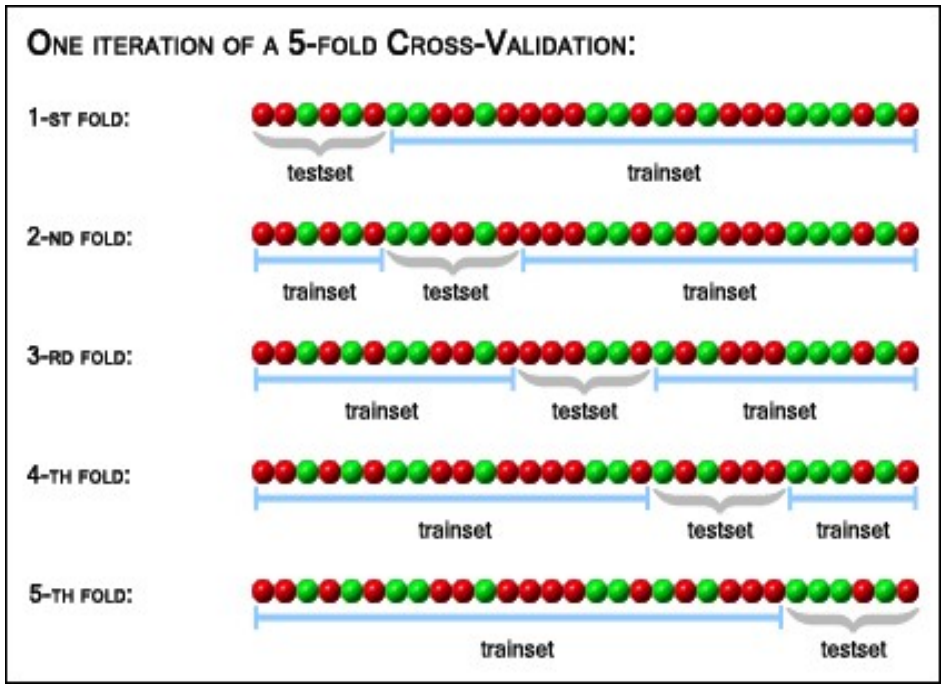
- Differential performance on training and validation sets



Atcold

How can we avoid overfitting ?

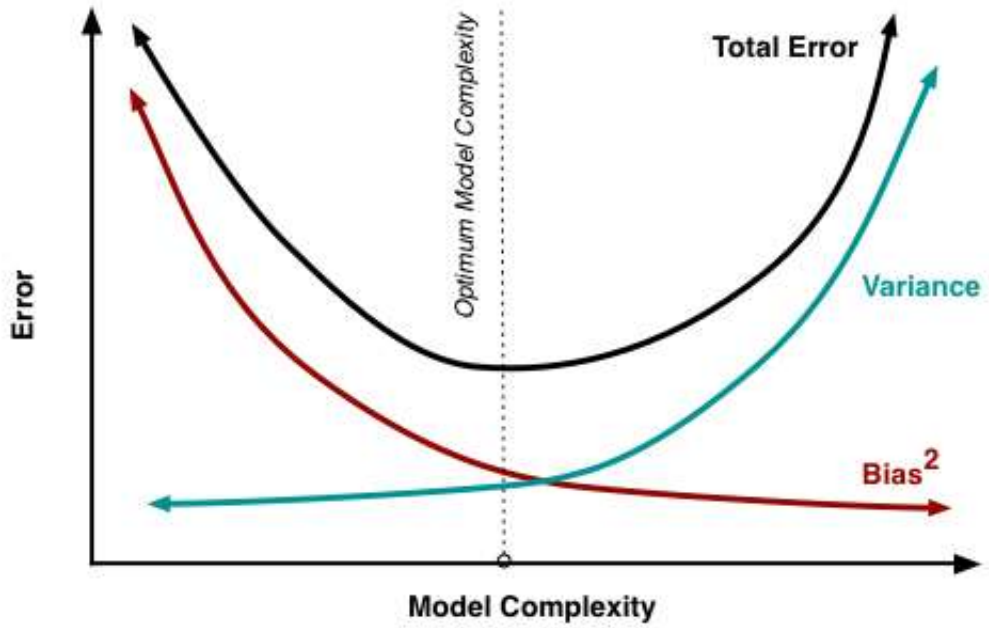
- Model cross-trained and tested on multiple large, heterogeneous datasets carved out of the original dataset
- Requires a lot of data and a lot of computing power



Shane : twitter @statalgo

The virtues of sparsity

- Fitting an optimal number of parameters can get us best results

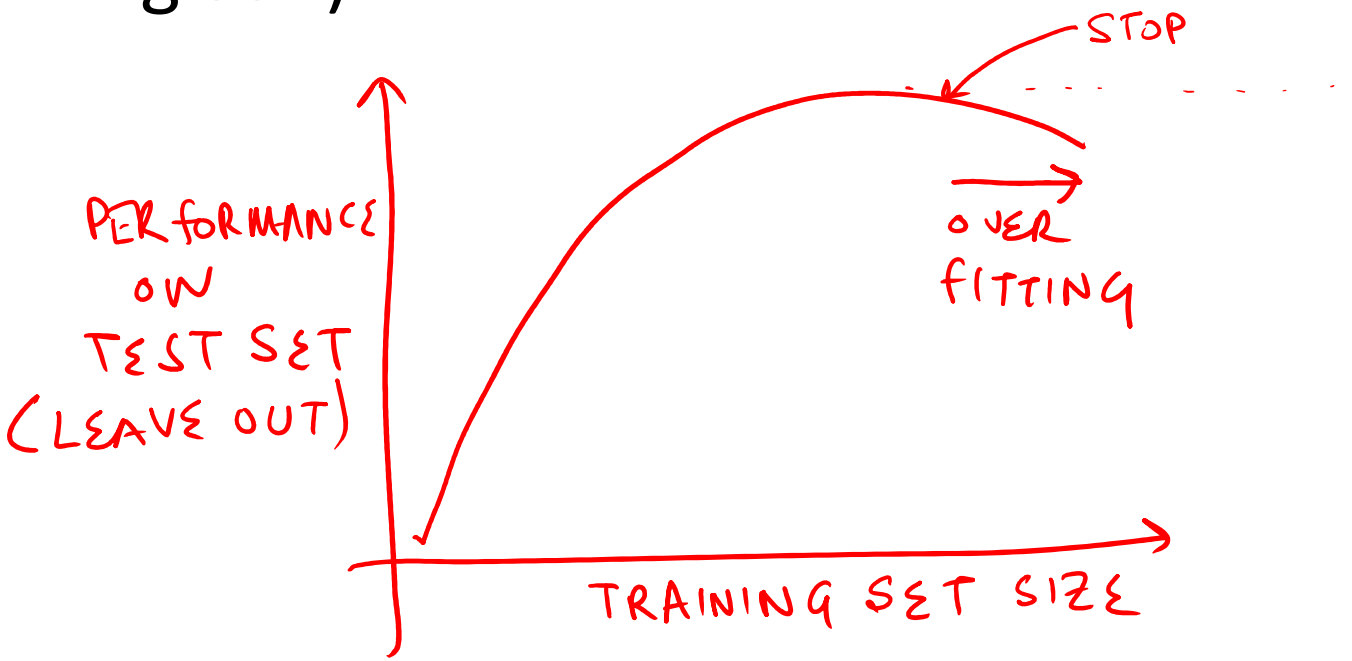


Scott Fortmann-Roe

- "Skinny jeans" problem : choose a jeans one size too large is easier to fit into than the perfect fit
- Choose a model slightly bigger than you need, and then reduce complexity

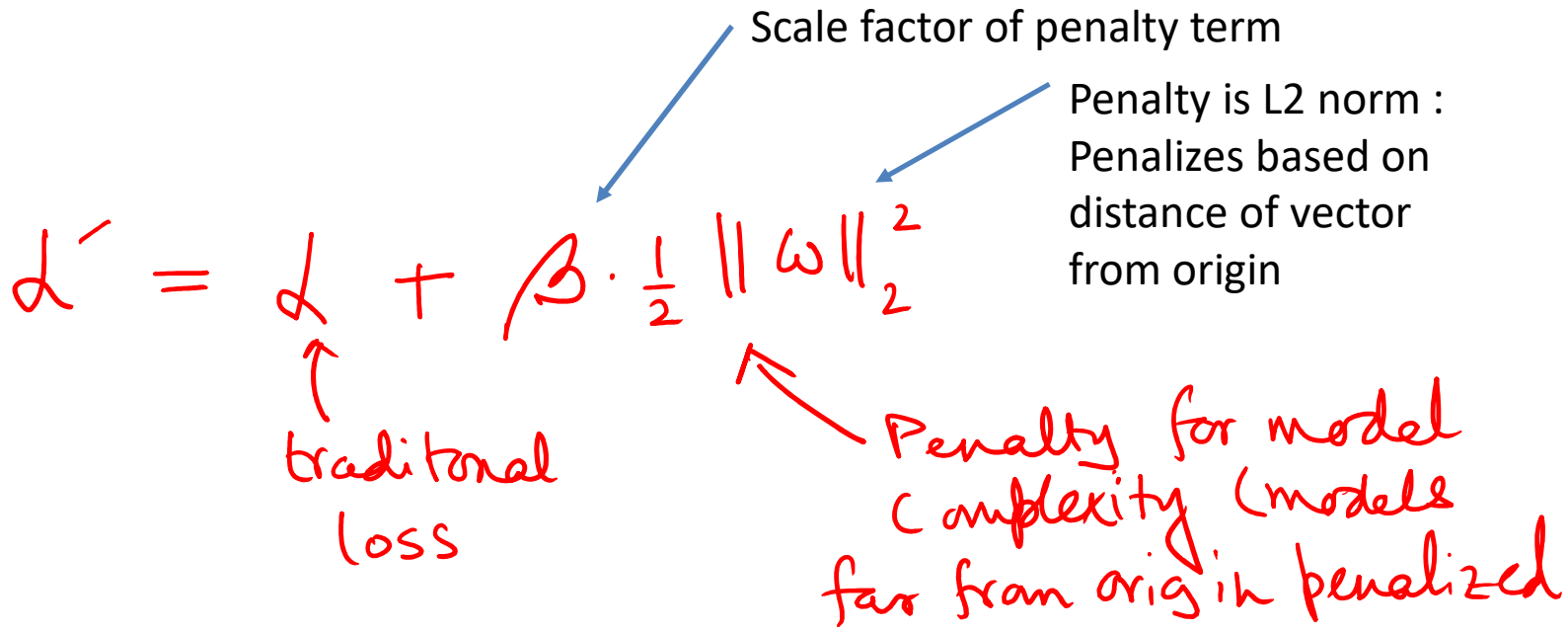
Making deep nets work

- Do not use more data than you have to : even with the right model complexity you need to guard against overfitting (overoptimizing on training set)



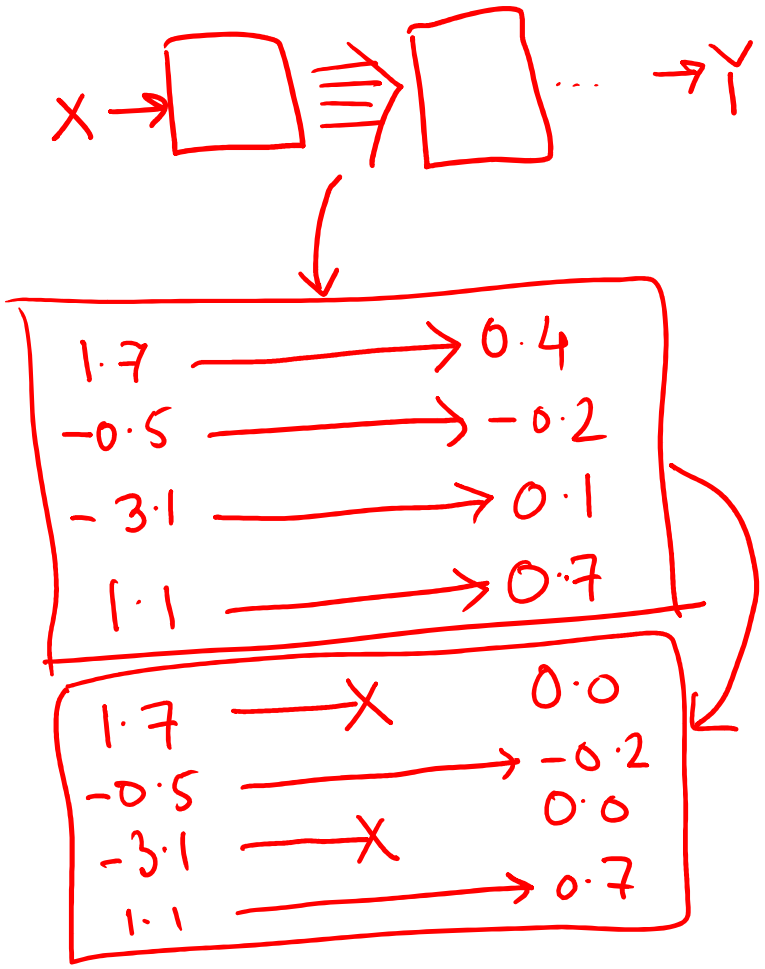
Making deep nets work

- Add a penalty term to the loss function to penalize complexity



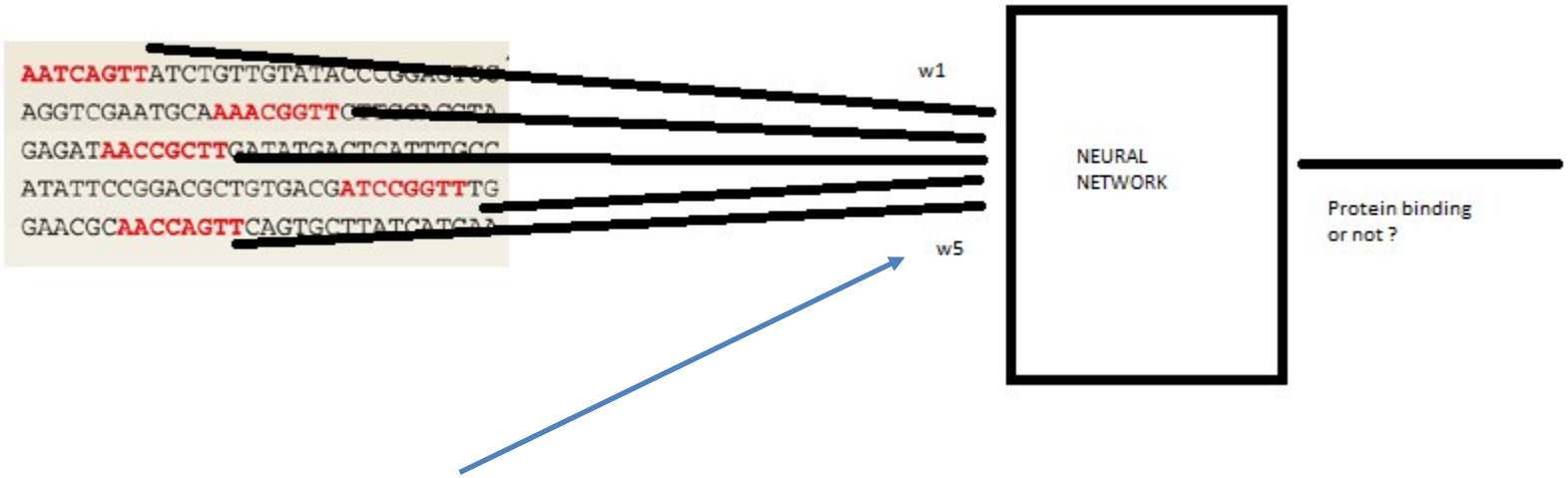
Making deep nets work

- Dropouts : “drop” (set to zero) a fraction (say, half) of the connections for each training example randomly
- Stops weights from “co-adapting” and breaks correlations between parameters



Making deep nets work

- Weight sharing : should the classifier treat some parameters equally ?



If the binding of protein to DNA is invariant of the exact genomic position of the motif, weights w1, w2, .. , w5 can be combined such that w1 = w2 = .. = w5 = w

Model simplification based on invariance properties !

Sequence from : Nature Biotechnology **24**, 959 - 961 (2006)

21st century Biology : high throughput assays

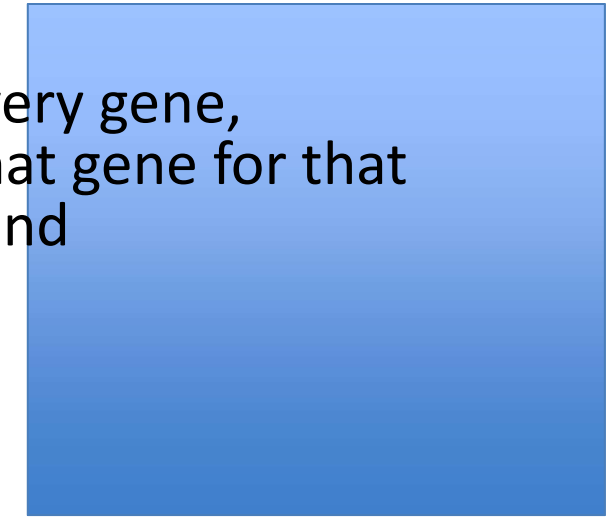
- Low throughput assays : eg. qPCR – queries a trait (in this case mRNA abundance) for one or a few locations in the genome / epigenome / transcriptome / proteome
- High throughput assays : eg. RNA-seq – queries a trait (in this case mRNA abundance) for many or all locations in the genome / epigenome / transcriptome / proteome **simultaneously**

How many high throughput dimensions ?

- One
  - RNA-seq : transcriptional level across every gene in the genome

x1, x2, x3, …, xN : for large N

- Two
  - Hi-C : For every pair of sites in the genome, characterize spatial proximity between the sites (both high throughput dimensions : genome)
  - Single cell RNA-seq : For every cell and every gene, characterize the transcriptional level in that gene for that cell (high throughput dimensions : cells and transcriptome)



x11, x12, x13, …, x1N : for large N

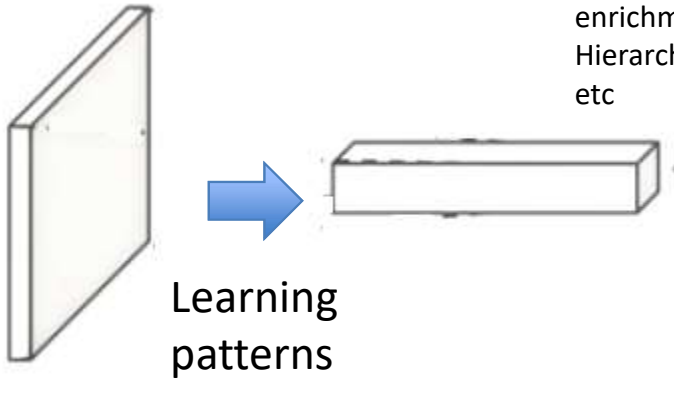
x11, x21, x31, …, xM1 : for large M

What do we want to learn from such datasets ?

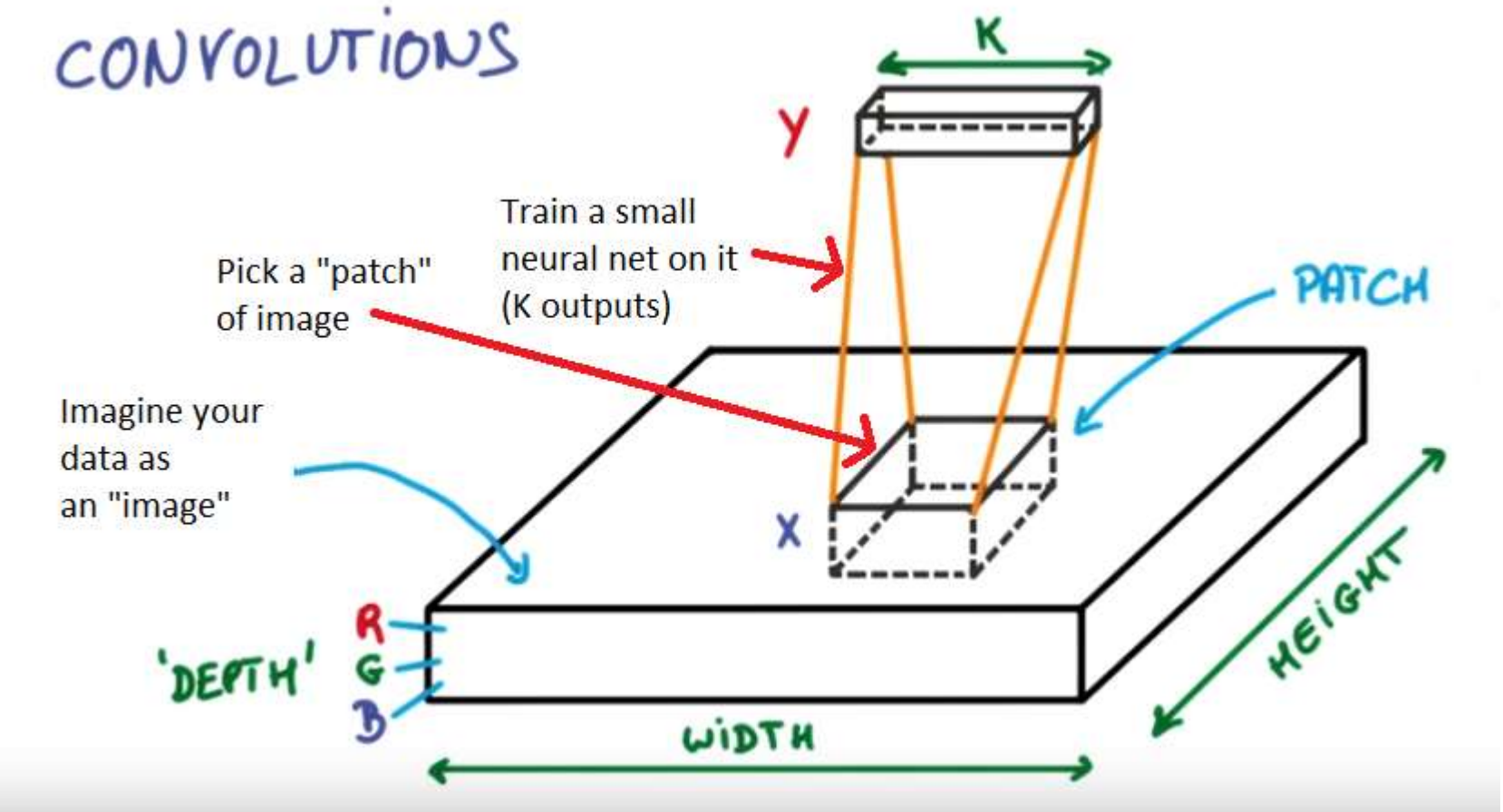
- Are some pathways / processes enriched ?
- Is there some typical correlation with a previously unrelated trait ?
- So, a set of structured outputs is desired which is common to the entire dataset(s)

A set of values consistent across the data characterizing correlations, enrichments, Hierarchy, etc

Data : Height and width – high throughput Dimensions
Depth – low throughput dimensions (healthy Vs normal)

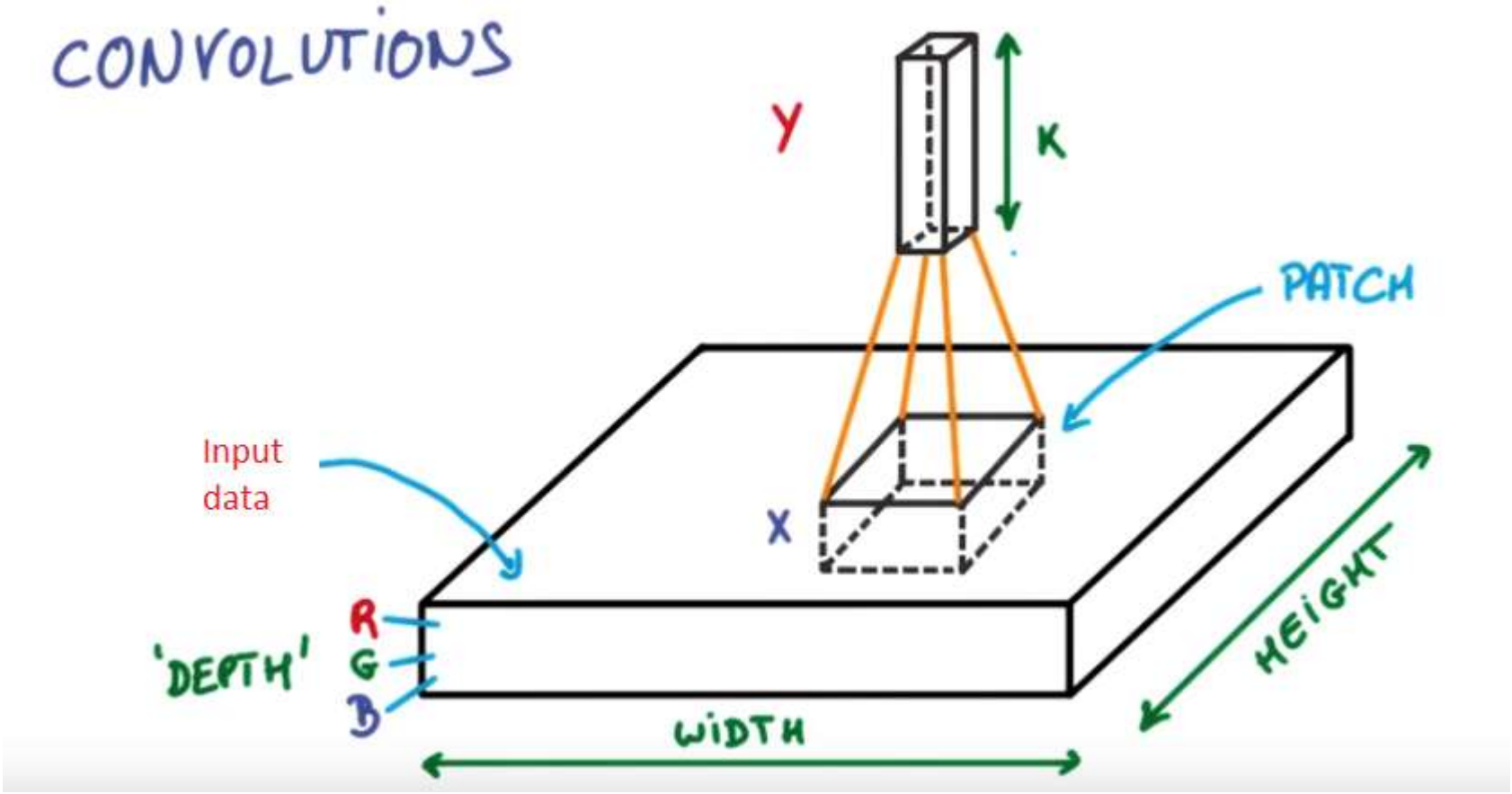


“Mix up” a small contiguous part of the dataset by training a neural net on it



Vincent Vanhoucke

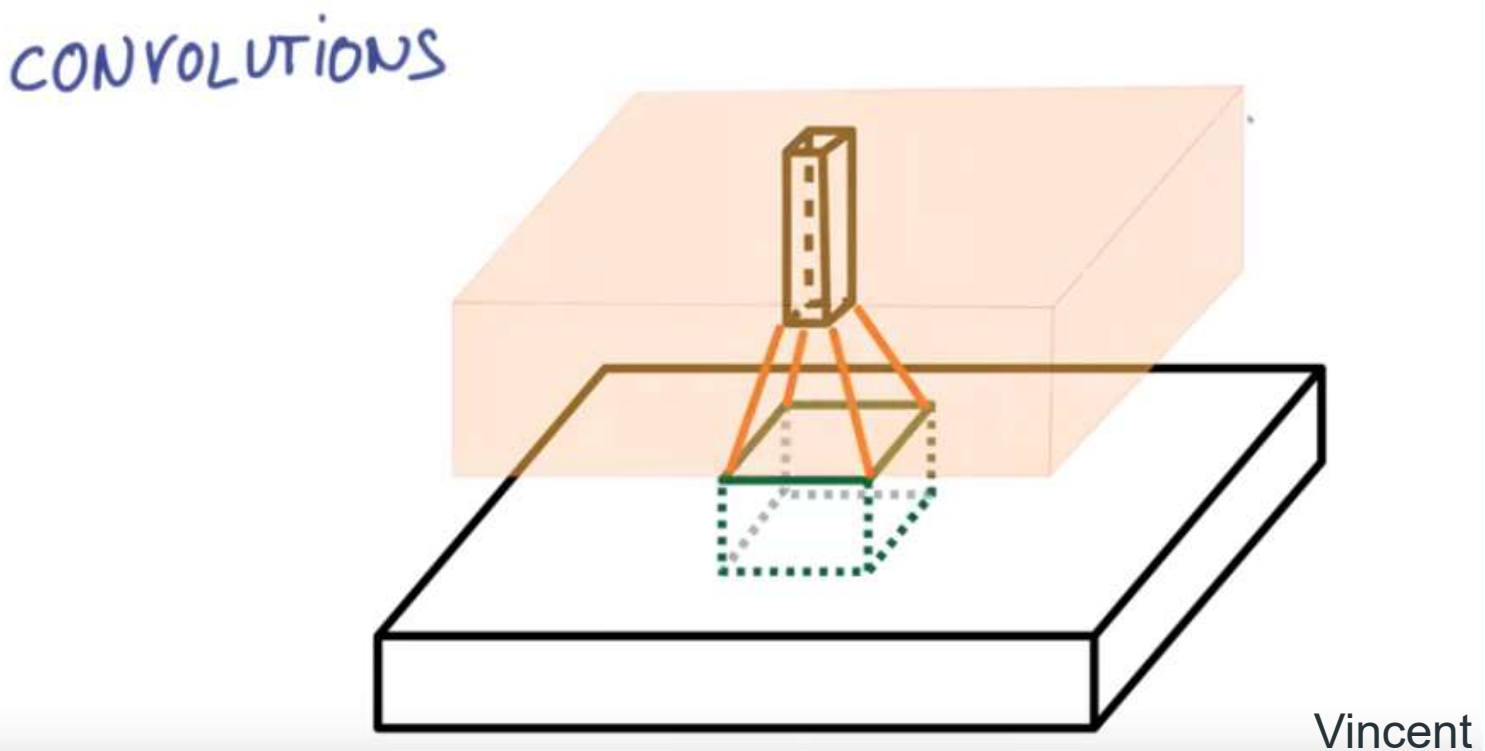
An equivalent representation



Vincent Vanhoucke

Create an image from NN outputs

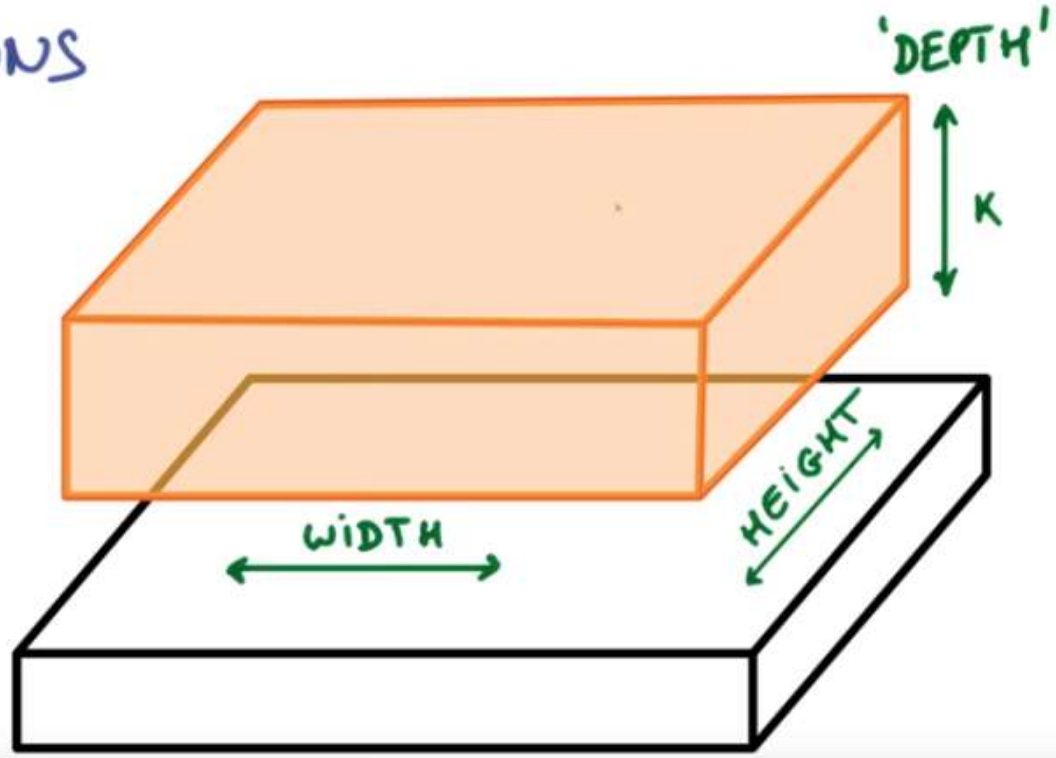
- Slide the NN across the image, and assemble the outputs to form an “output image”



What is the size of the output ?

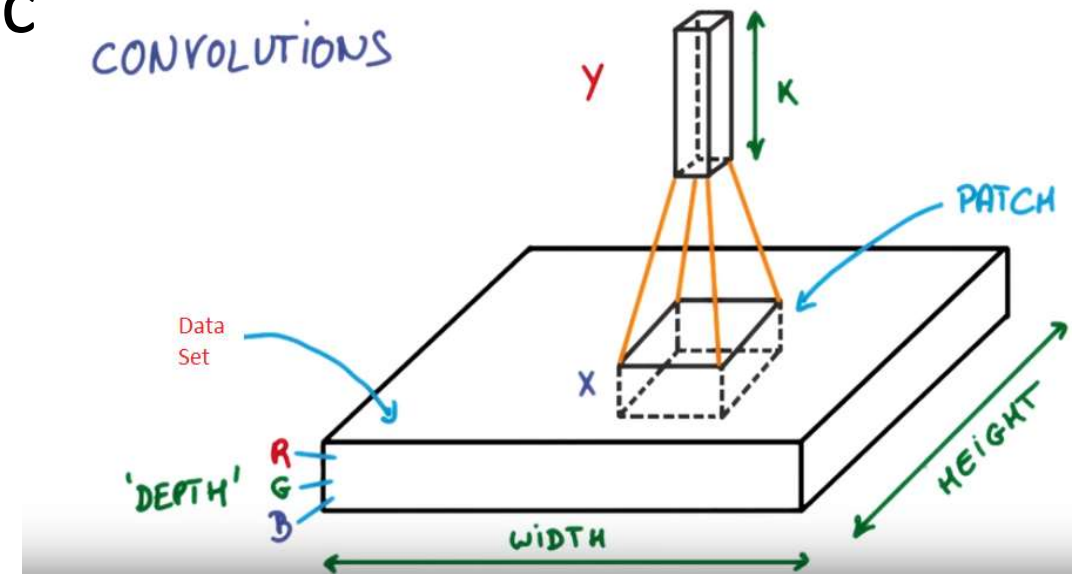
- Depth = k (output of neural net)
- Height and width = Height and width of original image scaled by the step-size of the sliding CONVOLUTIONS

Entire operation
= “convolution”

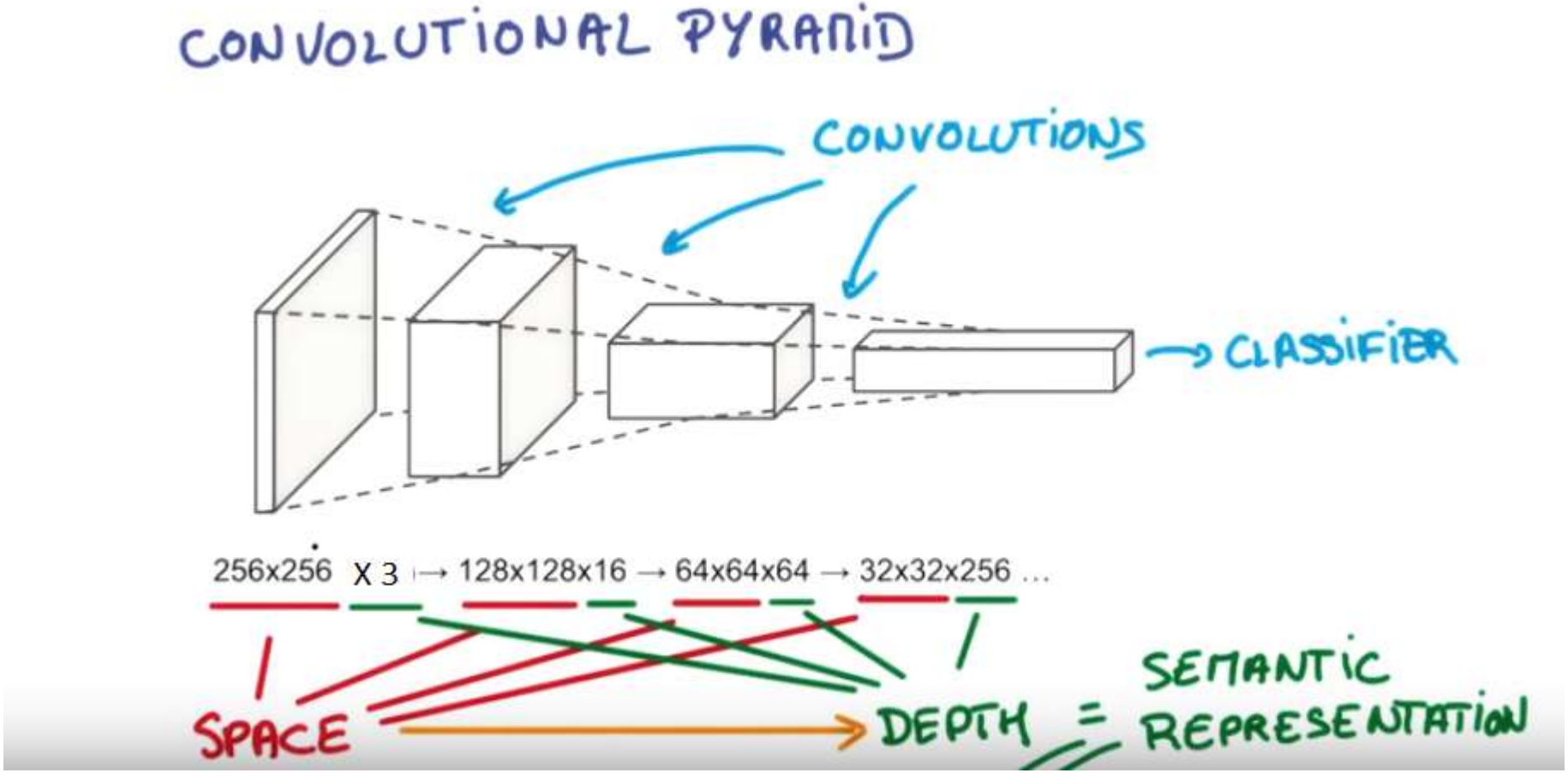


Patch size

- Patch size =1 X 1 : called a 1 X 1 convolution
- Patch size = image size : equivalent to a layer of a traditional neural network
- Weights are shared across the image (parameters specific to a region of the image will not be inferred)

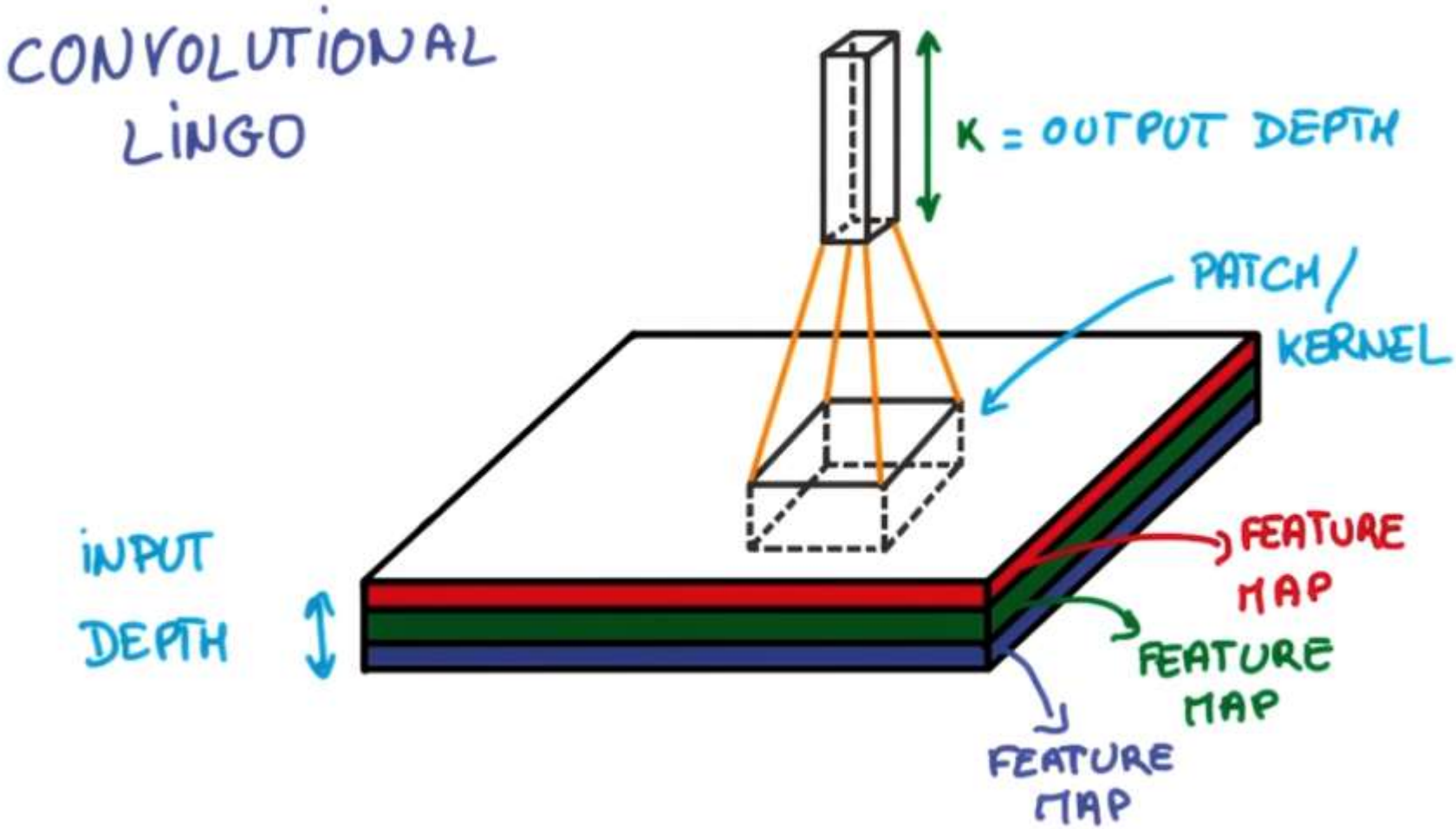


Stacking convolutions : the “deep” part of it



Vincent Vanhoucke

Increase depth (extracting info), decrease space (generalize) at each step



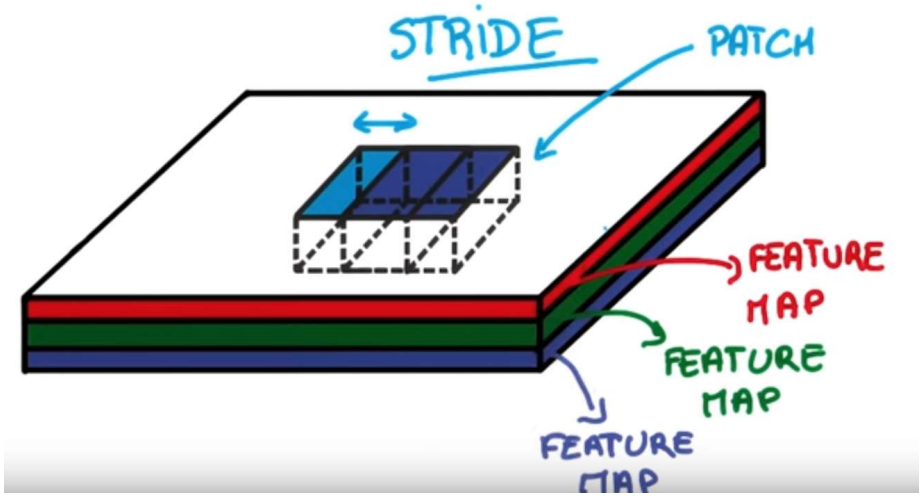
Vincent Vanhoucke

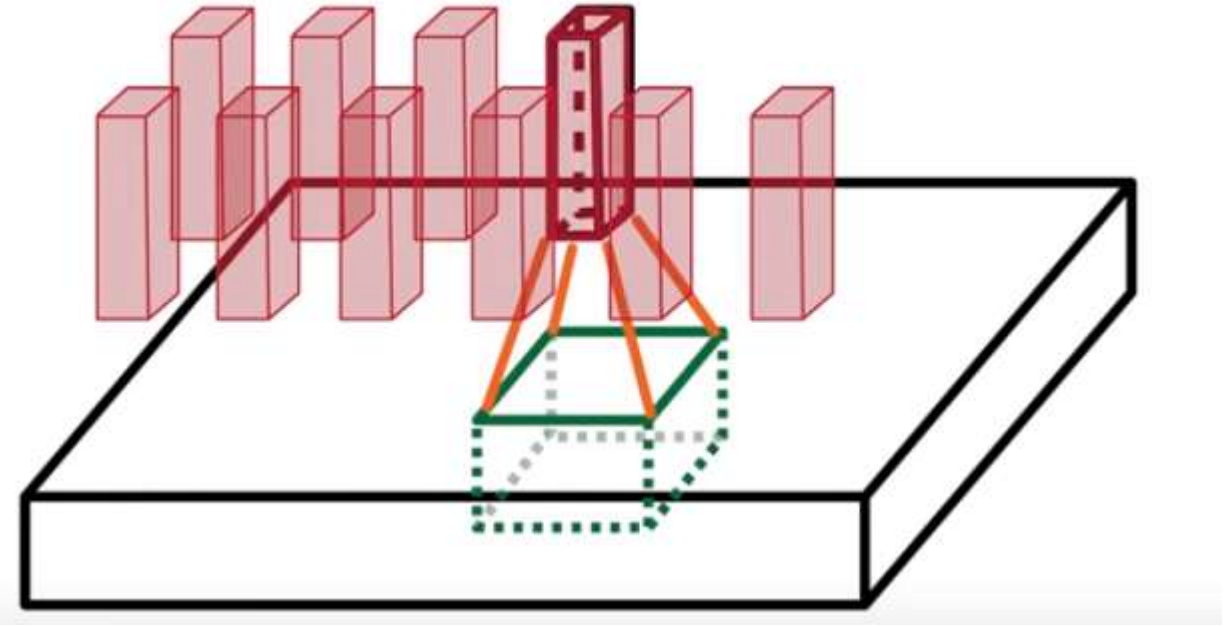
Stride

- Stride : resolution of analysis

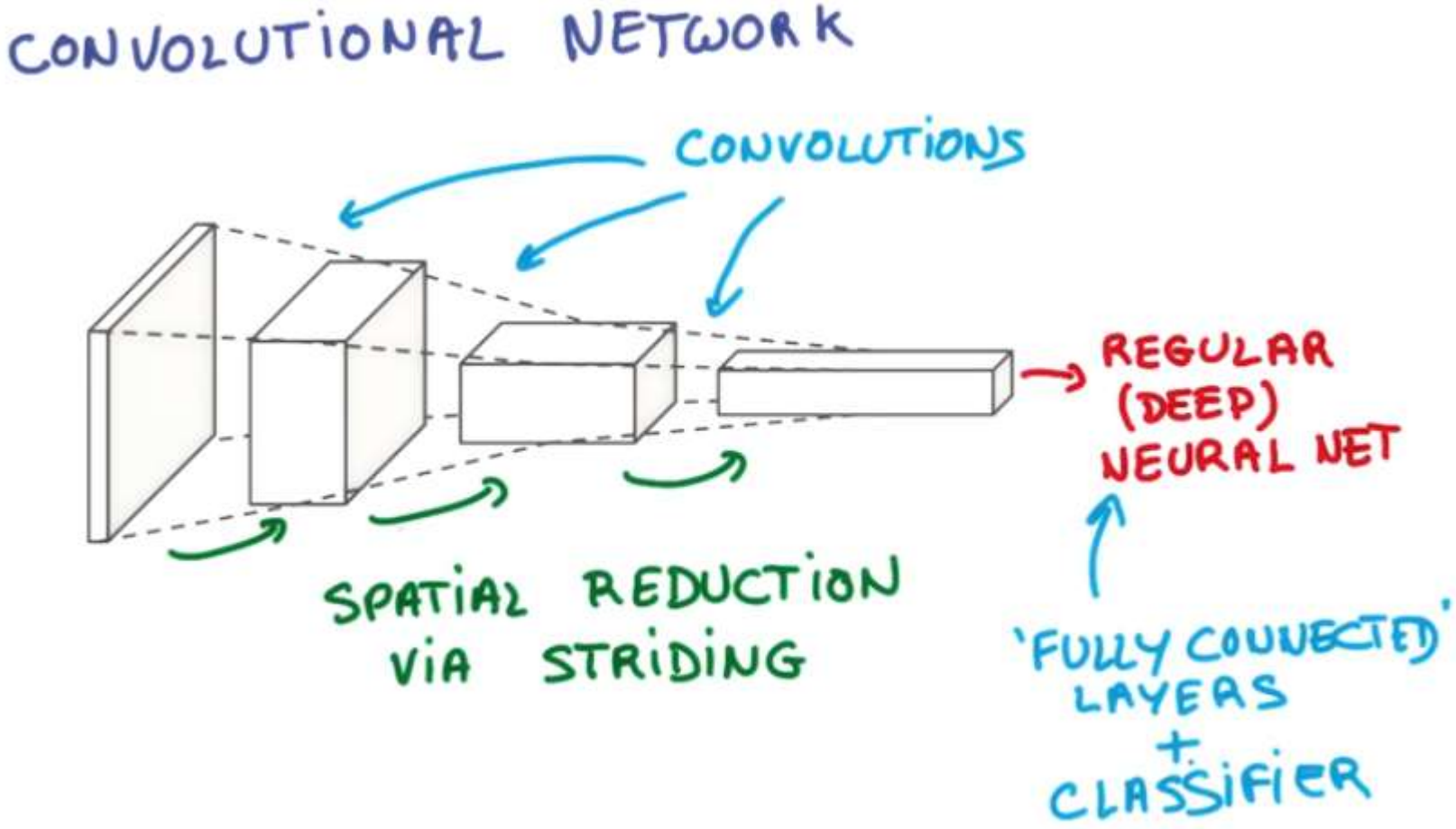
Step size of the convolution as the NN marches across image

- Paddings on the side



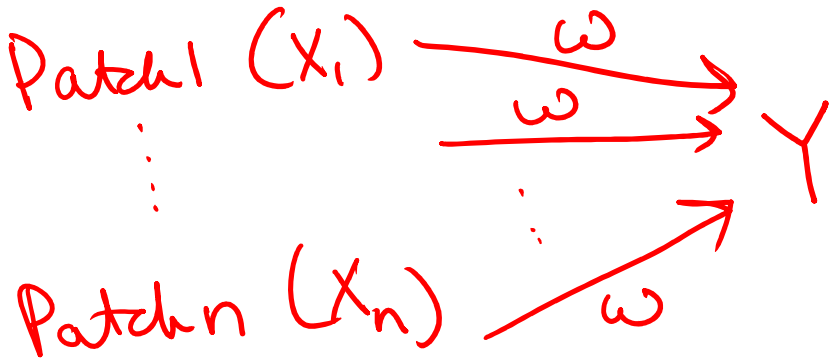


Hooking up conv-nets with regular deep nets



How to train the neural net ?

- How sharing works : works out of the box



$$\frac{\Delta d}{\Delta \omega} = \frac{\Delta d}{\Delta \omega}(\text{patch}_1) + \cdots + \frac{\Delta d}{\Delta \omega}(\text{patch}_n)$$

↑ add gradients from every patch

Reducing size at each step

- Increase stride /

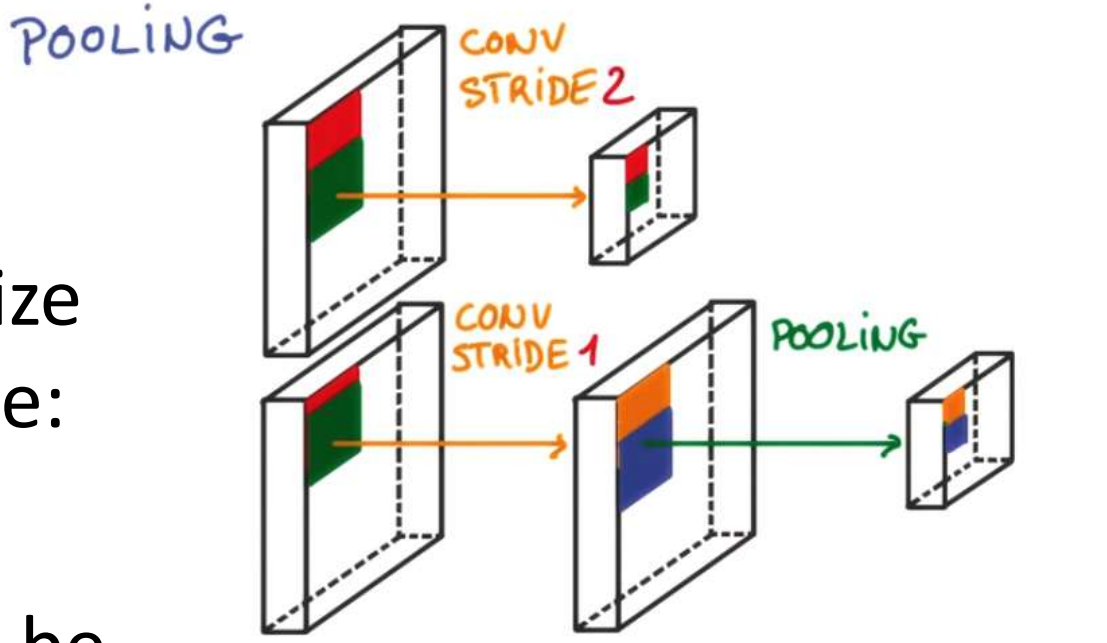
Step size

- Or, use a second stage to summarize parts of the image:

Pooling

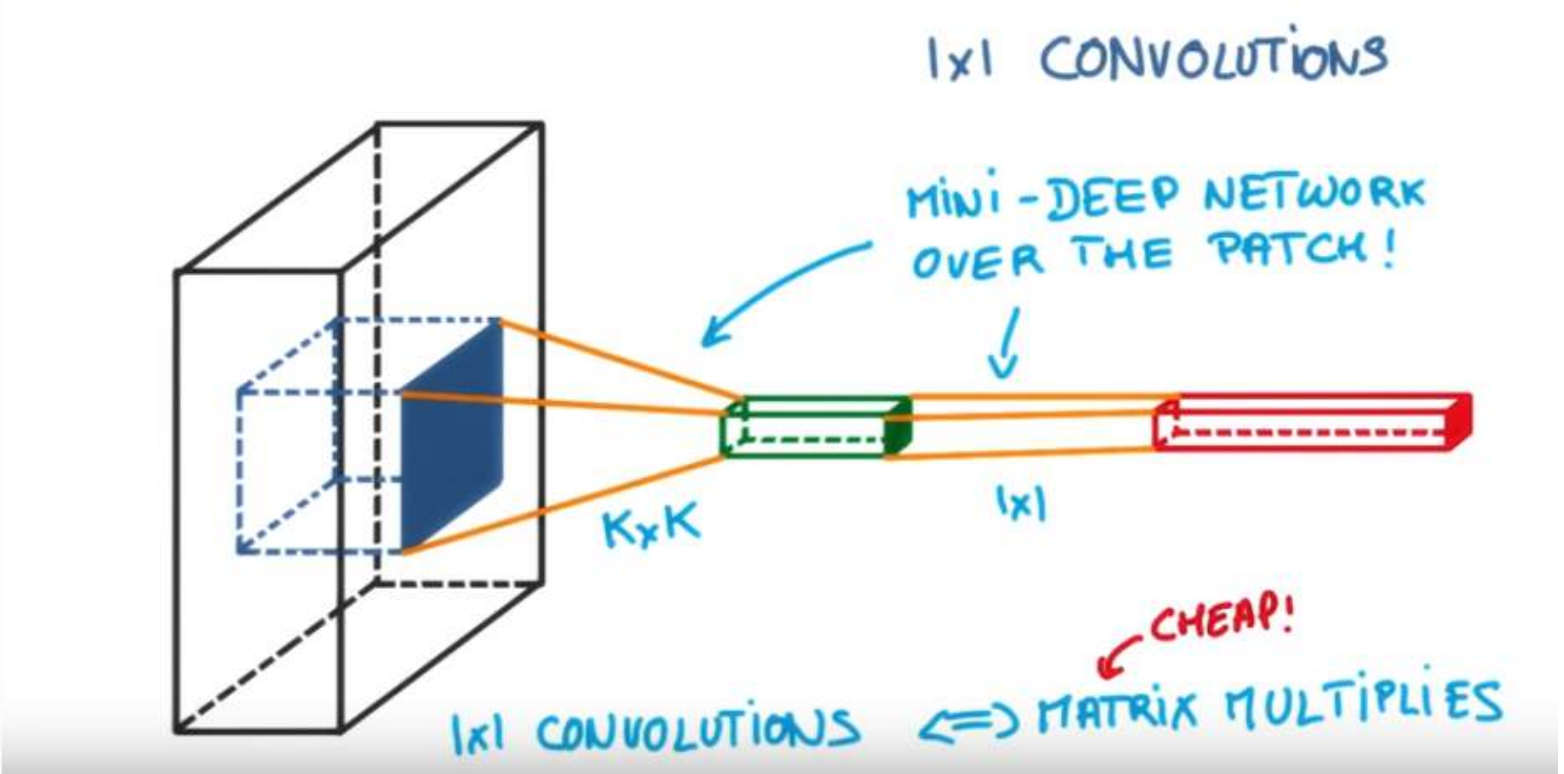
Summarization can be

Max, Average, etc



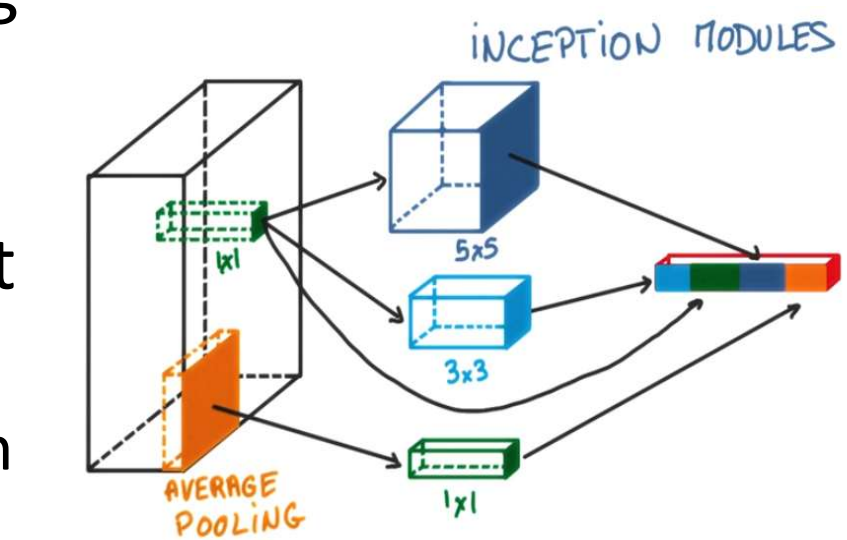
Why use 1X1 convolutions in a convolution stack ?

- Converts a linear classifier into a neural network : without too many new parameters



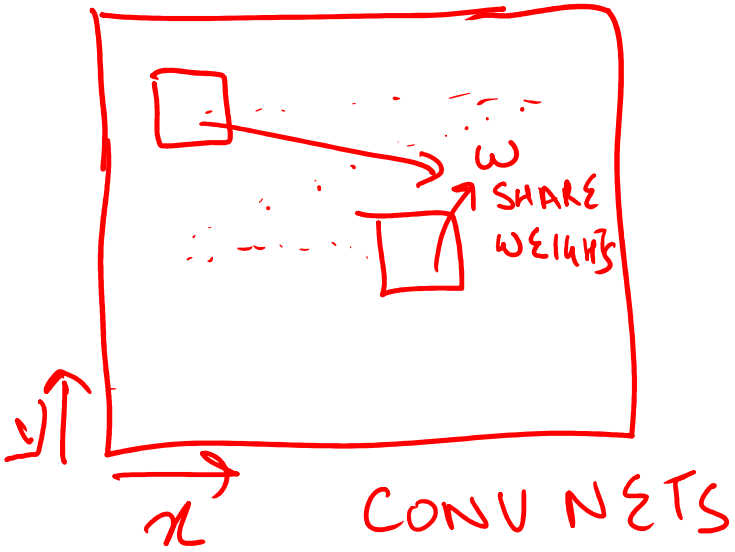
Inception modules

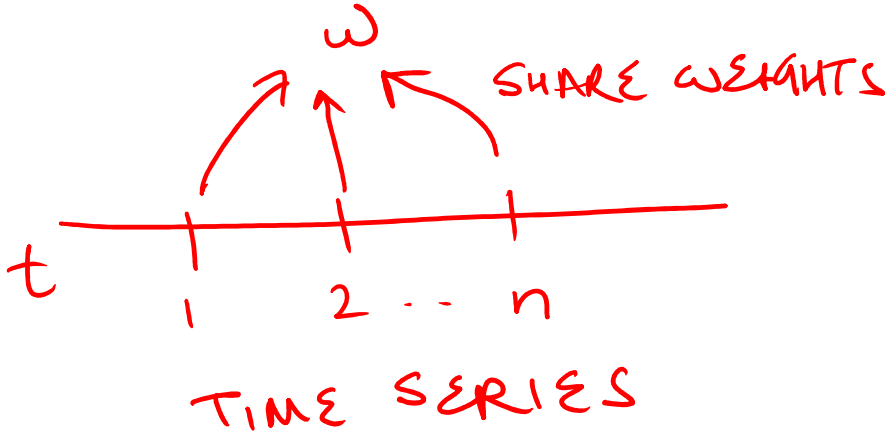
- At each step of the pyramid, a choice of using pooling or convolution (and of what size) ?
- You can hedge and have it all
  - Performance is better than using an individual step (different information captured by different convolutions)



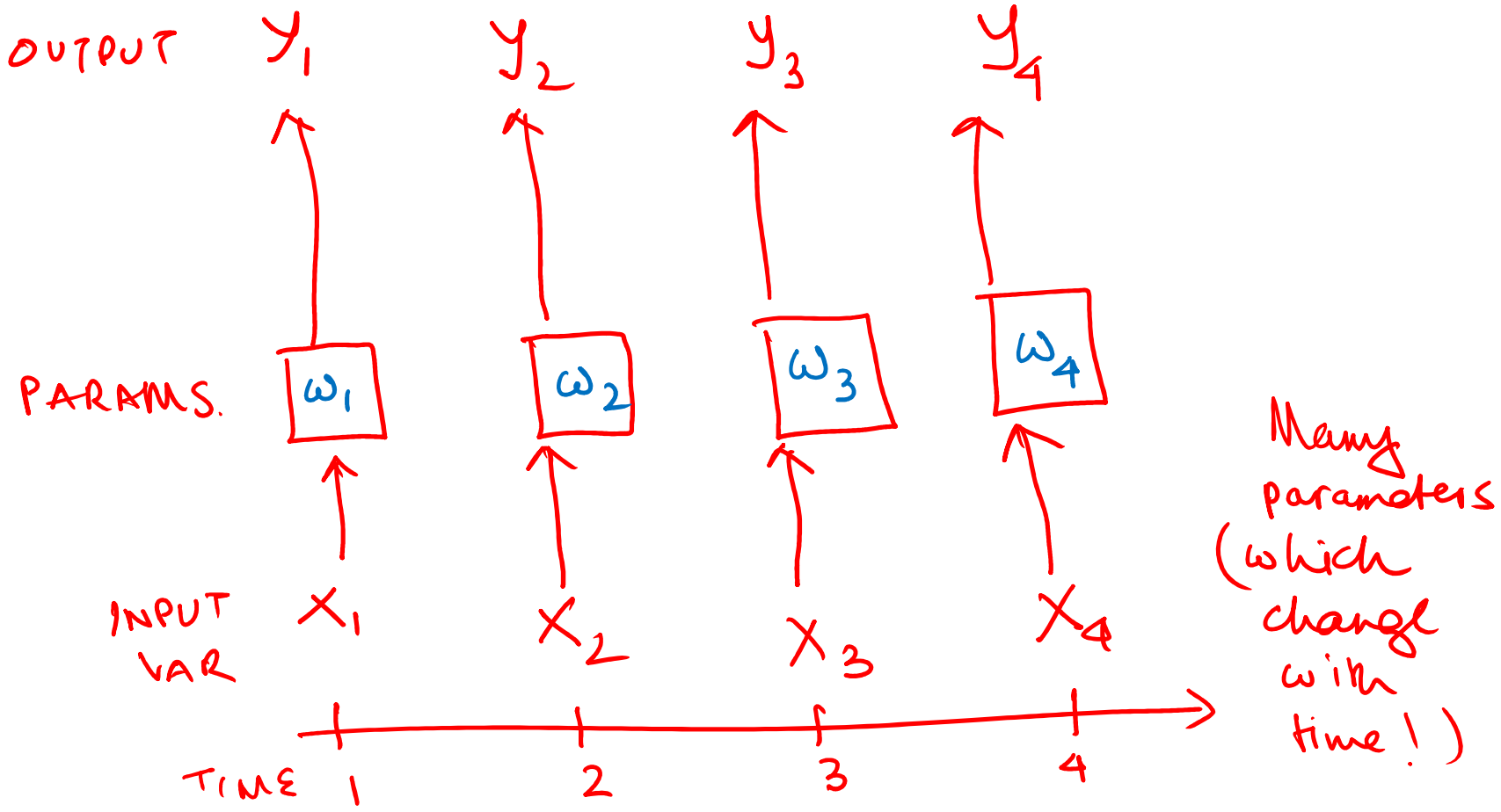
Time series data

- Weights shared across patches (hence across all dimensions)
- For time homogeneous models, we can share parameters across time dimension

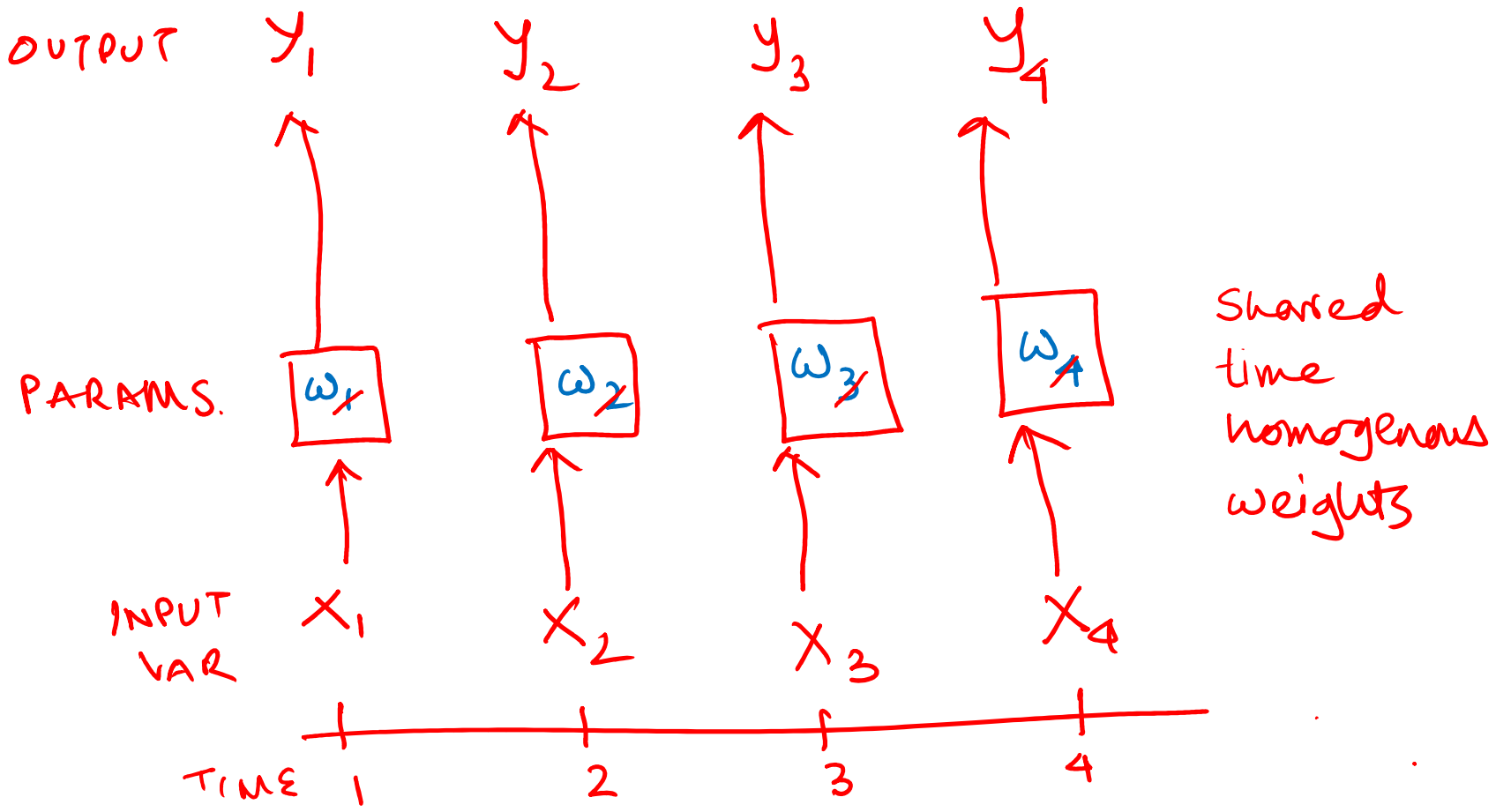




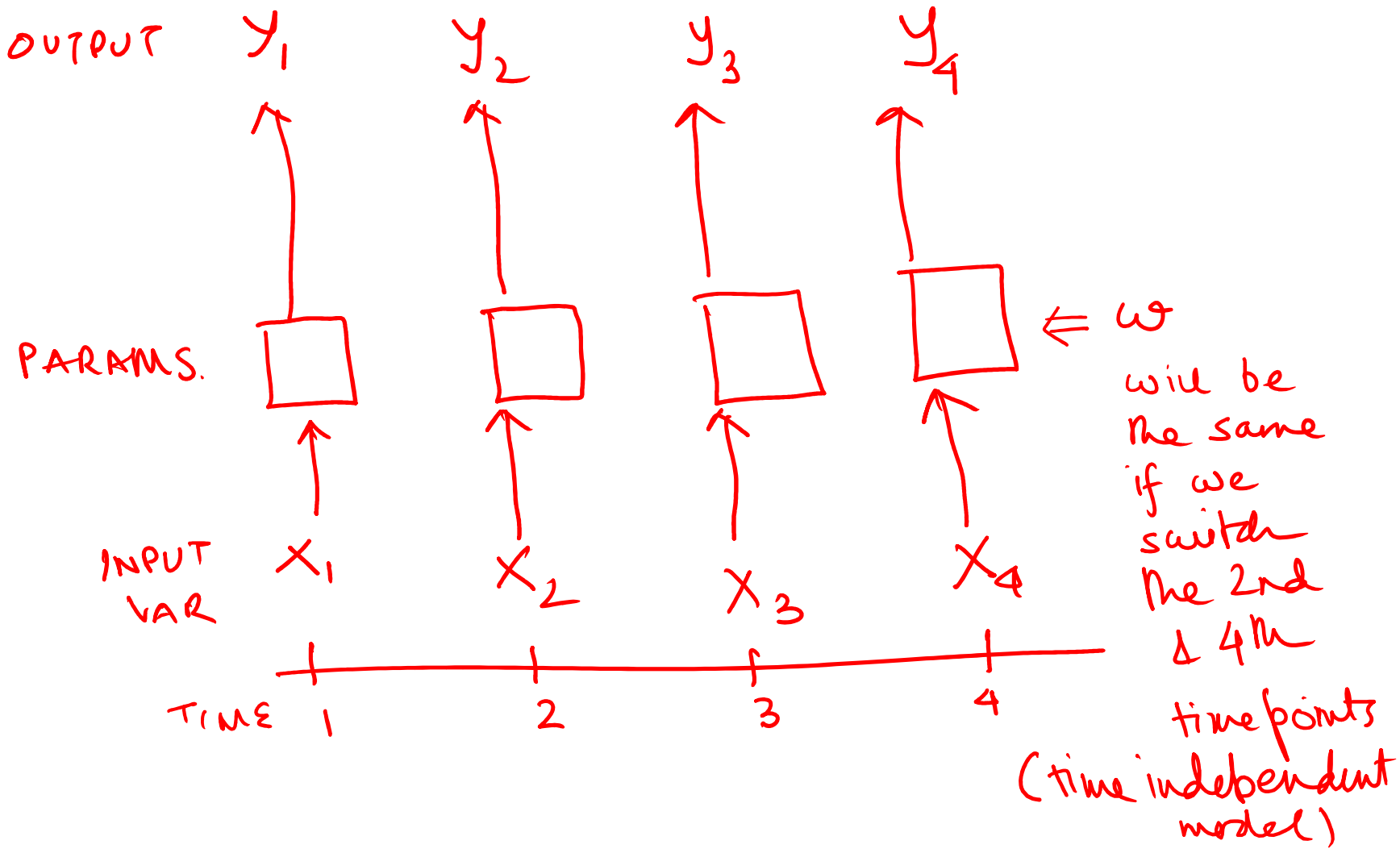
Neural networks with temporal data



Taking advantage of time homogeneity

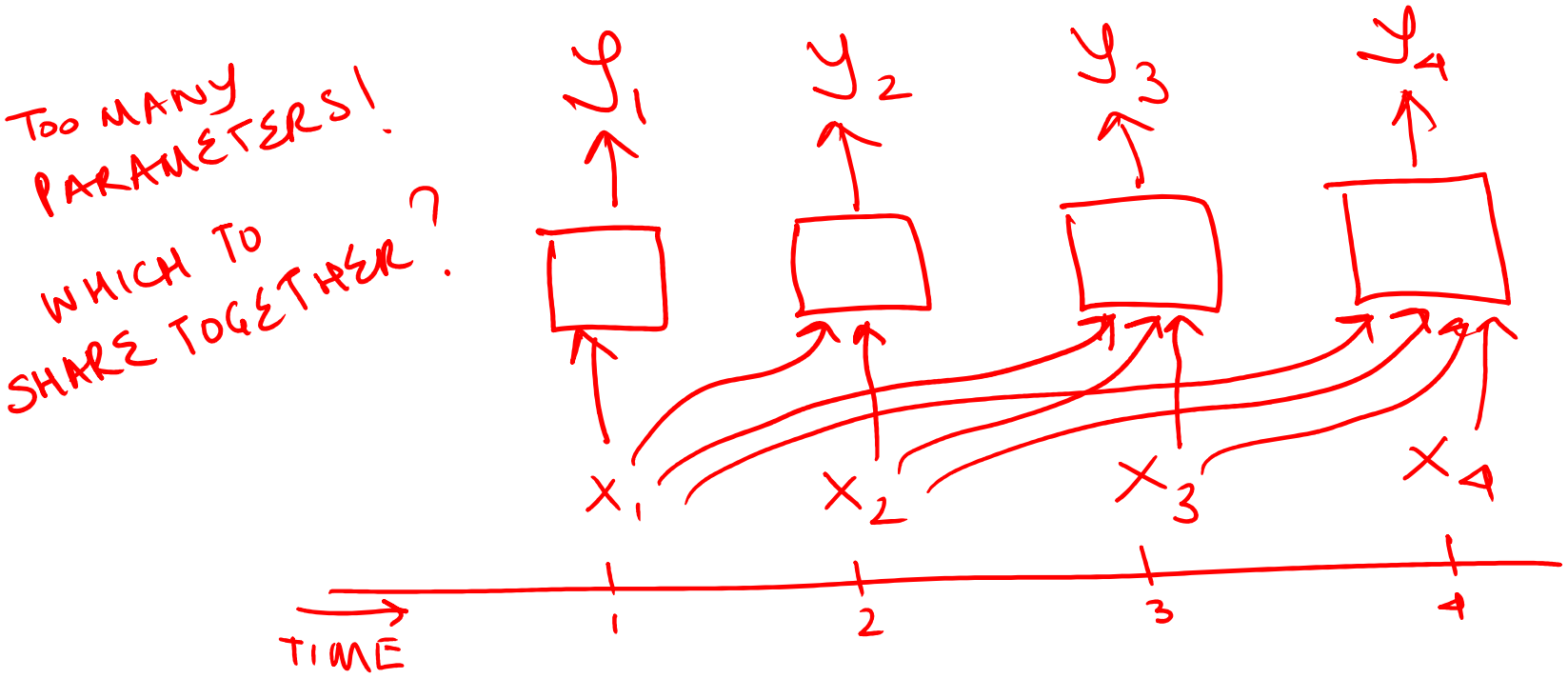


Time homogeneity and time agnosticism

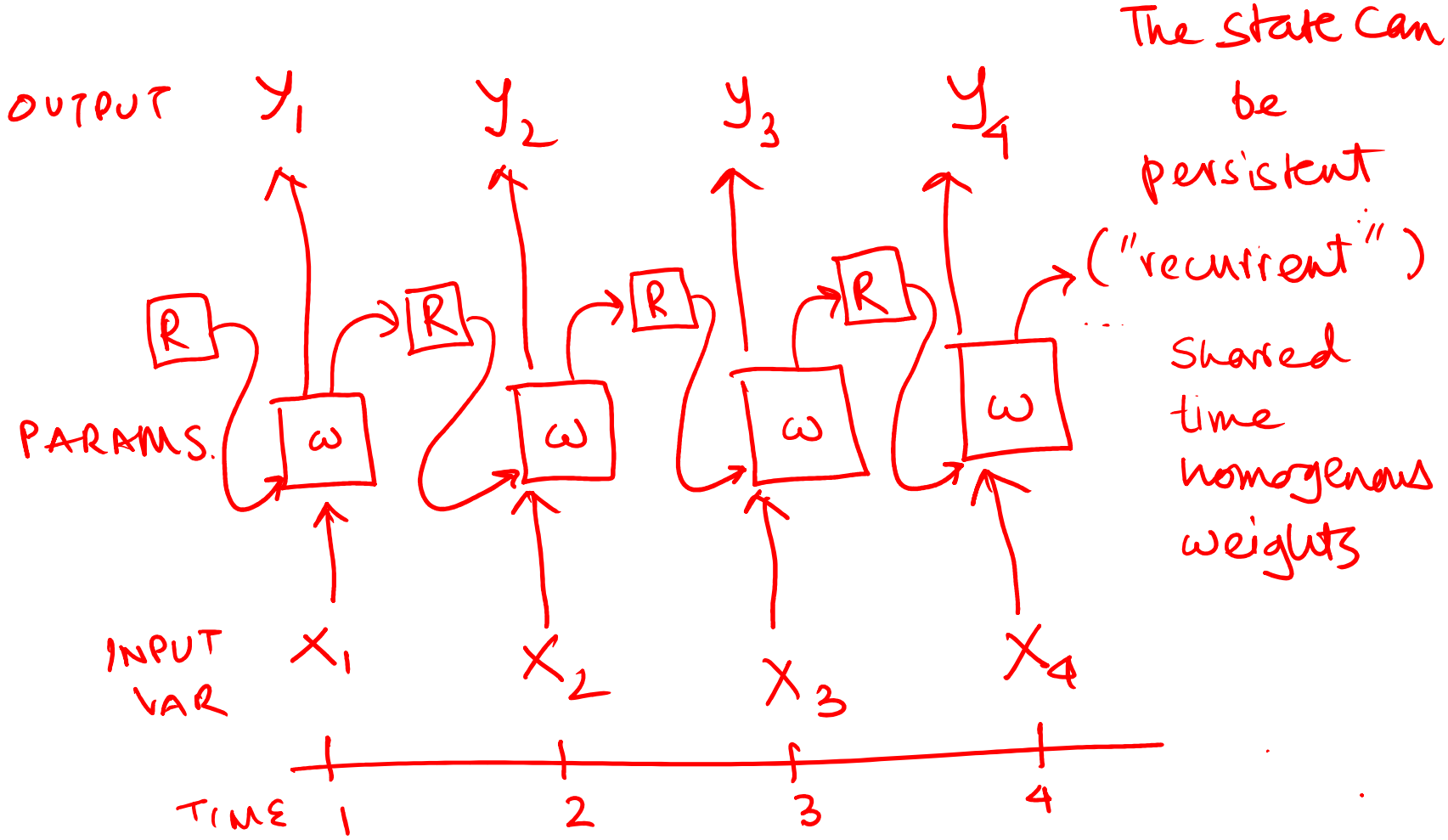


Using history

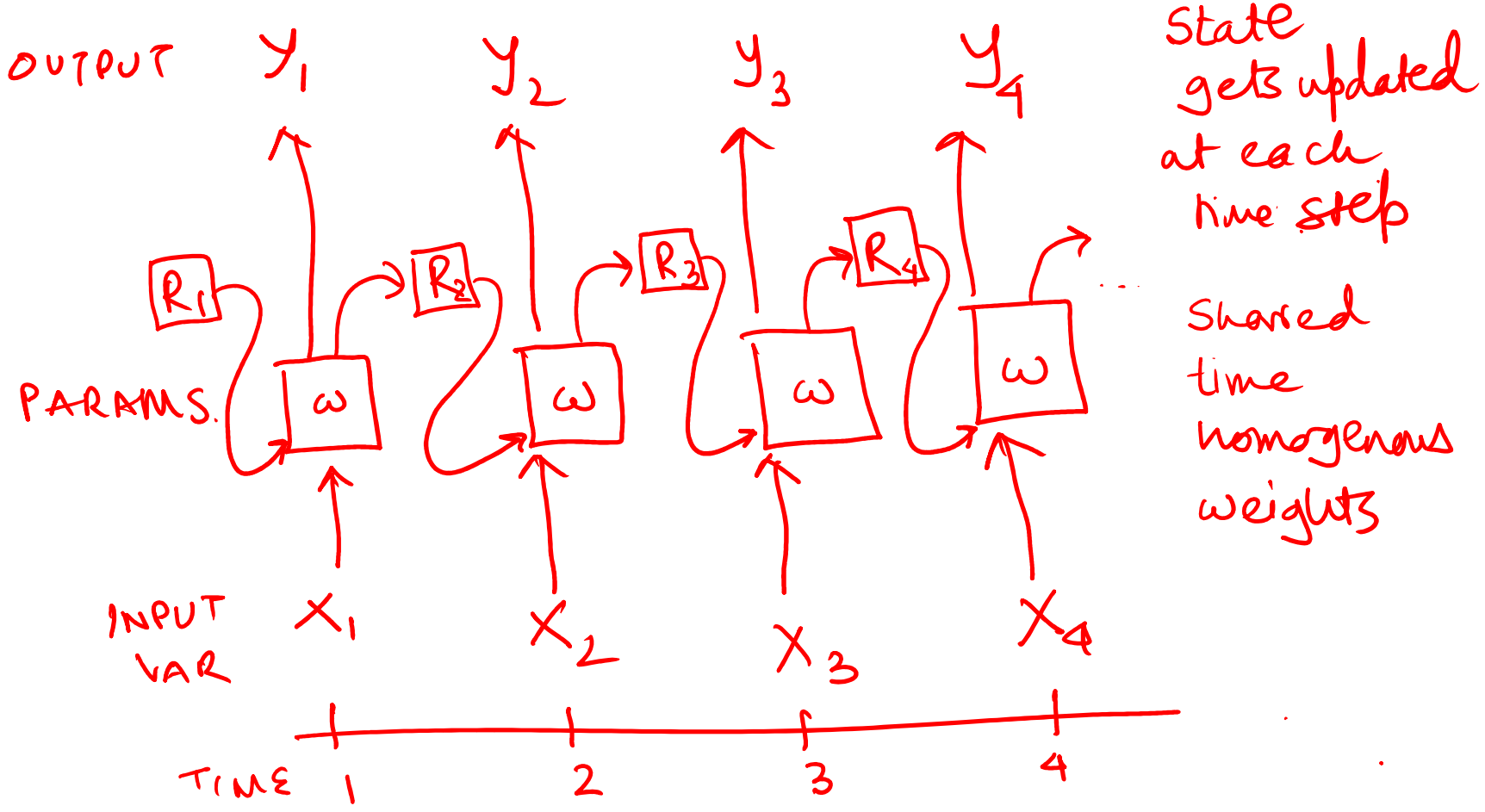
- What if all the previous timesteps can be fed in at time t ?



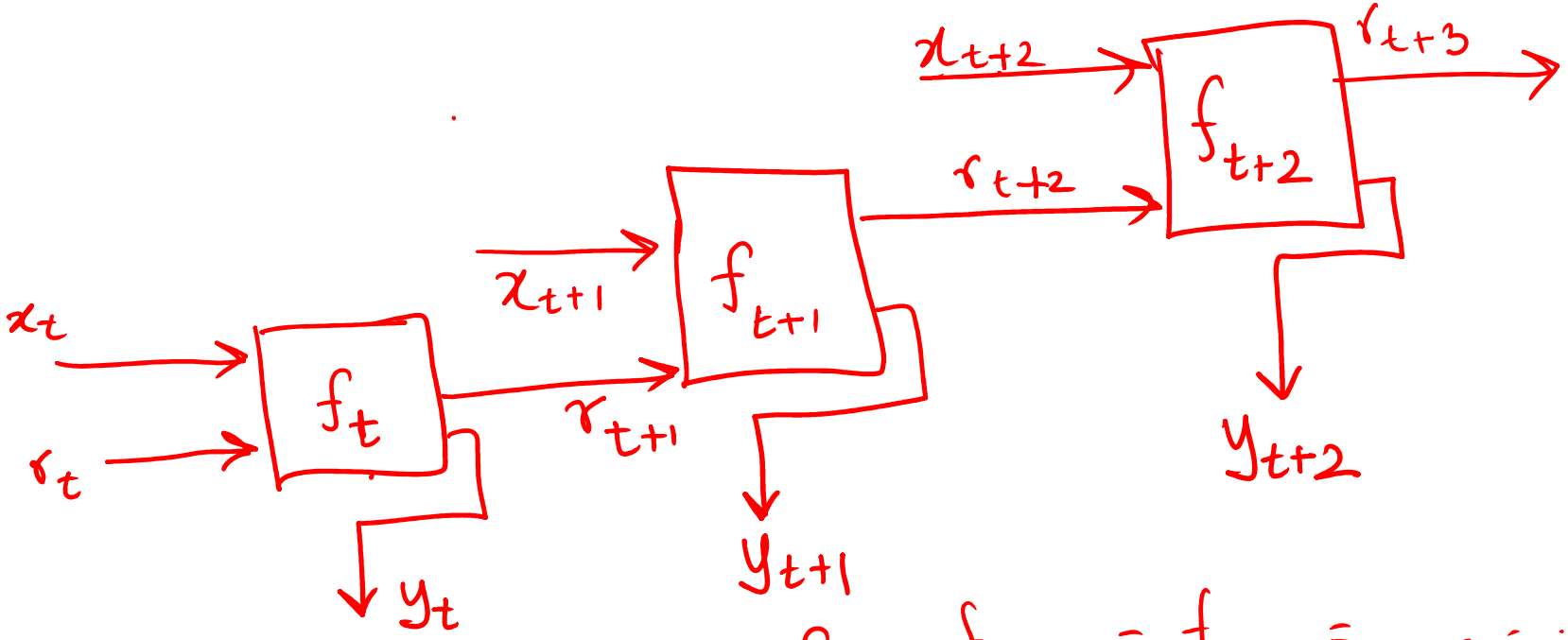
Designing an internal state



Designing an internal state



Linking input and output : unfolding in time

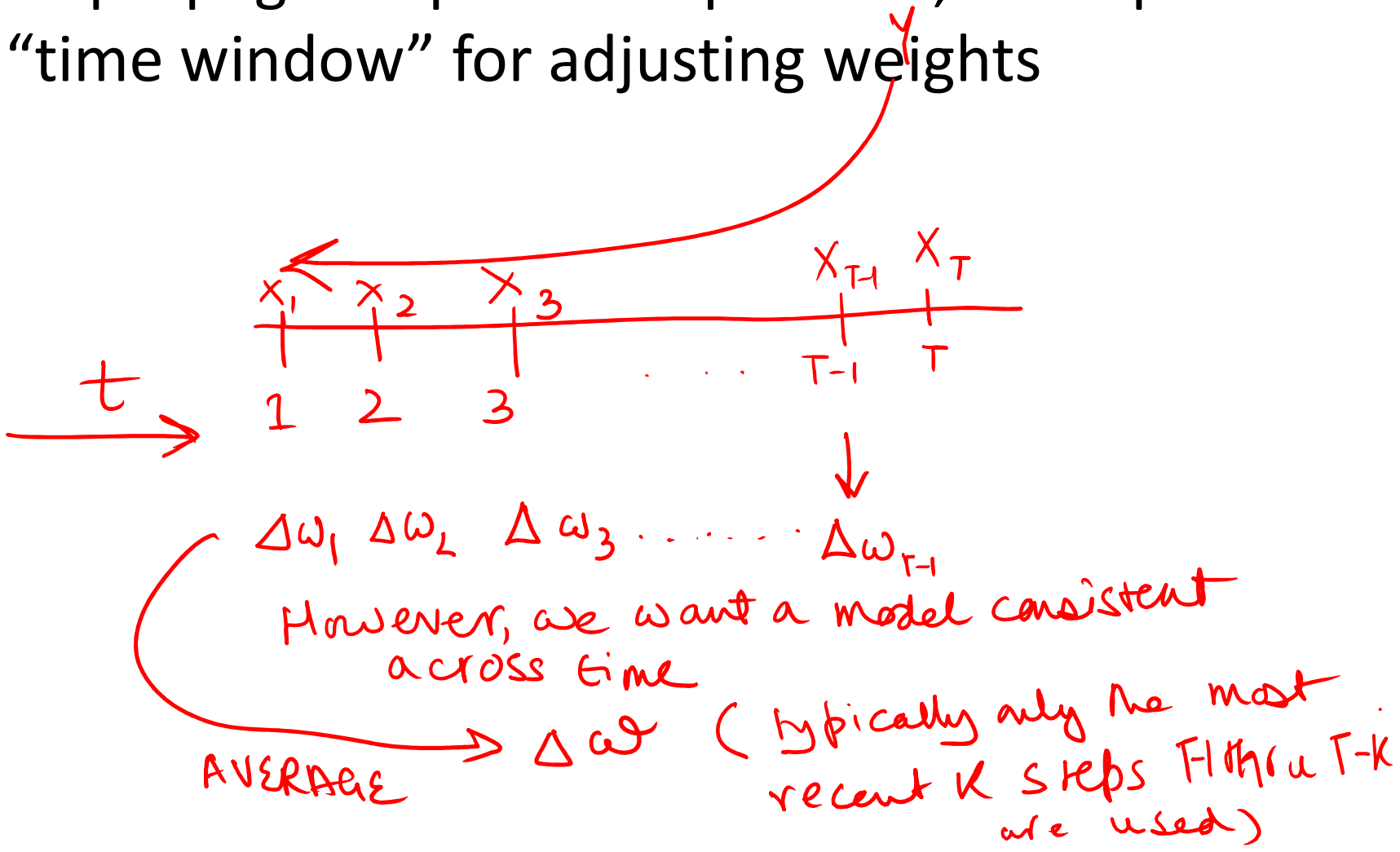


To share weights, $f_t = f_{t+1} = f_{t+2} = \ldots$

How to perform backprop?

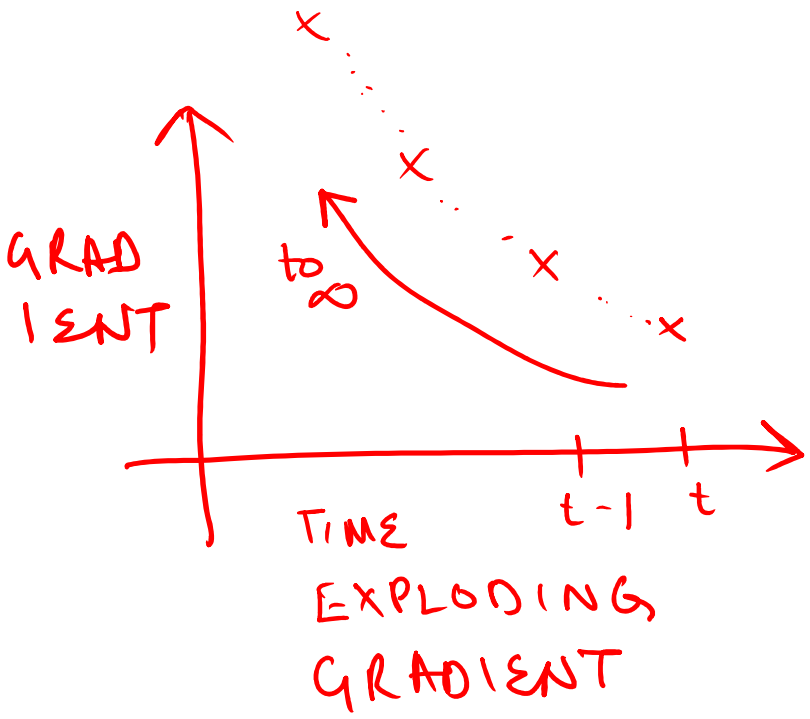
Backprop through time

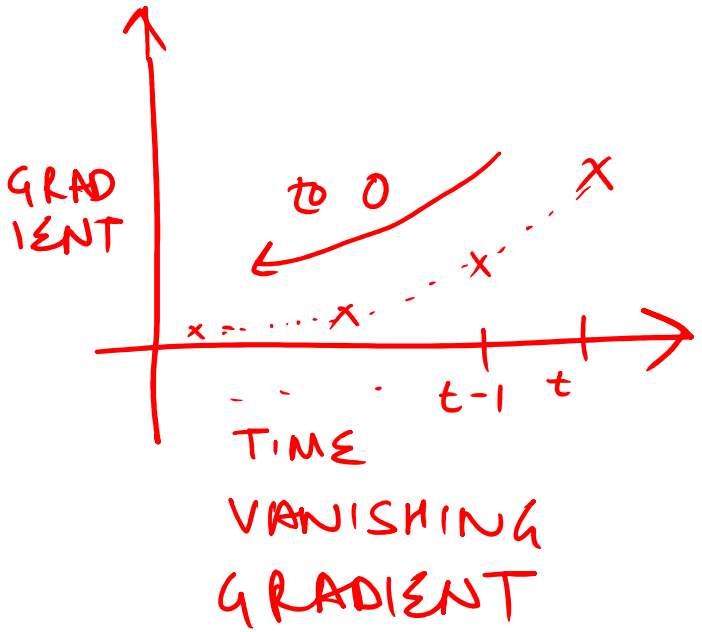
- Backpropagate upto t=0 if possible, else upto a "time window" for adjusting weights



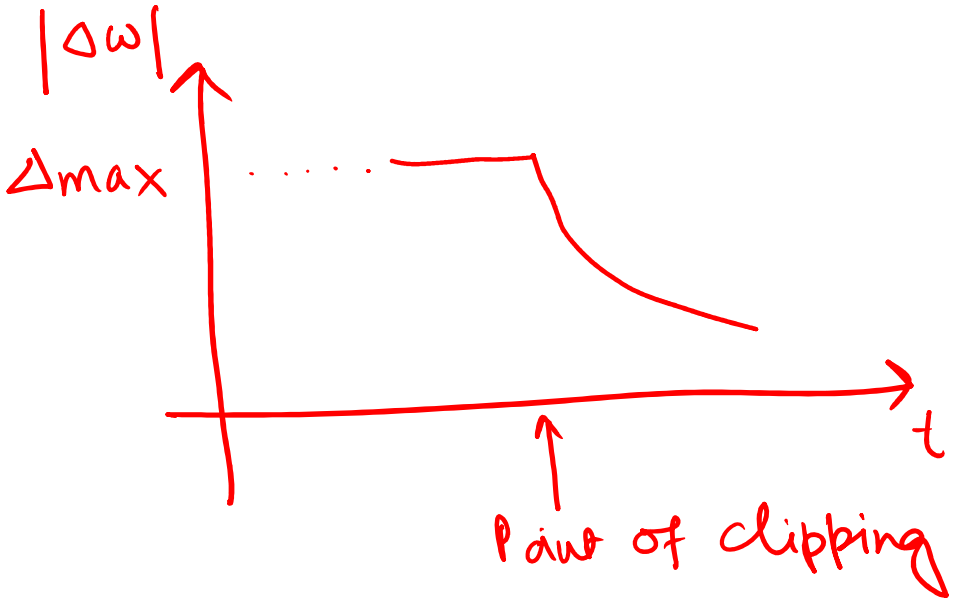
Dealing with feedback

- Standard feedback issues
  - Accentuation and dampening of weights





Exploding gradient : gradient clipping



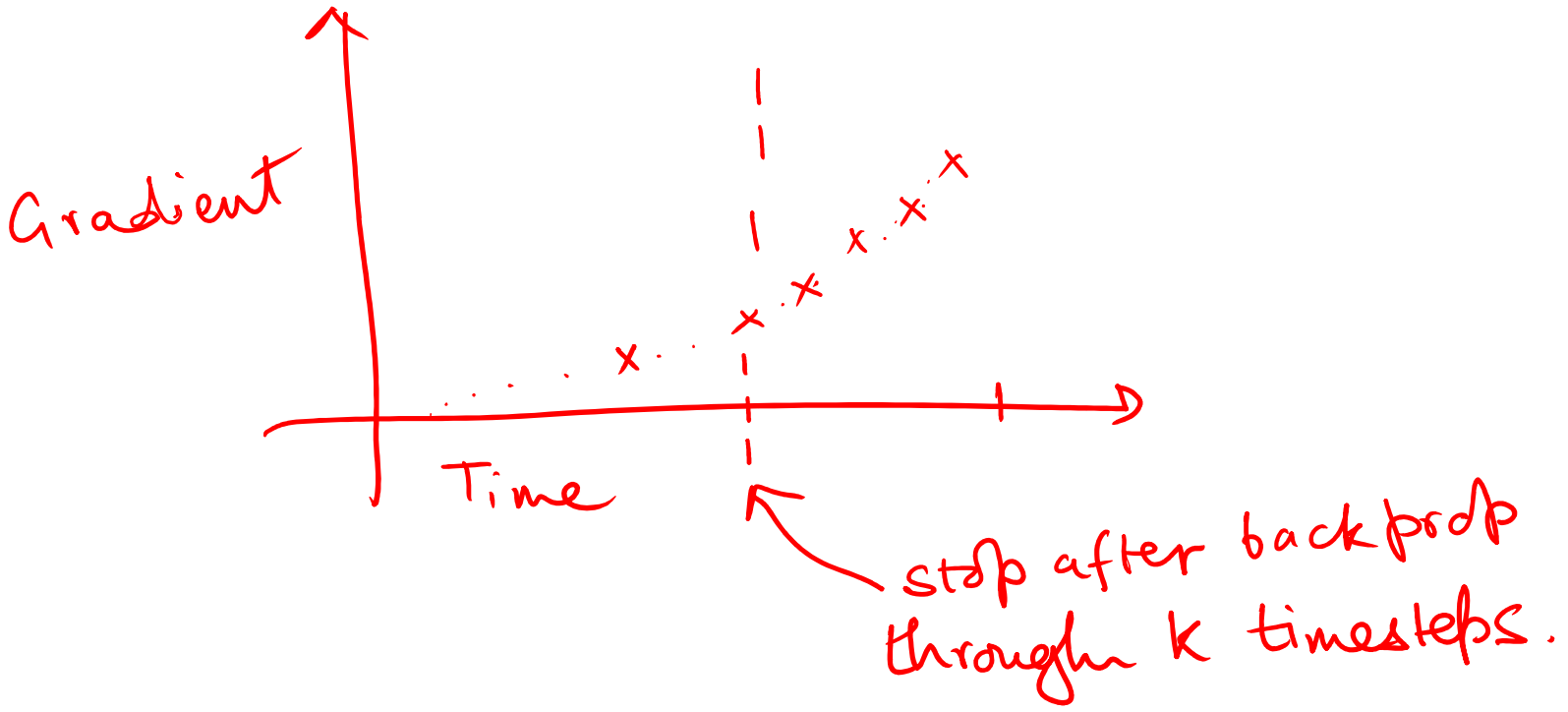
if $|\Delta w| > \Delta_{max}$

clipped Δw
$= \text{sign}(\Delta w) \times \Delta_{max}$

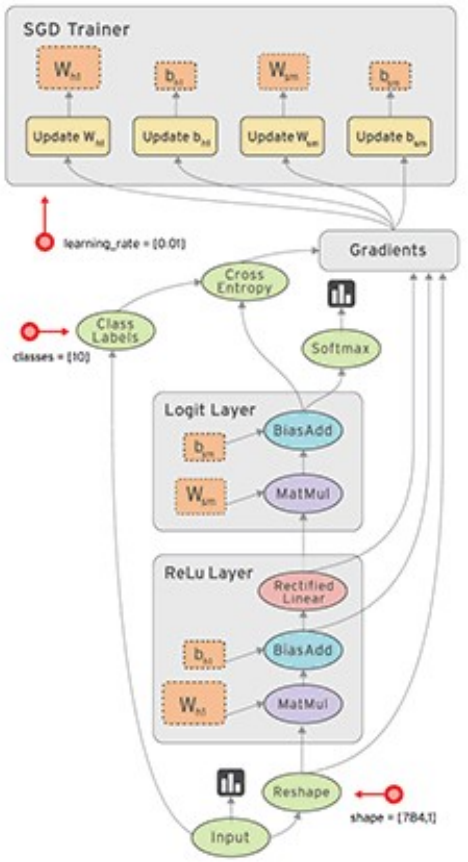
else
no clipping

Remember, $|\Delta w|$ is bounded, not $|w|$!

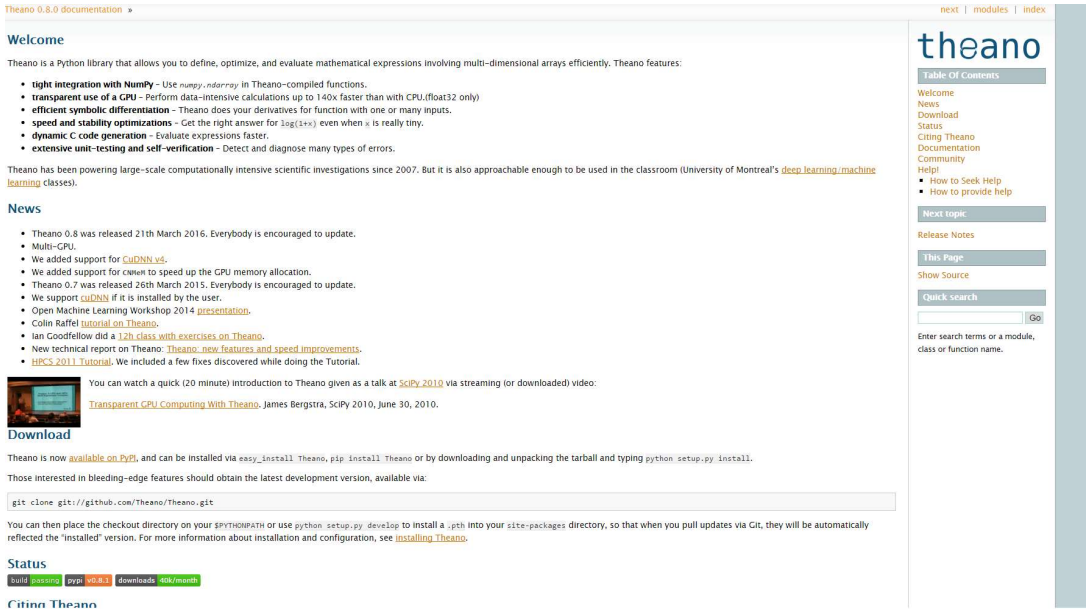
Vanishing gradient : limited backprop



What's out there



TensorFlow, Google's Deep Learning framework (open source)



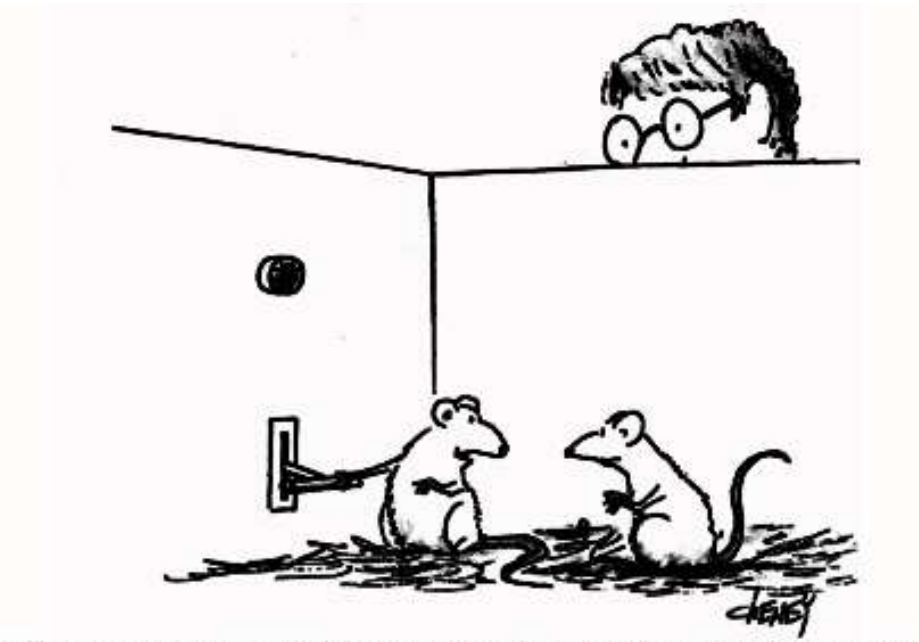
Theano, Python toolkit (open source)

Deep Genomics

Genomics Startup using deep learning

Limits of deep learning

- **Correlations cannot distinguish between implication, reverse implication and equivalence : what is the cause of good performance ?**
- **Semantics of learnt variables are grounded in scientific context : but hard to interpret parameters learnt by deep learning**
- **Adversarial samples cause issues in deep nets : bad for badly behaved data**
- **Lack estimates of uncertainty : how to quantify quality of prediction**
- **Many hyperparameters and parameters to tune : many ways to go wrong**



Terry's Lab Rat

Thanks !

- Further reading :
  - http://deeplearning.cs.cmu.edu : a course by Bhiksha Raj
  - http://www.nature.com/nbt/journal/v33/n8/full/nbt.3313.html
  - http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html