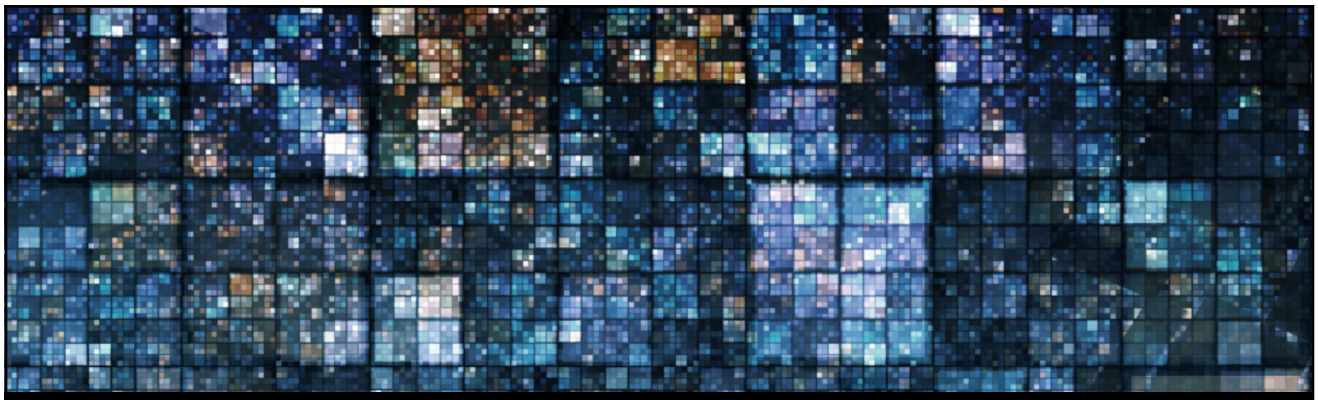
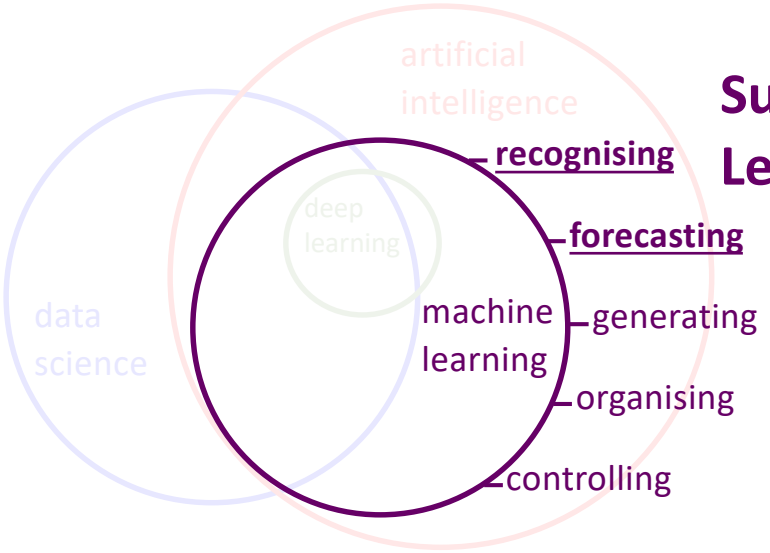


COMP47590
Advanced Machine Learning
Deep Learning - Regularisation

Dr. Brian Mac Namee

Information

Email:	Brian.MacNamee@ucd.ie
Course Materials:	All material posted on UCD CS moodlehttps://csmoodle.ucd.ie/moodle/course/view.php?id=663 Enrolment key **UCDAdvML2019**



Supervised Learning

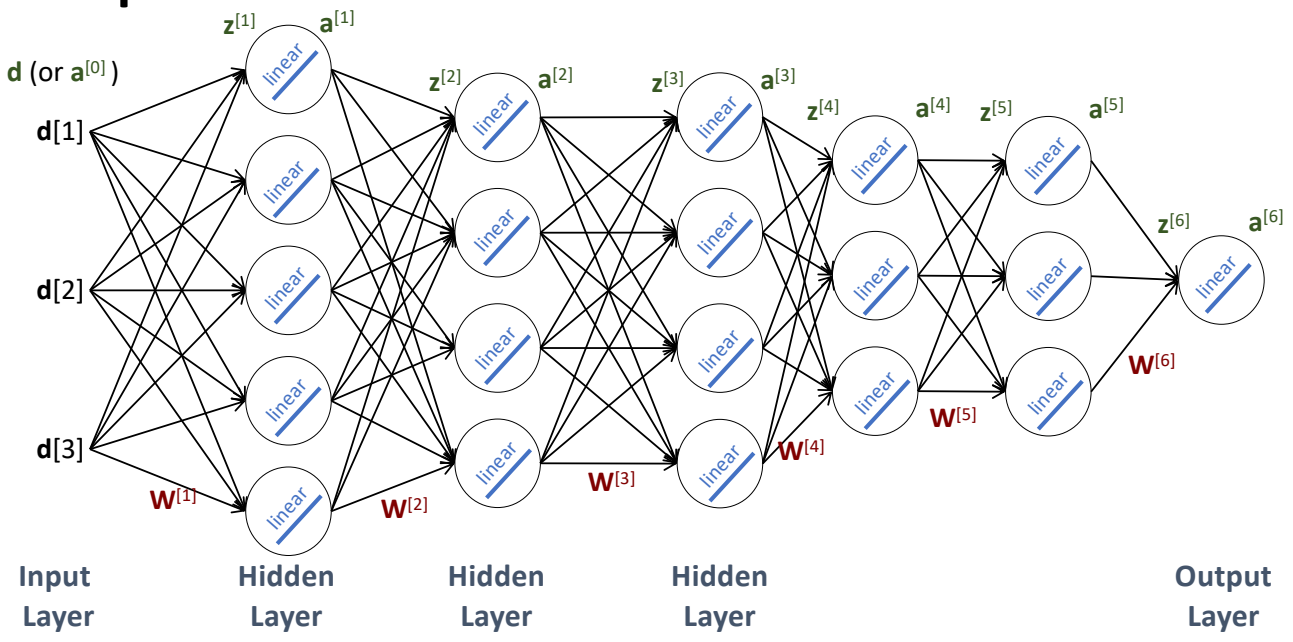
- recognising
- forecasting
- generating
- organising
- controlling

VANISHING/EXPLODING GRADIENTS

Vanishing/Exploding Gradients

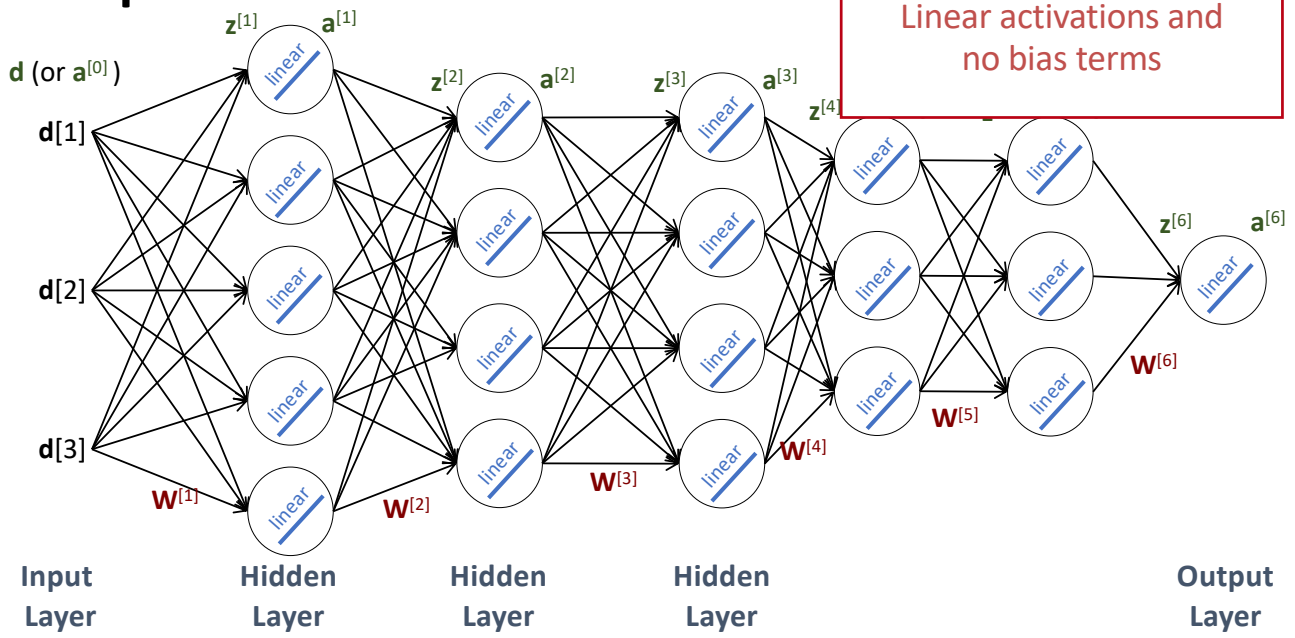
Early attempts at deep learning were often scuppered by what as referred to as **exploding** and **vanishing** gradients

Deep Feed Forward Network

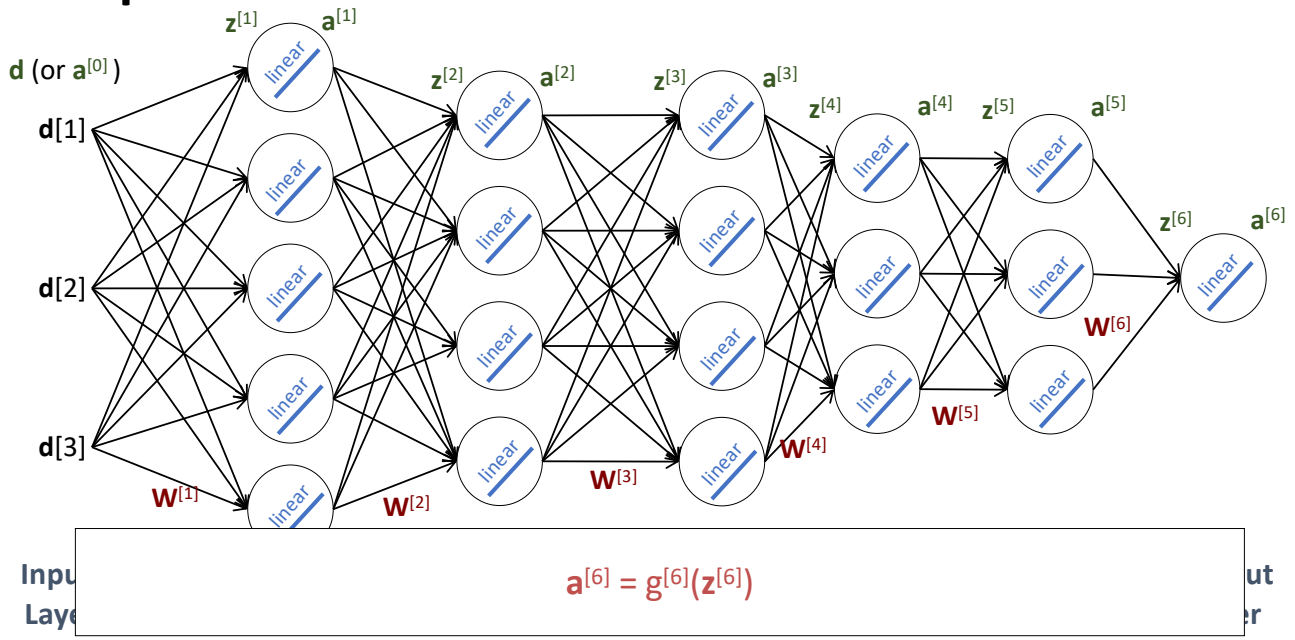


Deep Feed Forward Network

Linear activations and no bias terms

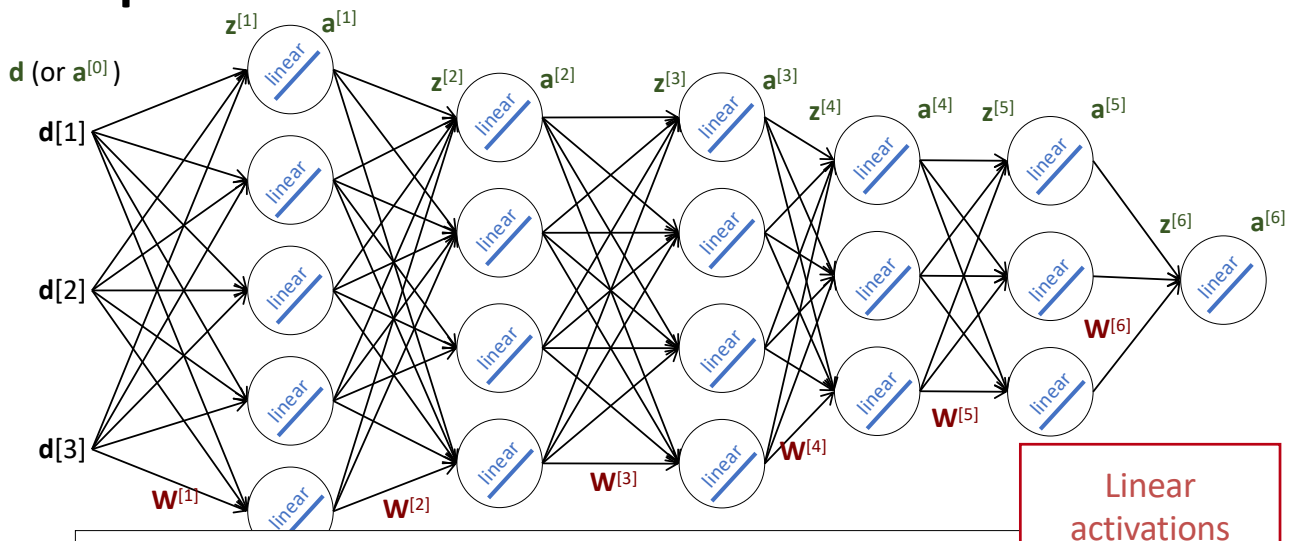


Deep Feed Forward Network

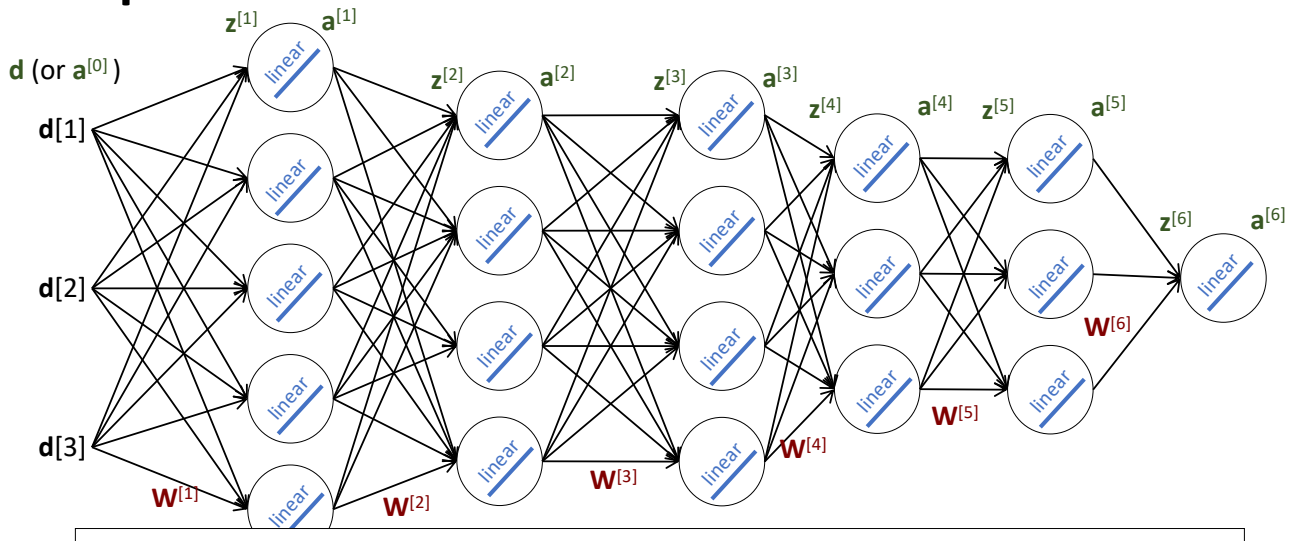


$a^{[6]} = g^{[6]}(z^{[6]})$

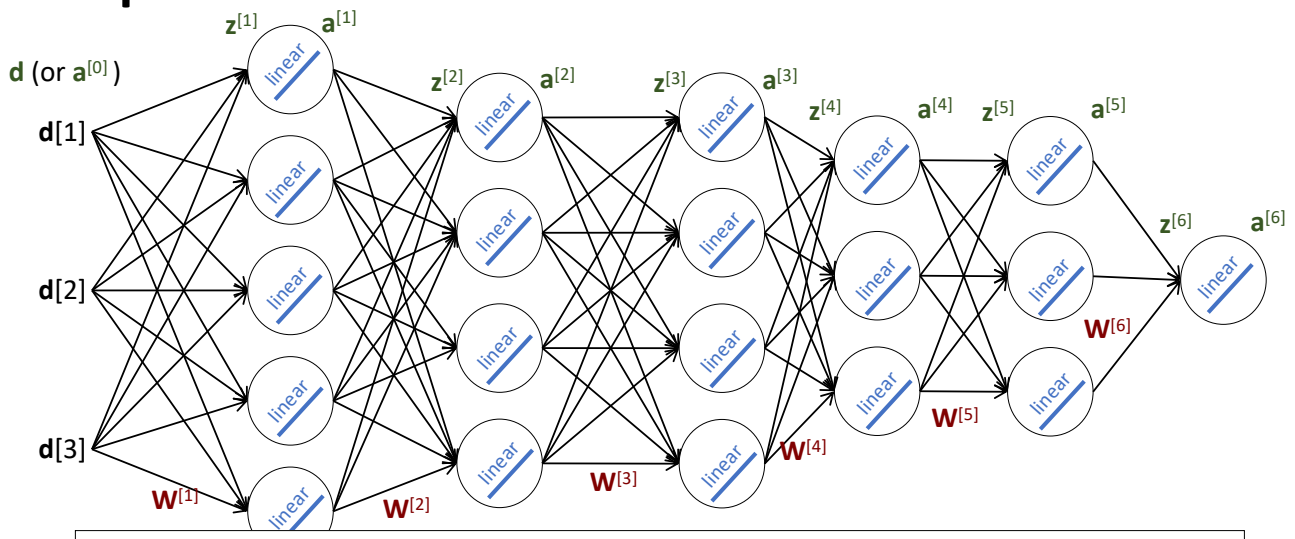
Deep Feed Forward Network



Deep Feed Forward Network



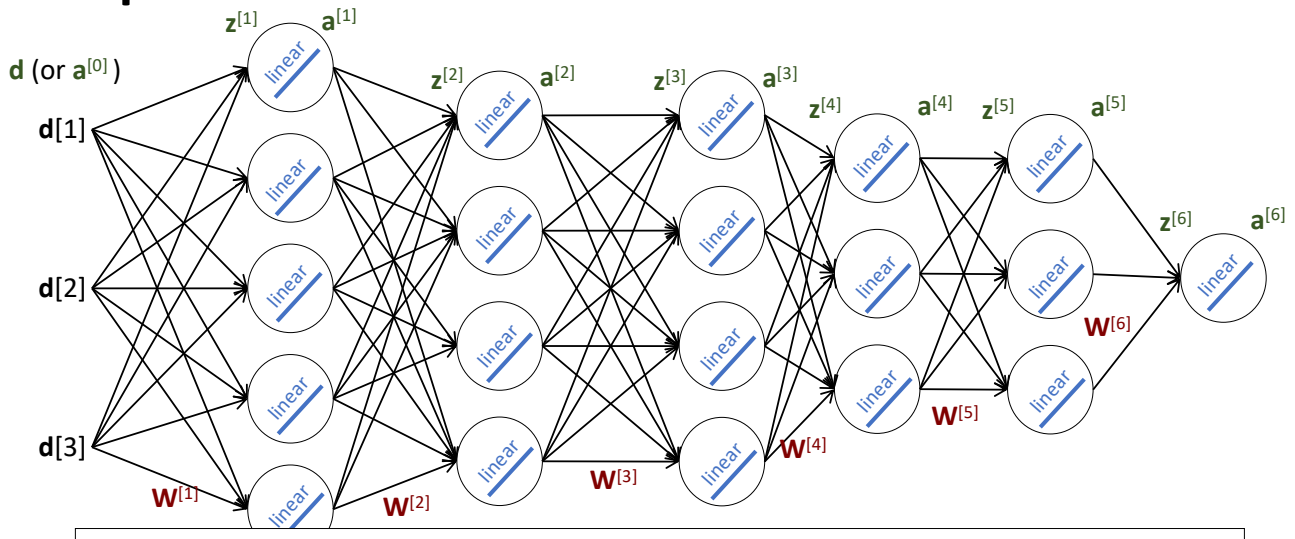
Deep Feed Forward Network



Inpu ut
Laye er

$$a^{[6]} = W^{[6]} (W^{[5]} a^{[4]})$$

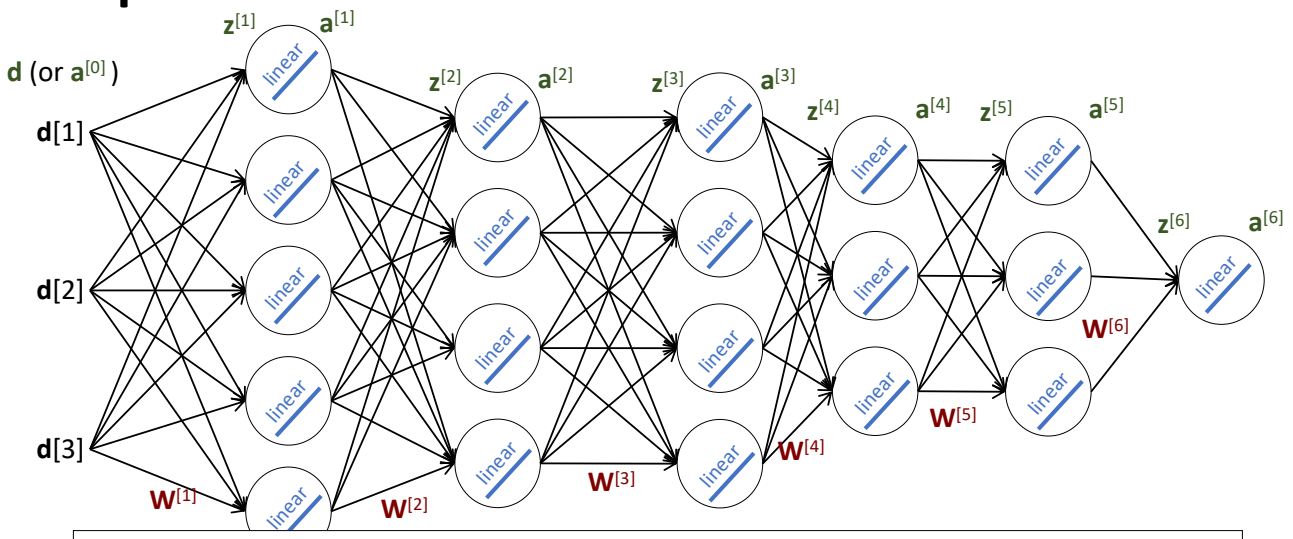
Deep Feed Forward Network



Inpu ut
Laye er

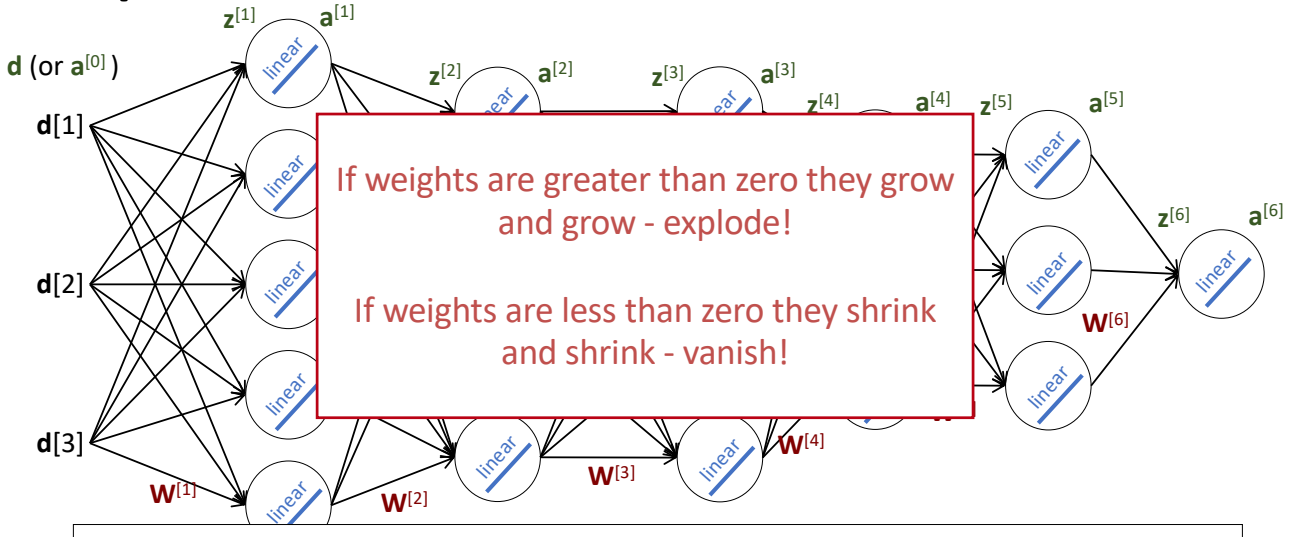
$$a^{[6]} = W^{[6]} (W^{[5]} (W^{[4]} a^{[3]}))$$

Deep Feed Forward Network



$$a^{[6]} = W^{[6]}\,(W^{[5]}\,(W^{[4]}\,(W^{[3]}\,(W^{[2]}\,(W^{[1]}\,d)))))$$

Deep Feed Forward Network



If weights are greater than zero they grow and grow - explode!

If weights are less than zero they shrink and shrink - vanish!

$$a^{[6]} = W^{[6]}\,(W^{[5]}\,(W^{[4]}\,(W^{[3]}\,(W^{[2]}\,(W^{[1]}\,d)))))$$

Combating Exploding & Vanishing Gradients

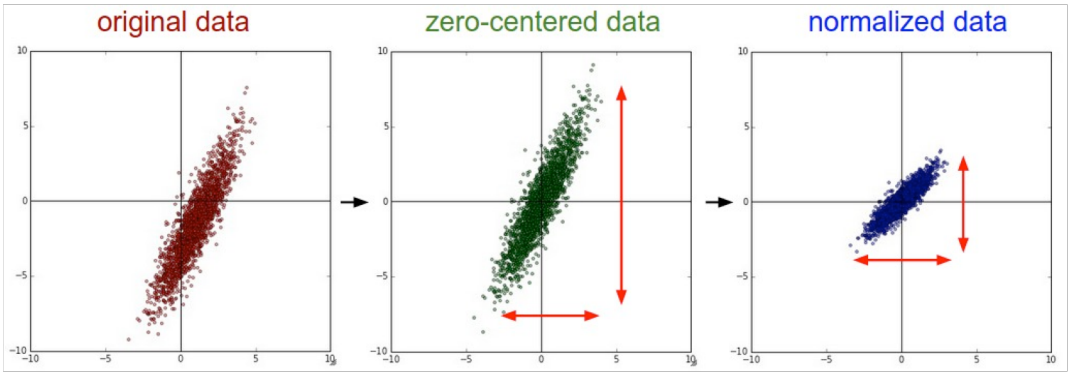
There are a few ways that have been shown to combat vanishing and exploding gradients

- Careful initialisation
- Regularisation
- Gradient clipping

STANDARDISING INPUTS & CHOOSING INITIAL WEIGHTS

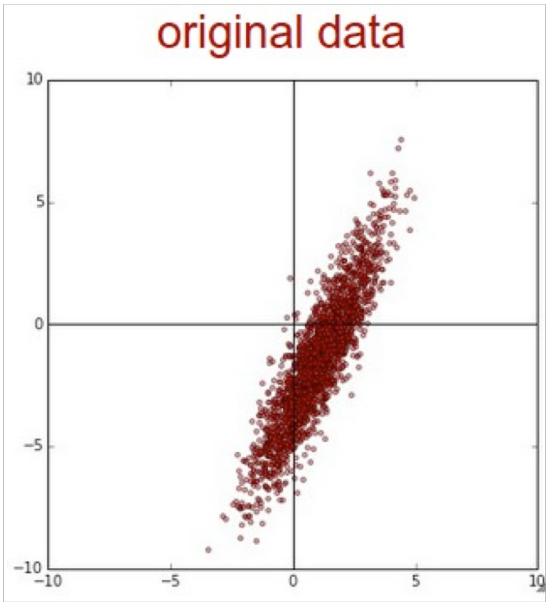
Standardising Inputs

Always normalise/standardise inputs - just makes everything behave better



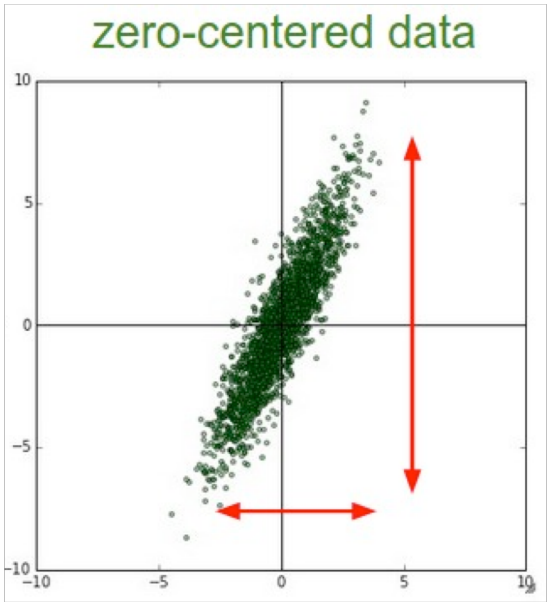
http://cs231n.github.io/neural-networks-2/

Standardising Inputs



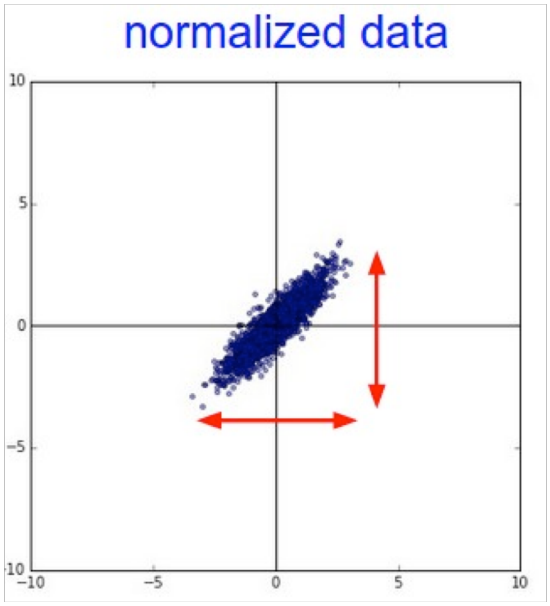
http://cs231n.github.io/neural-networks-2/

Standardising Inputs



http://cs231n.github.io/neural-networks-2/

Standardising Inputs



http://cs231n.github.io/neural-networks-2/

Choosing Initial Weights

There are different approaches to choosing initial weights

- Set all to zero - **never do this**
- Choose random values in fixed range, e.g. (-0.2, 0.2)
- Biases are often initialised to 0.01 or similar small value

Choosing Initial Weights

There are different approaches to choosing initial weights

- Scale weights with number of inputs -
  - Choose from a standard normal distribution but set variance to 1/n where n is the number of inputs
  - Scale random values with $\sqrt{\frac{1}{|\mathbf{a}^{[l-1]}|}}$

REGULARISATION

Regularisation

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

Goodfellow et al, 2016

Goodfellow, Bengio, Courville, Deep Learning, MIT Press, 2016

Regularisation

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right)$$

Regularisation

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right)$$

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right) + \lambda \times REG$$

where *m* is the size of the training dataset, λ is a hyper-parameter controlling the impact of regularisation, and *REG* is the regularisation term

Regularisation Methods For Regression

For regression models there are three common regularisation approaches:

- lasoo method
- ridge method
- elastic net method

Regularisation Methods For Regression
Lasoo

Least absolute shrinkage and selection operator (**lasoo**) method is an early regularisation approach

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right) + \frac{\lambda}{2m} \times ||\mathbf{w}||_1$$

where $||\mathbf{w}||_1 = \sum_j |\mathbf{w}_j|$

Regularisation Methods For Regression
Lasso

lasso regularisation is also known as L_1 regularisation

lasso regularisation can encourage **sparseness** within a weight matrix - weights set to zero

Regularisation Methods For Regression
Ridge

Ridge regularisation uses an L_2 regularisation and is written as

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right) + \frac{\lambda}{2m} \times ||\mathbf{w}||^2$$

where $||\mathbf{w}||^2 = \sum_j \mathbf{w}_j^2$

Regularisation Methods For Regression
Ridge

Ridge regularisation will not result ins sparse weights

Also known as *weight decay* or *parameter shrinkage*

Regularisation Methods For Regression
Elastic Net

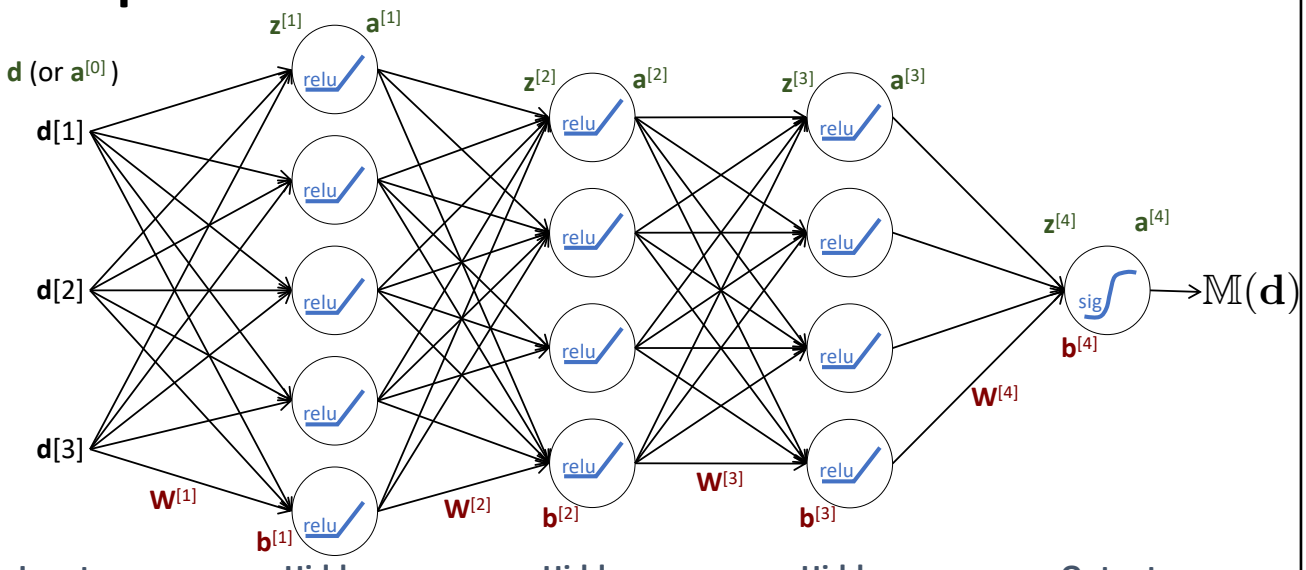
Elastic net regularisation combines lasso and ridge regularisations

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right) + \frac{\lambda_1}{2m} ||\mathbf{w}||_1 + \frac{\lambda_2}{2m} ||\mathbf{w}||^2$$

note that we now have two hyper-parameters λ_1 and λ_2

REGULARISATION IN NEURAL NETWORKS

Deep Feed Forward Network



L_2 Regularisation

In a deep neural network we have a set of weight matrices to which we apply an L_2 regularisation

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right) + \frac{\lambda}{2m} \sum_{l} \left\| \mathbf{W}^{[l]} \right\|_2^2$$

where $\left\| \mathbf{W}^{[l]} \right\|_2^2 = \sum_j \sum_k w_{j,k}^2$

L_2 Regularisation

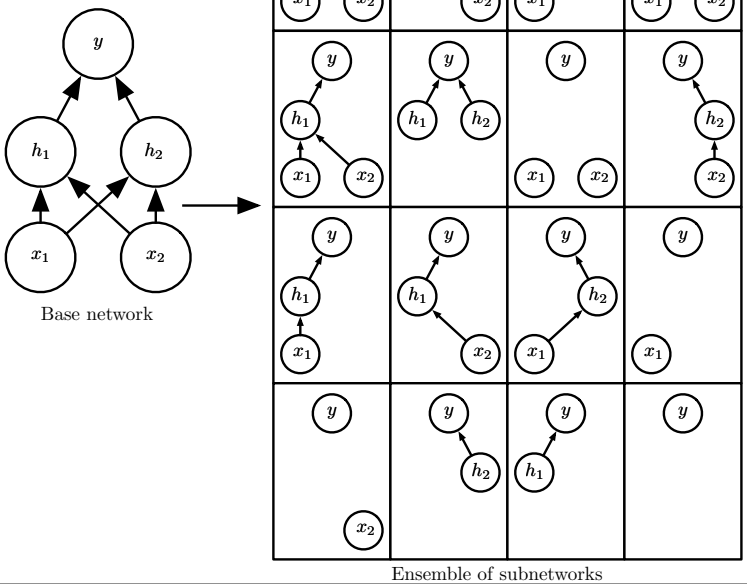
The weight update rule needs to then adjust slightly to take account of the regularisation term

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \left(d\mathbf{W}^{[l]} + \frac{\lambda}{m} \mathbf{W}^{[l]} \right)$$

Note: We typically don't bother applying regularisation to bias terms (although we can!)

INVERTED DROPOUT REGULARISATION

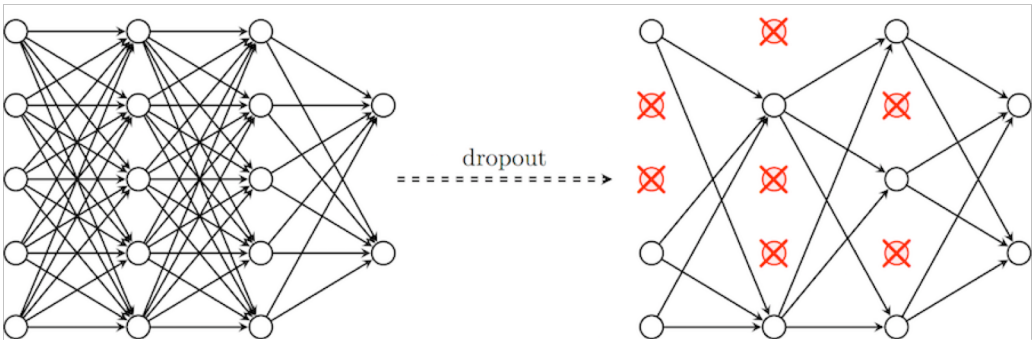
Regularisation: Dropout



Goodfellow, Bengio, Courville, Deep Learning, MIT Press, 2016

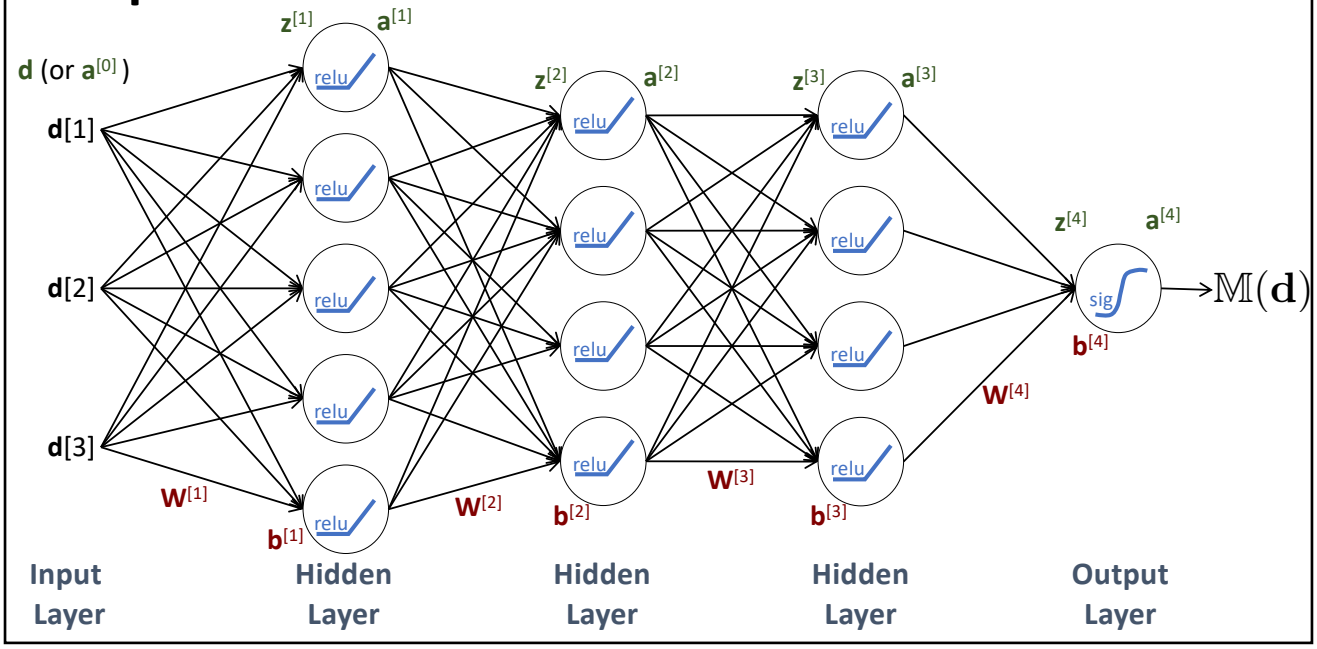
Inverted Dropout

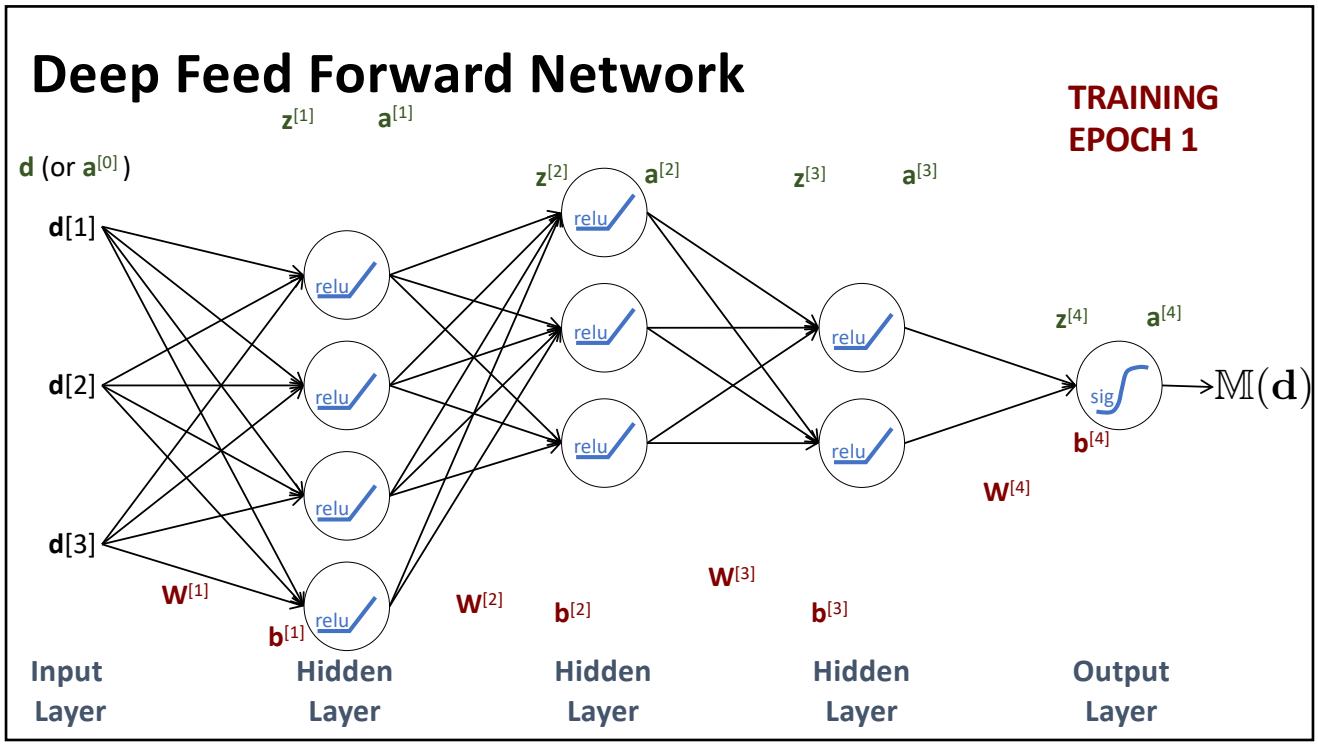
During training randomly drop out units according to a dropout probability at each training epoch

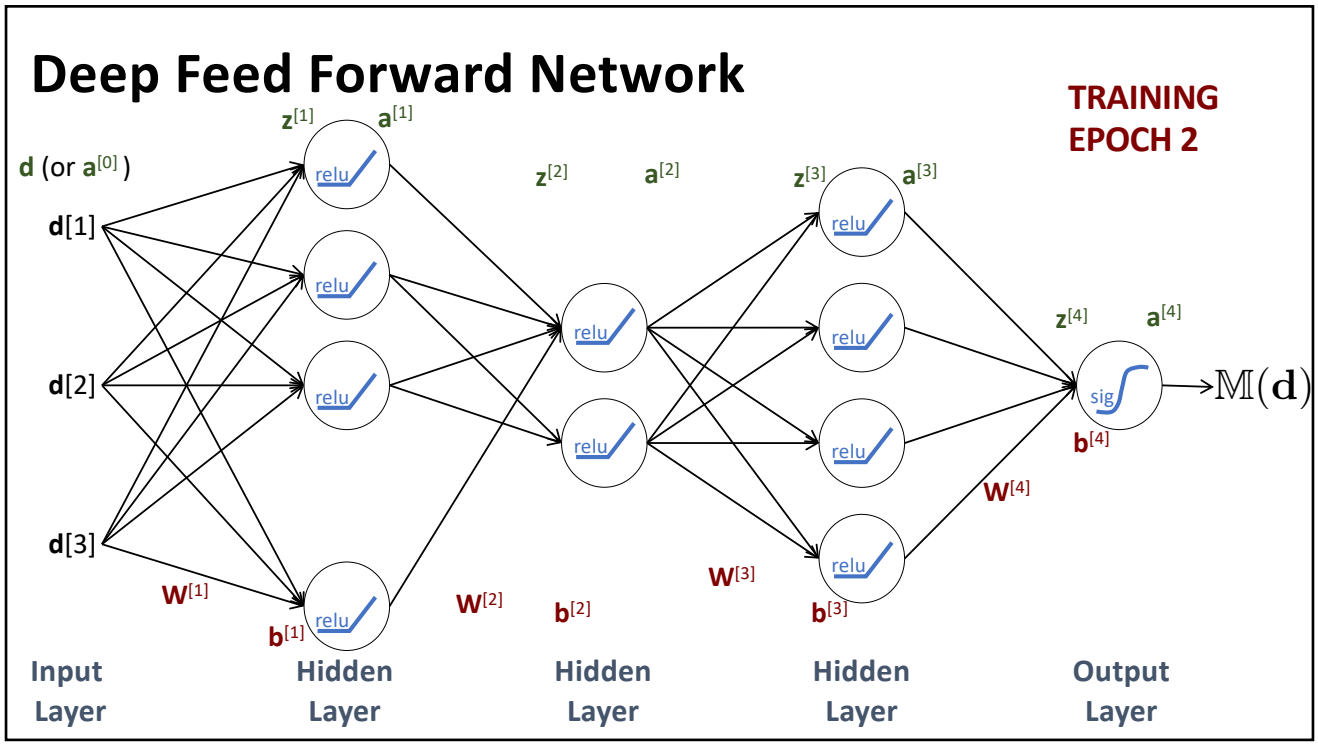


Credit to Petar Veličković

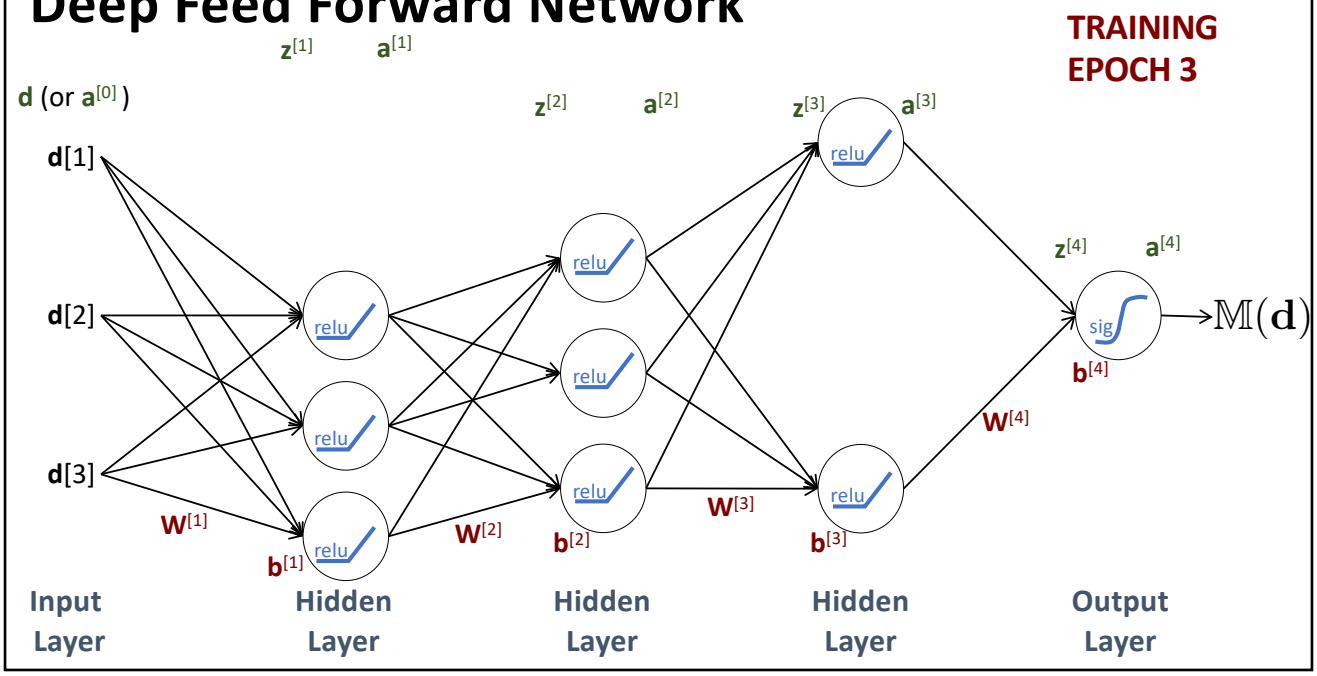
Deep Feed Forward Network




Deep Feed Forward Network
TRAINING EPOCH 1
d (or a[0])
d[1]
d[2]
d[3]
z[1]
a[1]
z[2]
a[2]
z[3]
a[3]
z[4]
a[4]
relu
sig
W[1]
b[1]
W[2]
b[2]
W[3]
b[3]
W[4]
b[4]
Input Layer
Hidden Layer
Hidden Layer
Hidden Layer
Output Layer


Deep Feed Forward Network
TRAINING EPOCH 2
d (or a[0])
d[1]
d[2]
d[3]
z[1]
a[1]
z[2]
a[2]
z[3]
a[3]
z[4]
a[4]
relu
sig
W[1]
b[1]
W[2]
b[2]
W[3]
b[3]
W[4]
b[4]
Input Layer
Hidden Layer
Hidden Layer
Hidden Layer
Output Layer

Deep Feed Forward Network



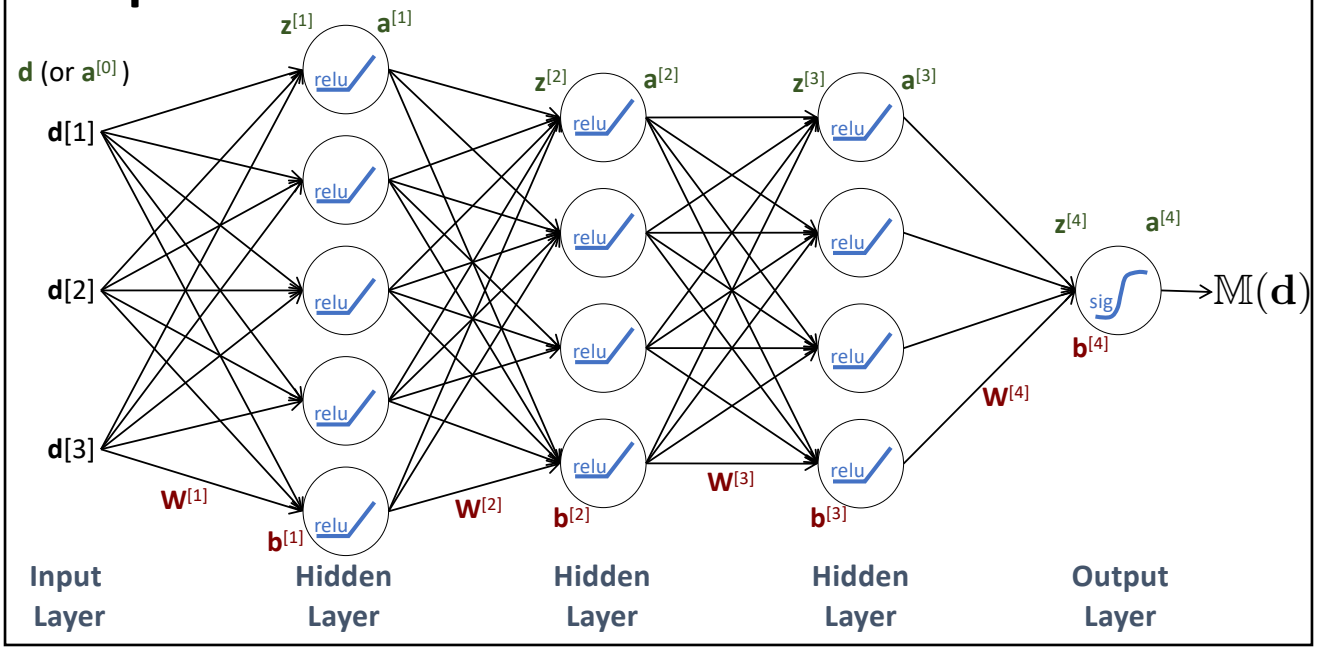
Inverted Dropout

During training scale up output of remaining units by dividing by dropout probability

- Ensures that overall activations remain in the same range throughout training process

When making predictions don't do anything different - no dropout applied

Deep Feed Forward Network



Why Does Dropout Work?

Spreads out weight because the model cannot rely on any one input

- Similar to L_2 regularisation

Unfortunately dropout introduces another hyper-parameter: dropout probability

- Often set differently for each layer - higher for layers with large numbers of weights

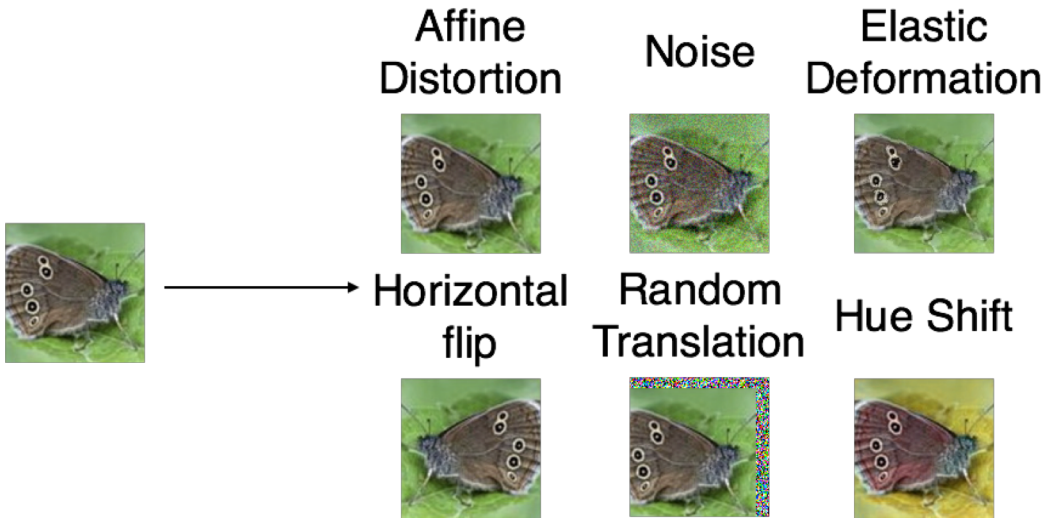
DATASET AUGMENTATION REGULARISATION

Dataset Augmentation

Synthesise examples by flipping, rotating, cropping, distorting, etc

Add these to the training dataset - makes dataset more robust

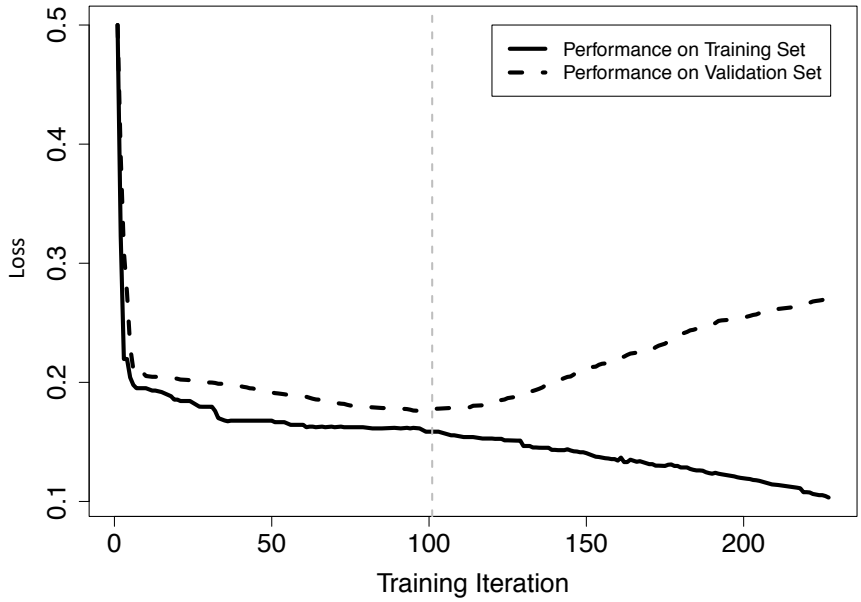
Dataset Augmentation



Goodfellow, Bengio, Courville, Deep Learning, MIT Press, 2016

EARLY STOPPING

Early Stopping



Early Stopping

Allow a network to overfit and then roll back to the poont at which loss curve on training set and validation set start to diverge

SUMMARY

Summary

There are lots of modifications to the basic gradient descent algorithm that we can introduce to improve the learning process and make it more likely that we arrive at accurate models quickly

Some important levers for this are:

- Model initialisation
- Regularisation
- Early stopping

Summary

There is no one answer, however, and new approaches are always being suggested

We can't avoid experimentation, however

Questions