Grasping Deep Learning from Fundamentals to Applications

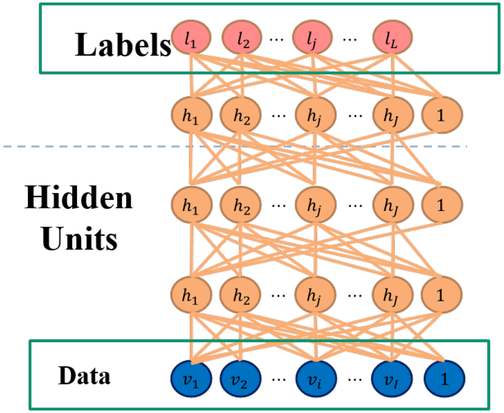
June 15, 2023

Lecture 1 – Intro to Deep Learning

Instructors: **Yufei Huang**, PhD; **Arun Das**, PhD

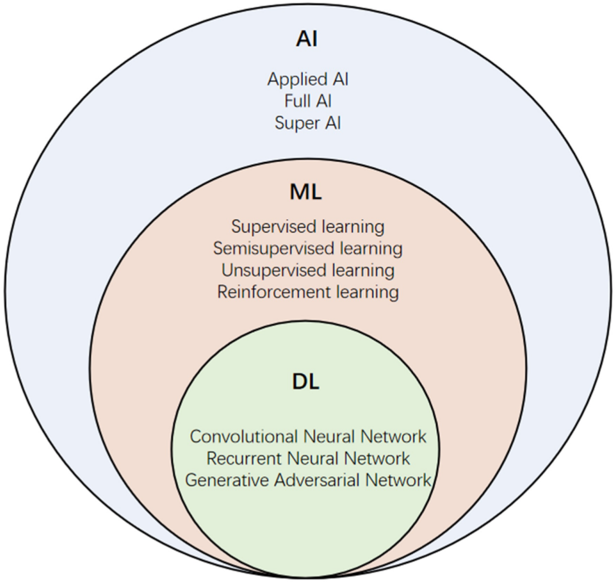
University of Pittsburgh

What is deep learning?



What is deep learning?

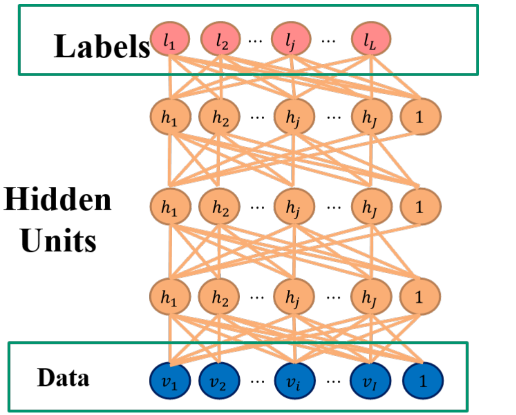
- A model paradigm
  - Neural networks
- A learning paradigm
  - Supervised
  - Unsupervised
  - Reinforcement
  - Self-supervised
  - Transfer
  - Contrastive
  - …

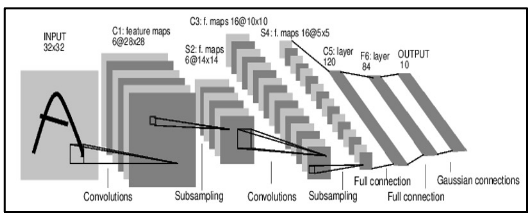


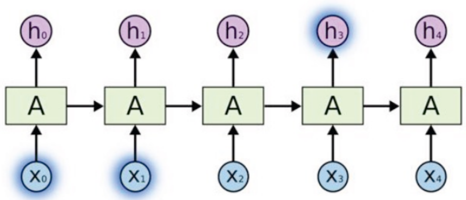
University of Pittsburgh

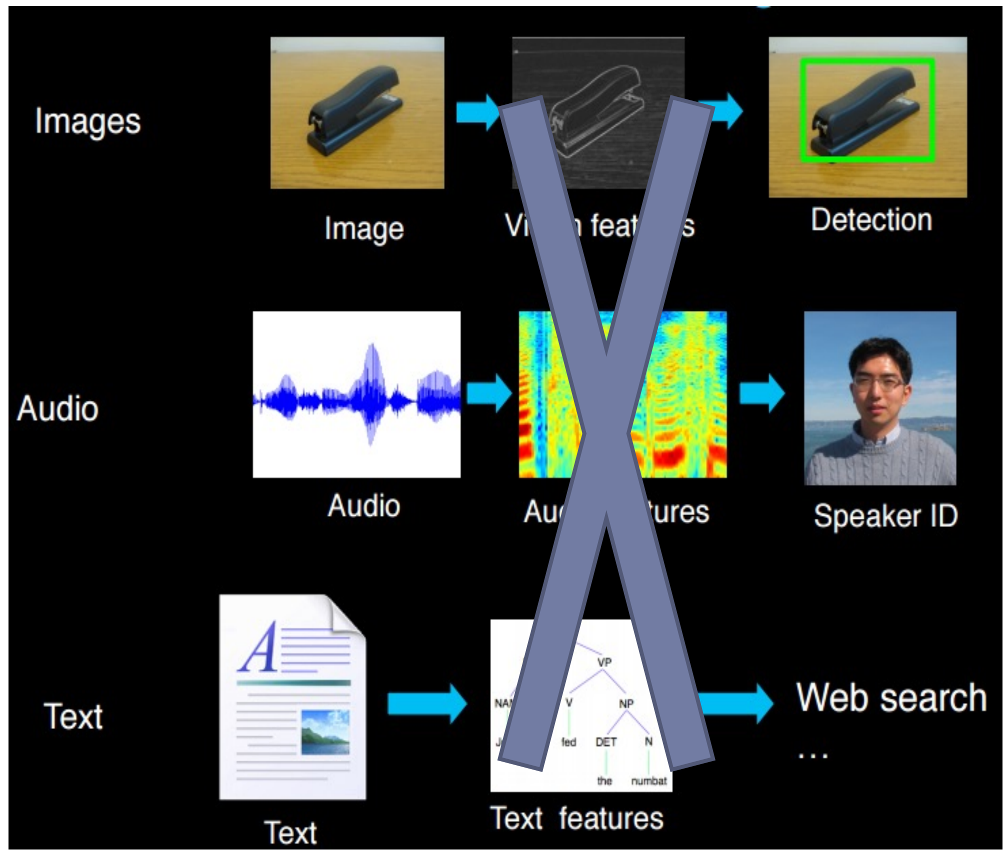
Deep Learning brings deep revolution for AI

Simplify AI systems





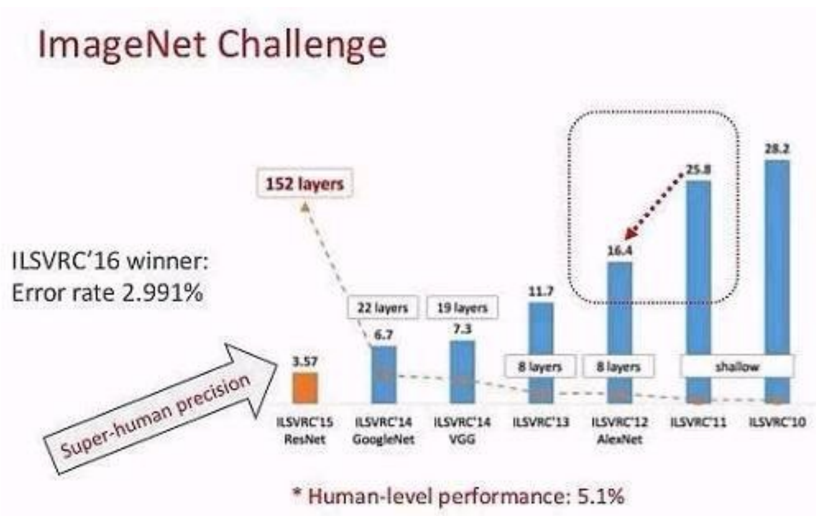




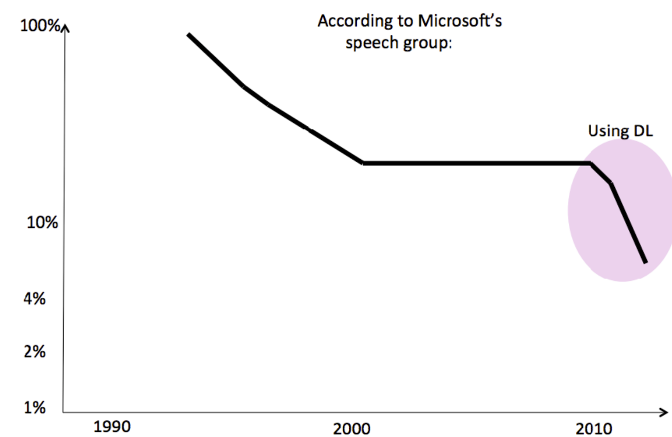
Deep Learning brings deep revolution for AI

Maximize AI performance

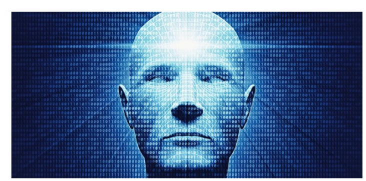
Computer vision



Speech recognition



Deep Learning in Speech Recognition

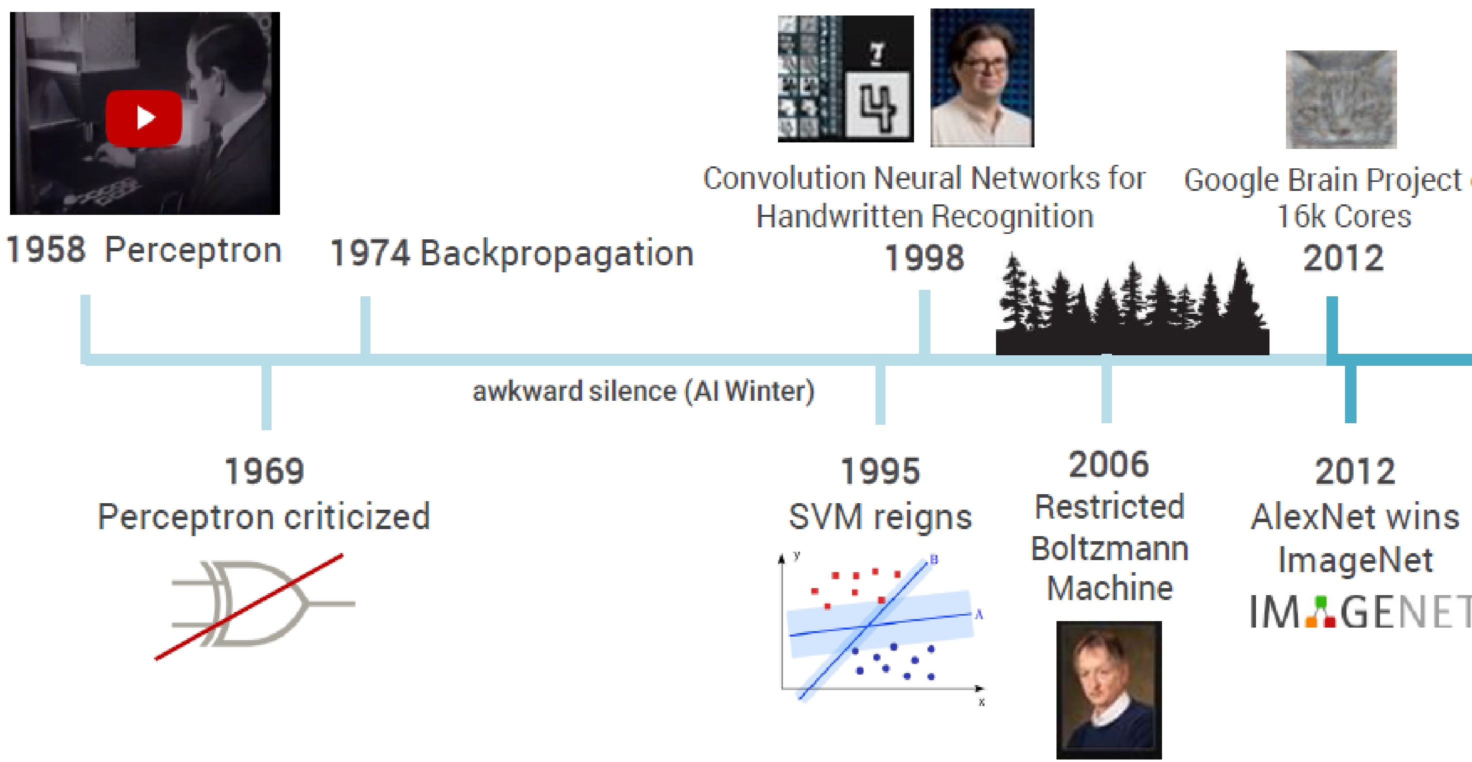


BRIEF

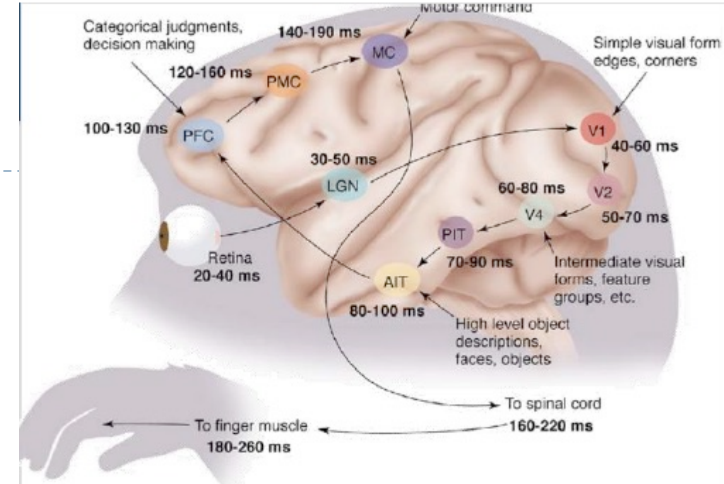
Microsoft achieves 'human parity' in speech recognition system

University of Pittsburgh

History of DL

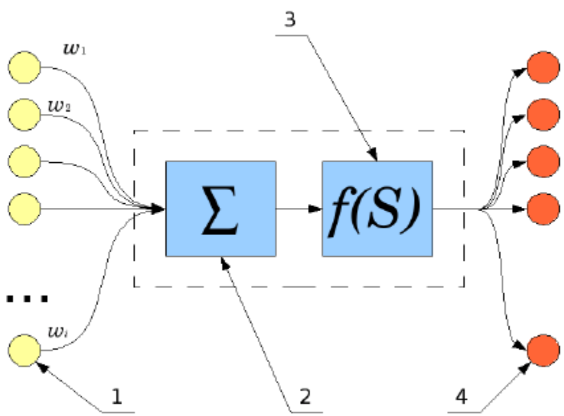


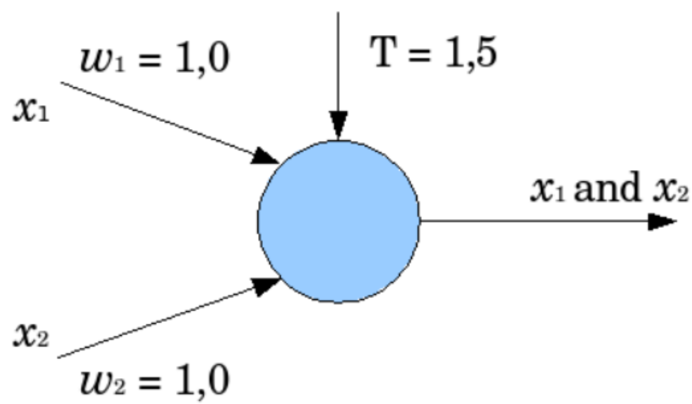
History of DL – Artificial Neuron, 1943



McCulloch and Pitts

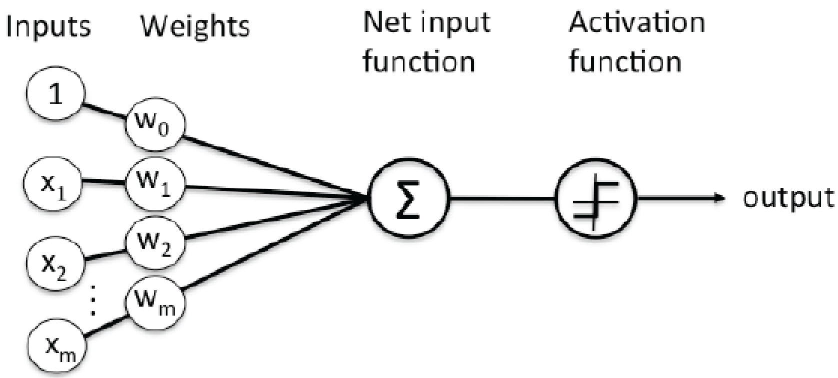
"A Logical Calculus of the Ideas Immanent in Nervous Activity"

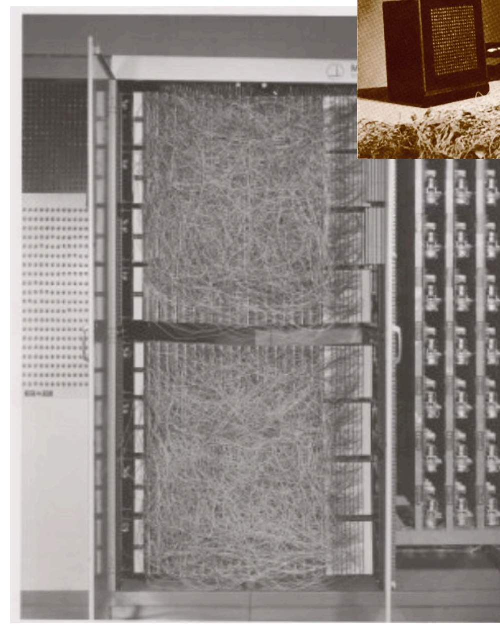




History of DL – Perceptron, 1958

1957 Frank Rosenblatt

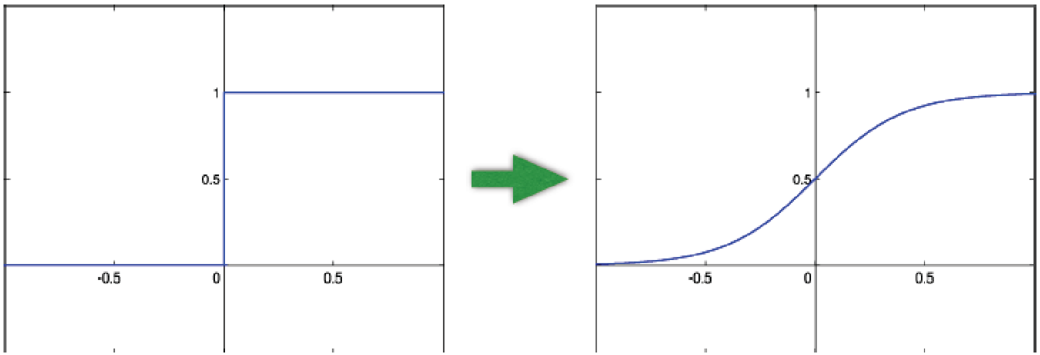




"[The Perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

THE NEW YORK TIMES

History of DL – Backpropagation, 1974



- Measure how **small changes** in weights **affect** output
- Can apply NN to **regression**

(1974) 1986

(Werbos) Rumelhart, Hinton, Williams

"Learning representations by back-propagating errors" (Nature)

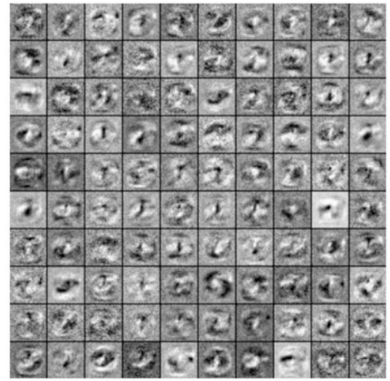
- **Multilayer** neural networks, etc.

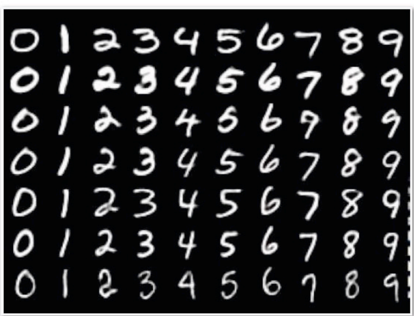
University of Pittsburgh

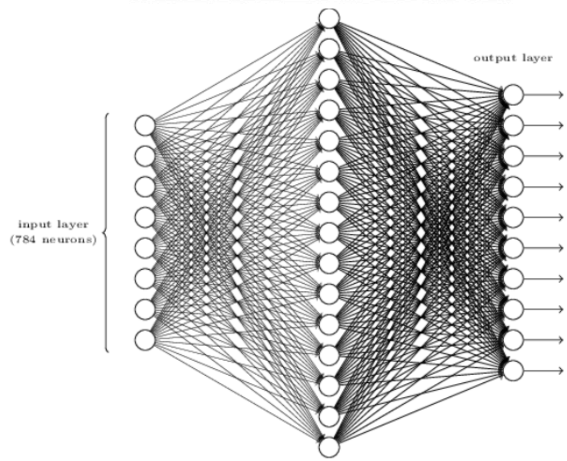
History of DL –
NN winter; late 1990 – early 2000

- Not enough data
- Not enough computing power
- Imperfect activation function

History of DL –
Restricted Boltzmann machine 2005



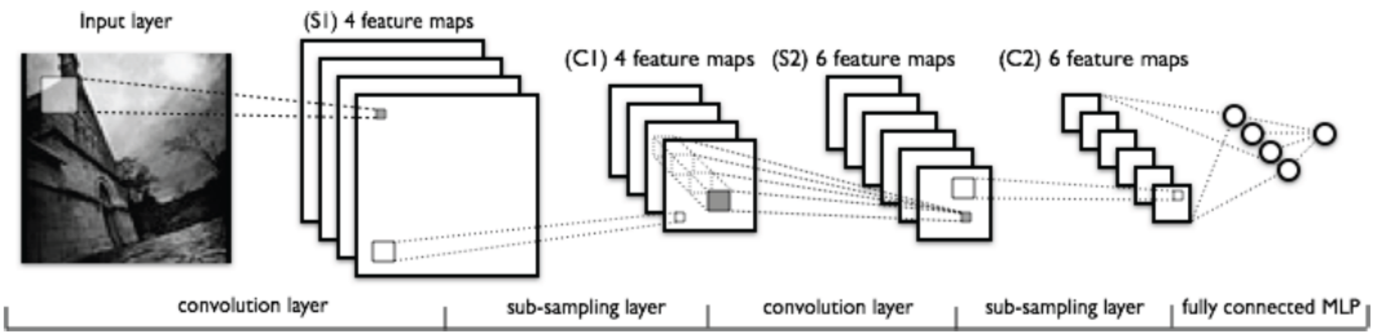




98.2% on the MNIST test set

History of DL – Convolution 1998, 2010

Yann LeCun, NYU & Meta



99.50% on the MNIST test set

CURRENT BEST: 99.77% by committee of 35 conv. nets

History of DL – ImageNet 2012 -

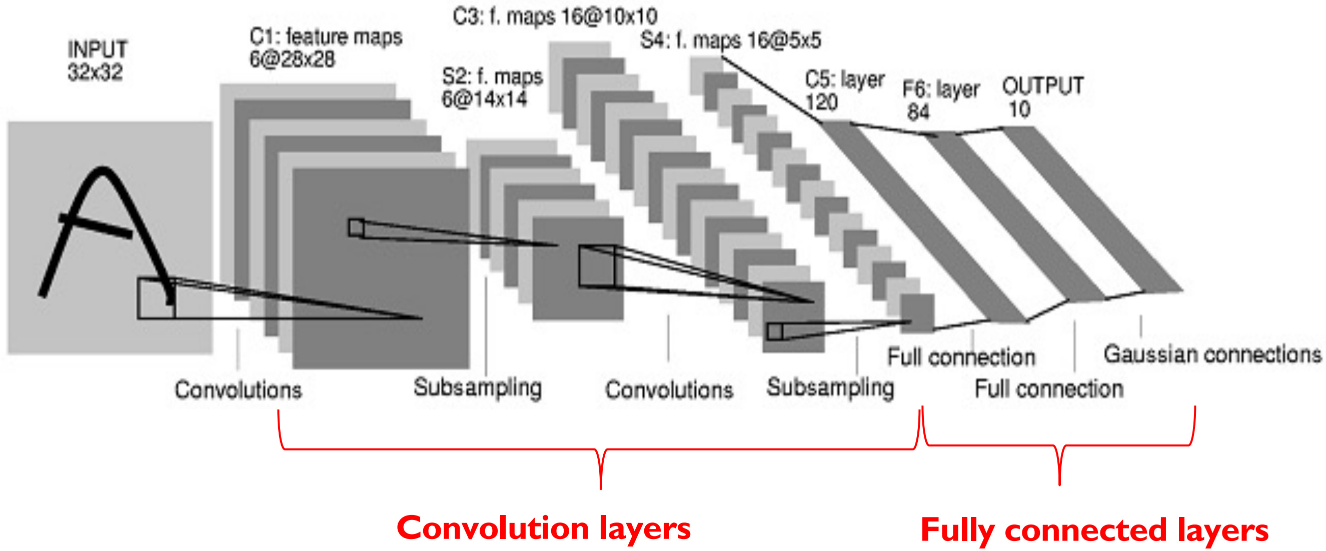


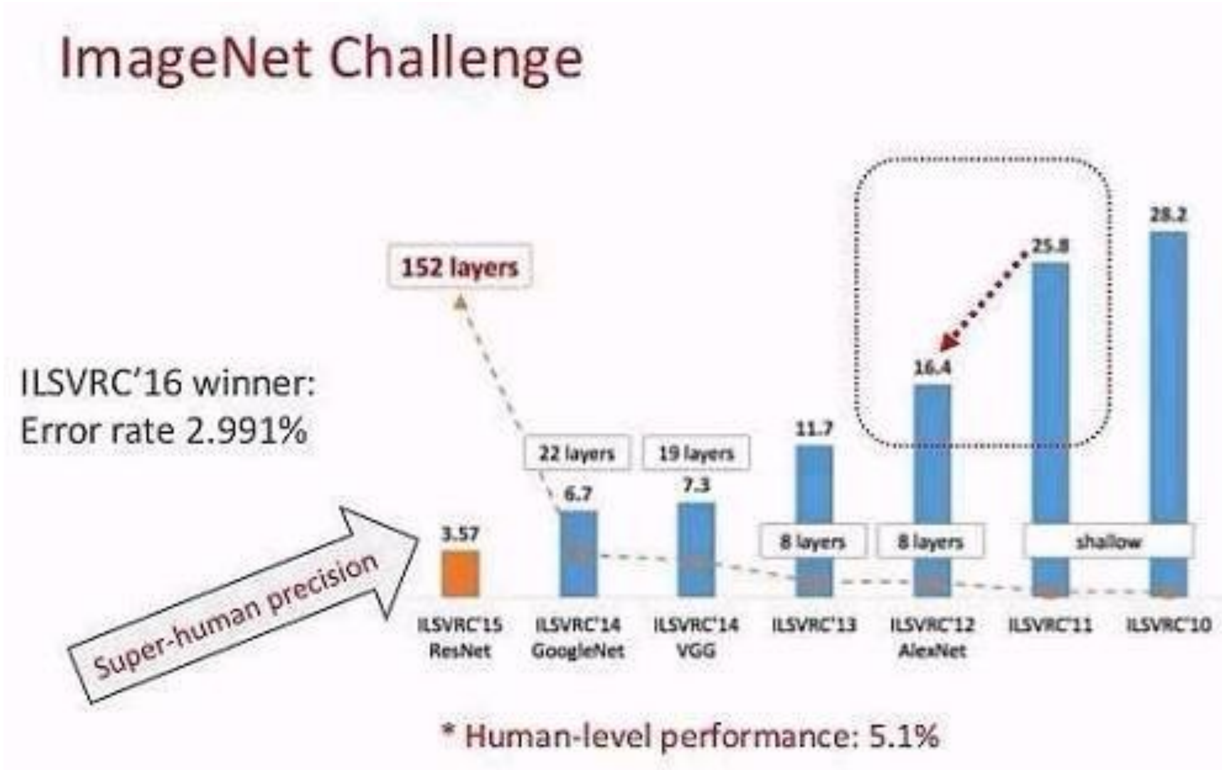
Image recognition

- **<u>AlexNet (2012)</u>**
  - 7 layers;
  - 1000 labels; >1.2M images
  - 17% vs 25.7%
- **GoogLeNet; VGG (2014)**
  - 19 layers (VGG)
  - ~7%

- **Deep Residual Net (2015)**
  - 152 layers
  - 3.57% > human recognition

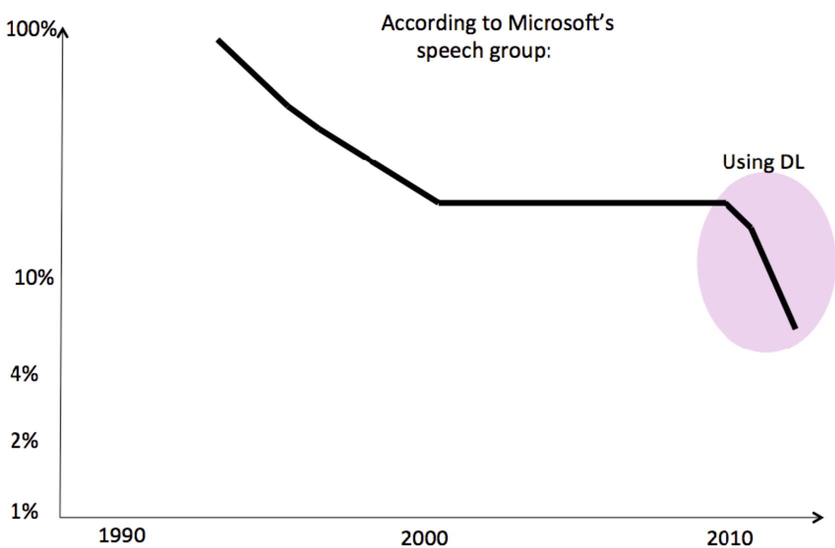
University of Pittsburgh

History of DL – ImageNet 2012 -



University of Pittsburgh

DL in speech recognition



Deep Learning in Speech Recognition

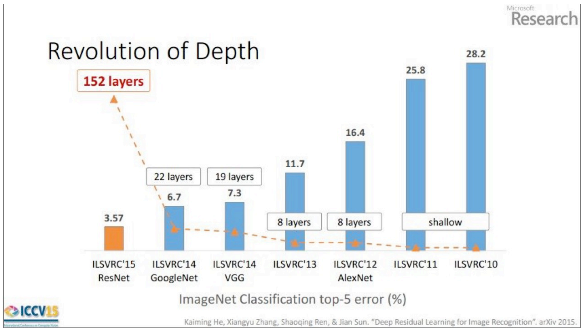
BRIEF

Microsoft achieves 'human parity' in speech recognition system

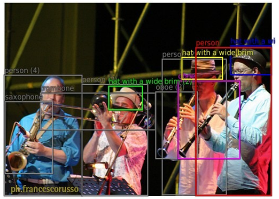
WER 5.1%

Deep learning applications

Image recognition



Object detection



Groundtruth:
person
hat with a wide brim
hat with a wide brim (2)
hat with a wide brim (3)
oboe
oboe (2)
saxophone
trombone
person (2)
person (3)
person (4)

Video game

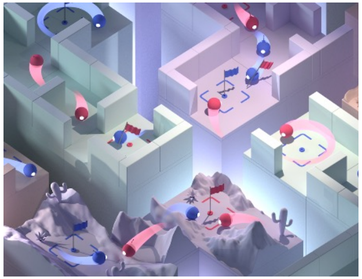
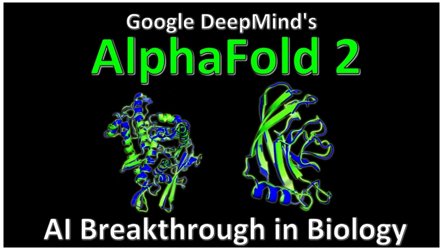


Image generation

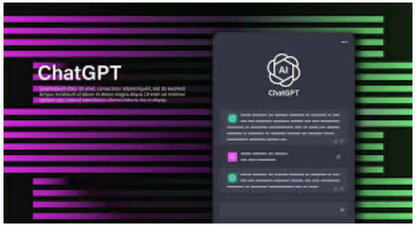
AlphaGO

AlphaFold



Medical diagnosis

ChatGPT



Categories of DL models

- Supervised
  - Deep (Dense) Neural Networks (DNN)
  - Convolutional Neural Networks (CNN)
  - Recurrent (RNN); LSTM, transformers

- Unsupervised
  - Auto-encoder (AE)
  - Variational AE (VAE)
  - Generative Adversarial Networks (GAN)
  - Diffusion models

Supervised learning (classification)

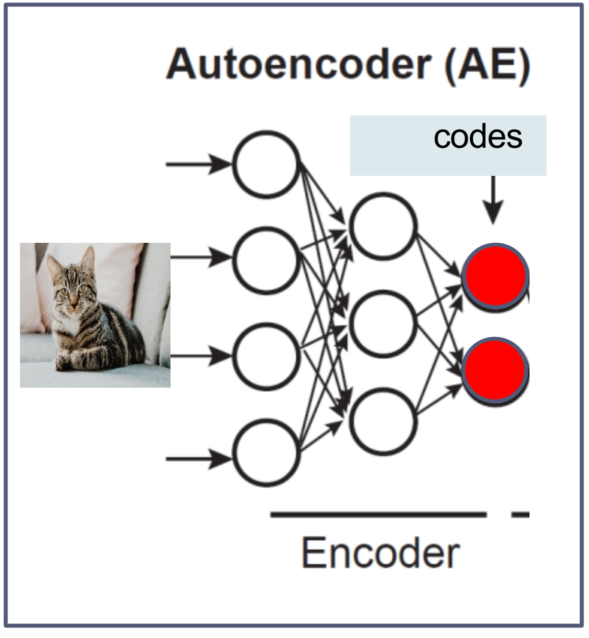
Goal

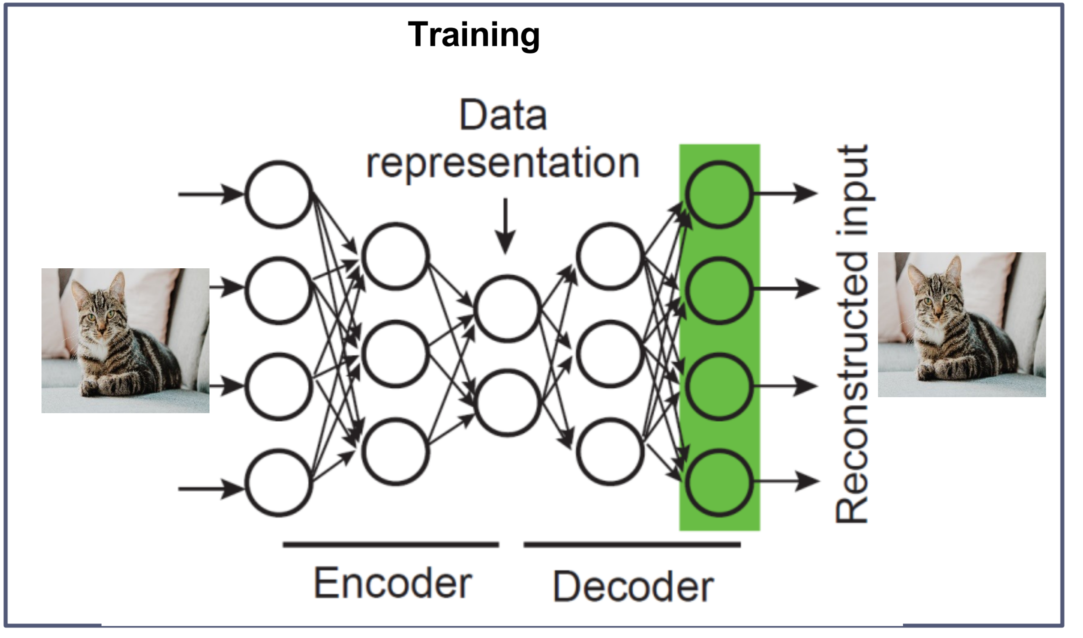
Labeled training data (lots of)

Training

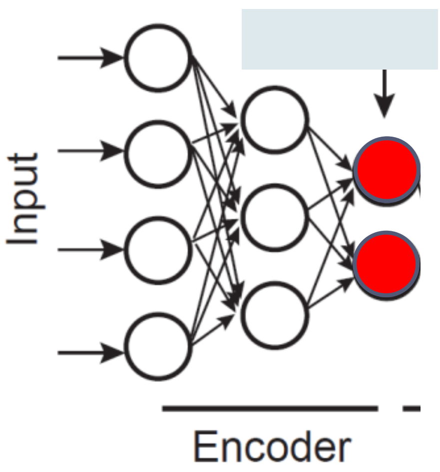
Unsupervised DL models (Autoencoder, generative adversarial networks(GAN))

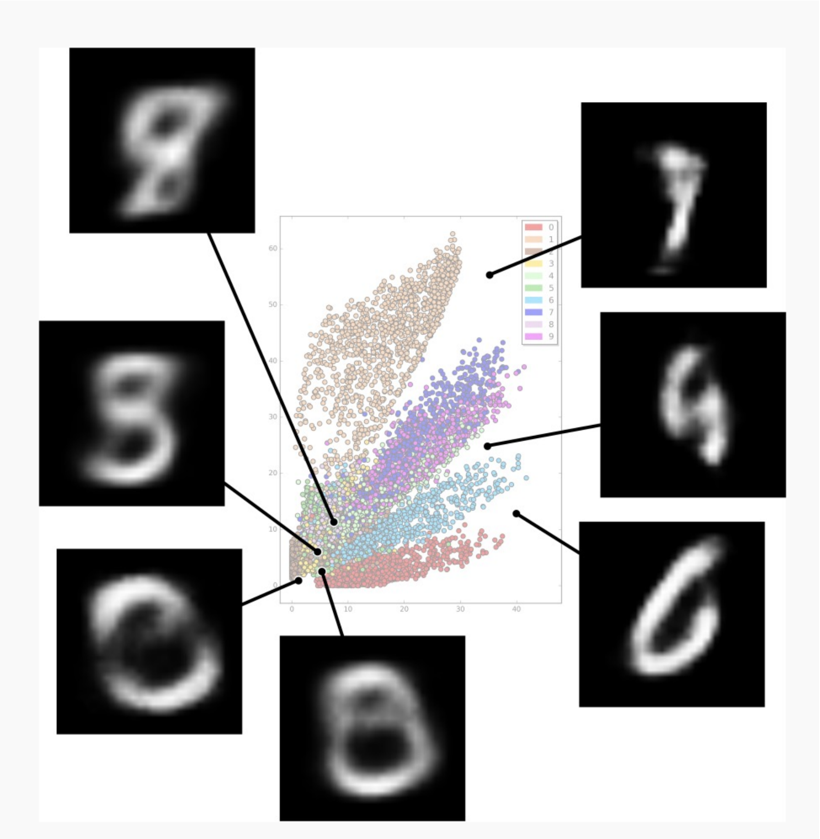
Goal: Learn the codes (latent representation) of data



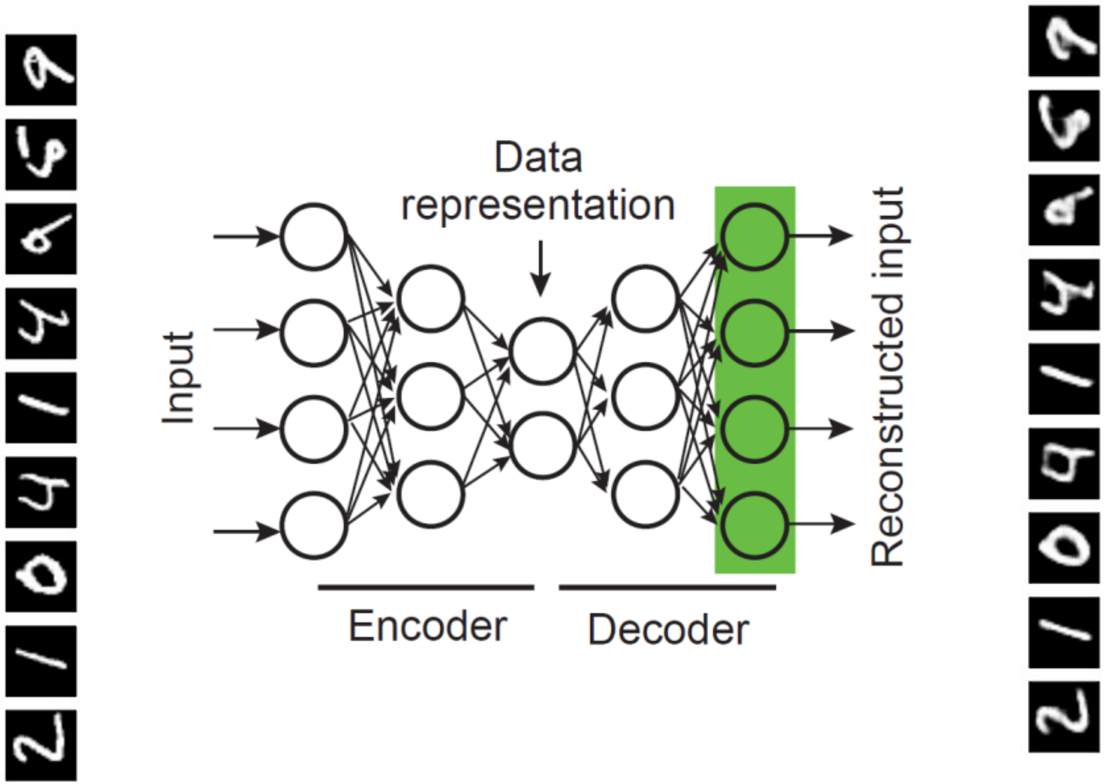


Visualizing the code





Reconstruct

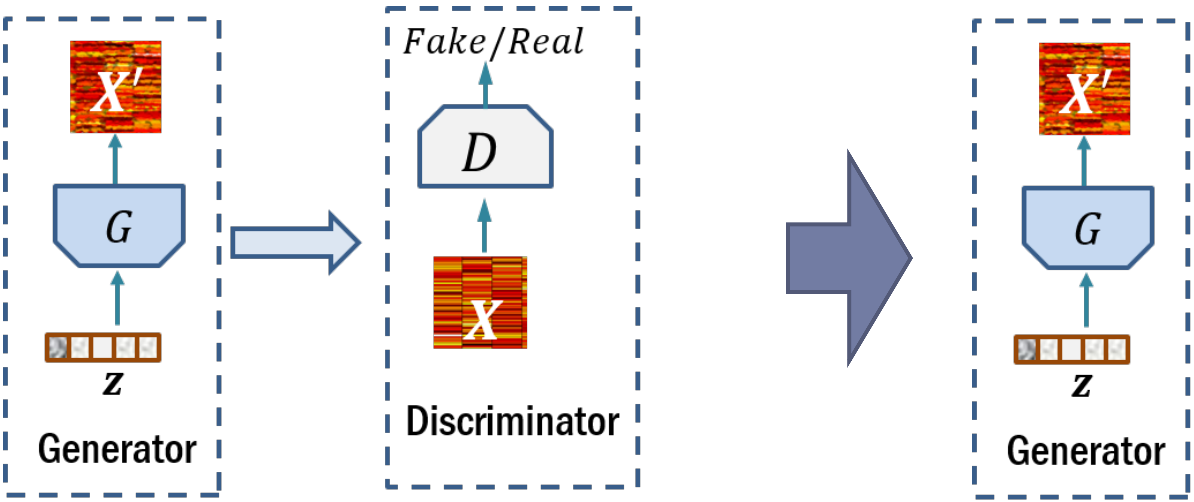


Generative Adversarial Networks (GAN)

Goal: Train a generator (decoder) G that learns from data to generate samples from data distribution

Training

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$



GAN Arts

Artificial intelligence takes on song-composing duties in Eurovision-inspired contest

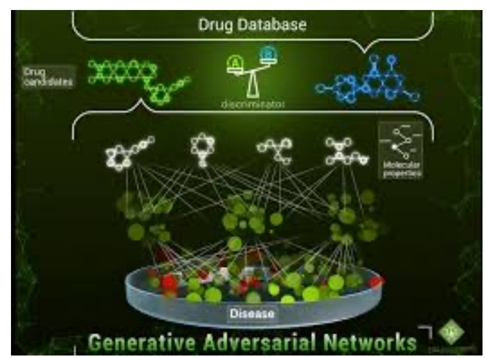
BY RODRIGO PÉREZ ORTEGA | APR. 24, 2020

Deepfake

You can now watch the video.

https://d285xazlytdv8t.cloudfront.net/output.mp4

Drug design



Processes of Building Deep Learning

- Observation/Data: $D = \{y, \boldsymbol{x}\}$
- ~~Feature extraction and selection~~
- Modeling
  - *Goal*: Model D
  - *Task: Define* f : $y = F(\boldsymbol{x}; \boldsymbol{w})$

DL

(hyper) parameters

- Training
  - *Goal*: Infer $\boldsymbol{w}$ (including hyper-parameters)
  - *Task 1*: *Loss function* - $L(D, \boldsymbol{w})$
  - *Task 2*: *optimization*
    - $\widehat{\boldsymbol{w}} = argmin_{\boldsymbol{w}} L(D, \boldsymbol{w})$

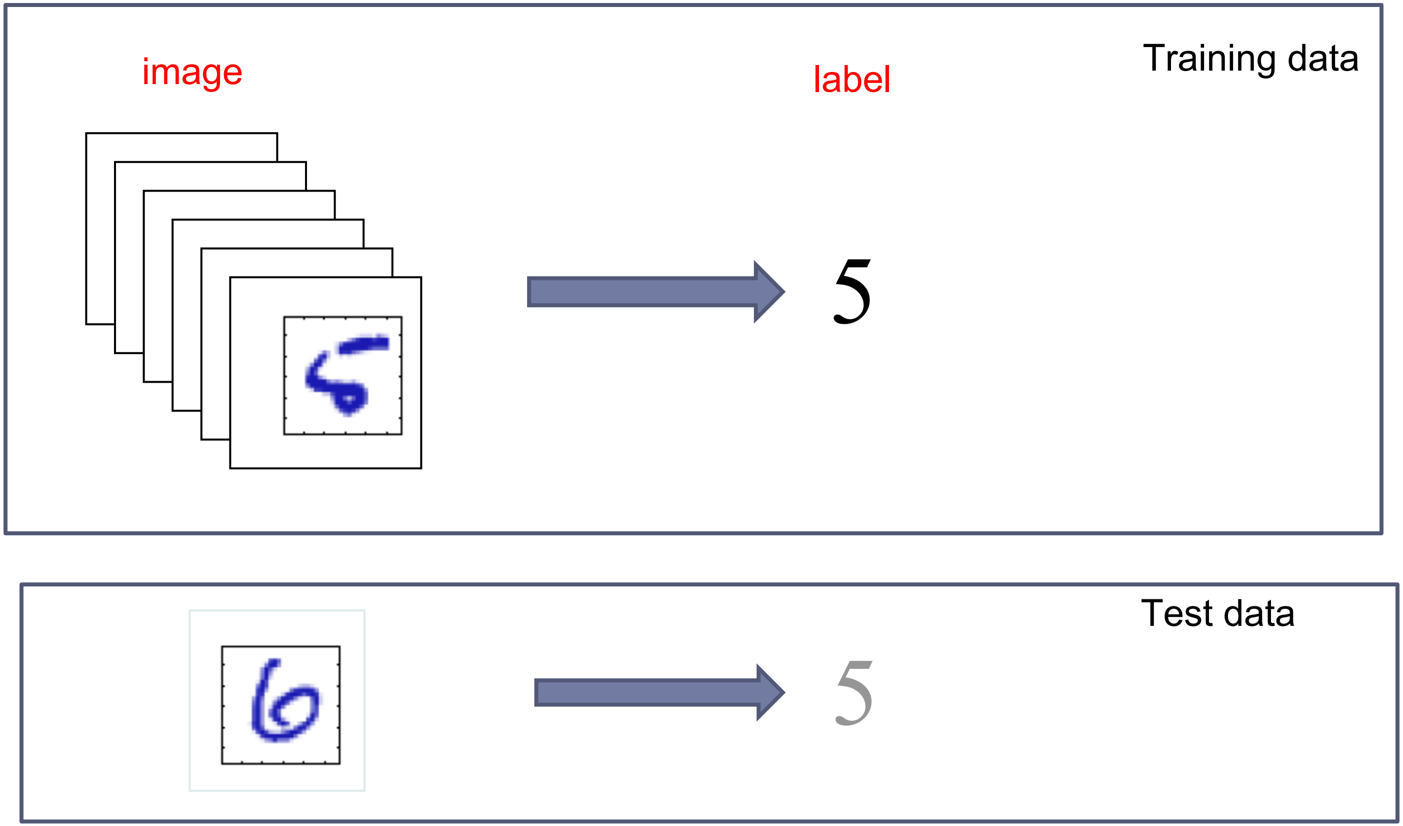
Cross entropy; MSE

Stochastic gradient descent

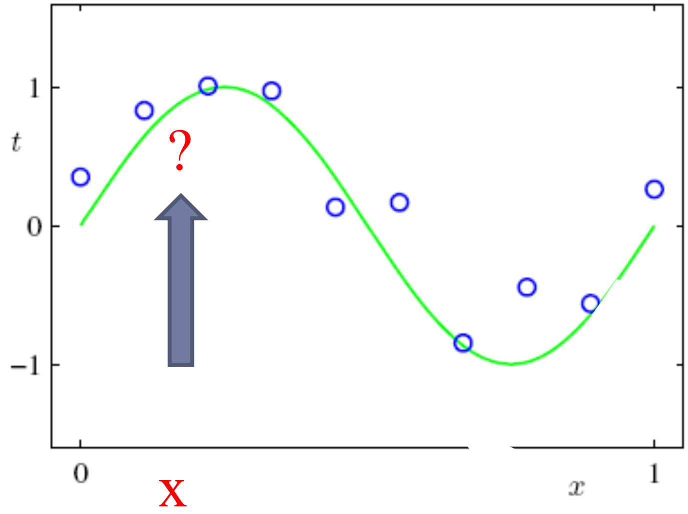
- Performance assessment
  - accuracy, AUC, PR, mAP, …

University of Pittsburgh

Classification



Curve fitting (regression)



Processes of Building Deep Learning

- Observation/Data: $D = \{y, \boldsymbol{x}\}$
- ~~Feature extraction and selection~~
- Modeling
  - *Goal*: Model D
  - *Task: Define f* : $y = F(\boldsymbol{x}; \boldsymbol{w})$

DL

(hyper) parameters

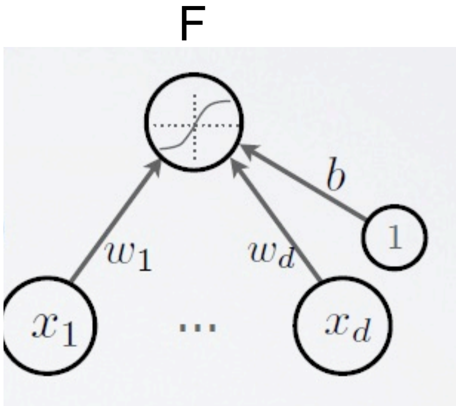
- Training
  - *Goal*: Infer $\boldsymbol{w}$ (including hyper-parameters)
  - *Task 1*: *Loss function* - $L(D, \boldsymbol{w})$
  - *Task 2*: *optimization*
    - $\widehat{\boldsymbol{w}} = argmin_{\boldsymbol{w}} L(D, \boldsymbol{w})$

Cross entropy; MSE

Stochastic gradient descent

- Performance assessment
  - accuracy, AUC, PR, mAP, …

Single neuron



- Pre-activation

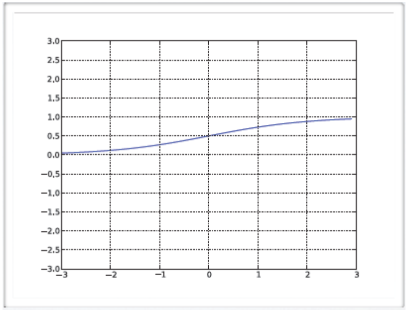
Input:	0,	8,	15,	22
Output:	32,	46.4,	59,	71.6

a = x*1.8 + 32

- Activation

Activation function (Sigmoid)

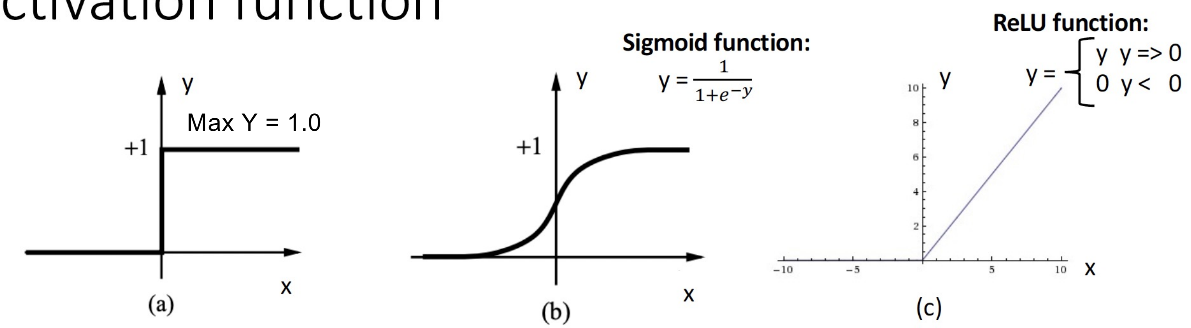
F = g(a) = g(x*1.8 + 32)



$$g(a) = \mathrm{sigm}(a) = \frac{1}{1+\exp(-a)}$$

Activation Functions

Activation function

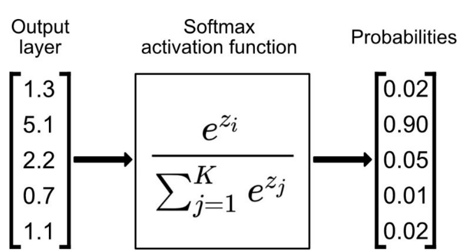


(a) Step function or threshold function

(b) Sigmoid function y = 1/(1+e-x); takes a real-valued input and squashes it to range between 0 and 1

(c) ReLU function

(d) Softmax



Classification Problems

Softmax converts the input vector to a probabilistic domain. This is very important for us for the final output layer.

University of Pittsburgh

Single hidden layer NN or fully connected layers

– Hidden layer pre-activation:

$$a_i = \boldsymbol{w}_i^{(1)T}\boldsymbol{x} + b_i^{(1)}$$

or

$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$$

– Hidden layer activation:

$$\boldsymbol{h}^{(1)} = \boldsymbol{g}(\boldsymbol{a}^{(1)})$$

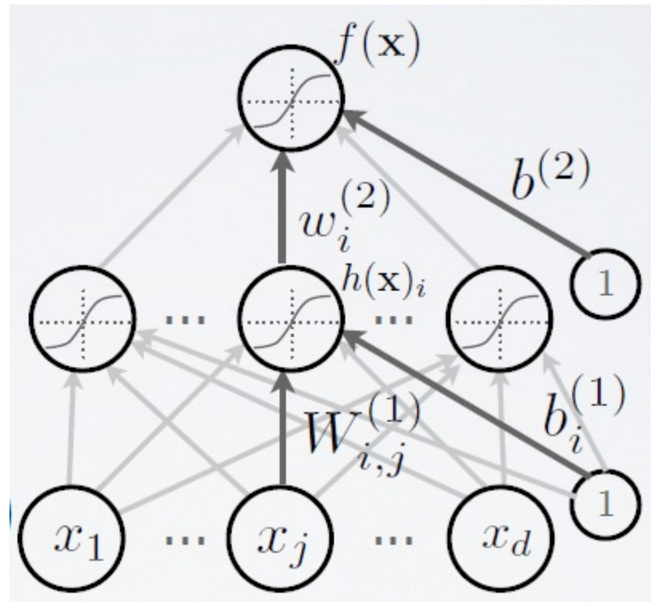
– Output layer pre-activation:

$$a^{(2)} = \boldsymbol{w}^{(2)^T}\boldsymbol{h}^{(1)} + b^{(2)}$$

– Output layer activation:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = o(a^{(2)})$$

O( ): output activation function



Deep Neural Networks (DNN)

- Could have L hidden layers:
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

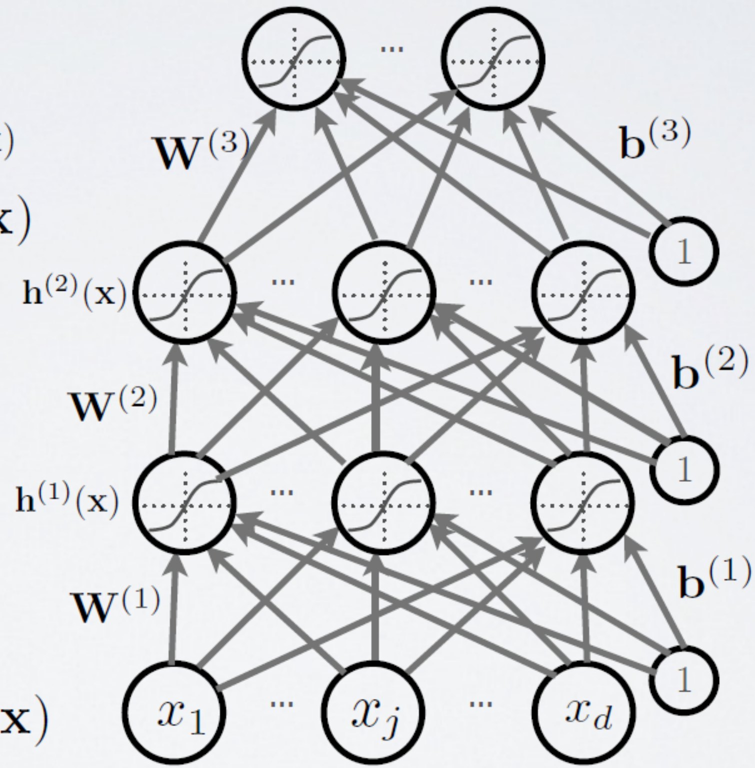
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation ($k=L+1$):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



Processes of Building Deep Learning

- Observation/Data: $D = \{y, \boldsymbol{x}\}$
- ~~Feature extraction and selection~~
- Modeling
  - *Goal*: Model *D*
  - *Task: Define f* : $y = F(\boldsymbol{x}; \boldsymbol{w})$

DL

(hyper) parameters

- Training
  - *Goal*: Estimate $\boldsymbol{w}$ (including hyper-parameters)
  - *Task 1*: *Loss function* - $L(D, \boldsymbol{w})$
  - *Task 2*: *optimization*
    - $\widehat{\boldsymbol{w}} = argmin_{\boldsymbol{w}}\, L(D, \boldsymbol{w})$

Cross entropy; MSE

Stochastic gradient descent

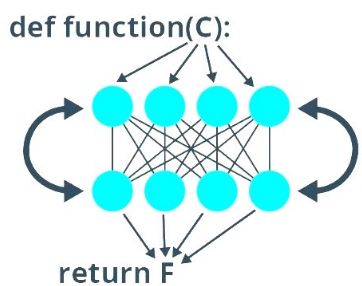
- Performance assessment
  - accuracy, AUC, PR, mAP, …

University of Pittsburgh

Goal of DL training

– Determine model weights (W, b), based on training data

W, b?



Training data

Input:	0,	8,	15,	22,	38
Output:	32,	46.4,	59,	71.6,	100.4

Ingredients of DL Training

– Training data (labeled data)

– Loss function $L(D, \boldsymbol{w})$

– Optimizers

$$\hat{\boldsymbol{w}} = \ argmin_{\boldsymbol{w}}\, L(D, \boldsymbol{w})$$

Loss functions

Hidden Layer	Output Layer (prediction)	Actual Output (label)
$h = f(x) = w_{it_1} * x + b_1$	12.1 ⟷	32
$h = f(x) = w_{it_2} * x + b_2$	19.9	32
…		
$h = f(x) = w_{it_n} * x + b_n$	31.4	32

- Assesses how good an estimate of (W, b) is
- Popular loss functions
  - Cross entropy loss (Classification)
  - Mean square error (regression; real-valued output)

Optimization

$$\hat{\boldsymbol{w}} = \ argmin_{\boldsymbol{w}}\, L(D, \boldsymbol{w})$$

– Solution (Optimizer): Stochastic gradient descent algorithm
– Efficient implementation: back-propagation

Optimizers

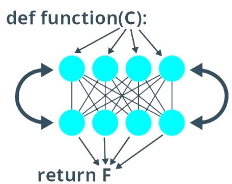
Name	Abbreviation	Mean momentum	Std. momentum	Strengths
Stochastic Gradient Descent	SGD	FALSE	FALSE	Easy to understand
Nesterov Momentum	SGD w/ Nesterov	TRUE	FALSE	
Root Mean Square Propagation	RMSProp	TRUE	FALSE	Works well w/ text input
Adaptive Moment Estimation	Adam	TRUE	TRUE	Good default

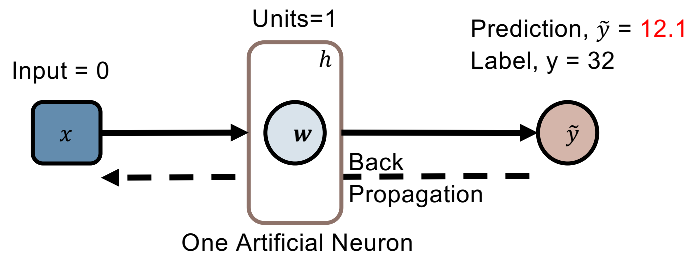
What does it look like?

```
def function(C):
    F = C * 1.8 + 32
    return F
```

Input: 0, 8, 15, 22

Output: 32, 46.4, 59, 71.6



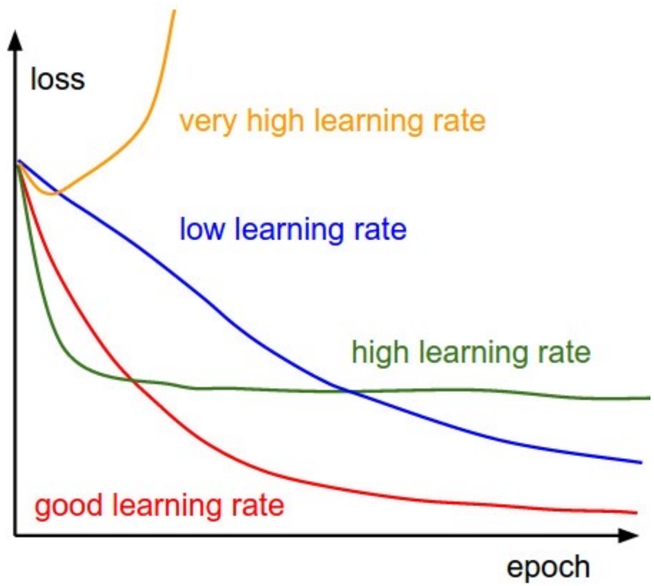


Iteration	**Input Layer (input)**	**Hidden Layer**	**Output Layer (prediction)**	Actual Output (label)	Compute Error (loss)	Loss function **MSE**
1	0	$h = f(x) = w_{it_1} * x + b_1$	12.1 ⟷	32	Loss(12, 32) -> very large	
2	0	$h = f(x) = w_{it_2} * x + b_2$	19.9	32	Loss(19.9, 32) -> moderately large	
		…				
n	0	$h = f(x) = w_{it_n} * x + b_n$	31.4	32	Loss(31.4, 32) -> very small	

University of Pittsburgh

Important concepts

- Terminology
  - Batch size: # of samples fed into an SGD step
  - Epoch: # of steps that takes to use all training samples
  - Learning rate
  - Initial value



Processes of Building Deep Learning

- Observation/Data: $D = \{y, \boldsymbol{x}\}$
- ~~Feature extraction and selection~~
- Modeling
  - *Goal*: Model D
  - *Task: Define f* : $y = F(\boldsymbol{x}; \boldsymbol{w})$

DL

(hyper) parameters

- Training
  - *Goal*: Estimate $\boldsymbol{w}$ (including hyper-parameters)
  - *Task 1*: *Loss function* - $L(D, \boldsymbol{w})$
  - *Task 2*: *optimization*
    - $\widehat{\boldsymbol{w}} = argmin_{\boldsymbol{w}} L(D, \boldsymbol{w})$

Cross entropy; MSE

Stochastic gradient descent

- Performance assessment
  - accuracy, AUC, PR, mAP, …

Evaluating classification performance

- Errors
  - False Positive: Incorrectly labeled as relevant
  - False Negative: Incorrectly labeled as not relevant

Prediction:

Image:

True Positive | **True** Negative | False Negative | False Positive

Evaluating classification performance

- Types of detection outcomes

Decision	$\mathcal{H}_0$ (no signal)	$\mathcal{H}_1$ (having signal)
$\mathcal{H}_0$	True negative	Miss, Type II, or F. negative
$\mathcal{H}_1$	FA, Type I, or F. positive	Detection, True postive

- Types of probabilities

Decision	$\mathcal{H}_0$ (no signal)	$\mathcal{H}_1$ (having signal)
$\mathcal{H}_0$	Specificity	P_{Miss}, β, False negative rate
$\mathcal{H}_1$	P_{FA}, α, False positive rate	P_D, $1-\beta$, power, sensitivity

Specificity and Sensitivity

- Specificity
  - True negative probability;
  - 1-False positive probability;
  - Percentage of negative examples that are correctly labeled
  - Specificity= (# true negatives) / (# negatives)
- Sensitivity (Recall)
  - True positive probability;
  - Percentage of positive examples that are correctly labeled
  - Recall = (# true positives) / (# positives)

Accuracy and precision

- Precision
  - Percentage of positive labels that are correct
  - Precision = (# true positives) / (# true positives + # false positives)
- Accuracy
  - Percentage of correct labels
  - Accuracy = (# true positives + # true negatives) / (# of samples)
  - Accuracy = 1 – P(error)

Example

Prediction:

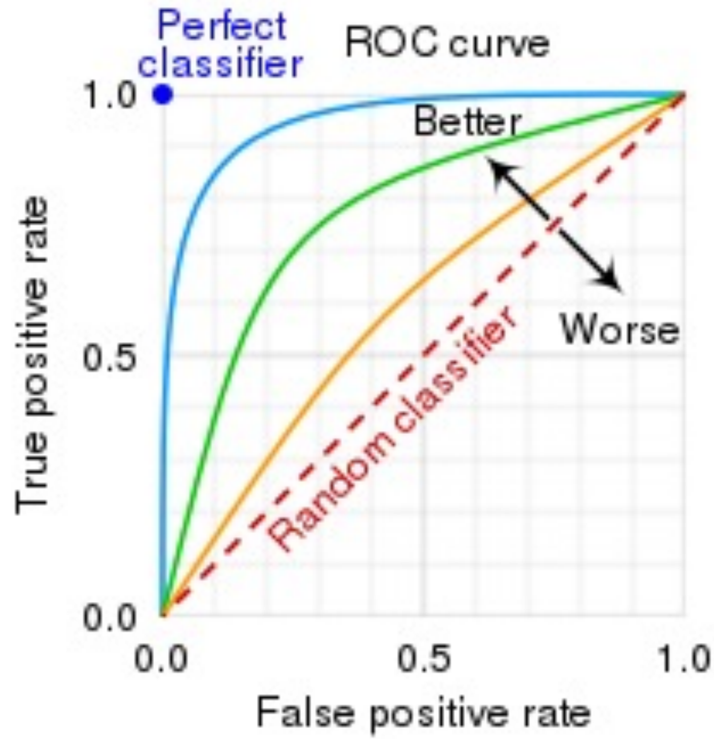
Image:

True Positive	**True** Negative	False Negative	False Positive	False Positive	**True** Positive

Specificity = 1/3 Recall = 2/3

Precision = 2/4 Accuracy = 3/6

ROC curve and Area Under the Curve (AUC)



Wikipedia.org

University of Pittsburgh

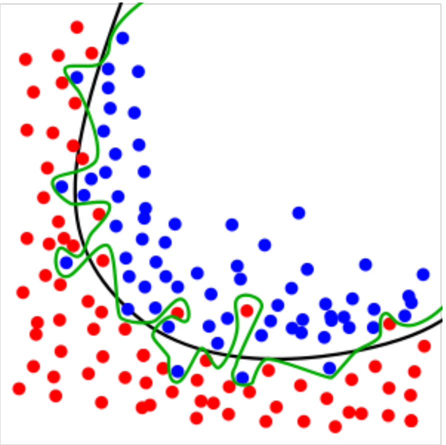
When to use which measure?

- No preferred labels and proportion of labels is unknown
  - ROC
- No preferred labels and proportion of labels is known
  - Accuracy
- Have a preferred label and proportion of the preferred label is small.
  - Precision vs. recall

Measuring Success/Failure for Classification

- Can we evaluate classification performance using training data?

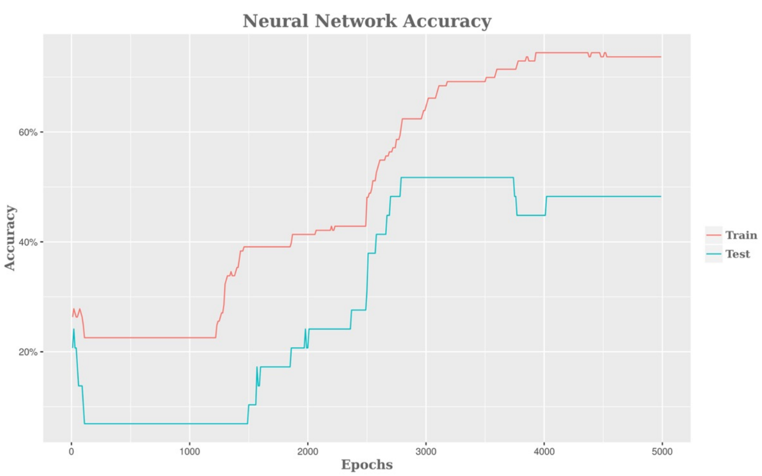
- No, because these could be a classifier that can produce 0 error on training data. This is called overfitting.
- Overfitting
  - Model performs well on training data but poorly on test data



Use test data to measure success

- Training Data
  - data used to learn a model
- Test Data
  - data used to assess the accuracy of model

- What to do when you only have training data?



University of Pittsburgh

Cross Validation

- To avoid overfitting

- train on part of available data, and test on rest
  - if dataset large (say, in 1000's), can simply set aside ≈ 1000 random examples as test
  - otherwise, use 10-fold cross validation
    - break dataset randomly into 10 parts
    - in turn, use each block as a test set, training on other 9 blocks