Introduction to Neural Networks and Deep Learning

ACC 690 - Predictive Analytics

Jesús Calderón

Learning Objectives

By the end of this week, students will be able to:

- Describe and explain Neural Networks.
- Discuss the advantages and limitations of these approaches.

Neural Networks

Perceptron

- Learns a hyperplane that separates the instances of different classes
- If the classes are linearly separable, the perceptron learning rule will find the separating hyperplane

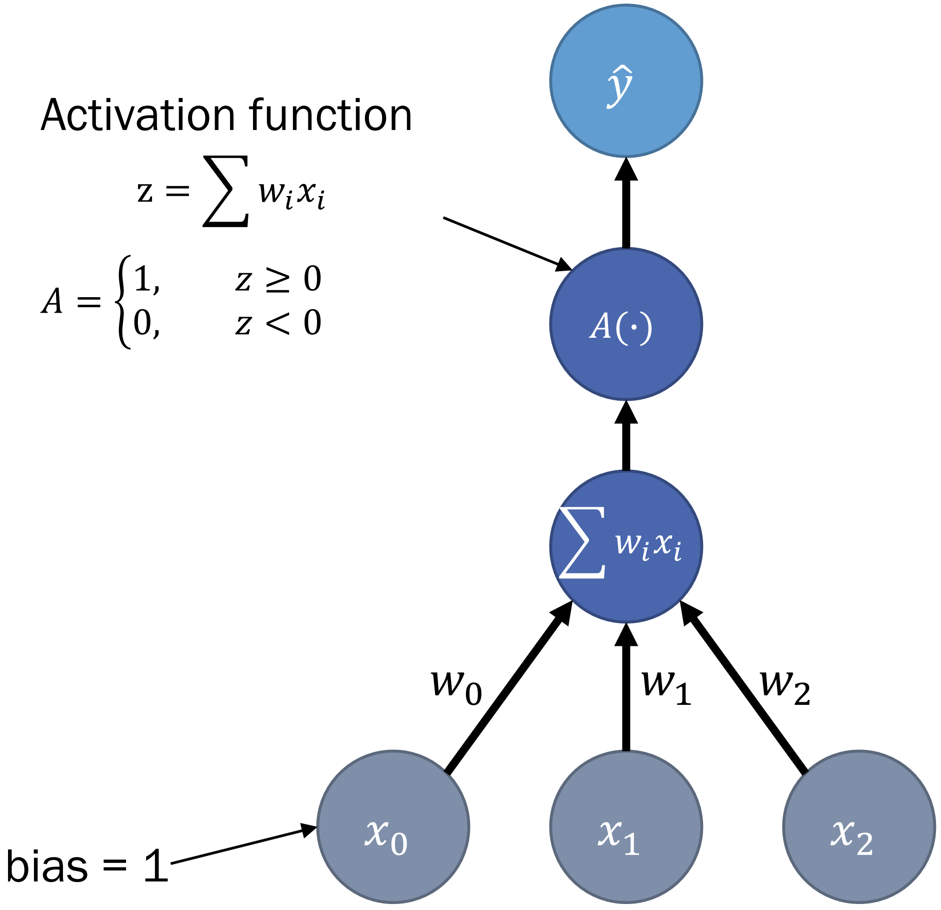
Perceptron learning rule

- Set all weights to zero
- Until all instances in the training data are classified correctly:
  - For each instance *k* in the data
    - If *k* is classified incorrectly by the perceptron
    - If *k* belongs to the first class add it to the weight vector, else subtract it from the weight vector

Activation function

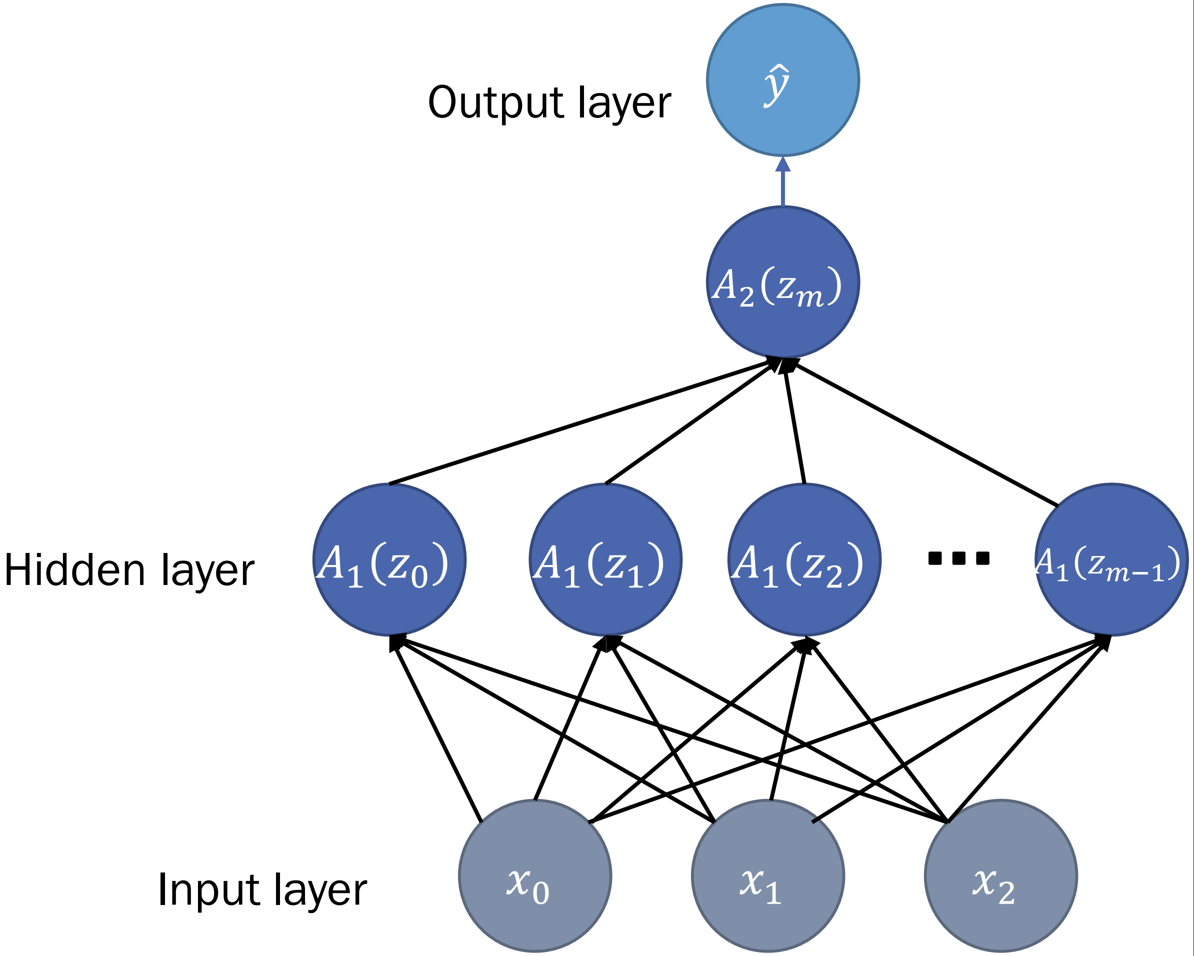
$$z = \sum w_i x_i$$

$$A = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

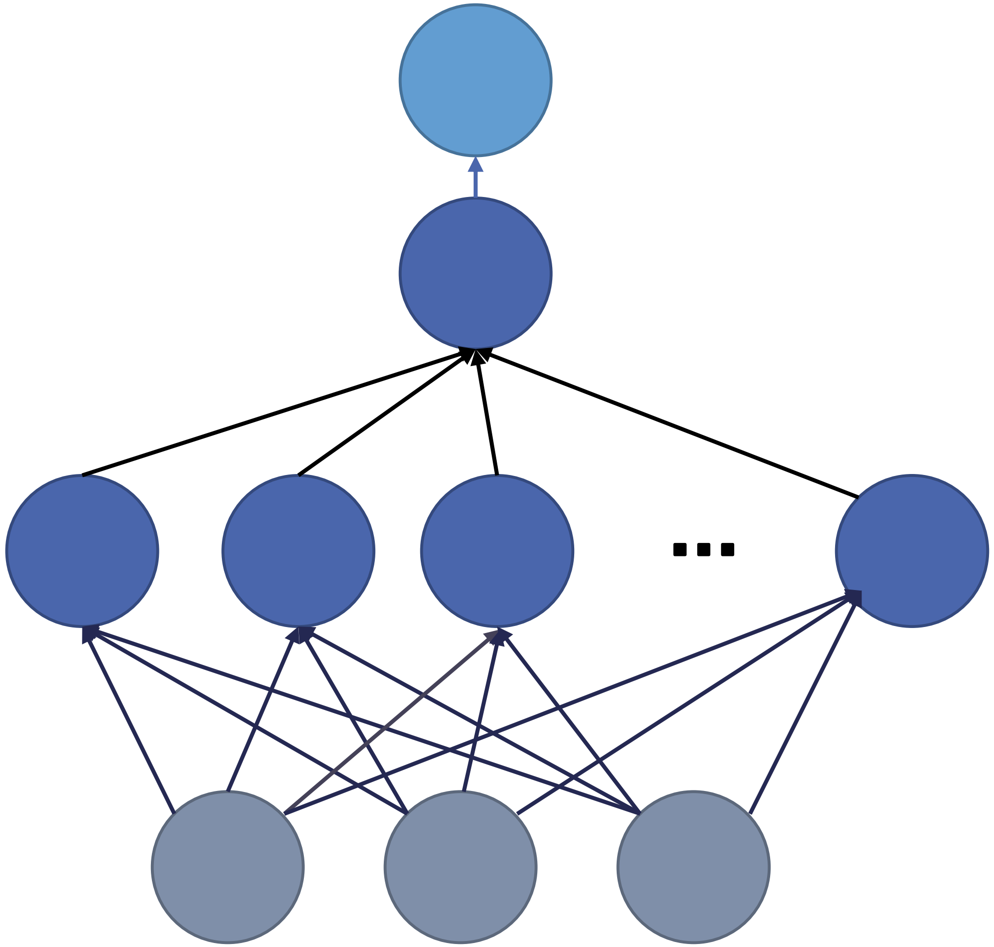


Neural Networks

- Perceptron is limited for all cases that are not linearly separable
- This limitation is overcome by neural networks, which:
  - Combine simple perceptron-like models in a hierarchical structure
  - Use (mostly) differentiable activation functions such as sigmoid or ReLU, such that gradient-based optimization can be applied
- The learning problem becomes:
  - Determine the network structure or architecture
  - Determine the weight of each connection



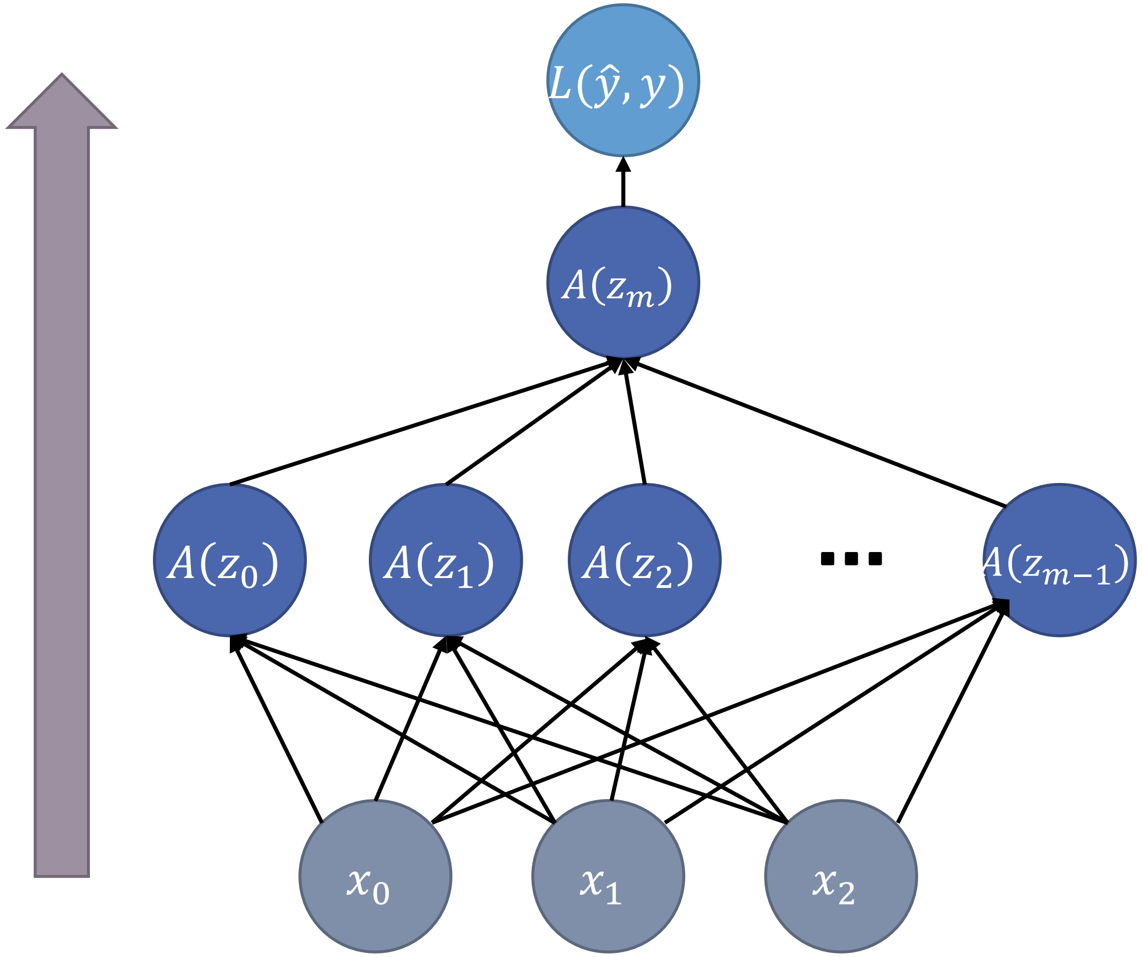
Backpropagation



- Backpropagation
  - Algorithm determines the weights for a given network structure
  - Computes the chain rule with a specific order of operations that is highly efficient
- Consists of two steps:
  - Forward computation: maps inputs to final output and defines the activation of the network
  - Back computation: calculate first order derivatives with respect to weights, avoiding double computations
- Update rule

$$w_i^{(t+1)} = w_i^{(t)} - lr \cdot \frac{\partial L(\hat{y}, y)}{\partial w_i}$$

Forward computation



- For illustration, assume squared-error loss and a sigmoid activation functions

$$L(\hat{y}, y) = \frac{1}{2}(y - A(z))^2$$

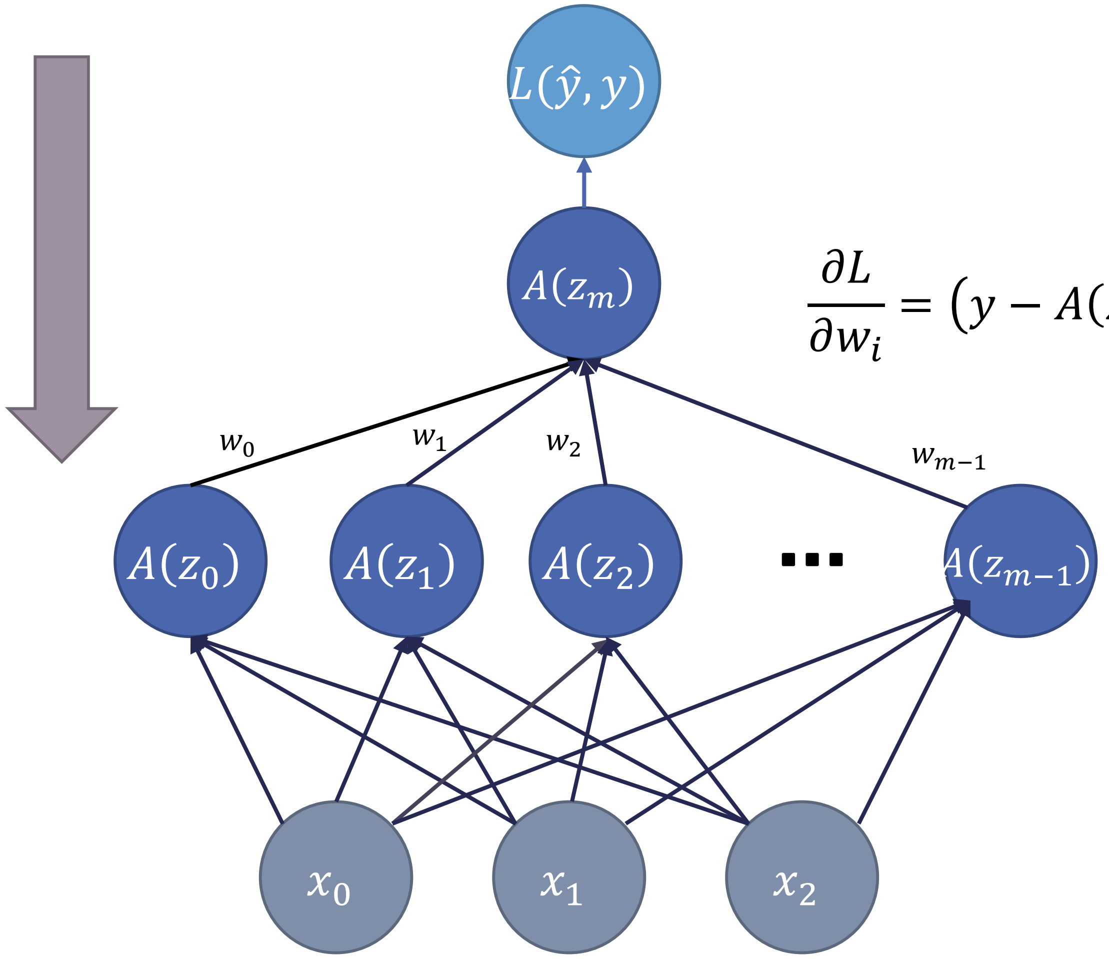
$$A(z) = \frac{1}{1 + e^{-z}}$$

$$z_i = w_{k,0} + \sum w_{k,j} x_{k,j}$$

Substitute z_j, accordingly

- Weights can be initialized with small random values, bias with zero or small positive values
- A regularization term could be added to loss function
- Forward computation calculates all values of network

Backward Computation



$$\frac{\partial L}{\partial w_i} = \left(y - A(z_m)\right)\frac{\partial A(z_m)}{\partial w_i}$$

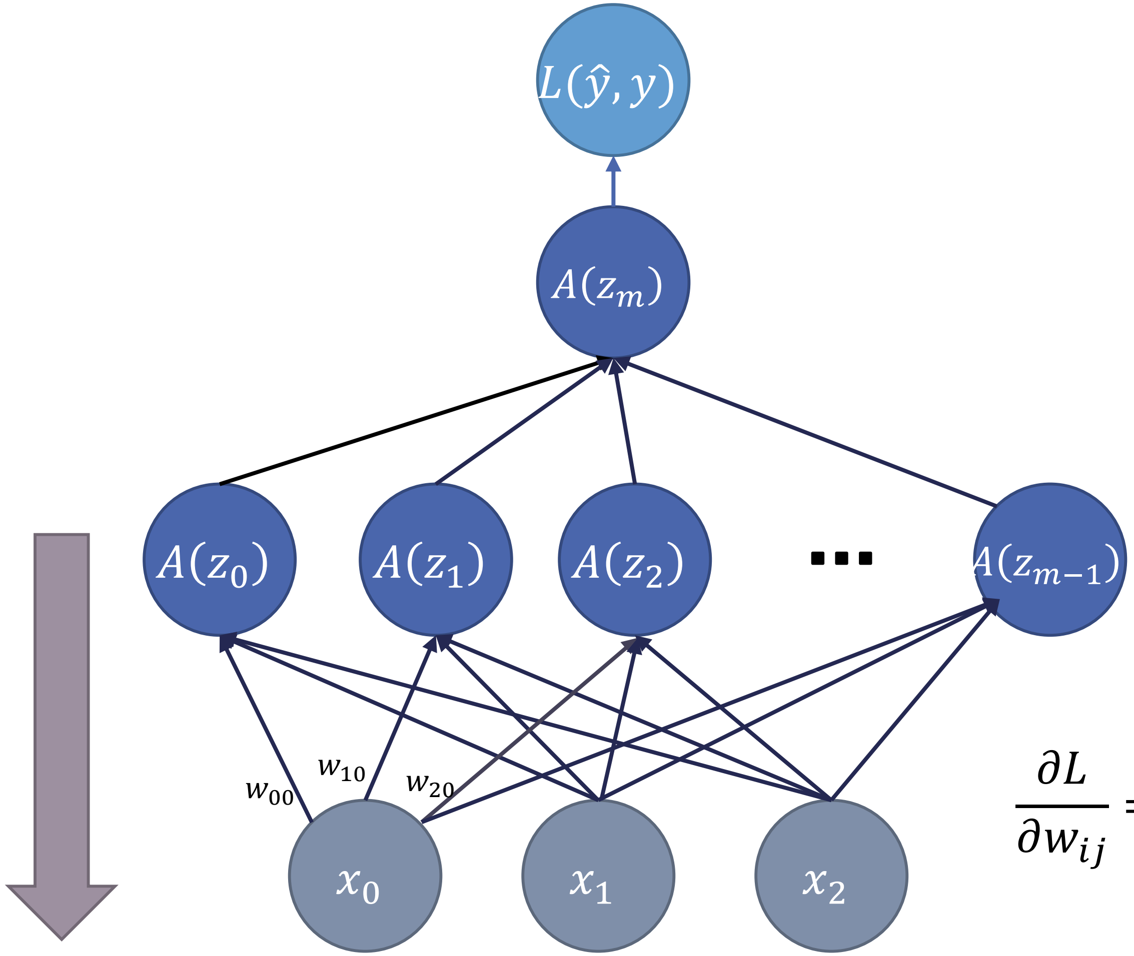
$$\frac{\partial A(z_m)}{\partial w_i} = \frac{\partial A(z_m)}{\partial z_m}\frac{\partial z_m}{\partial w_i}$$

$$\frac{\partial A(z_m)}{\partial z_m} = A(z_m)\left(1 - A(z_m)\right) = A'(z_m)$$

$$\frac{\partial z_m}{\partial w_i} = \mathrm{A}(z_i)$$

$$\frac{\partial L}{\partial w_i} = \left(y - A(z_m)\right)A'(z_m)\mathrm{A}(z_i)$$

Backward Computation



$$\frac{\partial L}{\partial w_{ij}} = \left(y - A(z_m)\right)\boxed{A'(z_m)}\boxed{w_i A'(z_i) x_i}$$

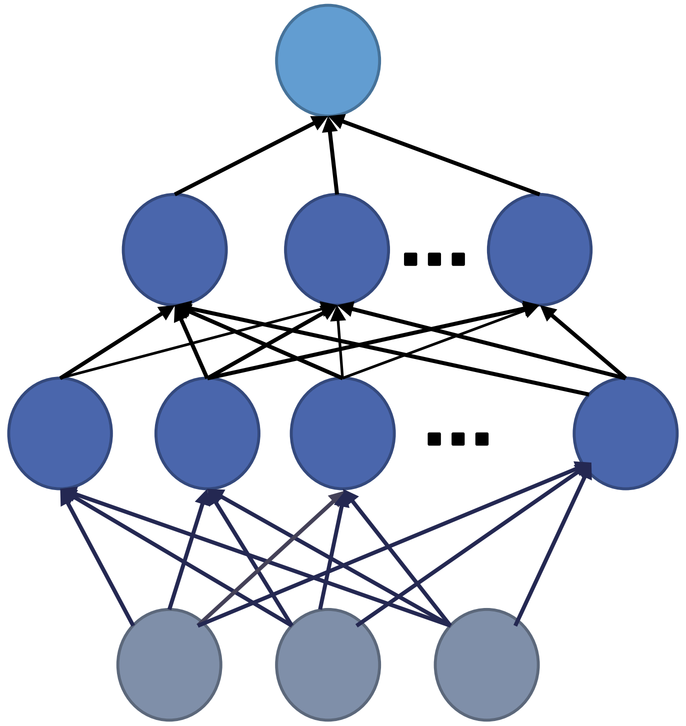
$$\boxed{\frac{\partial z_m}{\partial w_{ij}}} = \frac{\partial \sum w_i A(z_i)}{\partial w_{ij}} = w_i \frac{\partial A(z_i)}{\partial w_{ij}} = w_i A'(z_i) x_i$$

$$\boxed{\frac{\partial A(z_m)}{\partial z_m} = A(z_m)\left(1 - A(z_m)\right) = A'(z_m)}$$

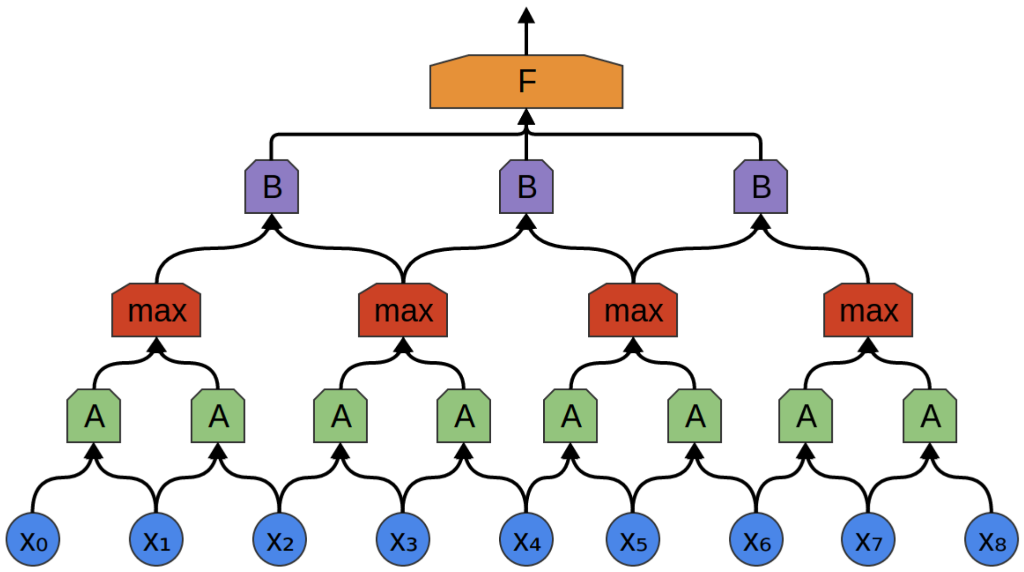
$$\frac{\partial L}{\partial w_{ij}} = \left(y - A(z_m)\right)\boxed{\frac{\partial A(z_m)}{\partial w_{ij}}} \rightarrow \boxed{\frac{\partial A(z_m)}{\partial w_{ij}}} = \boxed{\frac{\partial A(z_m)}{\partial z_m}}\boxed{\frac{\partial z_m}{\partial w_{ij}}}$$

Network architecture

- Architecture refers to the number of units in the network and the connections among them
- Most networks are organized as layers and most layers are arranged in a chain structure
  - Depth: the number of layers
  - Width: the number of units in each layer
- A feedforward network with a single layer is sufficient to represent any function
  - The layer may be infeasibly large
  - The model may fail to learn and generalize correctly
- Some family of functions can be approximated by a deep architecture



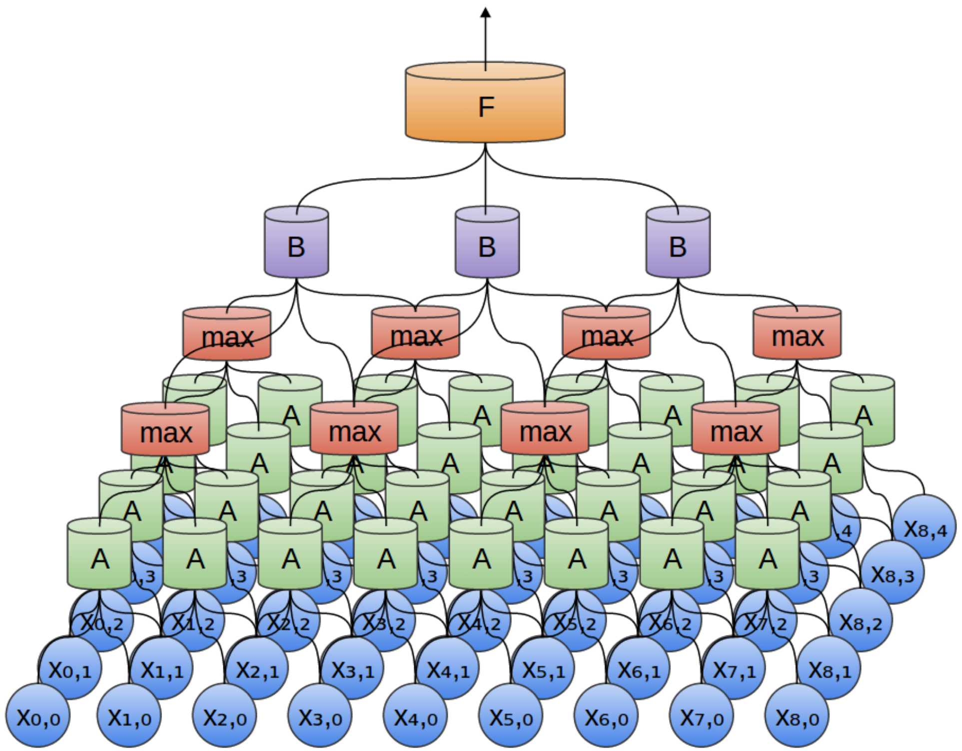
Convolutional Neural Networks

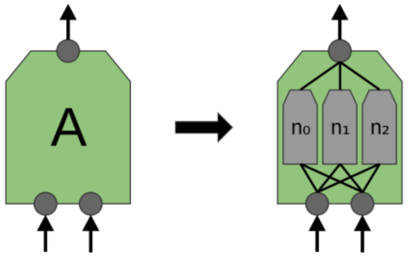


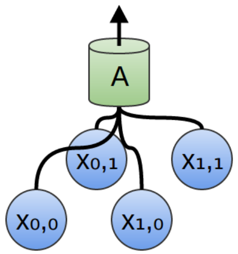
(Olah, 2014)

- ConvNets can be seen as a NN that uses many identical copies of the same neuron
- They allow computationally large models, while limiting the number of parameters
- At lower layers, ConvNets focus on local properties looking for them across different inputs
- Convolutional layers are composable

ConvNets in higher dimensions







(Olah, 2014)

- In 2012, Krizhevsky, Sutskever and Hinton outperformed image classification results using:
  - GPUs
  - ReLU units
  - DropOut for reducing over-fitting
  - Large image data set (ImageNet)
  - Deep convolutional NN

Deep Learning

Why Go Deep?

Universal Approximation Theorem

- A feedforward network:
  - Linear output
  - At least, one hidden layer with any ‘squashing’ activation function
- Can approximate any Borel measurable function
  - Any desired amount of error
  - Provided the network is given enough hidden units
- Extension of Theorem for ReLU
- Derivatives of the function can also be approximated by the network

(Hornik et al, 1989; Cybenko, 1989; Leshno et al. 1993)

- A feedforward network with a single layer is sufficient to represent any function:
  - The layer may be infeasibly large
  - May fail to learn and generalize correctly
- Using deeper models
  - Can reduce the number of units required to represent the desired function
  - Can reduce generalization error

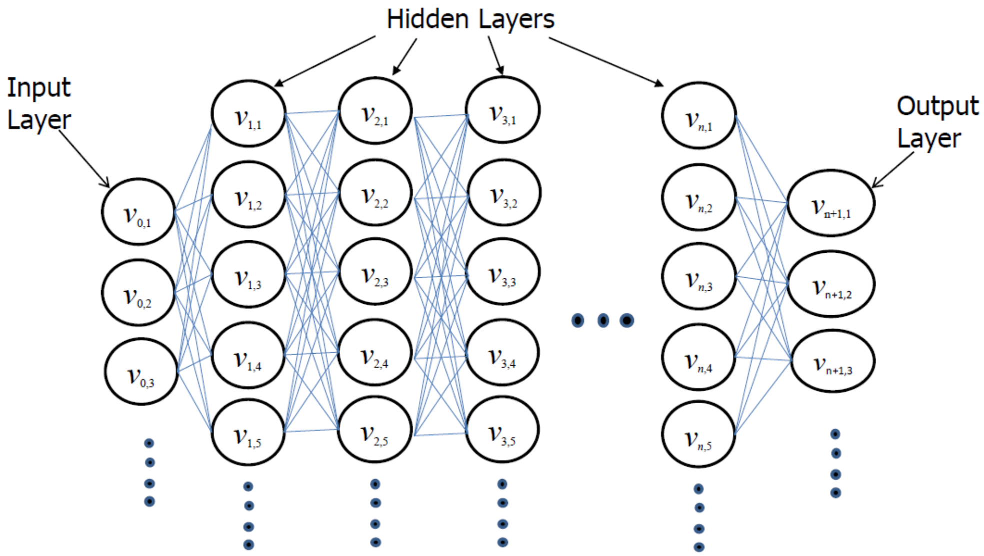
(Goodfellow et al., 2017)

A Note on Complexity

- In a fully connected network
  - *F:* number of features or inputs
  - *H*: number of hidden layers
  - *M*: neurons per layer
  - *T*: targets
- Number of parameters is given by

$$(F+1)M + M(M+1)(H-1) + (M+1)T$$

- Ex., a network with M = 200, F = 20, H = 6, and T = 1, requires more than 200 thousand parameters to be estimated



(Hull, 2019)


(Harris, 2016)

Deep Learning

- Conventional ML techniques have limited ability to process natural data:
  - Require feature engineering and domain knowledge
  - Transform the data, then learn classifier
- Representation learning
  - Input raw data
  - Automatically produce a representation needed for detection/classification
- Deep learning
  - Learn representations
  - Multiple levels of representation
  - Each level: simple, non-linear transformations
  - By stacking layers, very complex functions can be learned

(LeCun, Bengio, Hinton, 2015)

Deep Learning Reduces Engineering by Hand

- Deep Learning can be used to discover intricate structures in high-dimensional data:
  - Record-breaking performance in image recognition and speech recognition
  - Predict activity of potential drug molecules
  - Analyze particle accelerator data
  - Reconstruct brain circuits
  - Predict effects of mutations in non-coding DNA on gene expression and disease
  - Natural language understanding: topic classification, sentiment analysis, question answering, and language translation
- Requires little engineering by hand and takes advantage of data and computation availability

(LeCun, Bengio, Hinton, 2015)

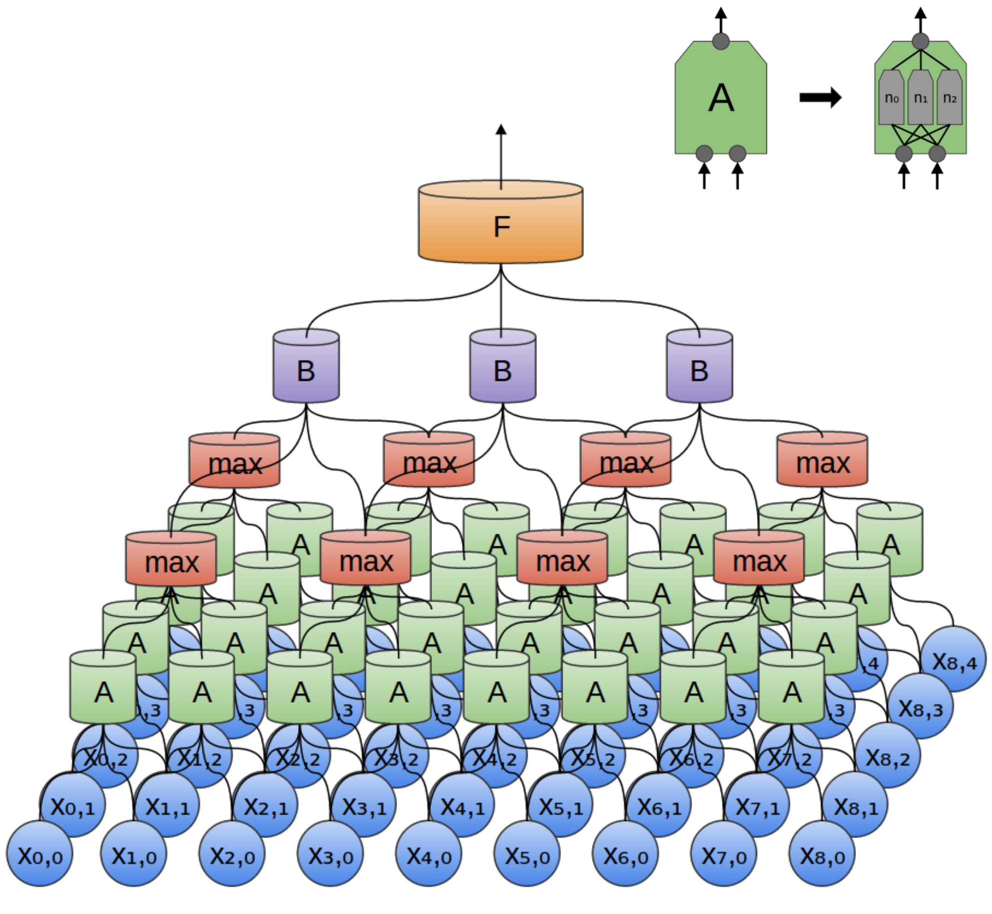
(Spratt, 2017)

Convolutional Neural Networks

Convolutional Neural Networks

(ConvNets)

- Two types of layers:
  - Convolutional layer
  - Pooling layer
- Four main ideas in ConvNets
  - **Local connections:** local groups of values are highly correlated, forming distinctive motifs
  - **Shared weights:** local statistics are invariant to location (a motif is a motif, regardless where it appears)
  - **Pooling:** semantically merge similar features into one
  - **Use of many layers:** natural signals are compositional hierarchies, higher-level features are obtained by composing lower-level ones

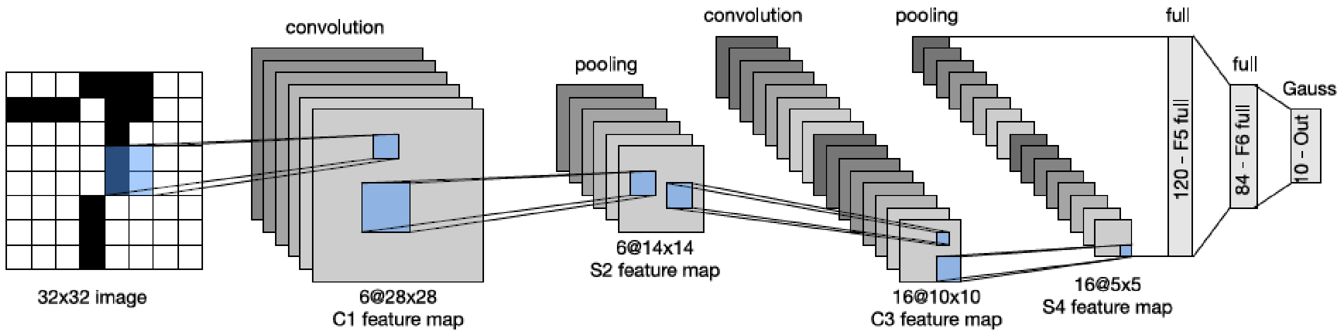


(Olah, 2014)

(LeCun, Bengio, Hinton, 2015)

Why Use a ConvNet?

LeNet5: the input is a handwritten digit,
the output is a probability over 10 possible outcomes.

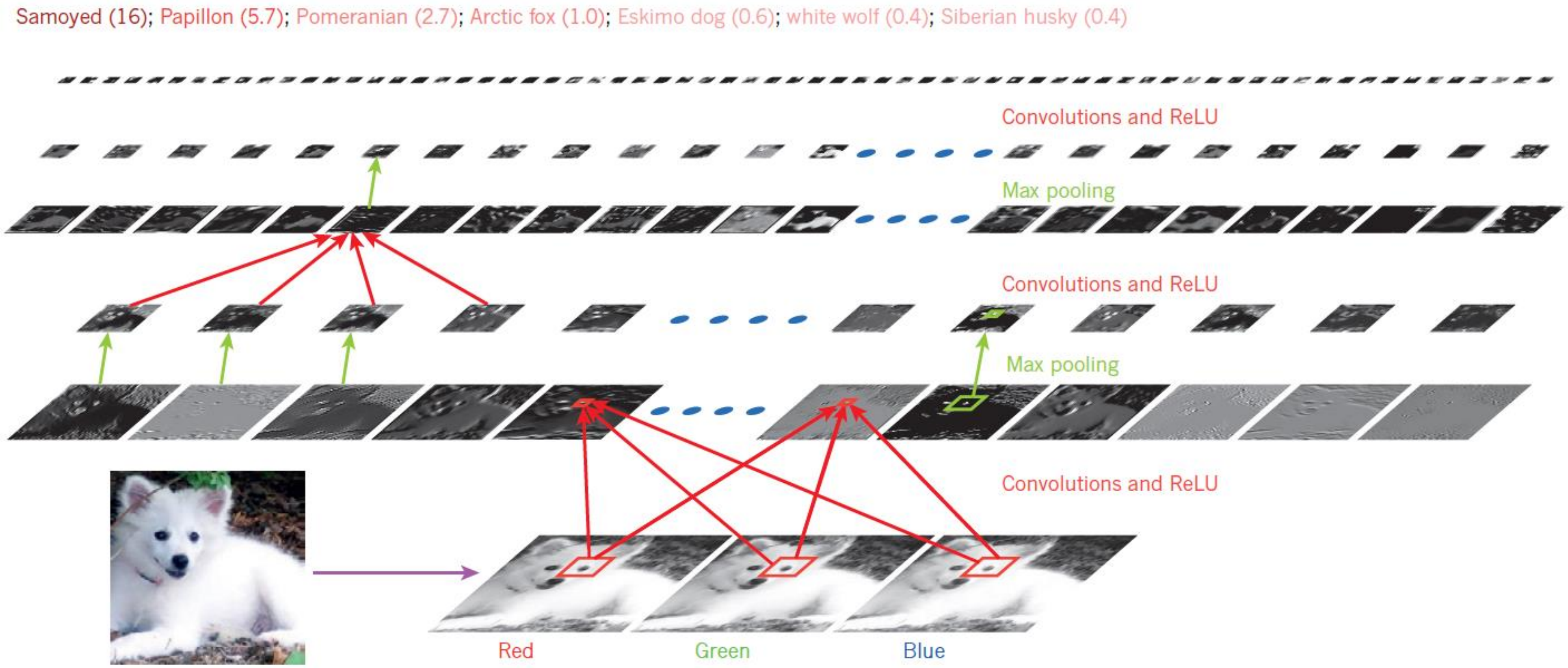


(Smola et al., 2019)

- In a fully connected NN, number of parameters grows dramatically with each layer
- Convolutional Neural Nets use convolution and pooling to reduce the number of parameters per layer

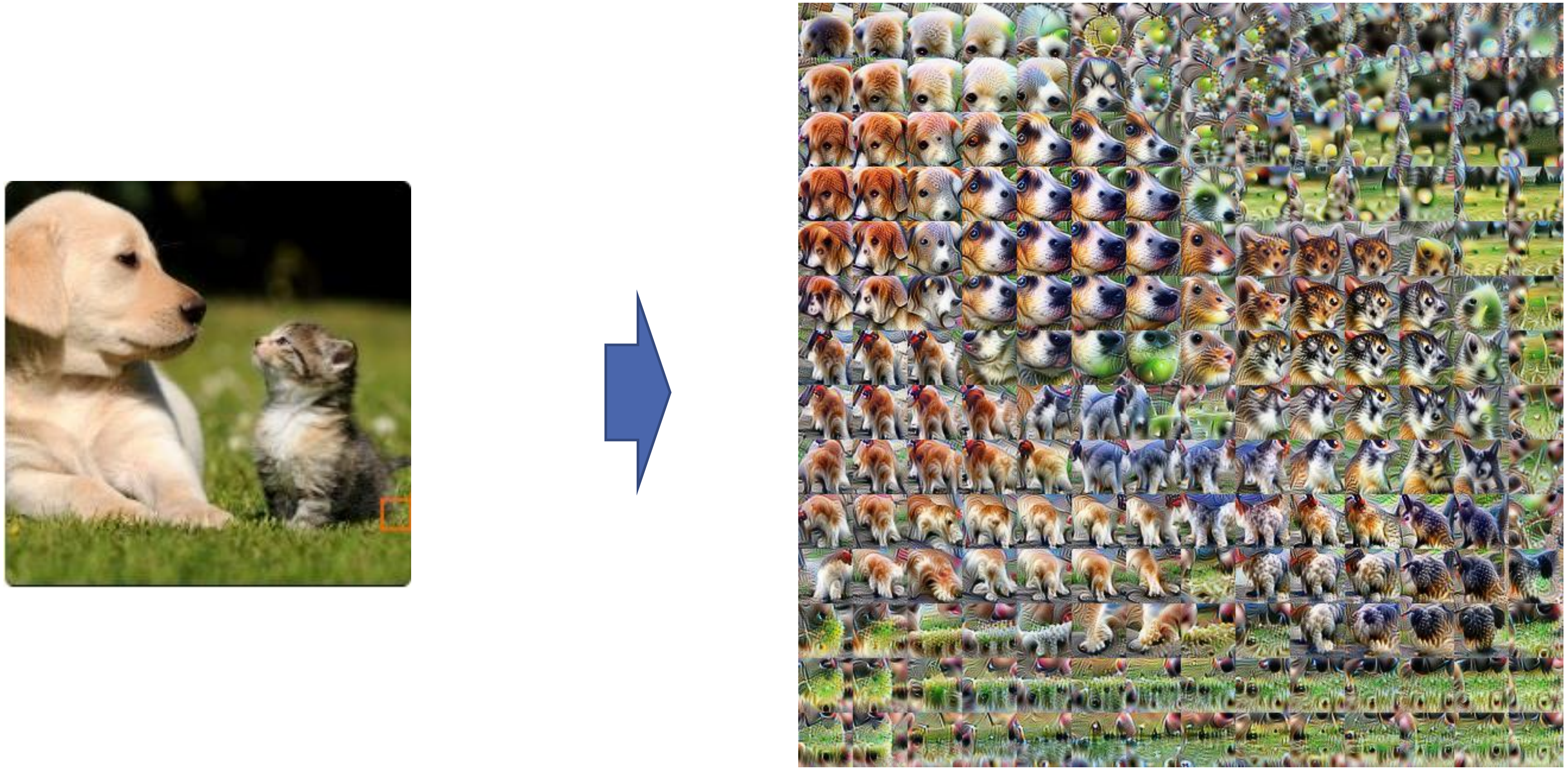
Example ConvNet

Applications of ConvNets have traditionally focused on image and image-like data



(LeCun et al., 2015)

Visualizing What a ConvNet ‘Sees’



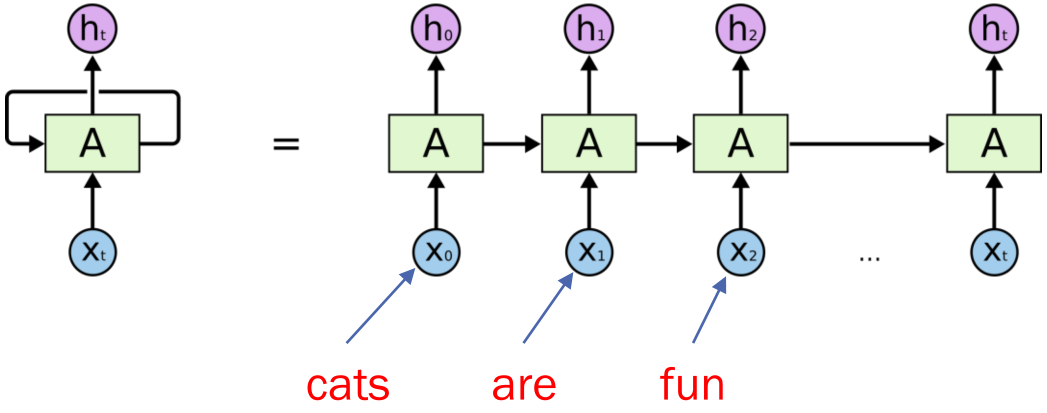
(Olah et al, 2018)

Recurrent Neural Networks

Recurrent Neural Nets

- Applied to sequential input such as text or speech
- Process an input sequence one element at a time
- Objective is to predict the next word in a sequence
  - To do this, a network needs to 'remember' previous words that it has seen
  - State vector
- Conceptually, a RNN can be seen as a very deep fully connected network with two inputs:
  - Previous state of the network
  - New feature (word)

(Osinga, 2018)

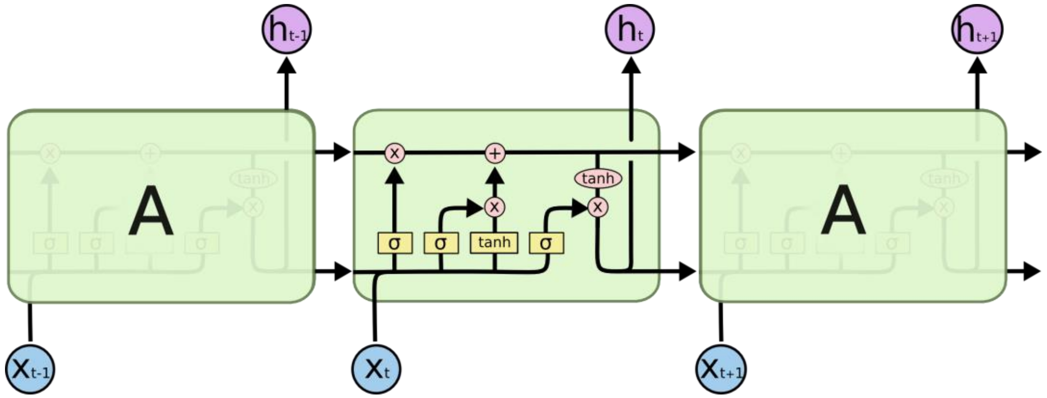


RNN unfolded in time

(Olah, 2014)

Recurrent Neural Networks
(RNN)

- Training RNNs can be challenging as gradients may explode or vanish
- Log Short-Term Memory (LSTM) models have special hidden units that serve as memory



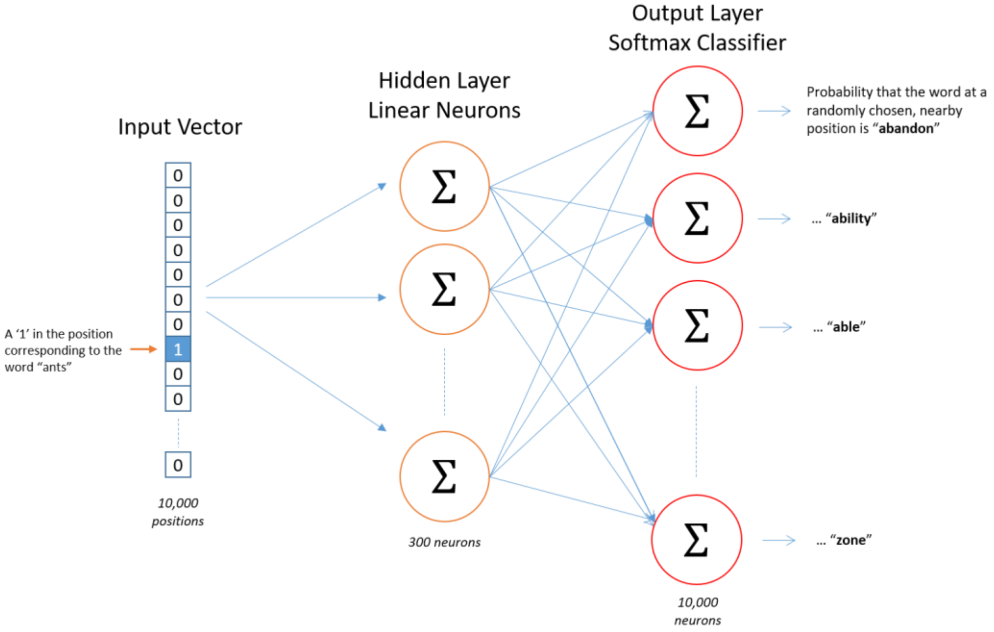
Long Short-Term Memory activation

(Olah, 2014)

Natural Language Processing

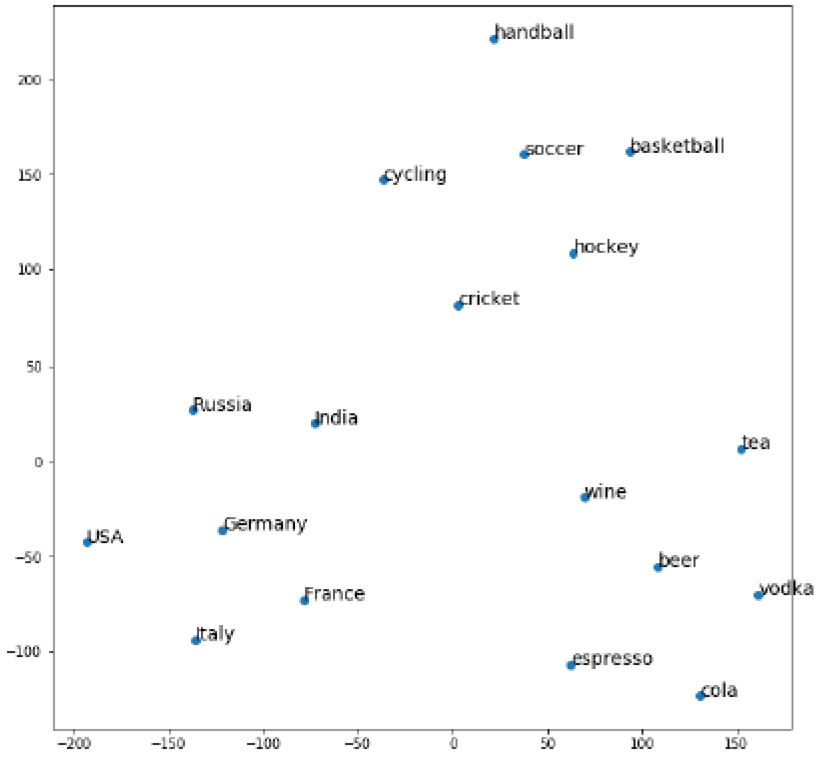
Word Embeddings and Representation

- Training NN is computationally intensive and time-consuming
- Assume that an NN has been trained
  - Pre-trained NN are available
  - Training sets include general language (news articles, Wikipedia, etc.), as well as specialized language (legal, medical, etc.)
- Word embeddings map words in vocabulary with an n-dimensional vector
- A popular embedding is Word2Vec
  - Similar to an autoencoder, obtained through a NN as a by-product
  - NN predicts a word from its context



(Gilyadov, 2018)

t-SNE: Dimensionality Reduction



(Osinga, 2018)

- From an input similar to a bag of words, one can obtain a lower-dimensional representation
- Word embeddings associate each word of a vocabulary (ex., 10k words) to an n-dimensional vector (ex., 300)
- But it is still not low enough to be visualized
- A popular technique to visualize embeddings is t-SNE
  - t-distributed stochastic neighbor embedding

Thank you!