# The Deep Learning Revolution

Raymond J. Mooney

University of Texas at Austin

# Deep Learning Revolution

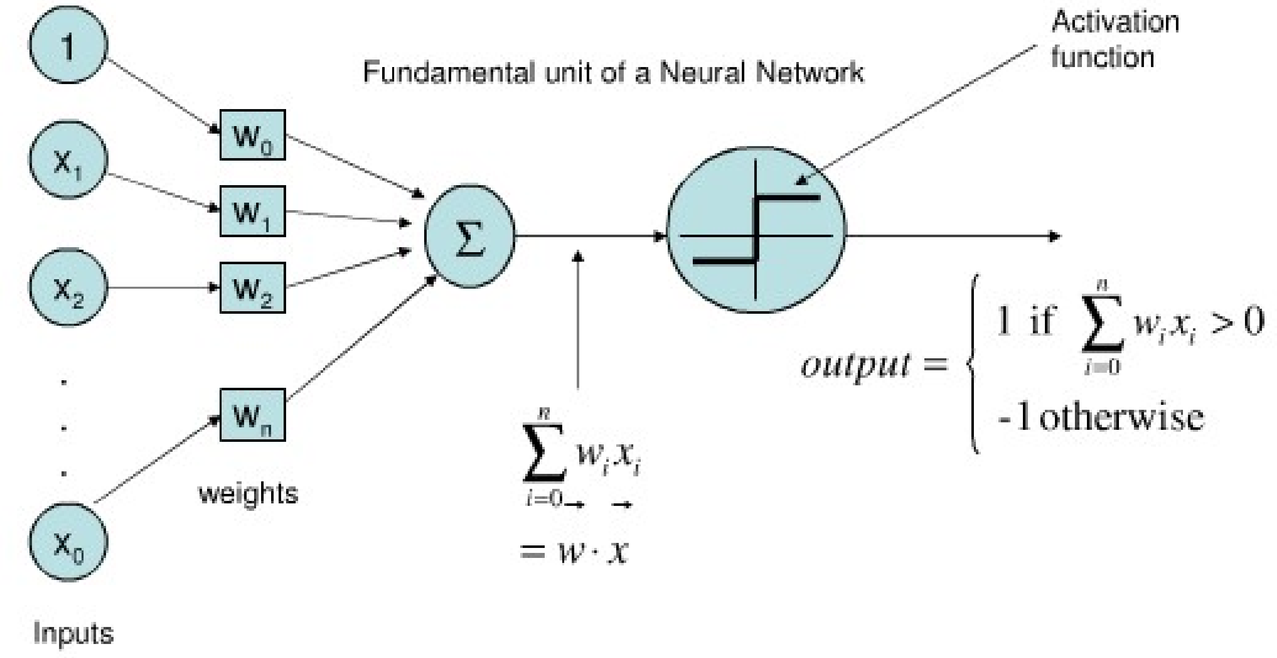
- Recent machine learning methods for training “deep” neural networks (NNs) have demonstrated remarkable progress on many challenging AI problems (e.g. speech recognition, visual object recognition, machine translation, game playing).
- However, their capabilities are prone to “hype.”
- Deep learning has not “solved” AI and current methods have clear limitations.

# Very Brief History of Machine Learning

- Single-layer neural networks (1957-1969)
- Symbolic AI & knowledge engineering (1970-1985)
- Multi-layer NNs and symbolic learning (1985-1995)
- Statistical (Bayesian) learning and kernel methods (1995-2010)
- Deep learning (CNNs and RNNs) (2010-?)

# Single-Layer Neural Network
## (Linear Threshold Unit)

- Mathematical model of an individual neuron.


Fundamental unit of a Neural Network
Activation function
1
$x_1$
$x_2$
$x_0$
Inputs
$w_0$
$w_1$
$w_2$
$w_n$
weights
$\Sigma$
$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

# Perceptron

- Rosenblatt (1957) developed an iterative, hill-climbing algorithm for learning the weights of single-layer NN to try to fit a set of training examples.
- Unable to learn or represent many classification functions (e.g. XOR), only the “linearly separable” ones are learnable.

# Perceptron Learning Rule

- Update weights by:

$$w_i = w_i + \eta(t - o)x_i$$

where η is the "learning rate," *t* is the teacher output, and *o* is the network output.

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs

# Perceptron Learning Algorithm

- Iteratively update weights until convergence.

```
Initialize weights to random values
Until outputs of all training examples are correct
    For each training pair, $E$, do:
        Compute current output $o$ for $E$ given its inputs
        Compare current output to target value, $t$ , for $E$
        Update weights using learning rule
```

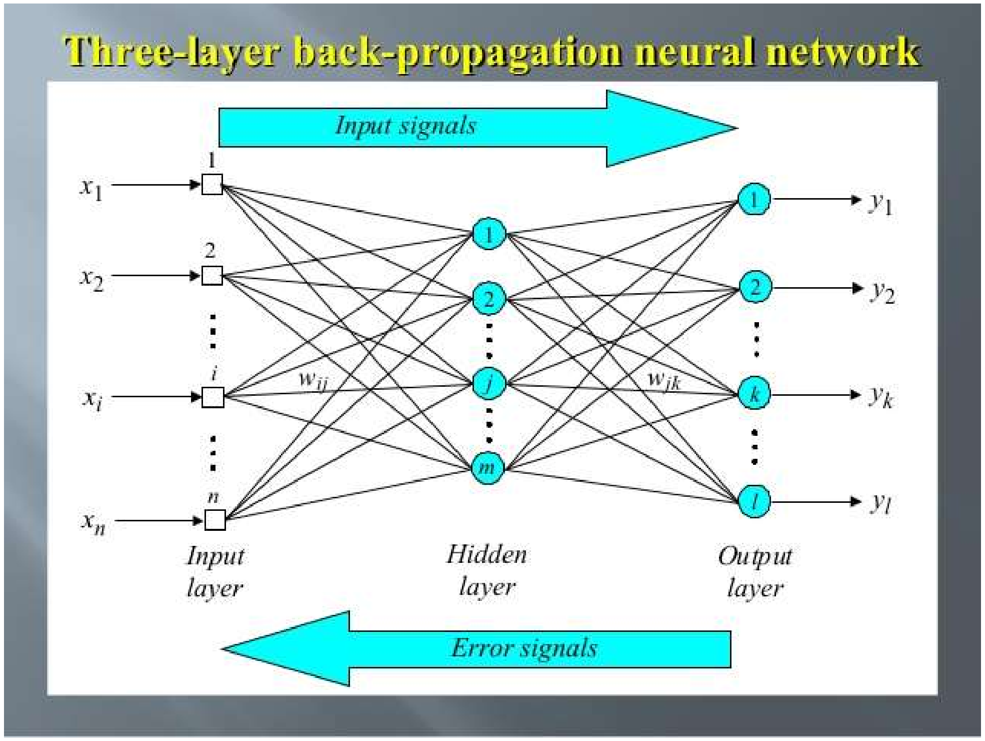
# Perceptron Demise

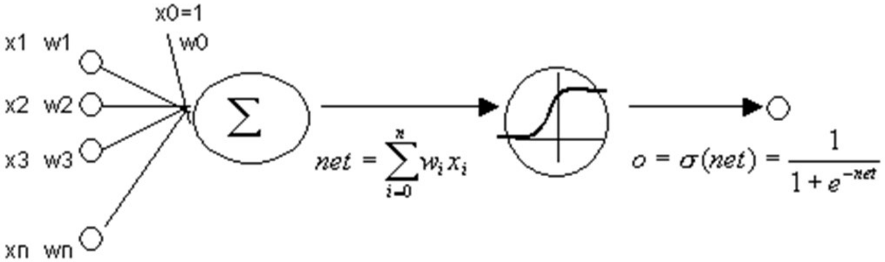
- *Perceptons* (1969) by Minksy and Papert illuminated the limitations of the perceptron.
- Work on neural-networks dissipated during the 70’s and early 80’s.

# Neural Net Resurgence (1986)

- Interest in NNs revived in the mid 1980’s due to the rise of “connectionism.”
- Backpropagation algorithm popularized for training three-layer NN’s.
- Generalized the iterative “hill climbing” method to approximate fitting two layers of synaptic connections, but no convergence guarantees.

# 3-Layer NN Backpropagation




x0=1
w0
x1 w1
x2 w2
x3 w3
xn wn
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$

[The sigmoid threshold unit]

# Second NN Demise (1995-2010)

- Generic backpropagation did not generalize that well to training deeper networks.
- Little theoretical justification for underlying methods.
- Machine learning research moved to graphical models and kernel methods.

# Deep Learning Revolution (2010…)

- Improved methods developed for training deep neural works.
- Particular successes with:
  - Convolutional neural nets (CNNs) for vision.
  - Recurrent neural nets (RNNs) for machine translation and speech recognition.
  - Deep reinforcement learning for game playing.

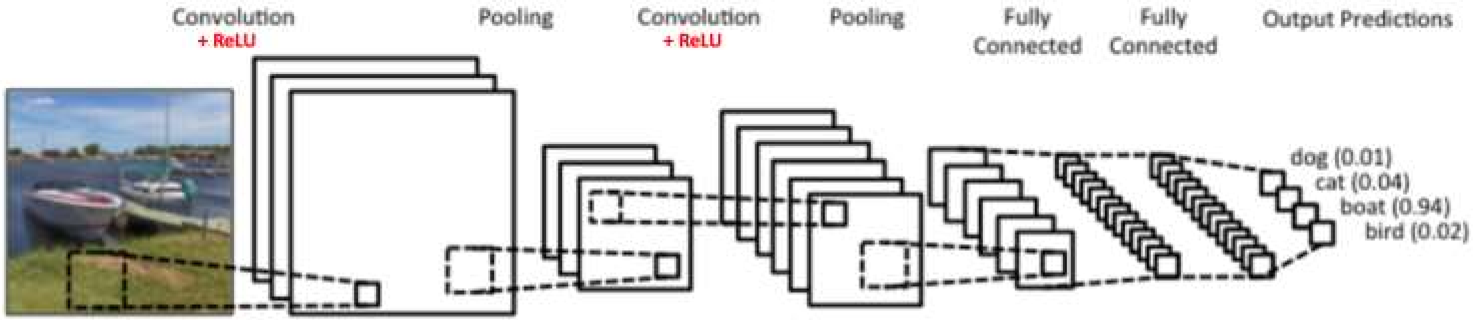
# Massive Data and Specialized Hardware

- Large collections of supervised (crowdsourced) training data has been critical.
- Efficient processing of this big data using specialized hardware (Graphics Processing Units, GPUs) has been critical.

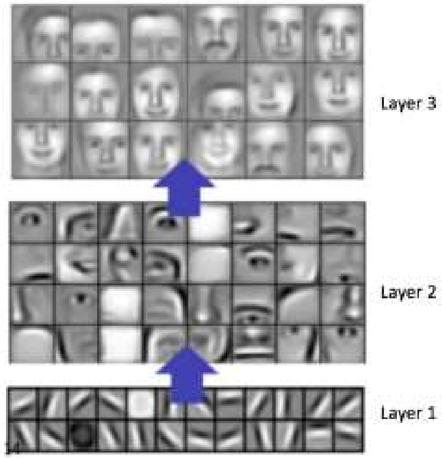
# CNNs

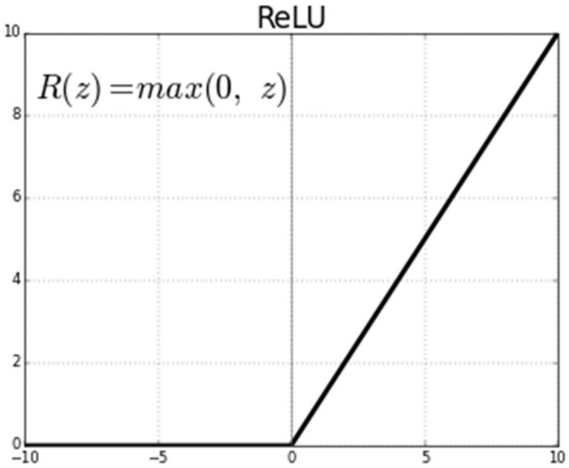
- Convolutional layers learn to extract local features from image regions (receptive fields) analogous to human vision (LeCun, et al., 1998).
- Deeper layers extract higher-level features.
- Pool activity of multiple neurons into one at the next layer using max or mean.
- Nonlinear processing with Rectified Linear Units (ReLUs)
- Decision made using final fully connected layers.

# CNNs



Increasingly broader local features extracted from image regions





# ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Recognize 1,000 categories of objects in 150K test images (given 1.2M training images).

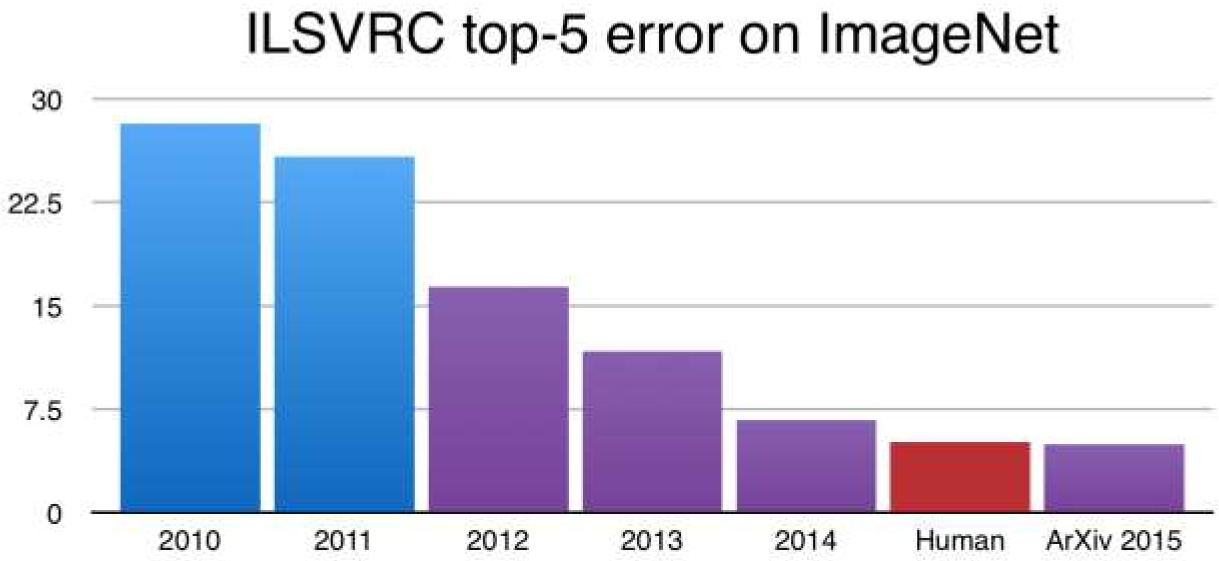
Mongoose

Canoe

Missile

Trombone

# ImageNet Performance Over Time



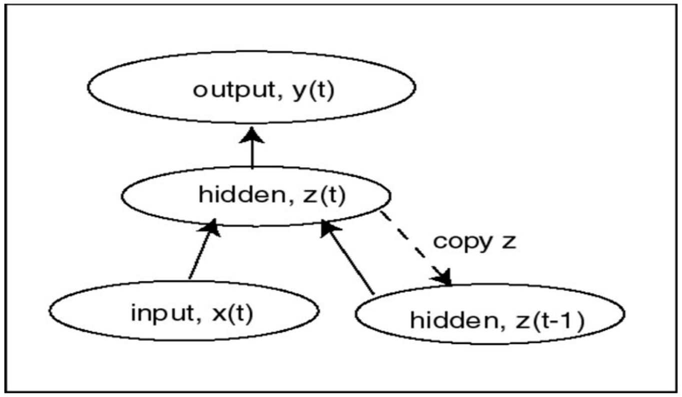
CNNs introduced

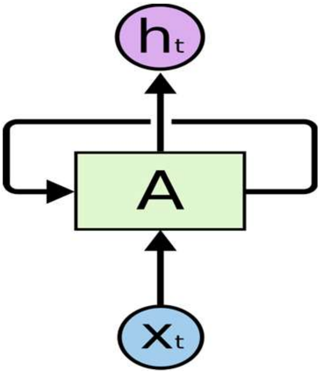
# Recurrent Neural Networks (RNNs)

- Add feedback loops where some units' current outputs determine some future network inputs.
- RNNs can model dynamic finite-state machines, beyond the static combinatorial circuits modeled by feed-forward networks.

# Simple Recurrent Network (SRN)

- Initially developed by Jeff Elman ("*Finding structure in time*," 1990).
- Additional input to hidden layer is the state of the hidden layer in the previous time

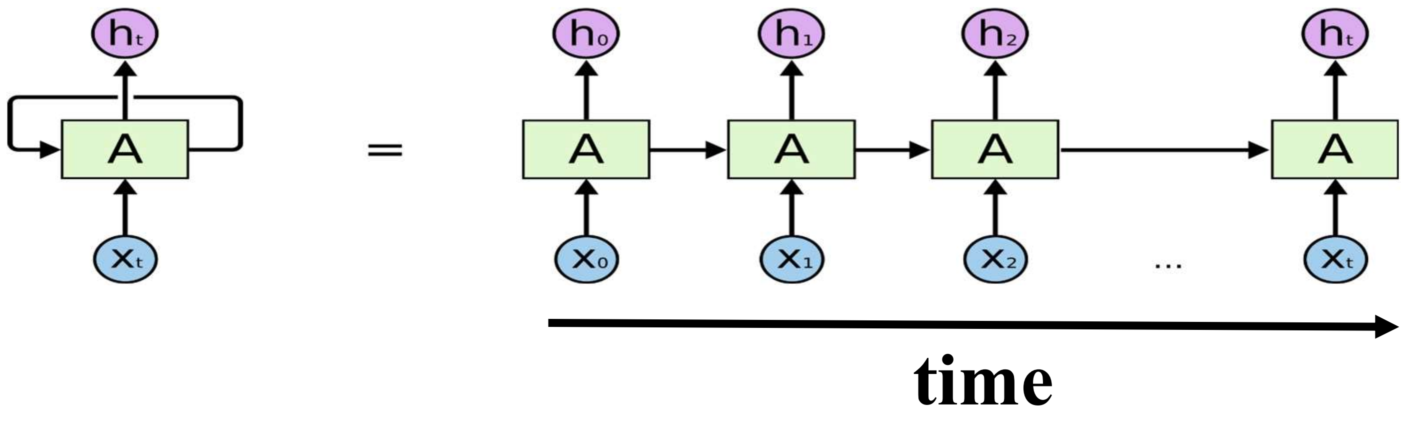




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

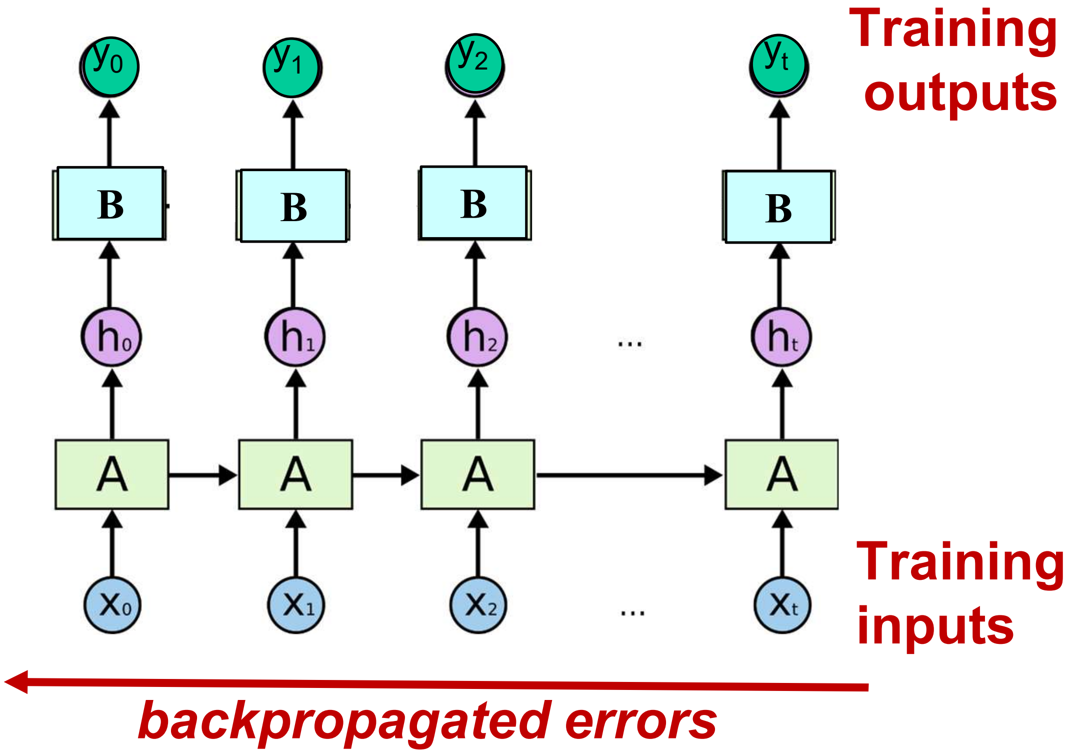
# Unrolled RNN

- Behavior of RNN is perhaps best viewed by "unrolling" the network over time.



# Training RNN's

- RNNs can be trained using "backpropagation through time."
- Can viewed as applying normal backprop to the unrolled network.



Training outputs

Training inputs

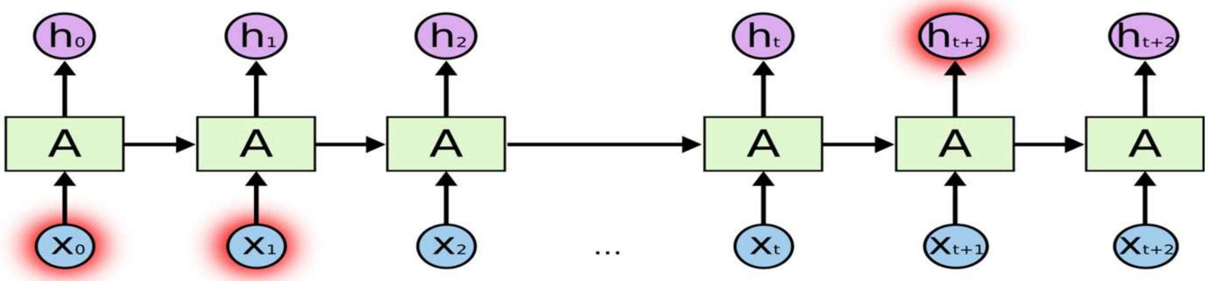
*backpropagated errors*

# Vanishing/Exploding Gradient Problem

- Backpropagated errors multiply at each layer, resulting in exponential decay (if derivative is small) or growth (if derivative is large).
- Makes it very difficult train deep networks, or simple recurrent networks over many time steps.

# Long Distance Dependencies

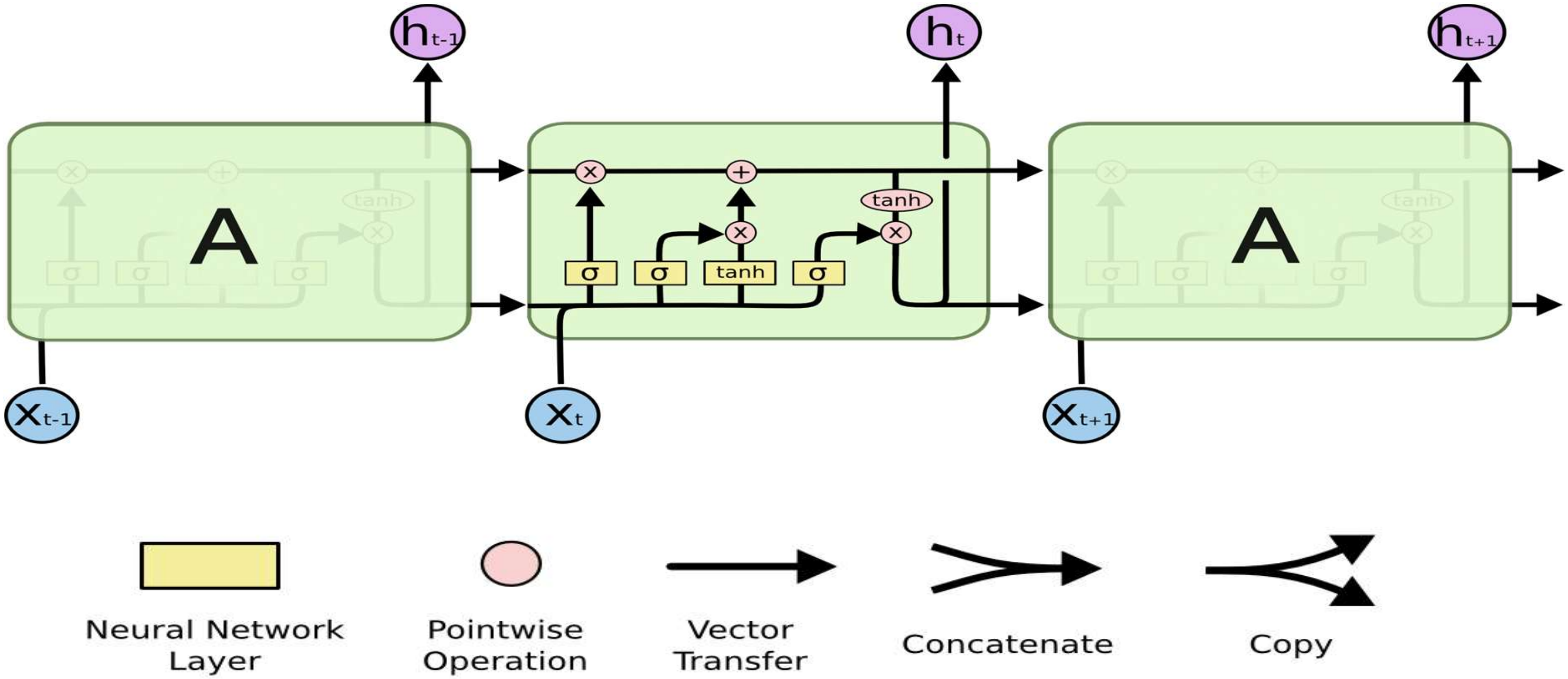
- It is very difficult to train SRNs to retain information over many time steps.
- This make is very difficult to learn SRNs that handle long-distance dependencies, such as subject-verb agreement.



# Long Short Term Memory (LSTM)

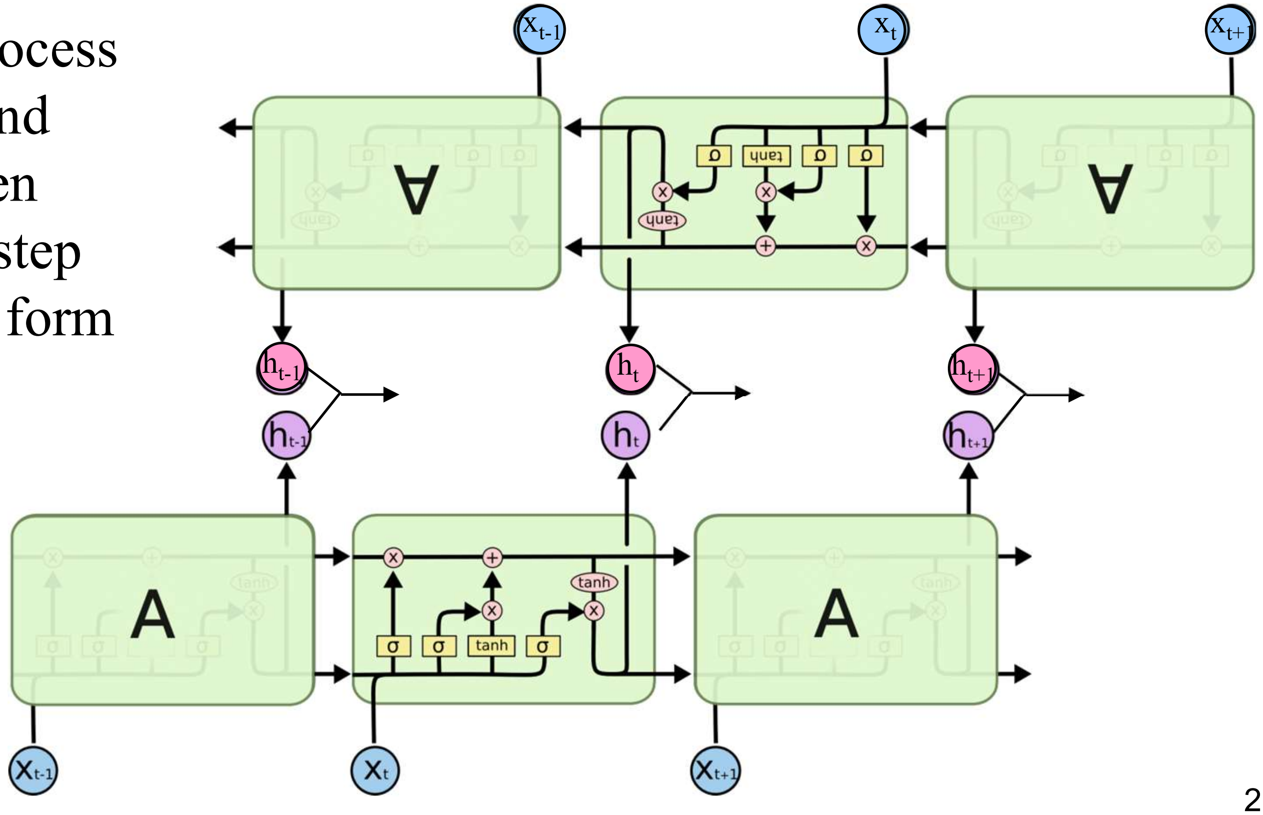
- LSTM networks, add additional gating units in each memory cell (Hochreiter & Schmidhuber, 1997).
  - Forget gate
  - Input gate
  - Output gate
- Prevents vanishing/exploding gradient problem and allows network to retain state information over longer periods of time.

# LSTM Network Architecture



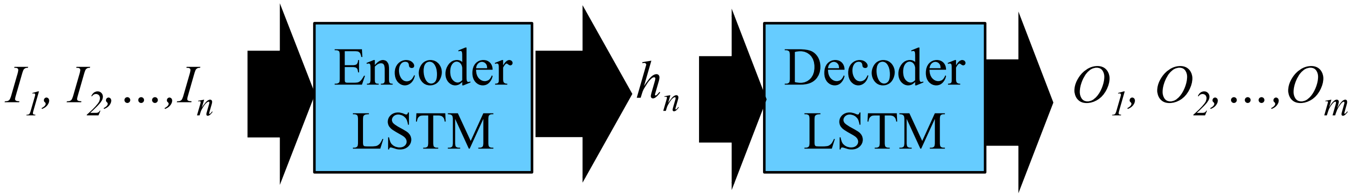
# Bi-directional LSTM (Bi-LSTM)

- Separate LSTMs process sequence forward and backward and hidden layers at each time step are concatenated to form the cell output.



# Sequence to Sequence (Seq2Seq) Transduction

- Encoder/Decoder framework maps one sequence to a "deep vector" then another LSTM maps this vector to an output sequence (Sutskever et al., 2014).



- Train model "end to end" on I/O pairs of sequences.

# Neural Machine Translation (NMT)

- LSTM Seq2Seq has lead to a new approach to translating human language.
- NMT modestly outperforms previous statistical learning approaches to MT (SMT).

# NMT Results (Wu et al., 2016)

- Experimental results using automated (BLEU) and human evaluation for English→ French translation.

| Method | BLEU | Human Rating |
|---|---|---|
| SMT | 37.0 | 3.87 |
| NMT | 40.35 | 4.46 |
| Human | | 4.82 |

# LSTM Application Architectures

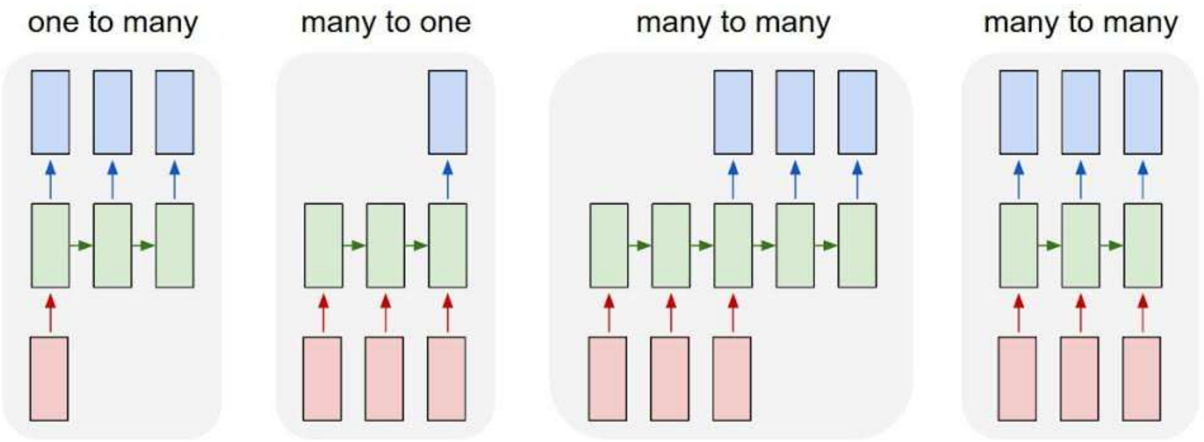


Image Captioning

Video Activity Recog
Text Classification

Video Captioning
Machine Translation

POS Tagging, IE,
Language Modeling

# Independent Word Vectors

- Represent word meanings as vectors based on words with which they co-occur.
- Neural approaches based on predicting a word’s context (skip-grams) from its vector (Word2Vec, Mikolov et al., 2013).
- Fails to account for lexical ambiguity or dependence of word meaning on context.

# Bidirectional Language Model

- A standard statistical language model predicts the probability of the next word based on the previous context.
  - Your program for Project 4 does not _____
- A bidirectional language model (BiLM) predicts the word at each position based on both prior and posterior context encoded using an RNN (e.g. LSTM).

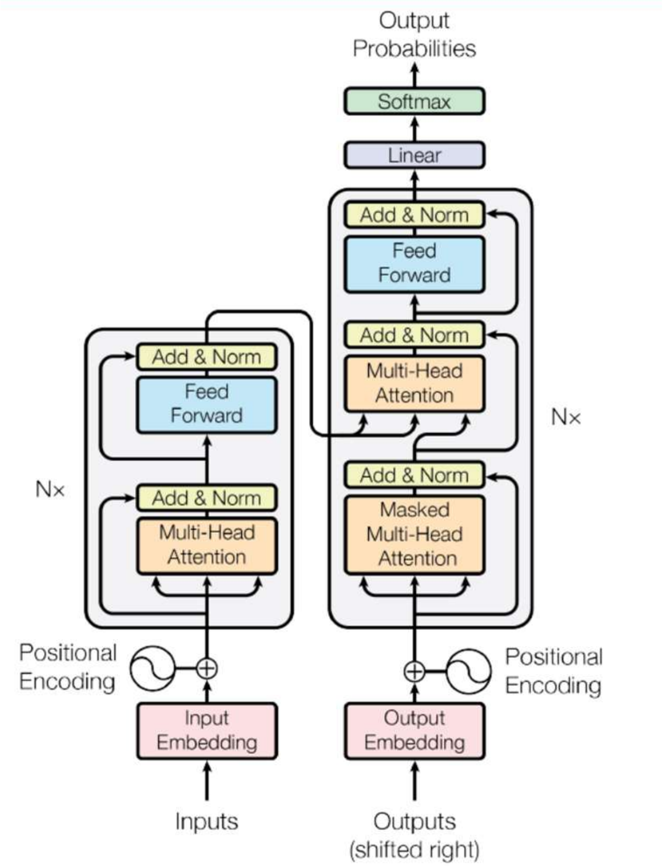
# Contextualized Word Embeddings

- Produce a vector representation for a specific occurrence of a word, by using textual context to compute its meaning.
- ELMo (Embeddings from Language Models, Peters et al., 2018) uses the hidden state of a BiLM to compute contextualized word embeddings.

# Transformer Networks

- An alternate Seq2Seq neural architecture based on attention rather than recurrence (Vaswani et al., 2017).
- Attention mechanisms compute the output at each position in the sequence by varying “attention” across different positions in the input sequence.

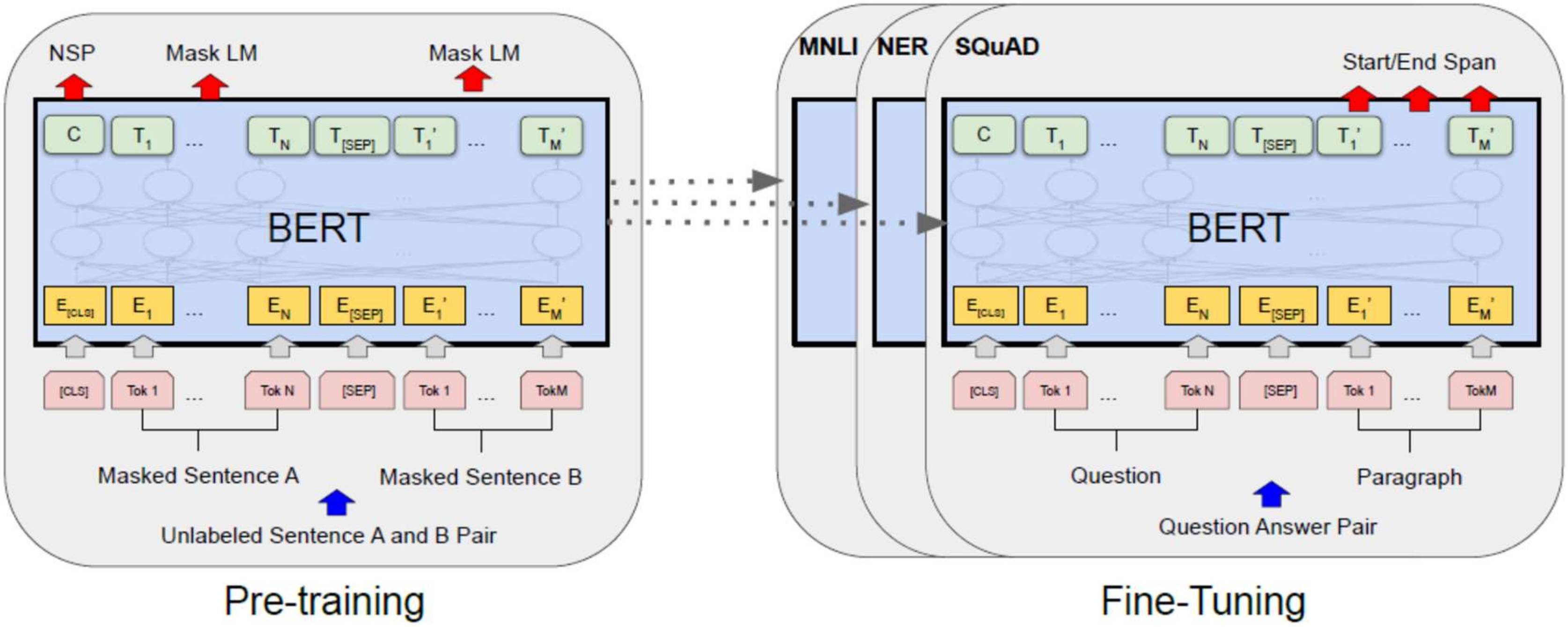
# Transformer Architecture



# BERT Contextualized Embeddings

- Bidirectional Encoder Representations from Transformers (BERT, Devlin et al., 2018)
- Trains a transformer network to predict a fraction of "masked" tokens in an input sentence, or predict the next sentence.

# BERT Architecture



# Neural Information Retrieval

- Word embeddings have been used to improve IR by allowing matching words based on semantic similarity.
- Most recent results (Dai & Callan, SIGIR-2019) show improvements to ad-hoc document retrieval using BERT transformer approach.

# BERT IR Results

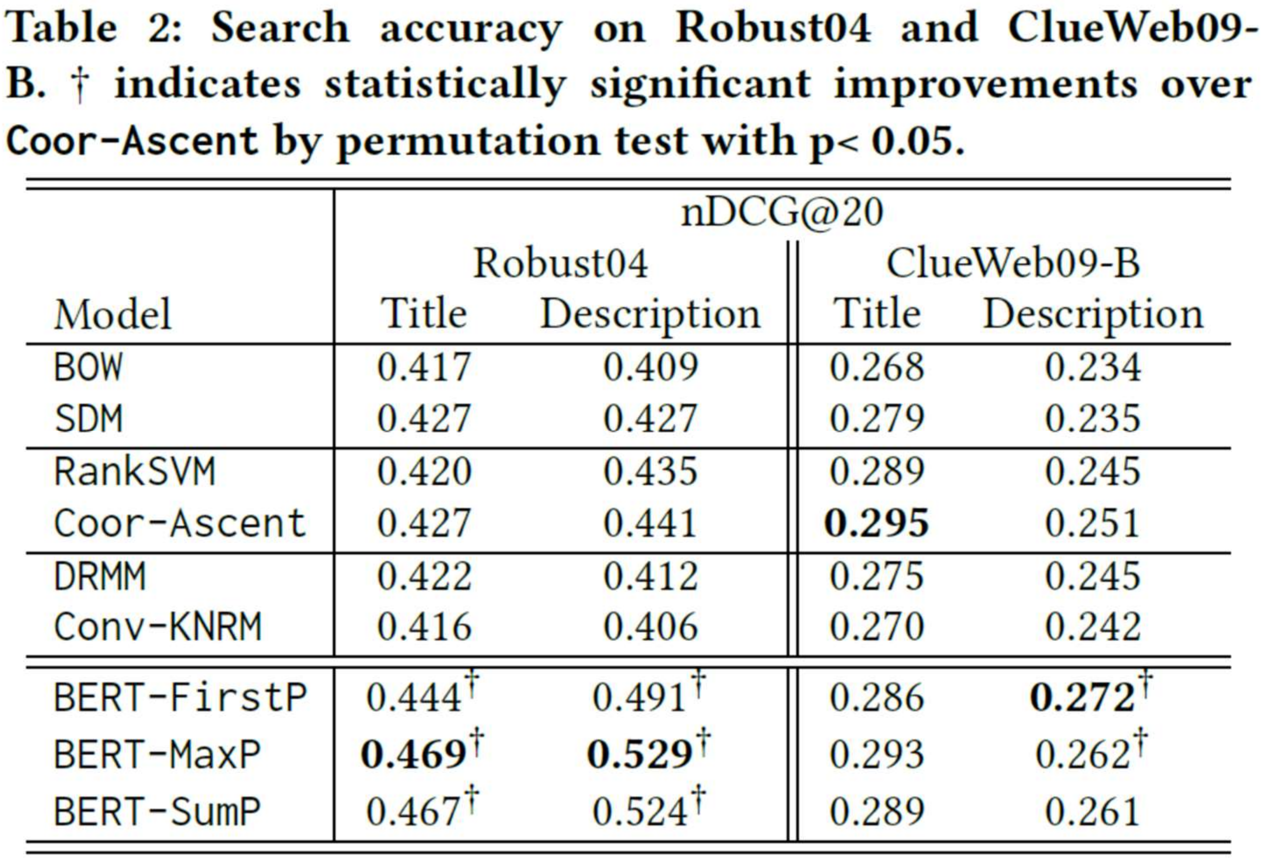


Table 2: **Search accuracy on Robust04 and ClueWeb09-B. † indicates statistically significant improvements over Coor-Ascent by permutation test with p< 0.05.**

| | nDCG@20 | | | |
|---|---|---|---|---|
| | Robust04 | | ClueWeb09-B | |
| Model | Title | Description | Title | Description |
| BOW | 0.417 | 0.409 | 0.268 | 0.234 |
| SDM | 0.427 | 0.427 | 0.279 | 0.235 |
| RankSVM | 0.420 | 0.435 | 0.289 | 0.245 |
| Coor-Ascent | 0.427 | 0.441 | **0.295** | 0.251 |
| DRMM | 0.422 | 0.412 | 0.275 | 0.245 |
| Conv-KNRM | 0.416 | 0.406 | 0.270 | 0.242 |
| BERT-FirstP | 0.444$^\dagger$ | 0.491$^\dagger$ | 0.286 | **0.272**$^\dagger$ |
| BERT-MaxP | **0.469**$^\dagger$ | **0.529**$^\dagger$ | 0.293 | 0.262$^\dagger$ |
| BERT-SumP | 0.467$^\dagger$ | 0.524$^\dagger$ | 0.289 | 0.261 |

# “Cramming” Meaning into Vectors

- DNNs force semantics to be encoded into real-valued vectors.
- Structured meaning representations that exploit trees, graphs, and logical representations are only imperfectly encoded as vectors.

# Complex Compositional Questions

“Has Woody Allen made more movies with Diane Keaton or Mia Farrow.”

$$\underset{\substack{X\in\{DianeKeaton,\\ MiaFarrow\}}}{arg\max}\ count(Y, Director(Y, WoodyAllen) \\ \wedge Cast(Y, X))$$

# Conclusions

- Machine learning, and specifically neural nets, has a a long, rich, varied history.
- Deep learning has made significant recent progress.
- Progress is continuing and holds promise of enabling revolutionary technology.
- However, progress has been exaggerated and core AI problems are a long way from completely solved.