# INTRODUCTION TO DEEP LEARNING

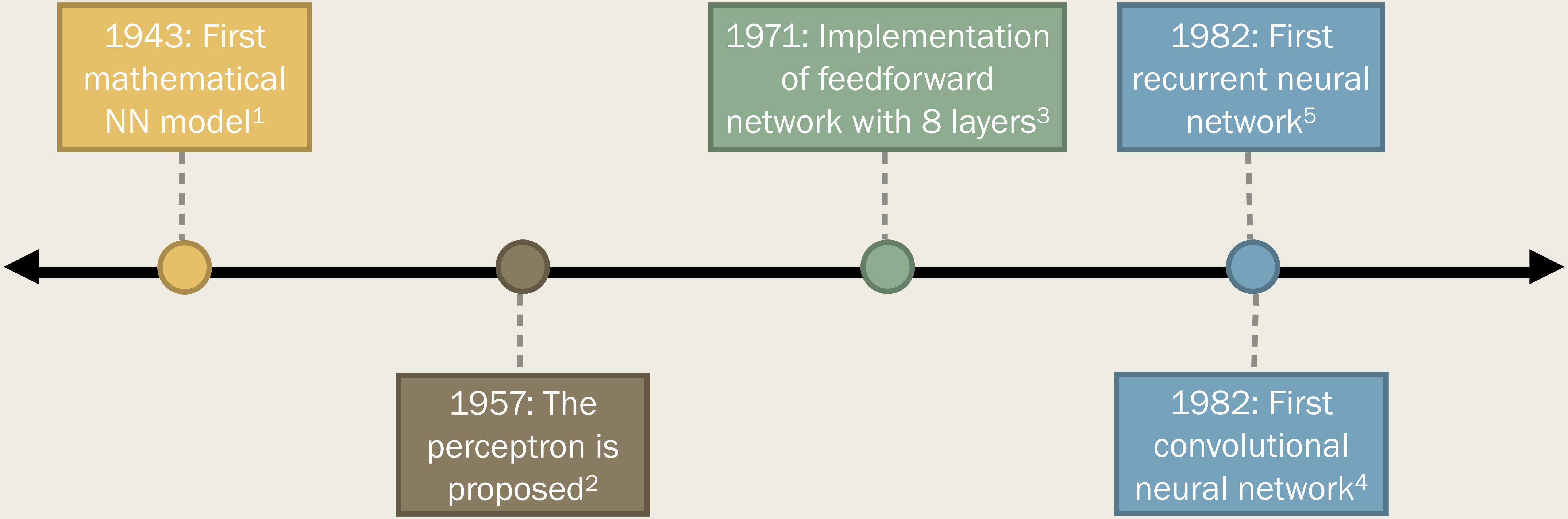
Natalie Parde
parde@uic.edu

CS 594: Language and Vision
Spring 2019

# What is deep learning?

- A machine learning approach that automatically learns features directly from data, employing a neural network with one or more **hidden layers** to do so.
- Often associated with **end-to-end learning**
  - *Put raw input in one end*
  - *Receive output from the other*

# Deep learning isn't new.



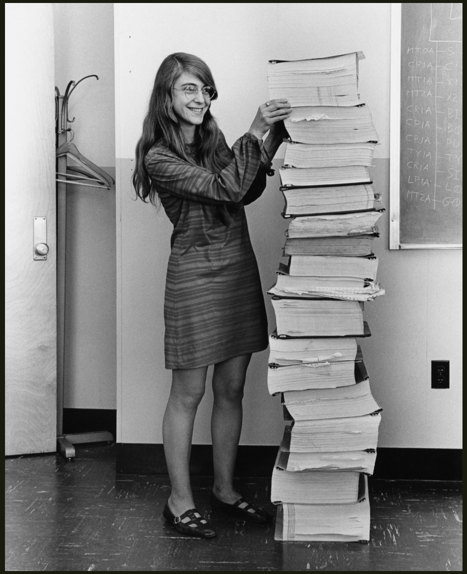

[1]McCulloch, W. S., and W. Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

[2]Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[3]Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, (4), 364-378.

[4]Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267-285). Springer, Berlin, Heidelberg.

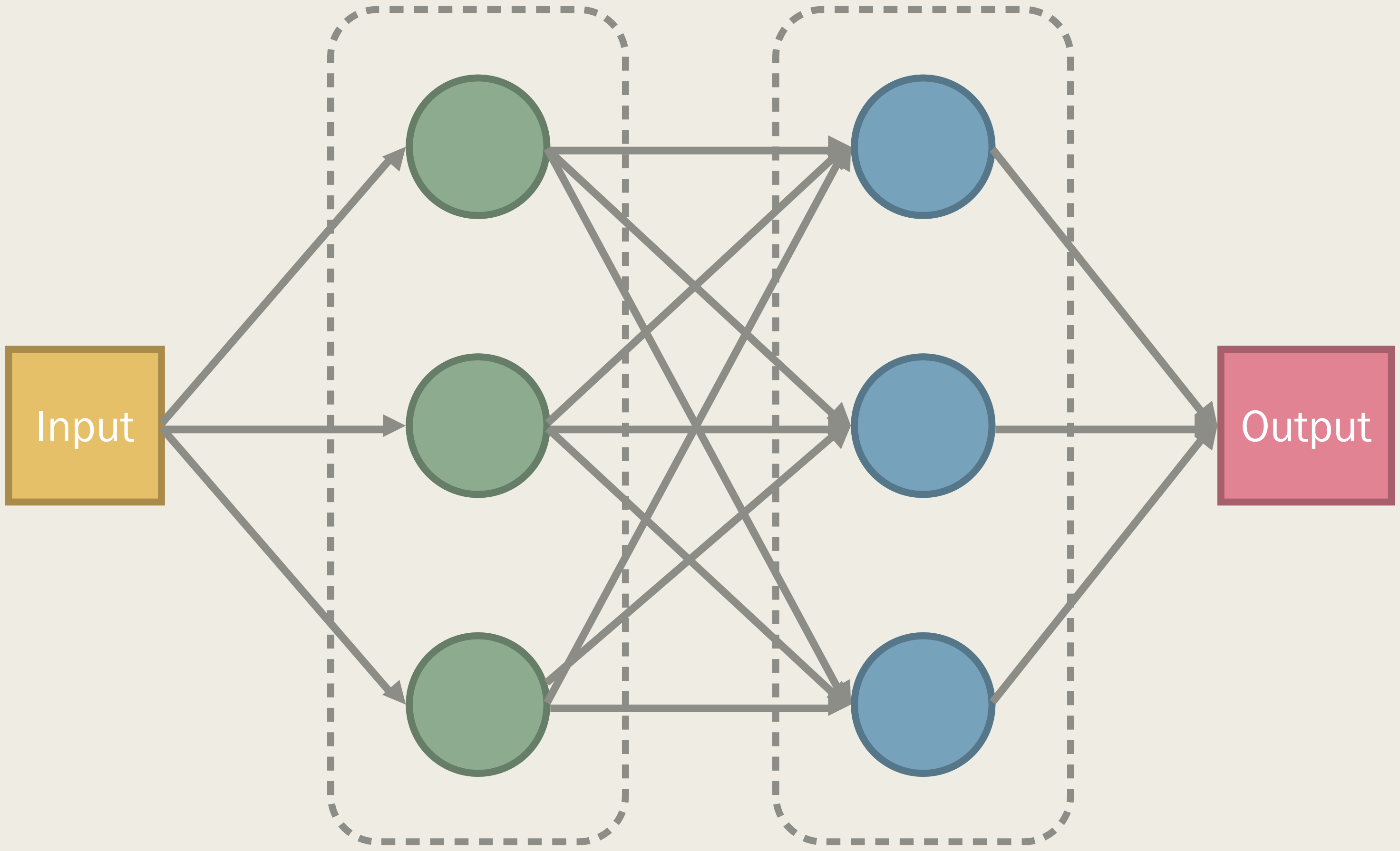
[5]Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554-2558.



# Why hasn’t it been a big deal until recently?

- Data
- Computing power

# Neural Networks



# Types of Neural Networks

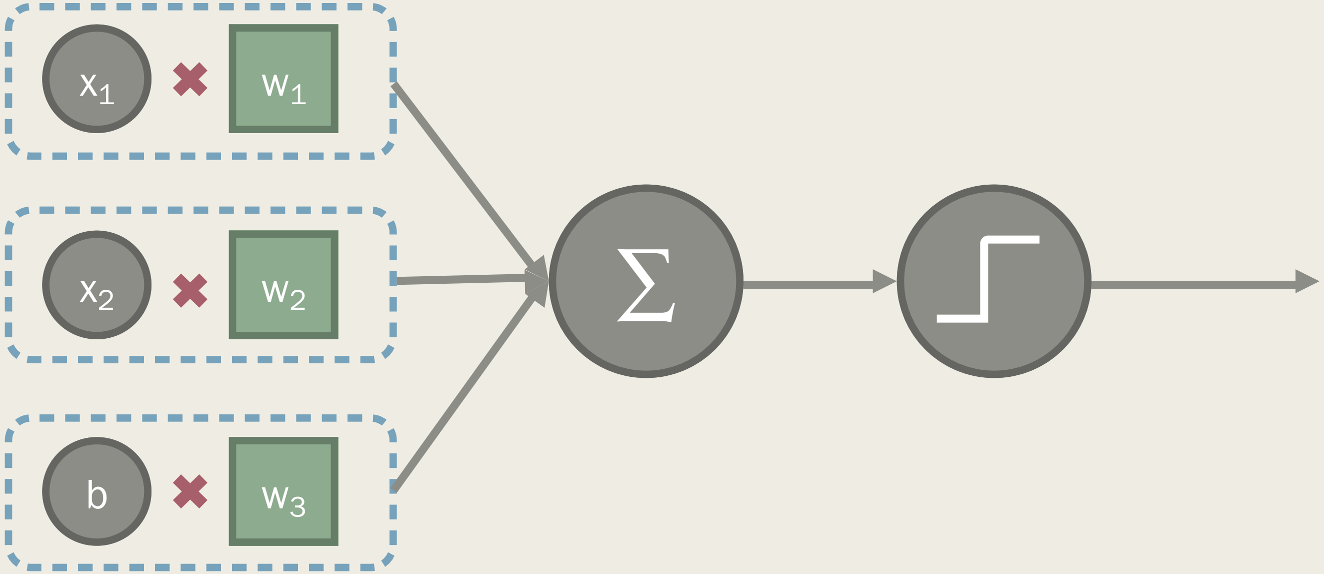
- Feedforward Neural Network
- Convolutional Neural Network
  - *LeNet*
  - *ResNet*
- Recurrent Neural Network
  - *LSTM*
  - *BiLSTM*
  - *GRU*
- Generative Adversarial Network
- Sequence-to-Sequence Network
- Autoencoder

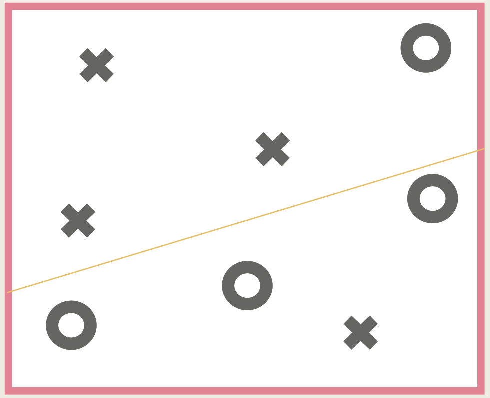
# Feedforward Neural Networks

- Earliest and simplest form of neural network
- Data is fed forward from one layer to the next
- Each layer:
  - *One or more perceptrons*
  - *A perceptron in layer n receives input from all perceptrons in layer n-1 and sends output to all perceptrons in layer n+1*
  - *A perceptron in layer n does not communicate with any other perceptrons in layer n*
- The outputs of all perceptrons except for those in the last layer are hidden from external viewers

# What is a perceptron?

- A function that outputs a binary value based on whether or not the product of its inputs and associated weights surpasses a threshold
- Learns this threshold iteratively by trying to find the boundary that is best able to distinguish between data of different categories

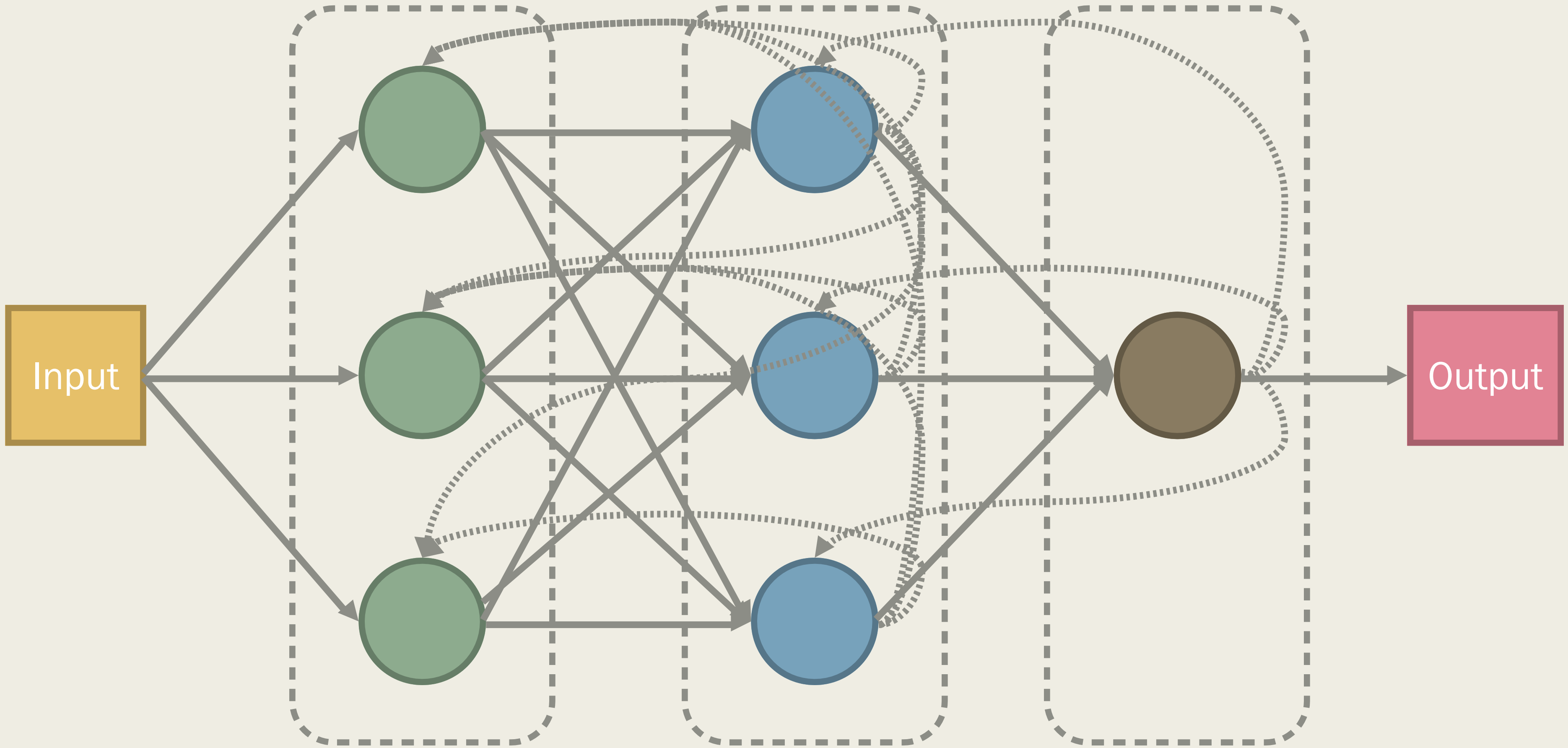


# How do feedforward neural networks improve over time?

- Backpropagation!
- Weights in each neuron (neuron = individual perceptron) are updated after a training epoch finishes to minimize the error between their real and desired output
- These updates begin at the output layer (where the error is known) and propagate backward through the network's hidden layers until the first layer is reached

# What does this look like altogether?



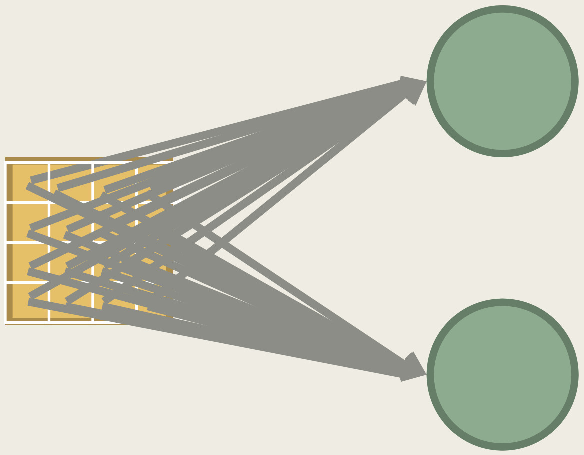
# Think, Pair, Share

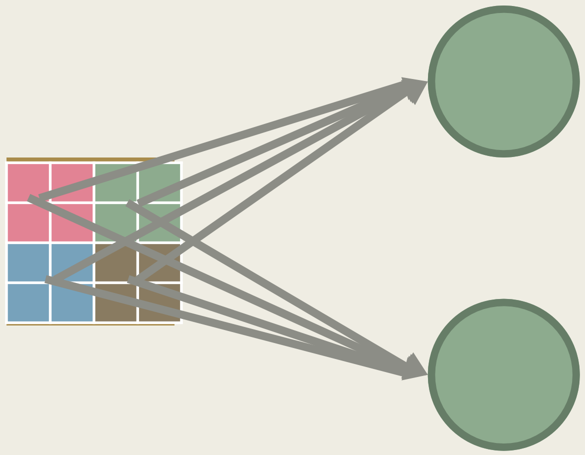
- Think of three shortcomings of standard feedforward neural networks, and one way that you might want to address each of those shortcomings, and write them on your notecard
- Share those ideas with a partner
- Choose one example to share with the class

- Timer: https://www.google.com/search?q=timer

# Convolutional Neural Networks

- Feedforward neural network with one or more **convolutional layers**
  - *Sliding windows that perform matrix operations on subsets of the input*
- Designed to reflect the inner workings of the visual cortex system …perhaps unsurprisingly, CNNs are primarily used for computer vision tasks!
- CNNs require that fewer parameters are learned relative to standard feedforward networks for equivalent input data





# Types of Layers in CNNs

## Convolutional layer

- *Computes products between the cells in a weight matrix and the original input matrix for a local region*

## Pooling layer

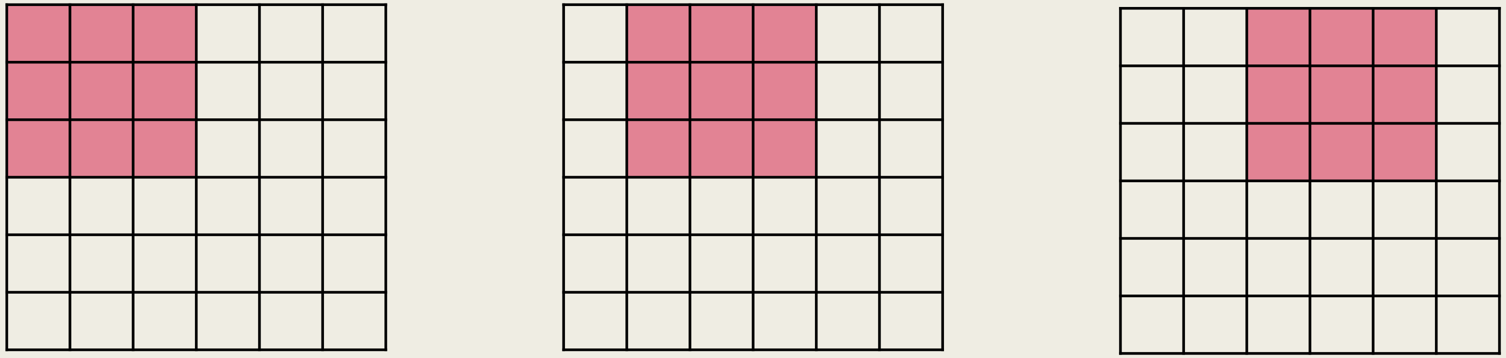
- *Reduces the dimensionality of the input by pooling the products computed in the convolutional layer to a single value*

## Fully-connected layer

- *Identical to that seen in standard feedforward neural networks*

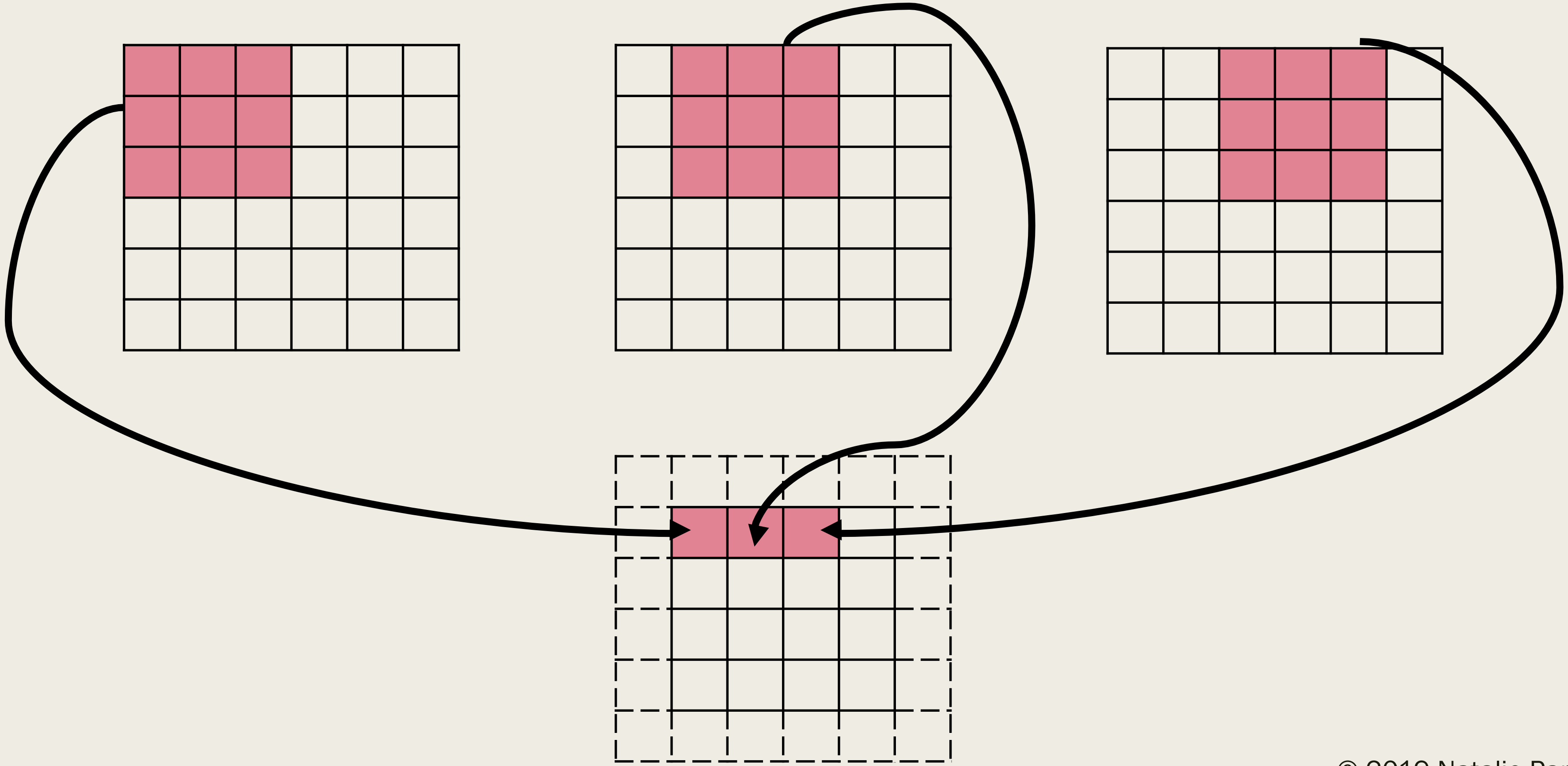
# Convolutional Layers

- First layer(s): low-level features
  - *Color, gradient orientation*
- Higher layer(s): high-level features
  - *Car, train, plane*
- Layers can have varying numbers of filters, or **feature maps**



# Pooling Layers

- Max Pooling
- Average Pooling



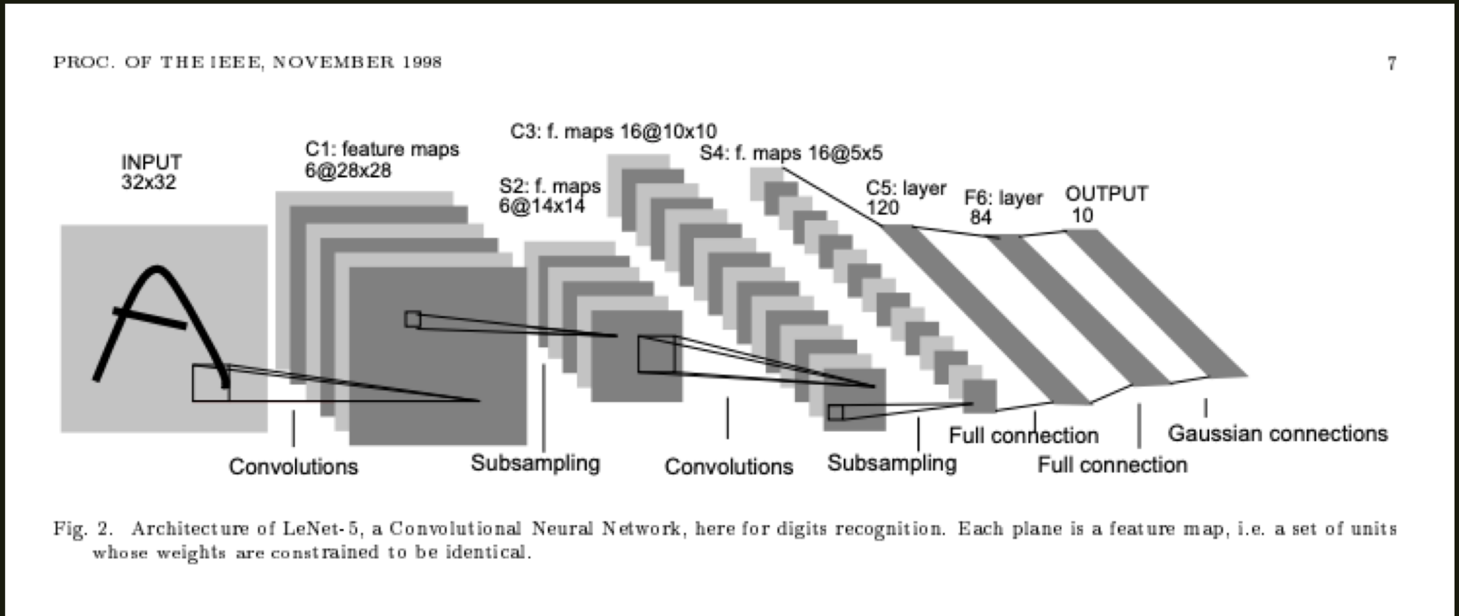


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

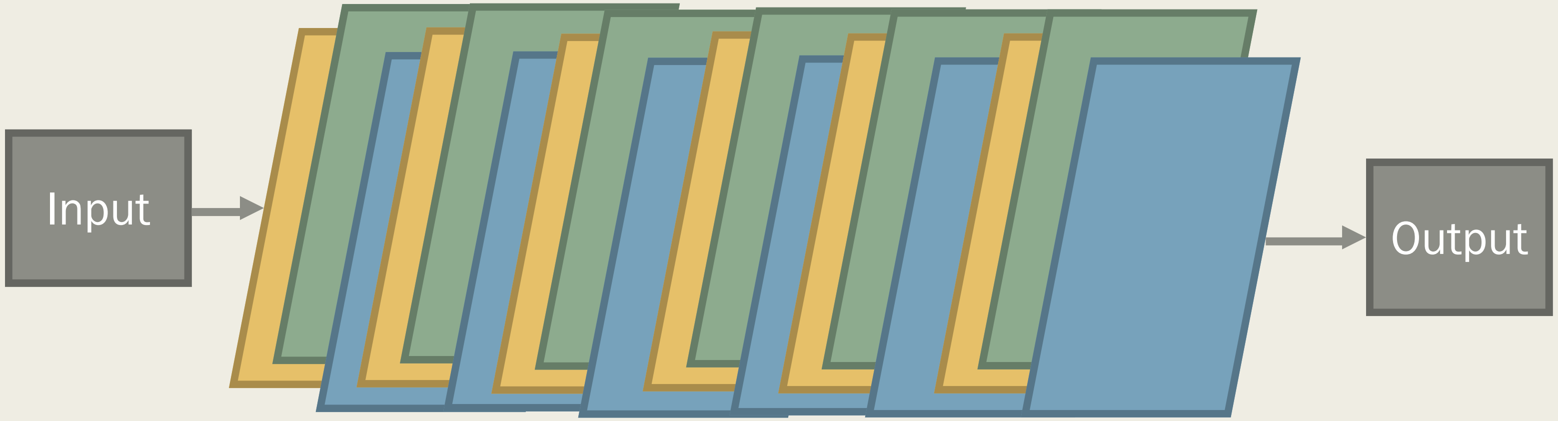
# LeNet

- First successful CNN[1]
- 7 layers
  - *3 convolutional*
  - *2 pooling*
  - *1 fully-connected*
  - *1 softmax output*
- 5x5 convolutions with stride size = 1
- 2x2 average pooling

[1]LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

# ResNet

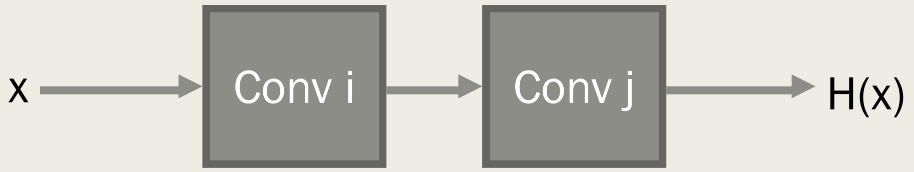
- Residual Network[1]
- Unique characteristics:
  - *Residual connections*
  - *No fully-connected layers at the end of the network*
- Opened the door to networks with hundreds or even 1000+ layers



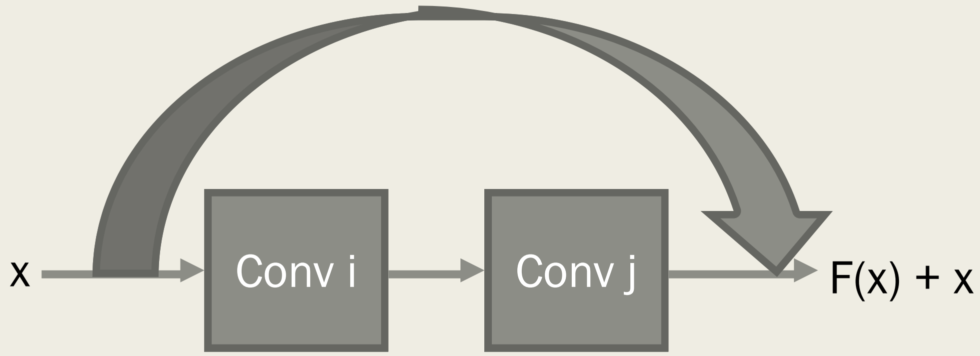

[1]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

# What is a residual connection?

- Rather than learning a full mapping (H(x)) from layer *i* to *j*, the model learns the difference (F(x)) between that mapping and the input to layer *i*
  - *More simply: What do we have to learn to get from x to H(x)?*



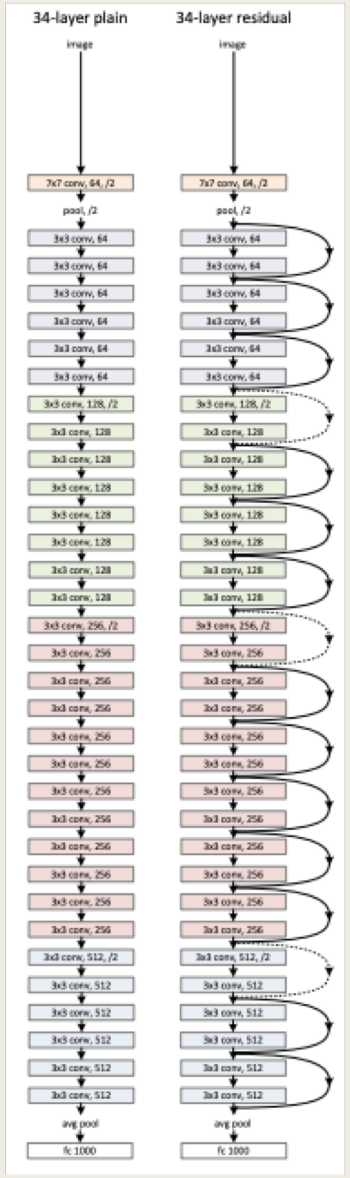

Normal Connection




Residual Connection

# ResNet Architecture

- Residual blocks:
  - *Two 3x3 convolutional layers*
- Periodically downsamples the data and doubles the number of feature maps in the convolutional layer




He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

# Recurrent Neural Networks

- Neural network model designed specifically to handle sequential data
- Particularly good for tasks like language modeling, image captioning, and other forms of predictive generation!

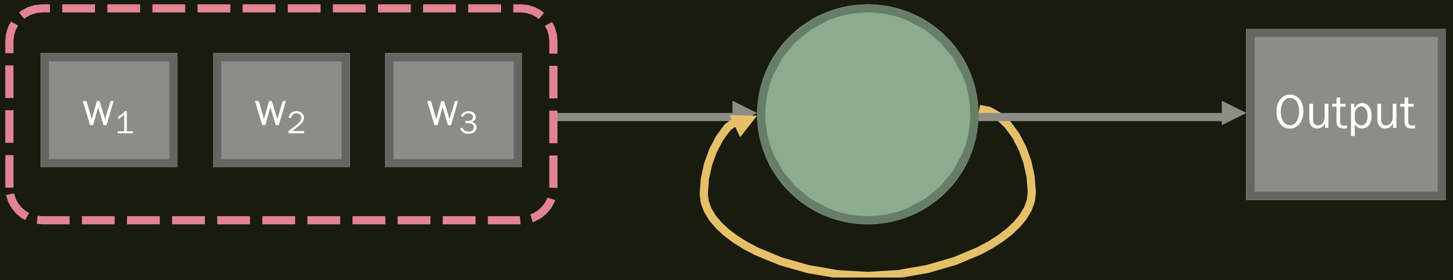
**Artificial intelligence can learn to write like Shakespeare. Can you tell the difference?**
**- Australian Broadcasting Corporation**

**The world's most prolific writer is a Chinese algorithm**
**- British Broadcasting Corporation**

**When an AI Goes Full Jack Kerouac**
**- The Atlantic**

# How do RNNs differ from standard feedforward neural networks?

- Memory!
  - *Loops in the network that allow information to persist over time*
- Information is stored between timesteps using an internal hidden state, and fed back into the model the next time it reads an input
  - *Some type of output is also predicted at each timestep*
- New hidden states are determined as a function of the existing hidden state and the new input at the current timestep
  - *This function remains the same across timesteps*




# Standard RNN

# Types of RNNs

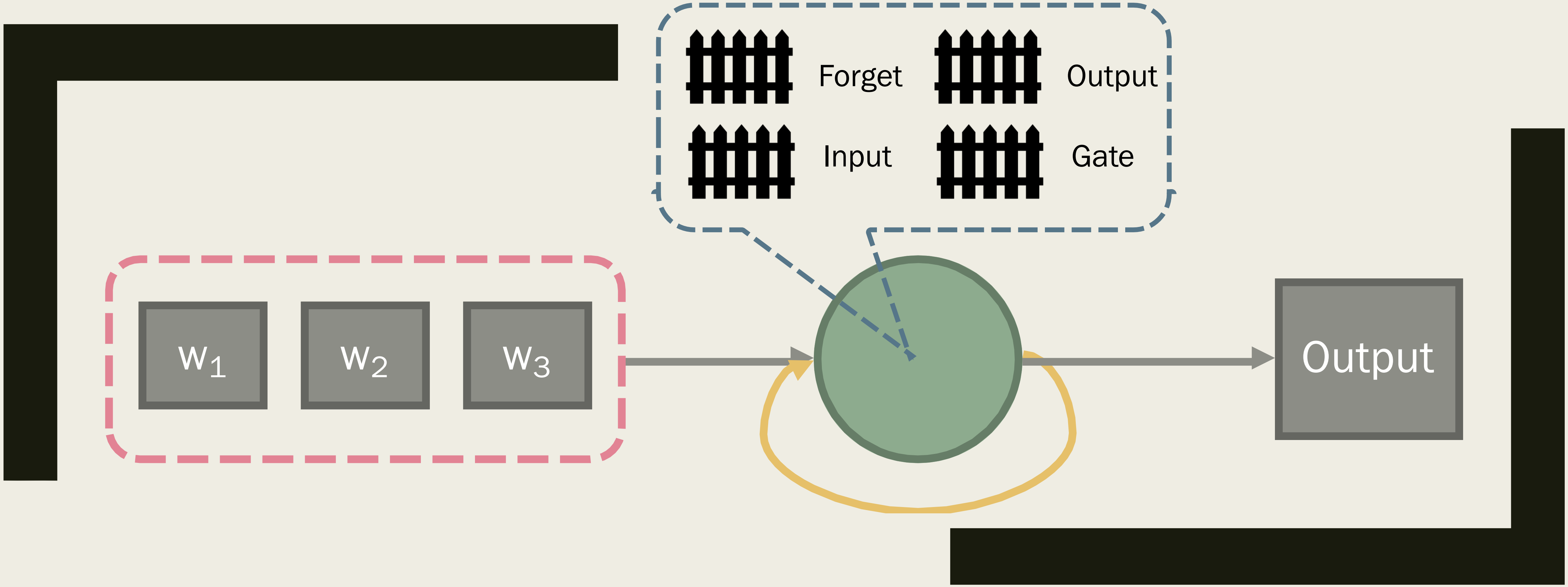
LSTM

BiLSTM

GRU

# Long Short Term Memory Networks

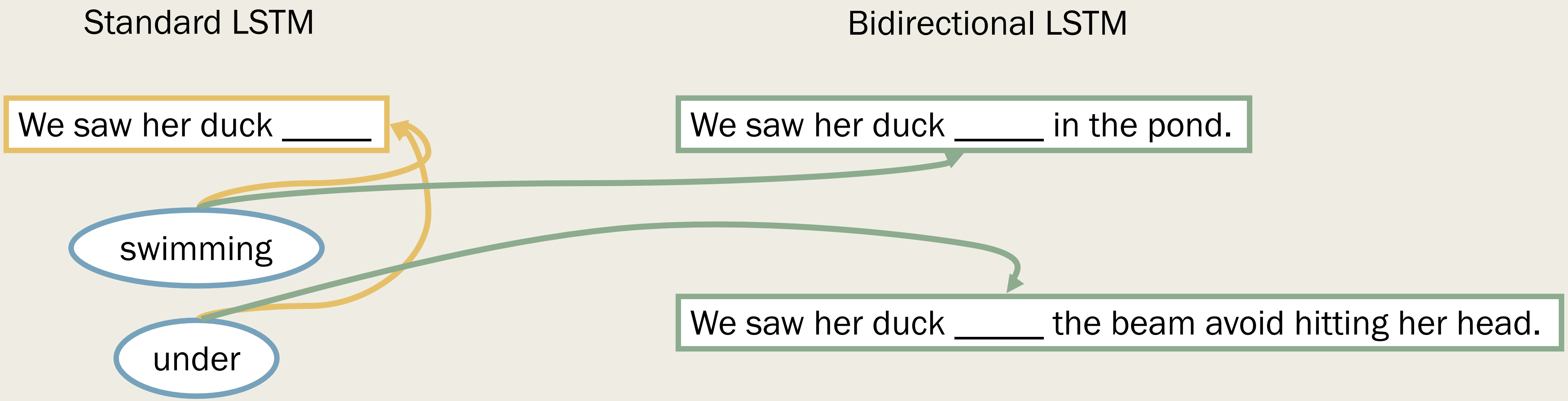
- Not one, but **two** hidden states persist through each timestep
  - *Hidden state*
  - *Cell state*
- The new input and the current hidden state are multiplied with a weight matrix to produce four gates:
  - *Forget gate: Should we erase this information from the cell?*
  - *Input gate: Should we write new information to the cell?*
  - *Gate gate: How much should we write?*
  - *Output gate: How much should we reveal as output?*
- The cell state is used to compute what information is in the new hidden state




# Long Short Term Memory Networks

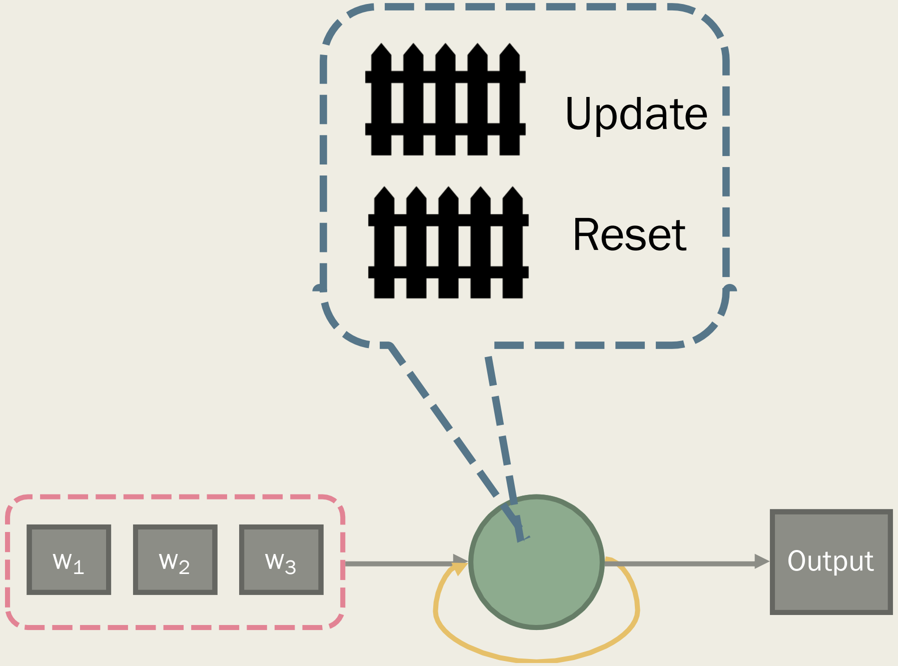
# Bidirectional LSTMs

- Basic idea: feed the input sequence to the LSTM model once from beginning to end, and once from end to beginning
- This means you have hidden states associated with both past and future information at a given timestep



# Gated Recurrent Units

- No cell state, but still has two gates
  - *Update: How much information from the past should be passed forward?*
  - *Reset: How much information from the past should be thrown out?*
- Why use GRUs instead of LSTMs?
  - *Computational efficiency: Good for scenarios in which you need to train your model quickly and don't have access to high-performance computing resources*
- Why use LSTMs instead of GRUs?
  - *Performance: LSTMs generally outperform GRUs at the same tasks*

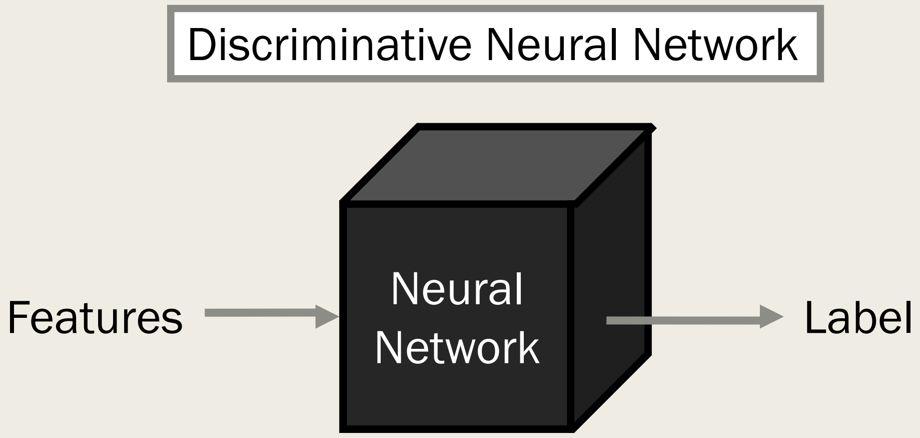


# Other Neural Network Models

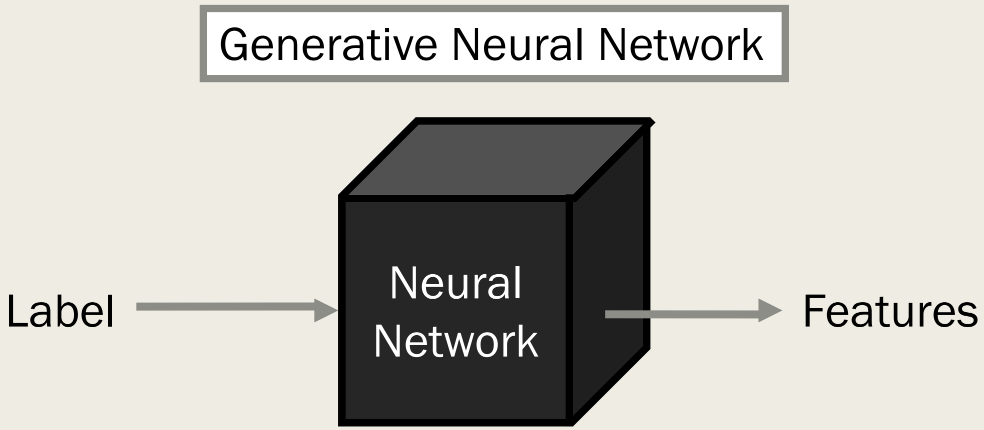
- Generative Adversarial Networks (GANs)
- Sequence to Sequence Networks (seq2seq)
- Autoencoders

# Generative Adversarial Networks

- Comprised of two neural networks that act as adversaries of one another
- Generative model rather than discriminative
  - *Generative: Learn the probability distributions of features associated with classes*
  - *Discriminative: Learn the boundary between classes*



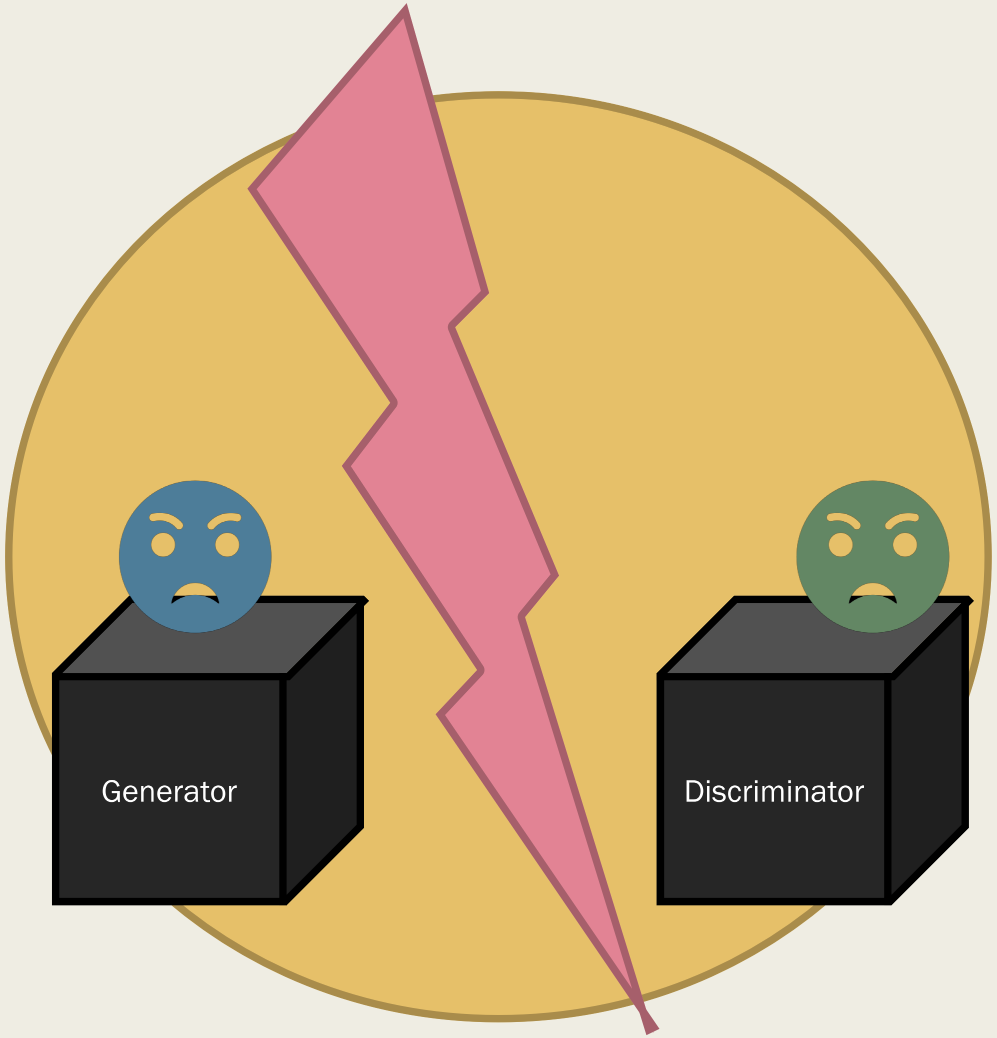

*What is the label, given what we know?*



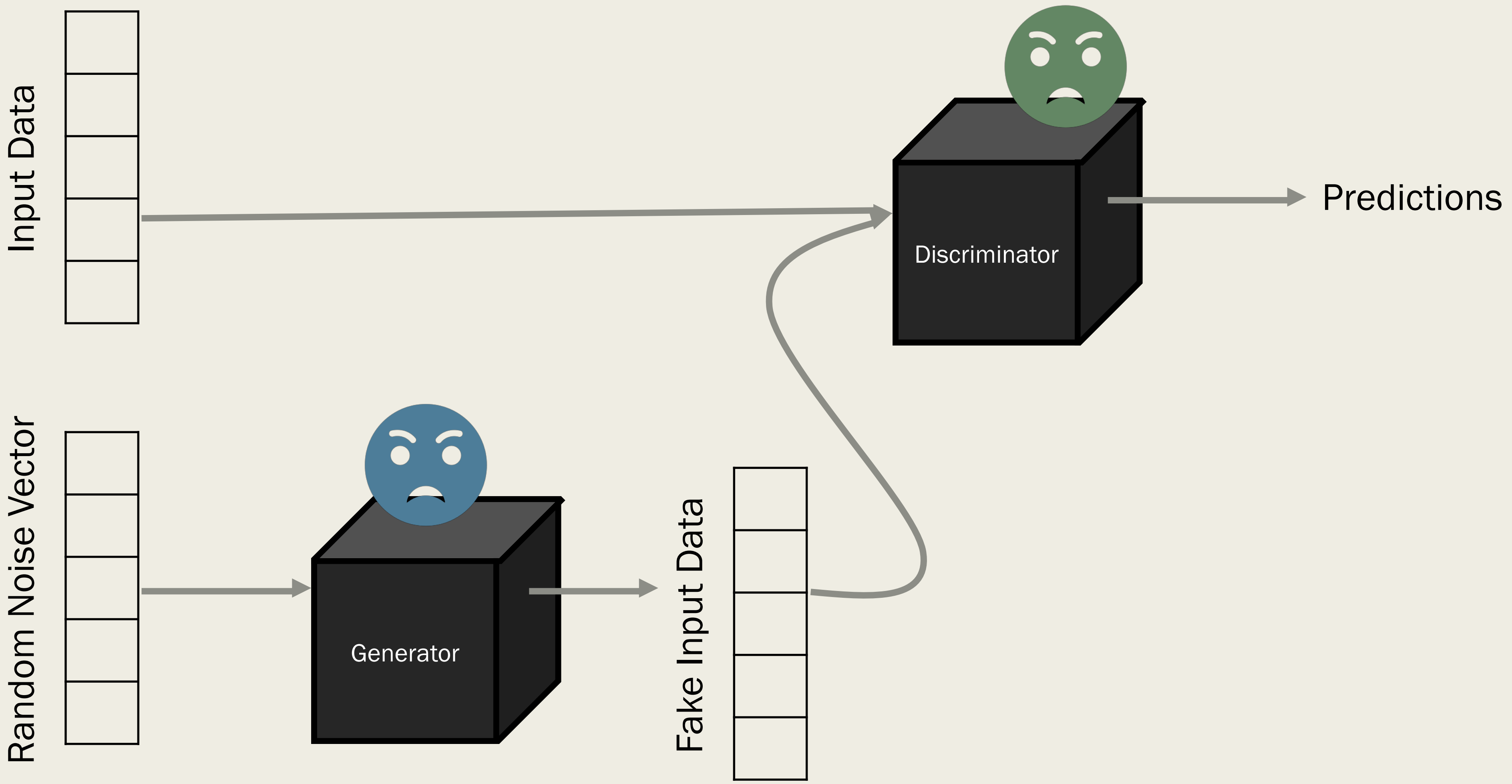

*How do we know that this is the label?*

# Generative Adversarial Networks

- Generator: “Inverse” convolutional neural network (upsamples random noise into an image) that generates new training instances
  - *Goal is to generate fake instances that are passable enough that the discriminator doesn’t detect them*
- Discriminator: Standard convolutional neural network that decides whether those instances are really part of the training dataset
  - *Goal is to discriminate between real instances and generated fake instances*



# Generative Adversarial Networks

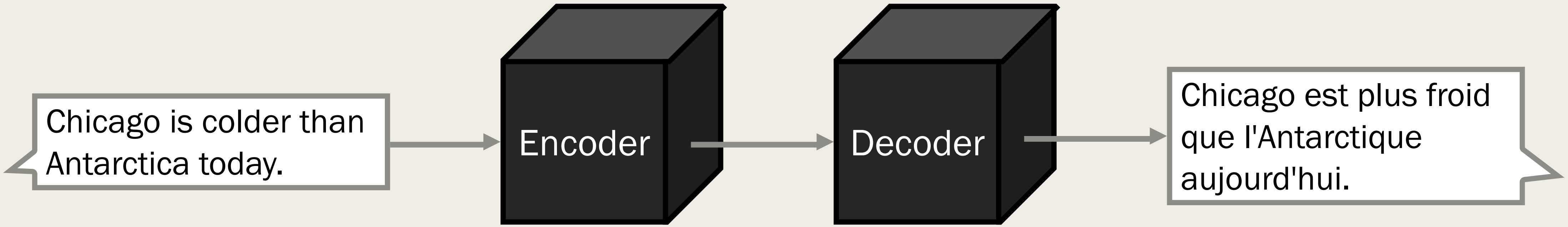


# When should GANs be used?

- Generally used in computer vision tasks
  - *Including text-to-image generation: https://github.com/zsdonghao/text-to-image*
- A few words of caution:
  - *Training can take a long time …you may want to avoid using GANs in time-sensitive projects*
  - *Tuning is also often difficult*
    - Sensitive to changes in hyperparameters
    - Generator can overpower discriminator, and vice versa

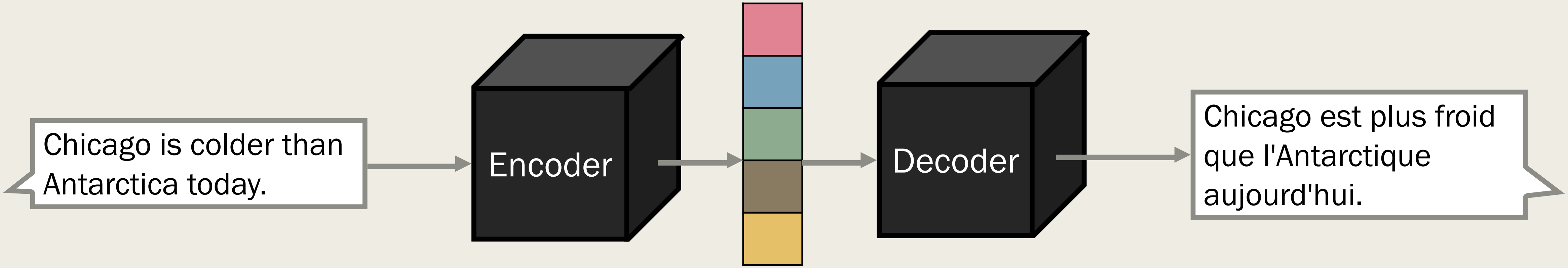
# Sequence-to-Sequence Networks

- **Encoder-decoder** models
- Accept sequential information as input, and return different sequential information as output
- Popular applications:
  - *Machine translation*
  - *Question answering*
  - *Summarization*



# What are encoders and decoders?

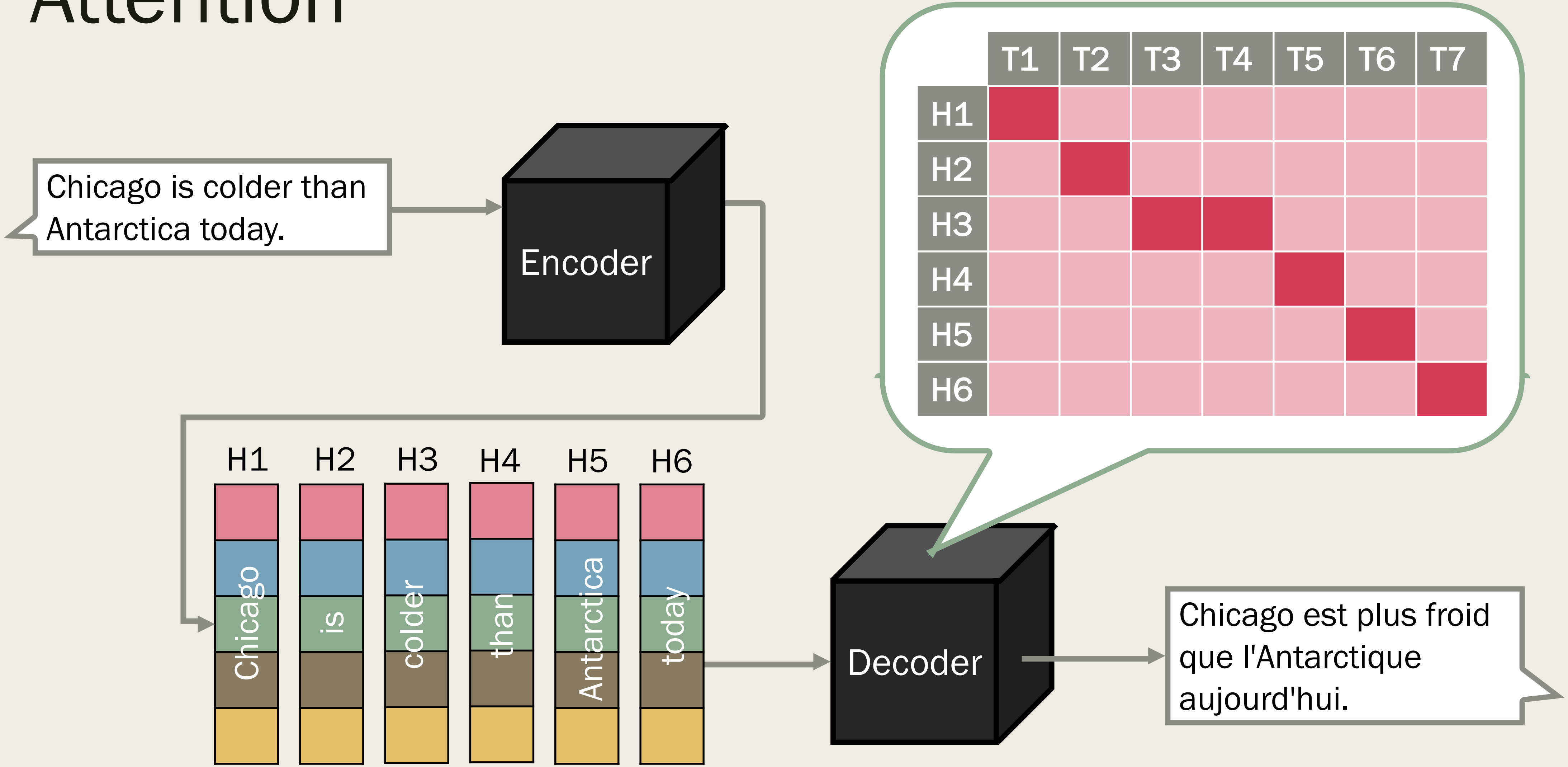
- In seq2seq models, encoders and decoders are typically LSTMs
- Encoders take sequential input and generate an encoded representation of it, often referred to as a **context**
  - *The context is equivalent to the last hidden state of the encoder network*
  - *Its features are indecipherable to us!*
- Decoders take a context as input and generate sequential (interpretable) output



# Seq2seq models often incorporate something called **attention**.

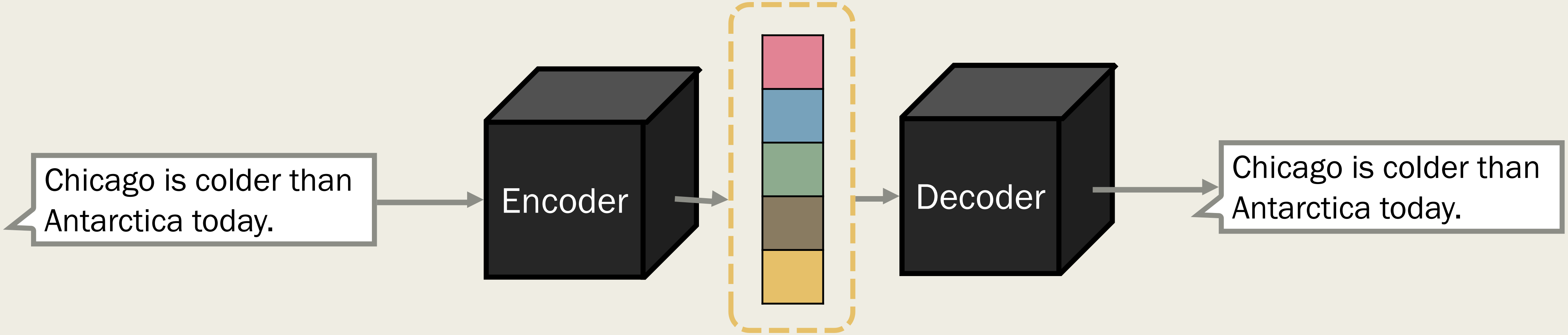
- Attention allows a decoder model to focus on (or **pay attention to**) particularly relevant parts of an input sequence
- In order to include attention in the seq2seq model, **all hidden states** must be passed to the decoder …not just the last one!
- At a given timestep, the decoder assigns a score to each hidden state in its input
- It then determines the input context for the timestep based on which hidden state(s) have the highest score

# Sequence-to-Sequence Model with Attention



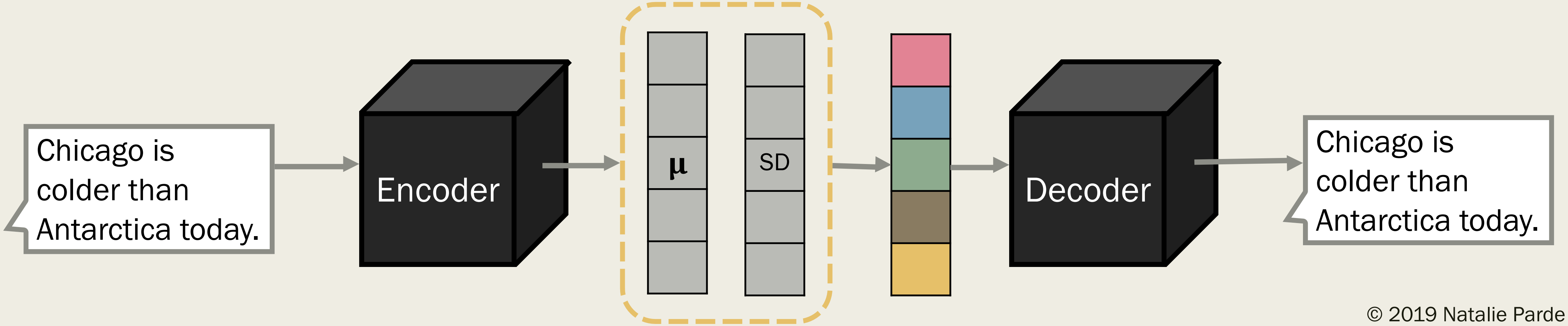
# Autoencoders

- Also **encoder-decoder** models
- The main difference:
  - *Autoencoders learn in a* ***self-supervised*** *manner*
- They do this by learning to predict their own input!
- This is a useful way to perform dimensionality reduction
  - *If a model's lower-dimensional hidden layer is capable of reconstructing its own input, it has learned how to represent that input in a lower-dimensional space*



# Variational Autoencoders

- Instead of learning a fixed representation at the **bottleneck** of the autoencoder, variational autoencoders learn a probability distribution
  - *Bottleneck = the hidden layer that is output from the encoder and input to the decoder*
- The hidden layer is replaced by two vectors:
  - *One representing its mean*
  - *One representing its standard deviation*
- The input to the decoder is then a **sample** of that probability distribution
- This change makes it possible for the variational autoencoder to act as a generative model, predicting values that did not exist in its input!



# Tool for Building Neural Networks

**TensorFlow**
- https://www.tensorflow.org/

**Keras**
- https://keras.io/

**PyTorch**
- https://pytorch.org/

**DL4J**
- https://deeplearning4j.org/

# Additional Deep Learning Resources

- Huge, curated list of deep learning books, courses, videos, tutorials, datasets, toolkits, etc.: https://github.com/ChristosChristofidis/awesome-deep-learning
- Top conference proceedings to check out:
  - *Neural Information Processing Systems (NeurIPS): https://neurips.cc/*
  - *International Conference on Machine Learning (ICML): https://icml.cc/*
  - *International Conference on Learning Representations (ICLR): https://iclr.cc/*
  - *AAAI Conference on Artificial Intelligence (AAAI): http://www.aaai.org/Conferences/conferences.php*
  - *International Joint Conferences on Artificial Intelligence (IJCAI): https://www.ijcai.org/*
- Tips for debugging deep neural networks: http://josh-tobin.com/troubleshooting-deep-neural-networks

# Wrapping up....

- Overview
- Feedforward Neural Networks
- Convolutional Neural Networks
  - *LeNet*
  - *ResNet*
- Recurrent Neural Networks
  - *LSTMs*
  - *BiLSTMs*
  - *GRUs*
- Generative Adversarial Networks
- Sequence-to-Sequence Networks
- Autoencoders
- Resources