

Deep Learning *at RichRelevance*

Nick Knowles

Data Science Research Team
nknowles@r...



Goals for the talk

- * Explore basic principles of Deep Learning
- * Share interesting research results
- * Give intuitions for how it can be used

-
- * What is Deep Learning?
 - * How is it different from other ML techniques?
 - * How does it work? (intuitively)
 - * Further steps

Outline

- * Deep Learning and Neural Networks
- * Working with sequences
- * Working with images
- * Limitations & Open Questions
- * QA

comments:

- Include benchmarks against existing deployments (see if compute is no prob)
- Inject some lesser known facts for ppl that already know this stuff

Deep Learning

Uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.

- * Implies use of Neural Network models
- * Used to be called *Connectionism* (beginning of time-2000)
- * Massively under-hyped before 2012 (Perceptrons by Marvin Minsky and Seymour Papert, 1969)
- * Now a very popular ML approach

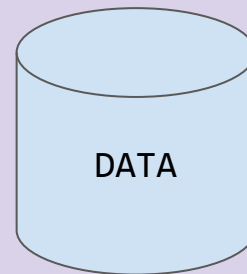
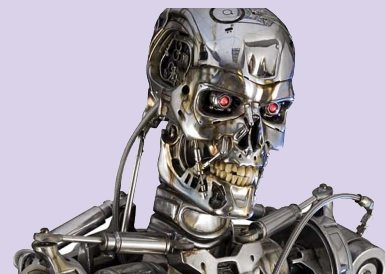
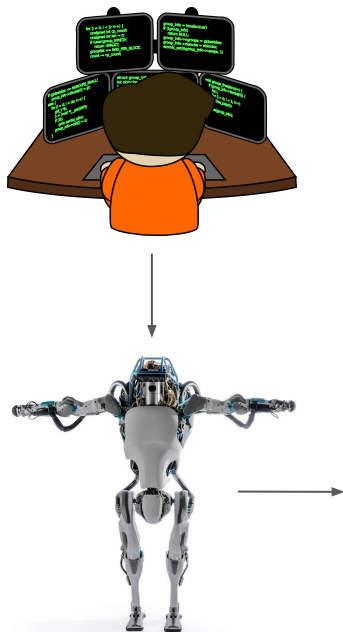
Deep Learning

Similar to connectionist (PDP/Perceptron) models popular in the '80s and '90s, but with:

- * Web-scale **data** sets
- * More **compute** (GPU, Cloud, Nvidia Tensor Cores)
- * Insanely good **software** tools (TensorFlow)
- * **Innovations** (more layers, LSTMs, attention, ect..)
- * Hype & attention: <http://paperscape.org/>

Machine Learning

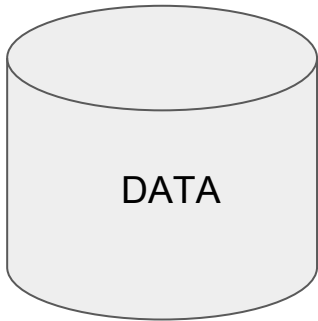
Telling vs. *Showing*



Machine Learning

Training Data

$[x]$



Supervised ML

Training Data

$[x]$

Labels/Targets

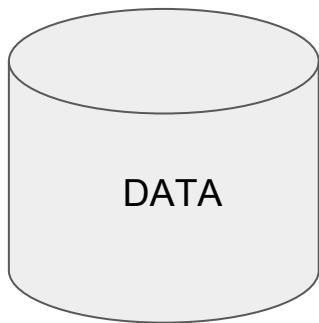
$[y]$

Supervised ML

$$f([x]) \approx [y]$$

Model

Rule-based Model

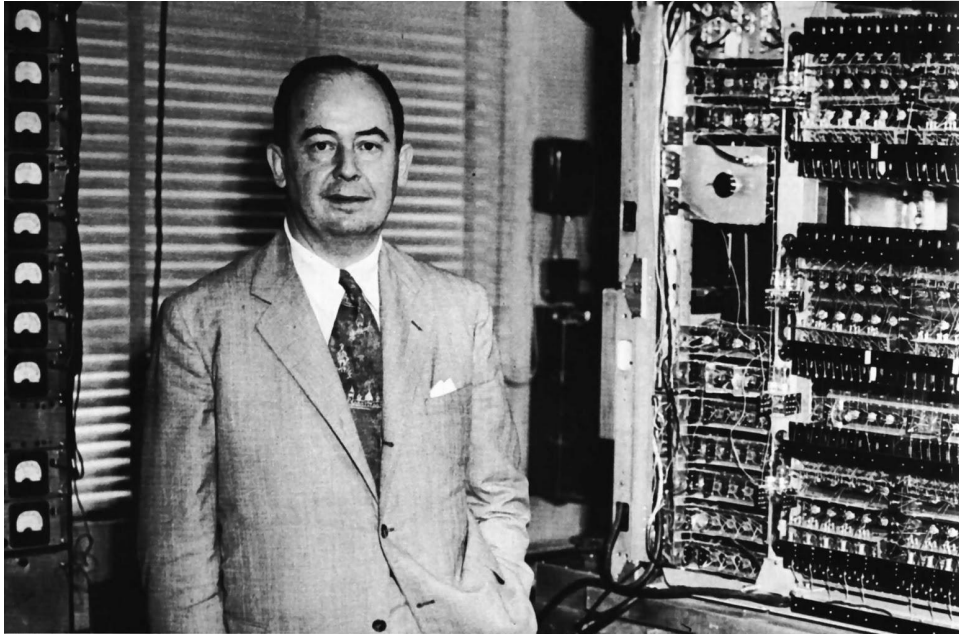


Model

```
int y = helpdesk_model("Where are the bananas?")  
// value of y procs some behavior like keyword search
```

```
int helpdesk_model(String data){  
    if (data.contains("where")){  
        return 2;  
    }  
    if (data.contains("why")){  
        return 1;  
    }  
    return 0;  
}
```

Motivation



"Truth is much too complicated to allow anything but approximations."

-John Von Neumann

Supervised ML

$$\mathbf{f}([\mathbf{x}]) \approx [\mathbf{y}]$$

Model

Regression (targets are continuous scalar values)

eg; $f(\mathbf{x}) = 85.12 \approx 86.21$

Classification: (targets are probability vectors)

eg; $f(\mathbf{x}) = [0.02, 0.01, 0.95, 0.02] \approx [0, 0, 1, 0]$

Models (Architectures) vs. Algorithms

Models	Algorithms
Recurrent Neural Networks (RNN)	Markov Chain Monte Carlo (MCMC)
Convolutional Neural Networks (CNN)	Genetic Algs (GA)
Deep Neural Networks (DNN)	Stochastic Gradient Descent (SGD)
Hidden Markov Models (HMM)	Variational Bayes
Boltzmann Machines	Conjugate Gradients
⋮	⋮

Models (Architectures) vs. Algorithms

THIS TALK

Models	Algorithms
Recurrent Neural Networks (RNN)	Markov Chain Monte Carlo (MCMC)
Convolutional Neural Networks (CNN)	Genetic Algs (GA)
Deep Neural Networks (DNN)	Stochastic Gradient Descent (SGD)
Hidden Markov Models (HMM)	Variational Bayes
Boltzmann Machines	Conjugate Gradients
⋮	⋮

High Level Takeaway

Practitioner Choices: Platforms

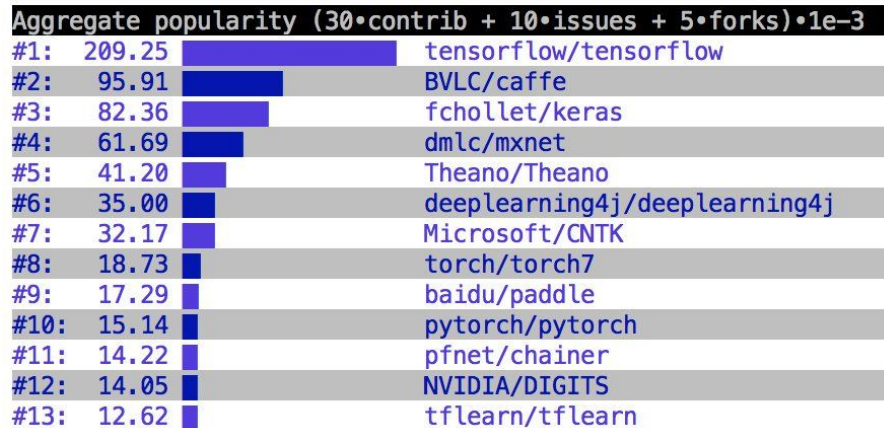
(cloud vs. GPU vs. CPU)

Practitioner Choices: Frameworks

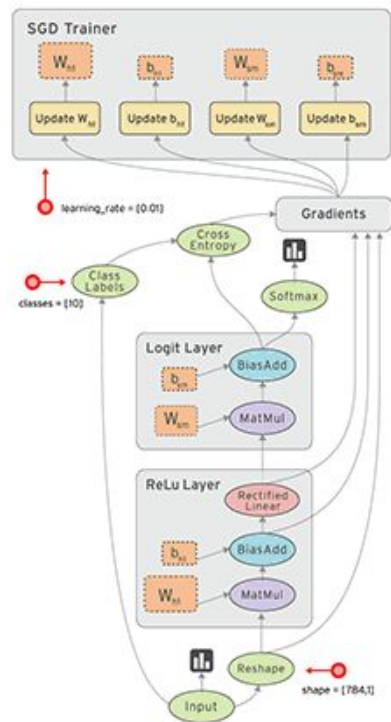


```
my_model.fit(algorithm='SGD', data=my_data)
```

Deep learning libraries: accumulated GitHub metrics
as of April 12, 2017



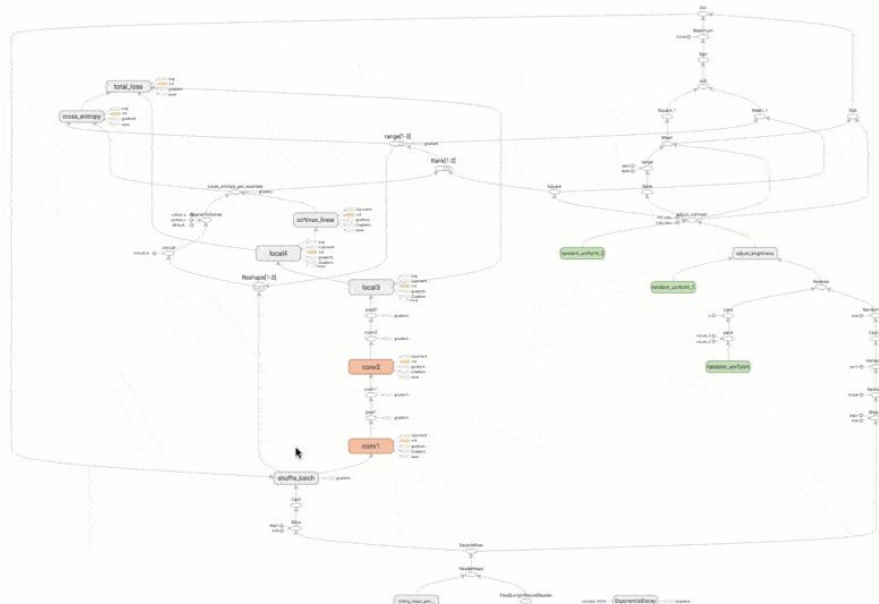
Tensorboard



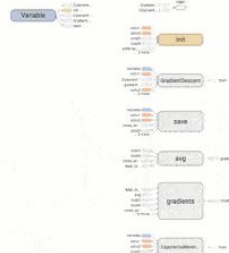
TensorBoard

Fit to screen
Run `cifar-train`
Upload Choose File
Color Structure
color: same substructure
gray: unique substructure

Main Graph



Auxiliary nodes



Graph (* = expandable)
Namespace*
OpNode
Unconnected series*
Connected series*
Constant
Summary
Dataflow edge
Control dependency edge
Reference edge

Practitioner Choices: Data

Type of data (image, text, graphs, ect..)

Features

Size

Supervised ML

$$f([x]) \approx [y]$$

How to feed x into the model?

Case study: Text data

Naive approach: cast the `char[]` to `int[]`

"cat" vs. "bat" vs. "fat"

cat -> [3, 1, 20]

bat -> [2, 1, 20]

fat -> [6, 1, 20]



Image of the vectors in 3 space

Case study: Text data

One hot encoding: give each word a unique sparse vector

"cat" vs. "bat" vs. "fat"

cat -> [1, 0, 0]

bat -> [0, 1, 0]

fat -> [0, 0, 1]



Image of the vectors in 3 space

Case study: Text data

Deep Learning: Point word vectors in directions w.r.t. their meanings

"cat" vs. "bat" vs. "fat"

cat \rightarrow $[x, y, z]$

bat \rightarrow $[x, y, z]$

fat \rightarrow $[x, y, z]$

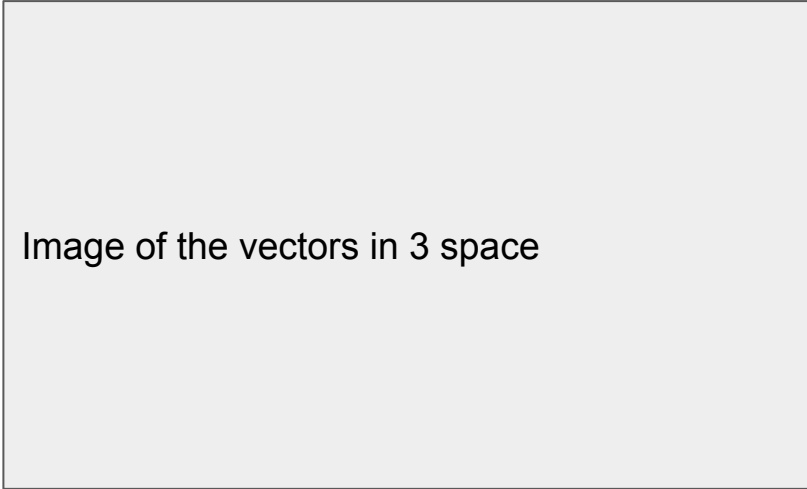


Image of the vectors in 3 space

Case study: Text data

High Level Takeaway

* How are Neural Network models different from other ML models?

Takeaway: They learn to point inputs in meaningful directions w.r.t. an objective.
-you can't *really* do **perception** without a neural network.

Supervised ML

$$f([x]) \approx [y]$$

Model --"representations of data"

'Interface for leveraging important aspects of data'

Parsimonious Models

$$PV = RT$$

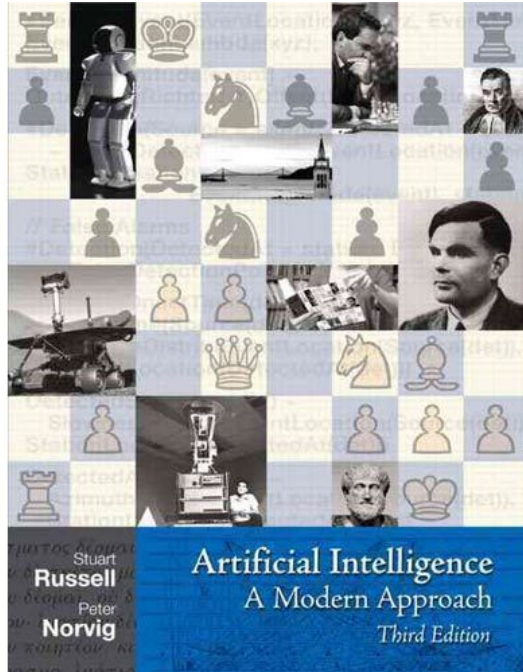
Relating pressure P , volume V , temperature T of an "ideal" gas via constant R

- * Based on physical observations of gas molecules and their behaviors
- * Not exactly true for any real gas
- * But provides good approximations that are useful

"Essentially, all models are wrong, but some are useful."

-George E.P. Box

Deep Learning Today



"So I'm re-writing the AI textbook right ... and we have a real problem because we have a Computer Vision chapter, and then Ian Goodfellow is writing the Deep Learning chapter and I suspect the Vision chapter is just gonna say: see chapter 19, and then we have a section on Speech Recognition that's just gonna say see chapter 19 ... and this worries me..." -Stuart Russell

Neural Nets: Tunable Functions

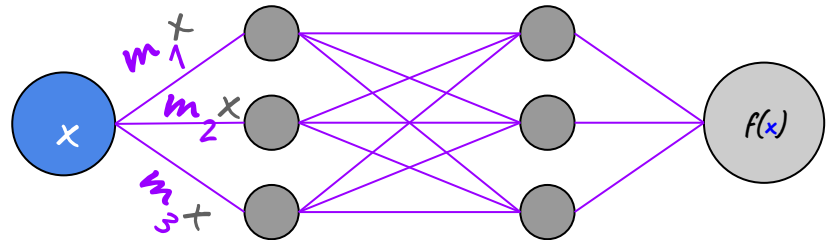
$$f(x) = mx + b \approx y$$

Model

Neural Nets: Tunable Functions (with many parameters)

$$f(x) = mx + b$$

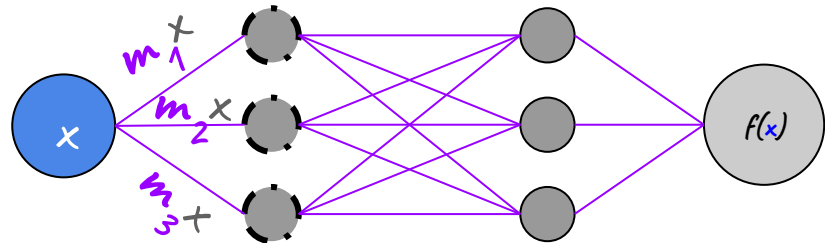
m: nxm Matrix []
b: vector [] or scalar



Neural Nets: Tunable **Nonlinear** Functions

$$f(x) = S(mx + b)$$

$$S = \max(0, k)$$

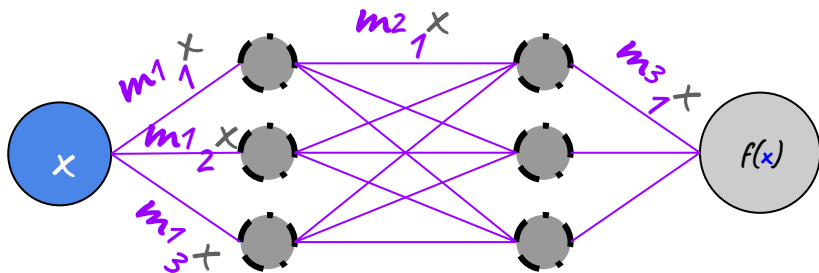


Arbitrarily many parameters

tuned via optimization

$$f(x) = S(m_2(S(m_1x + b_1)) + b_2)$$

$$S = \max(0, k)$$

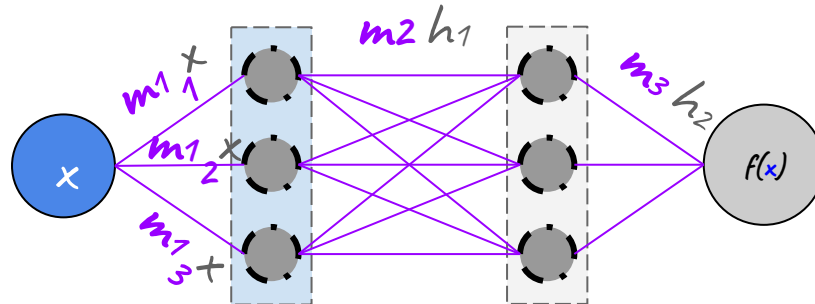


Usually described in terms of **Layers**

$$f(x) = S(m_2 (S(m_1 x + b_1)) + b_2)$$

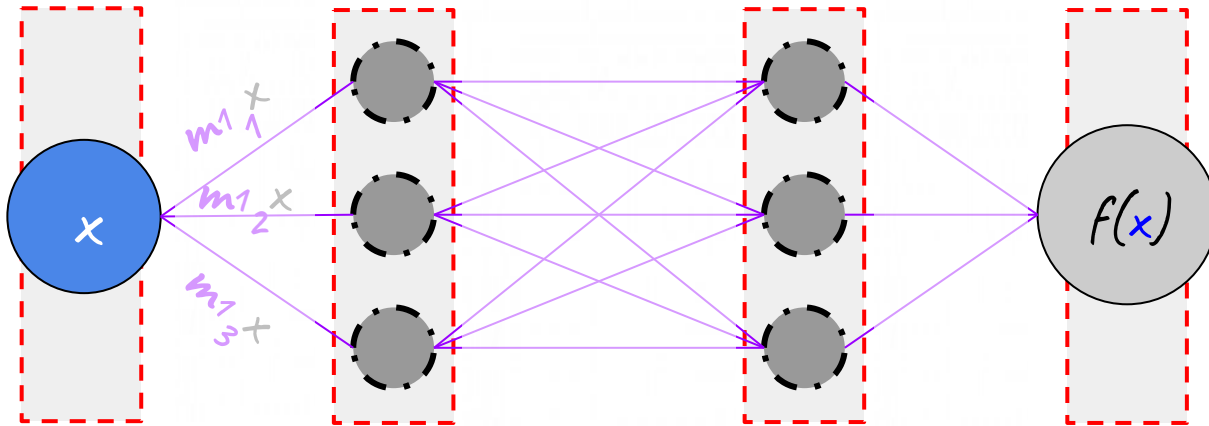
Diagram illustrating the layers of the function $f(x)$:

- Layer 1** (inner blue box): $S(m_1 x + b_1) = h_1$
- Layer 2** (outer grey box): $S(m_2 h_1 + b_2) = h_2$



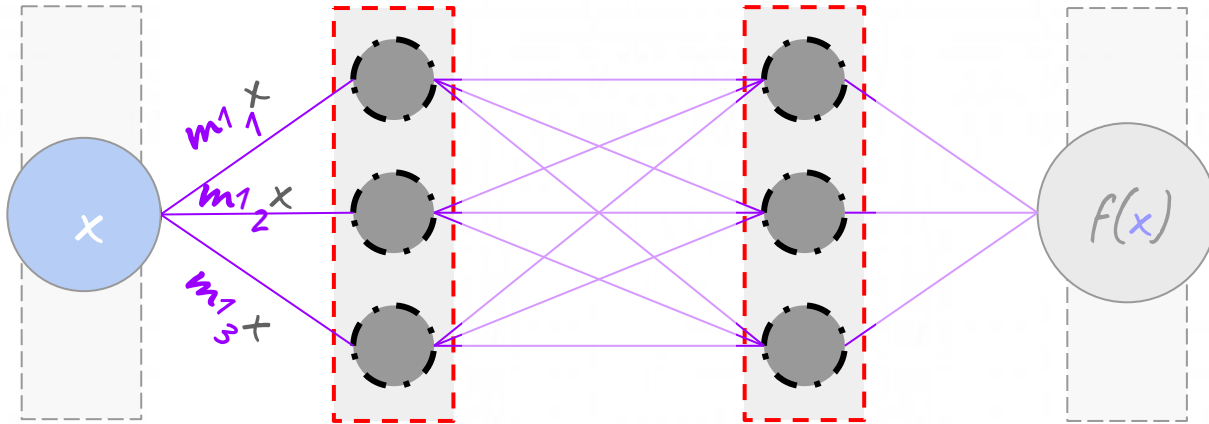
Detour: Neural Nets Vocab

Layers



Neural Nets Vocab

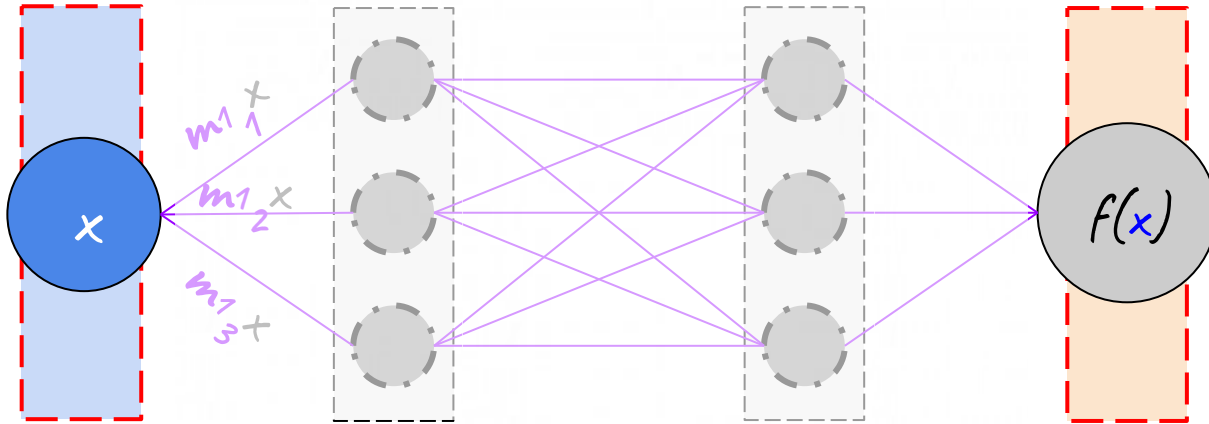
Hidden Layers



Neural Nets Vocab

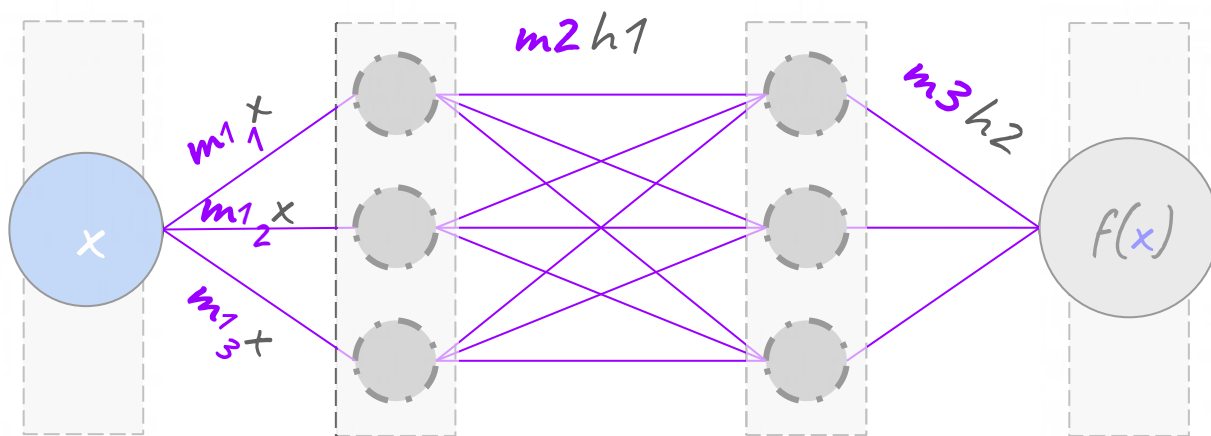
Input Layer

Output (Logit) Layer



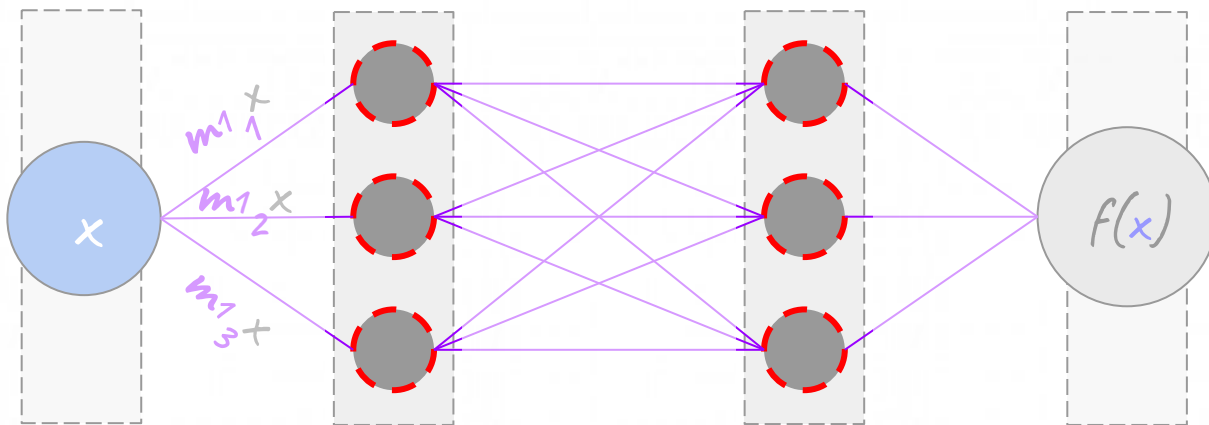
Neural Nets Vocab

Weights

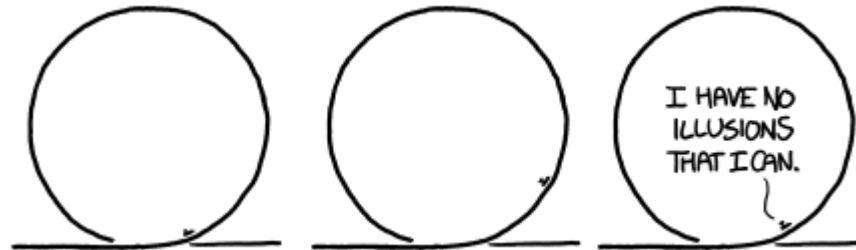
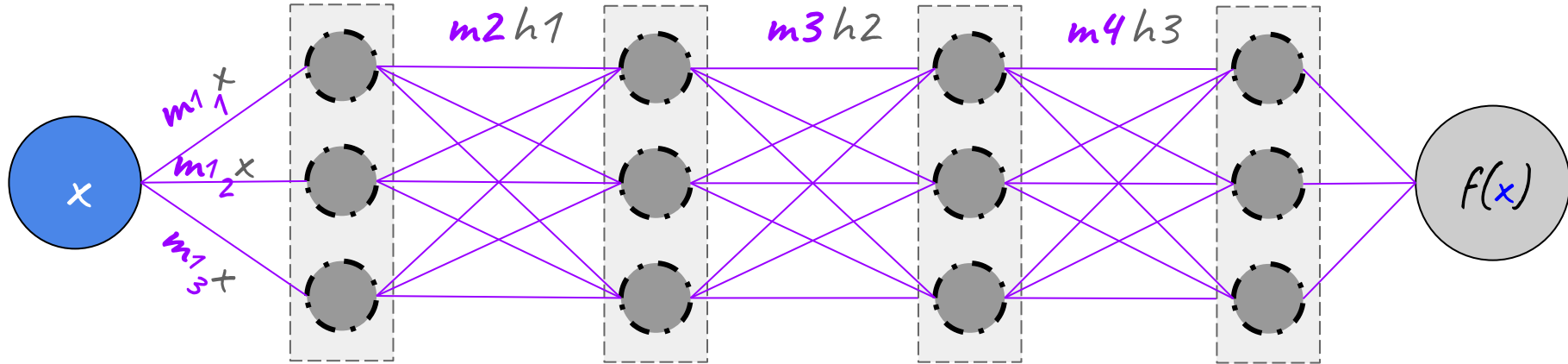


Neural Nets Vocab

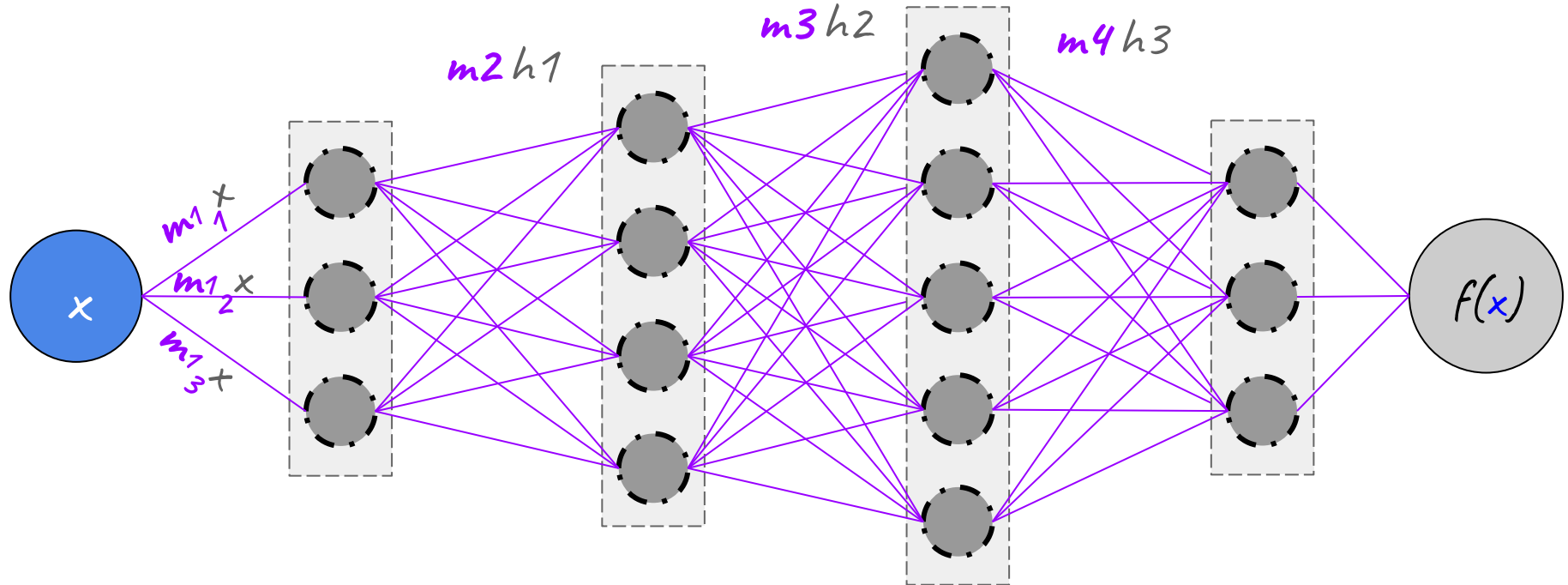
Hidden Units



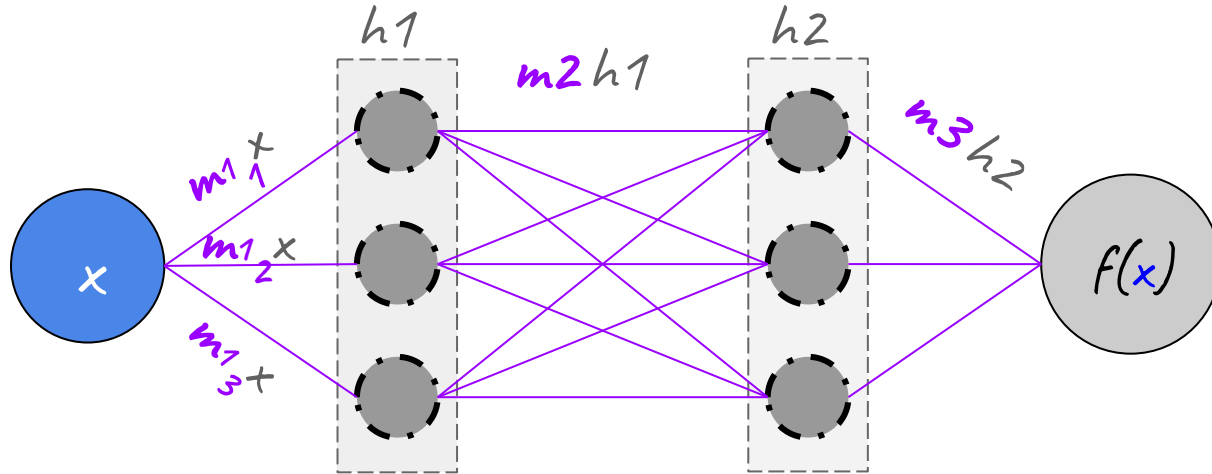
Deep Neural Networks (DNN)



Deep Neural Networks (DNN)

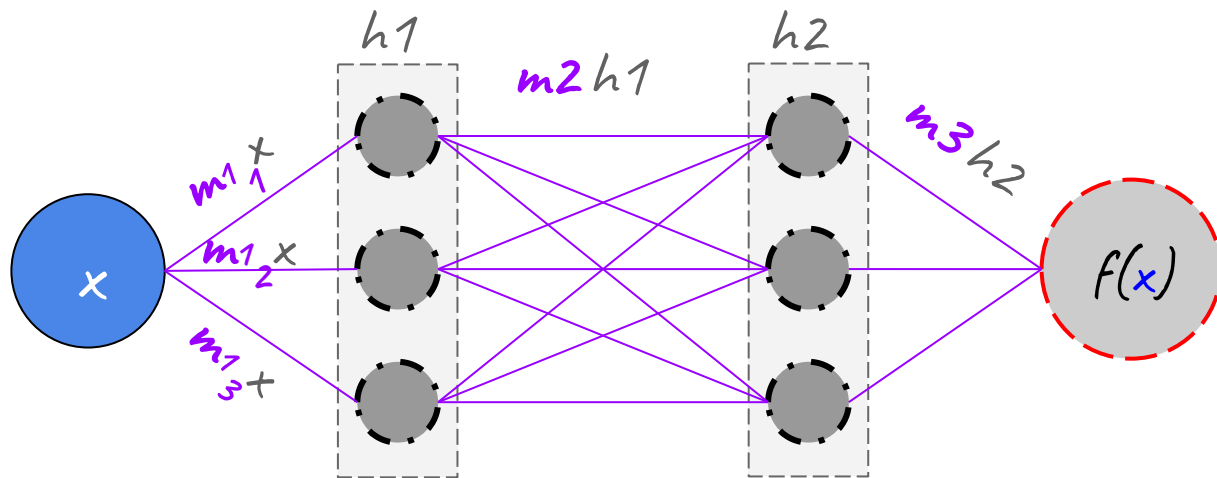


Deep Neural Networks (DNN)



$$f(x) = m3(H2(H1(x))) + b3 \approx y$$

Output Activation Functions



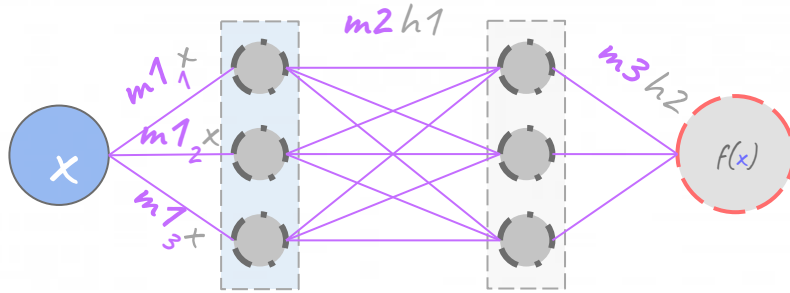
$$f(x) = a(m3H2 + b3) \approx y$$

$$a(k) = k$$

$$a(k) = 1/(1+e^{-k})$$

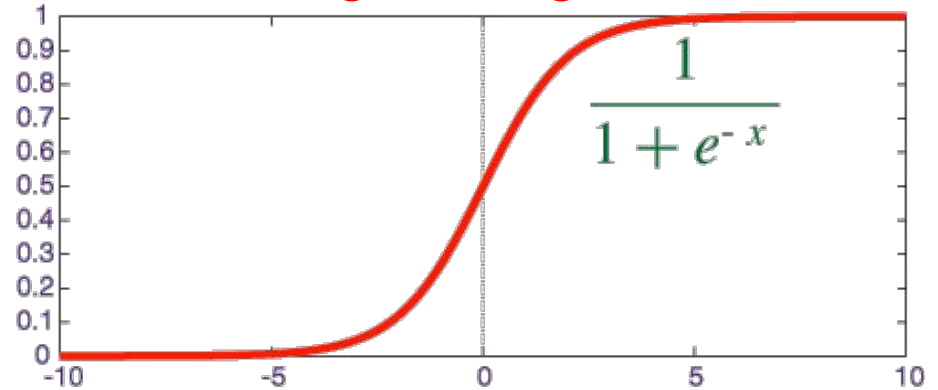
$$a([k]) = \text{softmax}([k])$$

Output Activation Functions



$$f(x) = a(m3H2 + b3) \approx y$$

Logistic sigmoid

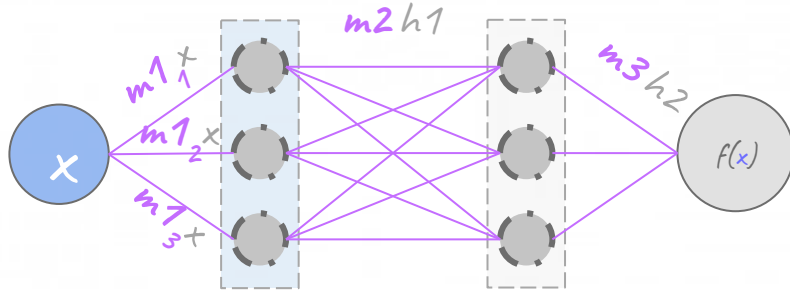


$$a(k) = k$$

$$a(k) = 1/(1+e^{-k})$$

$$a([k]) = \text{softmax}([k])$$

Output Activation Functions



563	0.8
-321	0.01
322	0.14
111	0.05

$$f(x) = a(m3(H2(H1(x))) + b3) \approx y$$

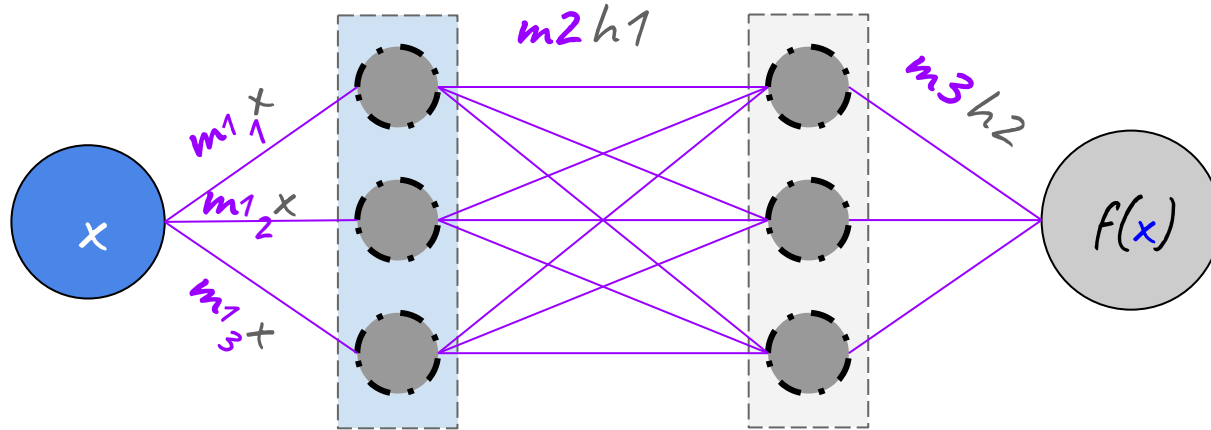
$$a(k) = k$$

$$a(k) = 1/(1+e^{-k})$$

Softmax

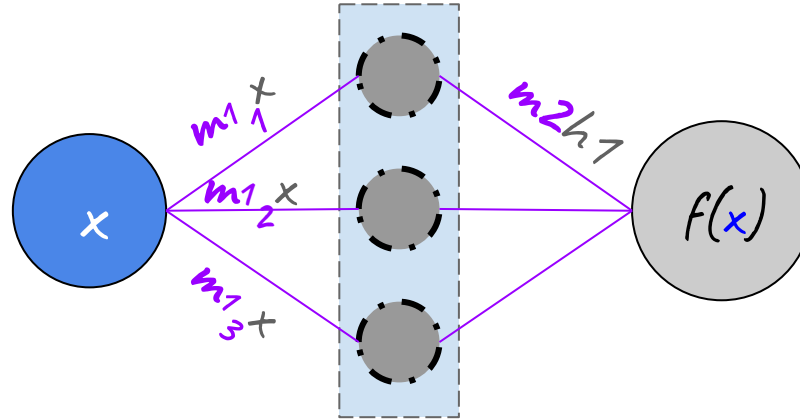
$$a([k]) = \text{softmax}([k])$$

Neural Networks



Deep Neural Networks (DNN) -has become very general
Fully Connected Networks (FCN) -new
Artificial Neural Networks (ANN) -old

Neural Networks

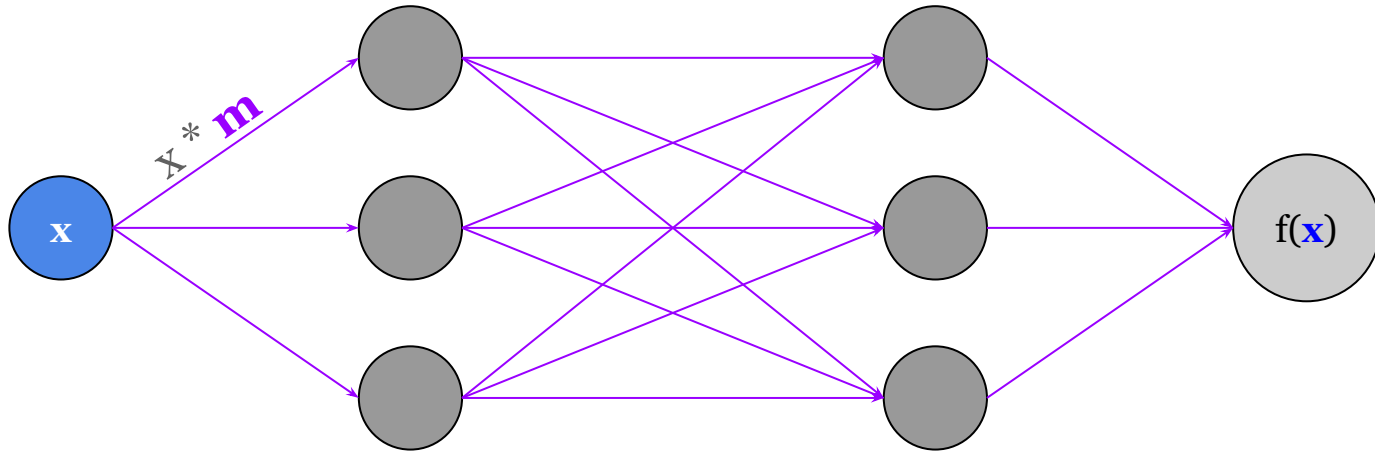


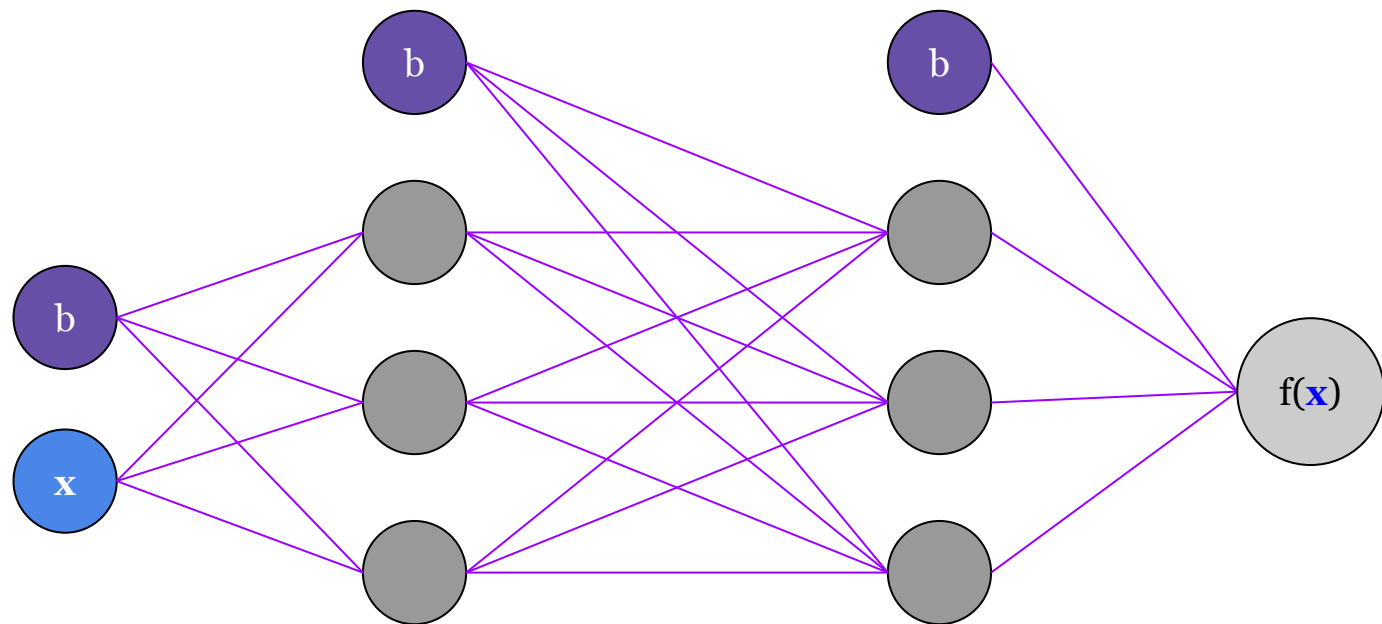
Multilayer Perceptron (MLP)

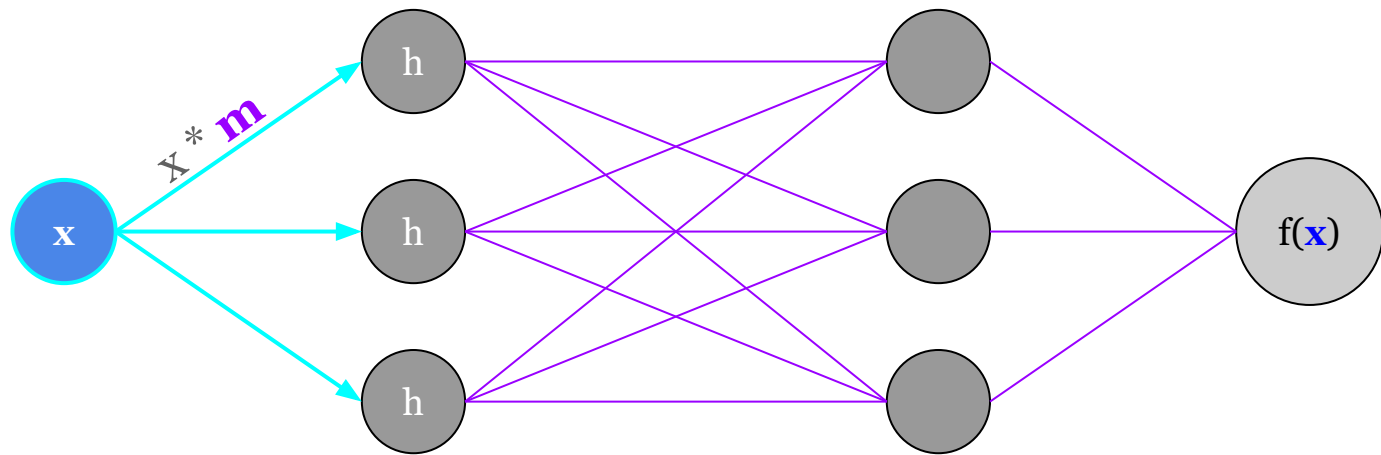
Deep Neural Network (DNN)

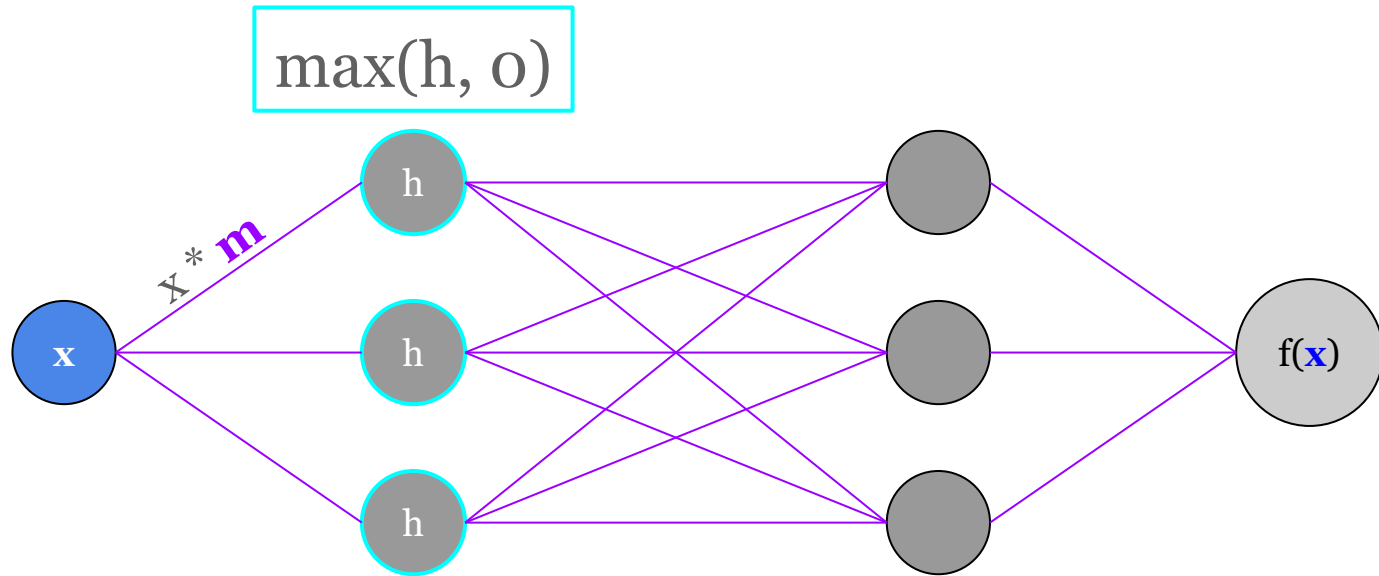
Edge = weighted connection (learned)

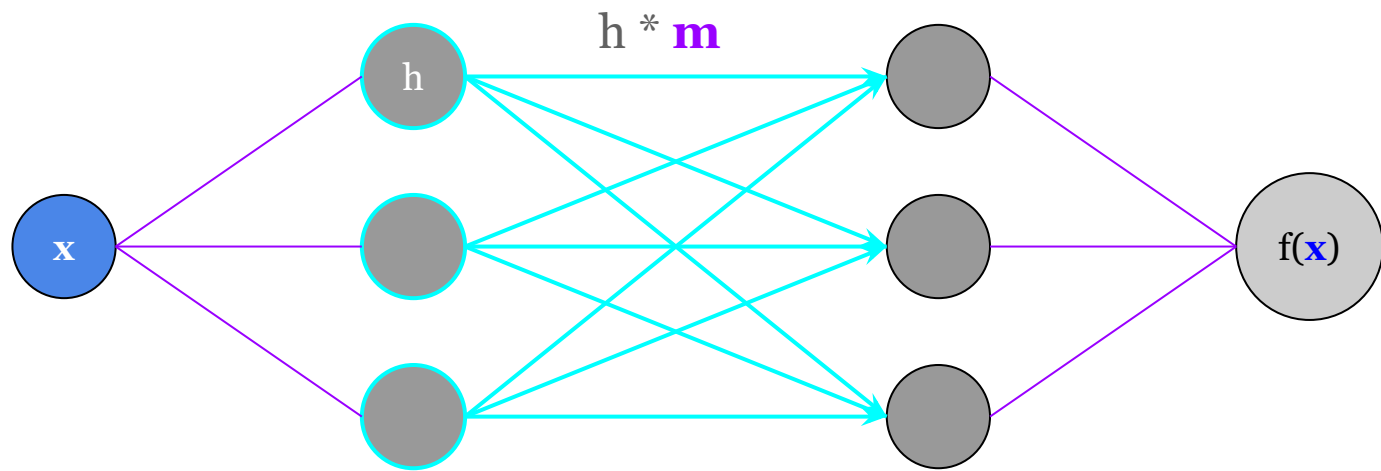
Nodes = inputs, hidden states

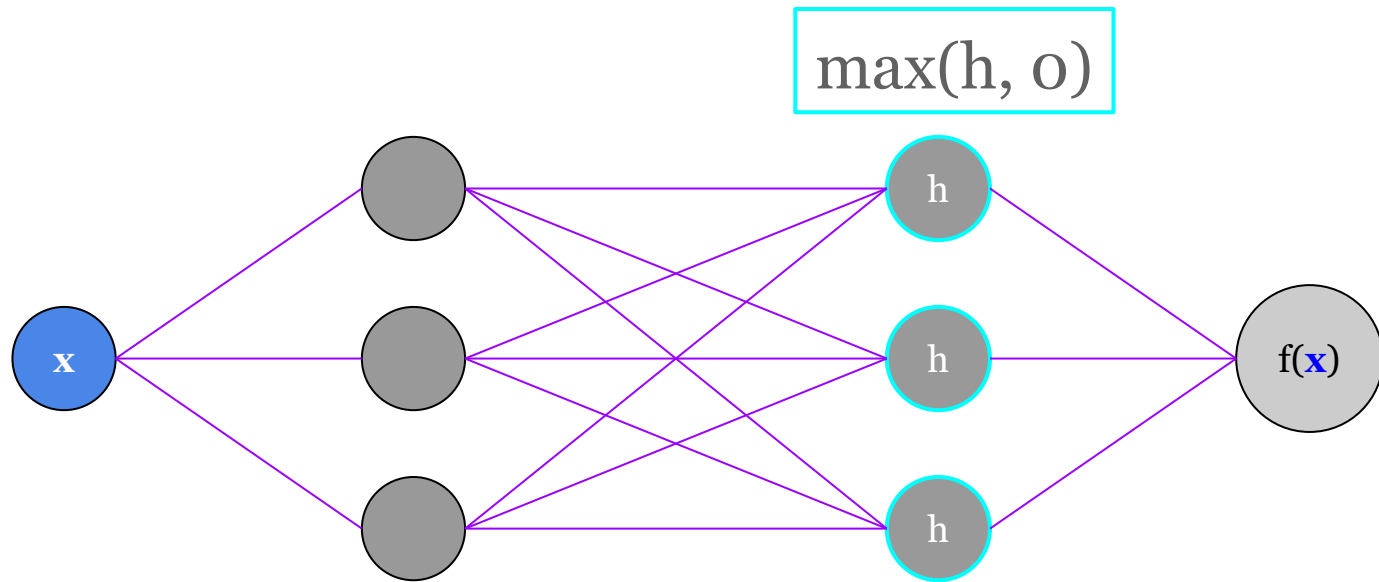


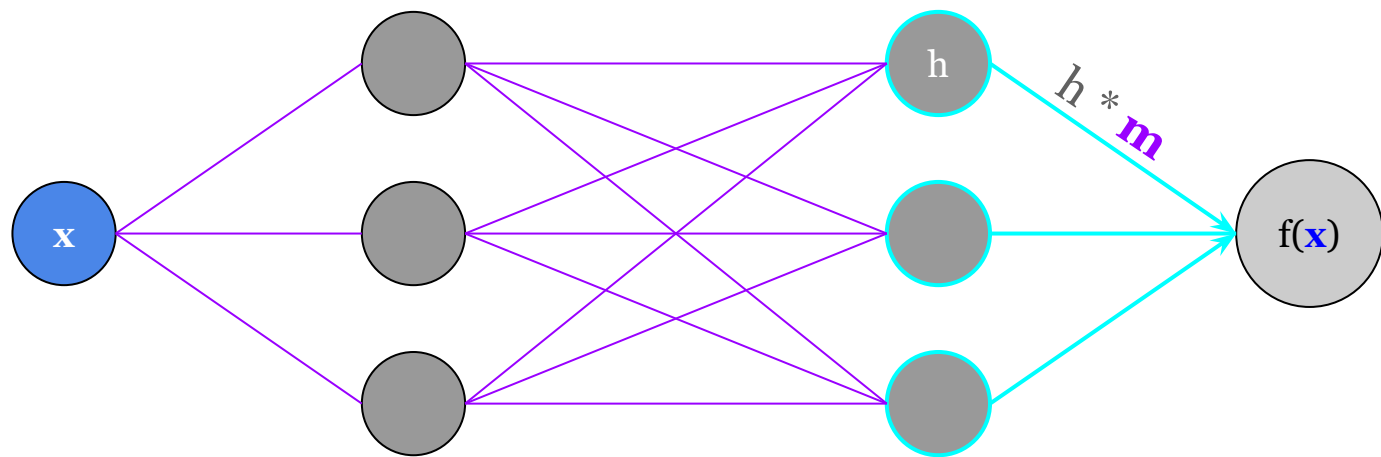




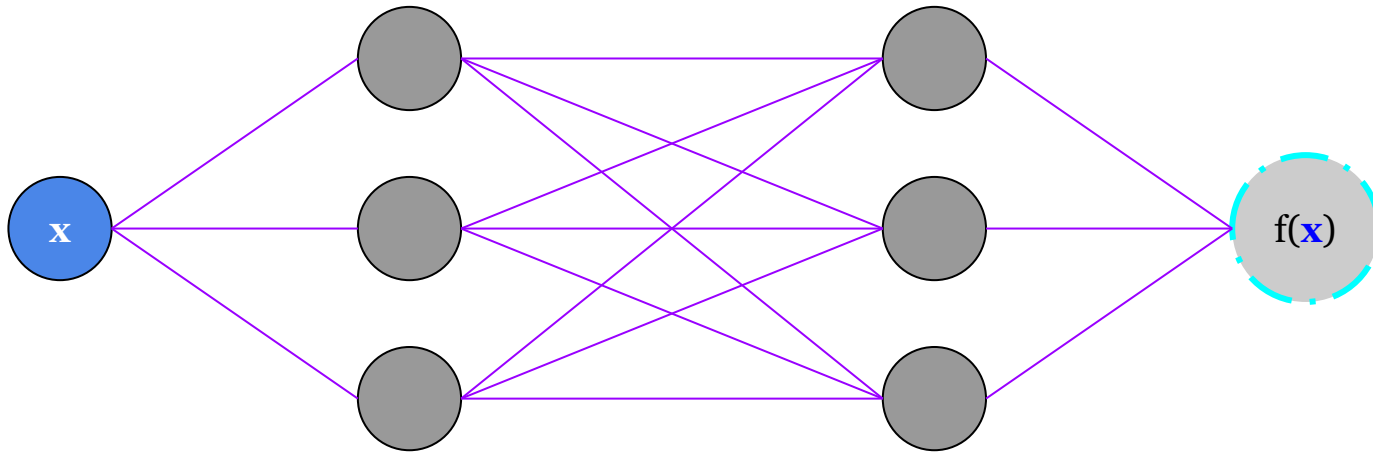


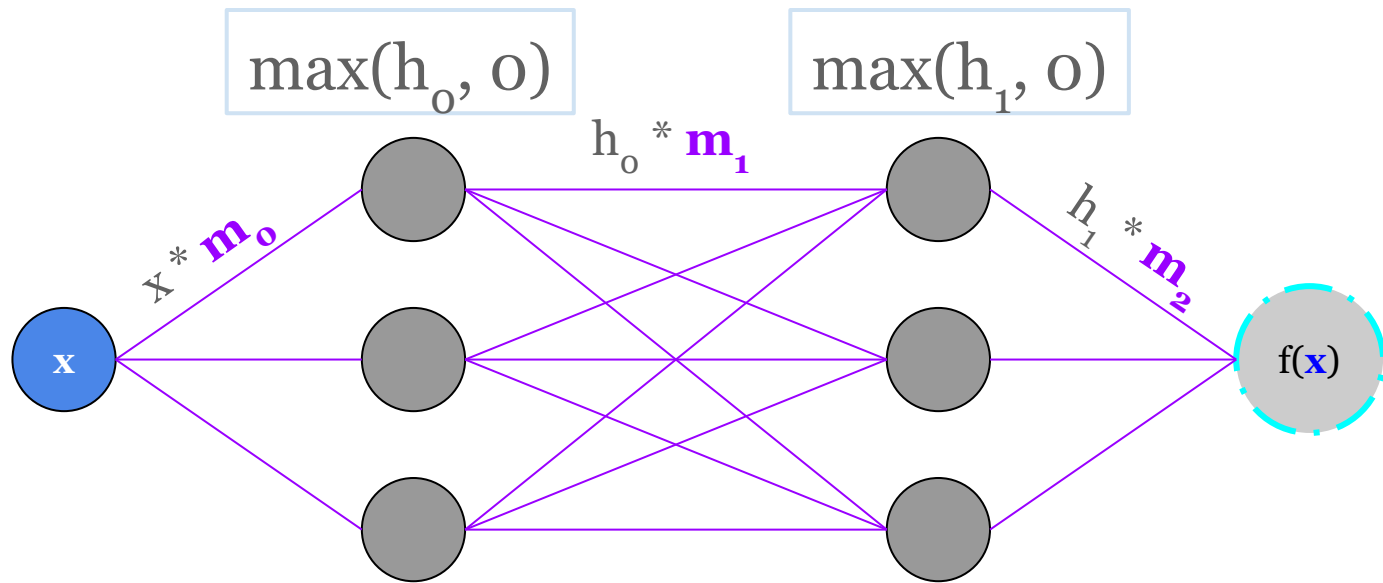






If doing regression: Don't do non-linearity for last layer.
If doing classification: sigmoid (binary) or softmax (multi-class)

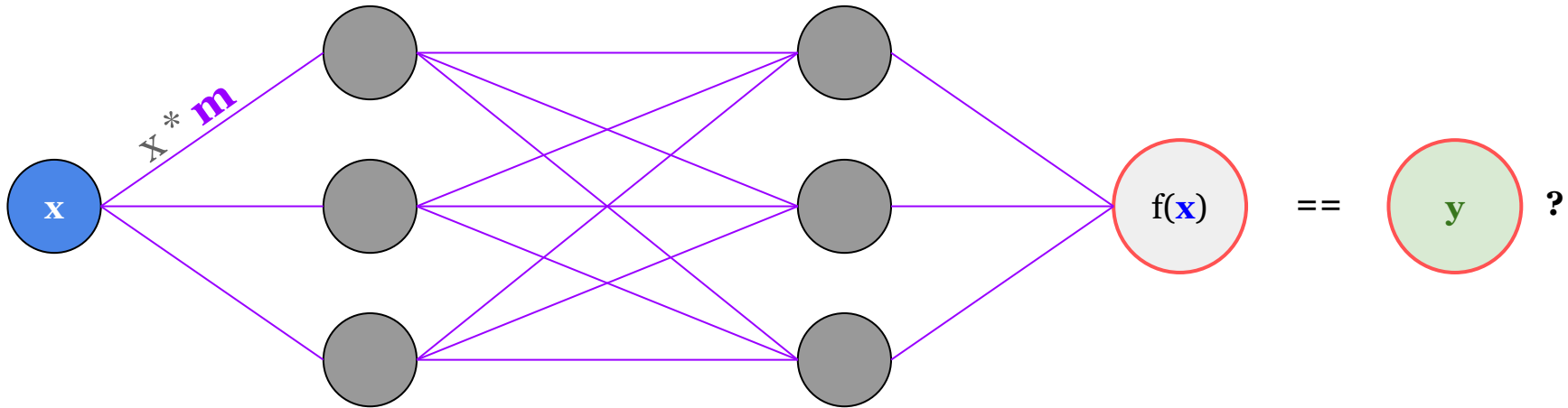




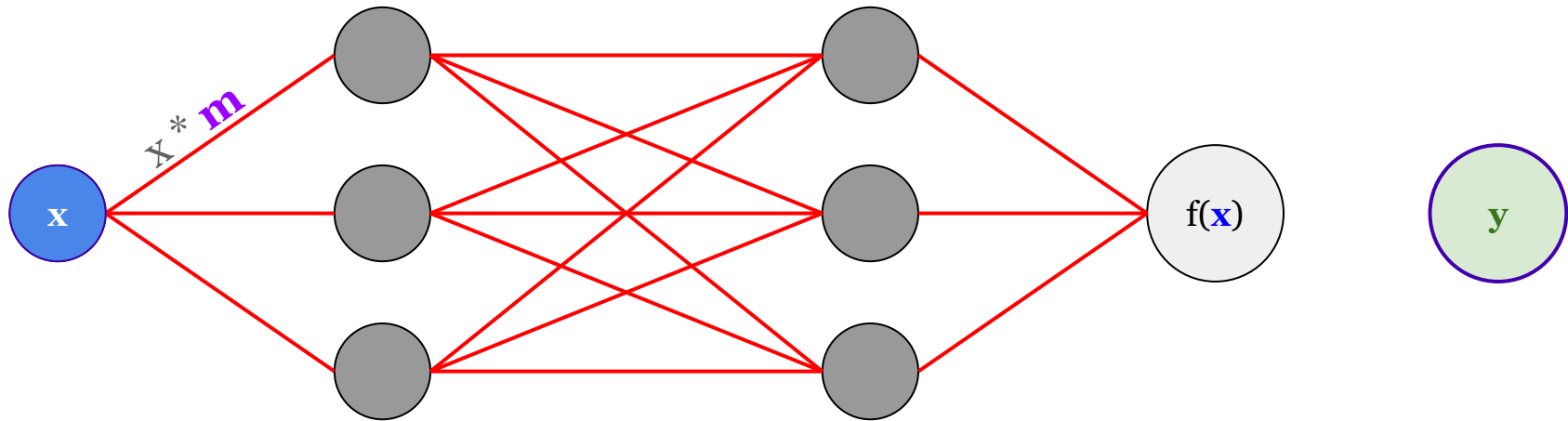
Loss Function

MSE if doing regression

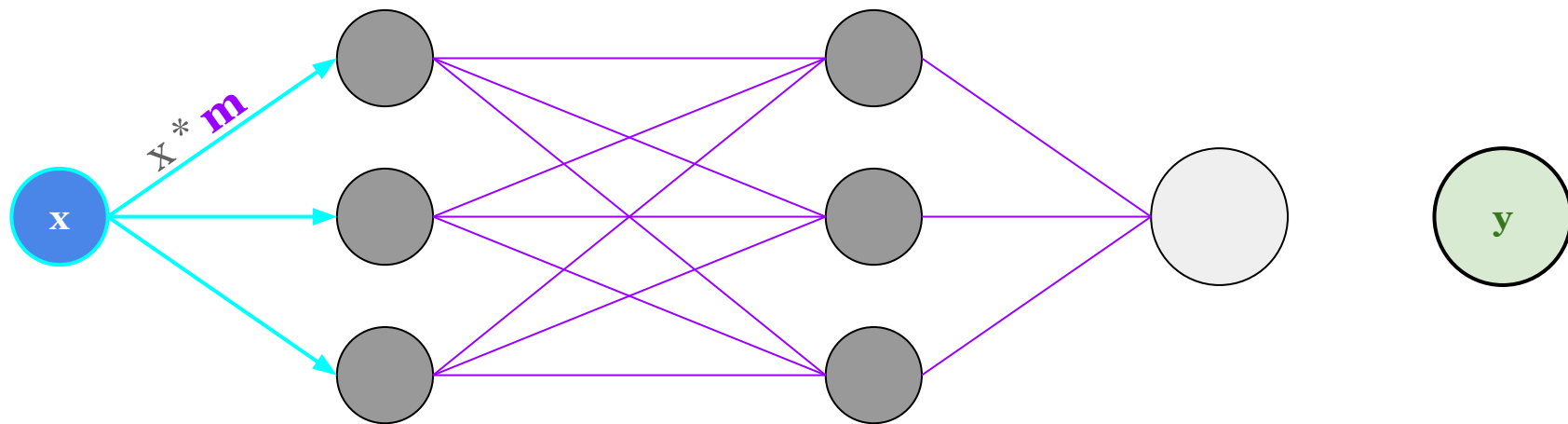
Cross entropy if classification: $-\log(P(y_{\text{true}}))$



Update the weights to better fit this $x \rightarrow y$ relationship
(via an optimization algorithm that minimizes selected loss)



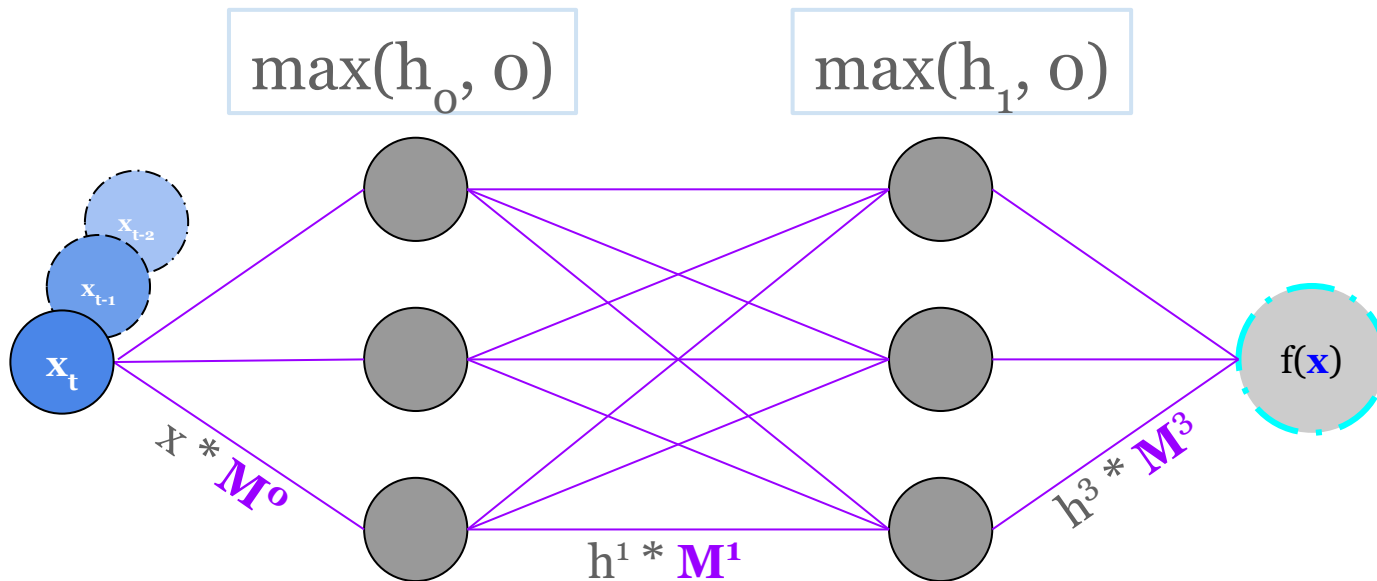
Sample a new (\mathbf{x} , \mathbf{y}) pair and repeat



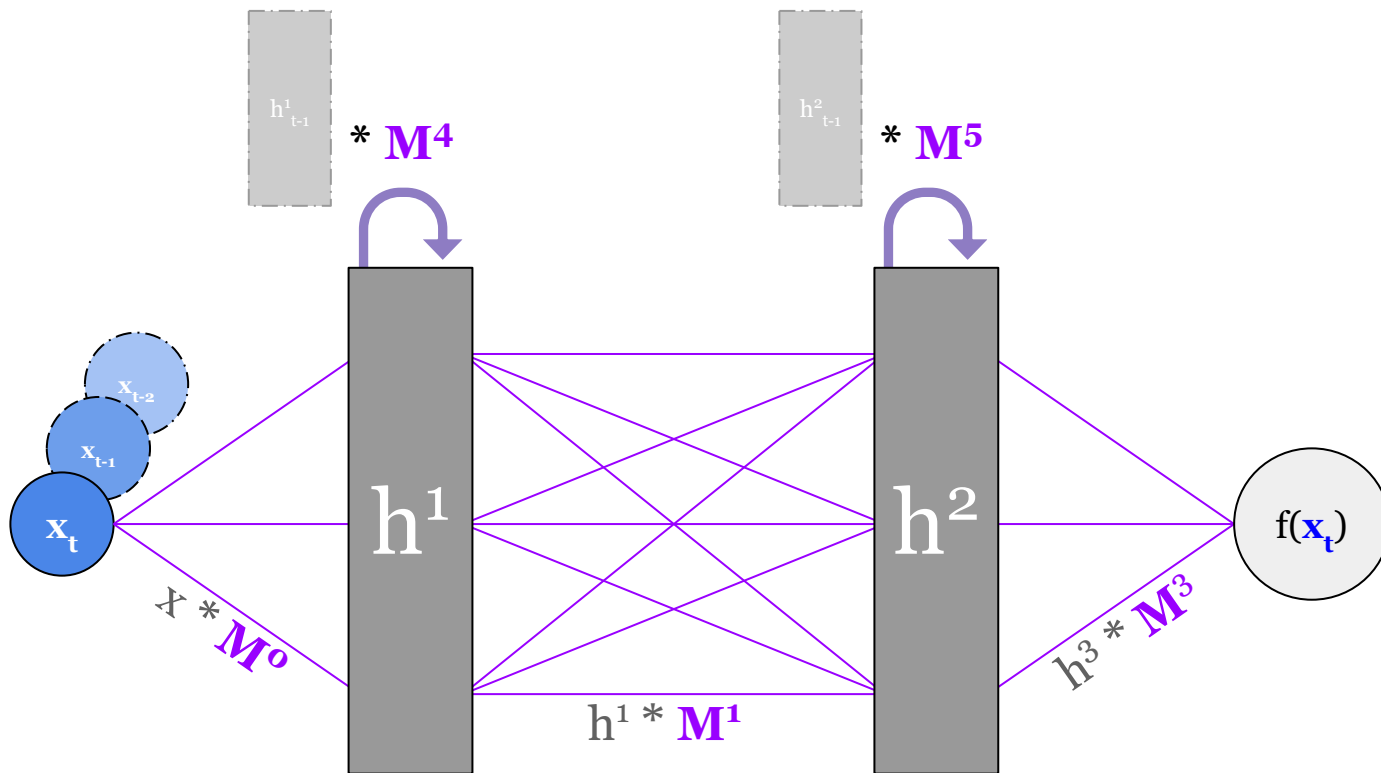
Questions?

Short FCN Demo/results

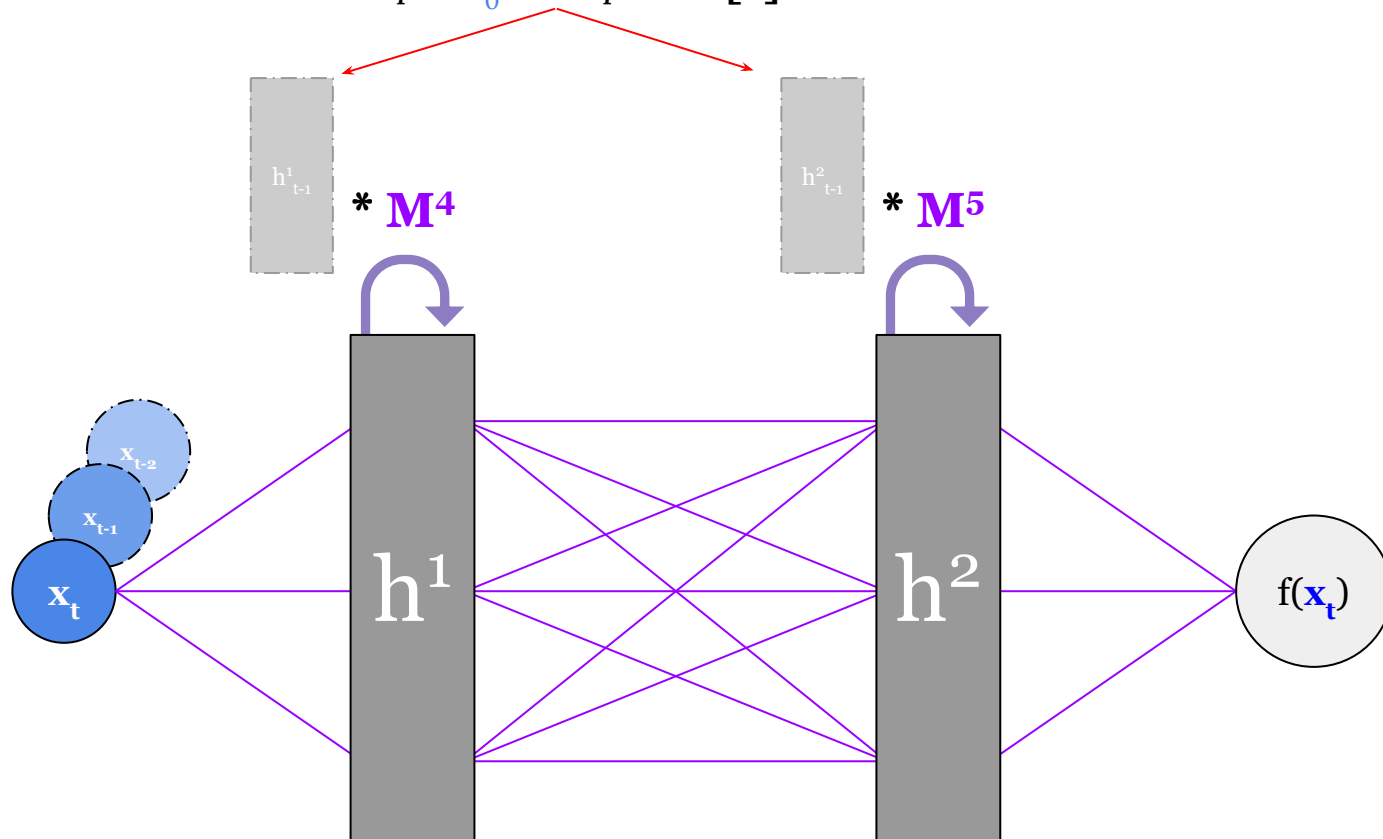
What about sequences?



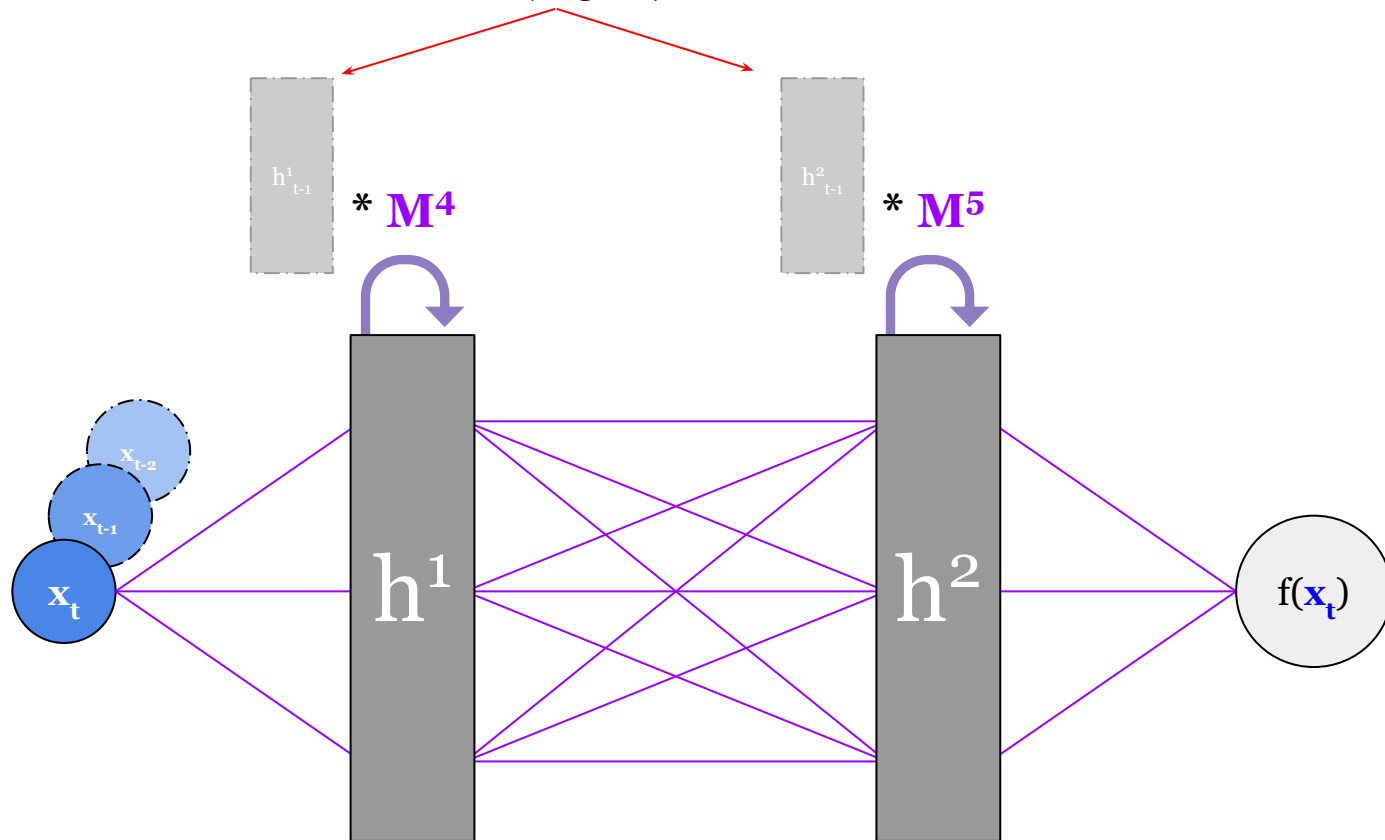
Recurrent Neural Networks

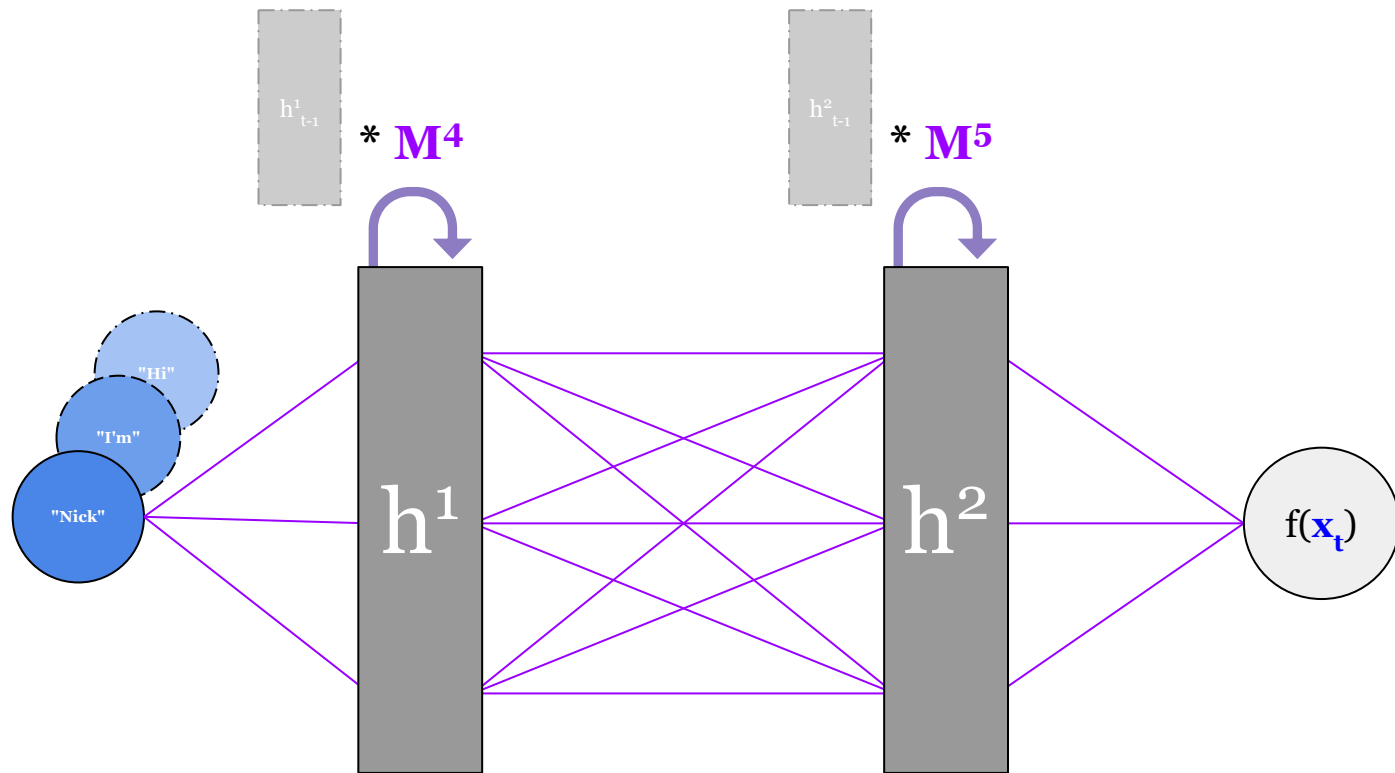


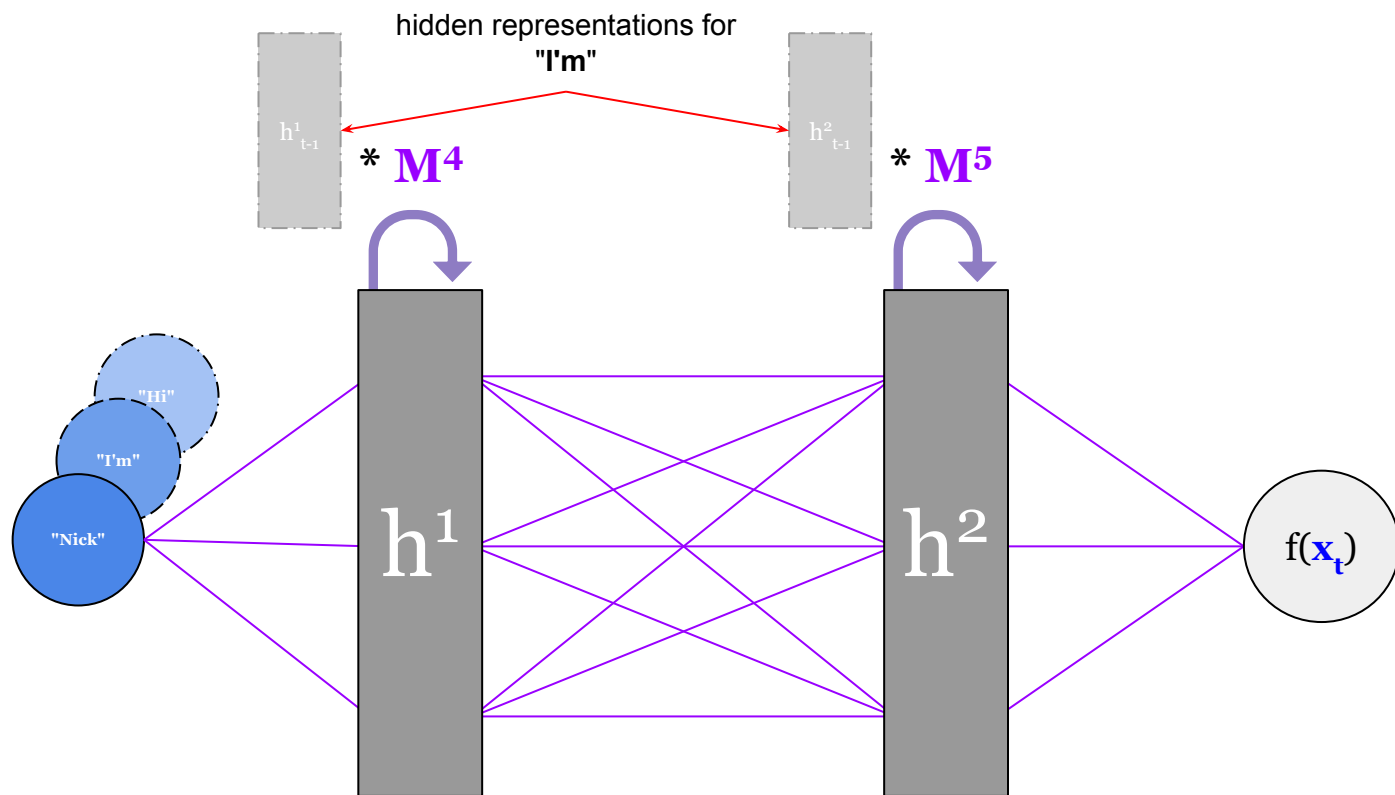
Set all \mathbf{h} to $[0]$ for first
input \mathbf{x}_0 in sequence $[\mathbf{X}]$



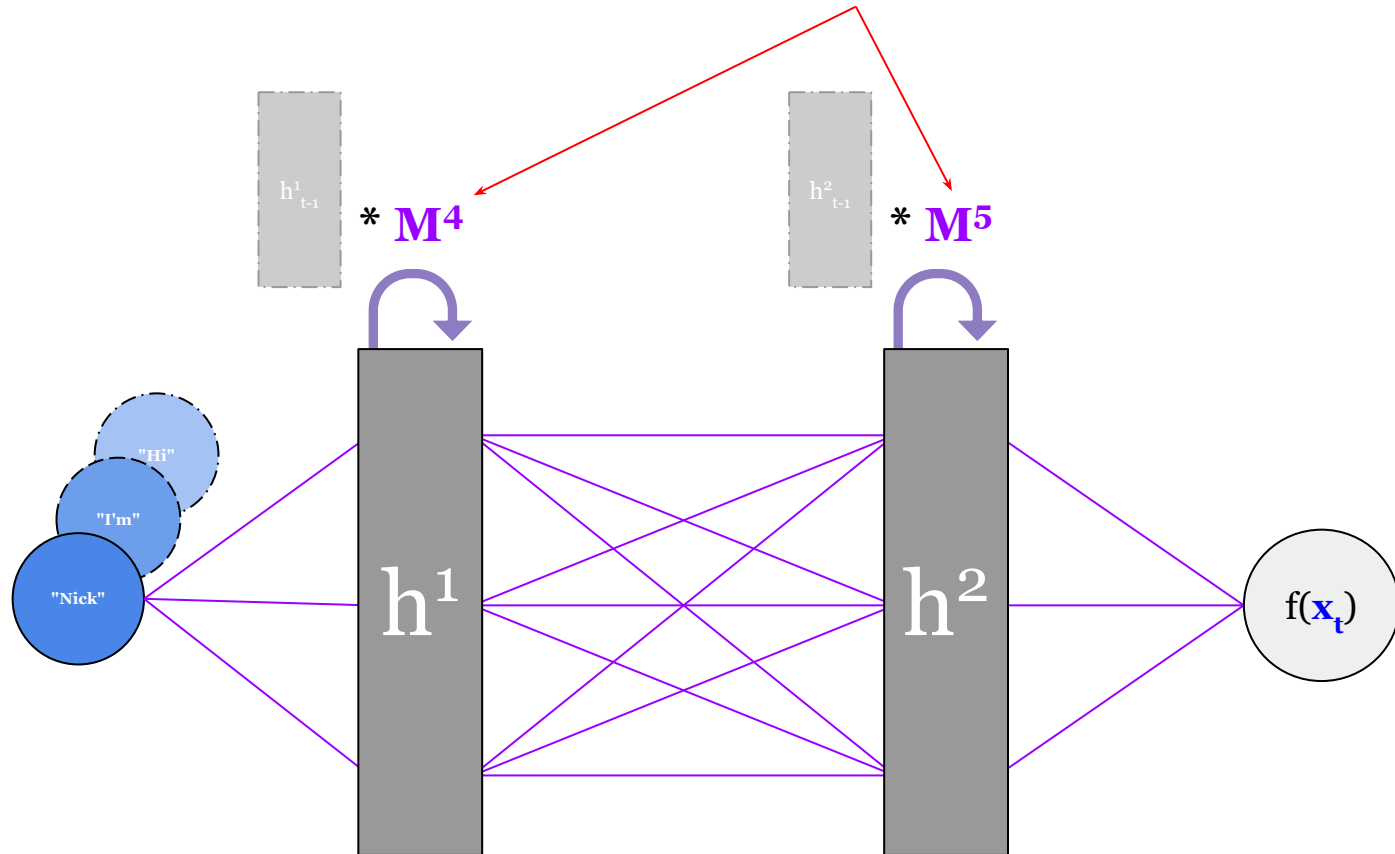
Or set to last \mathbf{h} seen for
this user/agent/ect..

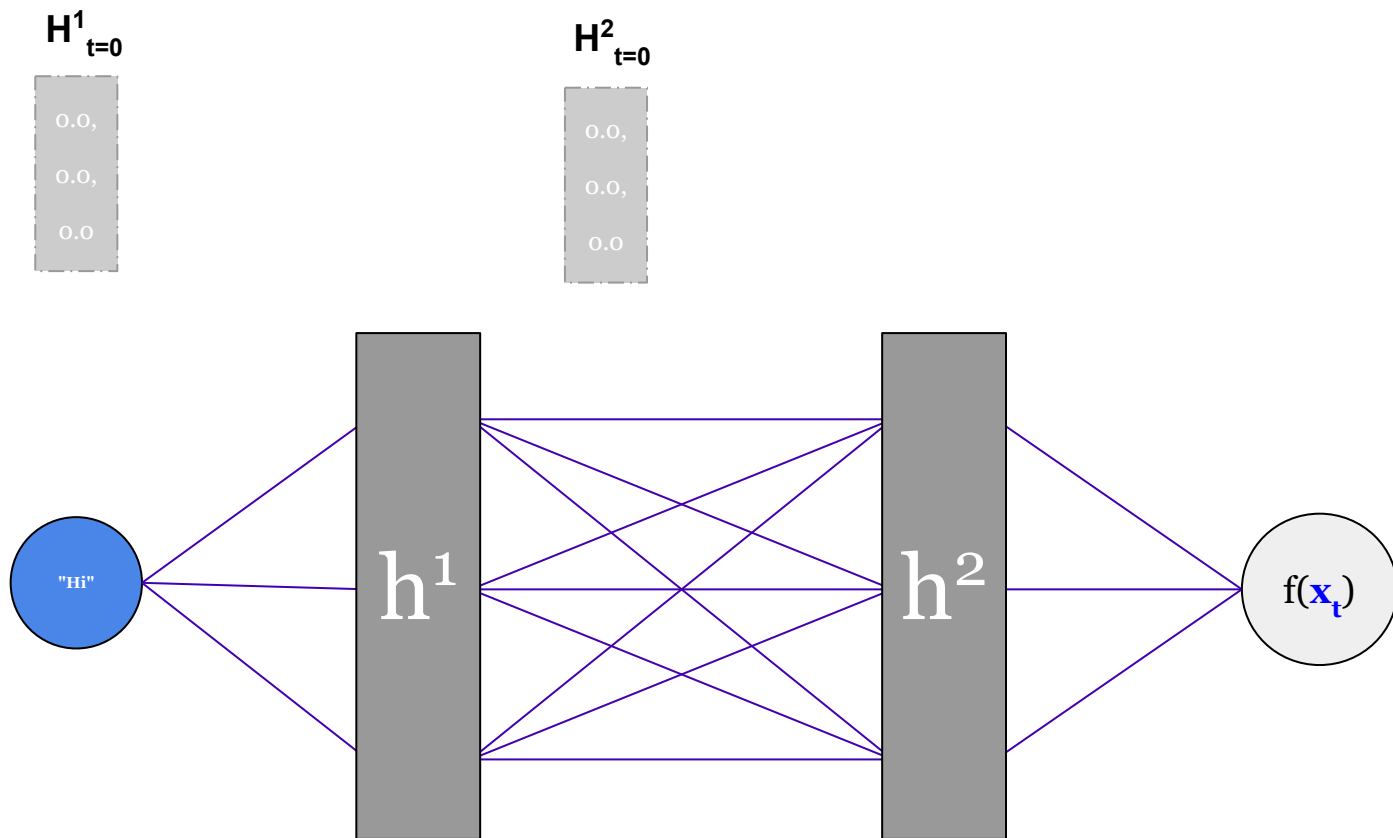


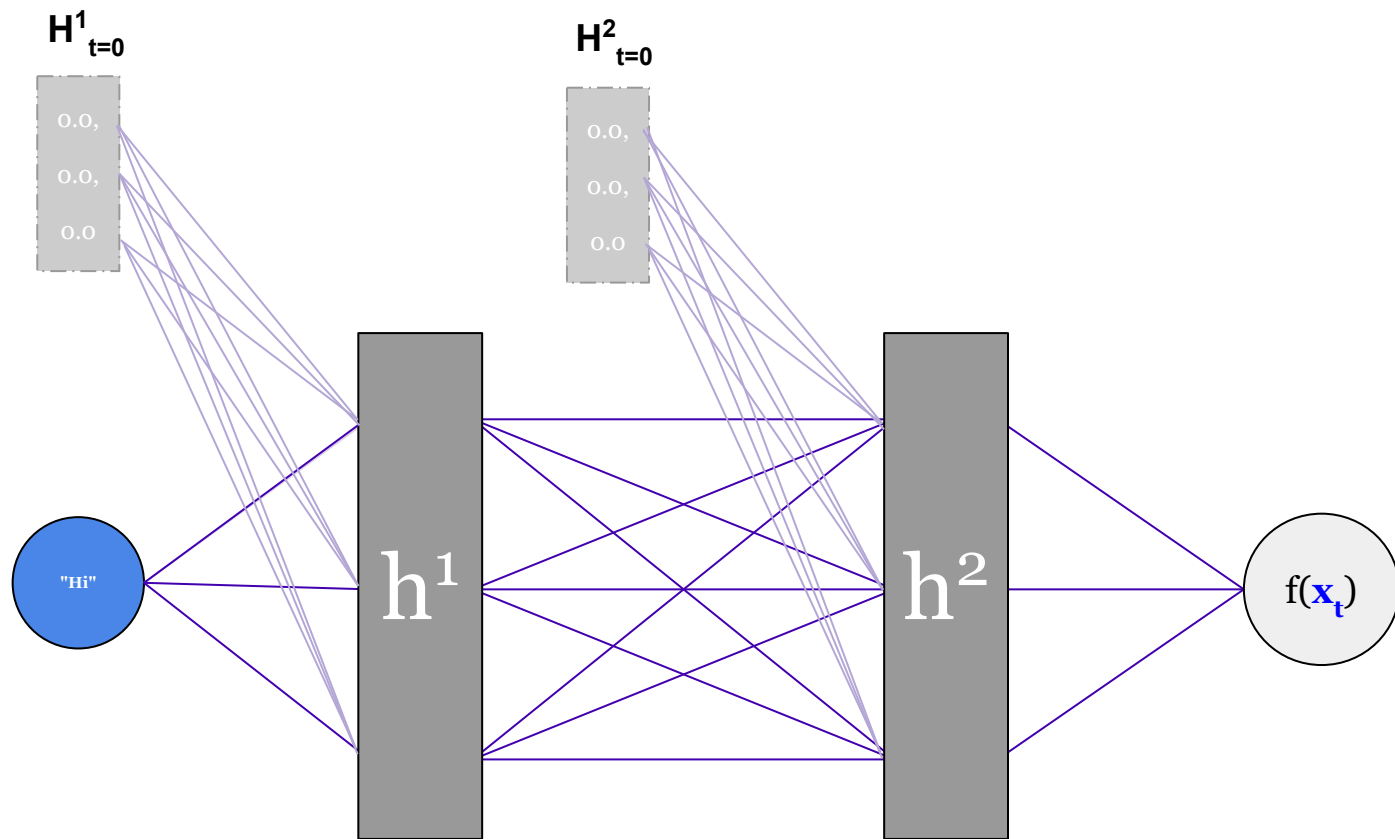


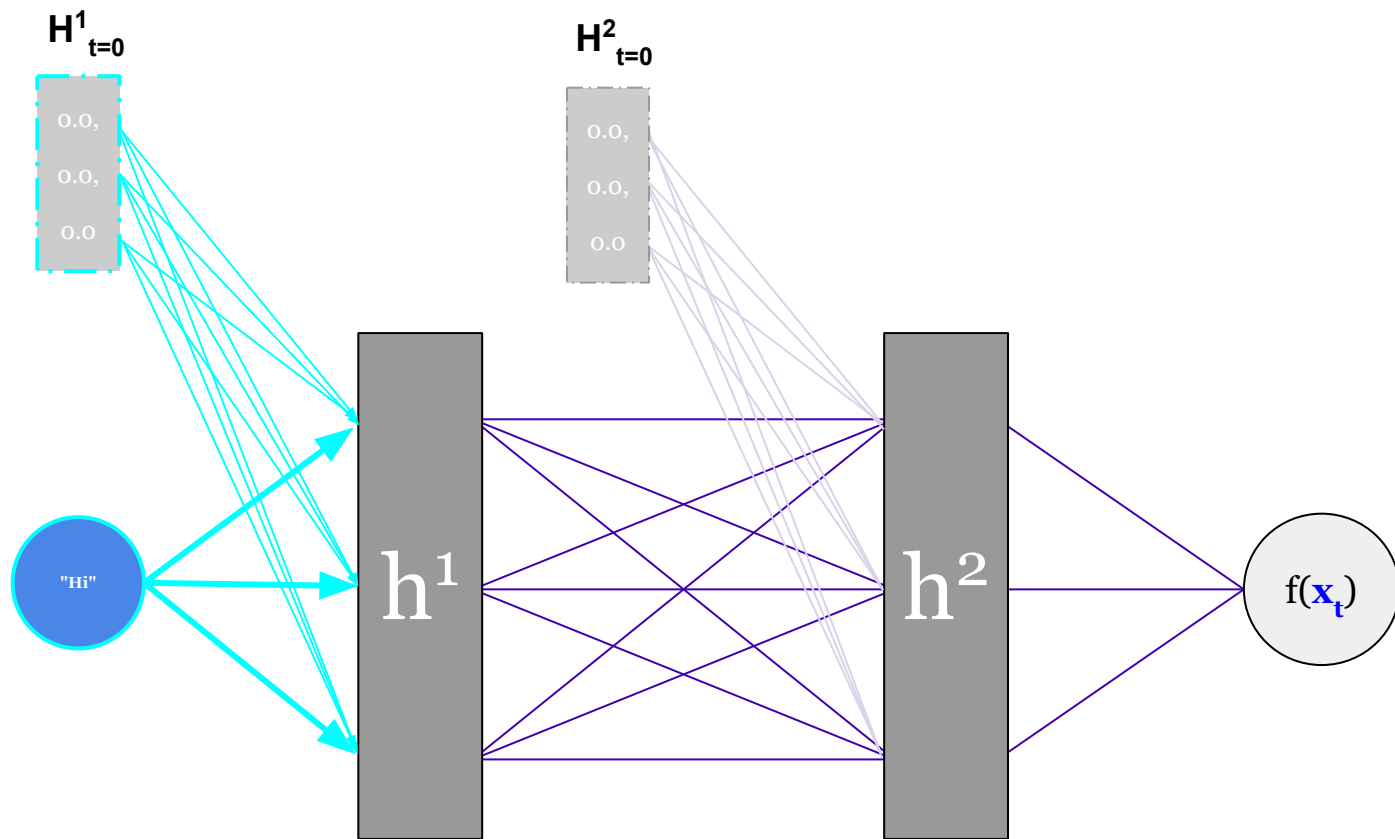


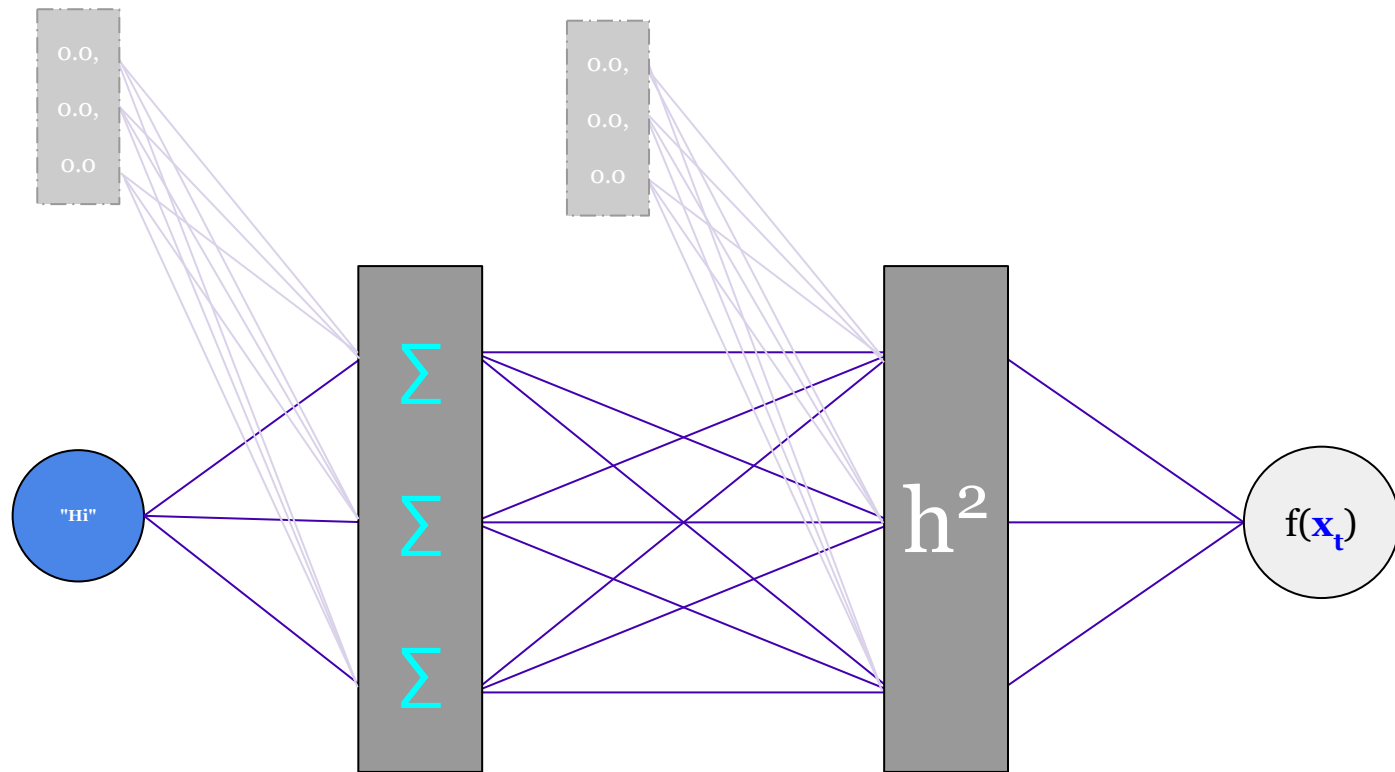
Learning to transform past data
into a "previous event" context

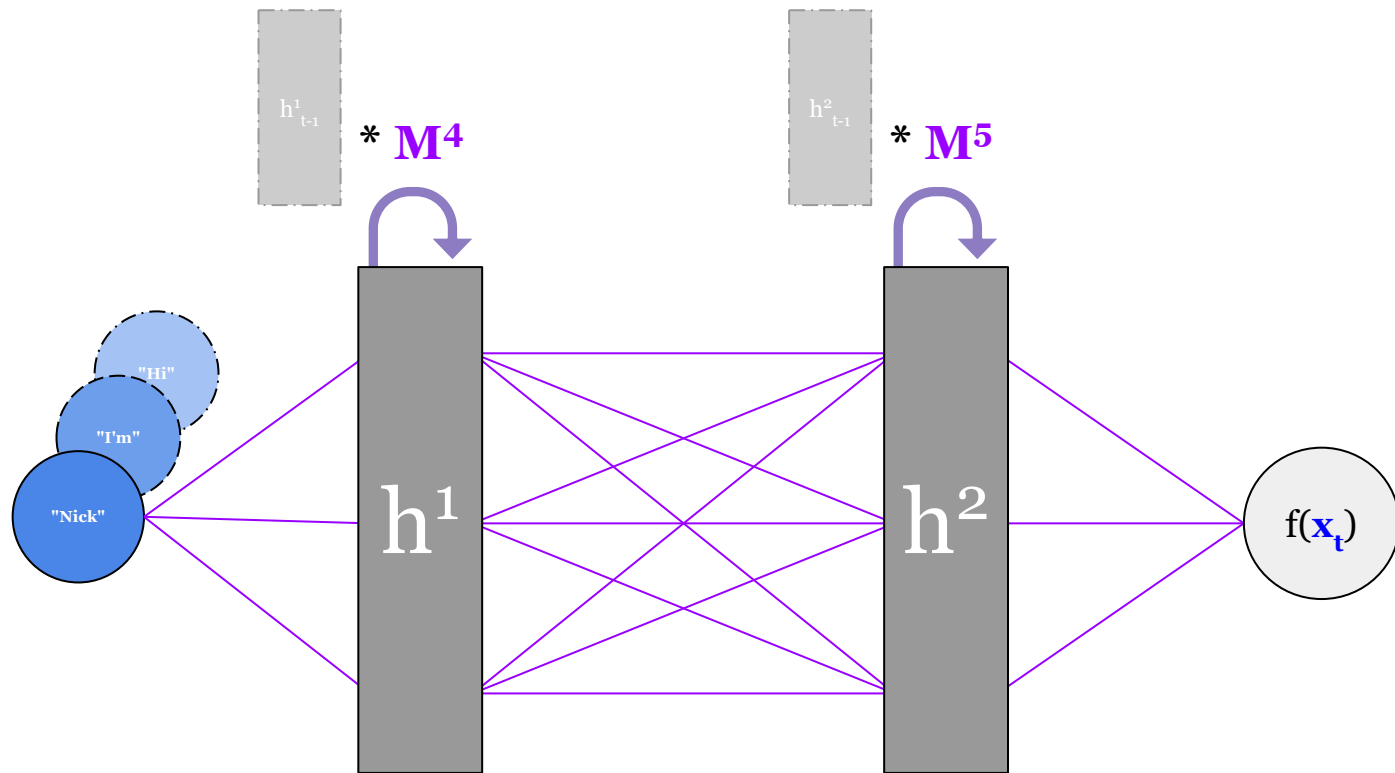












Demo Language model ?

Learning a Function

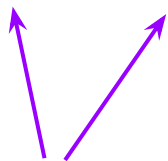


$$m\mathbf{x} + b = \mathbf{y}$$

Data containing **input** and **output** pairs

Learning a Function

$$y = mx + b$$



Learn some **parameters** that make this true for as many (x, y) pairs possible.

ML Models as Code

Linear Models (eg; Logistic Regression):

- * Very simple algorithms (SUMPRODUCT)
- * Fast and easy to train

Deep Neural Nets:

- * Arbitrarily complex algorithms
- * Slow to train, requires GPU hardware

Deep Learning History

Timeline Turing -> 80s Yann Lecun, Hinton, Schmidhuber, Bengio

Impractical compute obstacle

Impractical data obstacle

Training algorithm had flaws

Deep Learning History

90s -> 2000s

Training algorithm figured out

Lots of excitement "connectionists"

--SVM still winning vs. Neural Nets

Still huge compute obstacle

Still huge data obstacle

Deep Learning History

2010s -> present

Algorithms vastly expanded upon + researched

Compute solved w/ GPUs

Data solved w/ internet, smartphones, social media, ect..

Hinton students:

- Illya now director of OpenAI
- LeCun now director of FAIR
- Alex Graves (deepmind)
- Vlad Mnih (deepmind)

Michael Jordan students:

* Andrew NG

Deep Learning Today

Many fields currently **hijacked** by Deep Learning:

- * Computer Vision (CV)
- * Natural Language Processing (NLP)
- * Speech Recognition
- * Generative Modeling
- * Reinforcement Learning

Fields **created** by Deep Learning:

- * Adversarial Machine Learning
- * Adversarial Training
- * Neural Style Transfer
- * Automatic Video Generation and Alteration (?)