Deep Learning and Neural Nets

-a practical overview

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

---

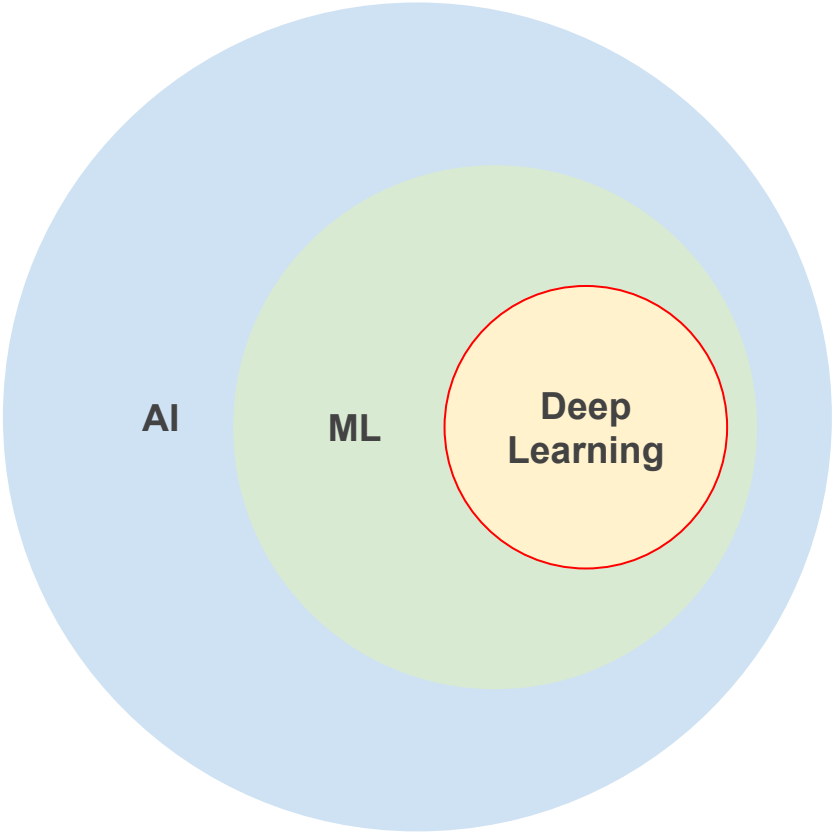
* What is Deep Learning?

* How is it different from other ML techniques?

* How does it work? (generally)

* Further steps

AI Landscape



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. ~ Wikipedia

* Implies use of Neural Network models

* Used to be called *Connectionism* (beginning of time-2000)

* Under-hyped before 2012 ("Perceptrons" by Marvin Minsky and Seymour Papert, 1969)

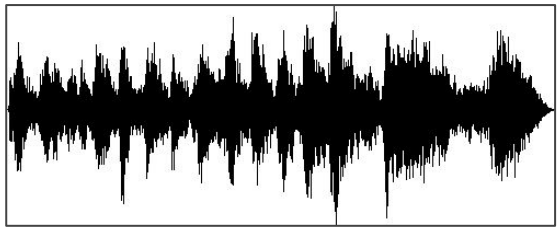
* Now very popular ML approach, major economic impacts, 'ai renaissance', ect..

Types of Data

Structured Data

	name	sex	age	height	weight
1	Aubrey	M	41	74	170
2	Ron	M	42	68	166
3	Carl	M	32	70	155
4	Antonio	M	39	72	167
5	Deborah	F	30	66	124
6	Jacqueline	F	33	66	115
7	Helen	F	26	64	121
8	David	M	30	71	158
9	James	M	53	72	175
10	Michael	M	32	69	143
11	Ruth	F	47	69	139
12	Joel	M	34	72	163
13	Donna	F	23	62	98
14	Roger	M	36	75	160
15	Yao	M		70	145
16	Elizabeth	F	31	67	135
17	Tim	M	29	71	176
18	Susan	F	28	65	131

Unstructured Data



KANYE WEST
@kanyewest
Follow

Super inspired by my visit to Ikea today , really amazing company… my mind is racing with the possibilities…

2:42 AM - 9 Mar 2016

24,772 45,313

Drawbacks of Neural Networks

- Data hungry
  - Not recommended if < 1000's of samples
  - Ideally have > 100,000 samples

- Can be expensive to train (tons of matrix/tensor ops & calculus)

- Many practitioner choices

- Debugging and interpreting individual model decisions is not trivial

Deep Learning Today

Similar to connectionist models popular in the '80s and '90s, but with:

* Web-scale **data** sets

* More **compute** (GPU, Cloud, Nvidia Tensore Cores)

* Powerful **software** tools (TensorFlow, Keras, Torch, ect..)

* **Innovations** (more layers, LSTMs, attention, faster training, ect..)

* Research interest: http://paperscape.org/

Models and Supervised Learning

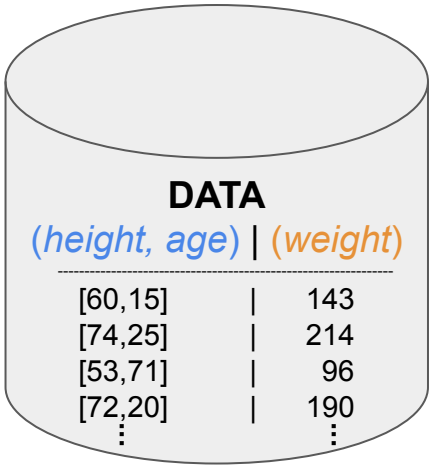
Supervised Learning

Inputs

[x]

Labels/Target outputs

[y]



Supervised Learning

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

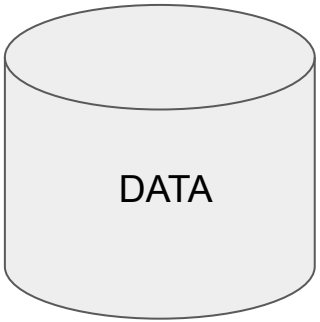
Model

Neural Nets: Tunable Functions

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

Model

Rule-based Model



Model

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

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

Parsimonious Models

$$PV = nRT$$

Relating pressure P, volume V, number of moles n, and 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

Neural Embeddings

Preparing the Data for Math

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

How to feed X into the model?

Preparing the Data for Math

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

cat vs. bat vs. car vs. paw

"cat" = [99, 97, 116]

"bat" = [98, 97, 116]

"car" = [99, 97, 114]

"paw" = [112, 97, 119]

```
word_vector = []

for character in word:
    word_vector.append(int(character))
```

Preparing the Data for Math

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

cat vs. bat vs. car vs. paw

"cat" = [99, 97, 116]

"bat" = [98, 97, 116]

"car" = [99, 97, 114]

"paw" = [112, 97, 119]

Euclidean/L2 Distance

$$D(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

D(*"cat"*, *"cbt"***)** = 1

D(*"cat"*, *"bat"***)** = 1

D(*"cat"*, *"paw"***)** = 178

Preparing the Data for Math

Basic approach: One hot encoding

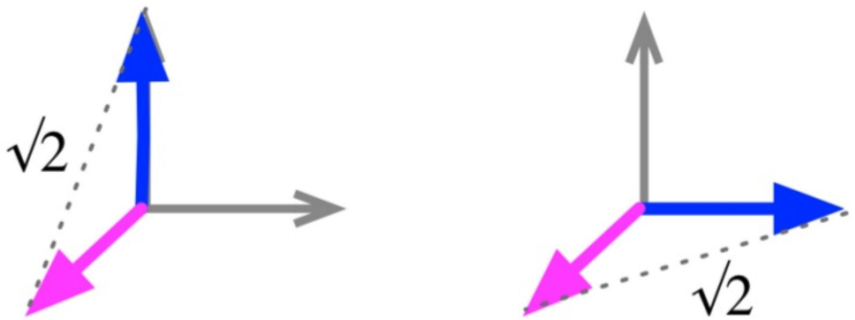
cat vs. bat vs. car vs. paw

"cat" = [1, 0, 0, 0]

"bat" = [0, 1, 0, 0]

"car" = [0, 0, 1, 0]

"paw" = [0, 0, 0, 1]



Preparing the Data for Math

Basic approach: One hot encoding

cat vs. bat vs. car vs. paw

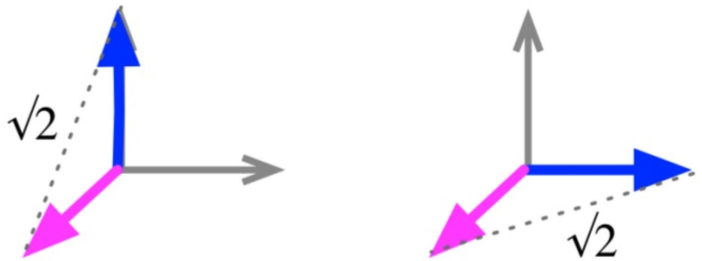
"cat" = [1, 0, 0, 0, ... , 0]

"bat" = [0, 1, 0, 0, ... , 0]

"car" = [0, 0, 1, 0, ... , 0]

"paw" = [0, 0, 0, 1, ... , 0]

$$\mathbf{D}(\textit{any}\ \mathbf{v}^{\mathbf{a}}, \textit{any}\ \mathbf{v}^{\mathbf{b}}) = \mathbf{1.4142}$$



Preparing the Data for Math

Basic approach: One hot encoding

Vocab = English Language = 50,000 words

"cat" = [1, 0, 0, 0, ... , 0]

"bat" = [0, 1, 0, 0, ... , 0]

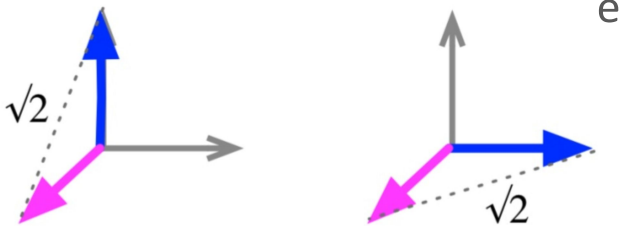
"car" = [0, 0, 1, 0, ... , 0]

"paw" = [0, 0, 0, 1, ..., 0]

len(cat) = 50,000

len(paw) = 50,000

ect...



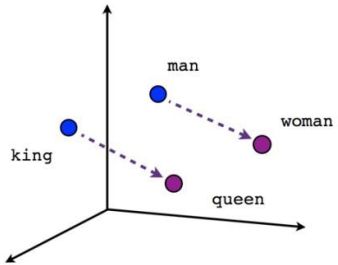
Preparing the Data for Math

Word2Vec Embeddings: try to point word vectors in directions w.r.t. lexical meaning & preserve semantic analogies

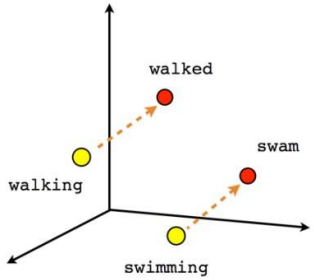
man -> [...]

woman -> [...]

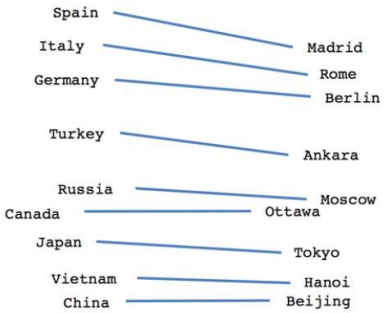
king -> [...]



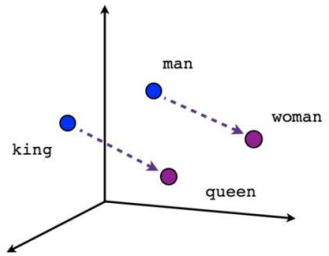
Male-Female



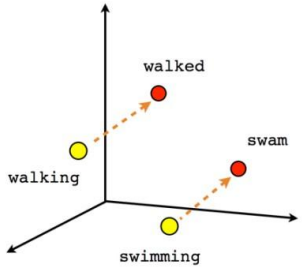
Verb tense



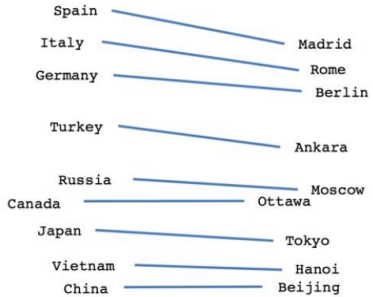
Country-Capital



Male-Female



Verb tense



Country-Capital

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

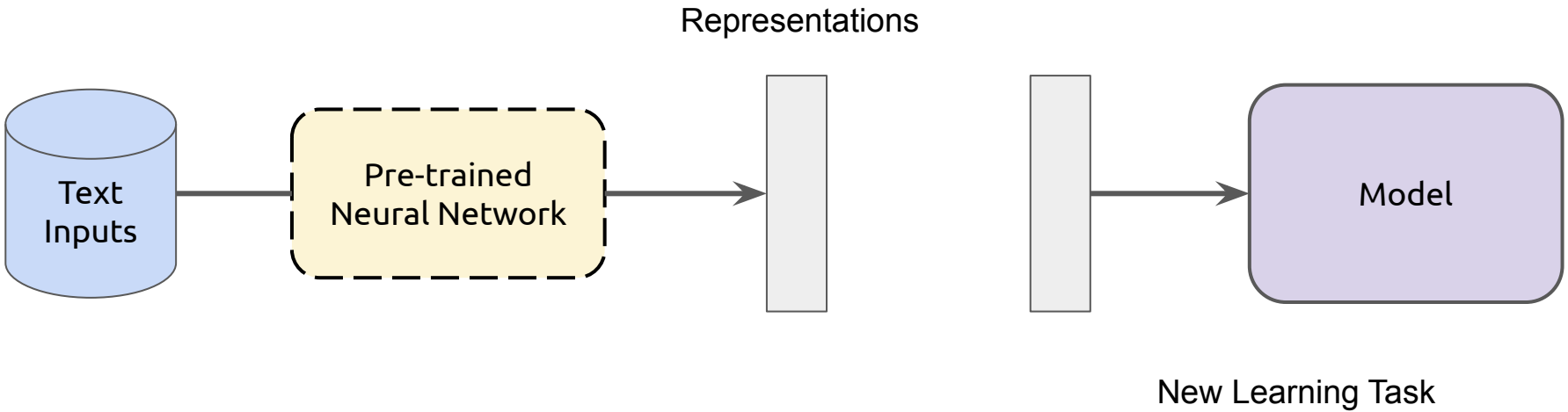
Algebra in the Latent Space

"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" Radford et al. 2016

Works for other Data

Embeddings

New NLP task



Neural Network Models

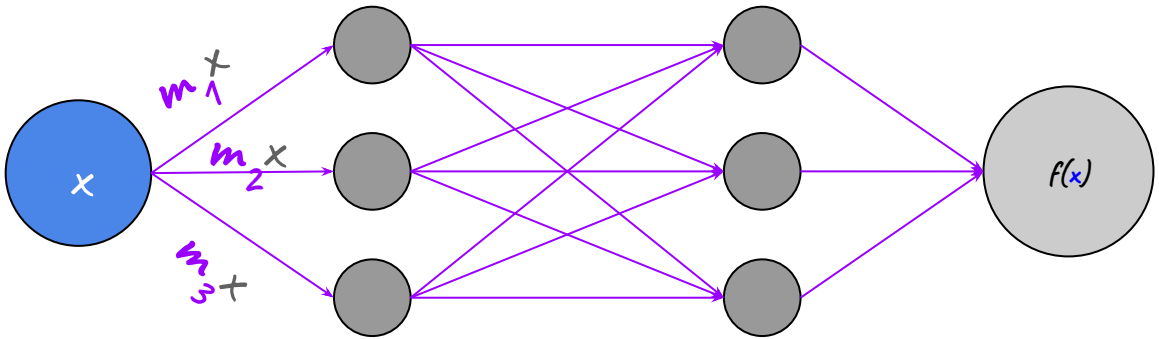
Neural Nets: Tunable Functions

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

Model

Neural Nets: Tunable Functions (with many parameters)

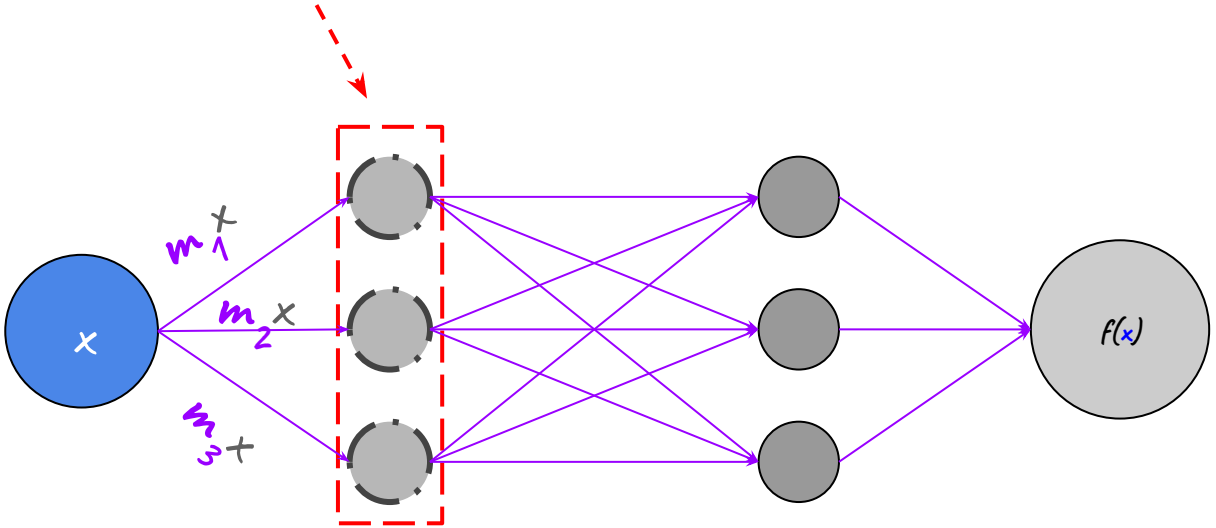
$$h_1(x) = [m]x + [b], \quad h_2(h_1) = [m]h_1 + [b], \quad \ldots$$



Neural Nets: Tunable Functions (with many parameters)

$$h(x) = \text{ReLu}([m]x + [b])$$

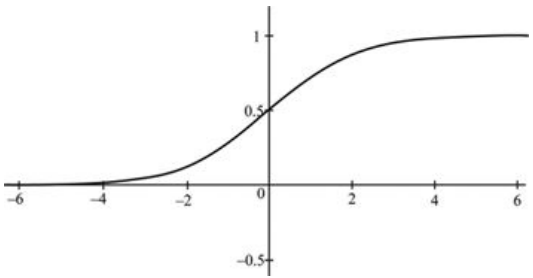
ReLu(k) = max(0, k)



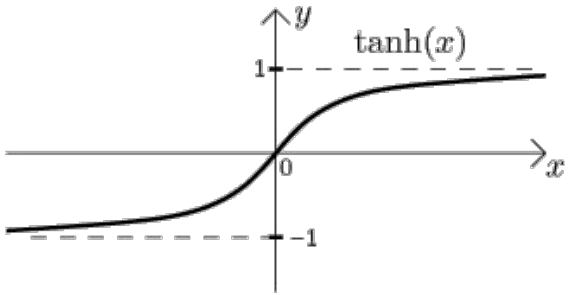
Neural Nets: Hidden Activation Functions

$$f(x) = \text{activation}(mx + b)$$

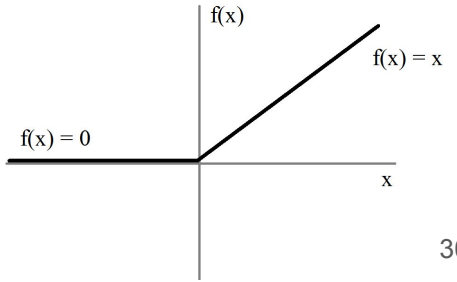
sig(x)



tanh(x)



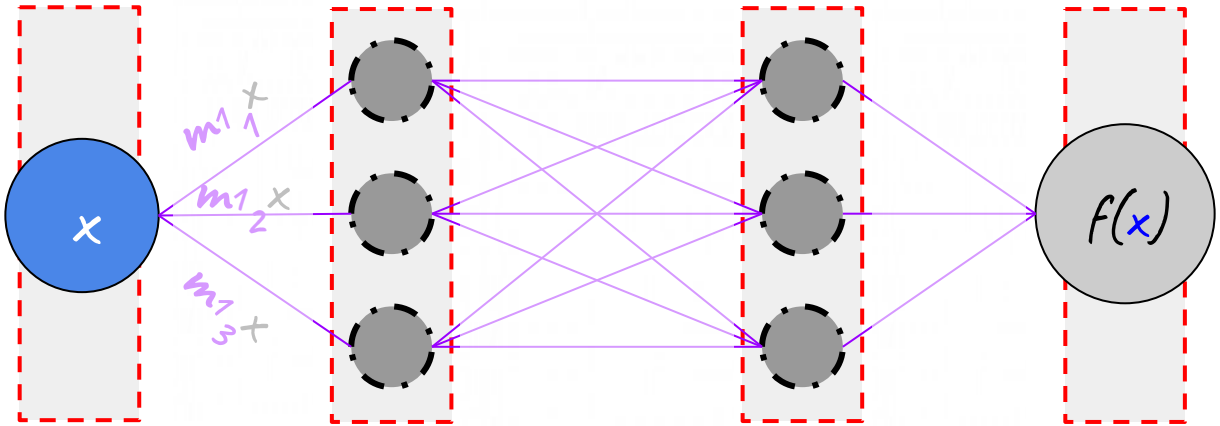
ReLu(x)



Vocab Detour

Vocab

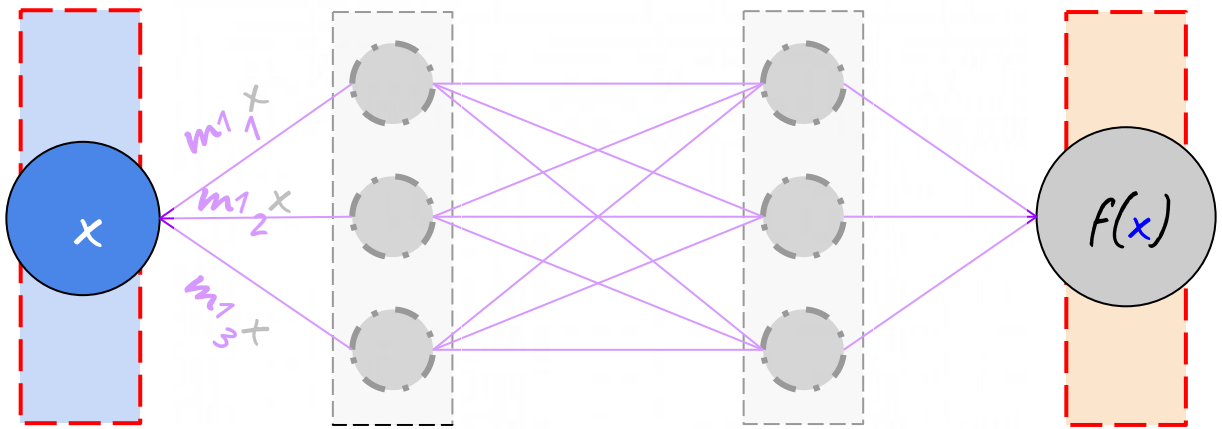
Layers



Vocab

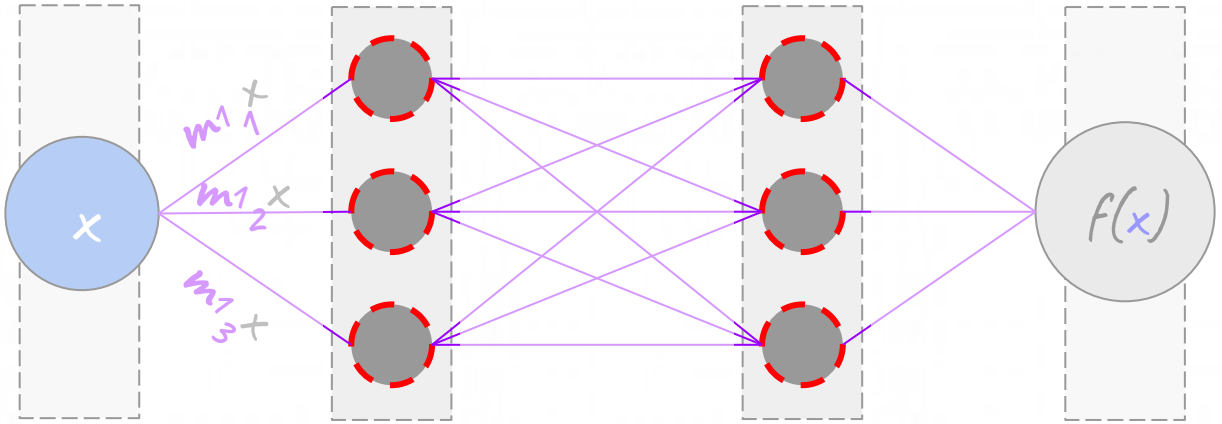
Input Layer

Output** (Logit) **Layer



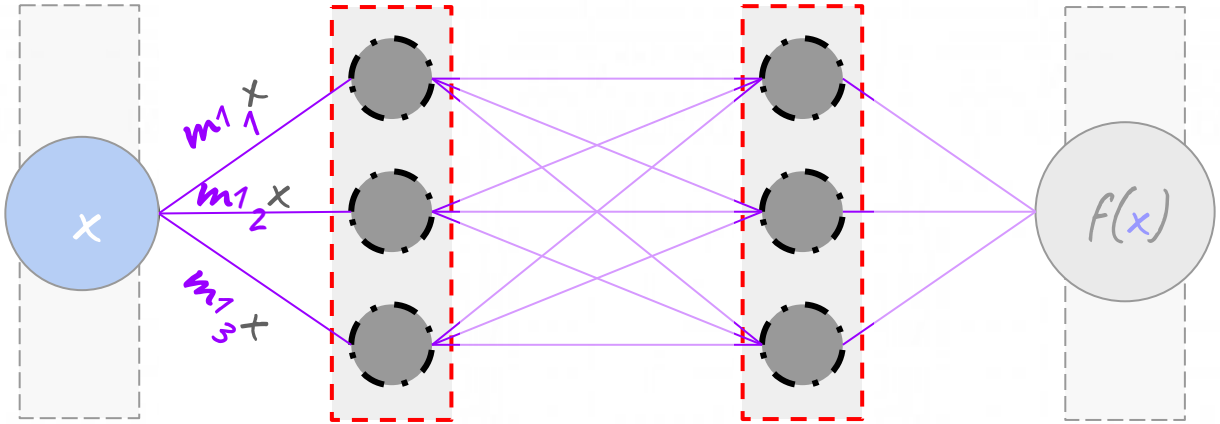
Vocab

Hidden Units



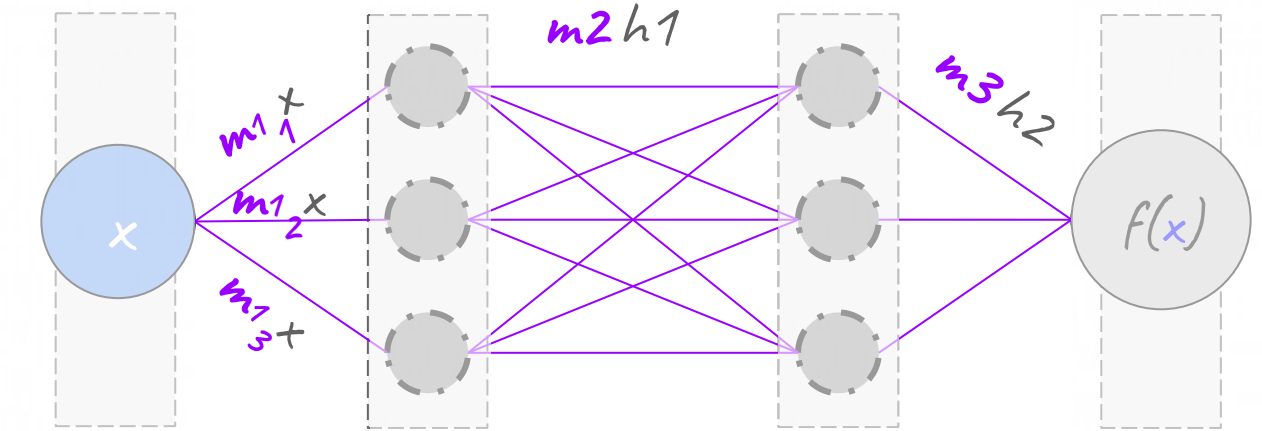
Vocab

Hidden/Dense Layers



Vocab

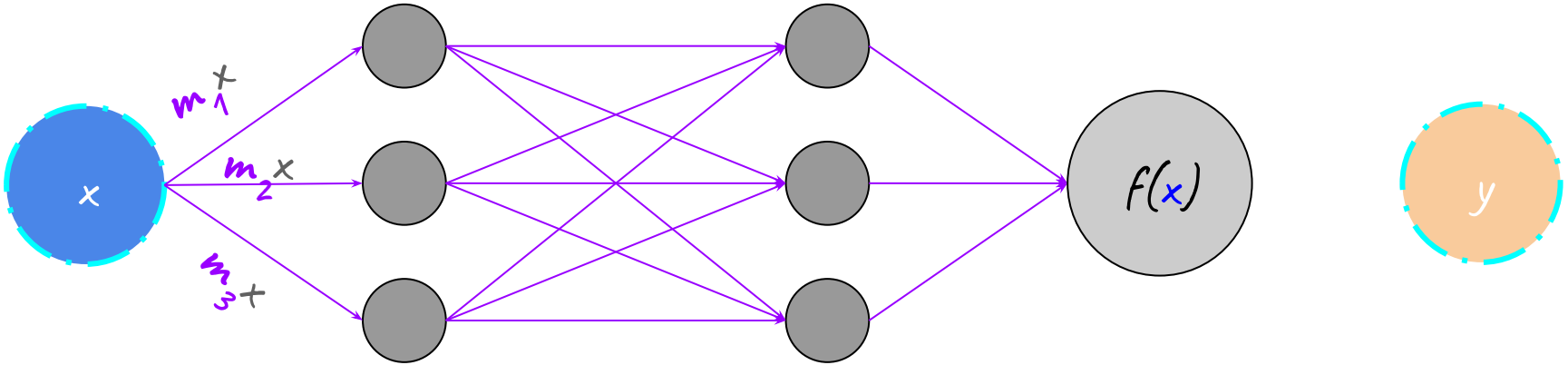
Weights



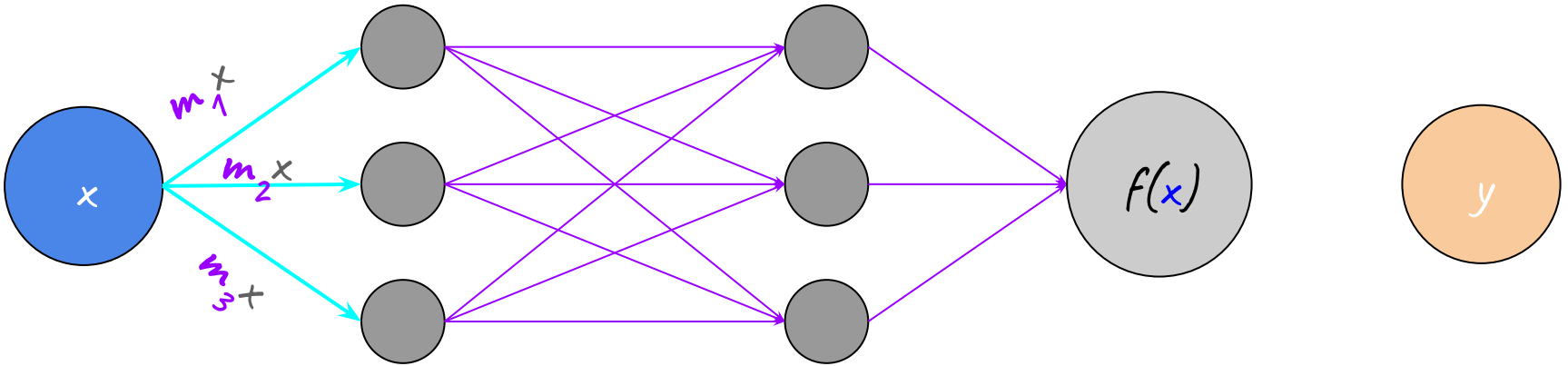
Training Loop

Neural Nets: Training Loop

Sample an (x,y) pair from data

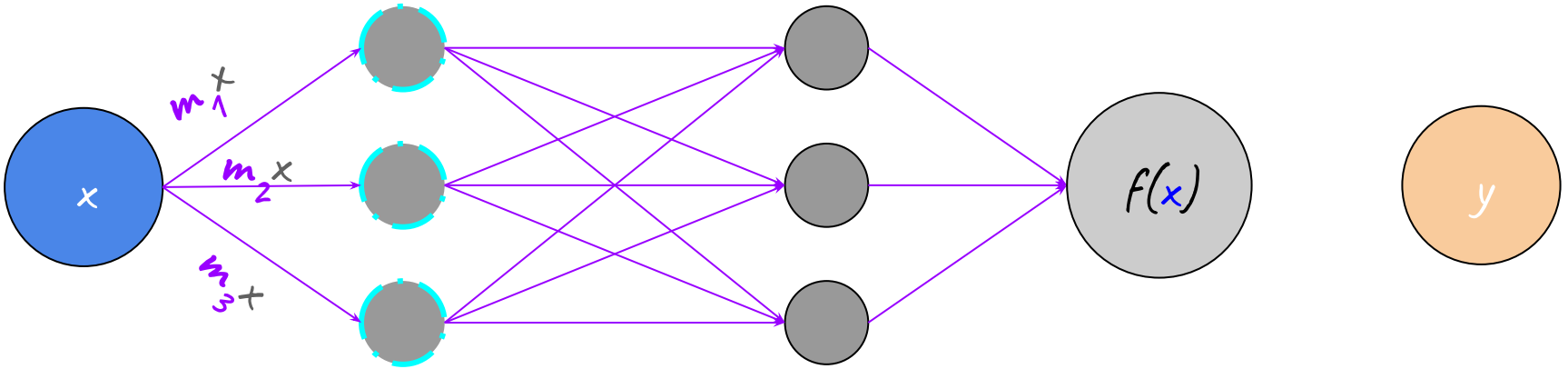


Neural Nets: Training Loop

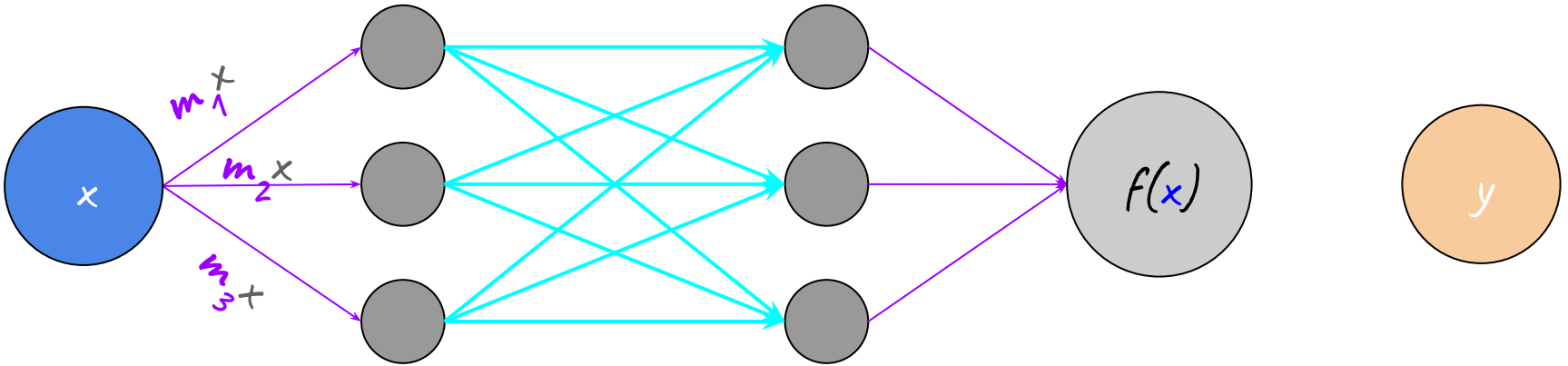


Neural Nets: Training Loop

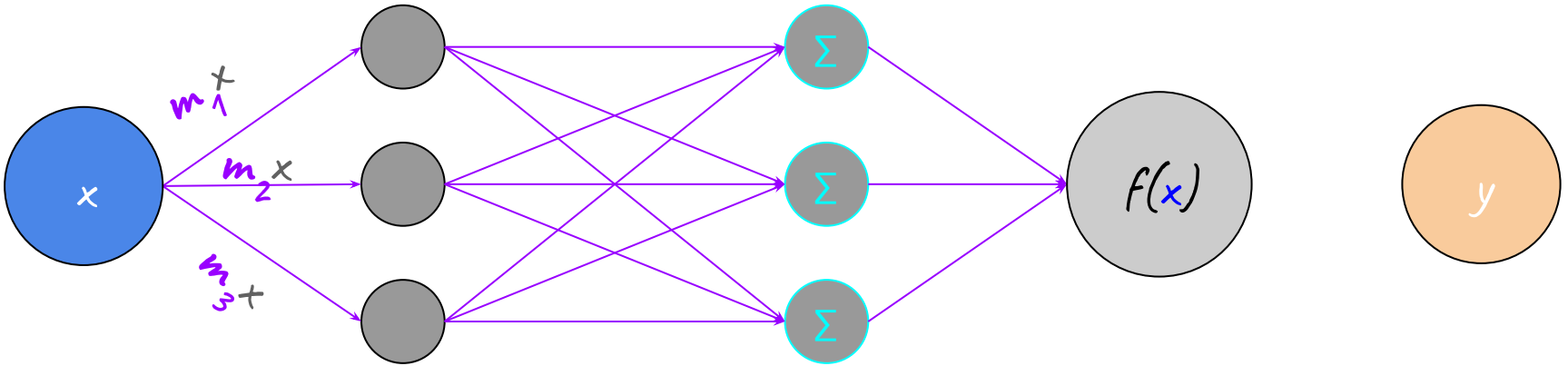
Set any negatives to 0



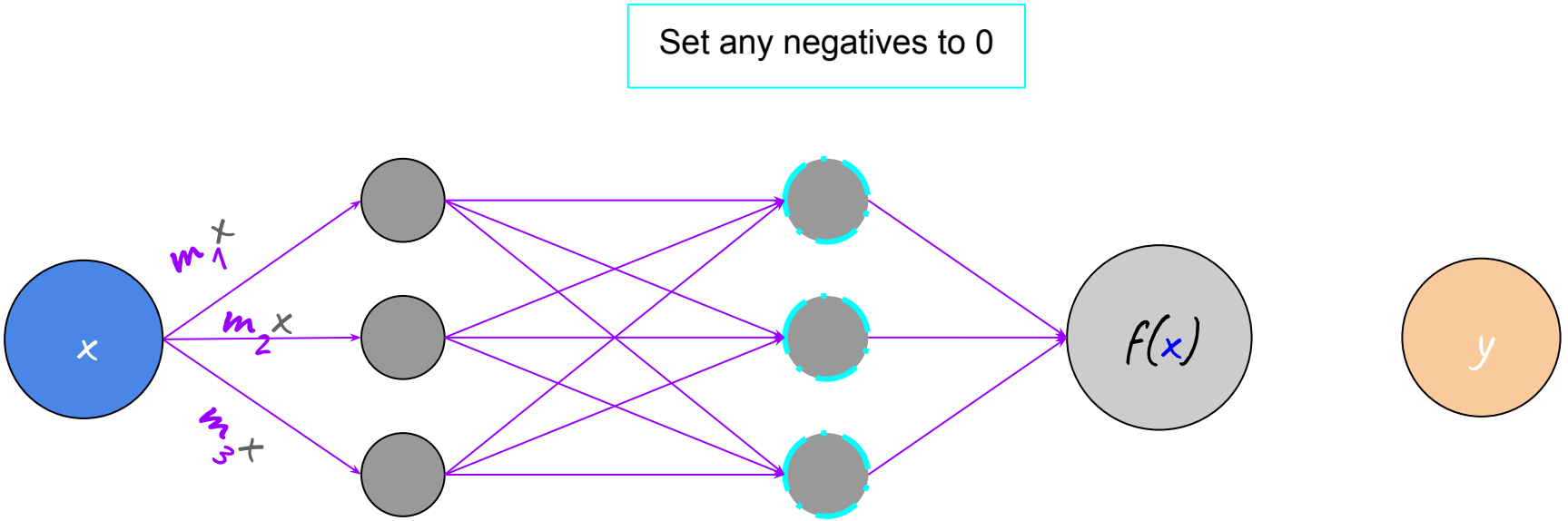
Neural Nets: Training Loop



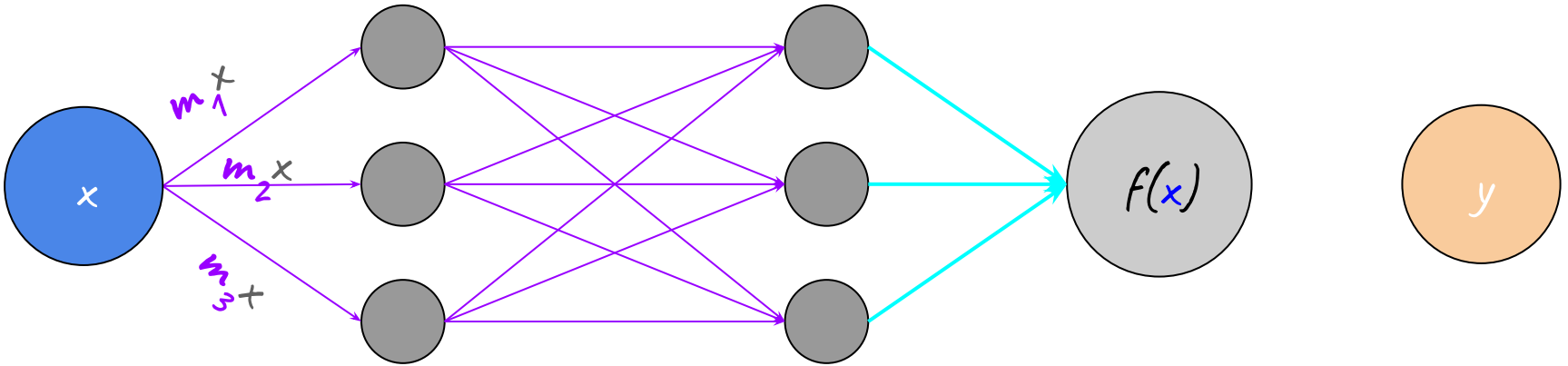
Neural Nets: Training Loop



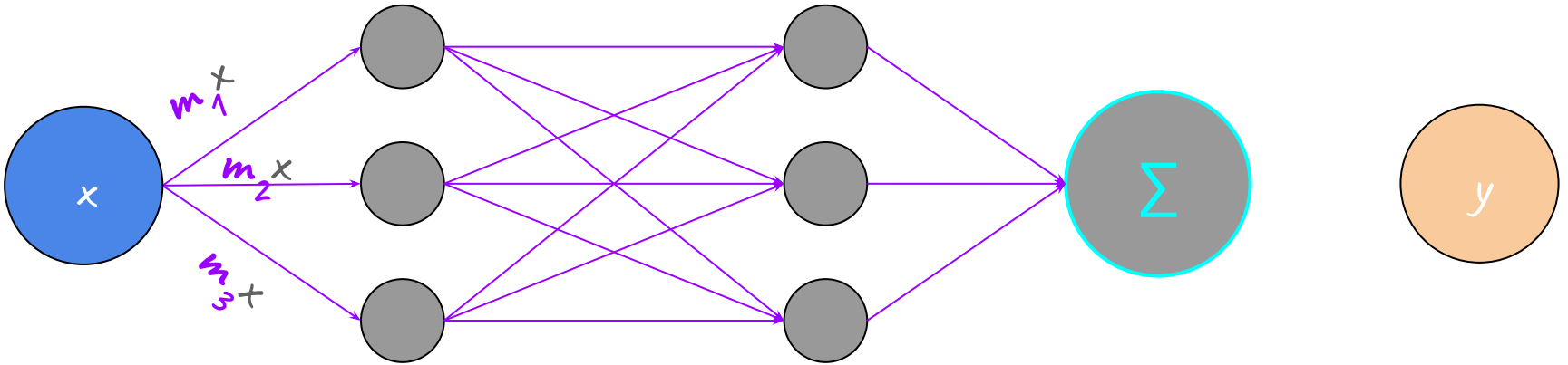
Neural Nets: Training Loop



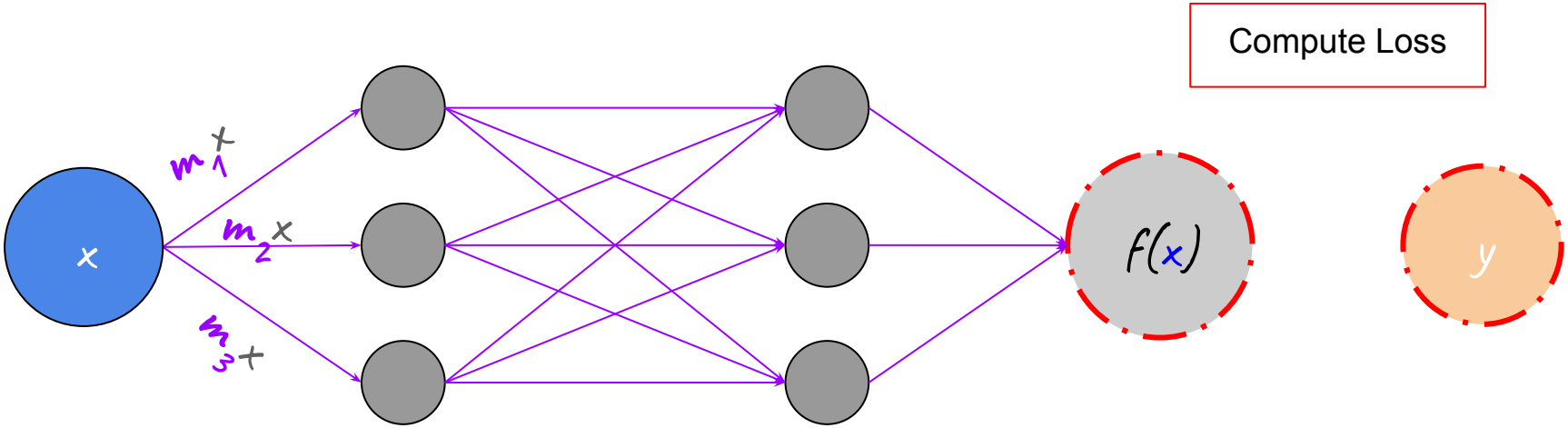
Neural Nets: Training Loop



Neural Nets: Training Loop

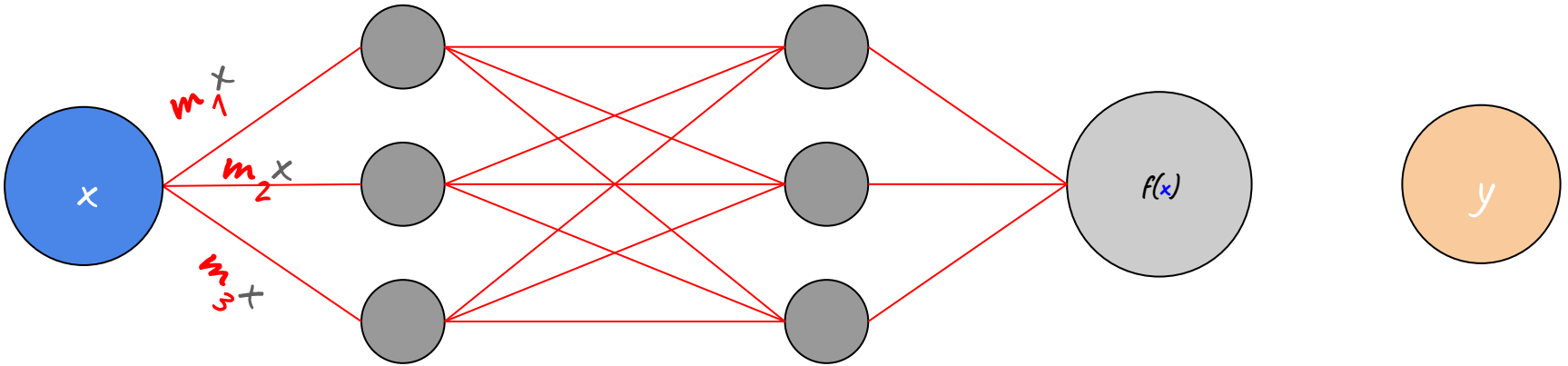


Neural Nets: Training Loop



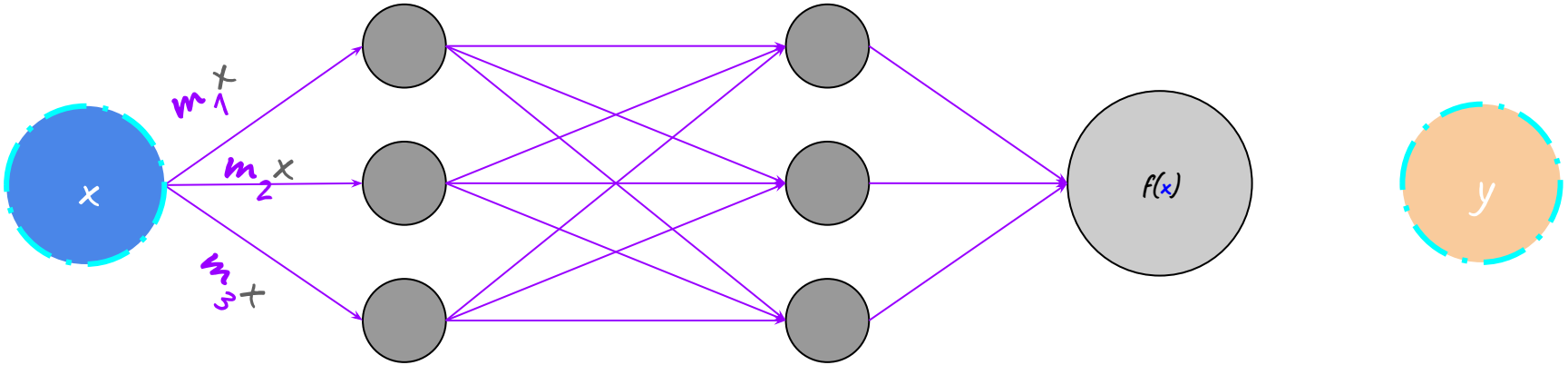
Neural Nets: Training Loop

Update parameters to reduce loss for this (x,y) pair

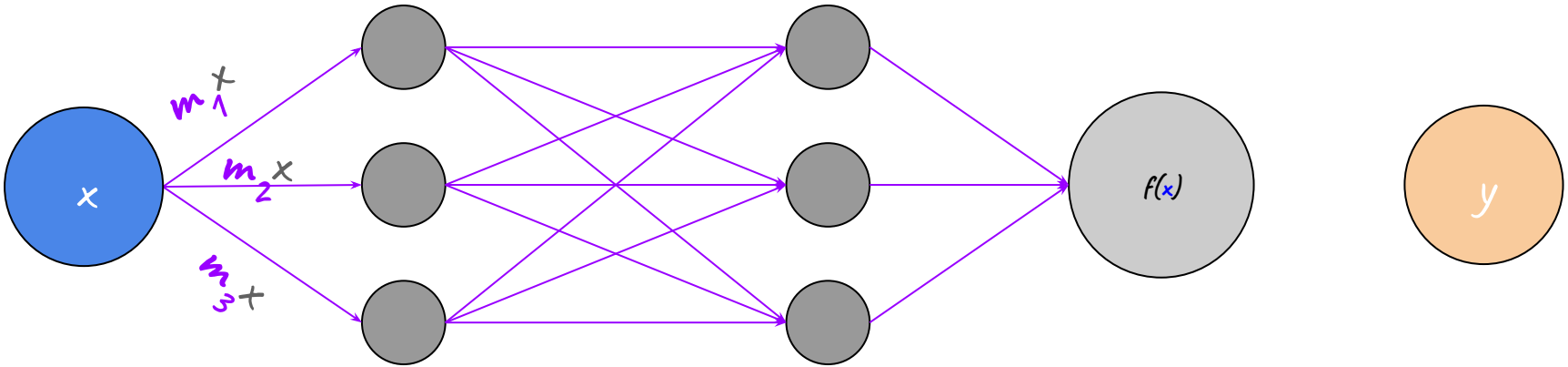


Neural Nets: Training Loop

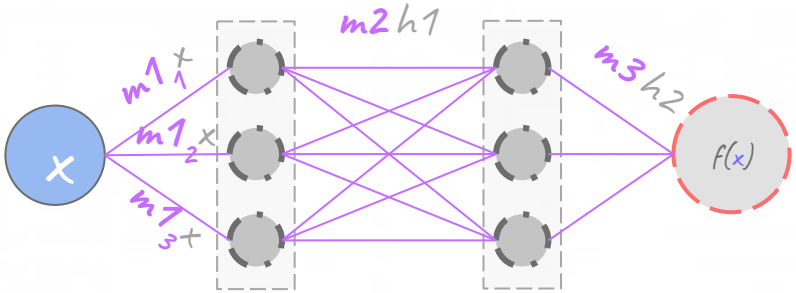
Sample a new (x,y) pair and repeat until loss is sufficiently low



Questions?

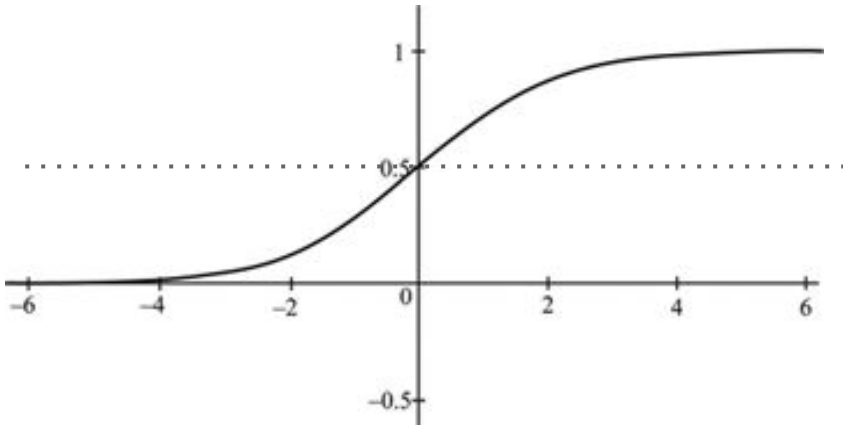


Output Activation (binary classification)



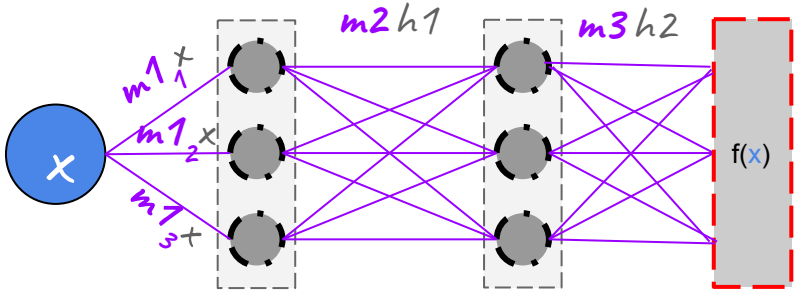
$f(x) = a(f(x)) \approx P(y \mid x)$

a = sigmoid



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

Output Activation (n-way classification)



$f(x) = \text{argmax}\, a([f(x)]) \approx P(y|x)$

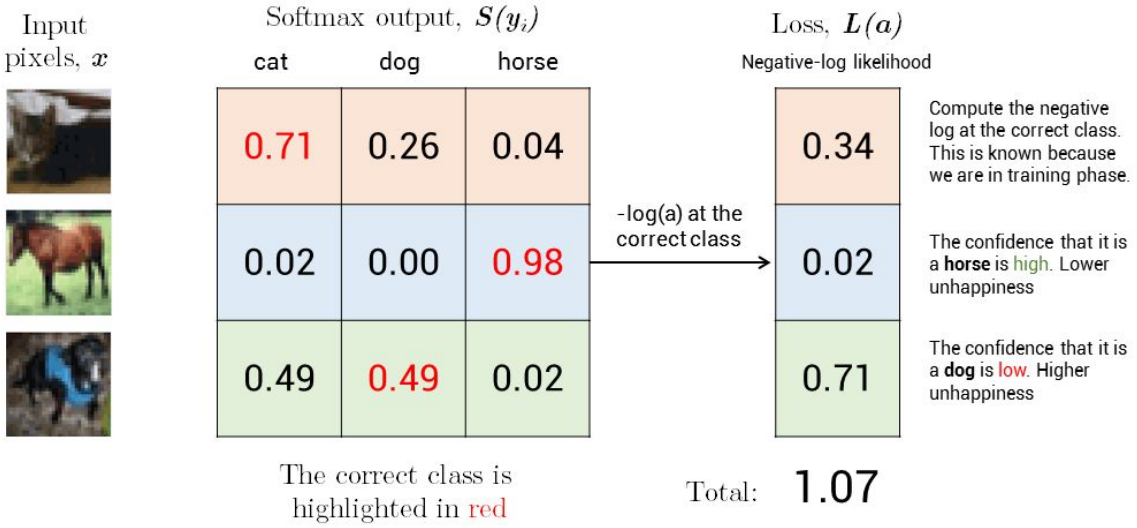
$a = \text{Softmax}([..])$

$$\begin{bmatrix} 5.0 \\ 4.0 \\ 3.0 \\ 2.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} 0.636 \\ 0.234 \\ 0.086 \\ 0.032 \\ 0.012 \end{bmatrix}$$

Loss Functions

Classification:

Loss = -log(P(*true_class*))



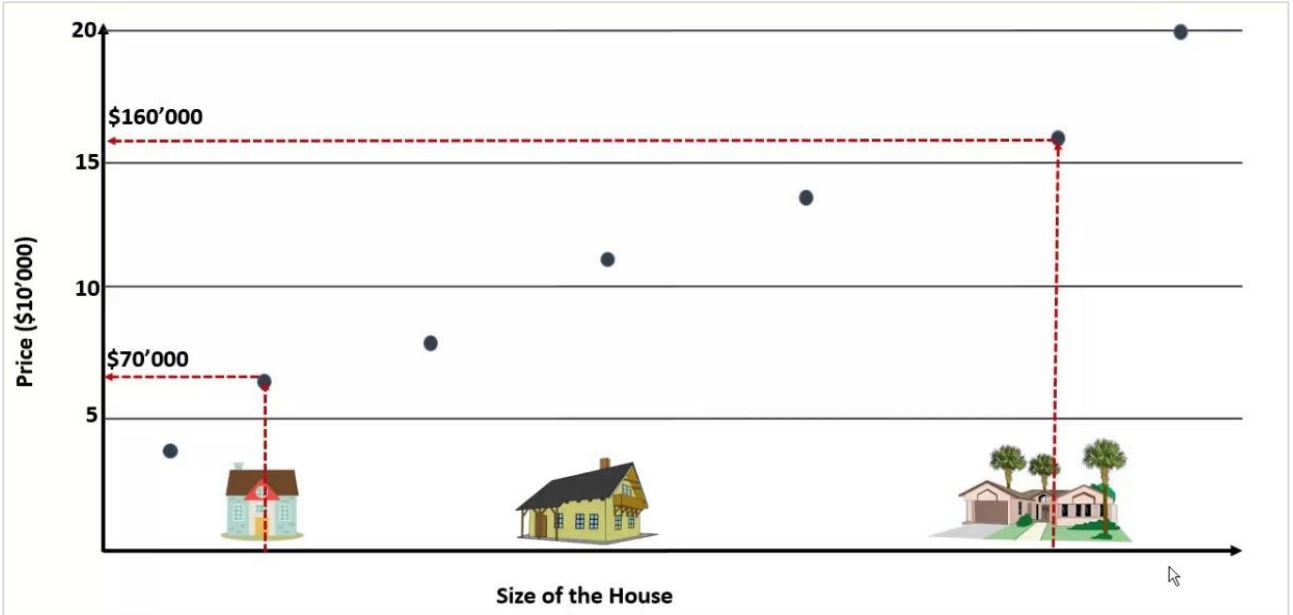
-log(0.0) = infinity

-log(1.0) = 0

Loss Functions

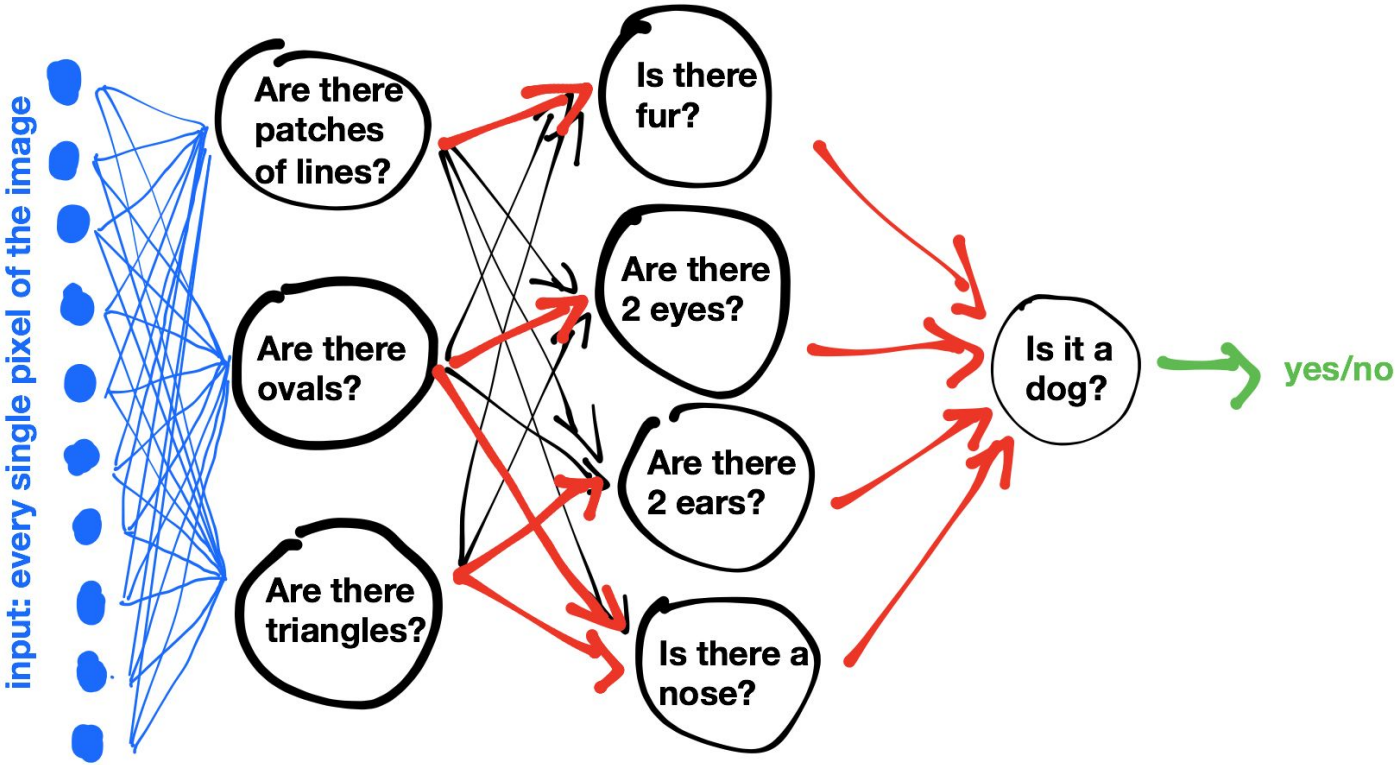
Regression:

Loss = Mean Squared Error $$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

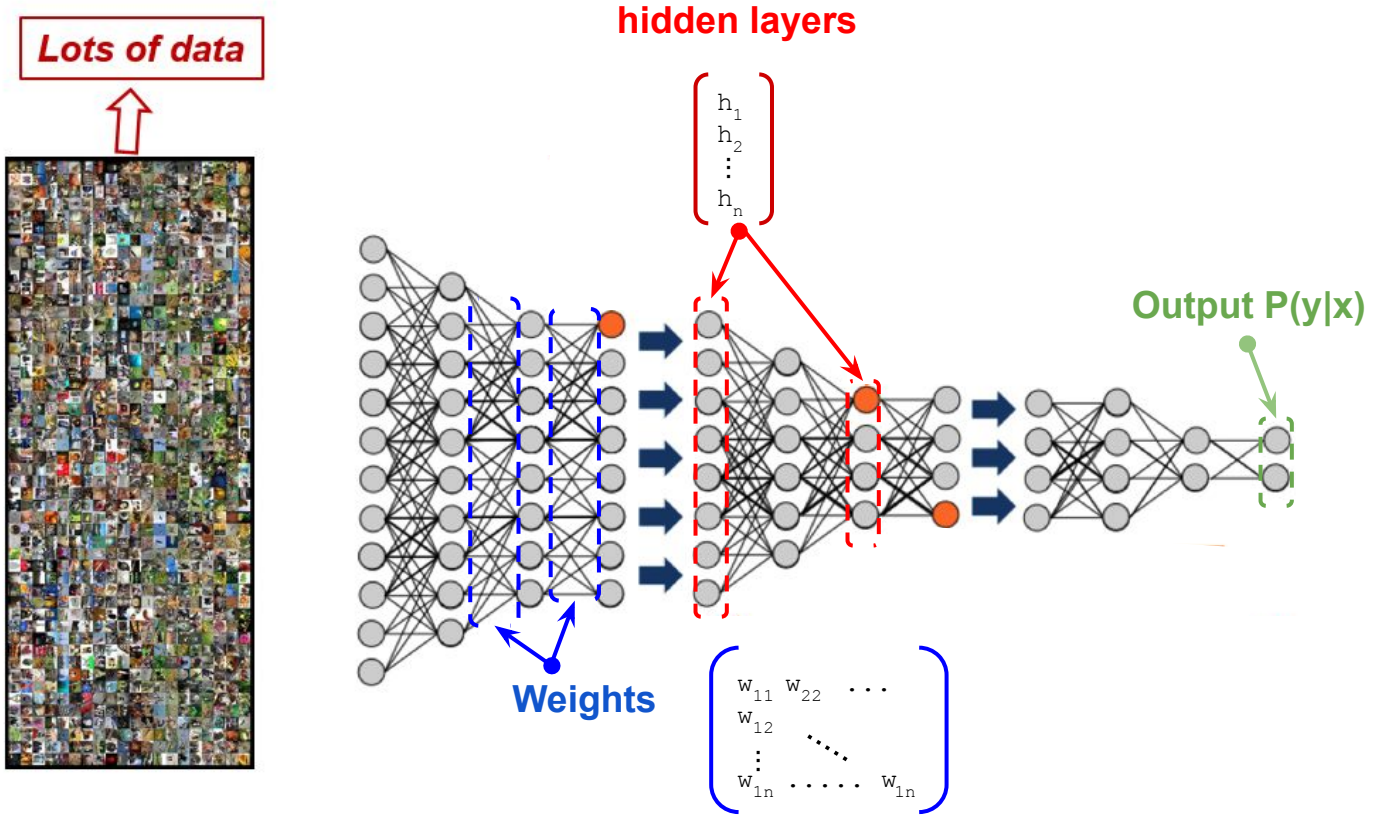


Feature Hierarchy

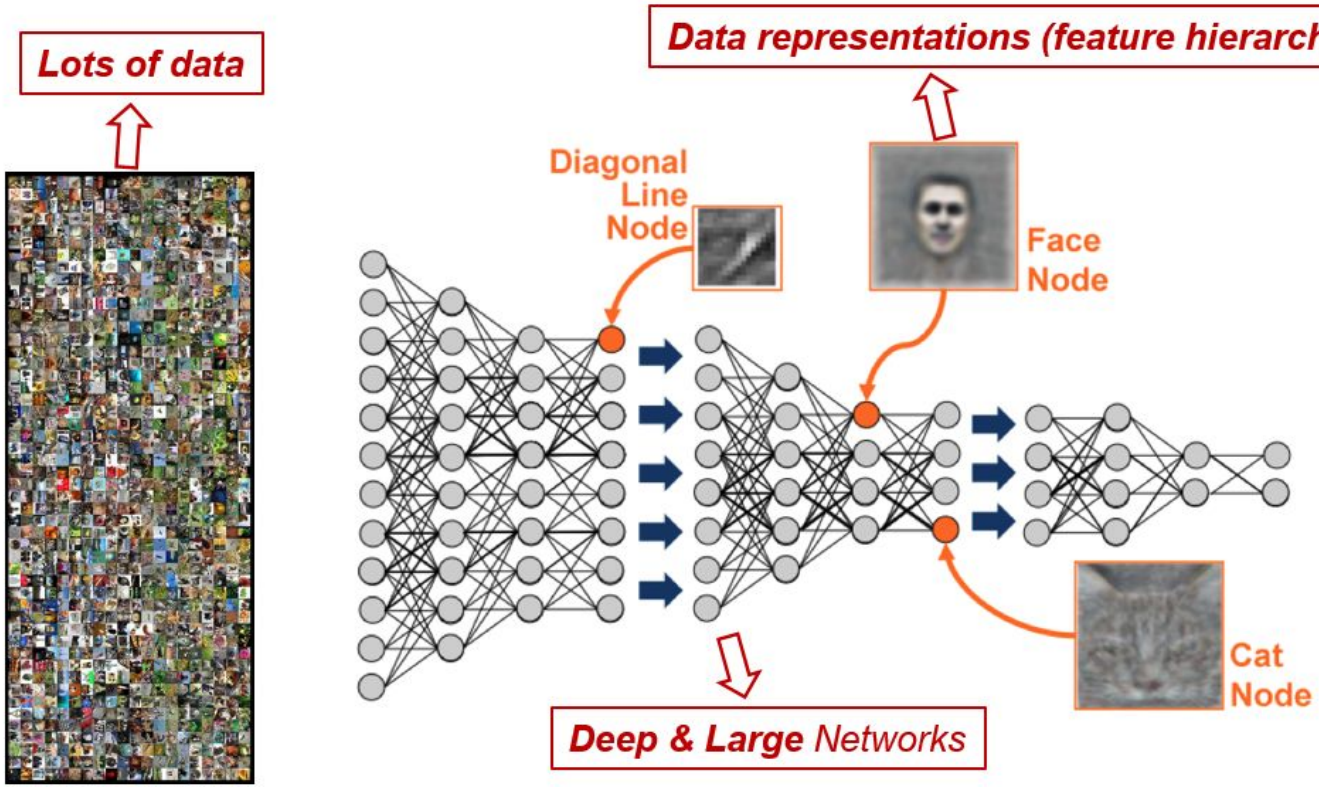
Neural Networks

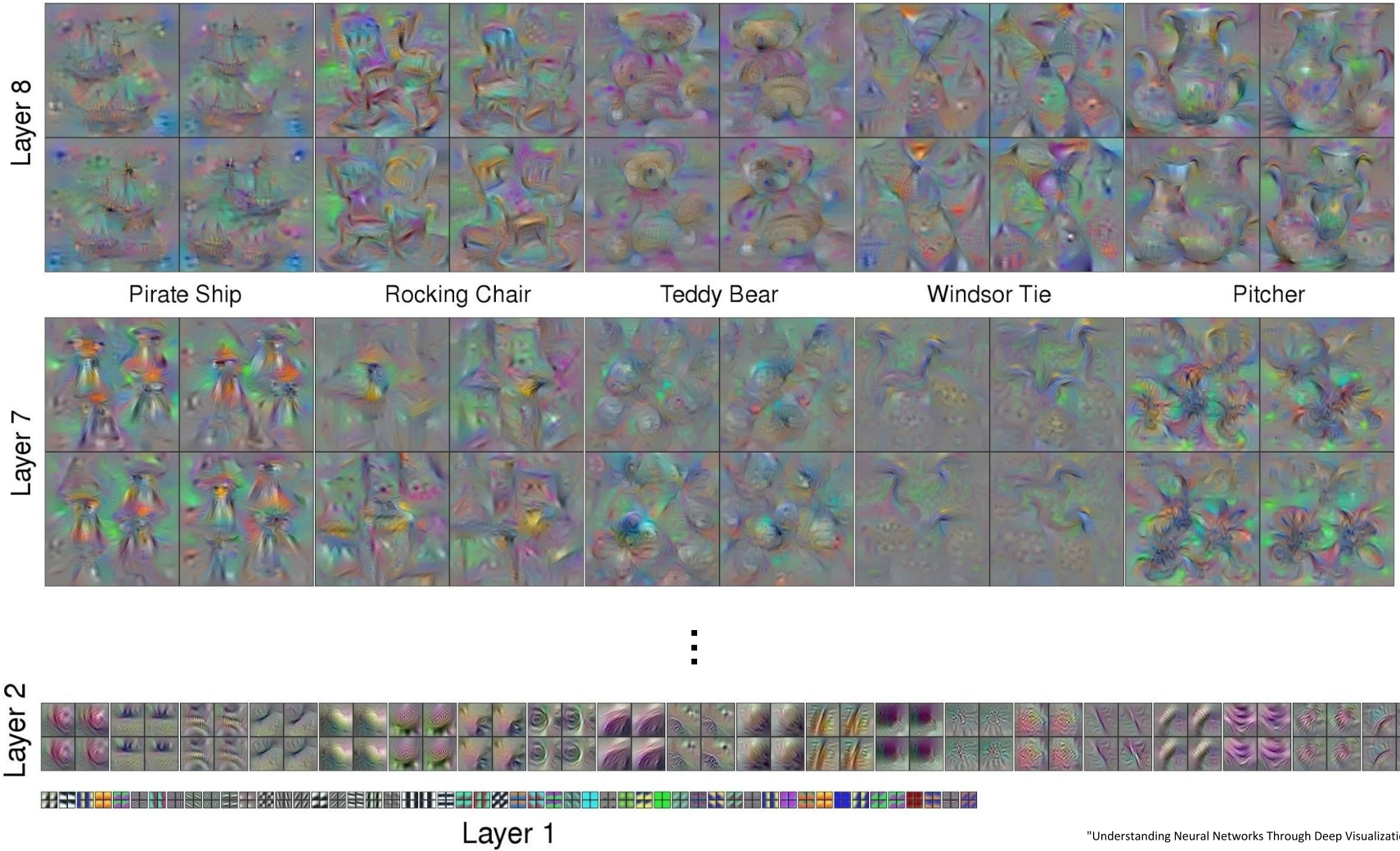


Neural Networks

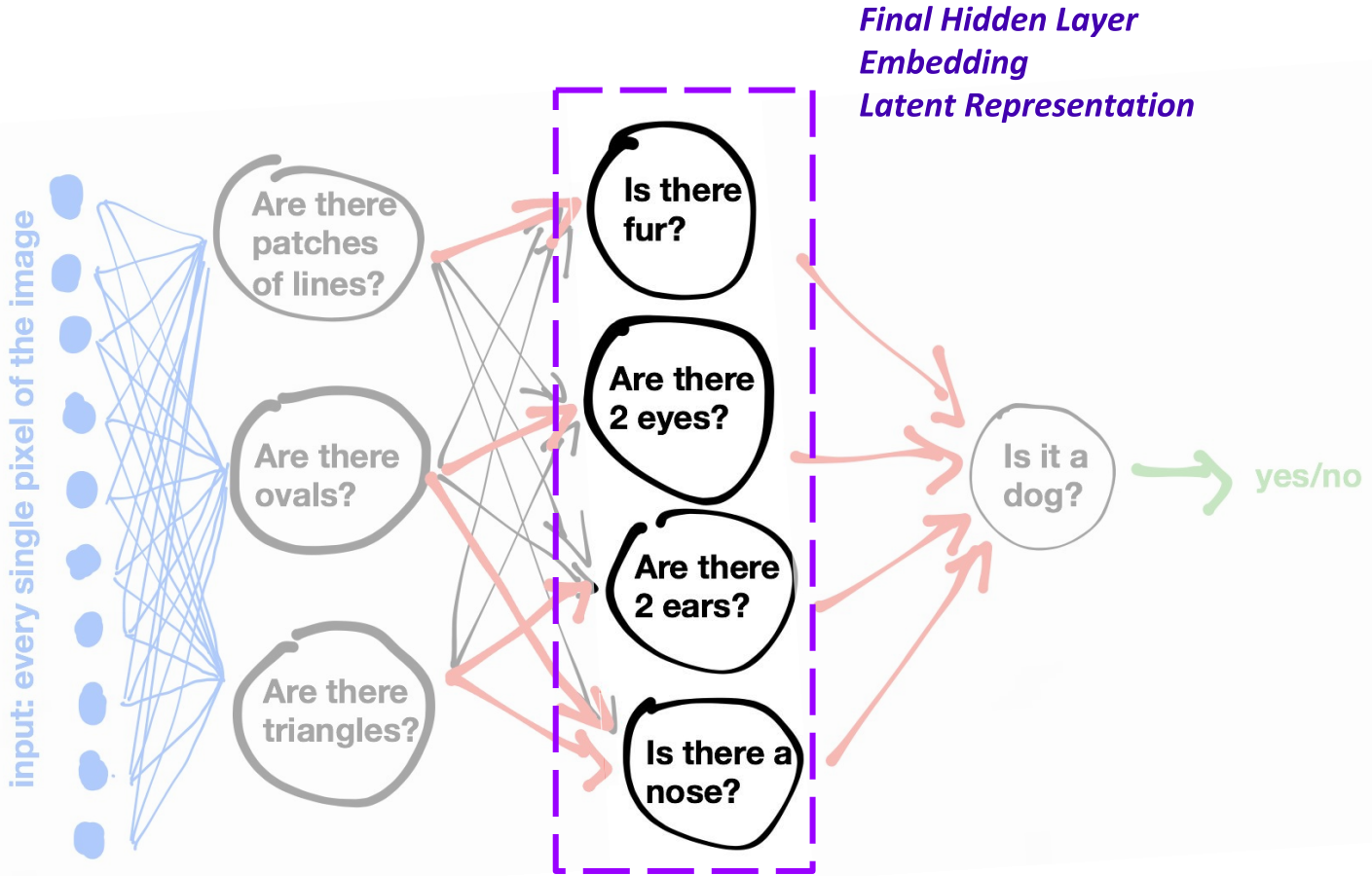


Neural Networks



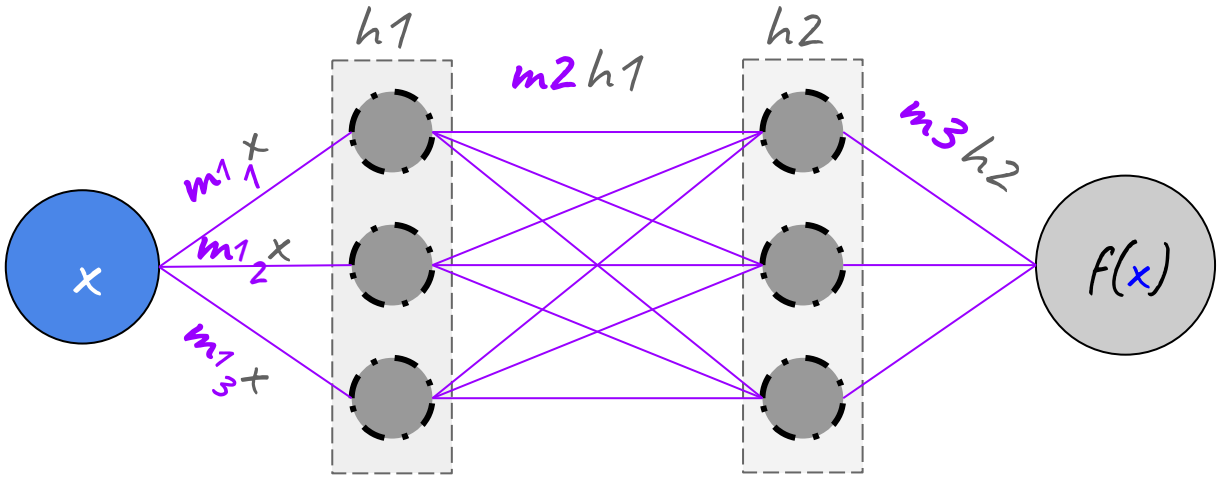


"Understanding Neural Networks Through Deep Visualization" Yosinski et al., 2015


Final Hidden Layer
Embedding
Latent Representation
input: every single pixel of the image
Are there patches of lines?
Are there ovals?
Are there triangles?
Is there fur?
Are there 2 eyes?
Are there 2 ears?
Is there a nose?
Is it a dog?
yes/no

Coding up Neural Nets

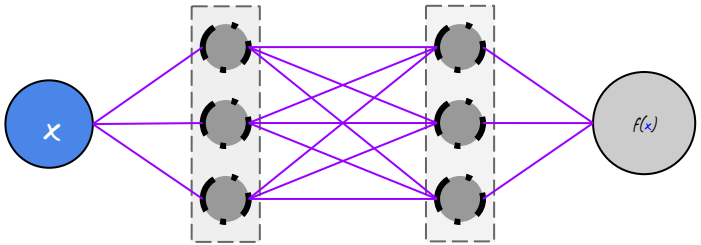
Deep Neural Networks (DNN)



Keras Code (Python)

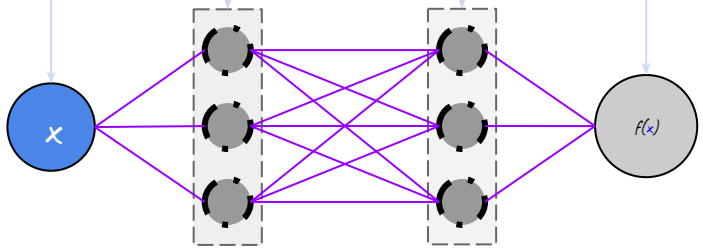
Keras

```python
import numpy as np
from keras.models import Model
from keras.layers import Input, Dense

# load some data
data = np.loadtxt('./inputs.txt')
labels = np.loadtxt('./labels.txt')

inputs = Input(shape=(1,))

hidden_layer_1 = Dense(units=3, activation='relu')(inputs)
hidden_layer_2 = Dense(units=3, activation='relu')(hidden_layer_1)
predictions = Dense(units=1)(hidden_layer_2)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='sgd',
              loss='mean_squared_error',
              metrics=['accuracy'])

model.fit(data, labels)  # starts training
```

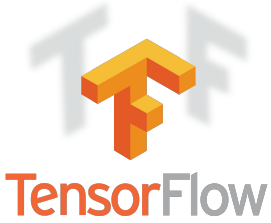


Keras Code (Python)

```python
import numpy as np
from keras.models import Model
from keras.layers import Input, Dense

# load some data
data = np.loadtxt('./inputs.txt')
labels = np.loadtxt('./labels.txt')

inputs = Input(shape=(1,))

hidden_layer_1 = Dense(units=3, activation='relu')(inputs)
hidden_layer_2 = Dense(units=3, activation='relu')(hidden_layer_1)
predictions = Dense(units=1)(hidden_layer_2)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='sgd',
              loss='mean_squared_error',
              metrics=['accuracy'])

model.fit(data, labels)  # starts training
```



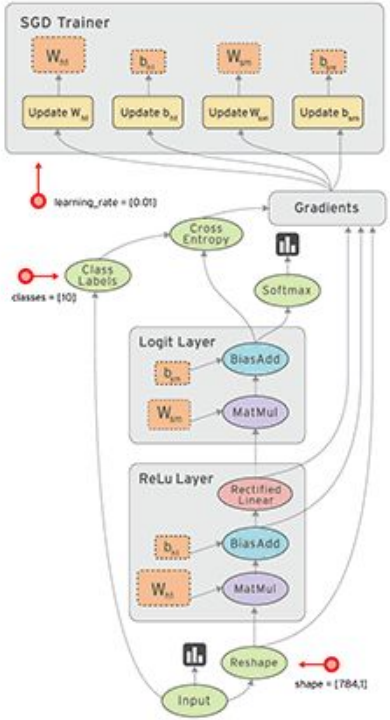
Deep Learning Frameworks

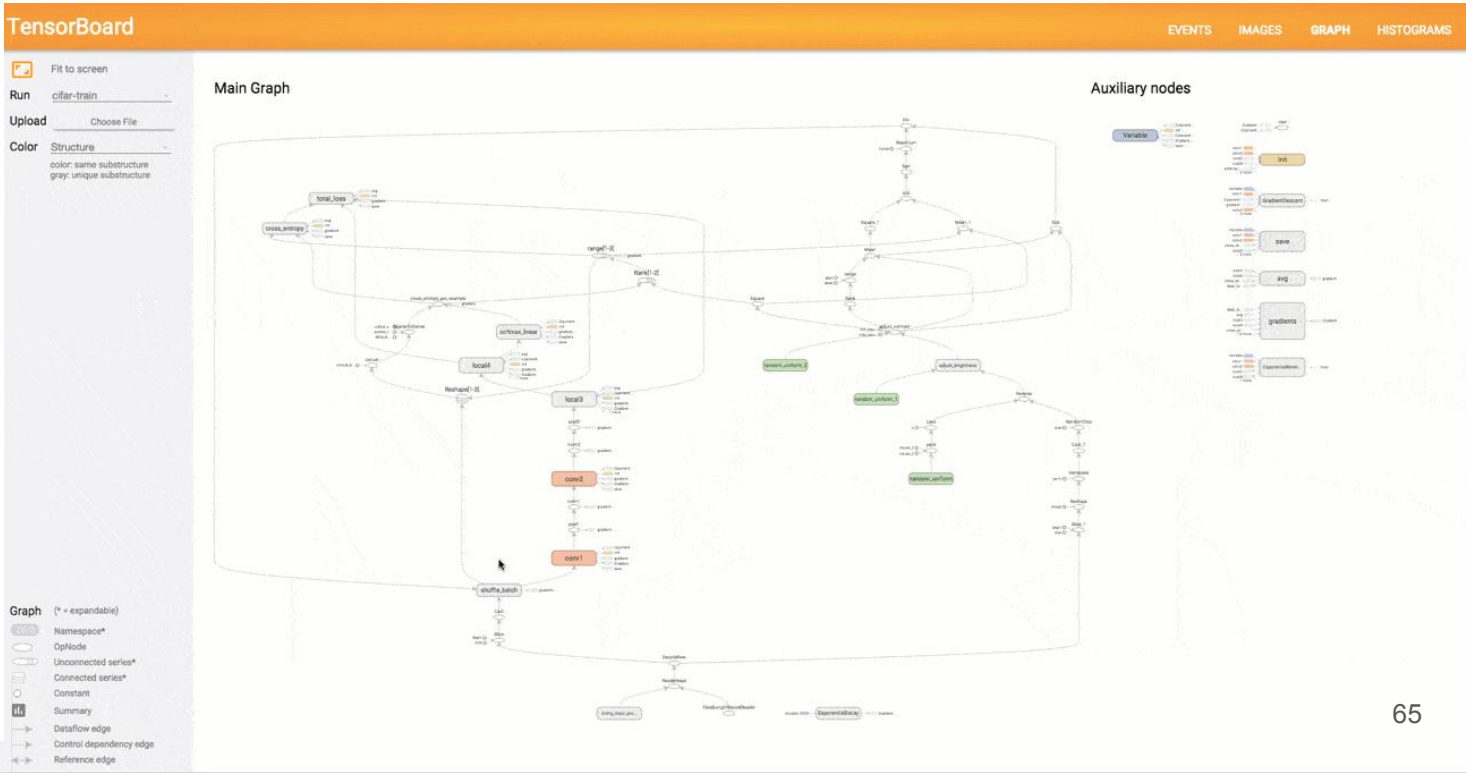


Deep learning libraries: accumulated GitHub metrics as of April 12, 2017

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	209.25	tensorflow/tensorflow
#2:	95.91	BVLC/caffe
#3:	82.36	fchollet/keras
#4:	61.69	dmlc/mxnet
#5:	41.20	Theano/Theano
#6:	35.00	deeplearning4j/deeplearning4j
#7:	32.17	Microsoft/CNTK
#8:	18.73	torch/torch7
#9:	17.29	baidu/paddle
#10:	15.14	pytorch/pytorch
#11:	14.22	pfnet/chainer
#12:	14.05	NVIDIA/DIGITS
#13:	12.62	tflearn/tflearn

Tensorboard



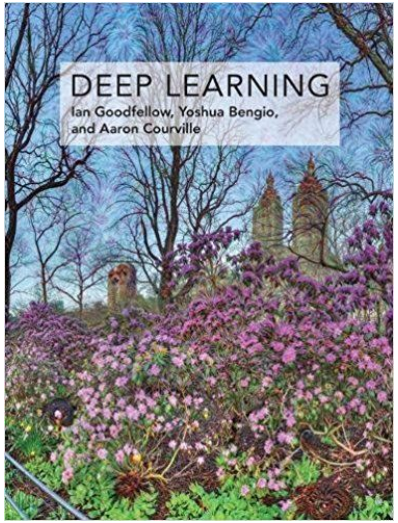


Resources/Next Steps

Deep Learning Course by Andrew Ng:
https://www.coursera.org/learn/neural-networks-deep-learning

Andrej Karpathy's "Hacker's guide to Neural Networks"
http://karpathy.github.io/neuralnets/

The Deep Learning Book
http://www.deeplearningbook.org/

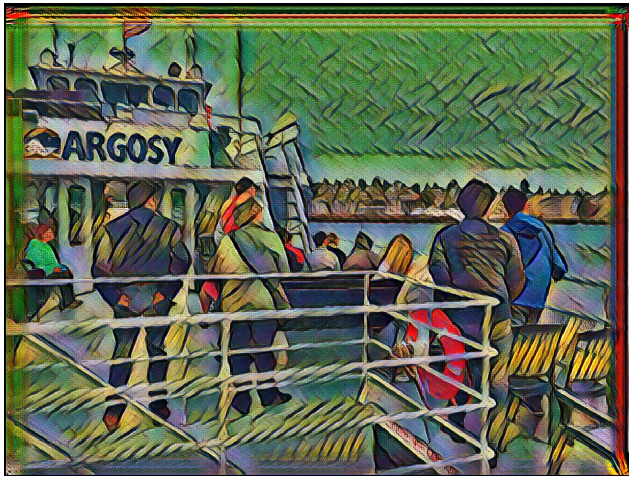


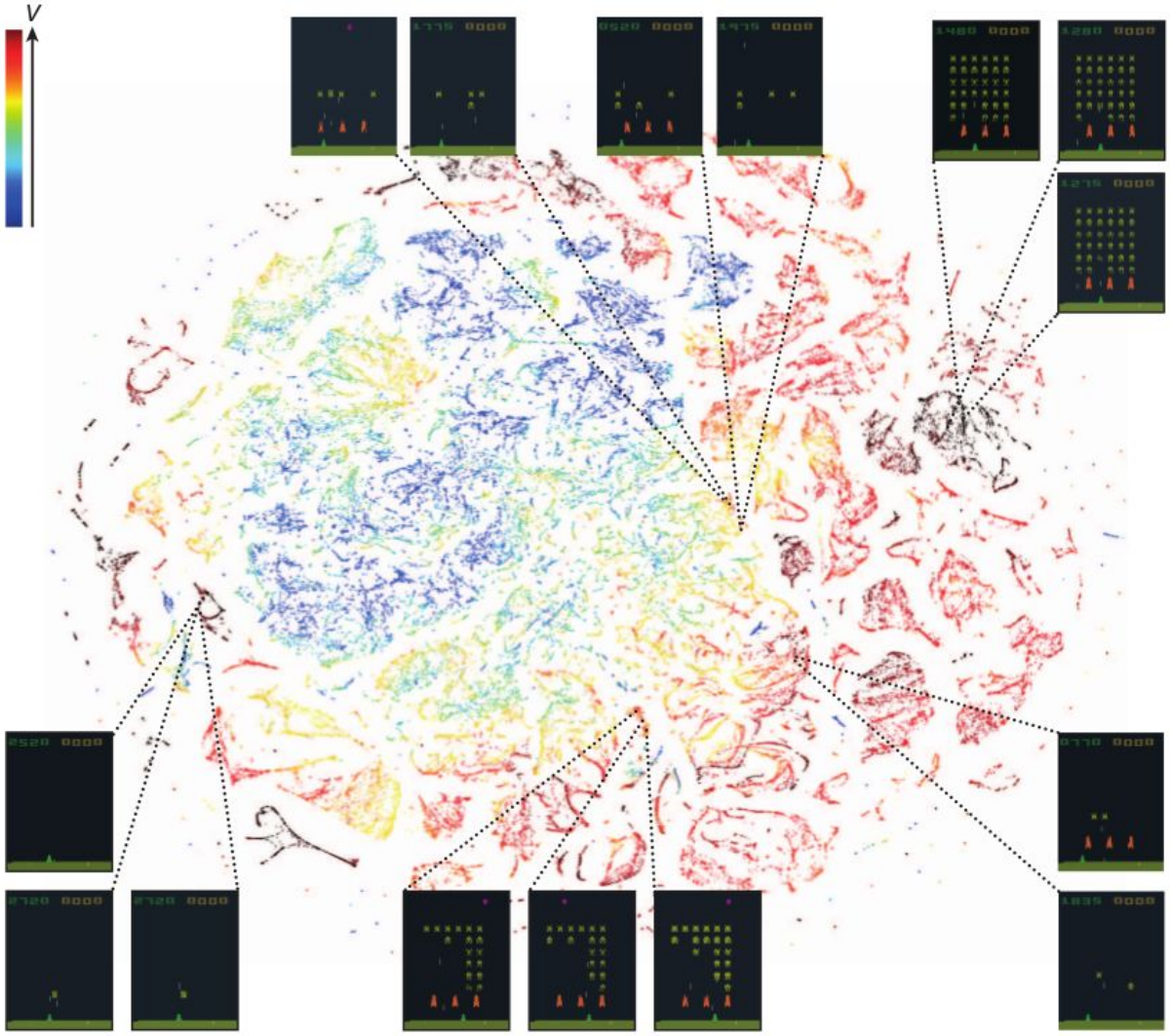
End: Q&A ?

Other stuff you can do..

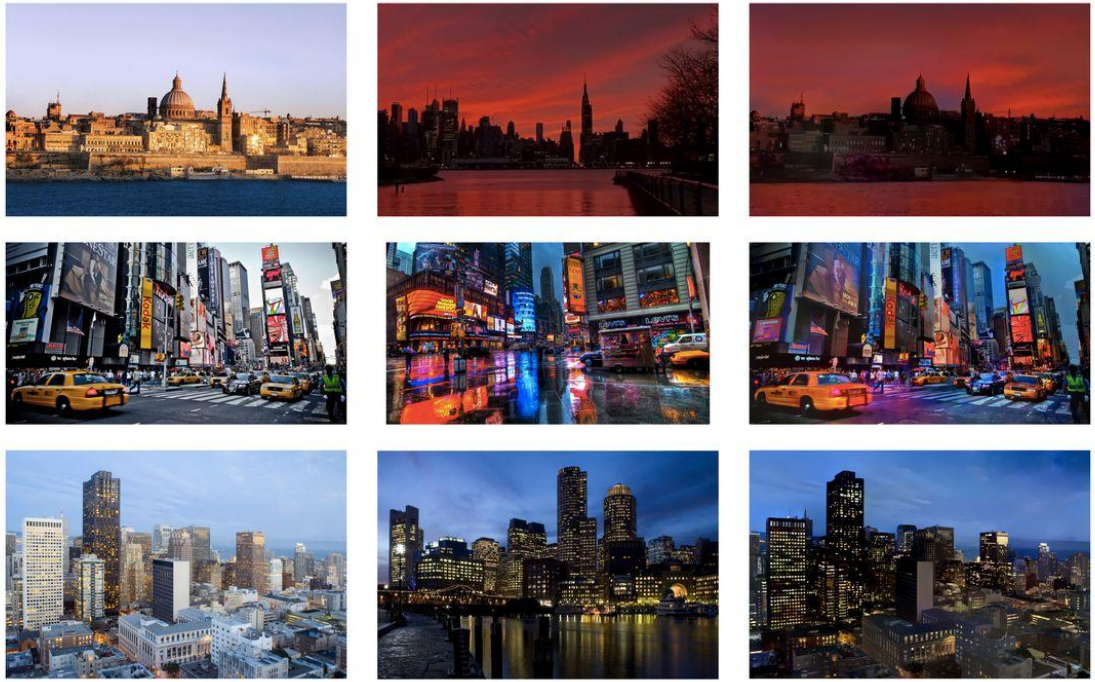
Style Transfer

=




V

Style Transfer



Original photo (raw input) ~ Reference photo (latent representation) = Result