Deep Learning
MSDS 631

Introduction to this course and Deep Learning

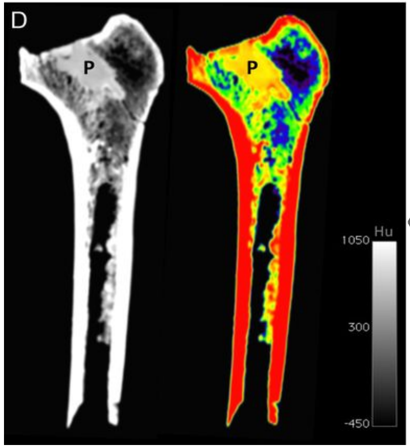
Michael Ruddy

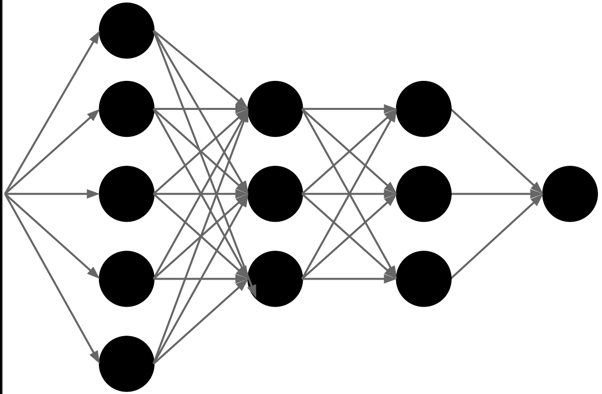
Overview

- What we will cover and why
- How this course will operate
- What is Deep Learning really?
- How do I train a neural network in Pytorch?

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.
- Models are usually based on **artificial neural networks** (ANNs or NNs).
    - Deep here refers to ANNs with many *layers*.



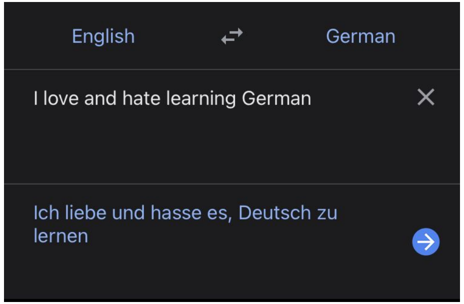


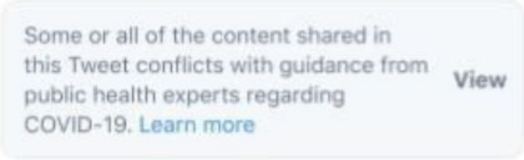
Why Deep Learning?

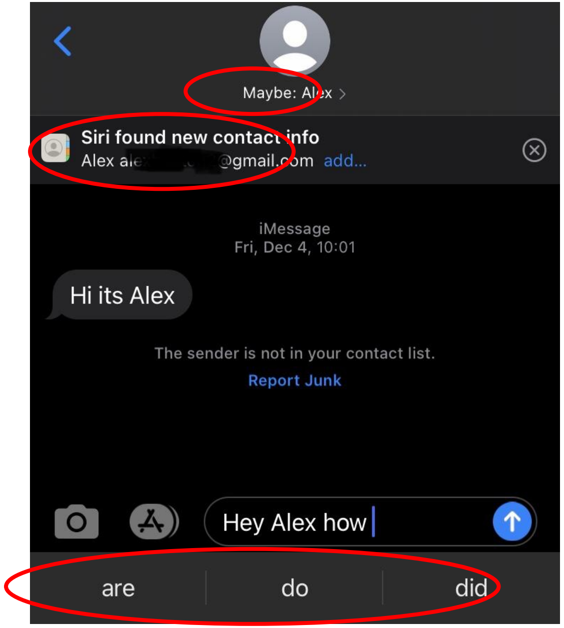
- Explosion in amount of data available and in computing power
    - Neural networks are often complicated models with many parameters, necessitating a lot of data and a lot of computing power
- Increasingly important aspect of data science
    - Image Science
    - Natural Language Processing

Why Deep Learning?

- Explosion in amount of data available and in computing power
  - Neural networks are often complicated models with many parameters, necessitating a lot of data and a lot of computing power
- Increasingly important aspect of data science
  - Image Science
  - Natural Language Processing

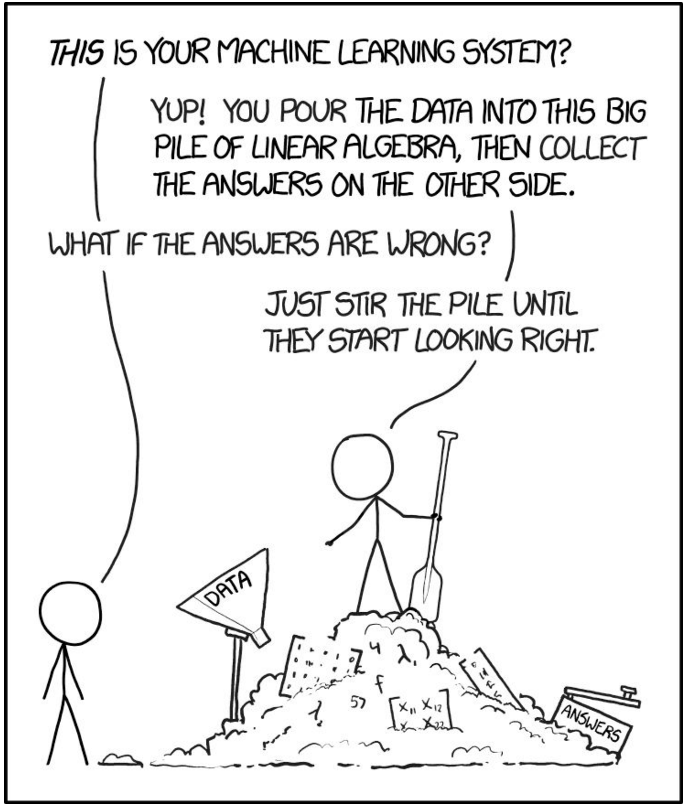






What we will learn

- How to effectively train neural networks using PyTorch
  - Learning rates, batch norm, regularization, data augmentation, transfer learning, ...



What we will learn

- How to effectively train neural networks using PyTorch
    - Learning rates, batch norm, regularization, data augmentation, transfer learning, ...
- Image data
    - Common imaging tasks (classification, segmentation, ...)
    - Architectures for spatial data: CNNs
- Text data
    - Common NLP tasks (classification, comparison, ...)
    - Architectures for sequences/text (RNNs, Attention)

What we will learn

- How to effectively train neural networks using PyTorch
  - Learning rates, batch norm, regularization, data augmentation, transfer learning, ...
- Image data
  - Common imaging tasks (classification, segmentation, ...)
  - Architectures for spatial data: CNNs
- Text data
  - Common NLP tasks (classification, comparison, ...)
  - Architectures for sequences/text (RNNs, Attention)
- Maybe, if time/interest
  - GANs (style transfer, synthetic images/text)
  - Something else?

Disclaimer

- Too much in Deep Learning to go over everything!
- Goals
  - Provide a good foundation for whatever most interests you
  - Good mix of conceptual understanding and implementation in PyTorch

Disclaimer

- Too much in Deep Learning to go over everything!
- Goals
    - Provide a good foundation for whatever most interests you
    - Good mix of conceptual understanding and implementation in PyTorch
- Changes every year (sometimes multiple times)
    - What techniques work the best
    - Why a certain technique works
    - What is really going on in Deep Learning
    - What is the hot stuff everyone wants to do

Disclaimer

- Too much in Deep Learning to go over everything!
- Goals
  - Provide a good foundation for whatever most interests you
  - Good mix of conceptual understanding and implementation in PyTorch
- Changes every year (sometimes multiple times)
  - What techniques work the best
  - Why a certain technique works
  - What is really going on in Deep Learning
  - What is the hot stuff everyone wants to do
- Don’t accept a research paper as the absolute truth

How this class will run: What you'll do

- On Tuesday/Thursday: **Live** Lecture
  - Quizzes graded for completion (Polls)
  - Active participation (answer if called on)
  - In-class work (group optional; determined before class)

How this class will run: What you'll do

- On Tuesday/Thursday: **Live** Lecture
  - Quizzes graded for completion (Polls)
  - Active participation (answer if called on)
  - In-class work (group optional; determined before class)
- Due Wednesday/Friday at Midnight: Labs
  - Graded "completion"-ish

How this class will run: What you'll do

- On Tuesday/Thursday: **Live** Lecture
  - Quizzes graded for completion (Polls)
  - Active participation (answer if called on)
  - In-class work (group optional; determined before class)
- Due Wednesday/Friday at Midnight: Labs
  - Graded "completion"-ish
- Due Periodically on Friday: Homework
  - About 3-4 assignments

How this class will run: What you'll do

- On Tuesday/Thursday: **Live** Lecture
  - Quizzes graded for completion (Polls)
  - Active participation (answer if called on)
  - In-class work (group optional; determined before class)
- Due Wednesday/Friday at Midnight: Labs
  - Graded "completion"-ish
- Due Periodically on Friday: Homework
  - About 3-4 assignments
- July 20th/August 12th: Final Project Dates
  - Description Presentation, Final Presentation
  - Presentations, GitHub
  - Group size: 2

How this class will run: Assessment

- Professionalism: 20%
  - Active Participation in Class/Quizzes
  - Respect your classmates, MSDS faculty, and any guests
- Assignments: 30%
  - Late Policy: 10% reduction in score per day late
  - May include Quizzes
- Labs: 20%
  - Late Policy: half credit if received less than one week late, zero otherwise
- Final Project: 30%
  - Description Presentation: 10%

How this class will run: Assessment

- Professionalism: 20%
  - Active Participation in Class/Quizzes
  - Respect your classmates, MSDS faculty, and any guests
- Assignments: 30%
  - Late Policy: 10% reduction in score per day late
  - May include Quizzes
- Labs: 20%
  - Late Policy: half credit if received less than one week late, zero otherwise
- Final Project: 30%
  - Description Presentation: 10%
- Important: Valid excuses communicated timely will be honored

How this class will run: Resources

- Office Hours
  - Tuesdays/Thursdays 3pm - 4pm
  - Online office hours
- Slack Channel
- Other MSDS faculty
- At some point (TBD)
  - In-person working sessions (maybe Wednesday or Friday)

How this class will run: Final Project

- Goal: Evidence that you are capable of utilizing deep learning to solve a task with real world data
    - Complement your existing work!
    - Get creative!
- Ideas/Starting Points
    - Kaggle Competitions (Active and Past)
        - https://www.kaggle.com/getting-started/16221
    - Inspiration here for images and text
    - Implement a Research Paper (very valuable!)
    - Generate Images/Text (specific categories, think Dr. Interian's recipes)
    - More inspiration

How this class will run

- Any Questions?

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

Why do you think this is a cat?

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

High Level Features

- Two Ears
- Two Eyes
- Whiskers
- Looks fluffy

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

High Level Features

- Two Ears
- Two Eyes
- Whiskers
- Looks fluffy

Collectively: a representation of the image

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

$$\rightarrow \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ .5 & .75 & 1 & \dots & .25 \\ \vdots & \vdots & \vdots & & \vdots \\ .333 & 0 & 1 & \dots & 0 \end{bmatrix}$$

Two Ears = ???

Two Eyes = ???

Whiskers = ???

Looks fluffy = ???

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

Low-level geometric Features

- Edge detection
- Noisiness
- Blob detection

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ .5 & .75 & 1 & \dots & .25 \\ \vdots & \vdots & \vdots & & \vdots \\ .333 & 0 & 1 & \dots & 0 \end{bmatrix}$$

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

DL high-level features

- learned from data
- Constructed from learned low-level features
- Usually NOT interpretable

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ .5 & .75 & 1 & \dots & .25 \\ \vdots & \vdots & \vdots & & \vdots \\ .333 & 0 & 1 & \dots & 0 \end{bmatrix}$$

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.

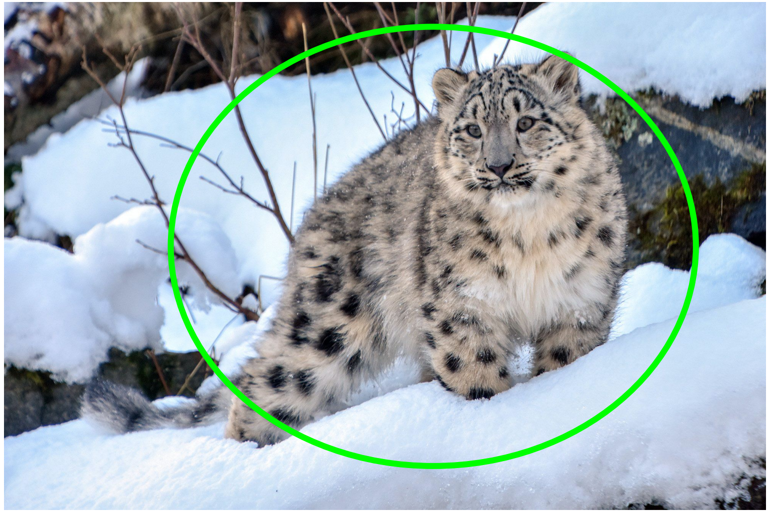
DL high-level features

- Usually NOT interpretable
- Only as good as your data...

Is this a snow leopard or regular leopard?

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.



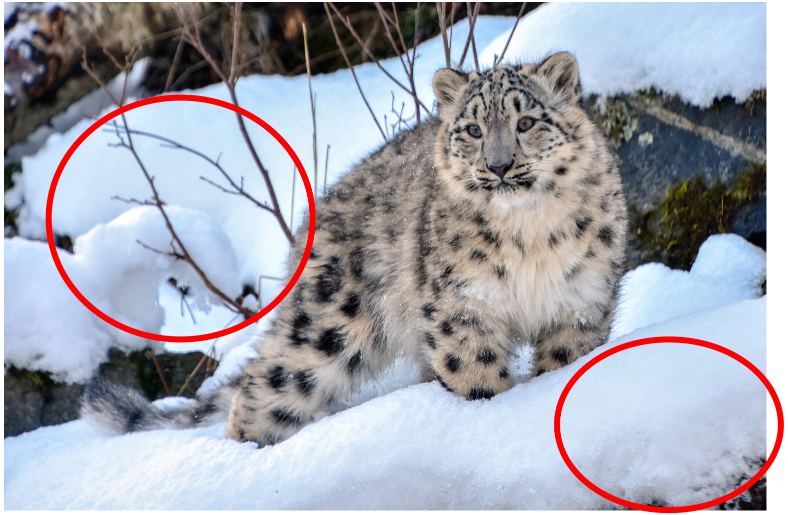
DL high-level features

- Usually NOT interpretable
- Only as good as your data...

Is this a snow leopard or regular leopard?

What is Deep Learning?

- **Deep Learning** (DL) is a subset of Machine Learning where algorithms perform tasks by extracting *high-level features* from datasets that are usually very large and unstructured.



DL high-level features

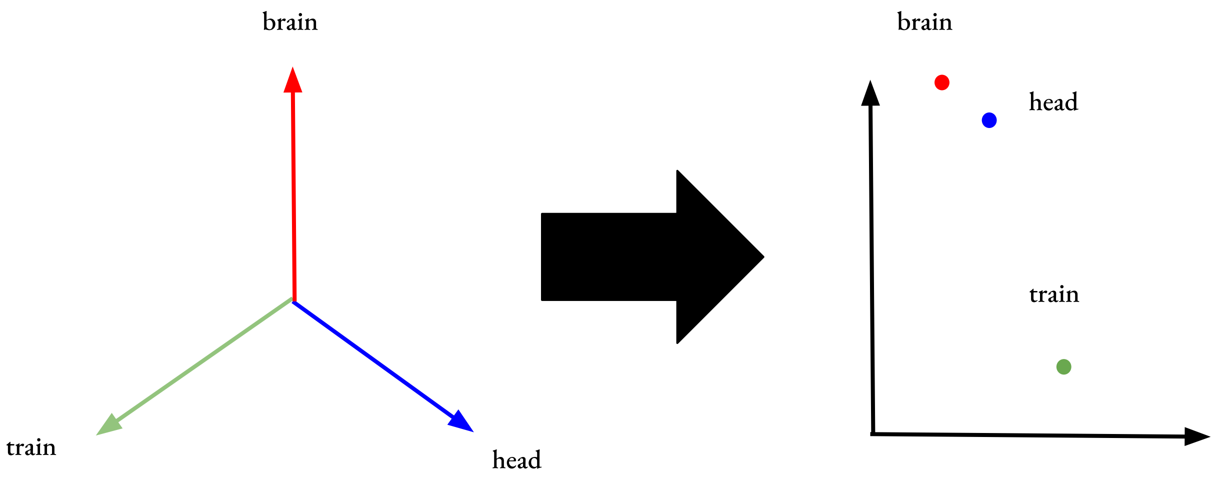
- Usually NOT interpretable
- Only as good as your data...

Is this a snow leopard or regular leopard?

What is Deep Learning?

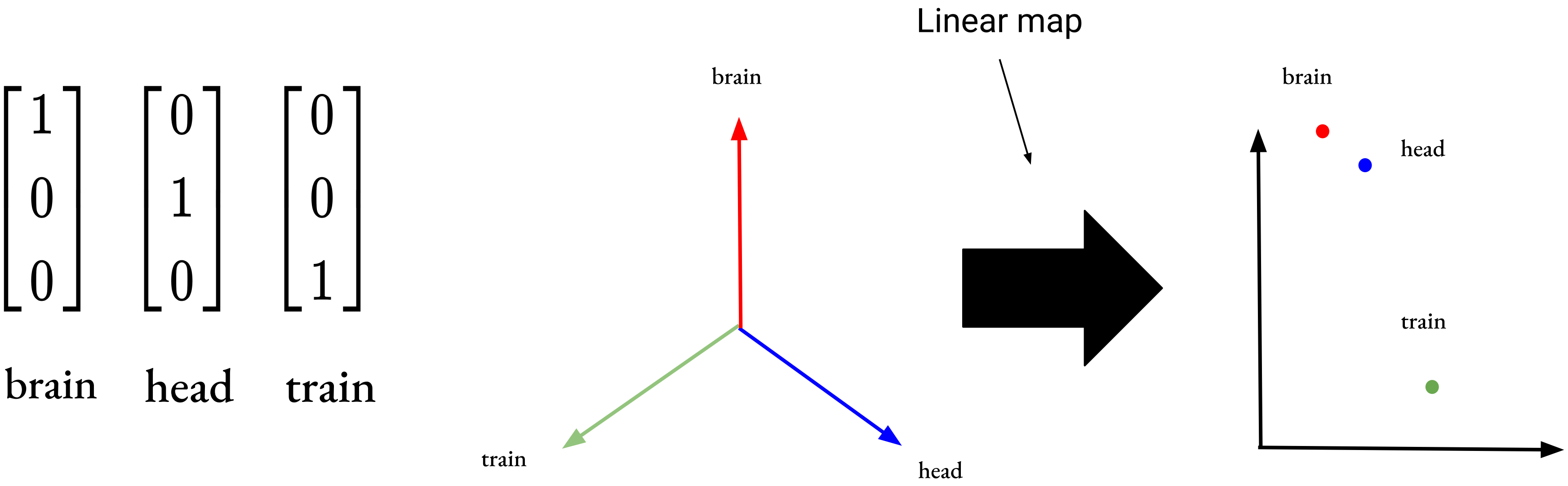
- One-Hot Embeddings -> Word Embeddings
- Unstructured data -> Represented by meaningful features
- Simple linear function

$$\begin{array}{ccc} \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \\ \text{brain} & \text{head} & \text{train} \end{array}$$



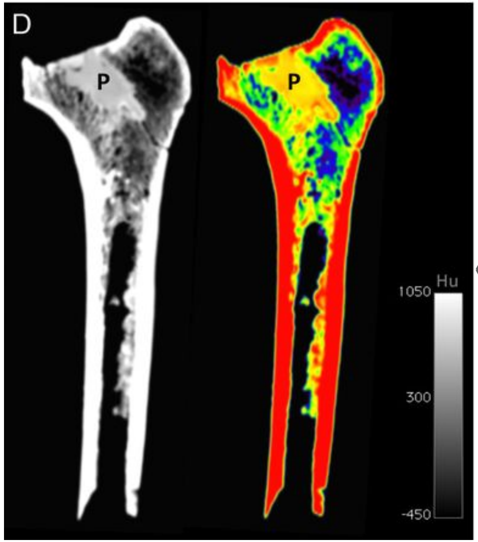
What is Deep Learning?

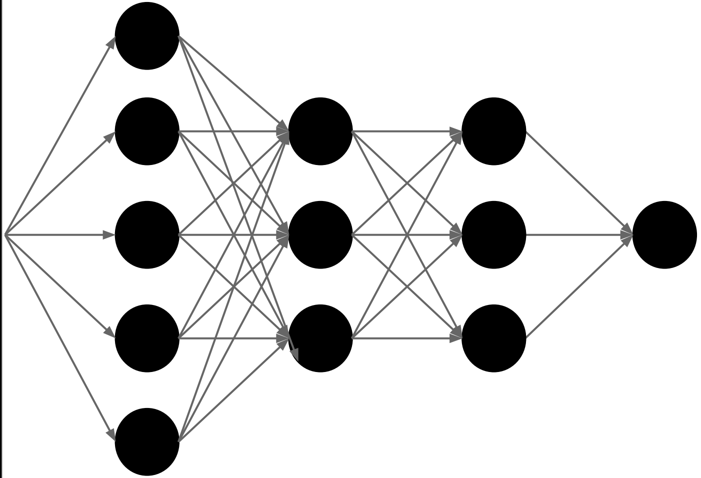
- One-Hot Embeddings -> Word Embeddings
- Unstructured data -> Represented by meaningful features
- Simple linear function



What is Deep Learning?

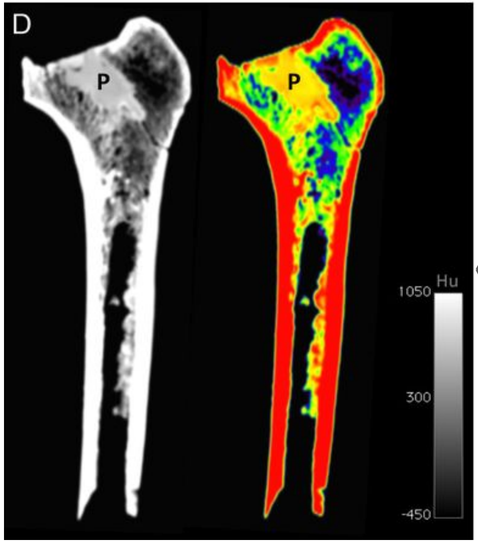
- Array -> Representation by “meaningful” features
- Simple linear functions stacked on top of each other

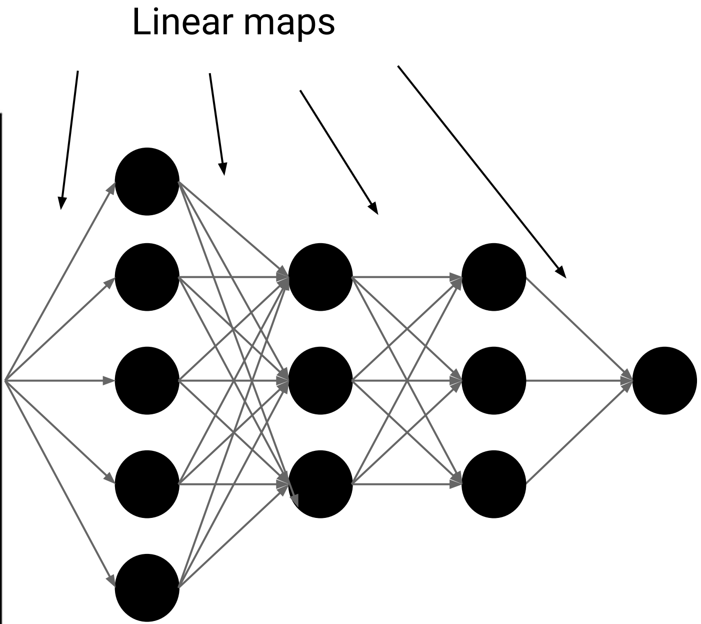




What is Deep Learning?

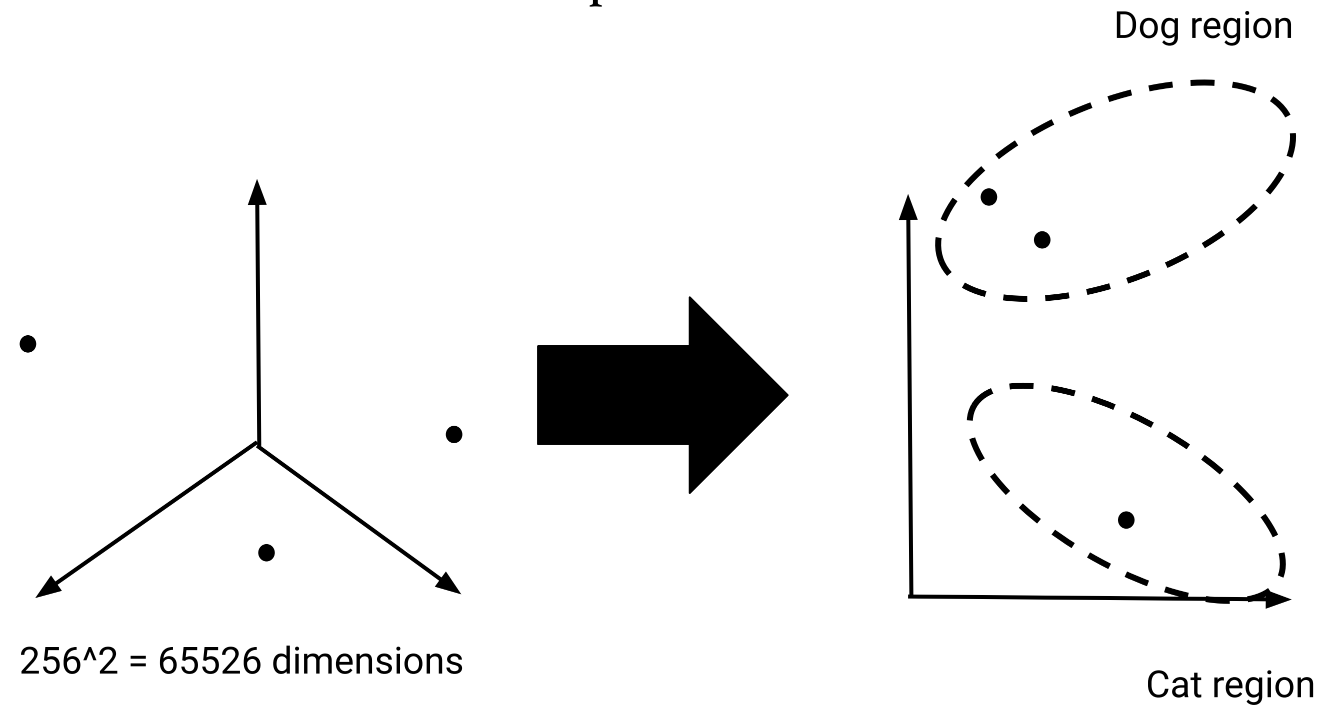
- Array -> Representation by “meaningful” features
- Simple linear functions stacked on top of each other





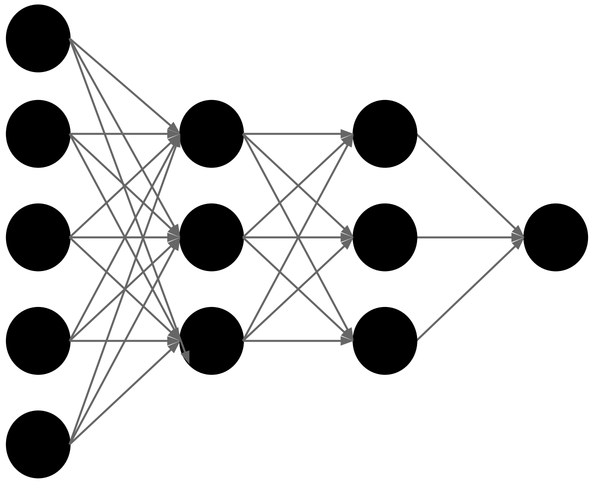
What is Deep Learning?

- Array -> Representation by “meaningful” features
- Simple linear functions stacked on top of each other



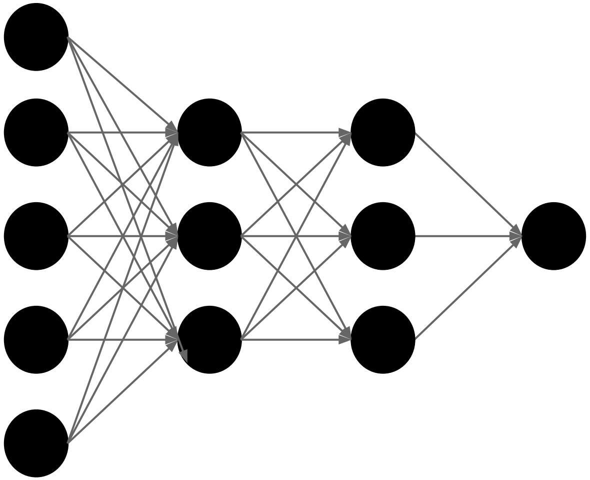
Neural Network

- Word embedding: simple linear mapping
- Neural Network: stack linear functions one after the other (layers)
- Any function can be approximated this way (rigorous!)



Neural Network

- Word embedding: simple linear mapping
- Neural Network: stack linear functions one after the other (layers)
- Any function can be approximated this way (rigorous!)
- Idea: create low level features in early layers to create high level features in later layers (none of these are necessarily interpretable!)



Neural Network

- Training a Neural Network: just a supervised learning problem!

Neural Network

- Training a Neural Network: just a supervised learning problem!
- Labelled data: $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$
- Model and parameters: $F(x; \theta)$
- Loss Function: $\mathcal{L}(F(x; \theta), y)$

Neural Network

- Training a Neural Network: just a supervised learning problem!
- Labelled data: $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$
- Model and parameters: $F(x; \theta)$
- Loss Function: $\mathcal{L}(F(x; \theta), y)$

Find parameters θ that minimize

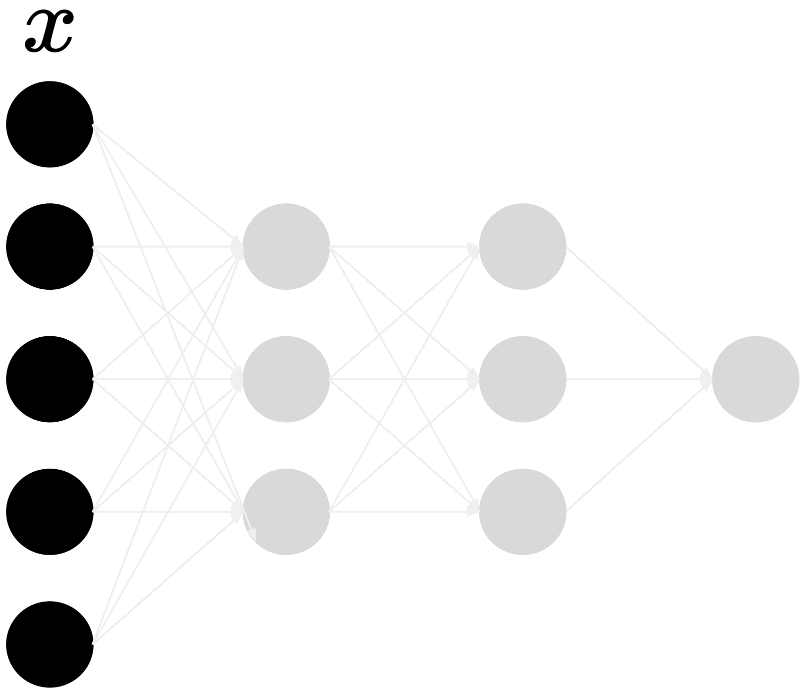
$$\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(F(x^{(i)}; \theta), y^{(i)}\right)$$

Neural Network

- What's so special about using a Neural Network for $F(x; \theta)$?

Neural Network

- What's so special about using a Neural Network for $F(x;\theta)$?

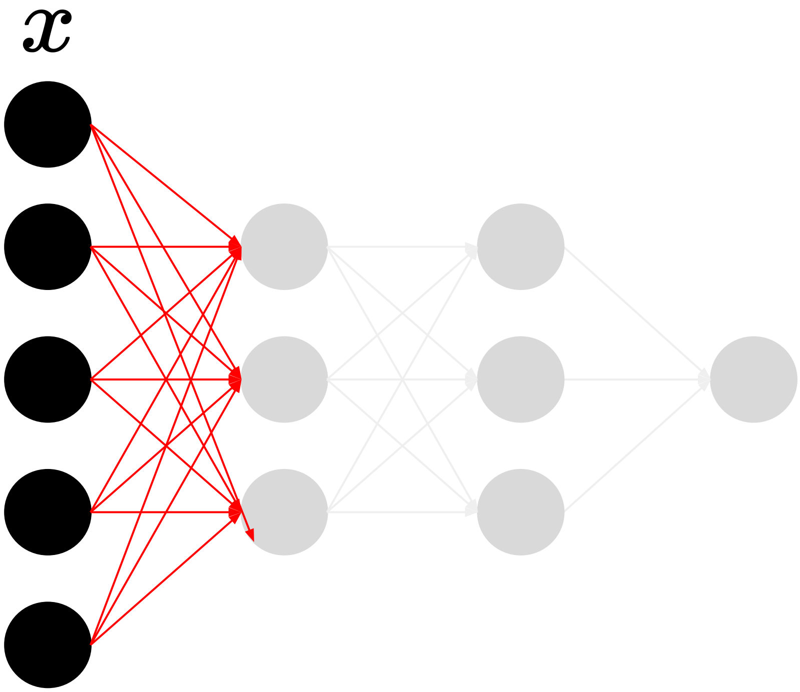


Input data: 5D vector
(Input layer)

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Neural Network

- First step: linear transformation

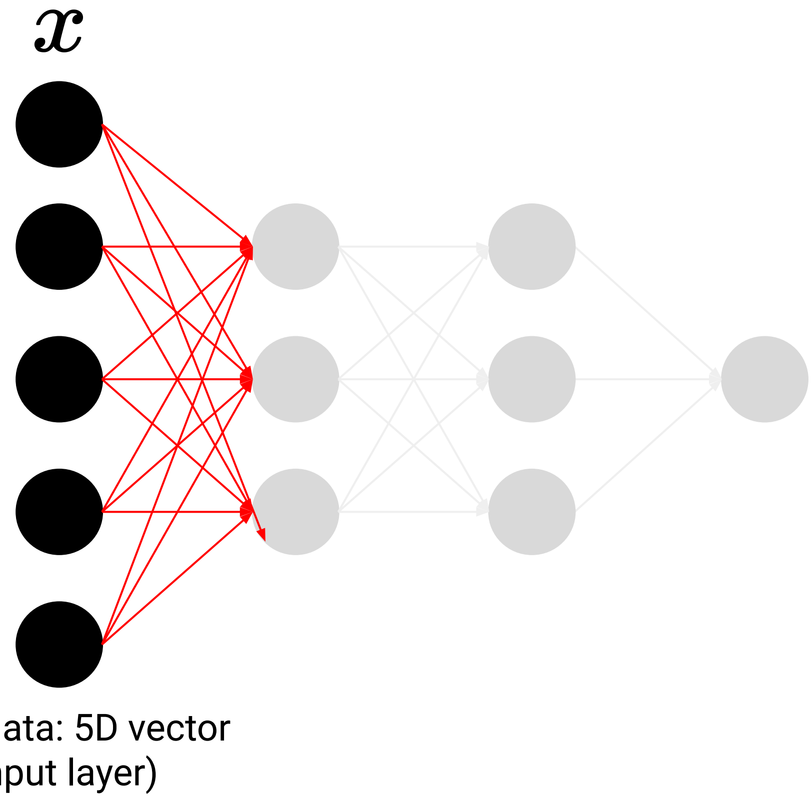


Input data: 5D vector
(Input layer)

$$W^{[1]} = ?$$

Neural Network

- First step: linear transformation



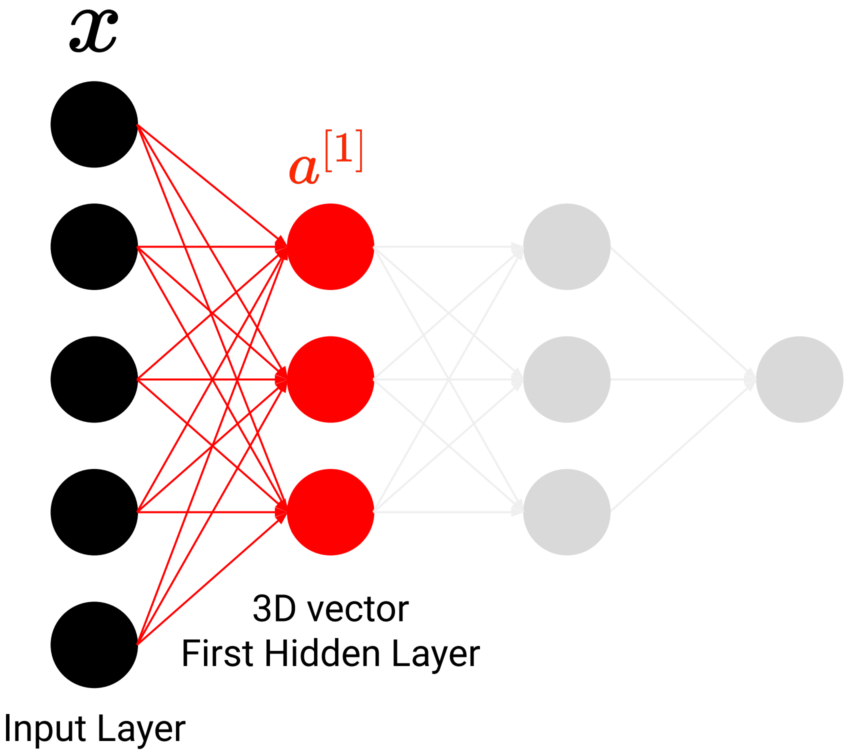
$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} & w_{14}^{[1]} & w_{15}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} & w_{24}^{[1]} & w_{25}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} & w_{34}^{[1]} & w_{35}^{[1]} \end{bmatrix}$$

$$b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}$$

$$W^{[1]}x + b^{[1]}$$

Neural Network

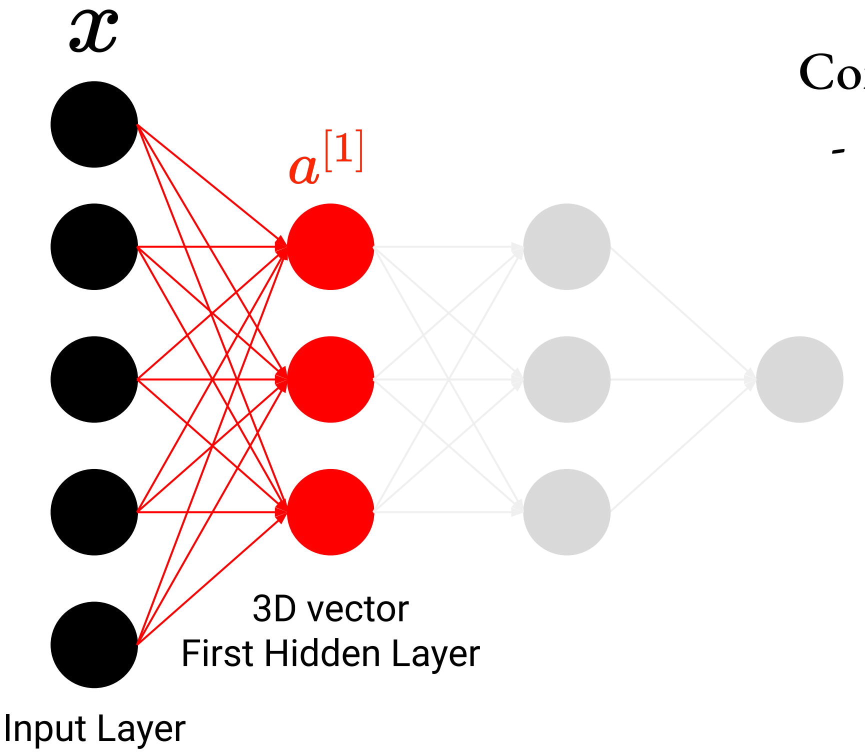
- Second step: nonlinearity (activation function)



$$a^{[1]} = h(W^{[1]}x + b^{[1]})$$

Neural Network

- Second step: nonlinearity (activation function)



Commonly used activation functions

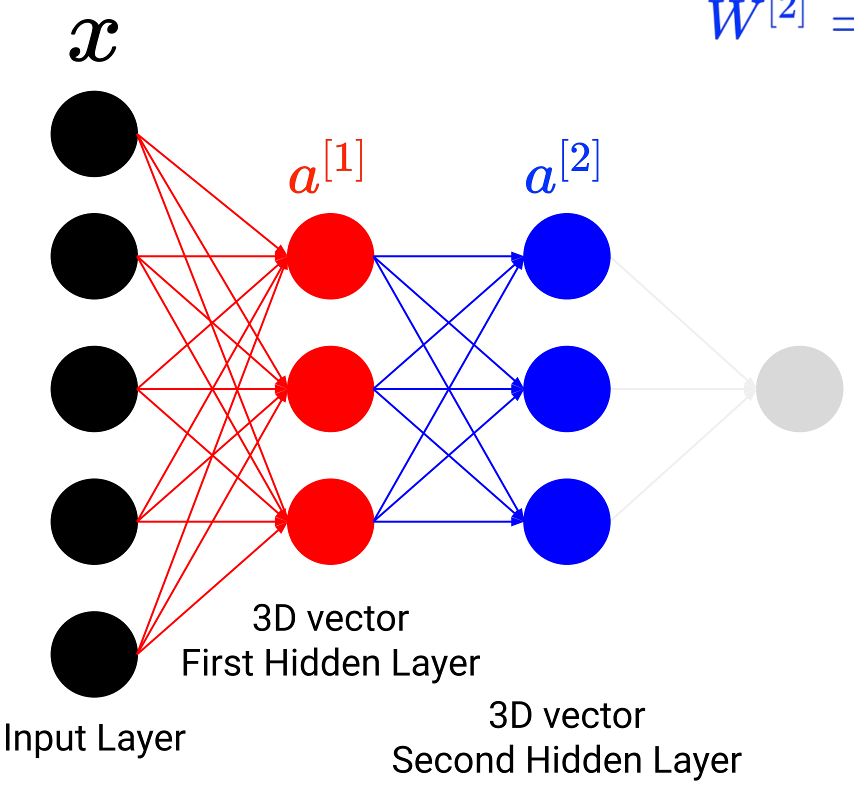
- ReLU, Sigmoid, Leaky ReLU, Tanh

$$a^{[1]} = h(W^{[1]}x + b^{[1]})$$

Neural Network

- Let's do it again!

$$W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{2} & w_{13}^{[2]} \\ w_{21}^{[2]} & w_{22}^{2} & w_{23}^{[2]} \\ w_{31}^{[2]} & w_{32}^{2} & w_{33}^{[2]} \end{bmatrix} \qquad b^{[2]} = \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \end{bmatrix}$$

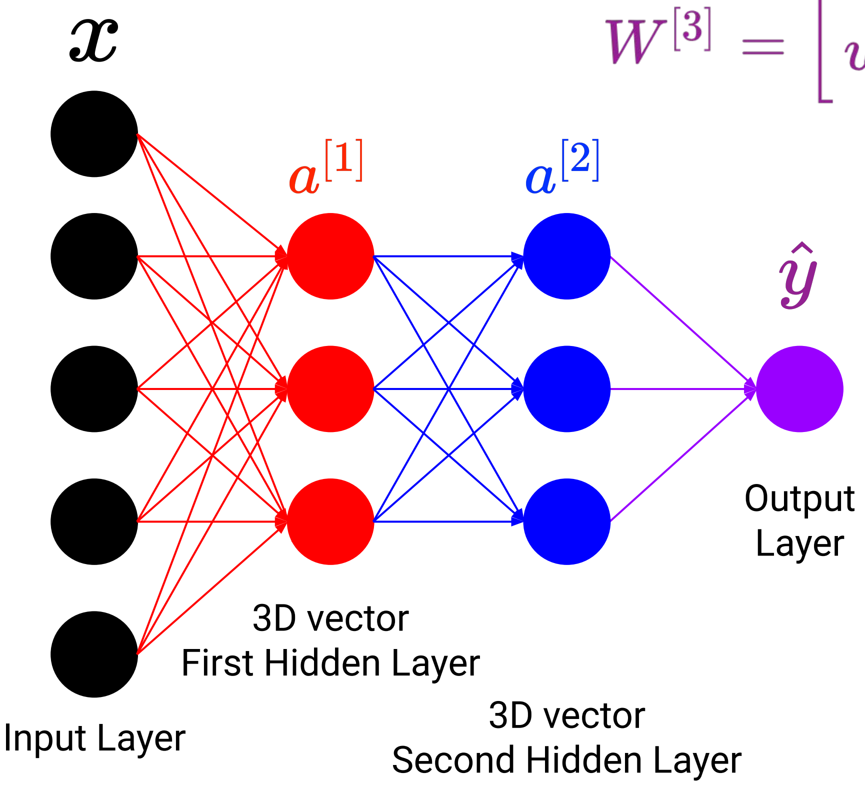


$$a^{[2]} = h(W^{[2]} a^{[1]} + b^{[2]})$$

$$a^{[1]} = h(W^{[1]} x + b^{[1]})$$

Neural Network

- Final Output



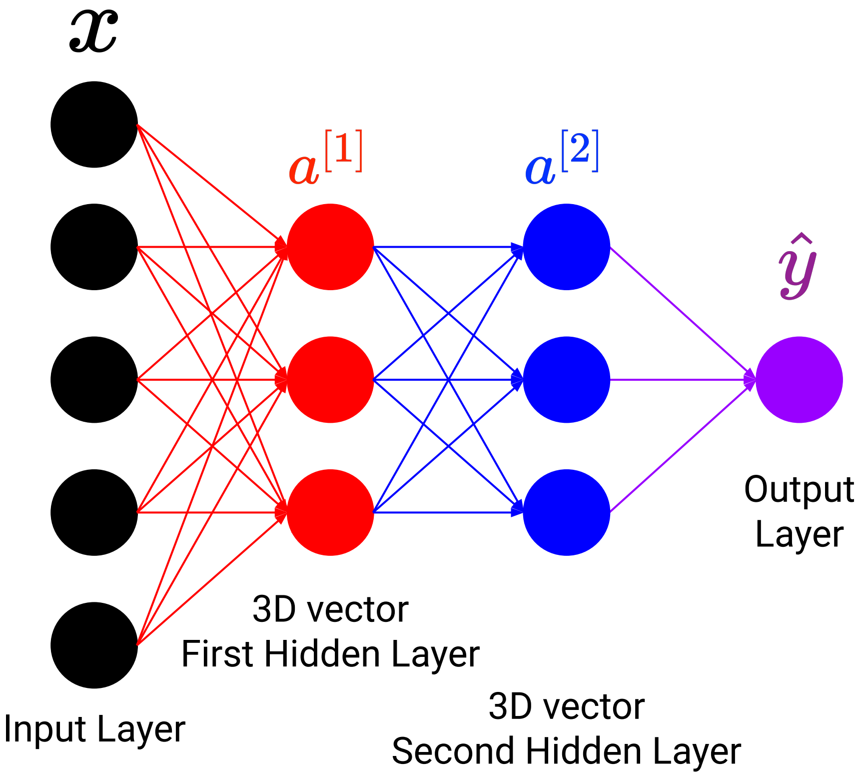
$$W^{[3]} = \begin{bmatrix} w_{11}^{[3]} & w_{12}^{[3]} & w_{13}^{[3]} \end{bmatrix} \qquad b^{[3]} = \begin{bmatrix} b_1^{[3]} \end{bmatrix}$$

$$F(x;\theta) = h(W^{[3]} a^{[2]} + b^{[3]})$$

$$a^{[1]} = h(W^{[1]} x + b^{[1]})$$
$$a^{[2]} = h(W^{[2]} a^{[1]} + b^{[2]})$$

Neural Network

- 3-layer "Feed-Forward" Neural Network



$$a^{[1]} = h(W^{[1]}x + b^{[1]})$$
$$a^{[2]} = h(W^{[2]}a^{[1]} + b^{[2]})$$
$$F(x;\theta) = h(W^{[3]}a^{[2]} + b^{[3]})$$

"Deeper" = more layers

Neural Networks

- All architectures are, at their core, linearity + nonlinearity successively
    - Easy to compute gradient this way (chain rule)
- In theory all you need is a FF NN
- In practice, intuition, experience, and understanding the problem are needed to make NNs work effectively
- We will not learn every architecture under the sun (boring)

Neural Networks

- All architectures are, at their core, linearity + nonlinearity successively
    - Easy to compute gradient this way (chain rule)
- In theory all you need is a FF NN
- In practice, intuition, experience, and understanding the problem are needed to make NNs work effectively
- We will not learn every architecture under the sun (boring)

Let's do this in PyTorch!