The Physics of Neural Networks

Using physics to quantify “goodness”

Hannah Day

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

What is a neural network?

(Spoiler alert: it's just fancy regression!)

- Give it some labeled data

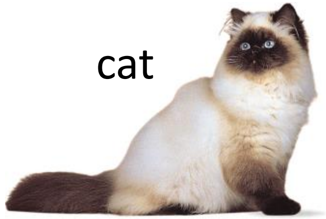
dog

cat

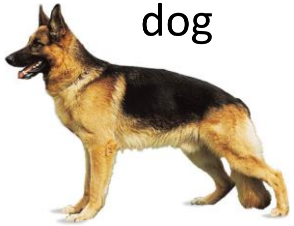
cat

dog

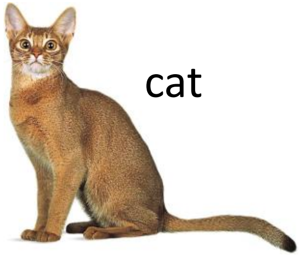
cat



dog



cat



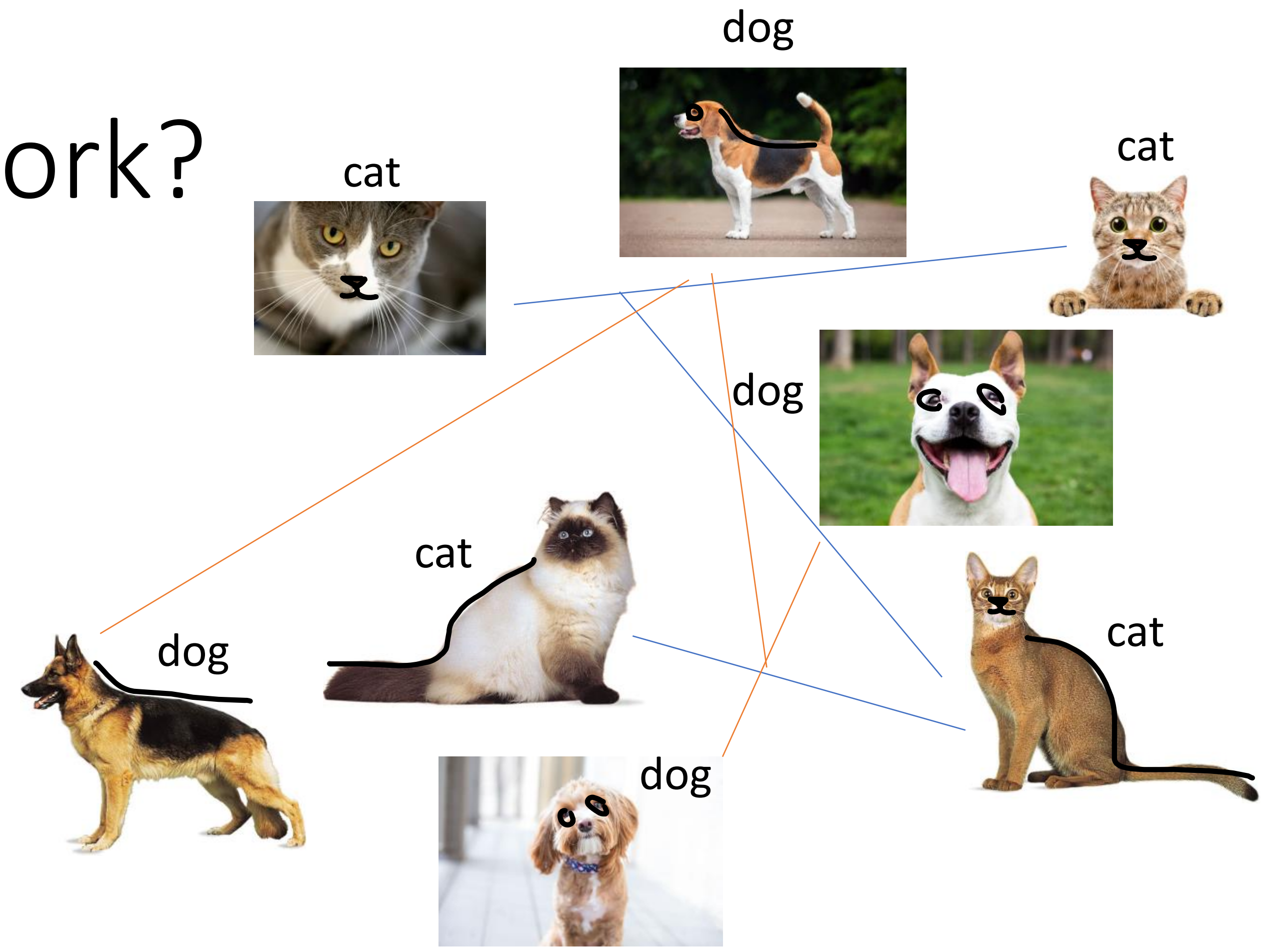
dog

What is a neural network?

(Spoiler alert: it’s just fancy regression!)

- Give it some labeled data

- Network learns patterns



What is a neural network?

(Spoiler alert: it's just fancy regression!)

- Give it some labeled data

- Network learns patterns

- Give it some unlabeled data

What is a neural network?

(Spoiler alert: it’s just fancy regression!)

- Give it some labeled data
- Network learns patterns
- Give it some unlabeled data
- Network assigns labels with some percent confidence

90% cat

87% dog

72% dog

81% cat

What is a neural network?

(Spoiler alert: it’s just fancy regression!)

- Give it some labeled data
- Network learns patterns
- Give it some unlabeled data
- Network assigns labels with some percent confidence
- Adjust initial network parameters to improve accuracy

98% cat

95% dog

89% dog

93% cat

What is a neural network?

(Spoiler alert: it's just fancy regression!)

- Give it some labeled data

95% dog

98% cat

- Network learns patterns

89% dog

- Give it some unlabeled data

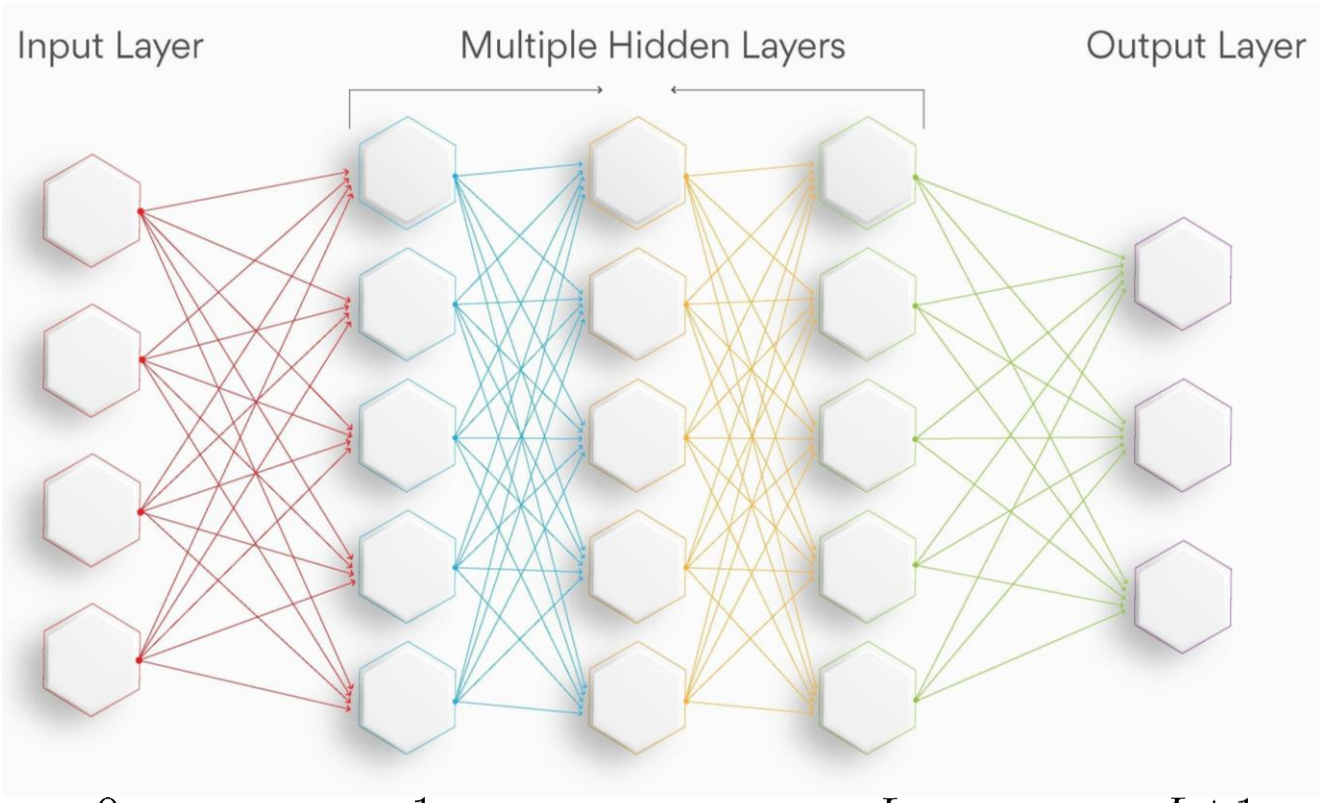
93% cat

- Network assigns labels with some percent confidence

We want to avoid this step

- ~~Adjust initial network parameters to improve accuracy~~

Basic neural network terminology

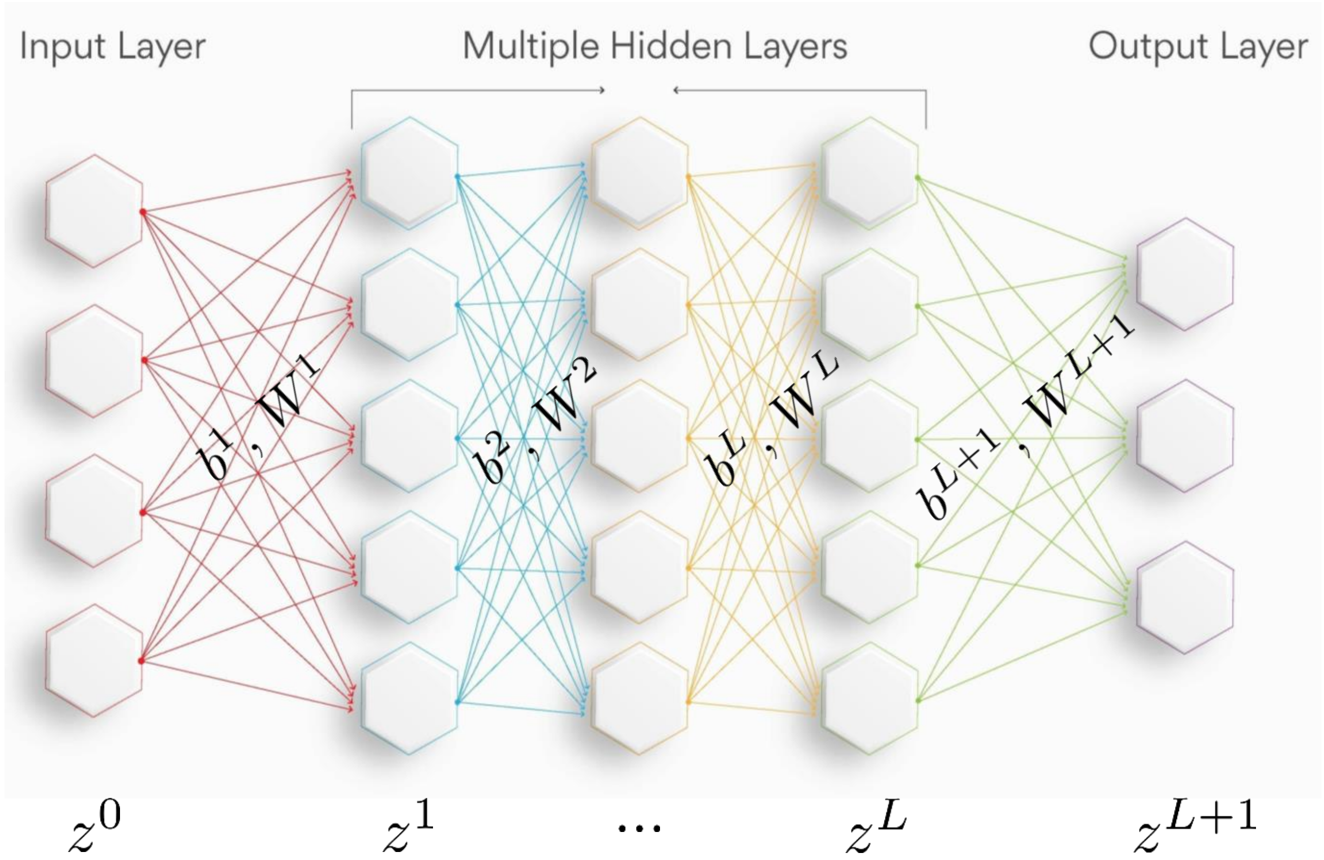


z^0 z^1 $\dots$ z^L z^{L+1}

$$\underbrace{\begin{pmatrix} z^{\ell+1} \end{pmatrix}}_{\text{layer vector}} = \underbrace{\begin{pmatrix} b^{\ell+1} \end{pmatrix}}_{\text{bias vector}} + \underbrace{\begin{pmatrix} W^{\ell+1} \end{pmatrix}}_{\text{weight matrix}} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied to each element of the vector

Basic neural network terminology

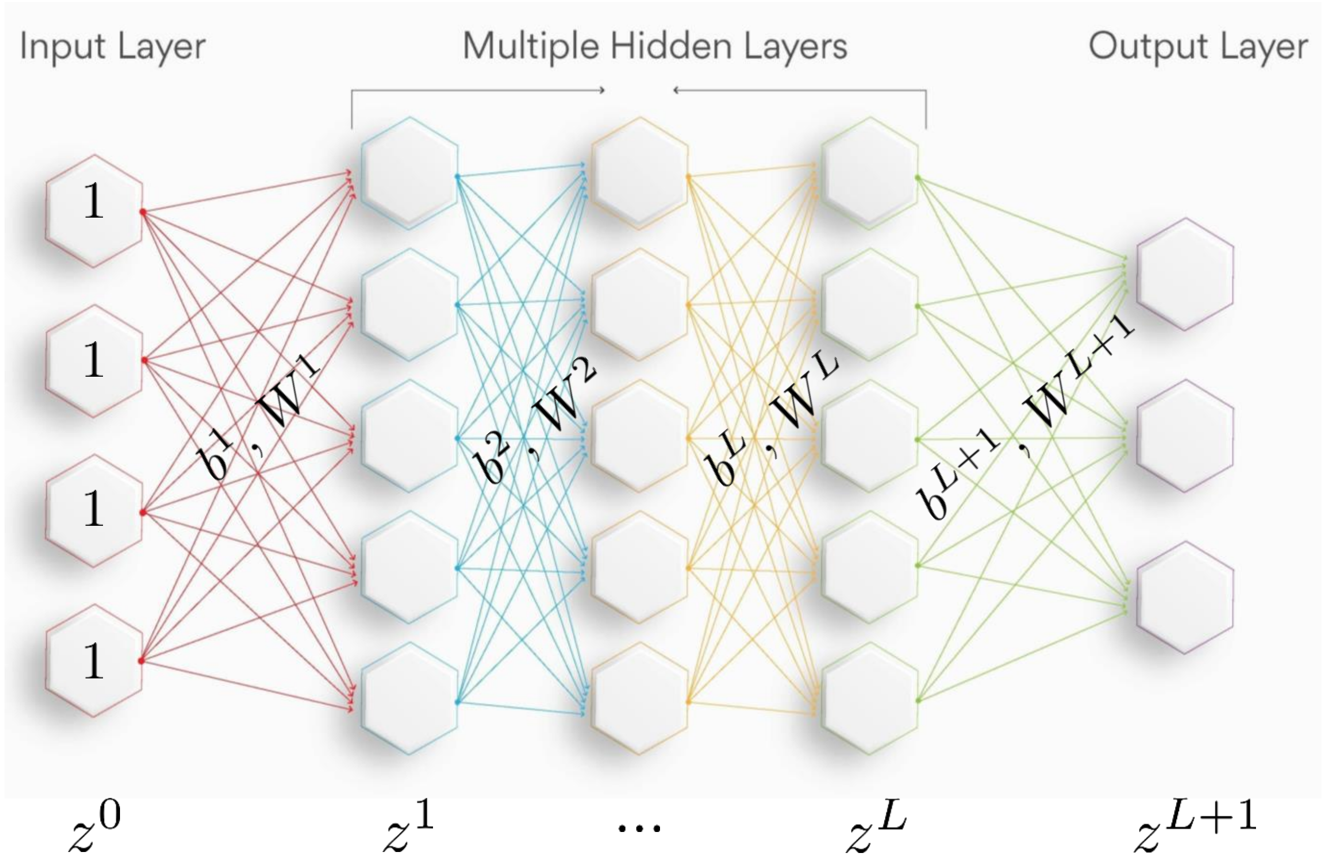


$$\overset{\text{layer vector}}{\begin{pmatrix} z^{\ell+1} \end{pmatrix}} = \overset{\text{bias vector}}{\begin{pmatrix} b^{\ell+1} \end{pmatrix}} + \overset{\text{weight matrix}}{\begin{pmatrix} \quad W^{\ell+1} \quad \end{pmatrix}} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied
to each element of the vector

- *Initializing the network* means picking <u>starting</u> values for biases and weights

Basic neural network terminology



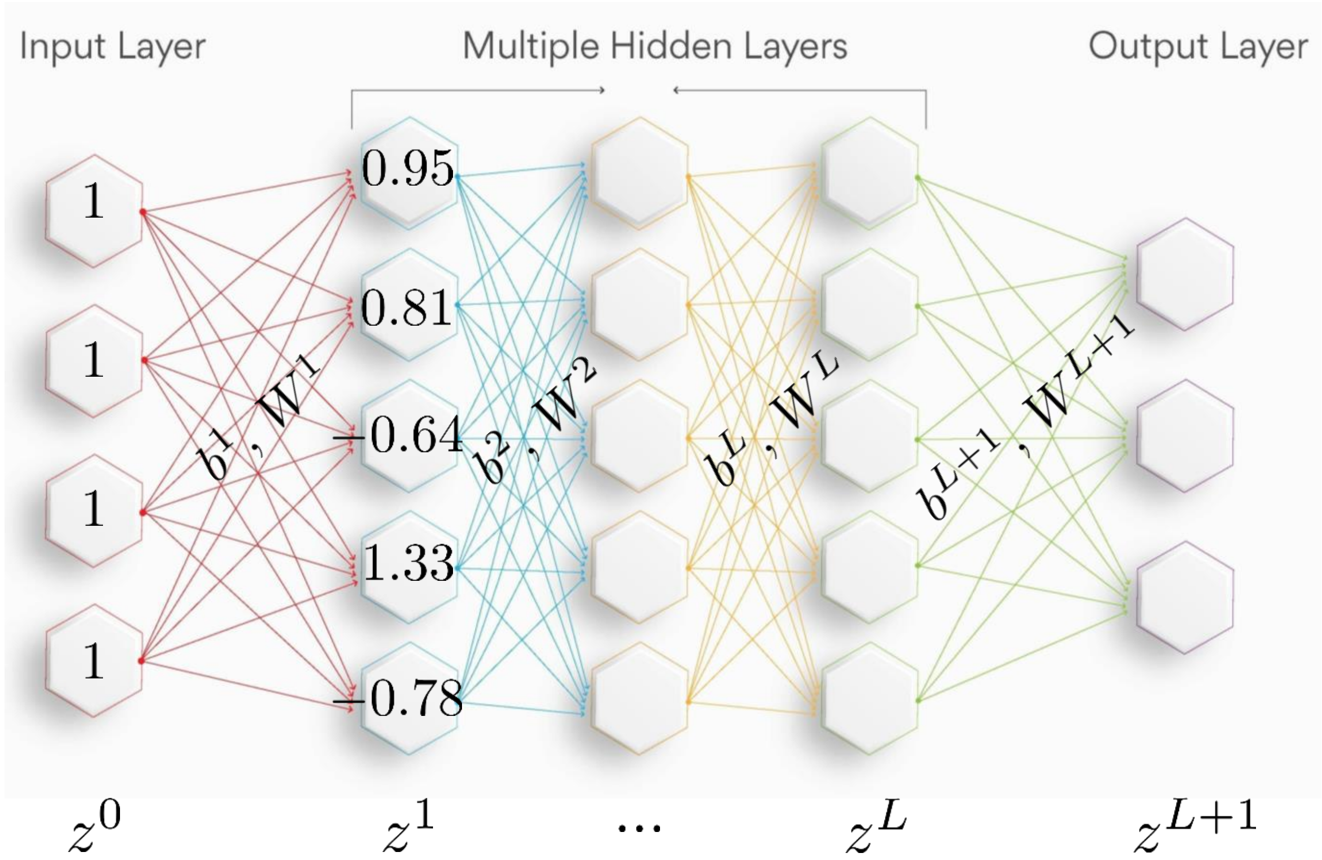
layer vector | bias vector | weight matrix

$$\begin{pmatrix} z^{\ell+1} \end{pmatrix} = \begin{pmatrix} b^{\ell+1} \end{pmatrix} + \begin{pmatrix} W^{\ell+1} \end{pmatrix} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied to each element of the vector

- *Initializing the network* means picking <u>starting</u> values for biases and weights
- Data *propagates* through the network

Basic neural network terminology



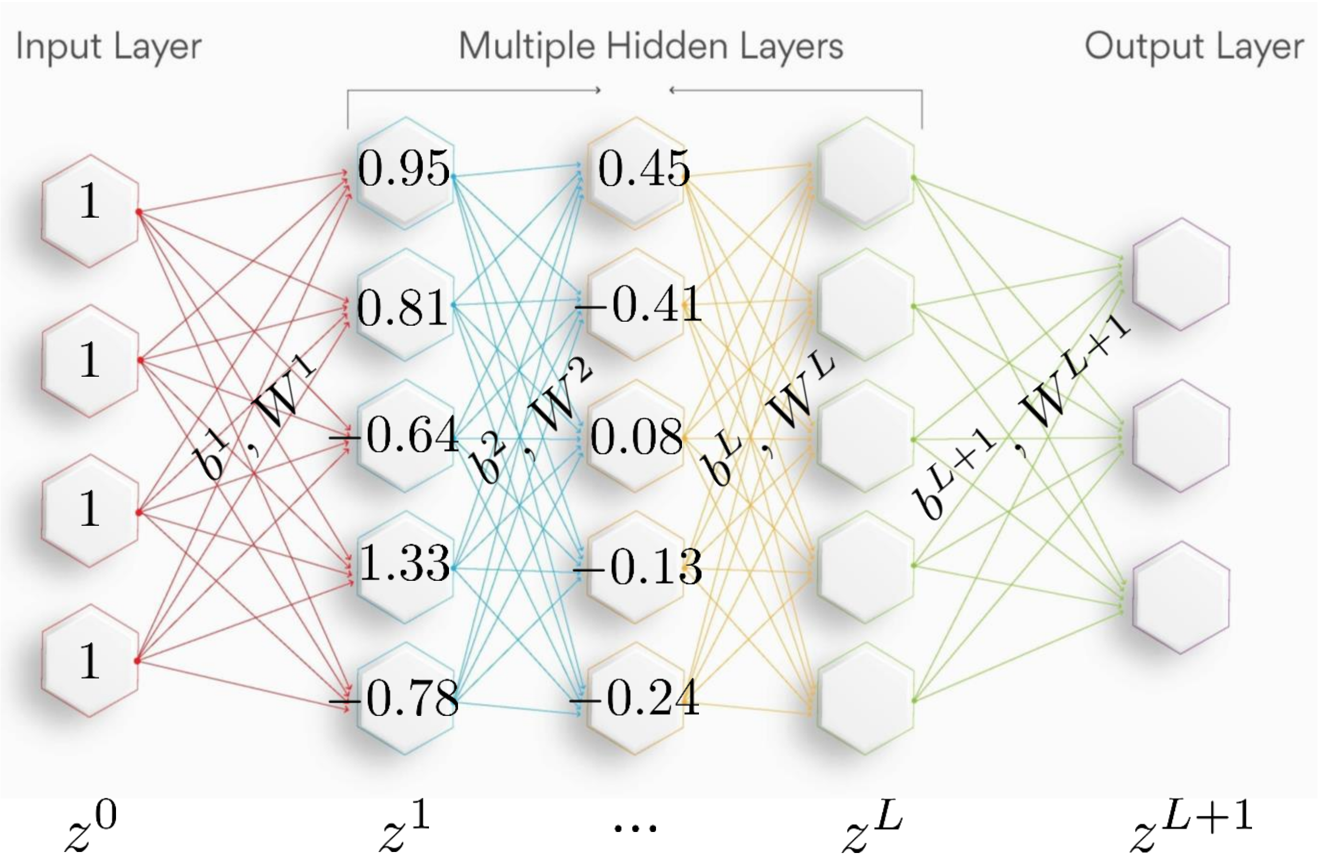
layer vector, bias vector, weight matrix

$$\left(z^{\ell+1} \right) = \left(b^{\ell+1} \right) + \left(\quad W^{\ell+1} \quad \right) \left(\sigma(z^\ell) \right)$$

activation function:
literally just some function applied to each element of the vector

- *Initializing the network* means picking <u>starting</u> values for biases and weights
- Data *propagates* through the network

Basic neural network terminology

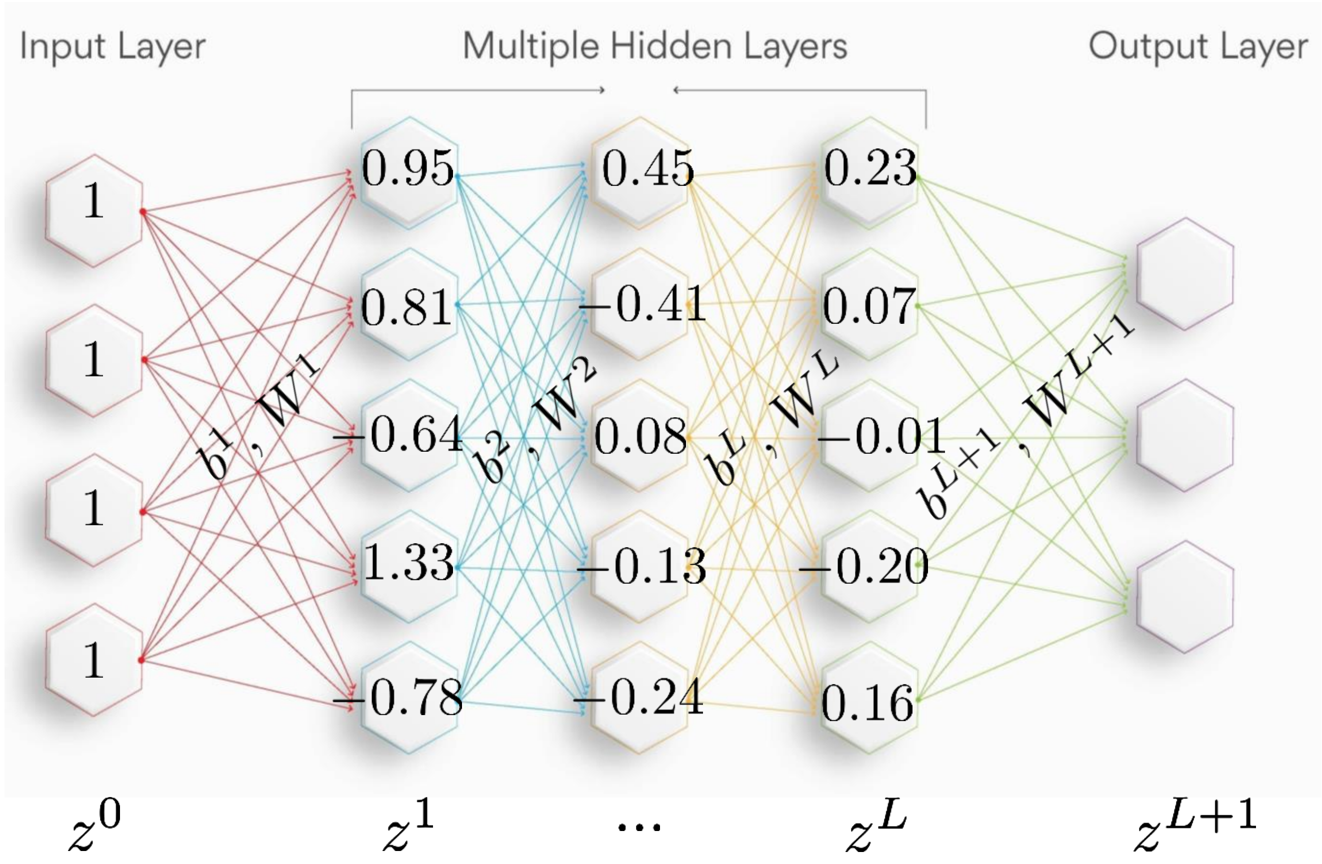


$$\overset{\text{layer vector}}{\begin{pmatrix} z^{\ell+1} \end{pmatrix}} = \overset{\text{bias vector}}{\begin{pmatrix} b^{\ell+1} \end{pmatrix}} + \overset{\text{weight matrix}}{\begin{pmatrix} \quad W^{\ell+1} \quad \end{pmatrix}} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied
to each element of the vector

- *Initializing the network* means picking <u>starting</u> values for biases and weights
- Data *propagates* through the network

Basic neural network terminology



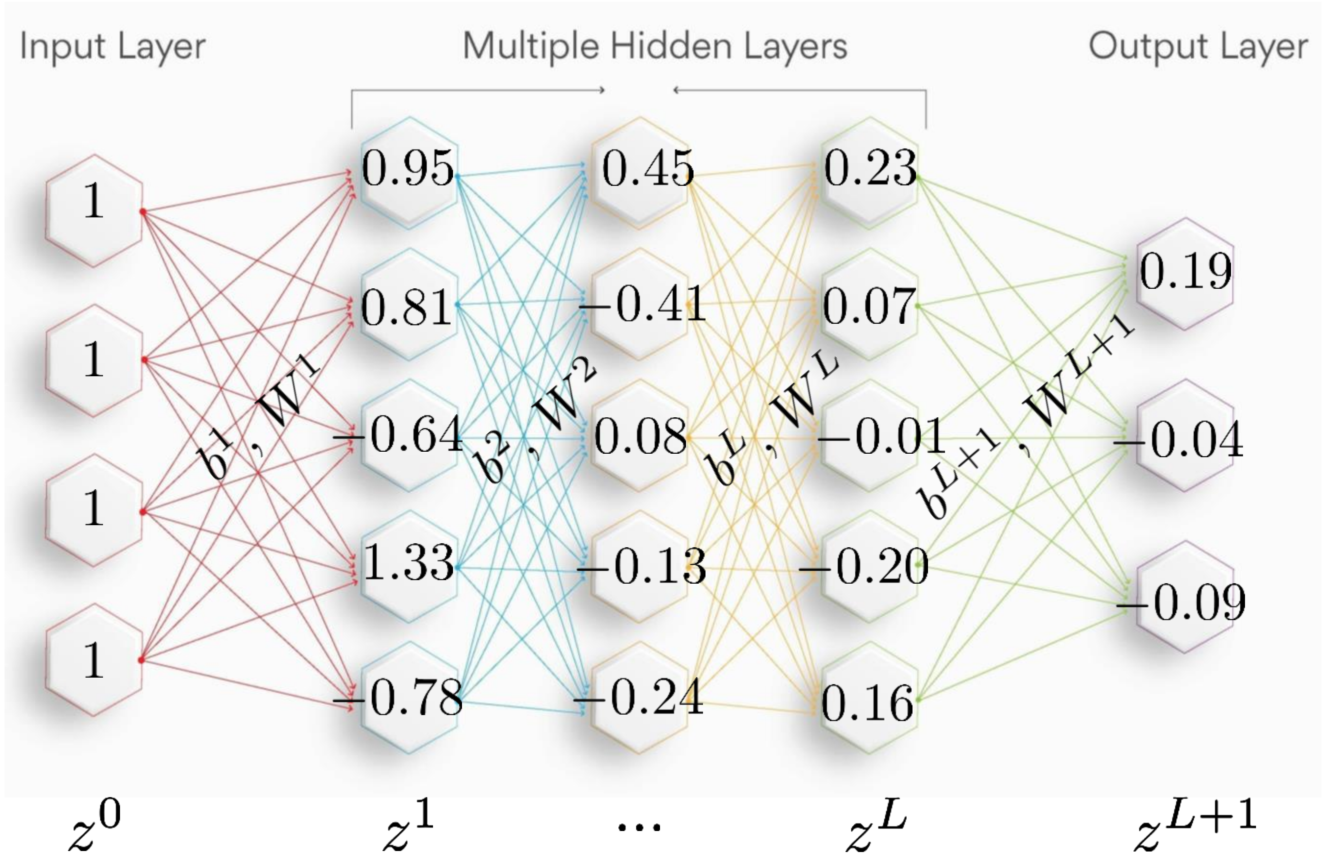
layer vector bias vector weight matrix

$$\begin{pmatrix} z^{\ell+1} \end{pmatrix} = \begin{pmatrix} b^{\ell+1} \end{pmatrix} + \begin{pmatrix} W^{\ell+1} \end{pmatrix} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied to each element of the vector

- *Initializing the network* means picking starting values for biases and weights
- Data *propagates* through the network

Basic neural network terminology



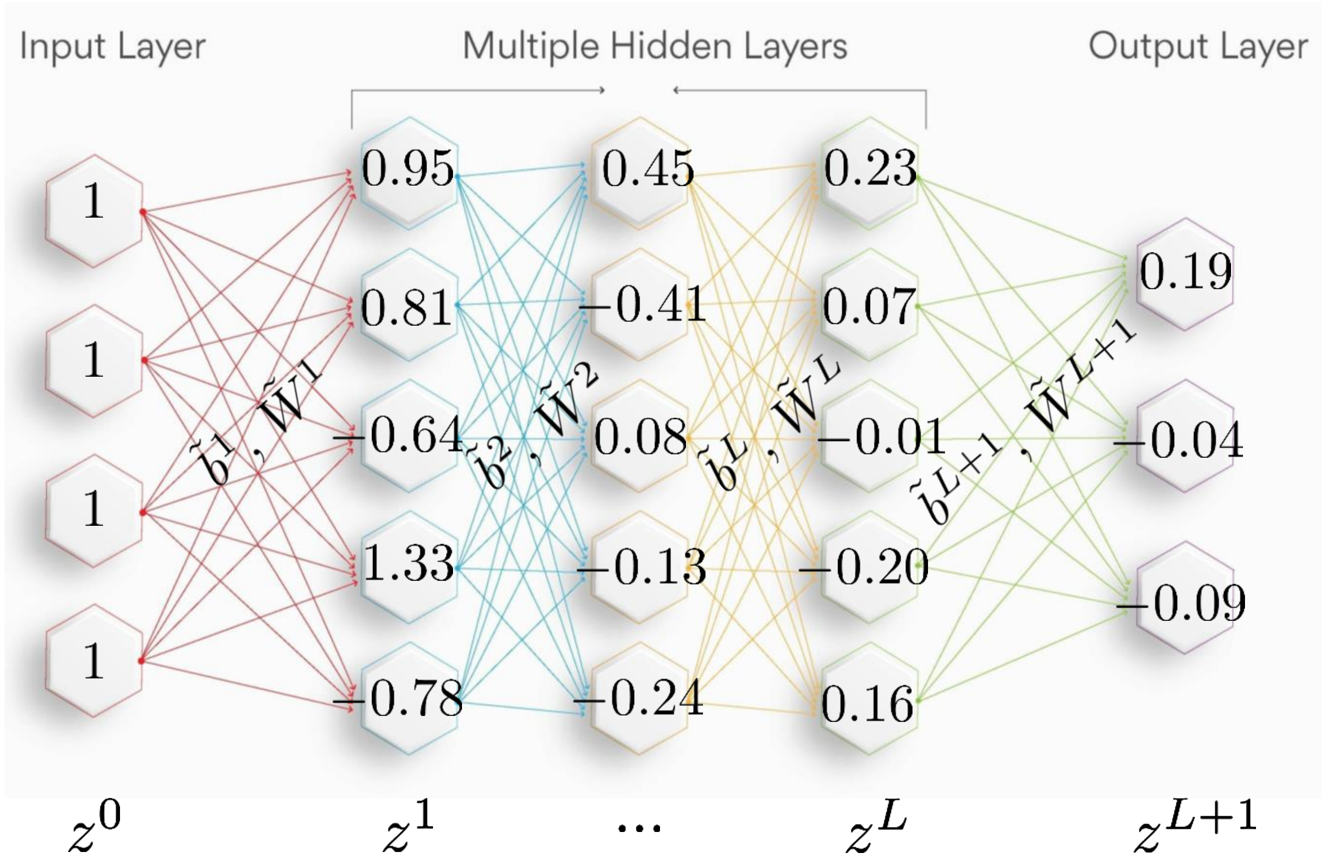
layer vector　bias vector　weight matrix

$$\begin{pmatrix} z^{\ell+1} \end{pmatrix} = \begin{pmatrix} b^{\ell+1} \end{pmatrix} + \begin{pmatrix} W^{\ell+1} \end{pmatrix} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function:
literally just some function applied to each element of the vector

- *Initializing the network* means picking starting values for biases and weights
- Data *propagates* through the network

Basic neural network terminology



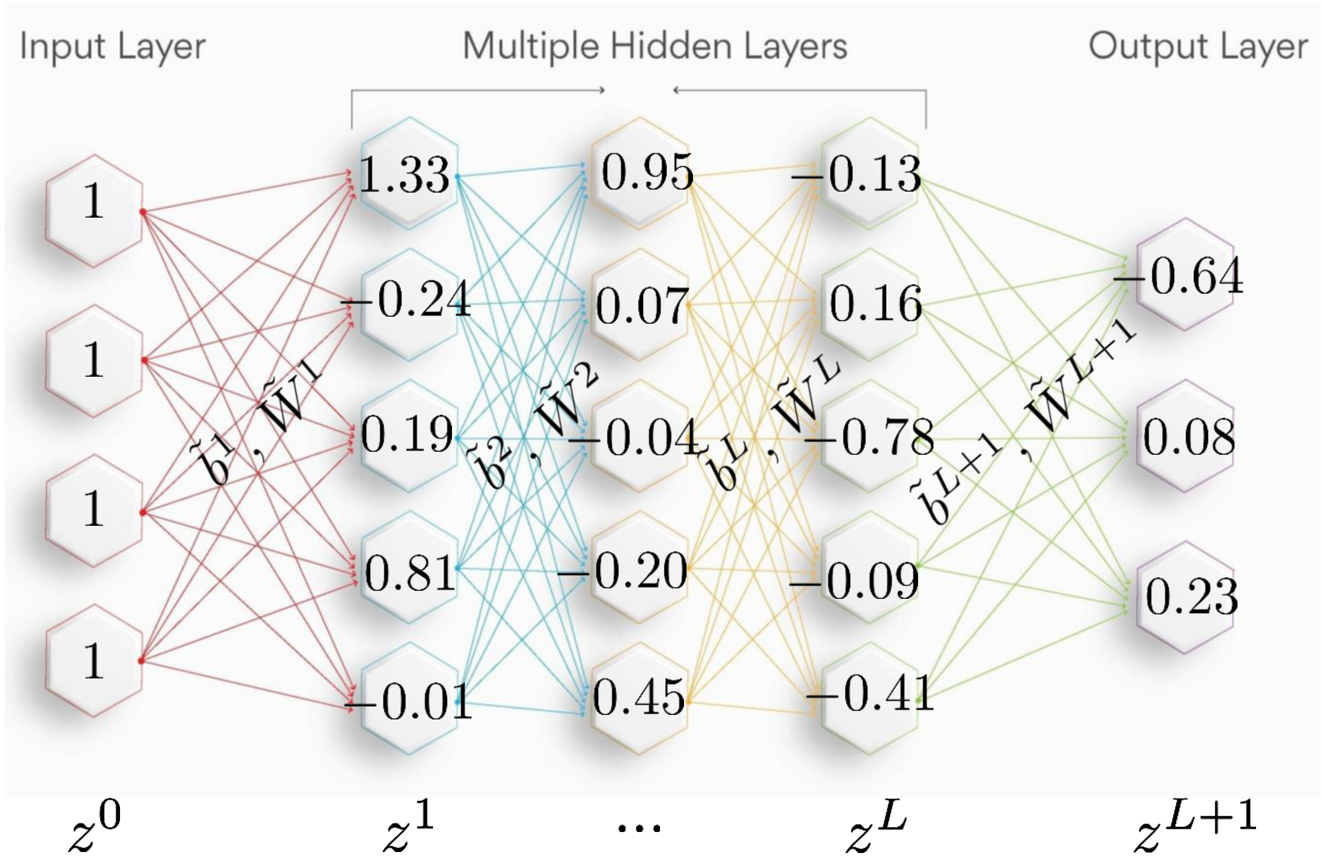
layer vector, bias vector, weight matrix

$$\begin{pmatrix} z^{\ell+1} \end{pmatrix} = \begin{pmatrix} b^{\ell+1} \end{pmatrix} + \begin{pmatrix} W^{\ell+1} \end{pmatrix} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

activation function: literally just some function applied to each element of the vector

- *Initializing the network* means picking <u>starting</u> values for biases and weights
- Data *propagates* through the network
- Biases and weights *evolve* during training

Basic neural network terminology

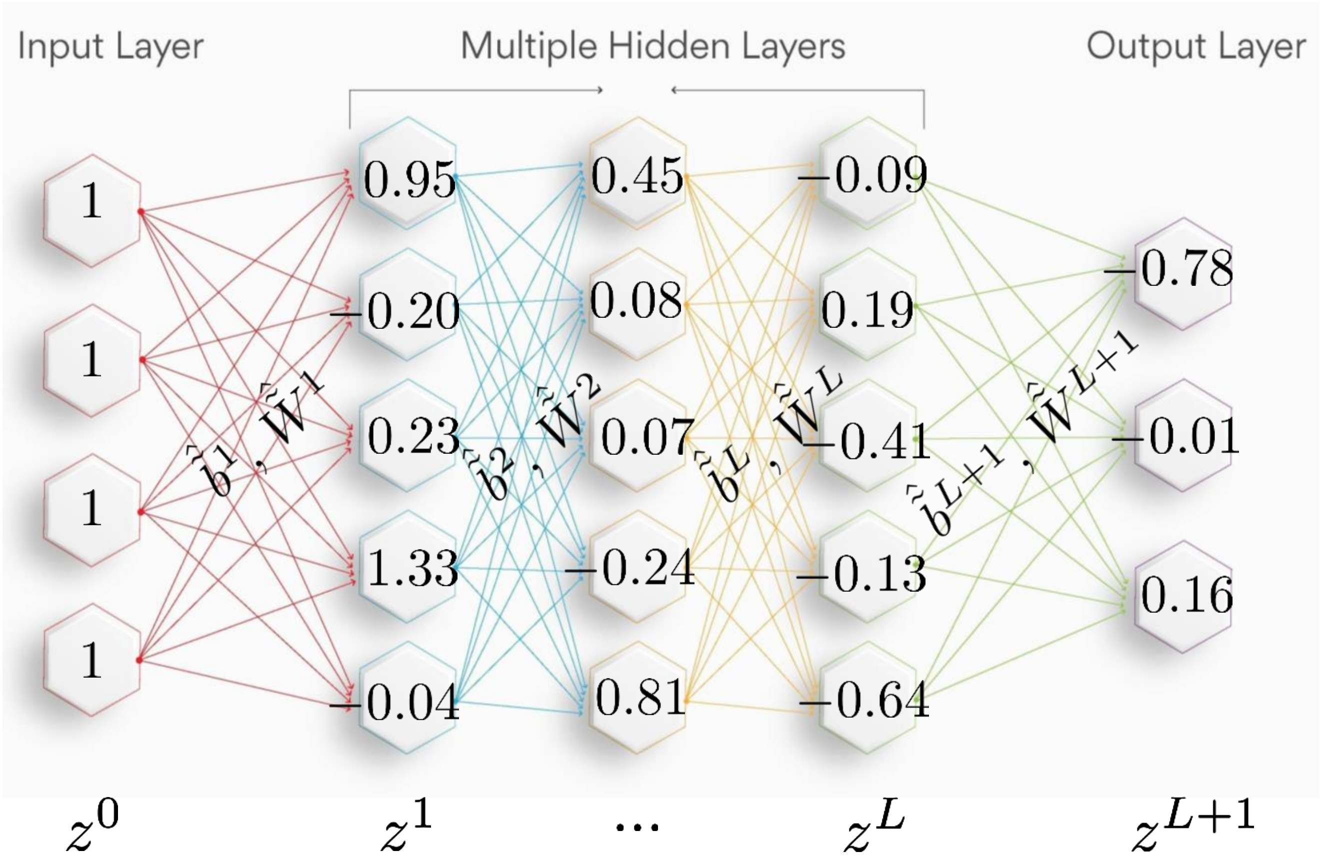


$$\underbrace{\left(z^{\ell+1} \right)}_{\text{layer vector}} = \underbrace{\left(b^{\ell+1} \right)}_{\text{bias vector}} + \underbrace{\left(\quad W^{\ell+1} \quad \right)}_{\text{weight matrix}} \left(\sigma(z^{\ell}) \right)$$

activation function:
literally just some function applied to each element of the vector

- *Initializing the network* means picking starting values for biases and weights
- Data *propagates* through the network
- Biases and weights *evolve* during training

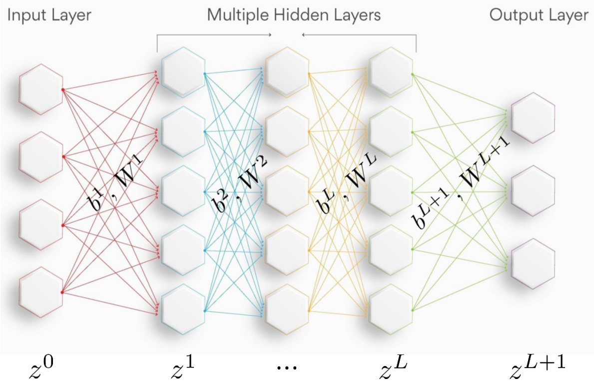
Basic neural network terminology



$$\underset{\text{layer vector}}{\begin{pmatrix} z^{\ell+1} \end{pmatrix}} = \underset{\text{bias vector}}{\begin{pmatrix} b^{\ell+1} \end{pmatrix}} + \underset{\text{weight matrix}}{\begin{pmatrix} \quad W^{\ell+1} \quad \end{pmatrix}} \begin{pmatrix} \sigma(z^\ell) \end{pmatrix}$$

activation function:
literally just some function applied to each element of the vector

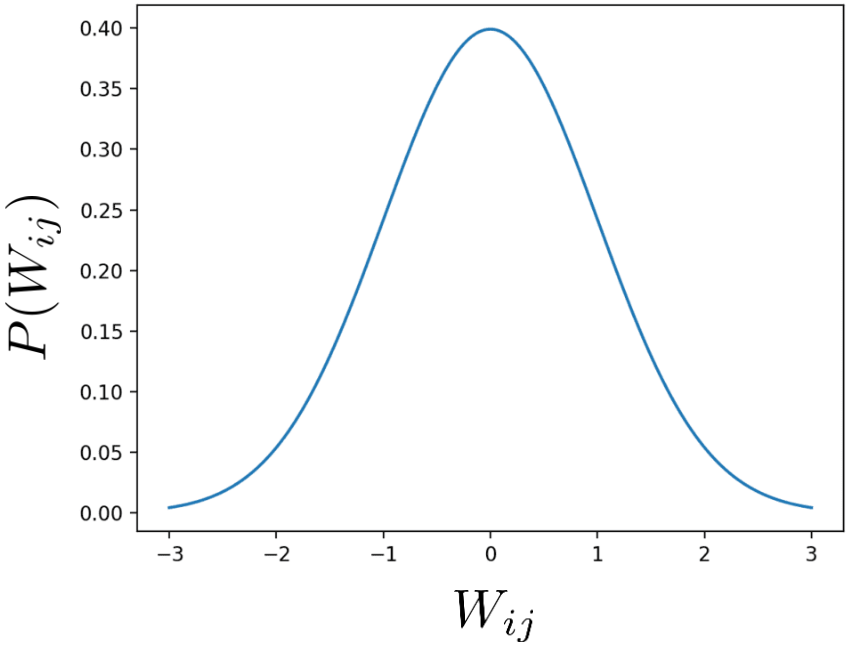
- *Initializing the network* means picking <u>starting</u> values for biases and weights
- Data *propagates* through the network
- Biases and weights *evolve* during training
- A *trained network* has “learned” the best biases and weights for optimal performance



Where's the physics?

- Initial weights and biases are randomly selected from a *distribution*

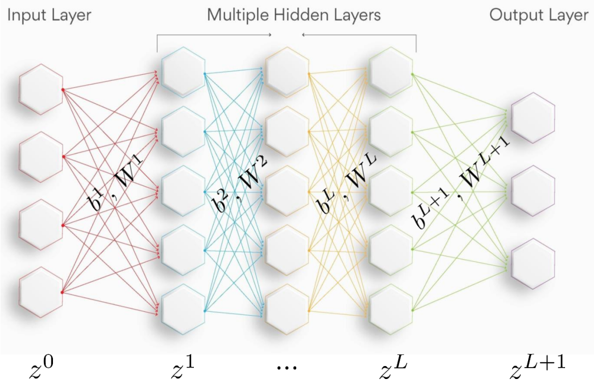
For example, sampling from a standard Gaussian distribution means the initial weight matrices could look like this:



$$W^1 = \begin{pmatrix} -0.34 & -0.09 & -0.05 & 0.03 \\ 0.76 & 0.98 & -0.53 & -1.13 \\ 0.50 & -0.97 & 0.51 & 0.61 \\ 0.59 & -0.14 & 0.76 & 0.69 \end{pmatrix}$$

$$W^2 = \begin{pmatrix} 0.11 & 0.54 & -0.30 & 0.42 \\ 0.01 & 0.06 & -0.10 & -0.47 \\ 0.58 & -0.08 & -0.27 & 0.76 \\ 0.53 & -0.37 & 0.28 & 0.92 \end{pmatrix}$$

$$W^3 = \begin{pmatrix} -0.02 & -0.06 & -0.50 & -0.73 \\ 0.54 & 0.49 & -0.35 & 0.67 \\ -0.99 & -0.02 & -0.56 & 0.01 \\ -0.68 & 0.29 & -0.03 & -0.40 \end{pmatrix}$$



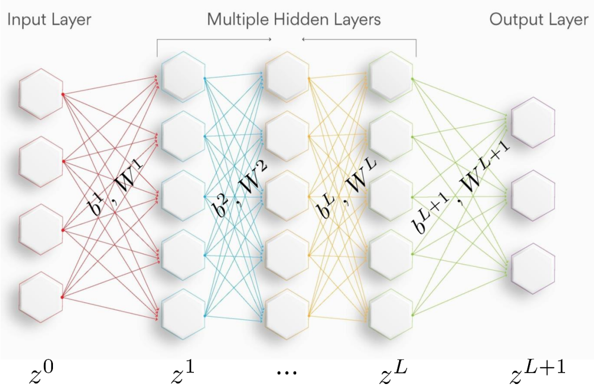
Where's the physics?

- Initial weights and biases are randomly selected from a *distribution*
- Deep networks must be tuned to *criticality*

$$z^0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \xrightarrow{b^1,\, W^1} z^1 = \begin{pmatrix} 0.95 \\ 0.81 \\ -0.64 \\ 1.33 \end{pmatrix} \xrightarrow{b^2,\, W^2} z^2 = \begin{pmatrix} 0.45 \\ -0.41 \\ 0.08 \\ -0.13 \end{pmatrix} \xrightarrow{b^3,\, W^3} z^3 = \begin{pmatrix} 0.23 \\ 0.07 \\ -0.01 \\ -0.20 \end{pmatrix}$$

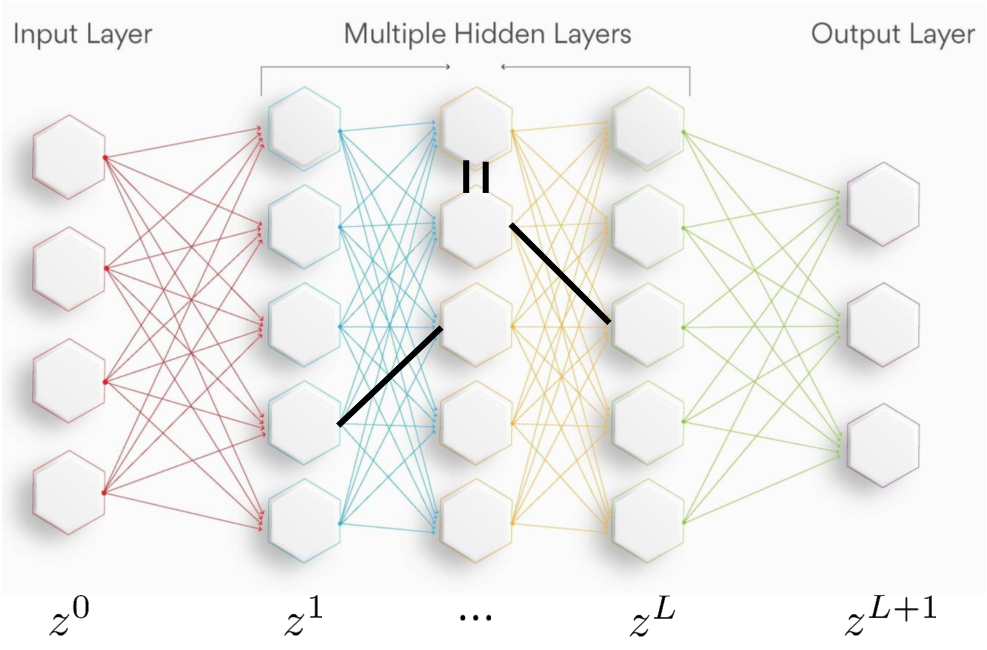
versus

$$z^0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \xrightarrow{b^1,\, W^1} z^1 = \begin{pmatrix} -1.59 \\ 0.70 \\ 1.66 \\ -0.23 \end{pmatrix} \xrightarrow{b^2,\, W^2} z^2 = \begin{pmatrix} 0.22 \\ 0.75 \\ -2.04 \\ 1.39 \end{pmatrix} \xrightarrow{b^3,\, W^3} z^3 = \begin{pmatrix} 1.03 \\ -1.34 \\ -0.77 \\ 1.12 \end{pmatrix}$$

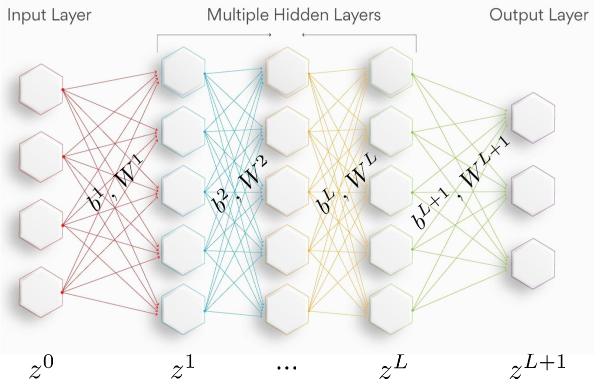


Where's the physics?

- Initial weights and biases are randomly selected from a *distribution*
- Deep networks must be tuned to *criticality*
- Interactions between network nodes can be quantified with *couplings*



Where’s the physics?

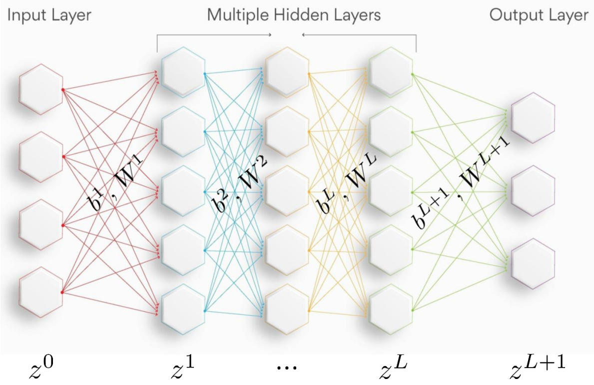


- Initial weights and biases are randomly selected from a *distribution*
- Deep networks must be tuned to *criticality*
- Interactions between network nodes can be quantified with *couplings*

Sounds suspiciously like stat mech!

- Infinite-width neural network = free field theory
- Finite width $\Longrightarrow$ interactions
- Signals propagation = renormalization group flow
- Critically tuned weights and biases = marginal couplings / critical point

Where’s the physics?

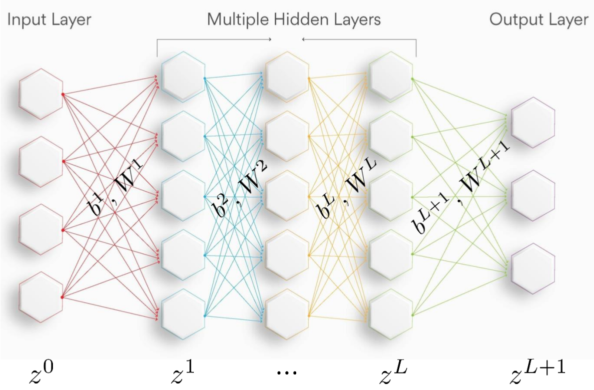


- Initial weights and biases are randomly selected from a *distribution*
- Deep networks must be tuned to *criticality*
- Interactions between network nodes can be quantified with *couplings*

Sounds suspiciously like stat mech!

Given our initial network conditions,

can we predict how the network will evolve?



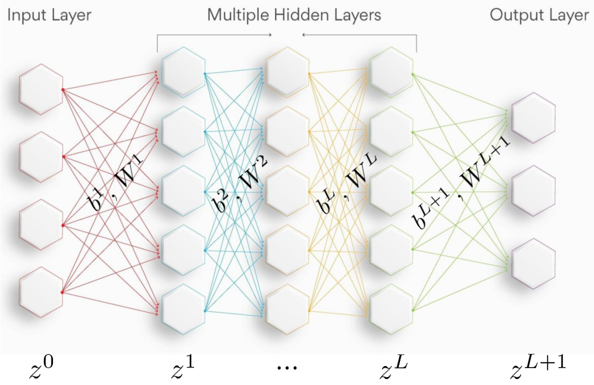
Where's the physics?

- Initial weights and biases are randomly selected from a *distribution*
- Deep networks must be tuned to *criticality*
- Interactions between network nodes can be quantified with *couplings*

Sounds suspiciously like stat mech!

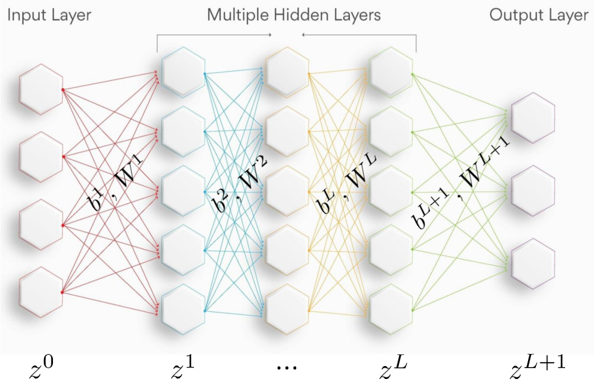
Given how we want the network to evolve,

can we determine the necessary initial conditions?



What makes a network "good"?

- High percentage of correct prediction
- Similar inputs should go to similar outputs
- Expect similar results every time you use the network

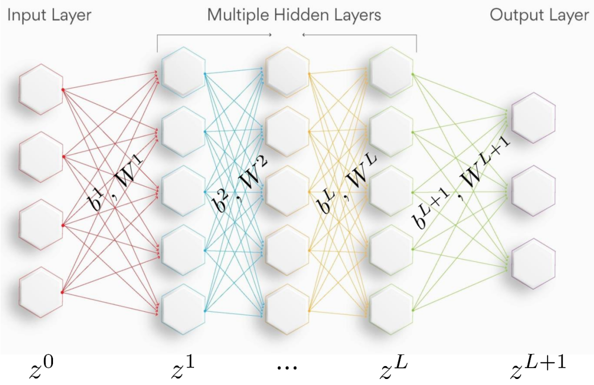


What makes a network “good”?

- High percentage of correct prediction
- Similar inputs should go to similar outputs
- Expect similar results every time you use the network

Especially important for physics applications

What makes a network "good"?

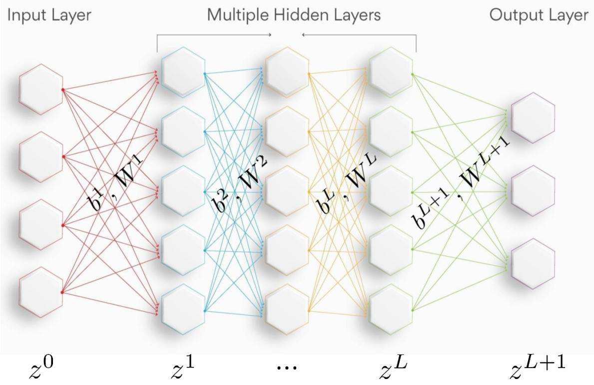


- High percentage of correct prediction
- Similar inputs should go to similar outputs
- Expect similar results every time you use the network

Especially important for physics applications:

E.g.
- 2 top quark jet images should receive similar classification
- That classification should be the same every time

What makes a network “good”?

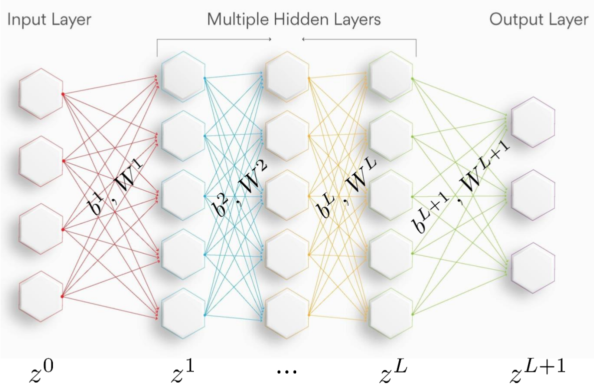


- High percentage of correct prediction
- $\boxed{\text{Similar inputs should go to similar outputs}}$
- $\boxed{\text{Expect similar results every time you use the network}}$

Especially important for physics applications

? Perhaps introducing physically motivated interactions will improve “goodness” of network

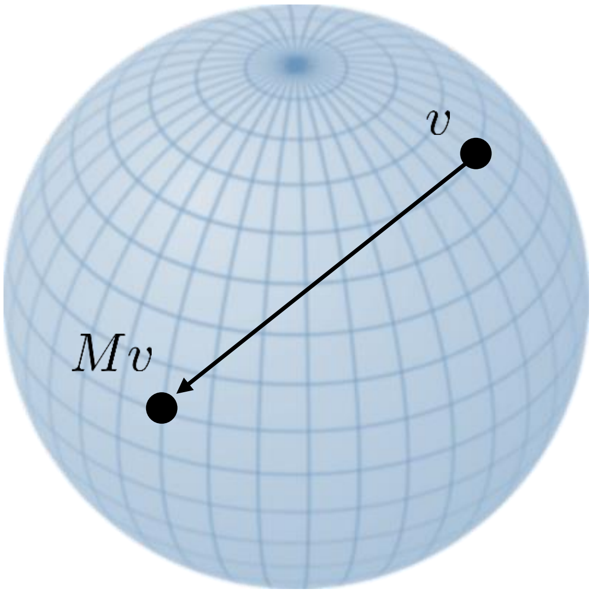
? Perhaps we can quantify the “goodness” of a network based on initial network parameters



The orthogonal distribution

(Physically motivated interactions)

- An orthogonal matrix rotates points on a sphere
  $\Rightarrow$ automatically preserves vector norms
- Naturally limits explosions and decays



layer vector, bias vector, weight matrix

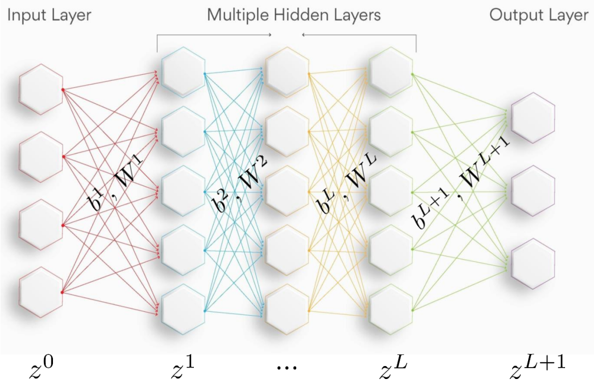
$$\left(z^{\ell+1} \right) = \left(b^{\ell+1} \right) + \left(\quad W^{\ell+1} \quad \right) \left(\sigma(z^{\ell}) \right)$$

bias initialization can always be set to zero

An orthogonal weight matrix will not change the magnitude of a vector

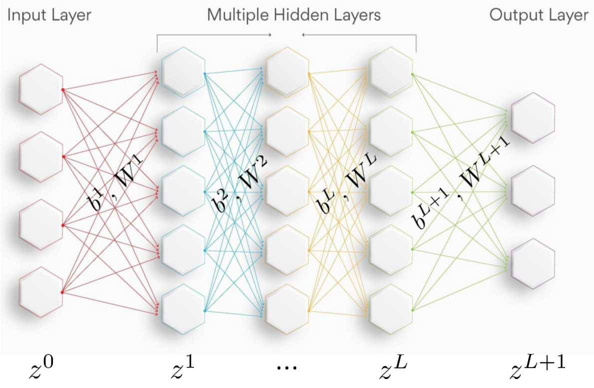
activation function: literally just some function applied to each element of the vector

What to measure?



- Stat mech relies on probabilities which require randomness
    - Initialize network 100 times to get 100 sets of parameters
    - Take averages over initializations to get expectation values
- Measure properties of initialization that inform <u>network performance</u>

What to measure?

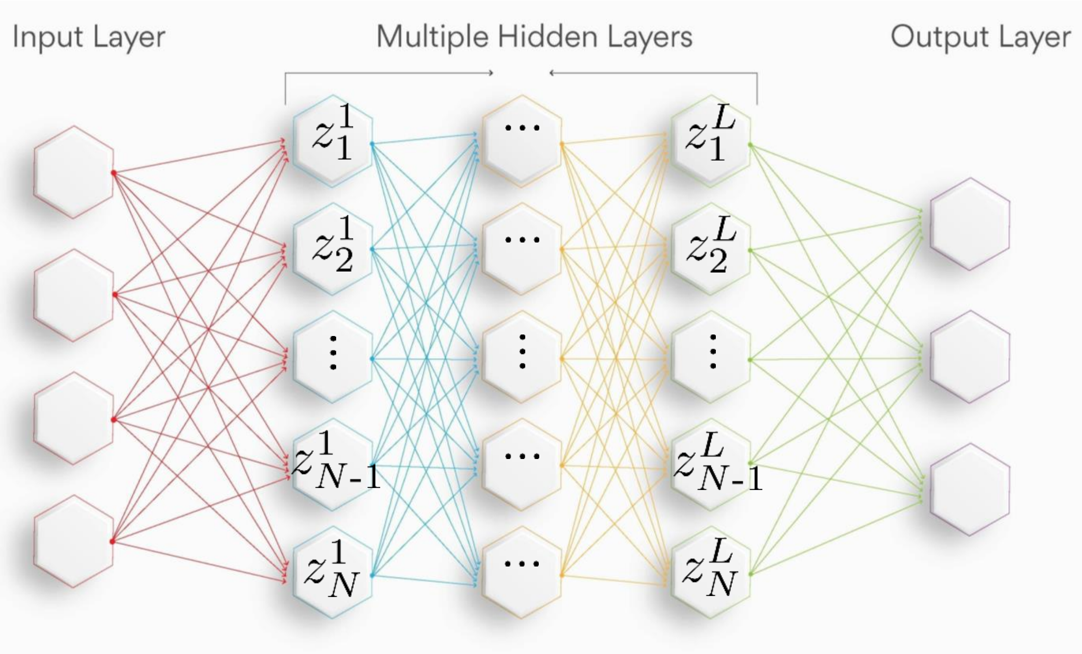


- Stat mech relies on probabilities which require randomness
  - Initialize network 100 times to get 100 sets of parameters
  - Take averages over initializations to get expectation values
- Measure properties of initialization that inform network performance
  - Similar inputs should go to similar outputs ⇒ n-point functions
  - Expect similar results every time you use the network ⇒ NTK

$$\overset{\text{layer vector}}{\begin{pmatrix} z^{\ell+1} \end{pmatrix}} = \overset{\text{bias vector}}{\begin{pmatrix} b^{\ell+1} \end{pmatrix}} + \overset{\text{weight matrix}}{\begin{pmatrix} \quad W^{\ell+1} \quad \end{pmatrix}} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

N-point functions

- Average of products of different combinations of neurons in each layer
- Similar inputs → similar outputs
  ⇒ want minimal layer-dependence
  (limit explosions and decays)

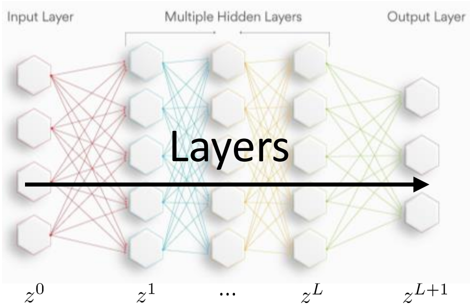


$$\mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_2}]$$

$$\mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_2} z^{\ell}_{i_3} z^{\ell}_{i_4}]$$

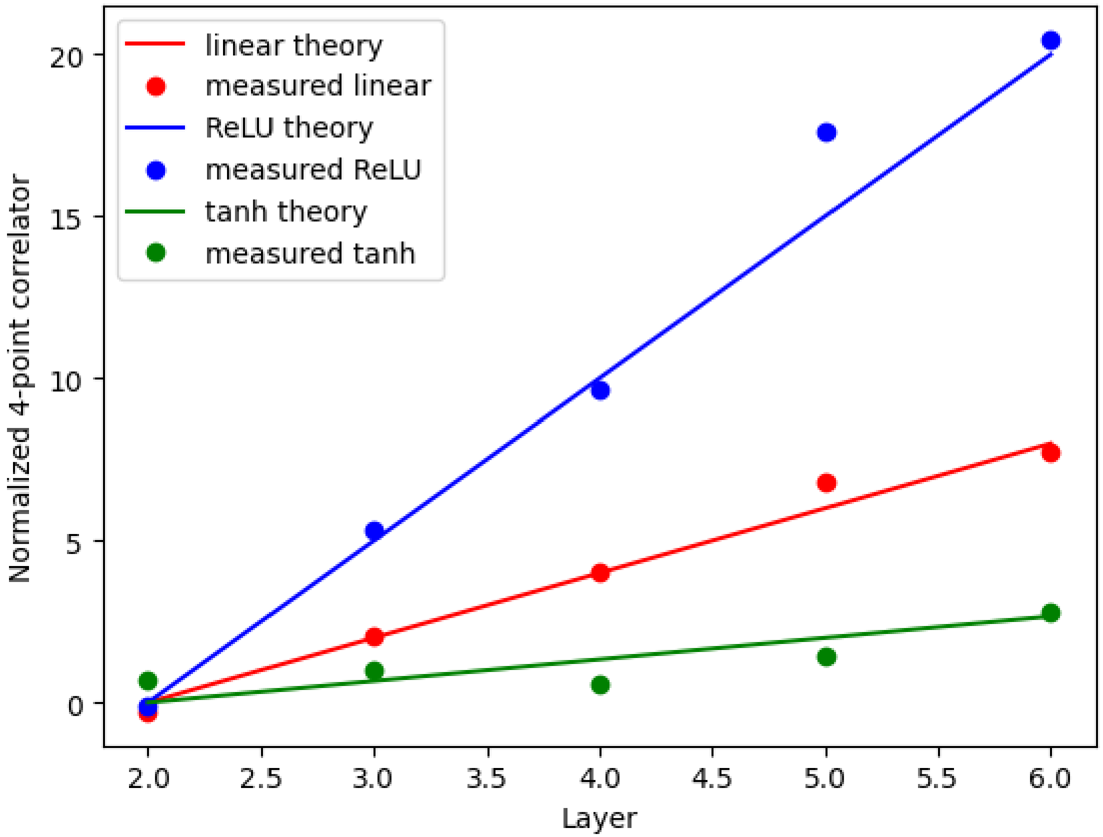
$$\mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_2} z^{\ell}_{i_3} z^{\ell}_{i_4}]\big|_{\text{conn.}} = \mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_2} z^{\ell}_{i_3} z^{\ell}_{i_4}] - \mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_2}]\mathbb{E}[z^{\ell}_{i_3} z^{\ell}_{i_4}] - \mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_3}]\mathbb{E}[z^{\ell}_{i_2} z^{\ell}_{i_4}] - \mathbb{E}[z^{\ell}_{i_1} z^{\ell}_{i_4}]\mathbb{E}[z^{\ell}_{i_2} z^{\ell}_{i_3}]$$

$$\vdots$$

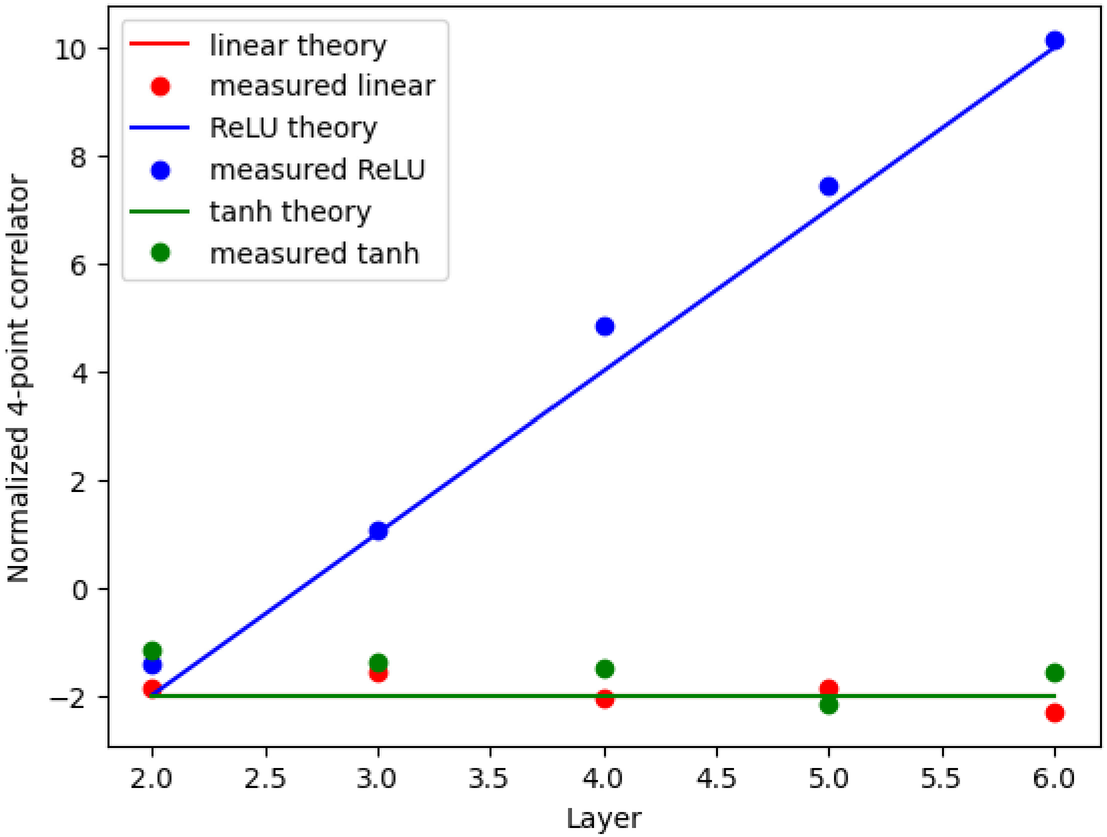


N-point functions

Gaussian weight initialization



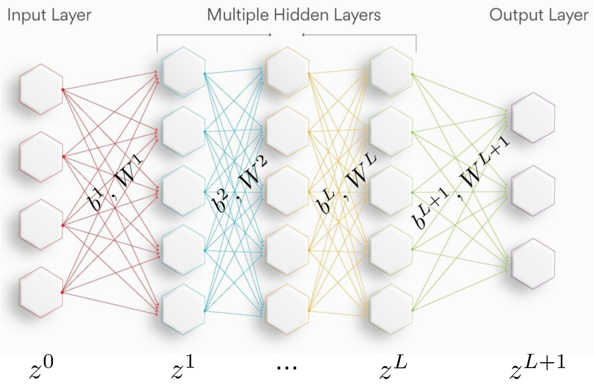
orthogonal weight initialization



Orthogonal initialization removes layer dependance!

HD, Y. Kahn, D. Roberts [arXiv:23XX.XXXX]

What to measure?



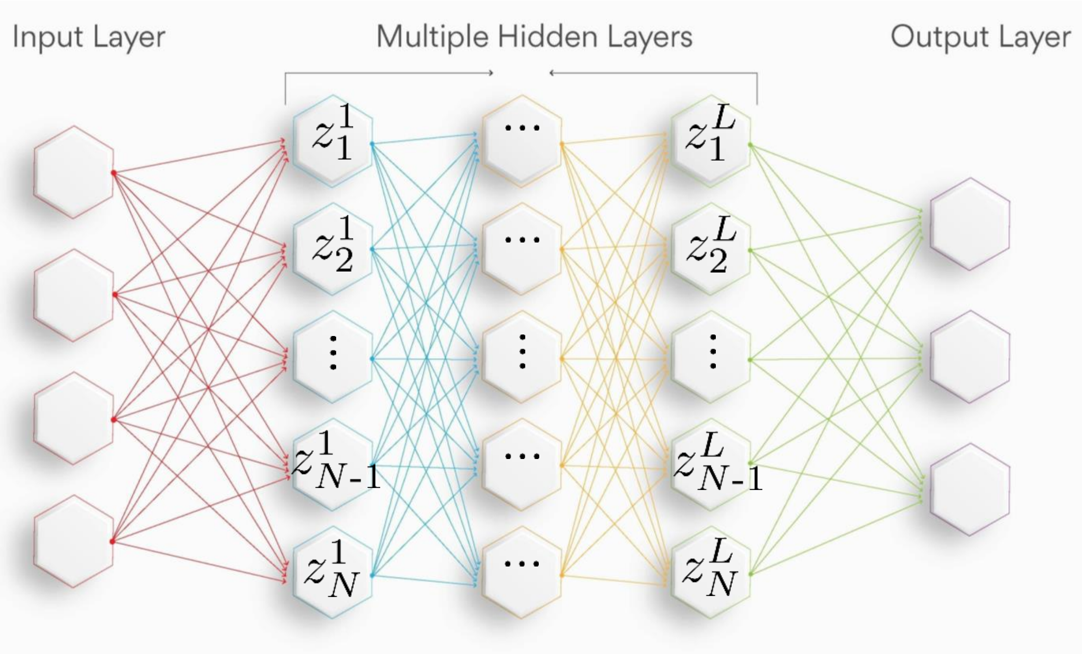
- Stat mech relies on probabilities which require randomness
  - Initialize network 100 times to get 100 sets of parameters
  - Take averages over initializations to get expectation values
- Measure properties of initialization that inform network performance
  - Similar inputs should go to similar outputs ⇒ n-point functions
  - Expect similar results every time you use the network ⇒ NTK

$$\overset{\text{layer vector}}{\left(z^{\ell+1} \right)} = \overset{\text{bias vector}}{\left(b^{\ell+1} \right)} + \overset{\text{weight matrix}}{\left(\quad W^{\ell+1} \quad \right)} \left(\sigma(z^{\ell}) \right)$$

The neural tangent kernel (NTK)

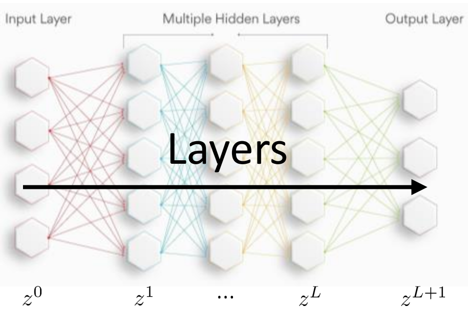
- Change in layer outputs as biases and weights are updated during training
- Governs *feature learning*, i.e. whether something useful happens during training
- Consistent results

  ⇒ want minimal layer-dependance
- But beware of tradeoff with learning, which requires layer-dependence



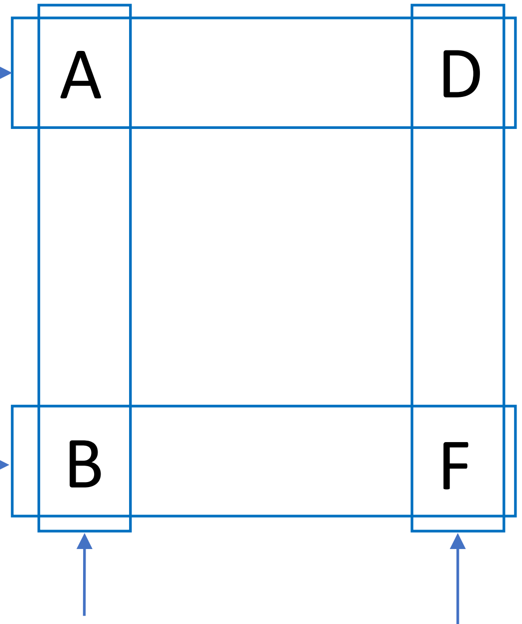
$$\hat{H}^{(\ell)} \propto \frac{dz^{(\ell)}}{d\theta}\frac{dz^{(\ell)}}{d\theta} \;, \quad \theta \in \{b, W\}$$

NTK: Gaussian vs. orthogonal



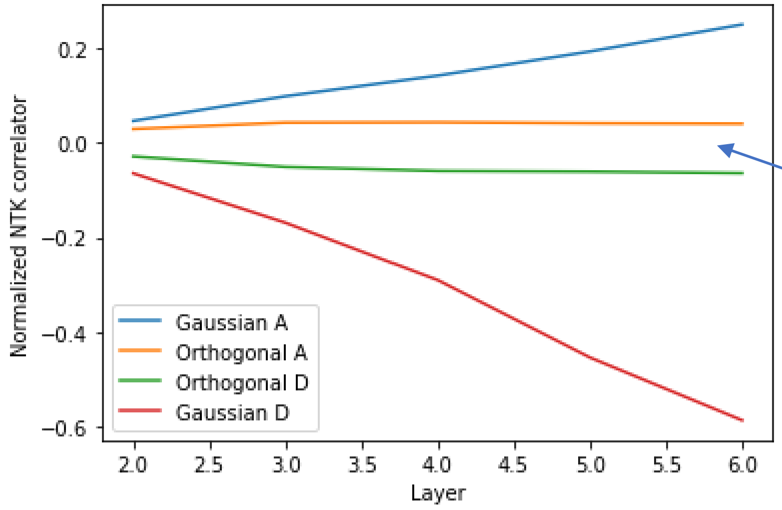
do not affect NTK evolution
(only induce noise)

update NTK during training
(allow feature learning)

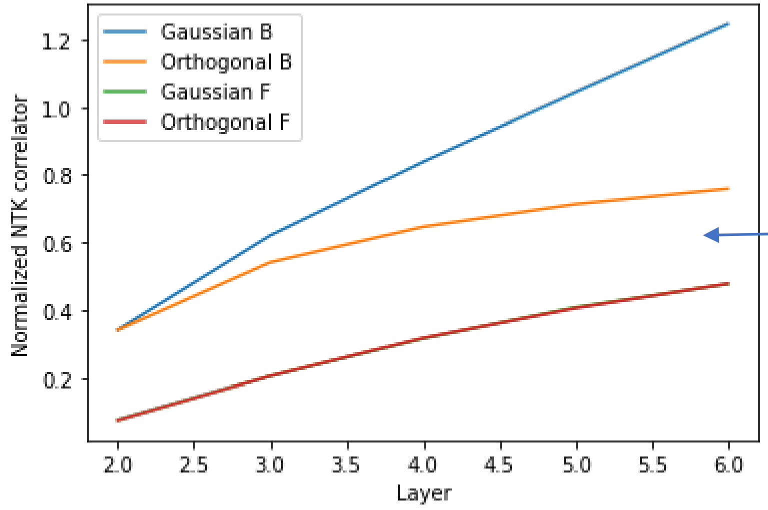


NTK fluctuations

interactions between
NTK and layer output



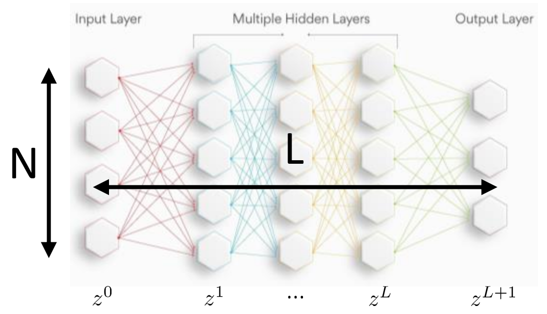
“bad” correlators become depth-independent



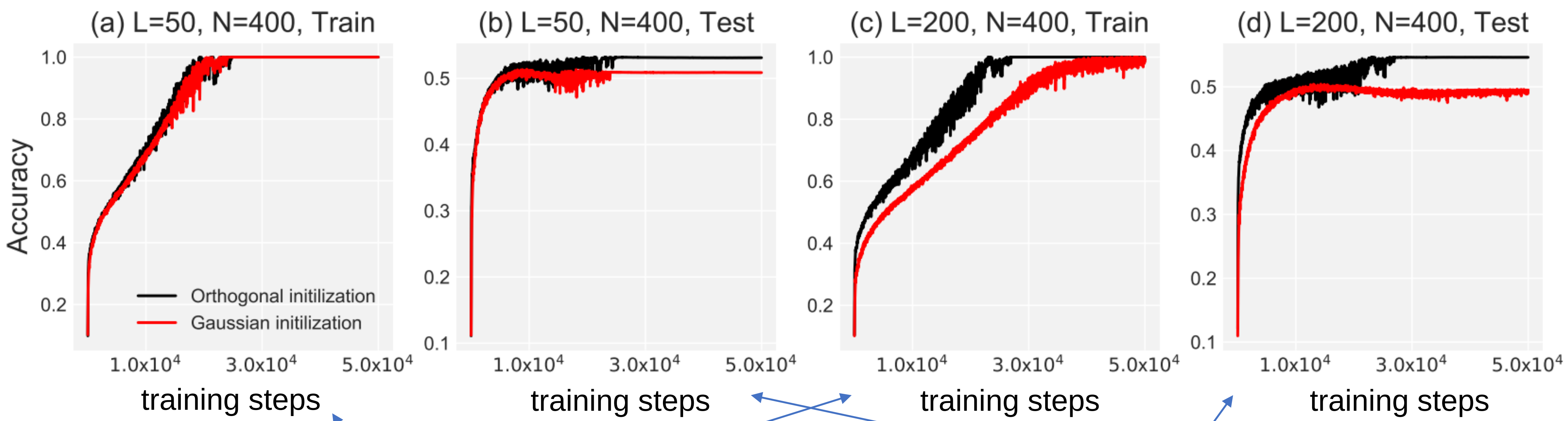
“good” correlators maintain depth-dependence

HD, Y. Kahn, D. Roberts [arXiv:23XX.XXXX]

Are the predictors right?



(Does reducing "bad" depth dependance improve network learning?)



Gaussian takes longer to train when depth increases
Orthogonal trains at approximately the same rate

Gaussian test accuracy plateaus
sooner and lower than orthogonal
as training steps increase

W. Huang, W. Du, R. Xu [arXiv:2004.05867]

Are the predictors right?

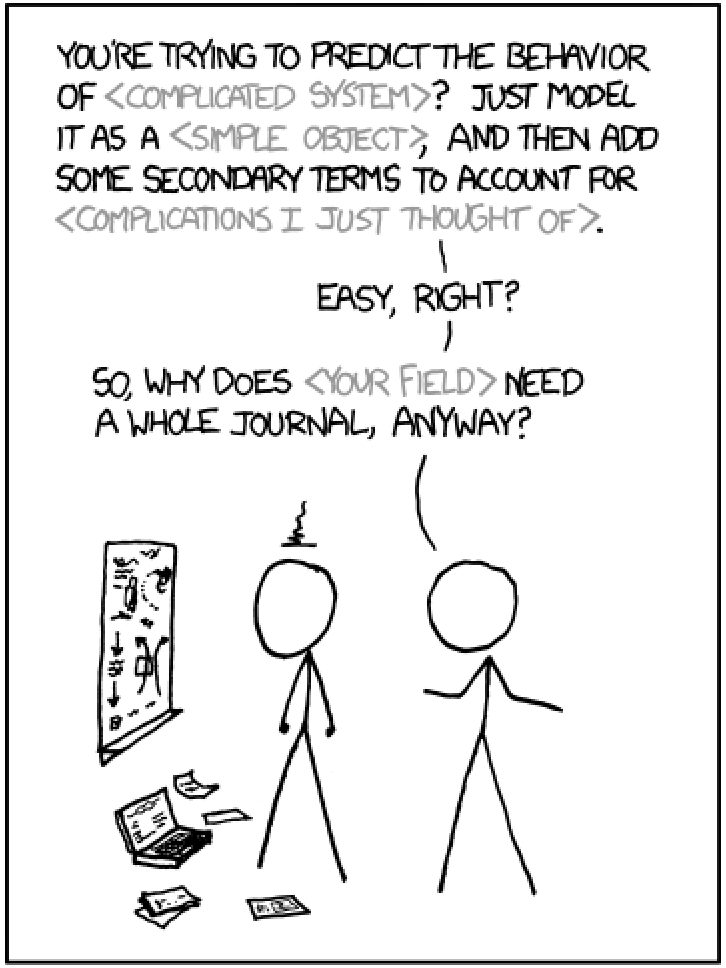
(Does reducing “bad” depth dependance improve network learning?)

Future work:

- Does variance in accuracy also decrease with orthogonal initialization?
- What happens with other types of networks (e.g. convolutional)?
- Can we generically determine the best initialization distribution?
- How does the type of data affect results?
- How far does the analogy go? (Feynman diagrams?)

Conclusion

- Neural networks can be described using statistical mechanics
- Network outputs can be both stochastic (governed by statistics) and deterministic (predictable) – just like in stat mech!
- Techniques from statistical mechanics can be used for network optimization
- Measurements at initialization can predict training success



LIBERAL-ARTS MAJORS MAY BE ANNOYING SOMETIMES, BUT THERE'S *NOTHING* MORE OBNOXIOUS THAN A PHYSICIST FIRST ENCOUNTERING A NEW SUBJECT.

Explosions and disappearances

(What does criticality mean?)

- Large weight initialization ⇒ large network output

$$|W^i| = \mathcal{O}(100) \Longrightarrow |z^L| \to \infty$$

- Small weight initialization ⇒ small network output

$$|W^i| = \mathcal{O}(0.01) \Longrightarrow |z^L| \to 0$$

- Network cannot learn in either case

Explosions and disappearances

(What does criticality mean?)

- Large weight initialization ⇒ large network output

$$|W^i| = \mathcal{O}(100) \Longrightarrow |z^L| \to \infty$$

- Small weight initialization ⇒ small network output

$$|W^i| = \mathcal{O}(0.01) \Longrightarrow |z^L| \to 0$$

- Network cannot learn in either case

~~Similar inputs should go to similar outputs~~

~~Expect similar results every time you use the network~~

Explosions and disappearances

(What does criticality mean?)

- Large weight initialization ⇒ large network output

$$|W^i| = \mathcal{O}(100) \Longrightarrow |z^L| \to \infty$$

- Small weight initialization ⇒ small network output

$$|W^i| = \mathcal{O}(0.01) \Longrightarrow |z^L| \to 0$$

- Network cannot learn in either case
- *Tuning to criticality* means preventing exploding and decaying signals

Similar inputs should go to similar outputs

Expect similar results every time you use the network

Explosions and disappearances

(What does criticality mean?)

- Large weight initialization ⇒ large network output

$$|W^i| = \mathcal{O}(100) \Longrightarrow |z^L| \to \infty$$

- Small weight initialization ⇒ small network output

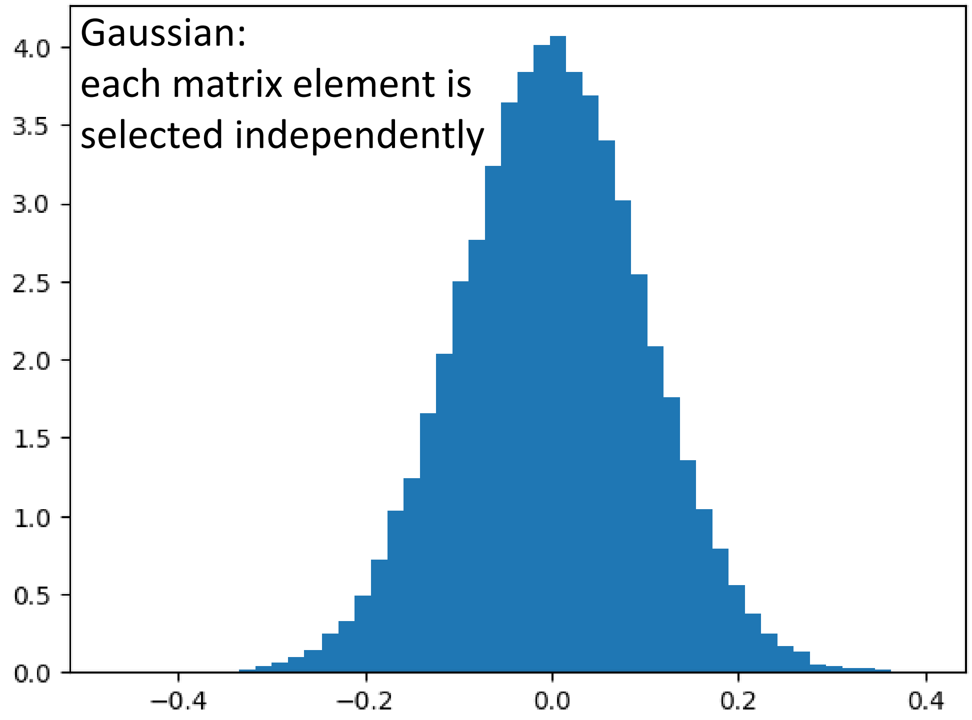
$$|W^i| = \mathcal{O}(0.01) \Longrightarrow |z^L| \to 0$$

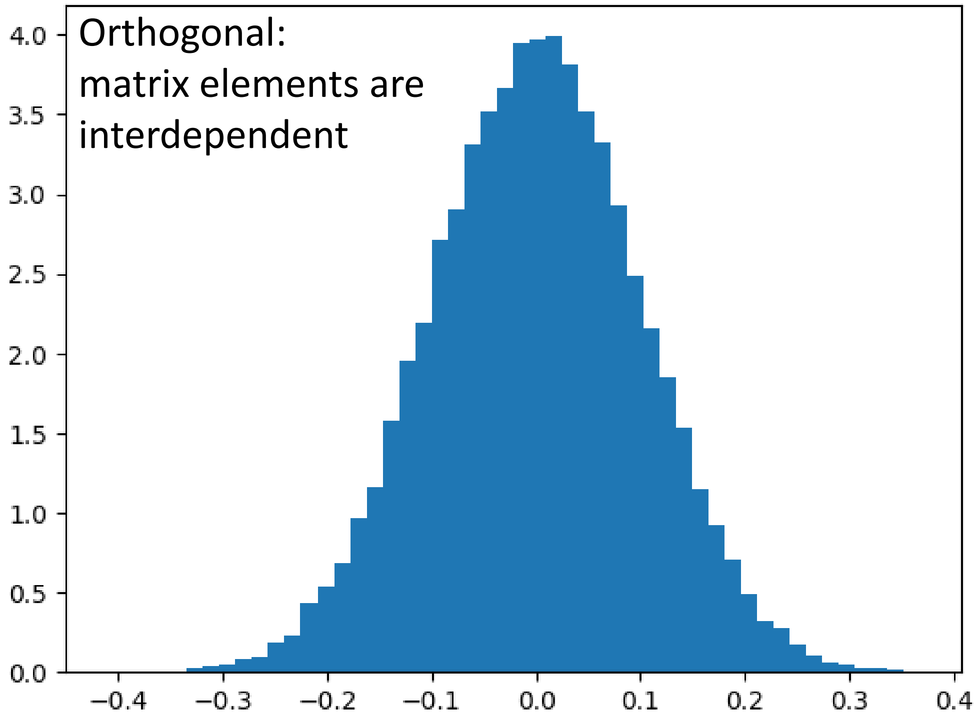
- Network cannot learn in either case
- *Tuning to criticality* means preventing exploding and decaying signals

*This maximizes the number of marginal couplings (couplings that do not grow or shrink), but does not guarantee that *all* couplings are marginal

Gaussian vs orthogonal weights

- Gaussian distribution is the limiting distribution – all distributions become Gaussian at infinite width
- Orthogonal distribution corresponds to points on a sphere





Comparing weight initialization distributions

Recall

- Initial weights and biases are randomly selected from a distribution
- Infinite-width neural network = free (Gaussian) field theory

- Finite width $\Longrightarrow$ interactions

$$\underbrace{\begin{pmatrix} z^{\ell+1} \end{pmatrix}}_{\text{layer vector}} = \underbrace{\begin{pmatrix} b^{\ell+1} \end{pmatrix}}_{\text{bias vector}} + \underbrace{\begin{pmatrix} \quad W^{\ell+1} \quad \end{pmatrix}}_{\text{weight matrix}} \begin{pmatrix} \sigma(z^{\ell}) \end{pmatrix}$$

Comparing weight initialization distributions

- Initial weights and biases are randomly selected from a distribution
- Infinite-width neural network = free (Gaussian) field theory

  All distributions become Gaussian at infinite width

- Finite width $\Longrightarrow$ interactions

  Non-Gaussian initializations are perturbations away from (infinite-width) Gaussian limit

$$\overset{\text{layer vector}}{\left(z^{\ell+1} \right)} = \overset{\text{bias vector}}{\left(b^{\ell+1} \right)} + \overset{\text{weight matrix}}{\left(\quad W^{\ell+1} \quad \right)} \left(\sigma(z^{\ell}) \right)$$

Comparing weight initialization distributions

- Initial weights and biases are randomly selected from a distribution
- Infinite-width neural network = free (Gaussian) field theory

➡ All distributions become Gaussian at infinite width

- Finite width $\Longrightarrow$ interactions

➡ Non-Gaussian initializations are perturbations away from (infinite-width) Gaussian limit

*even Gaussian initializations become non-Gaussian at finite-width!

Perhaps introducing physically-motivated interactions to our network will improve learning