香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

CSC6052/5051/4100/DDA6307/ MDS5110
Natural Language Processing

Lecture 6: Language Models (in a broader sense).

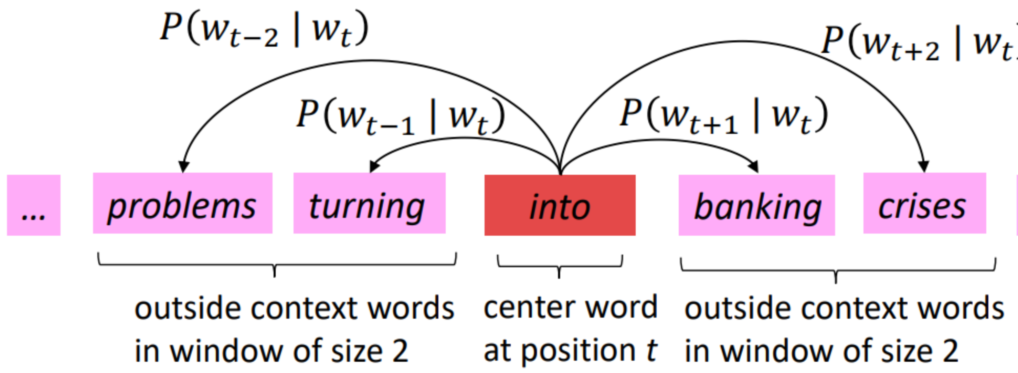
Spring 2025

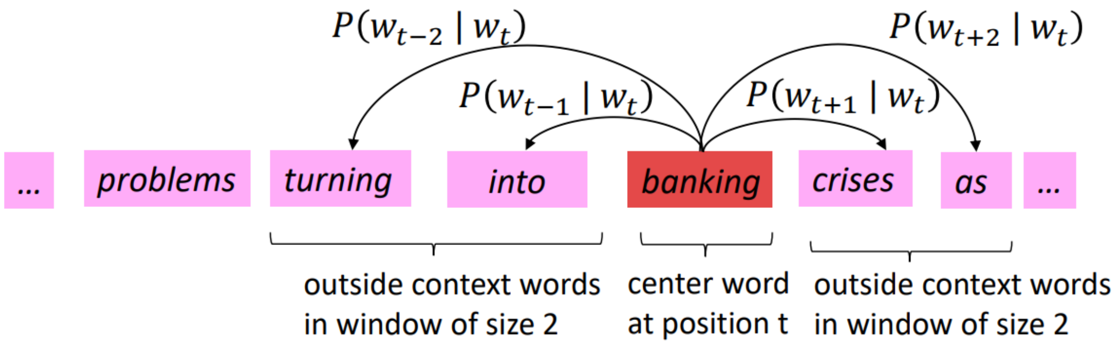
Benyou Wang
School of Data Science

To recap….

Word2Vec Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors





https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture01-wordvecs1.pdf

What’s wrong with word2vec?

- One vector for each word type

$$v(\text{bank}) = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix}$$

- Complex characteristics of word use: semantics, syntactic behavior, and connotations

- Polysemous words, e.g., bank, mouse

mouse^1 : .... a *mouse* controlling a computer system in 1968.
mouse^2 : .... a quiet animal like a *mouse*
bank^1 : ...a *bank* can hold the investments in a custodial account ...
bank^2 : ...as agriculture burgeons on the east *bank*, the river ...

Static vs. Contextualized

- **Problem**: Word embeddings are applied in a context free manner

open a **bank** account

on the river **bank**

[0.3, 0.2, -0.8, …]

- **Solution**: Train *contextual* representations on text corpus

[0.9, -0.2, 1.6, …]

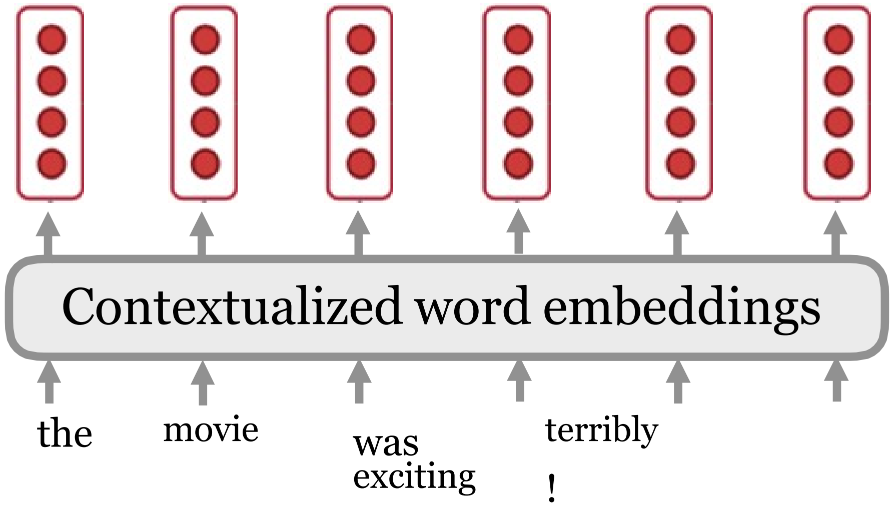
open a **bank** account

[-1.9, -0.4, 0.1, …]

on the river **bank**

Contextualized word embeddings

Let's build a vector for each word conditioned on its **context**!

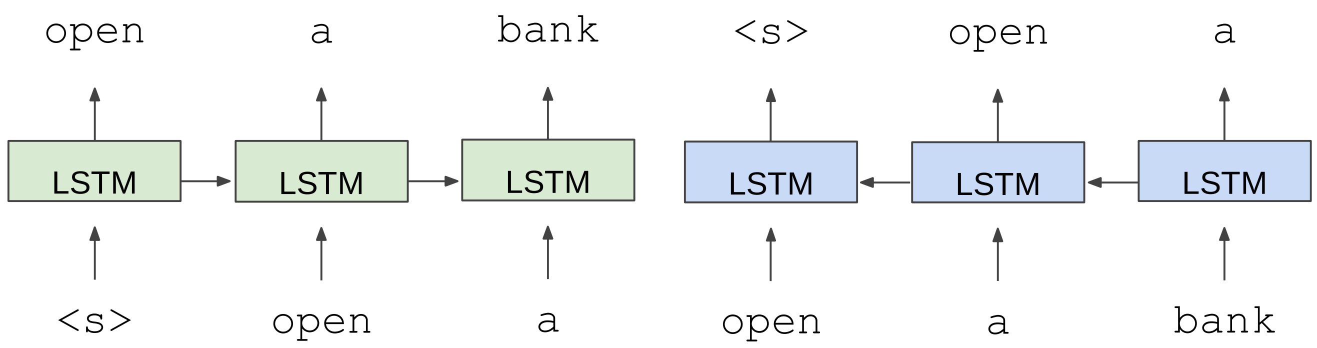


$$f : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

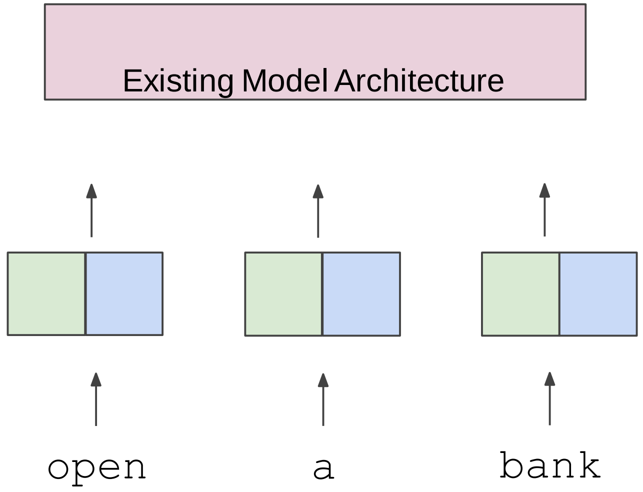
From static word vector to contextualized word vectors

- *ELMo: Deep Contextual Word Embeddings*, AI2 & University of Washington, 2017

Train Separate Left-to-Right and Right-to-Left LMs



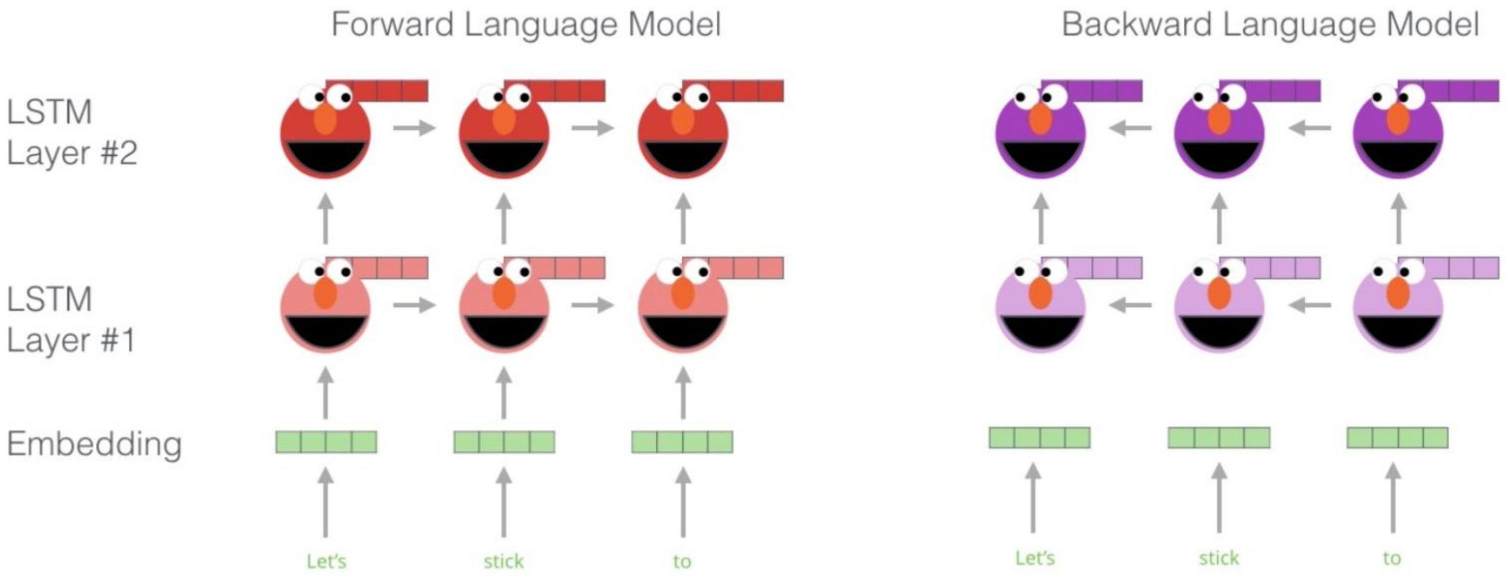
Apply as “Pre-trained Embeddings”



ELMo

- NAACL'18: Deep contextualized word representations
- Key idea:
  - Train an LSTM-based language model on some large corpus
  - Use the hidden states of the LSTM for each token to compute a vector representation of each word

ELMo



words in the sentence

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

input

softmax

How to use ELMo?

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \leftarrow \text{\# of layers} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{h}_{k,0}^{lM} = \mathbf{x}_k^{LM}, \mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]
$$

$$
\boxed{\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}}
$$

- γ^{task}: allows the task model to scale the entire ELMo vector
- s_j^{task}: softmax-normalized weights across layers
- Plug ELMo into any (neural) NLP model: freeze all the LMs weights and change the input representation to:

$$
[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]
$$

(could also insert into higher layers)

Use ELMo in practice

https://allennlp.org/elmo

Pre-trained ELMo Models

Model	Link(Weights/Options File)		# Parameters (Millions)	LSTM Hidden Size/Output size	# Highway Layers>
Small	weights	options	13.6	1024/128	1
Medium	weights	options	28.0	2048/256	1
Original	weights	options	93.6	4096/512	2
Original (5.5B)	weights	options	93.6	4096/512	2

```
from allennlp.modules.elmo import Elmo, batch_to_ids

options_file = "https://allennlp.s3.amazonaws.com/models/elmo/2x409
weight_file = "https://allennlp.s3.amazonaws.com/models/elmo/2x4096_

# Compute two different representation for each token.
# Each representation is a linear weighted combination for the
# 3 layers in ELMo (i.e., charcnn, the outputs of the two BiLSTM))
elmo = Elmo(options_file, weight_file, 2, dropout=0)

# use batch_to_ids to convert sentences to character ids
sentences = [['First', 'sentence', '.'], ['Another', '.']]
character_ids = batch_to_ids(sentences)

embeddings = elmo(character_ids)
```

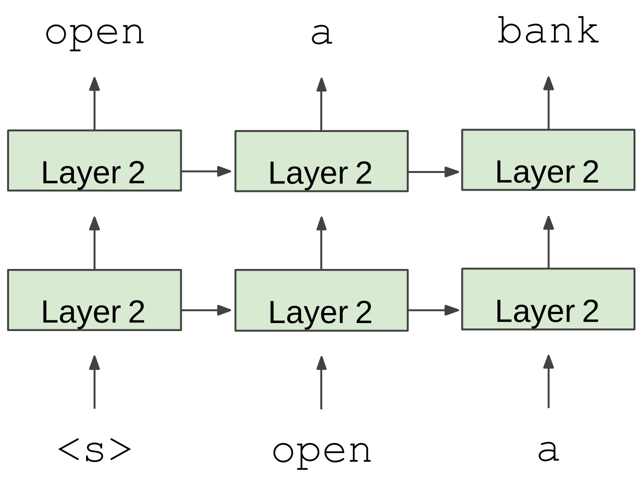
Also available in TensorFlow

How to use ELMo?

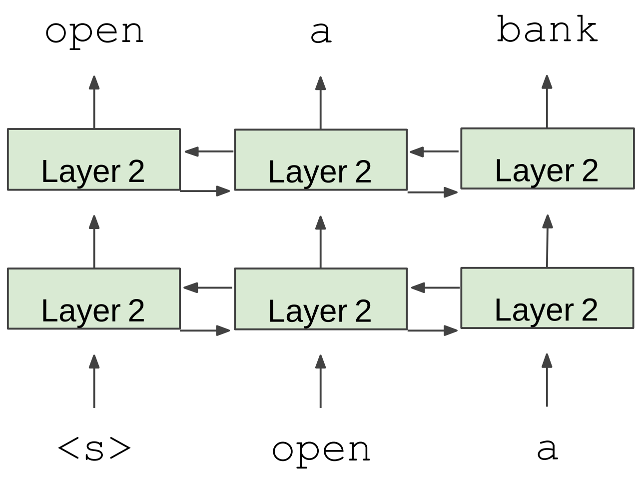
- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
  - We don't care about this.
- Reason 2: Words can "see themselves" in a bidirectional encoder.

Unidirectional vs. Bidirectional Models

Unidirectional context
Build representation incrementally



Bidirectional context
Words can “see themselves”



BERT

- First released in Oct 2018.
- NAACL'19: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

How is BERT different from ELMo?

#1. Unidirectional context vs bidirectional context

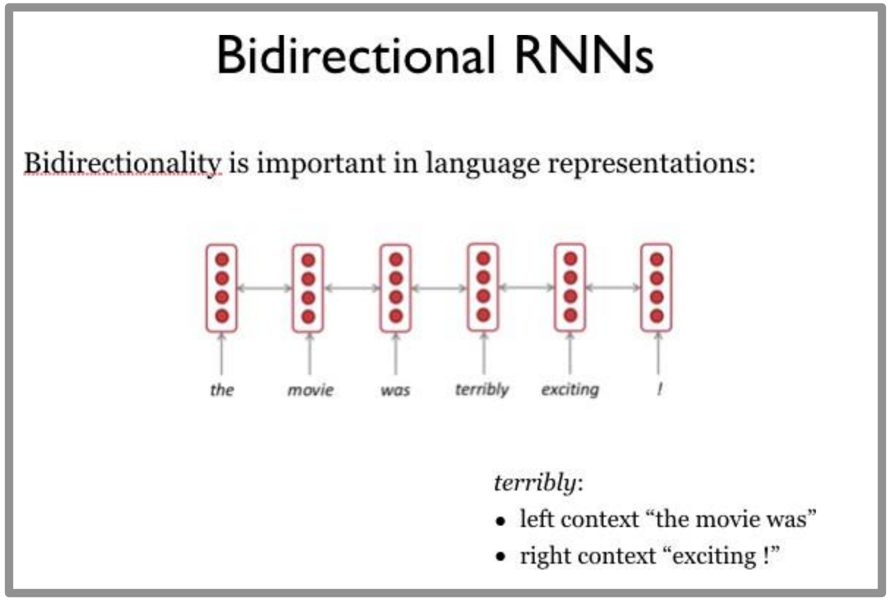
#2. LSTMs vs Transformers (will talk later)

#3. The weights are not freezed, called fine-tuning

Bidirectional encoders

- Language models only use left context or right context (although ELMo used two independent LMs from each direction).
- Language understanding is bidirectional

Lecture 9:

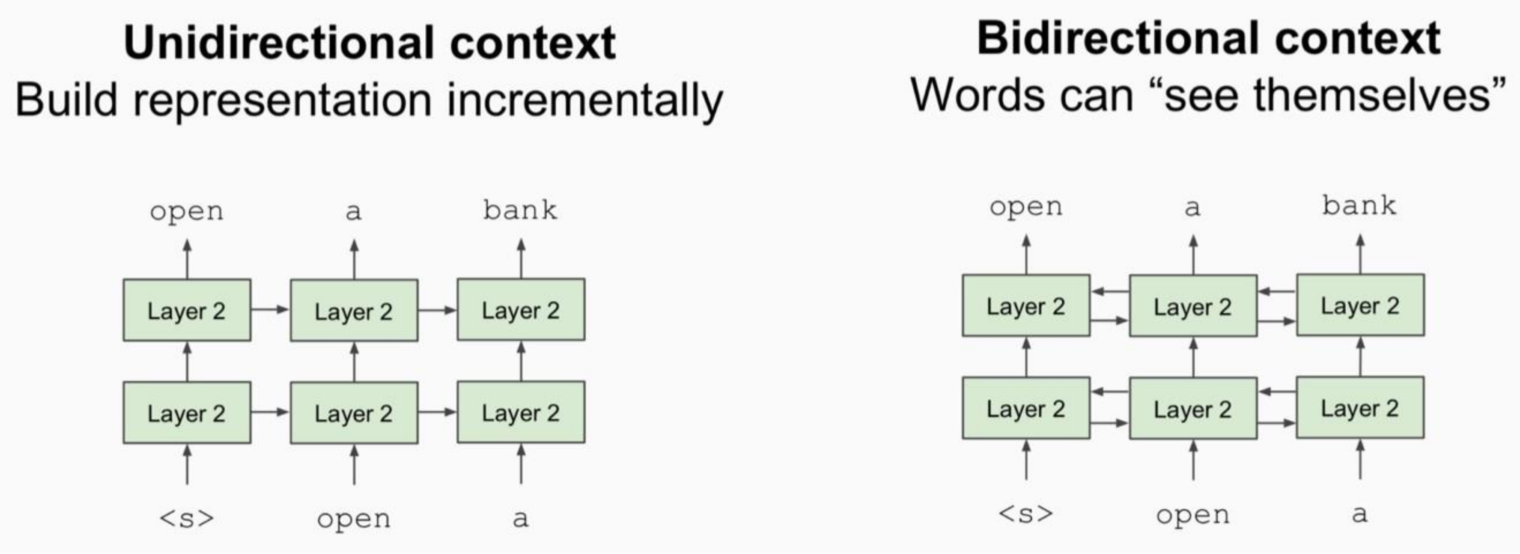


Why are LMs unidirectional?

Bidirectional encoders

- Language models only use left context or right context (although ELMo used two independent LMs from each
- direction).

  Language understanding is bidirectional



Masked language models (MLMs)

- Solution: Mask out 15% of the input words, and then predict the masked words

- Too little masking: too expensive to train
- Too much masking: not enough context

Masked language models (MLMs)

A little more complication:

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

We probably would not see [mask] in downstream tasks.

Usually, [MASK] would not exist when BERT is used in downstream tasks

Next sentence prediction (NSP)

Always sample two sentences, predict whether the second sentence is followed after the first one.

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]

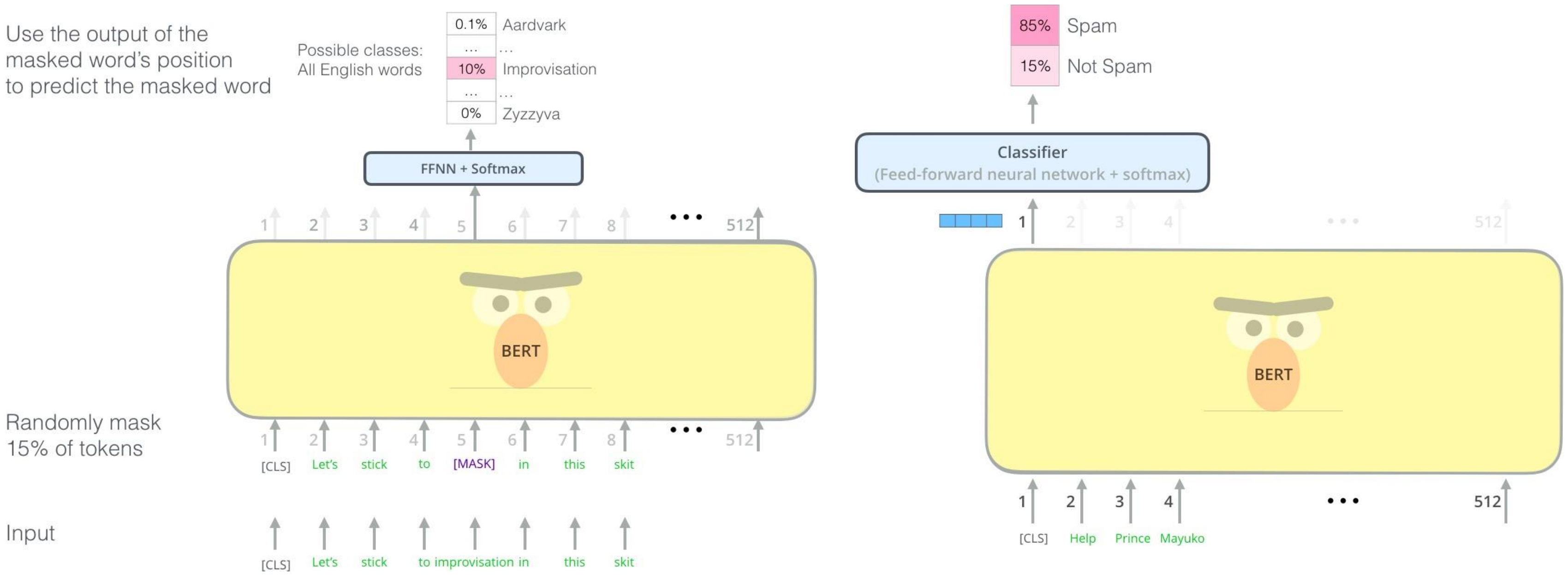
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

Recent papers show that NSP is not necessary, probably it becomes saturated quickly

(Joshi*, Chen* et al, 2019) :SpanBERT: Improving Pre-training by Representing and Predicting Spans (Liu et al, 2019): RoBERTa: A Robustly Optimized BERT Pretraining Approach

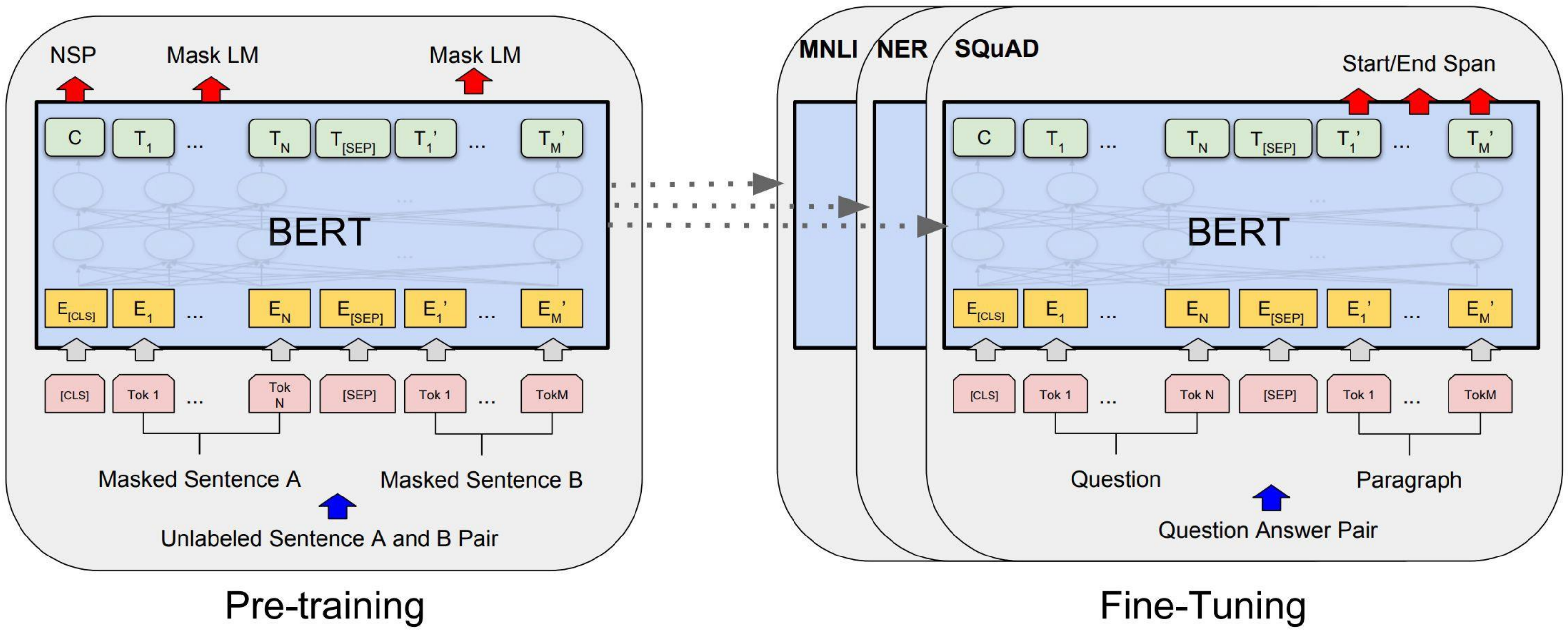
Pre-training and fine-tuning



Pre-training Fine-tuning

Key idea: all the weights are fine-tuned on downstream tasks

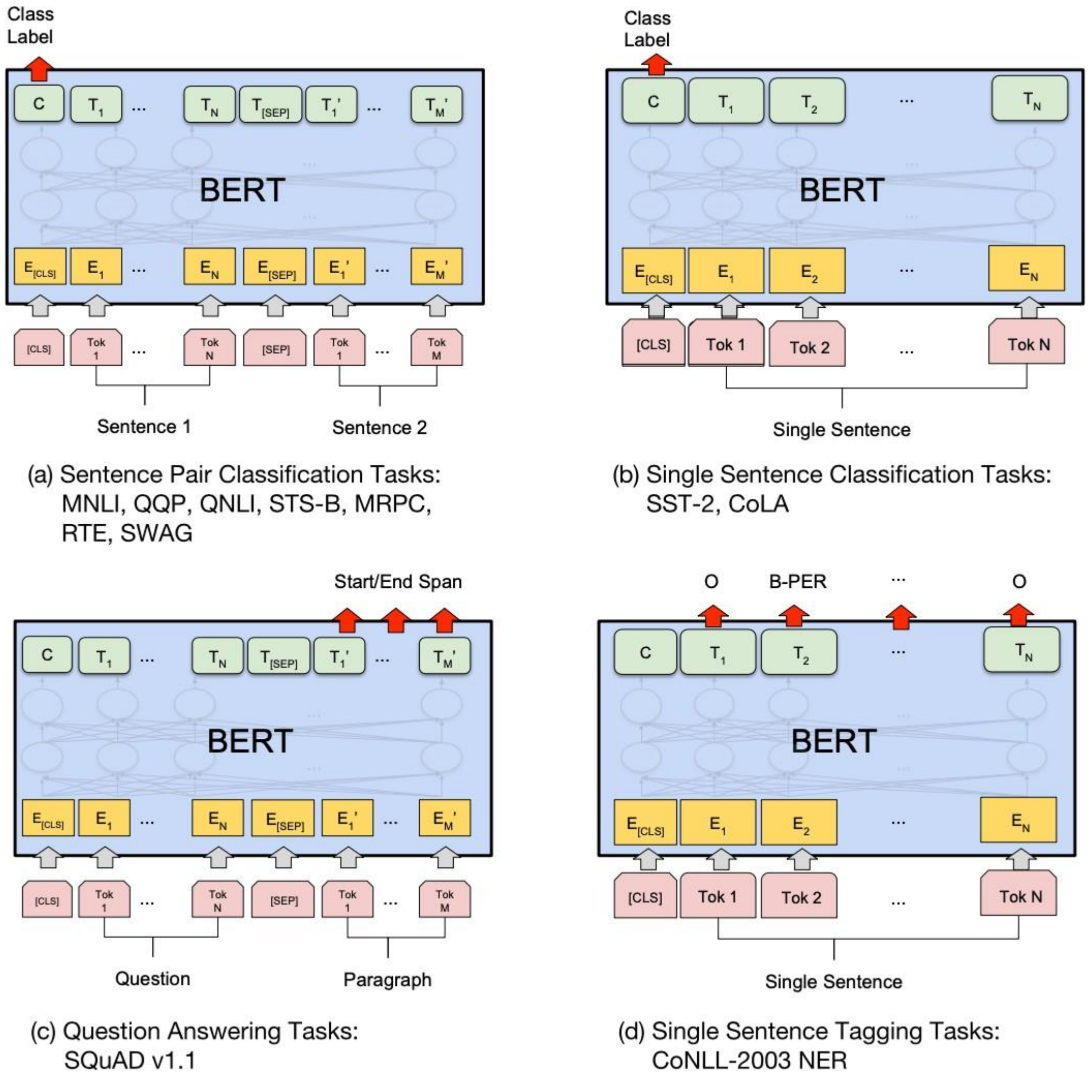
One pre-trained models is adapted everywhere



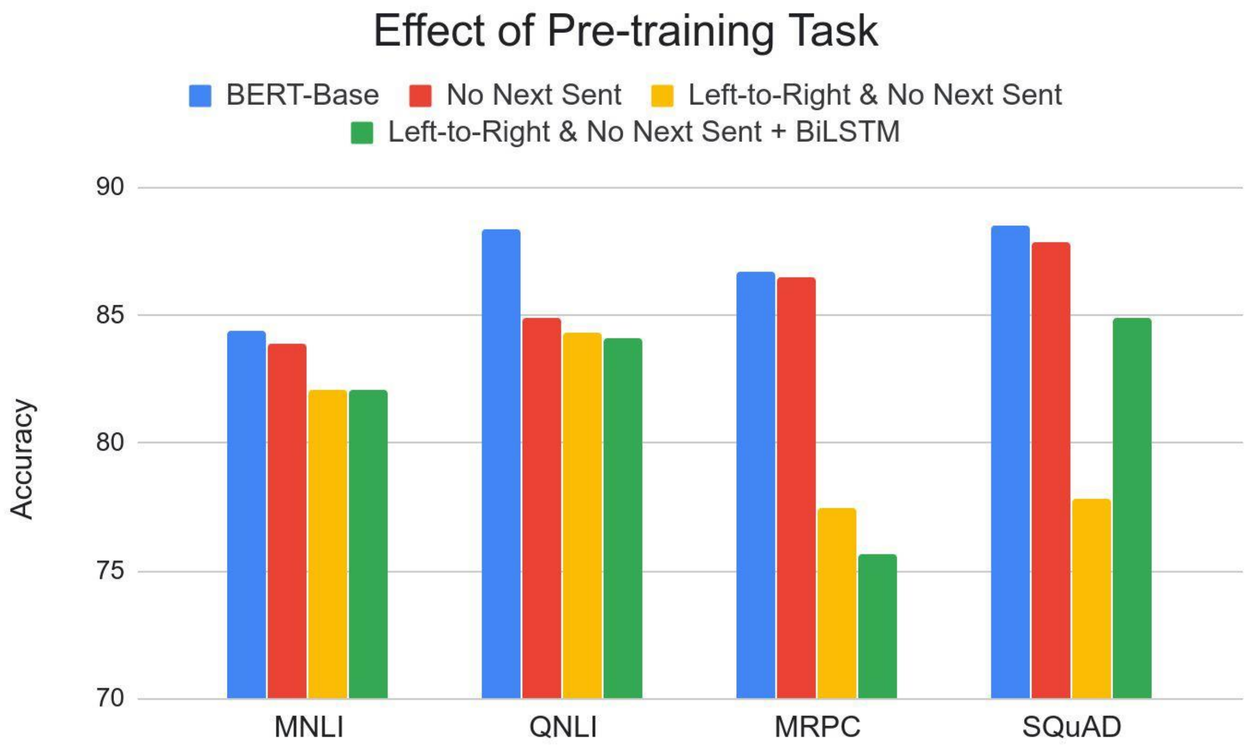
Maybe this is one of the first popular **Foundation model**

On the Opportunities and Risks of Foundation Models. https://arxiv.org/abs/2108.07258

Applications

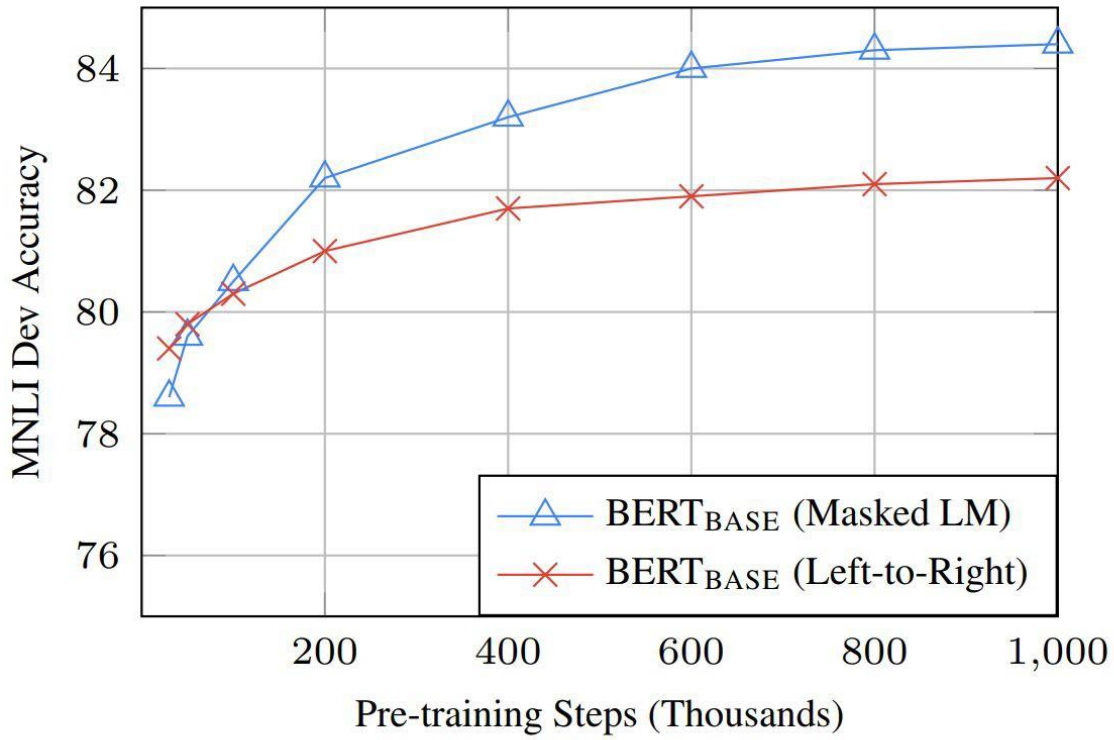


Effect of Pre-training Task



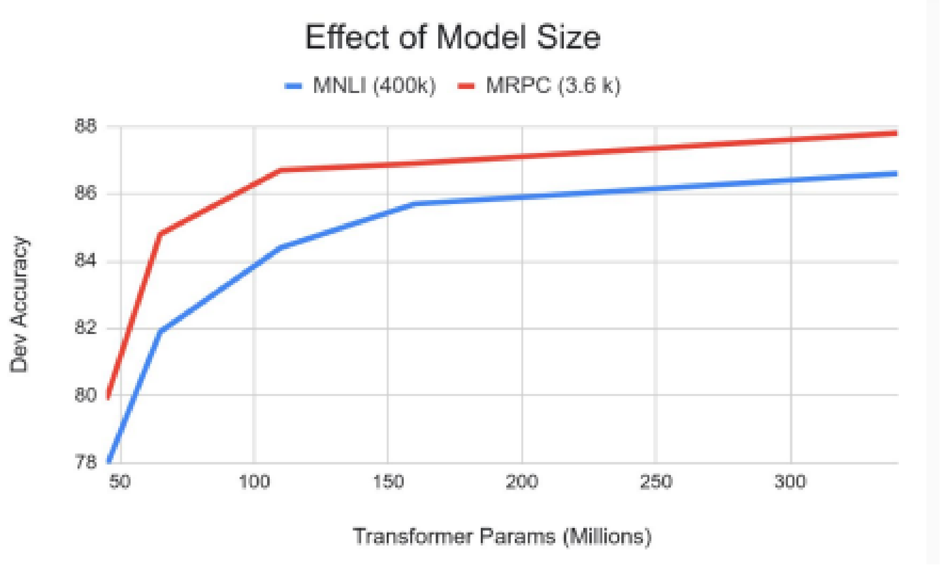
- Masked LM (compared to left-to-right LM) is very important on some tasks, Next Sentence Prediction is important on other tasks.
- Left-to-right model does very poorly on word-level task (SQuAD), although this is mitigated by BiLSTM

Directionality helps



- Masked LM takes slightly longer to converge because we only predict 15% instead of 100%
- But absolute results are much better almost immediately

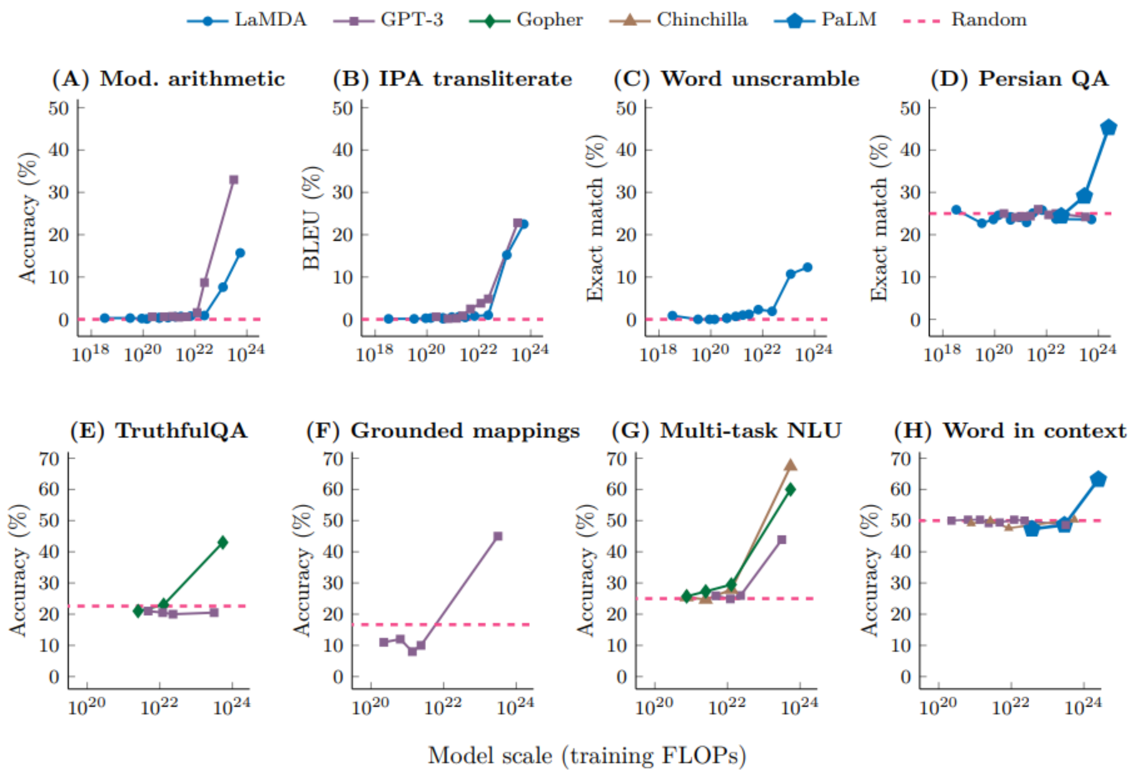
Scalability - BERT



It seems BERT cannot benefit that much from scaling;

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/slides/Jacob_Devlin_BERT.pdf

Scalability - GPT



GPT scales better!

Reason： generation scales, but no for discrimination

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/slides/Jacob_Devlin_BERT.pdf

From BERT/ELMO to more “general”“ language models

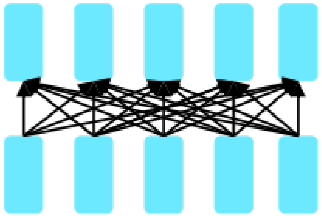
Overview

Model	Type	Architecture	Task
NLM [25]	static	1-layer MLP	$(a, b) \rightarrow c$ predicting the next word
Skip-Gram [200]	static	1-layer MLP	$b \rightarrow c, \quad b \rightarrow a$ predicting neighboring words
CBow [200]	static	1-layer MLP	$(a, c) \rightarrow b$ predicting central words
Glove [227]	static	1-layer MLP	$\vec{w_i}^T \vec{w_j} \propto logp(\#(w_i w_j))$ predicting the log co-occurrence count
ELMO [230]	contextualized	LSTM	$(a, b, c, d) \rightarrow e, \quad (e, d, c, b) \rightarrow a$ bi-directional language model
BERT [66], Roberta [185] ALBERT [154],XLNET [351]	contextualized	Transformers or Transformer-XL	$(a, [\texttt{mask}], c) \rightarrow (\_, b, \_)$ predicting masked words
Electra [54]	contextualized	Transformer	$(a, \hat{b}, c, \hat{d}) \rightarrow (0, 1, 0, 1)$ replaced token prediction
T5 [241] BART [158]	contextualized	Transformers	$(a, b, c,) \rightarrow (d, e)$ predicting the sequence
GPT [240]	contextualized	Transformers	$(a, b, c, d) \rightarrow e$ autoregressively predicting the next word

Benyou Wang et.al. Pre-trained Language Models in Biomedical Domain: A Systematic Survey. ACM Computing Survey.

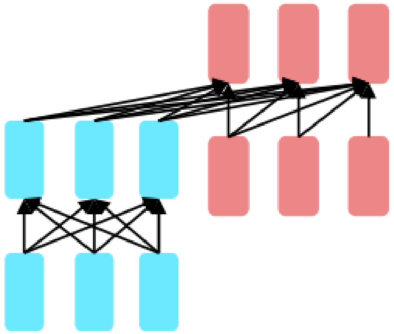
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



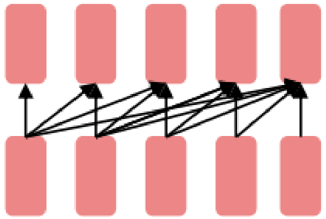
Encoders

- **Gets bidirectional context – can condition on future!**
- **How do we train them to build strong representations?**



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

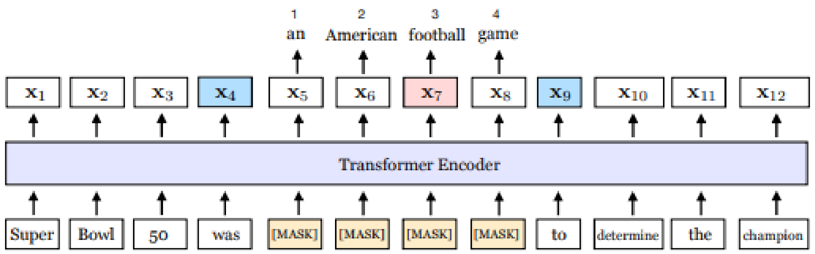
RoBERTA

- *RoBERTa: A Robustly Optimized BERT Pretraining Approach* (Liu et al, University of Washington and Facebook, 2019)
- Trained BERT for more epochs and/or on more data
  - Showed that more epochs alone helps, even on same data
  - More data also helps
- Improved masking and pre-training data slightly

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

SpanBERT

- *RoBERTa: SpanBERT: Improving Pre-training by Representing and Predicting Spans* (Joshi et al, 2019)
- Mask a whole Span



- Span masking helps

	SQuAD 1.1 EM	SQuAD 1.1 F1	SQuAD 2.0 EM	SQuAD 2.0 F1
Human Perf.	82.3	91.2	86.8	89.4
Google BERT	84.3	91.3	80.0	83.3
Our BERT	86.5	92.6	82.8	85.9
Our BERT-1seq	87.5	93.3	83.8	86.6
SpanBERT	**88.8**	**94.6**	**85.7**	**88.7**

BERT-wwm

- Pre-Training with Whole Word Masking for Chinese, Cui et.al. 2019

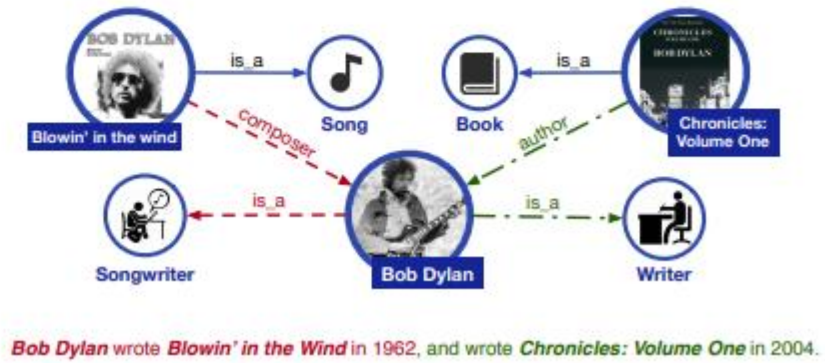
- Mask a whole Chinese work

	Chinese	English
Original Sentence	使用语言模型来预测下一个词的概率。	we use a language model to predict the probability of the next word.
+ CWS	语言 模型 来 预测 下 一个 词 的 概率 。	-
+ BERT Tokenizer	语 言 模 型 来 预 测 下 一 个 词 的 概 率 。	we use a language **model** to **pre** **##di** **##ct** the **pro** **##ba** **##bility** of the next word .
Original Masking	语 言 **[M]** 型 来 **[M]** 测 下 一 个 词 的 概 率 。	we use a language **[M]** to **[M]** **##di** **##ct** the **pro** **[M]** **##bility** of the next word .
+ WWM	语 言 **[M]** **[M]** 来 **[M]** **[M]** 下 一 个 词 的 概 率 。	we use a language **[M]** to **[M]** **[M]** **[M]** the **[M]** **[M]** **[M]** of the next word .
++ N-gram Masking	**[M]** **[M]** **[M]** **[M]** 来 **[M]** **[M]** 下 一 个 词 的 概 率 。	we use a **[M]** **[M]** to **[M]** **[M]** **[M]** the **[M]** **[M]** **[M]** **[M]** **[M]** next word .
+++ Mac Masking	语 法 建 模 来 预 见 下 一 个 词 的 几 率 。	we use a **text** **system** to **ca** **##lc** **##ulate** the **po** **##si** **##bility** of the next word .

ERNIE

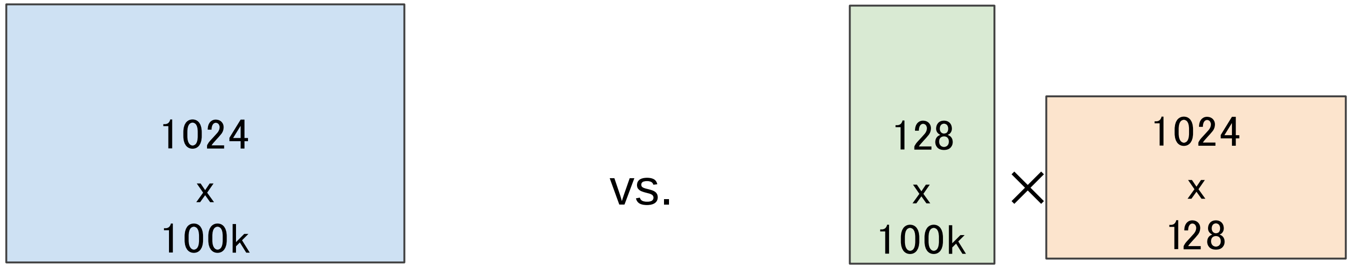
- ERNIE: Enhanced Language Representation with Informative Entities. Zhang et.al 2019

- Mask Informative Entities



ALBERT

- *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations* (Lan et al, Google and TTI Chicago, 2019)
- Innovation #1: Factorized embedding parameterization
  - Use small embedding size (e.g., 128) and then project it to Transformer hidden size (e.g., 1024) with parameter matrix



ALBERT

- Innovation #2: Cross-layer parameter sharing
  - Share all parameters between Transformer layers
- Results:

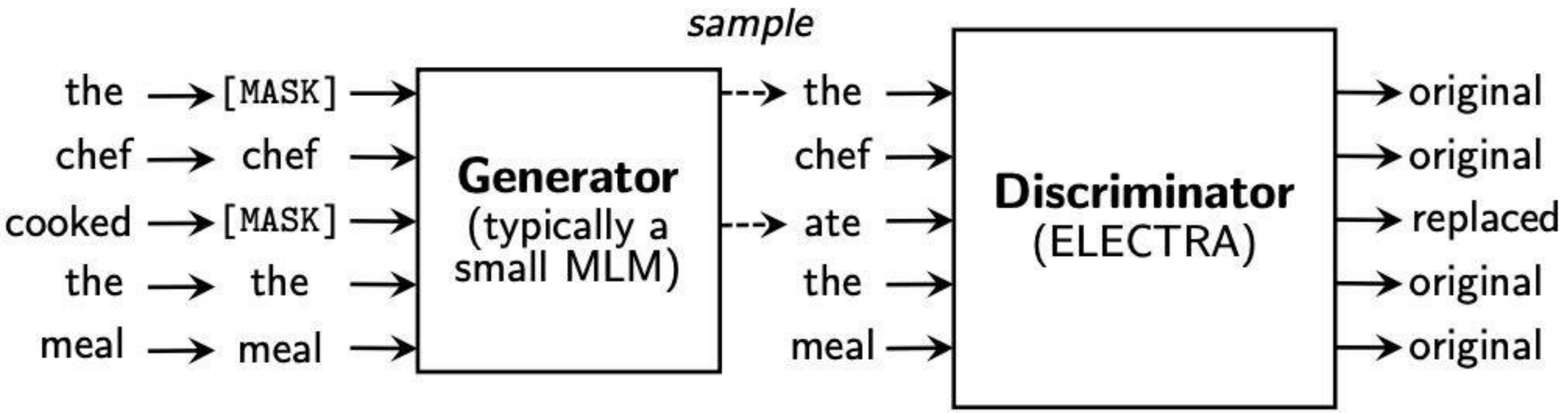
Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
Single-task single models on dev								
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**

- ALBERT is light in terms of *parameters*, not *speed*

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

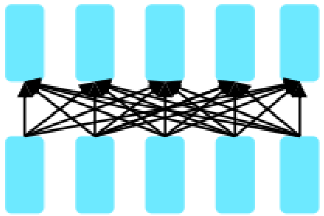
ELECTRA

- *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators* (Clark et al, 2020)
- Train model to discriminate locally plausible text from real text



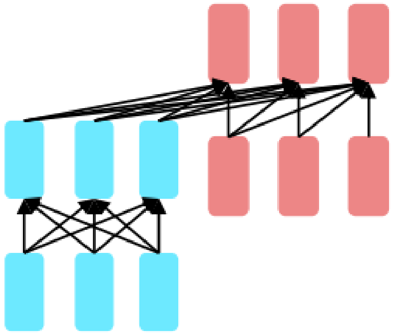
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



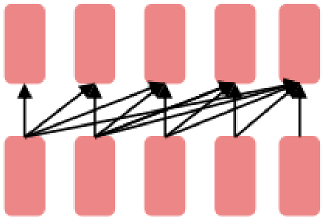
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-
Decoders

- **Good parts of decoders and encoders?**
- **What's the best way to pretrain them?**



Decoders

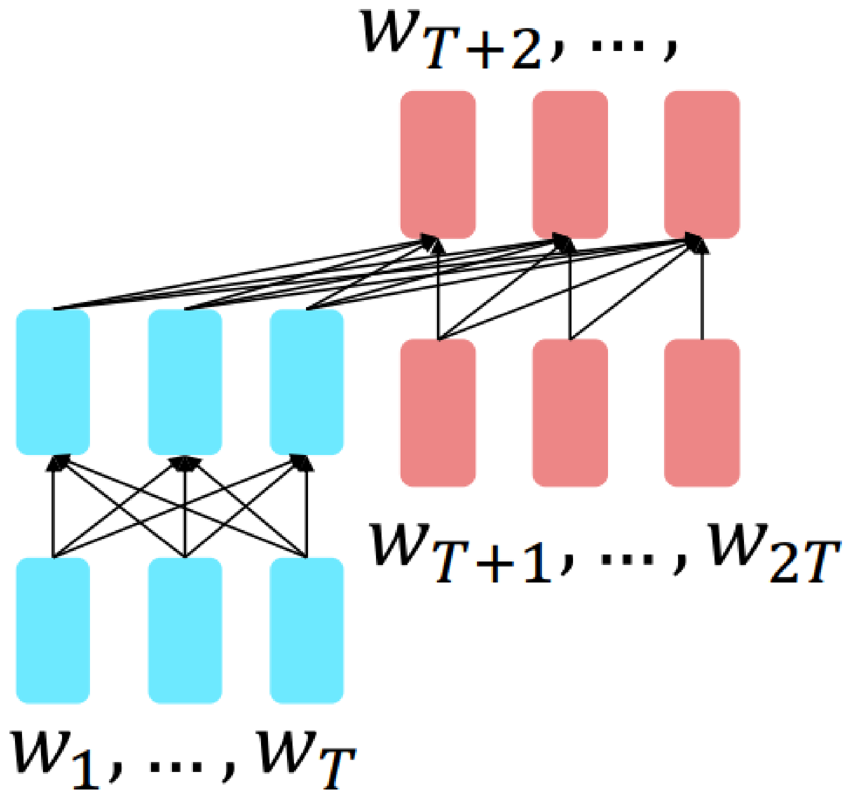
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Pretraining encoder-decoders: what pretraining objective to use?

For **encoder-decoders**, we could do something like **language modeling**, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = \text{Decoder}(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$



The **encoder** portion benefits from **bidirectional** context;
The **decoder** portion is used to train the whole model through language modeling.

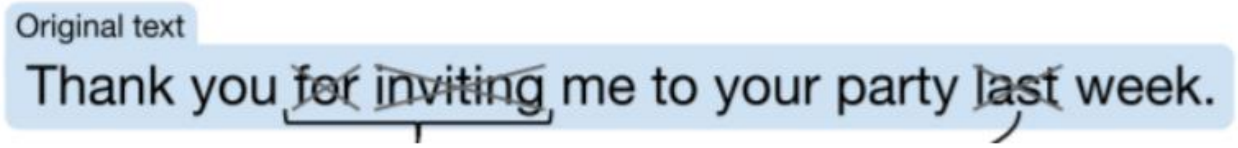
[Raffel et al., 2018]

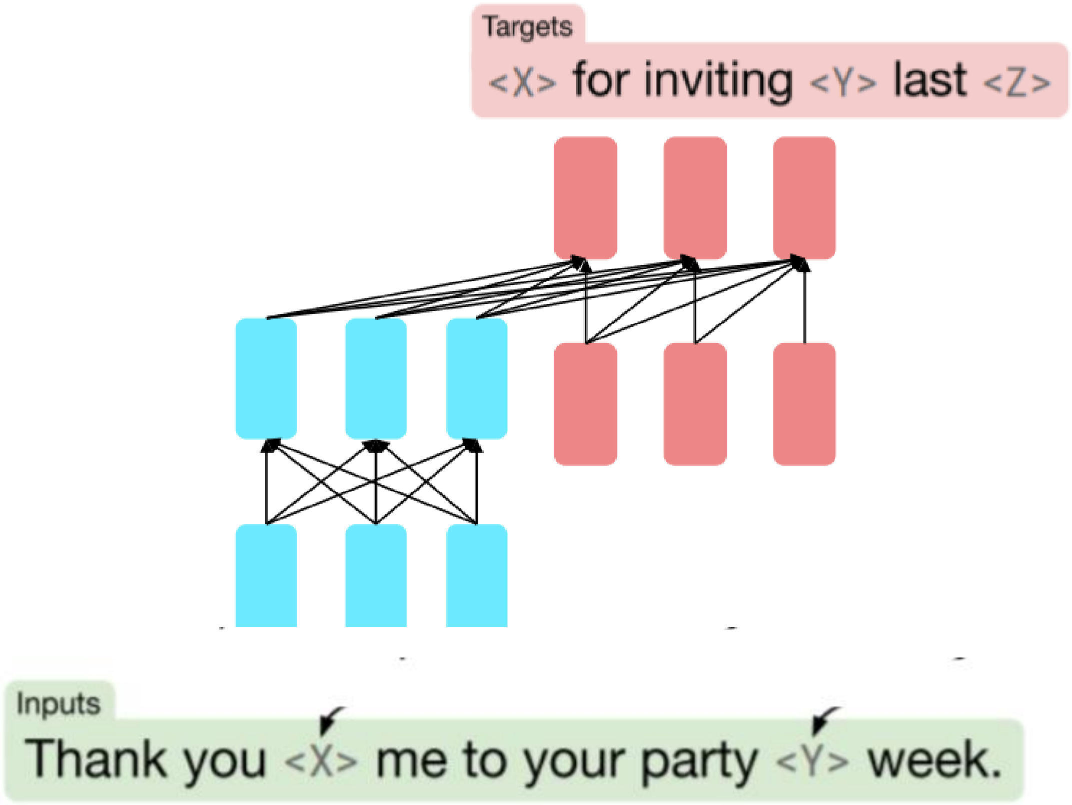
https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Pretraining encoder-decoders: what pretraining objective to use?

What [Raffel et al., 2018](#) found to work best was span corruption. Their model: T5.

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!



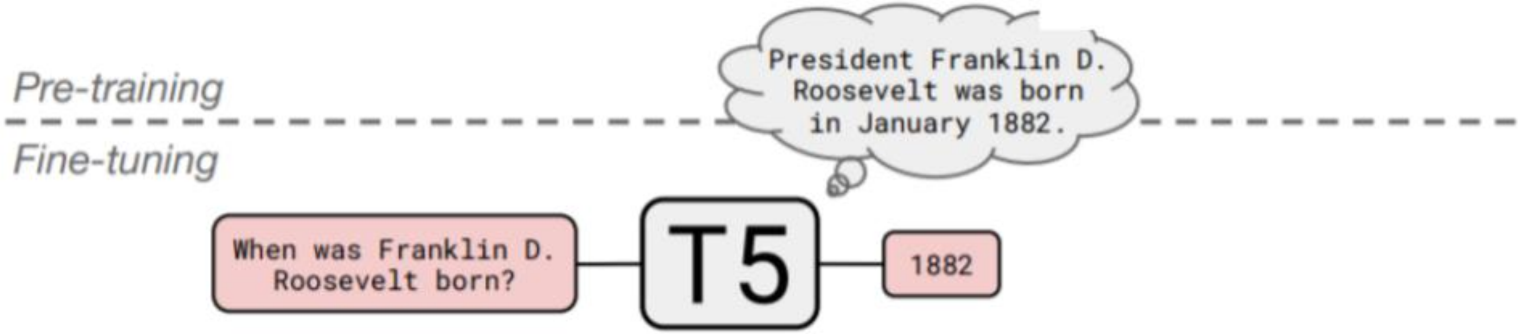


This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.

[Raffel et al., 2018]

Pretraining encoder-decoders: what pretraining objective to use?

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.



We may see a important concept called *instruction tuning, later used in large language models*

[Raffel et al., 2018]

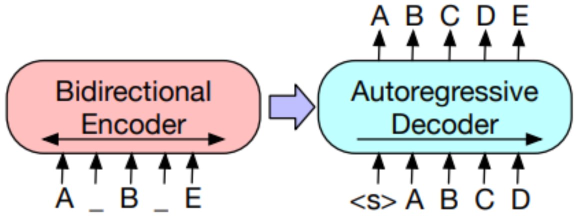
https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Pretraining encoder-decoders: what pretraining objective to use?

BART: **Denoising** Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. https://aclanthology.org/2020.acl-main.703.pdf

(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



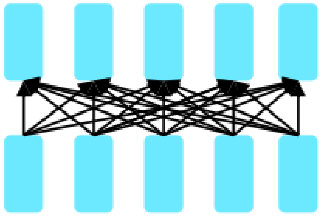
(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with a mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

[Raffel et al., 2018]

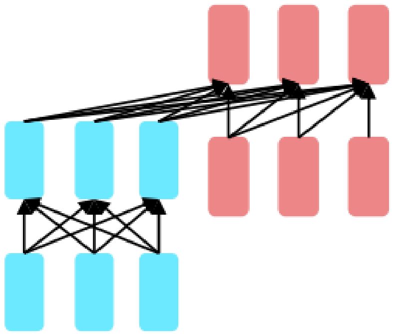
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



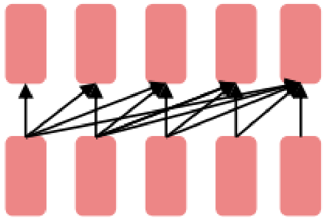
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- **Language models! What we've seen so far.**
- **Nice to generate from; can't condition on future words**

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Back to the language model (next word prediction)

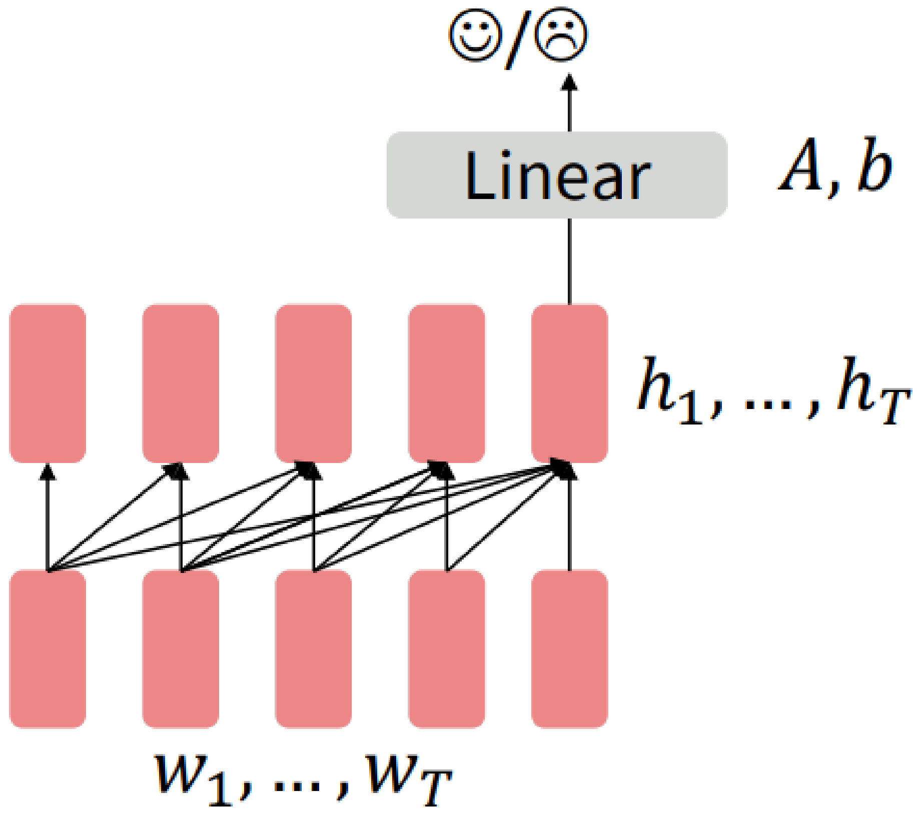
Pretraining decoders

When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t \mid w_{1:t-1})$

We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$y \sim A h_T + b$$

Where A and b are randomly initialized and specified by the downstream task. Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

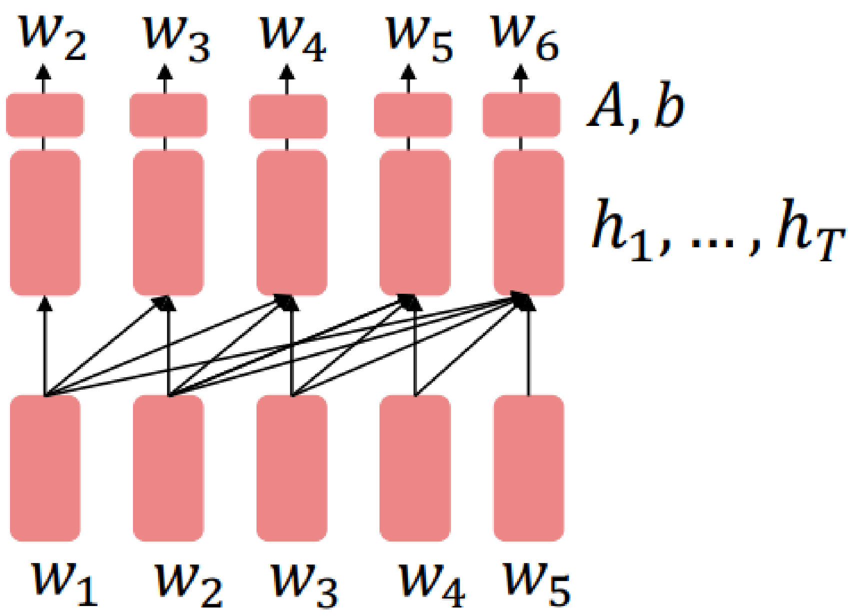
Pretraining decoders

It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t \mid w_{1:t-1})$

This is helpful in tasks **where the output is a sequence** with a vocabulary like that at pretraining time!
- Dialogue (context=dialogue history)
- Summarization (context=document)

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

Where A, b were pretrained in the language model!



[Note how the linear layer has been pretrained.]

Increasingly convincing generations (GPT2) [Radford et al., 2018]

We mentioned how pretrained decoders can be used in their capacities as language models. GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

GPT-3, In-context learning, and very large models

So far, we've interacted with pretrained models in two ways:
• Sample from the distributions they define (maybe providing a prompt)
• Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters. **GPT-3 has 175 billion parameters.**

- LM (next word prediction) is scalable
- LM does not need annotations
- LM is simple such that it is easily to adapt it many tasks
- LM could model human thoughts
- LM is efficient to capture knowledge (imagine use images to record knowledge?)
- Humans do LM everyday (do next-word/ next-second prediction)

What can we learn from reconstructing the input?

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___.

The woman walked across the street, checking for traffic over ___ shoulder.

I went to the ocean to see the fish, turtles, seals, and _____.

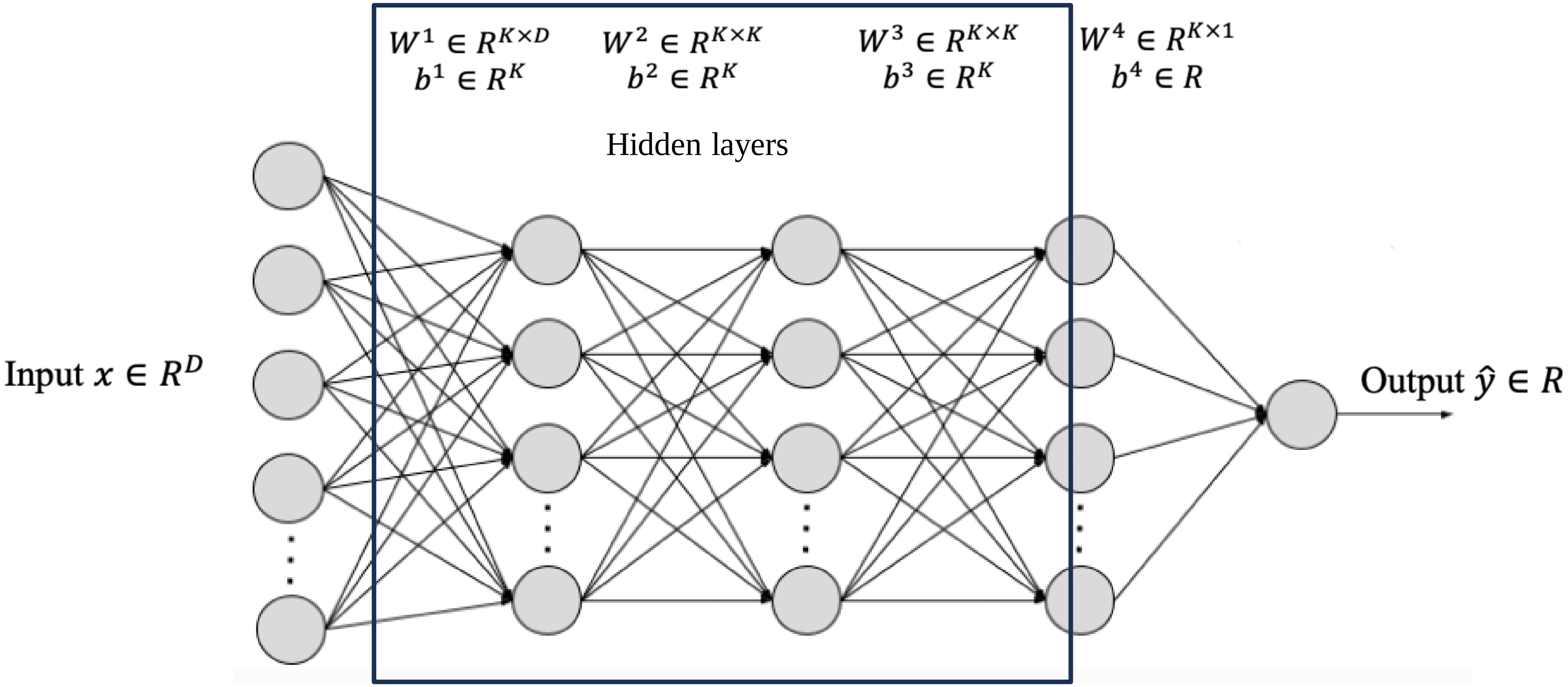
Acknowledgement

- Princeton COS 484: Natural Language Processing. Contextualized Word Embeddings. Fall 2019
- CS447: Natural Language Processing. Language Models. *http://courses.engr.illinois.edu/cs447*

Tutorial 1: Introduction to Overleaf, GitHub, Python, and Pytorch

Pytorch: Neural Network – Forward & Backward Propagation

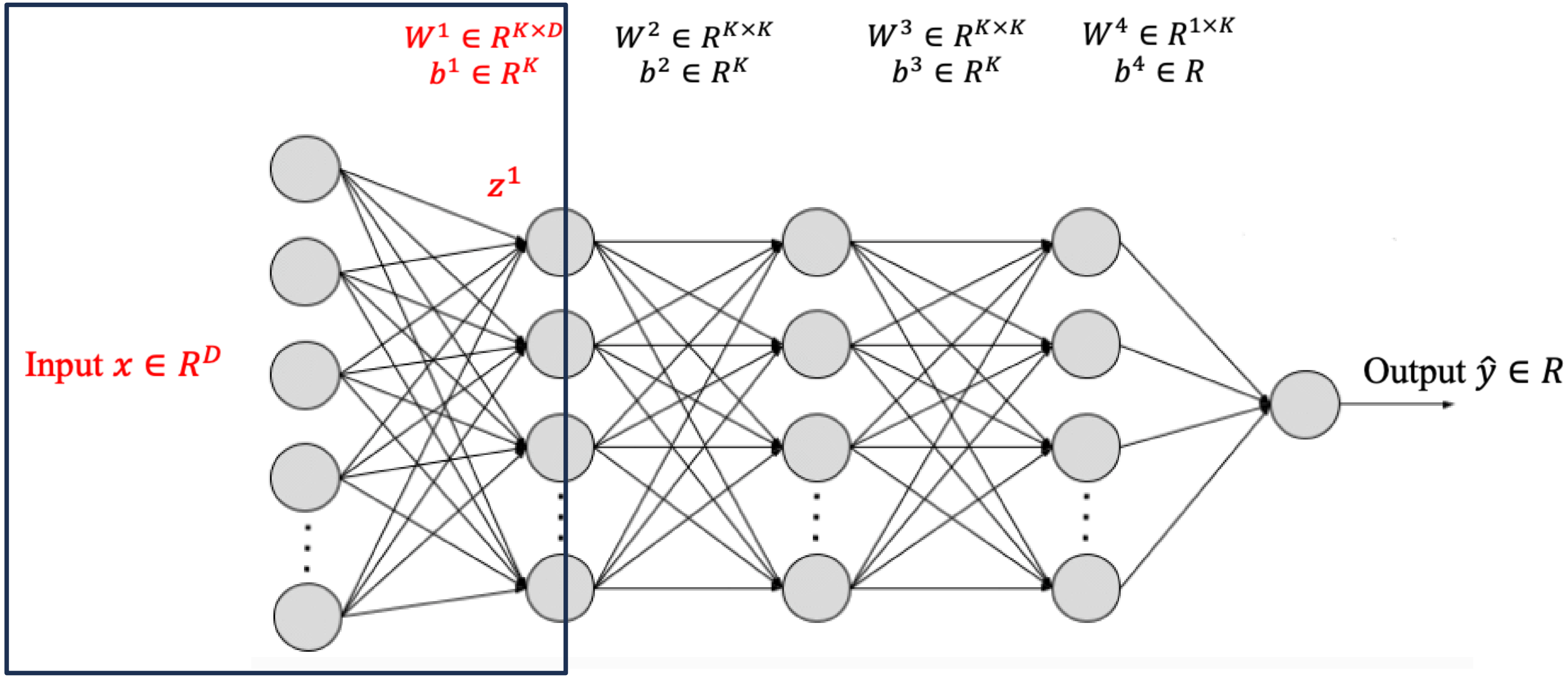
Neural Network



Suppose activation functions here are all sigmoid

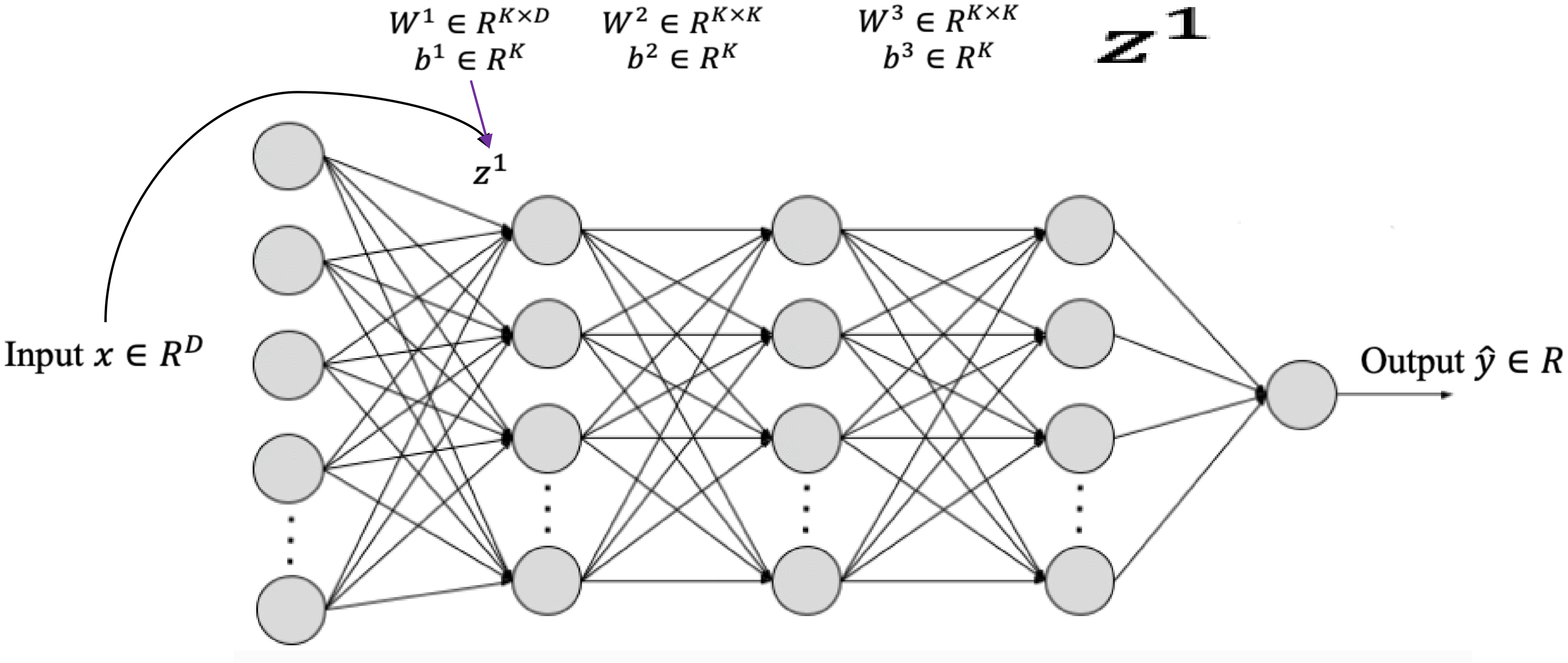
Forward Propagation

Forward Propagation



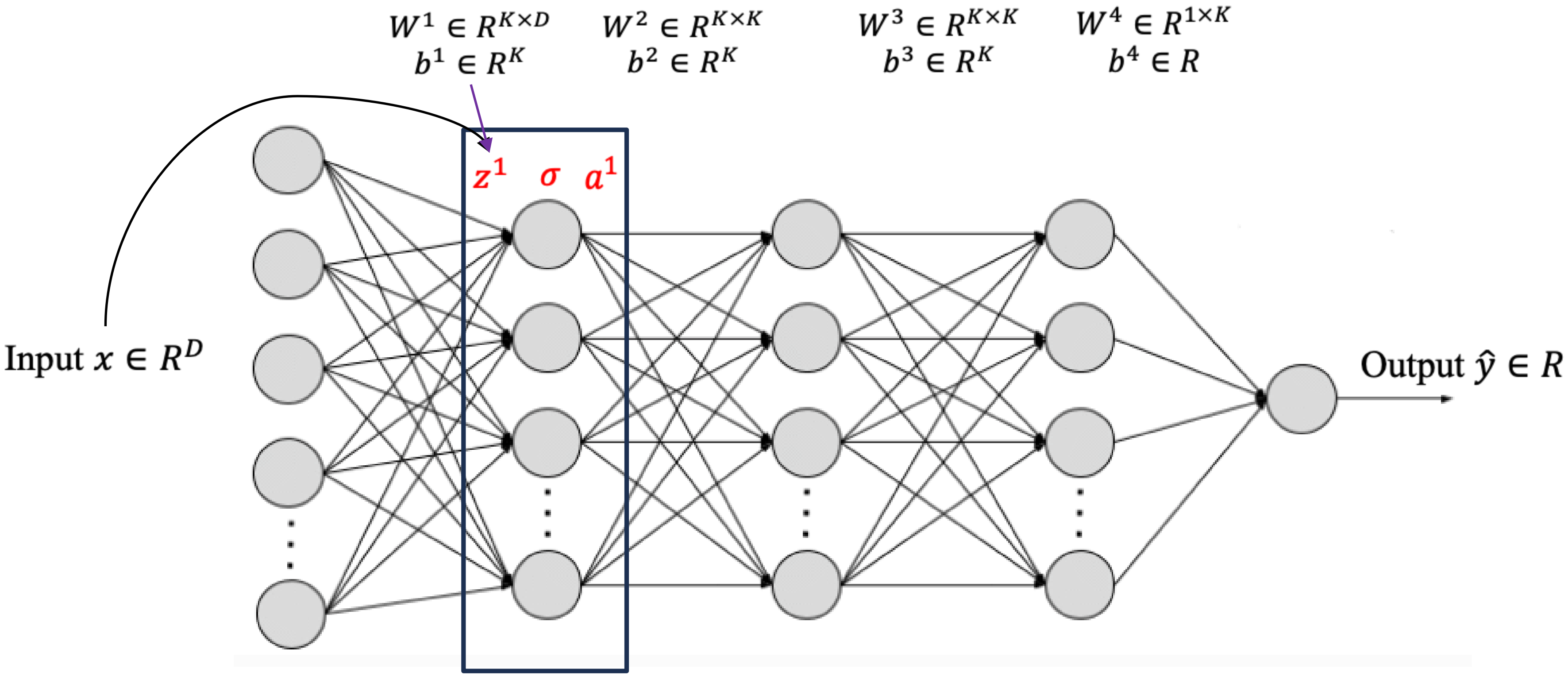
$z^1 = W^1 x + b^1$

Forward Propagation



$z^1 = W^1 x + b^1$

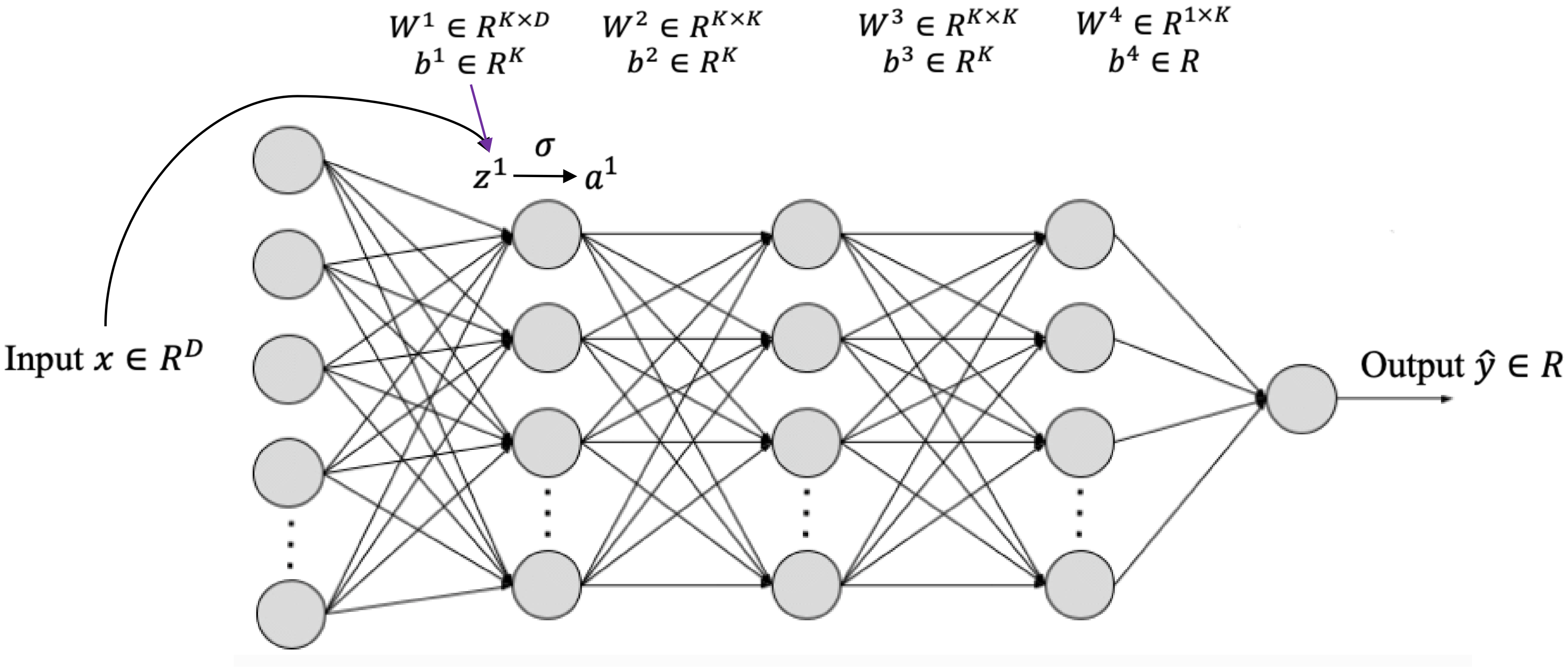
Forward Propagation



$z^1 = W^1x + b^1$

$a^1 = \sigma(z^1)$

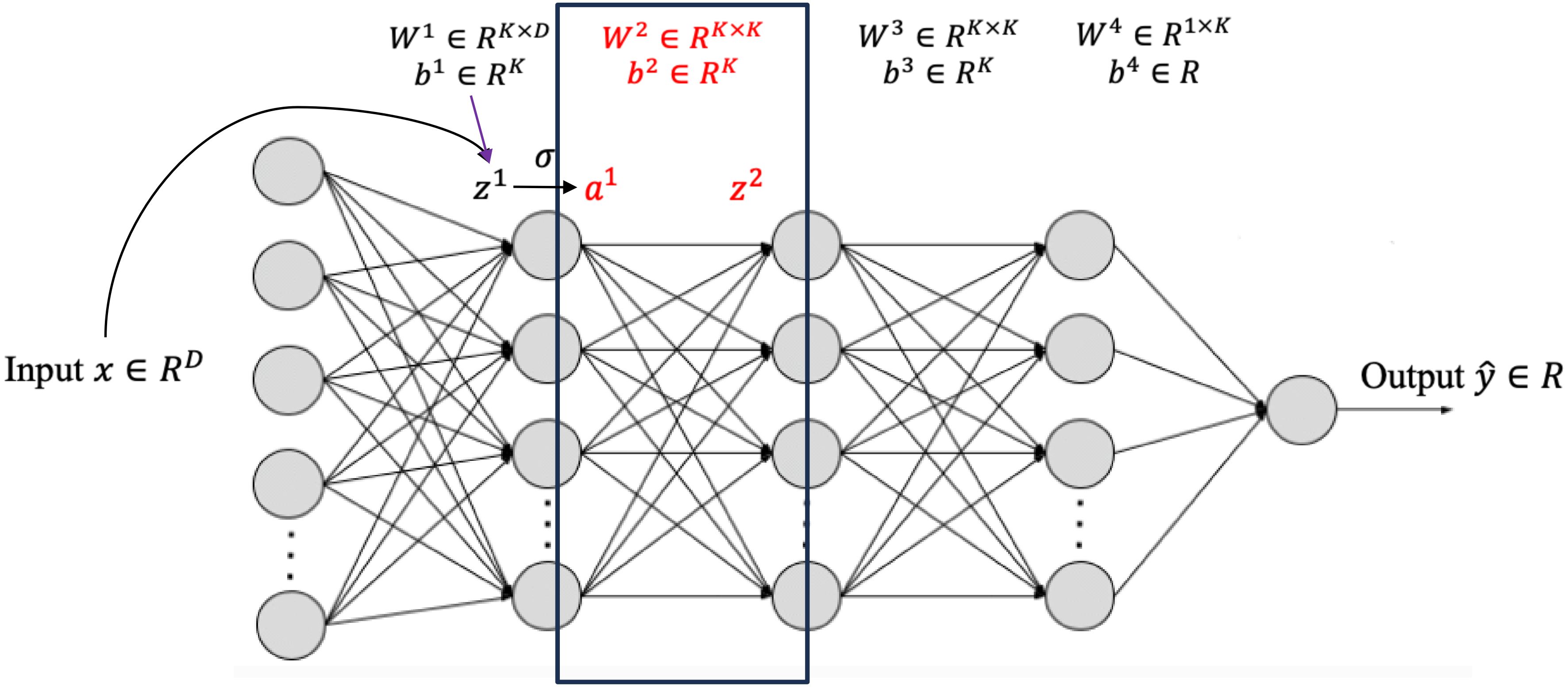
Forward Propagation



$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

Forward Propagation

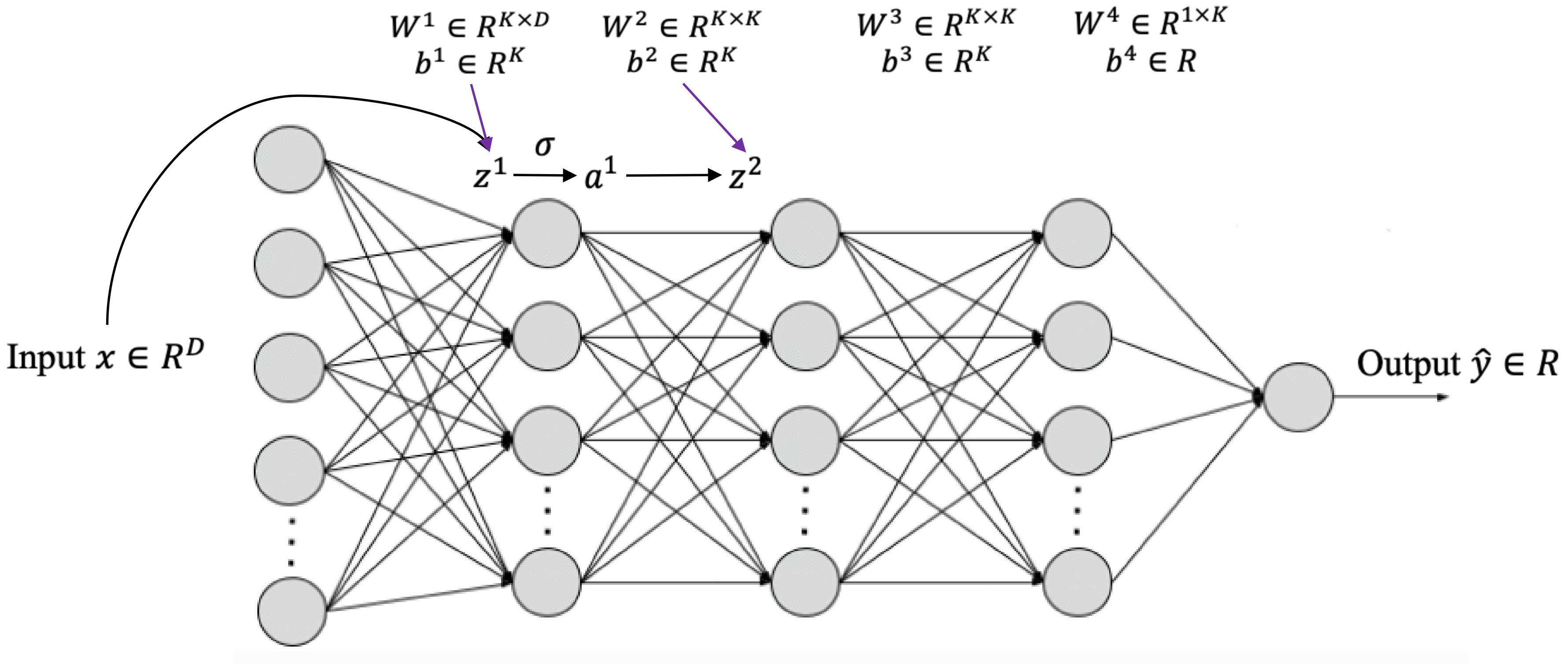


$z^1 = W^1x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2a^1 + b^2$

Forward Propagation

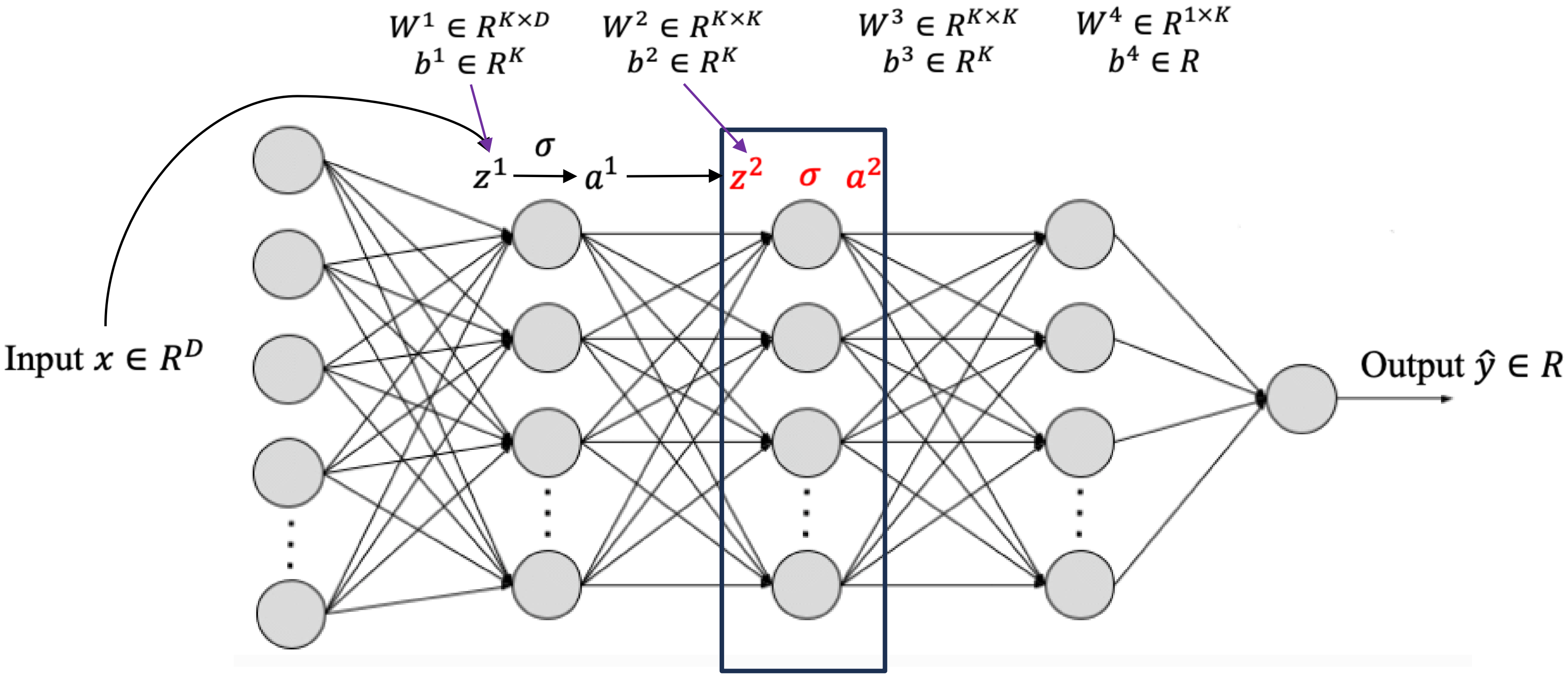


$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

Forward Propagation



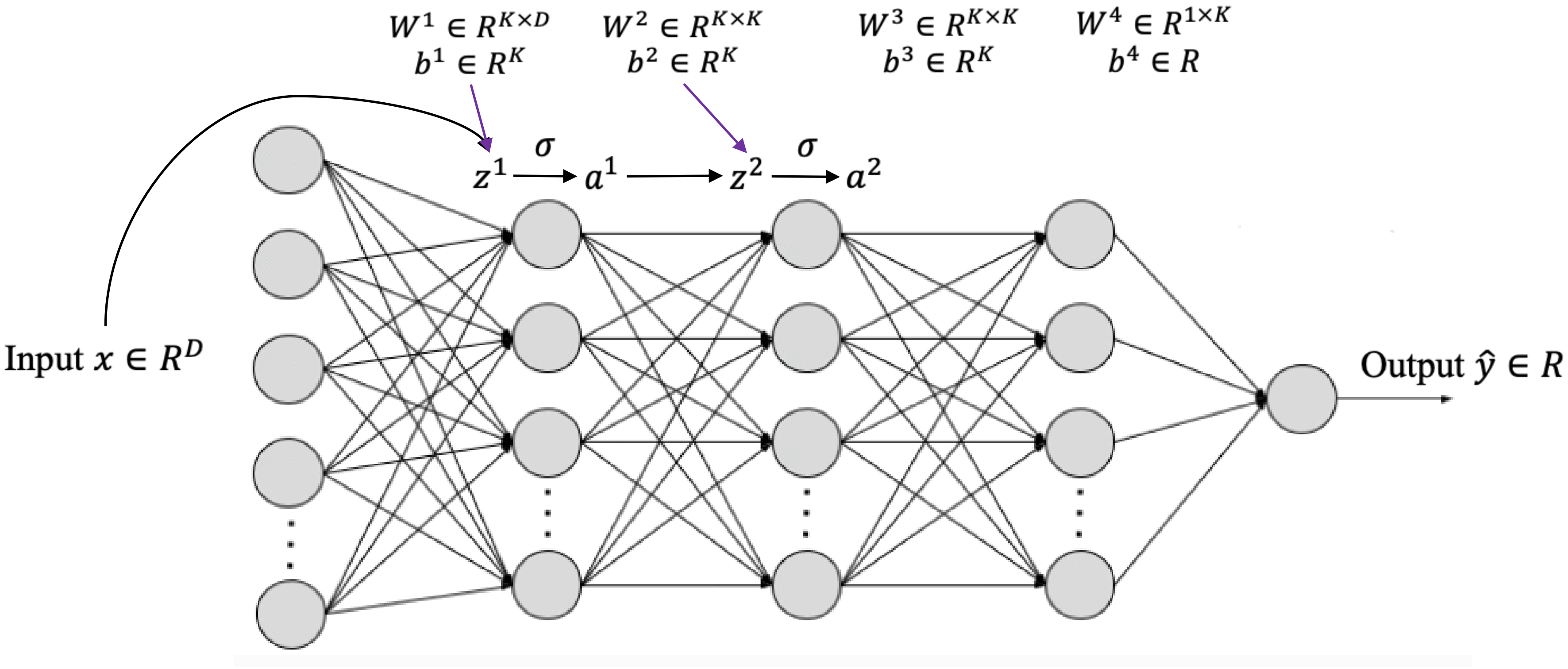
$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

Forward Propagation



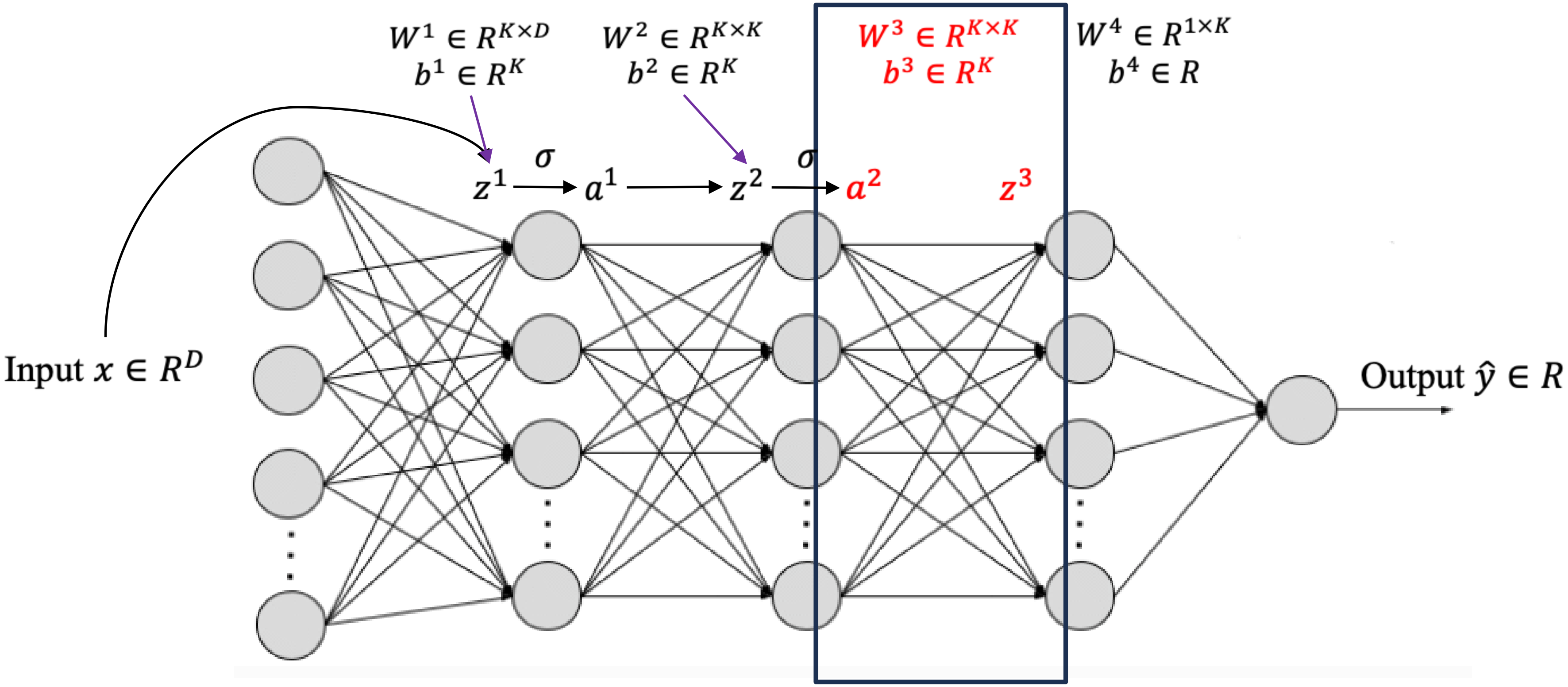
$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

Forward Propagation



$z^1 = W^1 x + b^1$

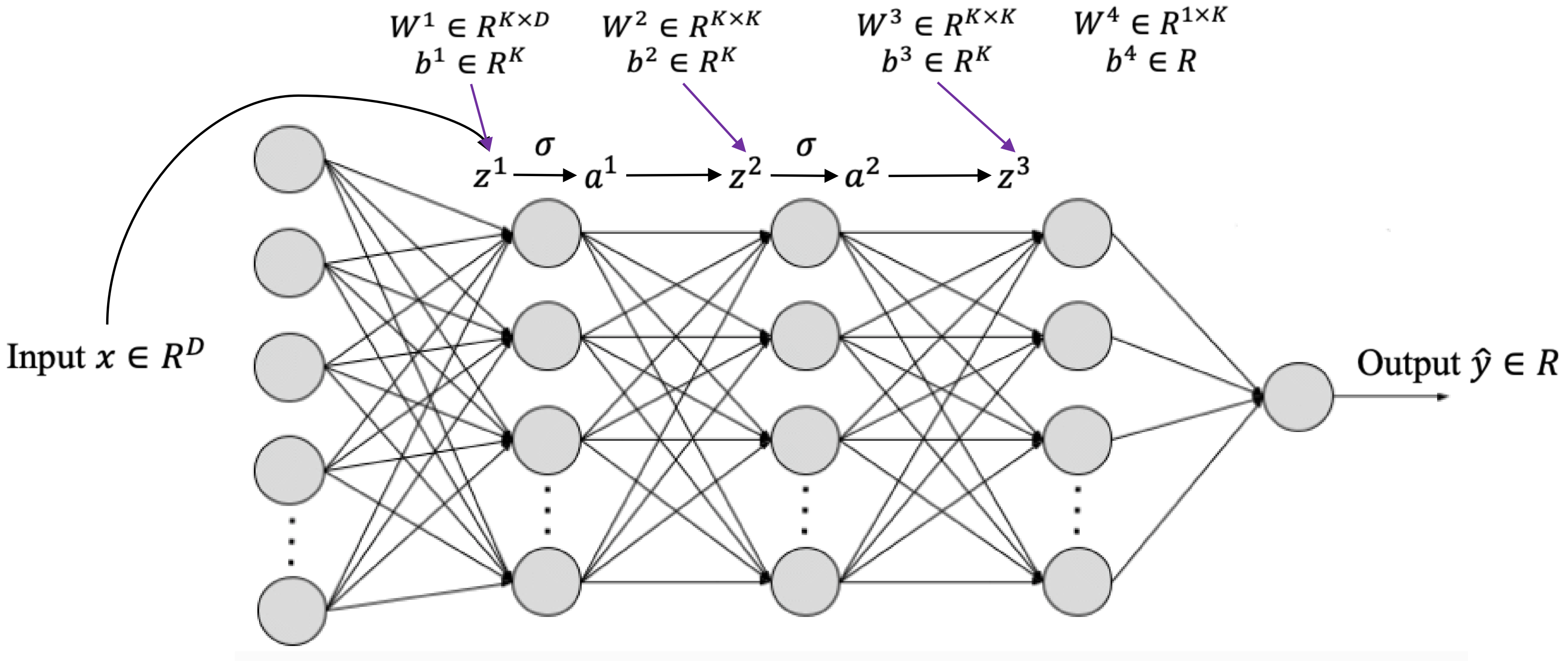
$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

$z^3 = W^3 a^2 + b^3$

Forward Propagation



$z^1 = W^1x + b^1$

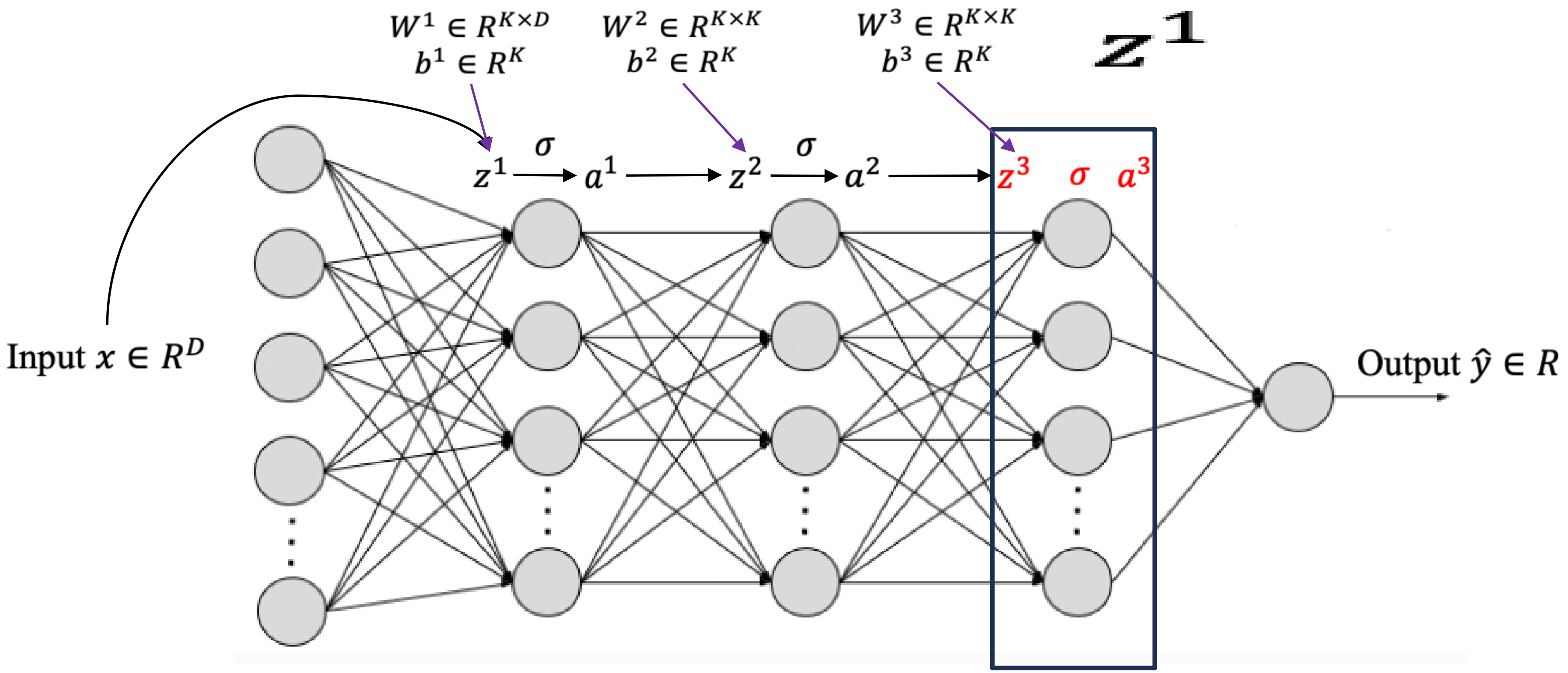
$a^1 = \sigma(z^1)$

$z^2 = W^2a^1 + b^2$

$a^2 = \sigma(z^2)$

$z^3 = W^3a^2 + b^3$

Forward Propagation



$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

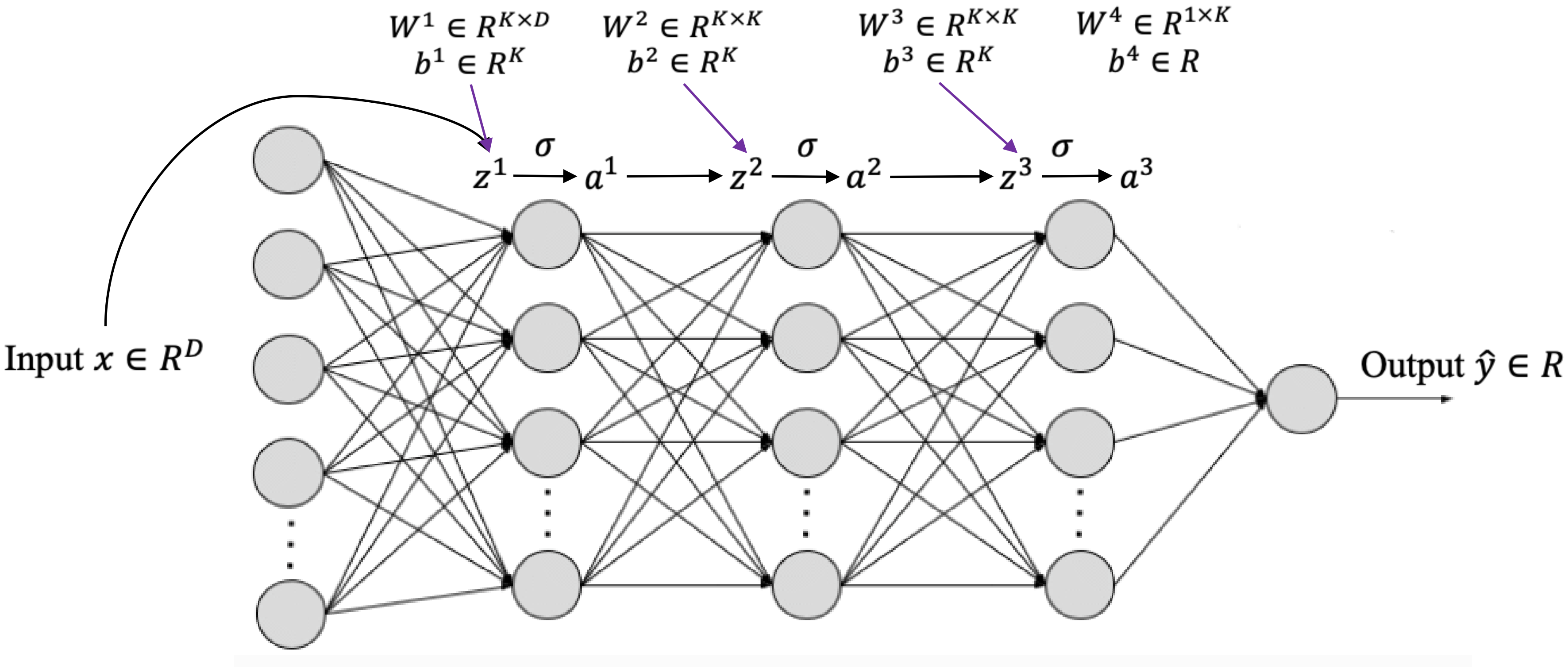
$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

$z^3 = W^3 a^2 + b^3$

$a^3 = \sigma(z^3)$

Forward Propagation



$z^1 = W^1x + b^1$

$a^1 = \sigma(z^1)$

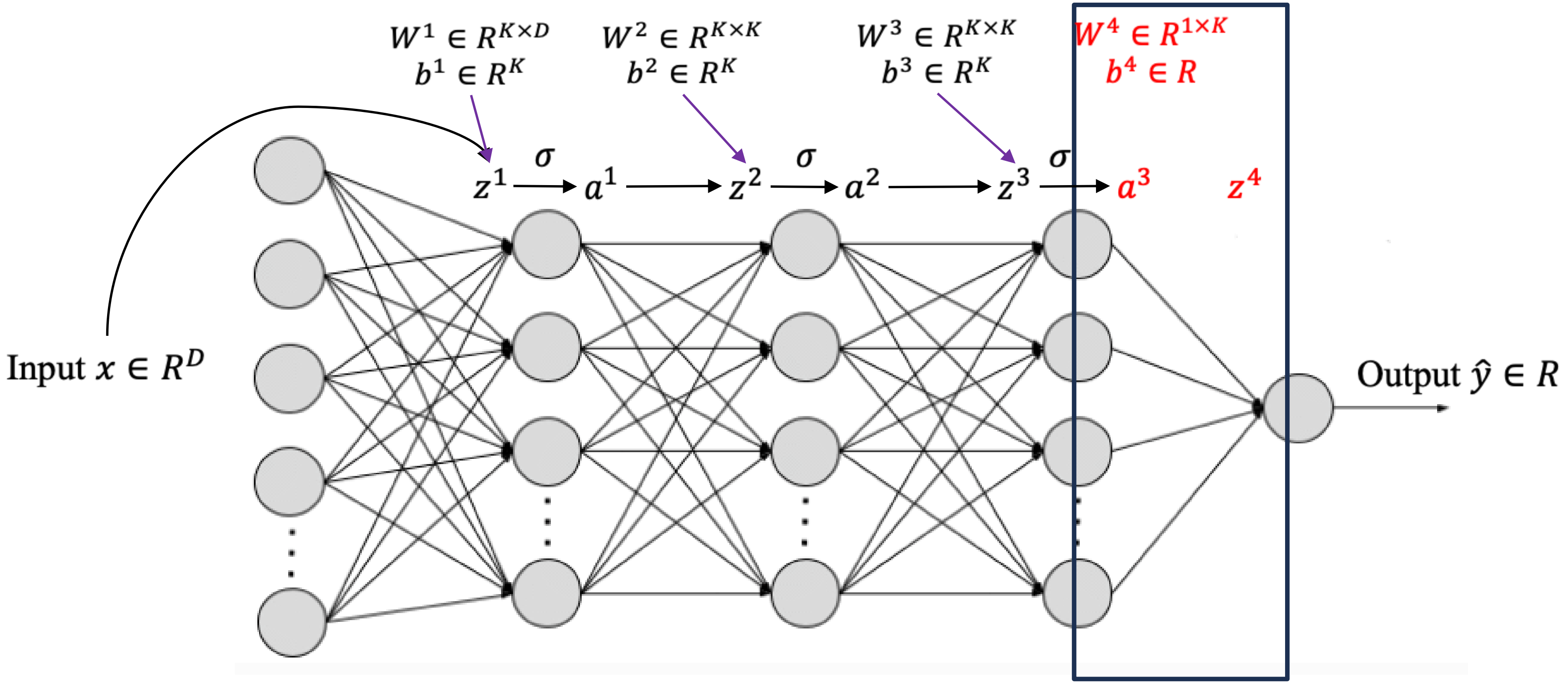
$z^2 = W^2a^1 + b^2$

$a^2 = \sigma(z^2)$

$z^3 = W^3a^2 + b^3$

$a^3 = \sigma(z^3)$

Forward Propagation



$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

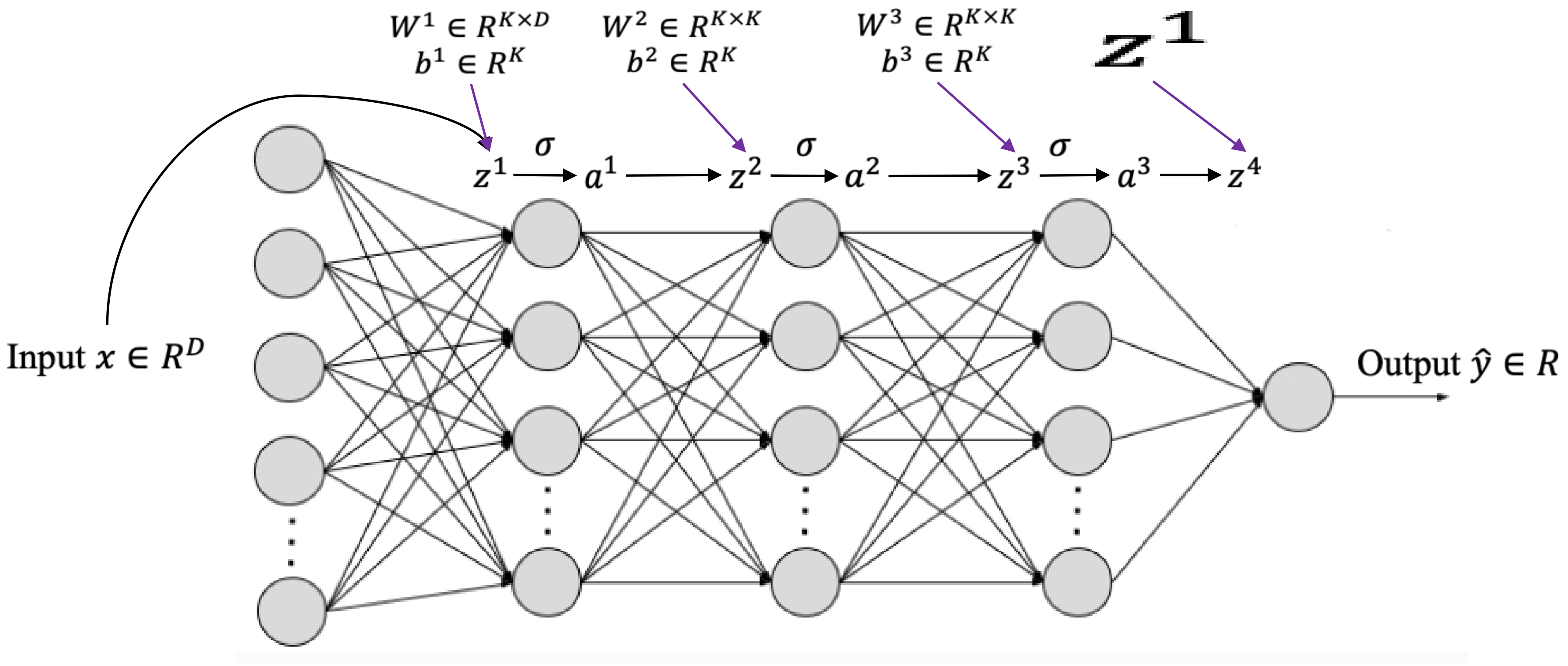
$a^2 = \sigma(z^2)$

$z^3 = W^3 a^2 + b^3$

$a^3 = \sigma(z^3)$

$z^4 = W^4 a^3 + b^4$

Forward Propagation



$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

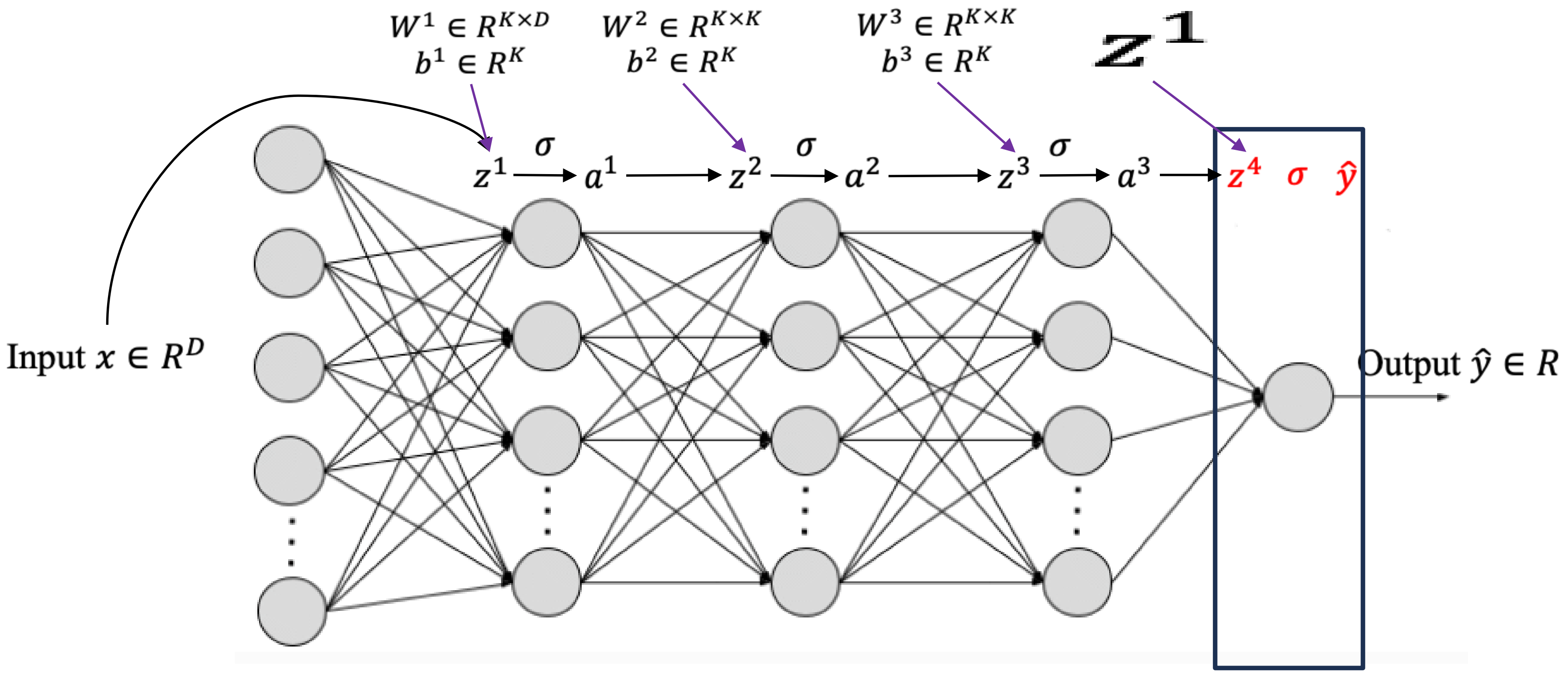
$a^2 = \sigma(z^2)$

$z^3 = W^3 a^2 + b^3$

$a^3 = \sigma(z^3)$

$z^4 = W^4 a^3 + b^4$

Forward Propagation



$z^1 = W^1x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2a^1 + b^2$

$a^2 = \sigma(z^2)$

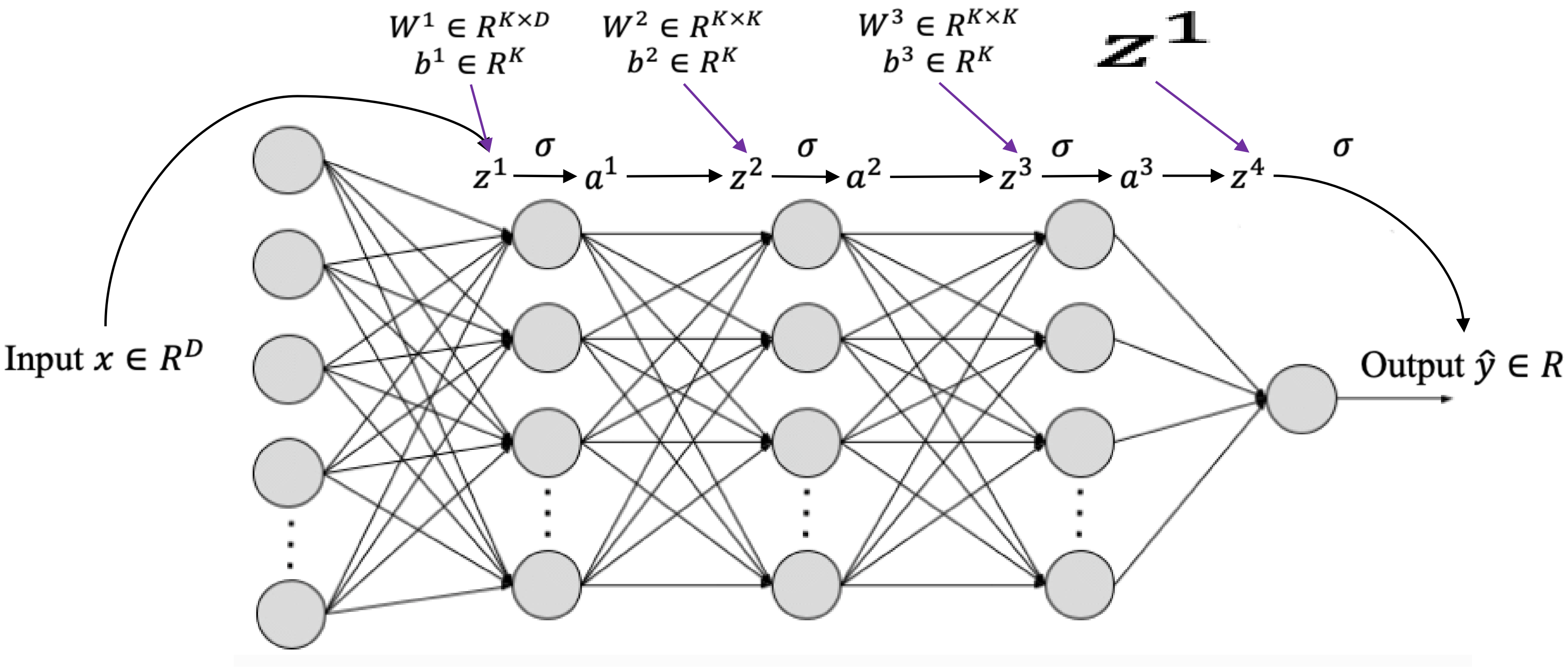
$z^3 = W^3a^2 + b^3$

$a^3 = \sigma(z^3)$

$z^4 = W^4a^3 + b^4$

$\hat{y} = \sigma(z^4)$

Forward Propagation



$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

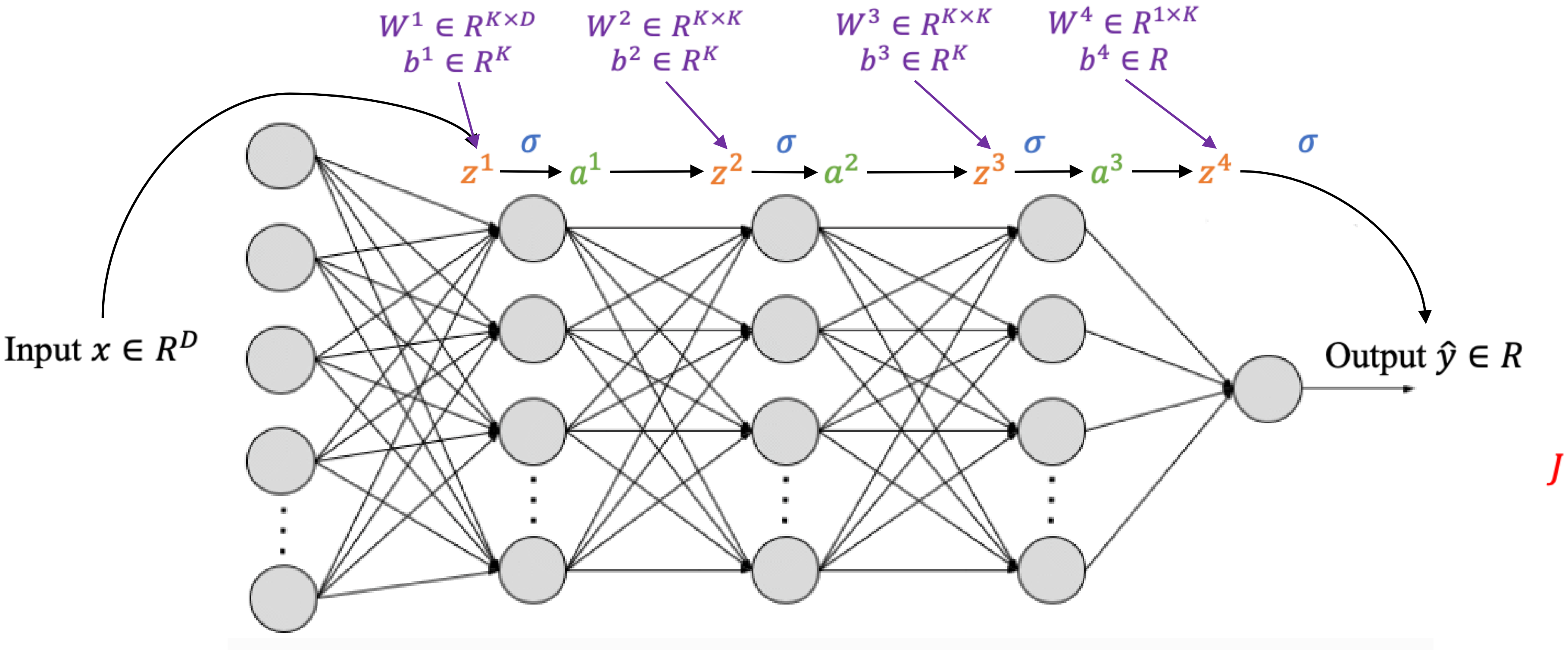
$z^3 = W^3 a^2 + b^3$

$a^3 = \sigma(z^3)$

$z^4 = W^4 a^3 + b^4$

$\hat{y} = \sigma(z^4)$

Forward Propagation



$$J = \frac{1}{2}(\hat{y} - y)^2$$

$z^1 = W^1 x + b^1$

$a^1 = \sigma(z^1)$

$z^2 = W^2 a^1 + b^2$

$a^2 = \sigma(z^2)$

$z^3 = W^3 a^2 + b^3$

$a^3 = \sigma(z^3)$

$z^4 = W^4 a^3 + b^4$

$\hat{y} = \sigma(z^4)$

$a^l = \sigma(z^l)$

$z^{l+1} = W^{l+1} a^l + b^{l+1}$

Parameter Update – Gradient Descent

Gradient Descent

current parameter

$$\boxed{W} \leftarrow \boxed{W} - \boxed{\alpha}\boxed{\frac{\partial J}{\partial \mathrm{W}}}$$

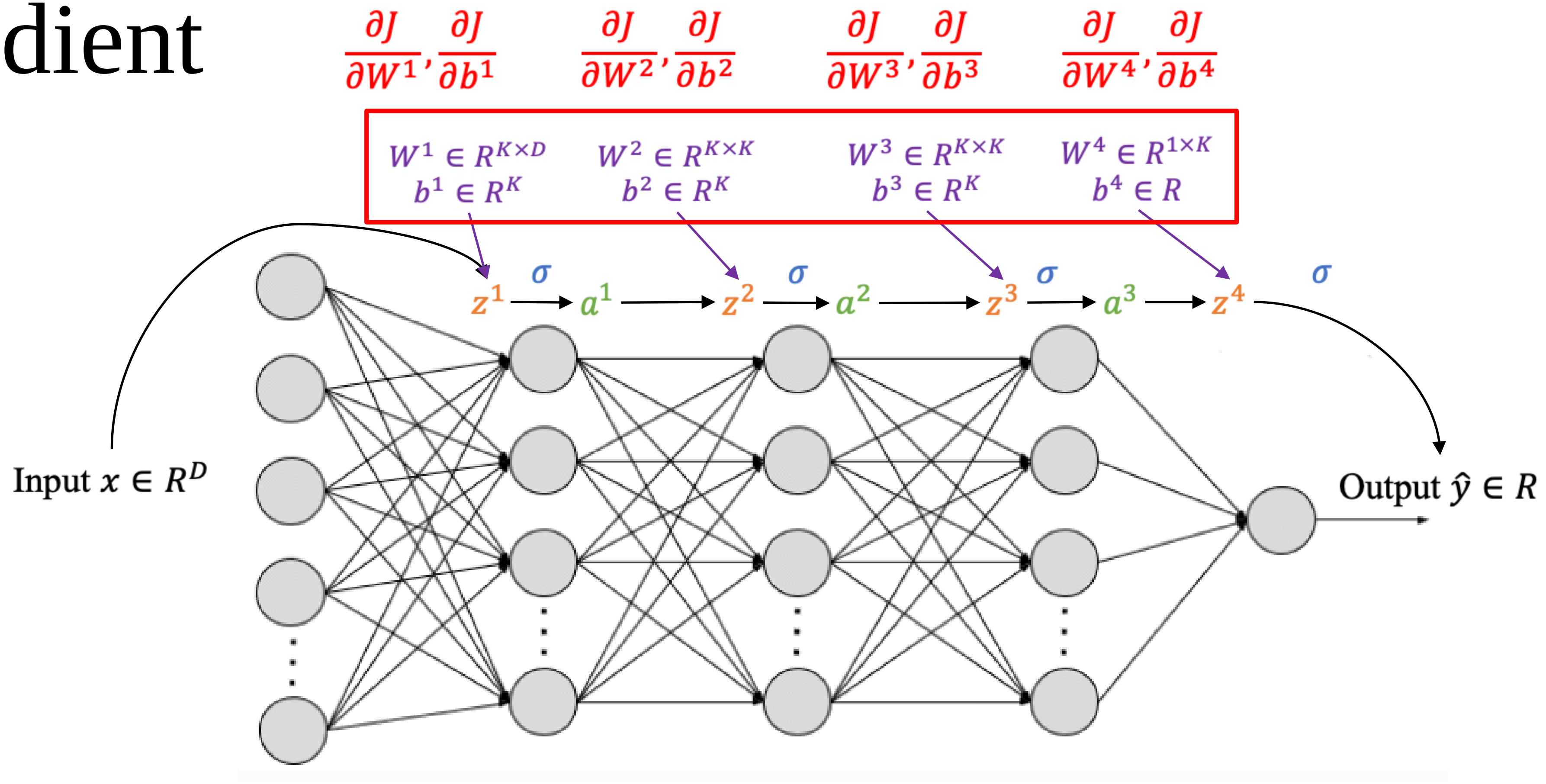
gradient

new parameter

learning rate

$$b \leftarrow b - \alpha\boxed{\frac{\partial J}{\partial \mathrm{b}}}$$

Gradient



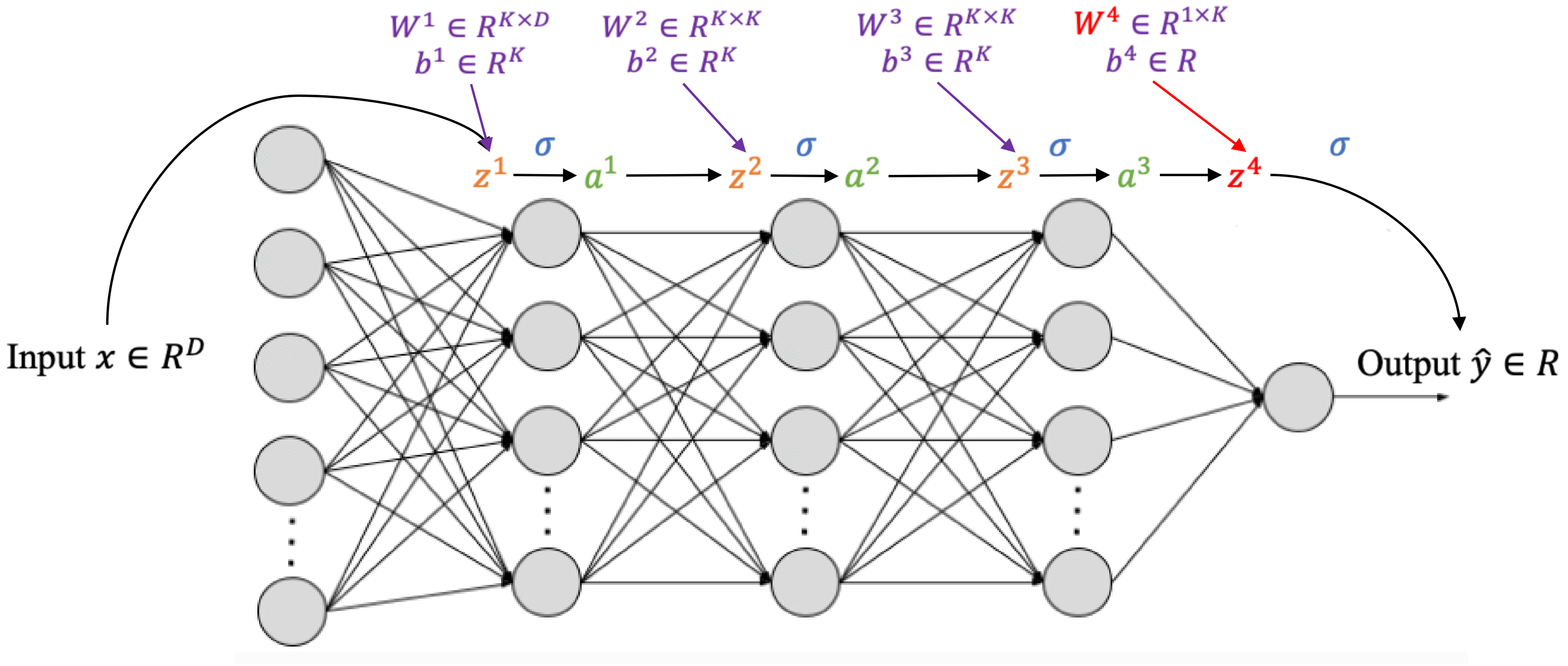
$$J = \frac{1}{2}(\hat{y} - y)^2$$

How? Backward propagation with chain rule!

Backward Propagation

Chain rule

$$\frac{\partial J}{\partial W^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial W^4}$$

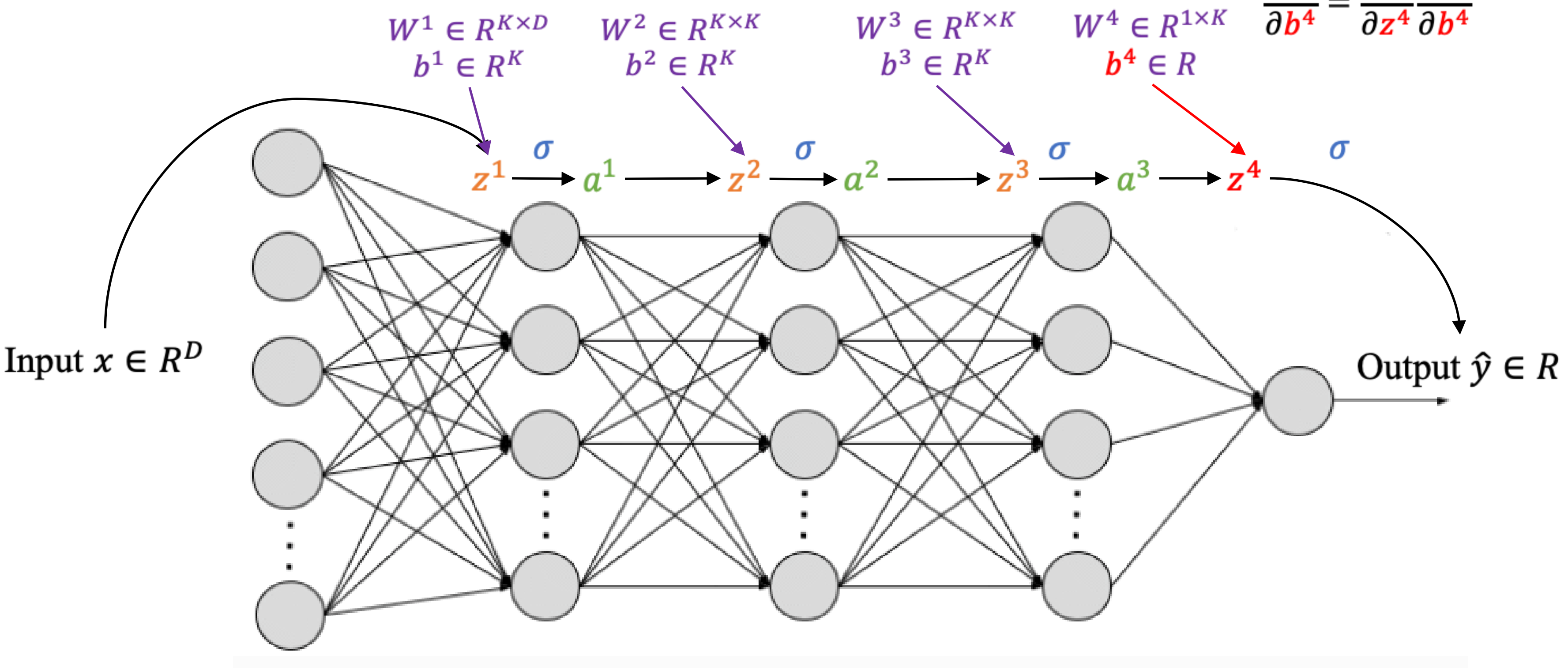


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial W^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial W^4}$$

$$\frac{\partial J}{\partial b^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial b^4}$$



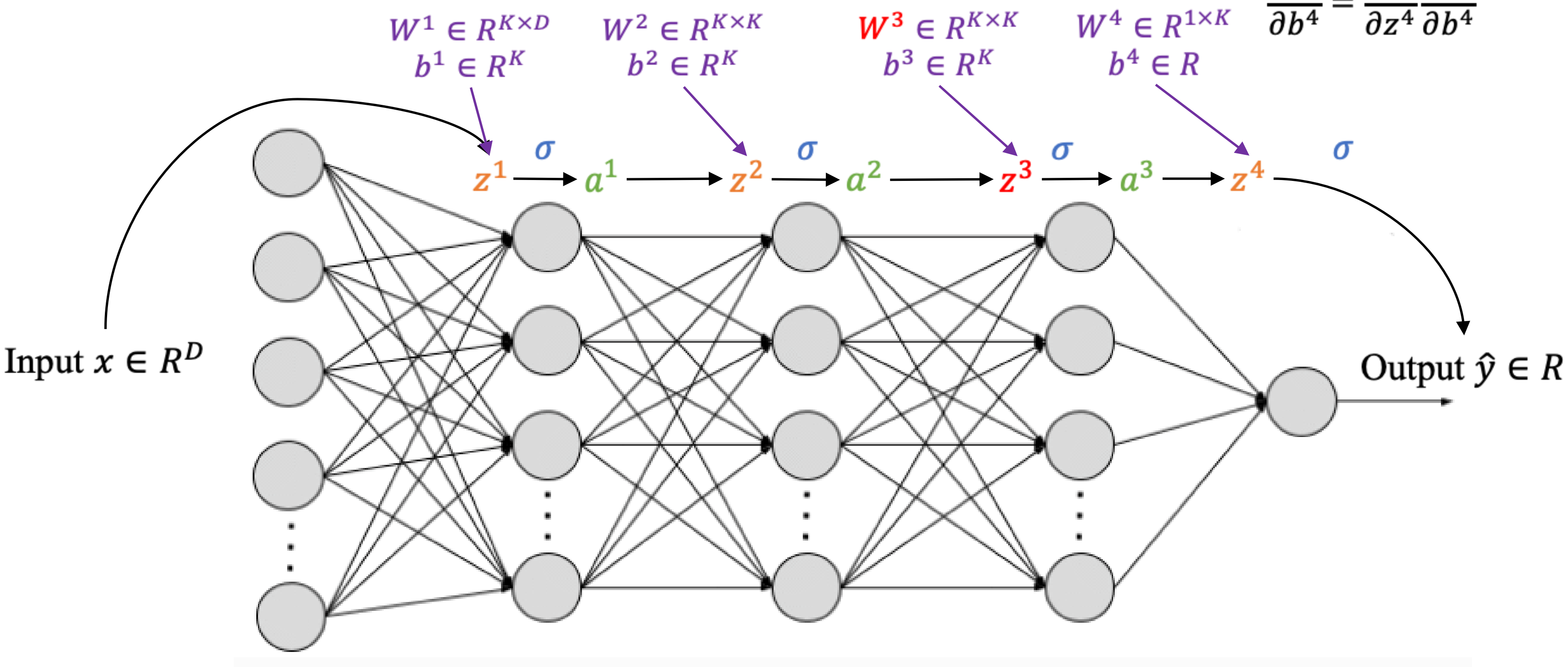
$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial W^3} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial W^3}$$

$$\frac{\partial J}{\partial W^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial W^4}$$

$$\frac{\partial J}{\partial b^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial b^4}$$

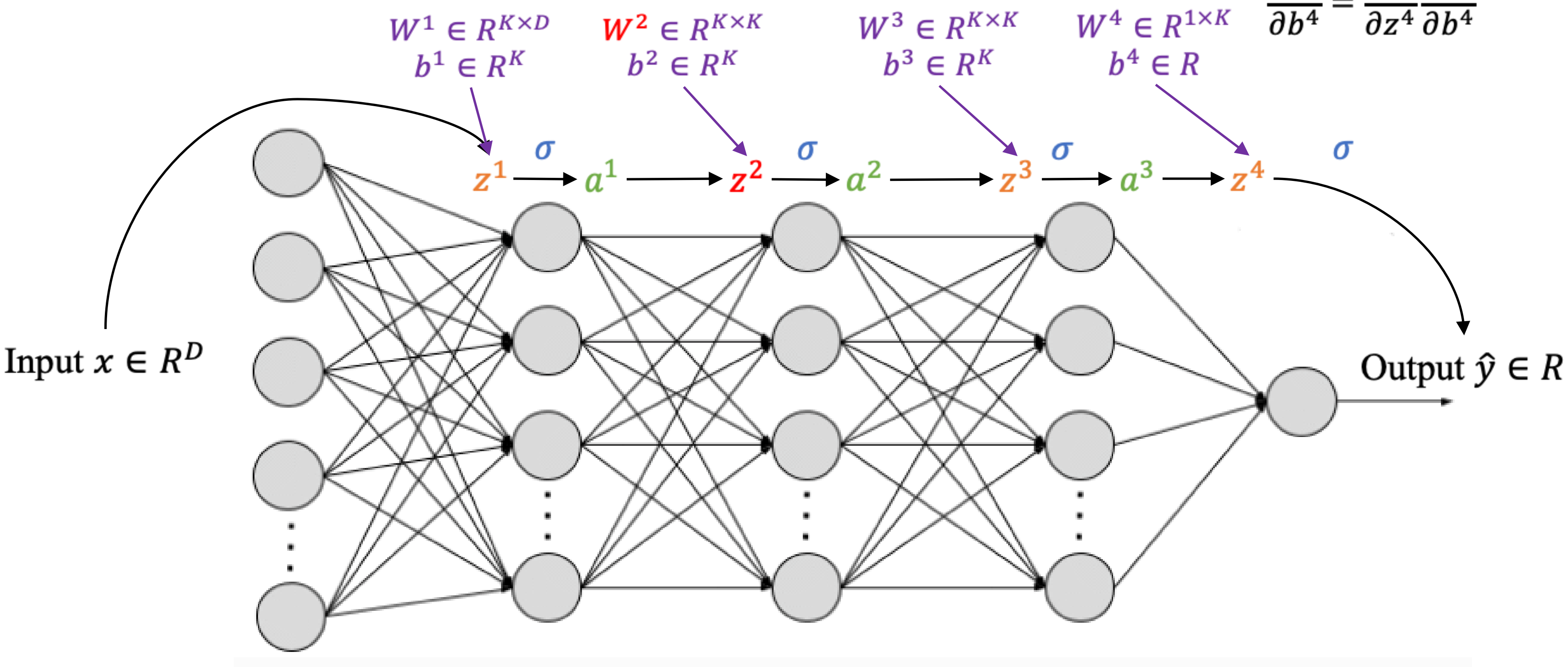


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial W^2} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial W^2} \quad \frac{\partial J}{\partial W^3} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial W^3} \quad \frac{\partial J}{\partial W^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial W^4}$$

$$\frac{\partial J}{\partial b^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial b^4}$$

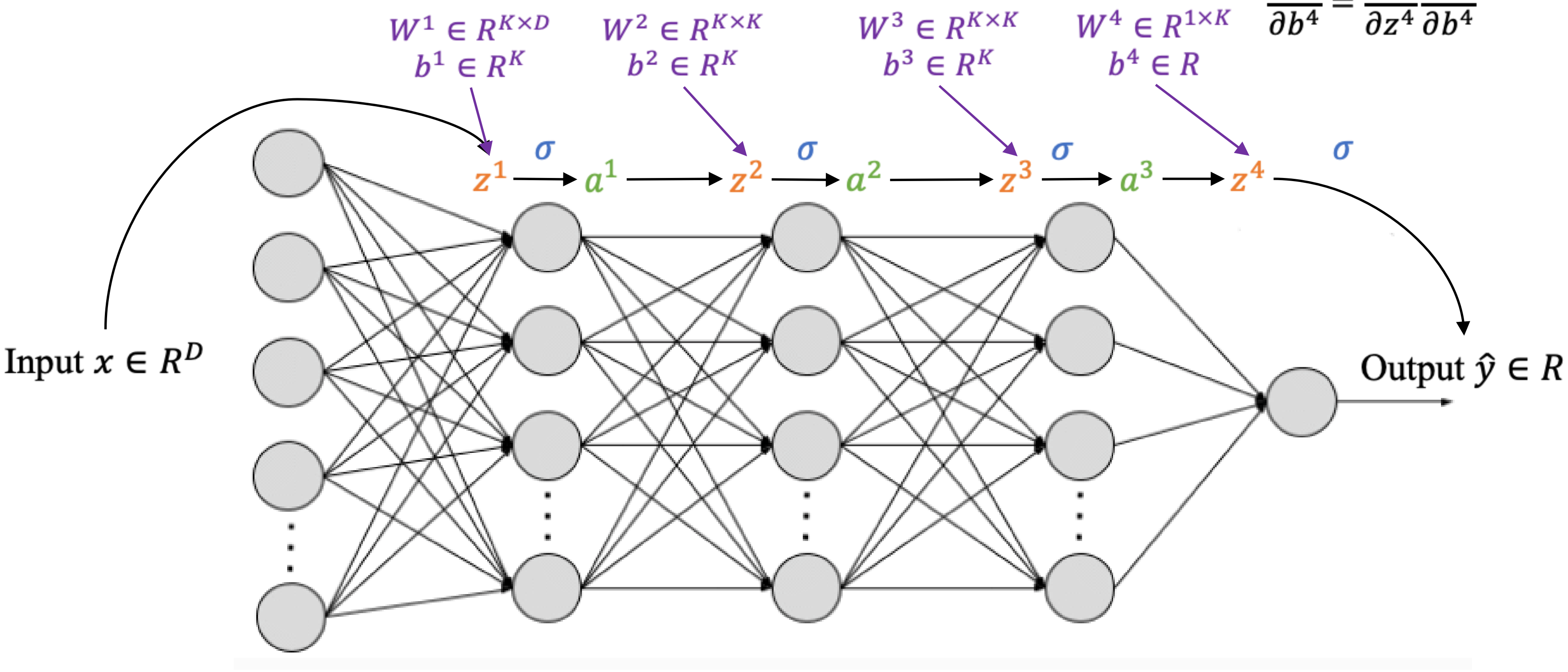


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial W^1} = \frac{\partial J}{\partial z^1}\frac{\partial z^1}{\partial W^1} \quad \frac{\partial J}{\partial W^2} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial W^2} \quad \frac{\partial J}{\partial W^3} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial W^3} \quad \frac{\partial J}{\partial W^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial W^4}$$

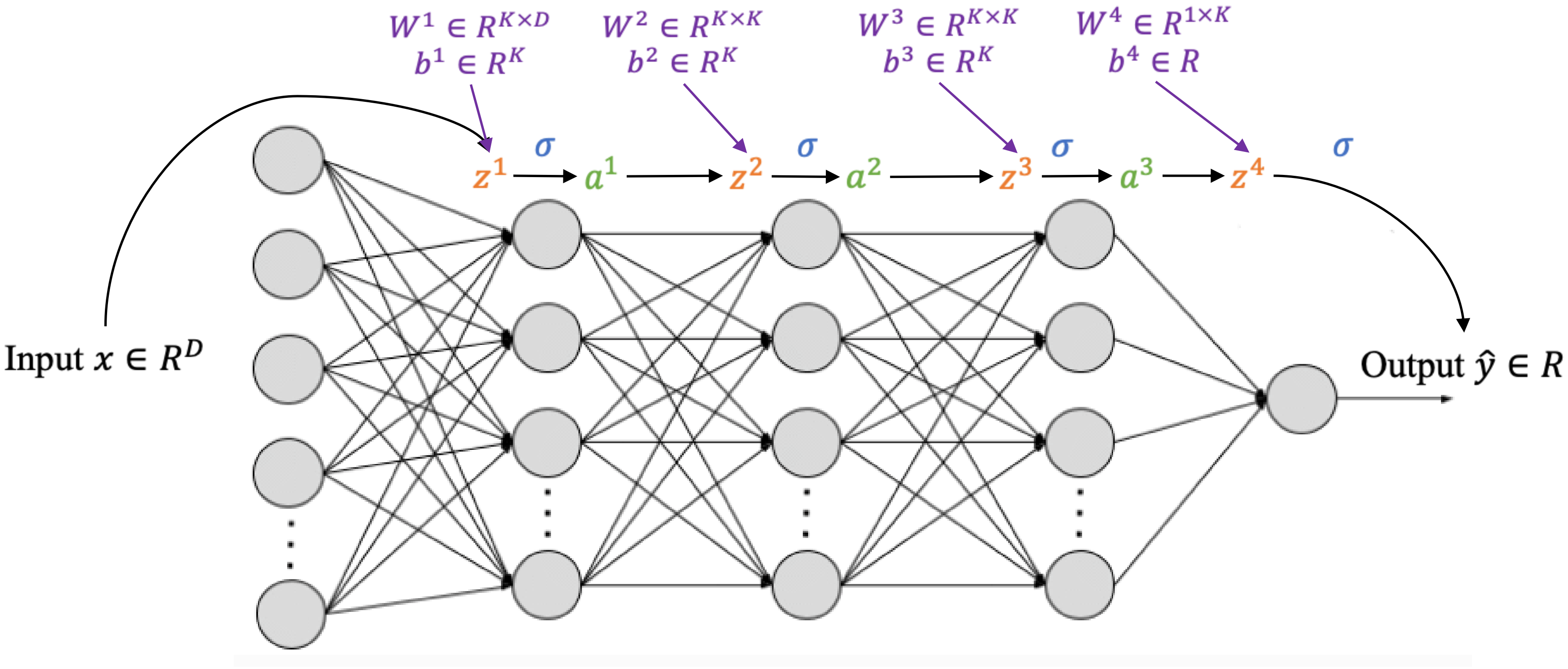
$$\frac{\partial J}{\partial b^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial b^4}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial W^l} = \frac{\partial J}{\partial z^l}\frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \frac{\partial J}{\partial z^l}\frac{\partial z^l}{\partial b^l}$$



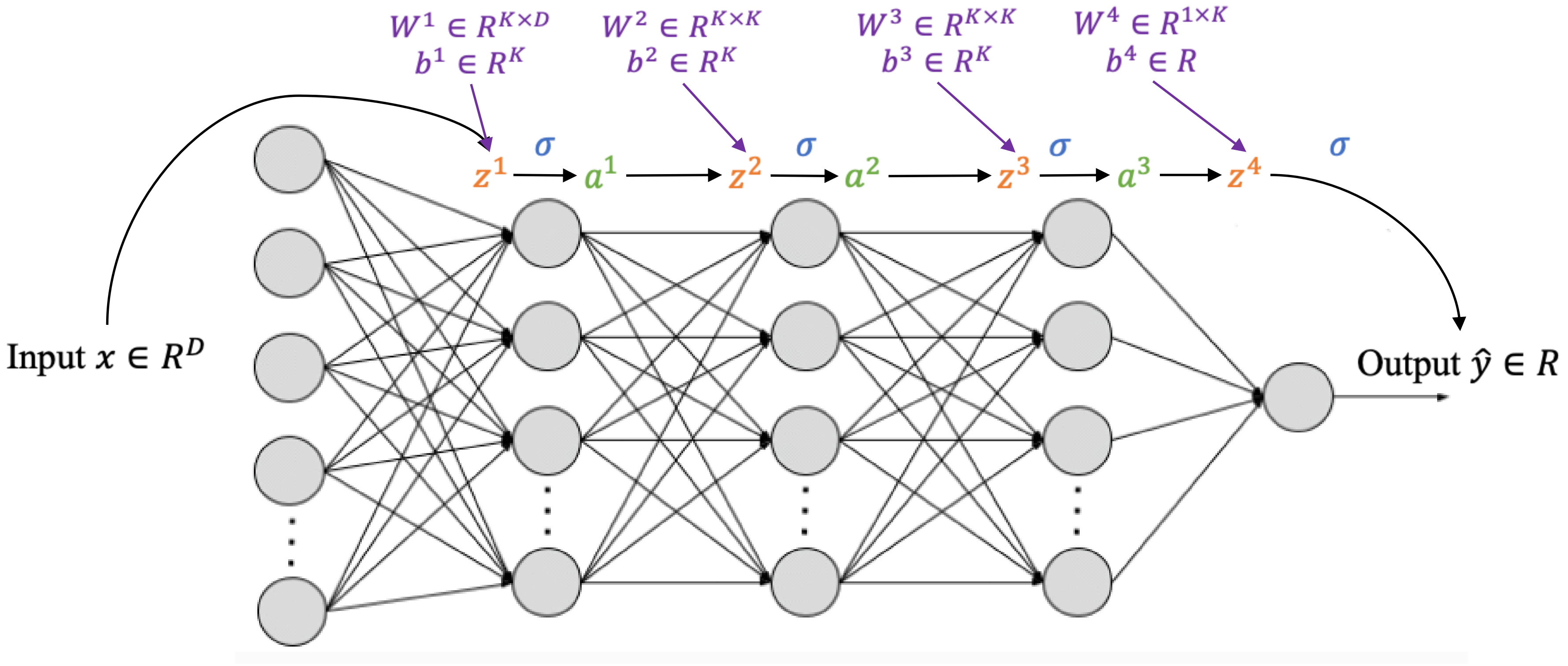
$$J = \frac{1}{2}(\hat{y} - y)^2$$

forward propagation:

$$z^l = W^l a^{l-1} + b^l \qquad \frac{\partial z^l}{\partial W^l} = a^{l-1} \qquad \frac{\partial z^l}{\partial b^l} = \mathbf{1}$$

Chain rule

$$\frac{\partial J}{\partial W^l} = \frac{\partial J}{\partial z^l}\frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \frac{\partial J}{\partial z^l}\frac{\partial z^l}{\partial b^l}$$



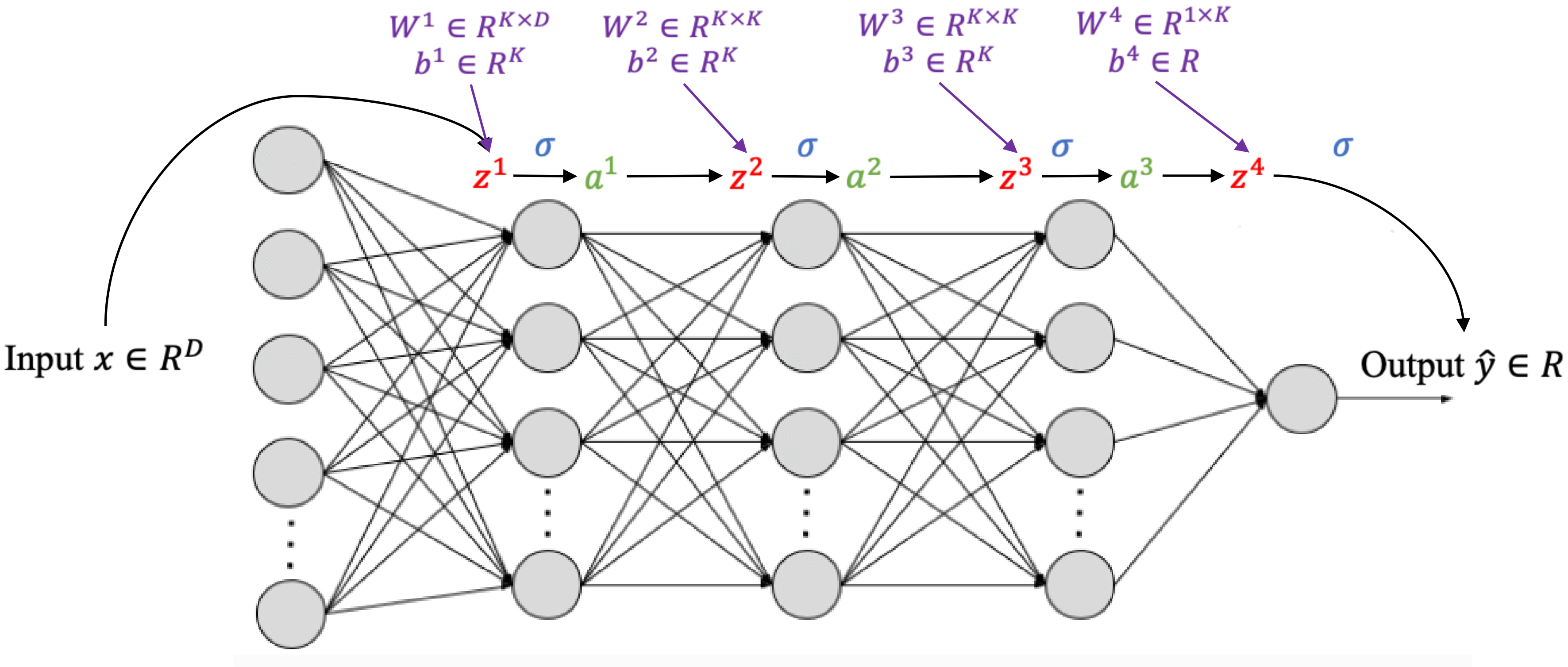
$$J = \frac{1}{2}(\hat{y} - y)^2$$

forward propagation:

$$z^l = W^l a^{l-1} + b^l \qquad \frac{\partial z^l}{\partial W^l} = a^{l-1} \qquad \frac{\partial z^l}{\partial b^l} = \mathbf{1}$$

Chain rule

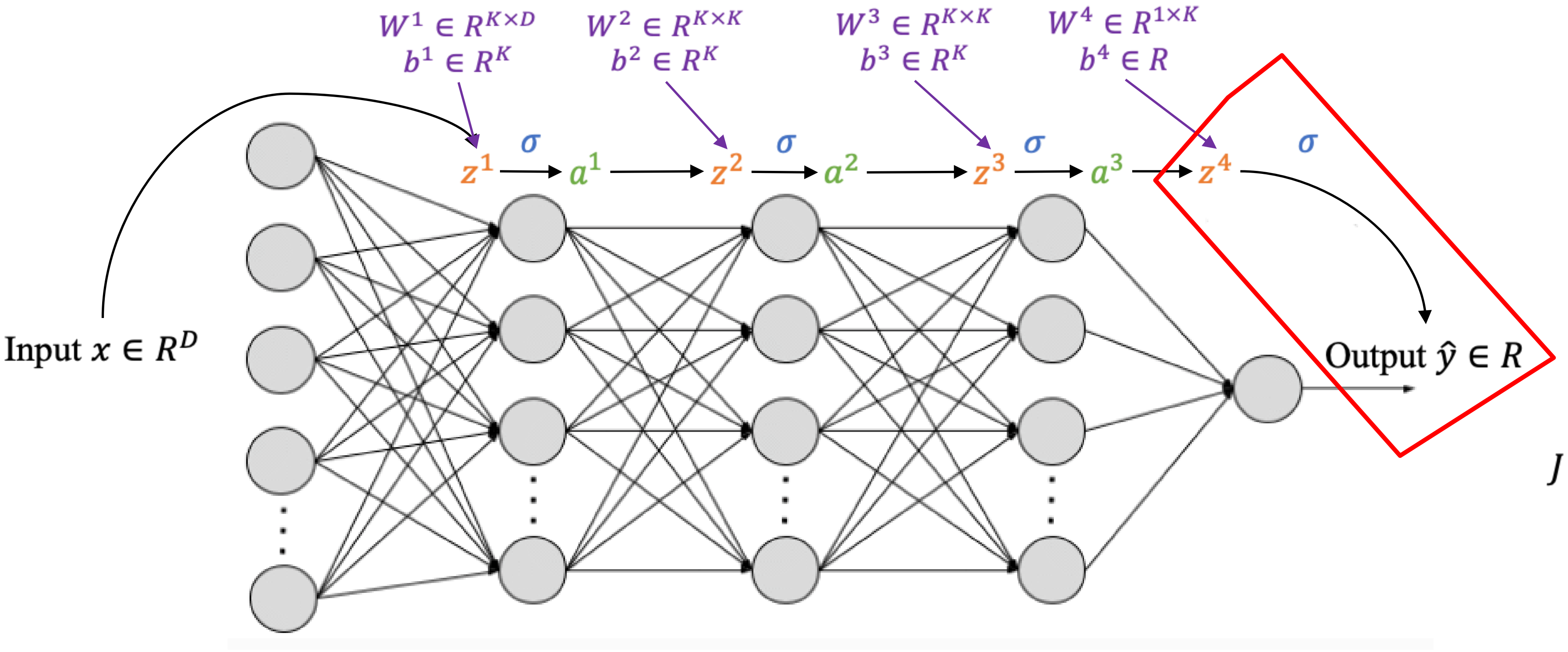
$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

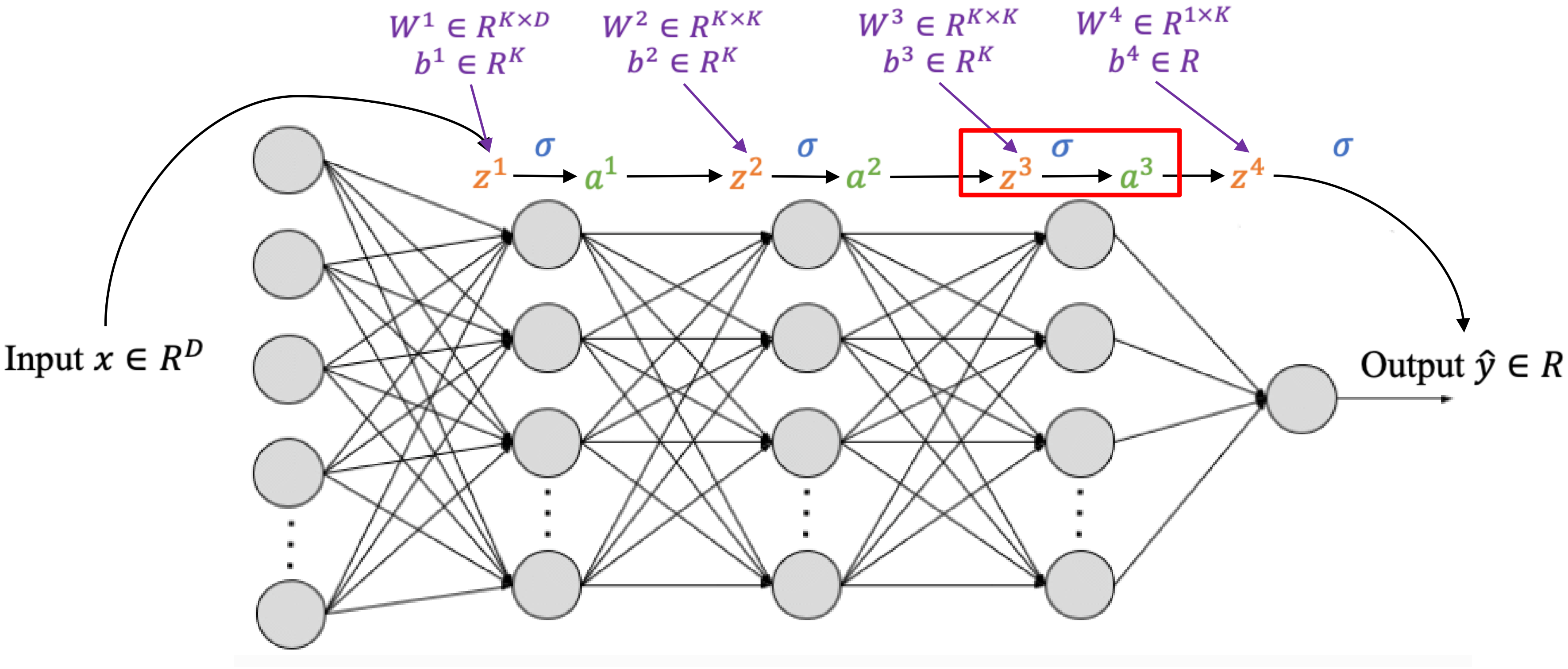
$$\frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^4}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

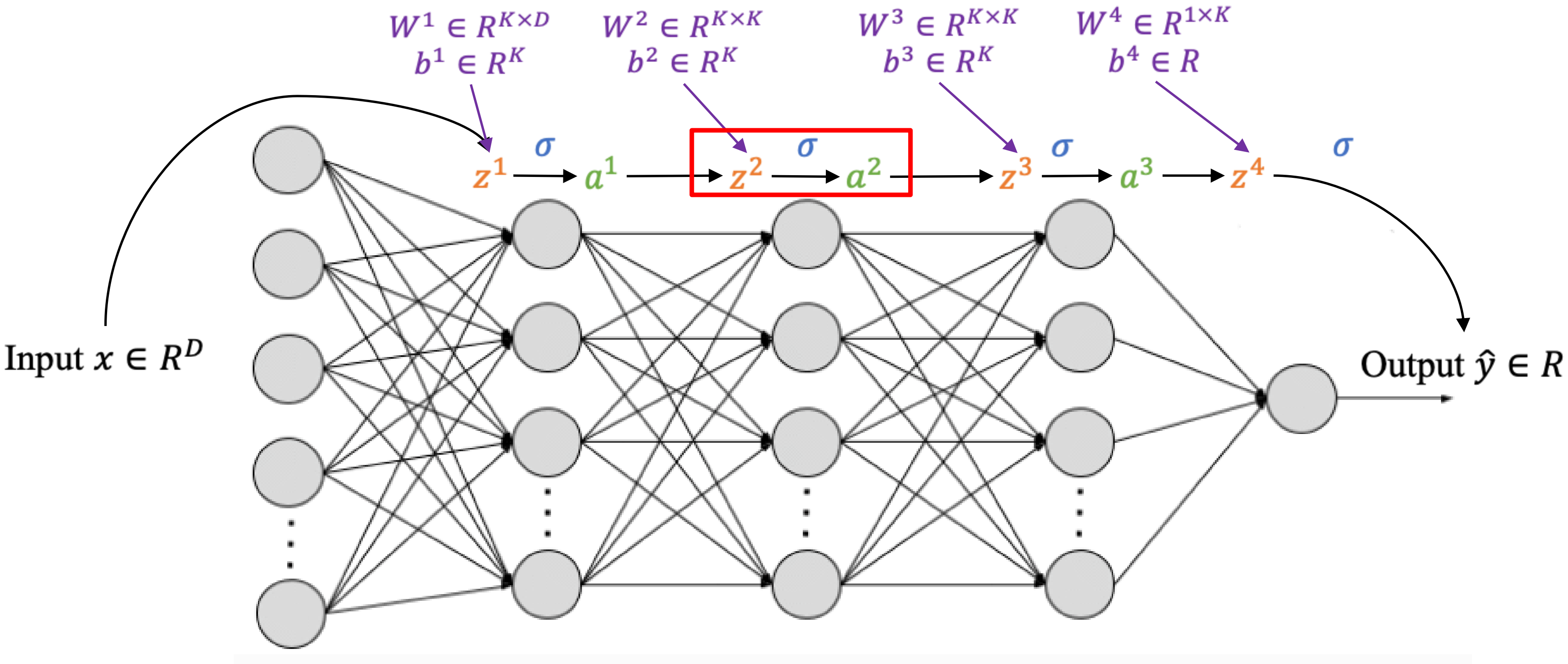
$$\frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial z^3} \qquad \frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^4}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

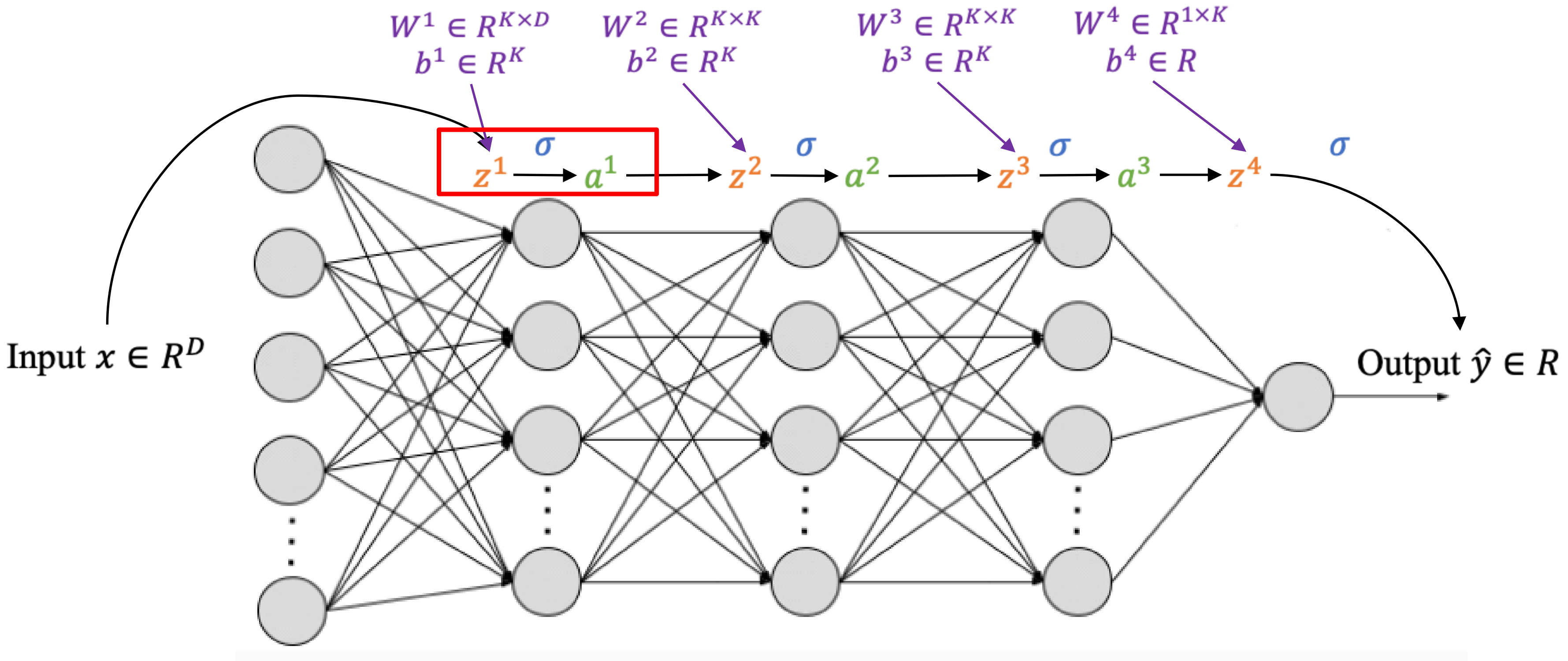
$$\frac{\partial J}{\partial z^2} = \frac{\partial J}{\partial a^2}\frac{\partial a^2}{\partial z^2} \qquad \frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial z^3} \qquad \frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^4}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

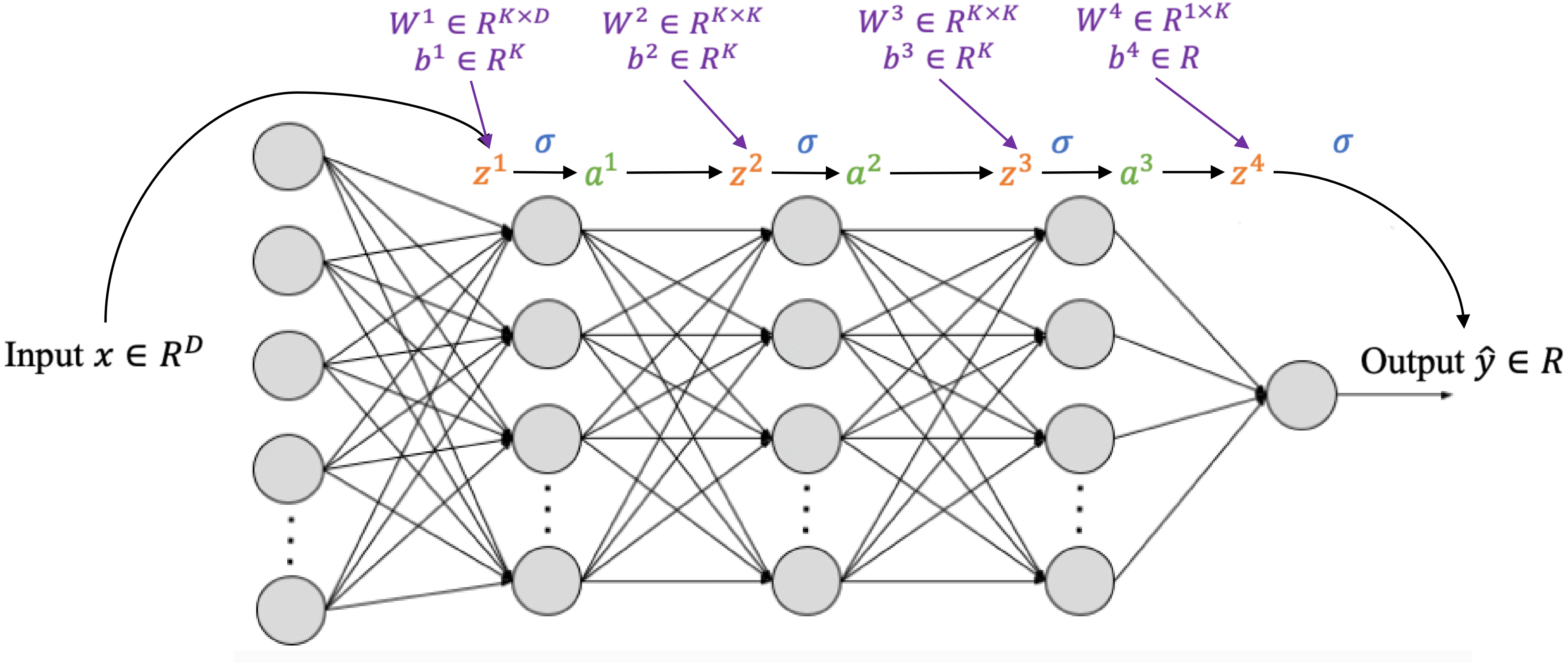
$$\frac{\partial J}{\partial z^1} = \frac{\partial J}{\partial a^1}\frac{\partial a^1}{\partial z^1} \quad \frac{\partial J}{\partial z^2} = \frac{\partial J}{\partial a^2}\frac{\partial a^2}{\partial z^2} \quad \frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial z^3} \quad \frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^4}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial z^l} = \frac{\partial J}{\partial a^l}\frac{\partial a^l}{\partial z^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

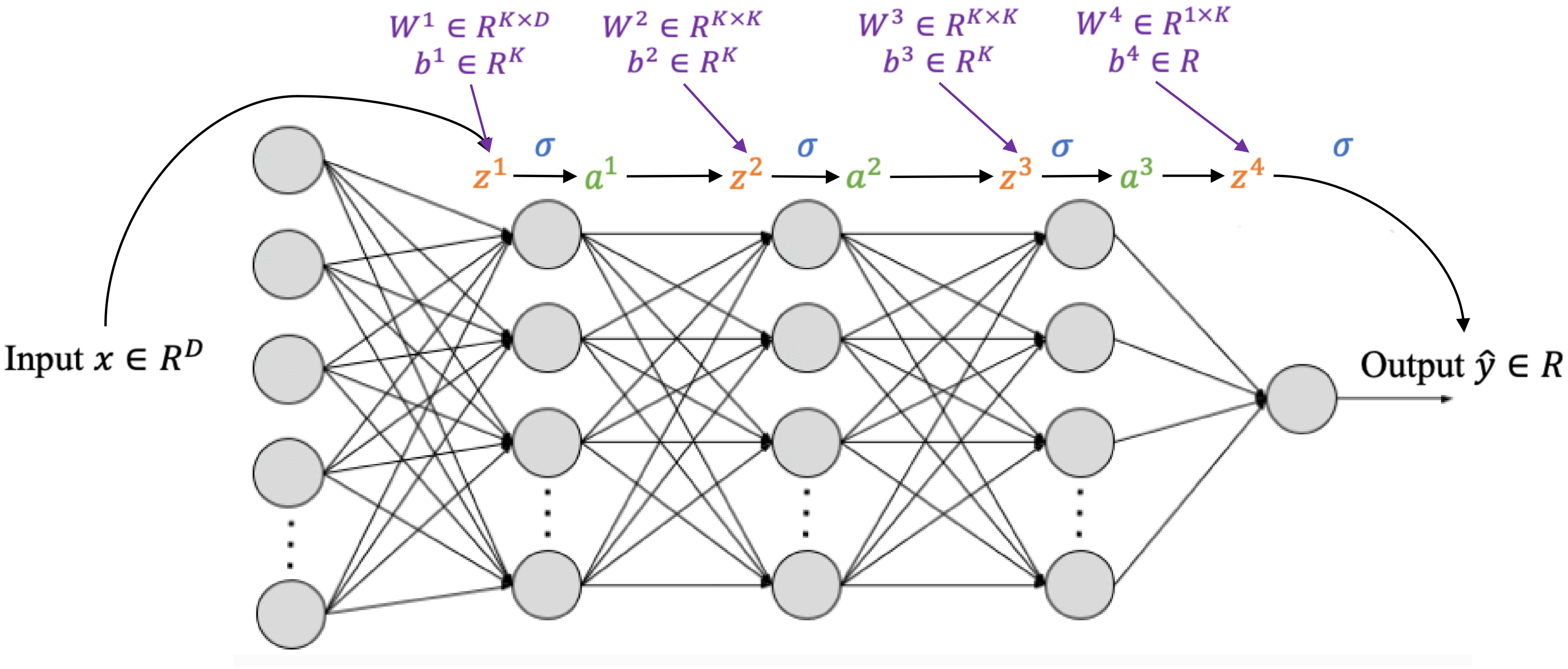
forward propagation:

$$a^l = \sigma(z^l)$$

$$\frac{\partial a^l}{\partial z^l} = a^l(1 - a^l)$$

Chain rule

$$\frac{\partial J}{\partial z^l} = \frac{\partial J}{\partial a^l}\frac{\partial a^l}{\partial z^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

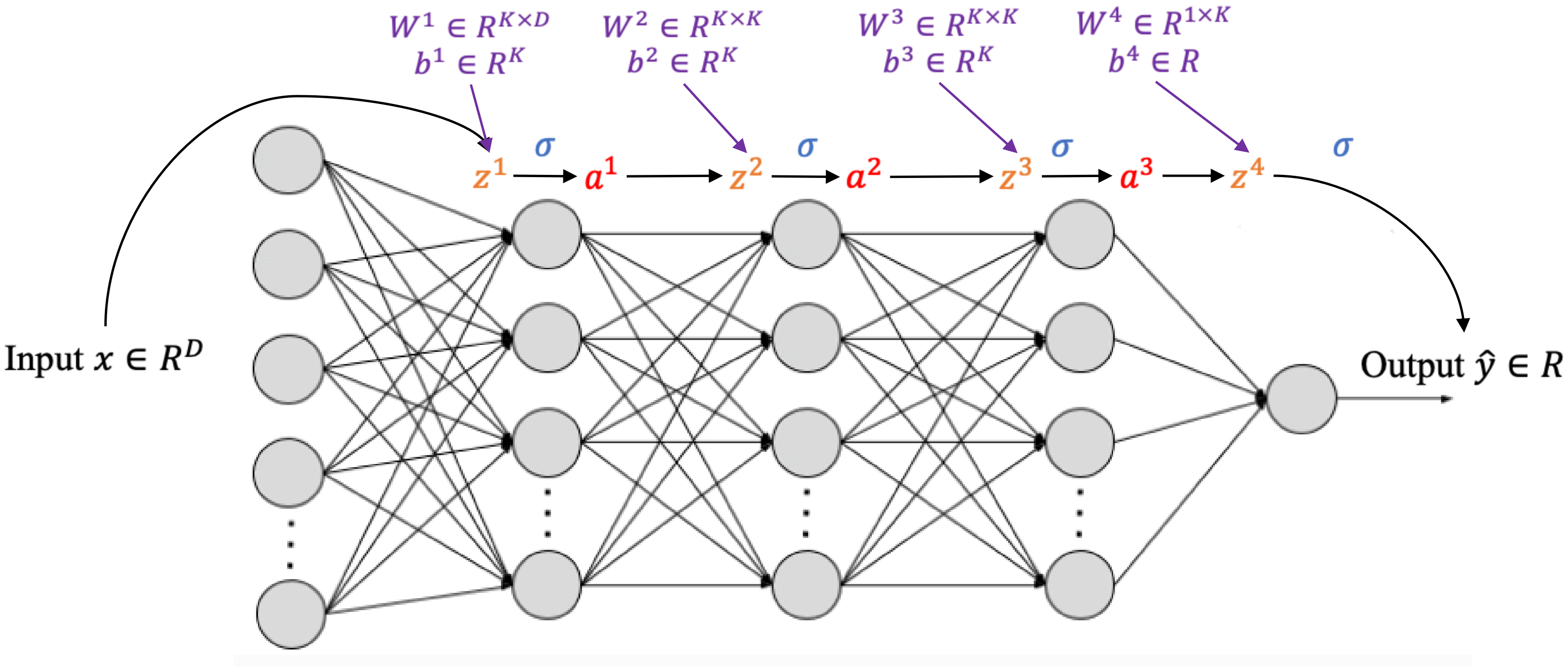
forward propagation:

$$a^l = \sigma(z^l)$$

$$\frac{\partial a^l}{\partial z^l} = a^l(1 - a^l)$$

Chain rule

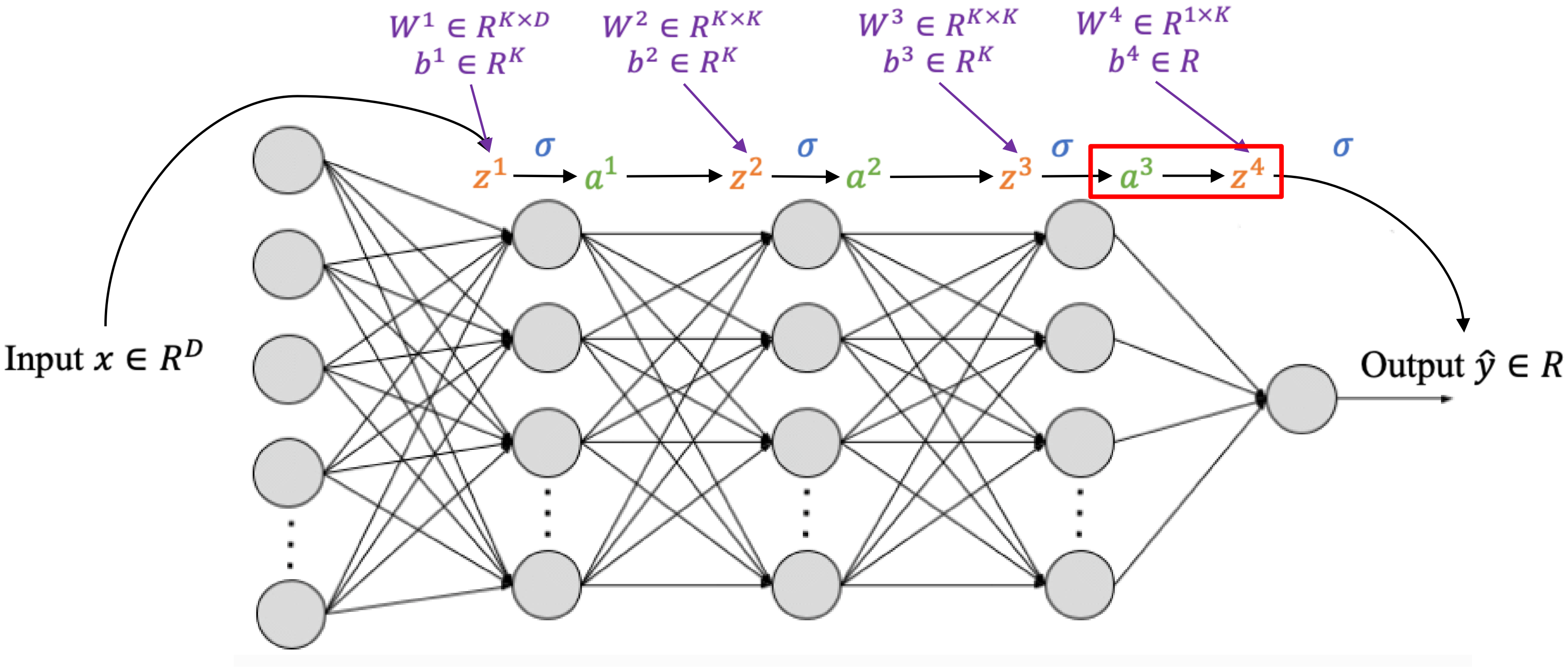
$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

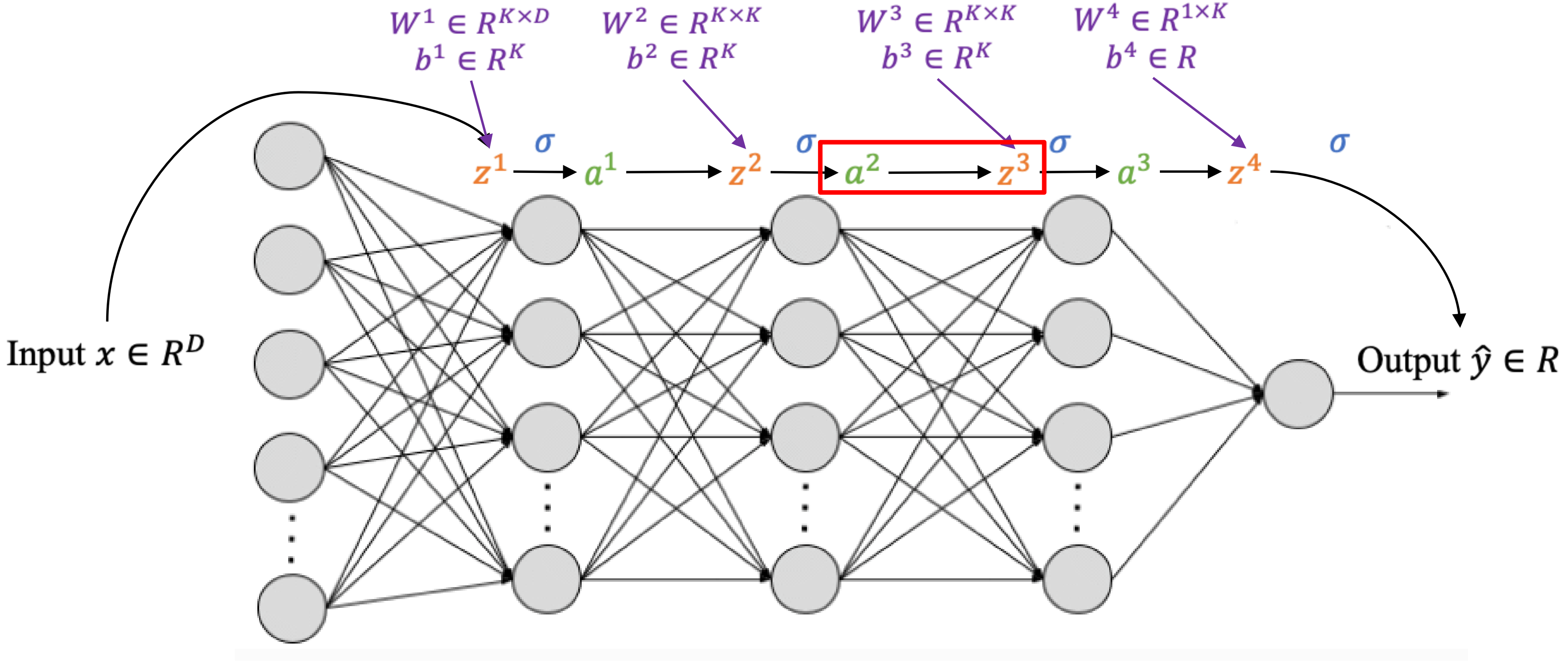
$$\frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial a^3}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

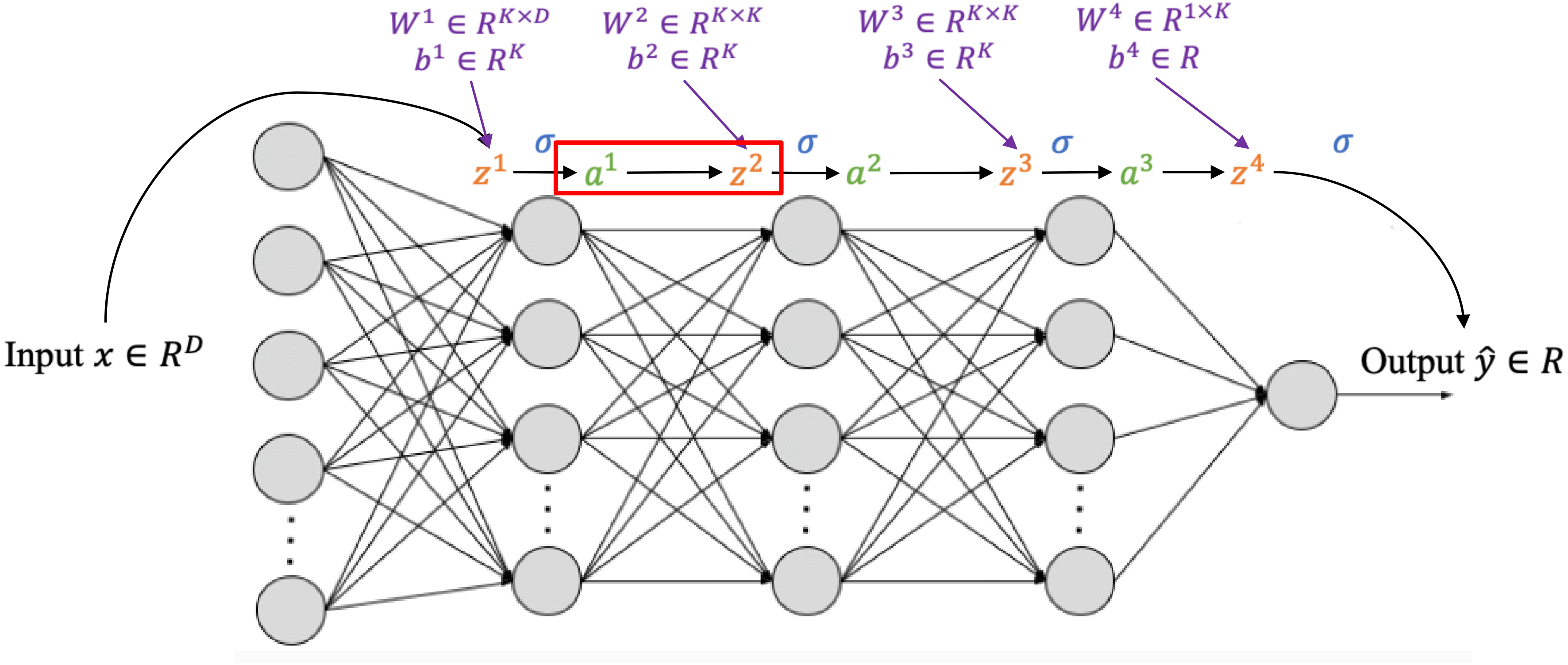
$$\frac{\partial J}{\partial a^2} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial a^2} \qquad \frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial a^3}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

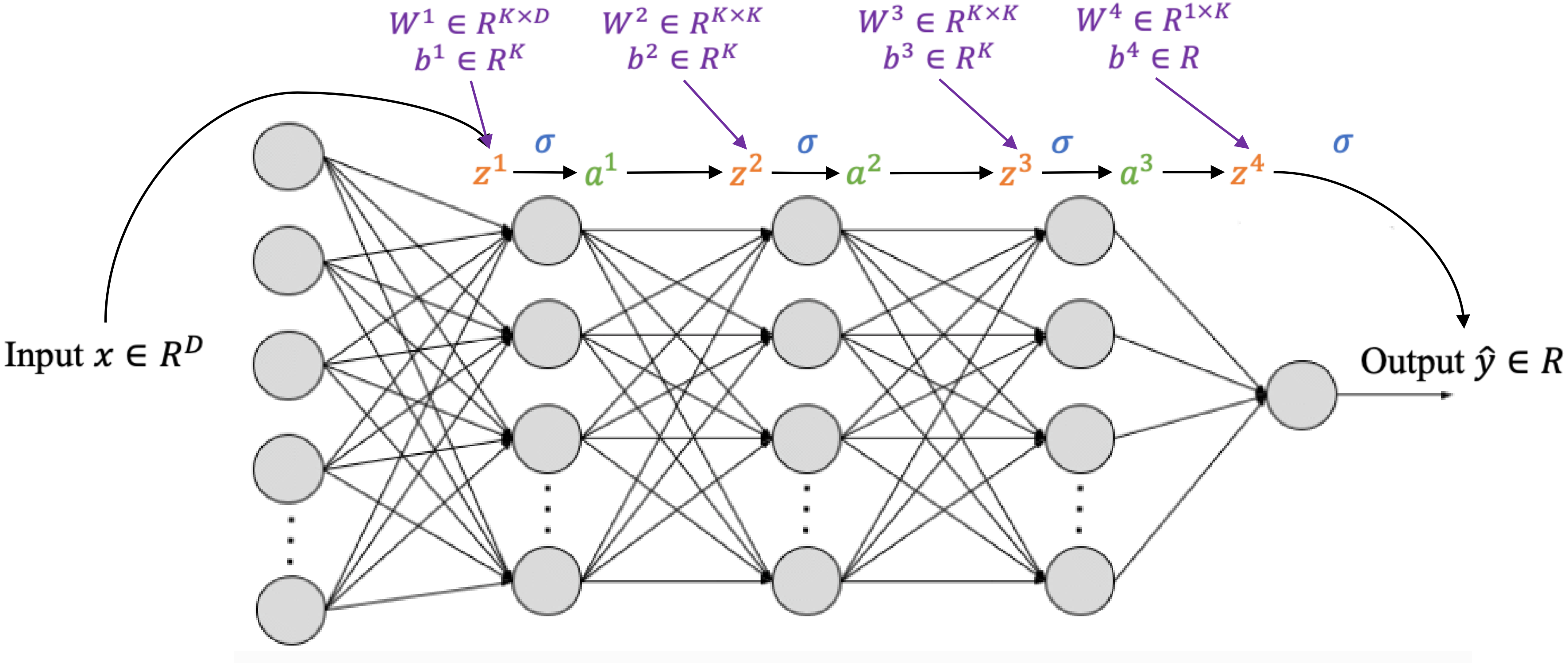
$$\frac{\partial J}{\partial a^1} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial a^1} \qquad \frac{\partial J}{\partial a^2} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial a^2} \qquad \frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial a^3}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

$$\frac{\partial J}{\partial a^l} = \frac{\partial J}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial a^l}$$



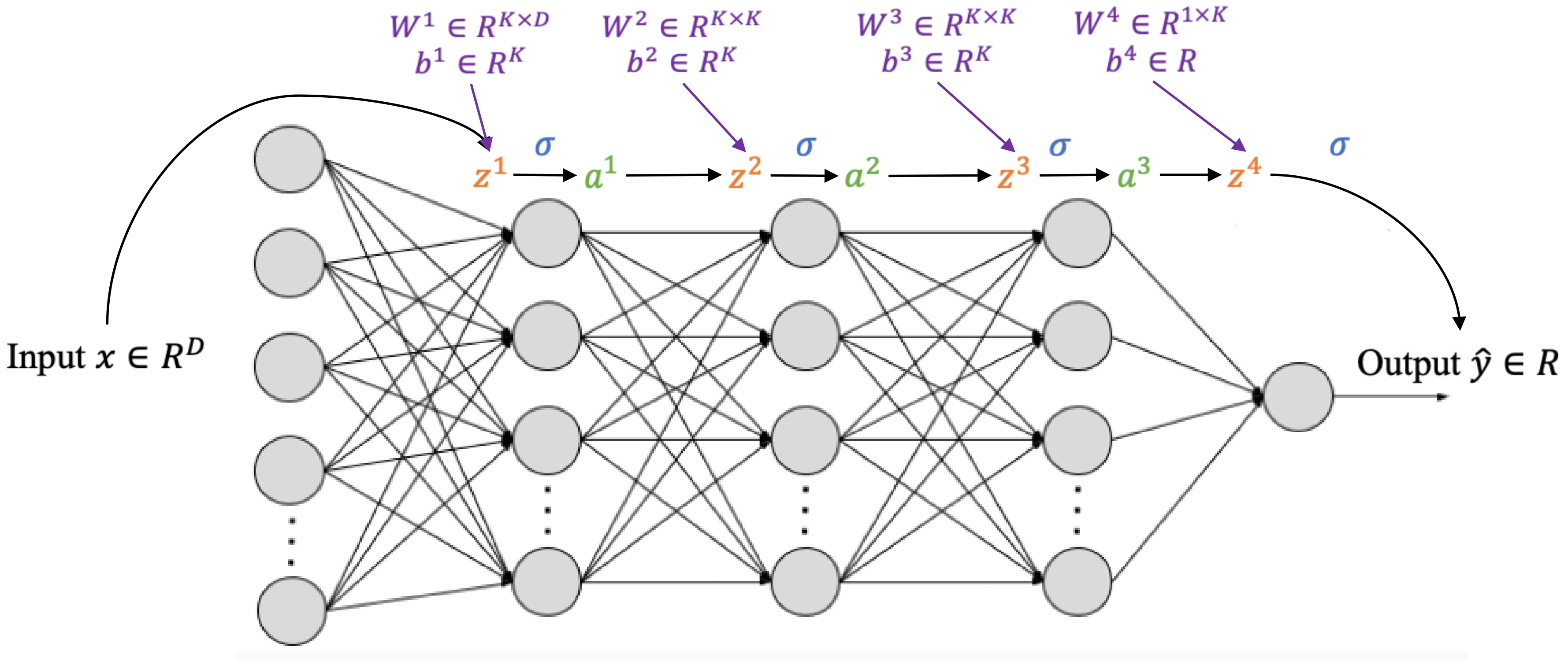
$$J = \frac{1}{2}(\hat{y} - y)^2$$

forward propagation:

$$z^{l+1} = W^{l+1}a^l + b^{l+1} \qquad \frac{\partial z^{l+1}}{\partial a^l} = W^{l+1}$$

Chain rule

$$\frac{\partial J}{\partial a^l} = \frac{\partial J}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial a^l}$$



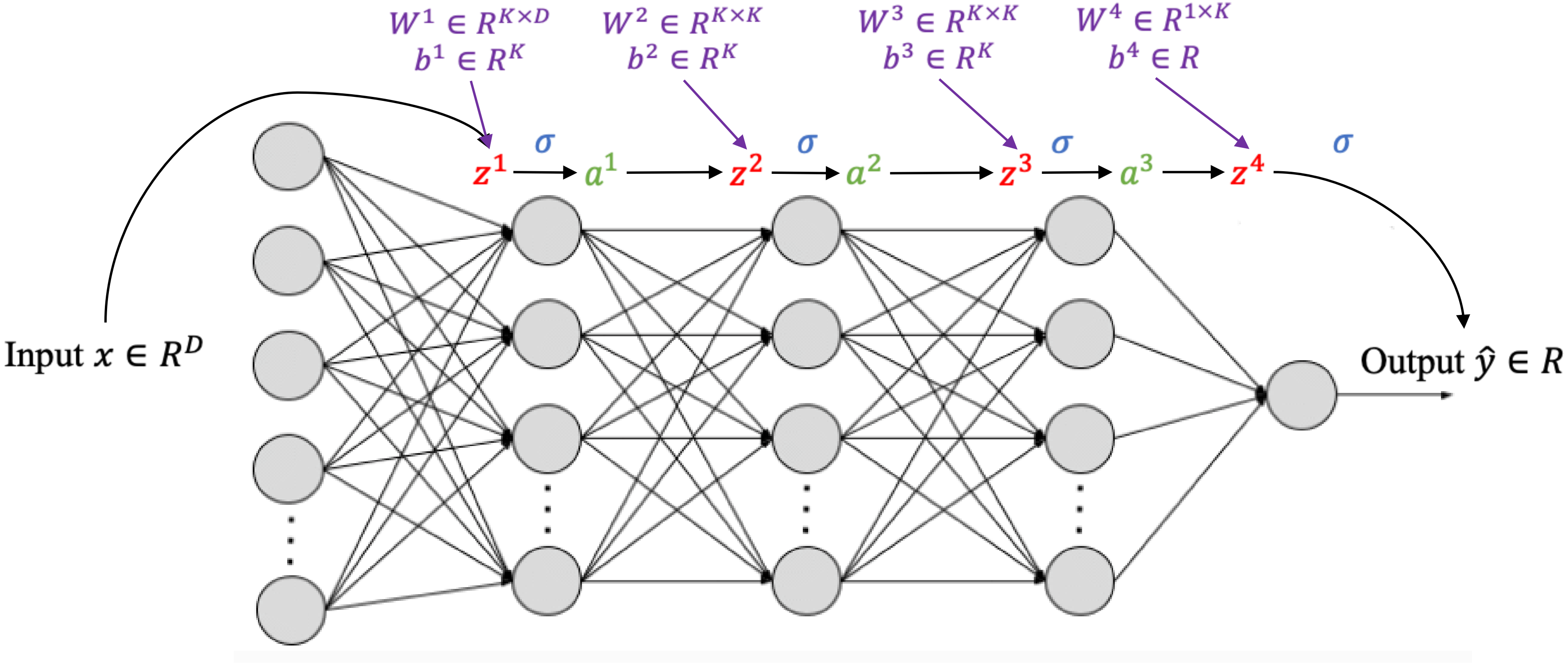
$$J = \frac{1}{2}(\hat{y} - y)^2$$

forward propagation:

$$z^{l+1} = W^{l+1} a^l + b^{l+1} \qquad \frac{\partial z^{l+1}}{\partial a^l} = W^{l+1}$$

Chain rule

$$\frac{\partial J}{\partial a^l} = \boxed{\frac{\partial J}{\partial z^{l+1}}} \frac{\partial z^{l+1}}{\partial a^l}$$

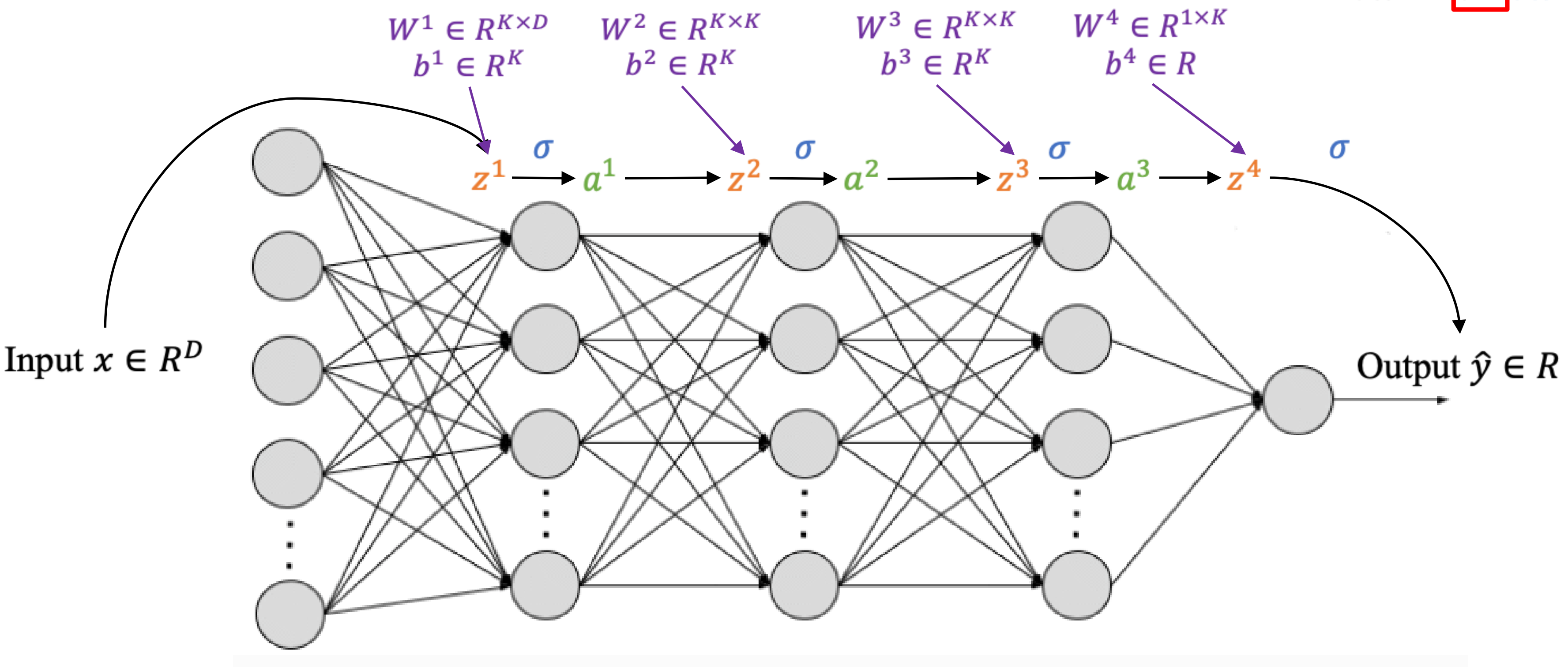


$$J = \frac{1}{2}(\hat{y} - y)^2$$

We come back!

Backward Propagation

$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}}\frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}}\frac{\partial z^l}{\partial b^l}$$



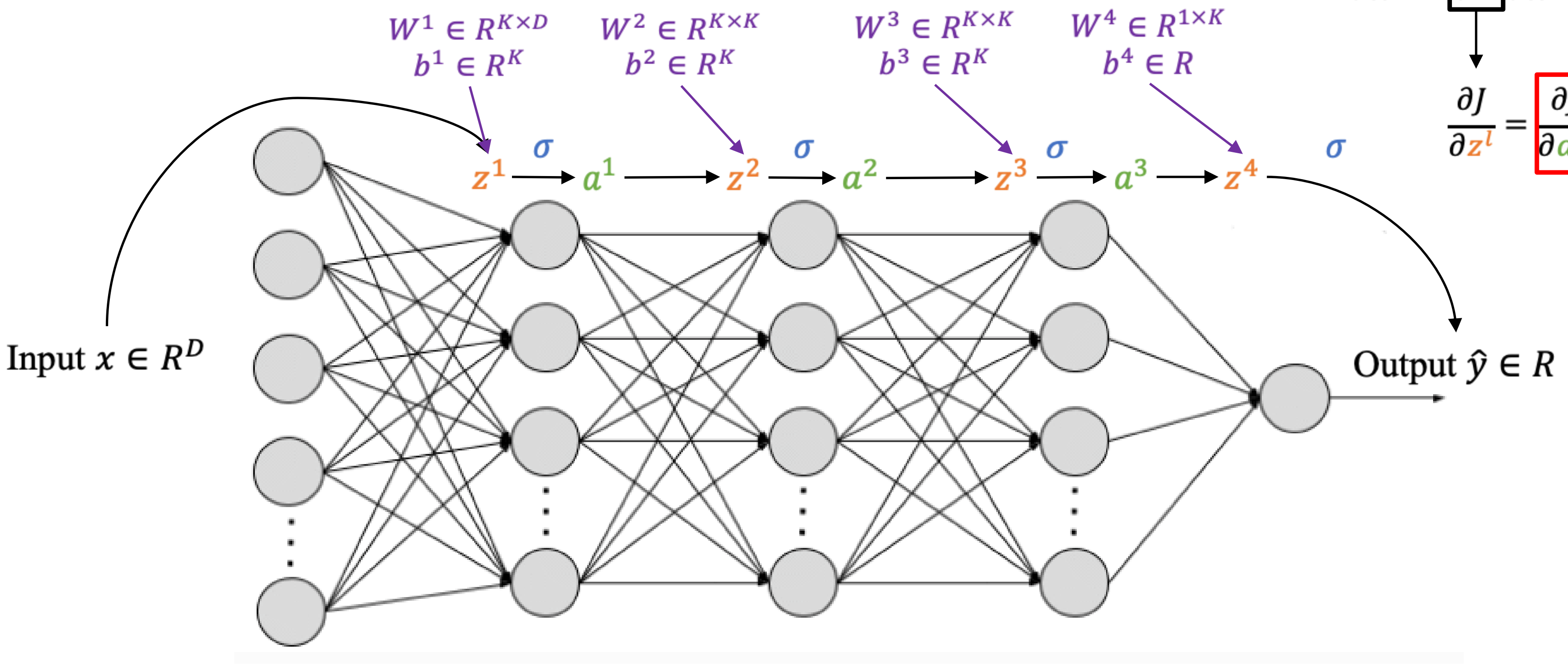
$$J = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{\partial z^l}{\partial W^l} = a^{l-1} \qquad \frac{\partial z^l}{\partial b^l} = \mathbf{1}$$

Backward Propagation

$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$

$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

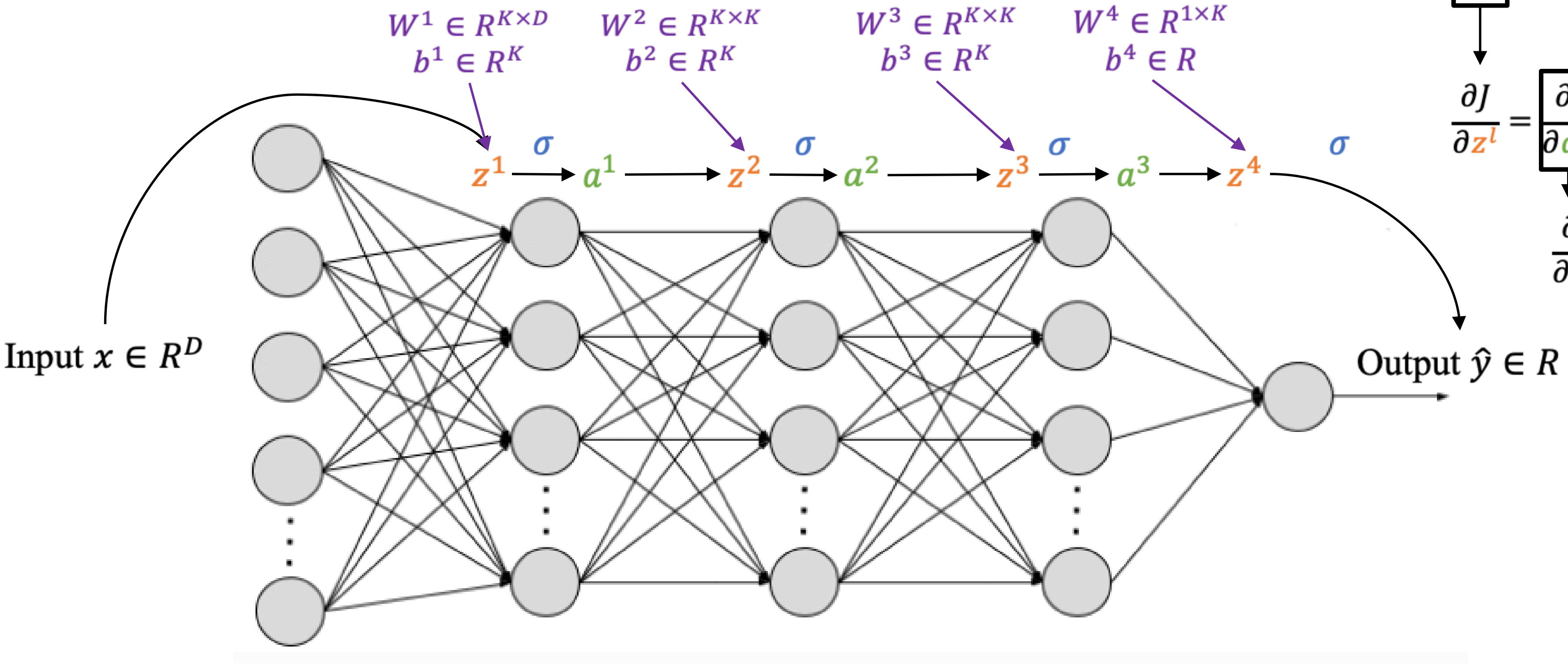
$$\frac{\partial a^l}{\partial z^l} = a^l(1 - a^l)$$

Backward Propagation

$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$

$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$

$$\frac{\partial J}{\partial a^l} = \boxed{\frac{\partial J}{\partial z^{l+1}}} \frac{\partial z^{l+1}}{\partial a^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

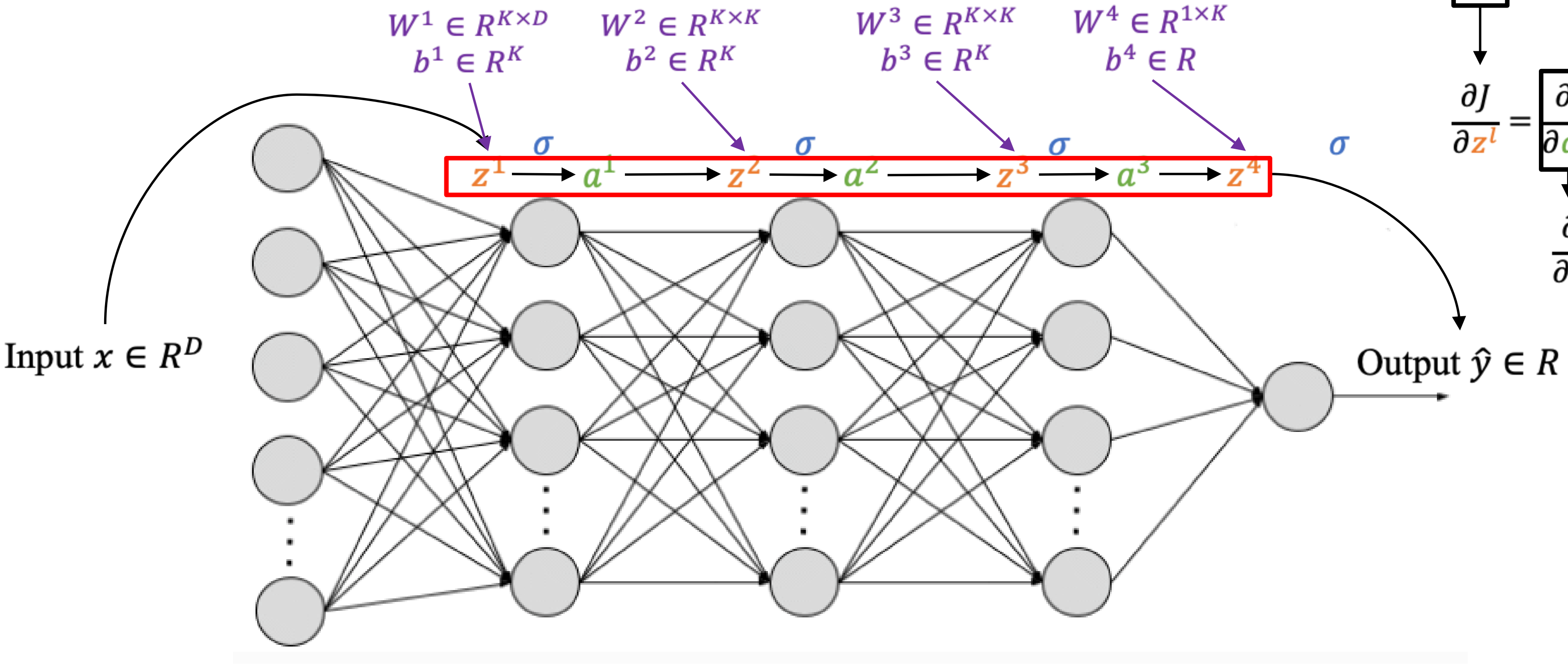
$$\frac{\partial z^{l+1}}{\partial a^l} = W^{l+1}$$

Backward Propagation

$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$

$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$

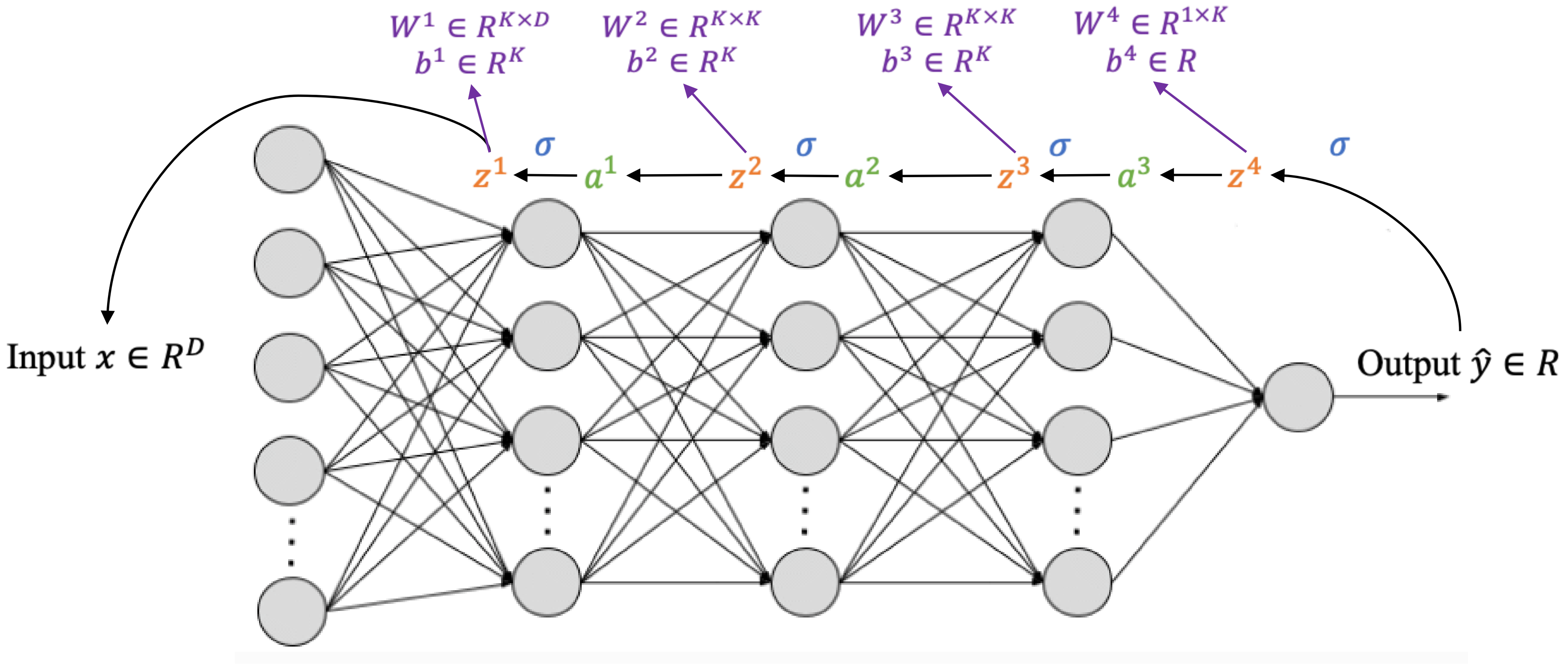
$$\frac{\partial J}{\partial a^l} = \boxed{\frac{\partial J}{\partial z^{l+1}}} \frac{\partial z^{l+1}}{\partial a^l}$$



$$J = \frac{1}{2}(\hat{y} - y)^2$$

Reverse the direction!

Backward Propagation



$$J = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$

$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$

$$\frac{\partial J}{\partial a^l} = \boxed{\frac{\partial J}{\partial z^{l+1}}} \frac{\partial z^{l+1}}{\partial a^l}$$