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 Modeling Cont.

Spring 2025

Benyou Wang
School of Data Science

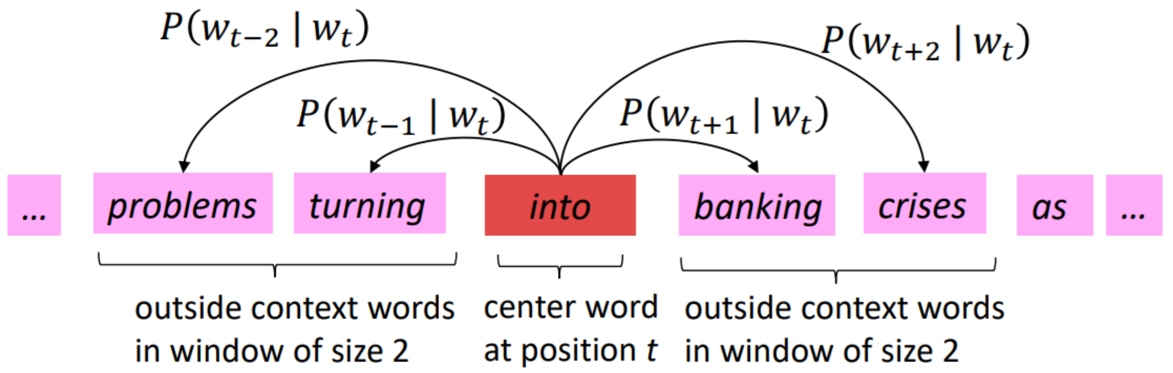
To recap….

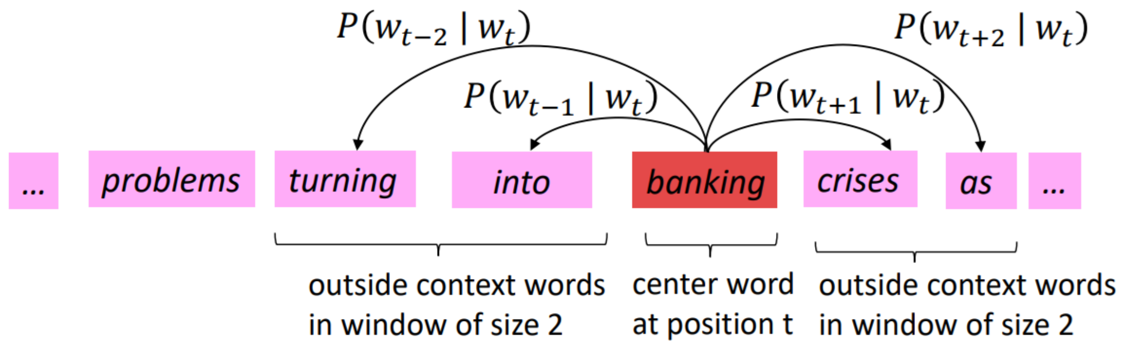
Word2Vec Overview

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

Idea:

- We have a large corpus ("body") of text: a long list of words
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context ("outside") words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability





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

Word2vec: objective function

- We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

- Question: How to calculate $P(w_{t+j} \mid w_t; \theta)$

Answer: We will use two vectors per word *w*:

- v_w when w is a center word
- u_w when w is a context word

Then for a center word *c* and *a* context word *o:* (softmax)

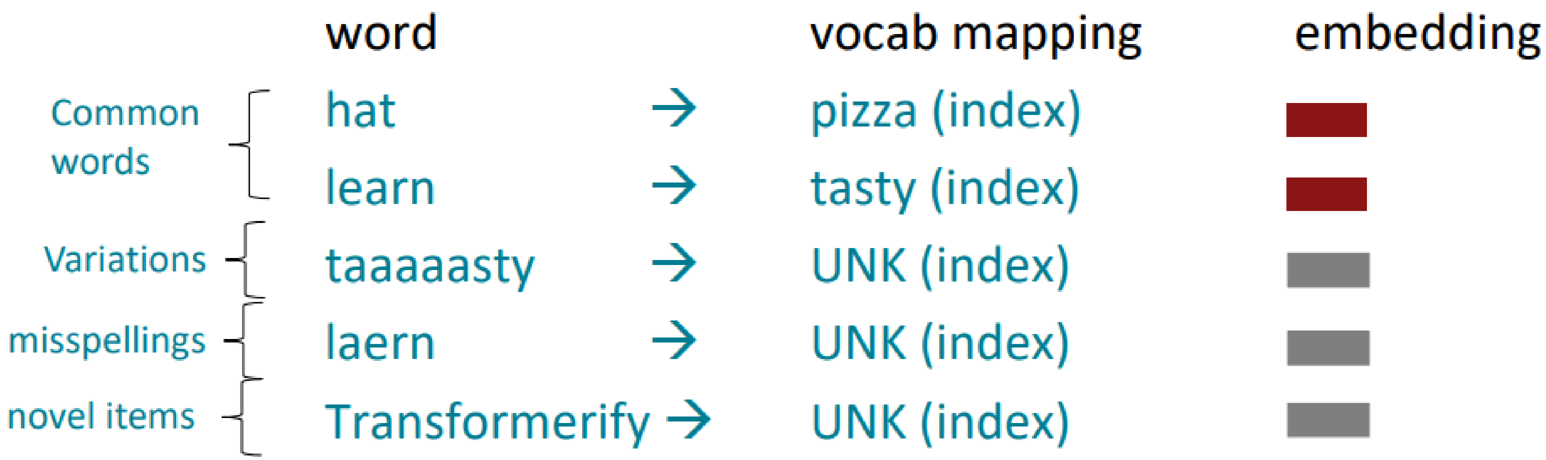
$$P(o \mid c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w \in V} \exp\left(u_w^T v_c\right)}$$

"max" because amplifies probability of largest
"soft" because still assigns some probability to smaller

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

Word structure and subword models

We assume a fixed vocab of tens of thousands of words, built from the training set.
All novel words seen at test time are mapped to a single UNK.



Finite vocabulary assumptions make even less sense in many languages.
- Many languages exhibit complex morphology, or word structure.
- The effect is more word types, each occurring fewer times.

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

From static word vector to contextualized word vectors

What’s wrong with word2vec?

- One vector for each word type

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

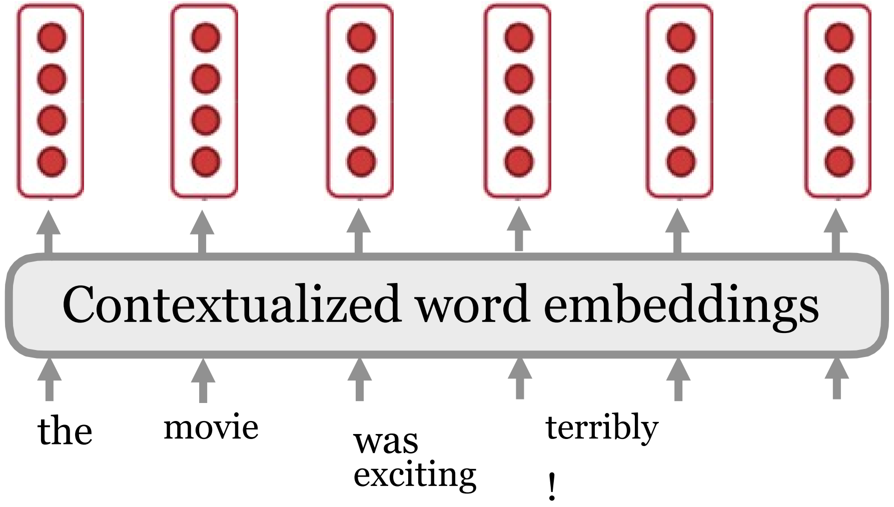
- Complex characteristics of word use: semantics, syntactic behavior, and connotations

- 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 ...

Contextualized word embeddings

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

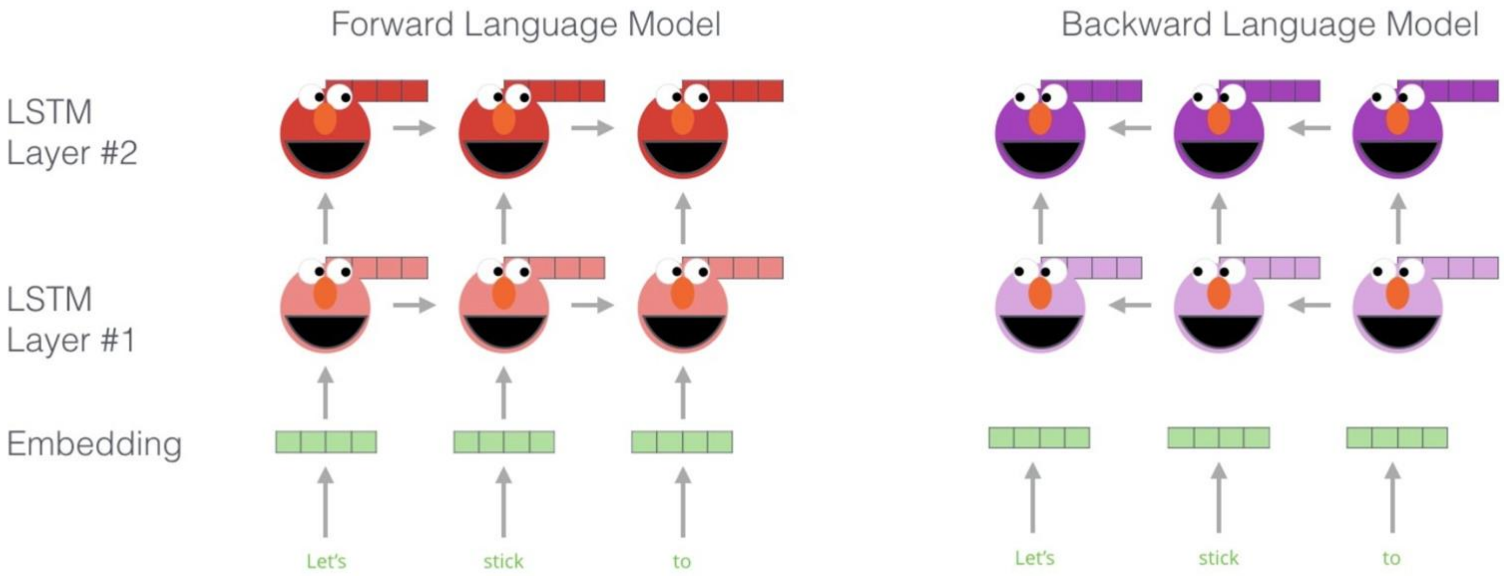


$$f : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

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

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)

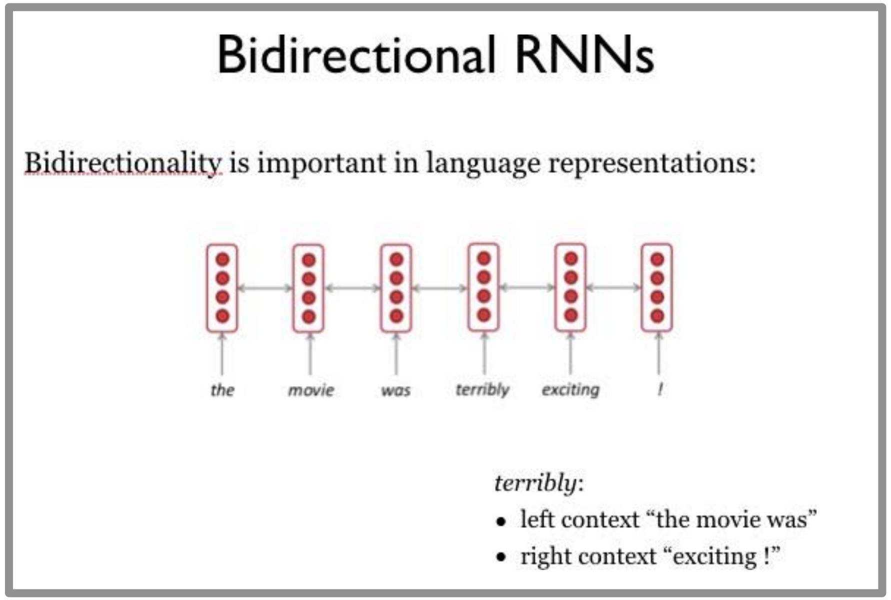
#3. The weights are not freezed, called fine-tuning

Bidirectional encoders

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

Language understanding is bidirectional

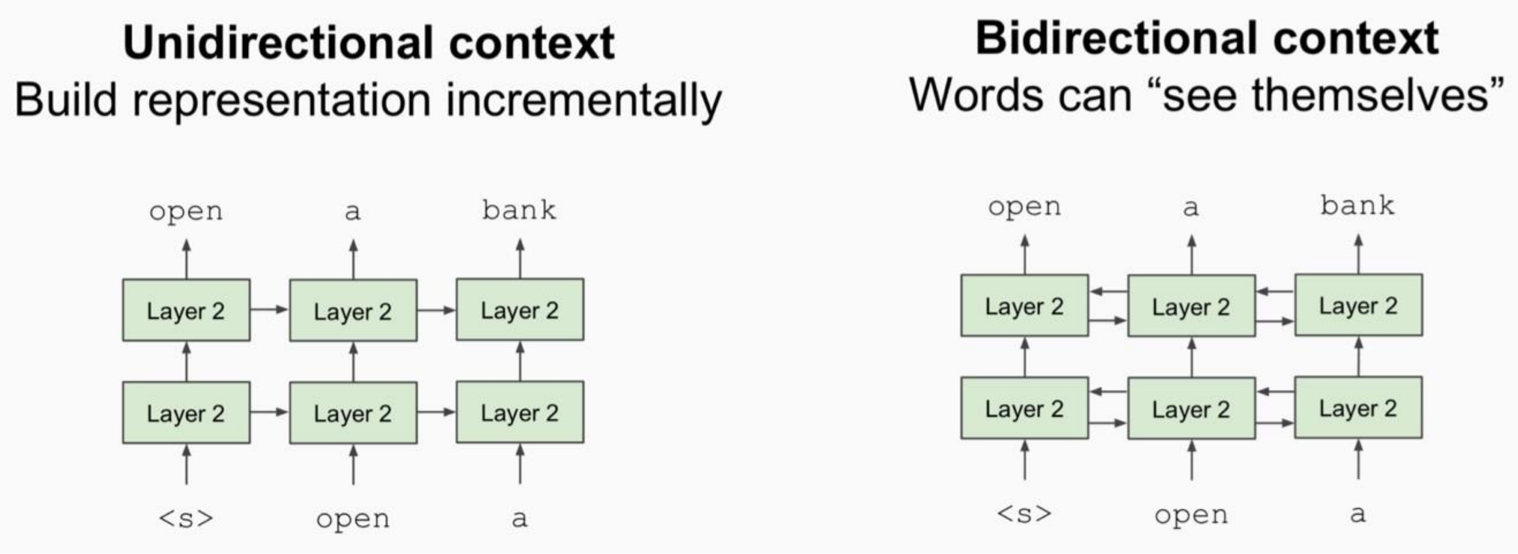
Lecture 9:



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

store ↑ gallon ↑

the man went to the [MASK] to buy a [MASK] of milk

- 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.

Because [MASK] is never seen when BERT is used...

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]`

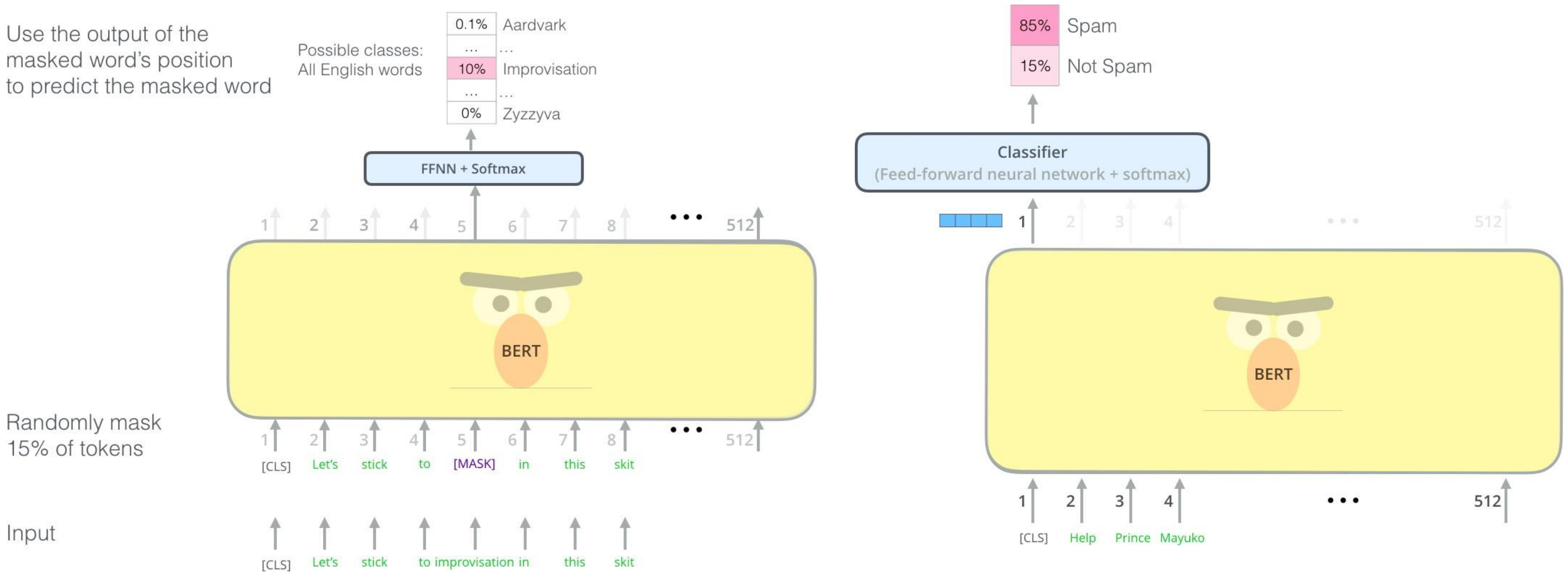
`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

Recent papers show that NSP is not necessary…

(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

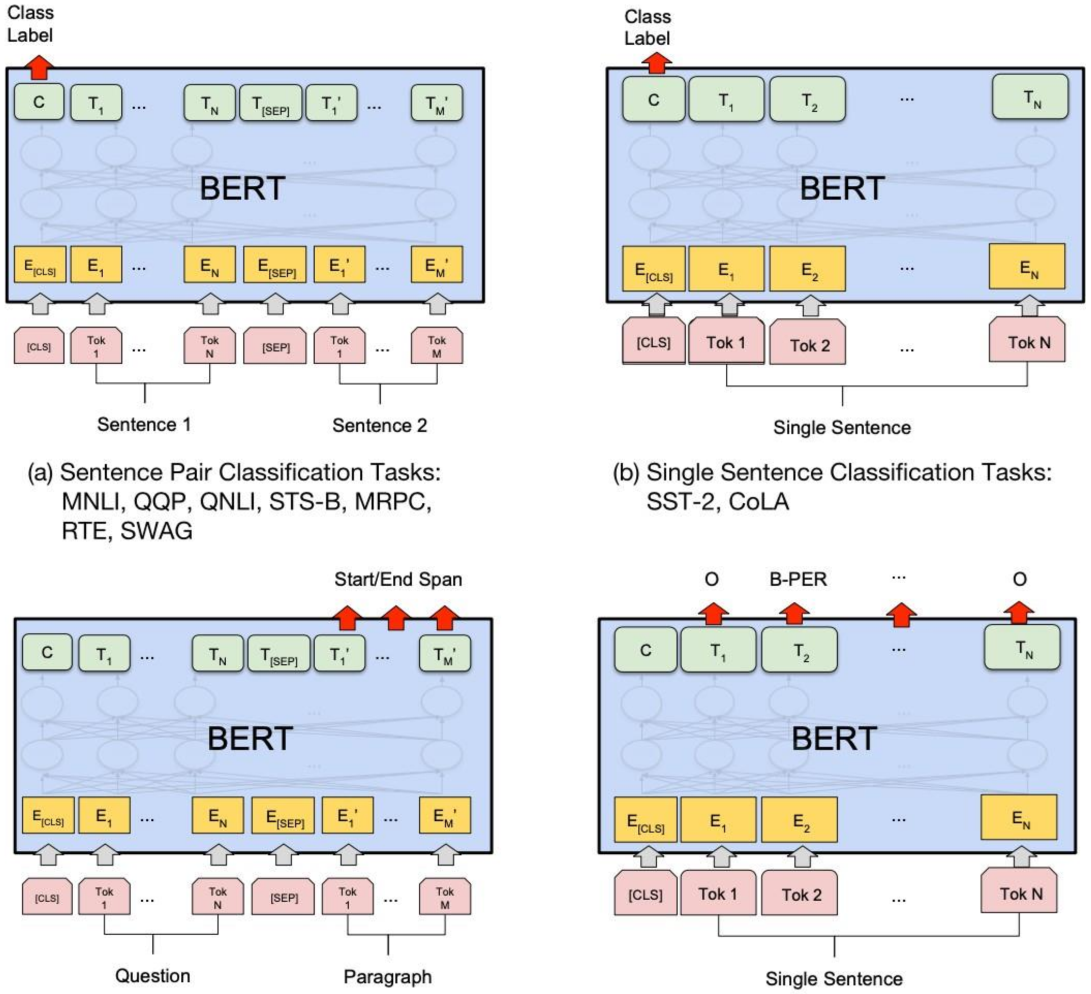
Pre-training and fine-tuning



Pre-training Fine-tuning

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

Applications



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

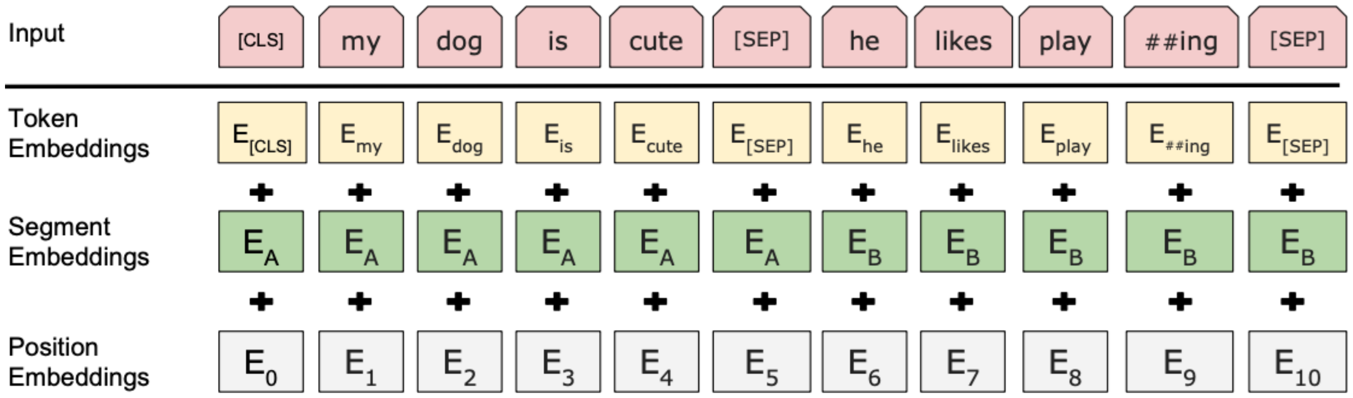
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

More details

- Input representations



- Use word pieces instead of words: playing => play ##ing ← Assignment 4
- Trained 40 epochs on Wikipedia (2.5B tokens) + BookCorpus (0.8B tokens)
- Released two model sizes: BERT_base, BERT_large

Variants of contextualized word vectors

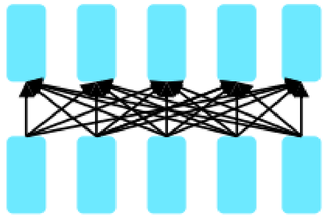
Overview

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

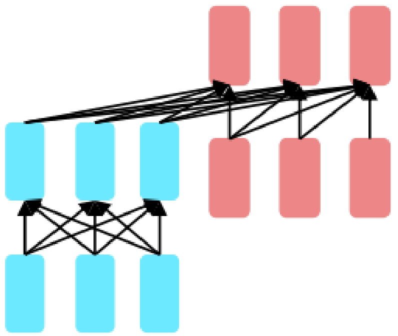
Benyou Wang et.al. Pre-trained Language Models in Biomedical Domain: A Systematic Survey. ACM Computing Survey.

Pretraining for three types of architectures

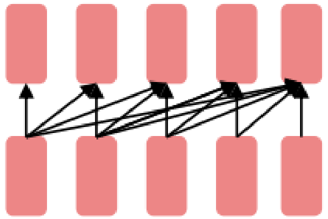
The neural architecture influences the type of pretraining, and natural use cases.



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



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



- 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

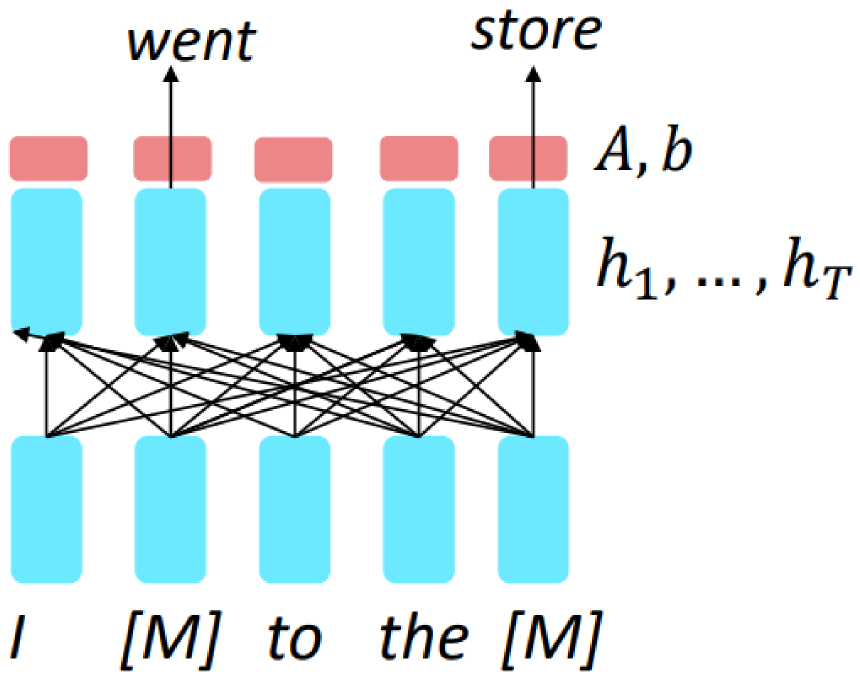
Pretraining encoders: what pretraining objective to use?

So far, we've looked at language model pretraining. But **encoders get bidirectional context**, so we can't do language modeling!

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim A w_i + b$$

Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x \mid \tilde{x})$. Called **Masked LM.**



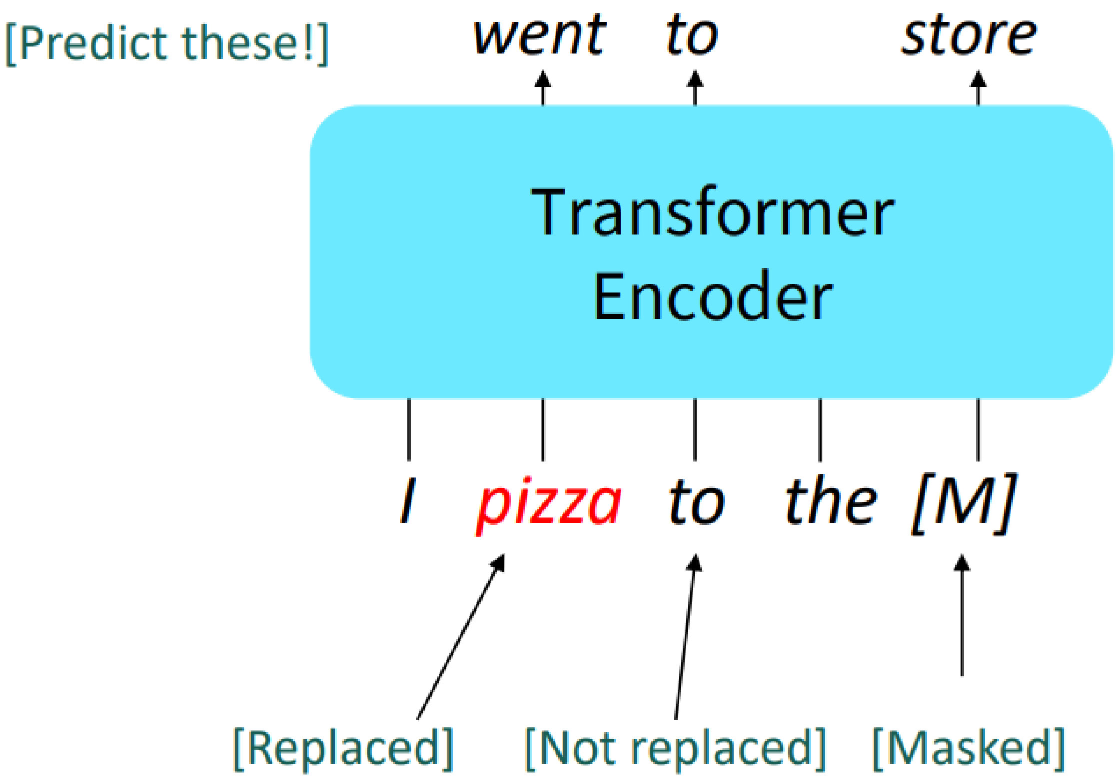
[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and released the weights of a pretrained Transformer, a model they labeled BERT.

Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)

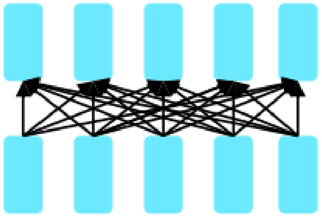


[Devlin et al., 2018]

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

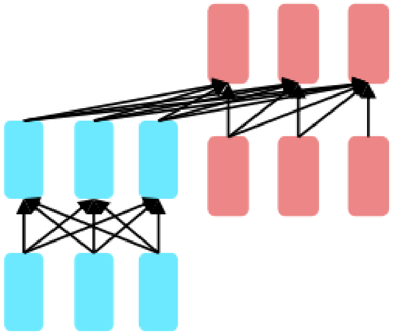
Pretraining for three types of architectures

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



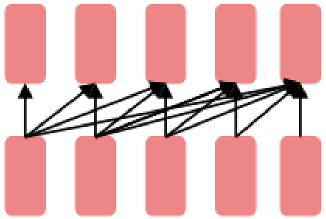
Encoders

- 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

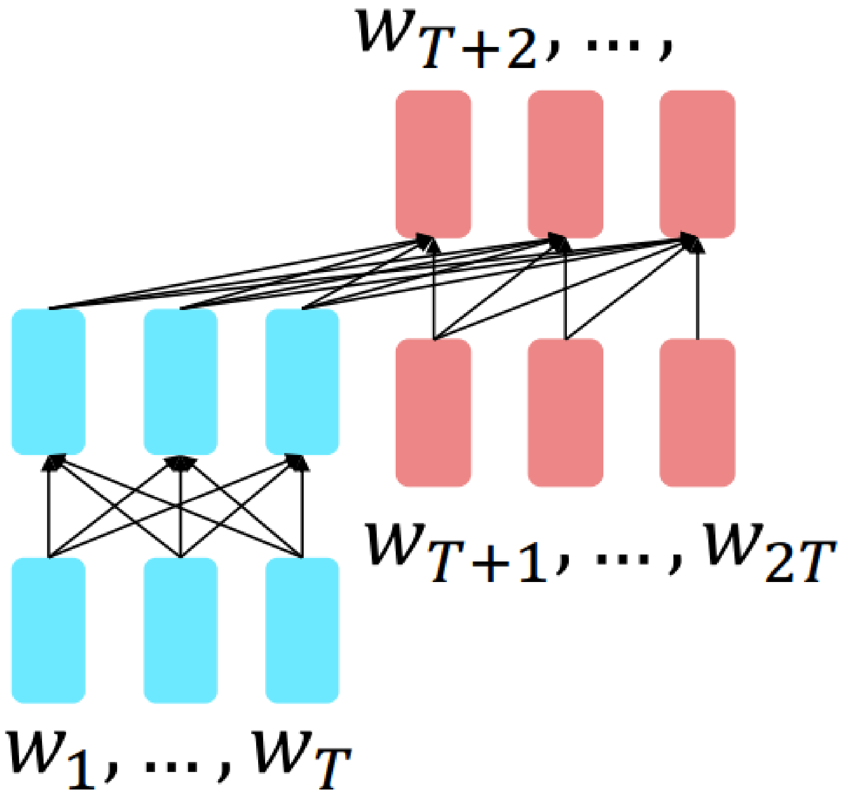
- Language models! What we’ve seen so far.
- Nice to generate from; can’t condition on future words

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.

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



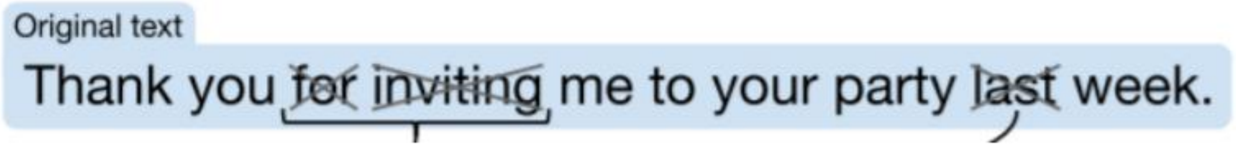
The **encoder** portion benefits from bidirectional context;
The **decoder** portion is used to train the whole model through language modeling.

[Raffel et al., 2018]

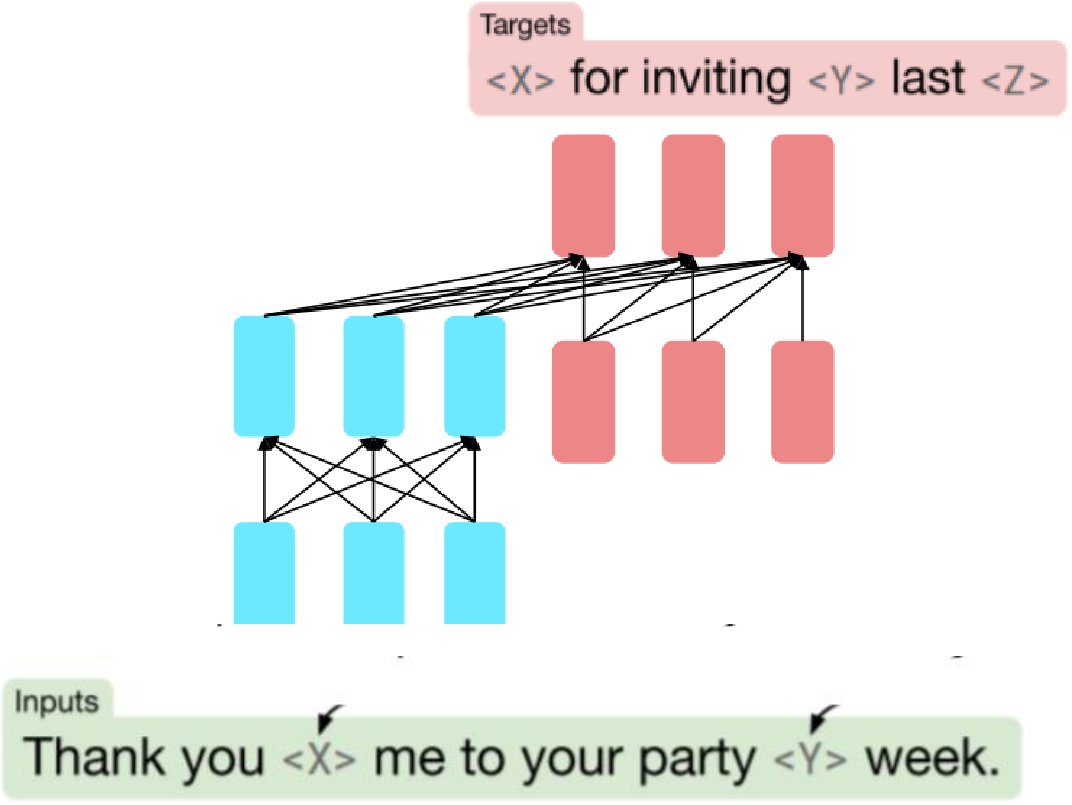
Pretraining encoder-decoders: what pretraining objective to use?

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!



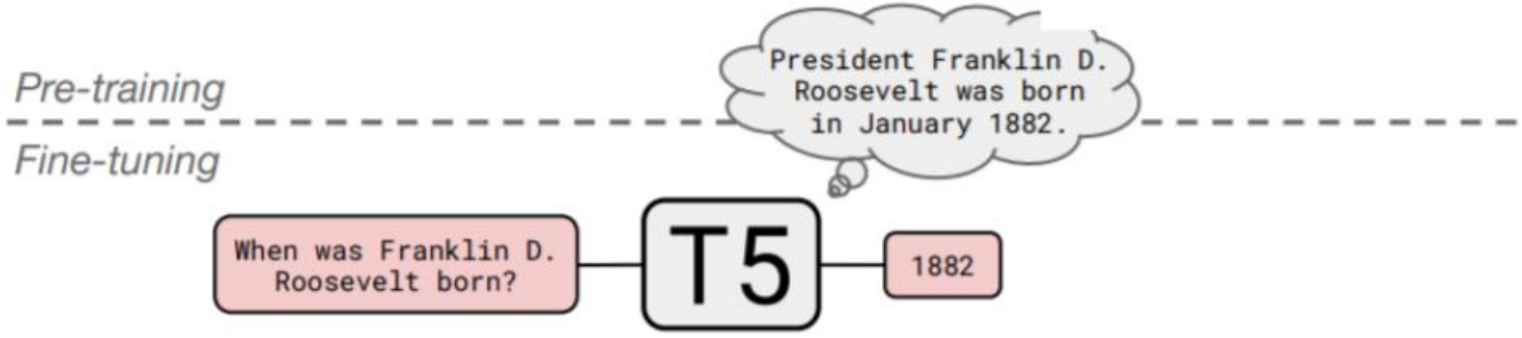
This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



[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.



	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**	

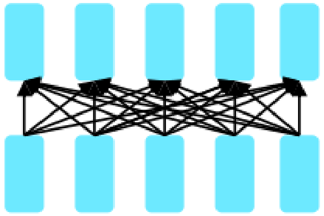
NQ: Natural Questions
WQ: WebQuestions
TQA: Trivia QA

All "open-domain" versions

[Raffel et al., 2018]

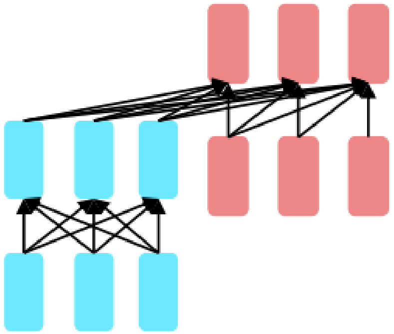
Pretraining for three types of architectures

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



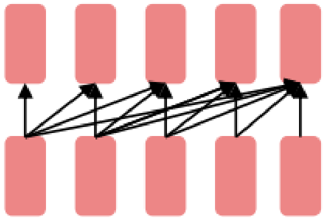
Encoders

- 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**

Back to the language model (next word predict)

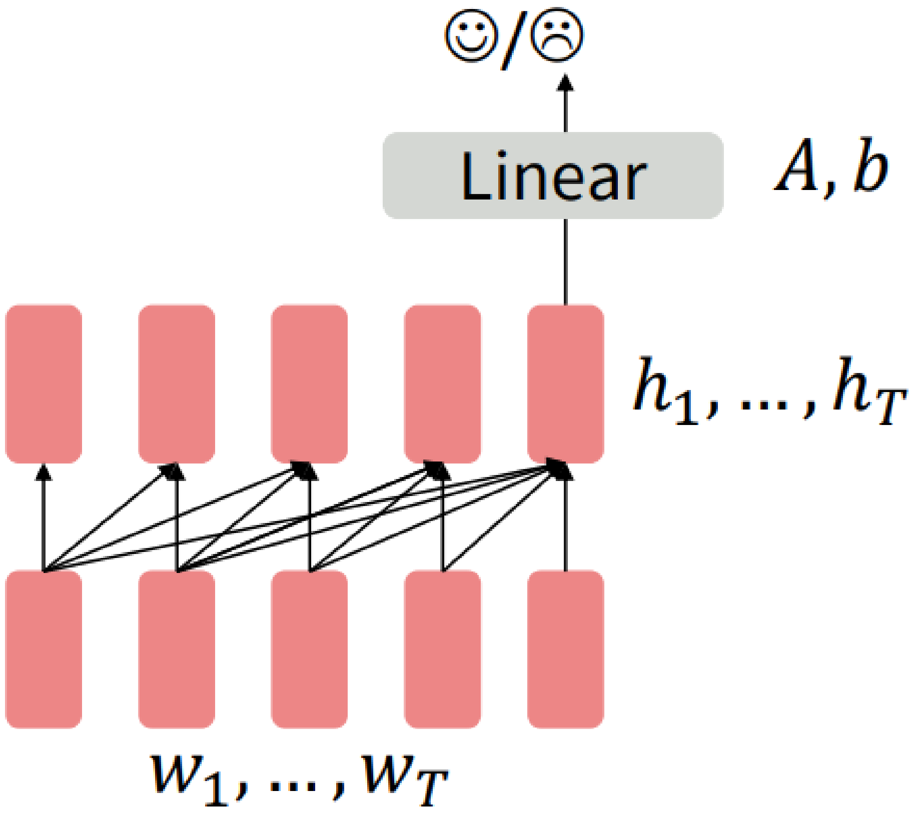
Pretraining decoders

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.

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

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

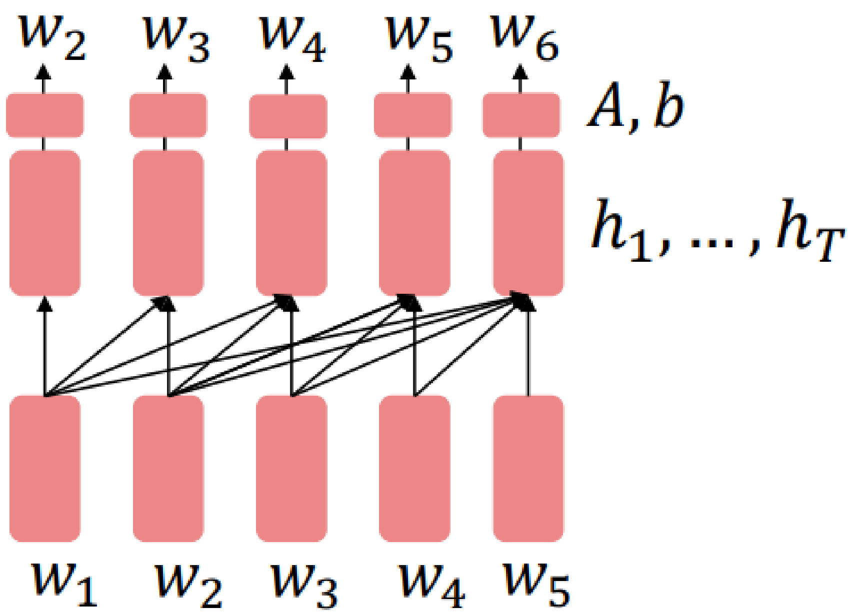
It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t \mid w_{1:t-1})$

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.**

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

LLaMA, Open-Source Models

Meta hopes to advance NLP research through LLAMA, particularly in the **academic exploration** of large language models.

LLAMA can be customized for **a variety of use cases**, especially in **research and non-commercial projects** where it demonstrates greater suitability.

Through architectural optimizations, LLAMA can achieve performance similar to GPT-3 while using fewer computational resources.

Llama 2: Open Foundation and Fine-Tuned Chat Models. https://arxiv.org/pdf/2307.09288

Phi-3, Small but Strong,

Despite the compact size of the Phi-3 model, it has demonstrated performance on par with or even **superior** to larger models on **various academic benchmarks** in the market.

Phi-3's training method, inspired by children's learning, uses a "**curriculum-based**" strategy. It starts with simplified data, **gradually guiding the model to grasp complex concepts.**

Phi-3 adopts an architecture optimized specifically for **mobile devices**, with a design that supports **significant extension of the model's context length** through the **LongRope system**, thereby enhancing its ability to handle **long-sequence** data.

https://azure.microsoft.com/en-us/products/phi-3

Today's lecture

- **Language model in a narrow sense**
  (Probability theory, N-gram language model)

- Language model in broad sense

- **More thoughts on language model**

- LM (next word predict) 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 _____.

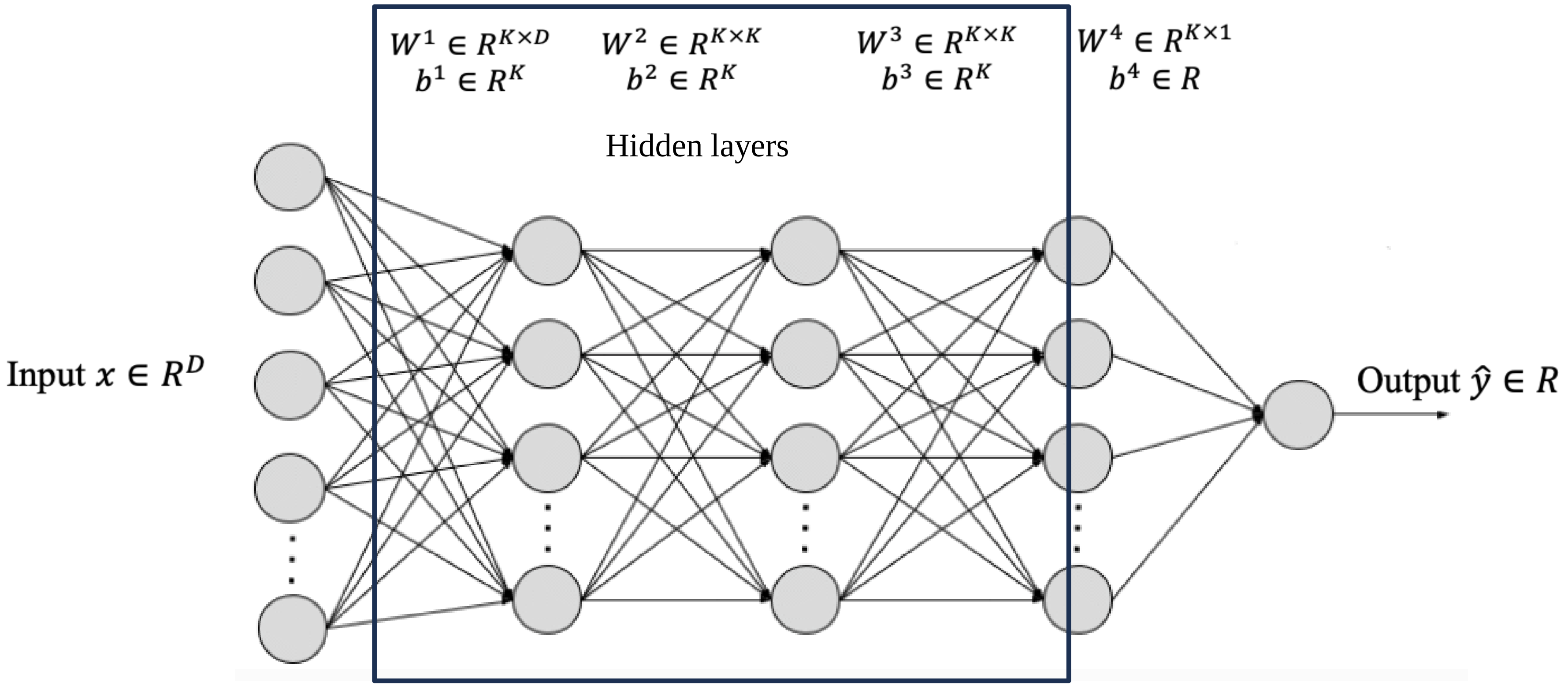
Acknowledgement

- 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

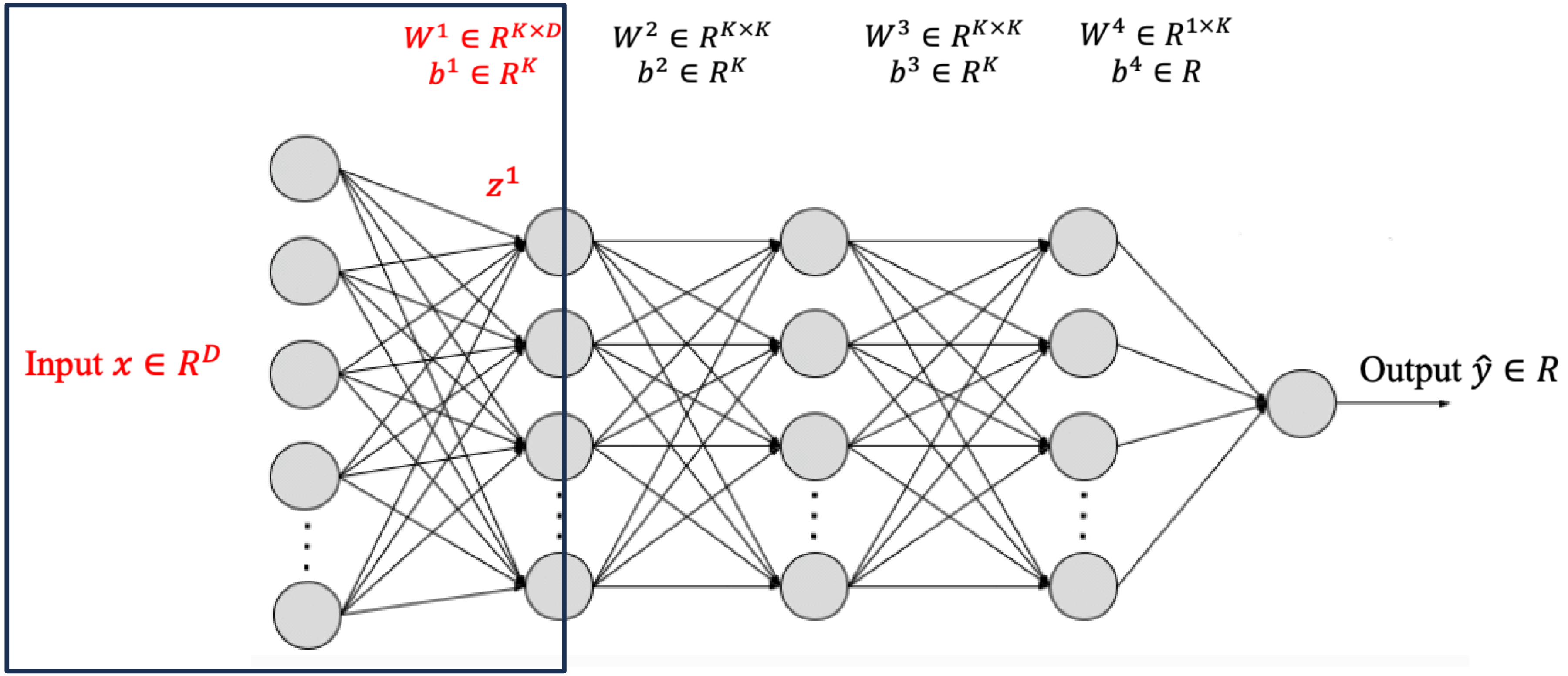
Neural Network



Suppose activation functions here are all sigmoid

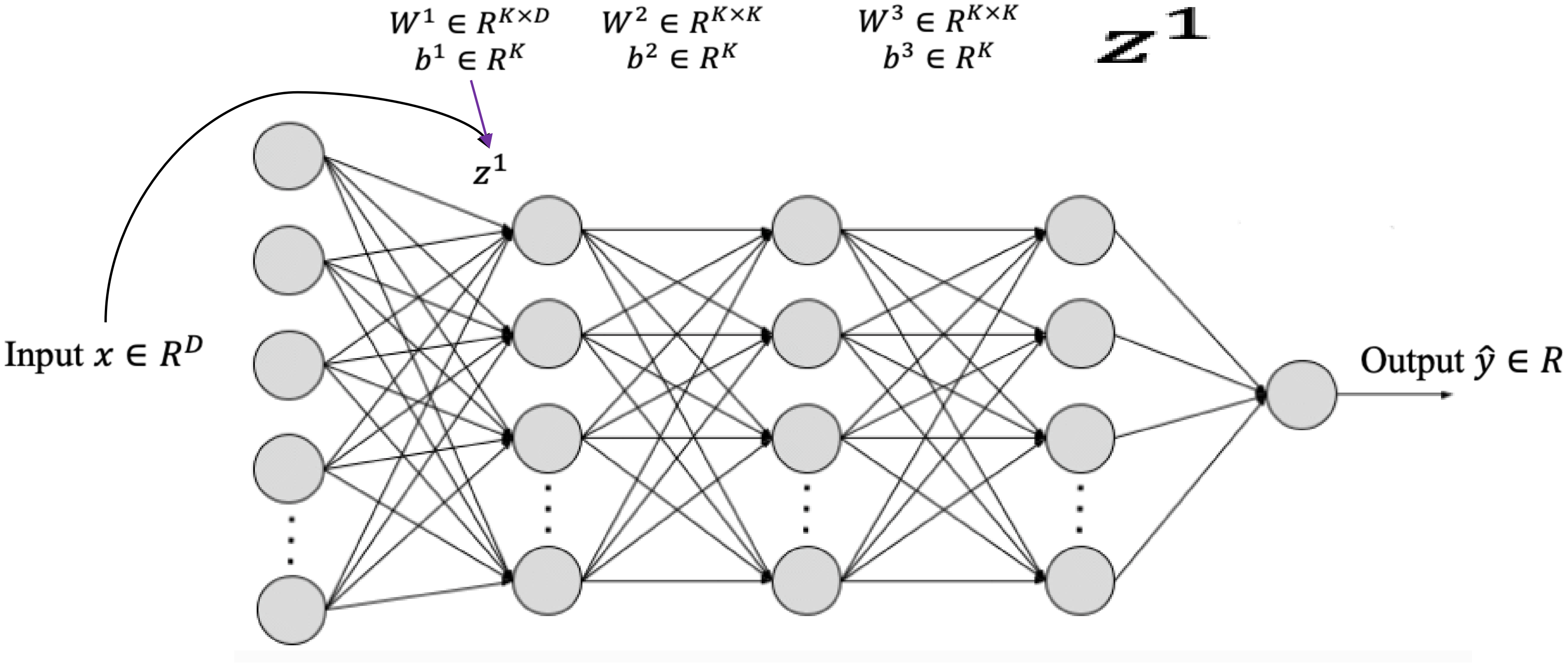
Forward Propagation

Forward Propagation



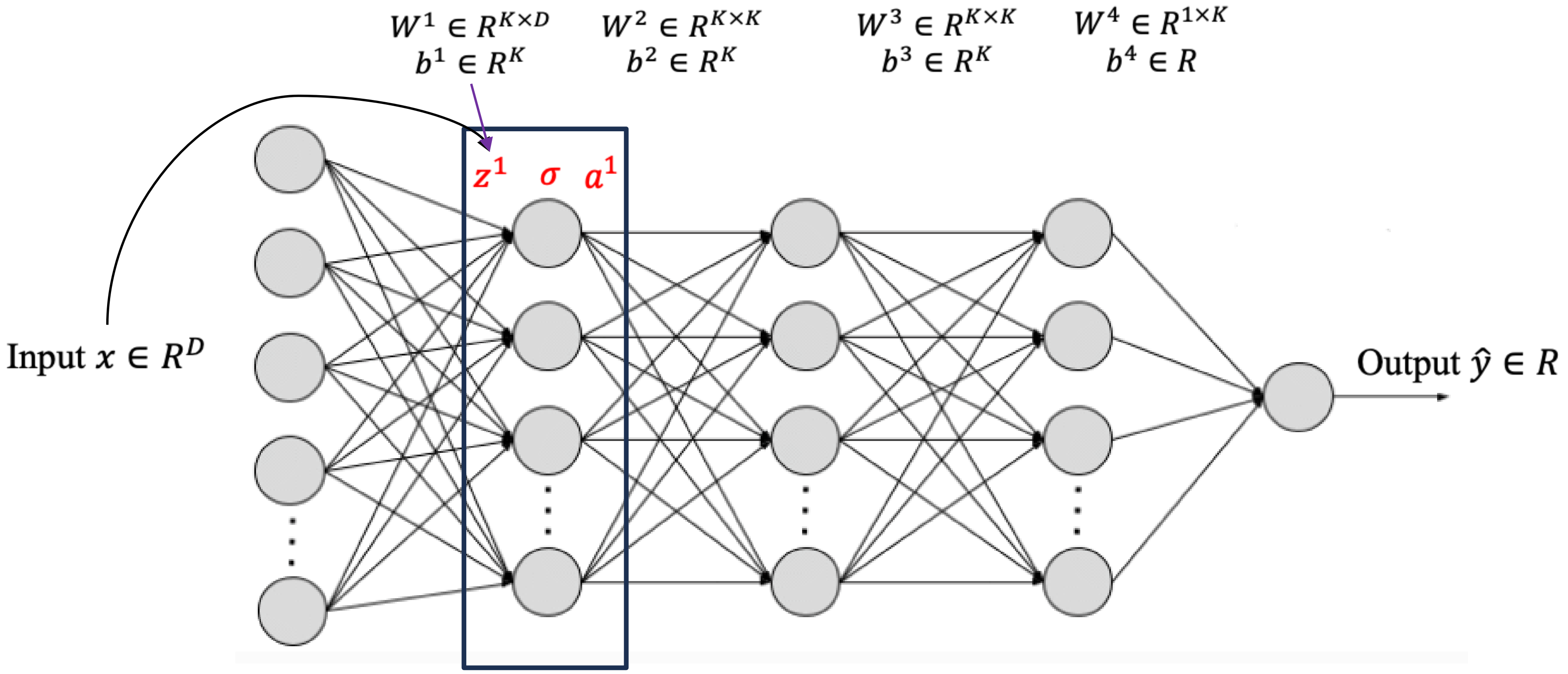
$z^1 = W^1x + b^1$

Forward Propagation



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

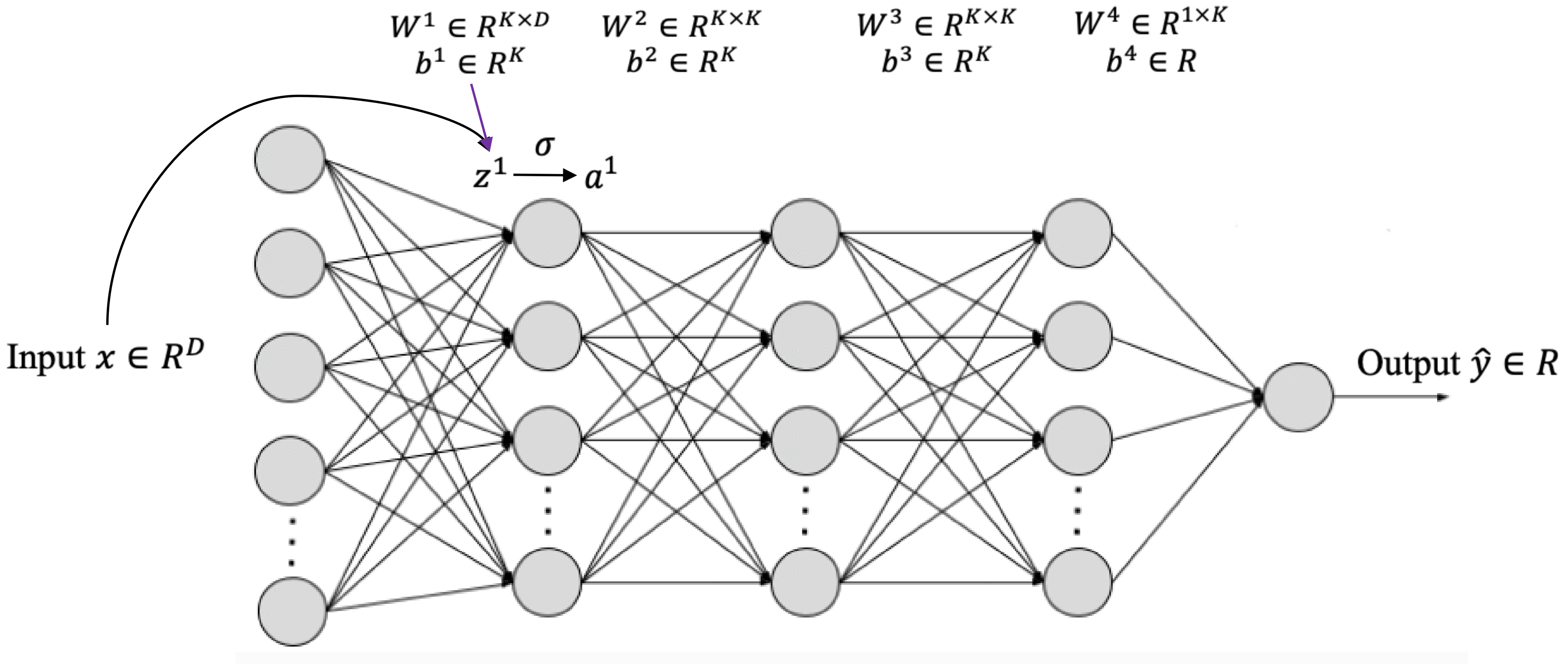
Forward Propagation



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

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

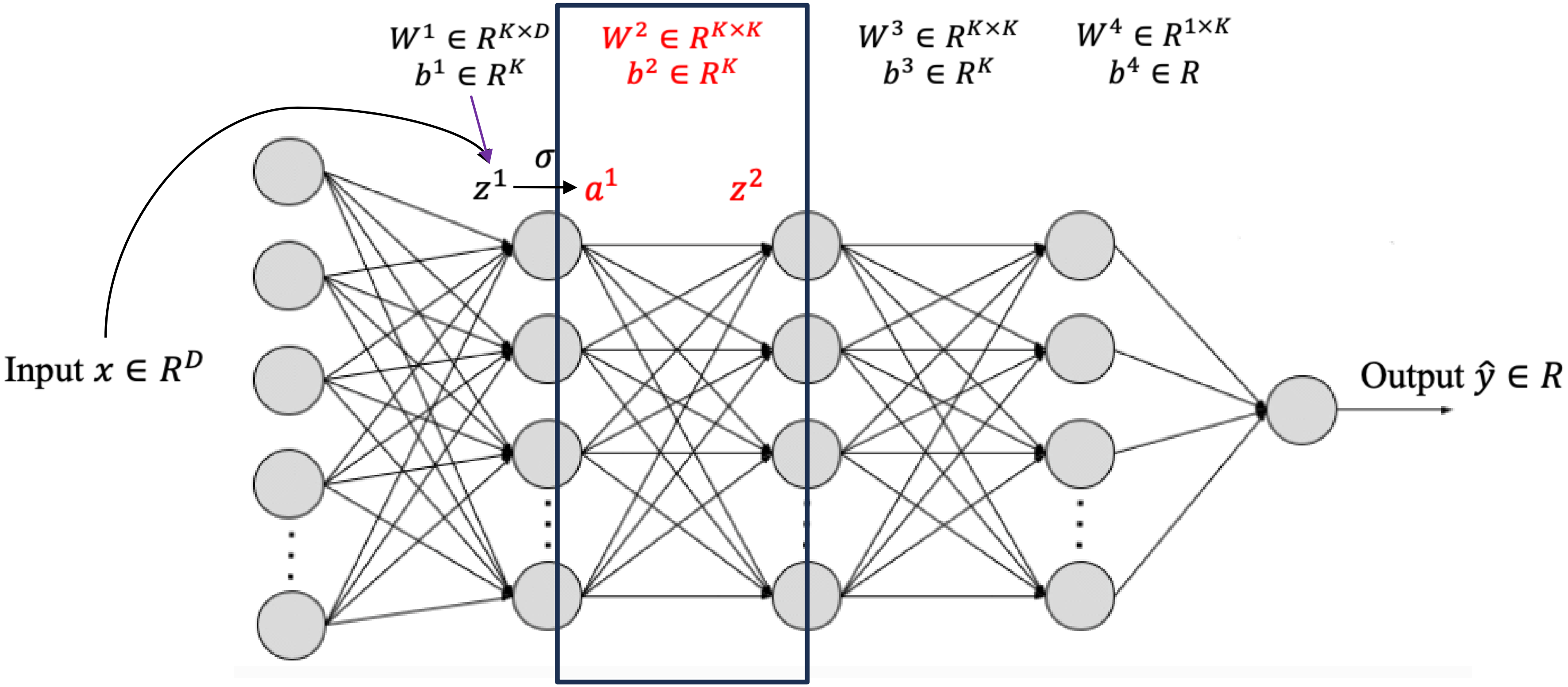
Forward Propagation



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

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

Forward Propagation

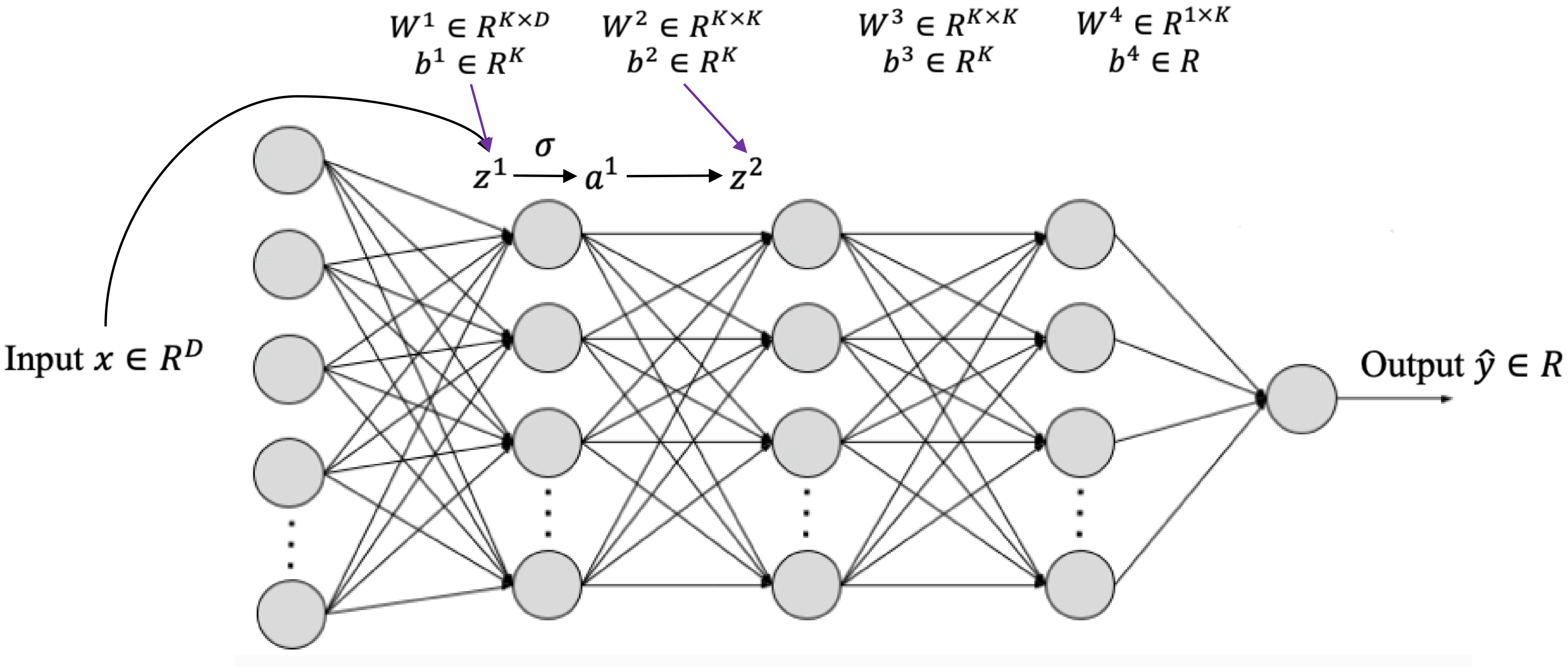


$z^1 = W^1 x + b^1$

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

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

Forward Propagation

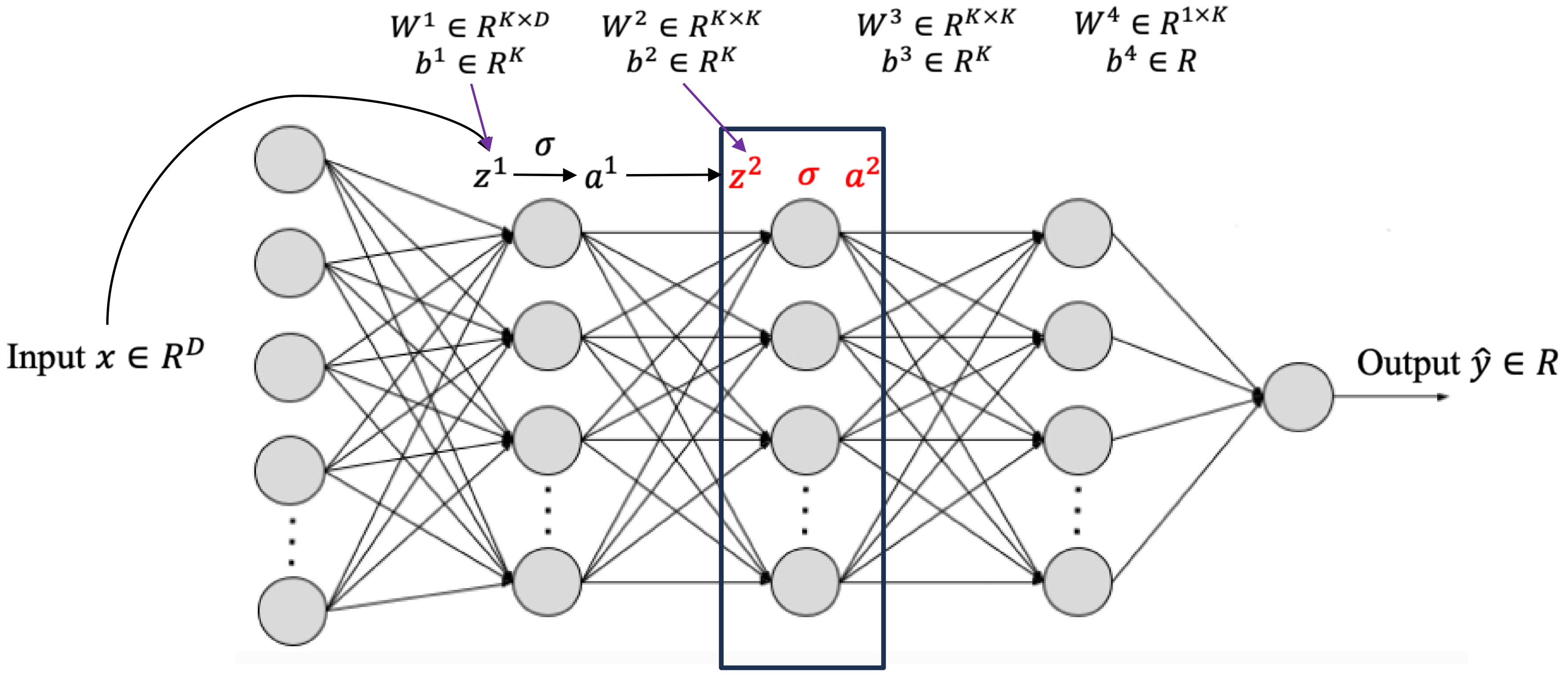


$z^1 = W^1 x + b^1$

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

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

Forward Propagation



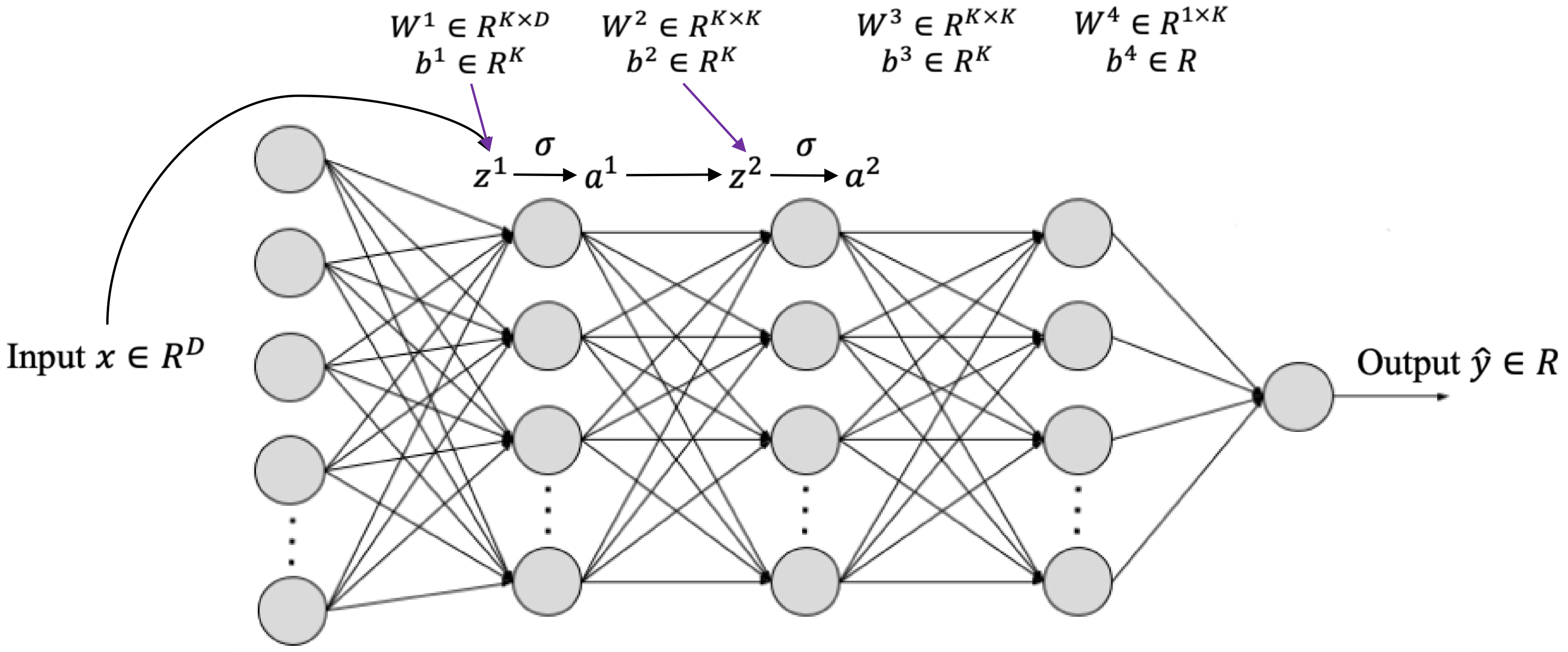
$z^1 = W^1 x + b^1$

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

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

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

Forward Propagation



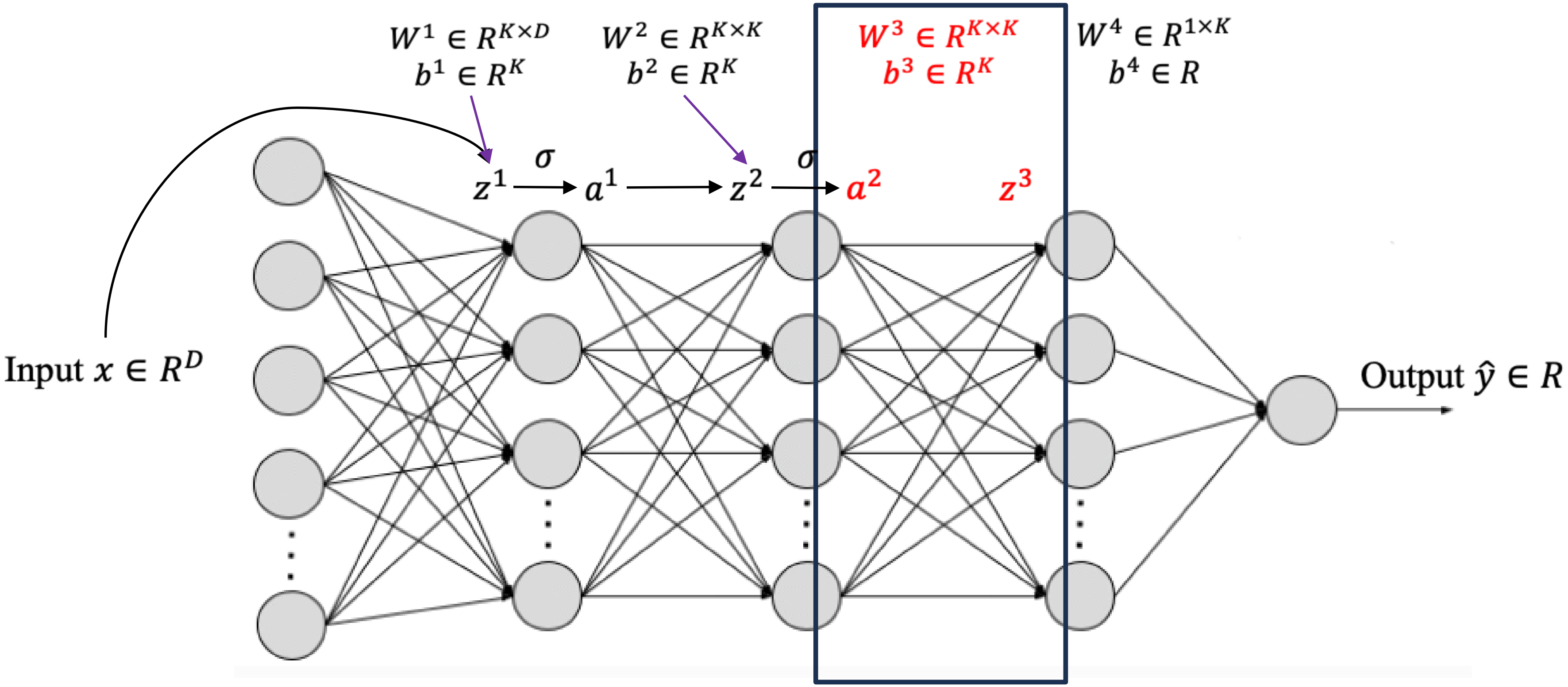
$z^1 = W^1 x + b^1$

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

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

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

Forward Propagation



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

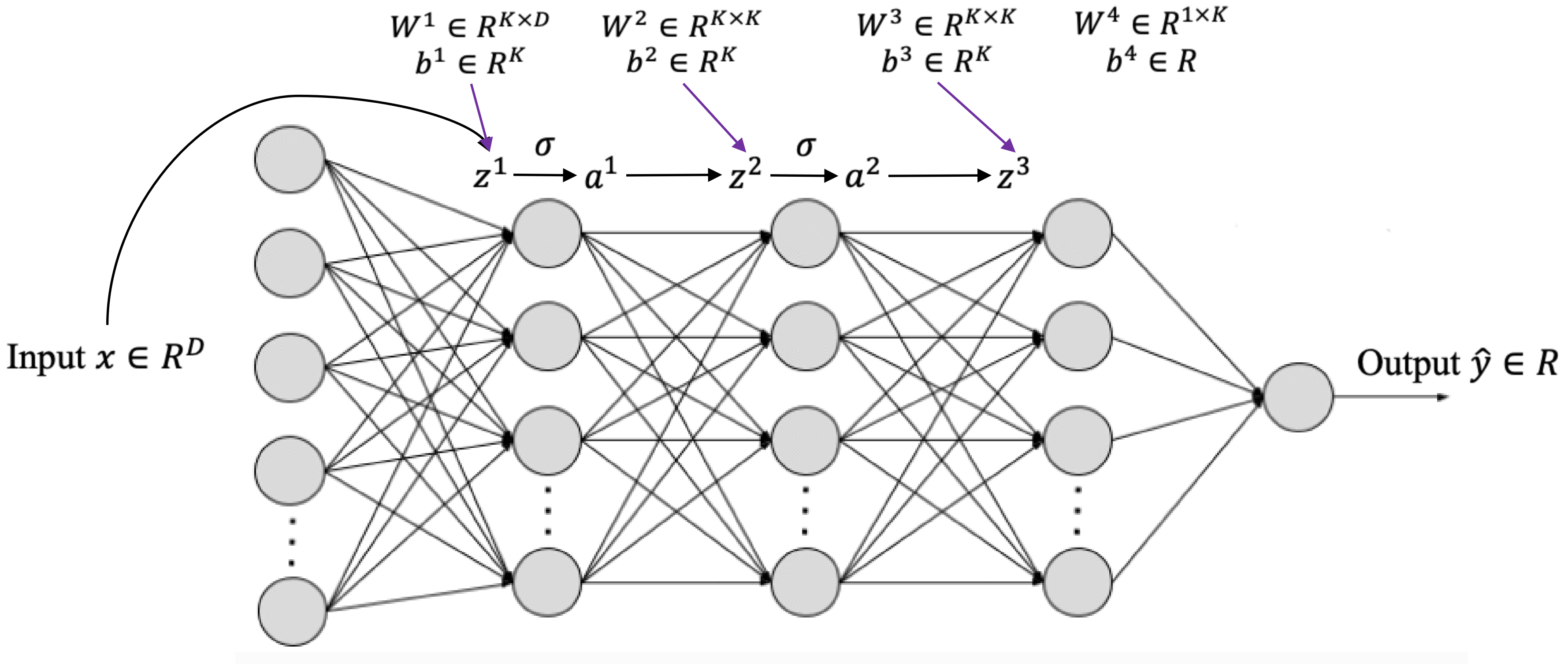
$a^1 = \sigma(z^1)$

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

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

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

Forward Propagation



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

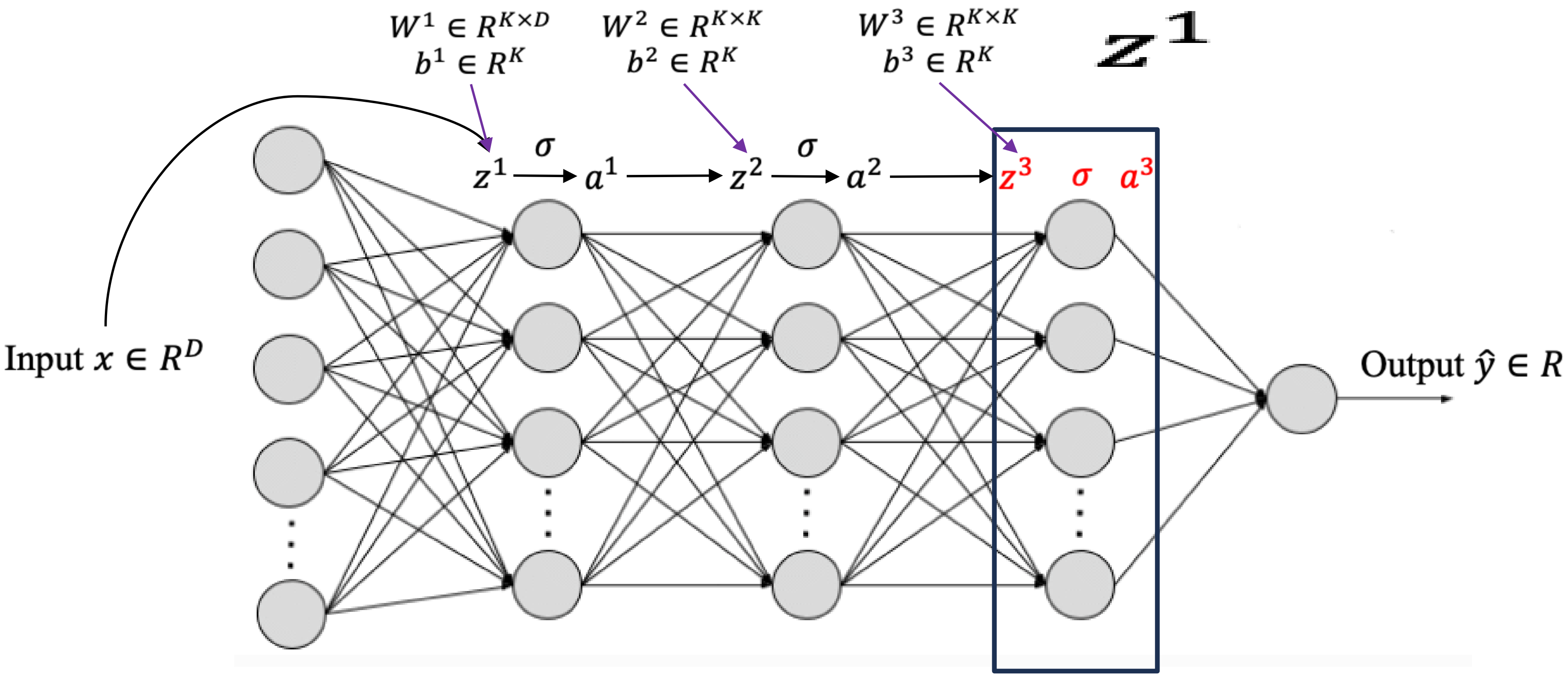
$$a^1 = \sigma(z^1)$$

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

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

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

Forward Propagation



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

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

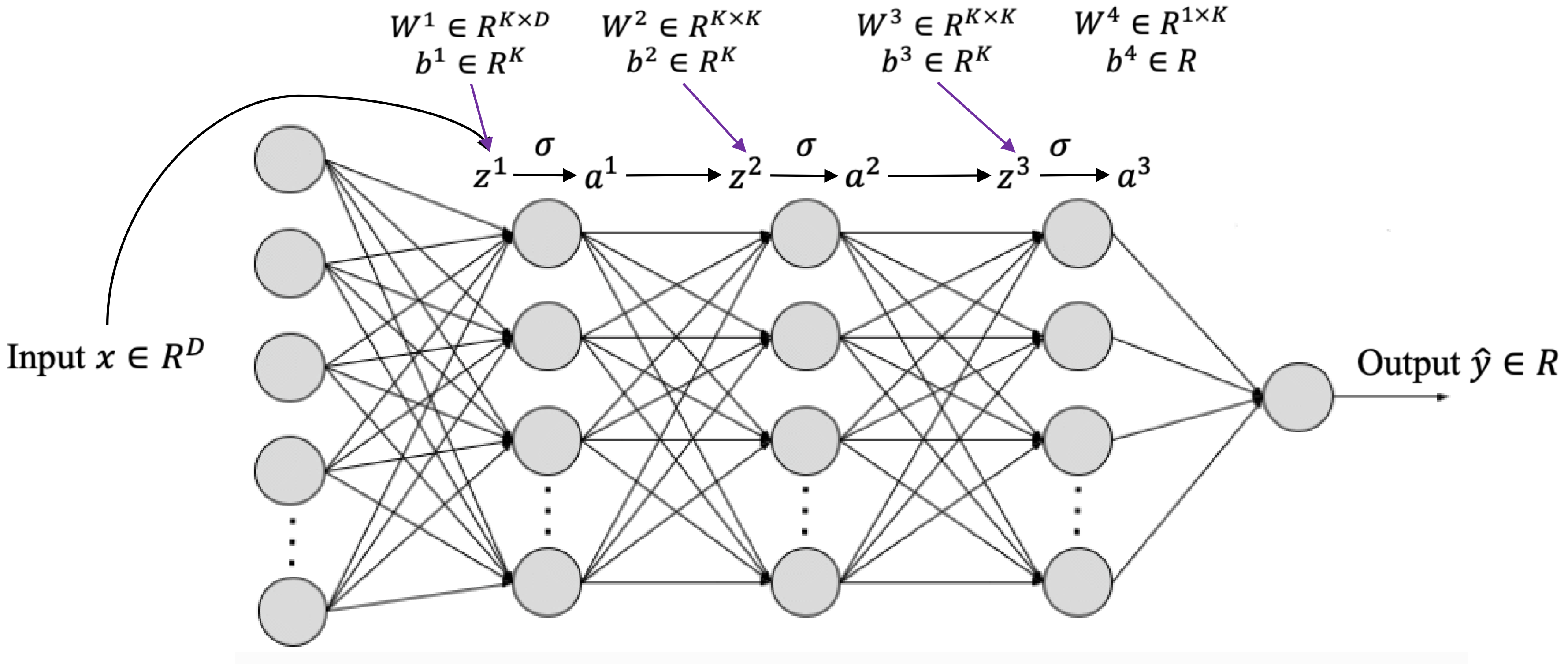
$z^2 = W^2 a^1 + b^2$

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

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

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

Forward Propagation



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

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

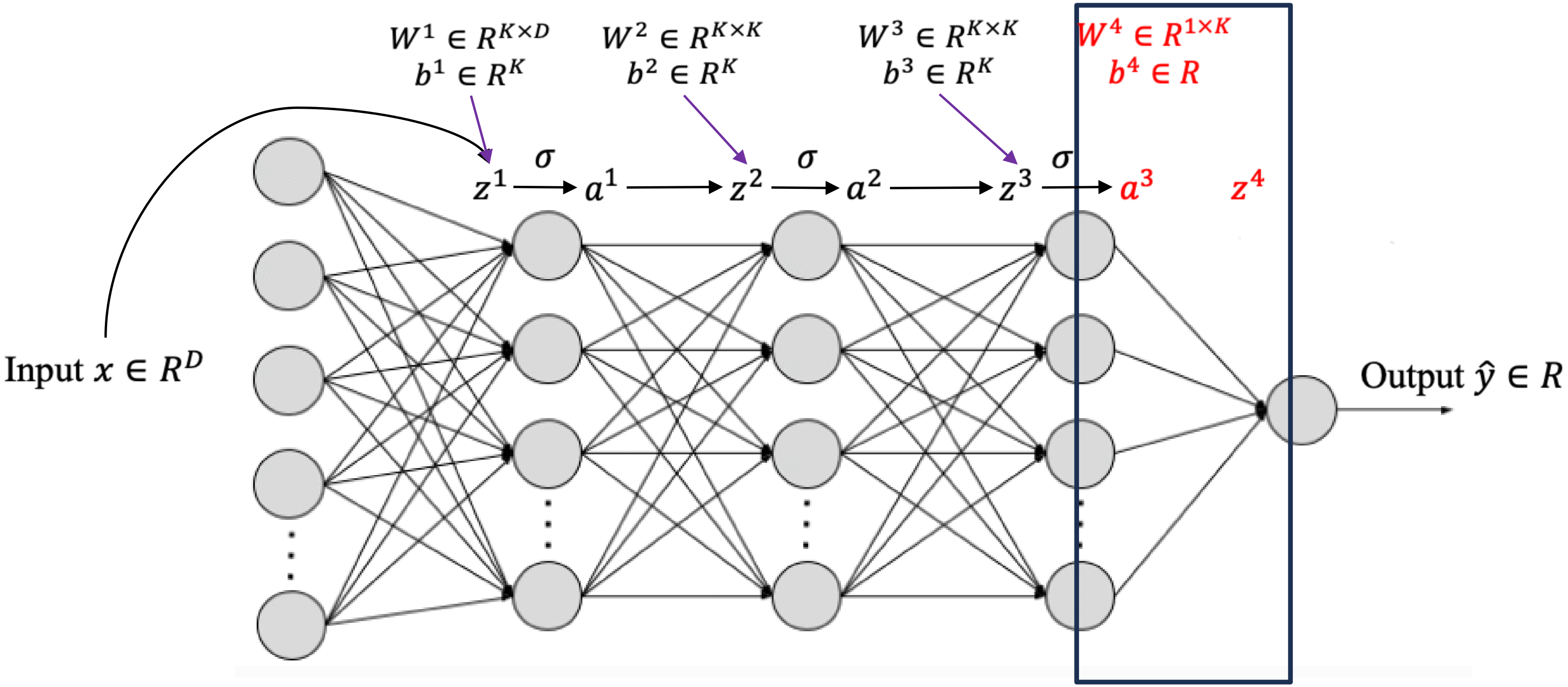
$z^2 = W^2a^1 + b^2$

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

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

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

Forward Propagation



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

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

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

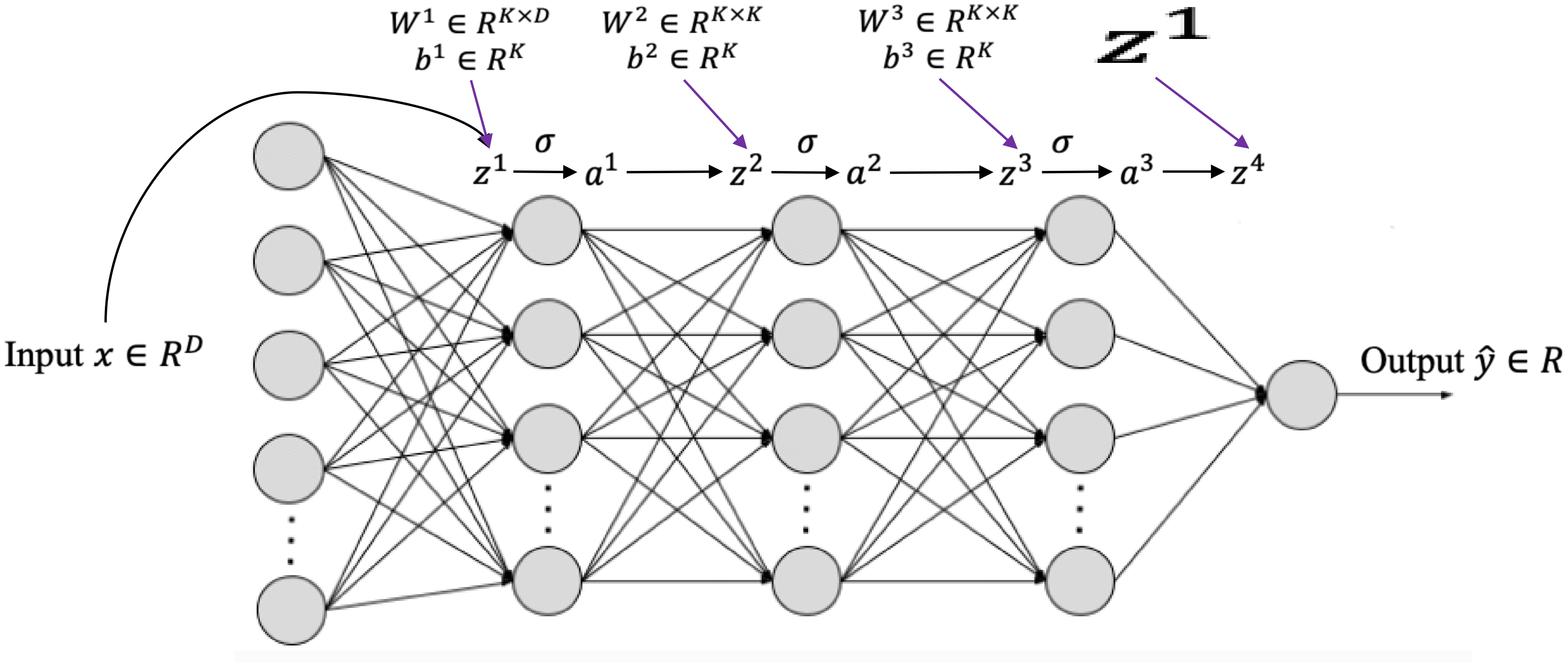
$a^2 = \sigma(z^2)$

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

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

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

Forward Propagation



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

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

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

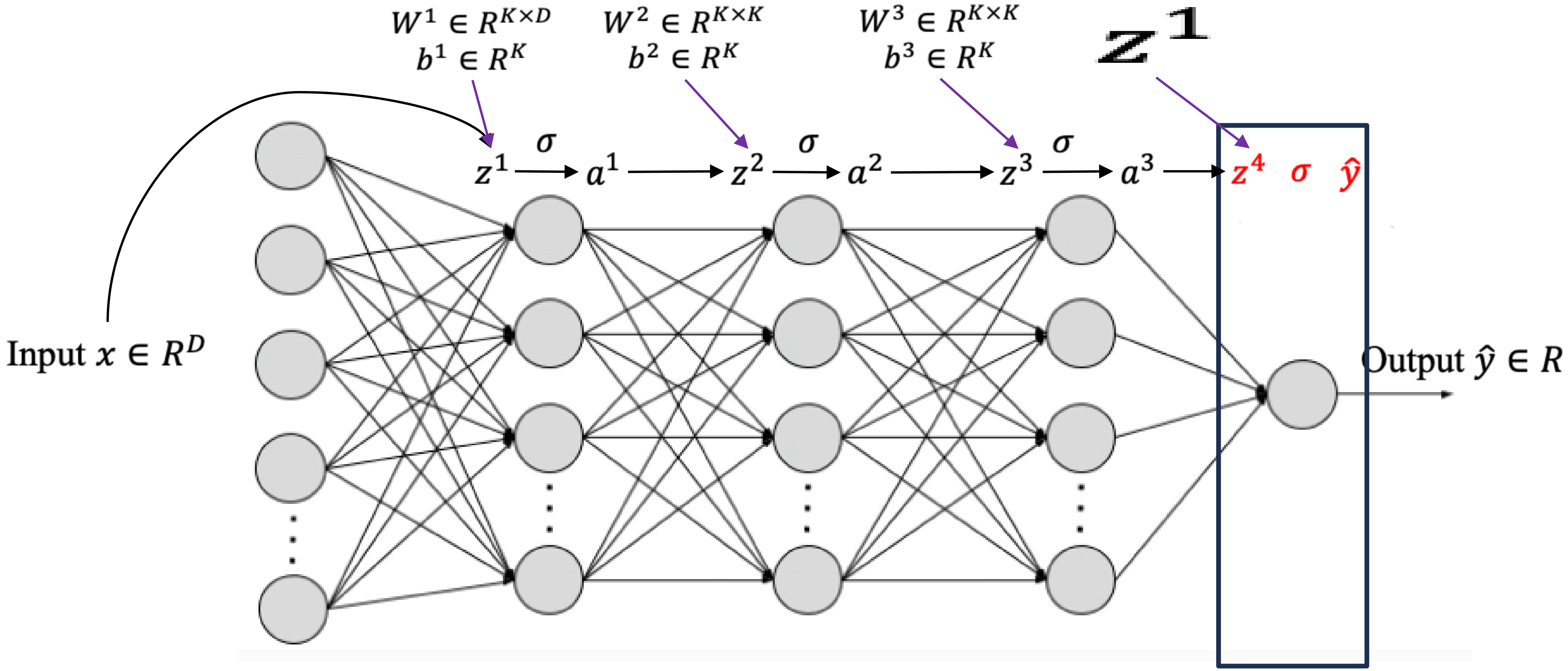
$a^2 = \sigma(z^2)$

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

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

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

Forward Propagation



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

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

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

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

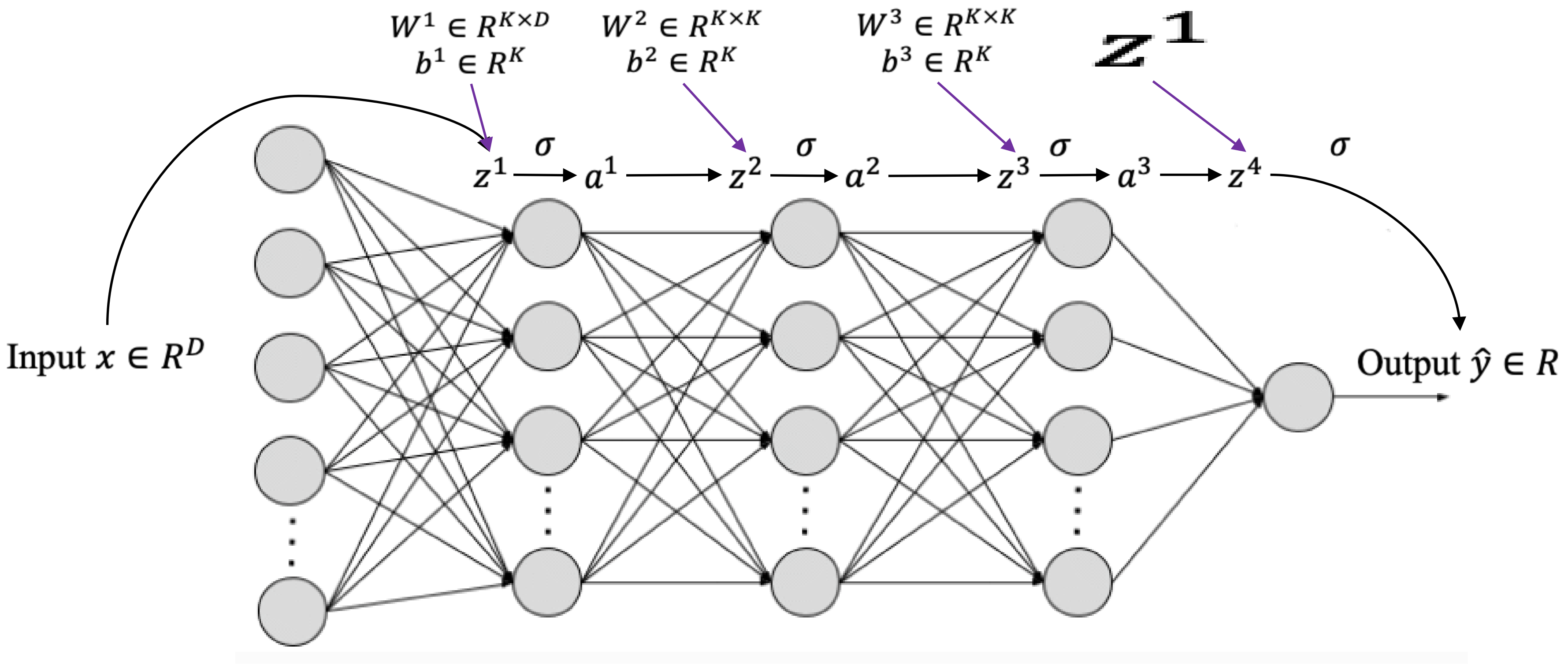
$z^3 = W^3 a^2 + b^3$

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

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

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

Forward Propagation



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

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

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

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

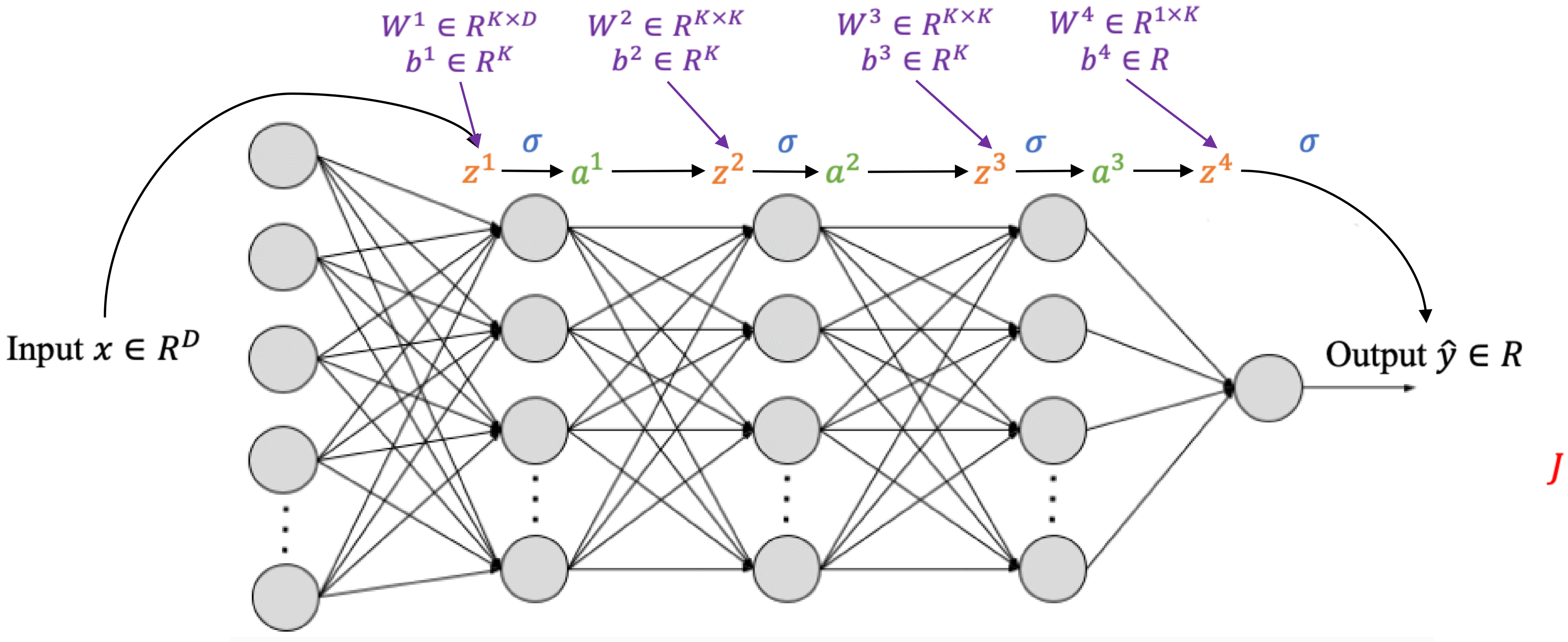
$z^3 = W^3a^2 + b^3$

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

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

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

Forward Propagation



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

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

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

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

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

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

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

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

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

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

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

Parameter Update – Gradient Descent

Gradient Descent

current parameter

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

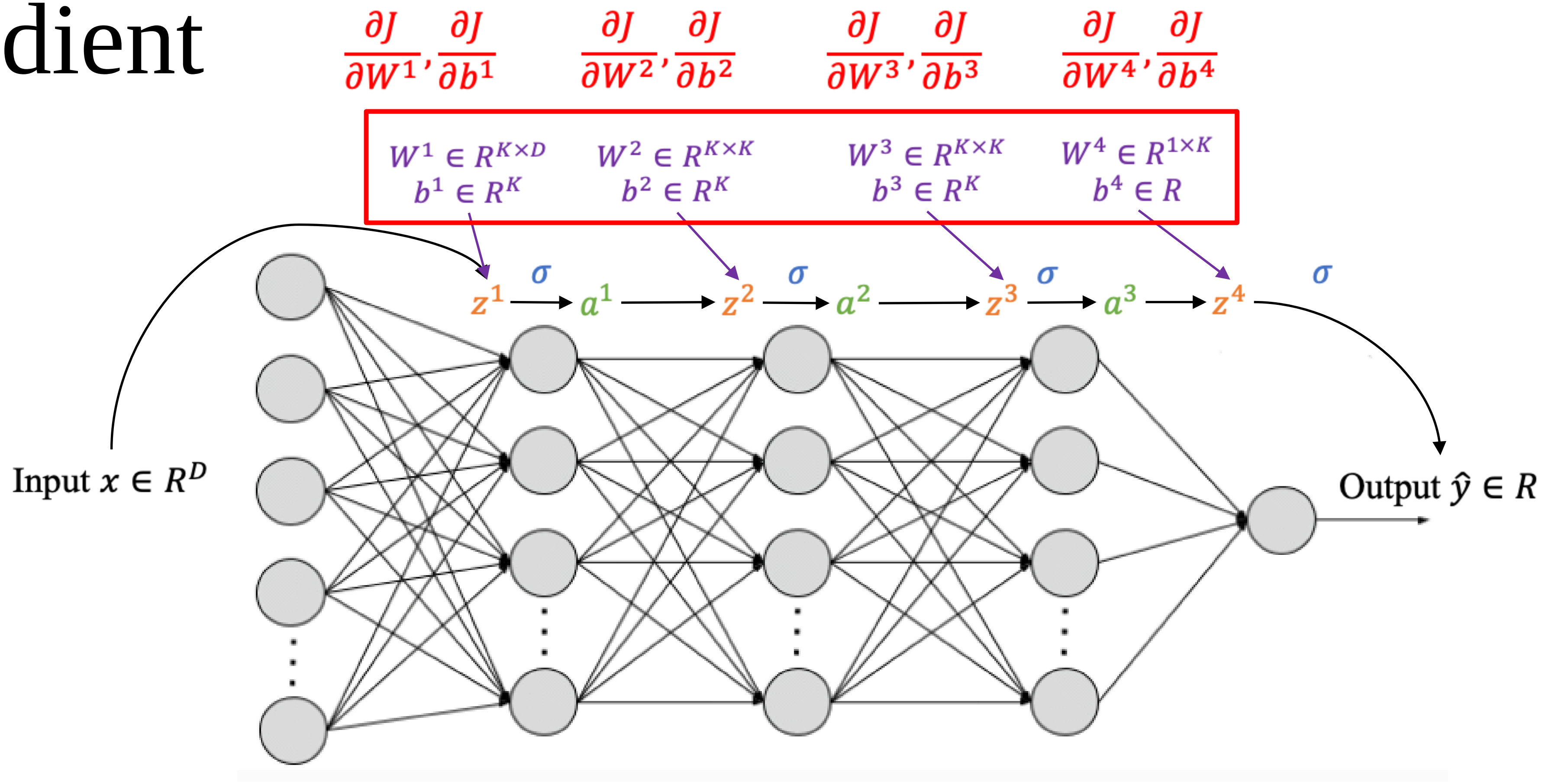
gradient

new parameter

learning rate

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

Gradient



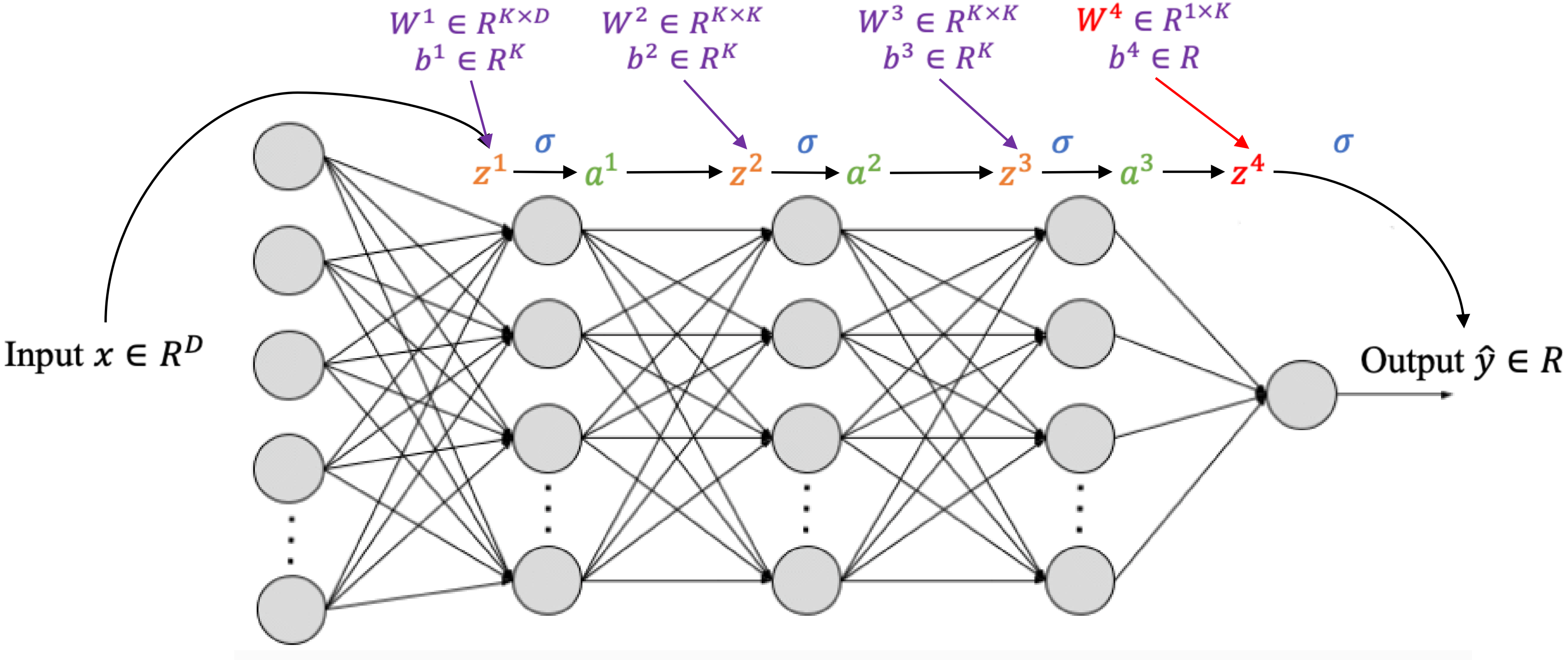
$$J = \frac{1}{2}(\hat{y} - y)^2$$

How? Backward propagation with chain rule!

Backward Propagation

Chain rule

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

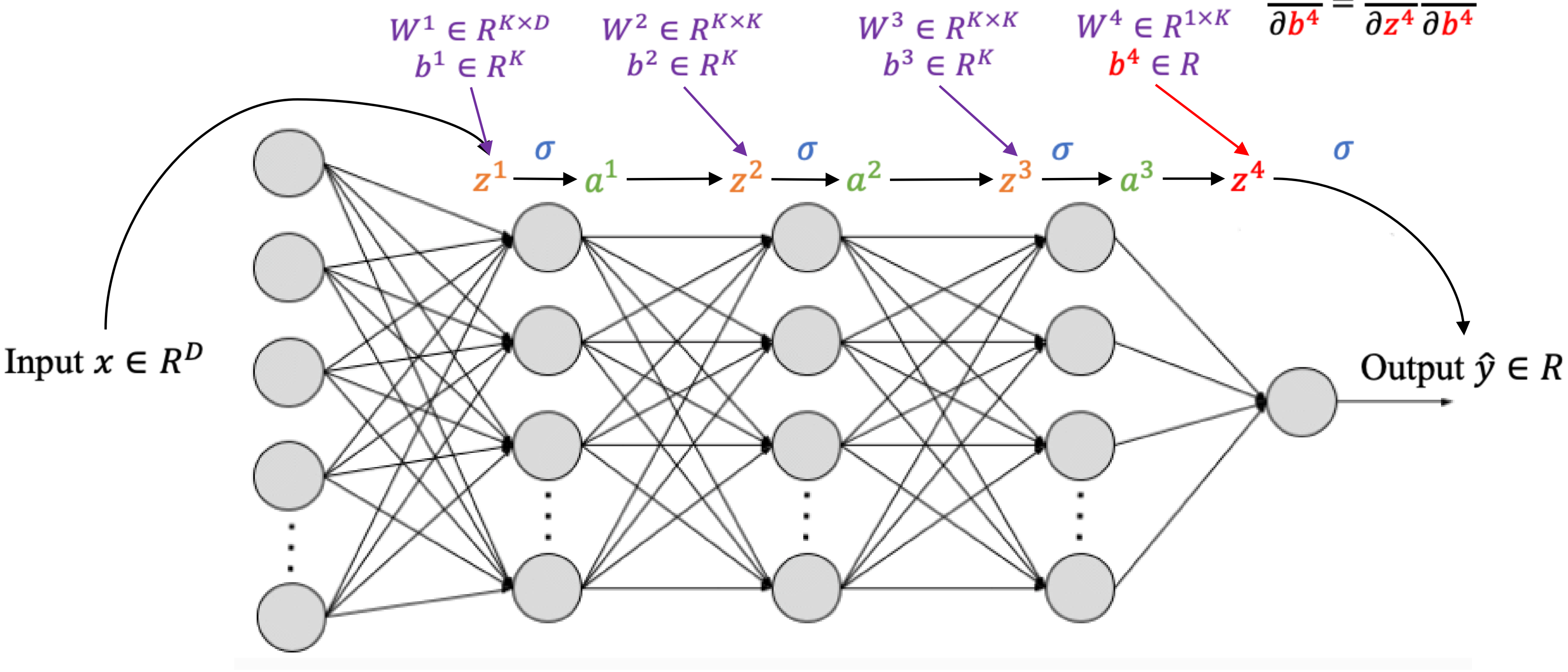


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

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

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



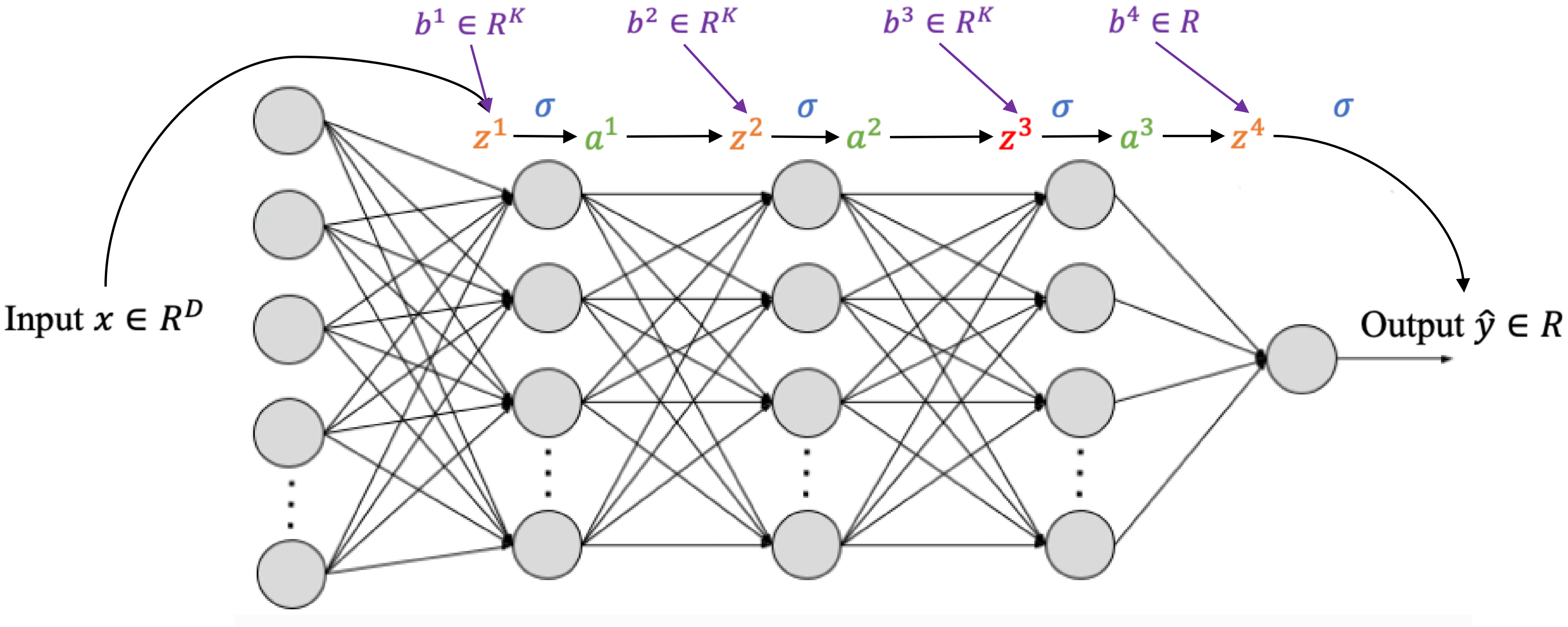
$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

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

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

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

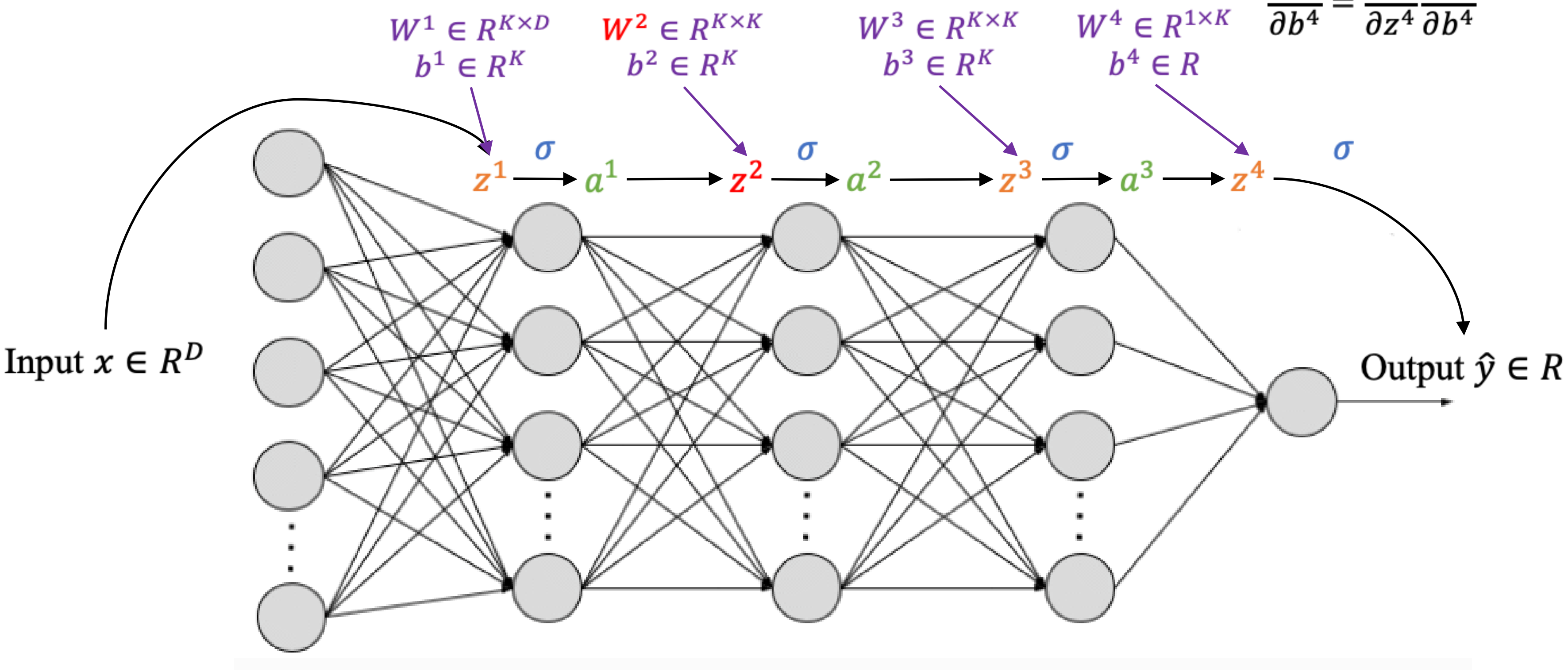


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

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

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

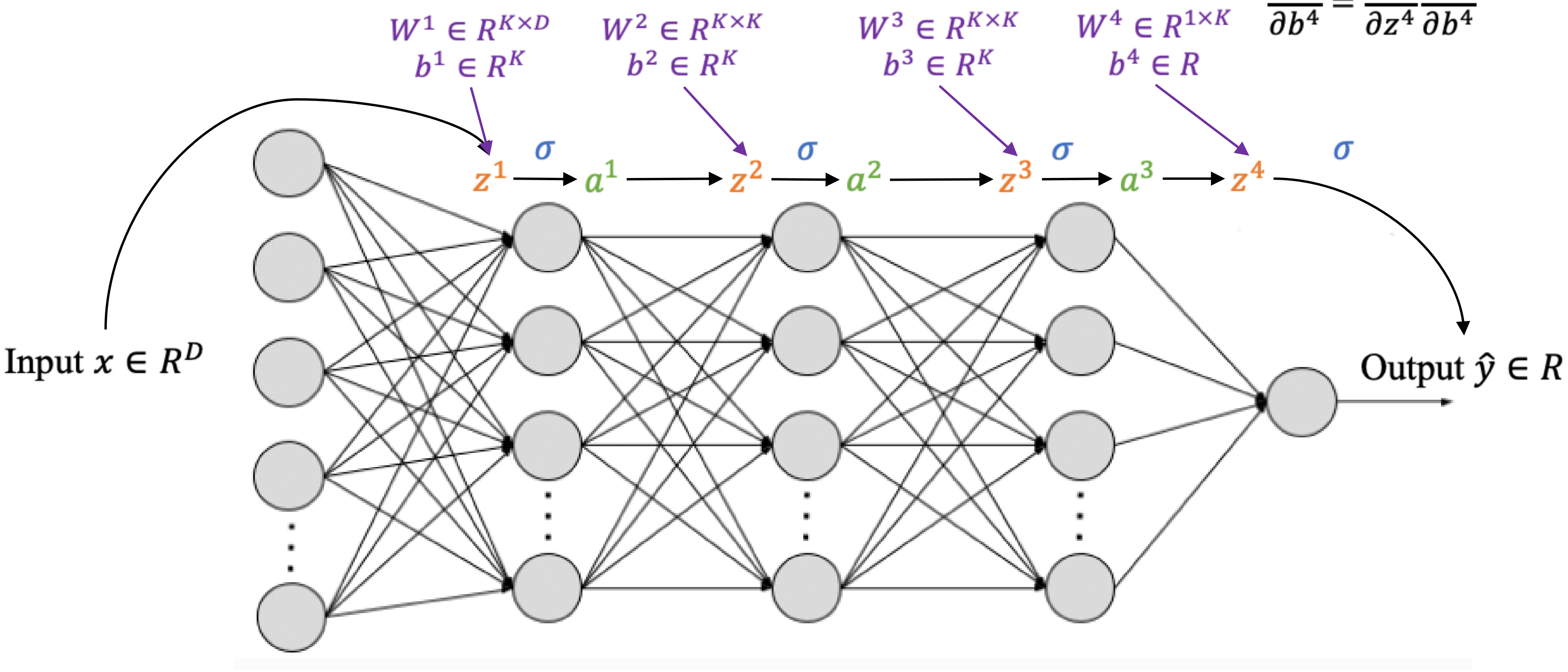


$$J = \frac{1}{2}(\hat{y} - y)^2$$

Chain rule

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

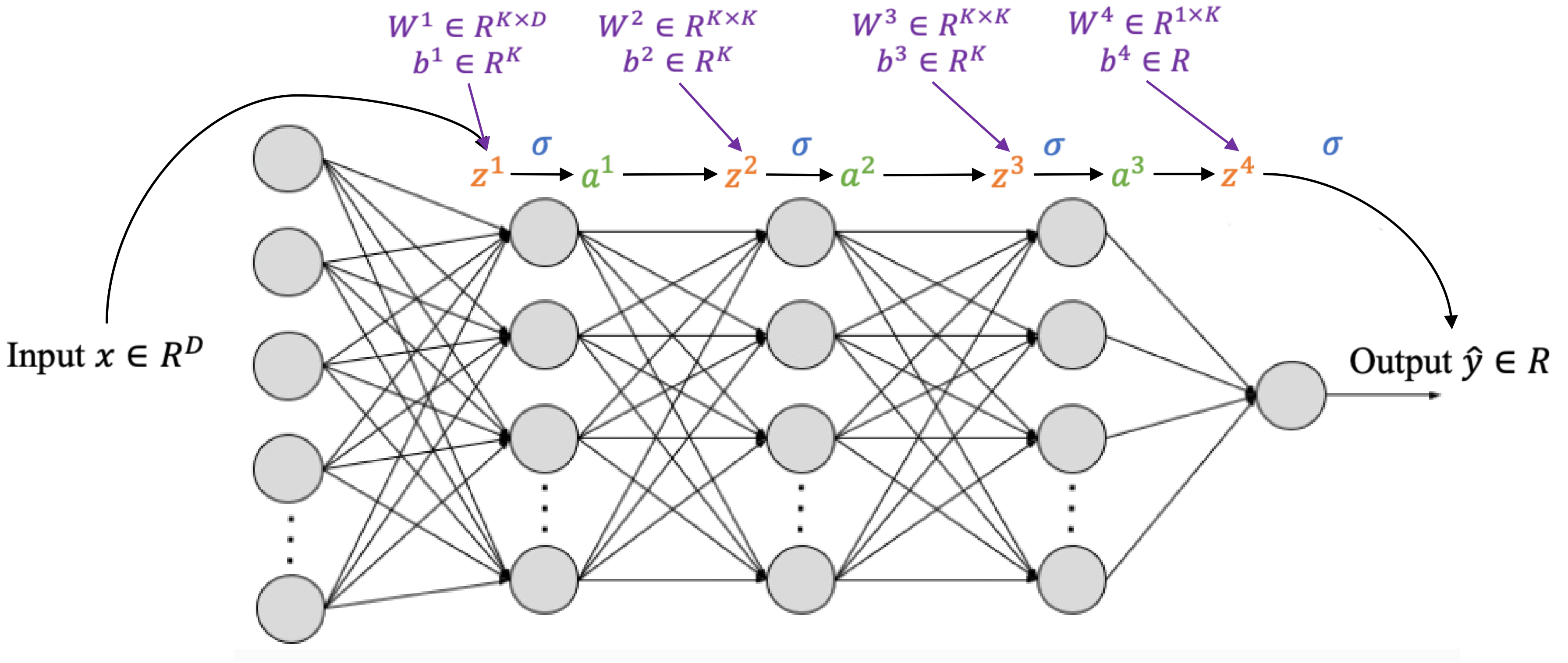
$$\frac{\partial J}{\partial b^4} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial b^4}$$



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

Chain rule

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



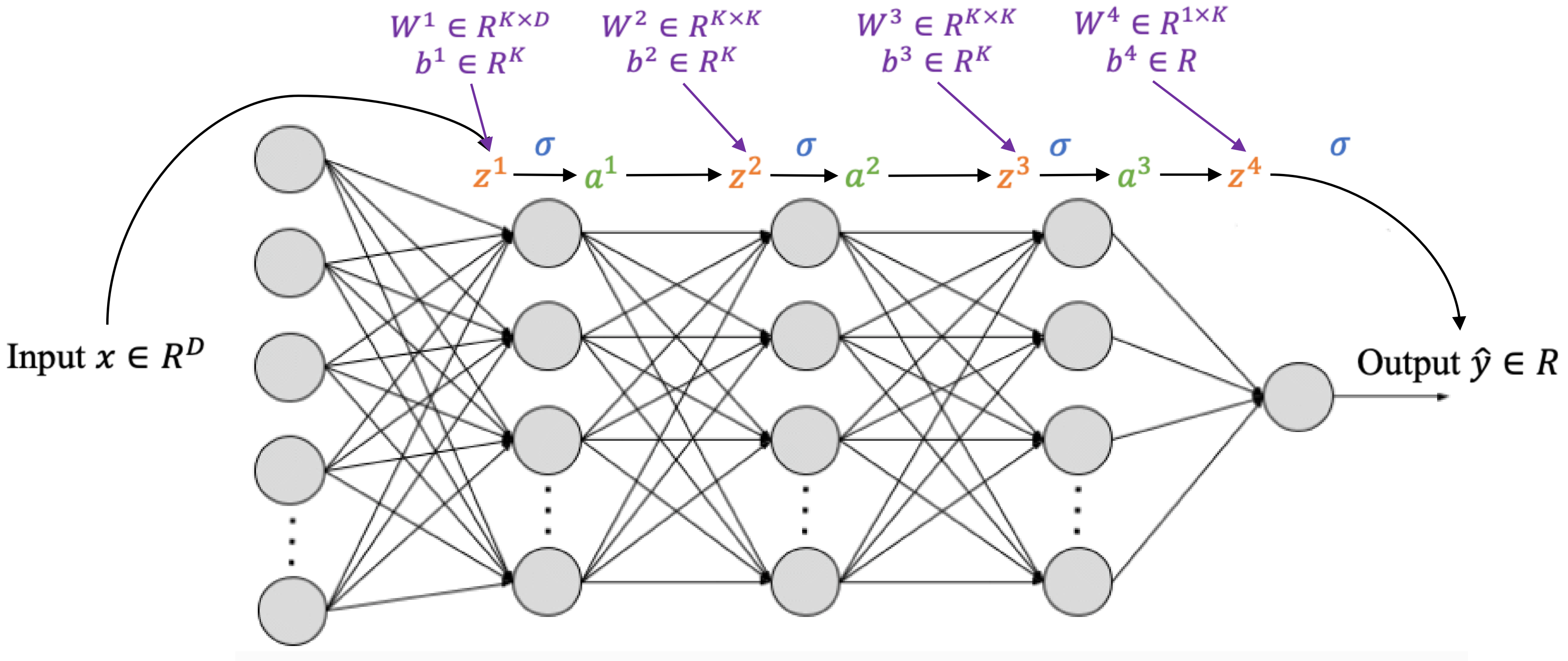
$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

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



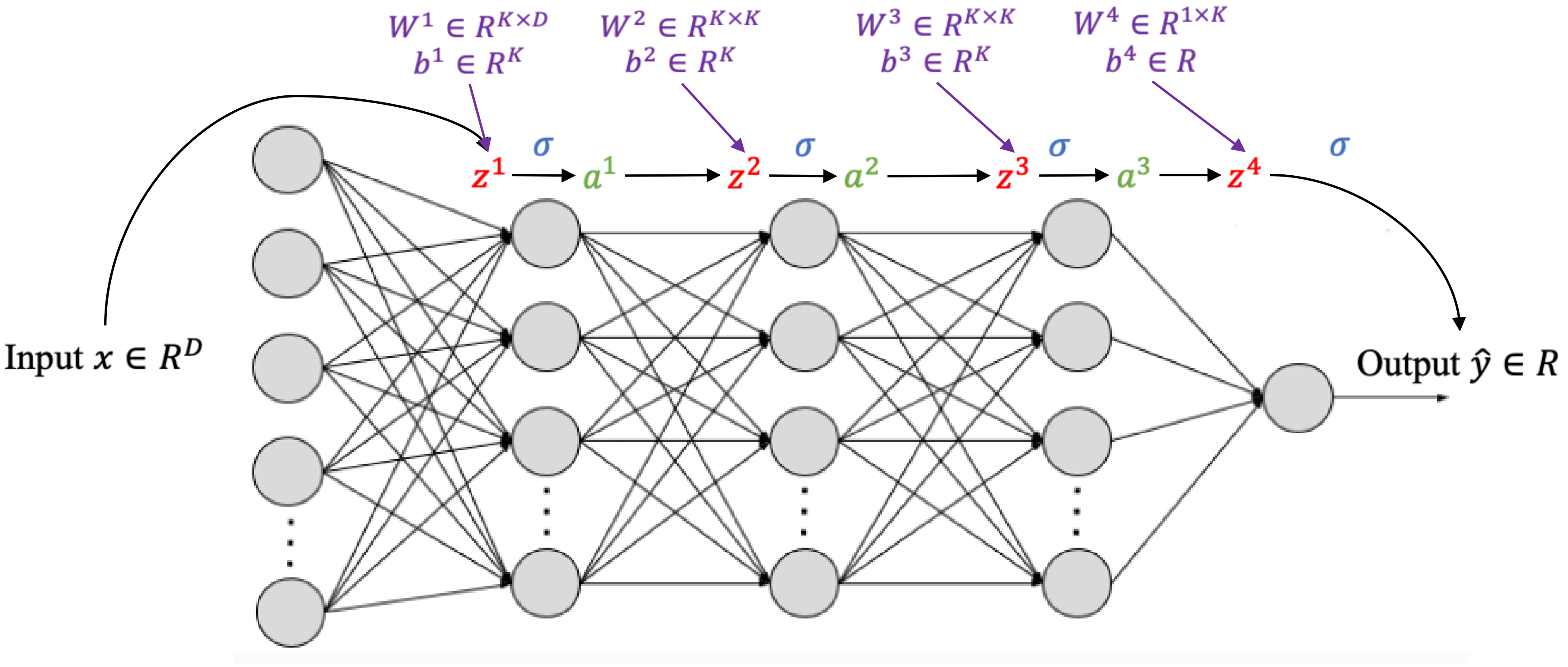
$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

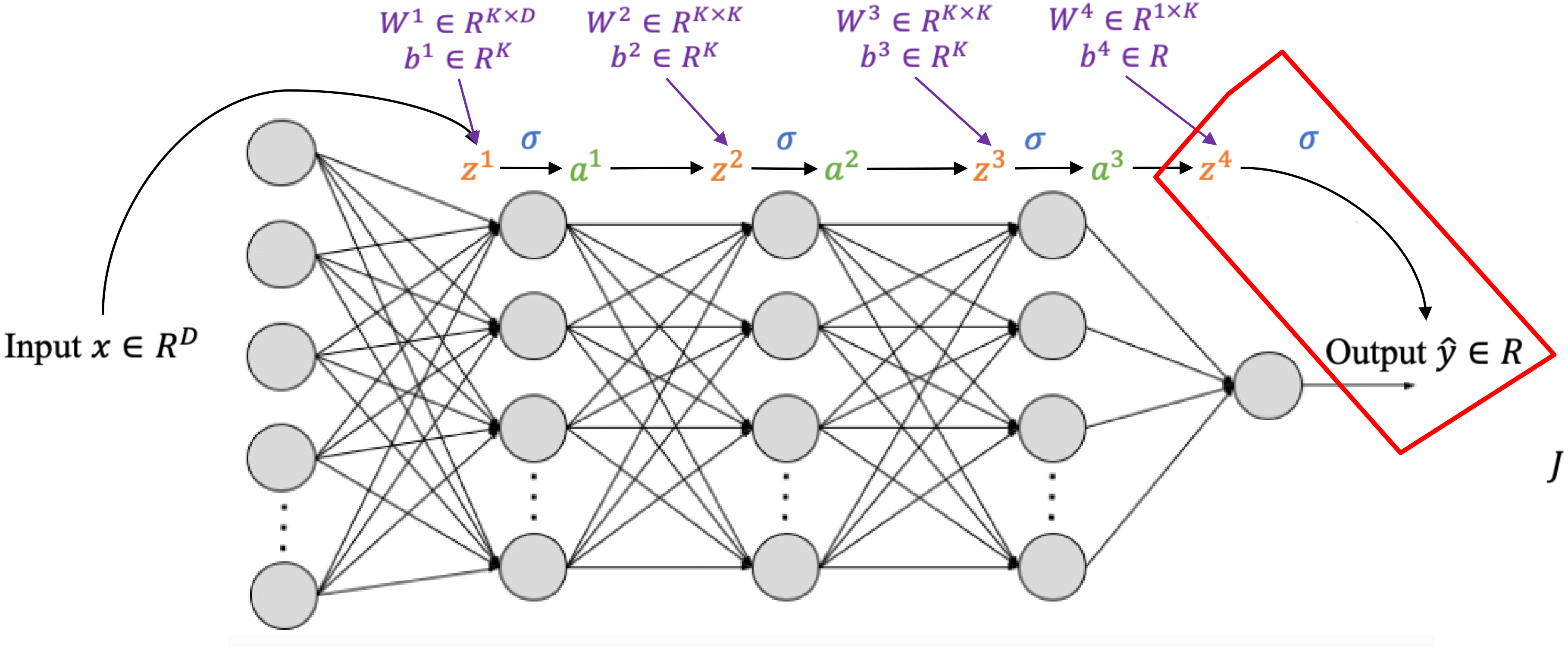
$$\frac{\partial J}{\partial W^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial W^l} \qquad \frac{\partial J}{\partial b^l} = \boxed{\frac{\partial J}{\partial z^l}} \frac{\partial z^l}{\partial b^l}$$



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

Chain rule

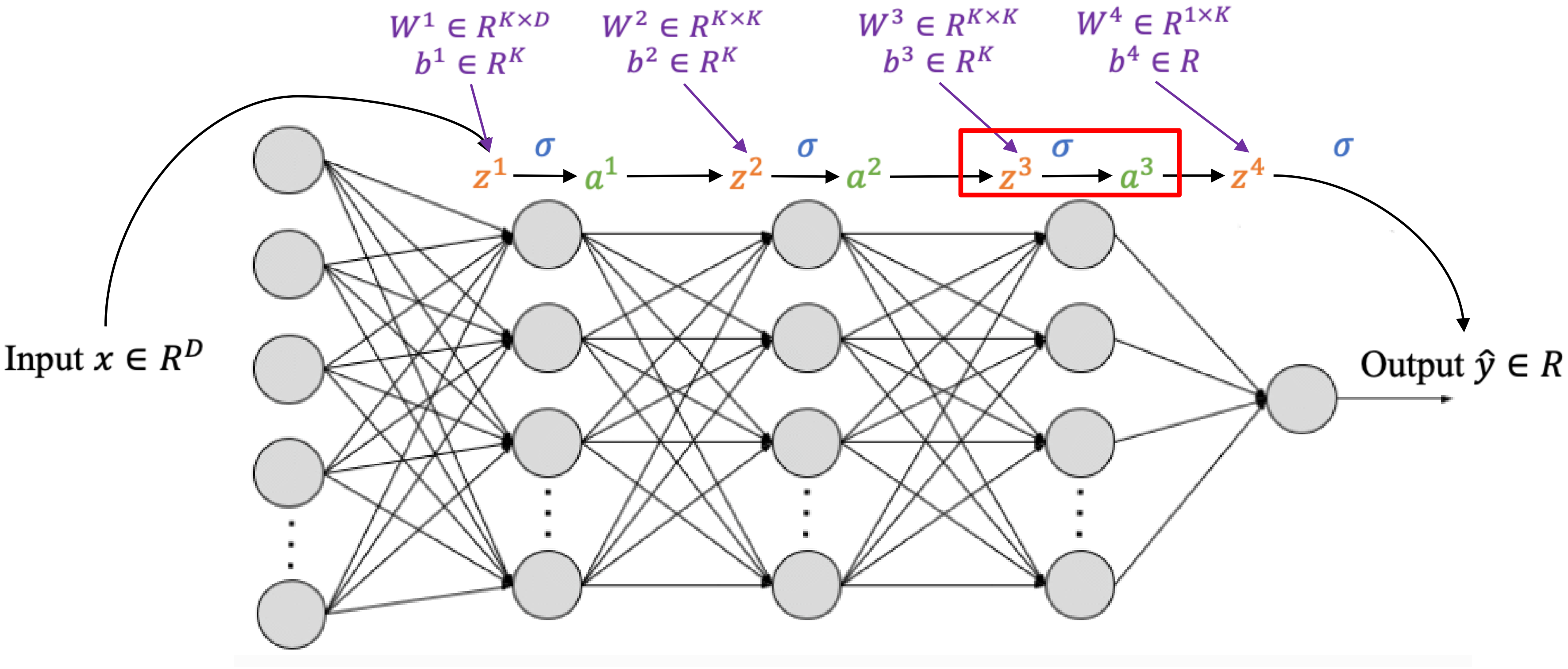
$$\frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^4}$$



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

Chain rule

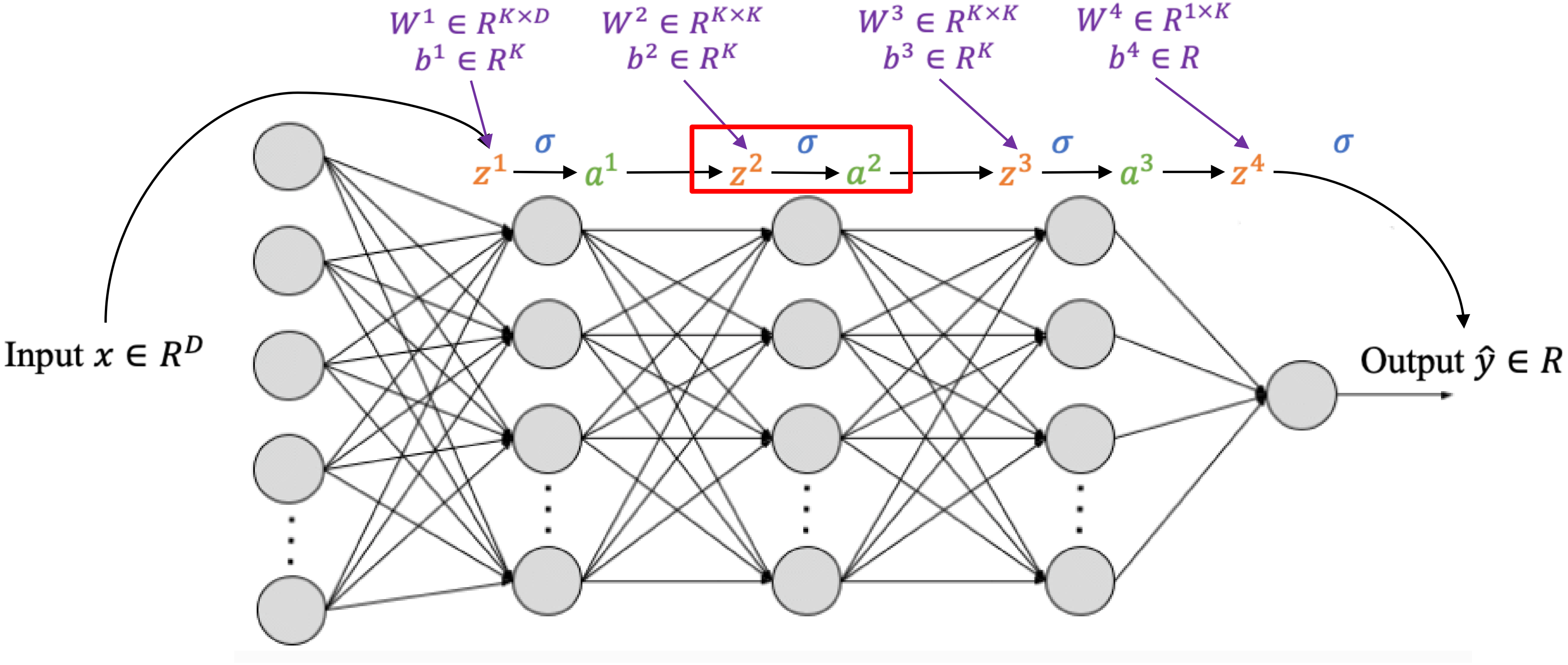
$$\frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial z^3} \qquad \frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^4}$$



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

Chain rule

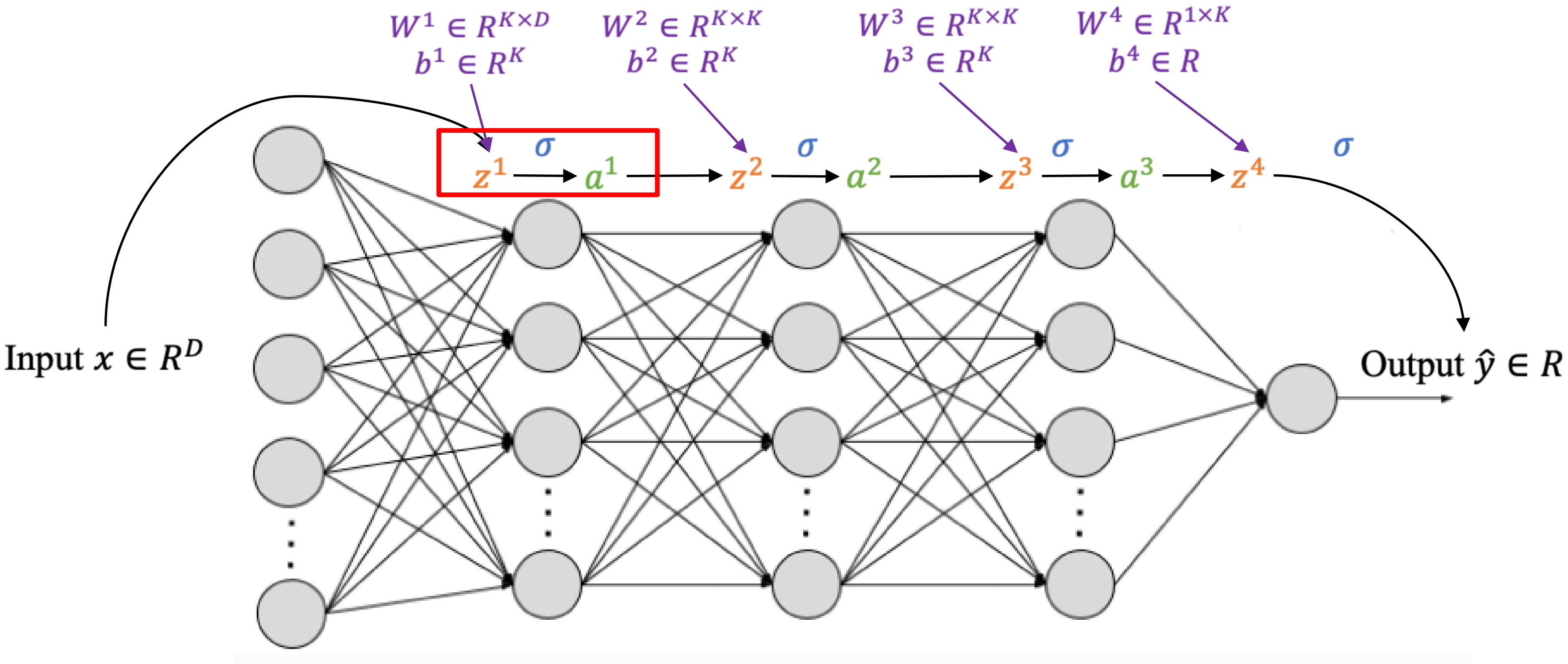
$$\frac{\partial J}{\partial z^2} = \frac{\partial J}{\partial a^2}\frac{\partial a^2}{\partial z^2} \qquad \frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial z^3} \qquad \frac{\partial J}{\partial z^4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial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



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

Chain rule

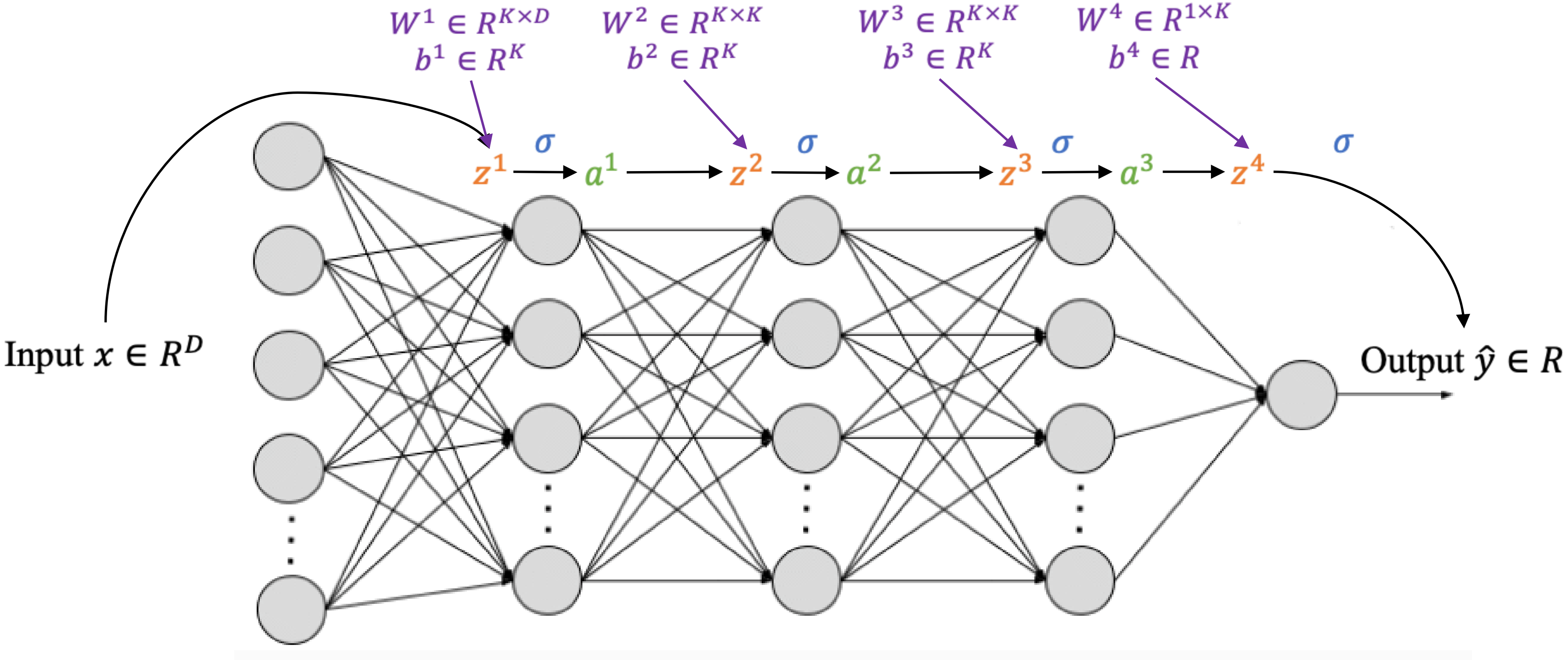
$$\frac{\partial J}{\partial z^1} = \frac{\partial J}{\partial a^1}\frac{\partial a^1}{\partial z^1} \quad \frac{\partial J}{\partial z^2} = \frac{\partial J}{\partial a^2}\frac{\partial a^2}{\partial z^2} \quad \frac{\partial J}{\partial z^3} = \frac{\partial J}{\partial a^3}\frac{\partial a^3}{\partial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



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

Chain rule

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



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

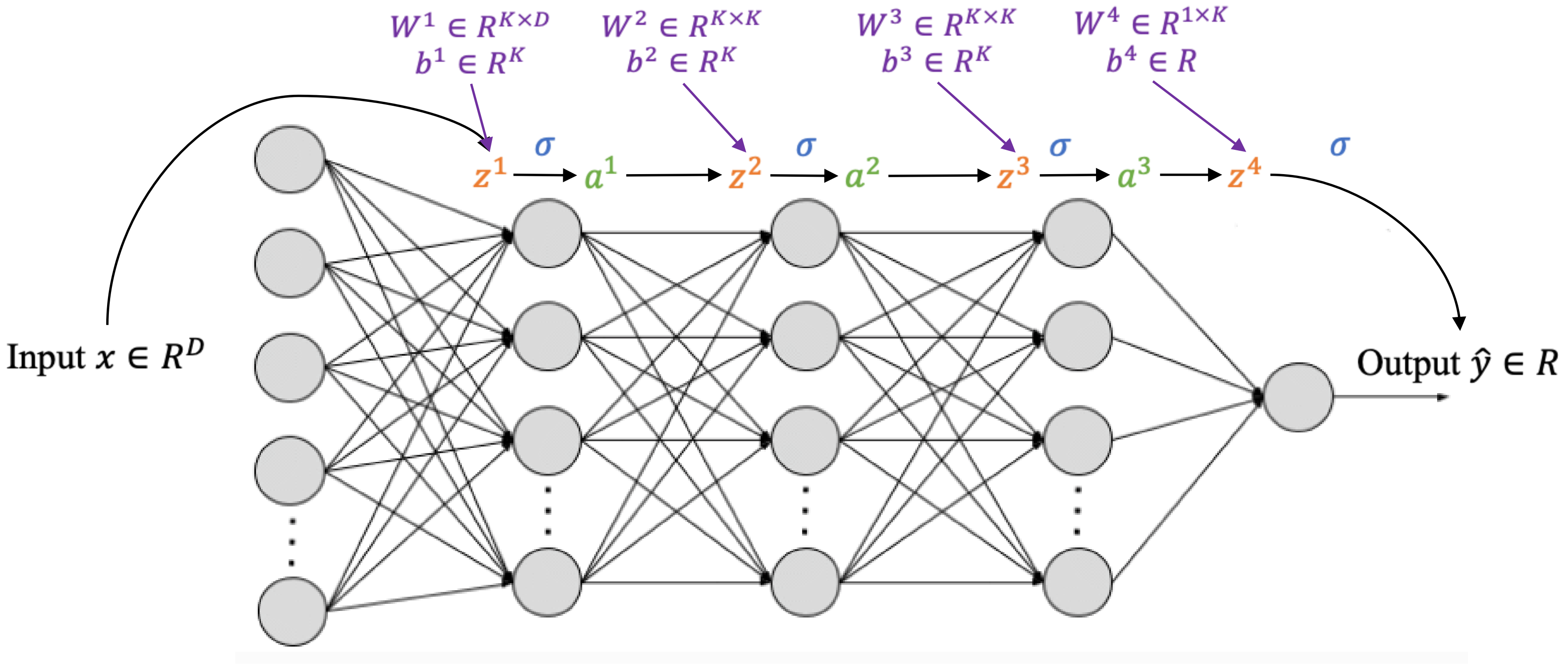
forward propagation:

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

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

Chain rule

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



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

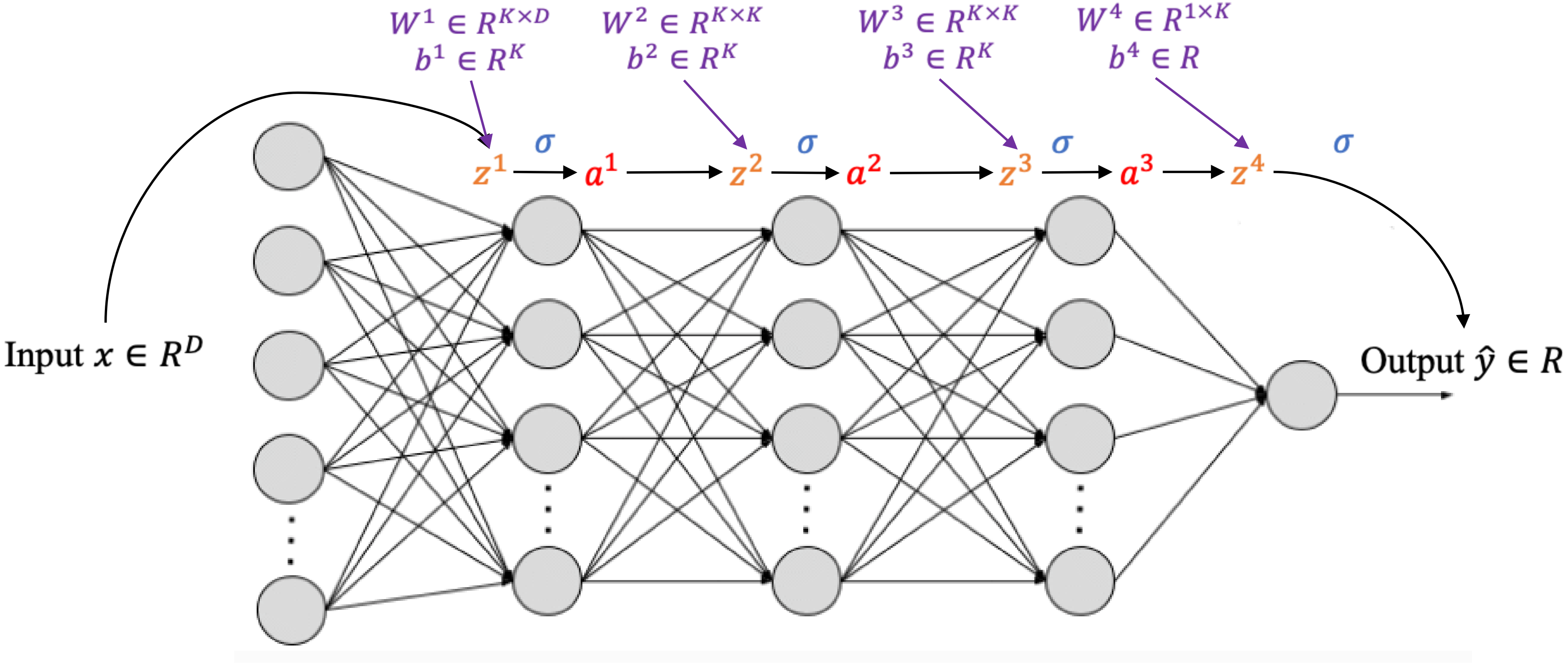
forward propagation:

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

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

Chain rule

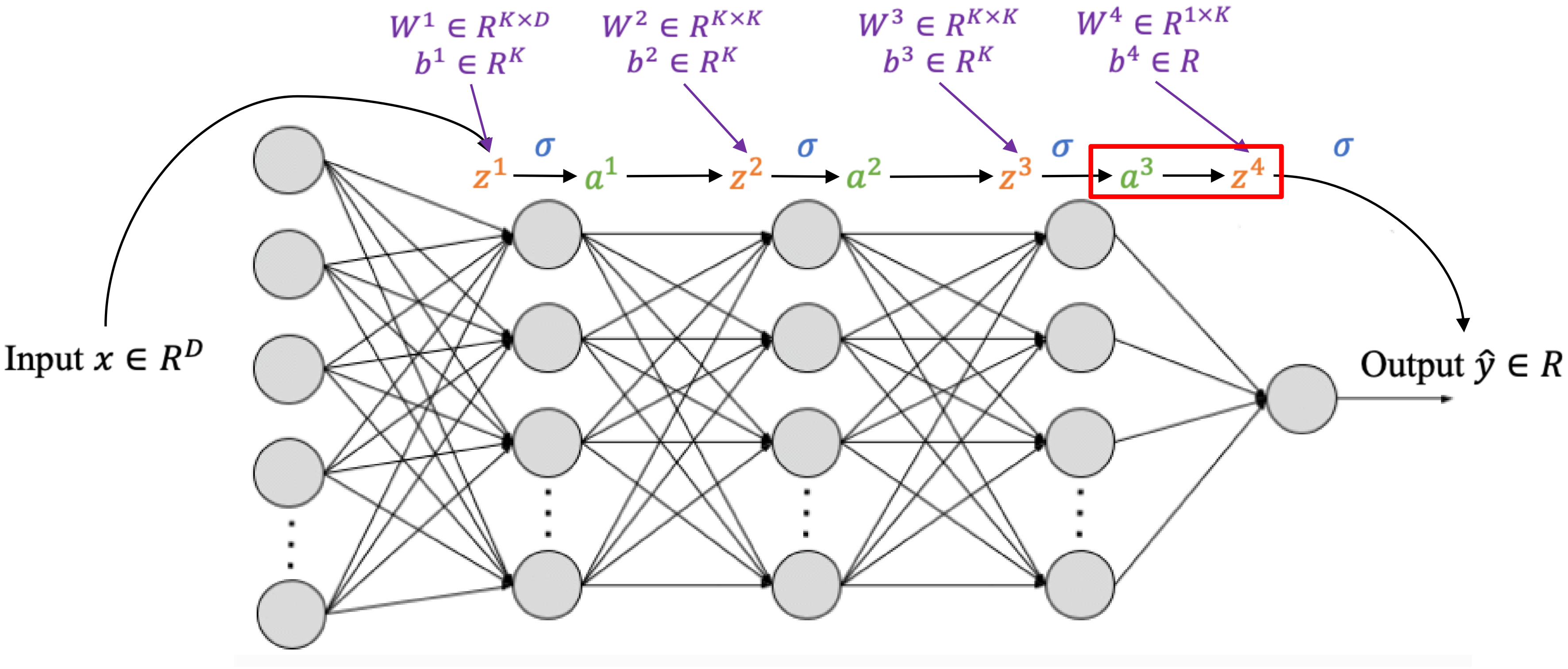
$$\frac{\partial J}{\partial z^l} = \boxed{\frac{\partial J}{\partial a^l}} \frac{\partial a^l}{\partial z^l}$$



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

Chain rule

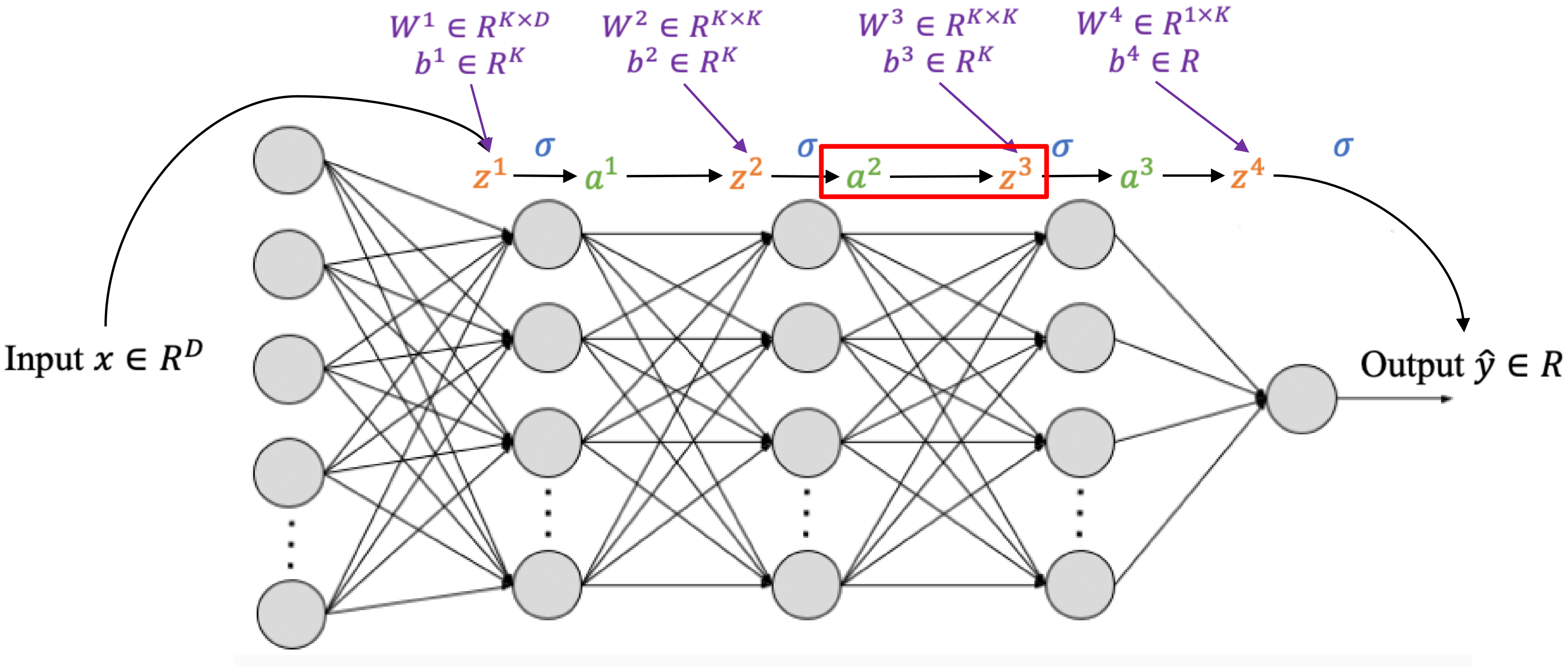
$$\frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4} \frac{\partial z^4}{\partial a^3}$$



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

Chain rule

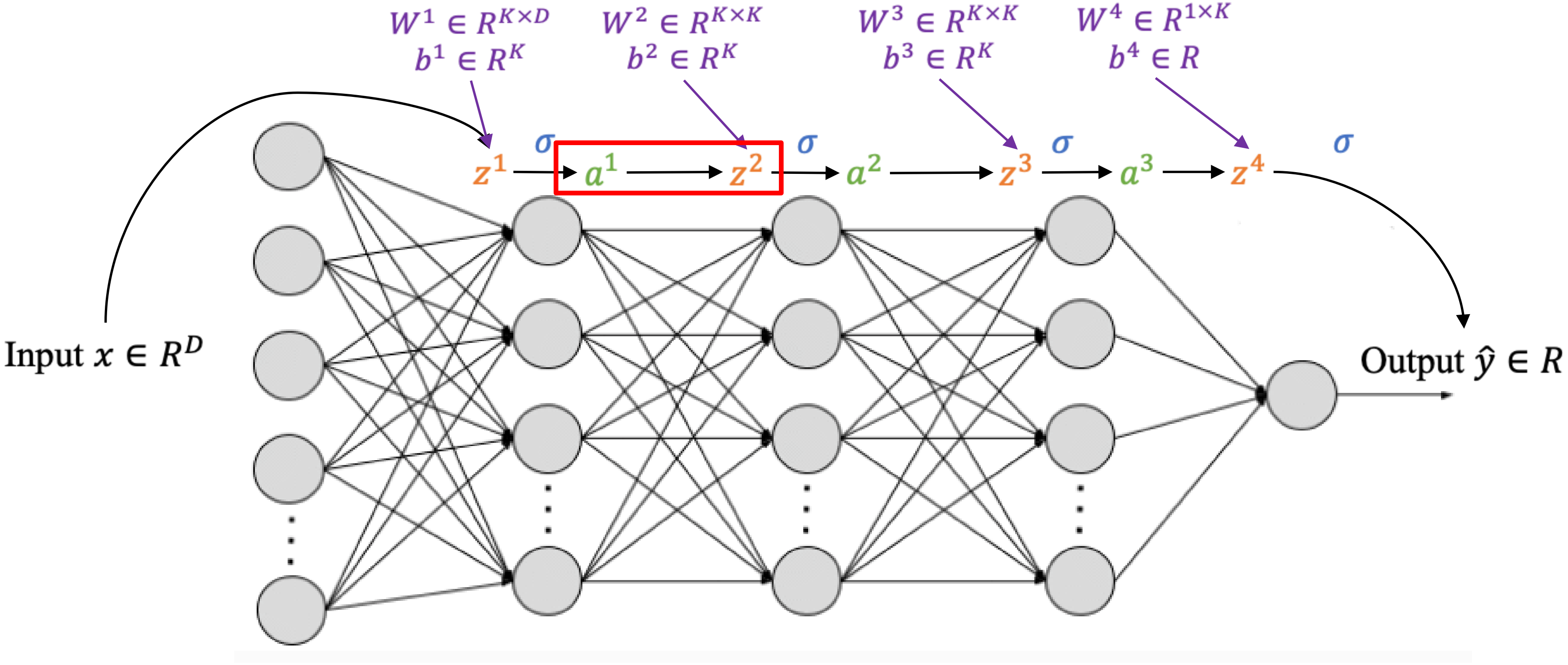
$$\frac{\partial J}{\partial a^2} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial a^2} \qquad \frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial a^3}$$



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

Chain rule

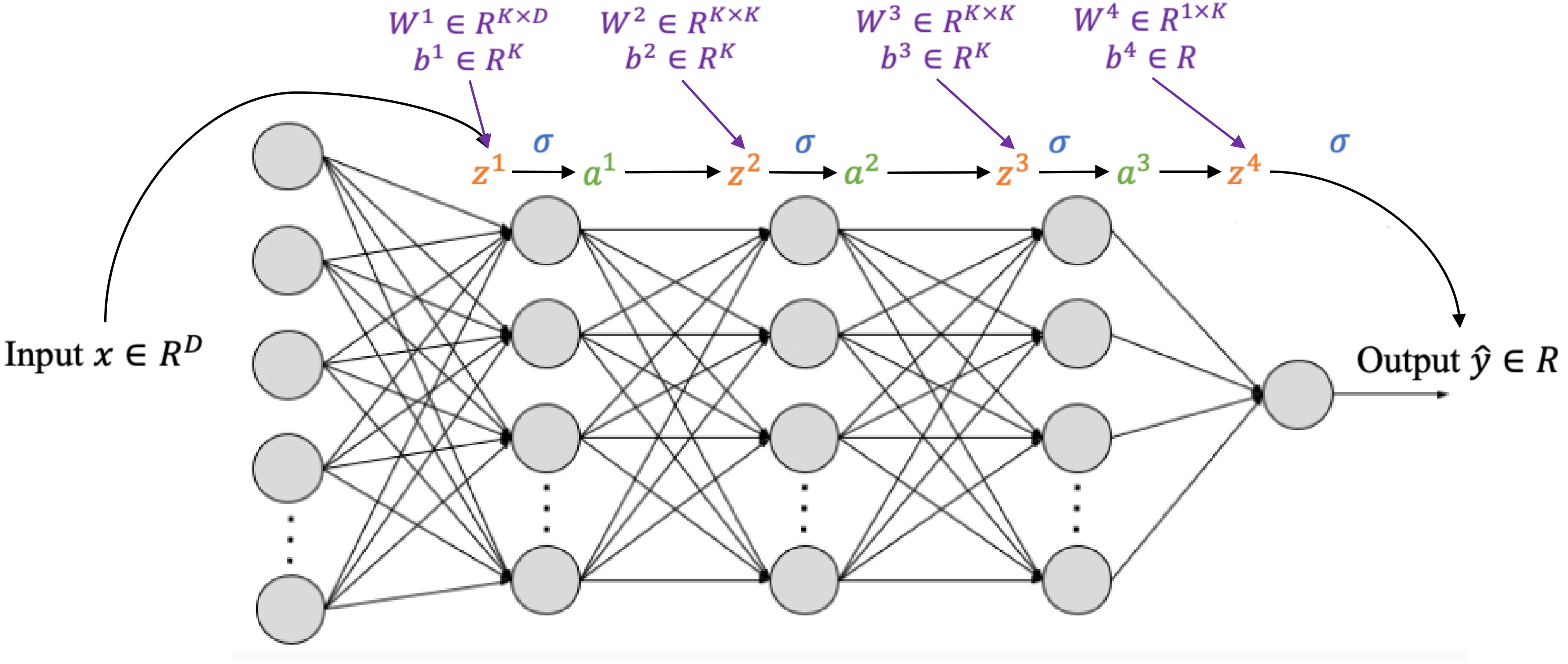
$$\frac{\partial J}{\partial a^1} = \frac{\partial J}{\partial z^2}\frac{\partial z^2}{\partial a^1} \qquad \frac{\partial J}{\partial a^2} = \frac{\partial J}{\partial z^3}\frac{\partial z^3}{\partial a^2} \qquad \frac{\partial J}{\partial a^3} = \frac{\partial J}{\partial z^4}\frac{\partial z^4}{\partial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



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

Chain rule

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



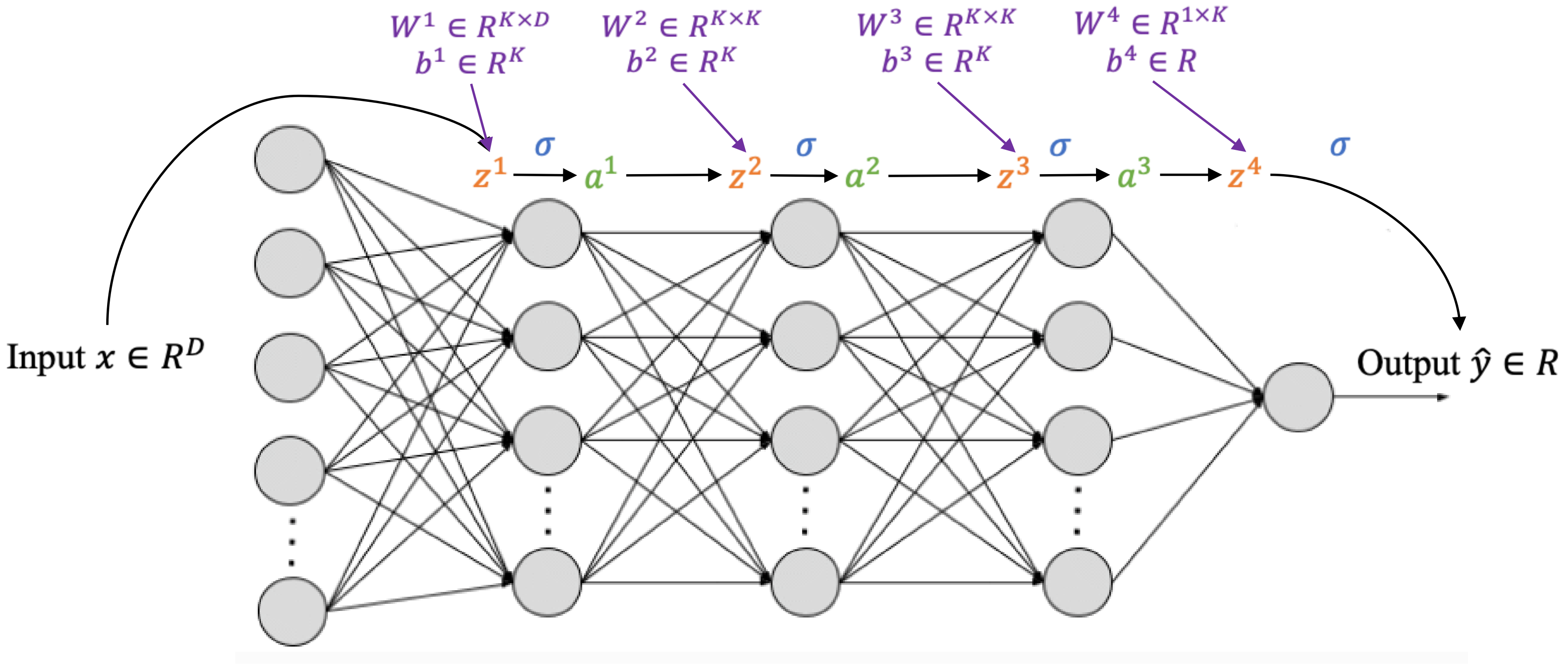
$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

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



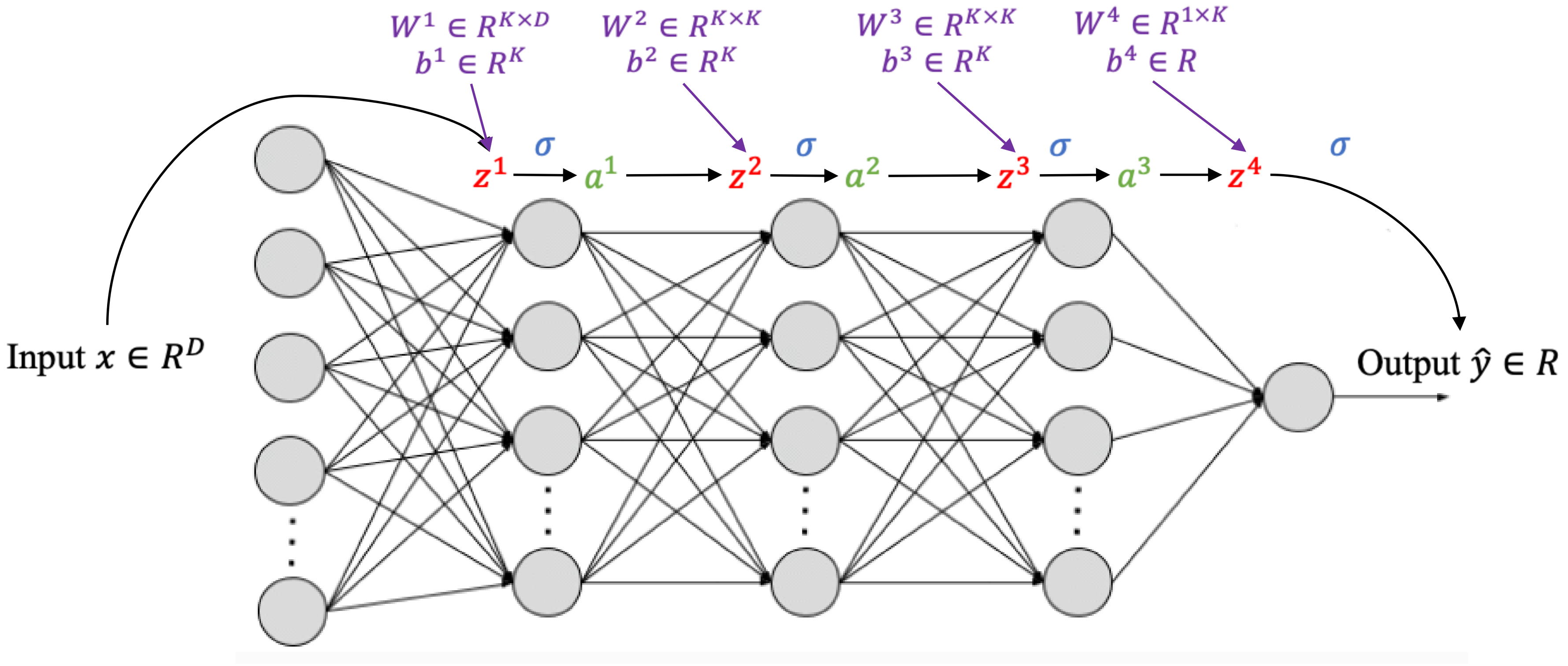
$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

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

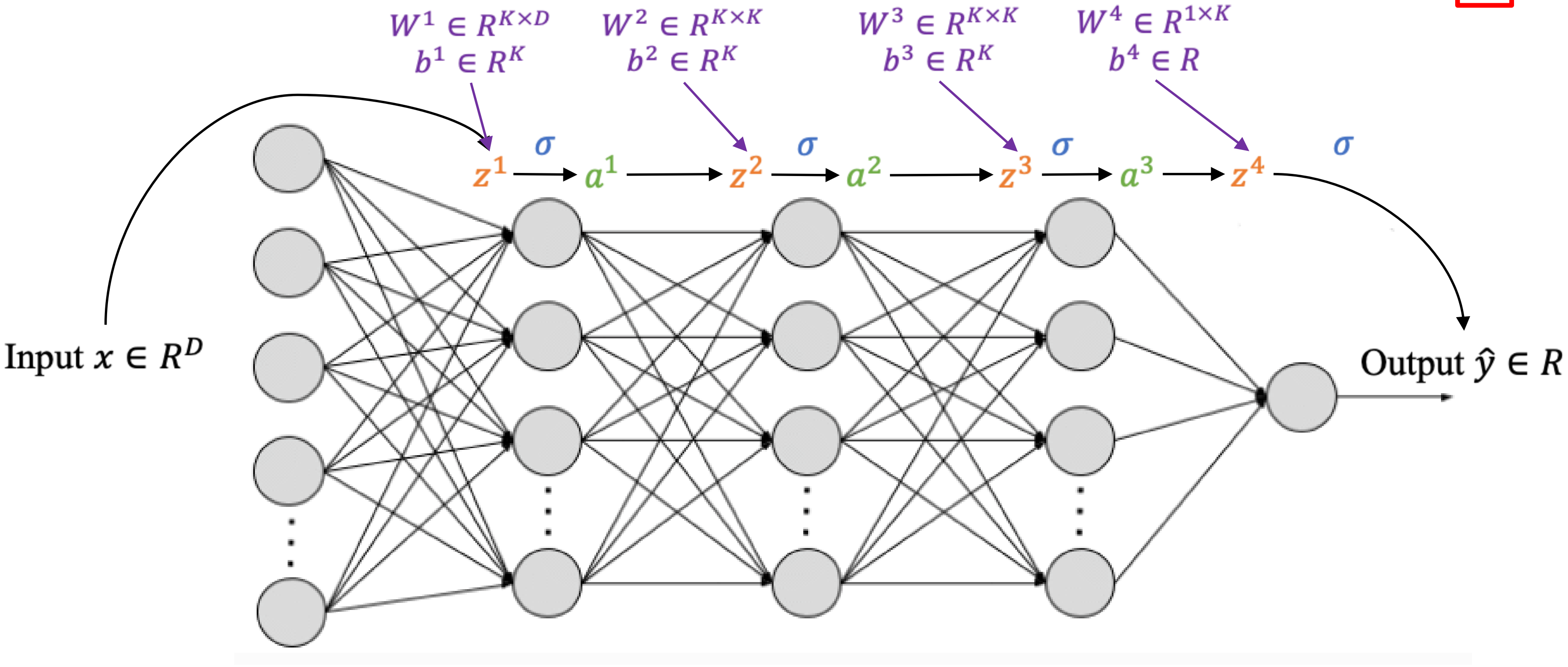


$$J = \frac{1}{2}(\hat{y} - y)^2$$

We come back!

Backward Propagation

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



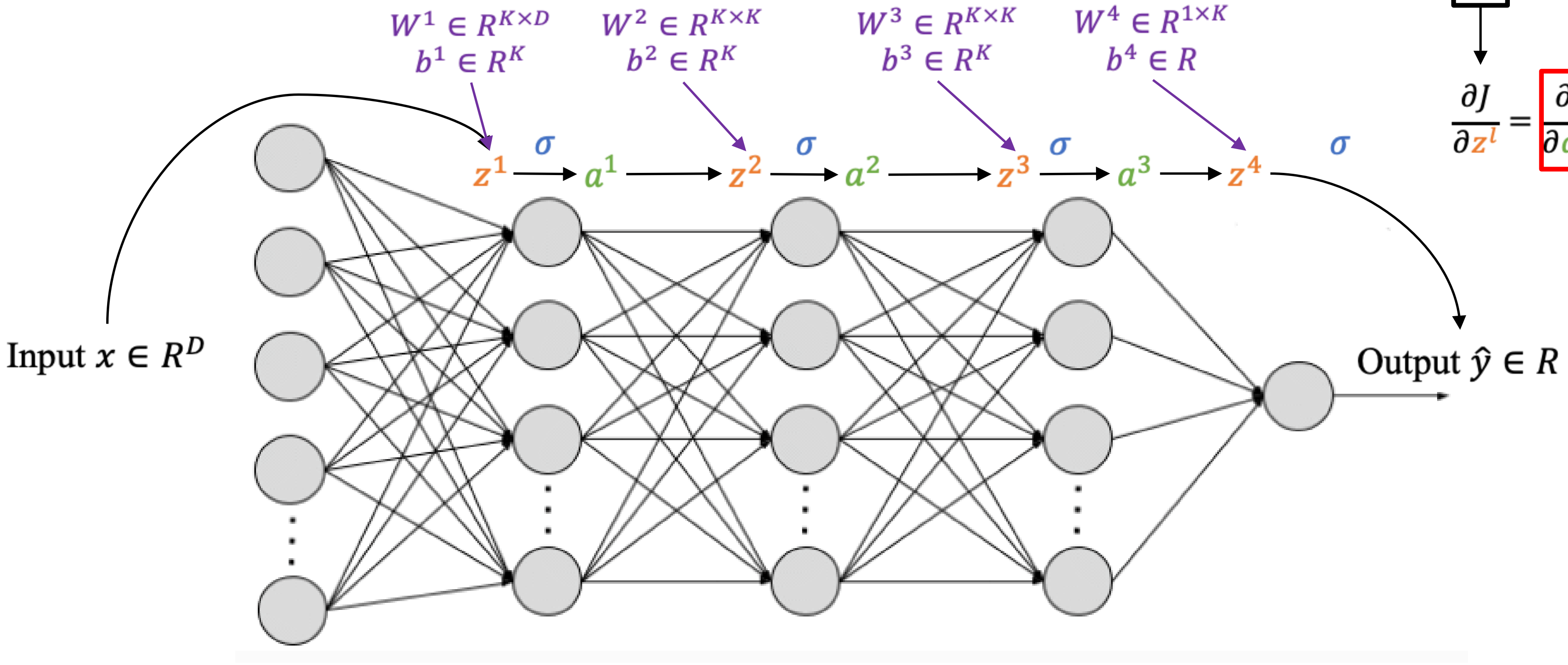
$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

Backward Propagation

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

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



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

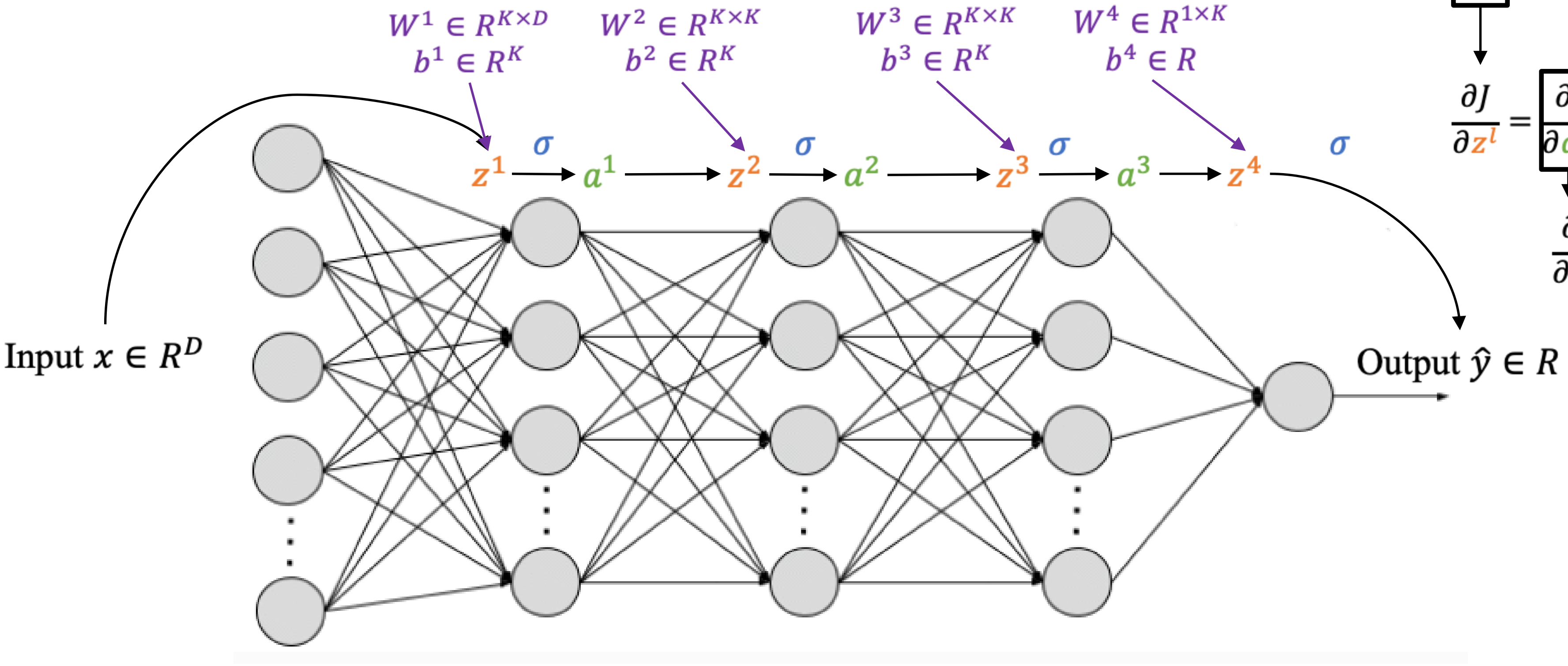
$$\frac{\partial a^l}{\partial z^l} = a^l(1 - a^l)$$

Backward Propagation

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

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

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



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

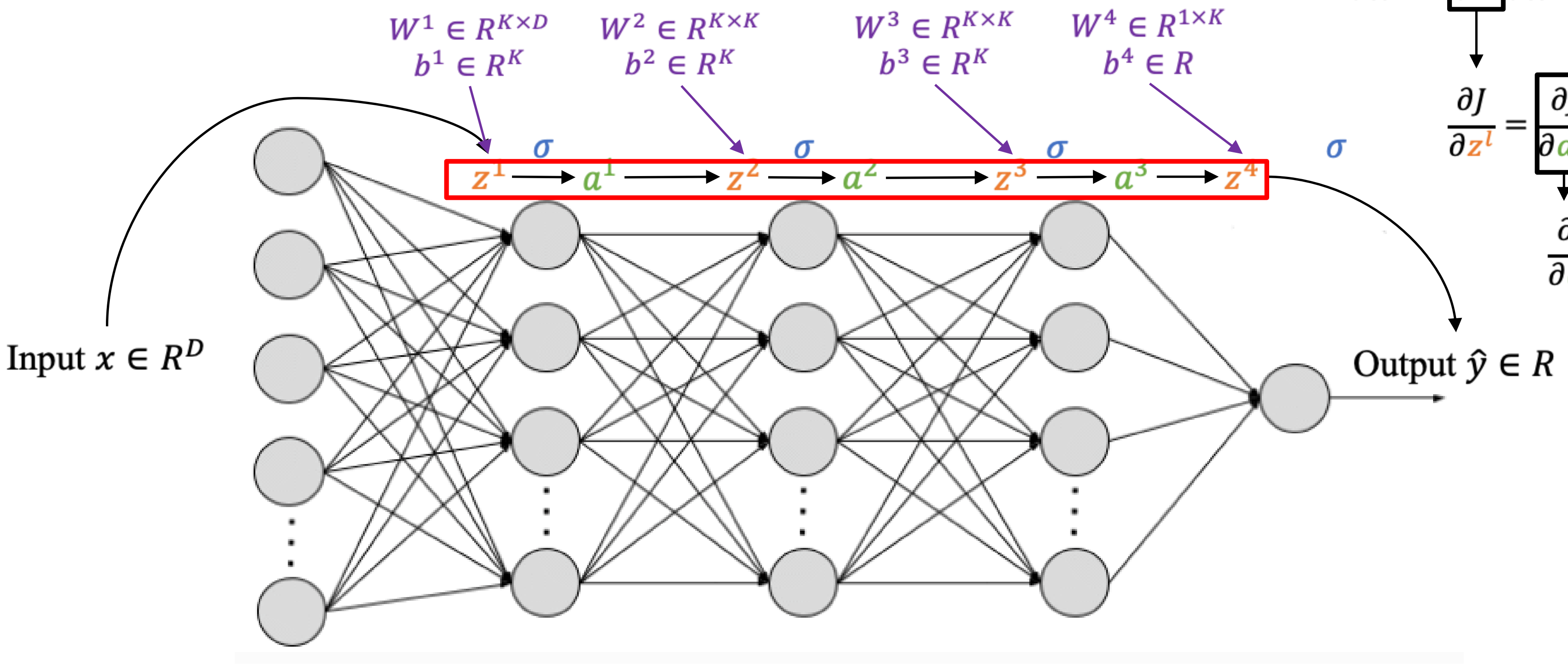
$$\frac{\partial z^{l+1}}{\partial a^l} = W^{l+1}$$

Backward Propagation

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

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

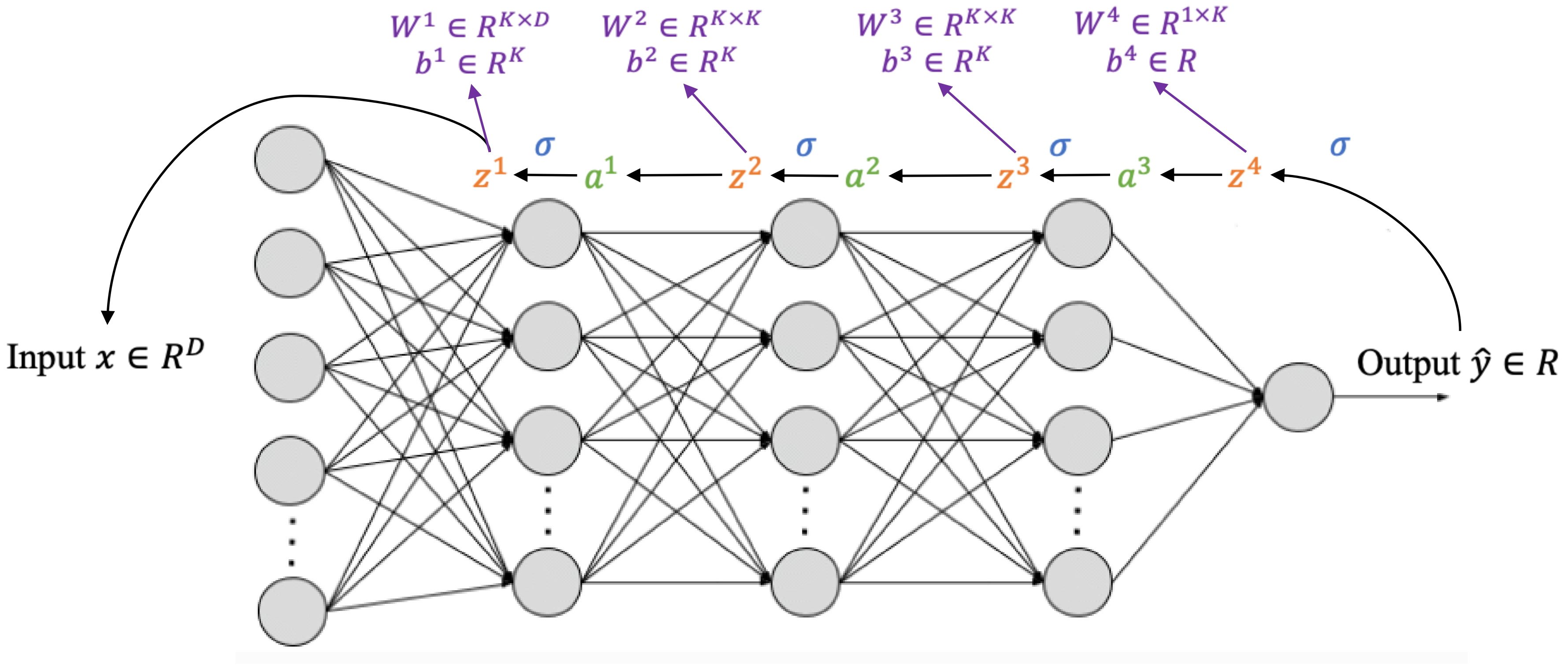
$$\frac{\partial J}{\partial a^l} = \boxed{\frac{\partial J}{\partial z^{l+1}}} \frac{\partial z^{l+1}}{\partial a^l}$$



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

Reverse the direction!

Backward Propagation

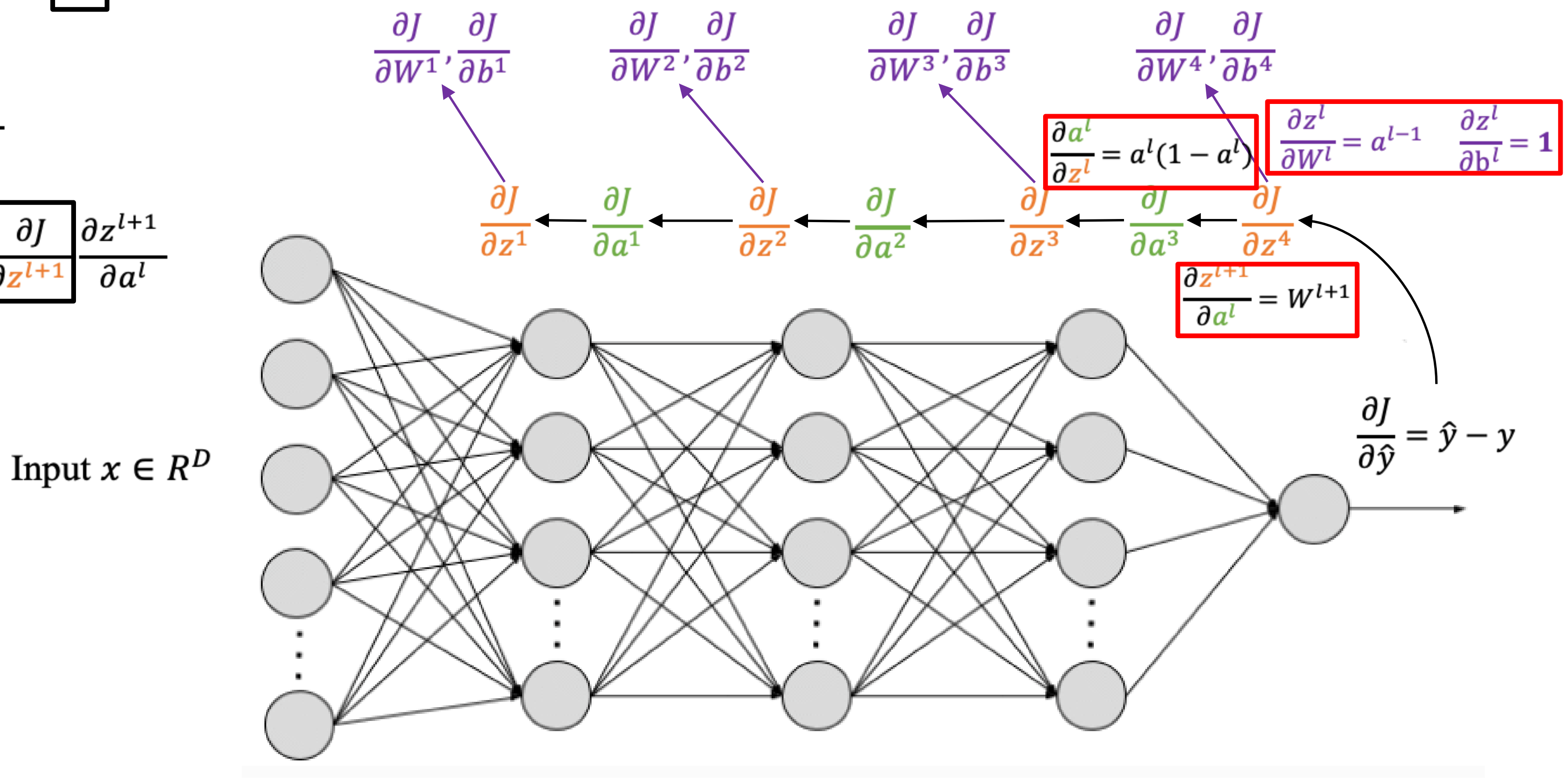


$$J = \frac{1}{2}(\hat{y} - y)^2$$

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

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

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



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