Systems for Machine Learning (CMSC828G)

# Deep Learning and Transformers

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

# Artificial neural networks

- Neural networks are parametrized function approximators
- Can work with high dimensional data: text, images, audio, …

Input $(x)$

$f(x; \theta)$

Model

Output $\hat{y}$

"A sitting dog"

- Neural networks can be used to model complex functions

# Supervised training

- Calculate loss and gradient of loss w.r.t. parameters

$(x, y)$

Parameters ($\theta$)

Model

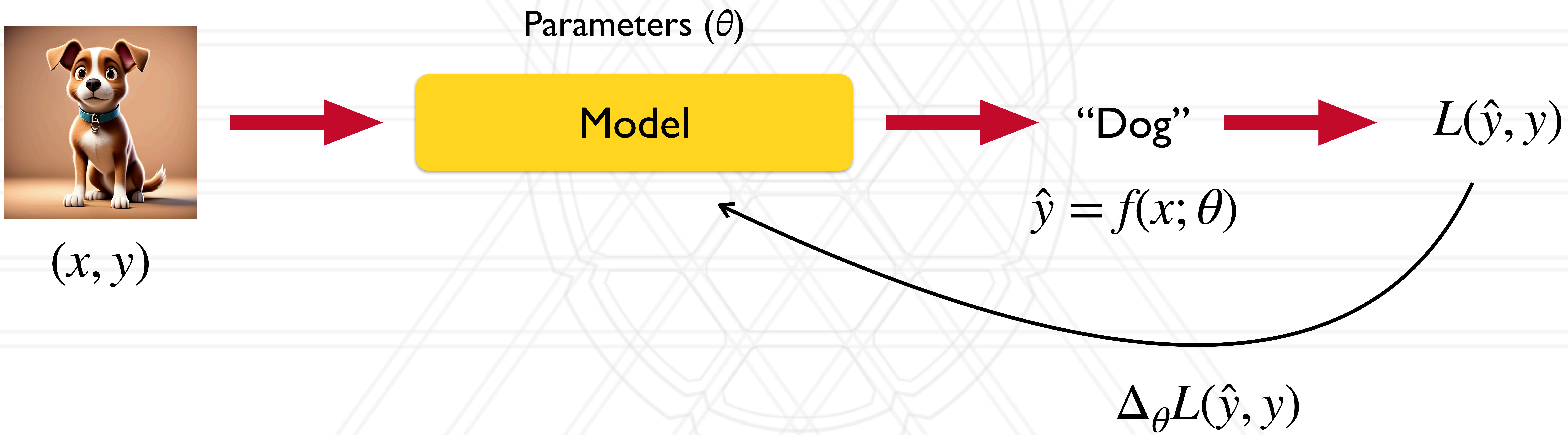
"Dog"

$\hat{y} = f(x; \theta)$

$L(\hat{y}, y)$

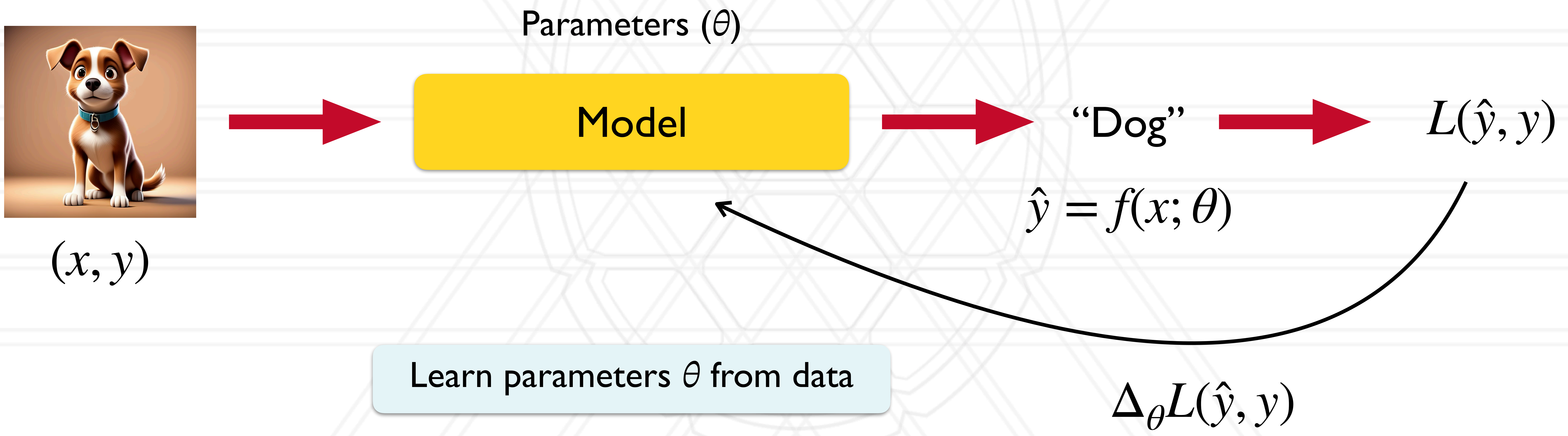
# Supervised training

- Calculate loss and gradient of loss w.r.t. parameters



# Supervised training

- Calculate loss and gradient of loss w.r.t. parameters



# Why neural networks?

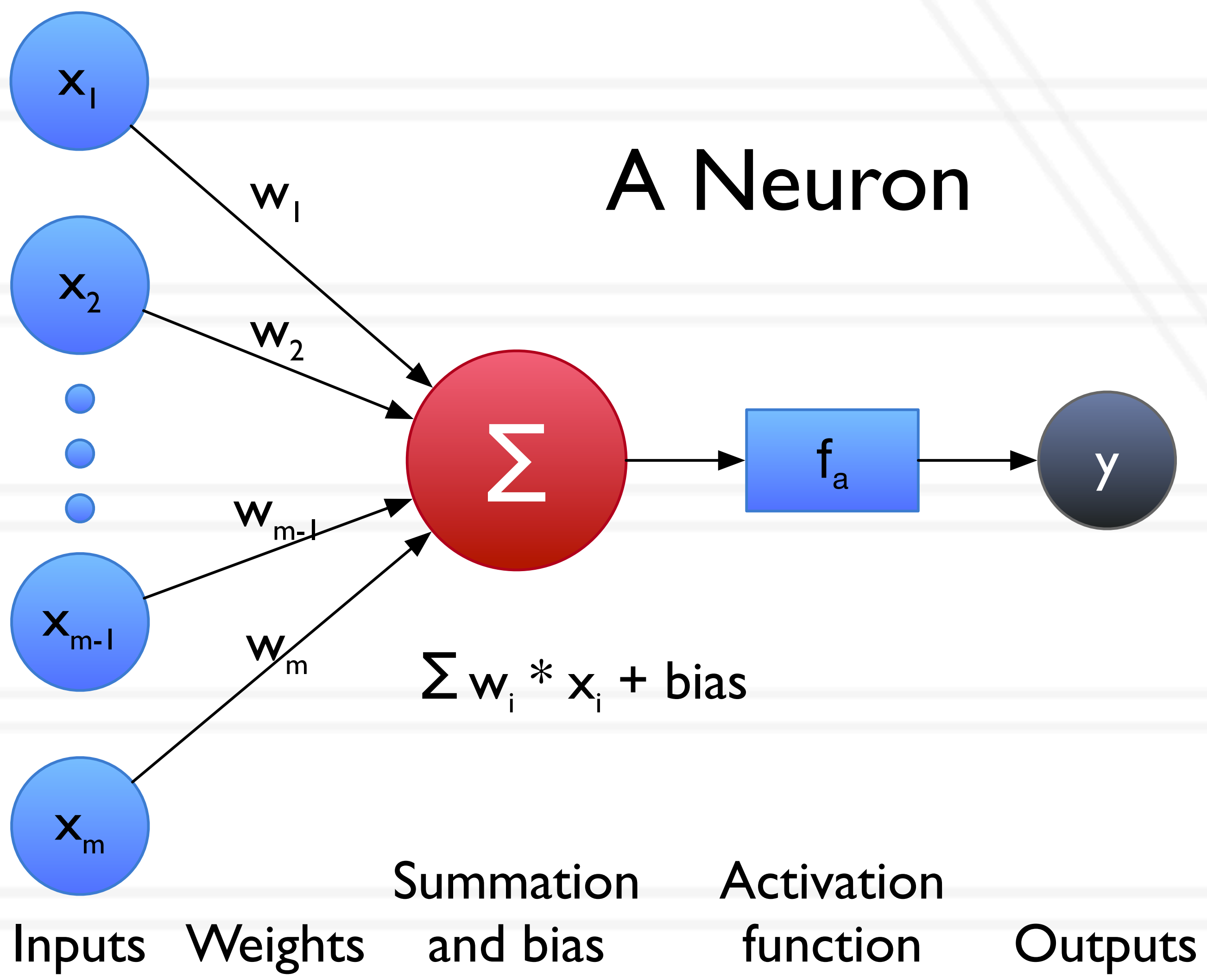
- Linear models are not always enough

$$f(x;\theta) = \theta^T x = \sum_{i=1}^{d} \theta_i x_i$$

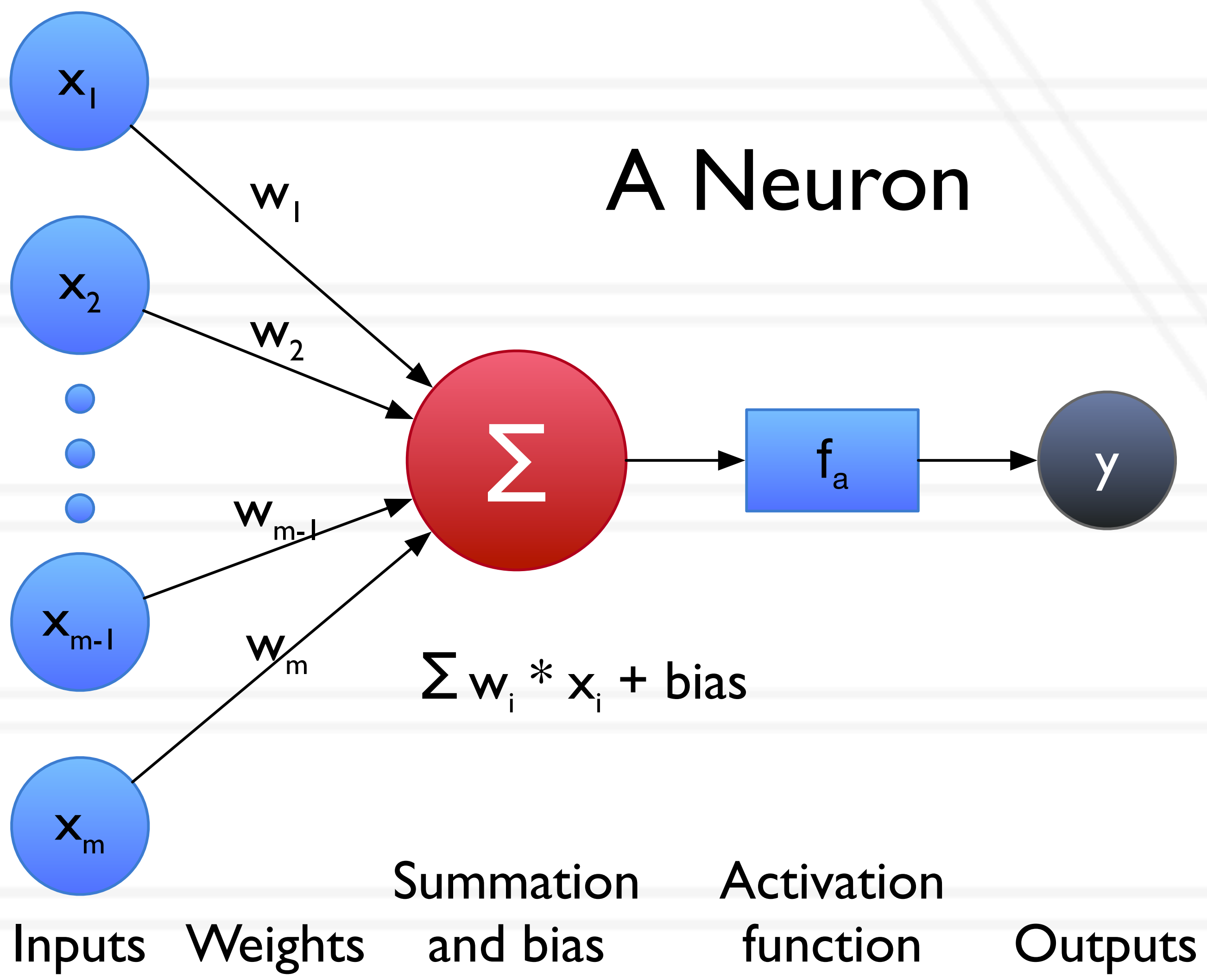
- Many real world problems are non-linear, irregular, hierarchical etc.
- Neural networks add levels of non-linearity
  - Via applying activations on a linear transformation

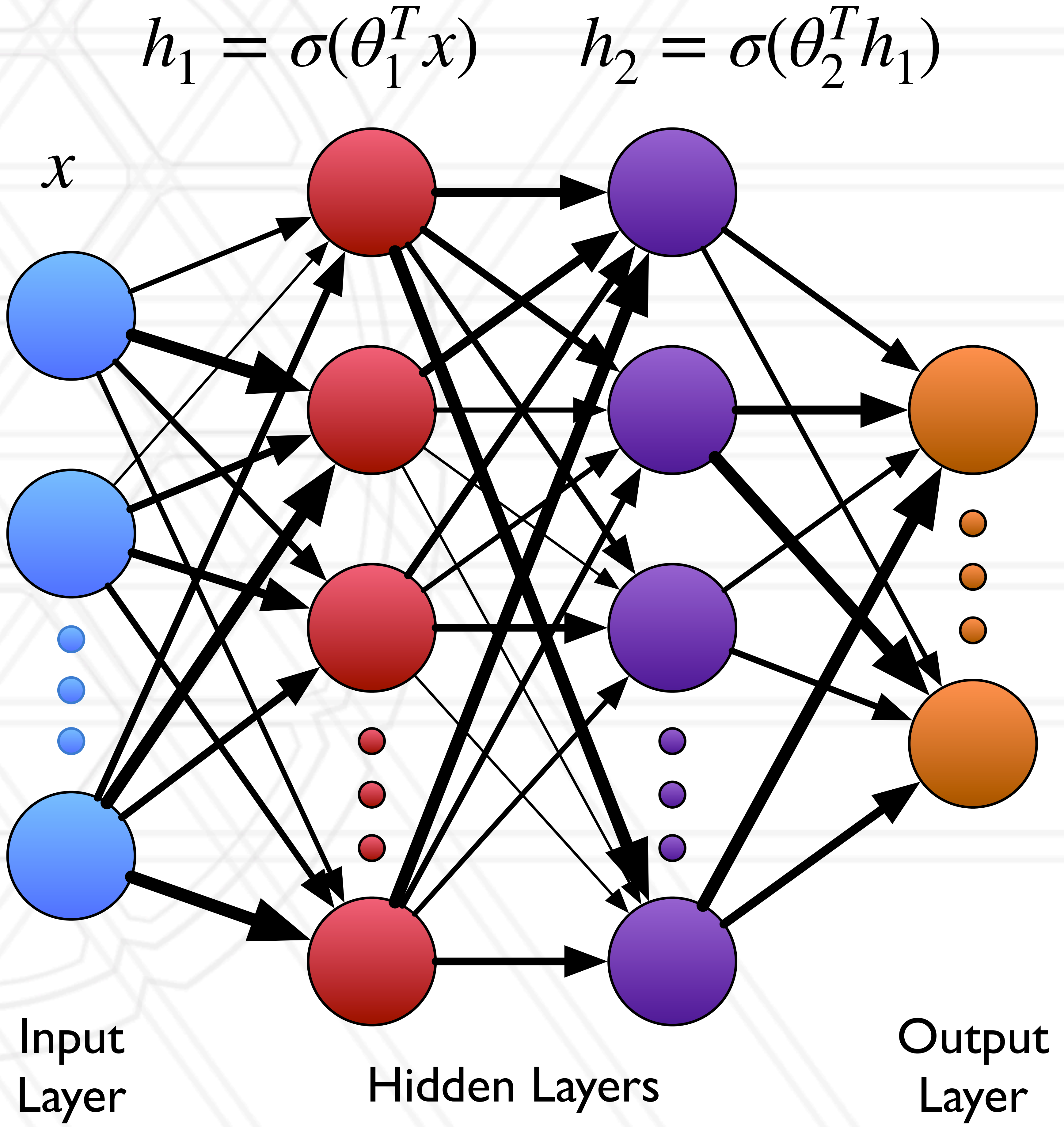
$$h = \sigma(\theta^T x)$$

# Deep neural networks (DNNs)

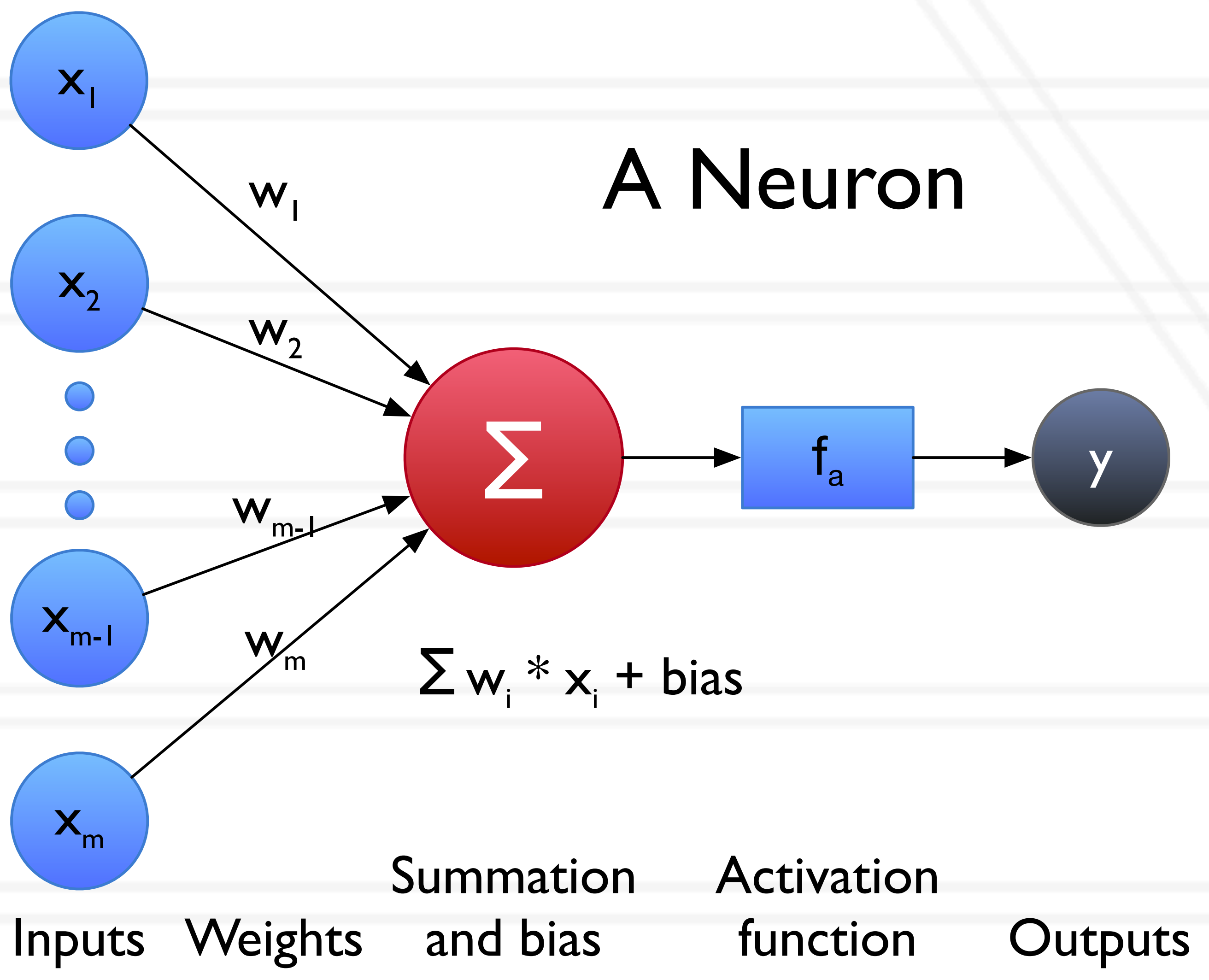


# Deep neural networks (DNNs)

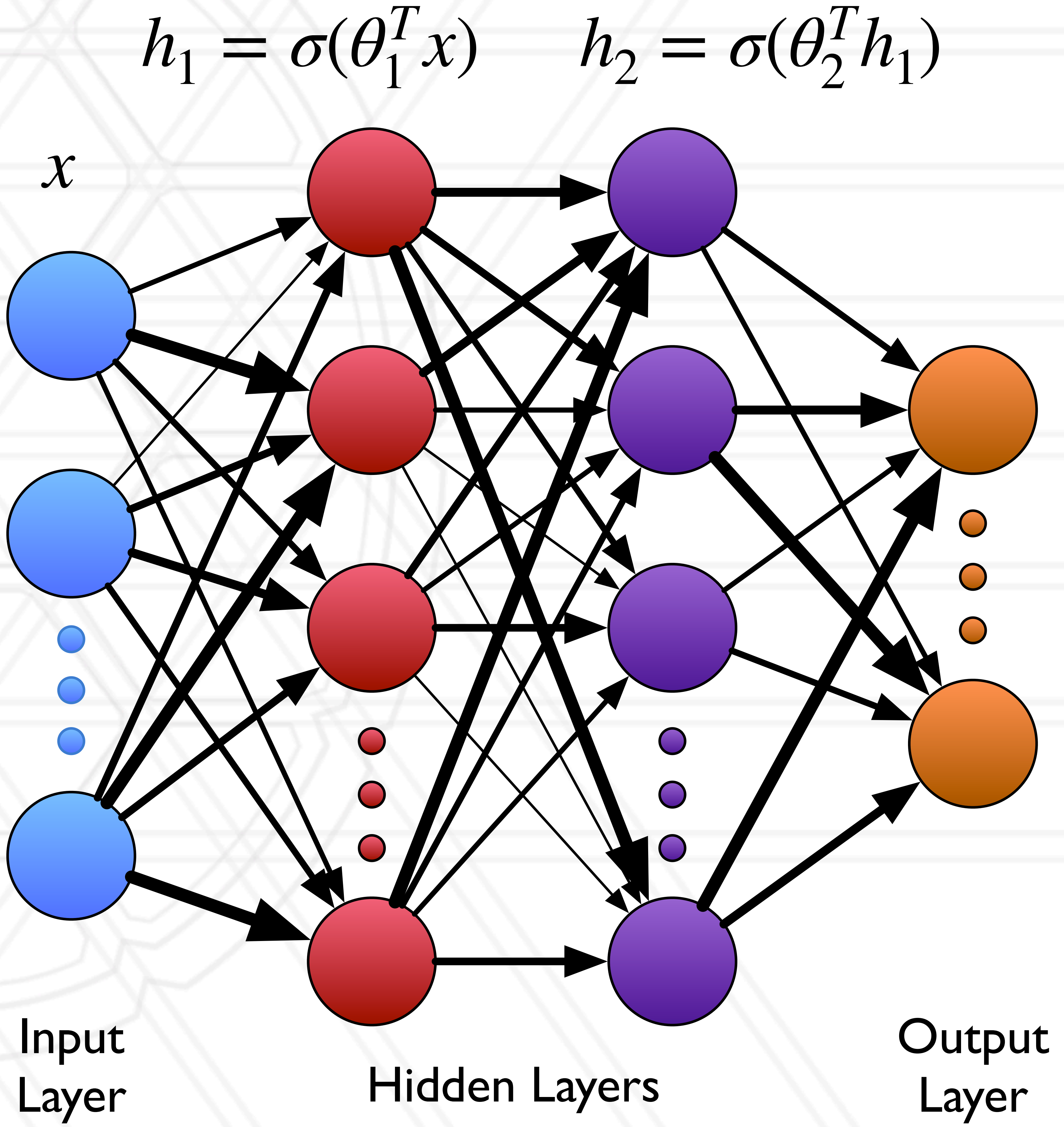


# Deep neural networks (DNNs)

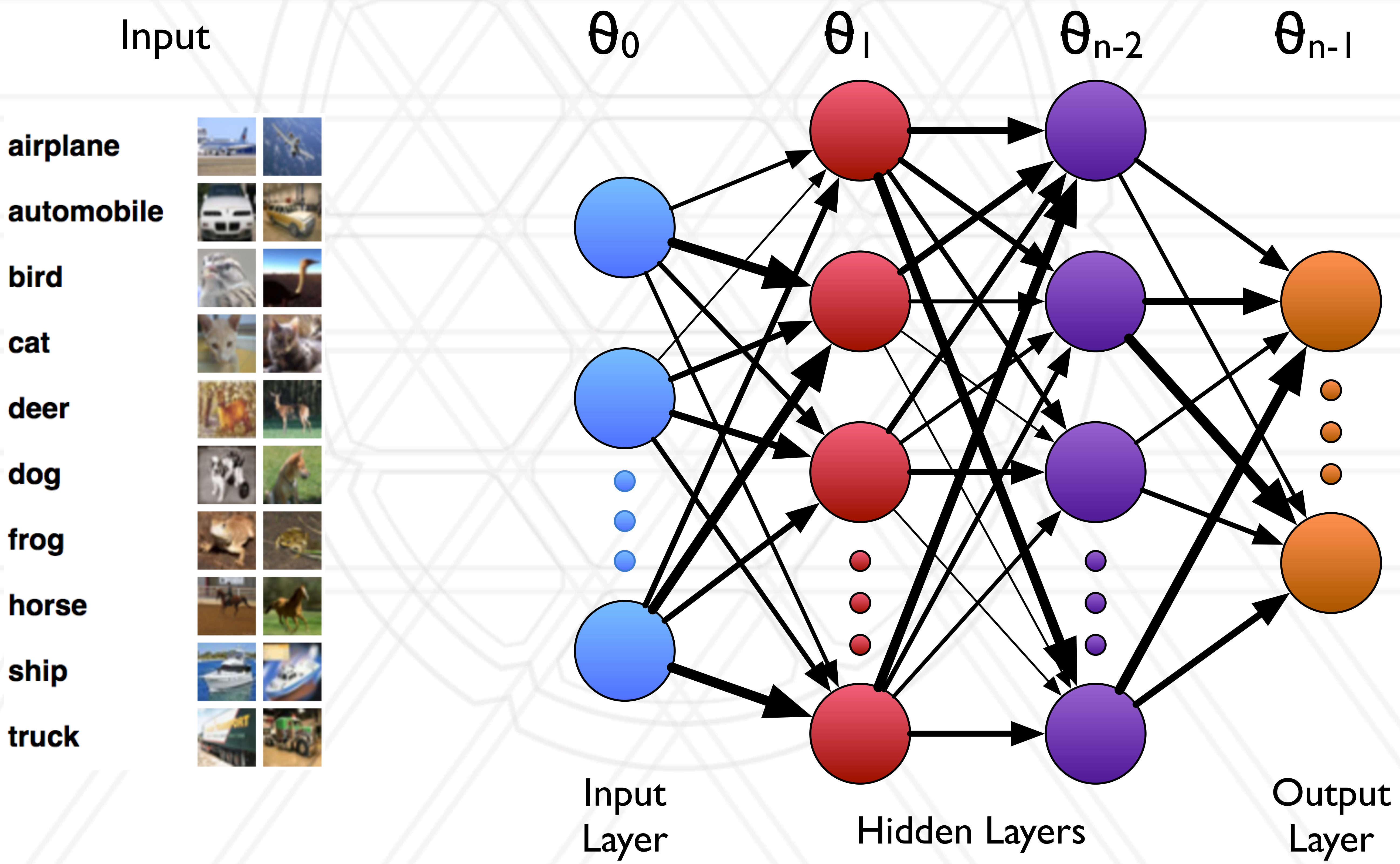



deep refers to having several layers



# Training a neural network

- Problem: Find a set of weights/parameters that best fits the function we are trying to learn over a given training dataset



# DNN training loop

```
while (data) {
    Read a single batch

    Forward pass: perform matrix multiplies to compute
        output activations

    Compute loss on this batch

    Backward pass: matrix multiplies to compute gradients of
        the loss w.r.t. parameters via backpropagation

    Optimizer step: use gradients to update the weights or
        parameters such that loss is gradually reduced
}
```

# Optimizer

- Used to update the parameters using the gradients
- They help minimize complex loss functions iteratively
- Gradient descent

$$\theta_{t+1} = \theta_t - \eta \Delta_\theta L$$

- Adam: Adaptive moment estimation
  - Adapts the learning rate per parameter

# Optimizer

- Used to update the parameters using the gradients
- They help minimize complex loss functions iteratively
- Gradient descent

$$\theta_{t+1} = \theta_t - \eta \Delta_\theta L$$

$\eta$ = learning rate

- Adam: Adaptive moment estimation
  - Adapts the learning rate per parameter

# Terms and definitions

- Learning/training: task of selecting weights ($\theta$) that lead to an accurate function
- Loss: a scalar proxy that when minimized leads to higher accuracy
- Gradient descent: process of updating the weights using gradients (derivates) of the loss weighted by a learning rate
- Mini-batch: Small subsets of the dataset processed iteratively
- Epoch: One pass over all the mini-batches

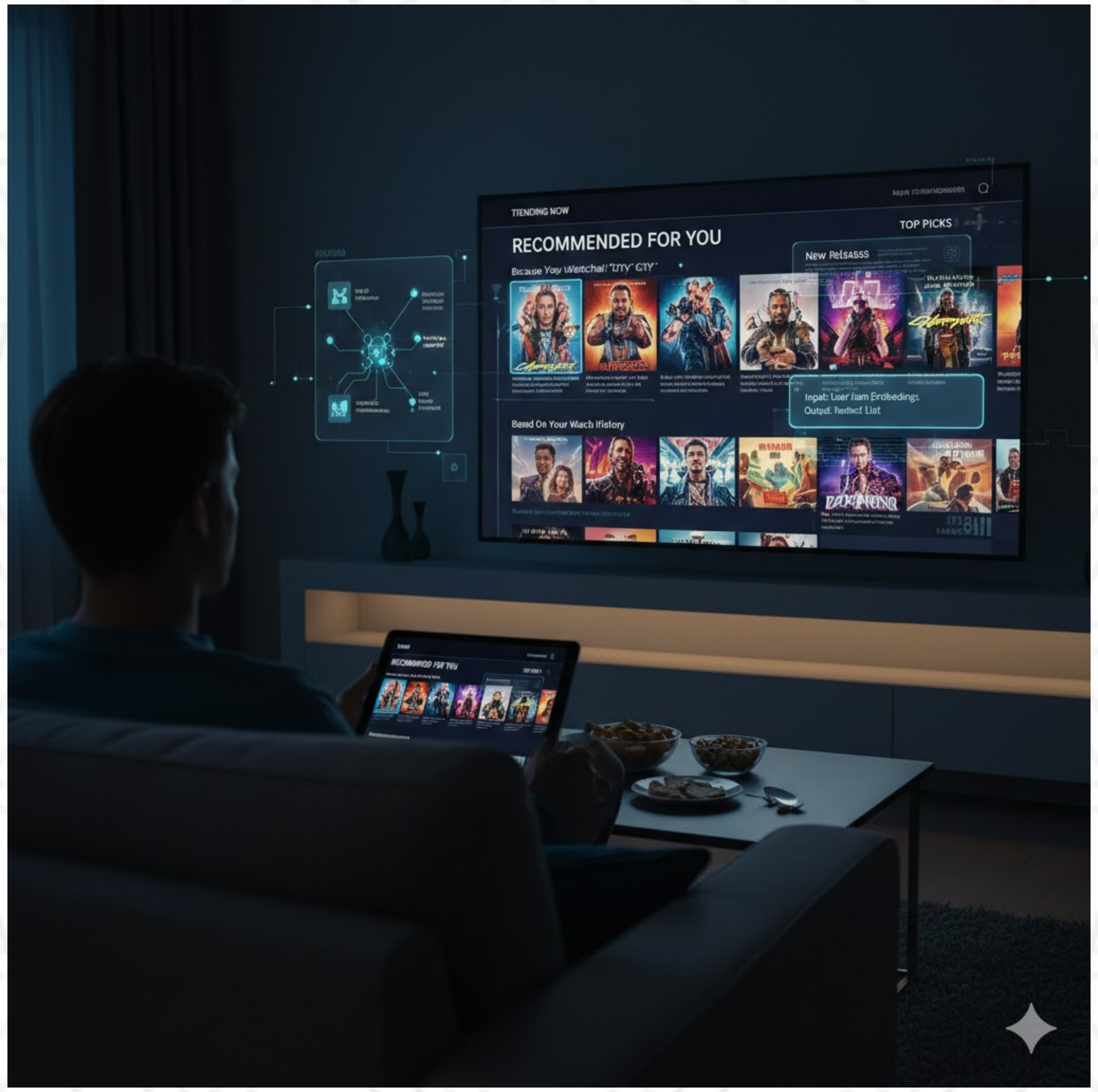
# Different types of DNNs

# Different types of DNNs


Convolutional NNs (CNNs)

Image Recognition

# Different types of DNNs

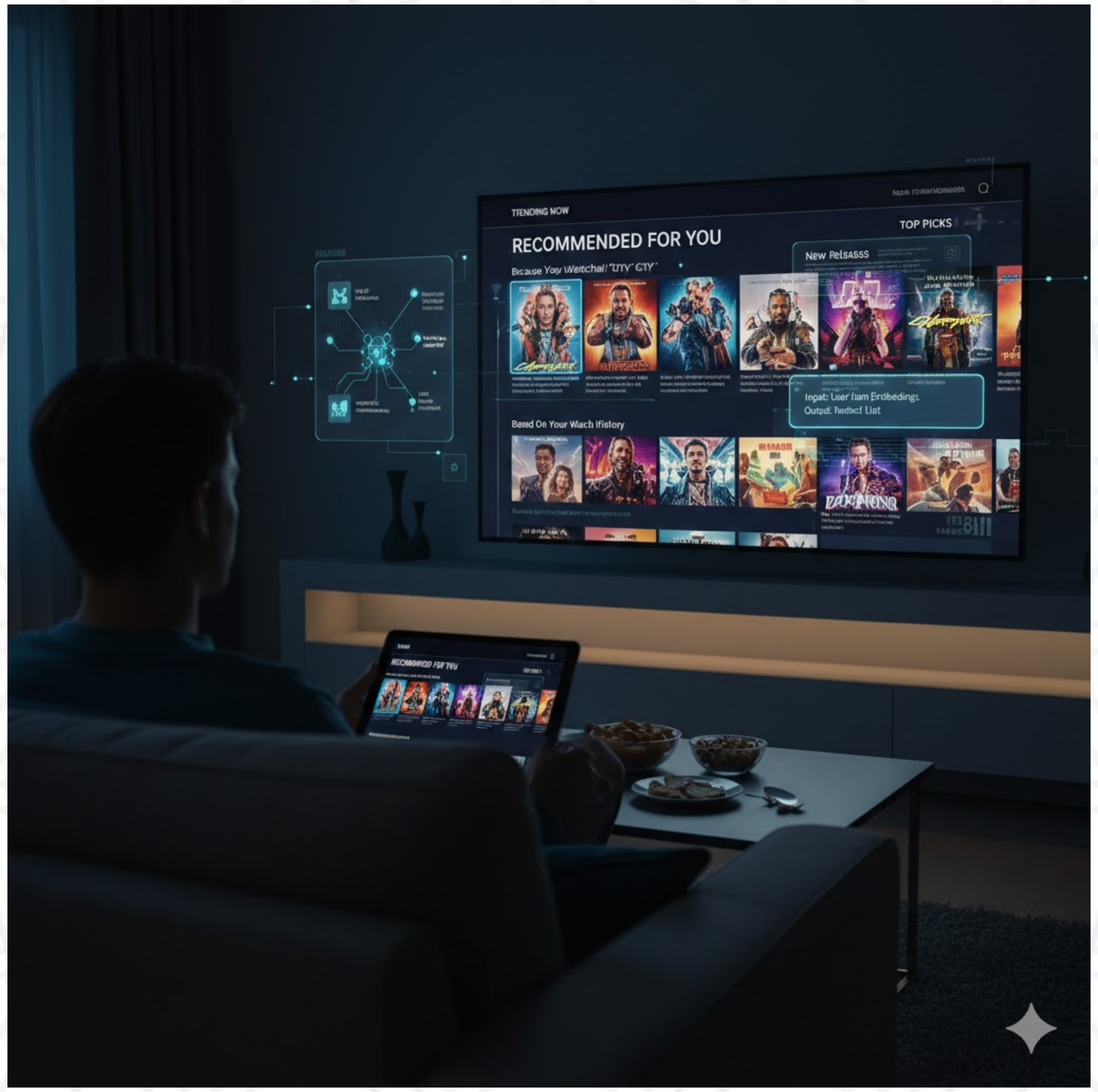

**Convolutional NNs (CNNs)**

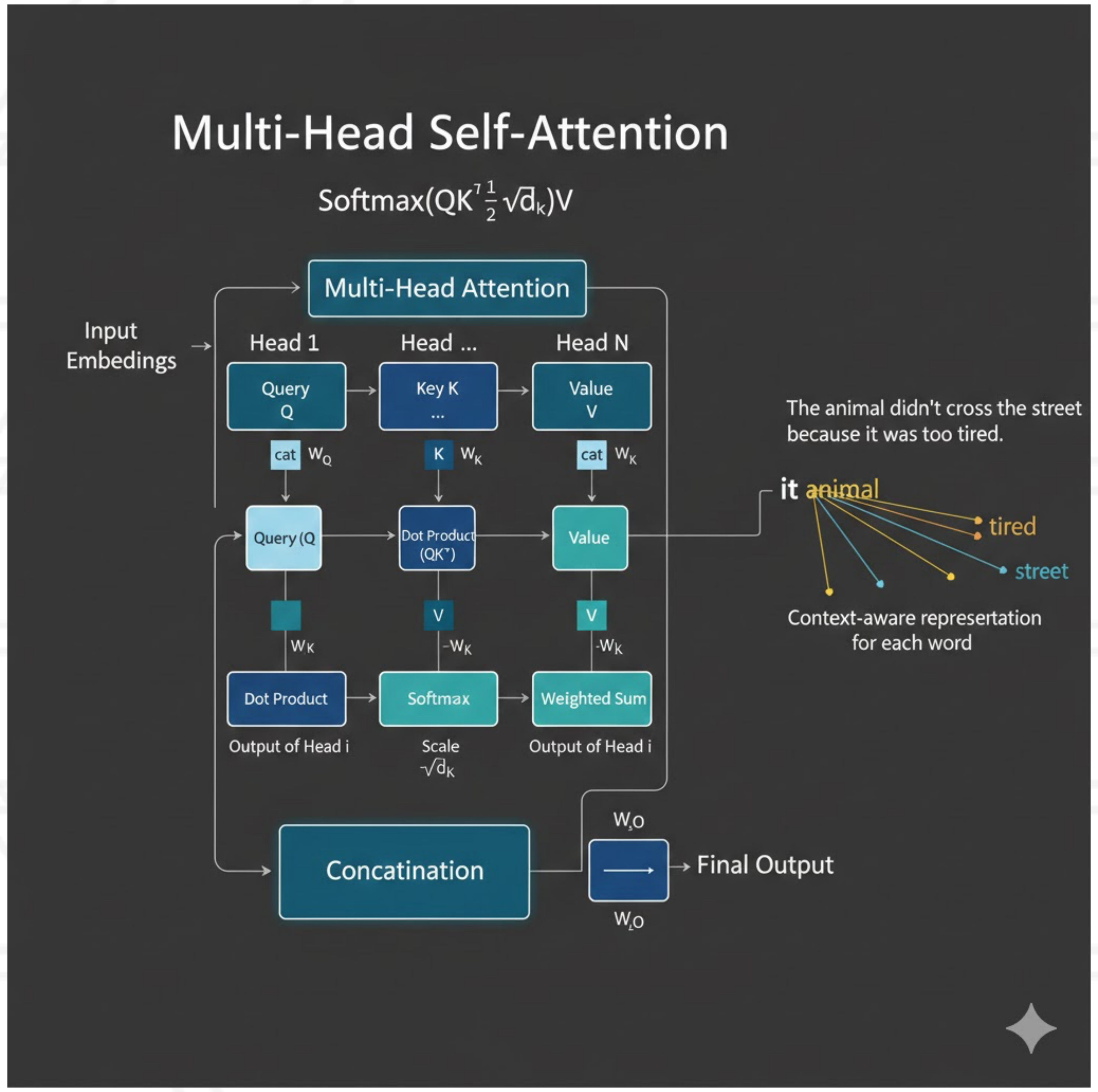
Image Recognition

**Recommendation Systems**

Recommend products, movies, …

# Different types of DNNs


Convolutional NNs (CNNs)

Image Recognition

Recommendation Systems

Recommend products, movies, …

Transformers and LLMs

Generate text

# Language modeling

- The idea is to predict the next word

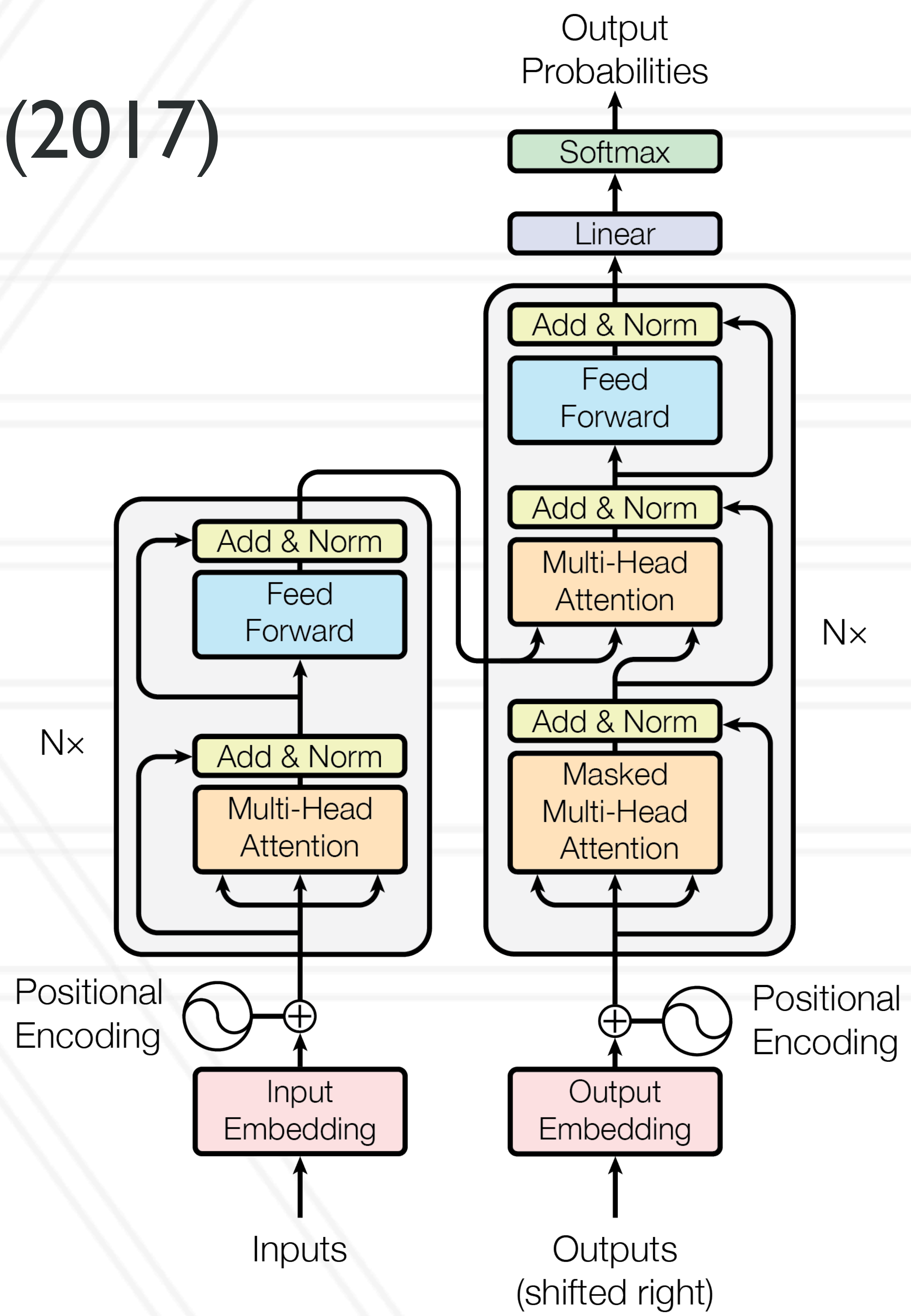
$$P(w_t \mid w_1, w_2, \ldots, w_{t-1})$$

- Ideally, we want to be able to capture long-range dependencies
  - These fade in other approaches such as RNNs

The animal didn't cross the street because it was …

# The Transformers Architecture

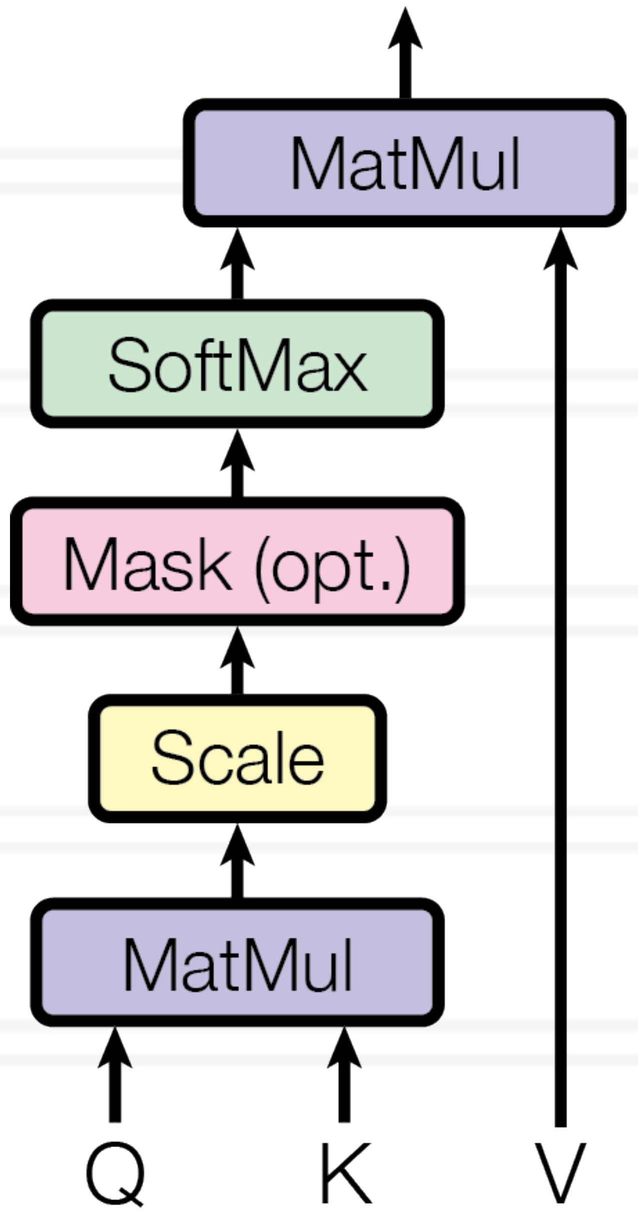
- A paradigm shift introduced in "Attention is All You Need" (2017)
- Relies entirely on attention
- Enables highly parallel computation
- Encoder stacks: MHSA + FFN
- Decoder stacks



# Scaled Dot-Product Attention

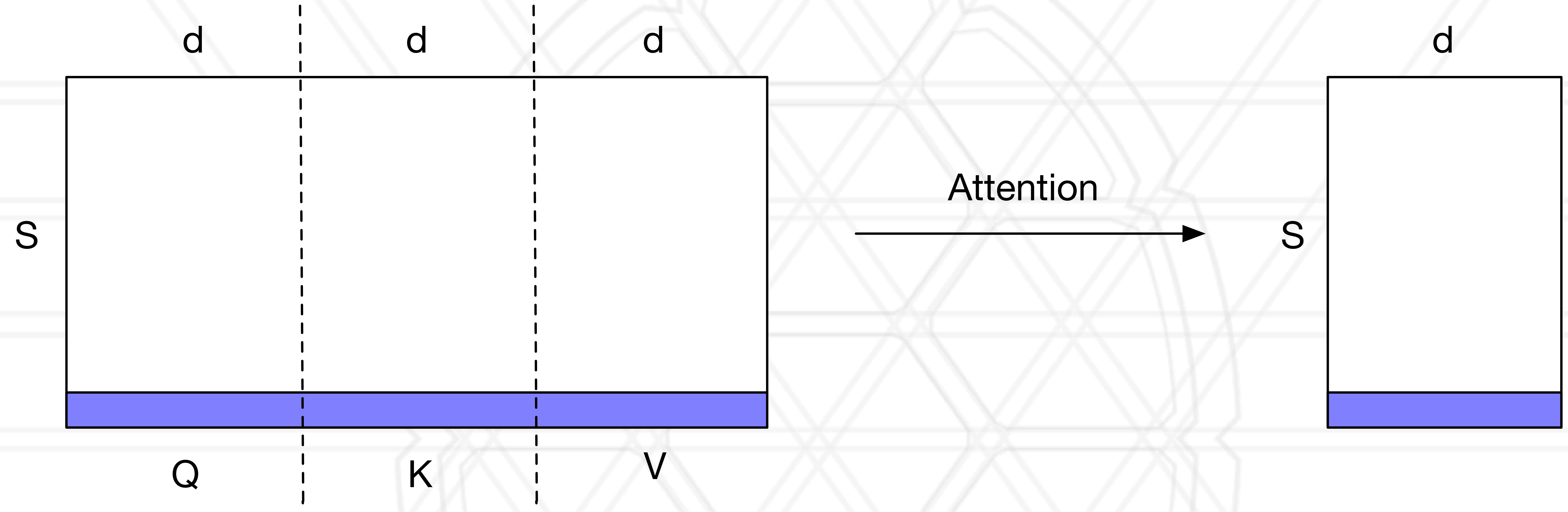
- Determine "how much" should a token "attend" to other tokens
  - When processing a word, which other words matter the most?

The animal didn't cross the street because ***it*** was too tired

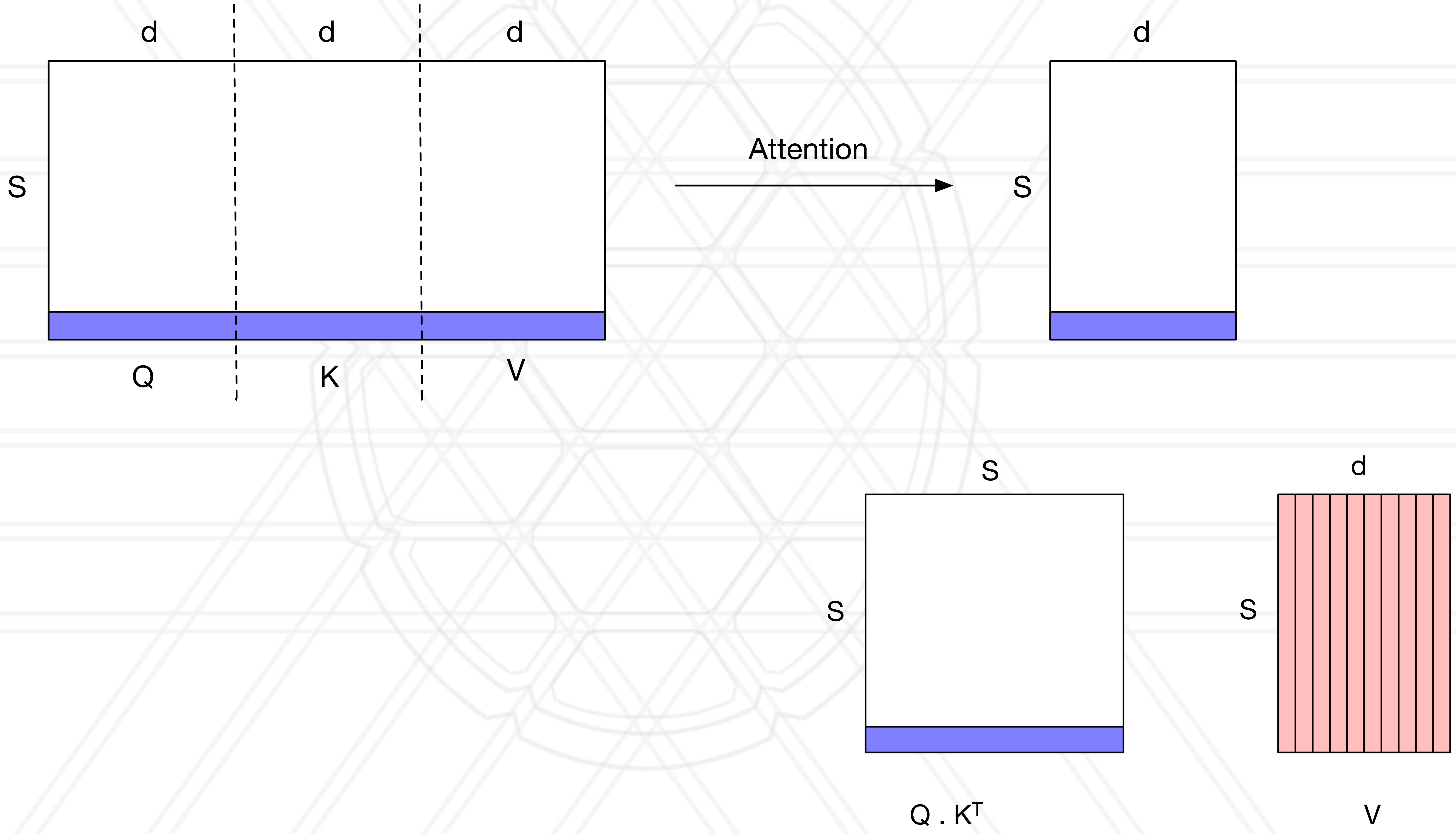



- Conceptually, each word produces three vectors:
  - Query (Q): What am I looking for?
  - Key (K): What do I contain?
  - Value (V): What information do I provide?

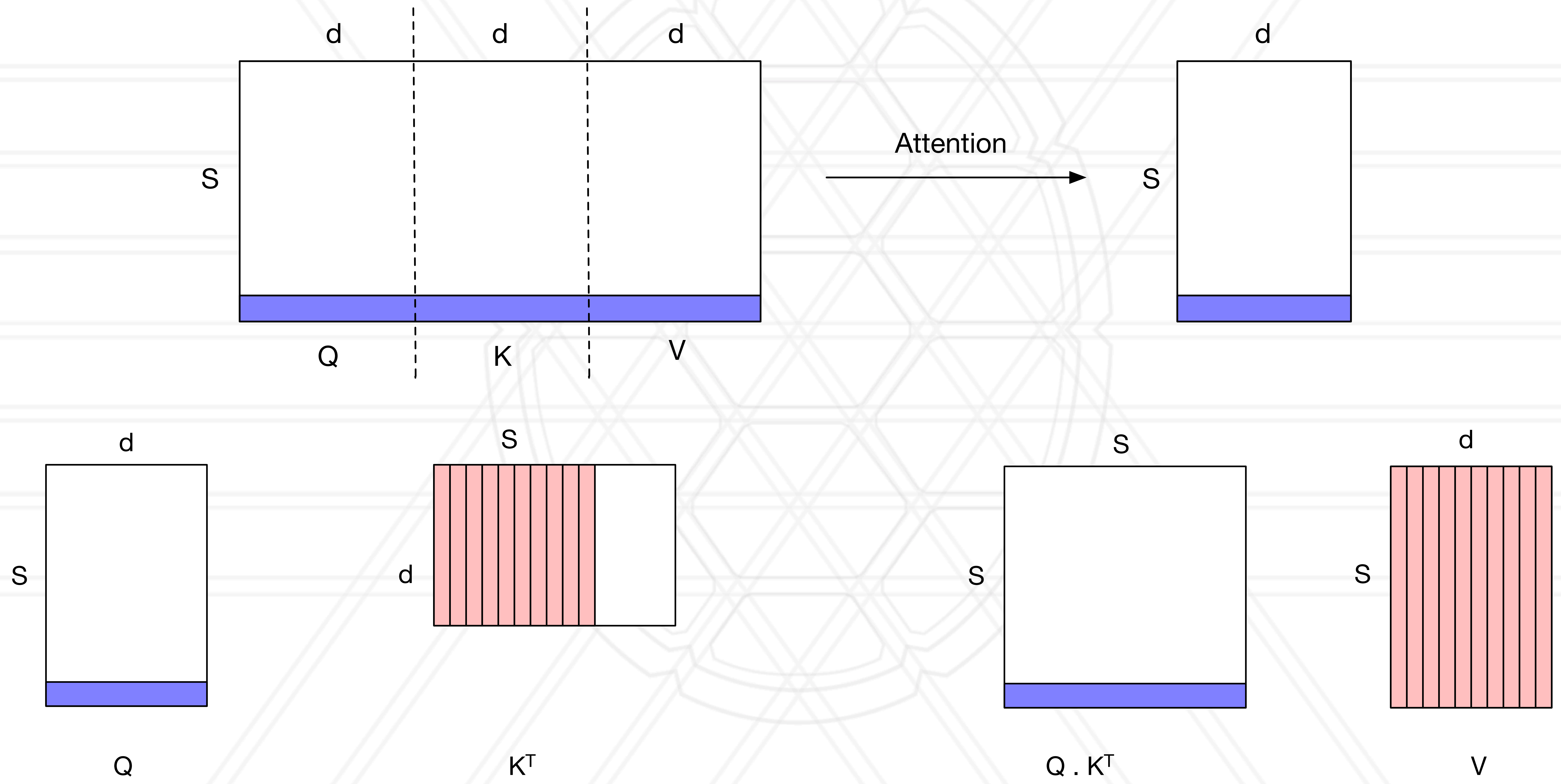
# Matrix multiplies in attention



# Matrix multiplies in attention



# Matrix multiplies in attention



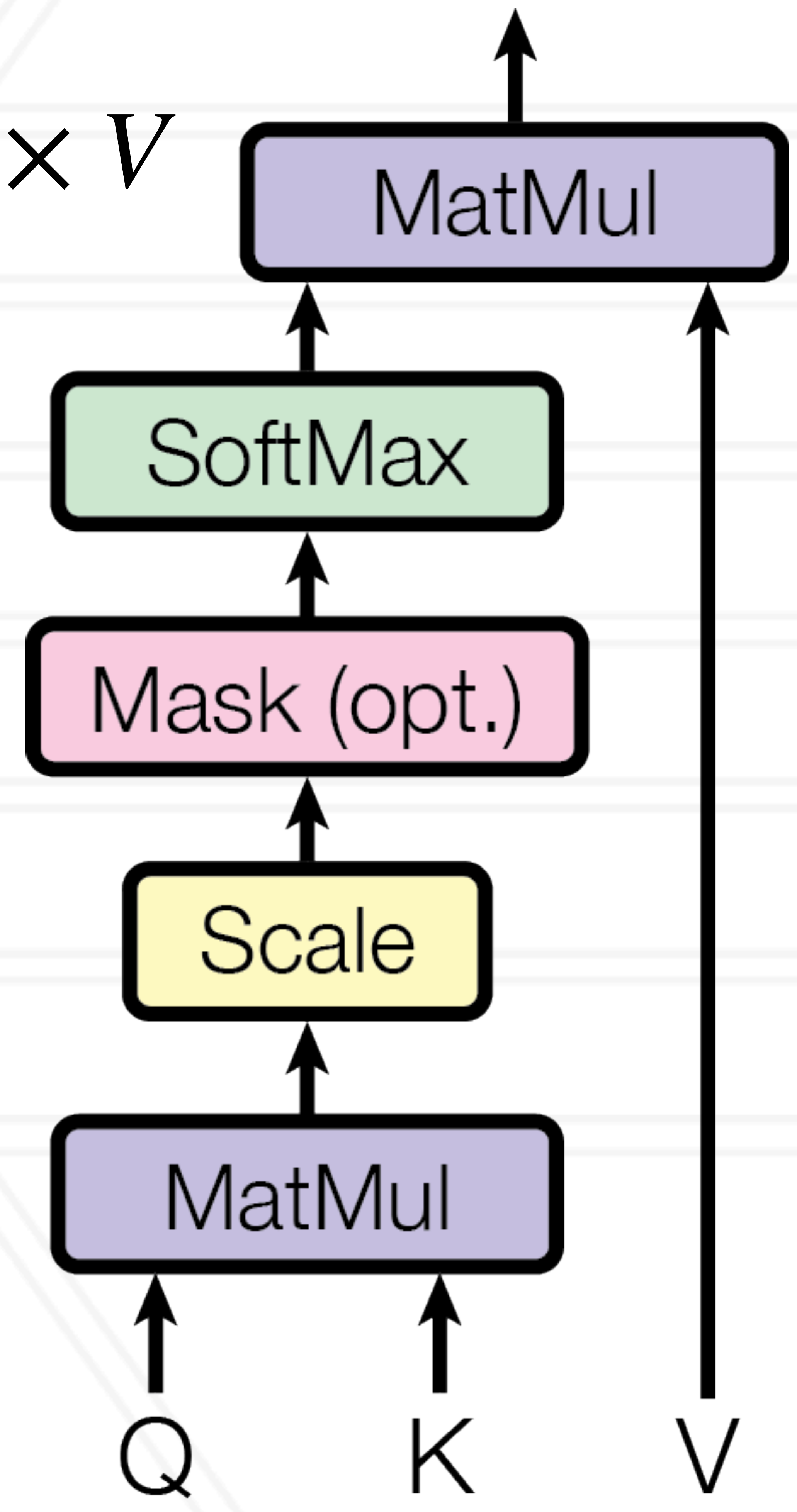
# Scaled Dot-Product Attention

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V$$

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)$$

$$\frac{Q \times K^T}{\sqrt{d_k}}$$

$$Q \times K^T$$



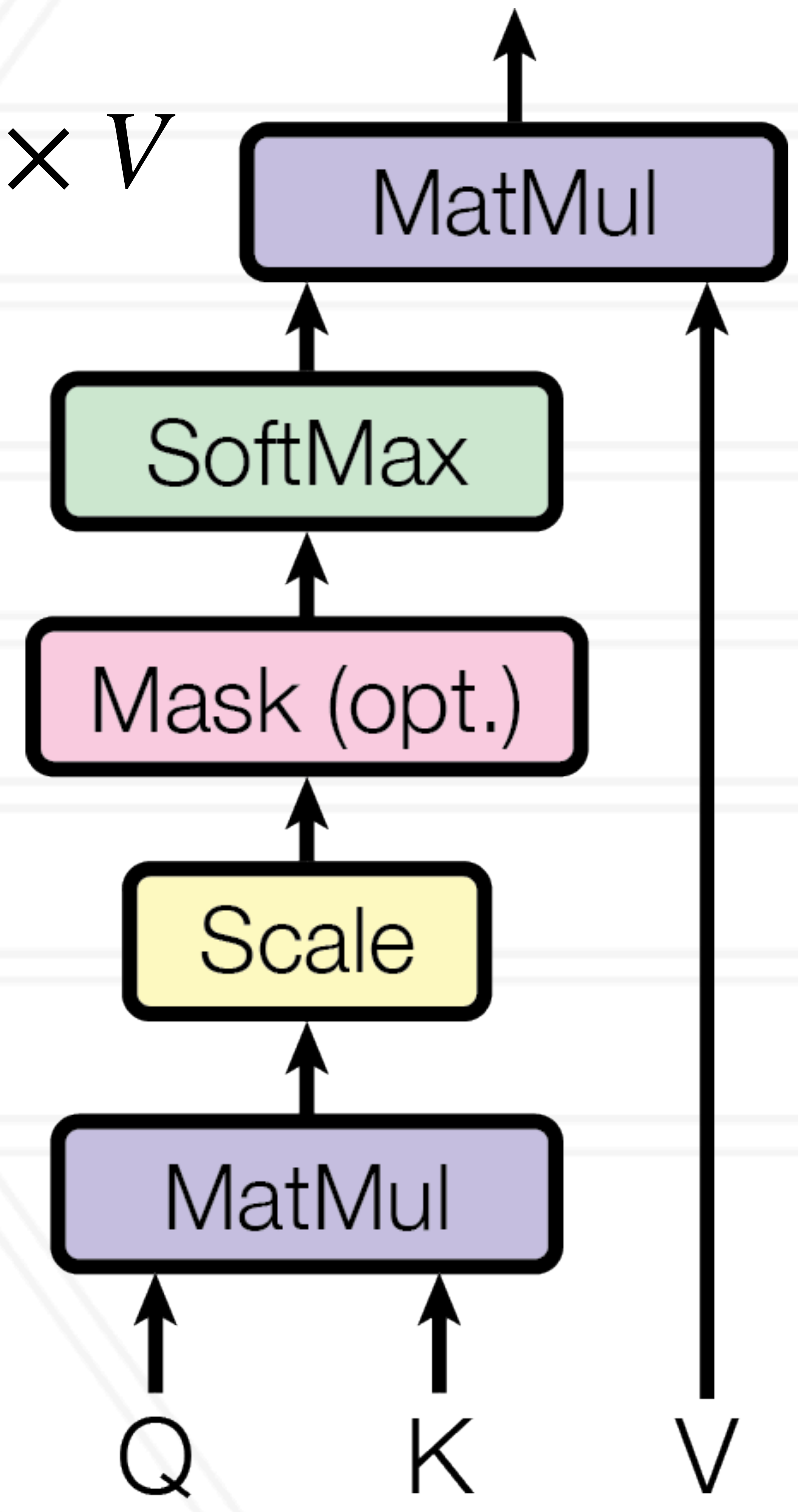
# Scaled Dot-Product Attention

Attention (Q, K,V) = $\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V$

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)$$

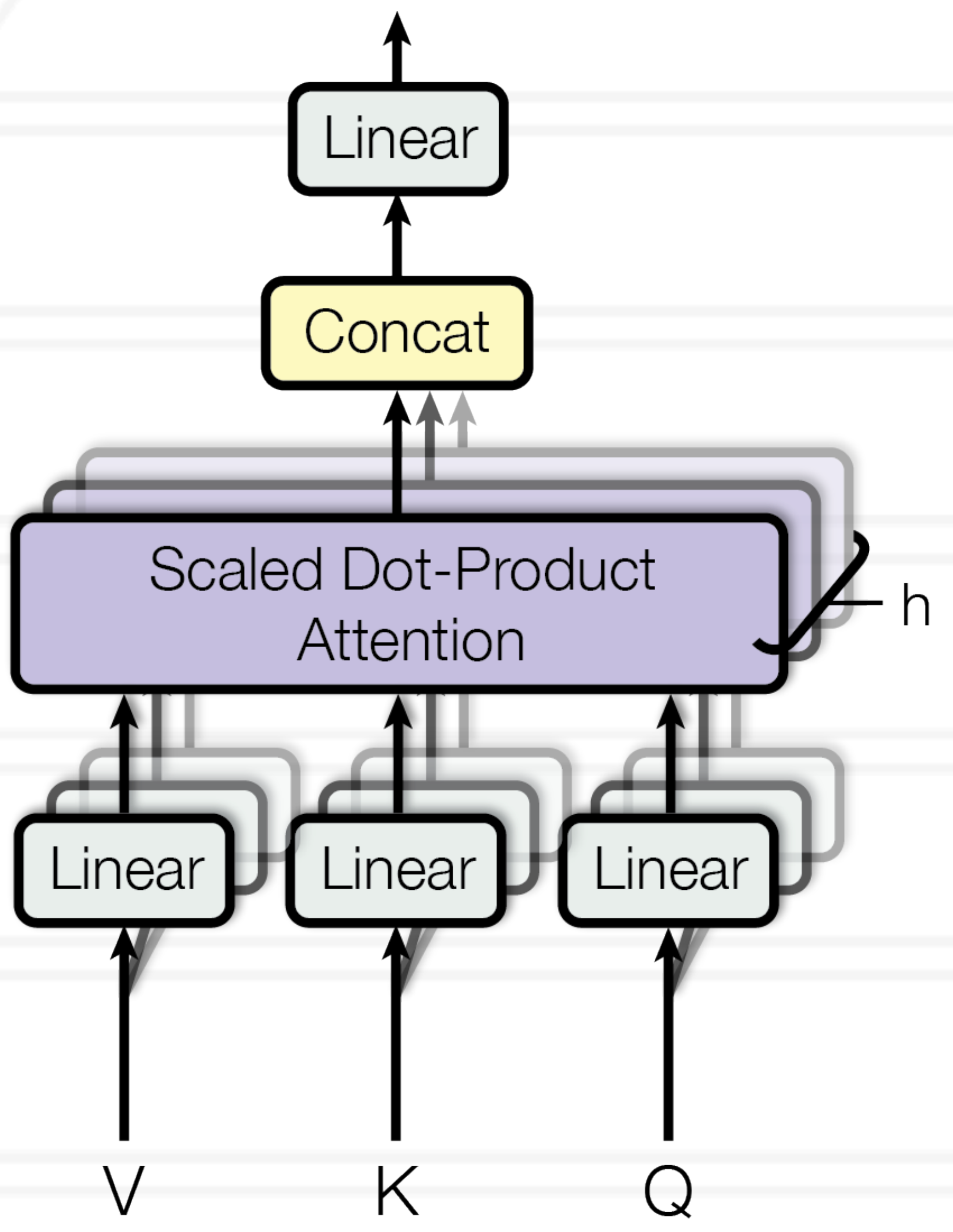
$$\frac{Q \times K^T}{\sqrt{d_k}}$$

$$Q \times K^T$$

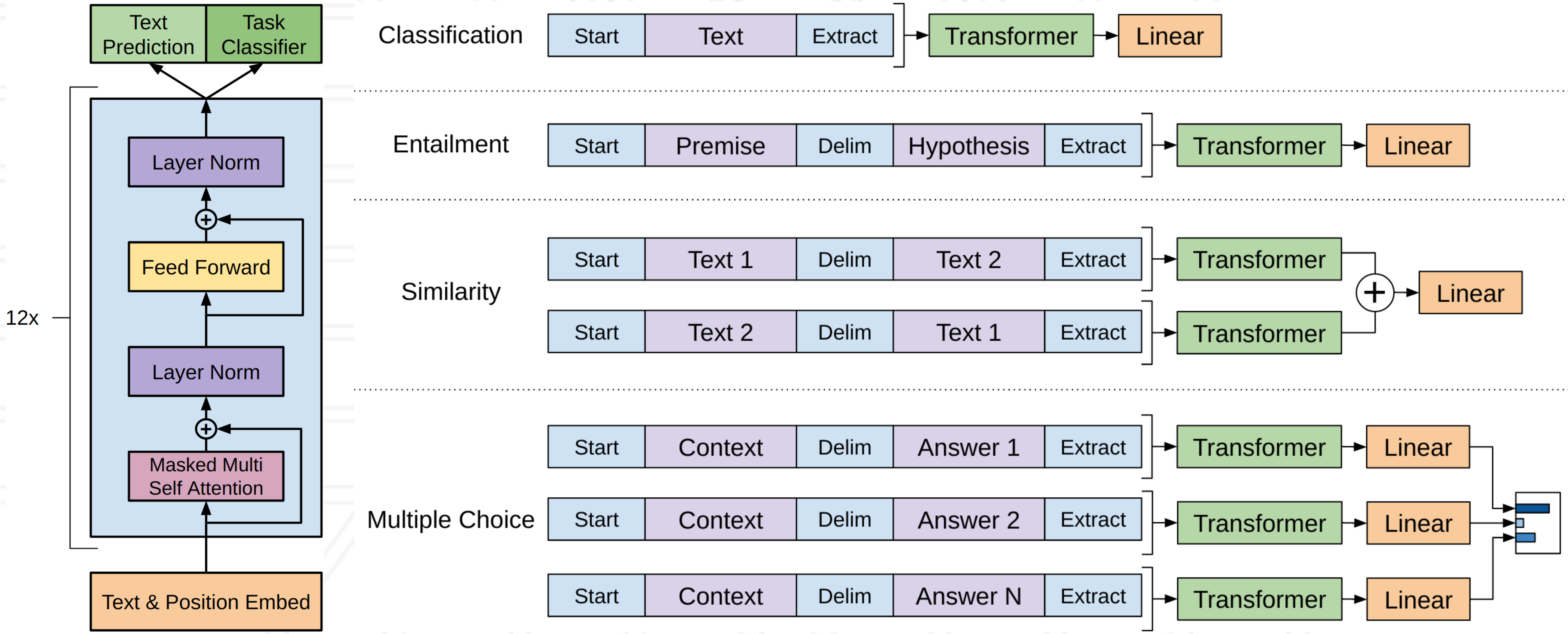


# Multi-head Attention

- Create h copies of Q, K,V
- Do attention in parallel

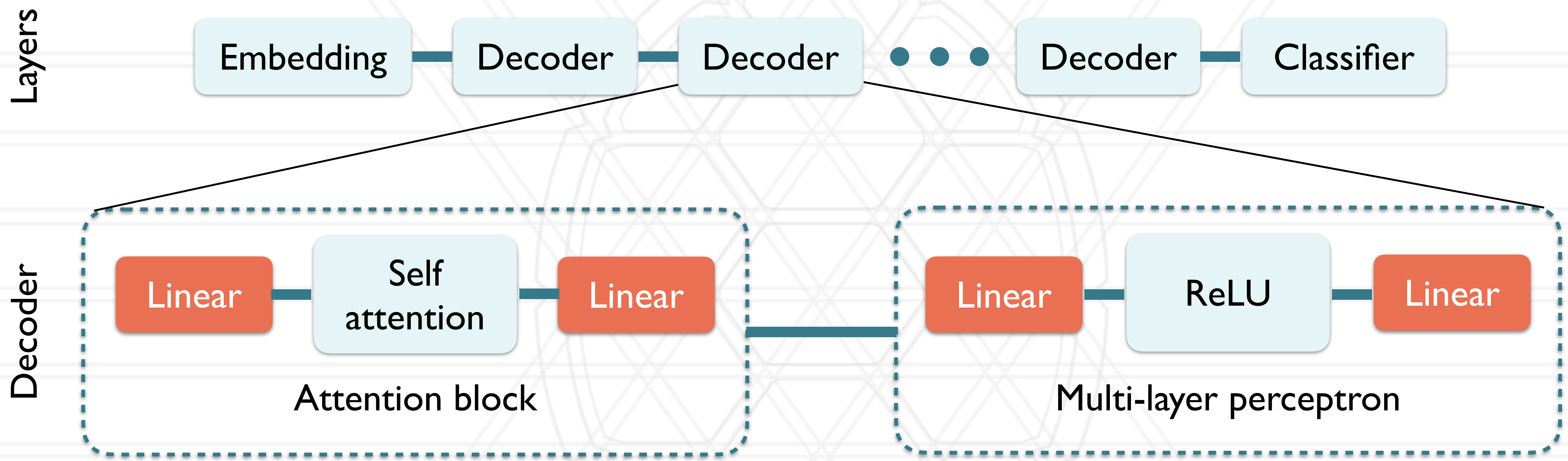


# Generative Pre-training (GPT)

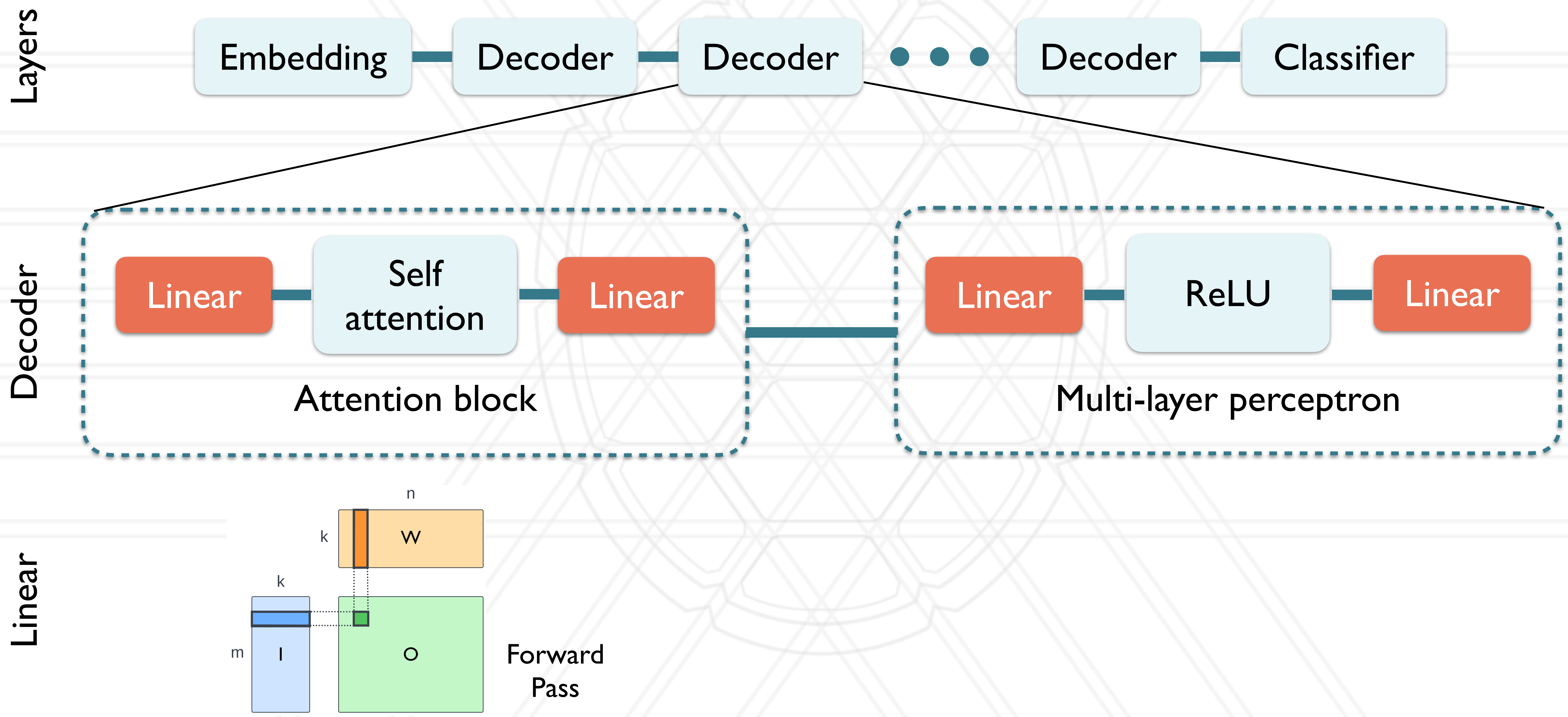


# Compute work in transformer models

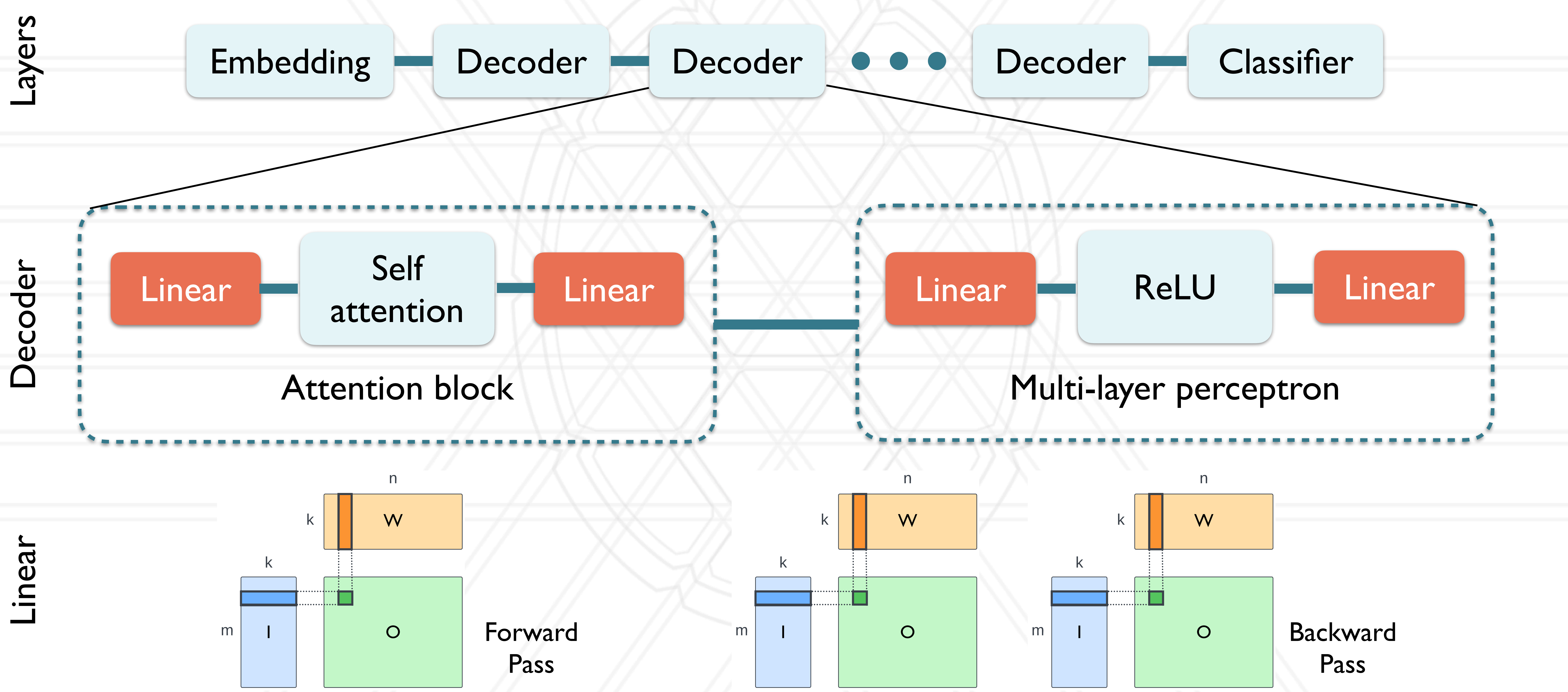
# Compute work in transformer models



# Compute work in transformer models



# Compute work in transformer models



UNIVERSITY OF
MARYLAND