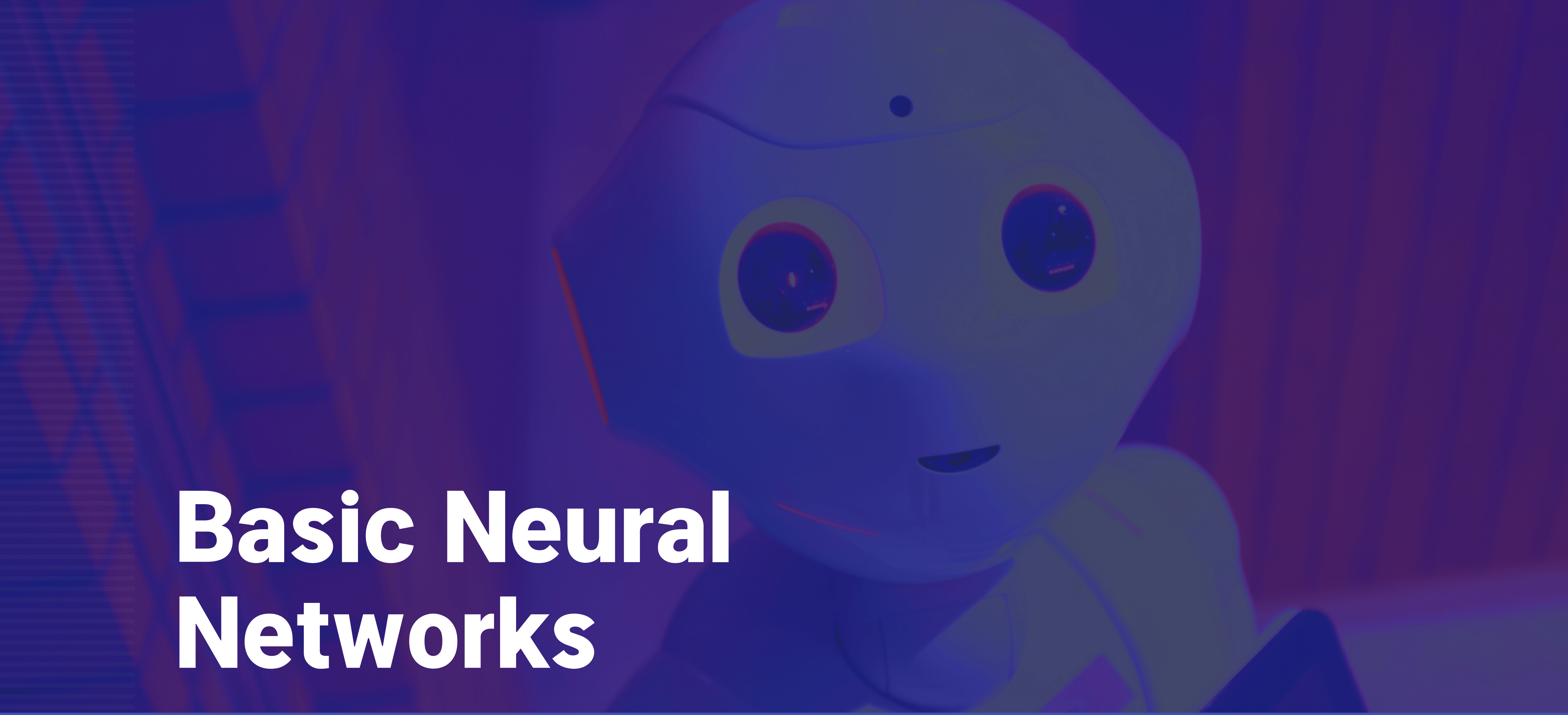


Basic Neural Networks

OCTOBER 18, 2017 — WORKSHOP #1

Student AI Group

Intro

We want to generate a prediction from some input

Output should give some probability of all our expected outputs

Network should give highest probability to correct class most of the time

Forward Pass

Student
AI
Group

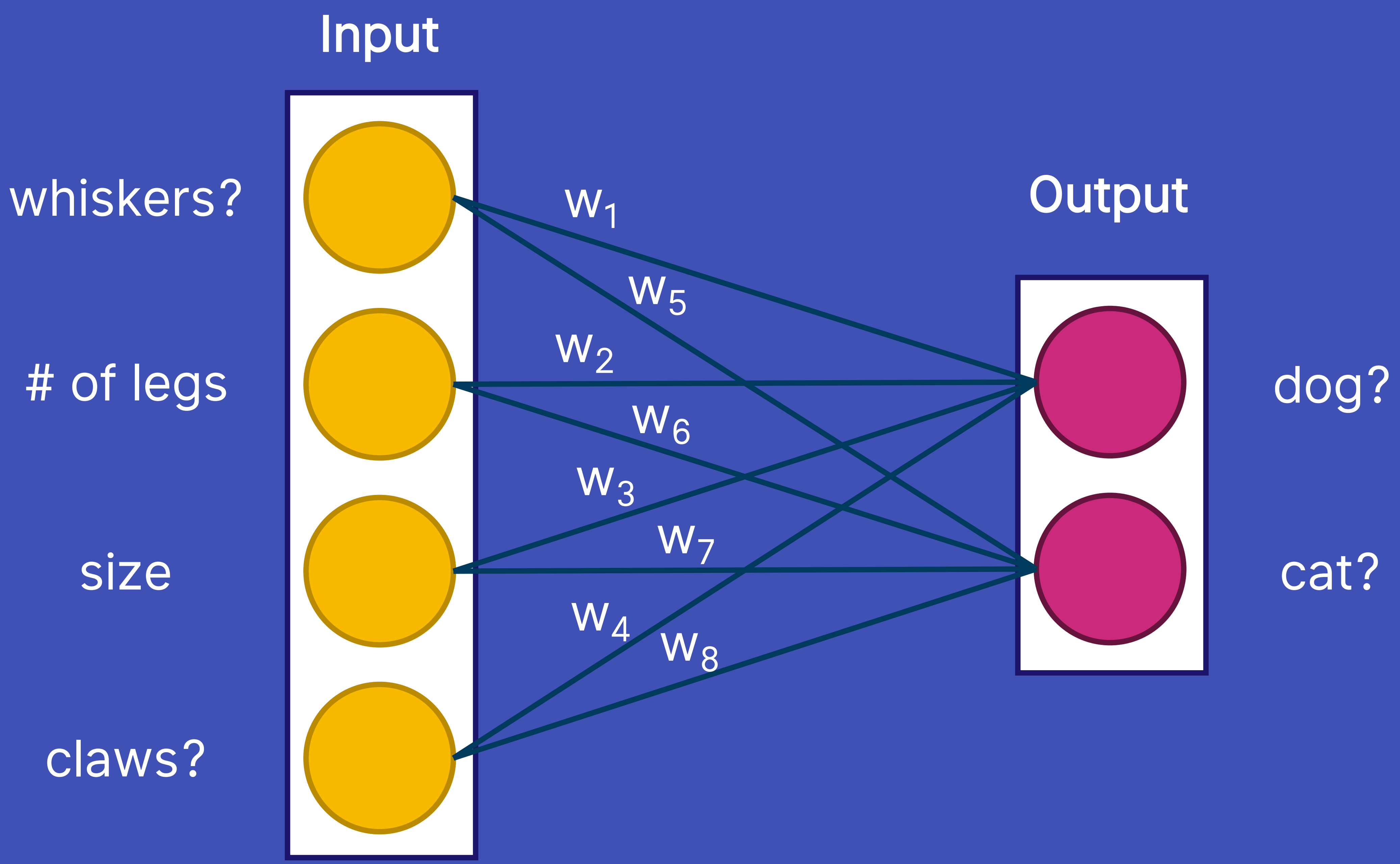
Fully-Connected Layers

Fully-connected layers are computations where every input neuron is connected to every output neuron

We transform the input by using weights to transform it into the output

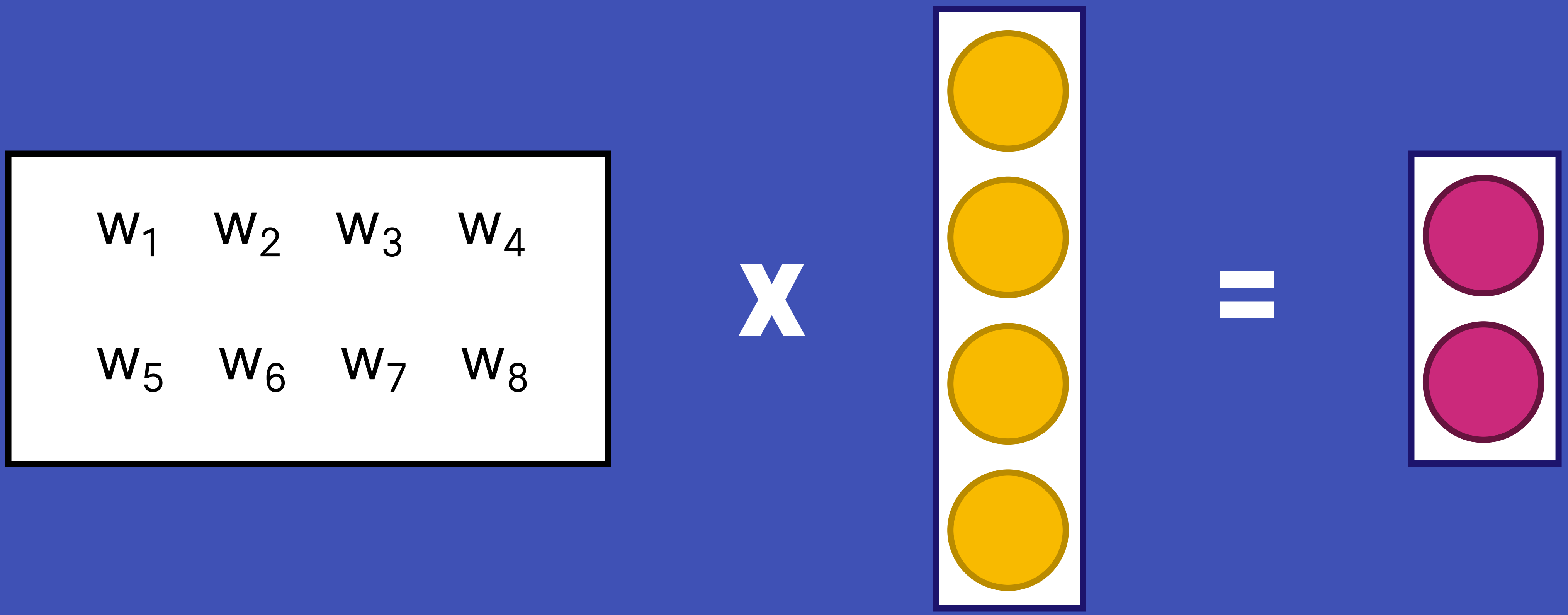
The output will have a different shape than the input

Fully-Connected Network



Student AI Group

FC Net as Matrix-Vec Mult



FORWARD PASS

Probability Distribution

The output of our network is arbitrarily scaled

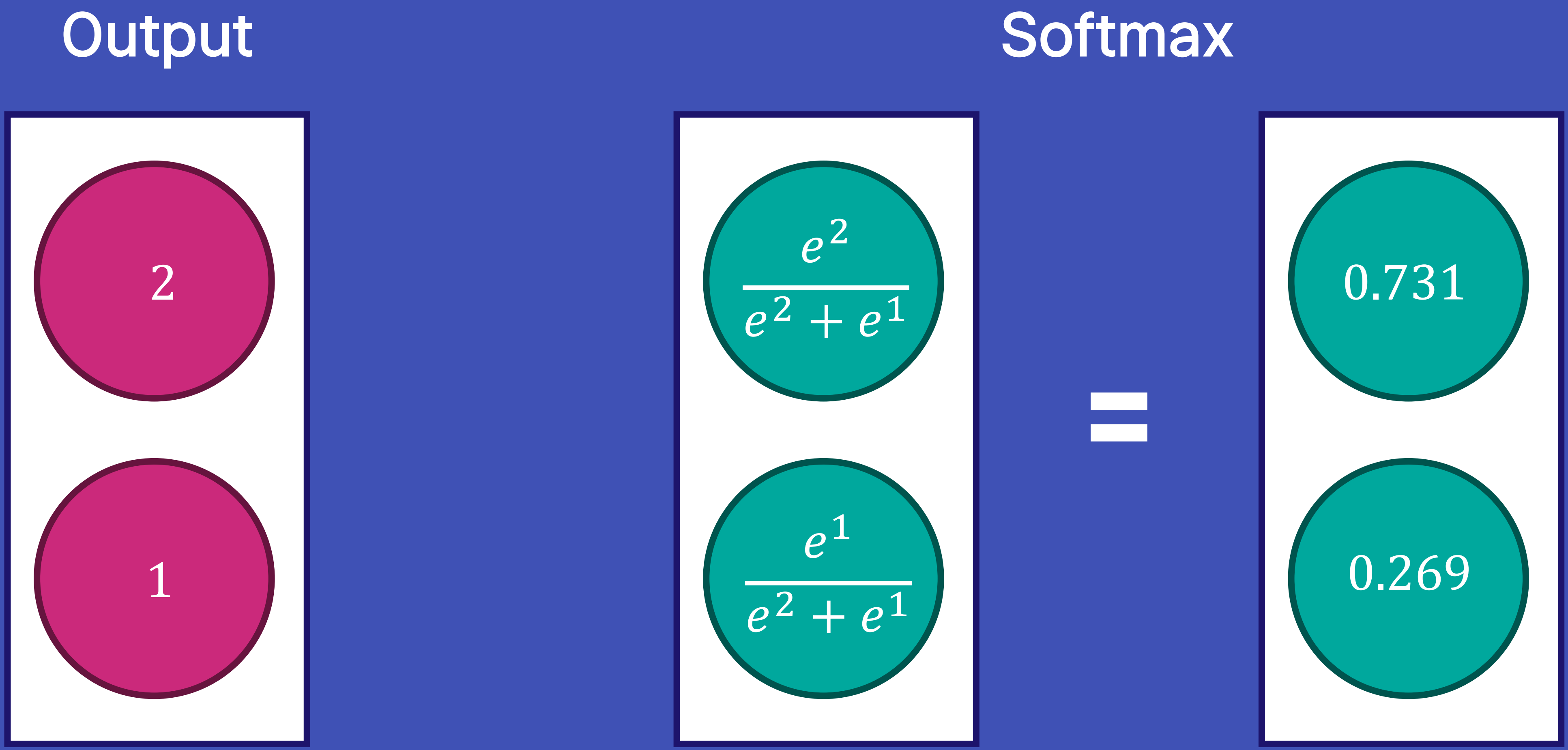
We want to turn the arbitrary scale into a proper probability distribution

We do so by using the softmax calculation:

$$p_i = \frac{e^{o_i}}{\sum_j e^{o_j}}$$

Student AI Group

Softmax Layer



Student AI Group

FORWARD PASS

Loss Value

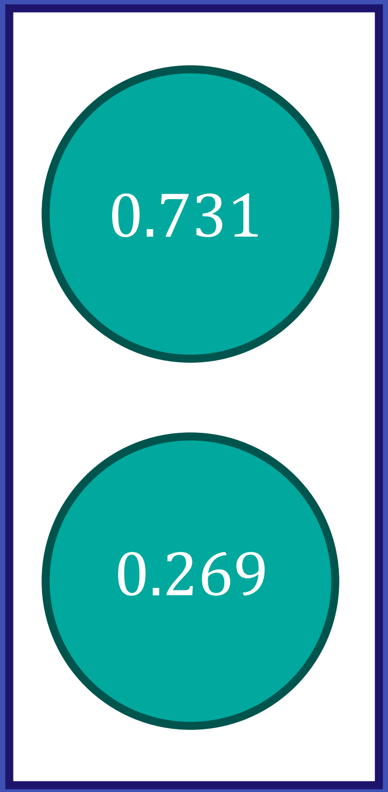
The network needs an "objective" to work towards—being accurate

We use the targets to figure out how "incorrect" the network was

The loss metric we use is cross-entropy loss:

$$L = -\log(p_c)$$

Softmax Layer



Loss

$$L = -\log(0.731) = 0.313$$

Student AI Group

FORWARD PASS

Data Splitting

We are interested in how our model does on unseen data above all

We can split data into training and test—but if we optimize on test, test becomes “indirectly” seen

Instead, we split into train, validation, and test; we run test only once

EXERCISE

Code Up the Forward Pass

Split the dataset into train, validation, and test

Initialize the parameters to random

Create the linear and softmax cross-entropy layers

Student AI Group

Backward Pass

Student
AI
Group

BACKWARD PASS

Backpropagation

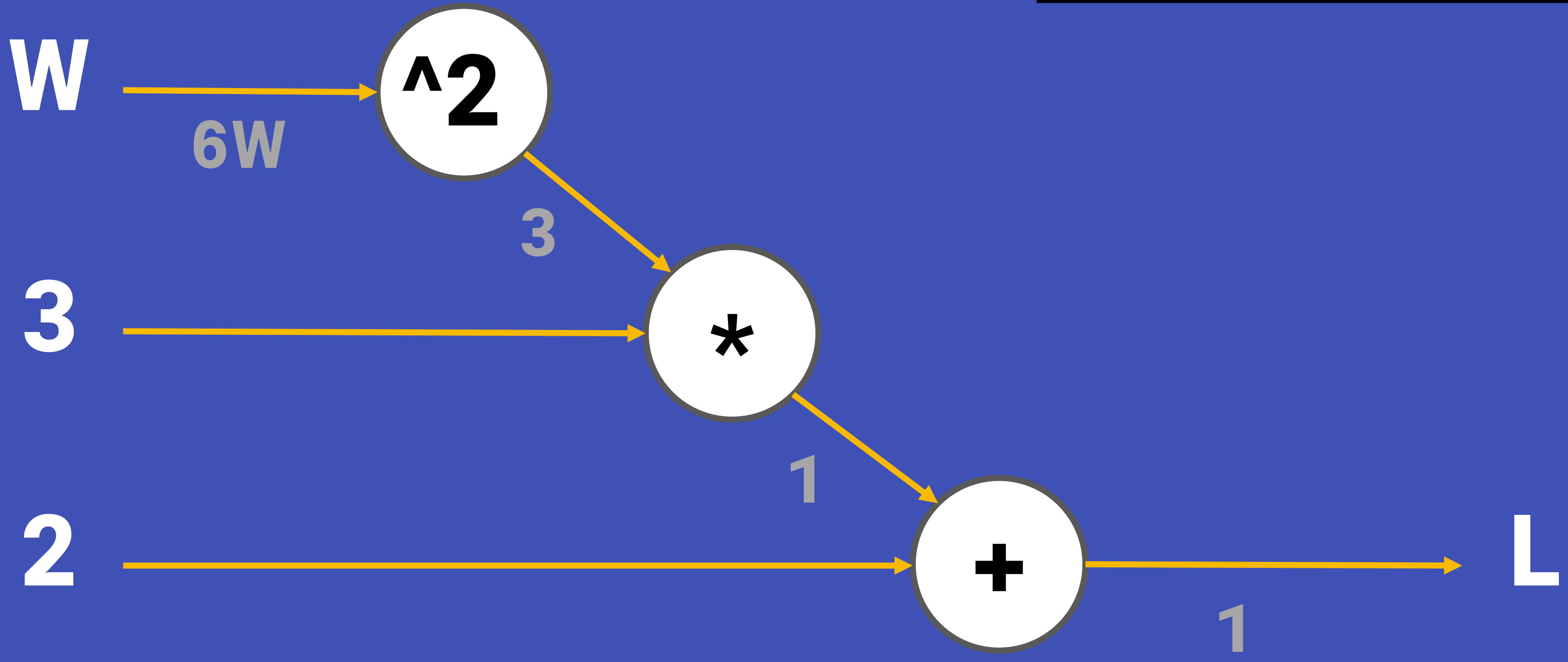
Each computation is represented as a node in a computational graph

Gradients are computed at each node of the graph

Chain rule is applied recursively to get total gradients on each weight

Simple Backprop

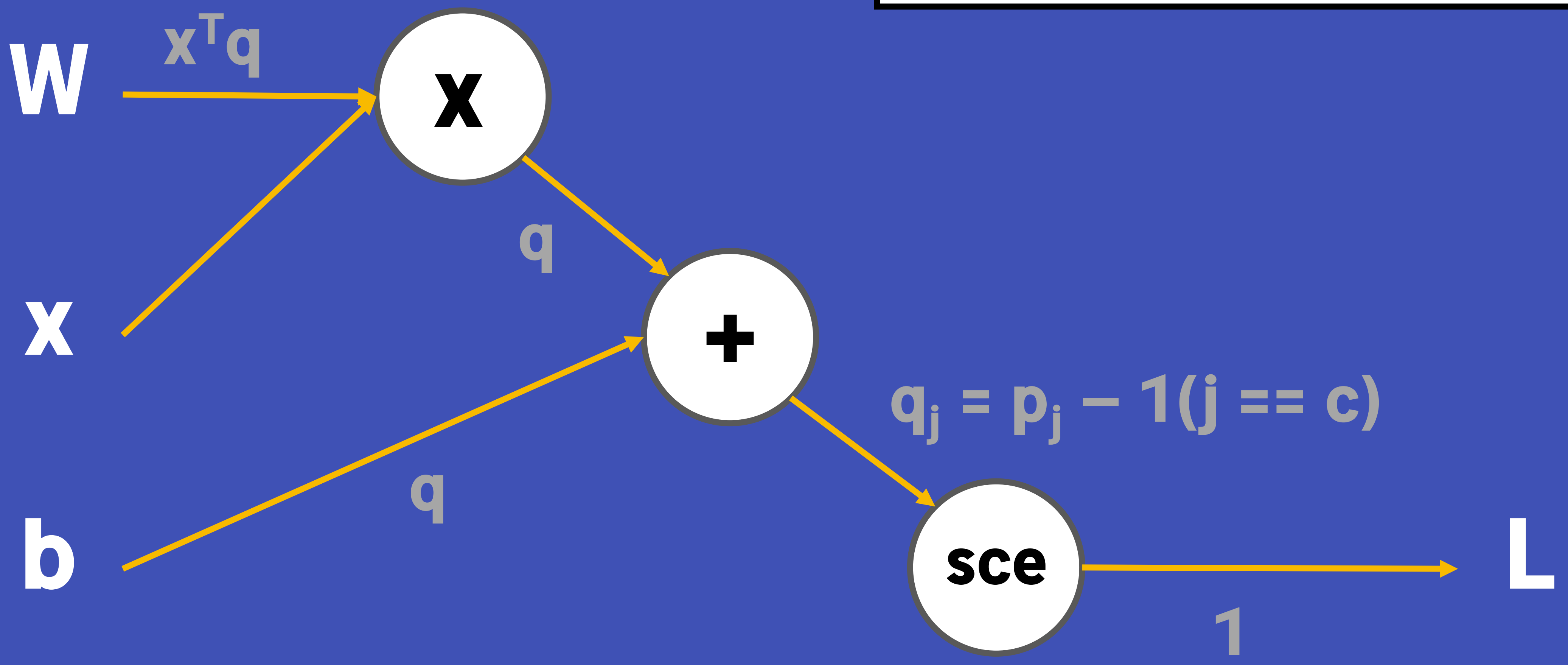
$$\boxed{L = 3W^2 + 2}$$



Student AI Group

Multidimensional Backprop

$$\boxed{L = \mathrm{sce}(Wx + b)}$$



EXERCISE

Code Up the Backward Pass

Create the backward pass through the linear layer

Create the backward pass through the softmax cross-entropy layer

Training

Student
AI
Group

TRAINING

Weight Update

Once we find gradients, we must change each weight in that direction

We can use a update rules to accomplish this; the simplest is SGD:

$$W_i := W_i - \alpha\left(\frac{\partial L}{\partial W_i}\right)$$

Student AI Group

Training Iterations

Each time we move the weights in the direction of the negative gradient, we get closer to a good value for the weights

We want to do this many times until we arrive at good weight values (i.e. converge)

We also want to periodically check the network's accuracy on the training and validation sets

EXERCISE

Code Up the Training Regime

Code the training iteration loop

Implement SGD weight updates

Periodically log the training and validation losses

Conclusion

Student
AI
Group