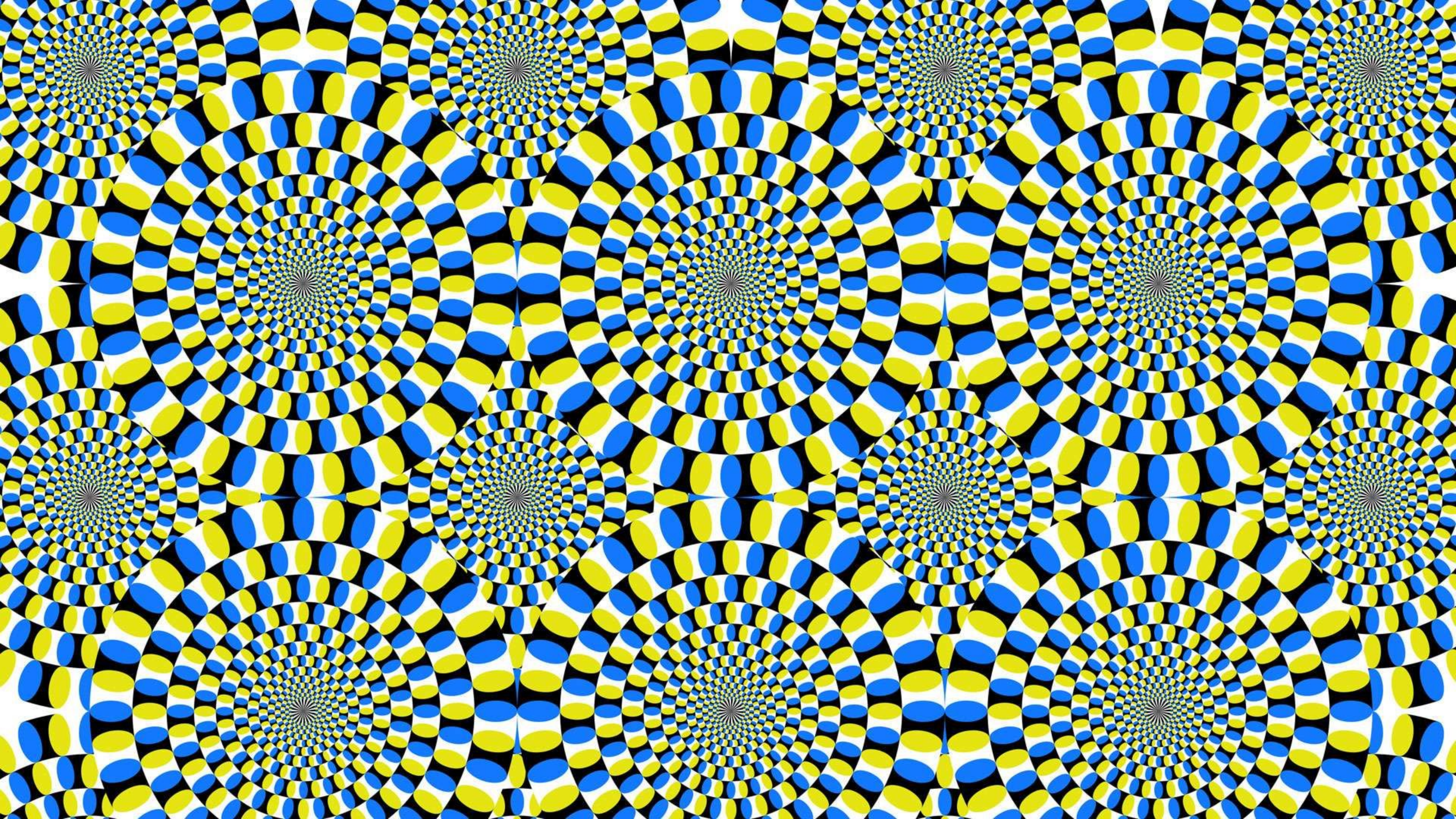
Adversarial Attacks in the Real World



Mathis Hammel

@MathisHammel

Independent consultant

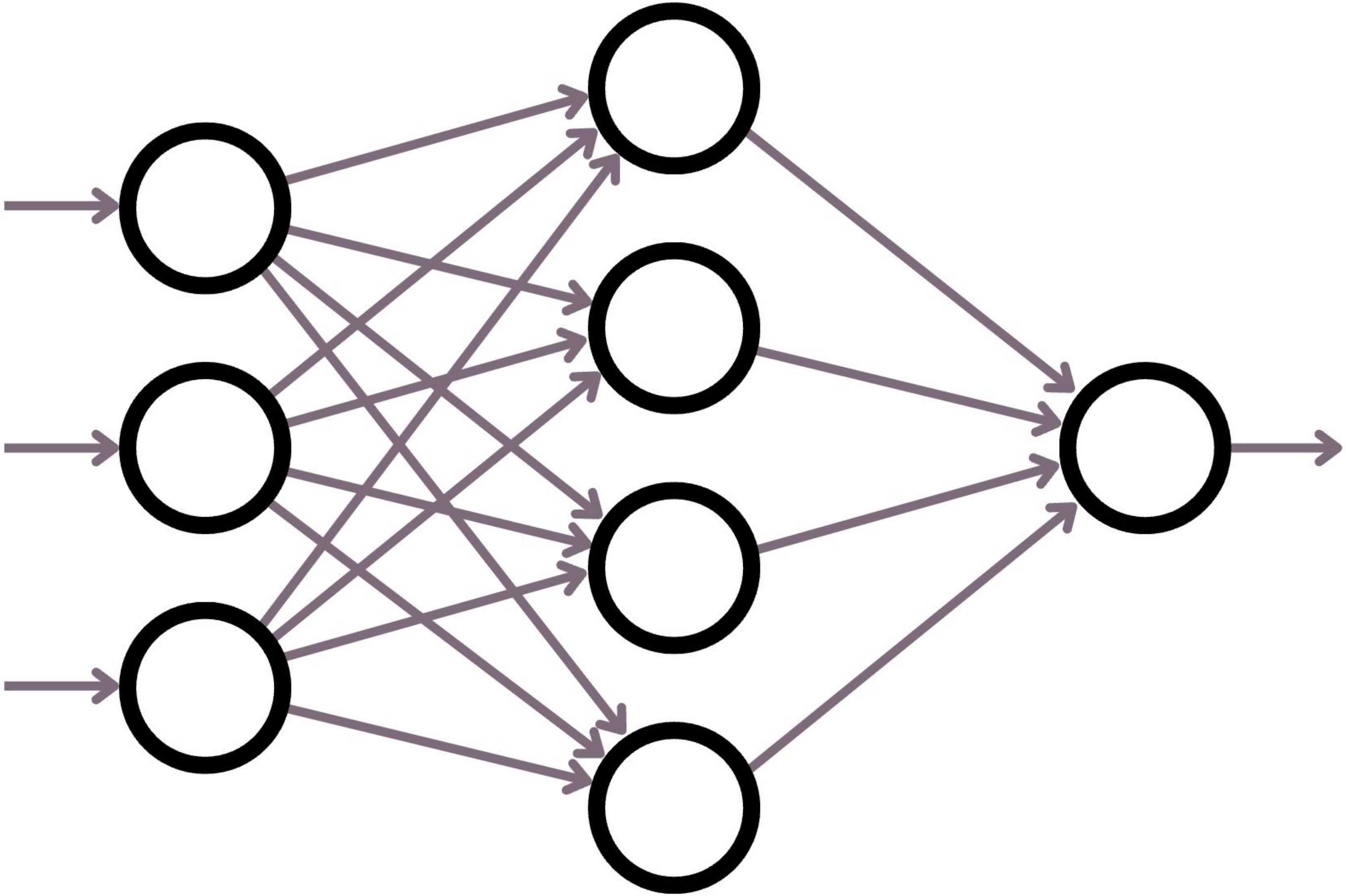
- Technical competitions
- Algorithms
- Cybersecurity R&D

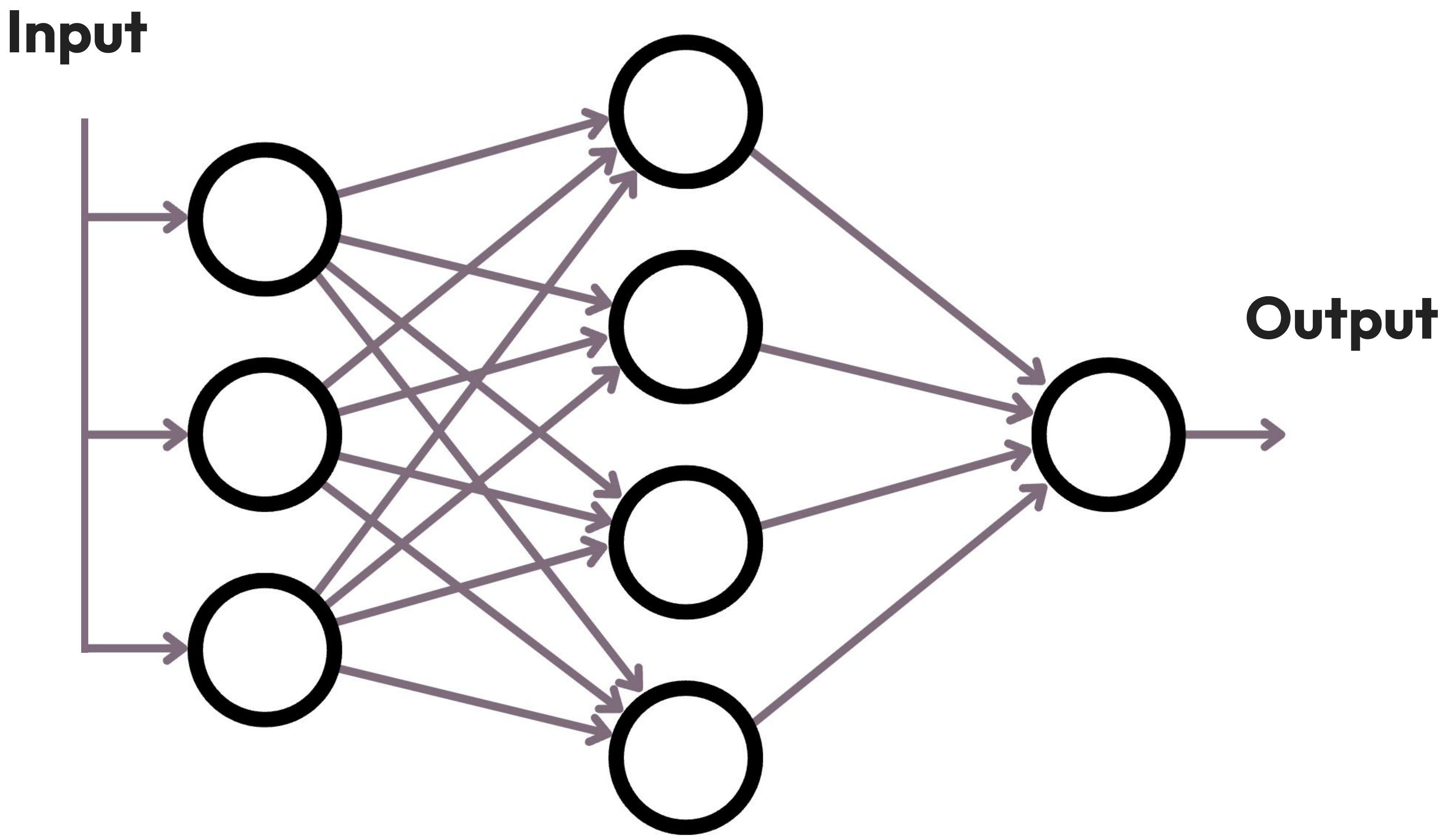
Experts 2024 Machine Learning

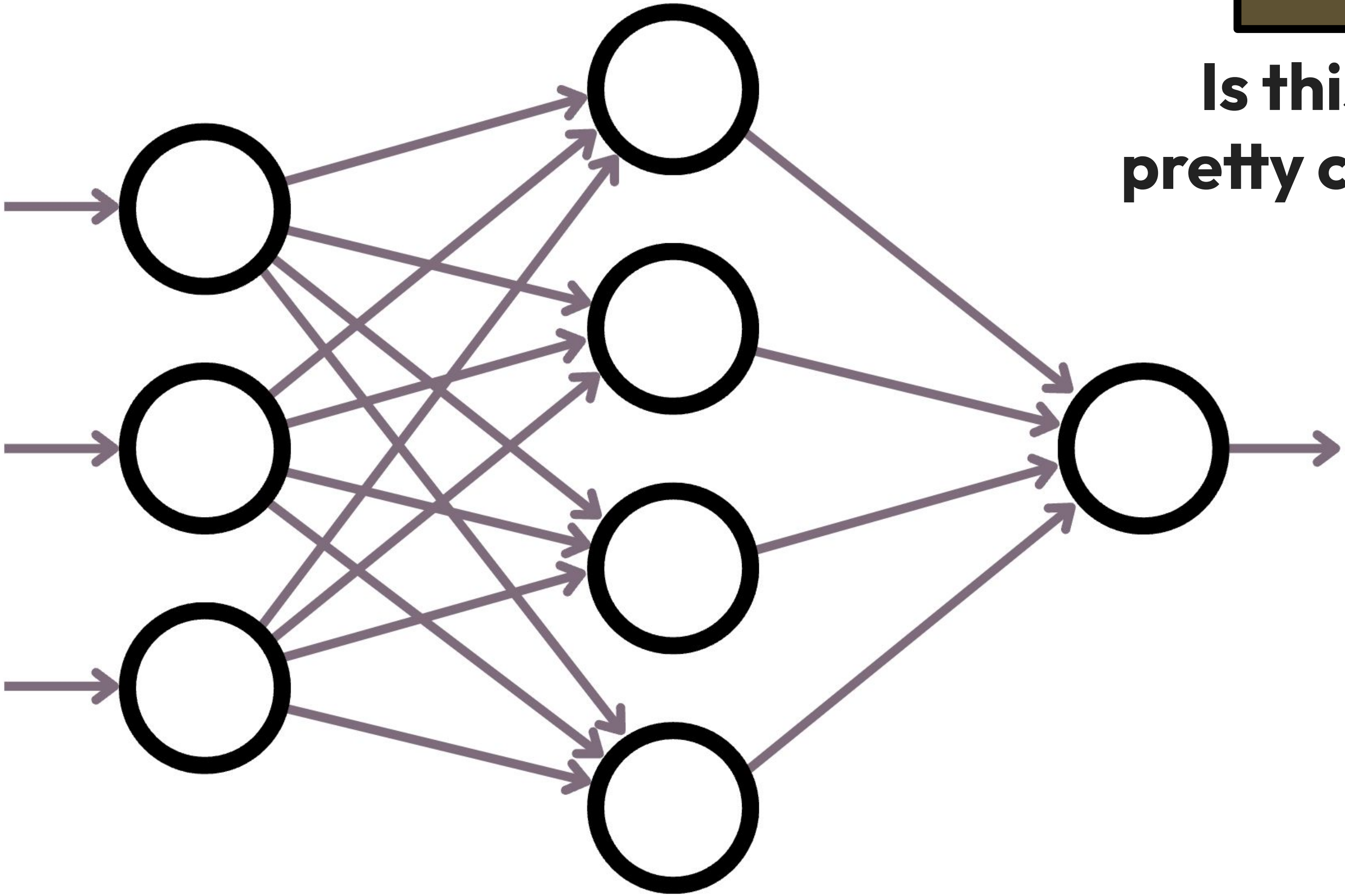
MVP Microsoft® Most Valuable Professional

Let’s start with the basics.

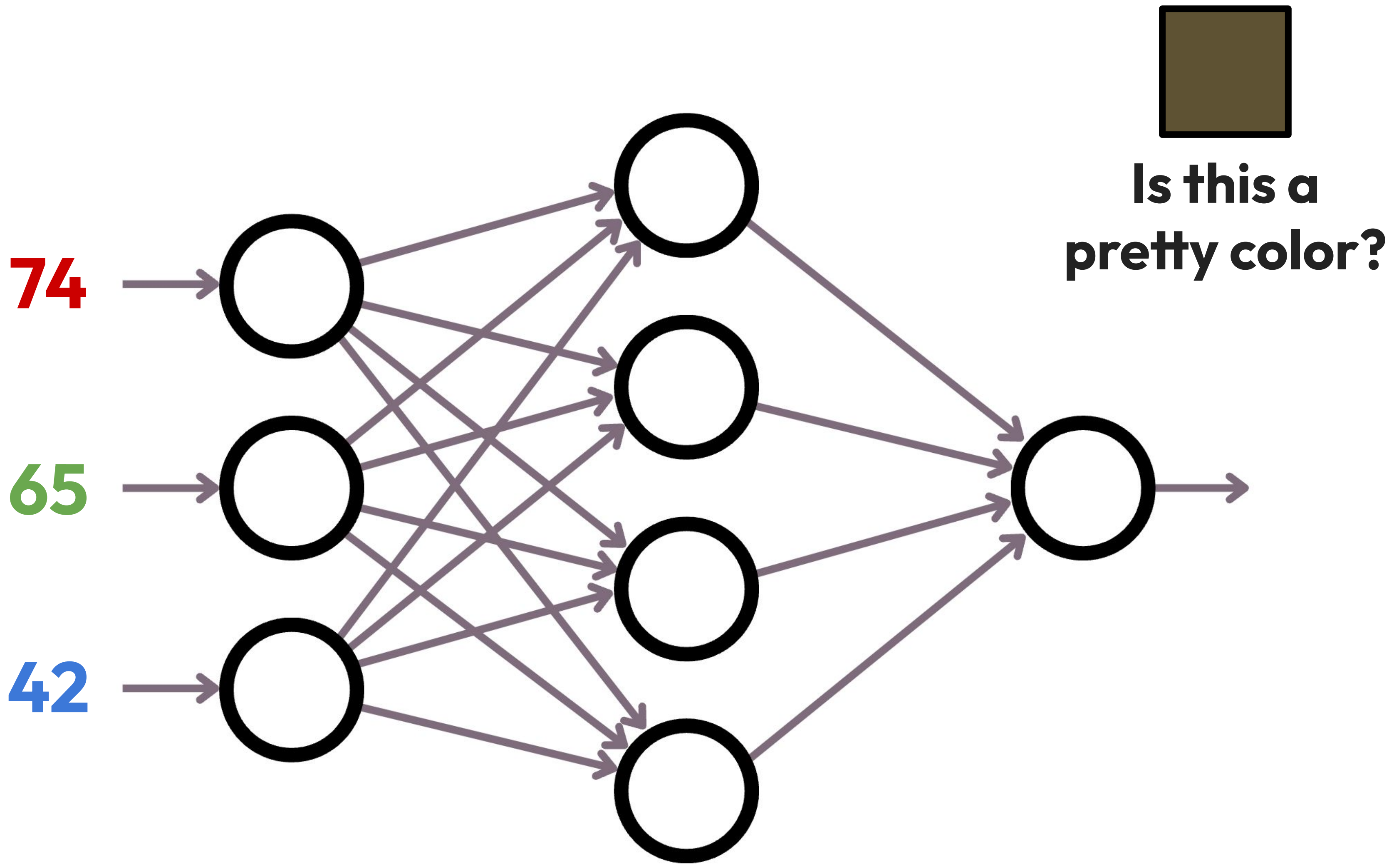
What *really* is a neural network?



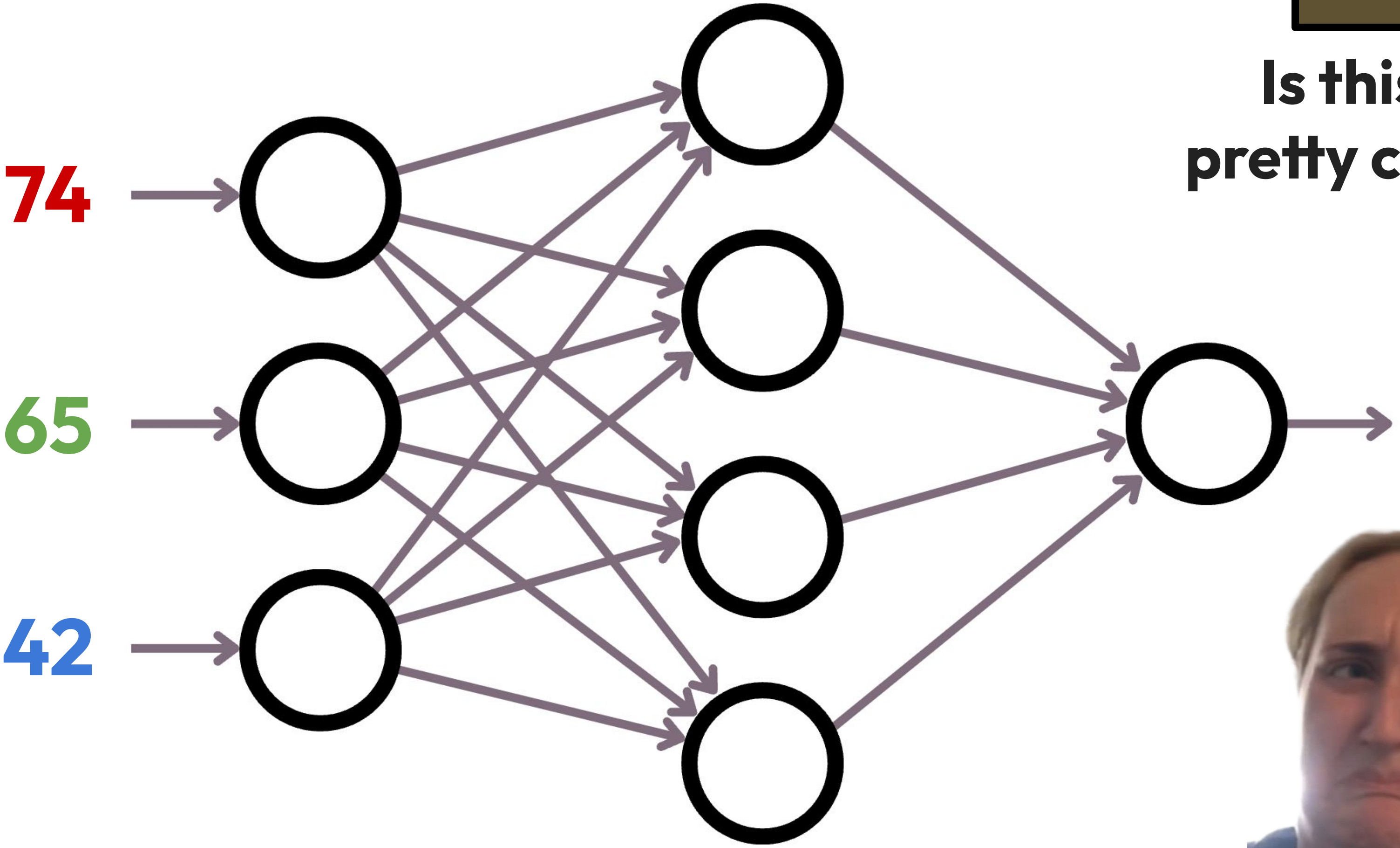

Input
Output



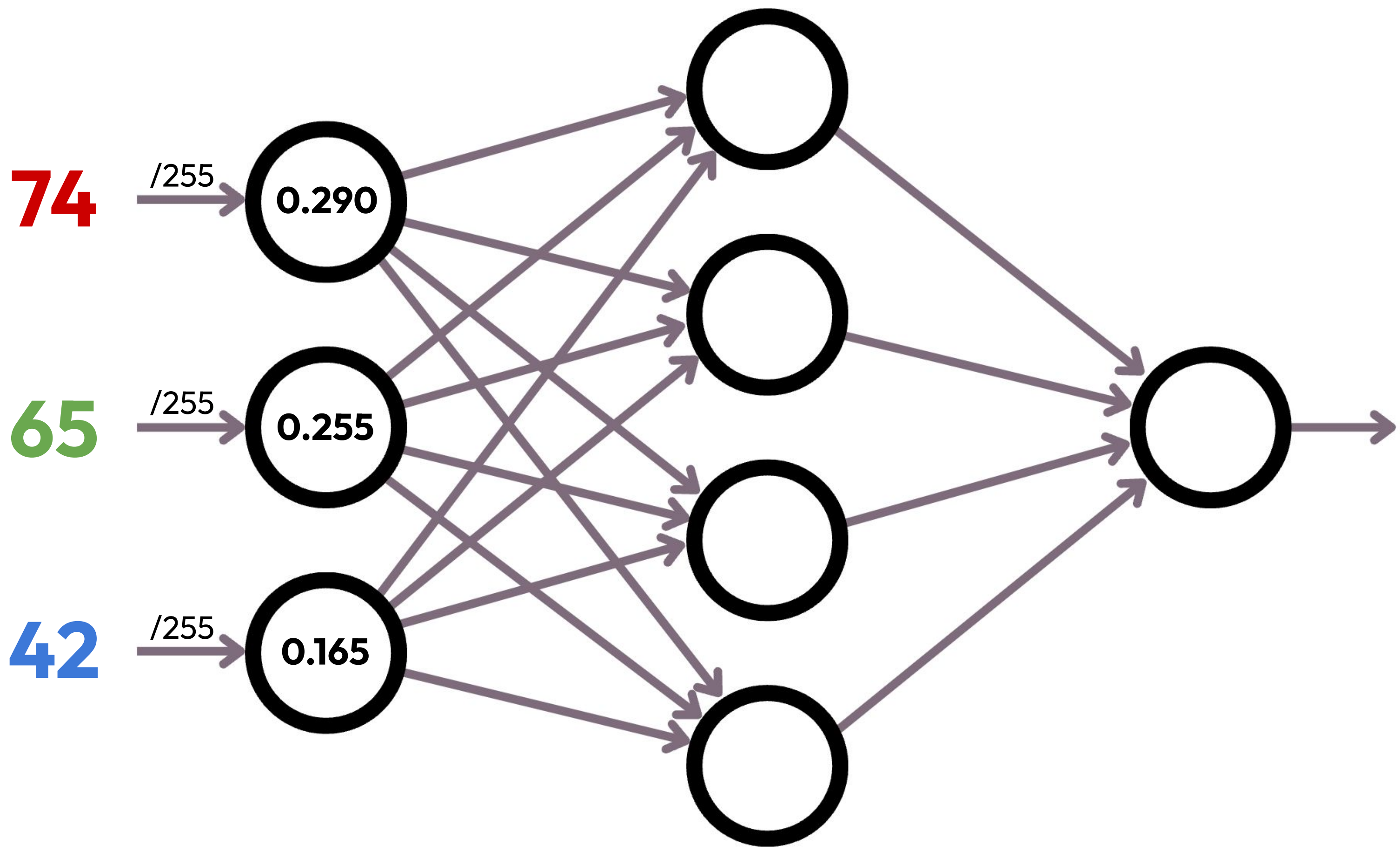
Is this a
pretty color?

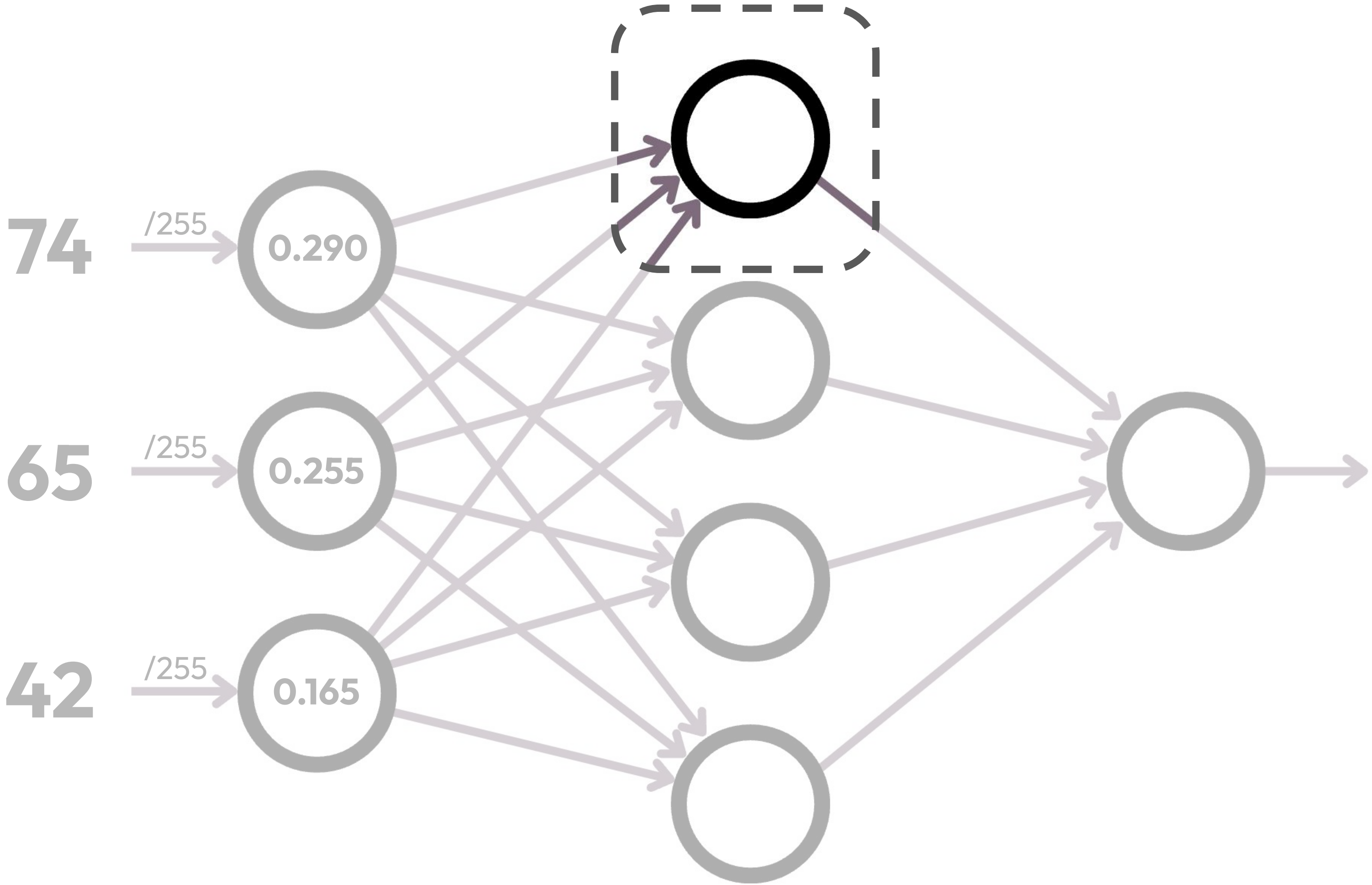

74
65
42
Is this a
pretty color?

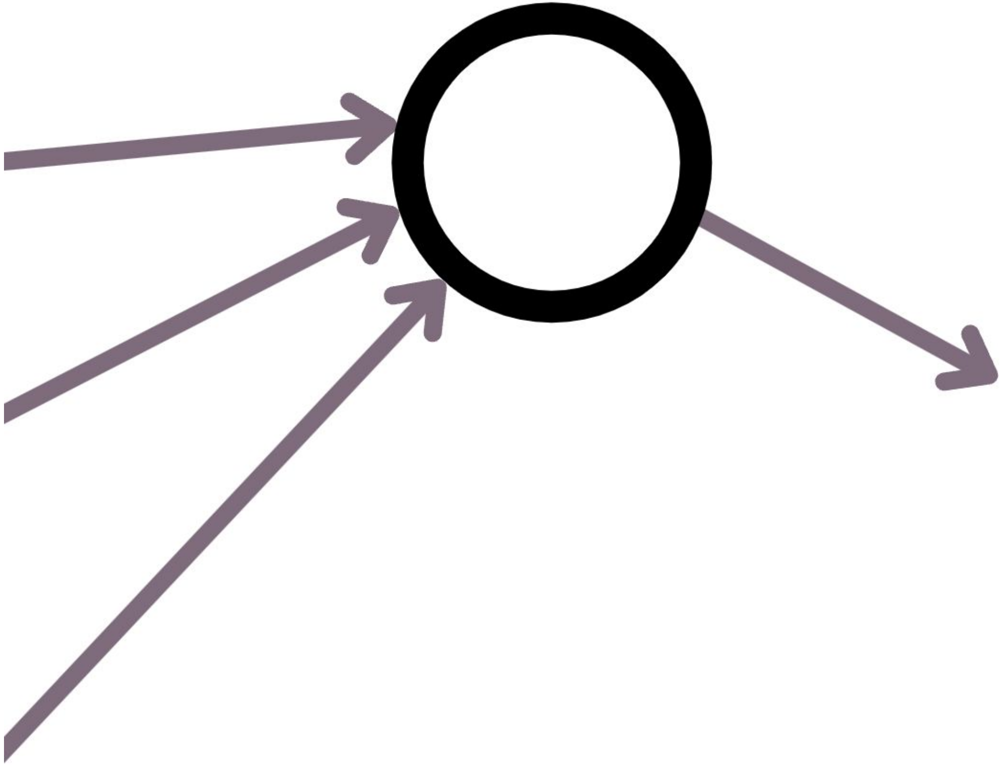
Is this a pretty color?



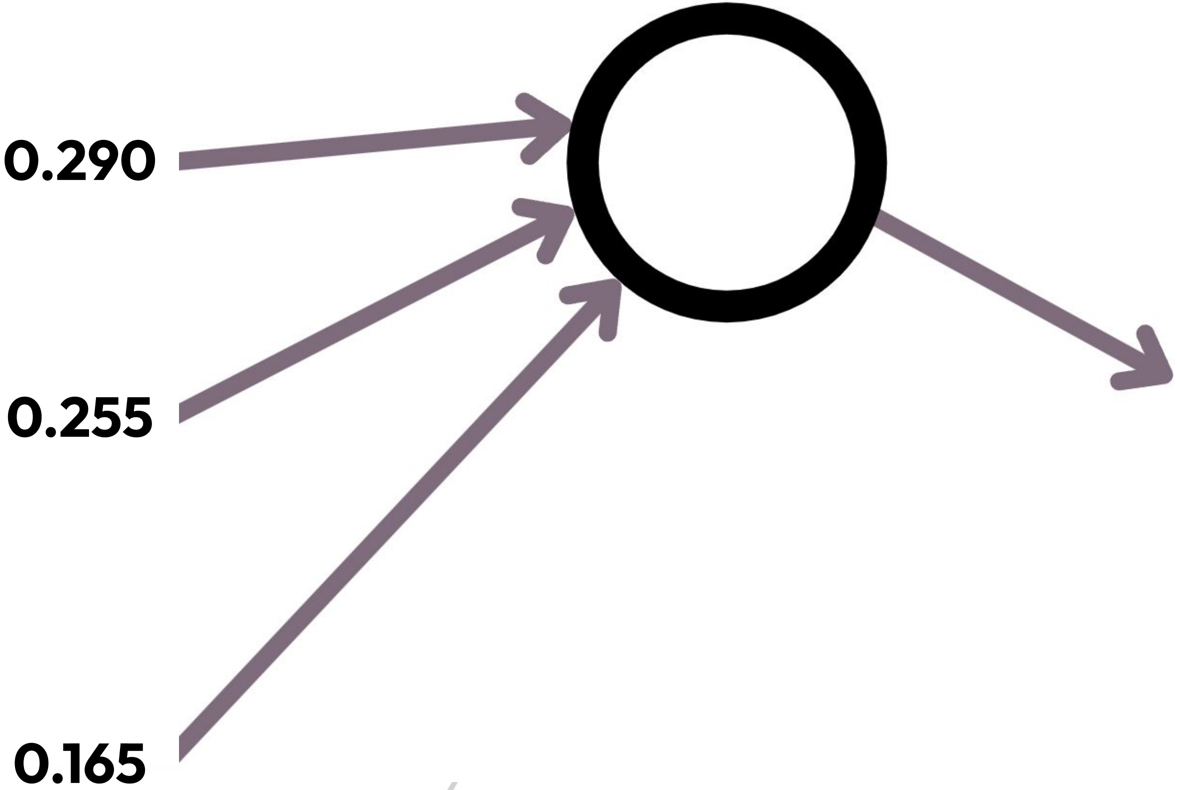
What really happens inside?


74
/255
0.290
65
/255
0.255
42
/255
0.165

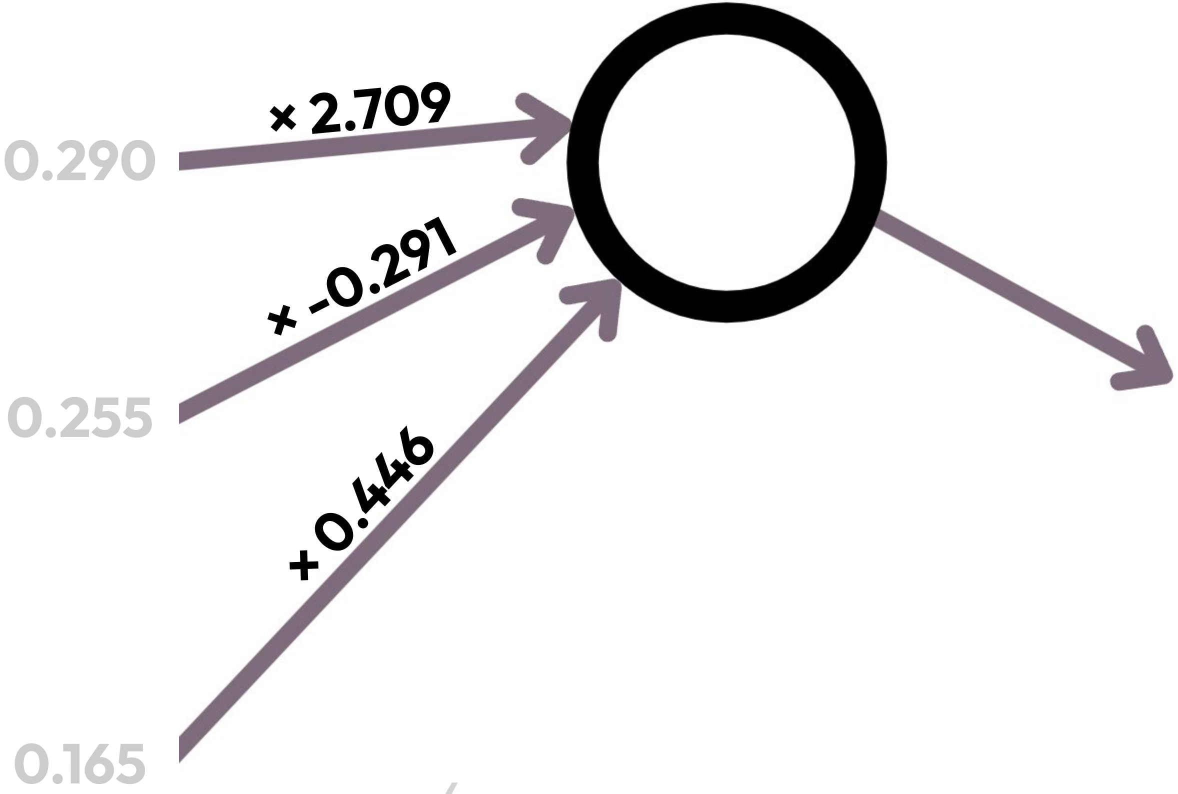

74
/255
0.290
65
/255
0.255
42
/255
0.165



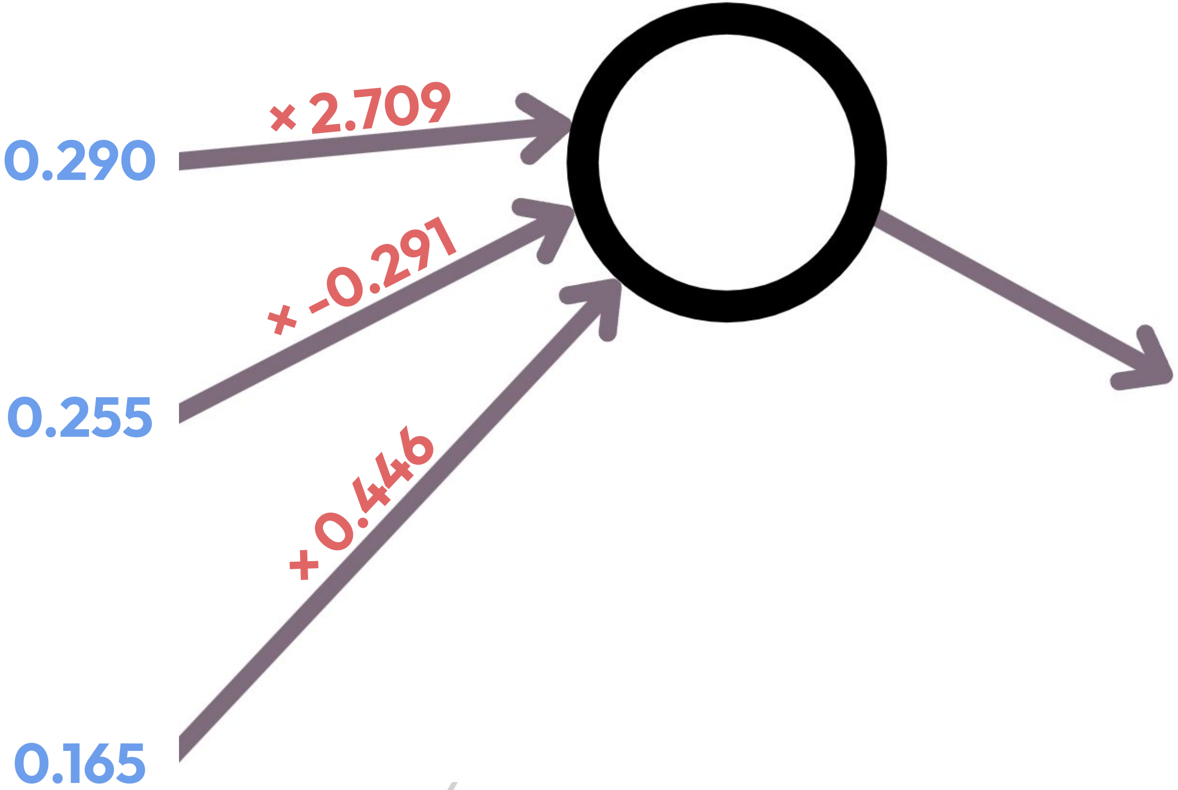
$$output = f\left(bias + \sum_i input_i \cdot weight_i\right)$$



$$output = f\left(bias + \sum_i input_i \cdot weight_i\right)$$

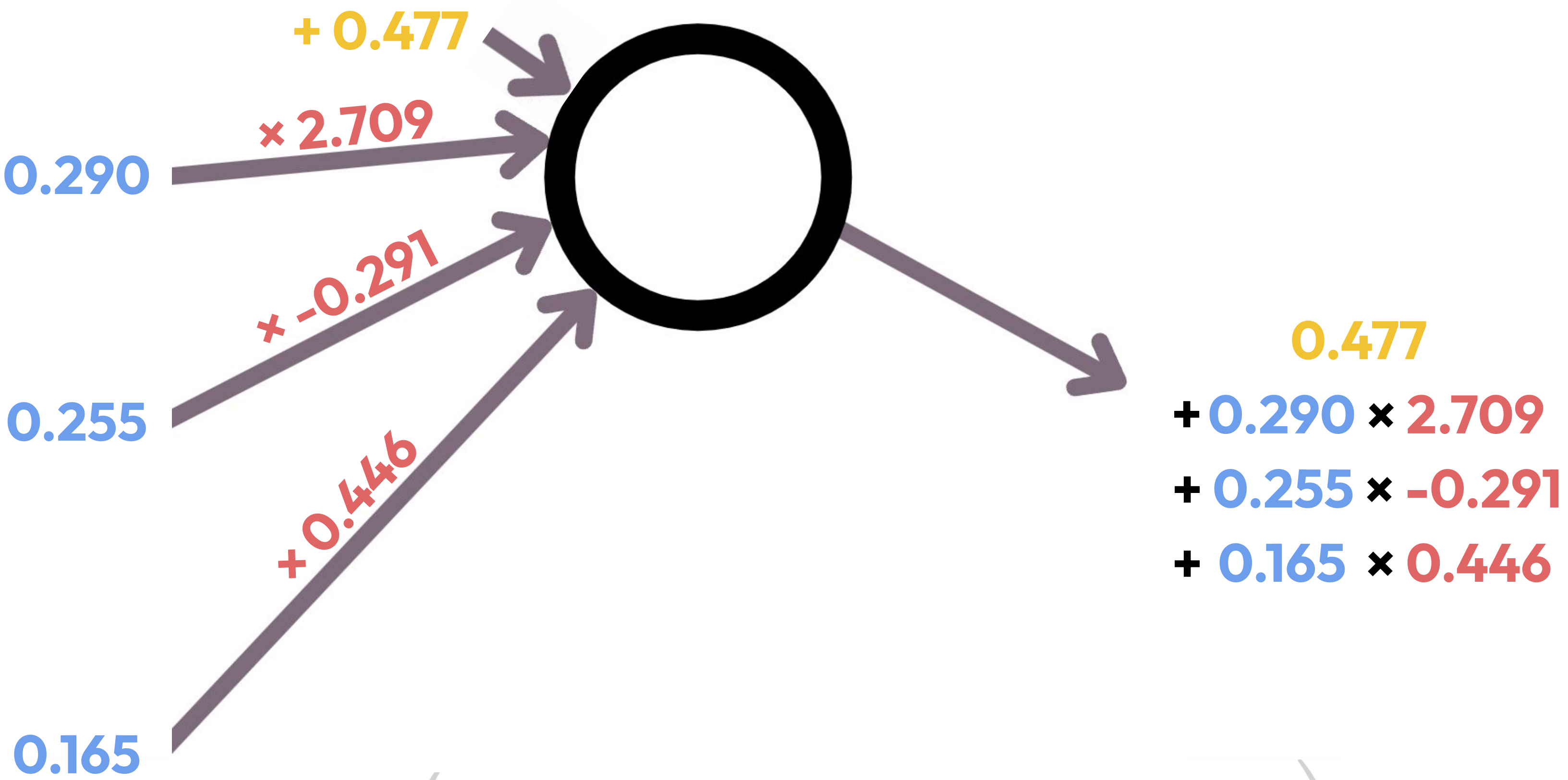


$$output = f\left(bias + \sum_i input_i \cdot weight_i\right)$$

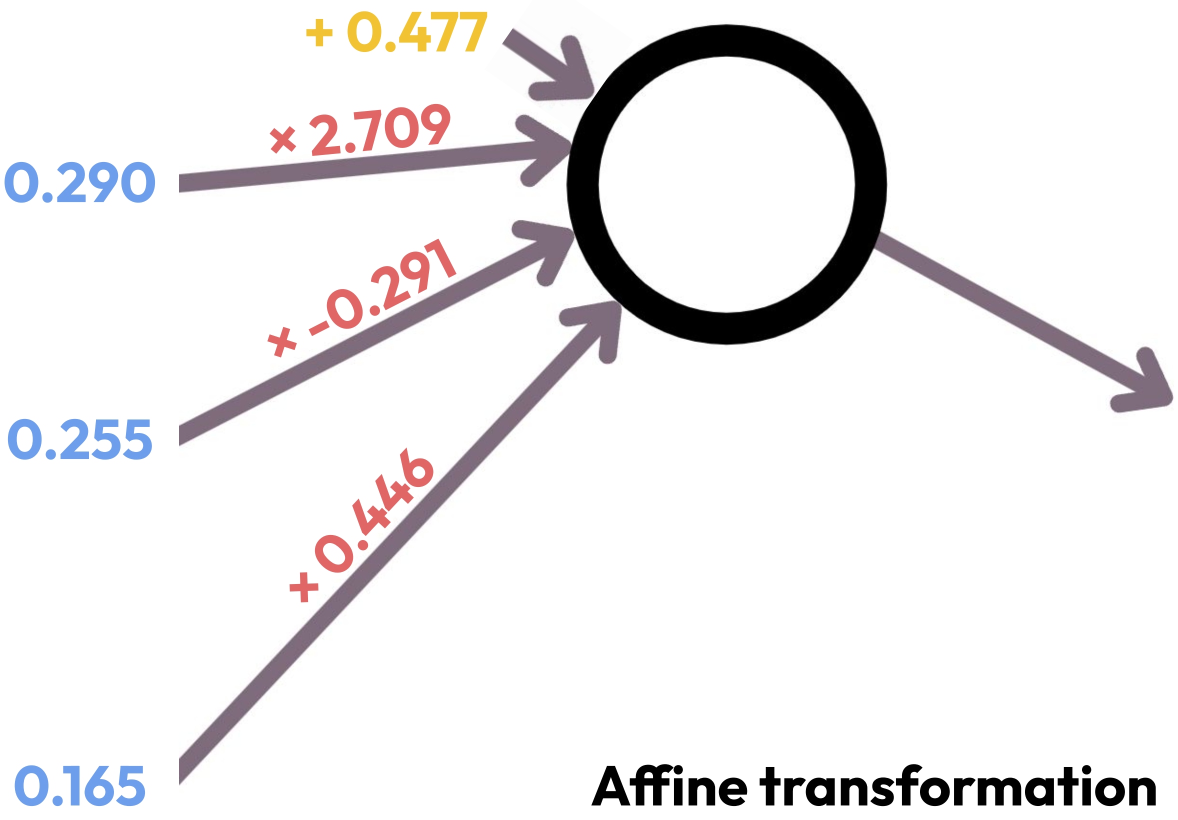


$$
\begin{aligned}
&0.290 \times 2.709 \\
+\ &0.255 \times -0.291 \\
+\ &0.165 \times 0.446
\end{aligned}
$$

$$output = f\left(bias + \sum_i \underline{input_i} \cdot \underline{weight_i}\right)$$



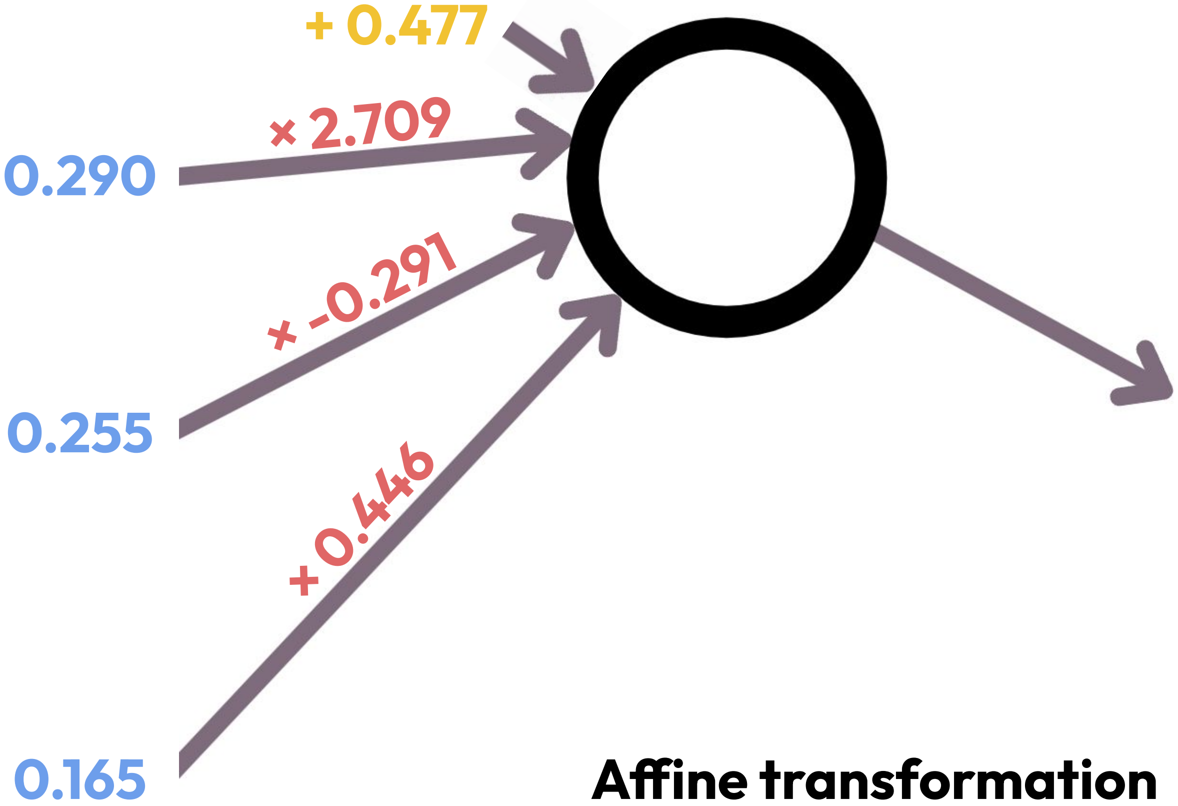
$$output = f\left(\underline{bias} + \sum_i \underline{input_i} \cdot \underline{weight_i}\right)$$



$$
\begin{aligned}
&0.477\\
&+ 0.290 \times 2.709\\
&+ 0.255 \times -0.291\\
&+ 0.165 \times 0.446
\end{aligned}
$$

Affine transformation

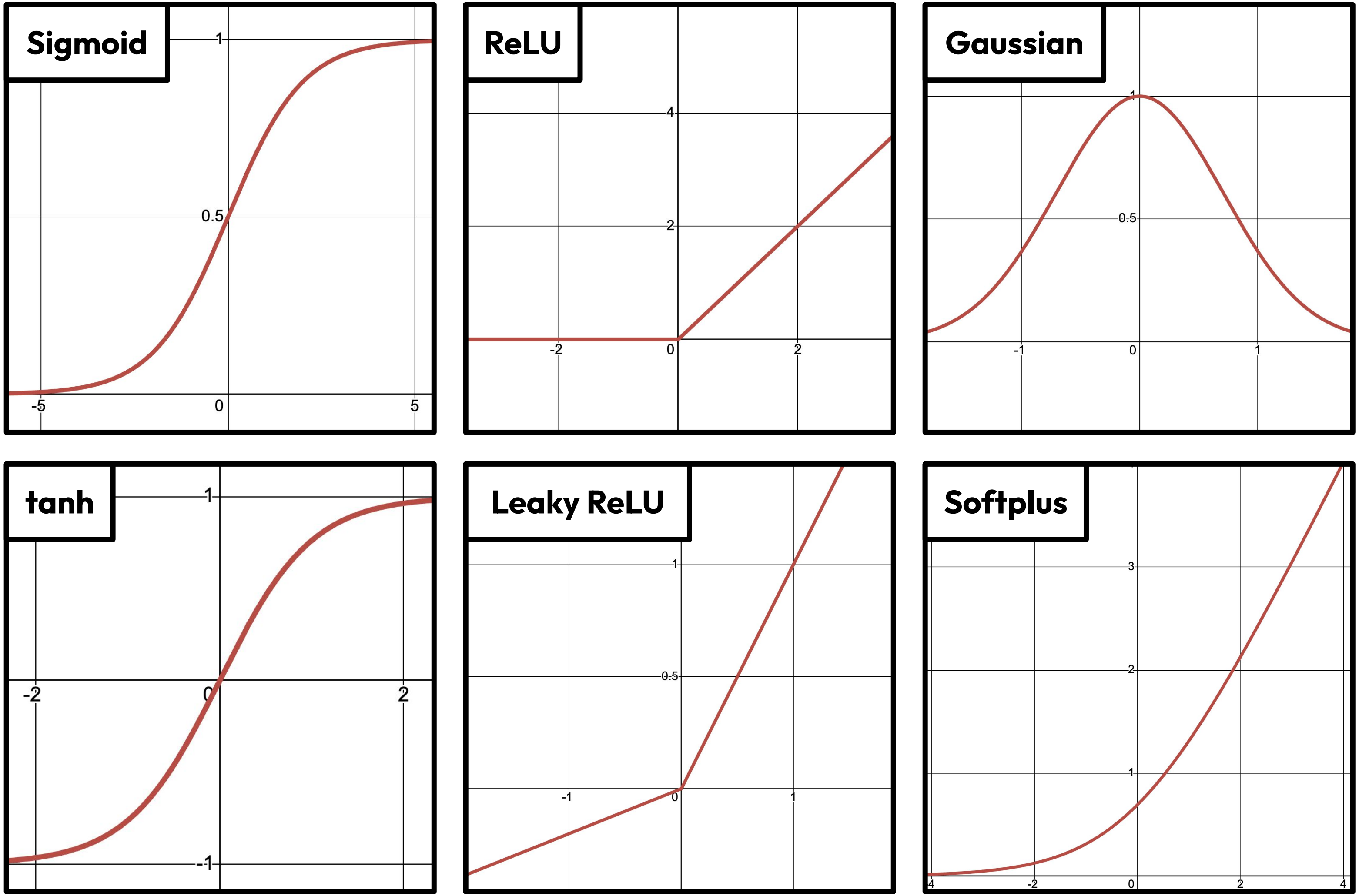
is <u>not</u> a universal approximator

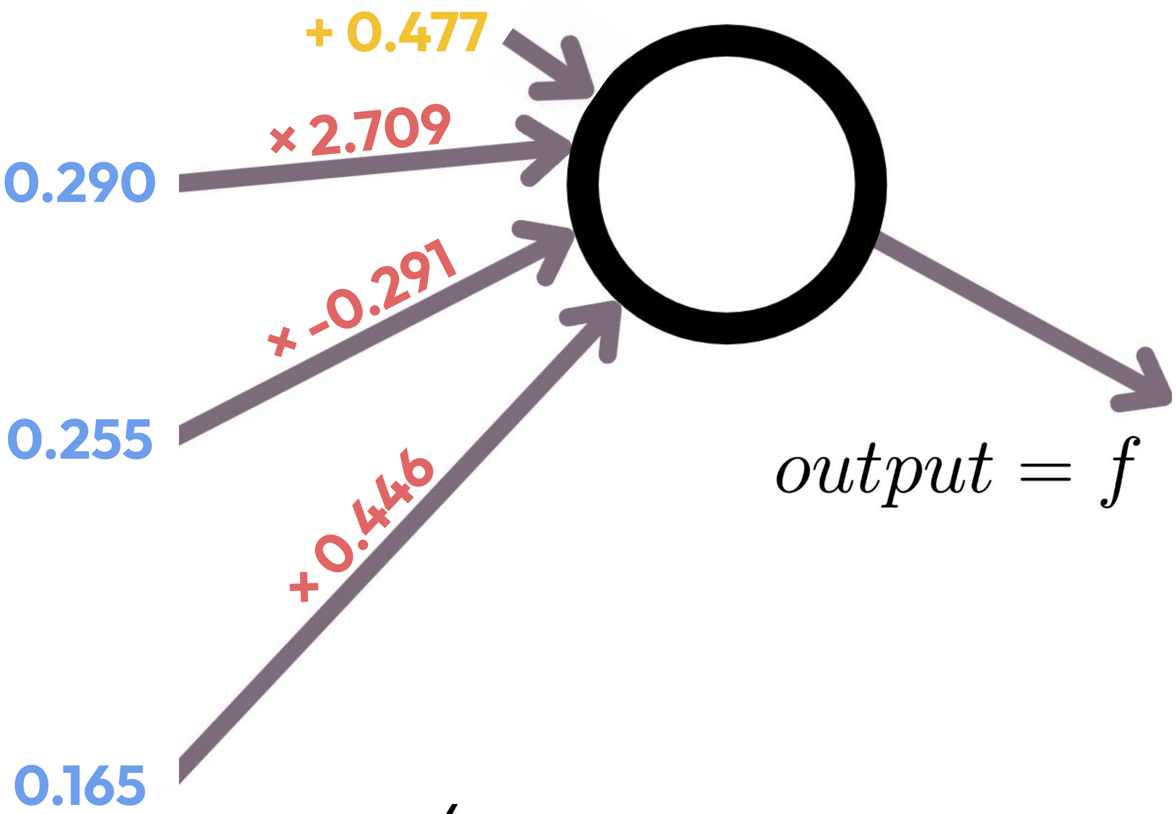


$$0.477 + 0.290 \times 2.709 + 0.255 \times -0.291 + 0.165 \times 0.446$$

Affine transformation
is <u>not</u> a universal approximator

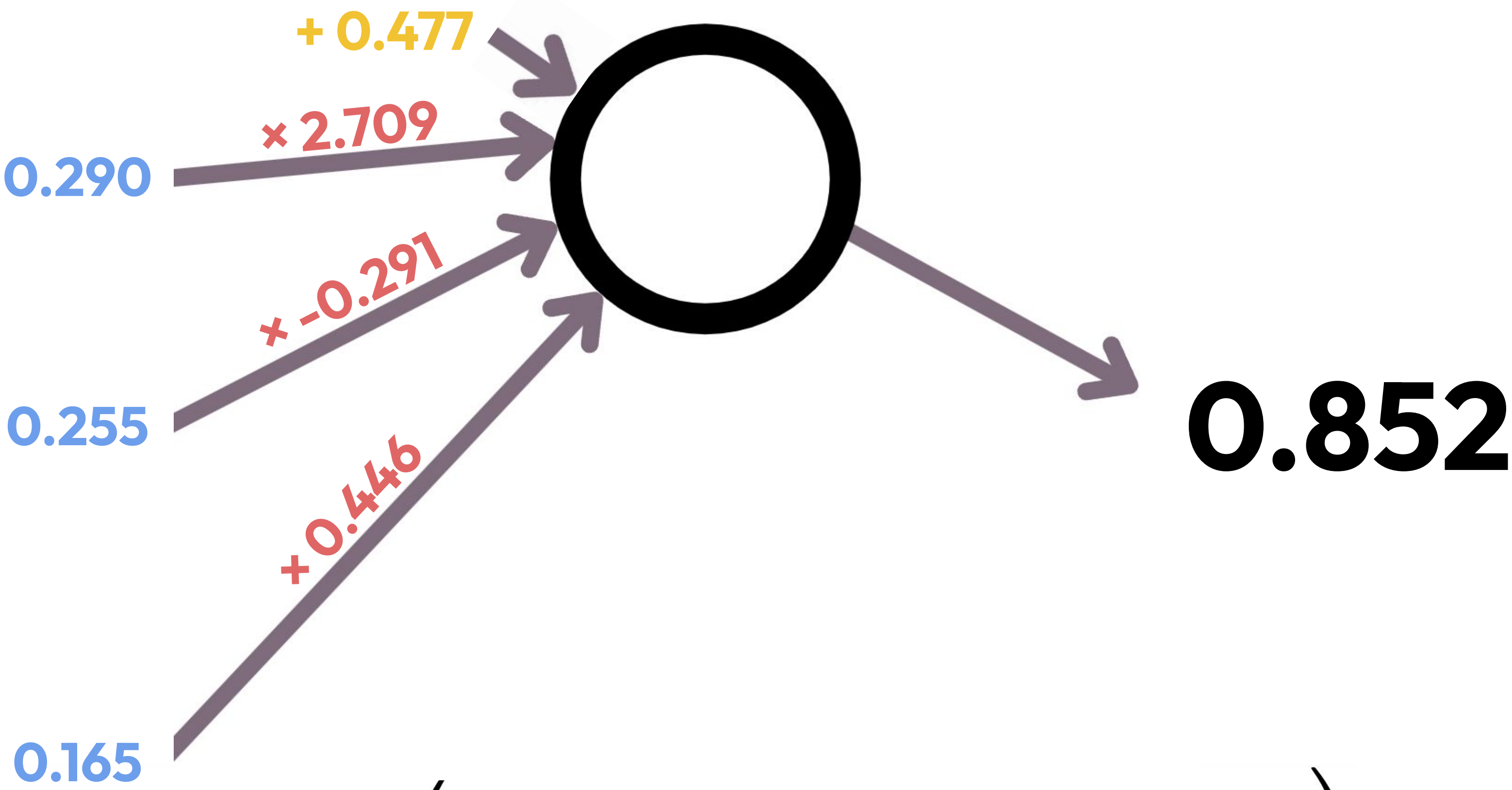
Let's add a non-linearity!


Sigmoid
1
0.5
-5
0
5
ReLU
4
2
-2
0
2
Gaussian
1
0.5
-1
0
1
tanh
1
-2
0
2
-1
Leaky ReLU
1
0.5
-1
0
1
Softplus
3
2
1
-2
0
2
4

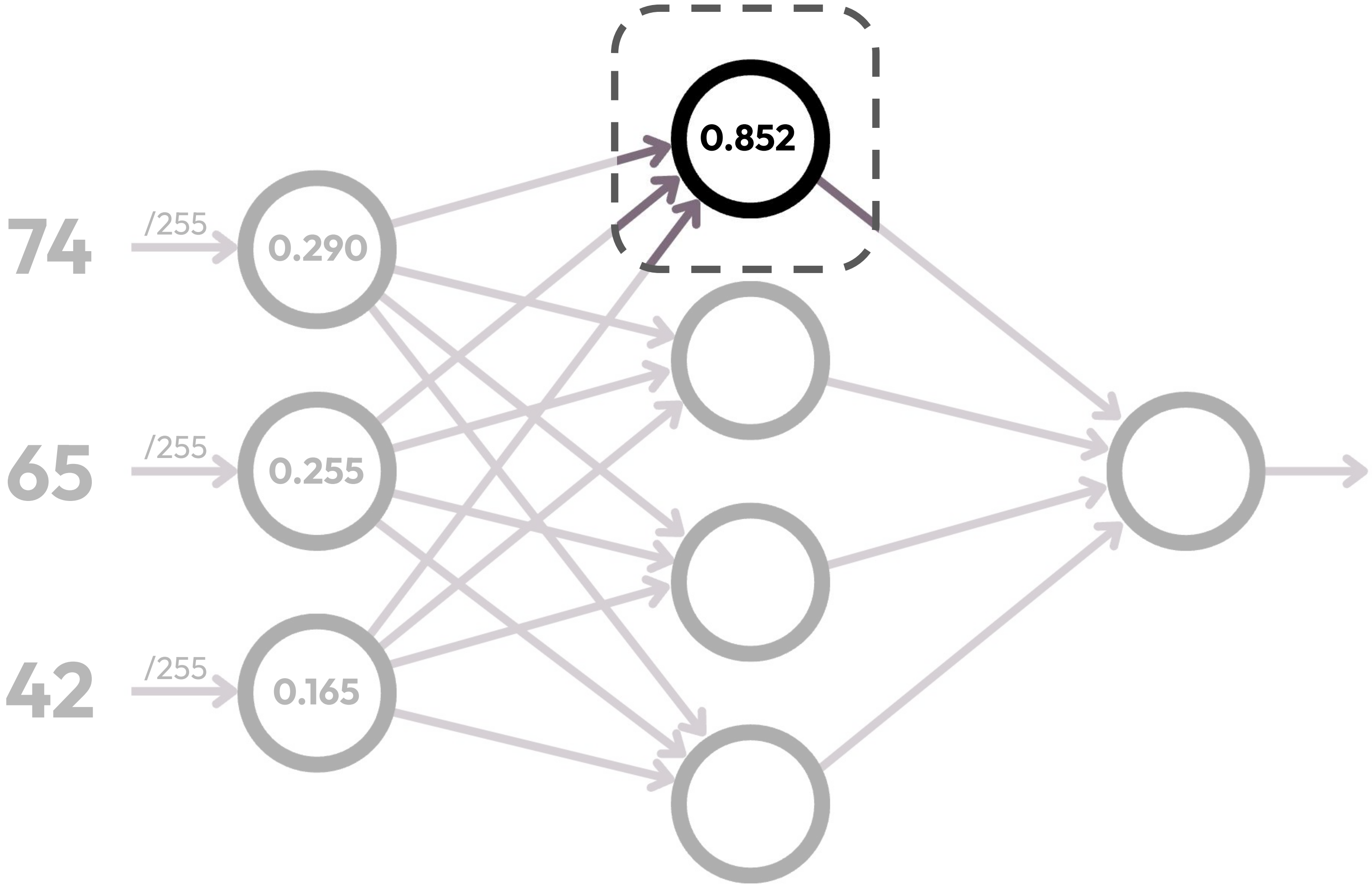


$$output = f\left(\begin{array}{c} 0.477 \\ + 0.290 \times 2.709 \\ + 0.255 \times -0.291 \\ + 0.165 \times 0.446 \end{array}\right)$$

$$output = f\left(bias + \sum_i input_i \cdot weight_i\right)$$



$$output = f\left(bias + \sum_i input_i \cdot weight_i\right)$$


74
/255
0.290
65
/255
0.255
42
/255
0.165
0.852

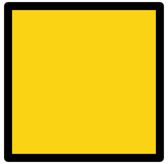

74
65
42
/255
/255
/255
0.290
0.255
0.165
0.852
0.431
-0.216
-0.191


74
/255
0.290
65
/255
0.255
42
/255
0.165
0.852
0.431
-0.216
-0.191
0.179
0.179

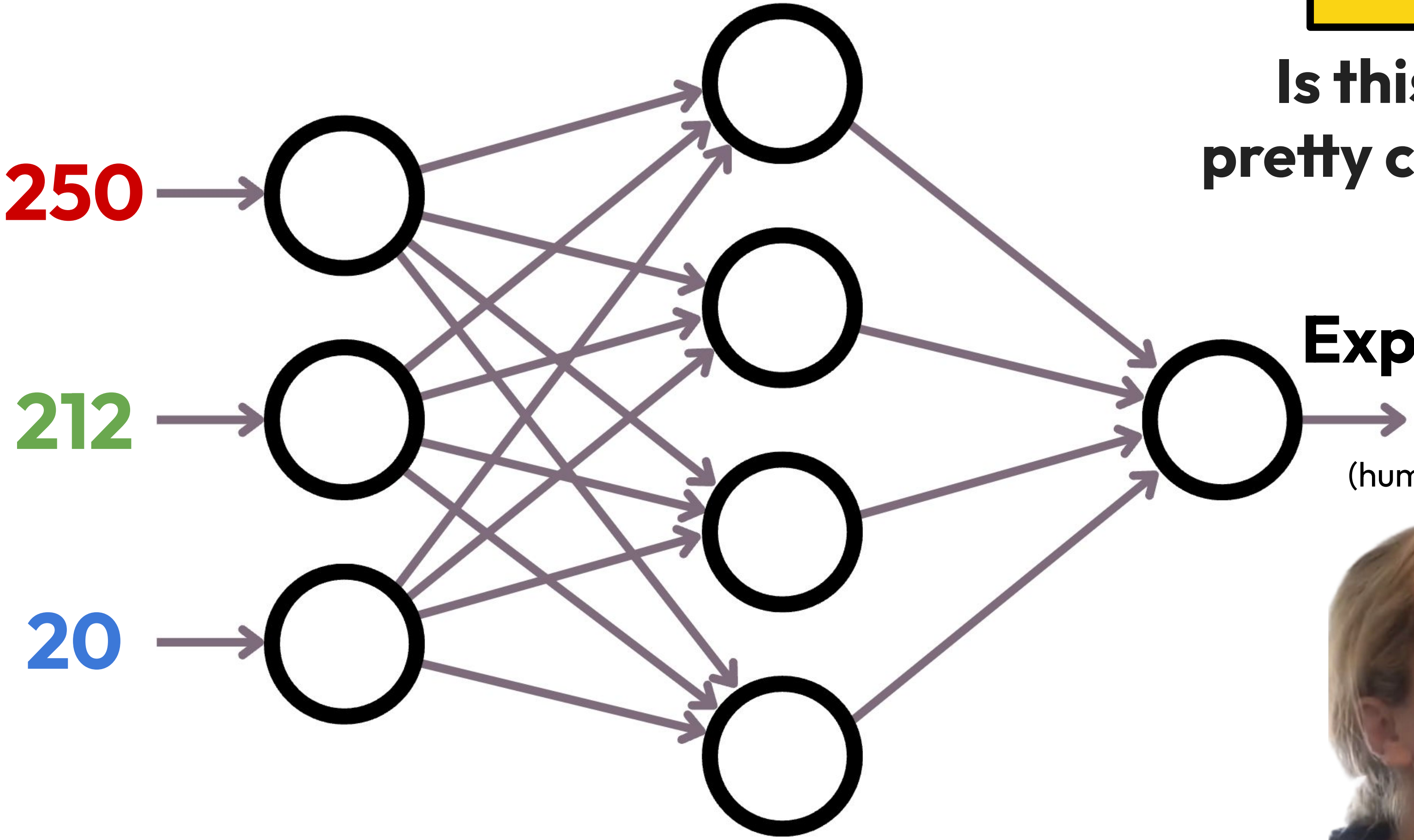
What is training?

Initially, the neural net does not know what to do

- It can approximate any function
- Adjust parameters (**weights** and **biases**)
- Learn their correct values through <u>training</u>

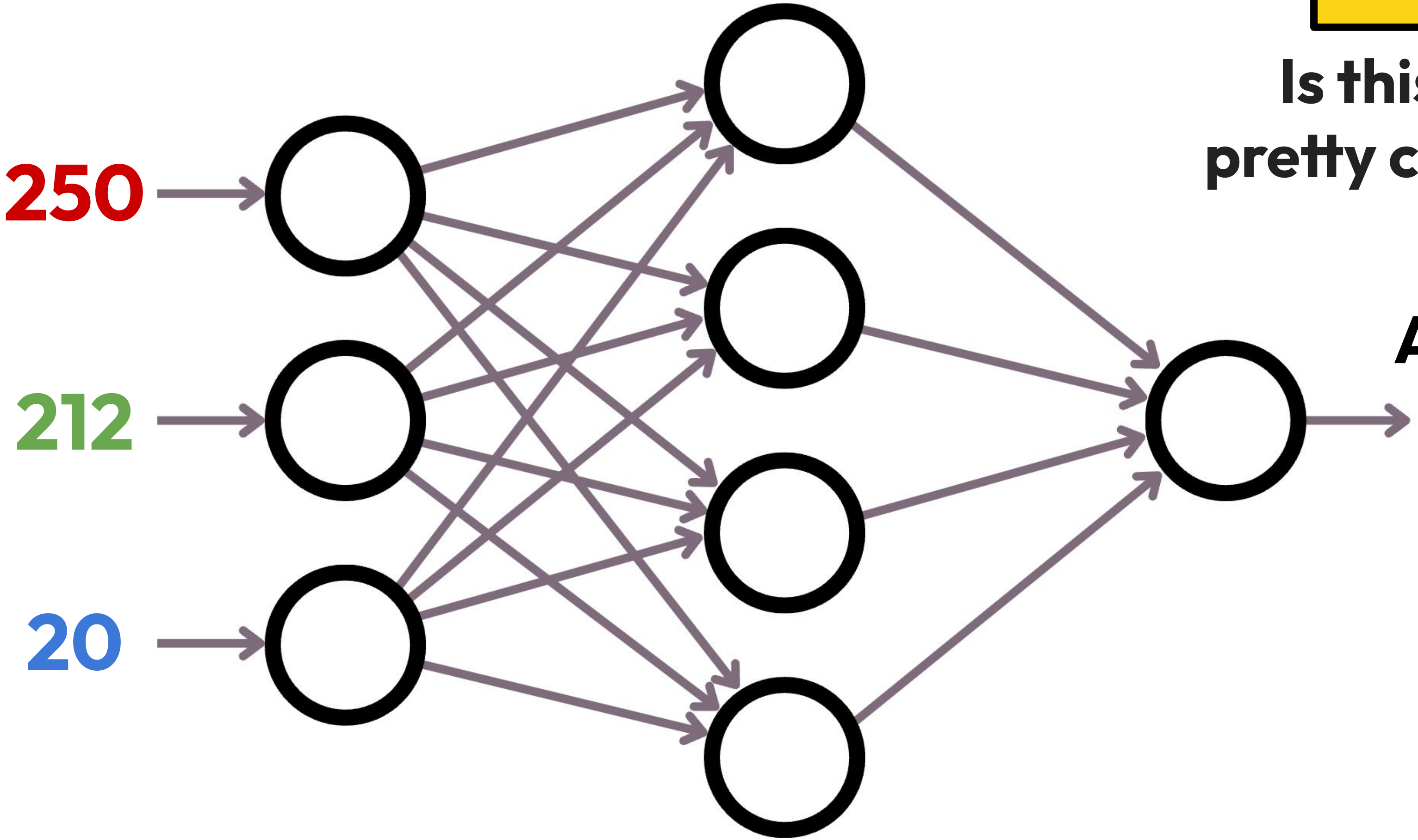


Is this a pretty color?

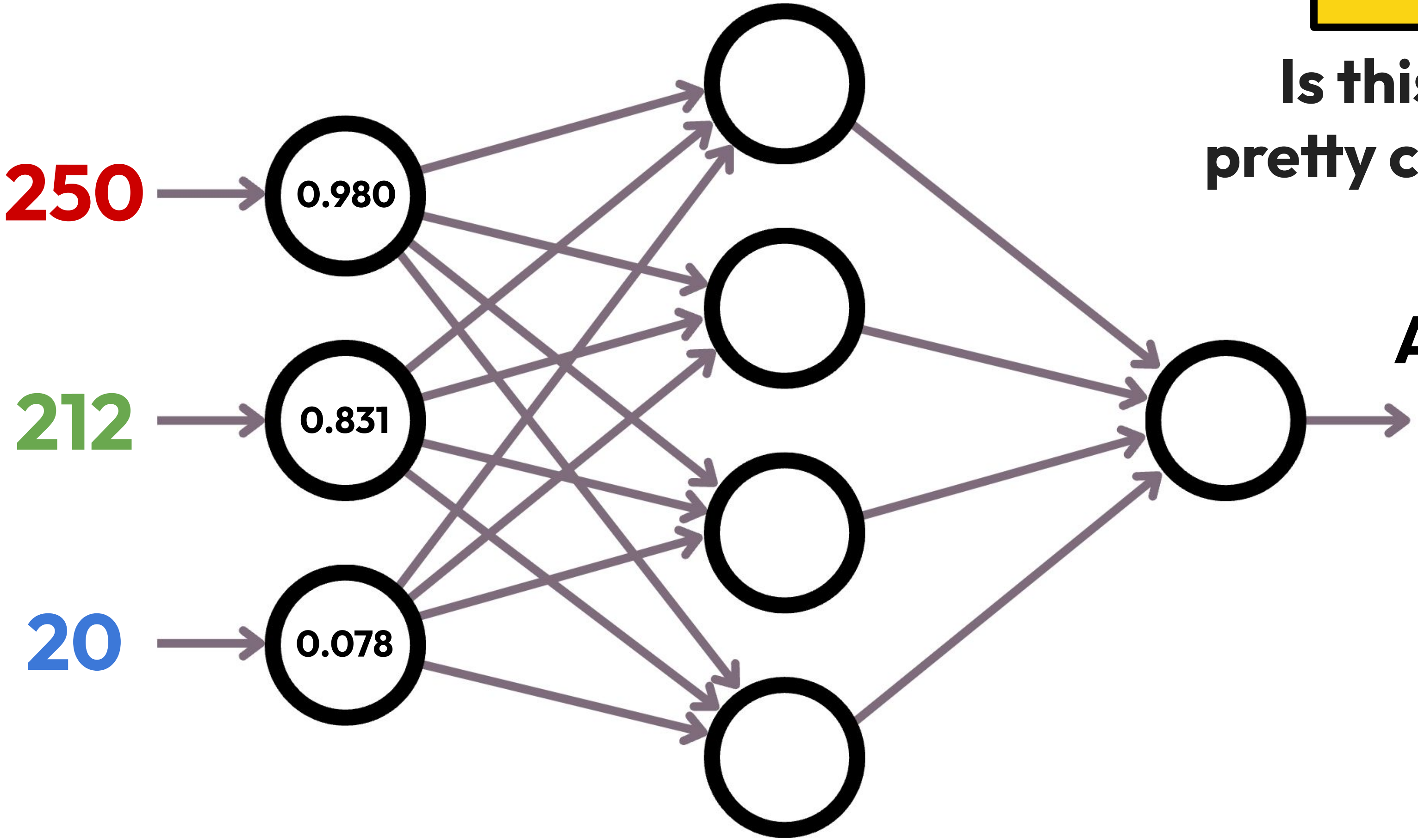


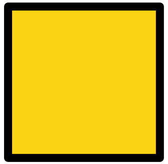
Expected:
0.943
(human annotation)

Is this a pretty color?



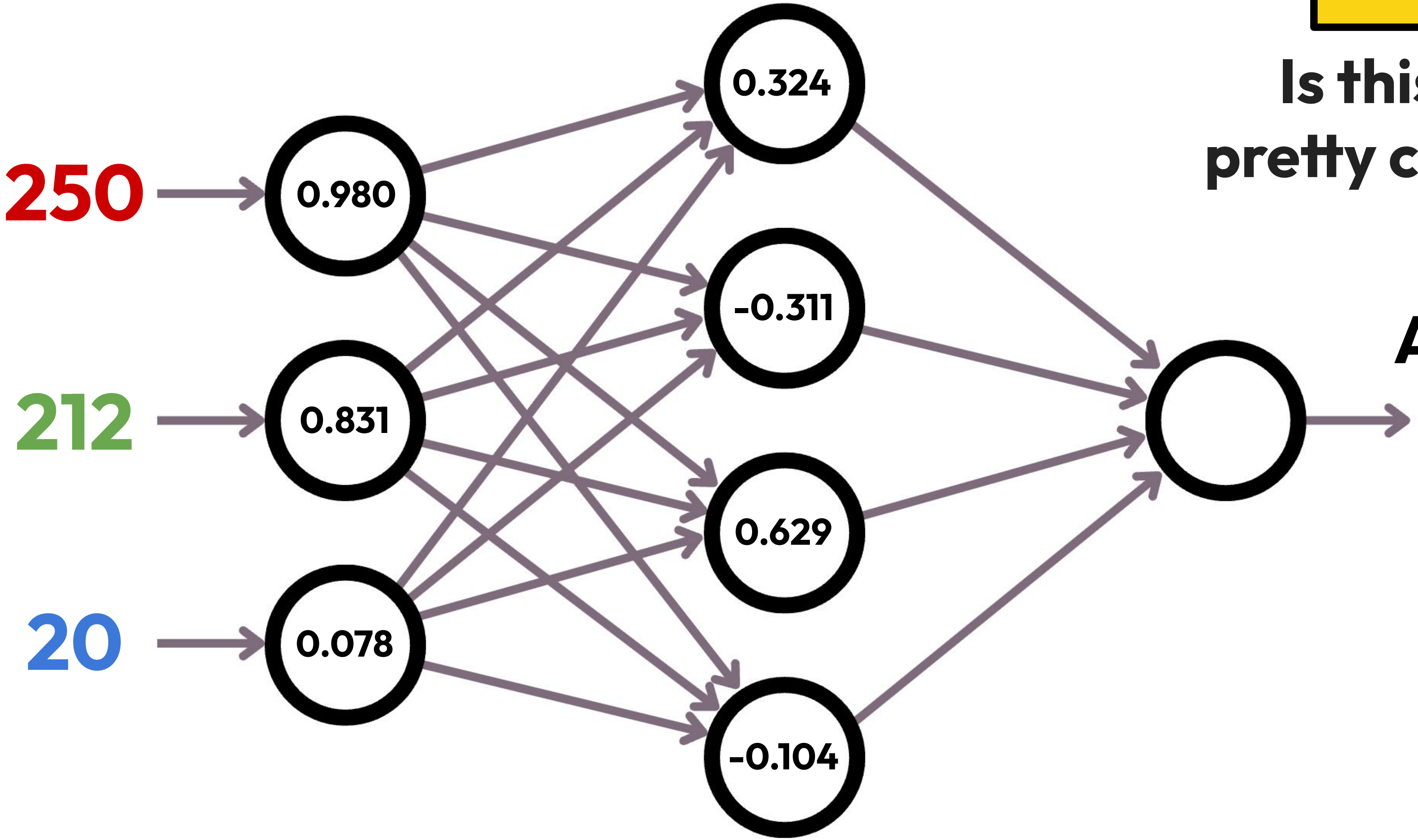
Actual:





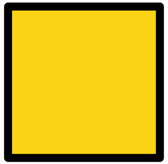
Is this a pretty color?

Actual:

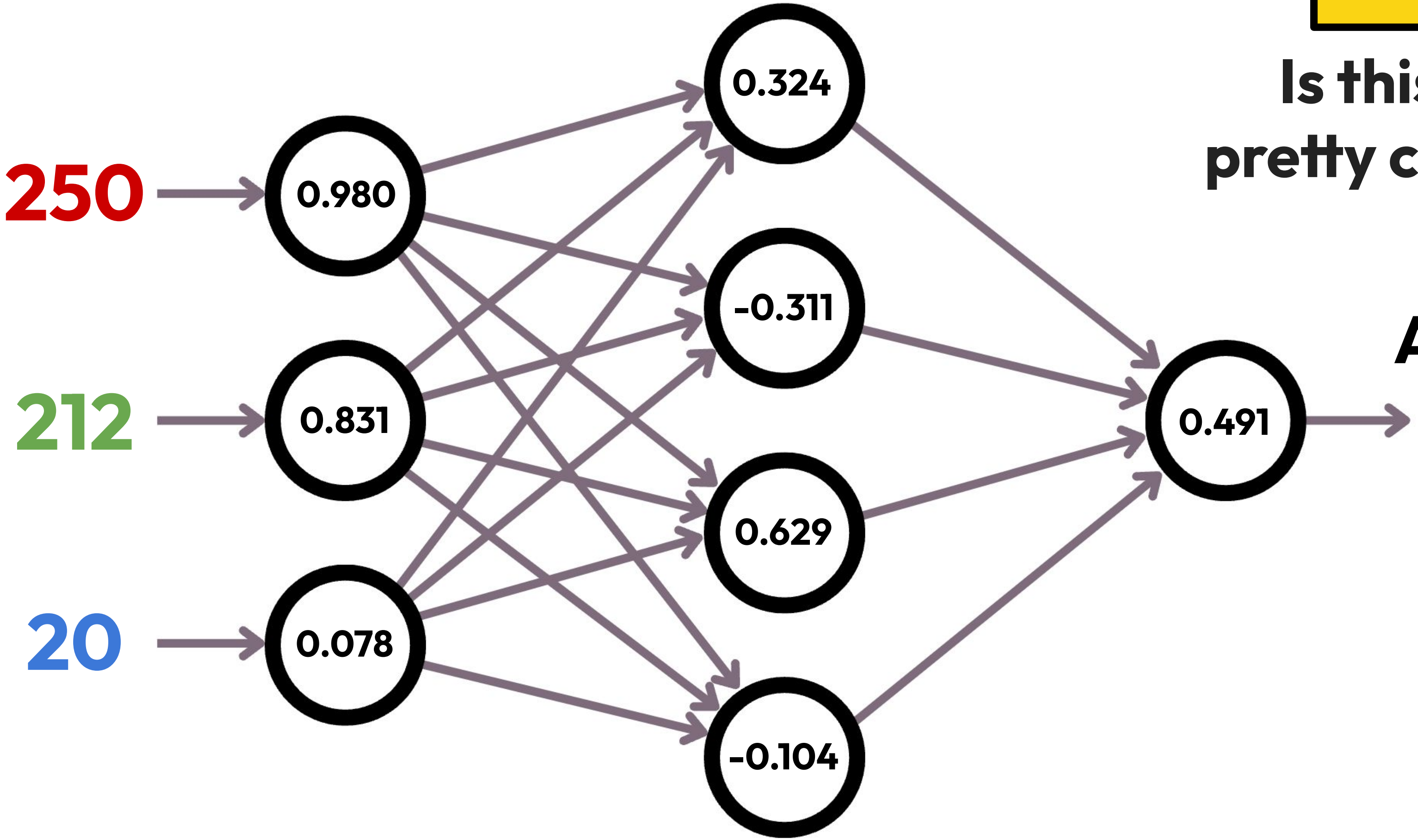


Is this a pretty color?

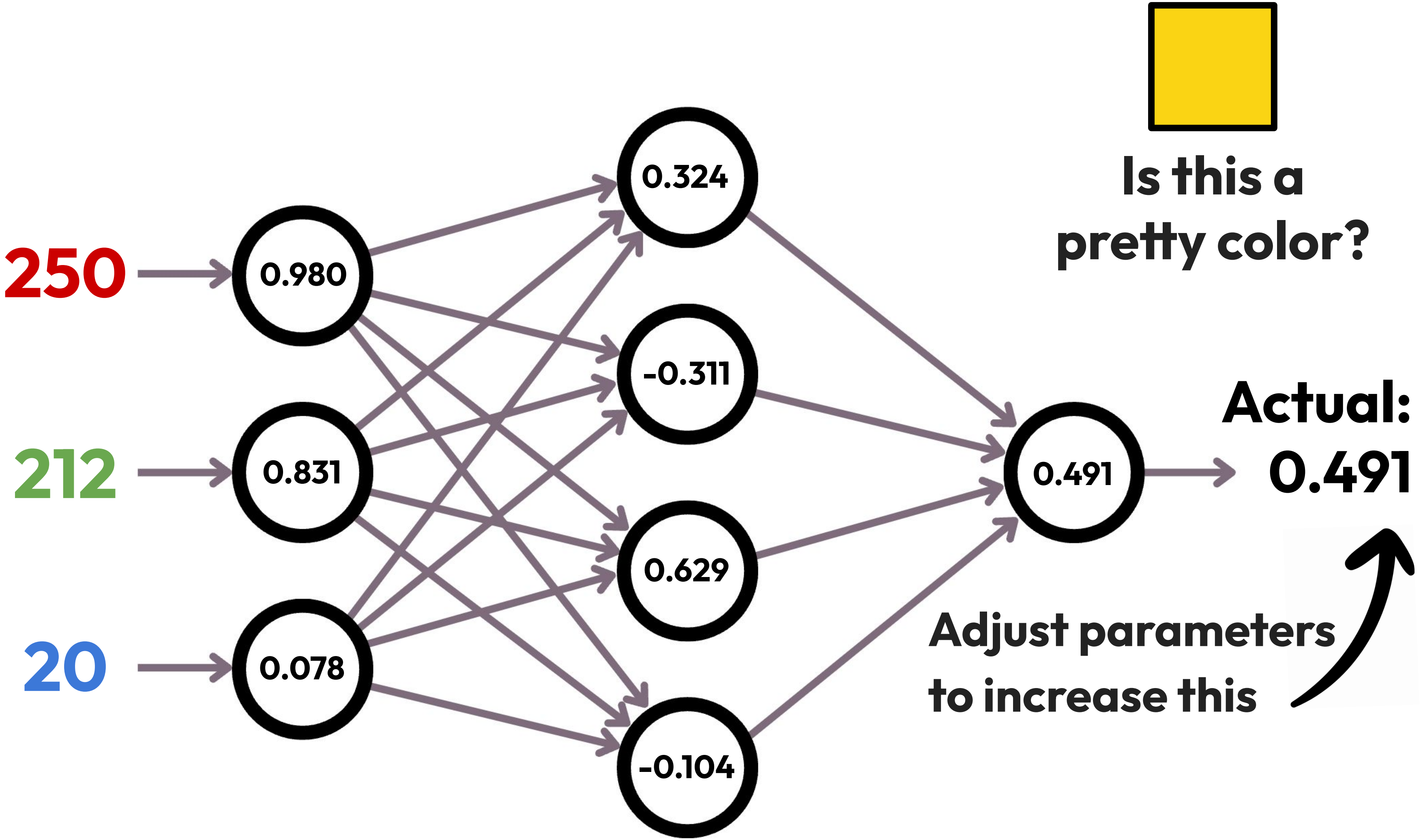
Actual:



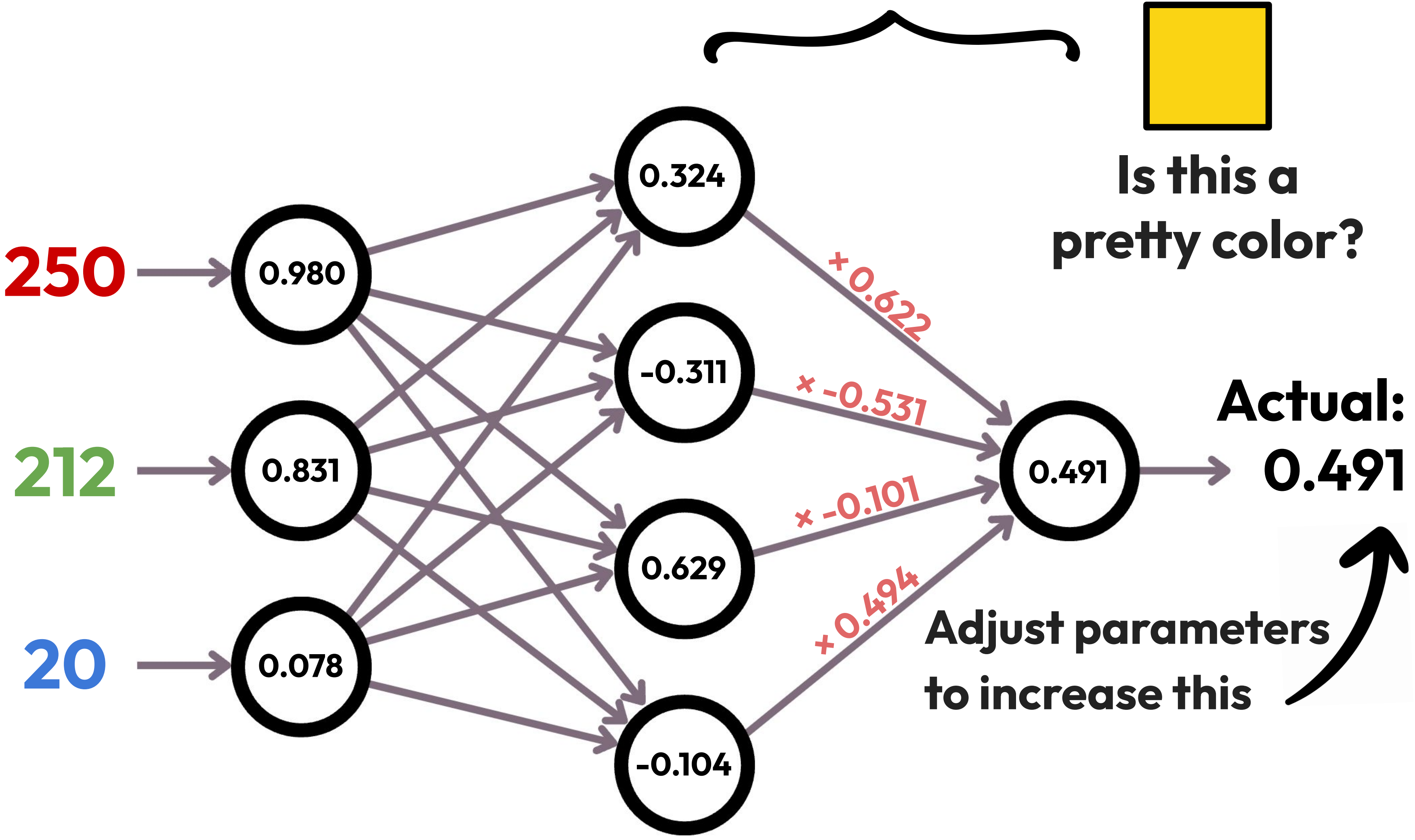
Is this a pretty color?

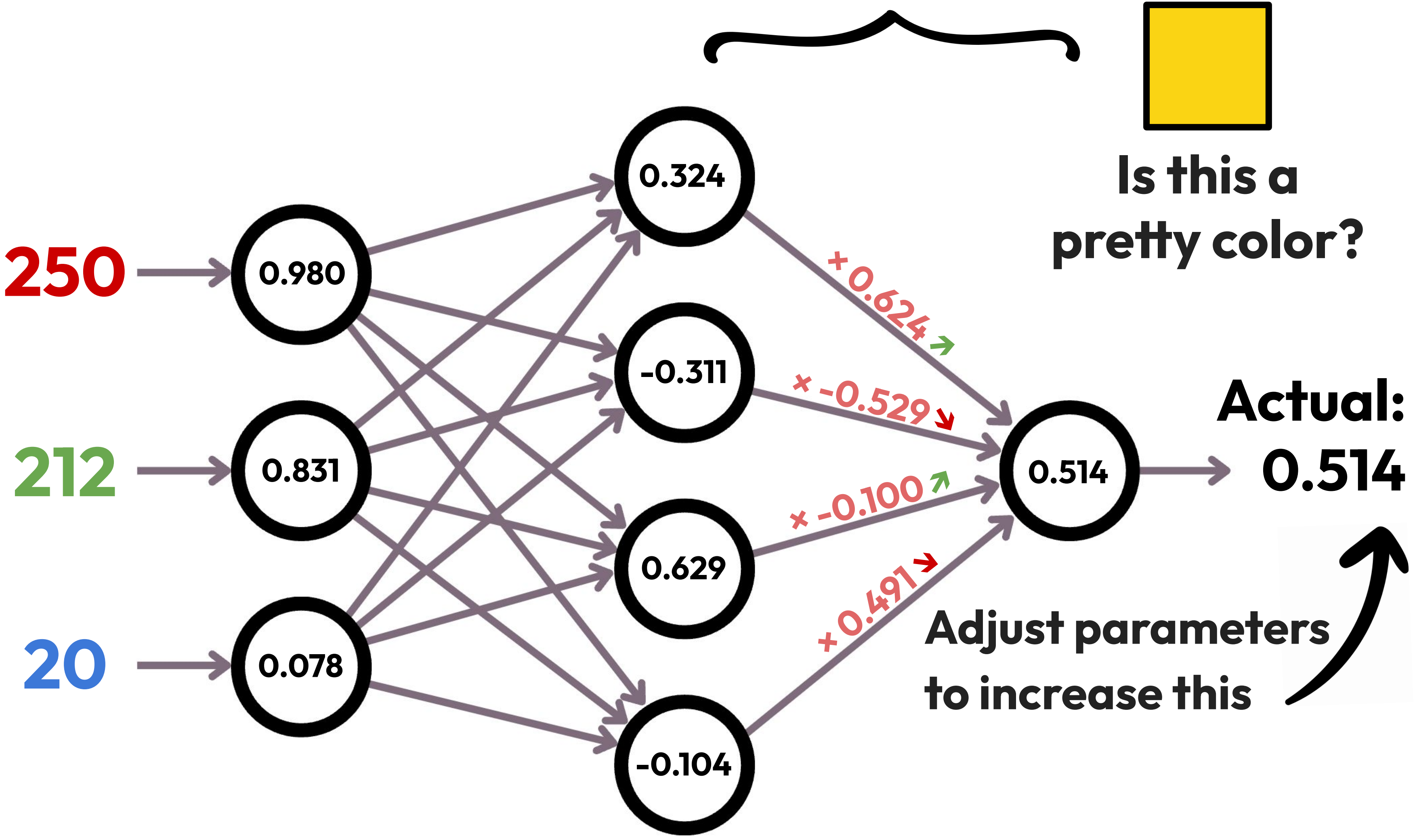


Actual:
0.491

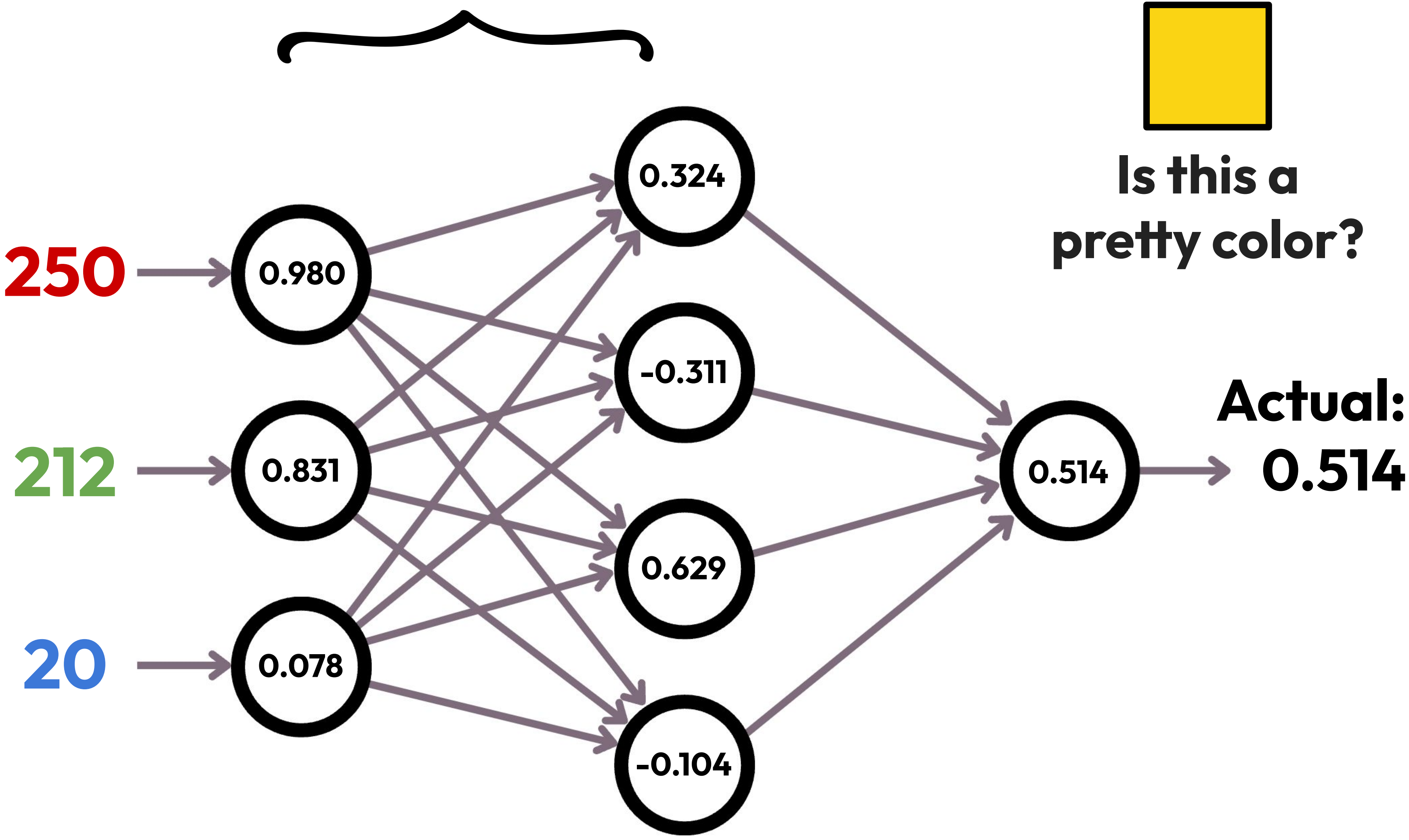


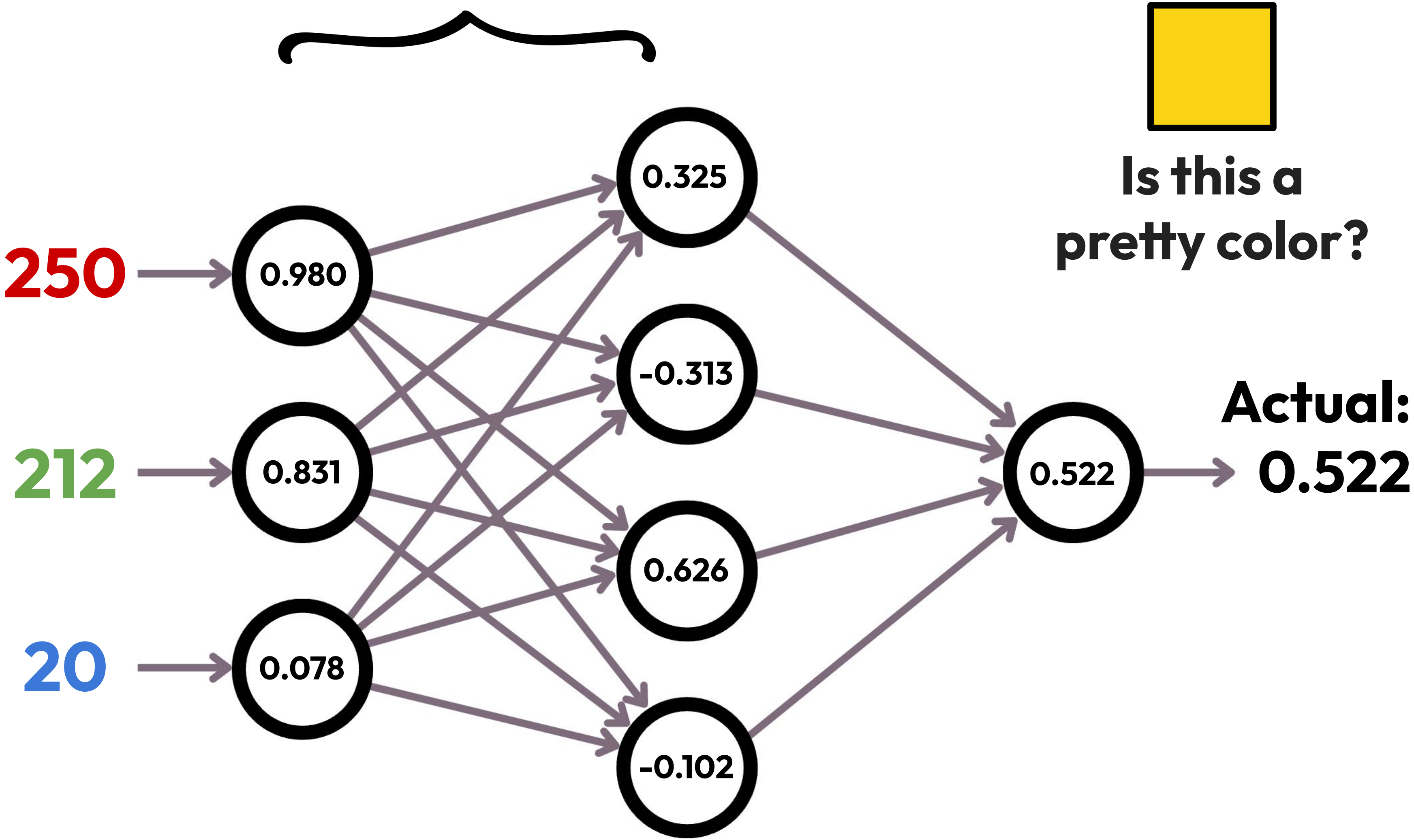
250
212
20
0.980
0.831
0.078
0.324
-0.311
0.629
-0.104
0.491
Is this a pretty color?
Actual:
0.491
Adjust parameters to increase this

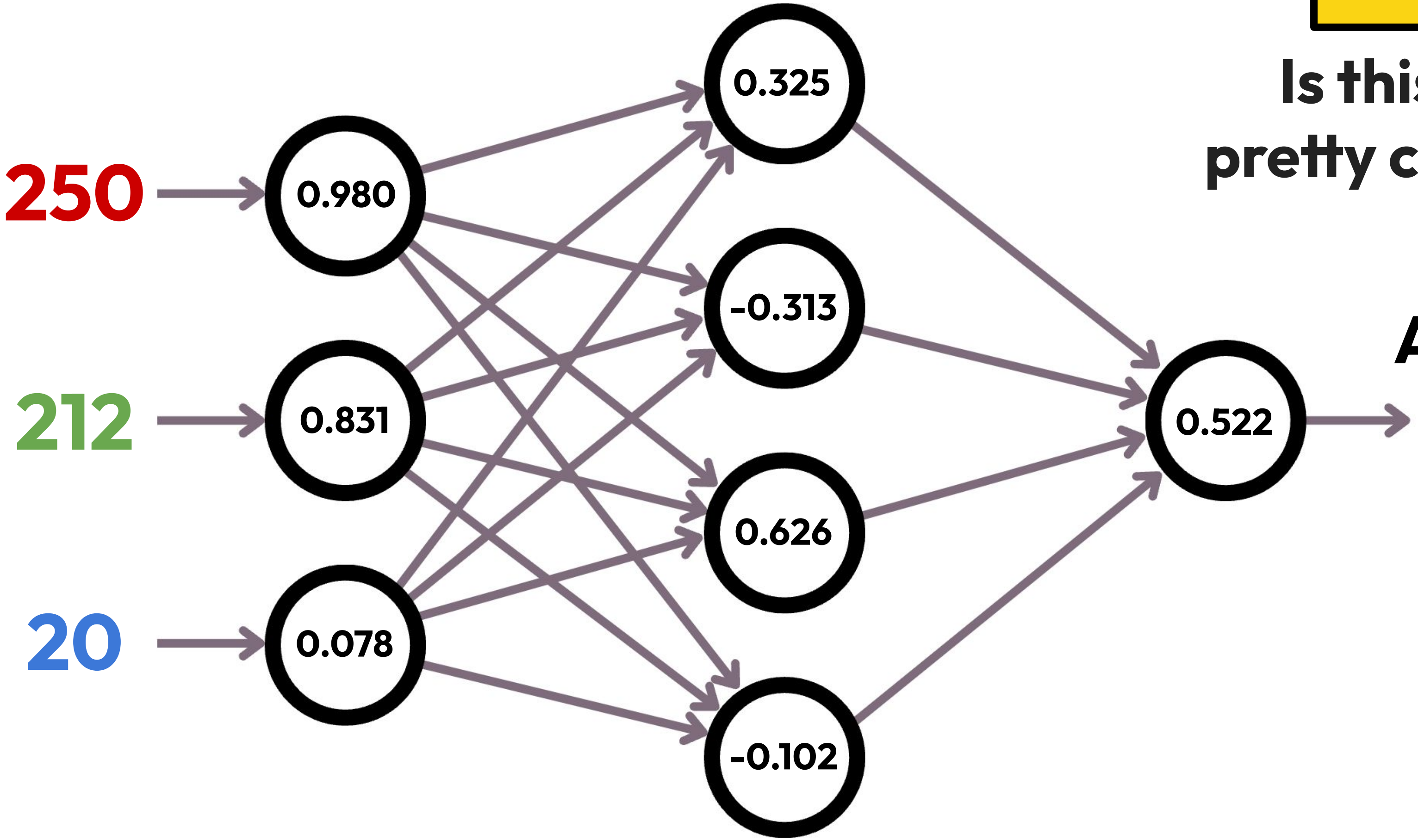

Is this a pretty color?
250
212
20
0.980
0.831
0.078
0.324
-0.311
0.629
-0.104
×0.622
× -0.531
× -0.101
×0.494
0.491
Actual:
0.491
Adjust parameters
to increase this

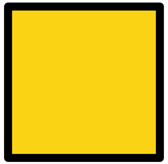


250
212
20
0.980
0.831
0.078
0.324
-0.311
0.629
-0.104
× 0.624
× -0.529
× -0.100
× 0.491
0.514
Is this a
pretty color?
Actual:
0.514
Adjust parameters
to increase this


Is this a pretty color?
250
212
20
0.980
0.831
0.078
0.324
-0.311
0.629
-0.104
0.514
Actual:
0.514


Is this a pretty color?
250
212
20
0.980
0.831
0.078
0.325
-0.313
0.626
-0.102
0.522
Actual:
0.522





Is this a pretty color?

Actual:
0.522

Repeat many times, over many examples

After enough training steps, the neural network will approximate human evaluation

Parameter adjustments use the <u>gradient</u>

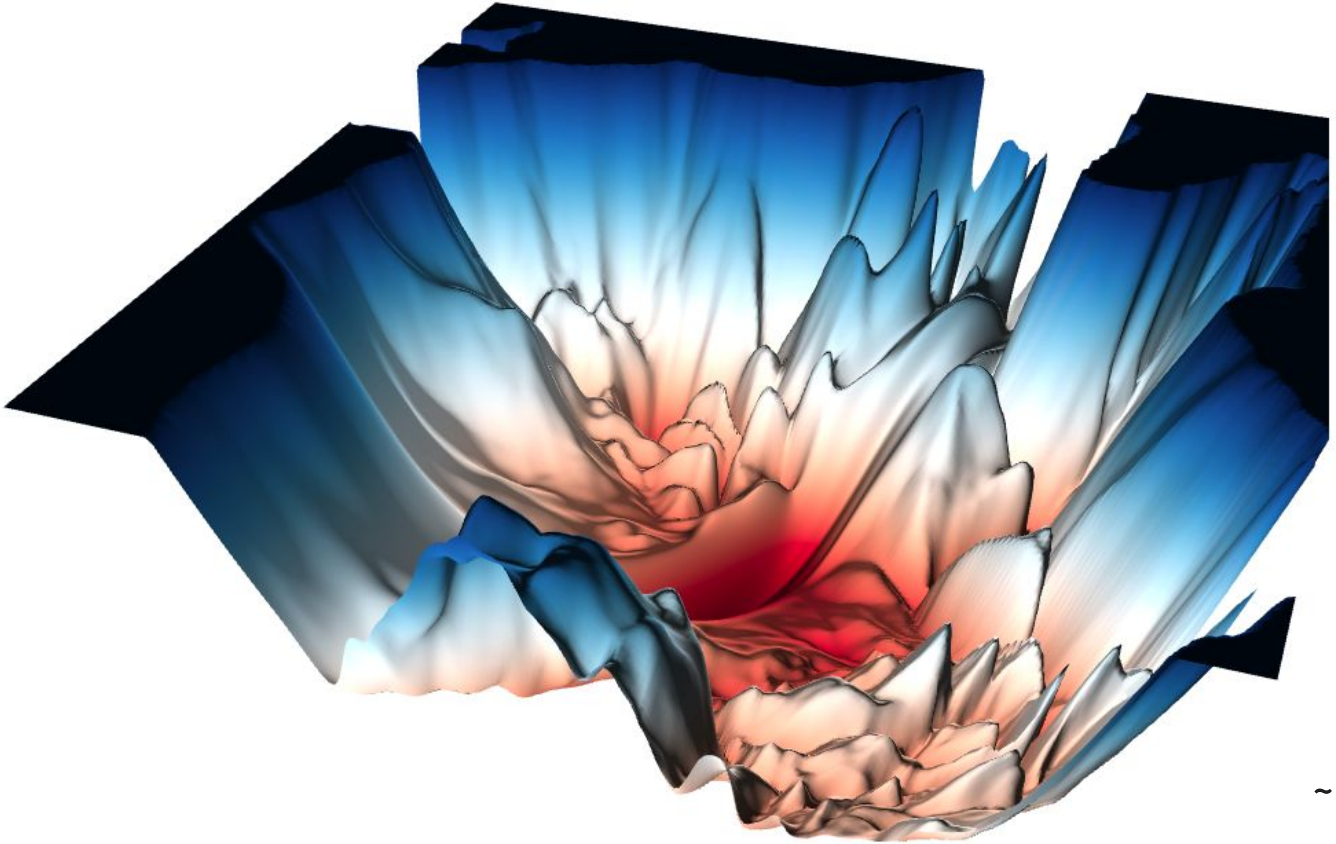
Neural nets must be **continuous** and **differentiable**

Parameter adjustments use the <u>gradient</u>

Neural nets must be **continuous** and **differentiable**

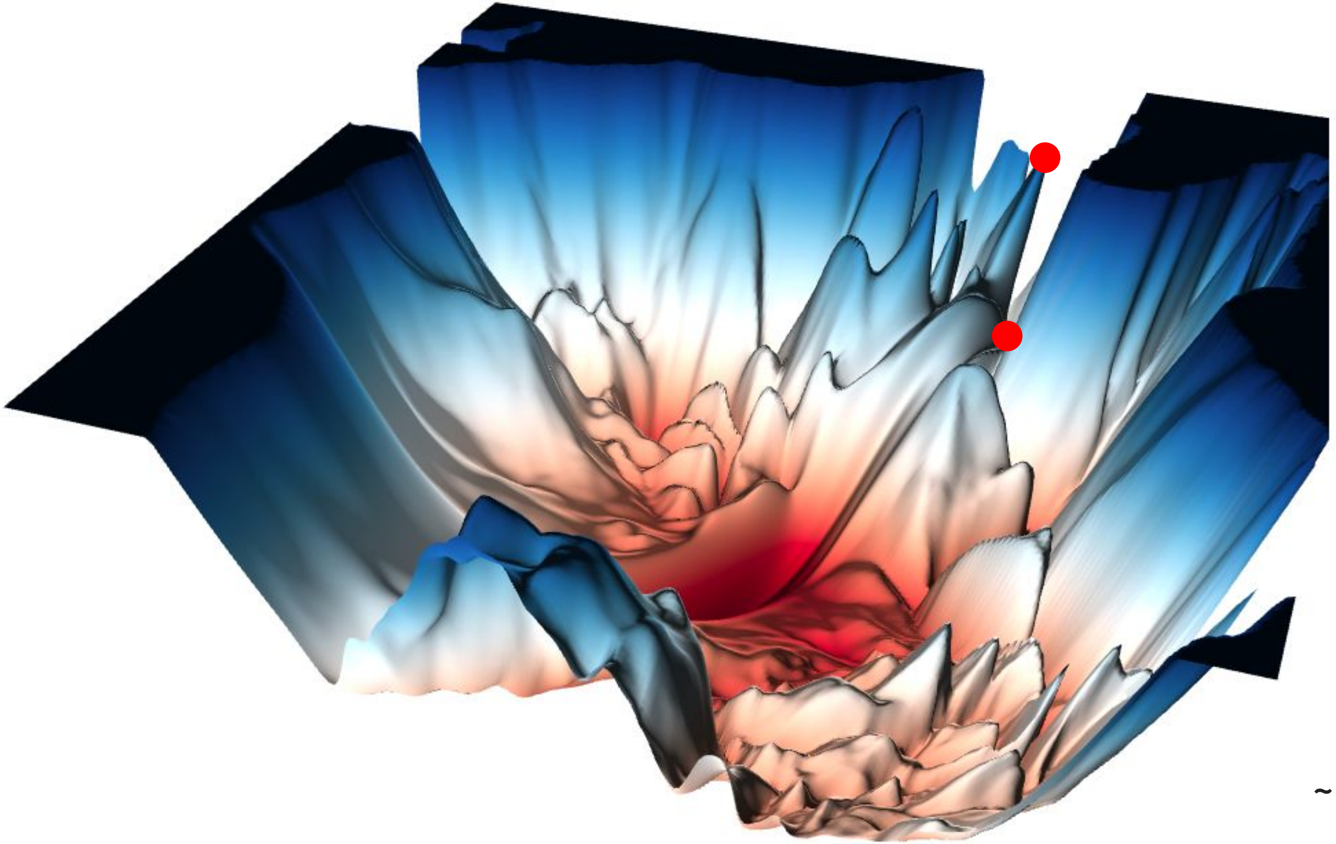
We can exploit these properties!

Neural networks are not “smooth”



~ Ankur Mohan

Neural networks are not “smooth”



~ Ankur Mohan

What do these irregularities mean?

A <u>small perturbation</u> on the input

(in a carefully chosen direction)

can change the output by a lot

Original

Infected

Each pixel is changed by 1 bit

Original

Panda

Infected

Chair (99.9%)

Pandas are nice,
but let’s break something cooler.

Self-driving cars!

Disclaimer

For legal and practical reasons, the following attacks will be demonstrated on a simulated environment.

(but they also work on actual cars)

Objective:

Get this sign to be detected as "speed limit 50"

How do we find the correct perturbation?

How do we find the correct perturbation?

Use the gradient!

Partial derivative

$$\frac{\partial score}{\partial pixel_{x,y}}$$

Partial derivative

$$\frac{\partial score}{\partial pixel_{x,y}}$$

how will the **score** change

if

this **pixel** changes slightly

Partial derivative

$$\frac{\partial score}{\partial pixel_{x,y}}$$

how will the **score** change

if

this **pixel** changes slightly

We can also see it as a correlation coefficient

Gradient-based adversarial attacks

Change **all** pixels **slightly** in the gradient direction

OR

Change **a few** most significant pixels **all the way**

FGSM

Many small changes

Speed limit 50 (99.9%)

JSMA

Few large changes

Speed limit 50 (99.9%)

These attacks are not **robust**

They are very sensitive to:

- Camera noise
- Object angle
- Lighting
- Background

Infected image
raw file

Speed limit 50 (99.9%)

Infected image
smartphone photo

Stop (97.6%)

Infected image

original perturbation

Speed limit 50 (99.9%)

Infected image

same perturbation, different pose

Stop (98.0%)

2 conditions needed

- Bit-perfect input
- White-box access to the model

How do we make a better attack?

We want a perturbation that is:

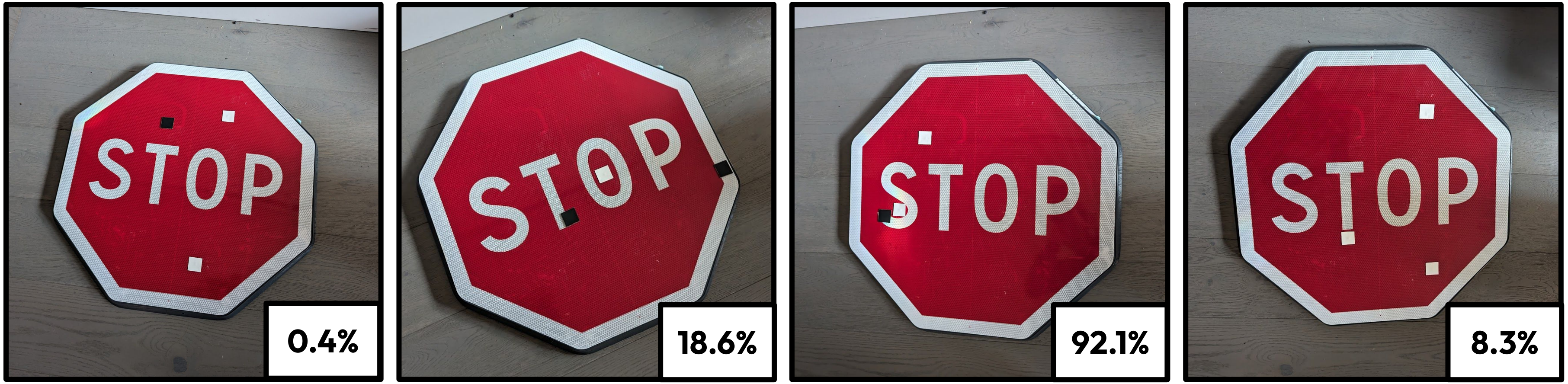
- physically applied to the sign
- as stealthy as possible
- robust on a wide range of conditions
- without access to the model gradients

Noise resistance can be achieved by using larger stickers

Black-box attack

We don’t have access to gradients, but we can still get the prediction score

→ Try many configurations and keep the best



Black-box attack

Not very efficient to do manually

- Need to try millions of combinations
- Not very reproducible

→ Work in a **virtual** attack setup


STOP

STOP

STOP

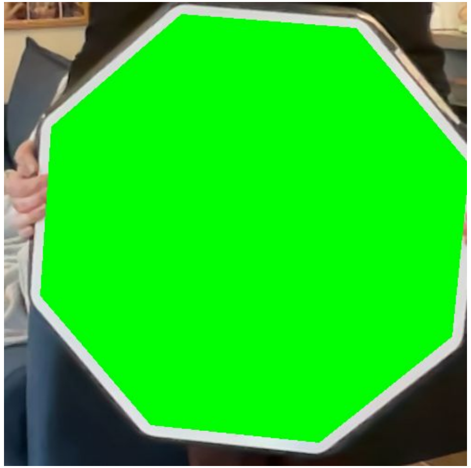
STOP

STOP

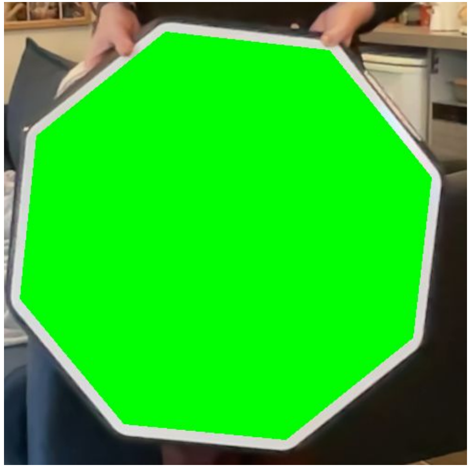
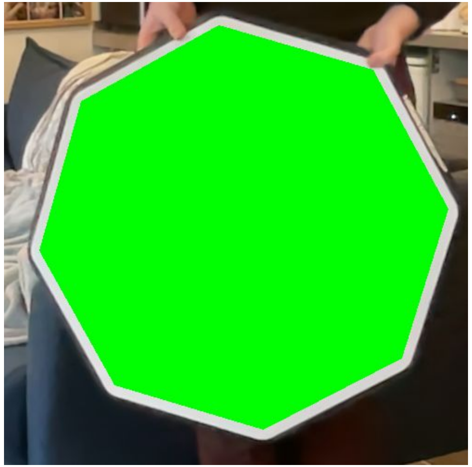
STOP

STOP

STOP




STOP
STOP
STOP
STOP
STOP
STOP
STOP
STOP

The same perturbation can now be simulated over many different conditions

Eliminates all external variables

- Noise
- Position/angle
- Lighting and reflections
- Background

How do we search for good perturbations?

Bruteforce - try all possible combinations

Guaranteed to find the optimal solution

...but takes 10^{25} years 😱

How do we search for good perturbations?

<u>Monte-Carlo</u> - try random perturbations

Good perturbations are rare (~1 in a billion)

Need to be **very lucky**

How do we search for good perturbations?

Work step by step

How do we search for good perturbations?

Work <u>step by step</u>

Test many configurations for sticker **1**, keep the best

How do we search for good perturbations?

Work step by step

Test many configurations for sticker **2**, keep the best

How do we search for good perturbations?

Work step by step

Test many configurations for sticker **3**, keep the best

How do we search for good perturbations?

Work step by step

etc.

How do we search for good perturbations?

Small improvement: add an **optimization** step to search locally for the best attack

Inspired by mutations in genetic algorithms

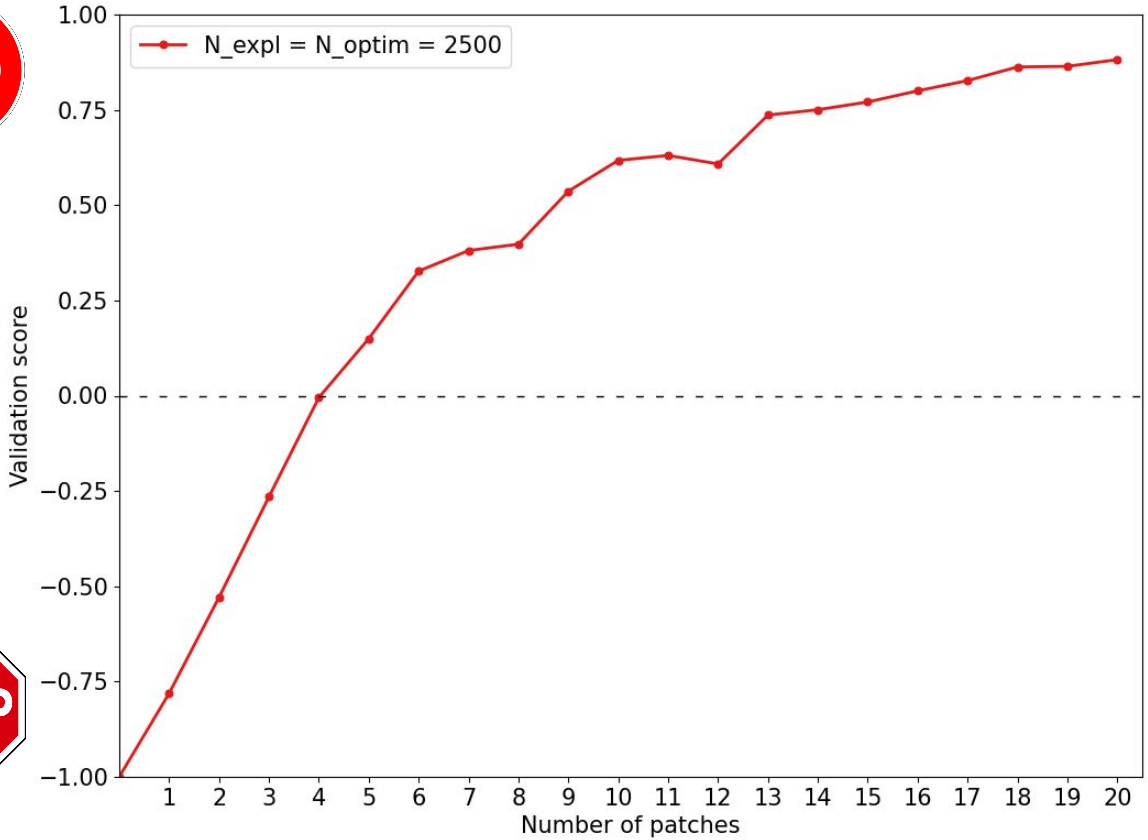
How do we search for good perturbations?

After adding each sticker, try moving the other ones slightly

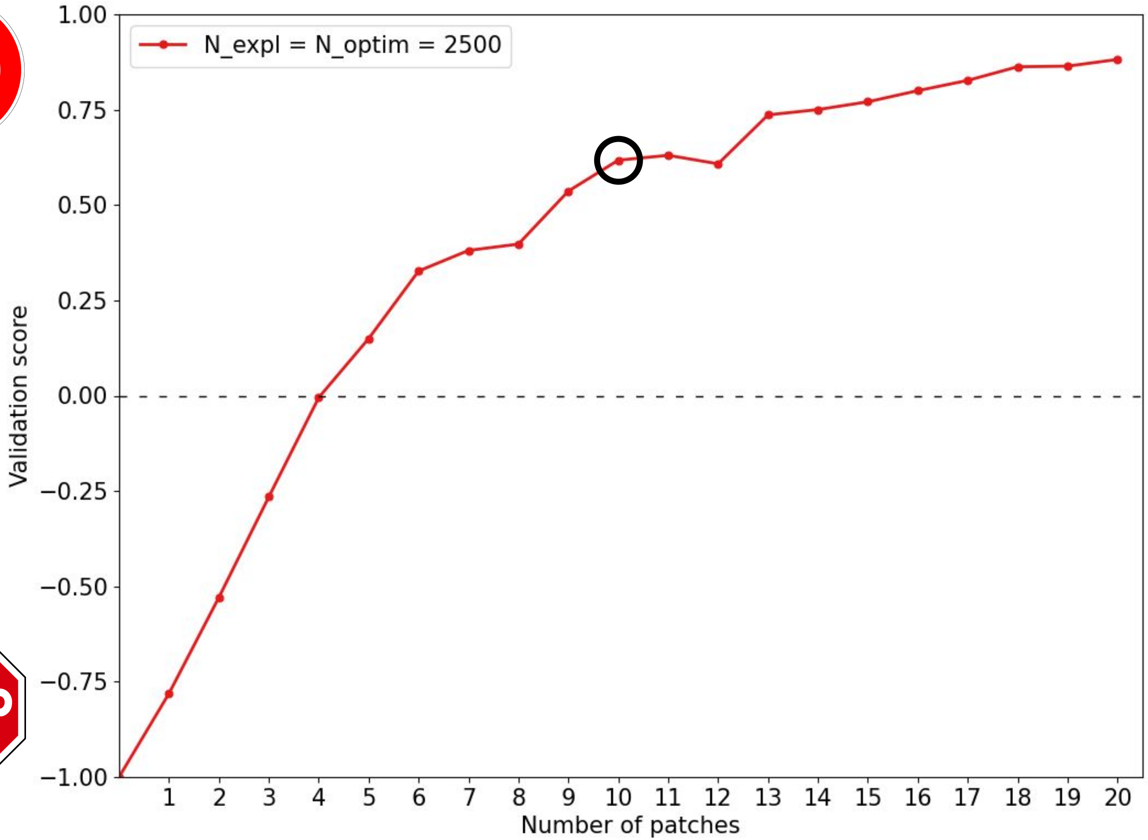
How do we search for good perturbations?

After adding each sticker, try moving the other ones slightly

Attack performance @ 3min per patch



Attack performance @ 3min per patch



Demo time!

Slides

Paper

We want a perturbation that is:

- physically applied to the sign
- as stealthy as possible
- robust on a wide range of conditions
- without access to the model gradients

We want a perturbation that is:

✅ physically applied to the sign

❓ as stealthy as possible

✅ robust on a wide range of conditions

✅ without access to the model gradients

This PoC can be improved. Ideas:

- More stealth
- Black-box++

All neural networks are vulnerable to adversarial attacks, and **no efficient protections** currently exist.

Thank you!

Any questions?

Mathis Hammel

@MathisHammel

mathishammel.com

Independent consultant

- Technical competitions
- Algorithms
- Cybersecurity R&D

Slides

Paper