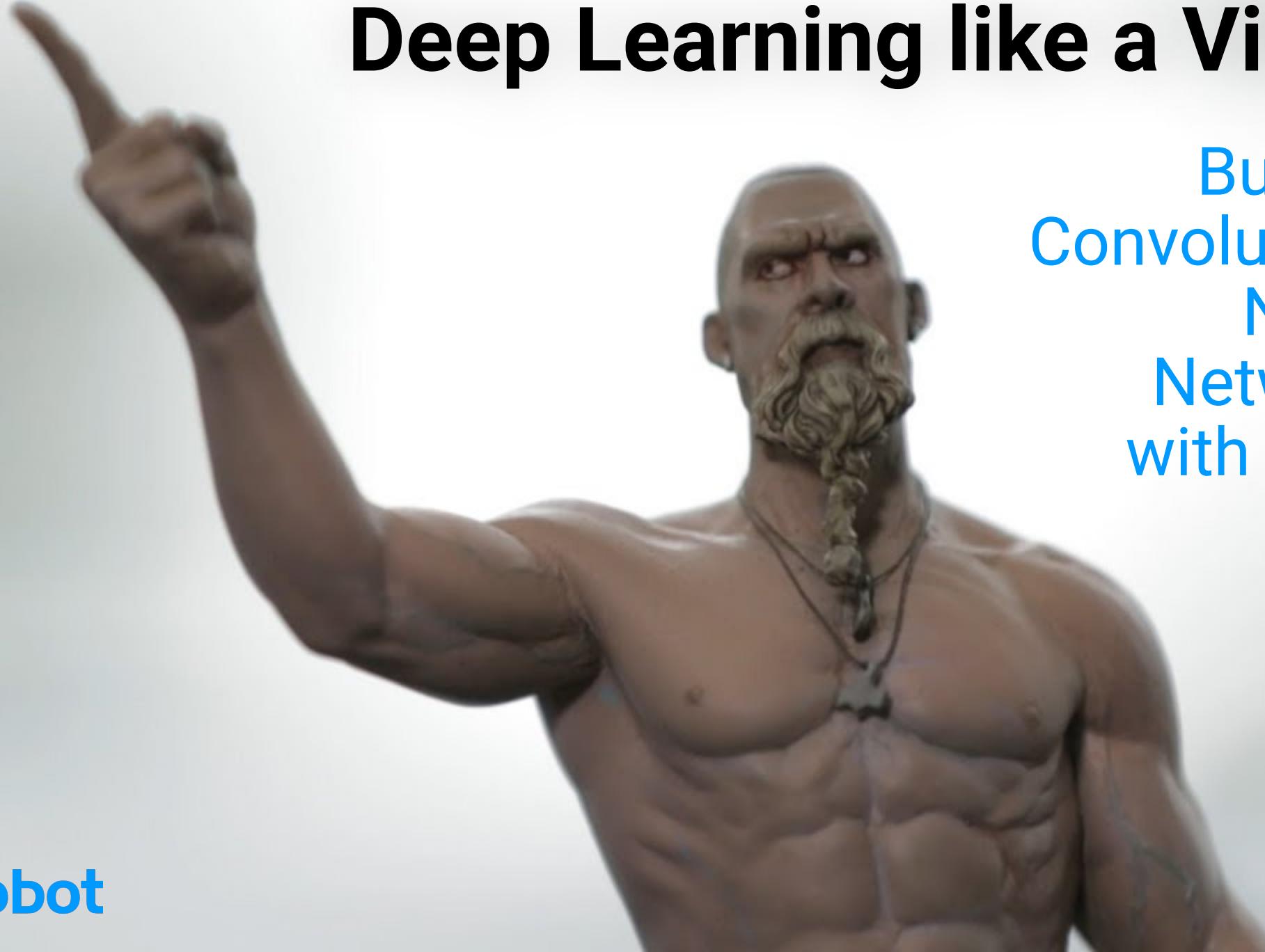
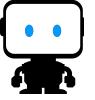


Data**Robot**

# Deep Learning like a Viking



Building  
Convolutional  
Neural  
Networks  
with Keras

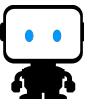


# **Guy Royse**

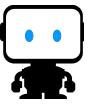
Developer Evangelist  
DataRobot

 @guyroyse

 [github.com/guyroyse](https://github.com/guyroyse)

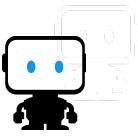


| \* + + ↑ ↘



# IANADS

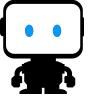
# The Younger Futhark



ᚠ	ᚢ	ᚦ	ᚩ	ᚪ	ᚱ	ᚴ	*	ᚷ	ᛁ	ጀ	ጀ	ጀ	ጀ	ᛖ	ጀ	ጀ	ጀ
fe	ur	thurs	as	reith	kaun	hagall	nauthr	isa	ar	sol	tyr	bjork	mathr	logr	yr		

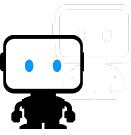
ᚫ\* | \* ᚻ\* | ᛉ

(Kai Rais)



# Lingsberg Runestones

Danr and Húskarl and Sveinn and Holmfríðr, the mother and (her) sons, had this stone erected in memory of Halfdan, the father of Danr and his brothers; and Holmfríðr in memory of her husbandman.

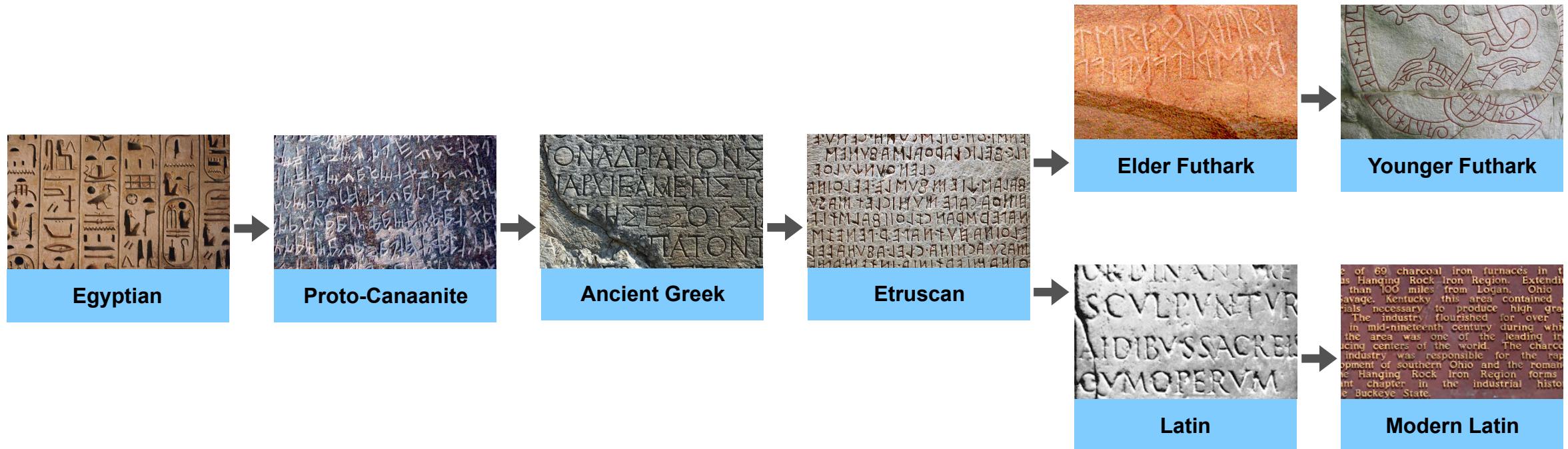
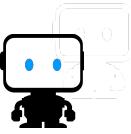


# Younger Futhark vs. Latin & Greek

ᚠ	ᚢ	ᚦ	ᚩ	*	ᚱ	ᚢ	*	ᚦ	ᛁ	ᛅ	ᚦ	ᚦ	ᛏ	ᛒ	ᛝ	ᚱ	ᚦ
fe	ur	thurs	as	reith	kaun	hagall	nauthr	isa	ar	sol	tyr	bjork	mathr	logr	yr		

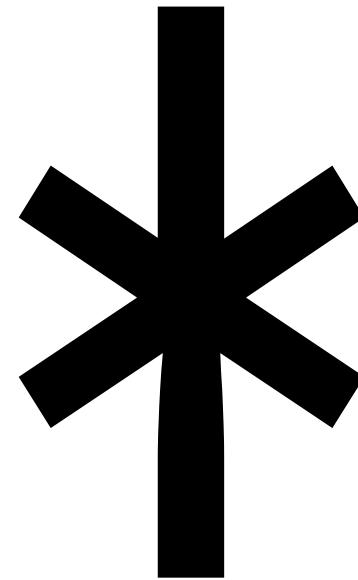
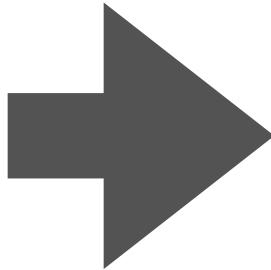
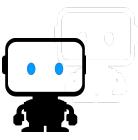
F	U			R			I		S	T	B		L				
		Θ		P			I		Σ	T	B		Λ				

# Common Ancestors

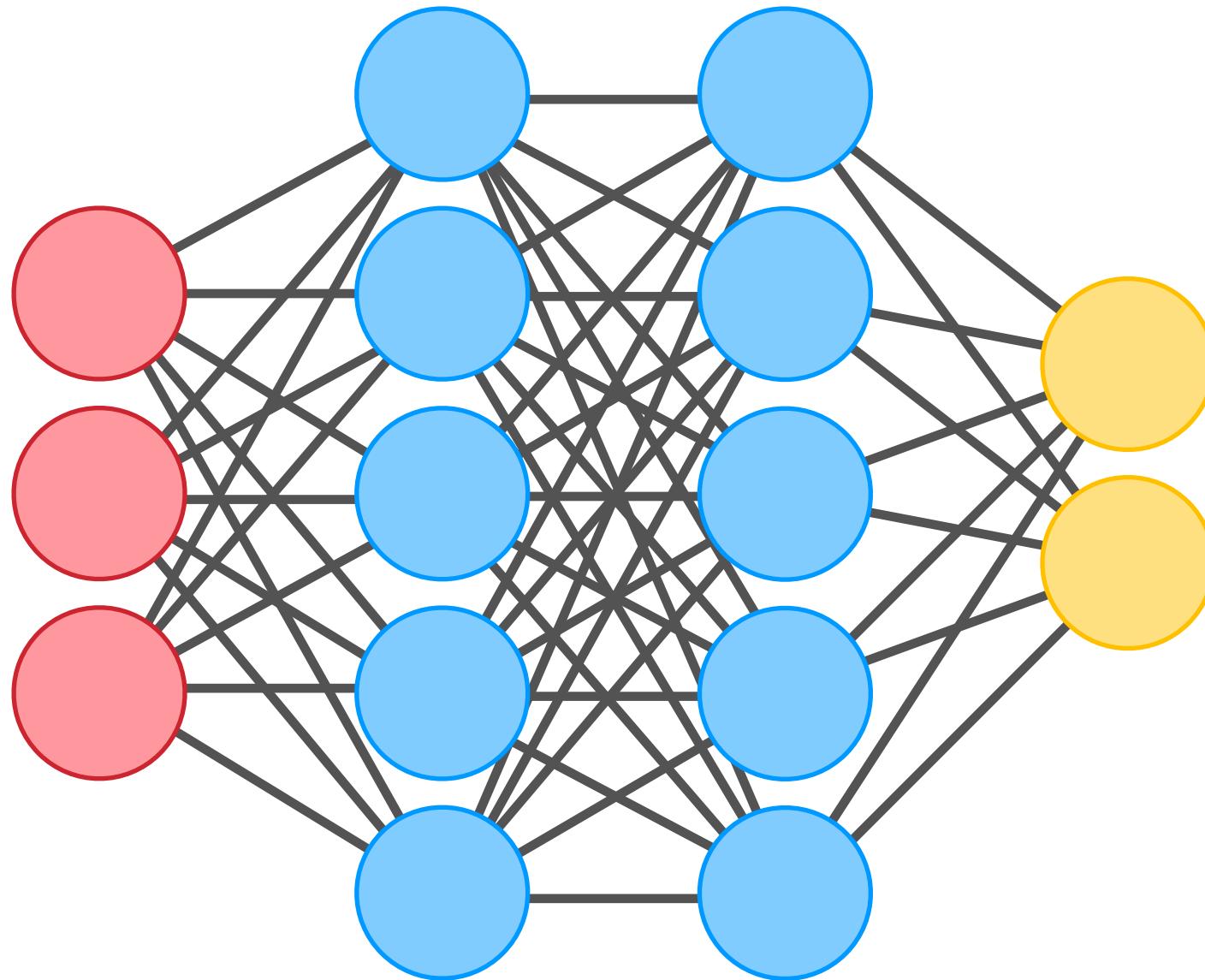
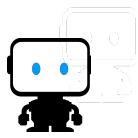


of 69 charcoal iron furnaces in the Hanging Rock Iron Region. Extending more than 100 miles from Logan, Ohio to Savage, Kentucky this area contained all the materials necessary to produce high grade iron. The industry flourished for over 50 years in mid-nineteenth century during which time the area was one of the leading iron producing centers of the world. The charcoal iron industry was responsible for the rapid development of southern Ohio and the surrounding areas. The Hanging Rock Iron Region forms an important chapter in the industrial history of the Buckeye State.

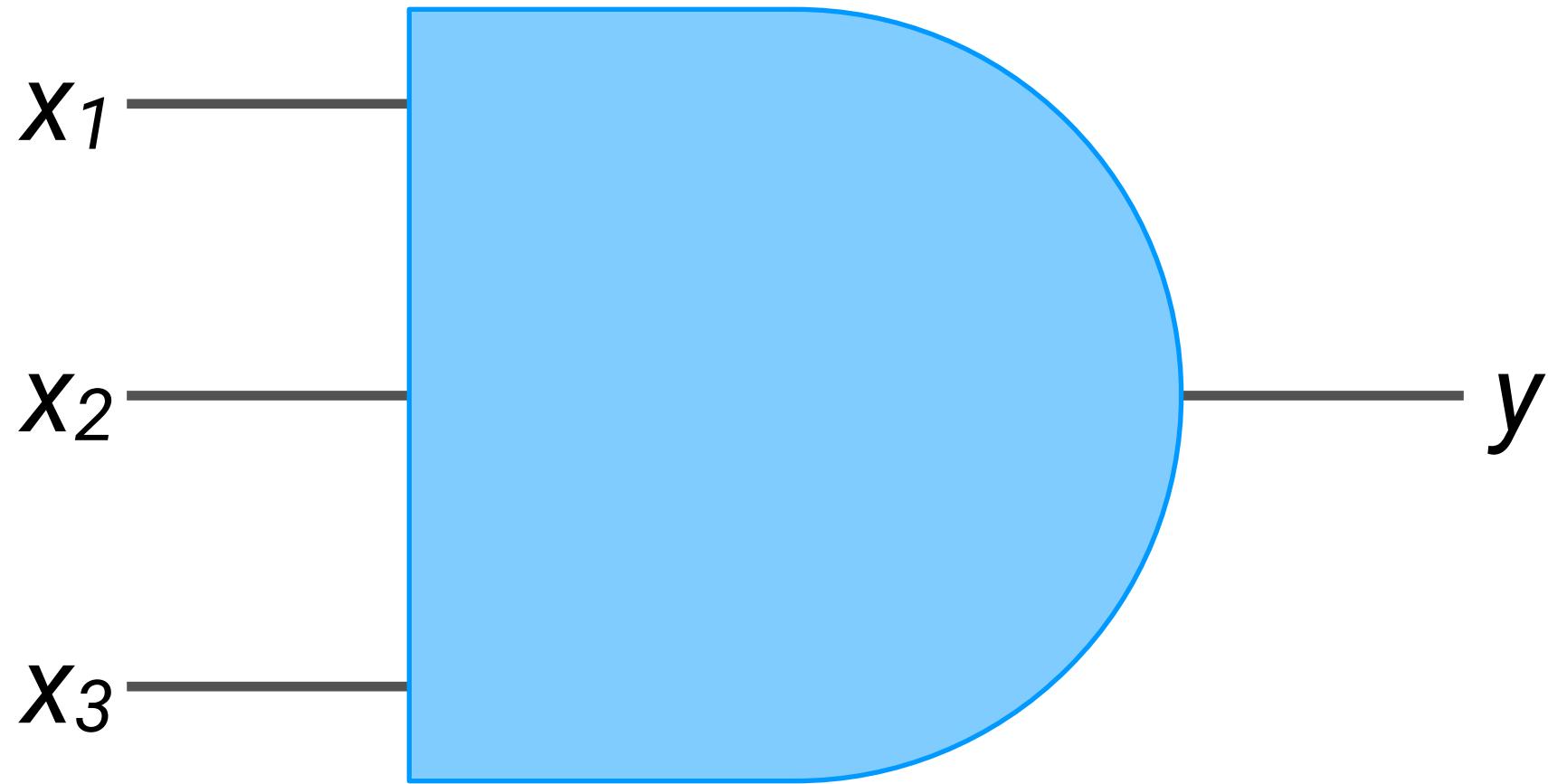
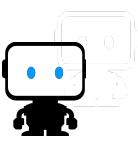
# Recognizing Runes



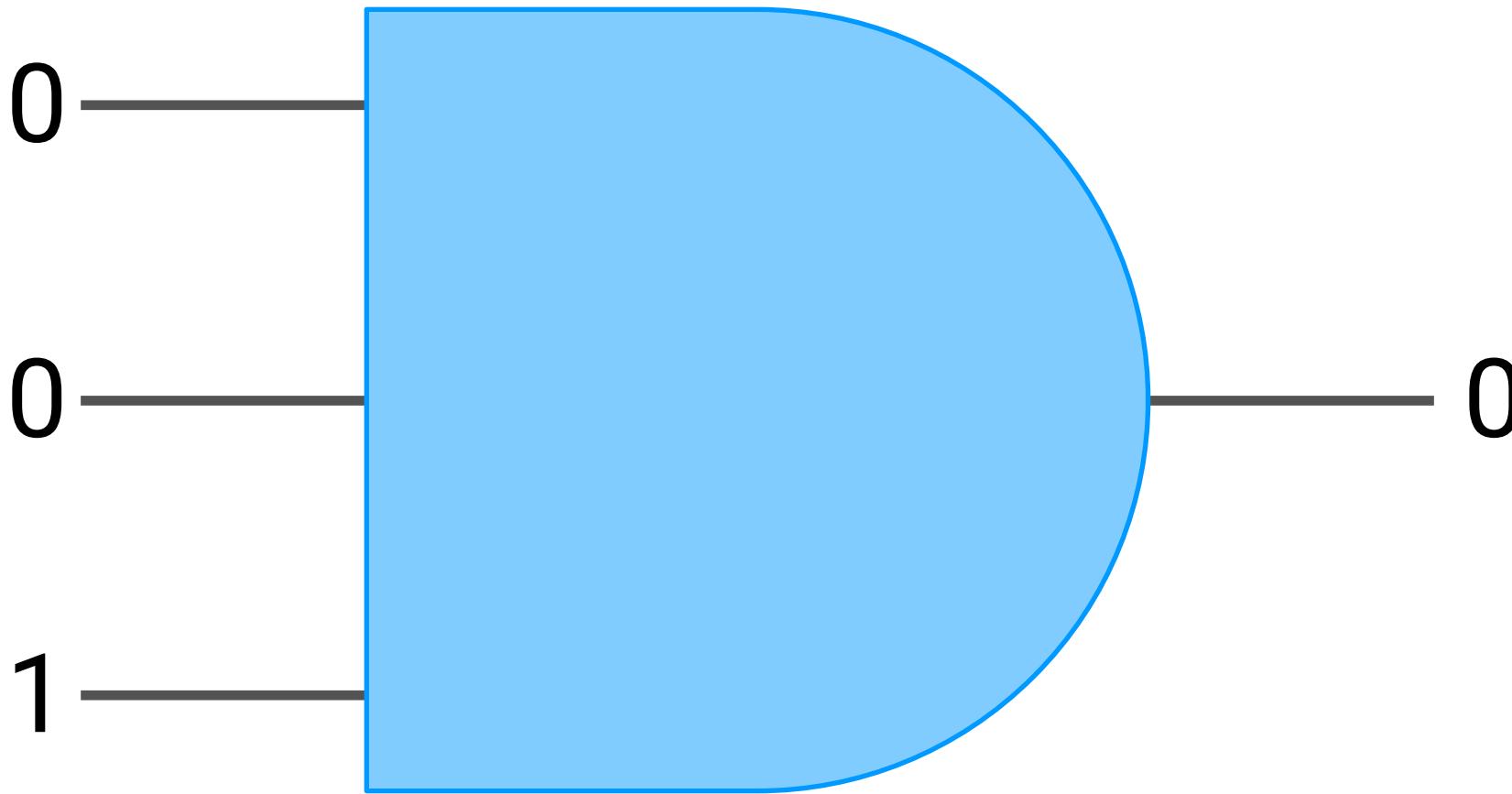
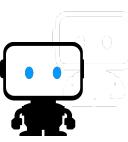
# Neural Networks



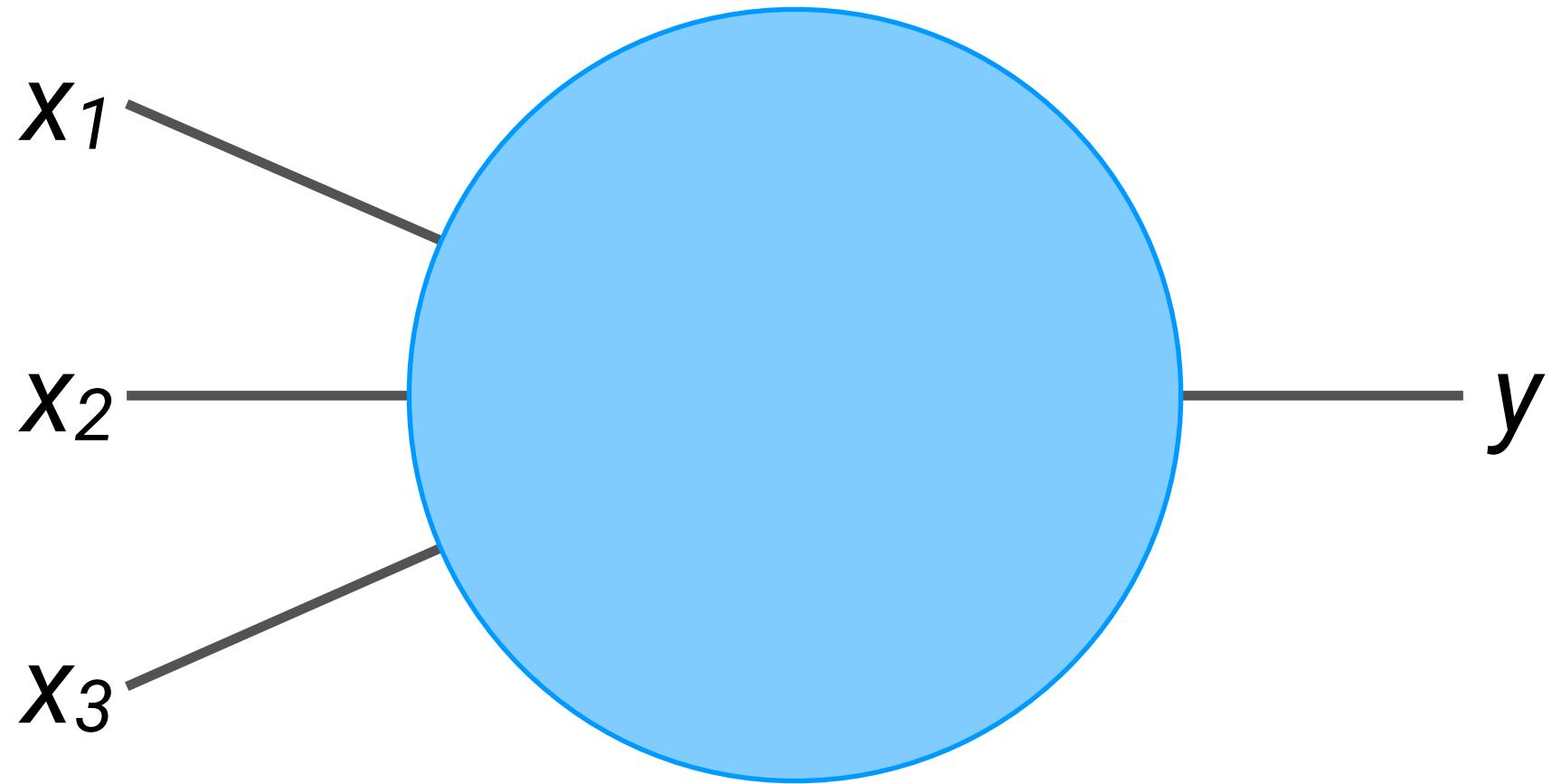
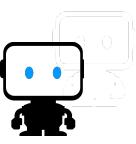
# Logic Gates



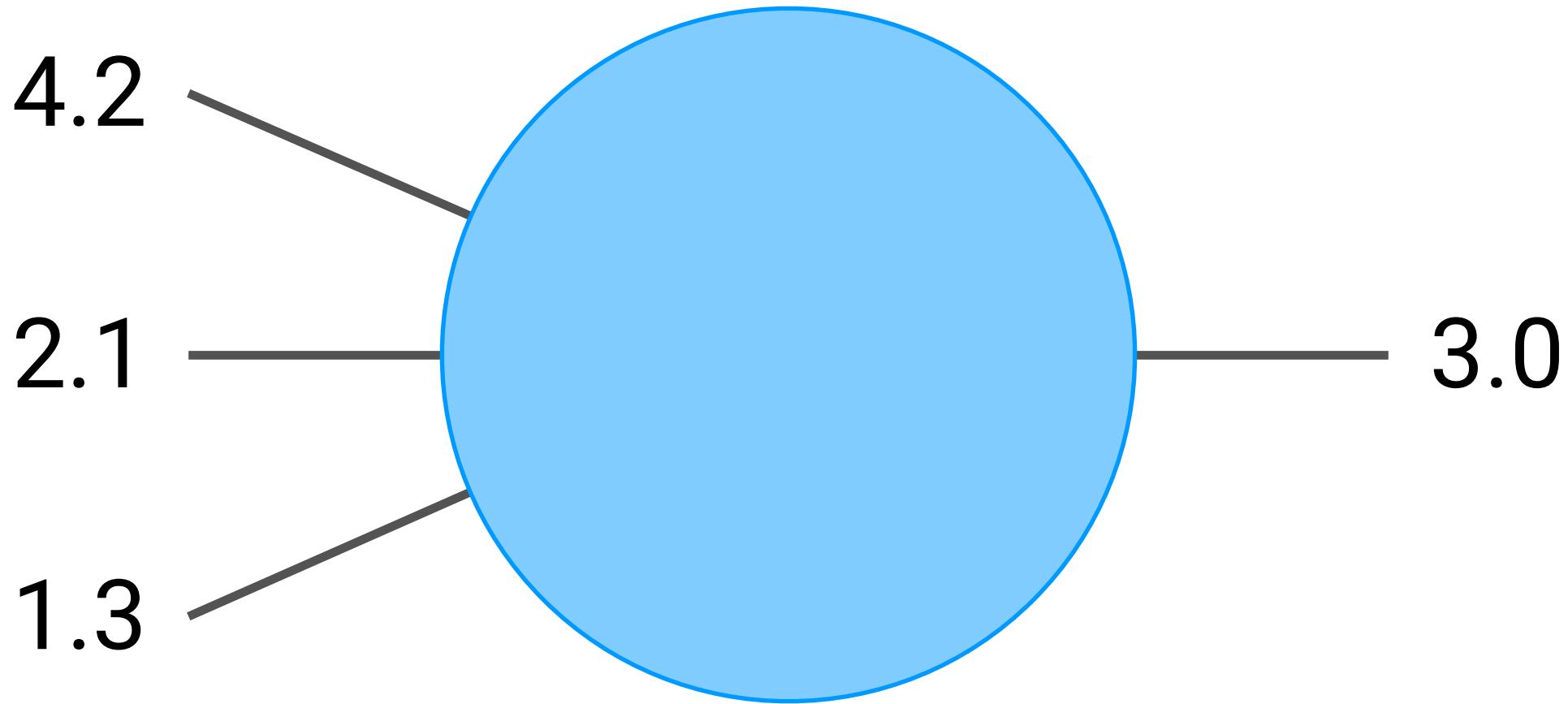
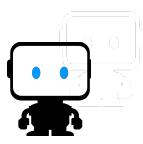
# Logic Gates with Actual Numbers



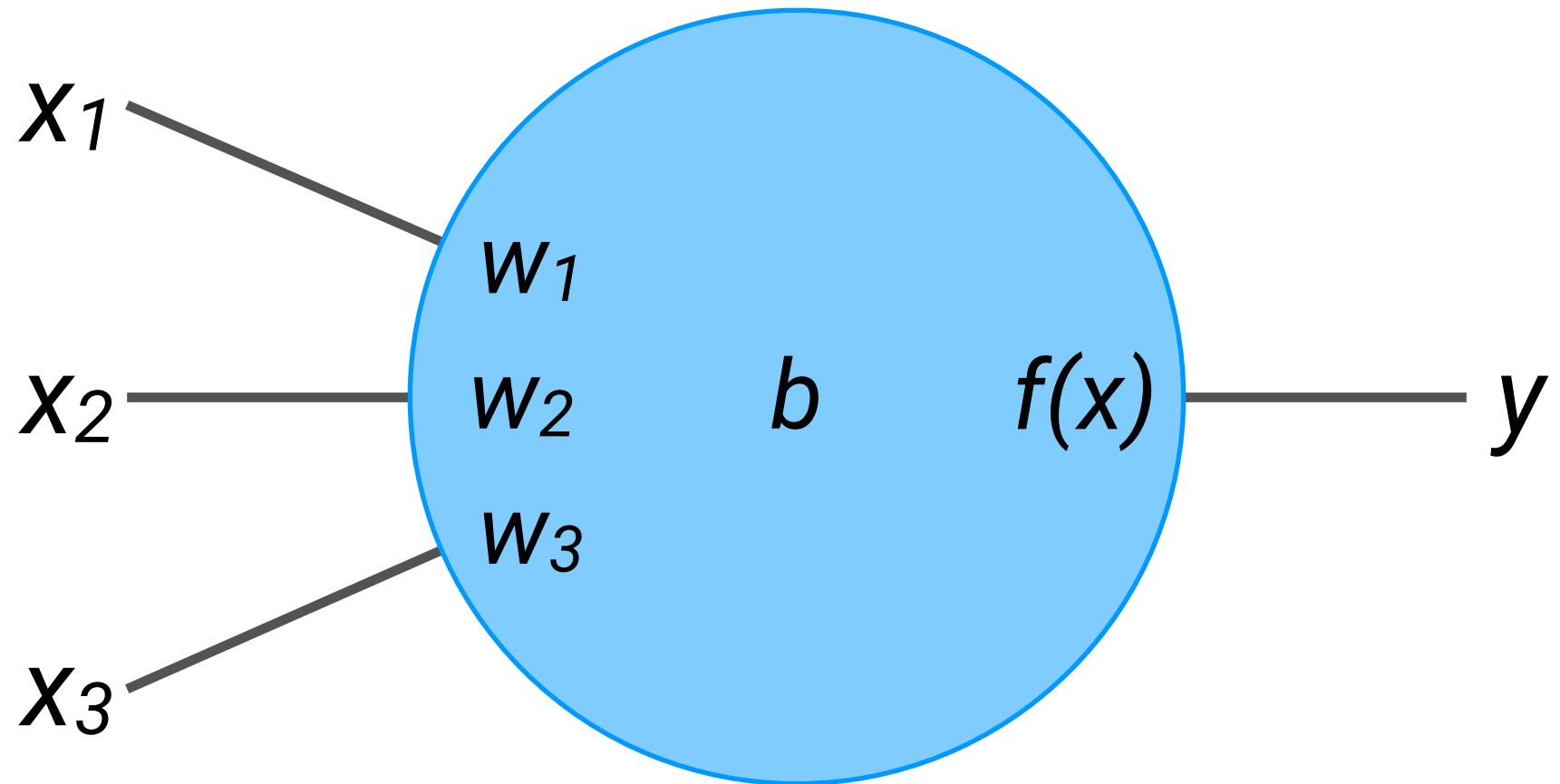
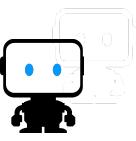
# Neurons



# Neurons with Actual Numbers

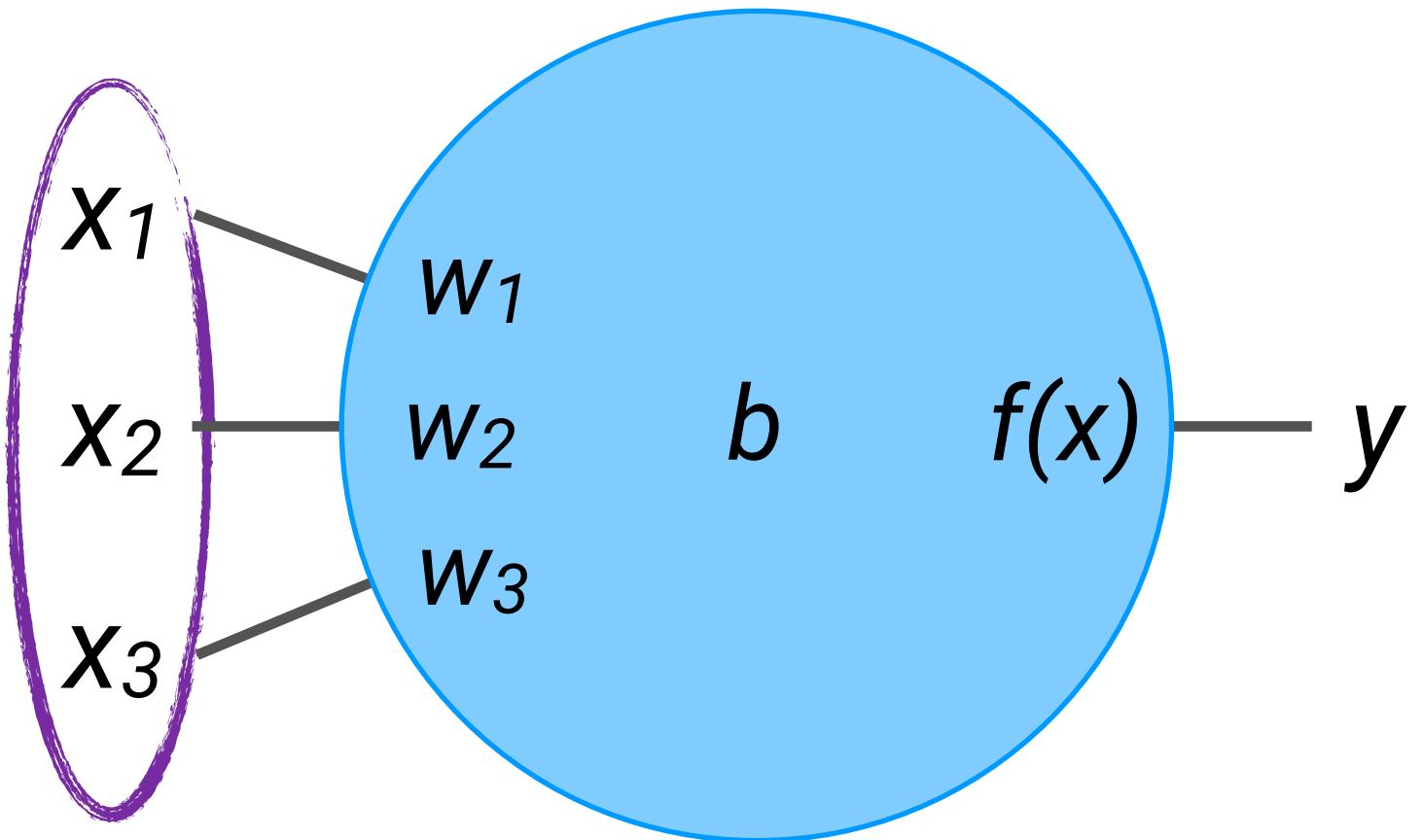


# Inside a Neuron

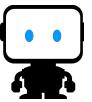




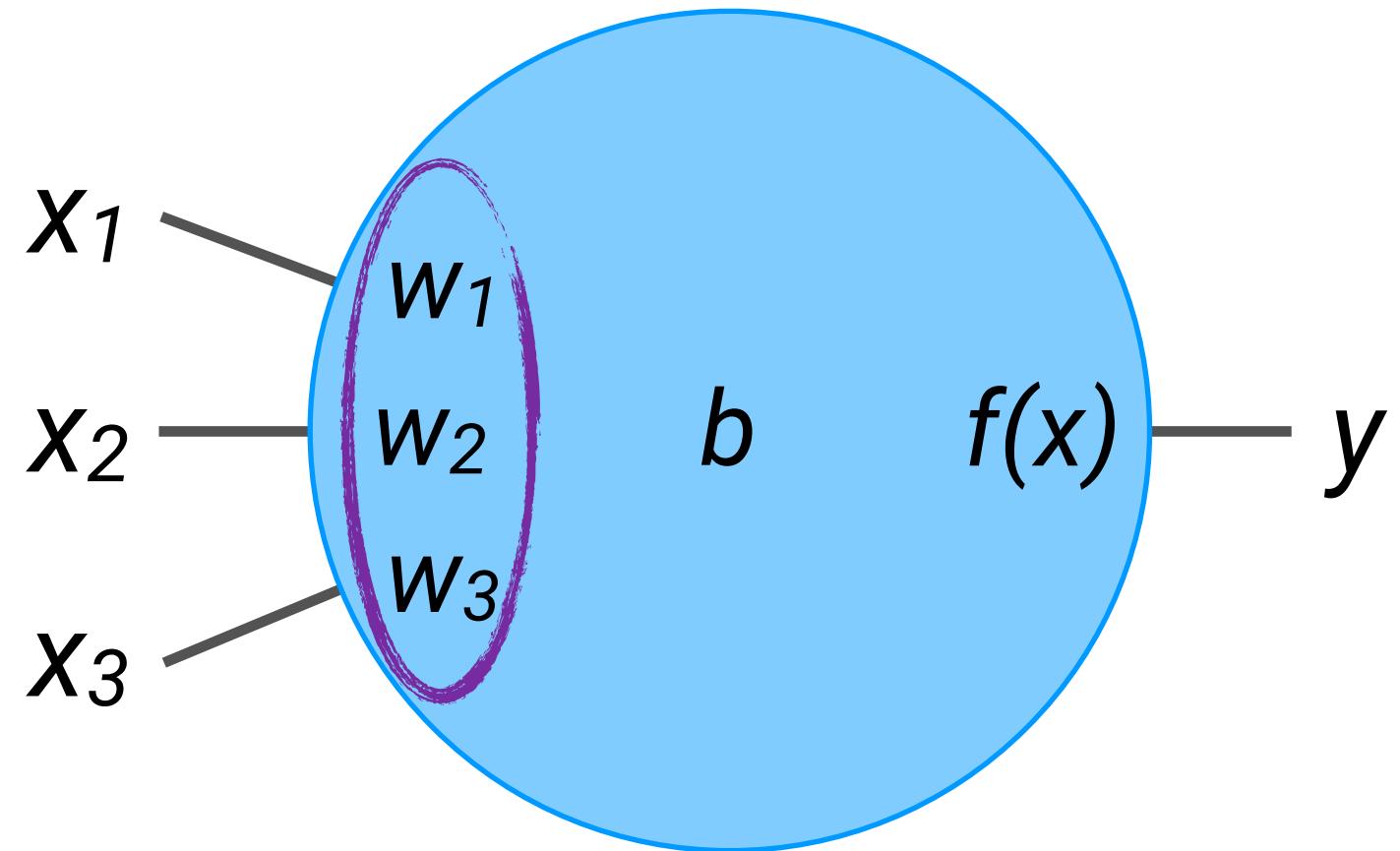
# Inputs



The values coming into the neuron.



# Weights

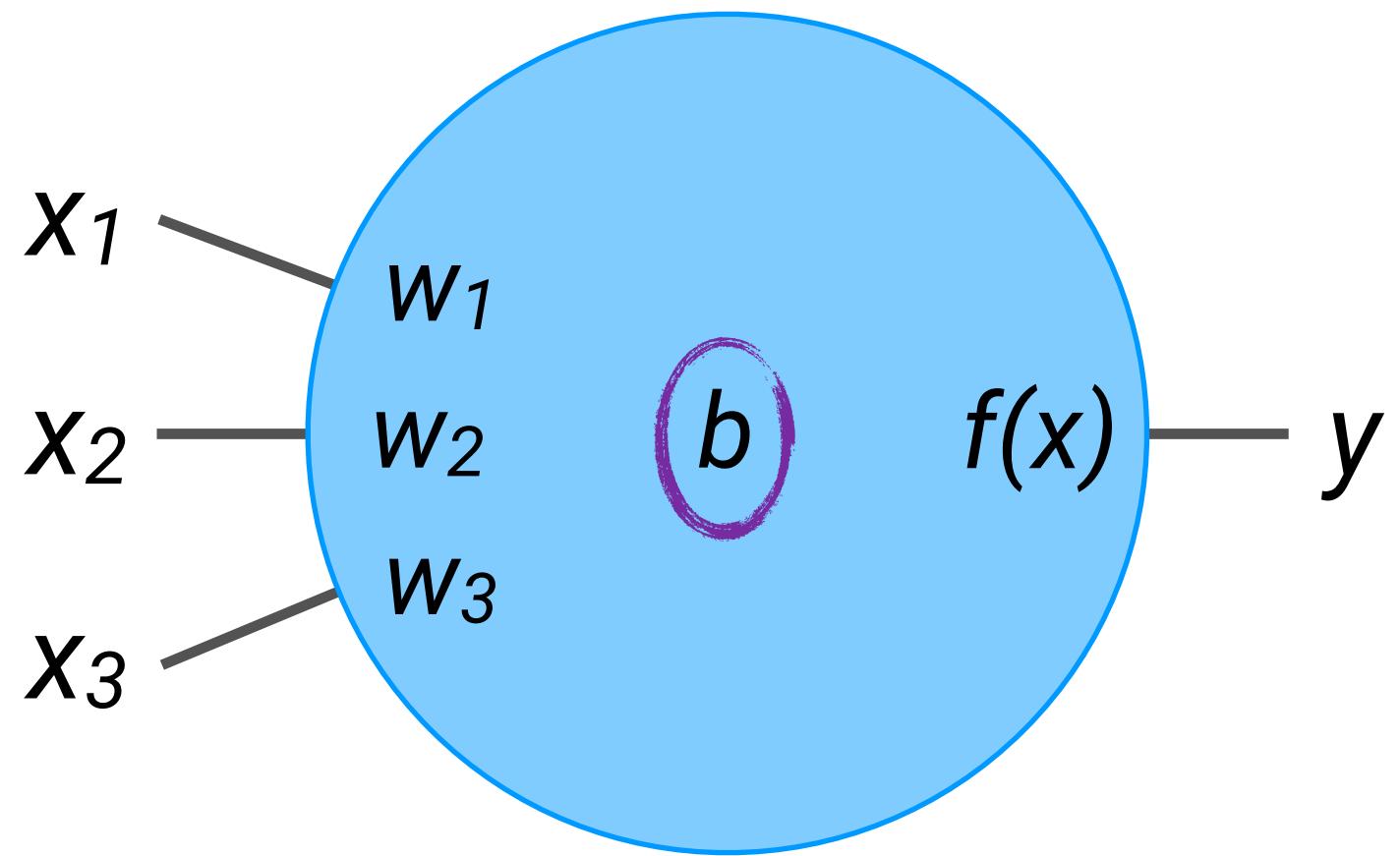


Makes an input more or less important.



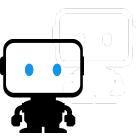


# Bias

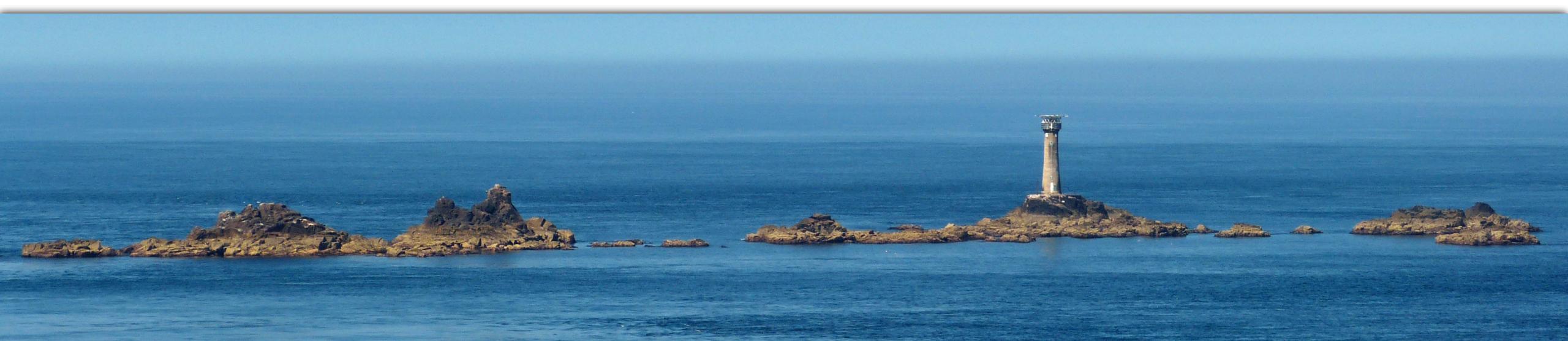


Makes the sum of the inputs more or less important.

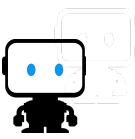
# The Math So Far



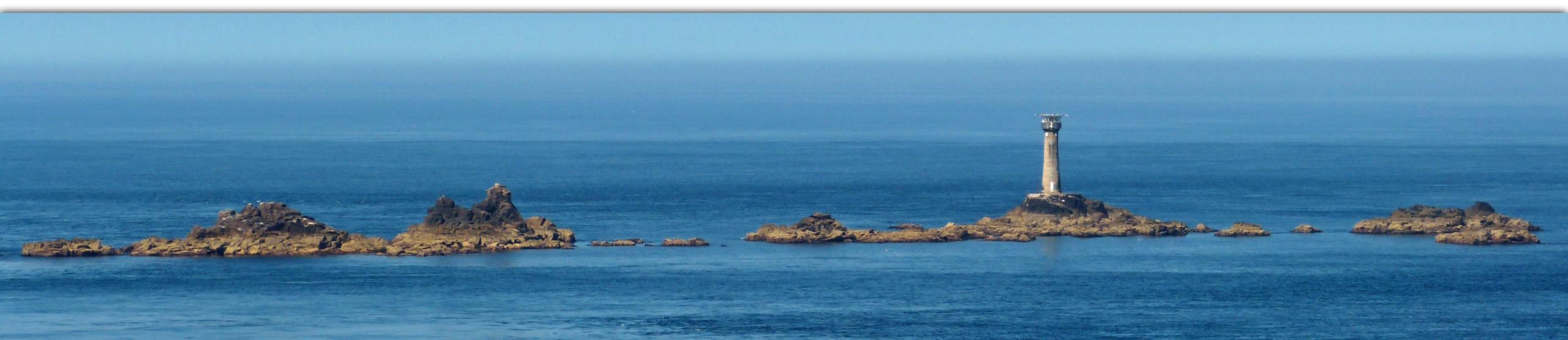
$$x_1w_1 + x_2w_2 + x_3w_3 + b$$

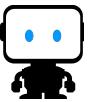


# Even Mathier

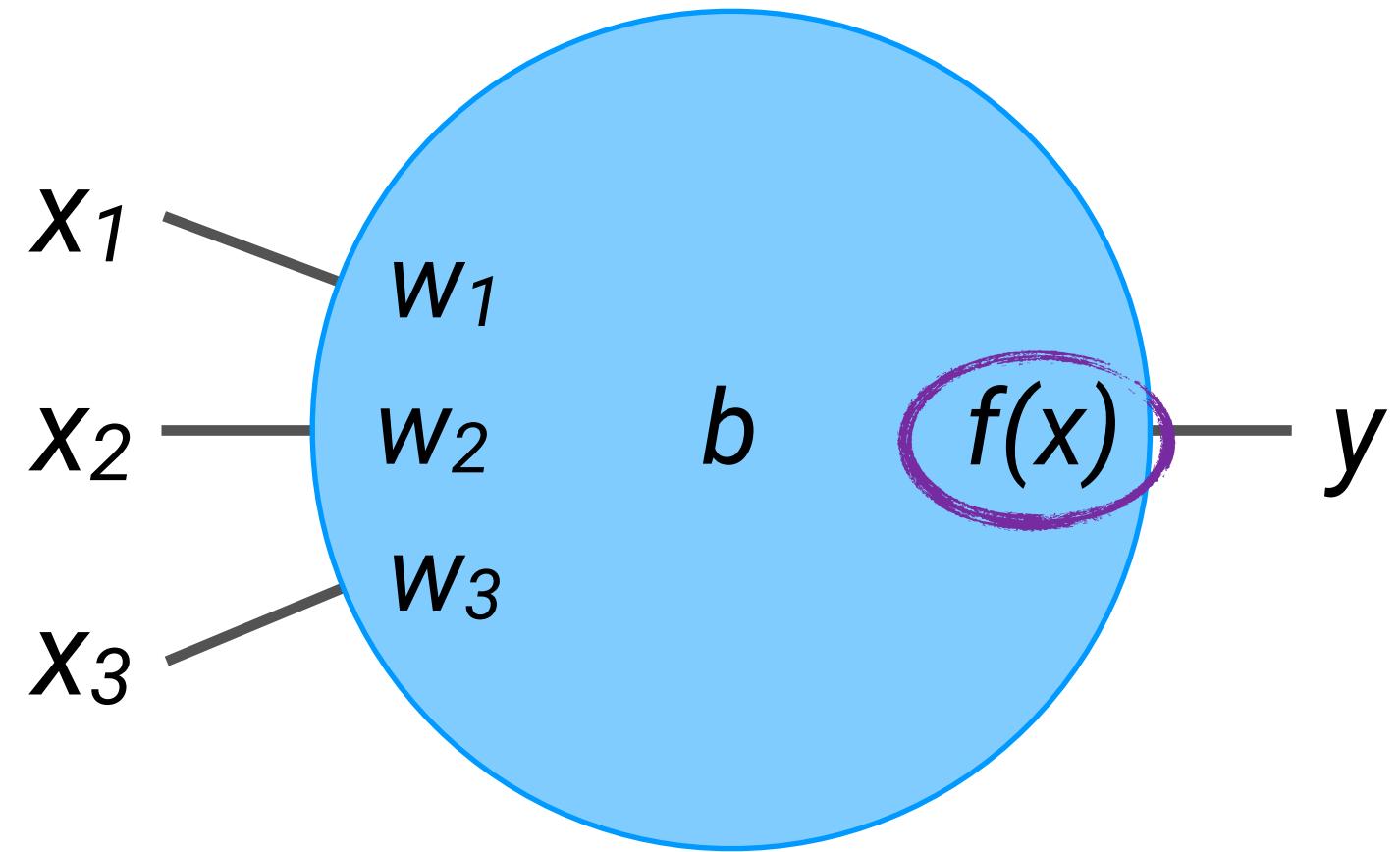


$$\sum_i x_i w_i + b$$





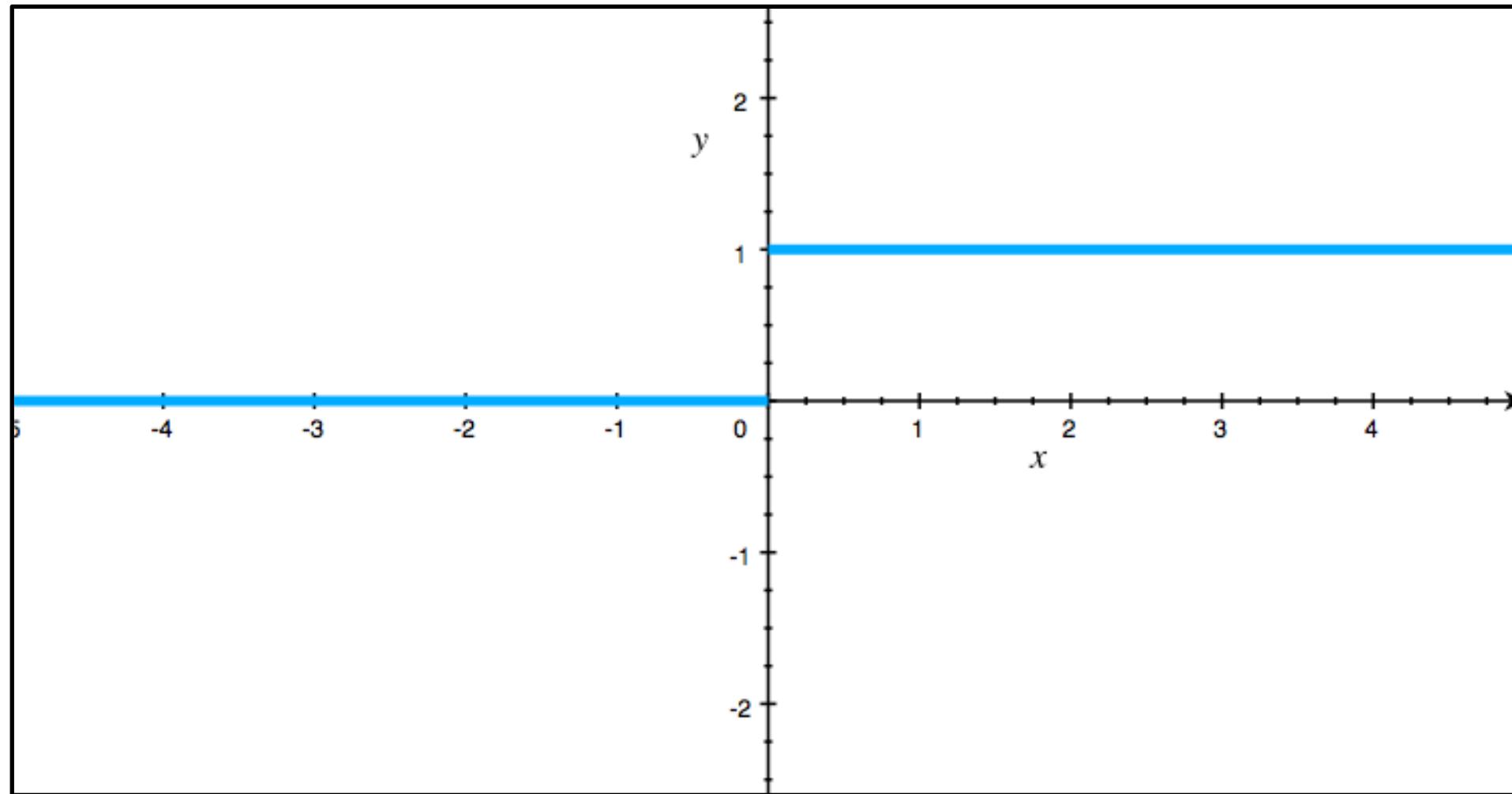
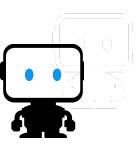
# Activation Function



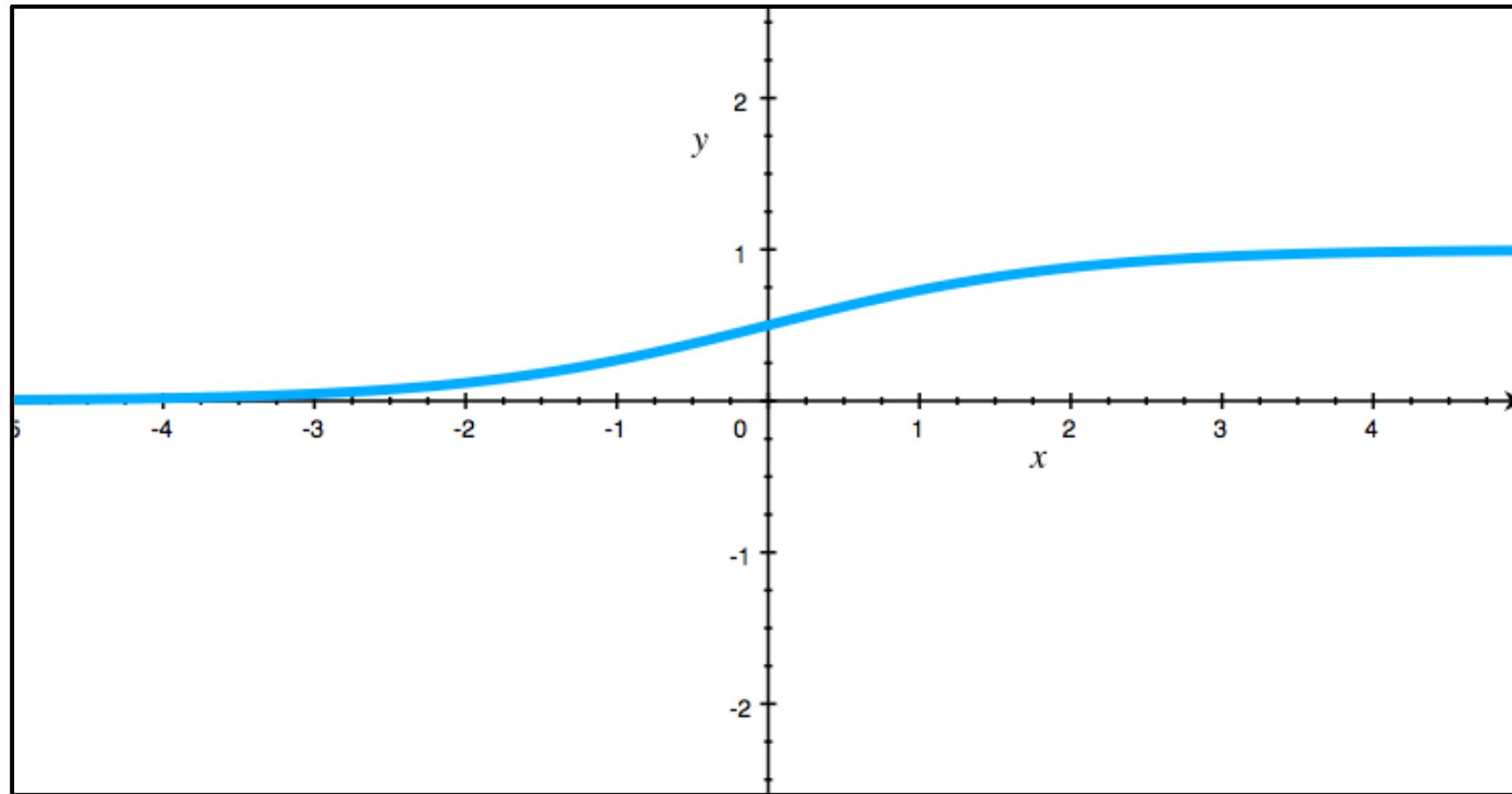
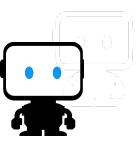
Adjusts the output of the neuron.



# Step Function



# Sigmoid Function



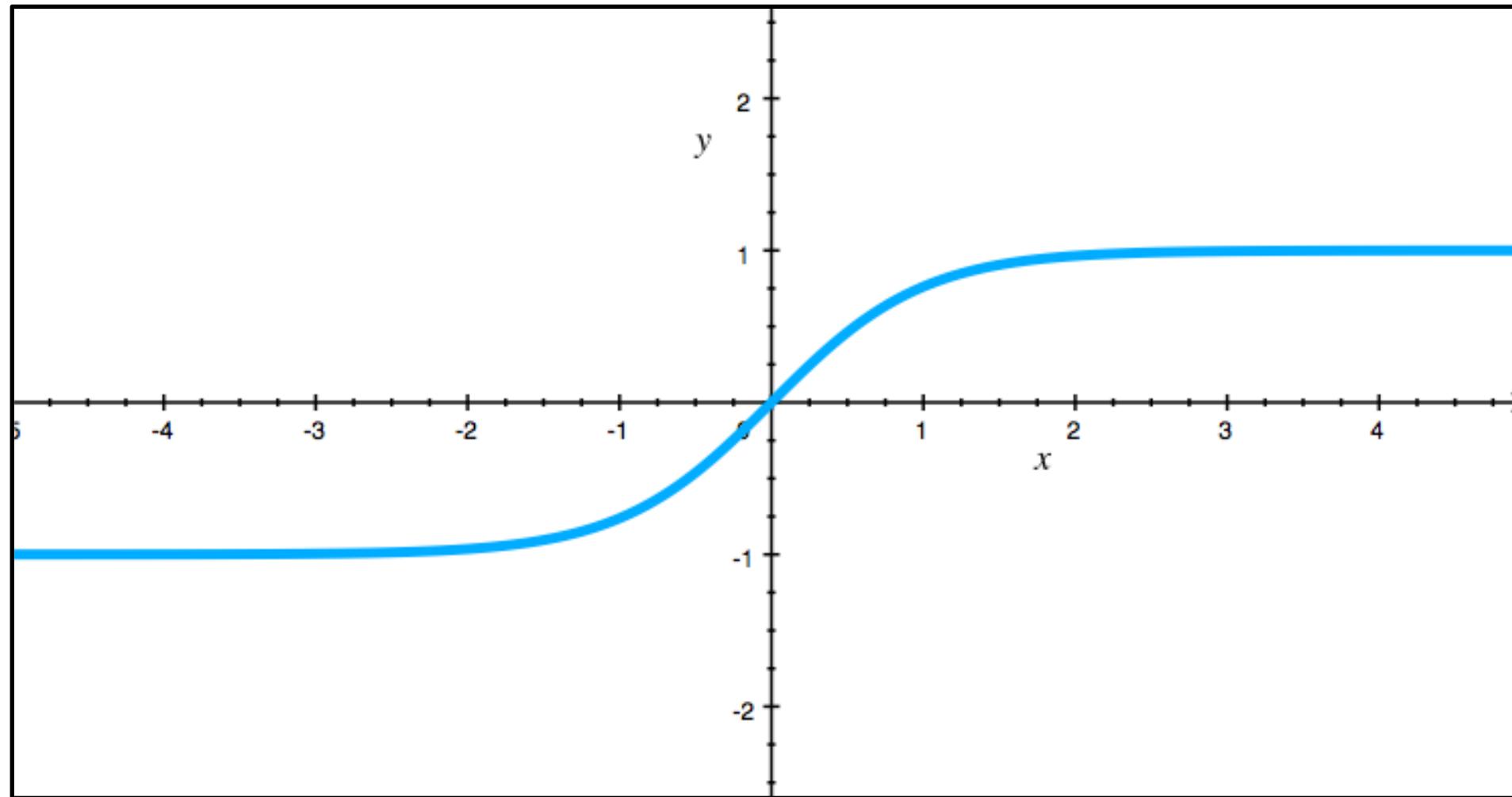
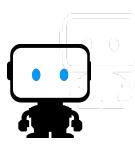


Σ

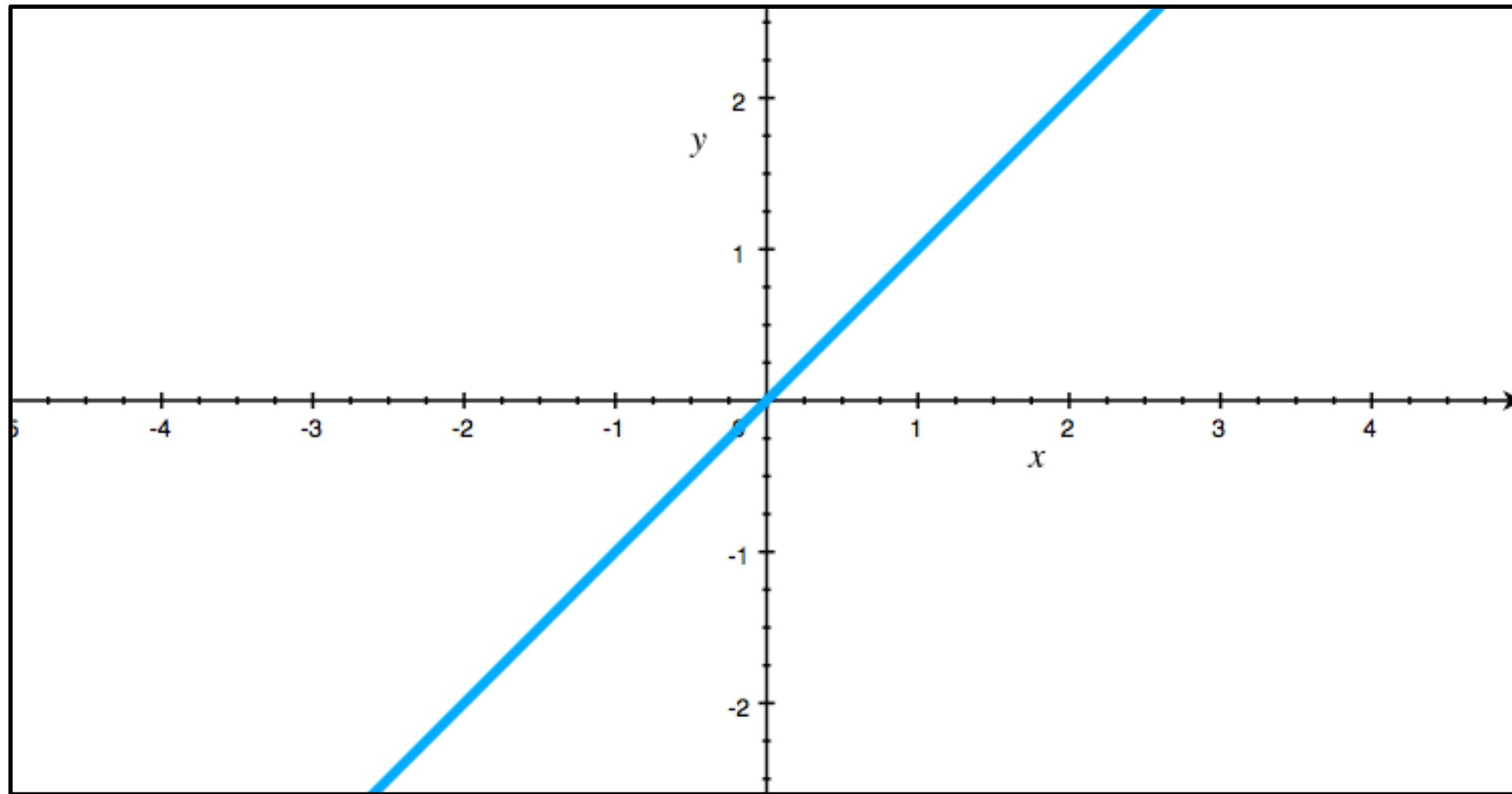
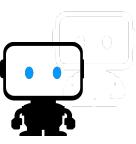
S

↪

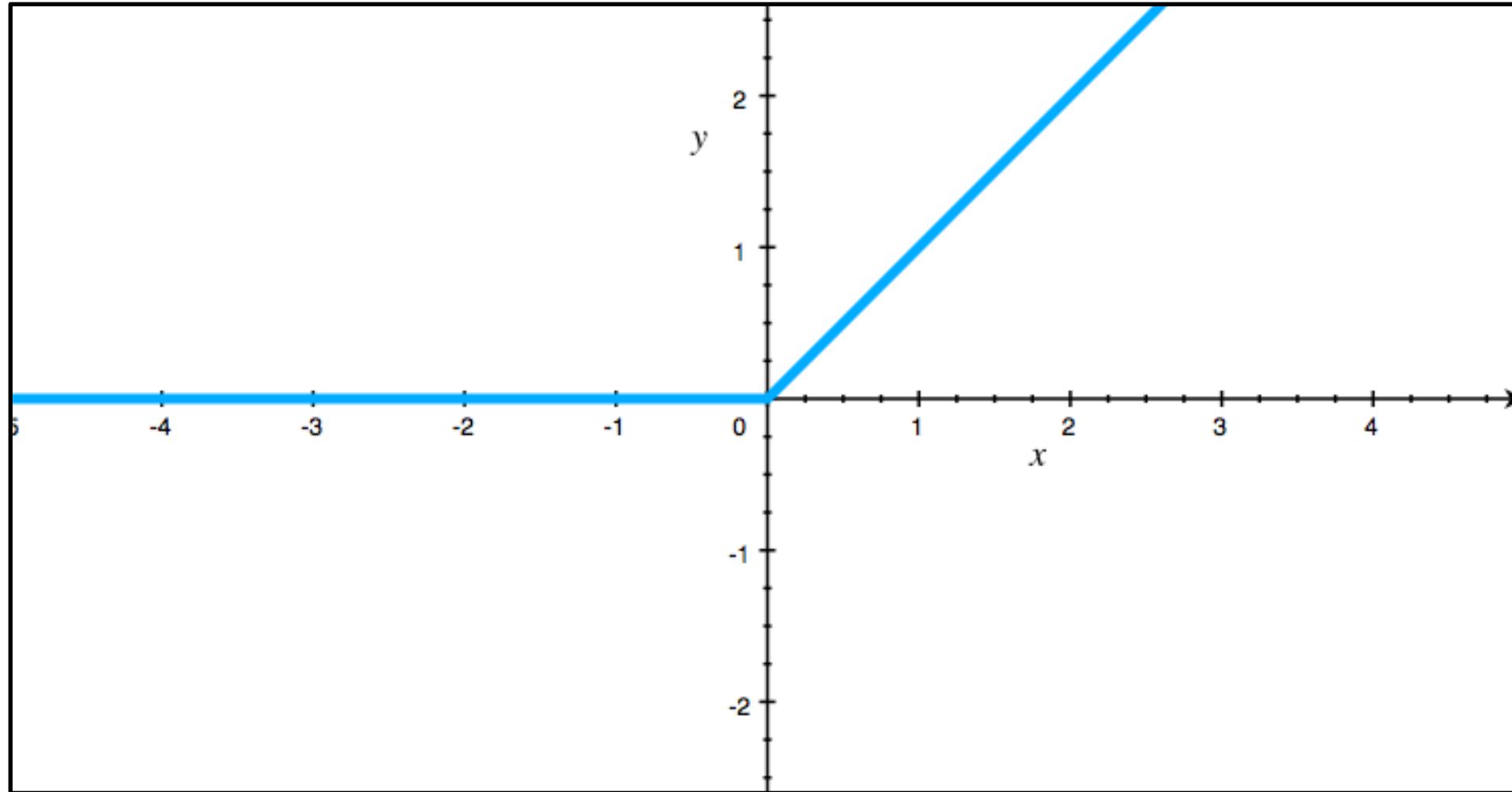
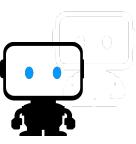
# tanh Function

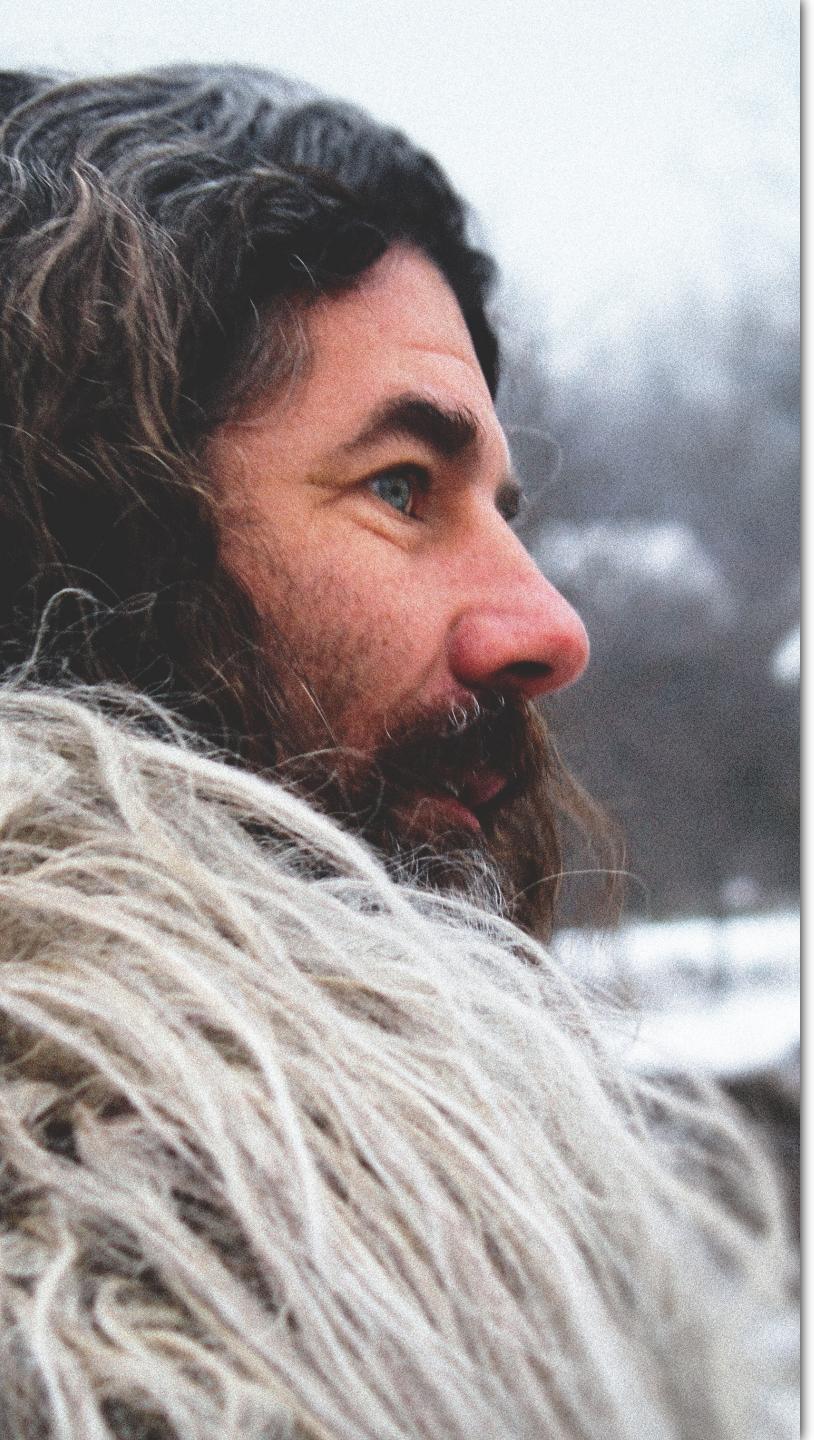


# Linear Function

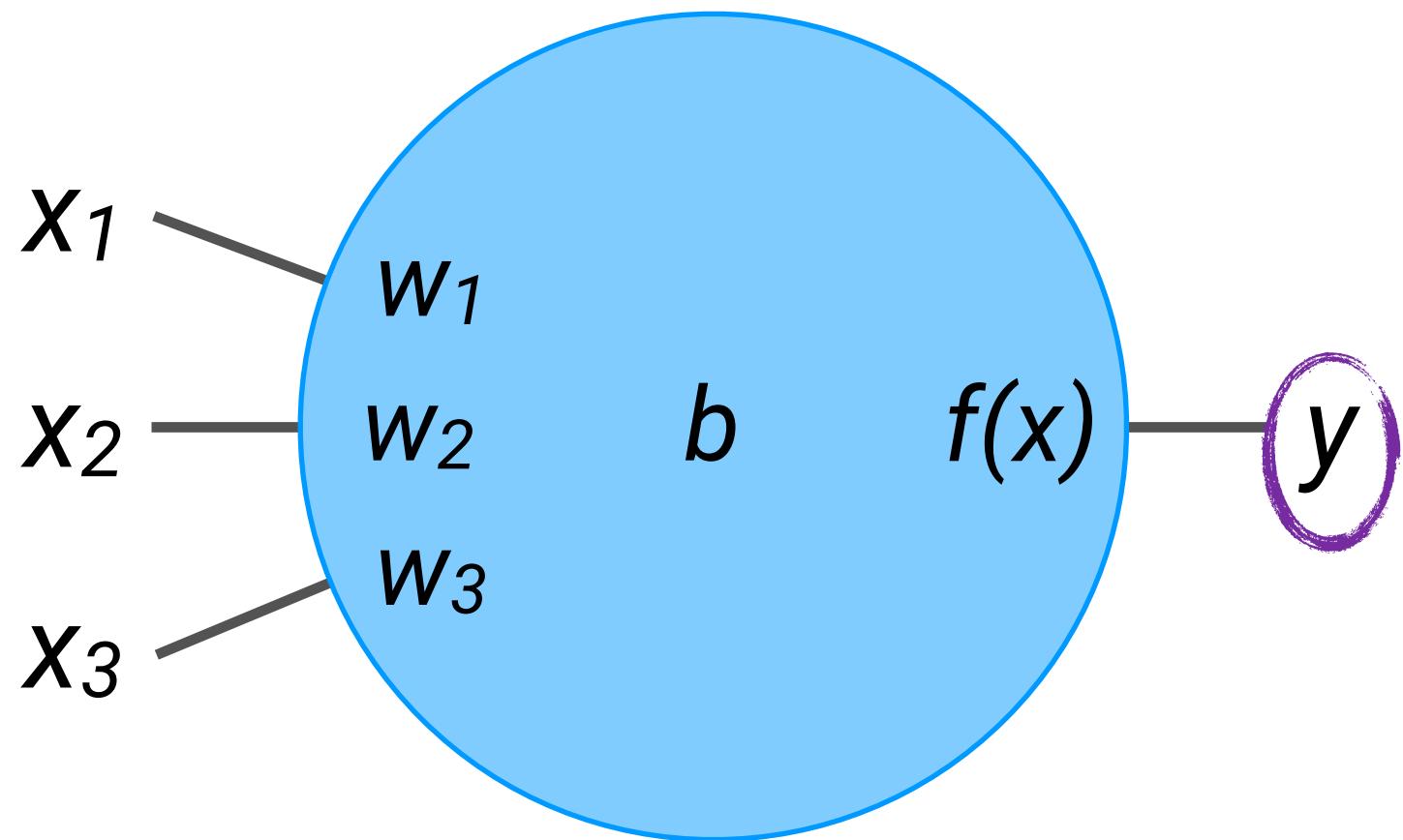
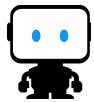


# Rectified Linear Units (ReLU)



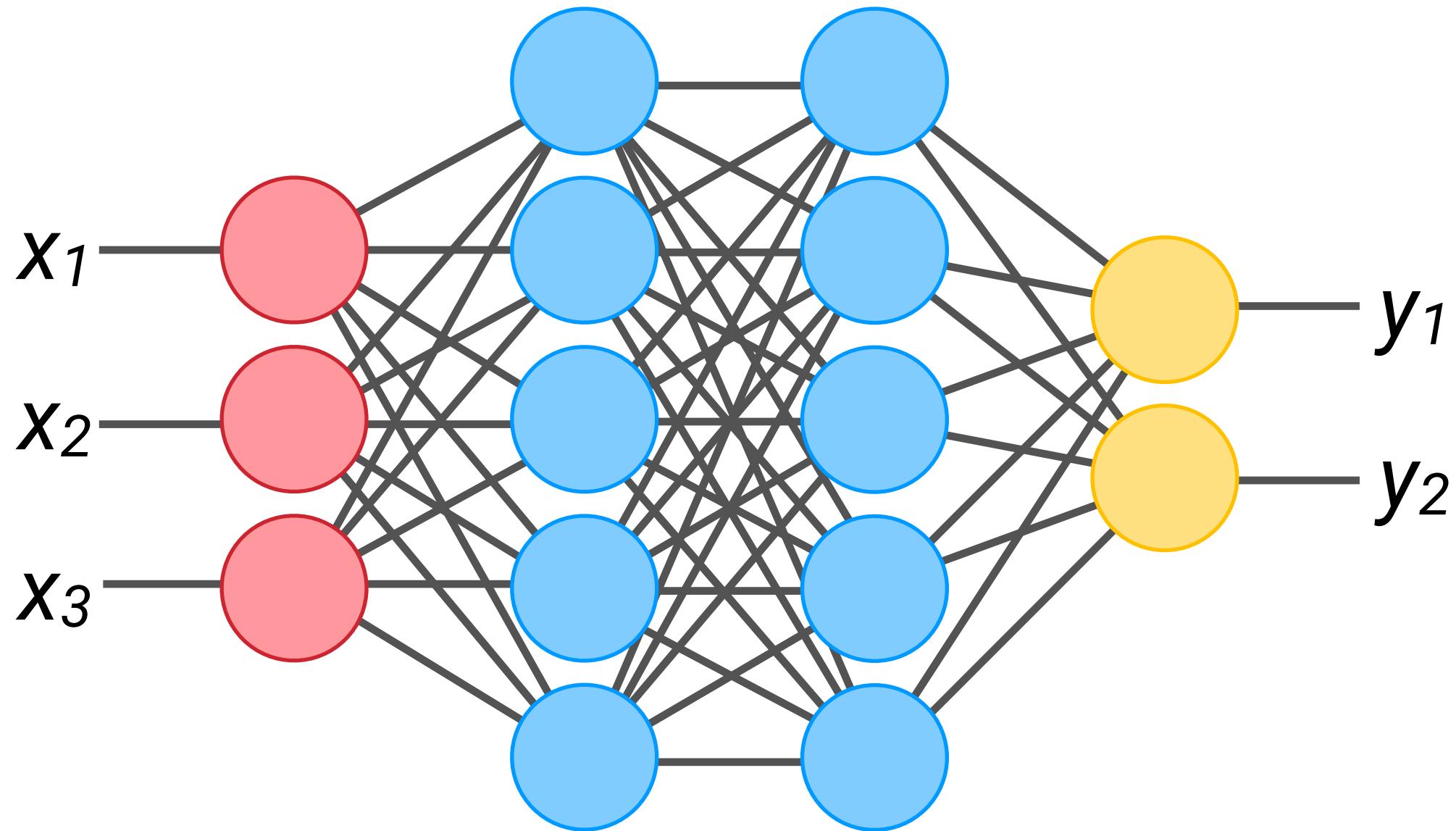
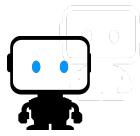


# Output

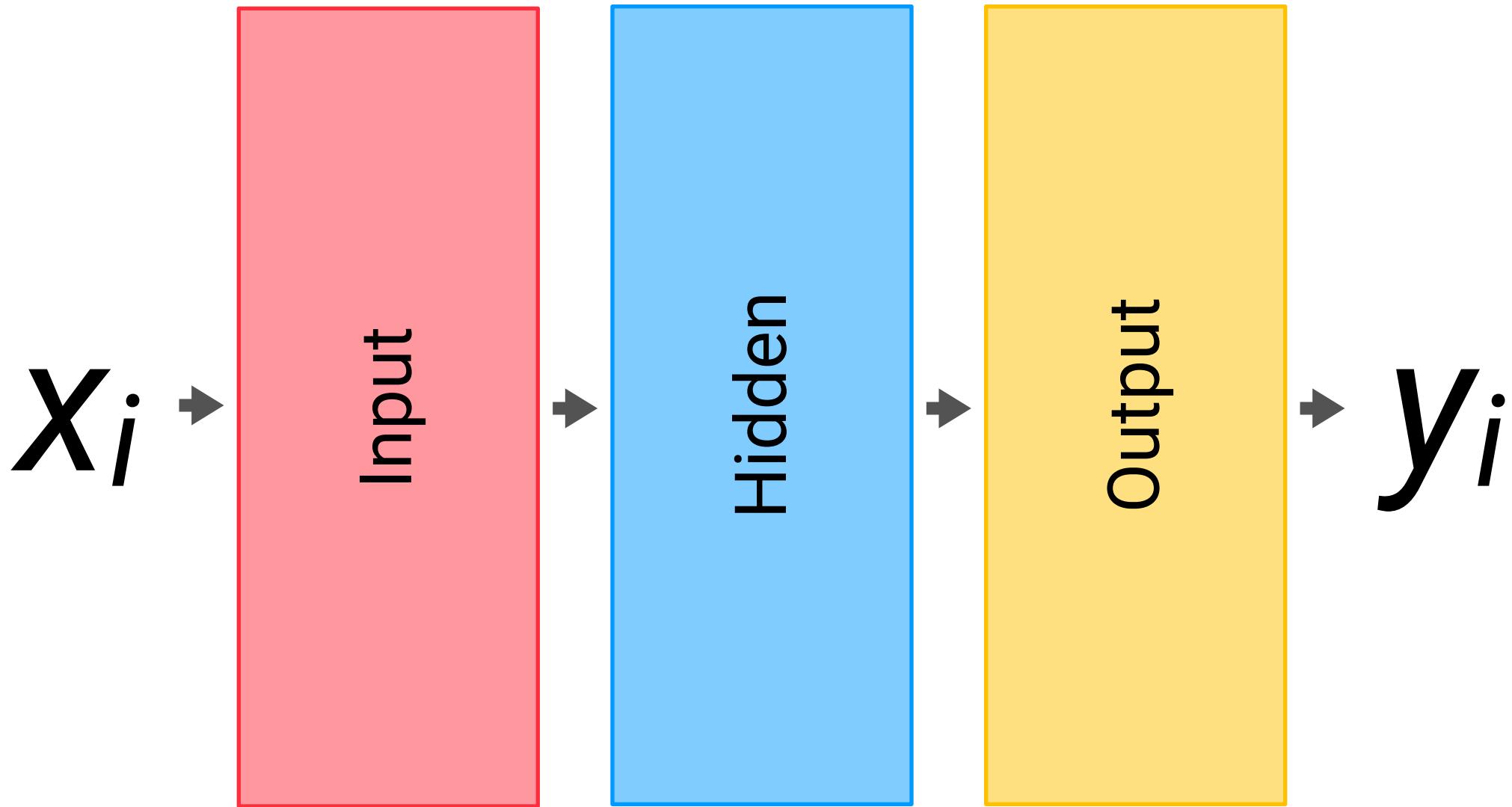
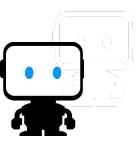


The value leaving the neuron.

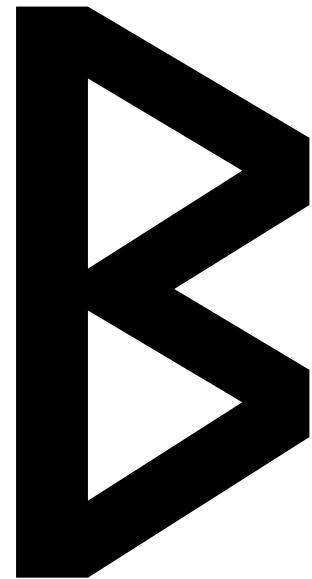
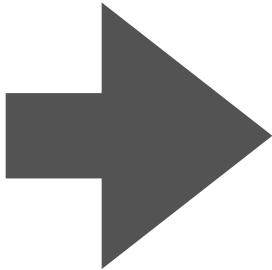
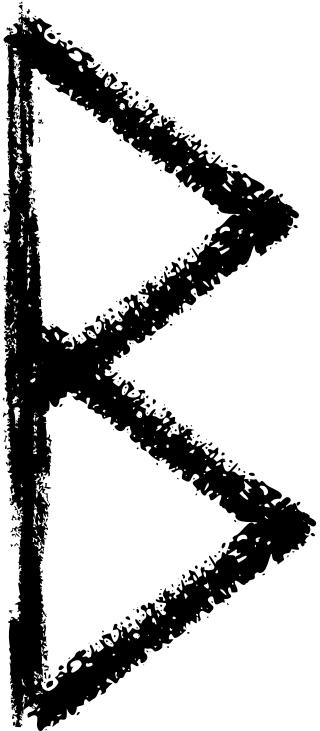
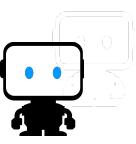
# Neural Networks



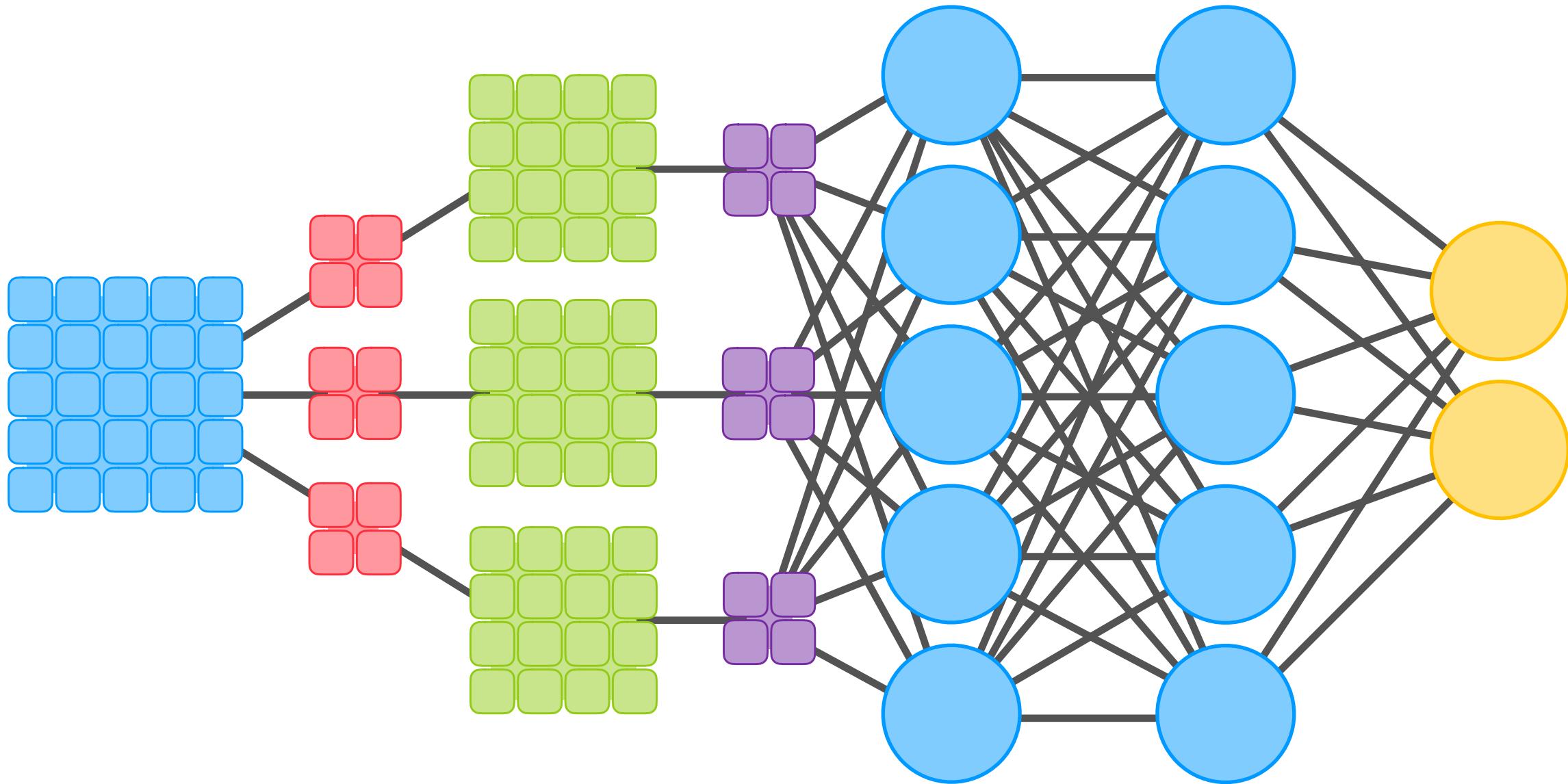
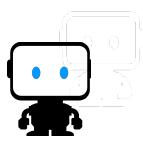
# Layers of a Neural Network



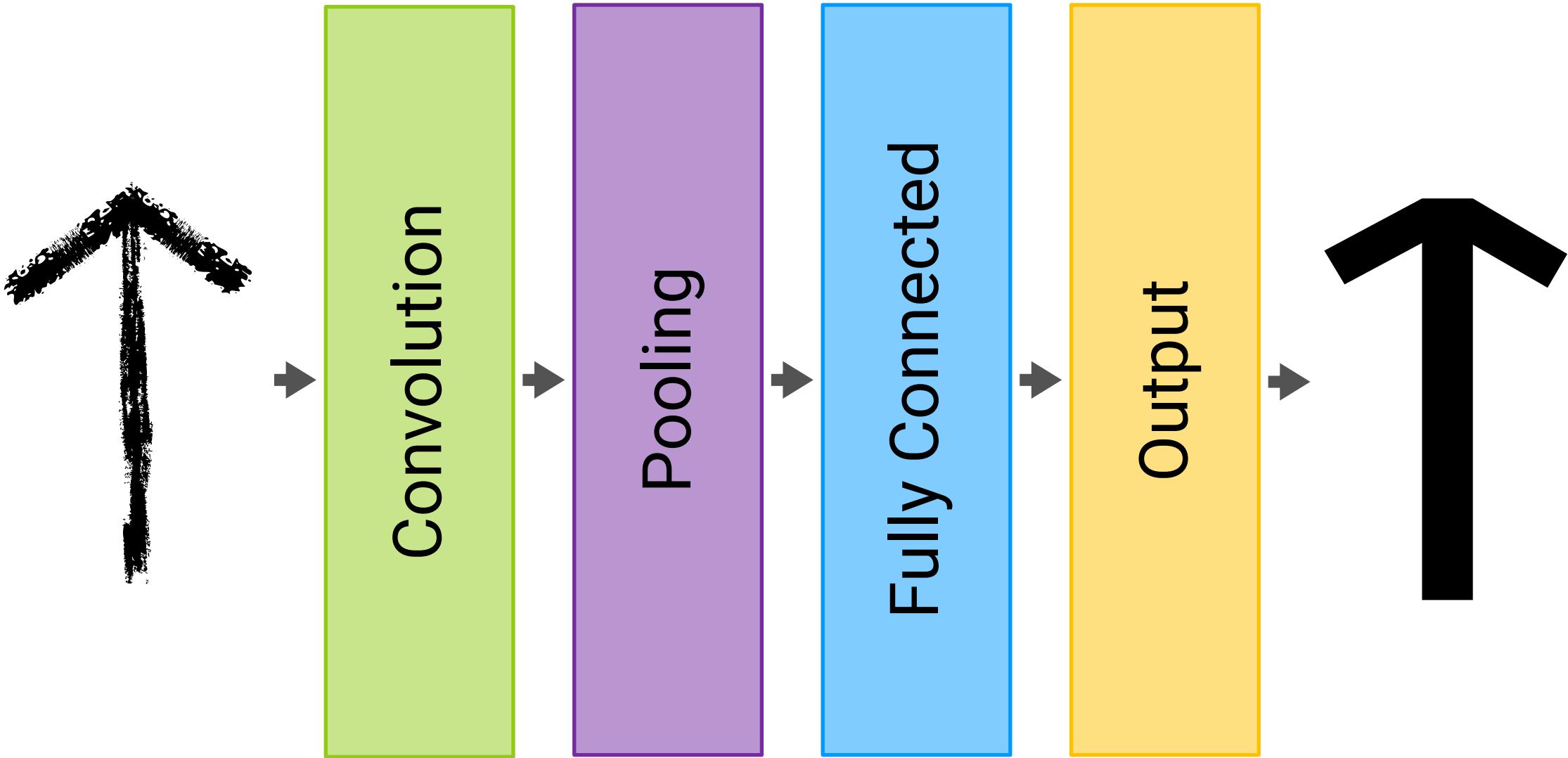
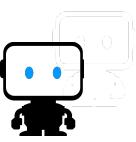
# Bad at Recognizing Runes



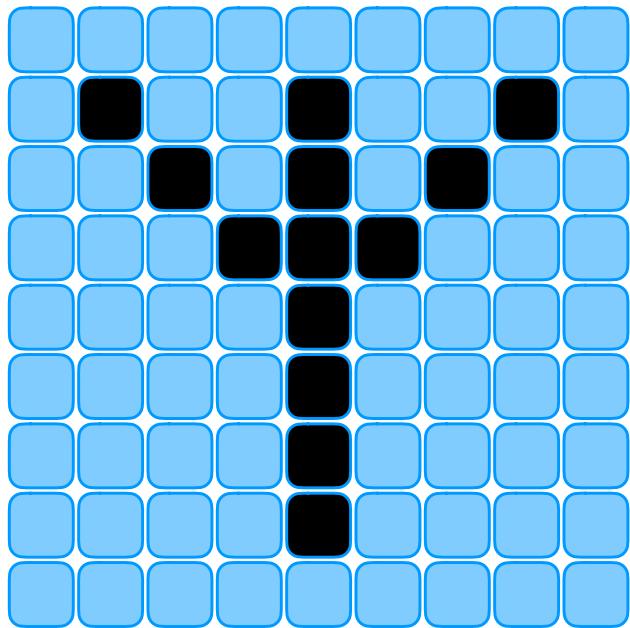
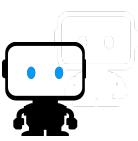
# Convolutional Neural Networks



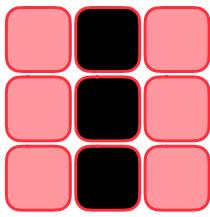
# Layers of a Convolutional Neural Network



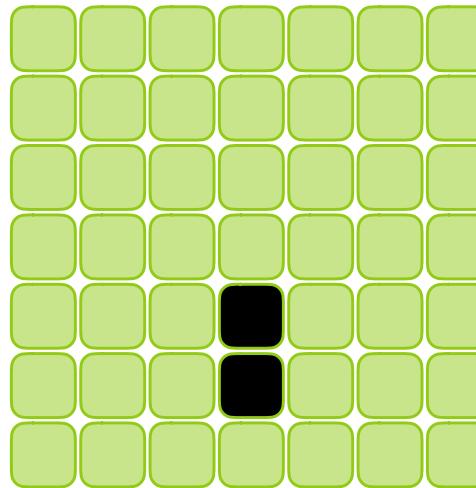
# Convolutions



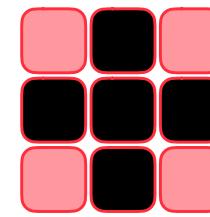
Input



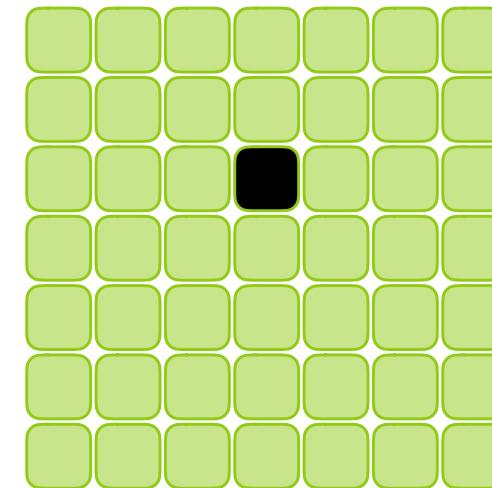
Filter



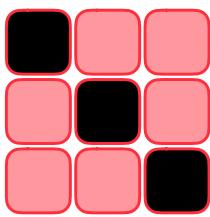
Output



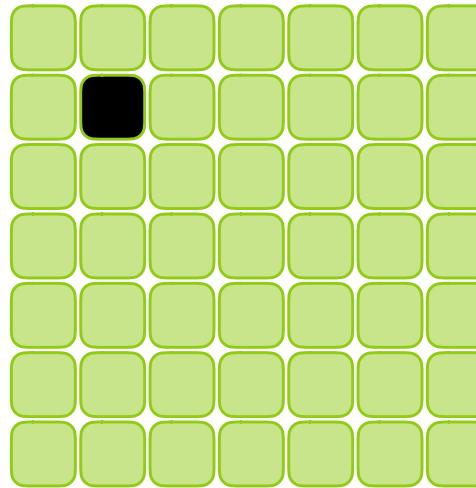
Filter



Output

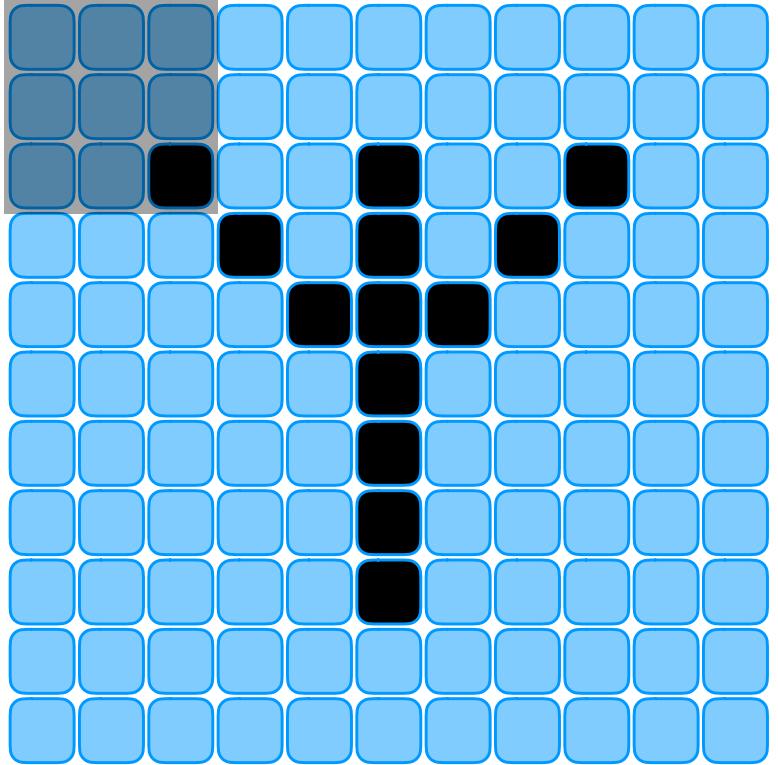
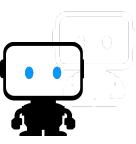


Filter

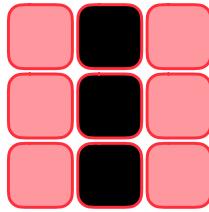


Output

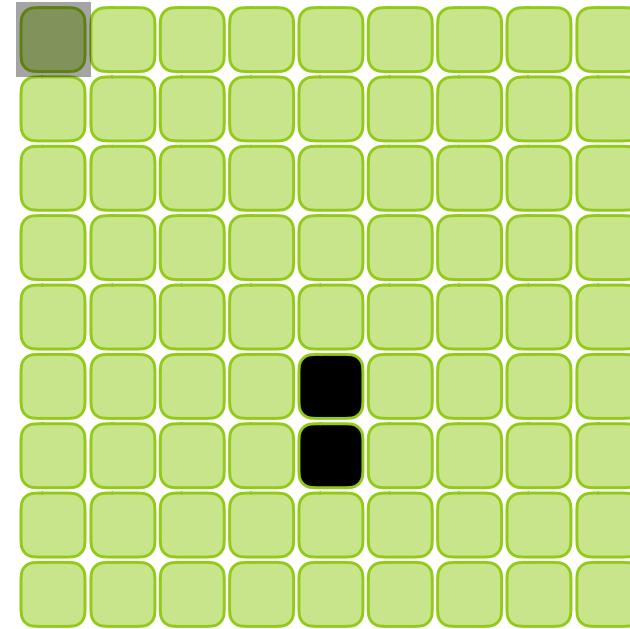
# Convolutions with Fancy Animation



Input

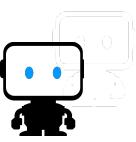


Filter



Output

# Convolutions with Numbers



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Input

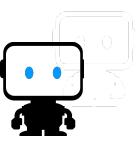
-1	1	-1
-1	1	-1
-1	1	-1
-2		

Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Output

# Convolutions with Familiar Looking Numbers



$X_i$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\sum_i X_i W_i + b$$

$W_i$

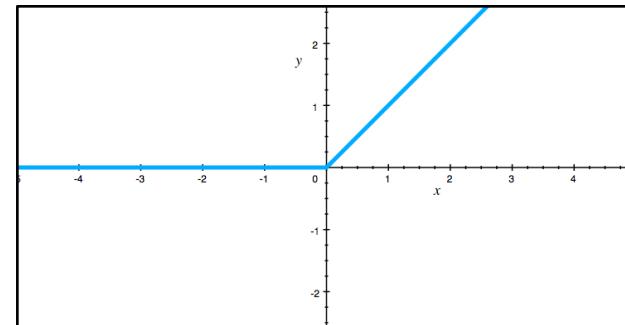
-1	1	-1
-1	1	-1
-1	1	-1

-2

$b$

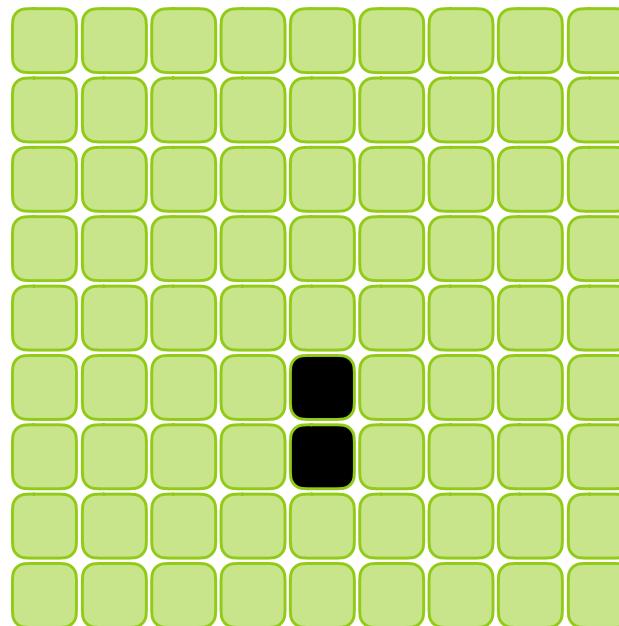
$y_i$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



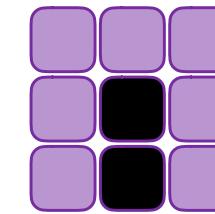


# Pooling



Output

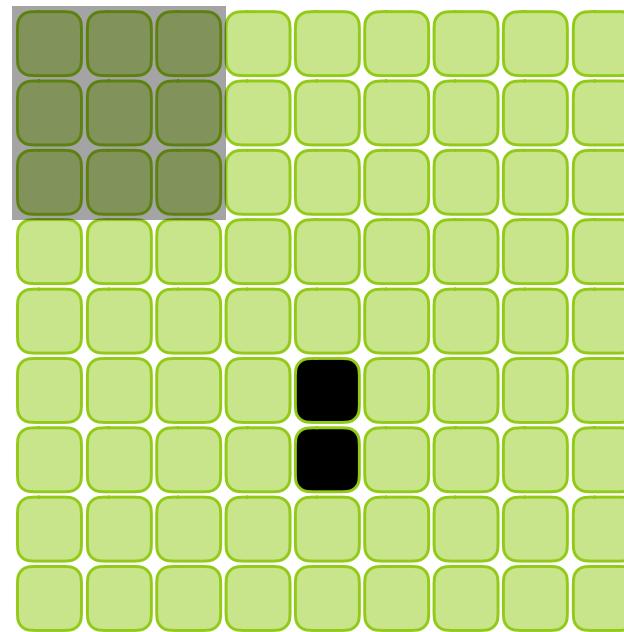
$3 \times 3$



Pooled

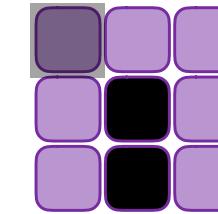


# Pooling with Fancy Animation



Output

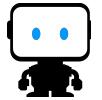
$3 \times 3$



Pooled



# Pooling with Numbers



0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Output

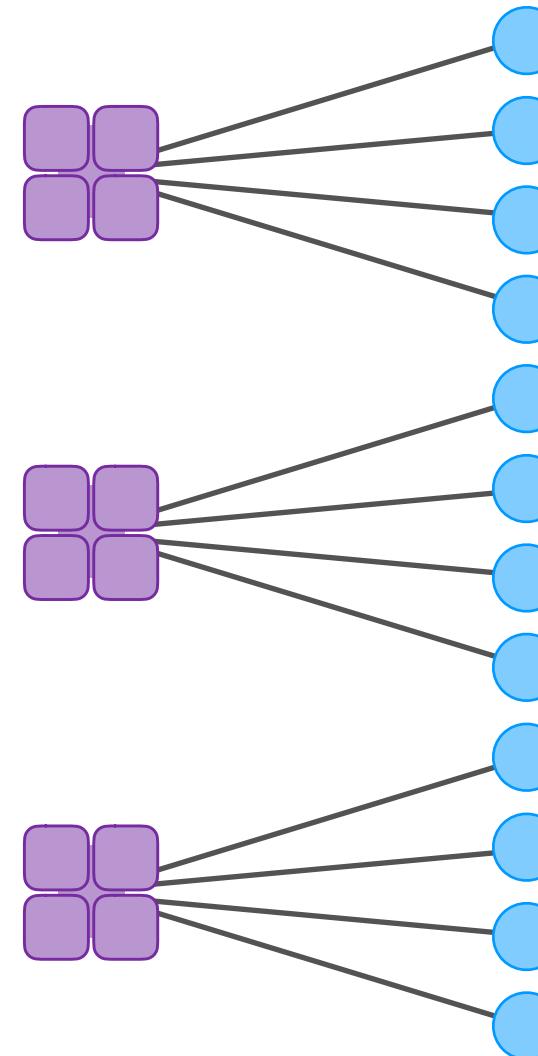
3 x 3

0	1	0
0	2	0
0	2	0

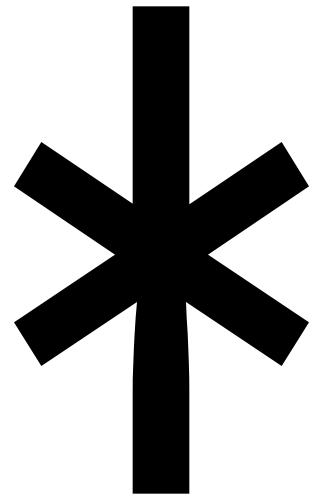
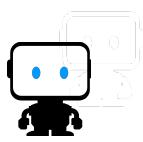
Pooled



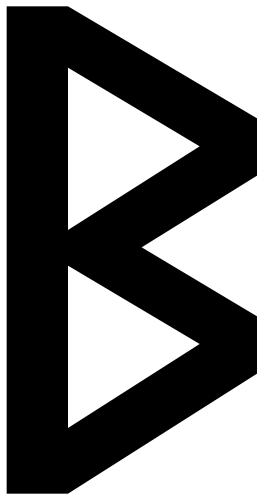
# Flattening



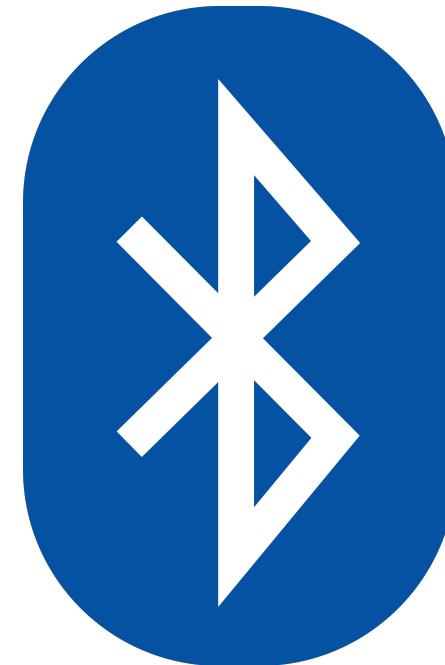
# Harald Bluetooth



+



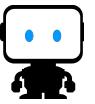
=





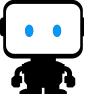
# Keras



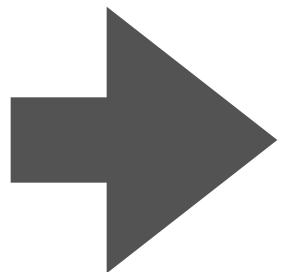


```
model = Sequential()  
  
model.add(Conv2D(48, (3, 3), activation='relu', input_shape=(1, 24, 24)))  
  
model.add(Conv2D(24, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2,2)))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dense(16, activation='softmax'))  
  
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])  
  
model.fit(X_train, Y_train, batch_size=32, epochs=20, verbose=1)
```



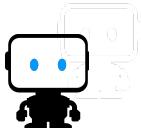


↑ | Ψ \*



Demo

(Tima)



# Resources

## Keras

<https://keras.io/>

## Microsoft CNTK

<https://www.microsoft.com/en-us/cognitive-toolkit/>

## TensorFlow

<https://www.tensorflow.org/>

## Theano

<http://www.deeplearning.net/software/theano/>

## Flask

<http://flask.pocoo.org/>

## VanillaJS

<http://vanilla-js.com/>

## Younger Futhark

[https://en.wikipedia.org/wiki/Younger\\_Futhark](https://en.wikipedia.org/wiki/Younger_Futhark)

## Neural Network Zoo

<http://www.asimovinstitute.org/neural-network-zoo/>

## A Beginner's Guide to Neural Networks

<https://towardsdatascience.com/a-beginners-guide-to-neural-networks-b6be0d442fa4>

## Hacker's Guide to Neural Networks

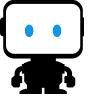
<http://karpathy.github.io/neuralnets/>

## CS231n Convolutional Neural Networks for Visual Recognition

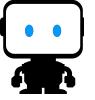
<http://cs231n.github.io/convolutional-networks/>

## An Intuitive Guide to Convolutional Neural Networks

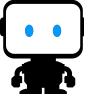
<https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>



**[https://github.com/guyroyse/  
deep-learning-like-a-viking](https://github.com/guyroyse/deep-learning-like-a-viking)**



# Questions?



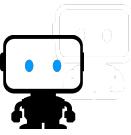
# **Guy Royse**

Developer Evangelist  
DataRobot

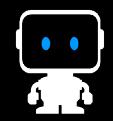
 @guyroyse

 [github.com/guyroyse](https://github.com/guyroyse)

# Image Credits



- <https://www.flickr.com/photos/frankdouwes/3985117642/>
- <https://www.flickr.com/photos/ecastro/4415693080/>
- <https://www.flickr.com/photos/torsven/2869528719/>
- [https://www.flickr.com/photos/arg\\_flickr/14732931742/](https://www.flickr.com/photos/arg_flickr/14732931742/)
- <https://www.flickr.com/photos/wwarby/23841229208/>
- <https://www.flickr.com/photos/hesim/6553273471/>



Data**Robot**