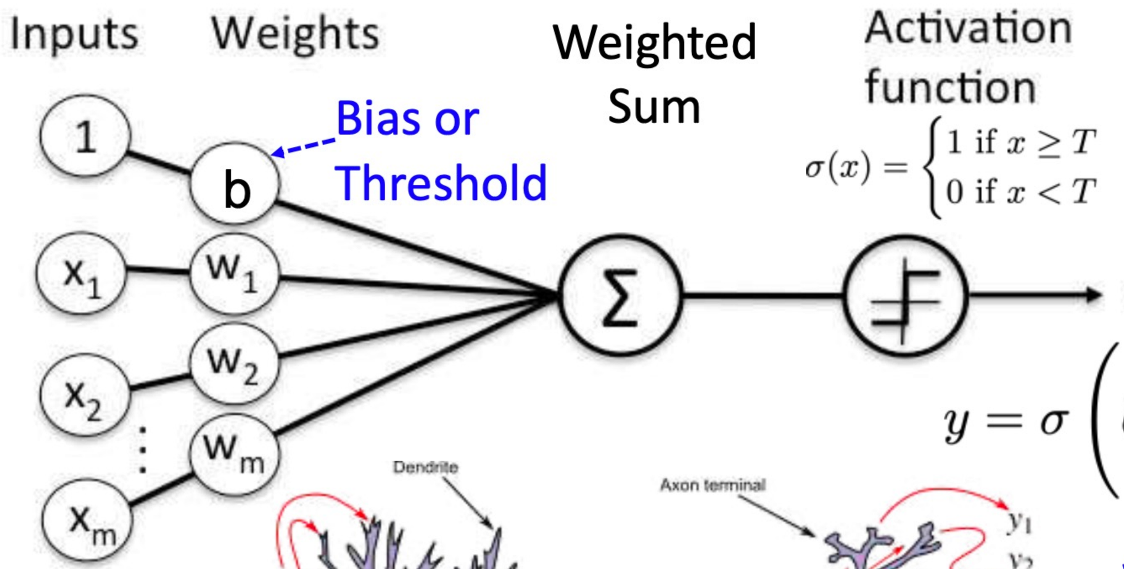
Deep Differentiable Logic Gate Networks

Felix Petersen, Christian Borgelt, Hilde Kuehne, Oliver Deussen

Interpreted by:
John Tan Chong Min

Perceptron

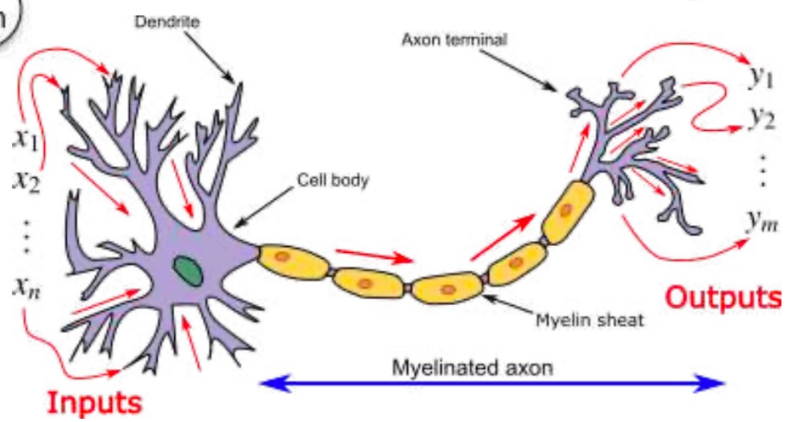


$$\sigma(x) = \begin{cases} 1 \text{ if } x \geq T \\ 0 \text{ if } x < T \end{cases}$$

$$y = \sigma\left(b + \sum_{i=1}^{m} w_i x_i\right)$$

Weighted linear combination of inputs

Compare to



The weights are bias are learned during training

Source: Mehul Motani’s Neural Network Notes

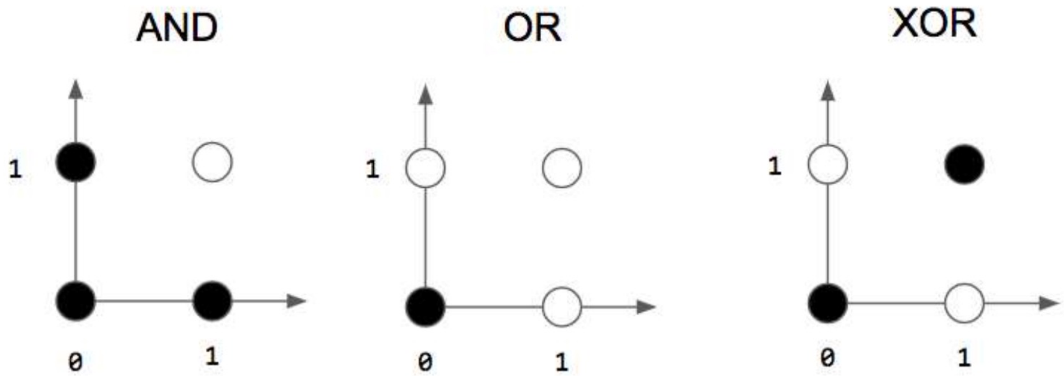
Logic Gates

AND | Inclusive OR | Exclusive OR

Inputs		Output
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

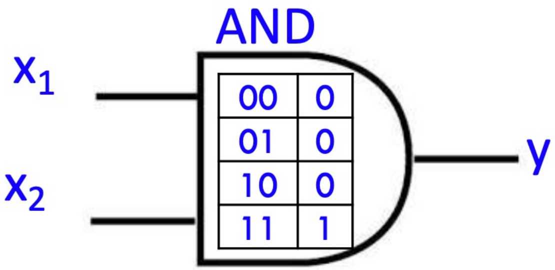
Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

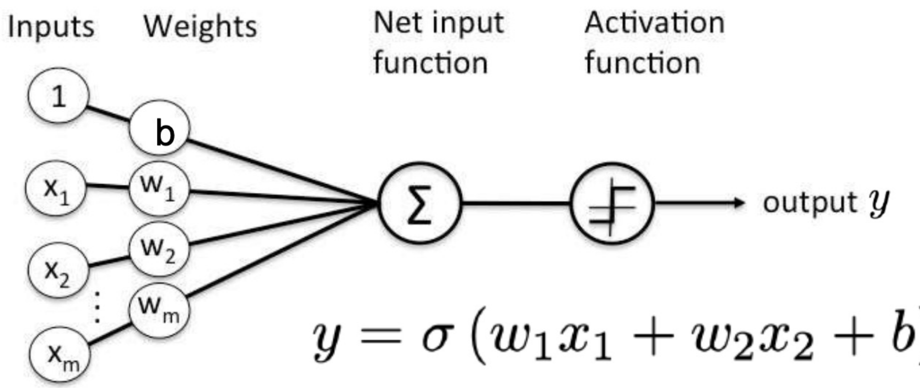


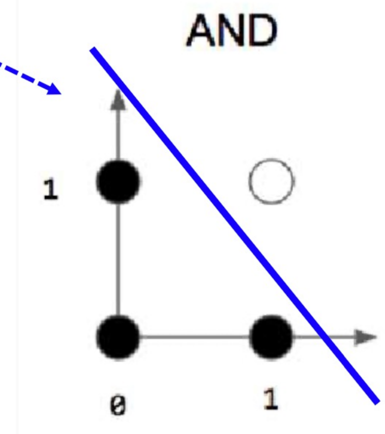
Source: Mehul Motani’s Neural Network Notes

Perceptron is a linear separator



Linearly separable set



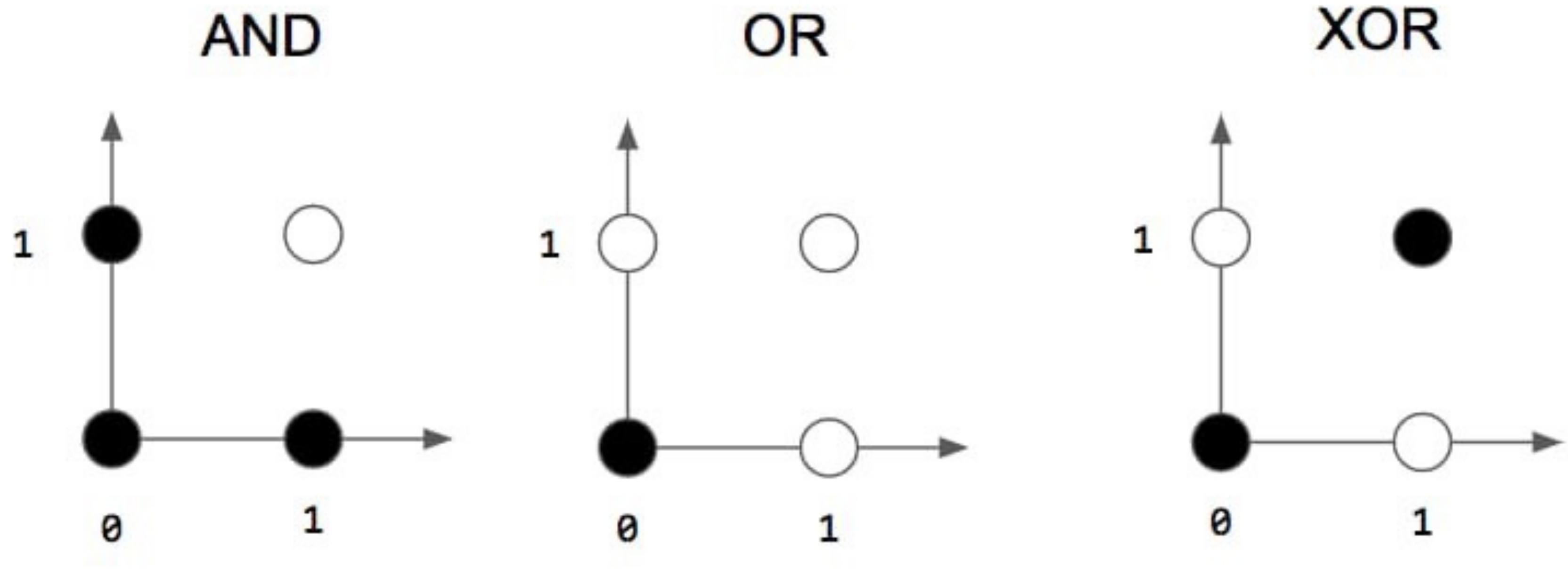


$$y = \sigma\left(w_1 x_1 + w_2 x_2 + b\right)$$ ⟵---- Equation of line

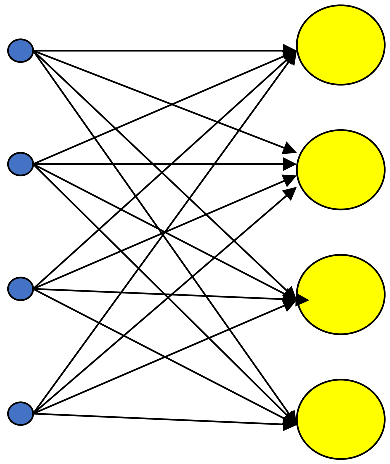
Source: Mehul Motani's Neural Network Notes

The XOR problem

- Try to draw a line to separate the black and white circles

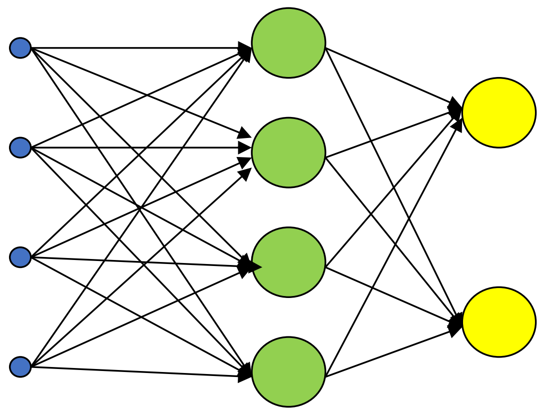


Multi-Layer Perceptron



Single Layer

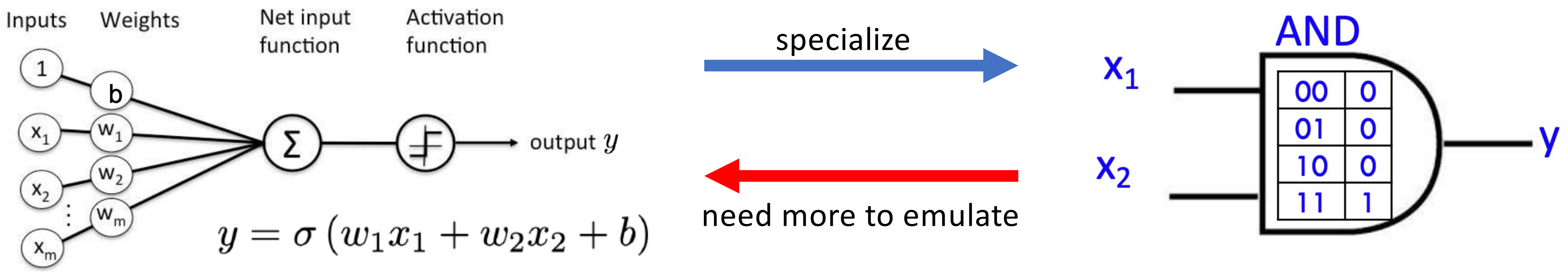
(Perceptron)



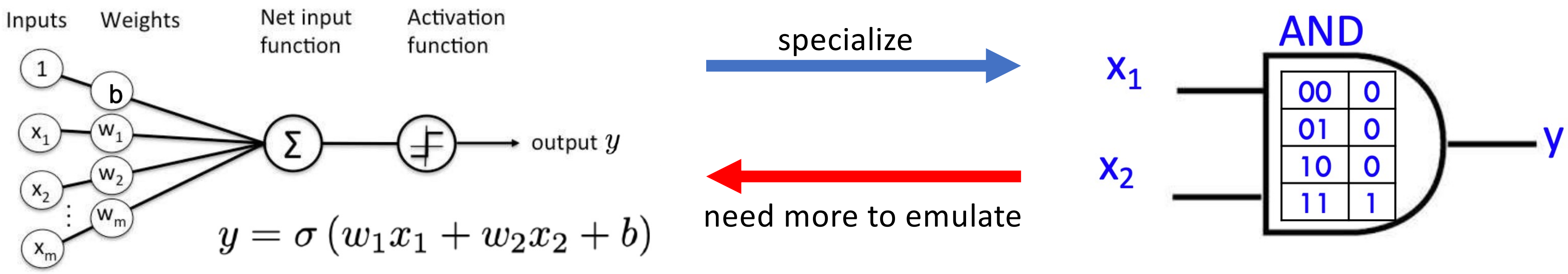
Multiple Layers

(Multi-Layer Perceptron)

Logic Gates vs Neurons



Logic Gates vs Neurons



$$y = \sigma\left(w_1 x_1 + w_2 x_2 + b\right)$$

Only models linear decision boundaries!

Can have non-linear decision boundaries!
E.g. XOR

Neural-like: Flexible learning of function

Algorithm-like: Fixed function

Slow learning and inference

Fast learning and inference

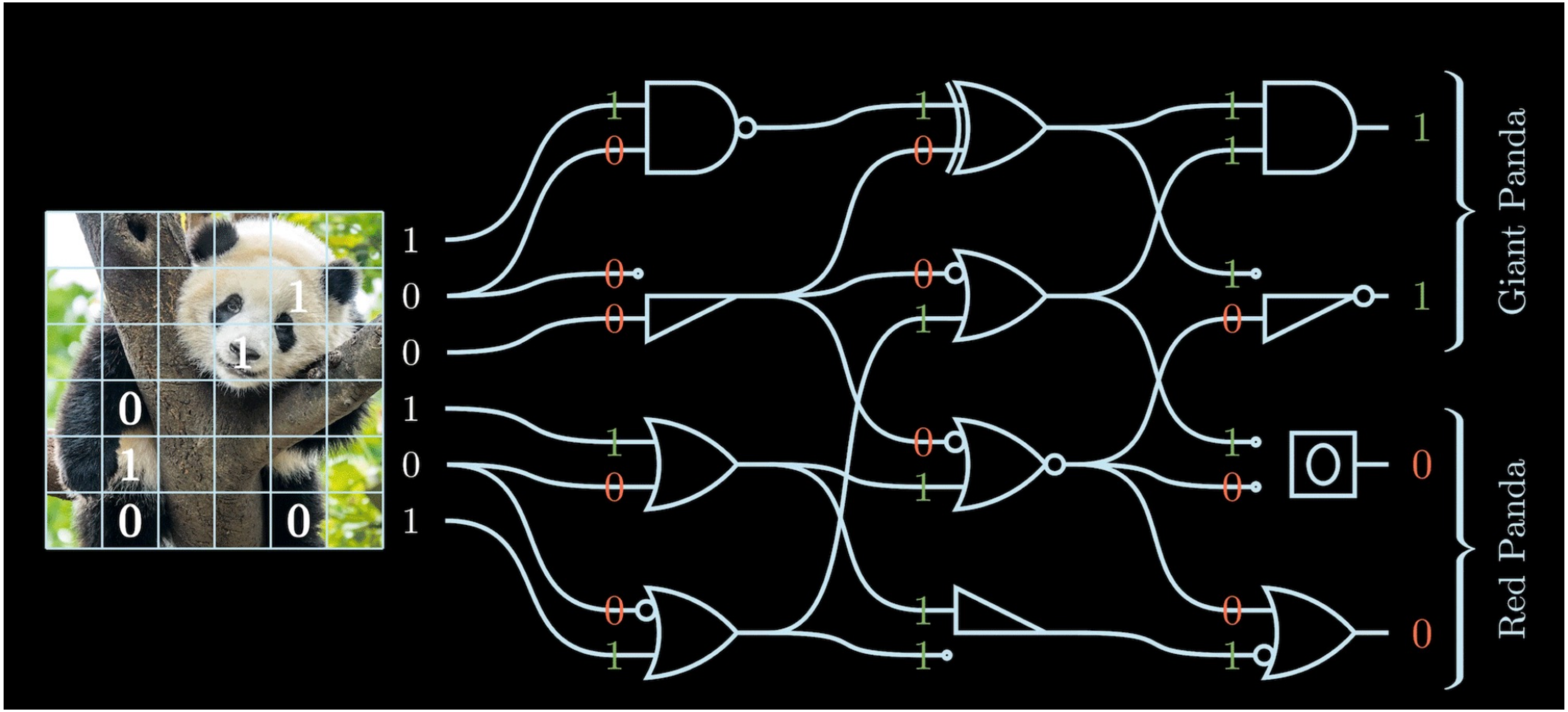
What Logic Gates to model?

- Model every possible 2-input gate

Table 1: List of all real-valued binary logic ops.

ID	Operator	real-valued	00	01	10	11
0	False	0	0	0	0	0
1	$A \wedge B$	$A \cdot B$	0	0	0	1
2	$\neg(A \Rightarrow B)$	$A - AB$	0	0	1	0
3	A	A	0	0	1	1
4	$\neg(A \Leftarrow B)$	$B - AB$	0	1	0	0
5	B	B	0	1	0	1
6	$A \oplus B$	$A + B - 2AB$	0	1	1	0
7	$A \vee B$	$A + B - AB$	0	1	1	1
8	$\neg(A \vee B)$	$1 - (A + B - AB)$	1	0	0	0
9	$\neg(A \oplus B)$	$1 - (A + B - 2AB)$	1	0	0	1
10	$\neg B$	$1 - B$	1	0	1	0
11	$A \Leftarrow B$	$1 - B + AB$	1	0	1	1
12	$\neg A$	$1 - A$	1	1	0	0
13	$A \Rightarrow B$	$1 - A + AB$	1	1	0	1
14	$\neg(A \wedge B)$	$1 - AB$	1	1	1	0
15	True	1	1	1	1	1

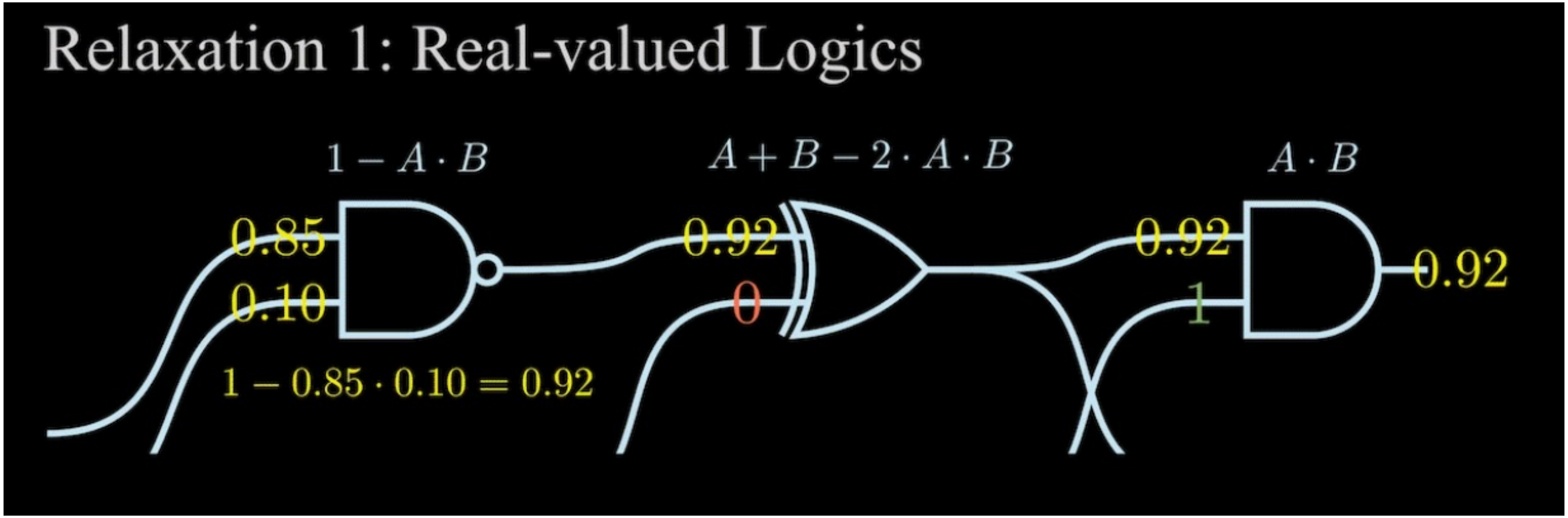
Classification



Difficulty of training logic gates

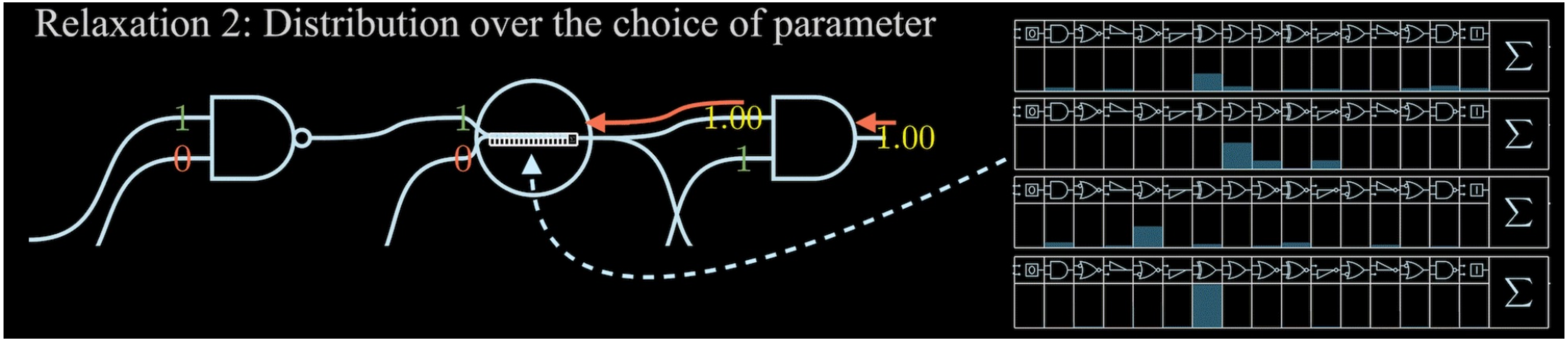
- Logic gates perform one fixed function, inherently not differentiable
- Could use evolutionary means to learn the logic gates -> not scalable
- Make network differentiable:
  - Link inputs to outputs using a differentiable function
  - Use some probabilistic way to select logic gates

Make output a function of the inputs



$$a' = f_i(a_1, a_2)$$

Make the gate selectable by backprop



$$a' = \sum_{i=0}^{15} \boxed{\boldsymbol{p}_i} \cdot f_i(a_1, a_2) = \sum_{i=0}^{15} \boxed{\frac{e^{\boldsymbol{w}_i}}{\sum_j e^{\boldsymbol{w}_j}}} \cdot f_i(a_1, a_2)$$

Categorical Distribution via Softmax

Training Setup

- **Initialization:** Randomly initialize connections and parameterization of each neuron (normal distribution), Adam optimizer, 0.01 learning rate

- **Logic Gate selection:** Each neuron will be converted to the logic gate with the highest probability based on the categorical probability distribution

Configuring to Output (Classification)

- k classes, n output neurons. Group neuron outputs in groups of size n/k, then aggregate the classification scores. Train by passing through all outputs with a softmax, and class with largest sum of post-softmax values is winner

Aggregation of Output Neurons Now, we may have n output neurons $a_1, a_2, ..., a_n \in [0, 1]$, but we may want the logic gate network to only predict $k < n$ values of a larger range than $[0, 1]$. Further, we may want to be able to produce graded outputs. Thus, we can aggregate the outputs as

$$\hat{y}_i = \sum_{j=i \cdot n/k+1}^{(i+1) \cdot n/k} a_j \, / \, \tau + \beta \tag{3}$$

where τ is a normalization temperature and β is an optional offset.

Results (Tabular - Classification)

Table 3: Results for the Adult and Breast Cancer data sets averaged over 10 runs.

Adult	Acc.	# Param.	Infer. Time	Space
Decision Tree Learner	79.5%	≈ 50	86ns	≈ 130B
Logistic Regression	84.8%	234	63ns	936B
Neural Network	84.9%	3810	635ns	15KB
Diff Logic Net (*ours*)	84.8%	1280	5.1ns	640B
Breast Cancer	Acc.	# Param.	Infer. Time	Space
Decision Tree Learner	71.9%	≈ 100	82ns	≈ 230B
Logistic Regression	72.9%	104	34ns	416B
Neural Network	75.3%	434	130ns	1.4KB
Diff Logic Net (*ours*)	76.1%	640	2.8ns	320B

Fast inference, good accuracy for classification tasks in tabular datasets

Results (Binarized MNIST)

Table 4: Results for MNIST, all of our results are averaged over 10 runs. Times (T.) are inference times per image, the GPU is an NVIDIA A6000, and the CPU is a single thread at 2.5 GHz. For our experiments, i.e., the top block, we use binarized MNIST.

MNIST	Acc.	# Param.	Space	T. [CPU]	T. [GPU]	**OPs**	FLOPs
Linear Regression	91.6%	4 010	16KB	3μs	2.4ns	(4M)	4K
Neural Network (*small*)	97.92%	118 282	462KB	14μs	12.4ns	(236M)	236K
Neural Network	98.40%	22 609 930	86MB	2.2ms	819ns	(45G)	45M
Diff Logic Net (*small*)	97.69%	48 000	23KB	625ns	6.3ns	48K	—
Diff Logic Net	98.47%	384 000	188KB	7μs	(50ns)	384K	—

Fast inference, good accuracy for larger image classification

Results (CIFAR)

Table 5: Results on CIFAR-10. Times (T.) are inference times per image, the GPU is an NVIDIA A6000, and the CPU is a single thread at 2.5 GHz. For our experiments, i.e., the top block, we use a color-channel resolution of 4 for the first 3 lines and a color-channel resolution of 32 for the *large* models. The other baselines were provided with the full resolution of 256 color-channel values. The numbers in parentheses are extrapolated / estimated.

CIFAR-10	Acc.	# Param	Space	T. [CPU]	T. [GPU]	**OPs**	FLOPs
Neural Network (color-ch. res. = 4)	50.79%	12.6M	48MB	1.2ms	370ns	(25G)	25M
Diff Logic Net (*small*)	51.27%	48K	24KB	1.3μs	19ns	48K	—
Diff Logic Net (*medium*)	57.39%	512K	250KB	7.3μs	29ns	512K	—
Diff Logic Net (*large*)	60.78%	1.28M	625KB	(18μs)	(73ns)	1.28M	—
Diff Logic Net (*large*$\times$2)	61.41%	2.56M	1.22MB	(37μs)	(145ns)	2.56M	—
Diff Logic Net (*large*$\times$4)	62.14%	5.12M	2.44MB	(73μs)	(290ns)	5.12M	—
Best Fully-Connected Baselines (color-ch. res. = 256)							
Regularized SReLU NN [28]	68.70%	20.3M	77MB	1.9ms	565ns	(40G)	40M
Student-Teacher NN [52]	65.8%	1M	4MB	112μs	243ns	(2G)	2M
Student-Teacher NN [52]	74.3%	31.6M	121MB	2.9ms	960ns	(63G)	63M

Fast inference, moderate accuracy for larger image classification

Exponential Growth of Gates!

Table 6: Logic gate network architectures.

Dataset	Model	Layers	Neurons / layer		Total num. of p.	τ
MONK-1	—	6	24		144	1
MONK-2	—	6	12		72	1
MONK-3	—	6	12		72	1
Adult	—	5	256	32	1 280	1/0.075
Breast Cancer	—	5	128	8	640	1/0.1
MNIST	small	6	8 000	128	48 000	1/0.1
	normal	6	64 000	2048	384 000	1/0.03
CIFAR-10	small	4	12 000	1024	48 000	1/0.03
	medium	4	128 000		512 000	1/0.01
	large	5	256 000		1 280 000	1/0.01
	large×2	5	512 000		2 560 000	1/0.01
	large×4	5	1 024 000		5 120 000	1/0.01

Exponential Increase

Requires more neurons than MLP (in orange)

Limitations

- Expensive Training – Higher training cost compared to conventional neural networks. Mainly due to the differentiable categorical distribution used to decide the logic gate

- Convolutions and Residual connections are not modelled

- Limited to small architectures due to exponential scaling of number of logic gates in subsequent layers

Some of my thoughts

Benefits and drawbacks (my own thoughts)

- Benefits:
  - No gradient vanishing / exploding
  - Mimics biological neurons better – output is always limited to 1

- Drawbacks:
  - Requires exponential amount of logic gates to model the input-output relation
  - Requires a lot of logic gate nodes as compared to the normal neurons

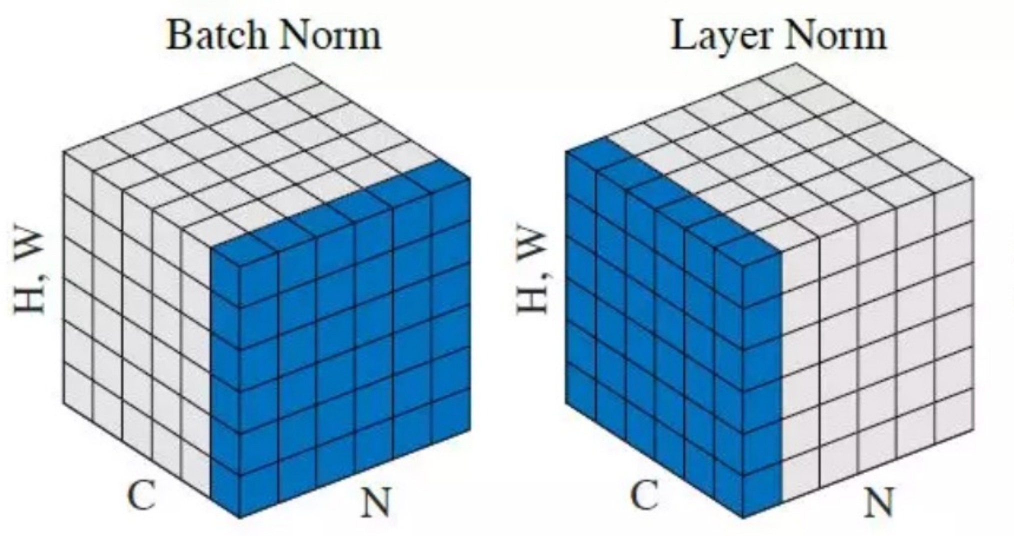
- Maybe take the paper's idea of constraining outputs, but use a more expressive architecture?

Artificial Tools used in Neural Networks to counter vanishing/exploding gradients

- Scaling of inputs to zero mean and unit variance (e.g. ImageNet)

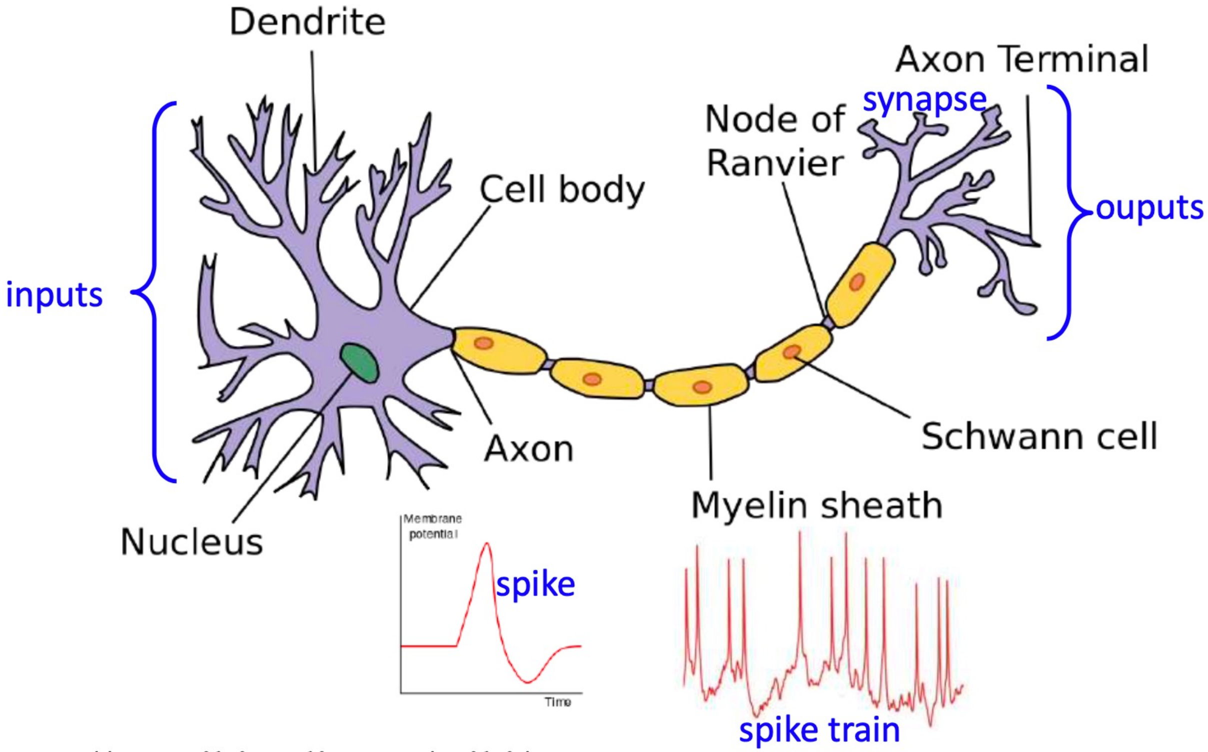
- Normalization
  - Batch Norm
  - Layer Norm

- Gated RNNs
  - Forget Gate
  - Input Gate
  - Output Gate

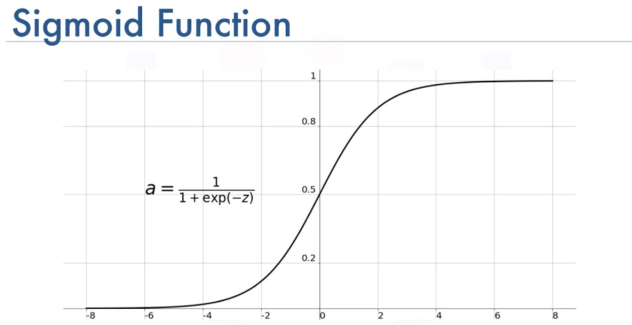


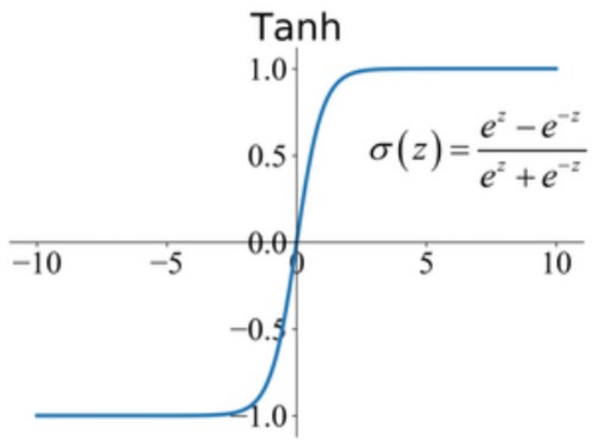
https://paperswithcode.com/method/layer-normalization

Action Potential is a fixed magnitude
Should we constrain outputs?



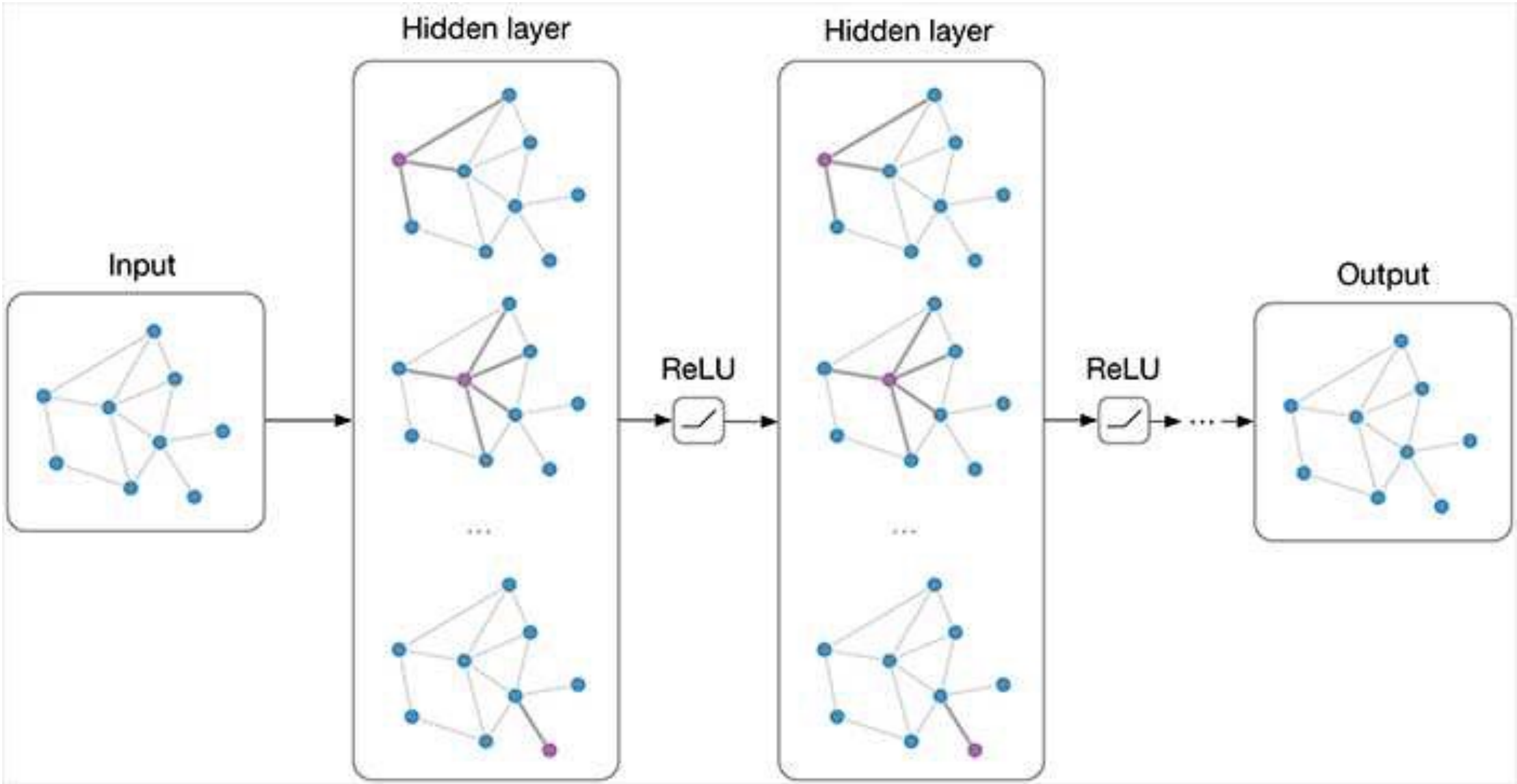
https://en.wikipedia.org/wiki/Neuron





Saturated gradients:
How to solve?

Neurons can communicate in clusters: Directed Graph Neural Networks?



Questions to Ponder

- Why restrict the input of each logic gate to just 2 inputs? Why not up to *n* inputs?
- Should the logic gate connections be pseudo-randomly initialized, or should there be a fixed set of initial connections?
- Why not keep the logic gates fixed throughout rather than having it being selected via categorical distribution?
- Can we do with fewer operators rather than all 16 gates?
- **Can we do a similar implementation on a neural network, but restrict output to magnitude 1 for all nodes?**