Neural Networks 2

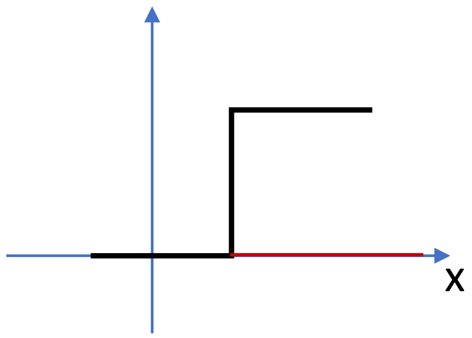
ATML Track1

Rik Sarkar

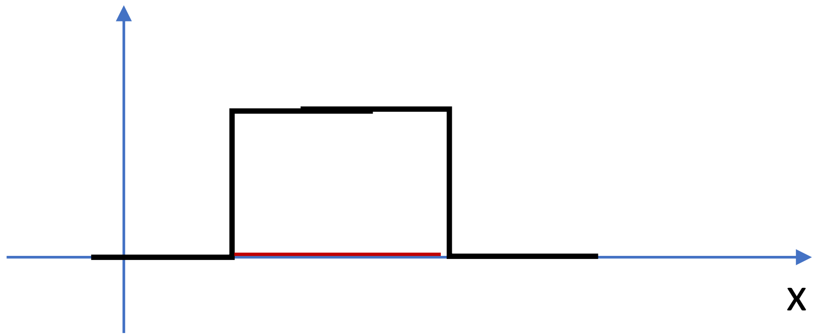
Today

- How neural networks divide the domain for classification
- Universal approximation
- The importance of depth in deep neural networks
- Internal representations in Neural networks – high dimensional embeddings

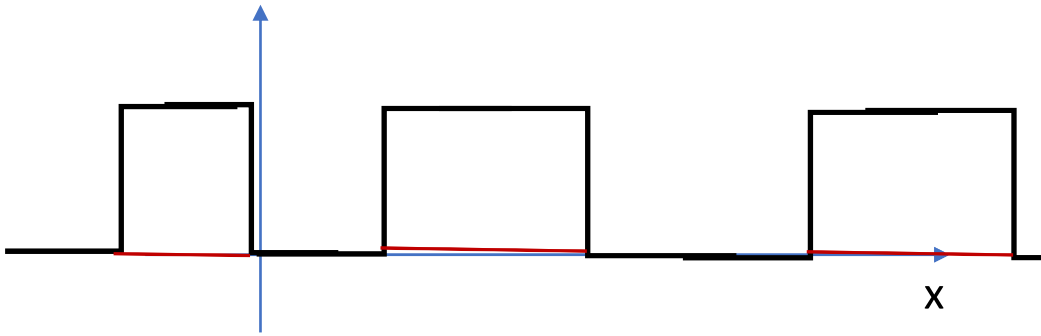
Step activation



- With suitable weights we can shift the step anywhere on X
- ANDing 2 neurons we can get a segment

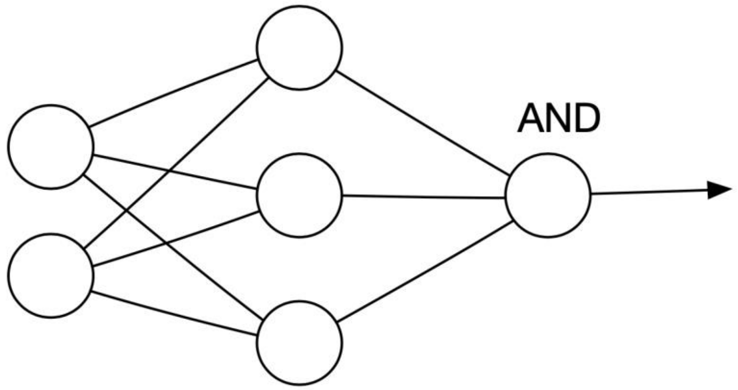


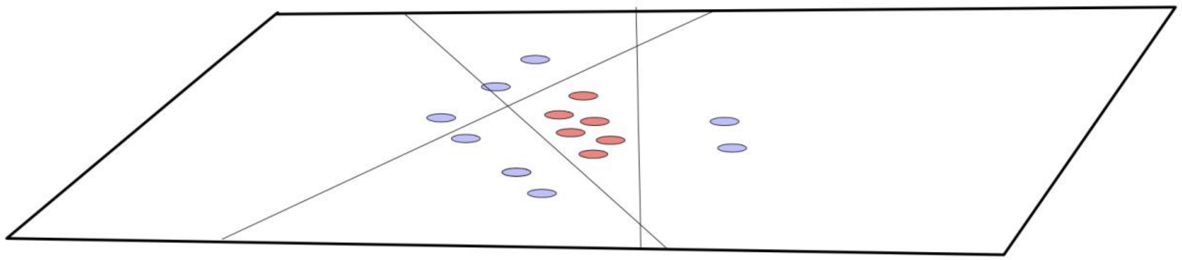
- Using several such configurations, we can get any binary function on $\mathbb{R}$
  - More neurons – more activation boundaries
    - Capable of representing more complex functions



In 2D: Creating shapes

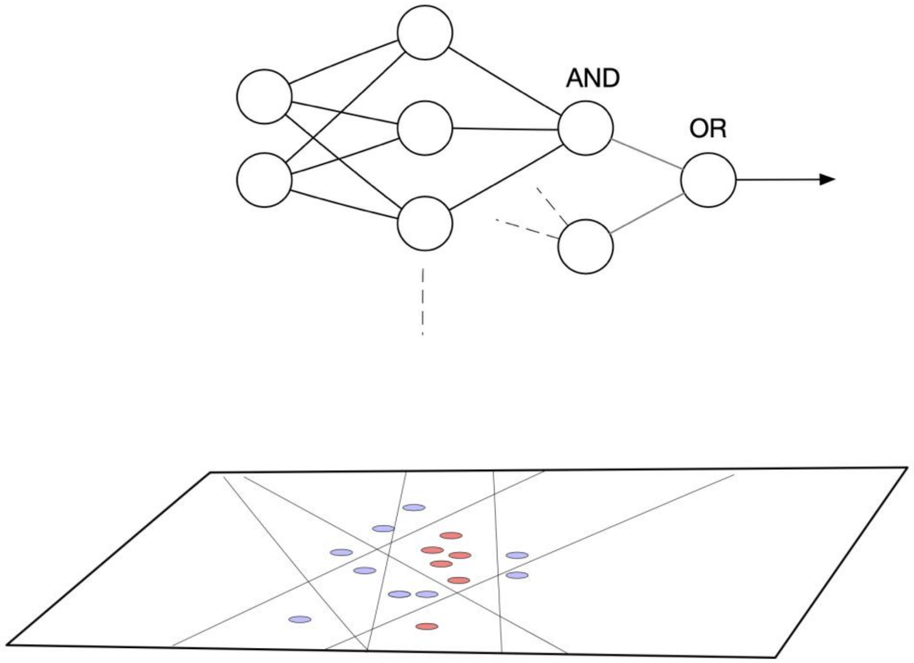
- Neurons with step activation
- Each hidden layer neuron identifies a half-plane
  - An activation boundary
- The output neuron performs AND
- We can create arbitrary polygons this way



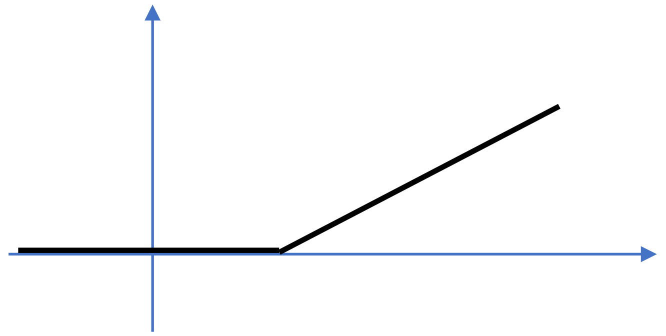


More complex shapes

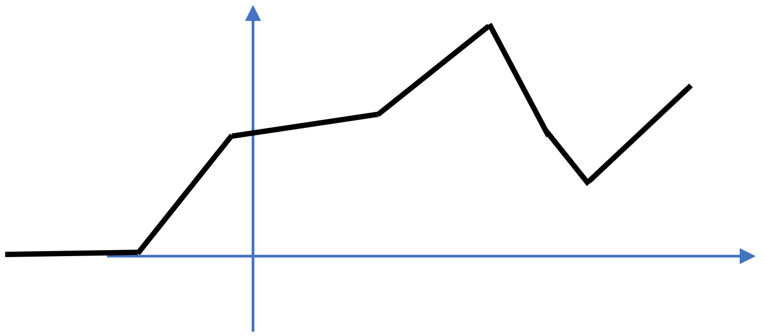
- Compute binary relations with respect to suitable half planes
- Use a Boolean formula to create the right area
- An optimization algorithm finds the right weights
  - to make the half planes
  - And make the Boolean expression
- More neurons adds more half plane boundaries
  - Splits the plane into smaller cells
  - Capable of finer divisions
  - Representing more complex functions



ReLU



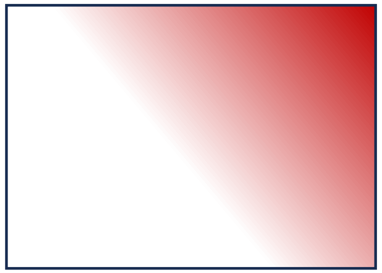
- Position and slope controlled by weights

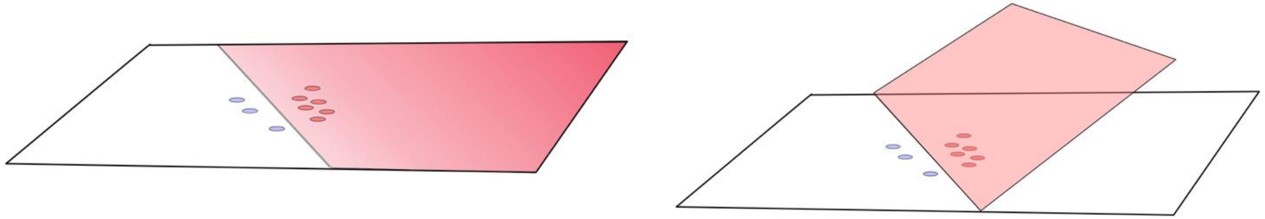


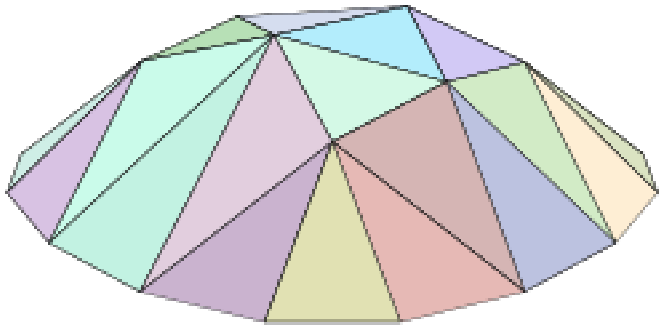
- Multiple neurons produce piecewise linear functions

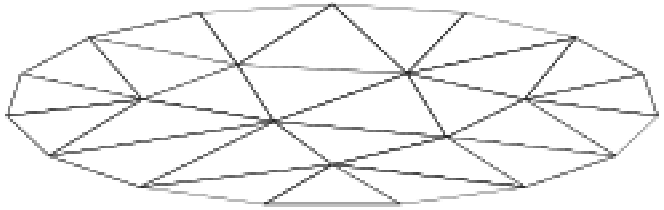
ReLU in 2D

- Multiple views of a ReLU function in 2D
- It divides with an activation boundary
- Plus adds a number on one side
  - How far it is from the boundary
  - Like a coordinate
- Can be used to create piecewise linear function shapes over 2D
- Try out ReLU activations on
  - https://playground.tensorflow.org/





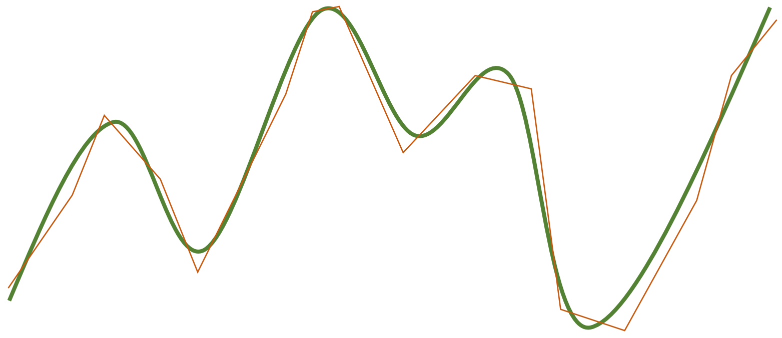




Universal approximation theorem

- Given: The task of
  - Approximating a continuous real valued function f in a bounded region $\mathcal{X} \subset \mathbb{R}^d$
  - Within an error bound ϵ
- Then, for any non-polynomial activation (E.g. ReLU, sigmoid) there is a neural network F,
  - with a single hidden layer
  - of width some suitable N
- Such that with suitable weights: $\forall x \in \mathcal{X}$: $|F(x) - f(x)| \leq \epsilon$

- With a wide enough hidden layer, we can approximate any function



Observation

- Essentially saying: For any function, for however good approximation we require:
  - There is a big enough neural network to achieve that.

- No one network does everything. Just that for any task, there is a suitable network.

Many variants of universal approximation

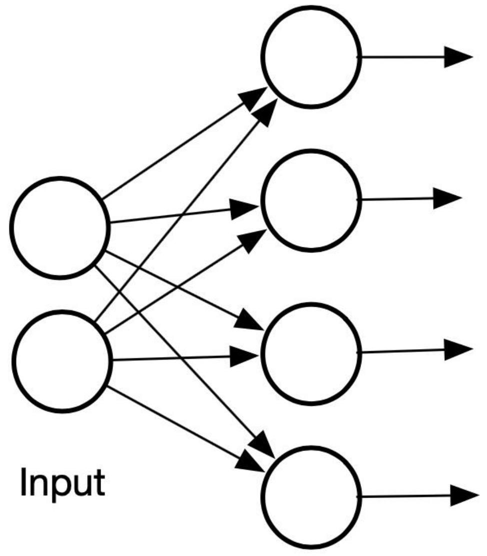
- Approximation result for deep networks
- Variations due to activation functions
- Various assumptions

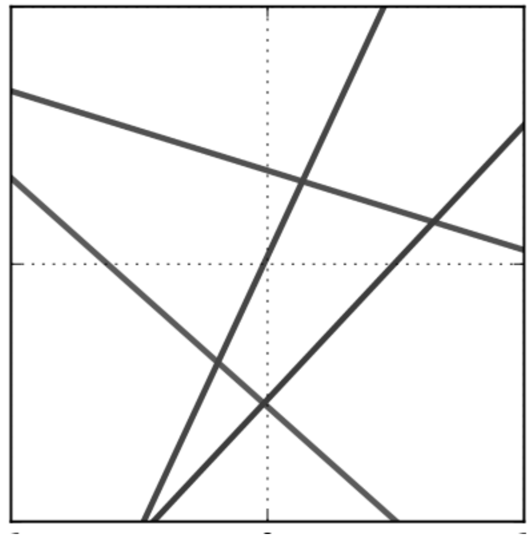
Why do we need deep neural networks?

- What does depth get us?

What happens as we add ReLU layers?

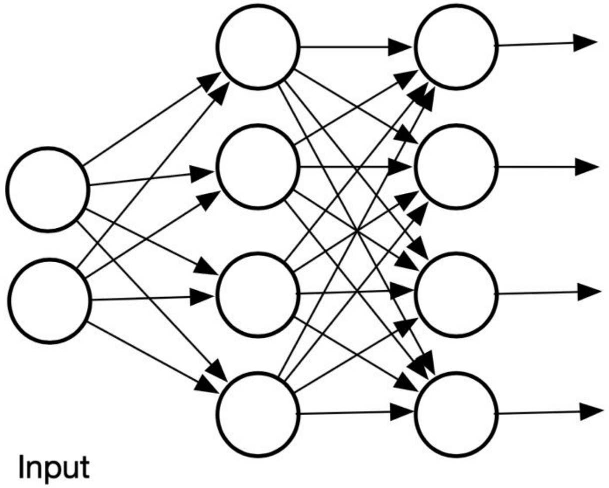
- Say, if we have one layer of 4 ReLUs
- Generates 4 lines (activation boundaries) that divide the space into ~ 9 cells
  - We can use some AND/OR logic to decide if point is in a particular cell
- Remember, ReLU also outputs how far from the boundary a point is
  - Can be used as a coordinate within each cell
  - Each cell has a coordinate system

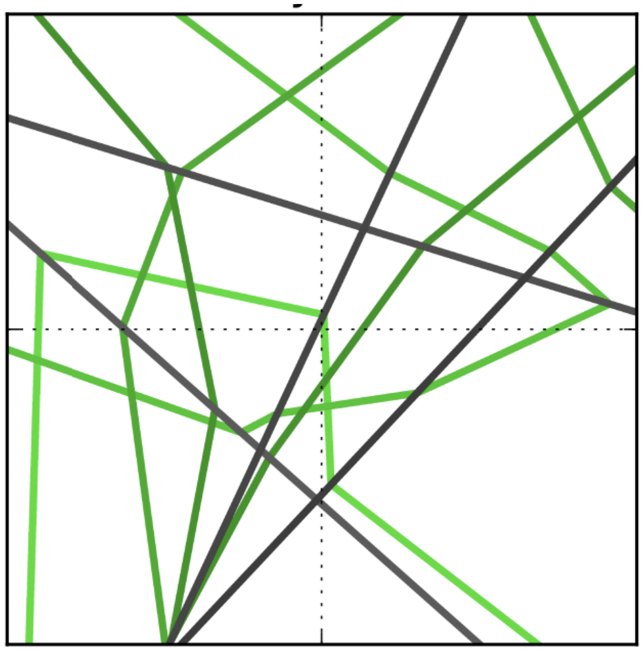




What happens with multiple ReLU layers?

- Add one more layer
- Each cell is now split into ~9 smaller cells
- The activation boundaries are different for each cell
- The boundaries from layer 2 bend at the boundary of layer 1 due to different coordinate systems
- Within each cell, each output value changes linearly



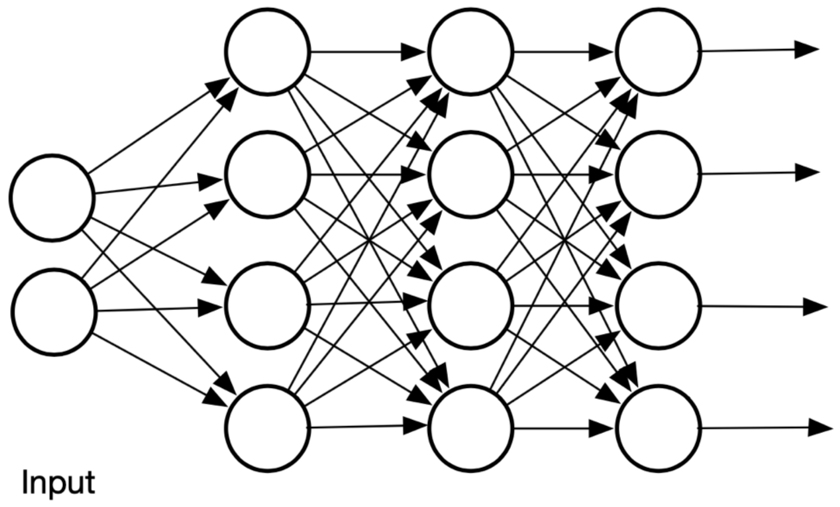


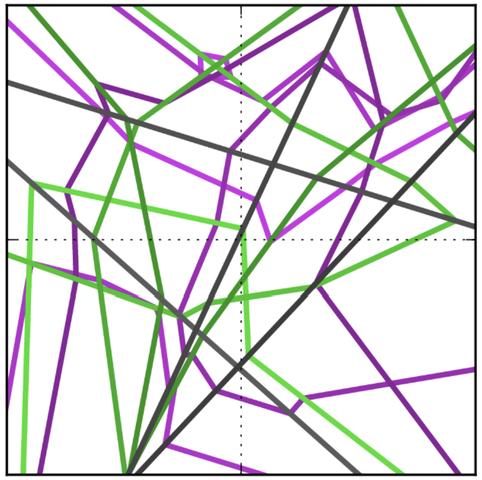
What happens with multiple ReLU layers?

- Add a third layer

- Each smaller cell now splits into even smaller cells.
  - Lines bend at both previous boundaries

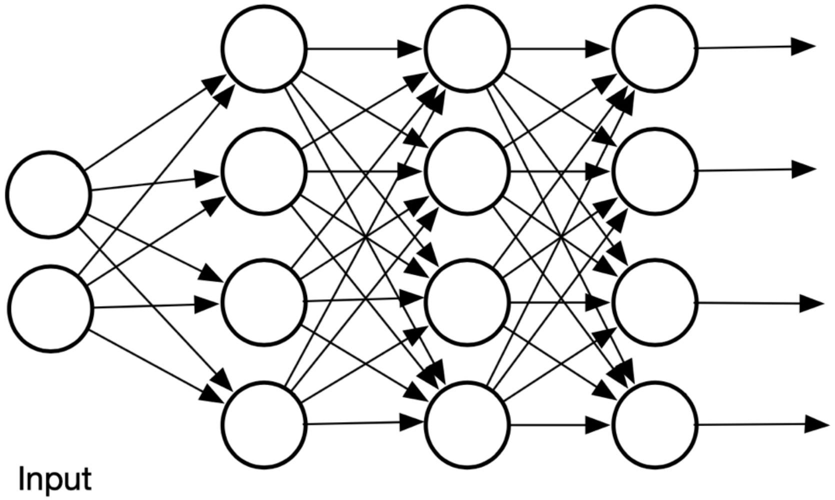
- Within each cell, each output value changes linearly
- At each layer, each cell gets split into multiple pieces
  - How does the number of cells grow?

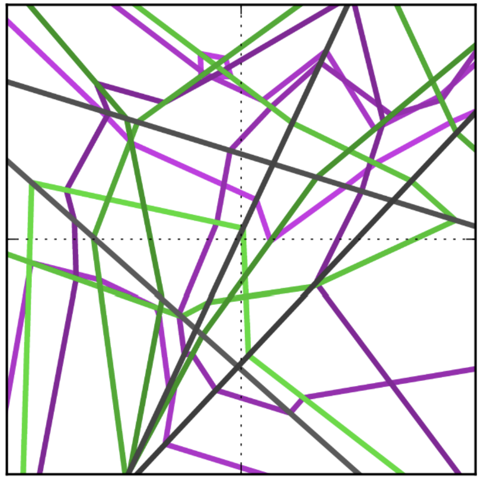




What happens with multiple ReLU layers?

- At each layer, each cell is split into multiple pieces
- Number of cells grow exponentially
- If we take a line in $\mathcal{X}$
  - The number of activation boundaries it crosses increases exponentially with depth of the network
- The complexity of the function computed by a neural network increases exponentially with depth
  - [Raghu, Kleinberg et al. 2017]





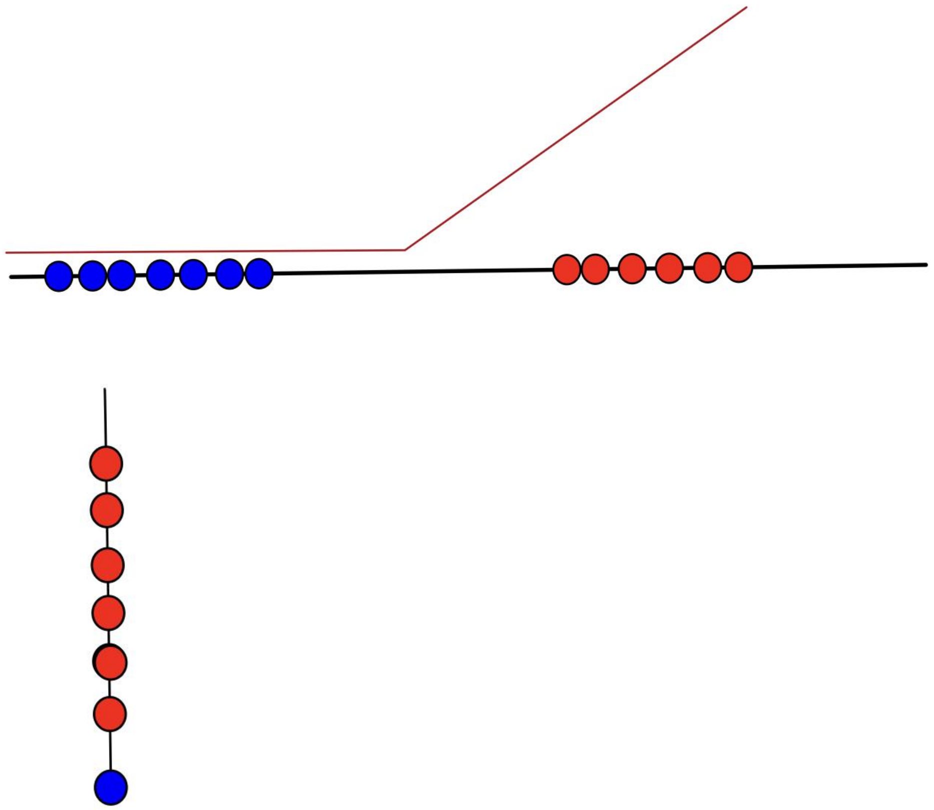
- Observe that in the increasing width case, a new neuron along the width splits *some* of the existing cells.
- But not *all* the existing cells

Internal representations: High Dimensional Embeddings

- We can think of the output of each layer as an embedding

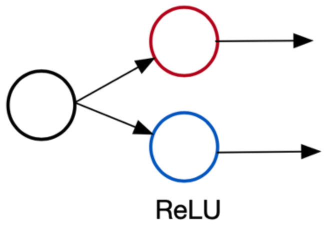
- Example: 1-D, 1 ReLU

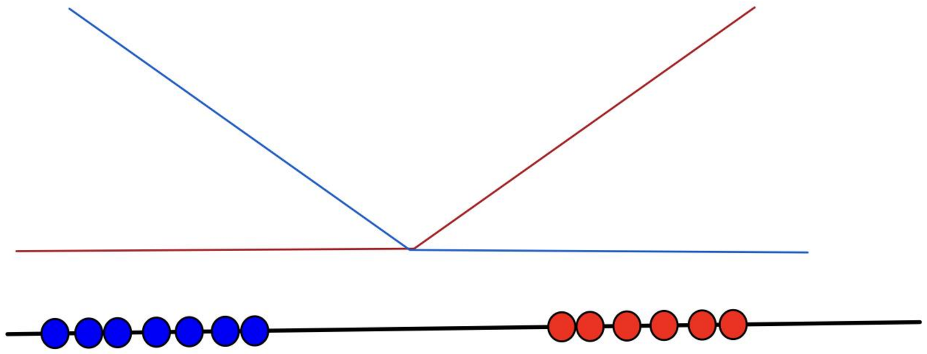
  - ReLU output:



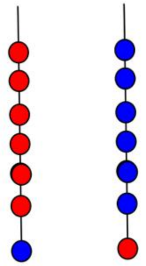
Internal representations: High D Embeddings

- Example: 1-D, 2 ReLU

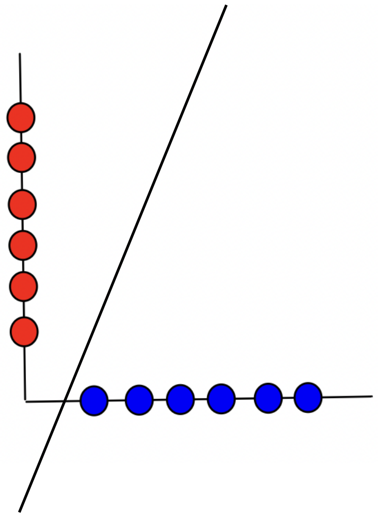




- ReLU output:

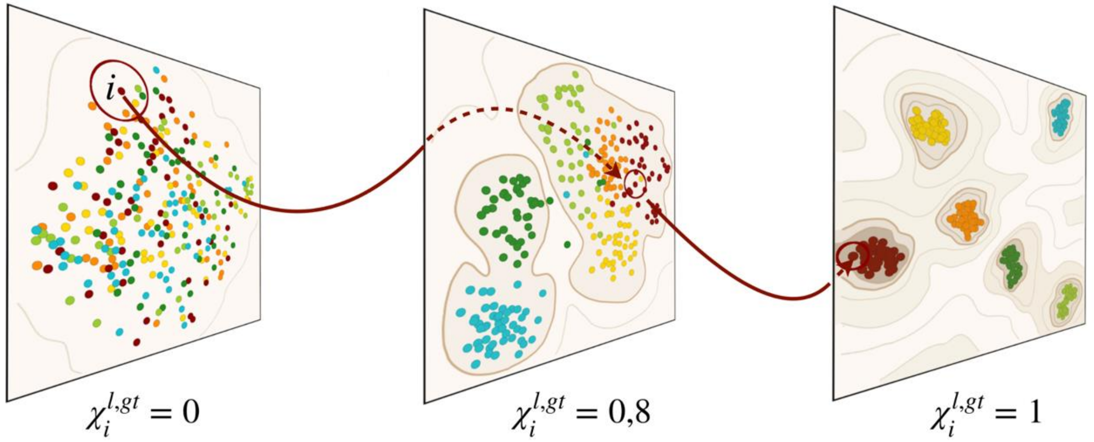


- More conveniently:
  - Along two axes
  - With easy classification



- In general, we can think of the output of each layer as a high dimensional representation of the input space
- The network takes inputs in $\mathcal{X}$,
  - Output of layer k maps to $\mathbb{R}^u$
  - Upto layer k, the NN is a map $\mathcal{X} \to \mathbb{R}^u$
- If the next layer has a width v then it is a map $\mathbb{R}^u \to \mathbb{R}^v$
  - Upto layer $k+1$, the NN is a map $\mathcal{X} \to \mathbb{R}^v$
  - ….

- With each layer the NN computes a different embedding in a high dimensional space
  - Alternatively, a different probability density in High dim space



- Example: ResNet152 (2D projections)
  - Input, Conv4, output
  - Initial input scattered and mixed
  - Slowly becomes more clustered
  - χ_i: fraction of 5 nearest neighbors of i that are in the same class

[Doimo 2020]: Hierarchical nucleation in deep neural networks