



IMPERIAL



Learning with Logical Constraints

An introduction

Eleonora Giunchiglia
PI of the DUCK Lab | Imperial College London





Overview

- Why are constraints important
- Integration of constraints into the loss function
- Integration of constraints into the topology
- The reasoning shortcuts problem
- Conclusions



Why is Machine Learning successful?

Because you do not need to give instructions, as everything can be learnt from data (that is at least what some believe...)

Imagine giving instructions to a computer vision model to detect pedestrians...





Machine Learning Limitations

Is data enough?

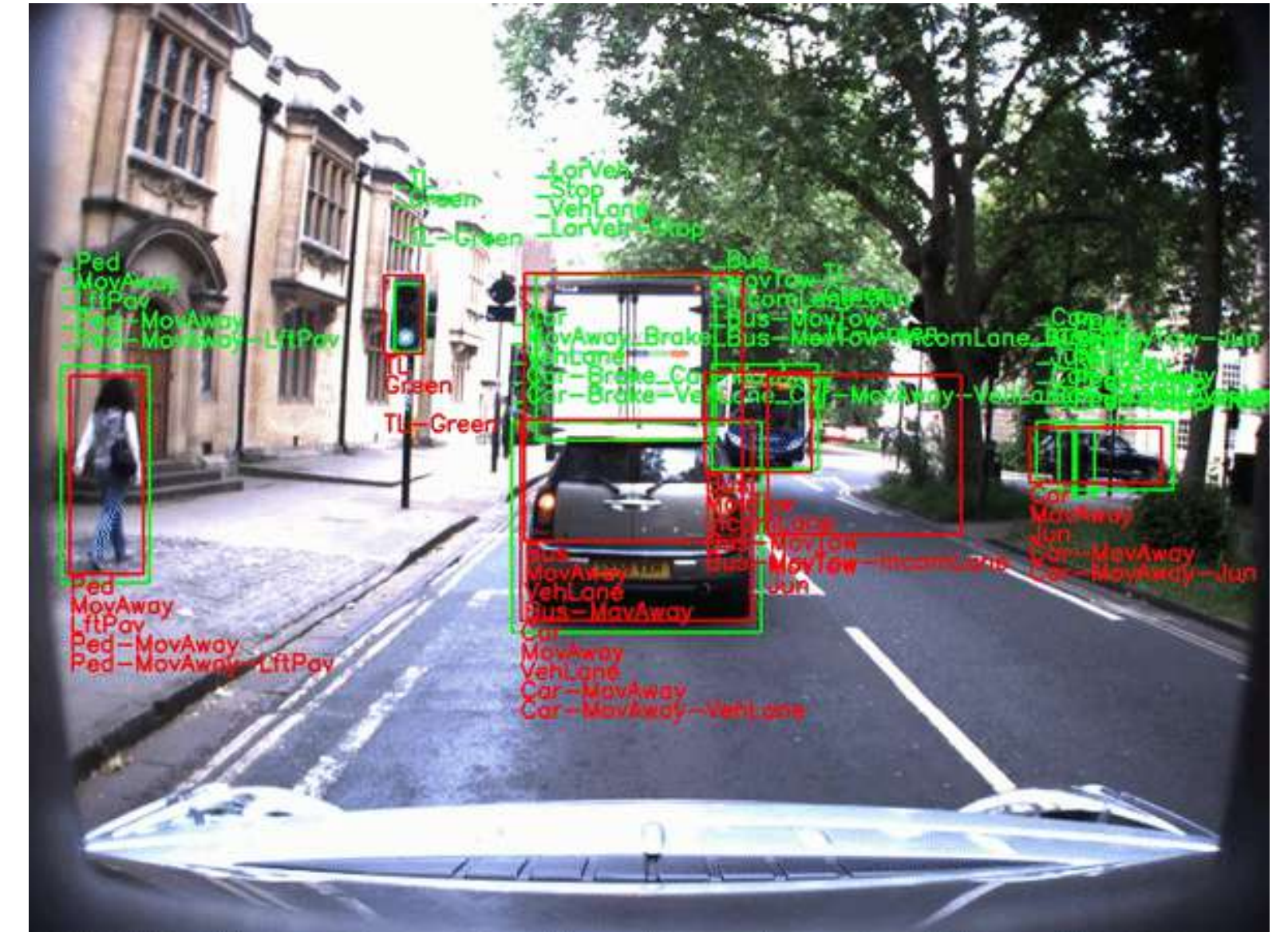
- In ML you need a huge amount of labelled data



Machine Learning Limitations

Is data enough?

- In ML you need a huge amount of labelled data

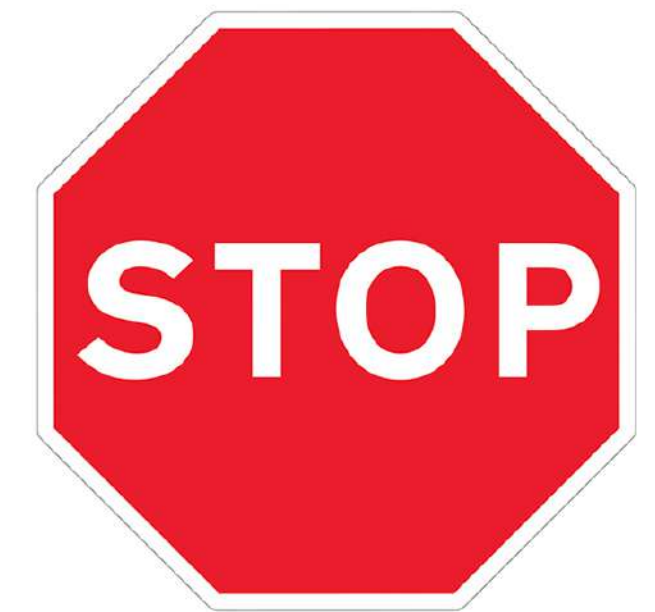




Machine Learning Limitations

Is data enough?

- In ML you need a huge amount of labelled data
- There is no way to inject any background knowledge





Machine Learning Limitations

Is data enough?

- In ML you need a huge amount of labelled data
- There is no way to inject any background knowledge
- There is no way to guarantee that the models are compliant with the background knowledge





How can we alleviate these problems?





How can we alleviate these problems?



Verification Techniques

An approach to reachability analysis for feed-forward ReLU neural networks

Alessio Lomuscio, Lalit Maganti
Imperial College London, London, UK

2017

Testing Techniques

DOI:10.1145/3361566
DeepXplore: Automated Whitebox Testing of Deep Learning Systems

By Kexin Pei, Yinzhì Cao, Junfeng Yang, and Suman Jana

2017



How can we alleviate these problems?



Neuro-Symbolic AI Techniques

Semantic Probabilistic Layers for Neuro-Symbolic Learning

Kareem Ahmed

Guy Van den Broeck

Stefano Teso

Kai-Wei Chang

Antonio Vergari

CCN⁺: A neuro-symbolic framework for deep learning with requirements

Eleonora Giunchiglia^{a,*}, Alex Tatomir^b, Mihaela Cătălina Stoian^b,
Thomas Lukasiewicz^{a,b}

^a Institute of Logic and Computation, TU Wien, Vienna, Austria

^b Department of Computer Science, University of Oxford, Oxford, UK

Verification Techniques

An approach to reachability analysis for feed-forward ReLU neural networks

Alessio Lomuscio, Lalit Maganti
Imperial College London, London, UK

2017

Testing Techniques

DeepXplore: Automated Whitebox Testing of Deep Learning Systems

By Kexin Pei, Yinzhao Cao, Junfeng Yang, and Suman Jana

DOI:10.1145/3361566

2017

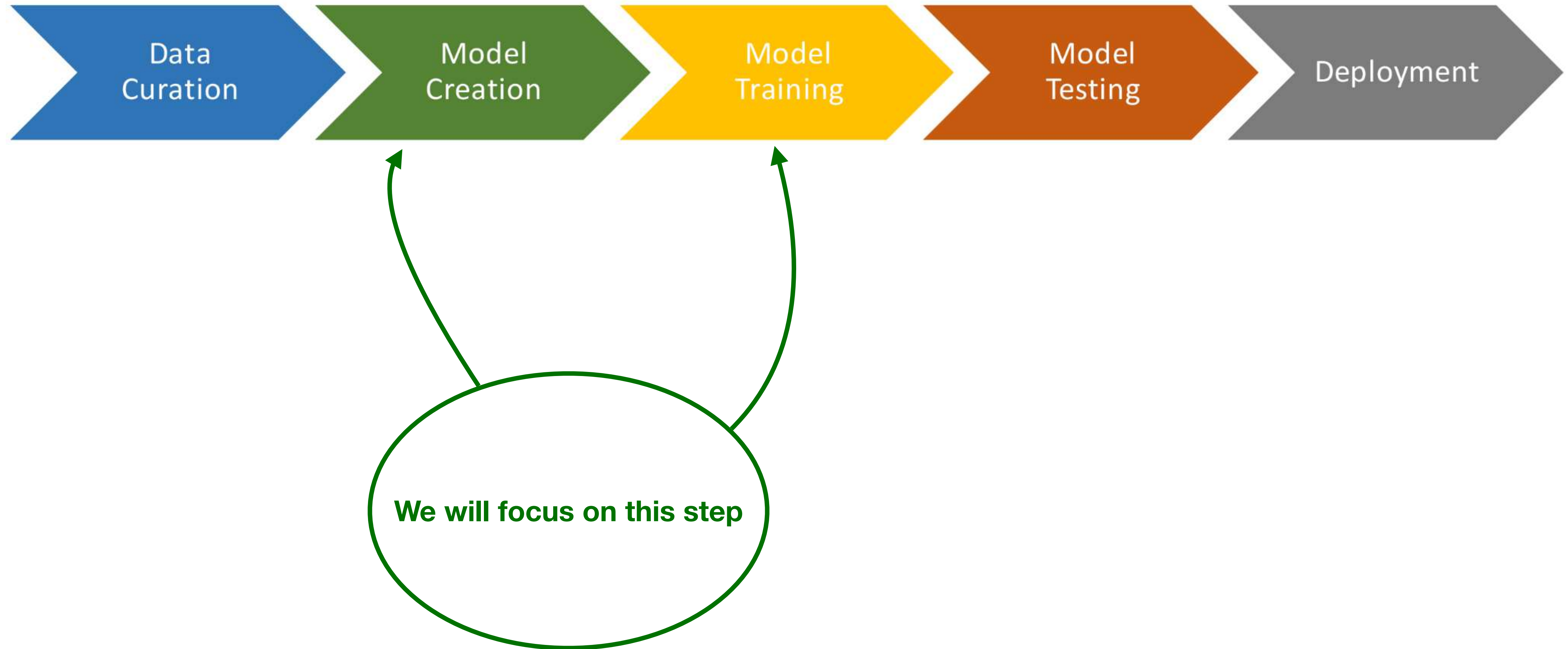


How can we alleviate these problems?





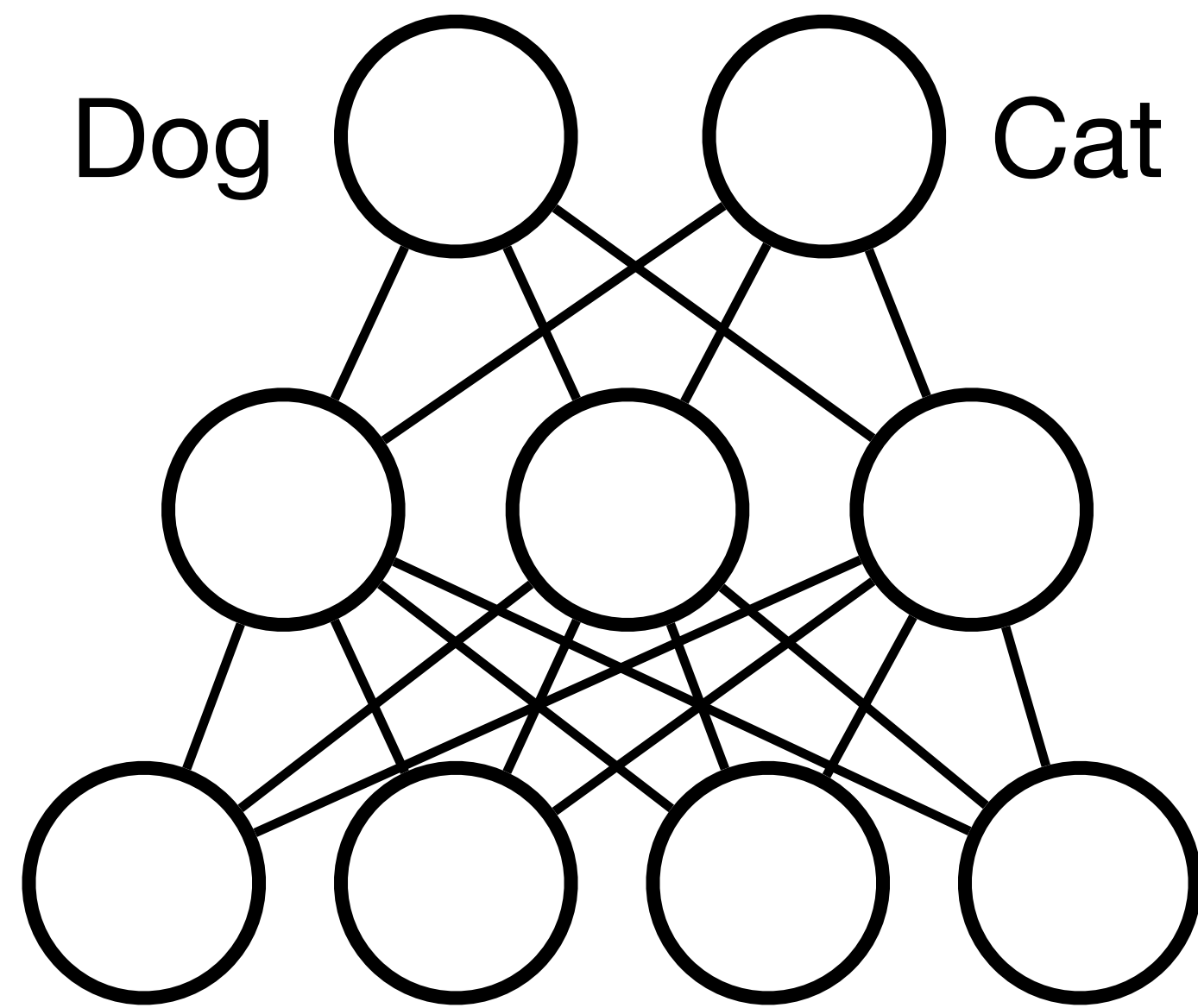
How can we alleviate these problems?





How to integrate constraints in the ML model?

Before answering... A small recap on ML



- Define the topology of the network $f, g, h \dots$
- Define a loss function using ground truth y_D and y_C
- Train the network using loss function $\mathcal{L}$
- Test it

How to integrate constraints in the ML model?

Before answering... A small recap on ML





How to integrate constraints in the ML model?

Before answering... A small recap on ML

Multi-class classification problem:

→ Mutually exclusive classes

$$(h_D + h_C = 1)$$



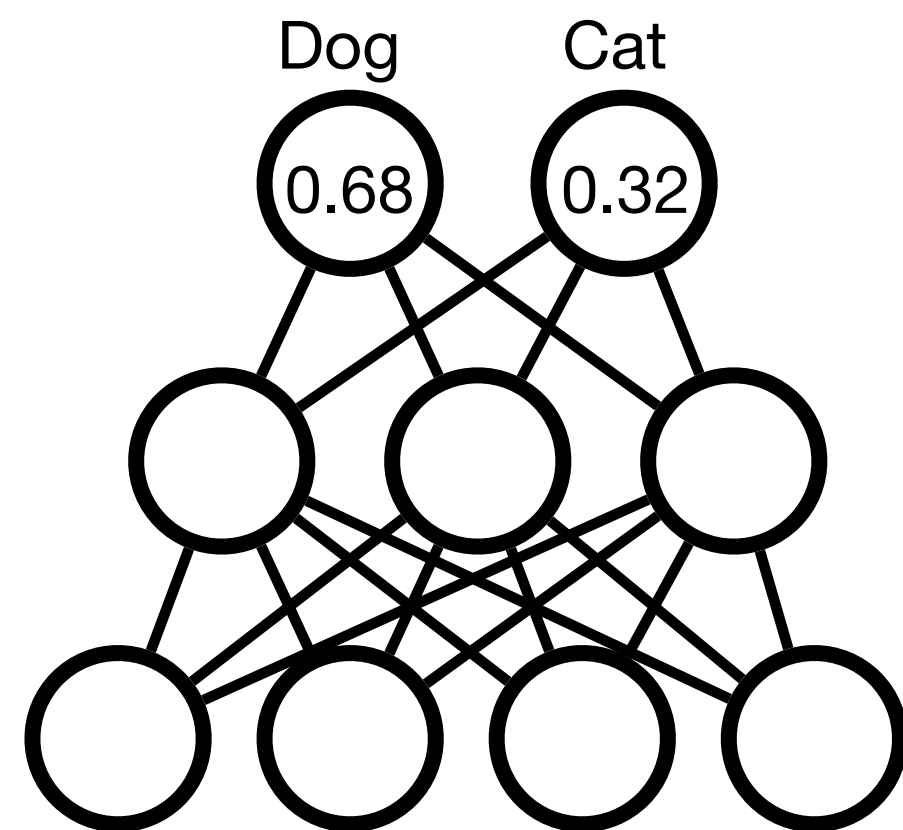
How to integrate constraints in the ML model?

Before answering... A small recap on ML

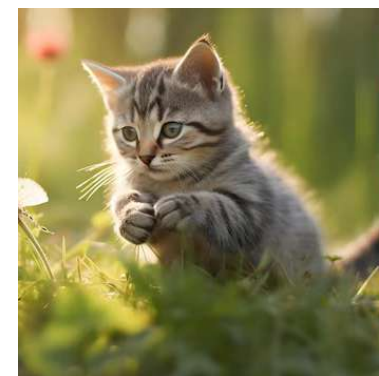
Multi-class classification problem:

→ Mutually exclusive classes

$$(h_D + h_C = 1)$$



$\mathbf{x}_1$



$\mathbf{x}_2$



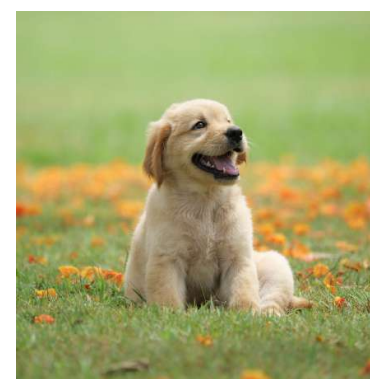
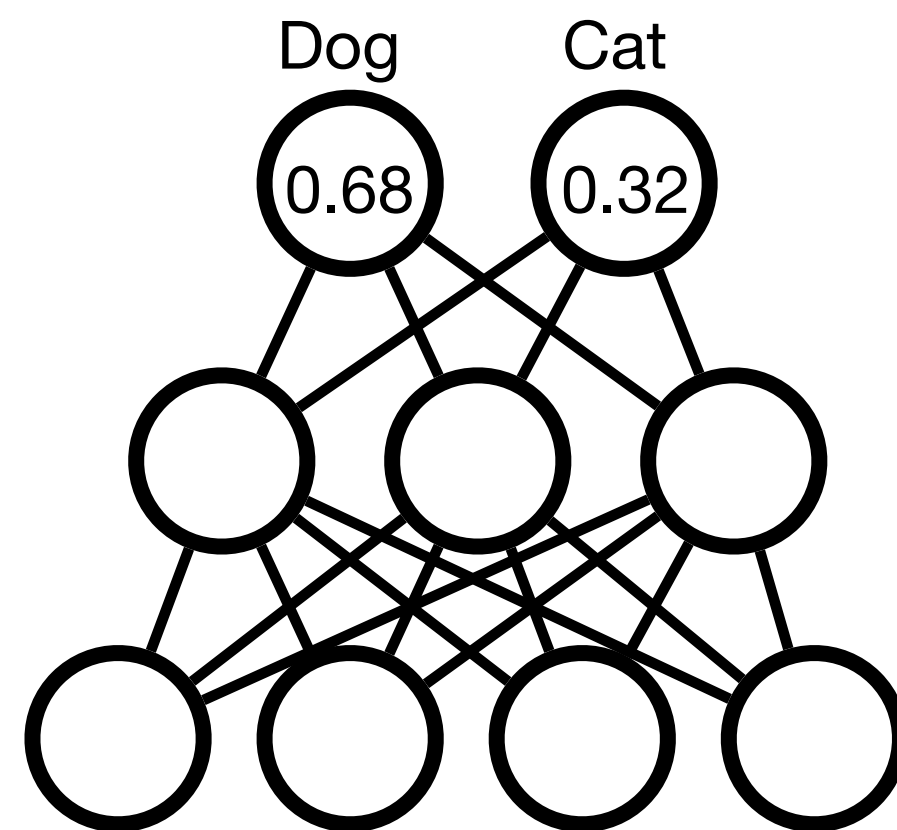
How to integrate constraints in the ML model?

Before answering... A small recap on ML

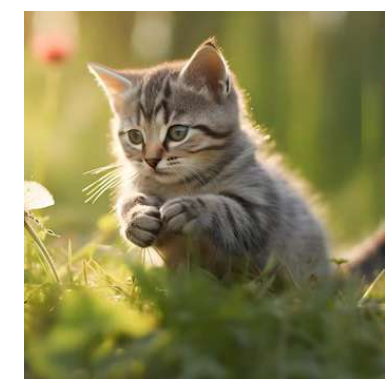
Multi-class classification problem:

→ Mutually exclusive classes

$$(h_D + h_C = 1)$$



$\mathbf{x}_1$



$\mathbf{x}_2$

Multi-label classification problem:

→ independent classes



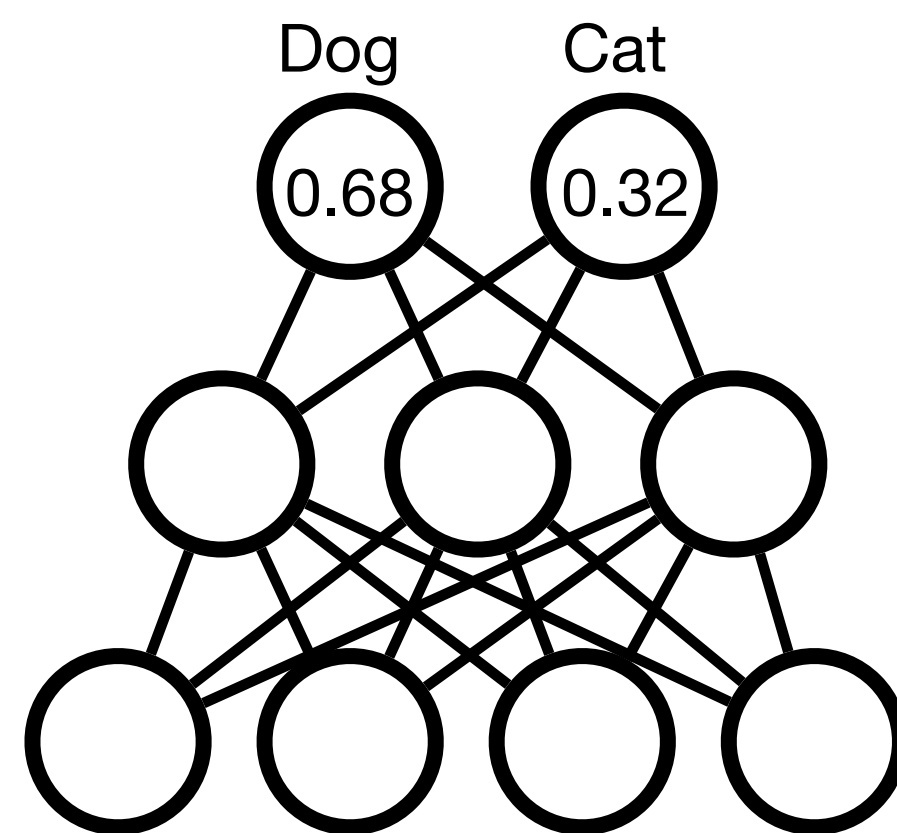
How to integrate constraints in the ML model?

Before answering... A small recap on ML

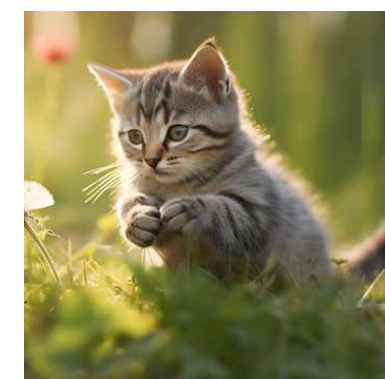
Multi-class classification problem:

→ Mutually exclusive classes

$$(h_D + h_C = 1)$$



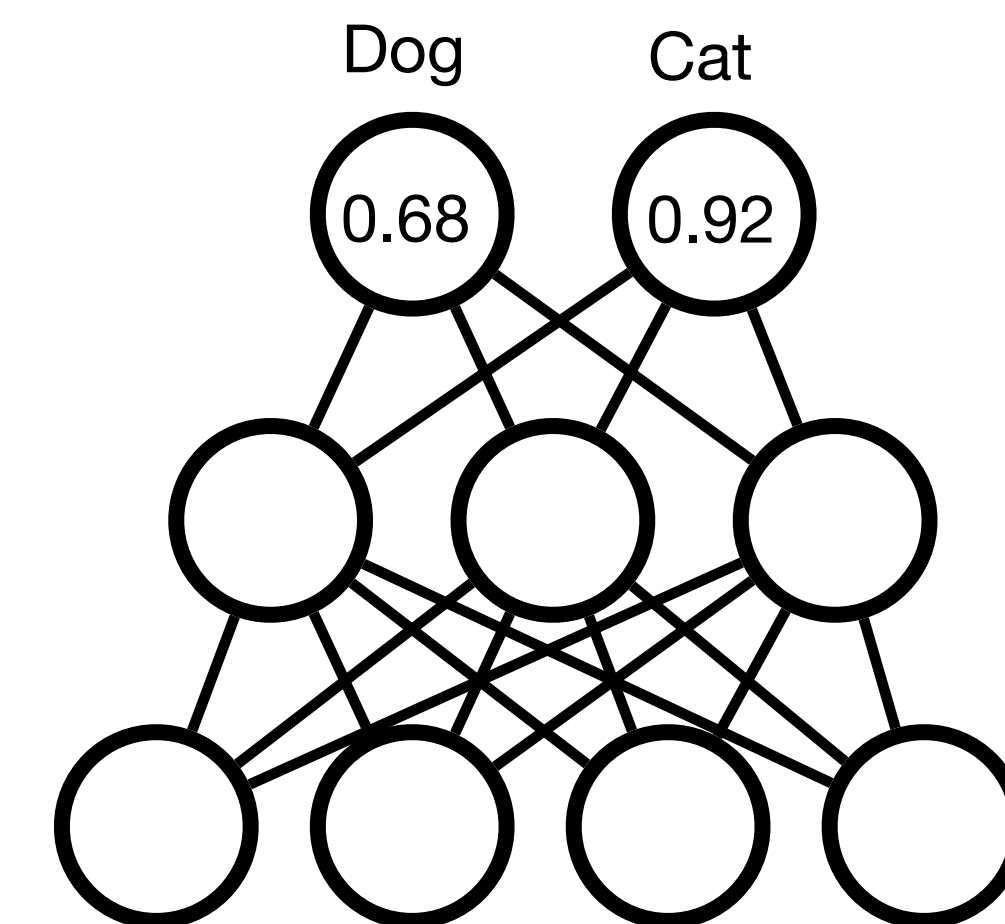
x_1



x_2

Multi-label classification problem:

→ independent classes



x_1



x_2



x_3

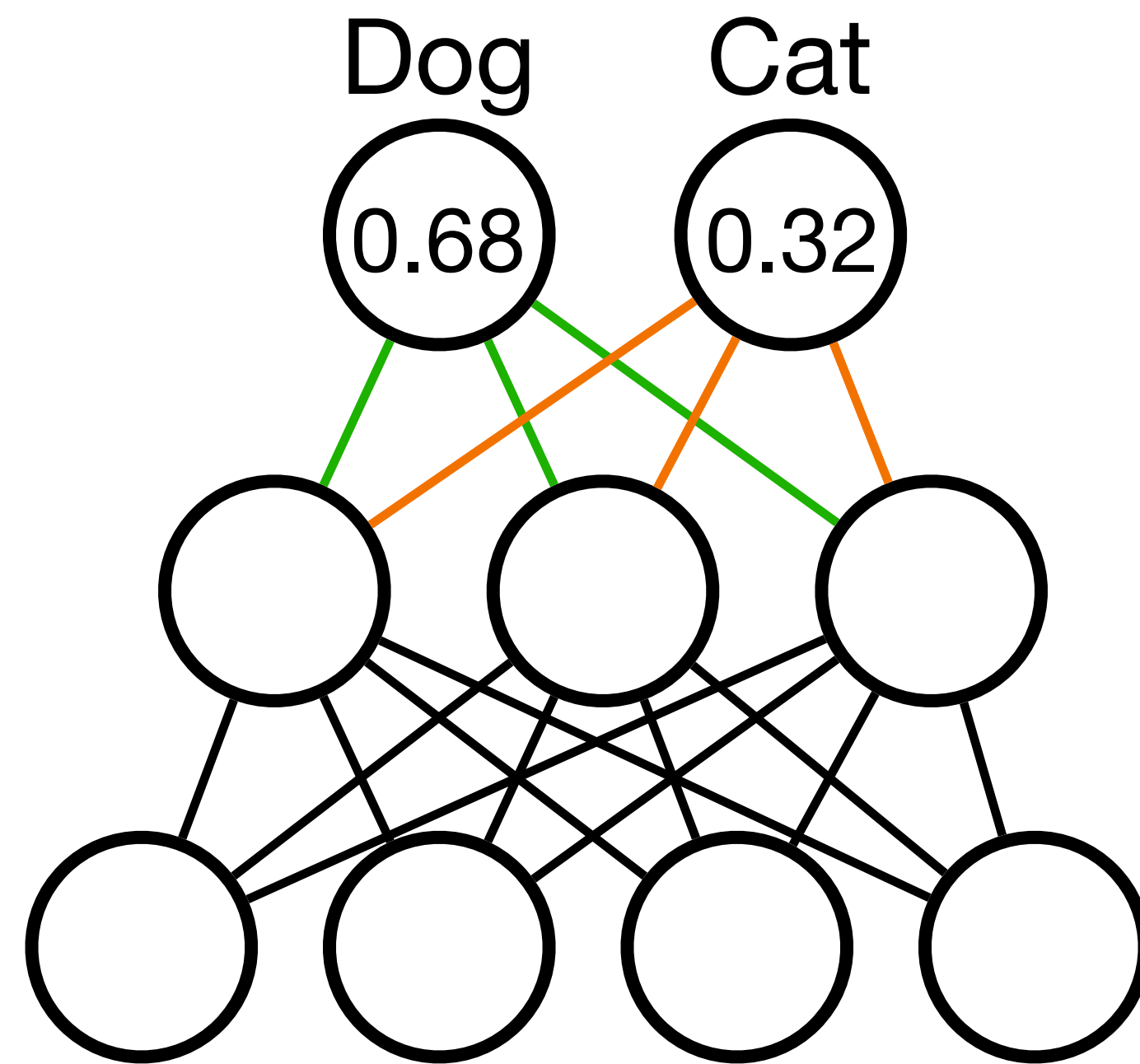


x_4

How to integrate constraints in the ML model?



Multi-class classification problems



$\mathbf{x}_1$

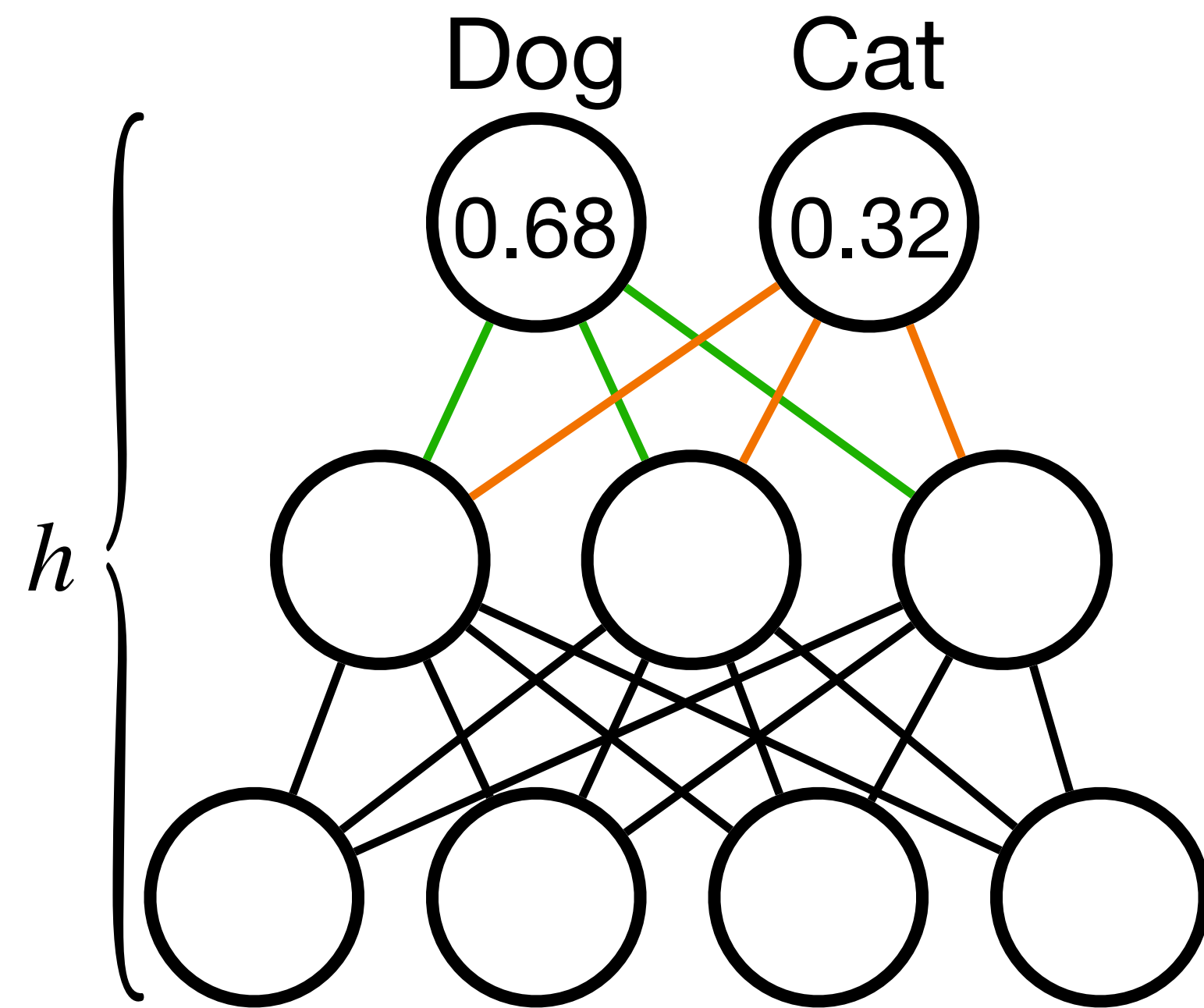


$\mathbf{x}_2$

How to integrate constraints in the ML model?



Multi-class classification problems



$\mathbf{x}_1$

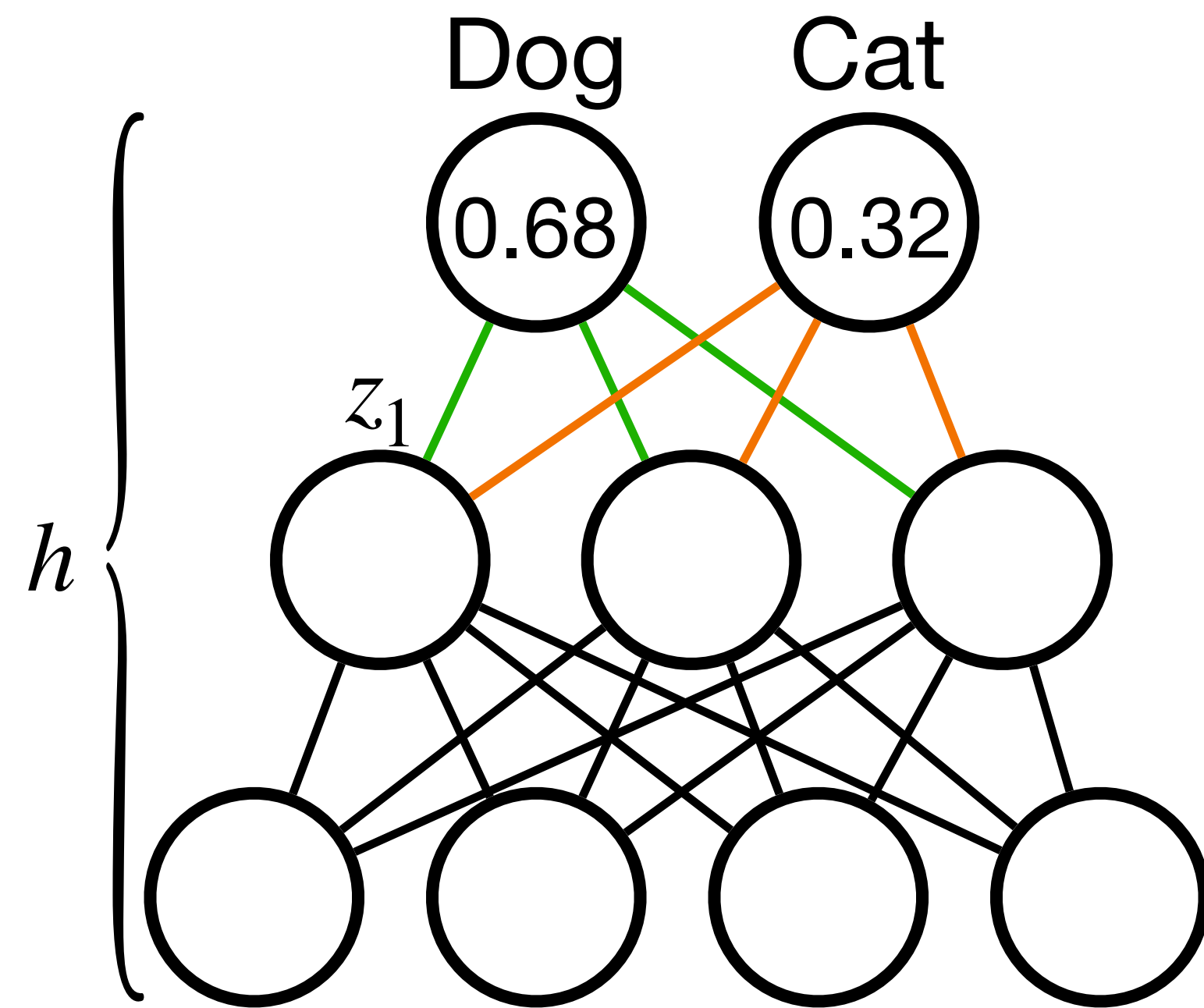


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

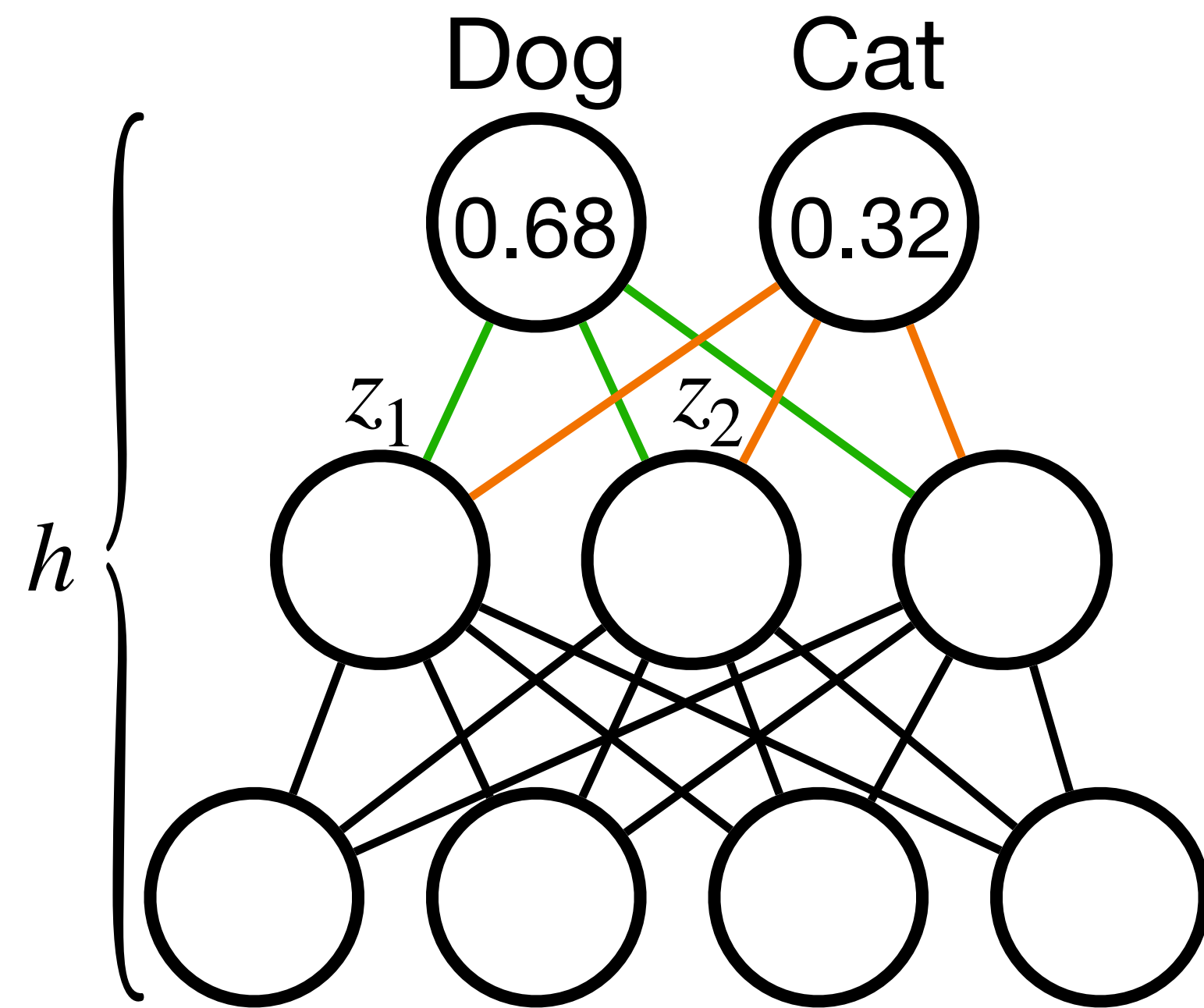


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

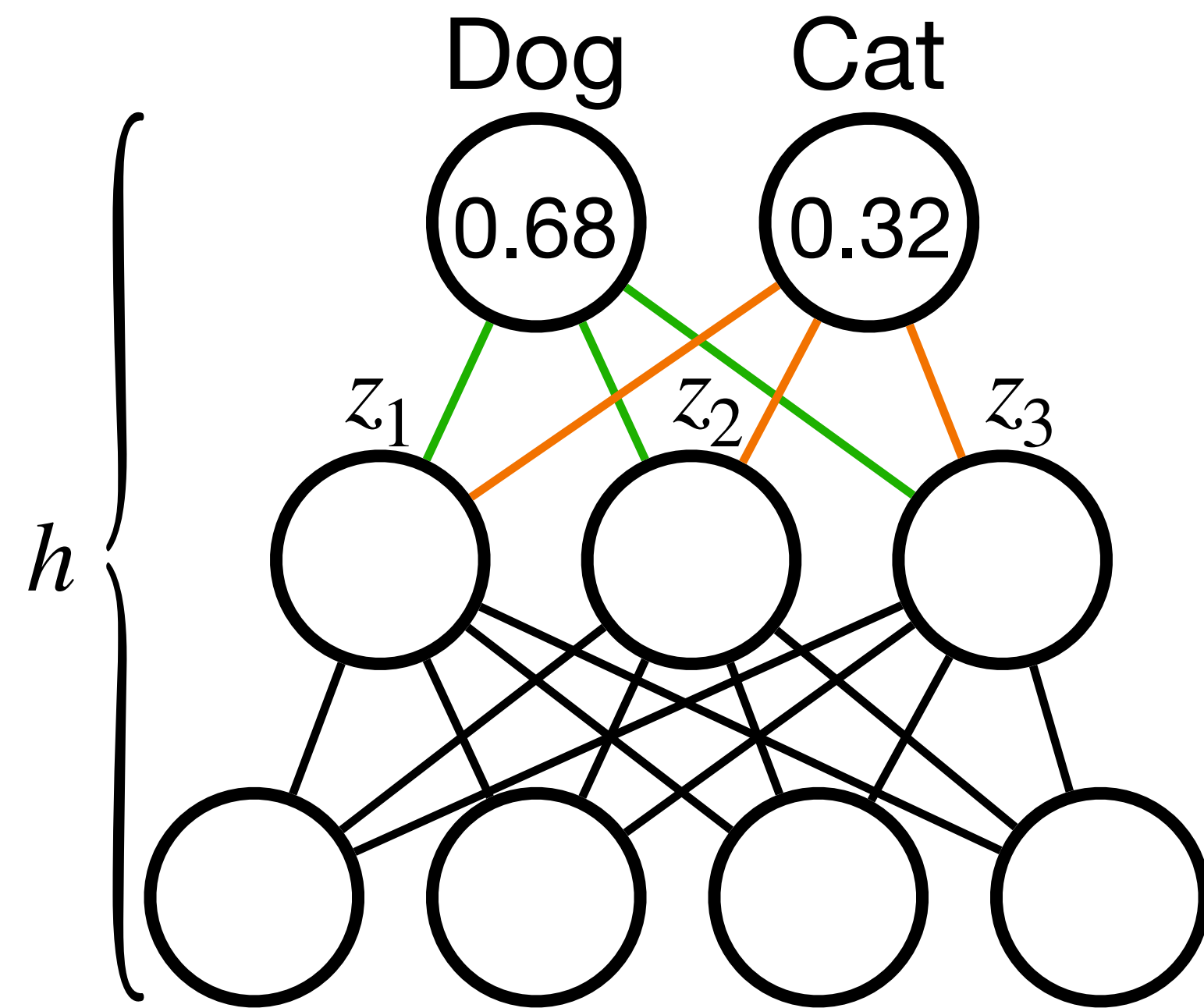


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

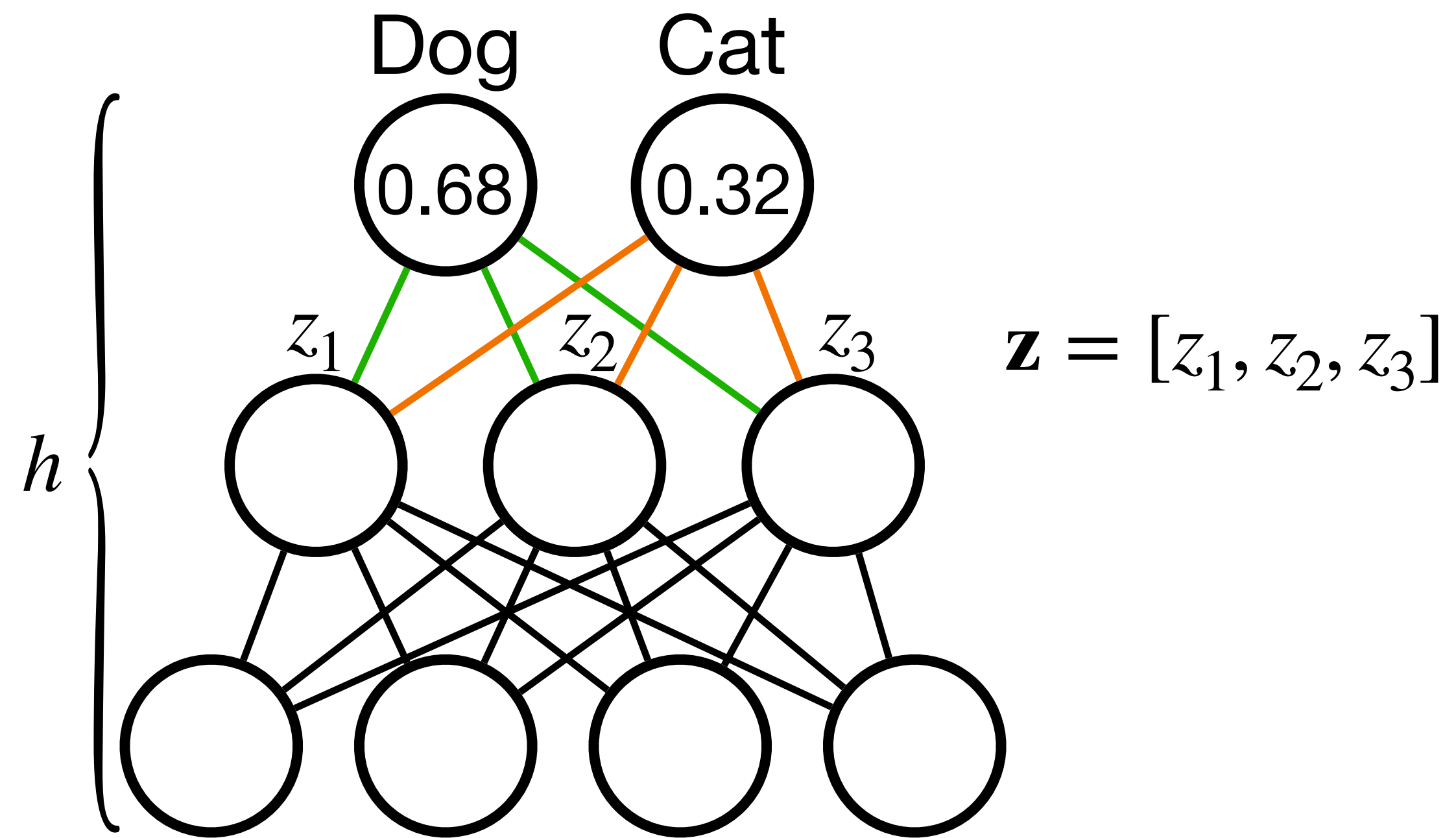


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

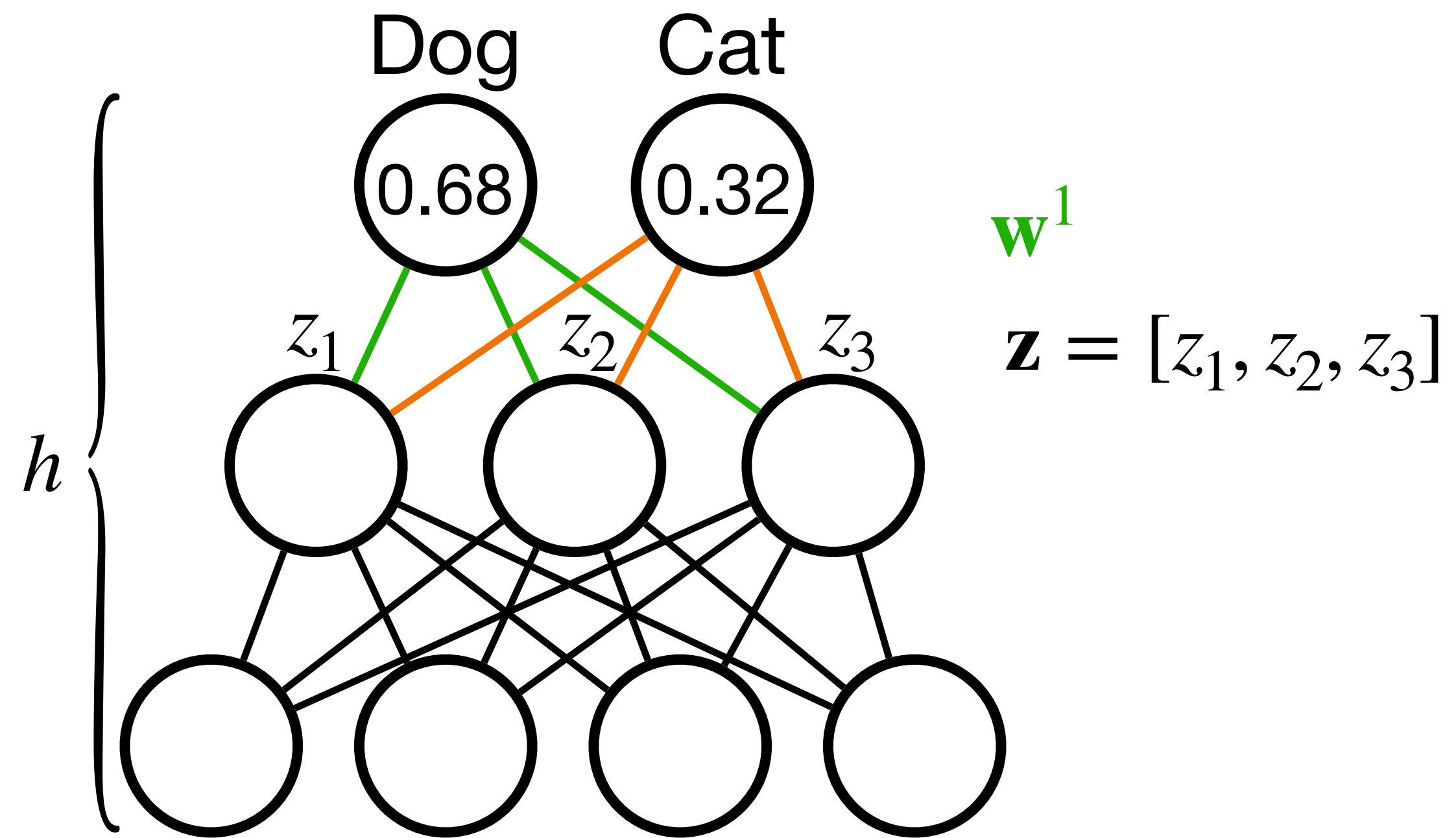


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

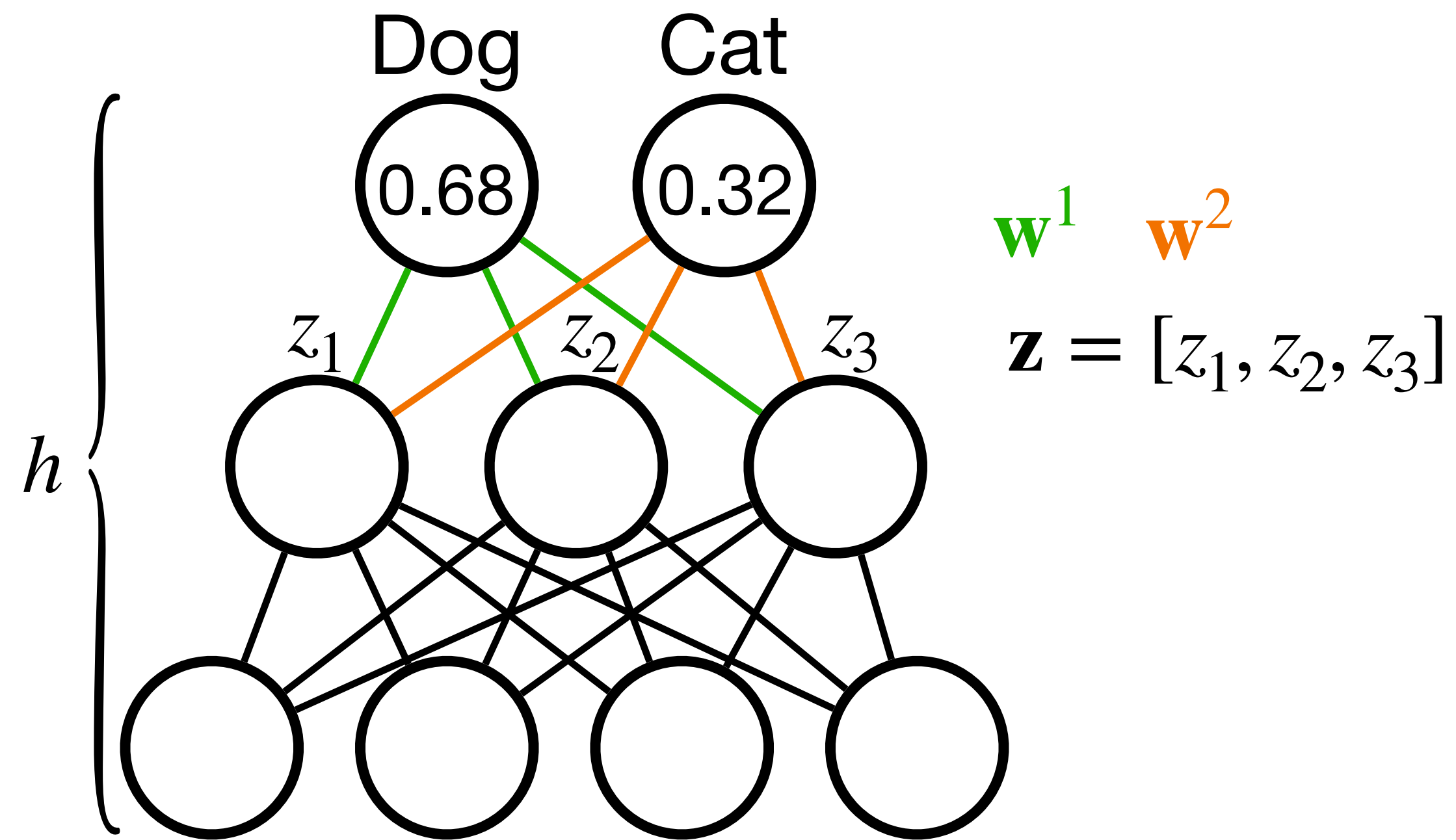


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$

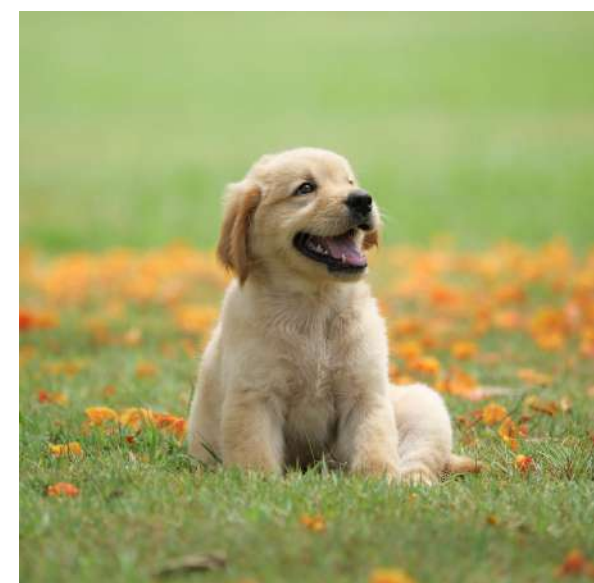
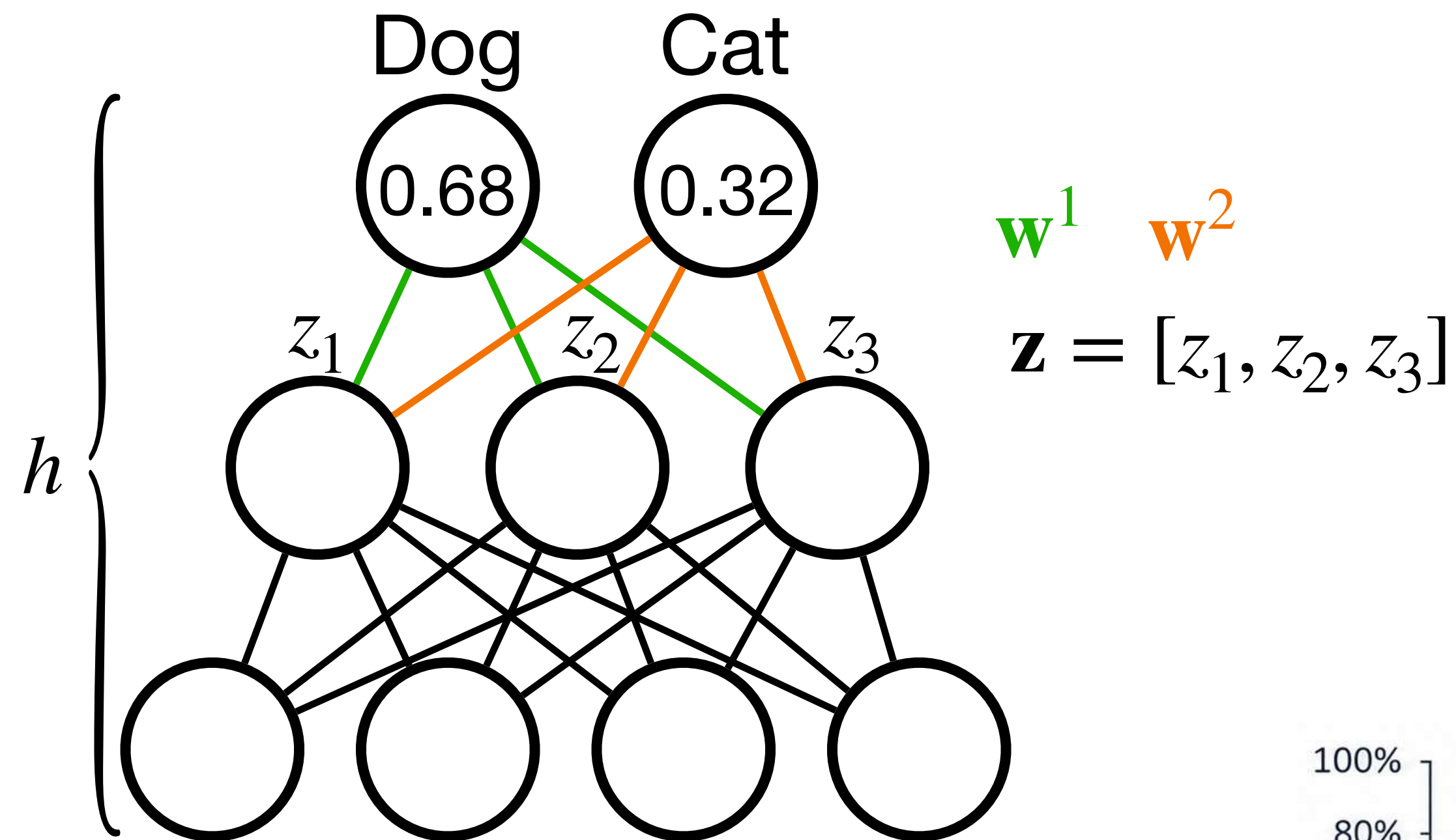


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



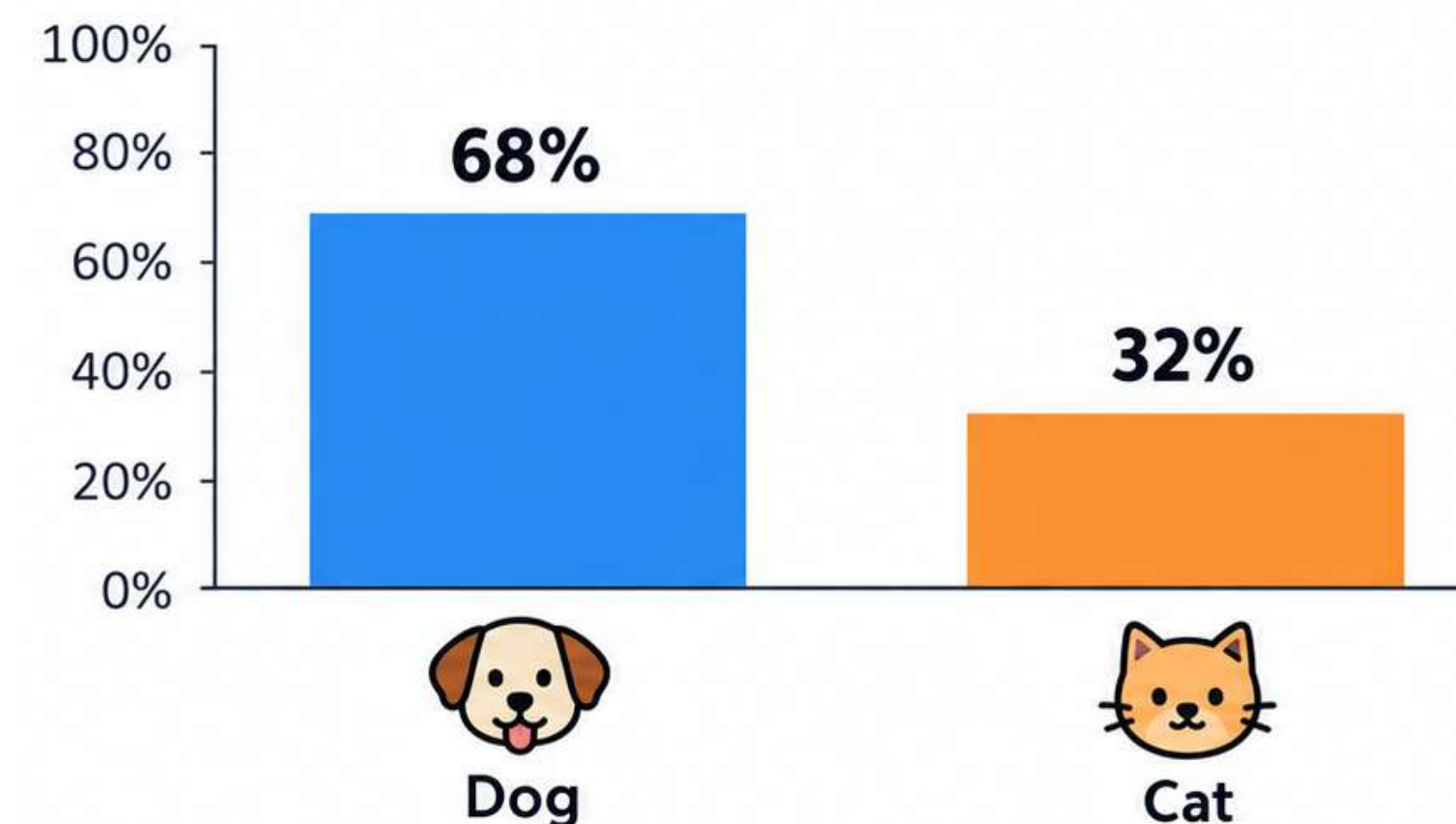
$\mathbf{x}_1$



$\mathbf{x}_2$

$$h_D = \text{softmax}(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{e^{\mathbf{w}^1 \mathbf{z} + b^1}}{\sum_{i=1}^N e^{\mathbf{w}^i \mathbf{z} + b^i}}$$

of classes

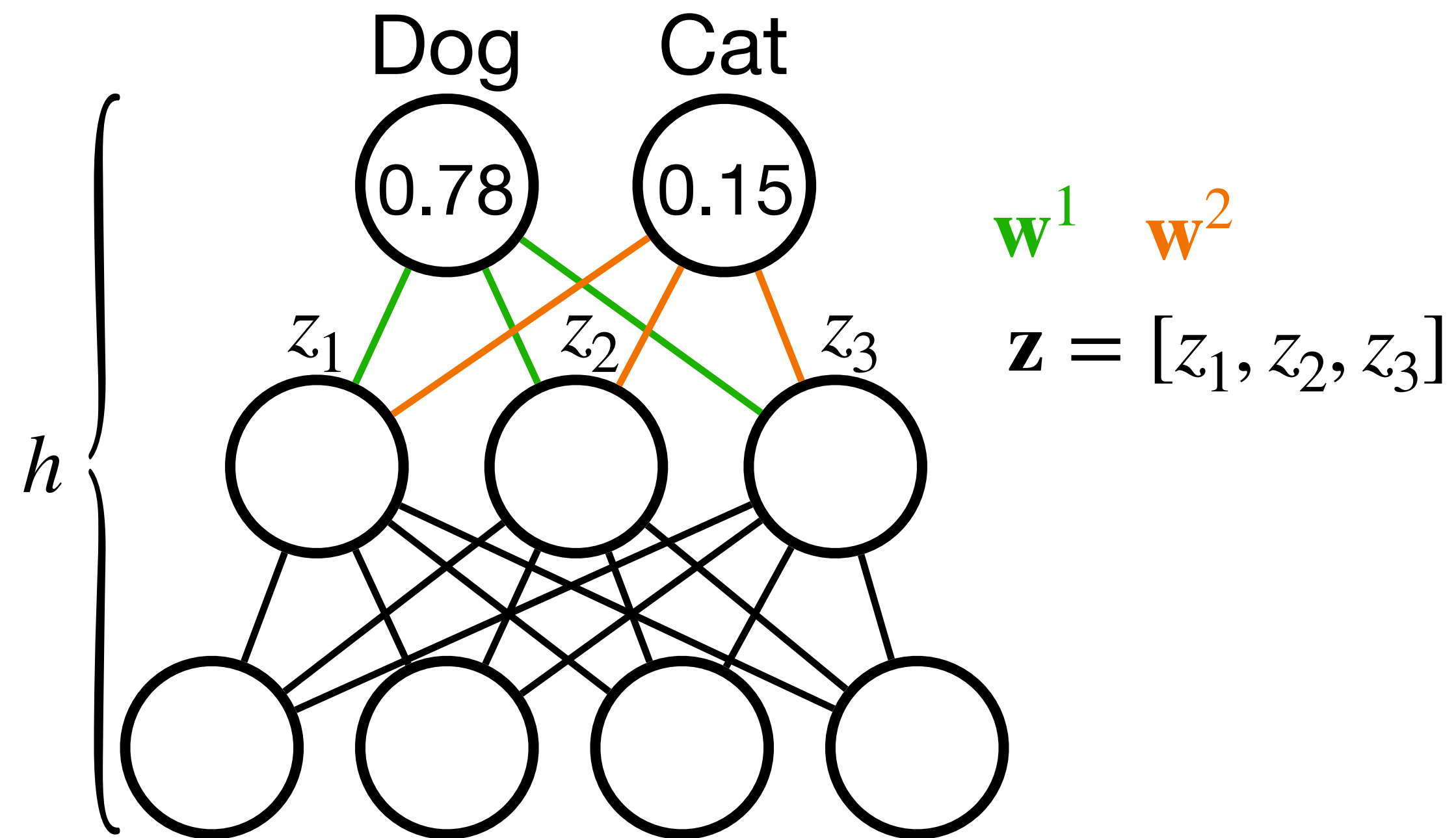


$$p = \text{argmax}(h_D, h_C)$$



How to integrate constraints in the ML model?

Multi-class classification problems



$$h_D = \text{softmax}(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{e^{\mathbf{w}^1 \mathbf{z} + b^1}}{\sum_{i=1}^N e^{\mathbf{w}^i \mathbf{z} + b^i}}$$

of classes



$\mathbf{x}_1$

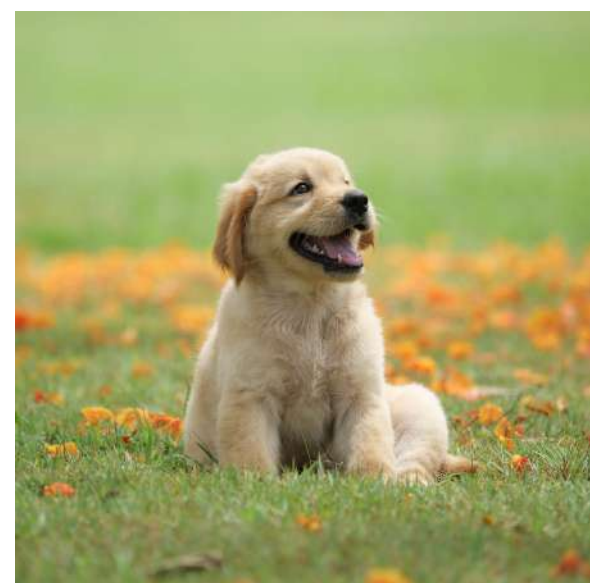
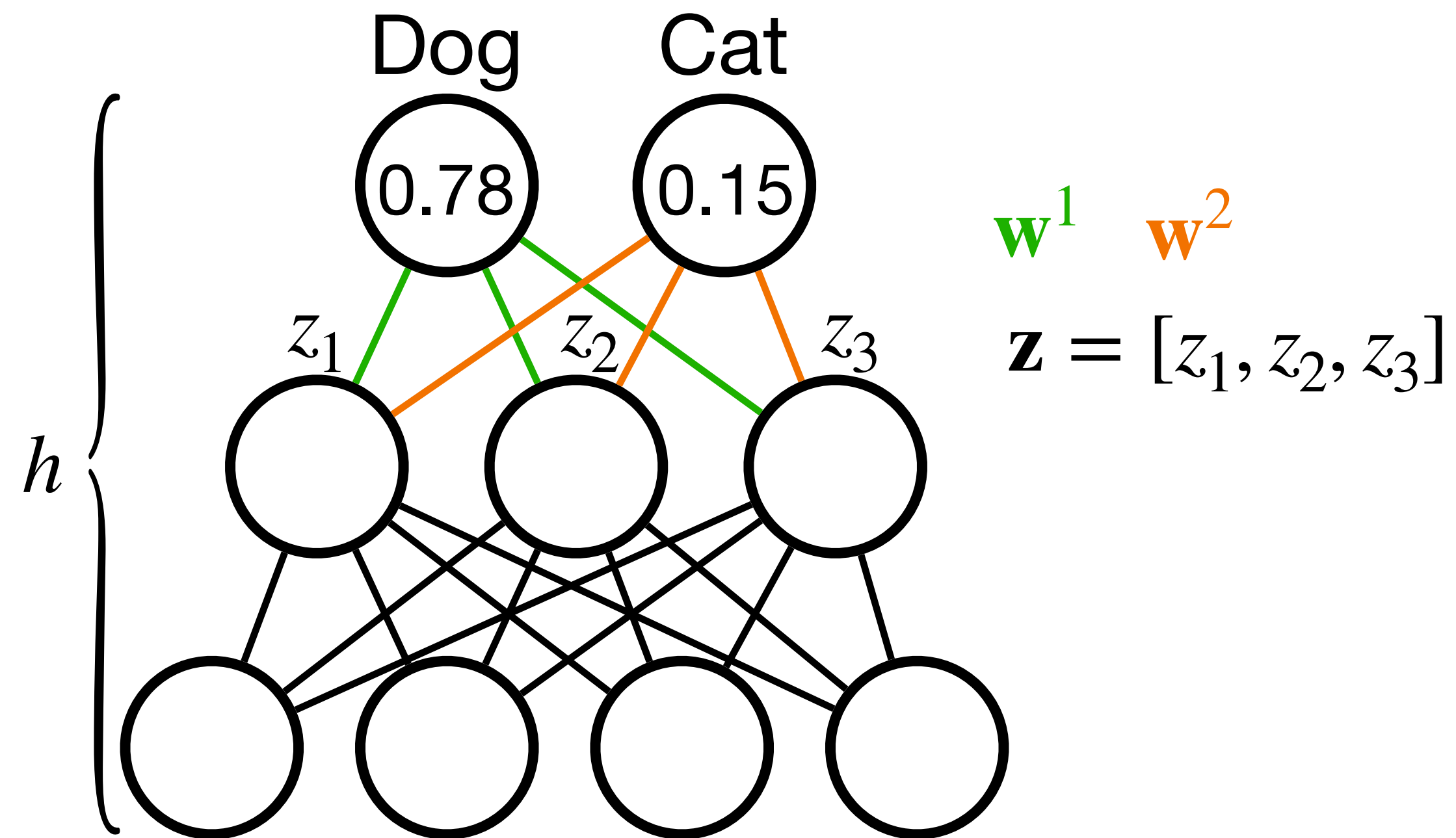


$\mathbf{x}_2$



How to integrate constraints in the ML model?

Multi-class classification problems



$\mathbf{x}_1$



$\mathbf{x}_2$

$$h_D = \text{softmax}(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{e^{\mathbf{w}^1 \mathbf{z} + b^1}}{\sum_{i=1}^N e^{\mathbf{w}^i \mathbf{z} + b^i}}$$

of classes

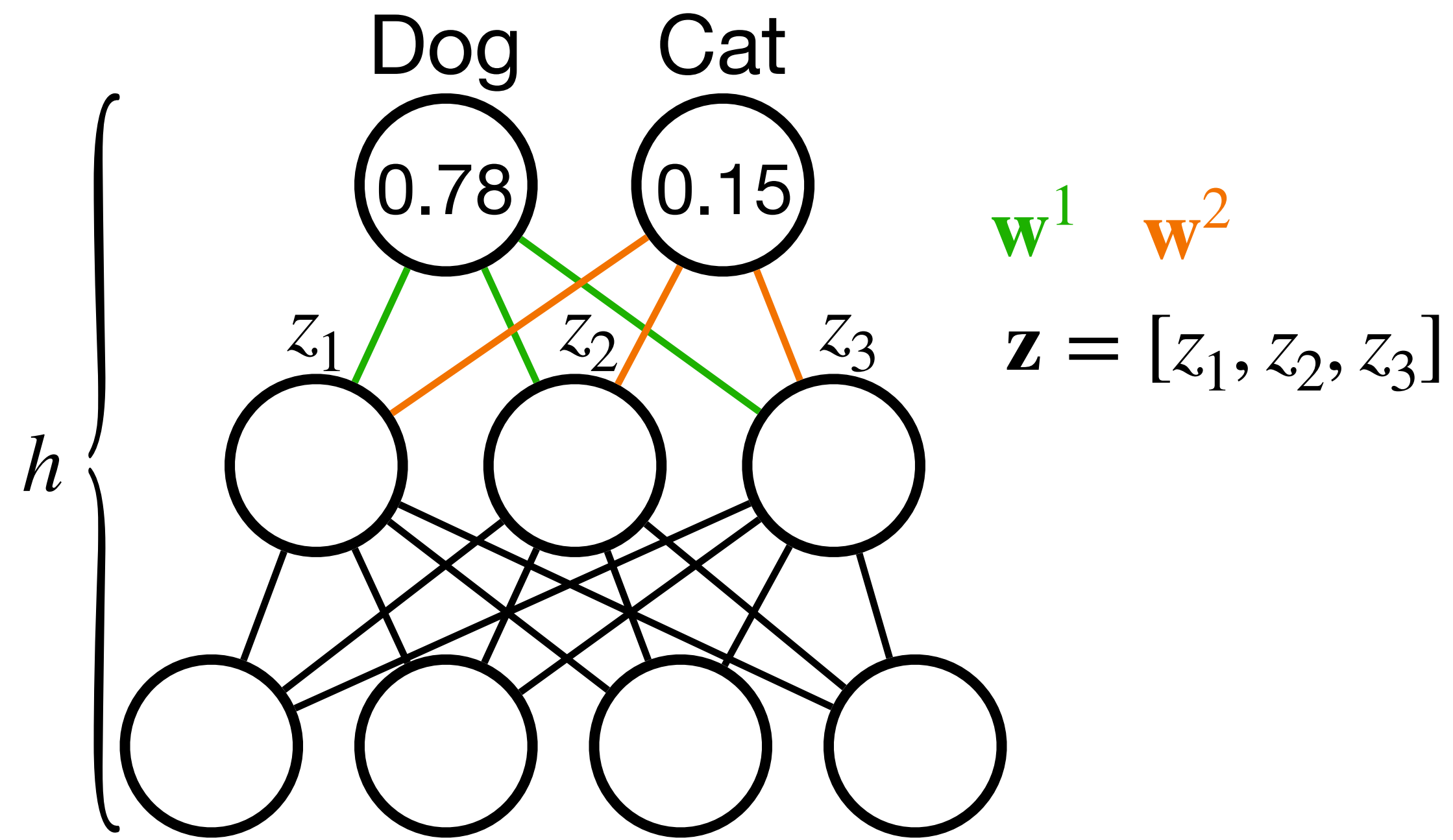
Cross-entropy function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \log(y_i \text{softmax}(\mathbf{w}^i \mathbf{z} + b^i))$$

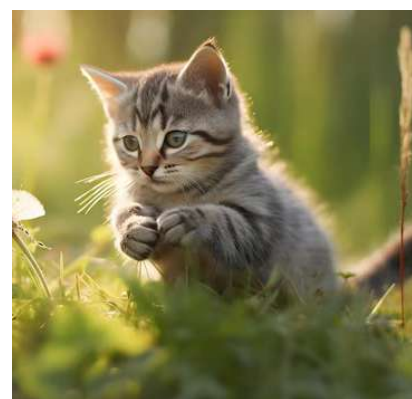


How to integrate constraints in the ML model?

Multi-label classification problems



$\mathbf{x}_1$



$\mathbf{x}_2$



$\mathbf{x}_3$

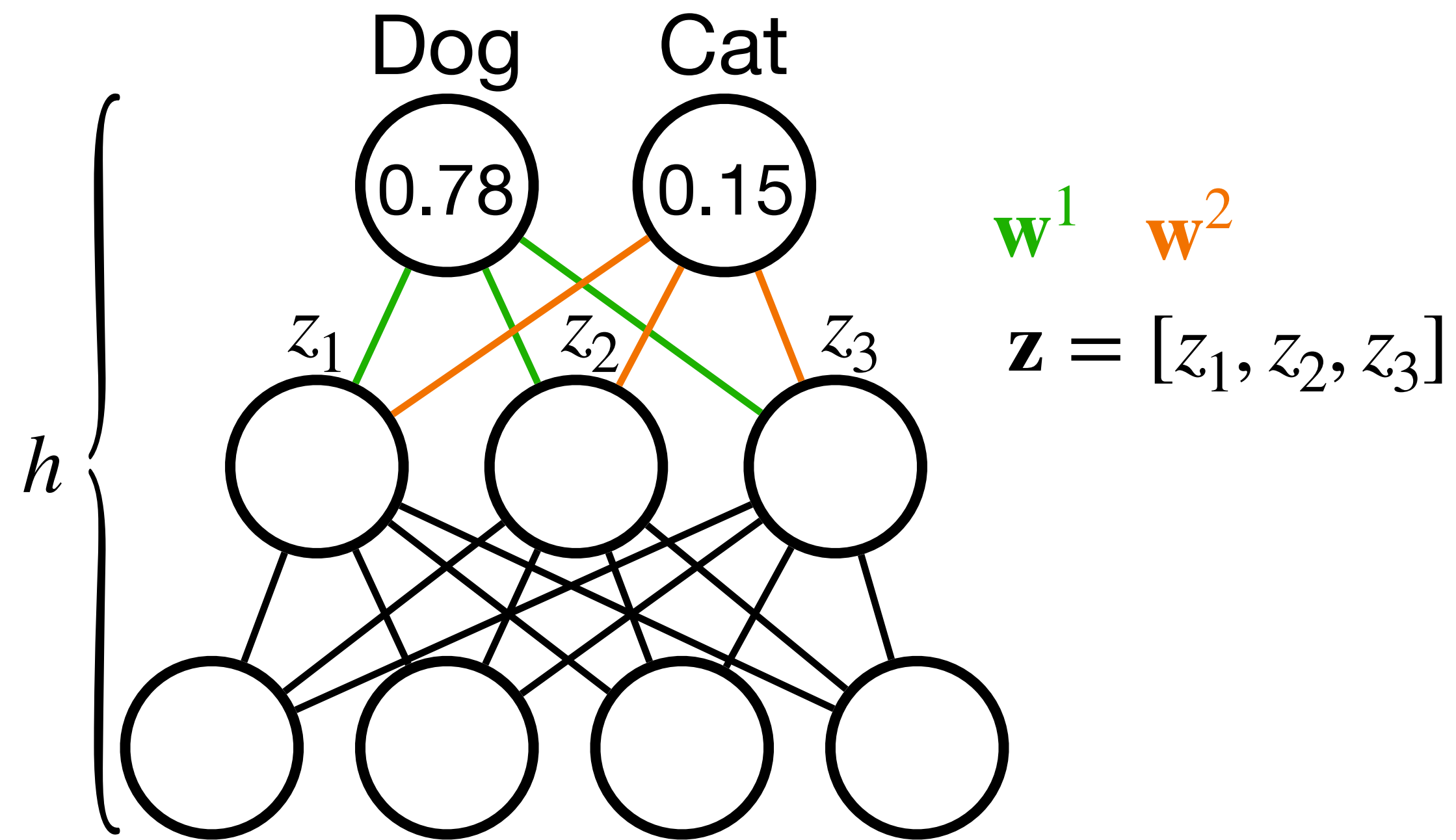


$\mathbf{x}_4$



How to integrate constraints in the ML model?

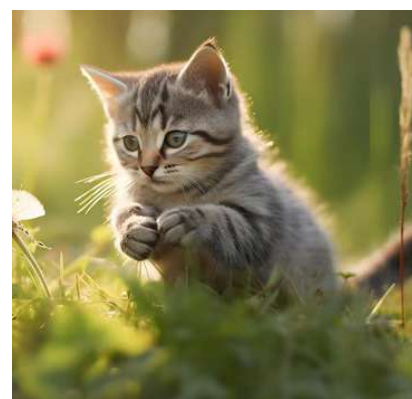
Multi-label classification problems



$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-\frac{1}{1}(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-\frac{1}{1}(\mathbf{w}^2 \mathbf{z} + b^2)}}$$



$\mathbf{x}_1$



$\mathbf{x}_2$



$\mathbf{x}_3$

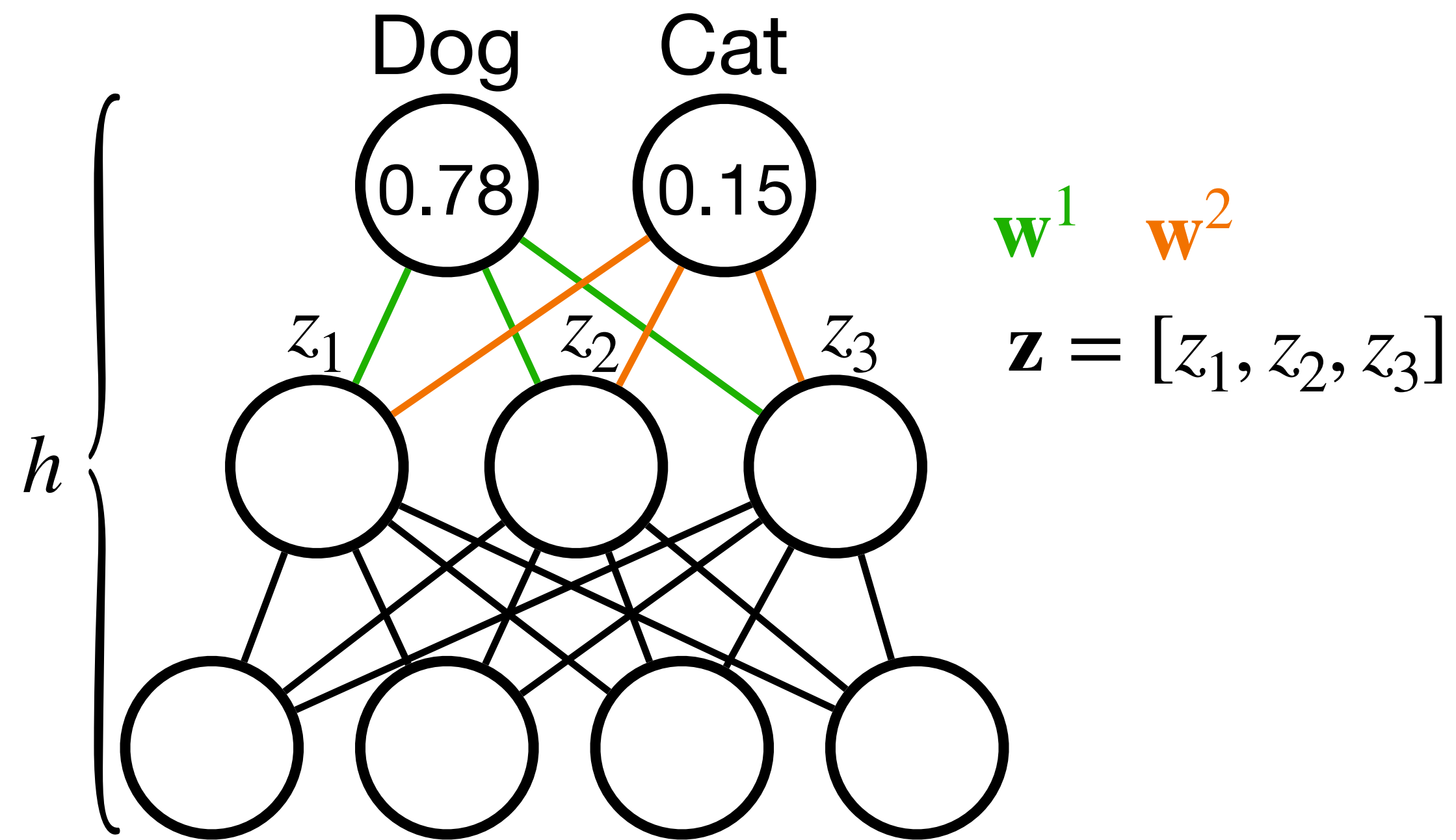


$\mathbf{x}_4$

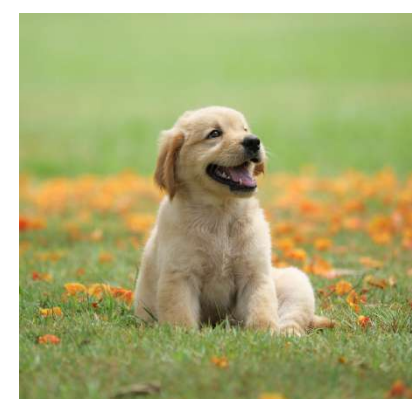


How to integrate constraints in the ML model?

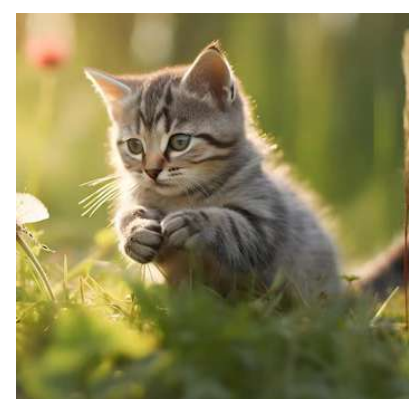
Multi-label classification problems



$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-1(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-1(\mathbf{w}^2 \mathbf{z} + b^2)}}$$



$\mathbf{x}_1$



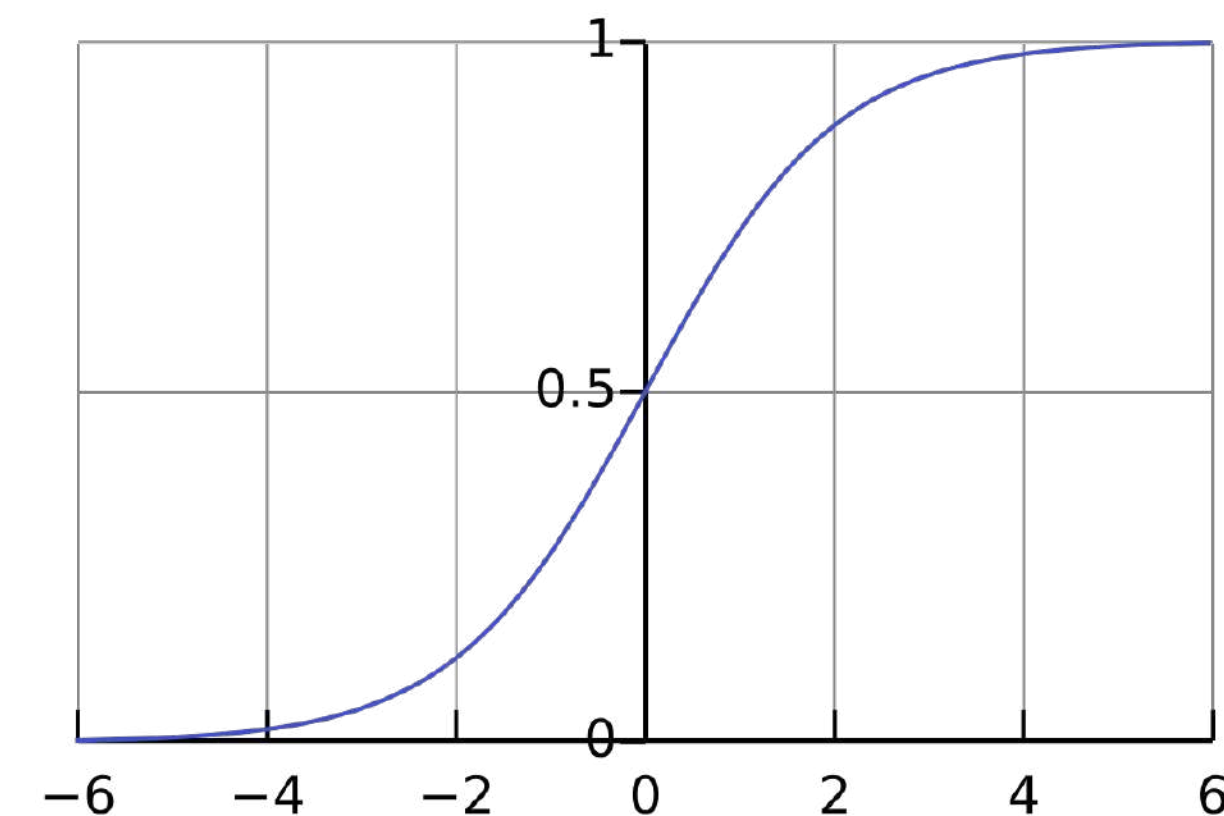
$\mathbf{x}_2$



$\mathbf{x}_3$



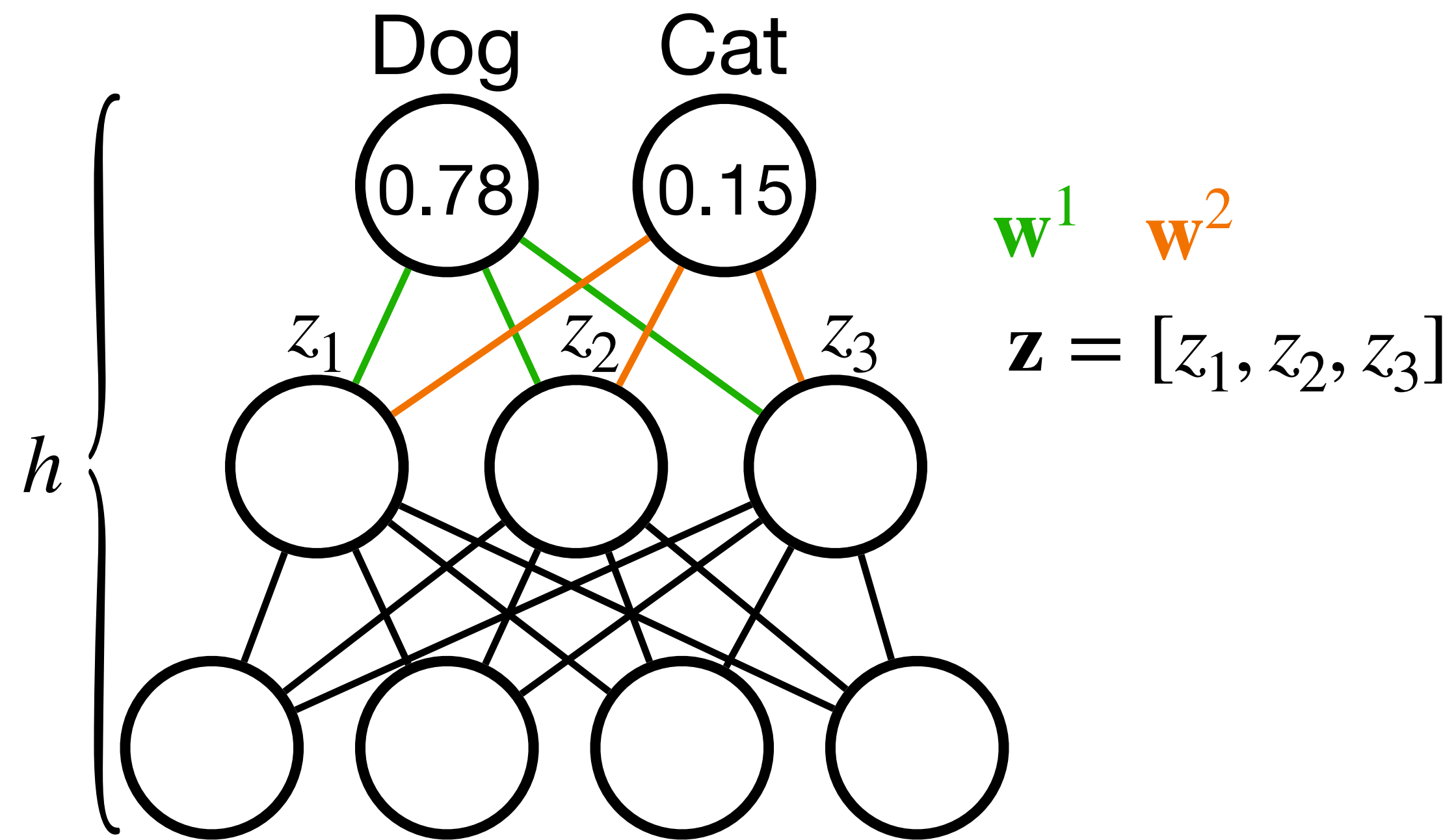
$\mathbf{x}_4$





How to integrate constraints in the ML model?

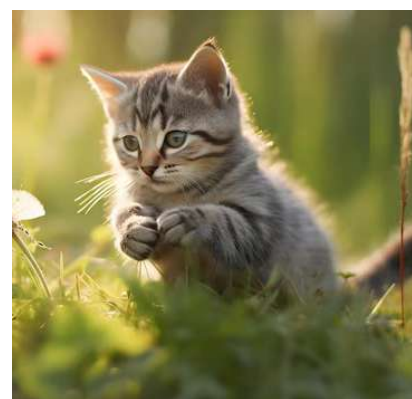
Multi-label classification problems



$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-1(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-1(\mathbf{w}^2 \mathbf{z} + b^2)}}$$



$\mathbf{x}_1$



$\mathbf{x}_2$



$\mathbf{x}_3$

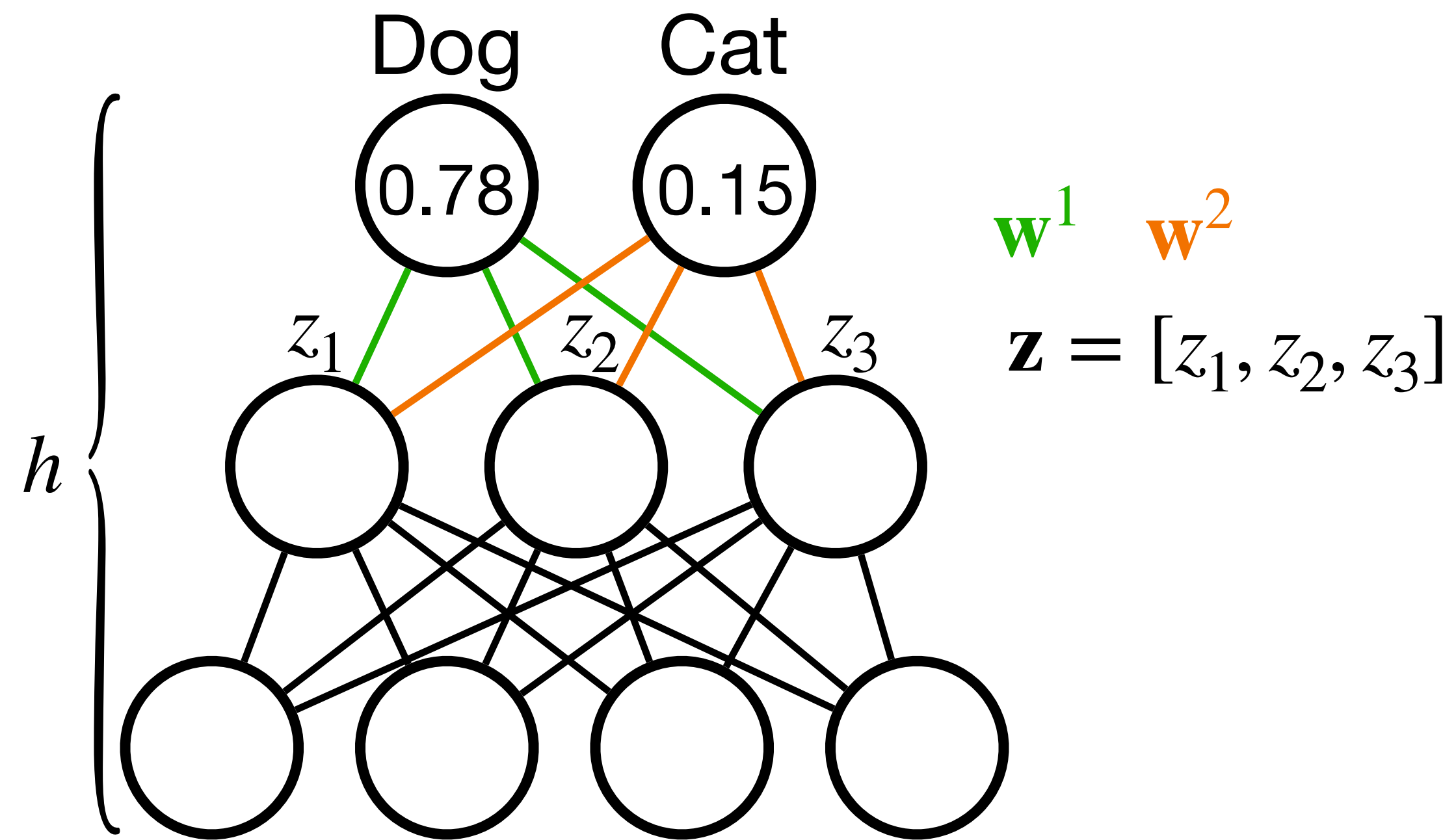


$\mathbf{x}_4$

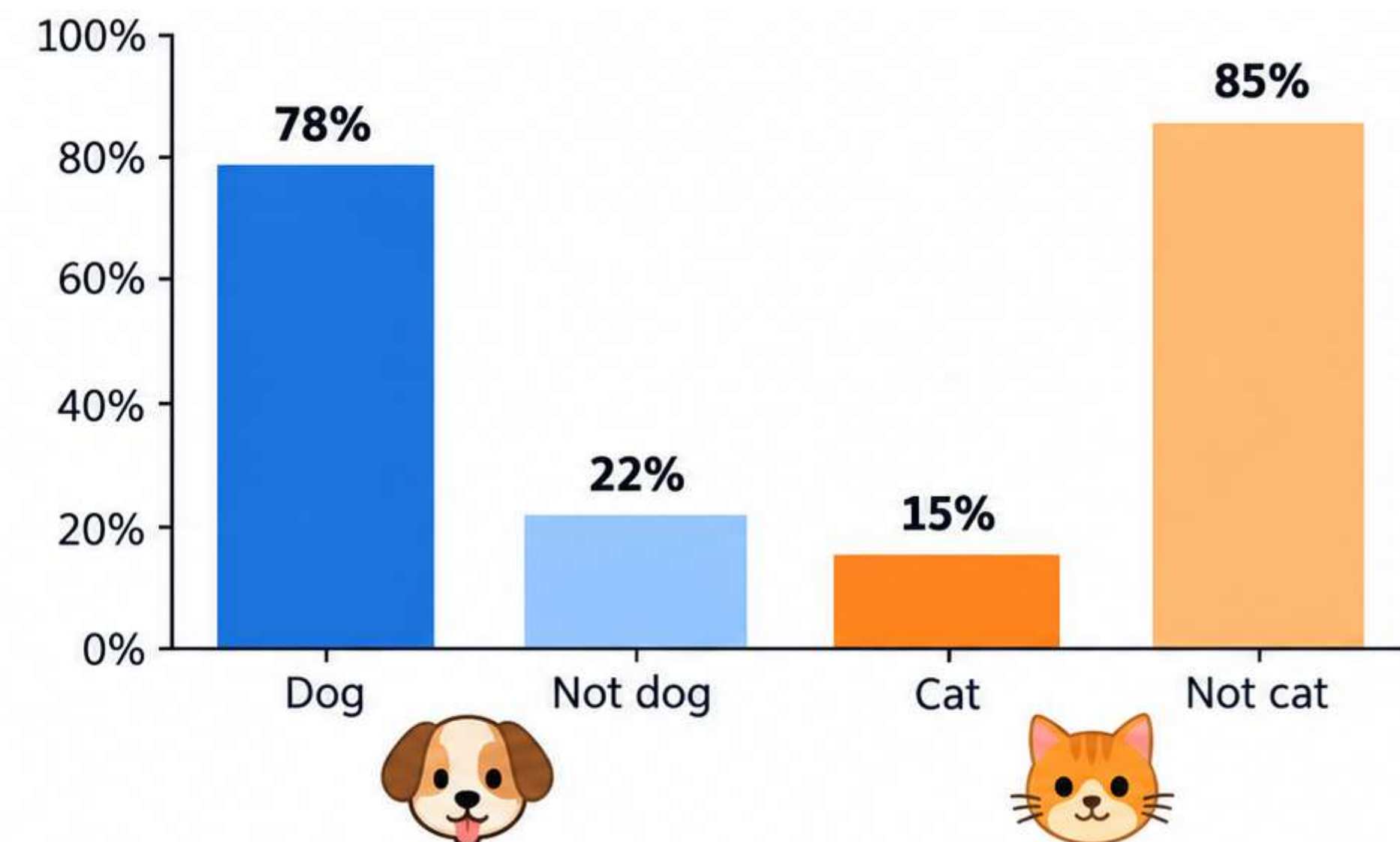
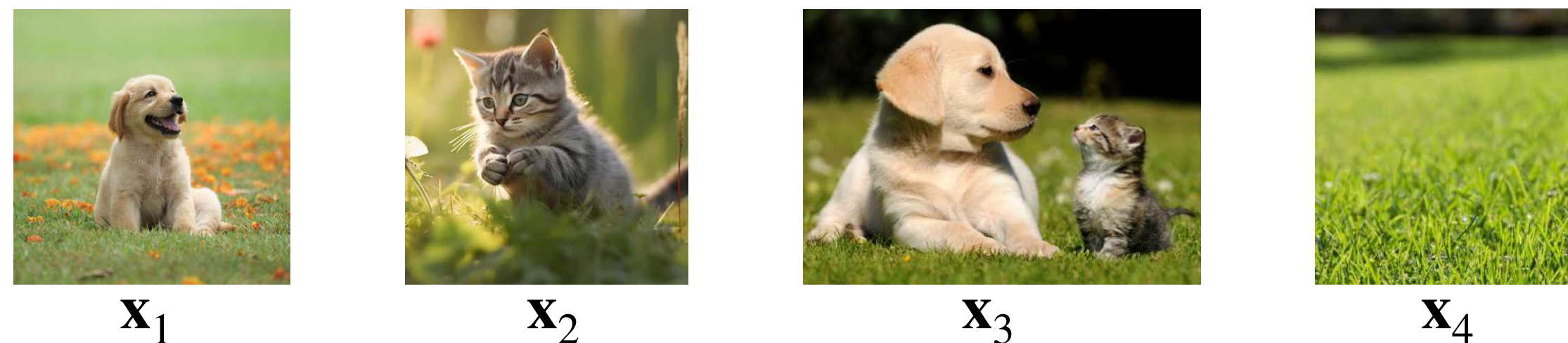


How to integrate constraints in the ML model?

Multi-label classification problems



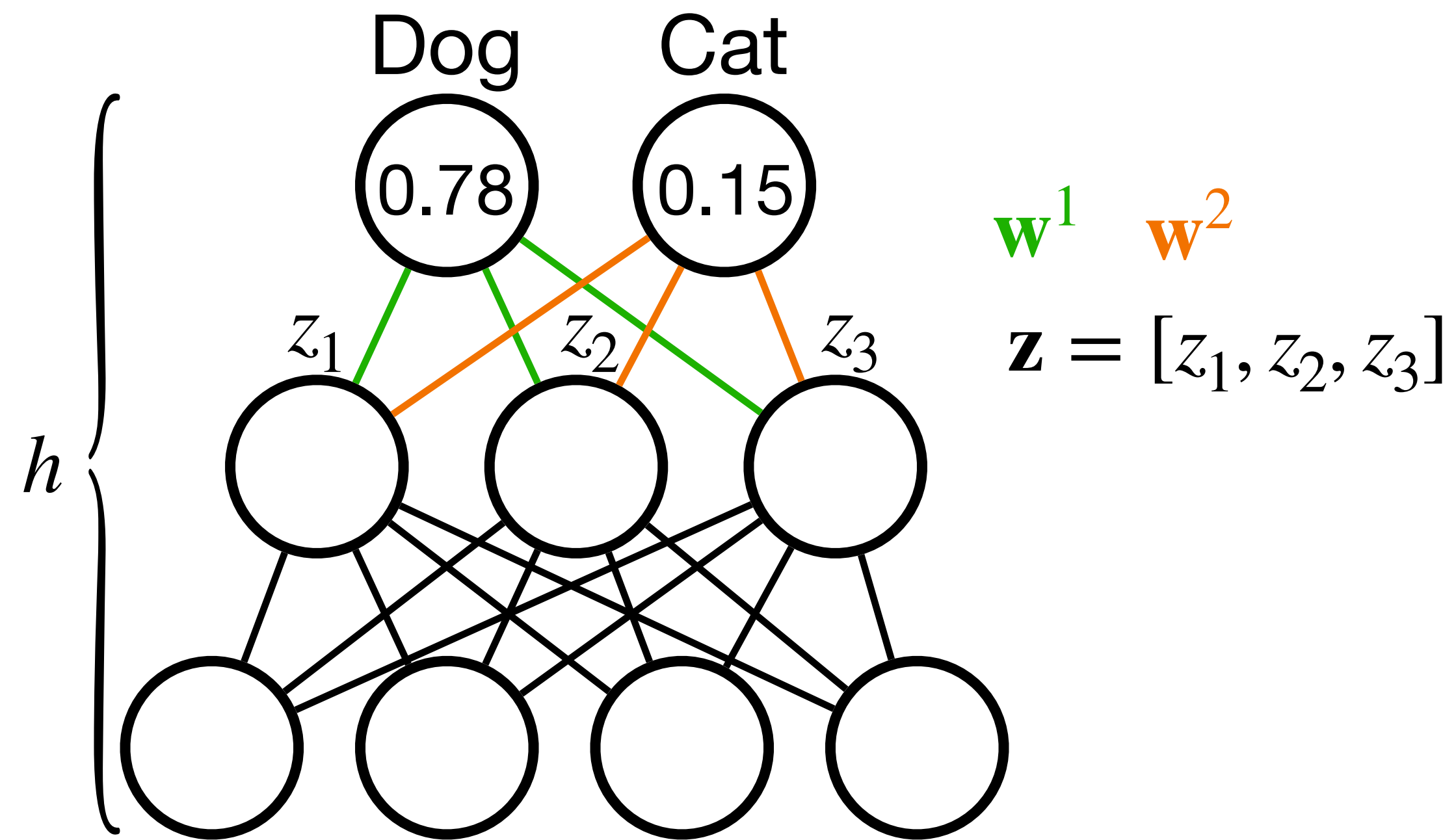
$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-1(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-1(\mathbf{w}^2 \mathbf{z} + b^2)}}$$





How to integrate constraints in the ML model?

Multi-label classification problems



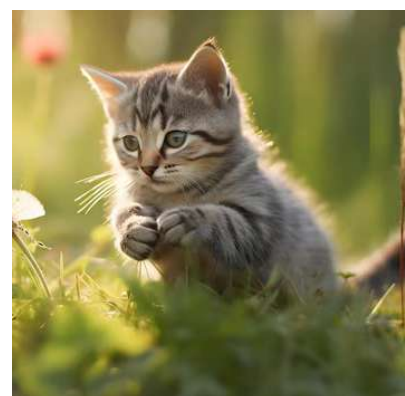
$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-1(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-1(\mathbf{w}^2 \mathbf{z} + b^2)}}$$

$$p_D = \mathbb{I}(h_D \geq \theta)$$

$$p_C = \mathbb{I}(h_C \geq \theta)$$



$\mathbf{x}_1$



$\mathbf{x}_2$



$\mathbf{x}_3$

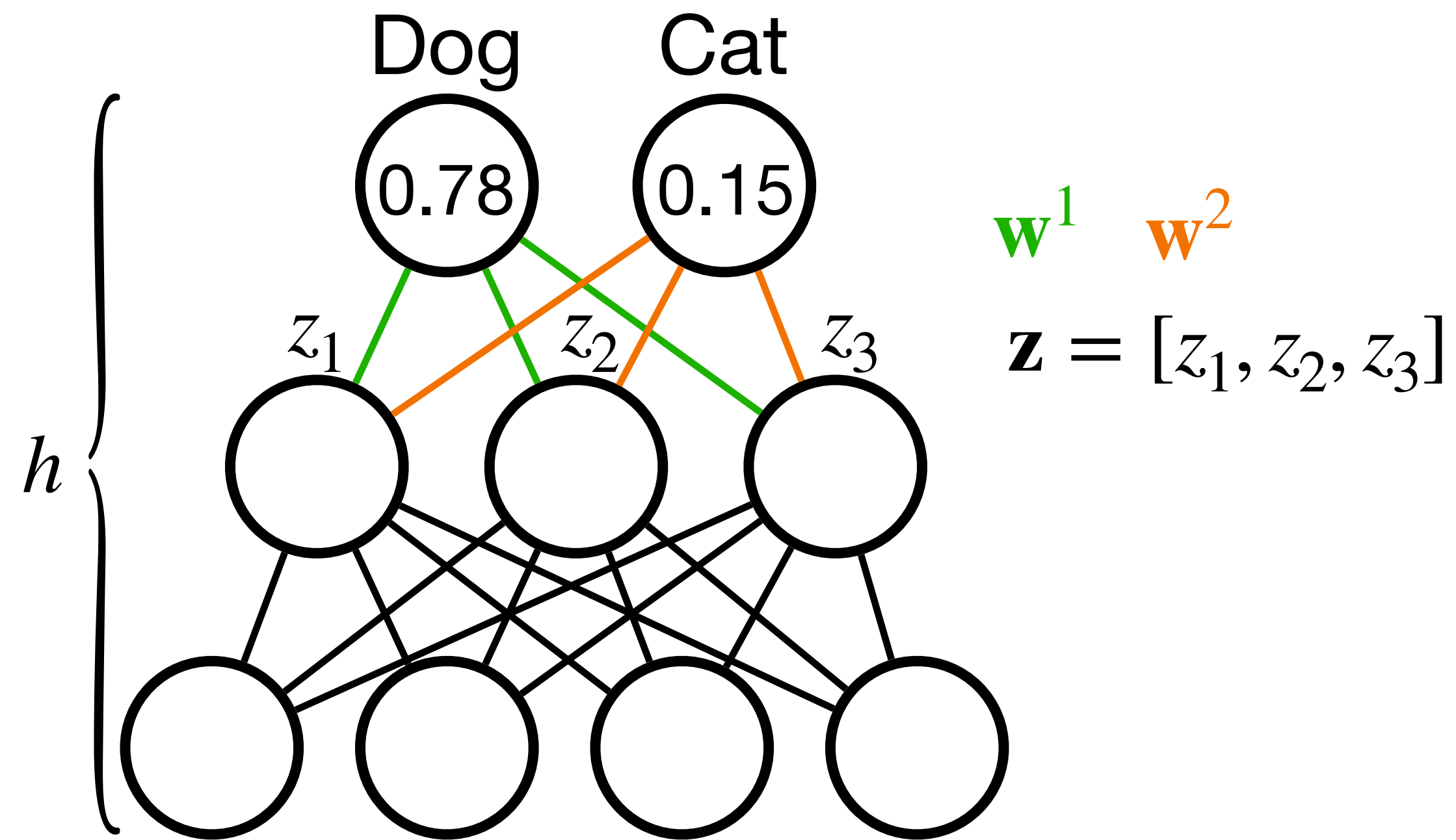


$\mathbf{x}_4$



How to integrate constraints in the ML model?

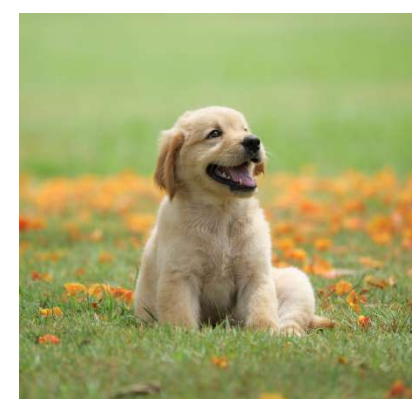
Multi-label classification problems



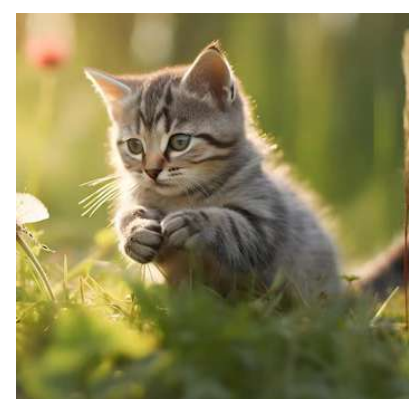
$$h_D = \sigma(\mathbf{w}^1 \mathbf{z} + b^1) = \frac{1}{1 + e^{-1(\mathbf{w}^1 \mathbf{z} + b^1)}}$$
$$h_C = \sigma(\mathbf{w}^2 \mathbf{z} + b^2) = \frac{1}{1 + e^{-1(\mathbf{w}^2 \mathbf{z} + b^2)}}$$

Binary Cross-Entropy loss function:

$$\mathcal{L} = - \sum_{i=1}^N y_i \log(\sigma(\mathbf{w}^i \mathbf{z} + b^i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^i \mathbf{z} + b^i))$$



$\mathbf{x}_1$



$\mathbf{x}_2$

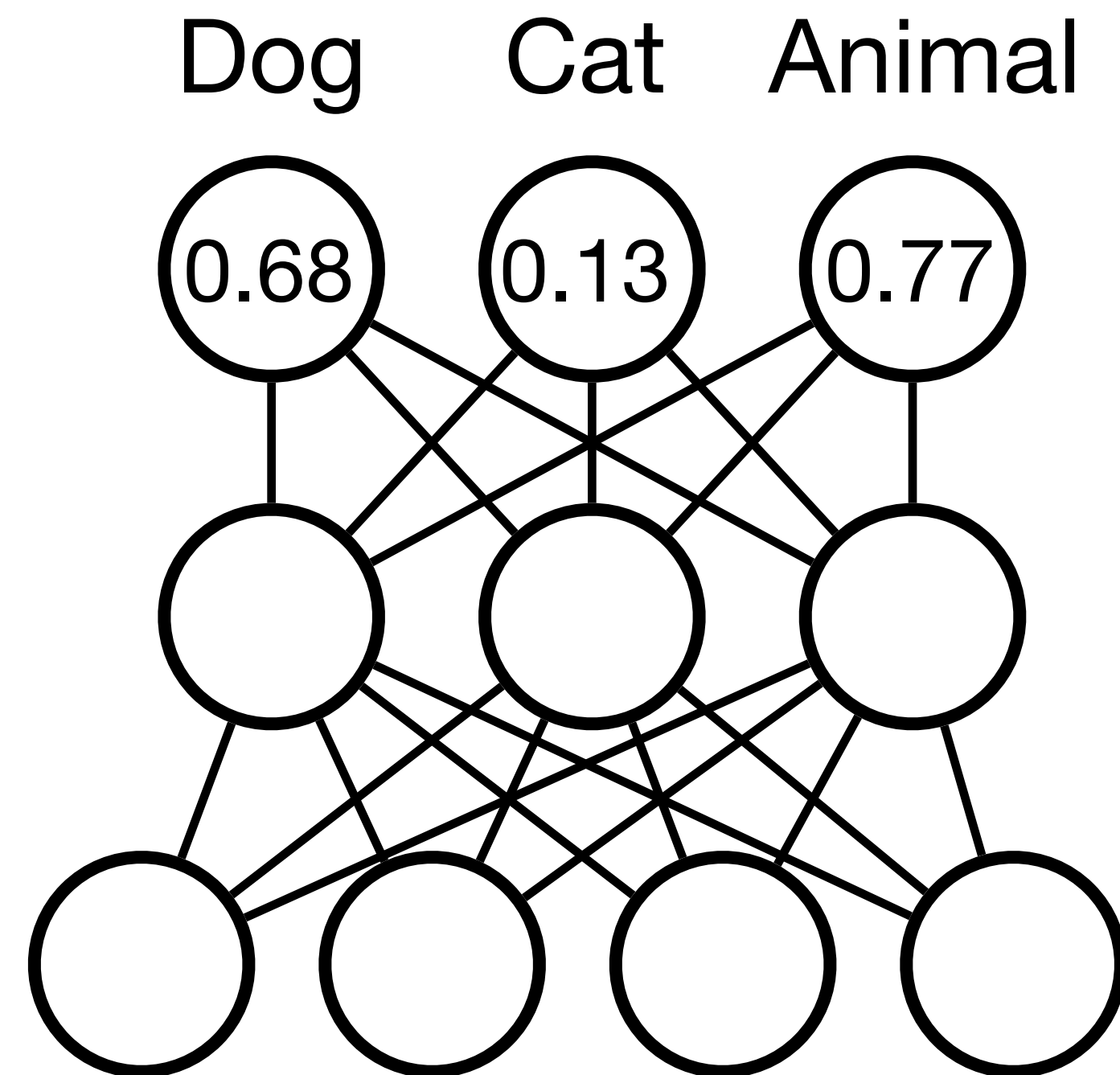


$\mathbf{x}_3$



$\mathbf{x}_4$

Integrate the requirements into the loss function!

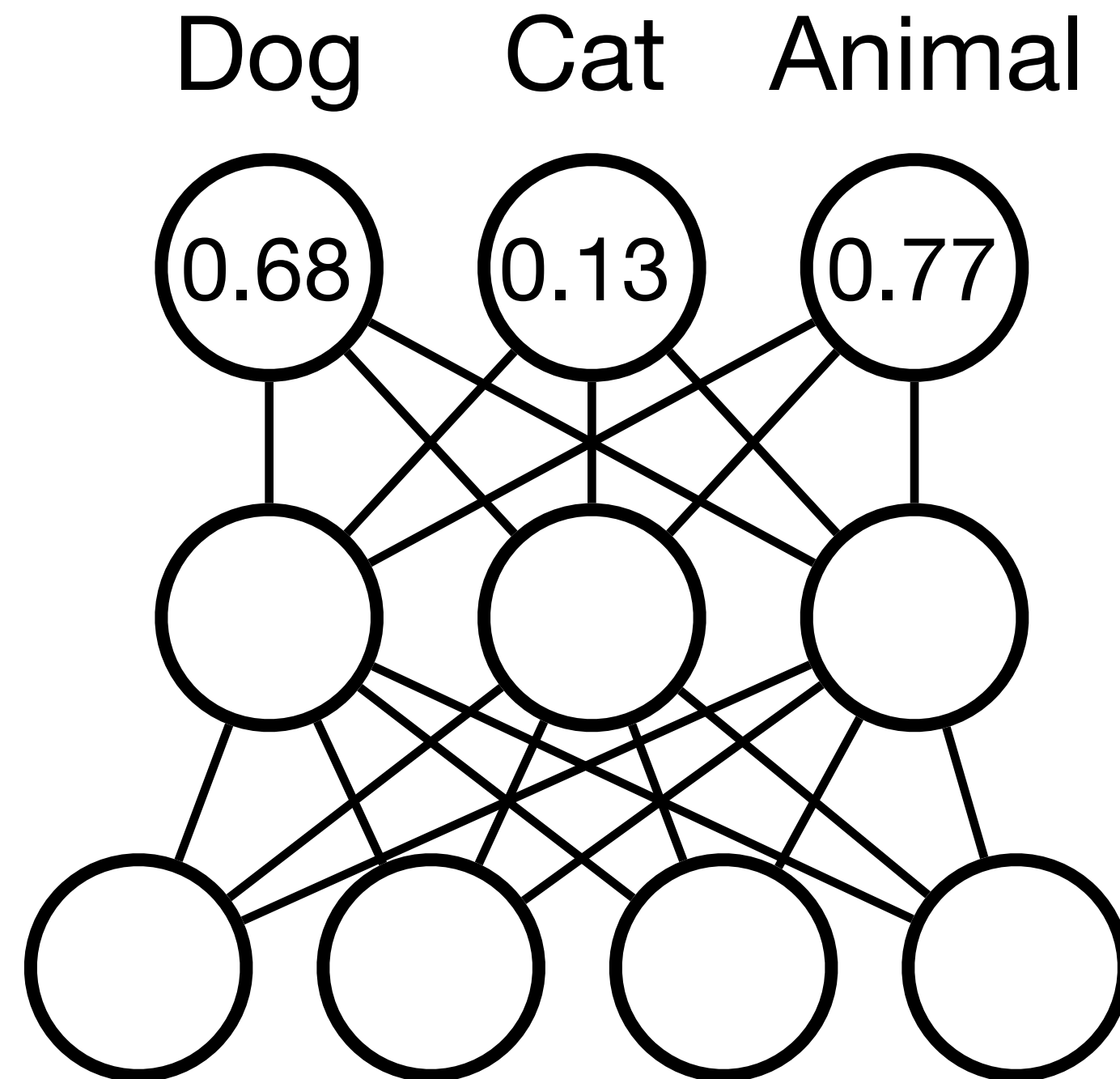




Integrate the requirements into the loss function!

In this scenario, we would like to inform our network that:

Dog $\rightarrow$ Animal Cat $\rightarrow$ Animal



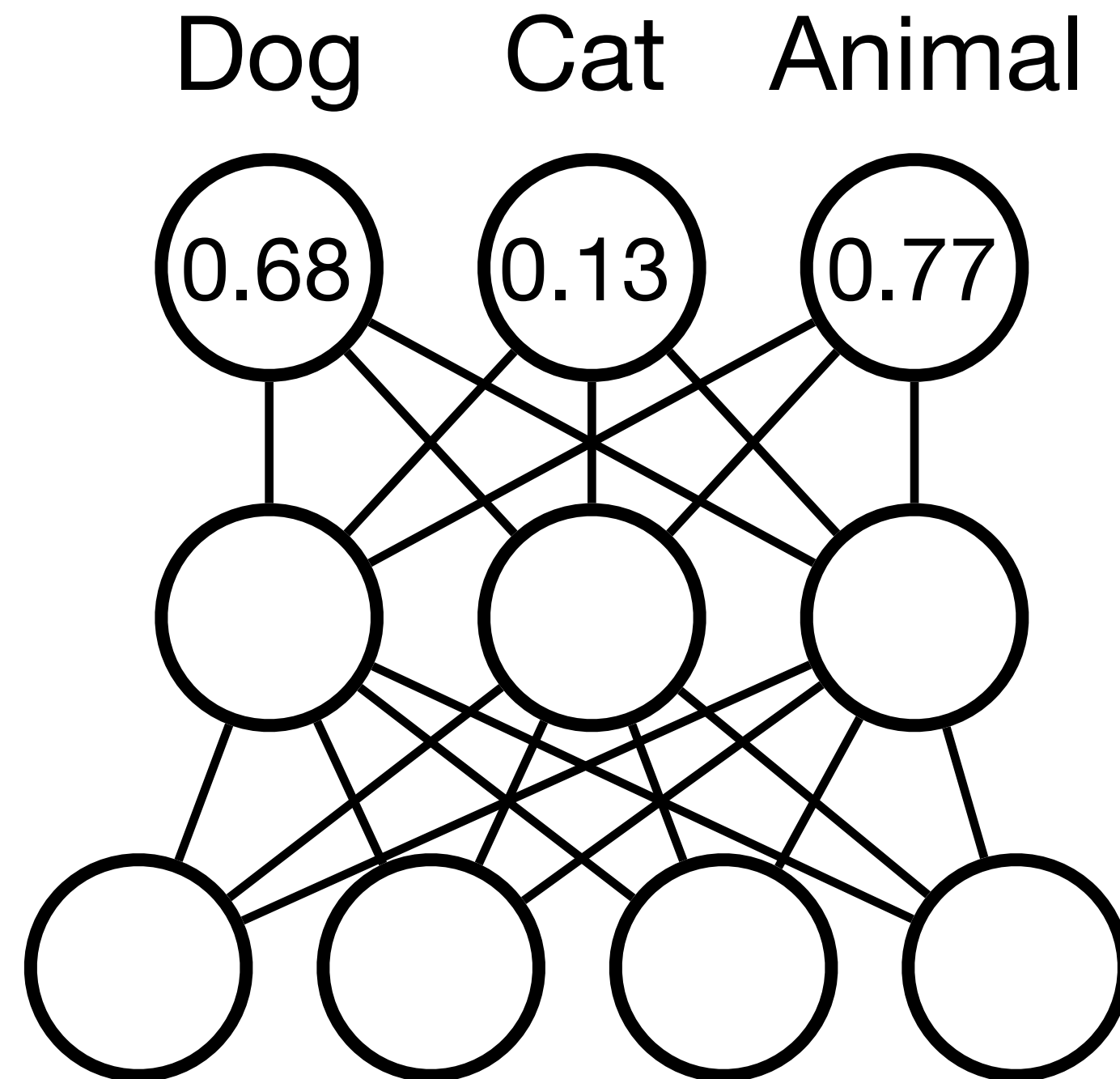


Integrate the requirements into the loss function!

In this scenario, we would like to inform our network that:

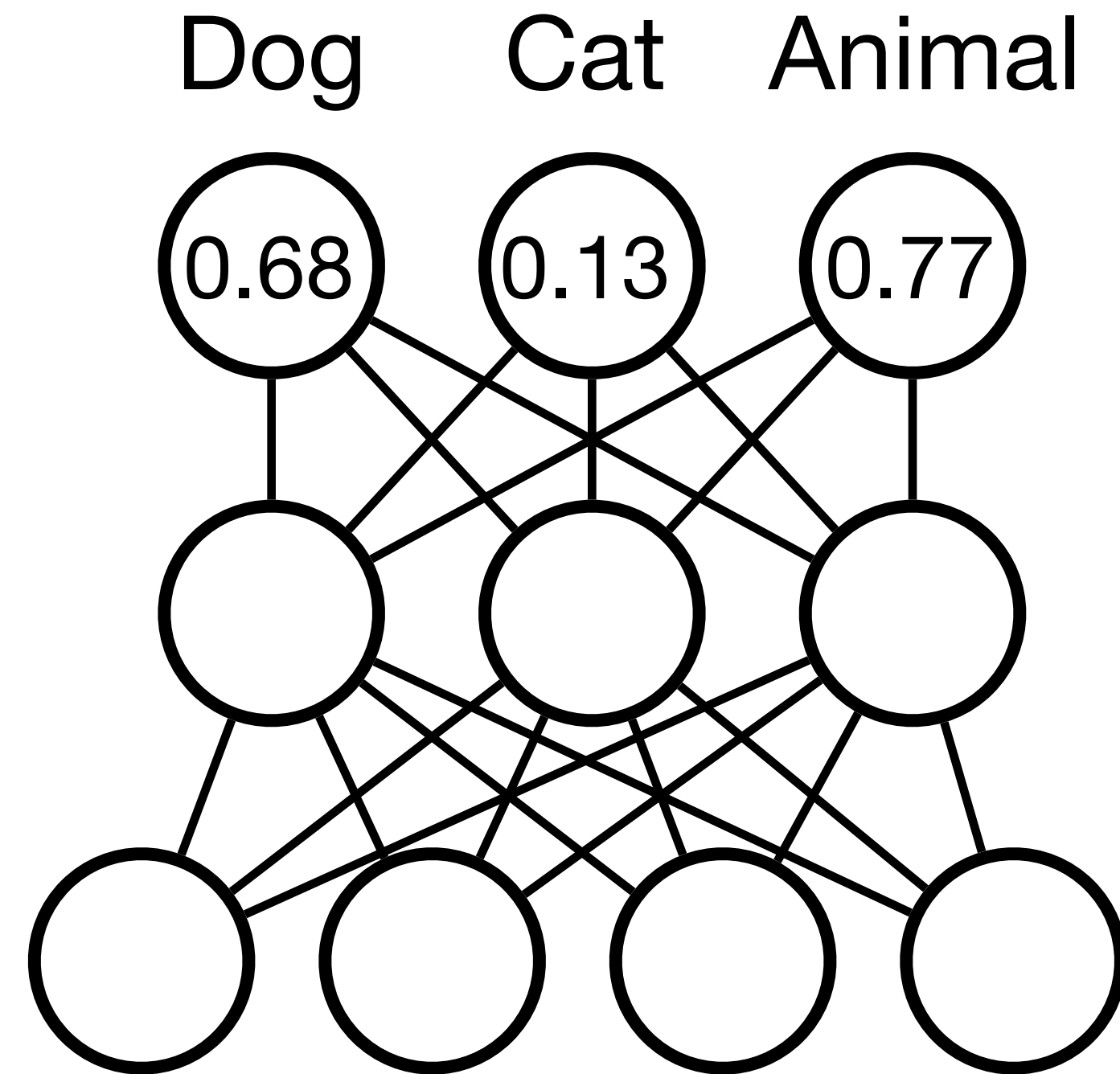
Dog $\rightarrow$ Animal Cat $\rightarrow$ Animal

We can use t-norms based losses!





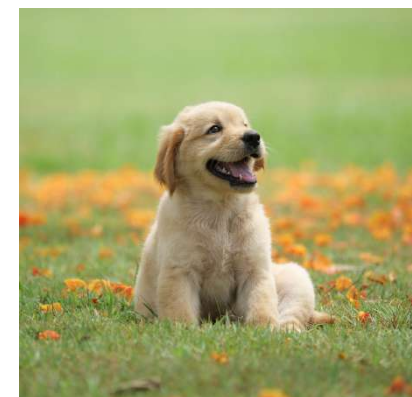
Integrate the requirements into the loss function!



In this scenario, we would like to inform our network that:

Dog $\rightarrow$ Animal Cat $\rightarrow$ Animal

We can use t-norms based losses!



AIJ Prominent Paper Award
2026!! 🏆

Fuzzy Logic inspired loss



Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

It is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ modelling the conjunction with properties:

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

It is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ modelling the conjunction with properties:

- **Commutativity:** $T(a, b) = T(b, a)$

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

It is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ modelling the conjunction with properties:

- **Commutativity:** $T(a, b) = T(b, a)$
- **Monotonicity:** if $a \leq c, b \leq d$, then $T(a, b) \leq T(c, d)$

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

It is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ modelling the conjunction with properties:

- **Commutativity:** $T(a, b) = T(b, a)$
- **Monotonicity:** if $a \leq c, b \leq d$, then $T(a, b) \leq T(c, d)$
- **Associativity:** $T(a, T(b, c)) = T(T(a, b), c)$

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

Intuition: we transform logical formulas into a differentiable form using t-norms

What is a t-norm?

It is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ modelling the conjunction with properties:

- **Commutativity:** $T(a, b) = T(b, a)$
- **Monotonicity:** if $a \leq c, b \leq d$, then $T(a, b) \leq T(c, d)$
- **Associativity:** $T(a, T(b, c)) = T(T(a, b), c)$
- **Identity element is 1:** $T(a, 1) = a$

Operation	Minimum/Gödel
$x \wedge y$	$\min(x, y)$
$\neg x$	$1 - x$
$x \vee y$	$\max(x, y)$



Fuzzy Logic inspired loss

There exist many different t-norms! The most used are:

Operation	Minimum/Gödel	Product	Lukasiewicz
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \wedge y$	$\min(x, y)$	$x \cdot y$	$\max(0, x + y - 1)$
$x \vee y$	$\max(x, y)$	$x + y - x \cdot y$	$\min(1, x + y)$



Fuzzy Logic inspired loss

There exist many different t-norms! The most used are:

Operation	Minimum/Gödel	Product	Lukasiewicz
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \wedge y$	$\min(x, y)$	$x \cdot y$	$\max(0, x + y - 1)$
$x \vee y$	$\max(x, y)$	$x + y - x \cdot y$	$\min(1, x + y)$

Idempotent ($T(a, a) = a$)
& Continuous





Fuzzy Logic inspired loss

There exist many different t-norms! The most used are:

Operation	Minimum/Gödel	Product	Lukasiewicz
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \wedge y$	$\min(x, y)$	$x \cdot y$	$\max(0, x + y - 1)$
$x \vee y$	$\max(x, y)$	$x + y - x \cdot y$	$\min(1, x + y)$

Idempotent ($T(a, a) = a$)
& Continuous

Strict (continuous & strictly
monotone) & Archimedean



Fuzzy Logic inspired loss

There exist many different t-norms! The most used are:

Operation	Minimum/Gödel	Product	Lukasiewicz
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \wedge y$	$\min(x, y)$	$x \cdot y$	$\max(0, x + y - 1)$
$x \vee y$	$\max(x, y)$	$x + y - x \cdot y$	$\min(1, x + y)$

Idempotent ($T(a, a) = a$)
& Continuous

Strict (continuous & strictly
monotone) & Archimedean

Continuous &
Archimedean

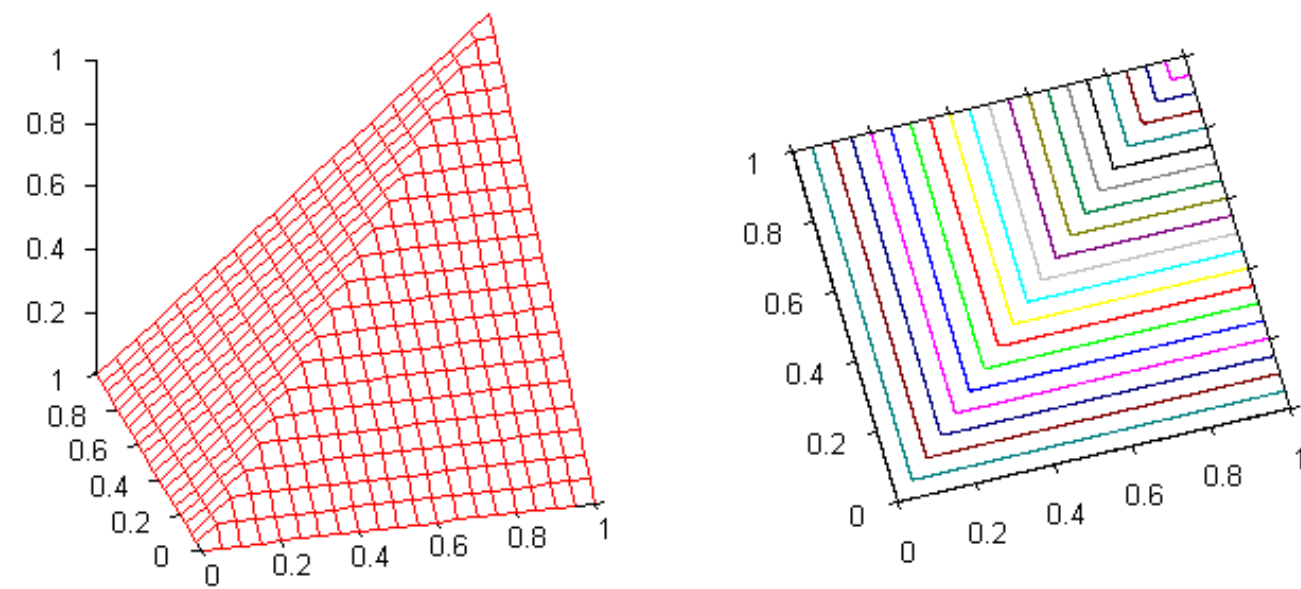


Why are these differences important?

Different t-norms lead to differently shaped loss functions!

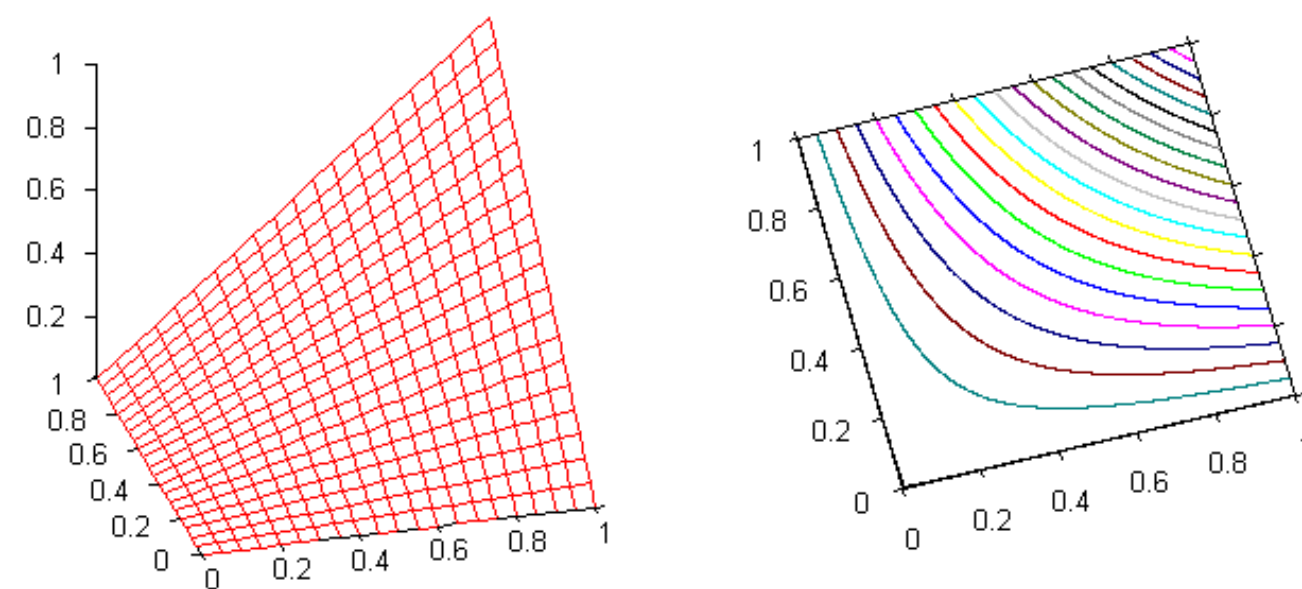
Gödel t-norm

$$T(x, y) = \min(x, y)$$



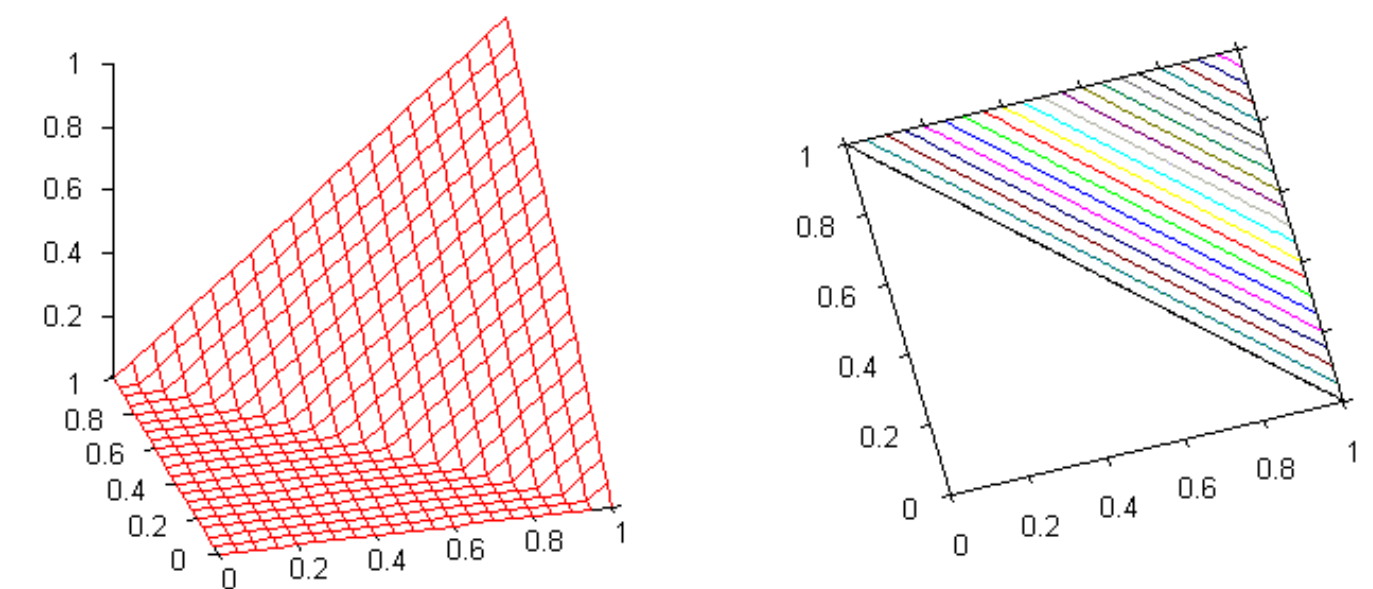
Product t-norm

$$T(x, y) = xy$$



Lukasiewicz t-norm

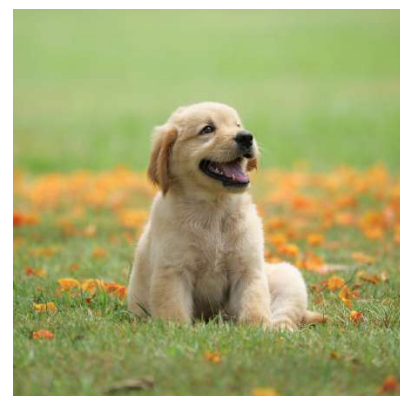
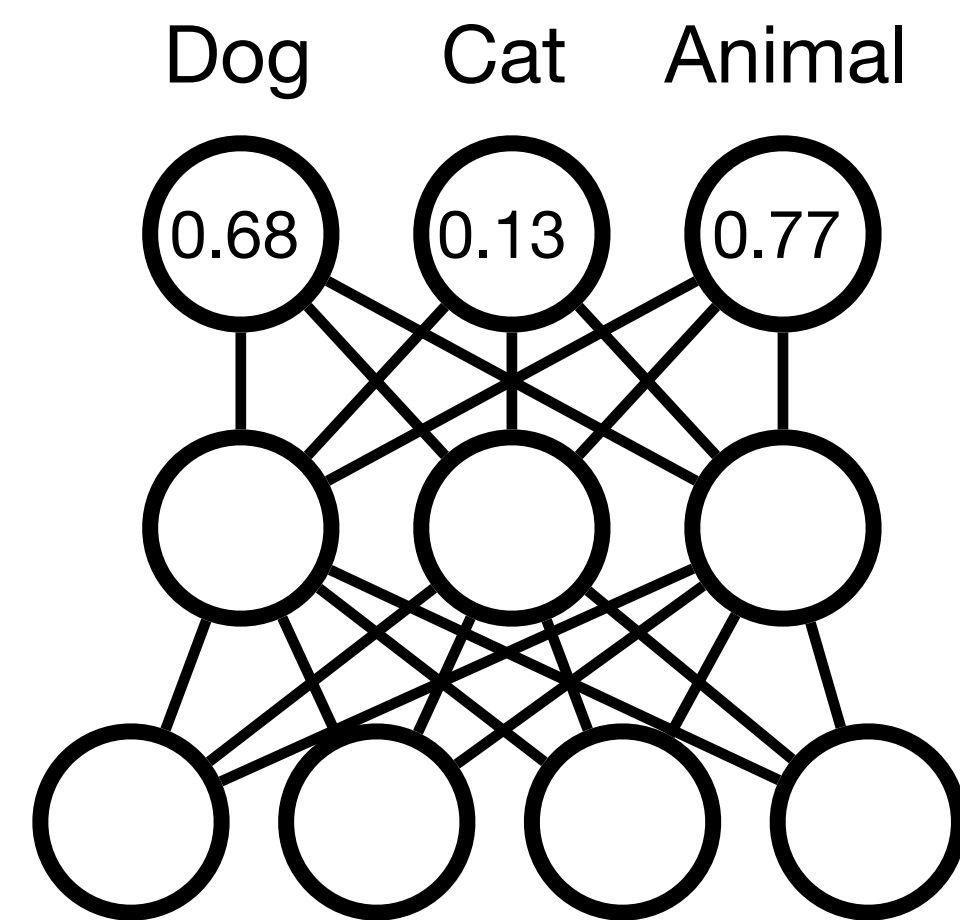
$$T(x, y) = \max(0, x + y - 1)$$





Let's go back to our example...

Dog $\rightarrow$ Animal, Cat $\rightarrow$ Animal



By using **product t-norm**, then we can model

$x \rightarrow y$ as $\neg x \vee y$ and hence $(1 - x) + y - y(1 - x)$

And thus:

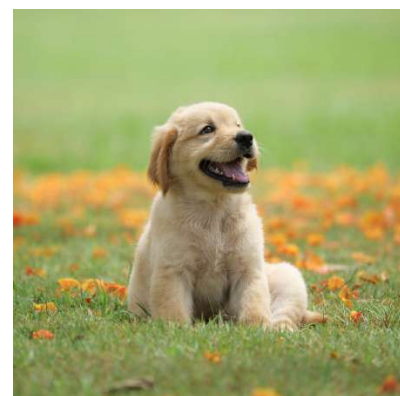
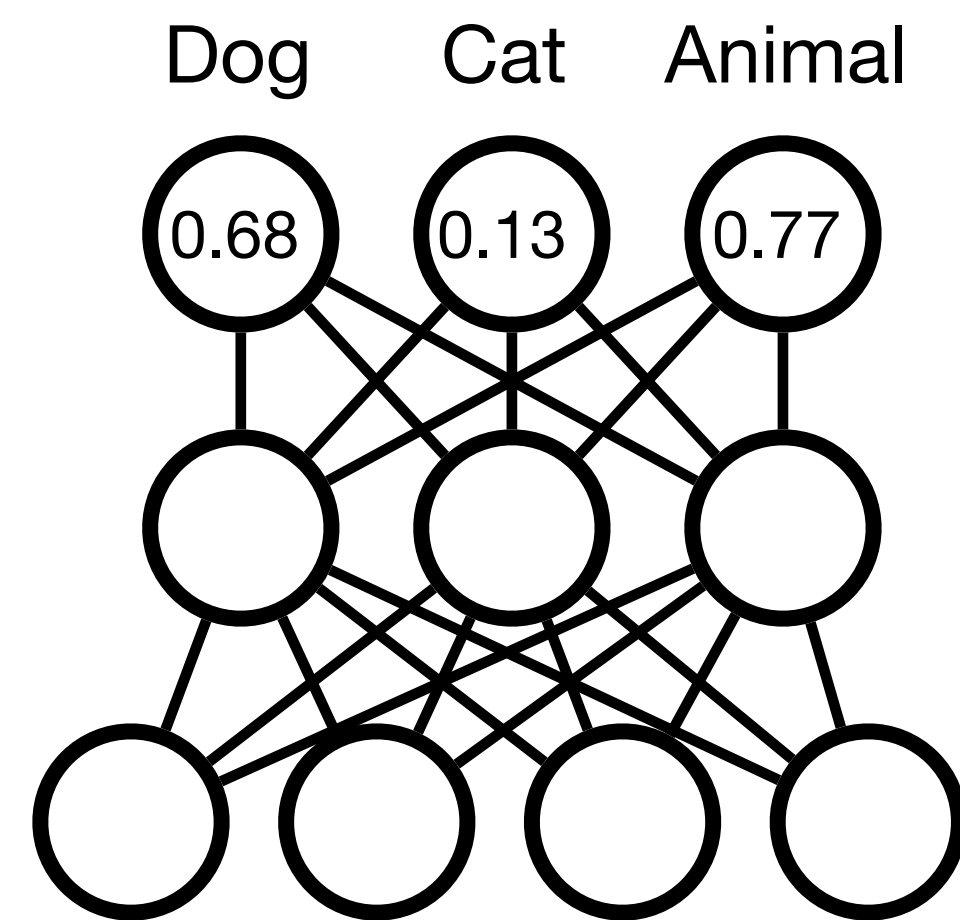
$$\mathcal{L}_1 = -((1 - h_{Dog}) + h_{Animal} - (1 - h_{Dog})h_{Animal}))$$

$$\mathcal{L}_2 = -((1 - h_{Cat}) + h_{Animal} - (1 - h_{Cat})h_{Animal}))$$



Let's go back to our example...

Dog $\rightarrow$ Animal, Cat $\rightarrow$ Animal



By using **product t-norm**, then we can model

$x \rightarrow y$ as $\neg x \vee y$ and hence $(1 - x) + y - y(1 - x)$

And thus:

$$\mathcal{L}_1 = -((1 - h_{Dog}) + h_{Animal} - (1 - h_{Dog})h_{Animal}))$$

$$\mathcal{L}_2 = -((1 - h_{Cat}) + h_{Animal} - (1 - h_{Cat})h_{Animal}))$$

The final loss will be the sum of the loss for each class
+ the loss for each constraint

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A + \mathcal{L}_1 + \mathcal{L}_2$$



Pros and Cons

Pros

Cons



Pros and Cons

Pros

- Intuitive to understand

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A + \mathcal{L}_1 + \mathcal{L}_2$$

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \mathcal{L}_1 + \mathcal{L}_2$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \boxed{\mathcal{L}_1 + \mathcal{L}_2}$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

$$\mathcal{L}_1 = -((1 - h_D) + h_A - (1 - h_D)h_A))$$

Cons



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \boxed{\mathcal{L}_1 + \mathcal{L}_2}$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

$$\mathcal{L}_1 = -((1 - h_D) + h_A - (1 - h_D)h_A))$$

Cons

- Not clear which t-norm to use



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \boxed{\mathcal{L}_1 + \mathcal{L}_2}$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

$$\mathcal{L}_1 = -((1 - h_D) + h_A - (1 - h_D)h_A))$$

Cons

- Not clear which t-norm to use
- No guarantee that the constraints will be satisfied



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \boxed{\mathcal{L}_1 + \mathcal{L}_2}$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

$$\mathcal{L}_1 = -((1 - h_D) + h_A - (1 - h_D)h_A)$$

Cons

- Not clear which t-norm to use
- No guarantee that the constraints will be satisfied
- Syntax dependent



Pros and Cons

Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

$$\mathcal{L}_{\text{Final}} = \boxed{\mathcal{L}_D + \mathcal{L}_C + \mathcal{L}_A} + \boxed{\mathcal{L}_1 + \mathcal{L}_2}$$

$$\mathcal{L}_D = -y_D \ln(h_D) - (1 - y_D) \ln(1 - h_D)$$

$$\mathcal{L}_1 = -((1 - h_D) + h_A - (1 - h_D)h_A))$$

Cons

- Not clear which t-norm to use
- No guarantee that the constraints will be satisfied
- Syntax dependent

Consider the Product t-norm:

$$T(a, a) \neq T(a, T(a, a))$$

How can we make it syntax independent?





How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then we can create a loss inspired by probabilistic logic:



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then we can create a loss inspired by probabilistic logic:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then we can create a loss inspired by probabilistic logic:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

Intuition: the loss represents the “surprise” of obtaining an assignment that satisfies the constraint.



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then we can create a loss inspired by probabilistic logic:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

Intuition: the loss represents the “surprise” of obtaining an assignment that satisfies the constraint.



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then the loss is defined as:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

Summing over all the
instantiations that satisfy α



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then the loss is defined as:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

Summing over all the
instantiations that satisfy α

Compute the probability of that instantiation
according to the neural network h



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then the loss is defined as:

$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$



How can we make it syntax independent?

Let α be a propositional logic formula over the set variables $\mathcal{Y} = \{Y_1, \dots, Y_n\}$.
Let $\mathbf{y}$ indicate the joint instantiation of all variables in $\mathcal{Y}$.

Given a neural network h and a propositional formula α ,
then the loss is defined as:

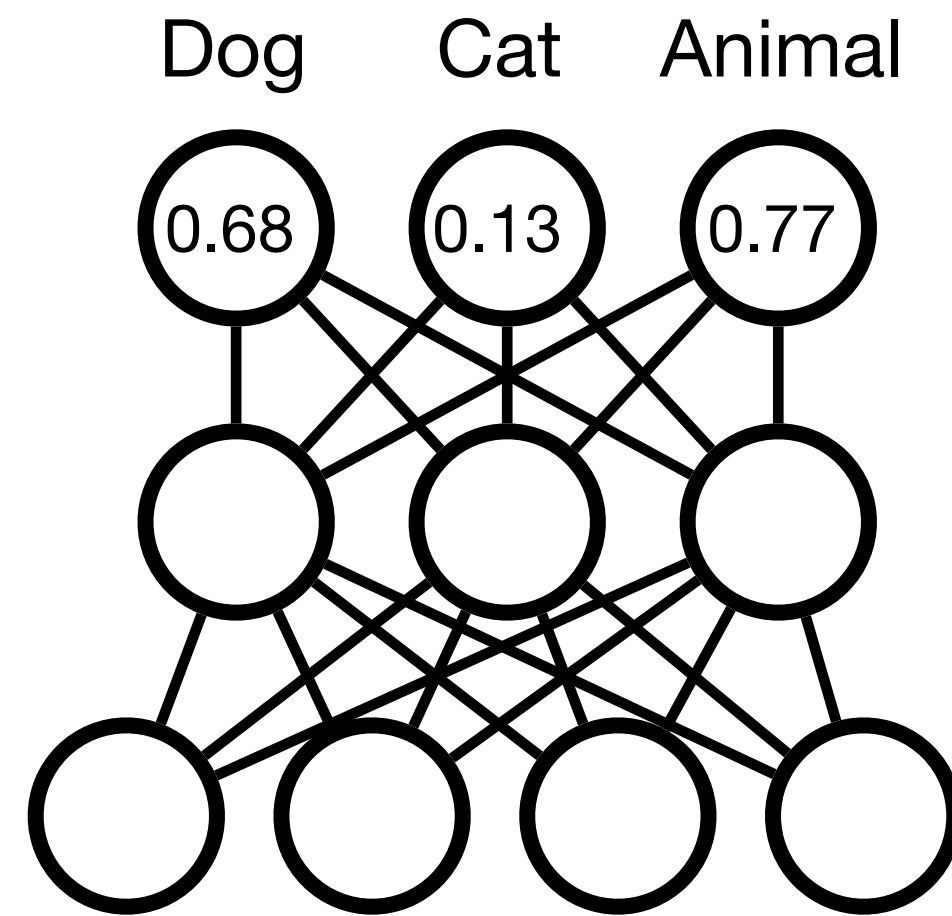
$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

probability of generating an instantiation that satisfies α , when sampling values according to h



Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$

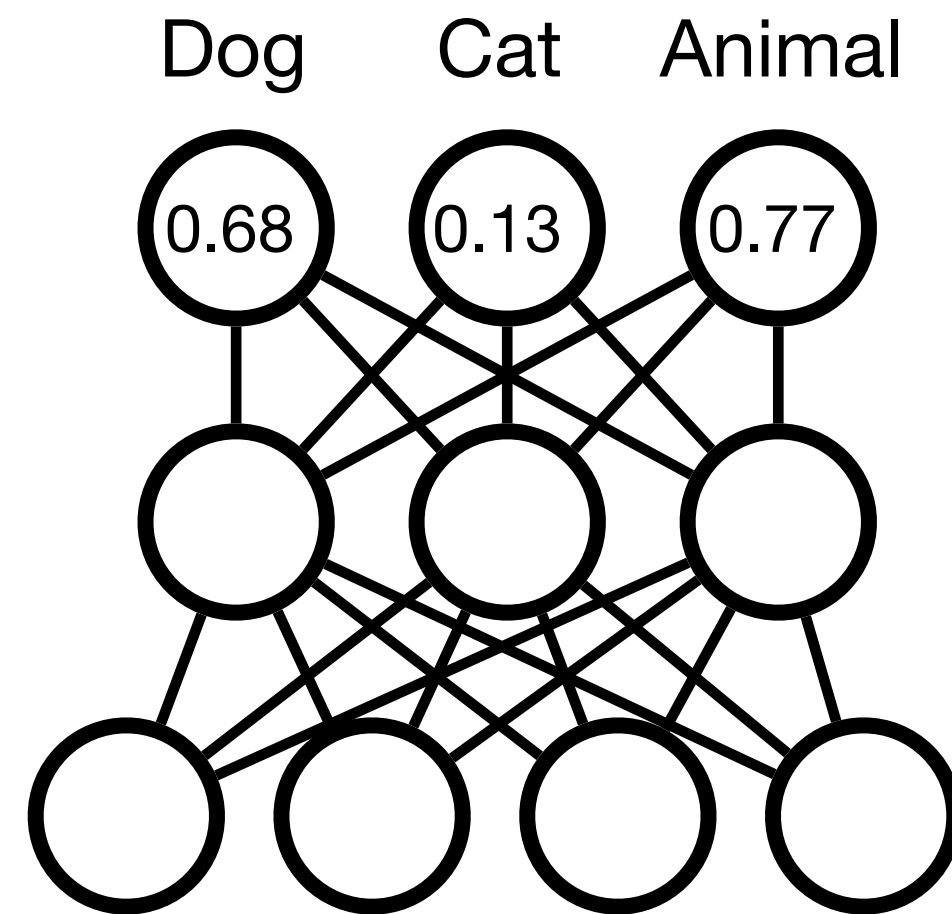




Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$

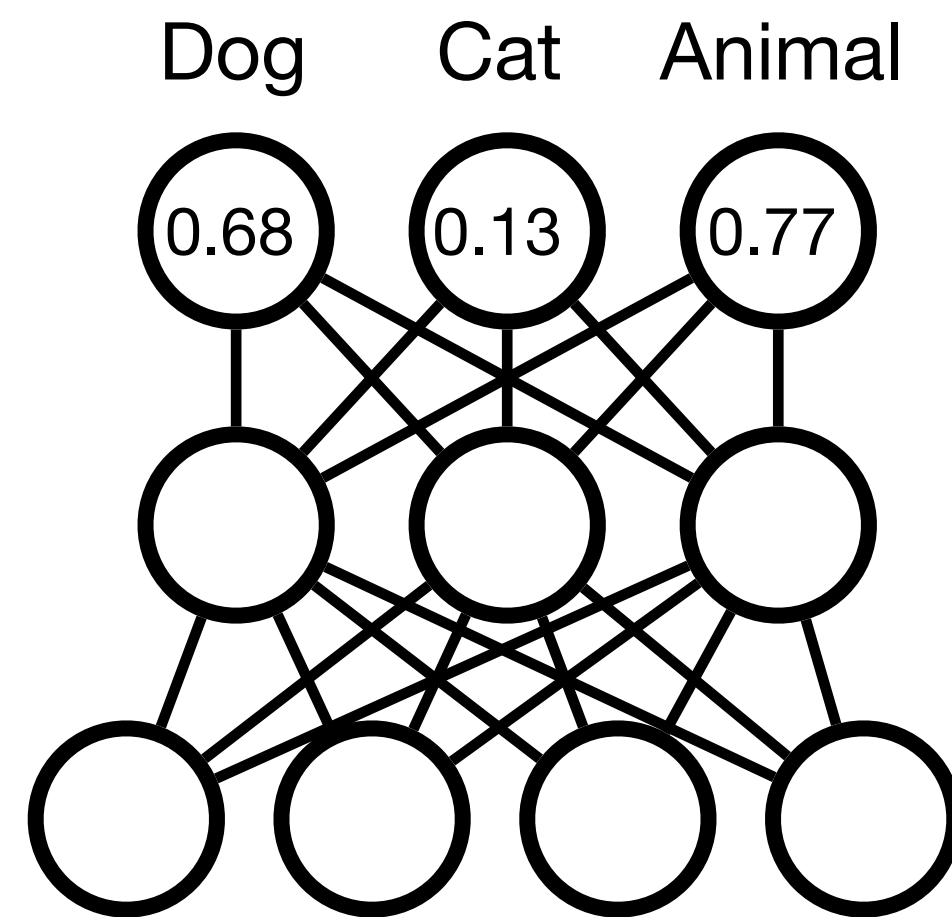
$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$





Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



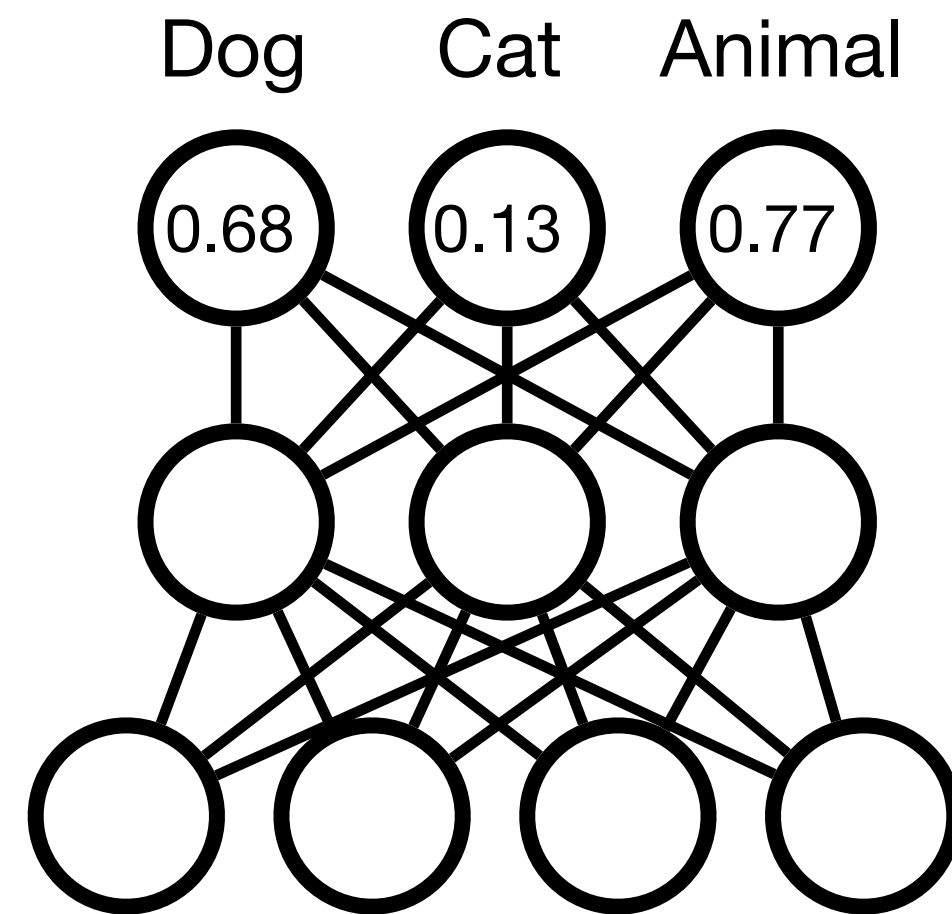
$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

$$\{\mathbf{y} \mid \mathbf{y} \models \alpha\} = \{(\text{Dog} = 0, \text{Cat} = 0, \text{Animal} = 0), \\ (\text{Dog} = 0, \text{Cat} = 0, \text{Animal} = 1), \\ (\text{Dog} = 1, \text{Cat} = 0, \text{Animal} = 1), \\ (\text{Dog} = 0, \text{Cat} = 1, \text{Animal} = 1), \\ (\text{Dog} = 1, \text{Cat} = 1, \text{Animal} = 1)\}$$



Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



$$\mathcal{L} \propto -\log \sum_{\mathbf{y} \models \alpha} \prod_{i: \mathbf{y} \models Y_i} h_i \prod_{i: \mathbf{y} \models \neg Y_i} (1 - h_i)$$

$$\{\mathbf{y} \mid \mathbf{y} \models \alpha\} = \{(\text{Dog} = 0, \text{Cat} = 0, \text{Animal} = 0), \\ (\text{Dog} = 0, \text{Cat} = 0, \text{Animal} = 1), \\ (\text{Dog} = 1, \text{Cat} = 0, \text{Animal} = 1), \\ (\text{Dog} = 0, \text{Cat} = 1, \text{Animal} = 1), \\ (\text{Dog} = 1, \text{Cat} = 1, \text{Animal} = 1)\}$$

$$\mathcal{L} \propto -\log \left((1 - h_D)(1 - h_C)(1 - h_A) + \right. \\ \left. (1 - h_D)(1 - h_C)h_A + (1 - h_D)h_C h_A + \right. \\ \left. h_D(1 - h_C)h_A + h_D h_C h_A \right)$$

Pros and Cons



Pros

Cons

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand

Cons

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute

Cons

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

- Intuitive to understand

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

- Intuitive to understand
- Syntax independent

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

- Intuitive to understand
- Syntax independent
- Can be used to train on unlabelled data

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

- Intuitive to understand
- Syntax independent
- Can be used to train on unlabelled data

- No guarantee that constraints will be satisfied

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- Efficient to compute
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- Syntax dependent

Fuzzy

- Intuitive to understand
- Syntax independent
- Can be used to train on unlabelled data

- No guarantee that constraints will be satisfied
- Not really efficient to compute

Probabilistic

Pros and Cons



Pros

- Intuitive to understand
- ***Efficient to compute***
- Can be used to train on unlabelled data

Cons

- Not clear which t-norm to use
- No guarantee that constraints will be satisfied
- ***Syntax dependent***

Fuzzy

- Intuitive to understand
- ***Syntax independent***
- Can be used to train on unlabelled data

- No guarantee that constraints will be satisfied
- ***Not really efficient to compute***

Probabilistic



Example

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.

We want to calculate the value of the loss function when using:

- The t-norm based loss with Gödel t-norm
- The probabilistic logic inspired loss



Solution

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.





Solution

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.

Gödel t-norm based loss:

$$\begin{aligned}\mathcal{L} &= -\max(h_C, \min(h_A, h_B)) \\ &= -\max(0.6, \min(0.3, 0.7)) \\ &= -0.6\end{aligned}$$



Solution

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.

Gödel t-norm based loss:

$$\begin{aligned}\mathcal{L} &= -\max(h_C, \min(h_A, h_B)) \\ &= -\max(0.6, \min(0.3, 0.7)) \\ &= -0.6\end{aligned}$$

Probabilistic logic inspired loss:



Solution

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.

Gödel t-norm based loss:

$$\begin{aligned}\mathcal{L} &= -\max(h_C, \min(h_A, h_B)) \\ &= -\max(0.6, \min(0.3, 0.7)) \\ &= -0.6\end{aligned}$$

Probabilistic logic inspired loss:

$$\begin{aligned}\{\mathbf{y} \mid \mathbf{y} \models \alpha\} &= \{(A = 1, B = 1, C = 0), (A = 1, B = 1, C = 1), \\ &\quad (A = 0, B = 1, C = 1), (A = 1, B = 0, C = 1), \\ &\quad (A = 0, B = 0, C = 1)\}\end{aligned}$$



Solution

Suppose we have a multi-label classification problem with classes $\{A, B, C\}$.

Let our requirement be $\alpha = (A \wedge B) \vee C$.

Given an input $\mathbf{x}$, suppose that the outputs are:

$h_A = 0.3$, $h_B = 0.7$ and $h_C = 0.6$.

Gödel t-norm based loss:

$$\begin{aligned}\mathcal{L} &= -\max(h_C, \min(h_A, h_B)) \\ &= -\max(0.6, \min(0.3, 0.7)) \\ &= -0.6\end{aligned}$$

Probabilistic logic inspired loss:

$$\begin{aligned}\{\mathbf{y} \mid \mathbf{y} \models \alpha\} &= \{(A = 1, B = 1, C = 0), (A = 1, B = 1, C = 1), \\ &\quad (A = 0, B = 1, C = 1), (A = 1, B = 0, C = 1), \\ &\quad (A = 0, B = 0, C = 1)\}\end{aligned}$$

$$\begin{aligned}\mathcal{L} &= -\log\left(h_A h_B (1 - h_C) + h_A h_B h_C + (1 - h_A) h_B h_C \right. \\ &\quad \left. + h_A (1 - h_B) h_C + (1 - h_A) (1 - h_B) h_C\right) \\ &= -\log(0.126 + 0.084 + 0.294 + 0.054 + 0.126) = -\log(0.684)\end{aligned}$$

So far...



So far...



We have seen how to incorporate the requirements into the loss function.



So far...

We have seen how to incorporate the requirements into the loss function.

But... this does not give us any guarantee that the constraints will not be violated.



So far...

We have seen how to incorporate the requirements into the loss function.

But... this does not give us any guarantee that the constraints will not be violated.

Then... can we even get this guarantee?



So far...

We have seen how to incorporate the requirements into the loss function.

But... this does not give us any guarantee that the constraints will not be violated.

Then... can we even get this guarantee?

The word "Yes" is written in a large, elegant, black cursive script. The letters are fluid and connected, with a prominent 'Y' and a long, sweeping 's'. The text is centered within a gray rectangular background.



So far...

We have seen how to incorporate the requirements into the loss function.

But... this does not give us any guarantee that the constraints will not be violated.

Then... can we even get this guarantee?

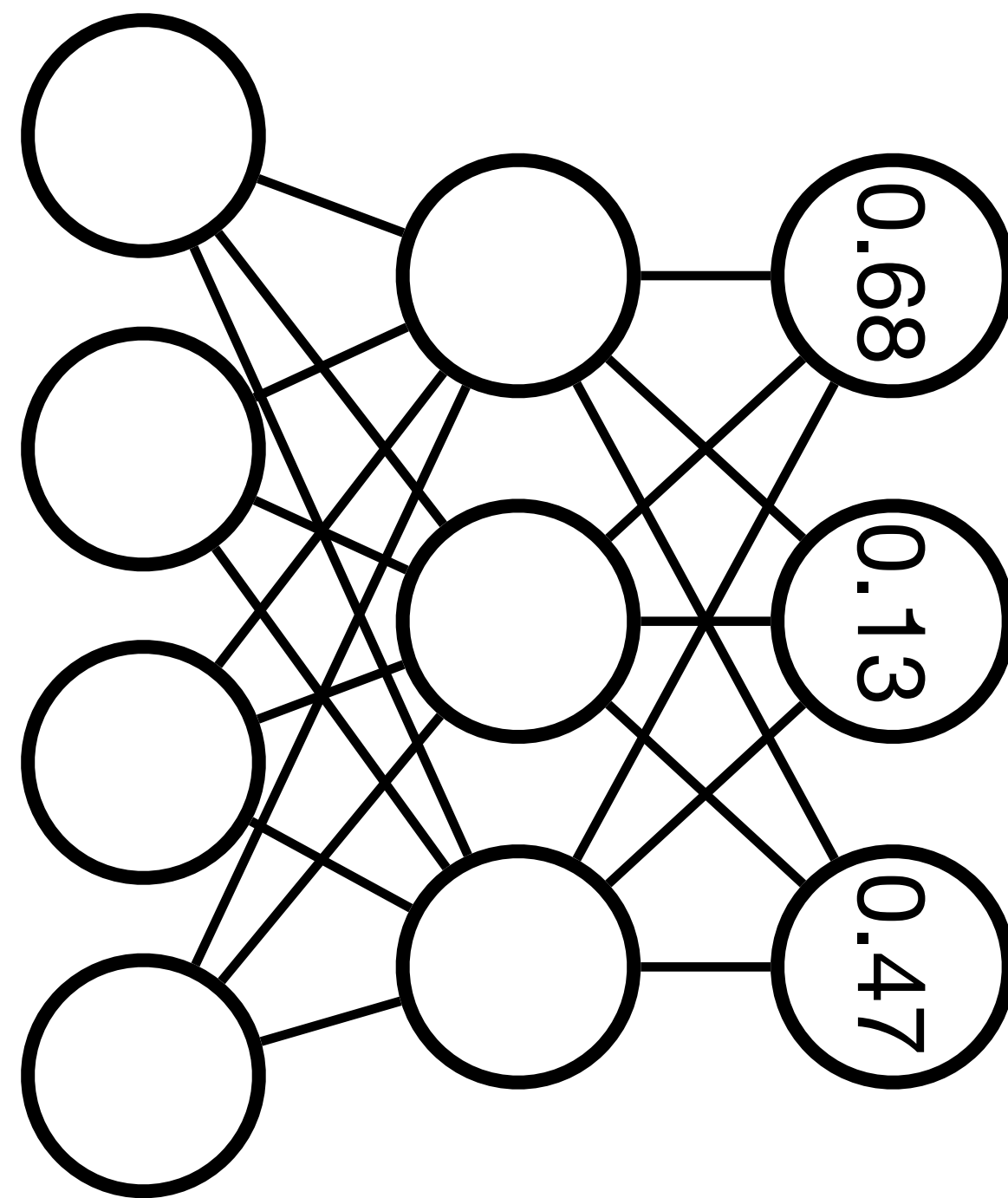


By compiling the requirements into the topology of the network itself!!!

What does it mean to compile into the topology?



$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



Dog

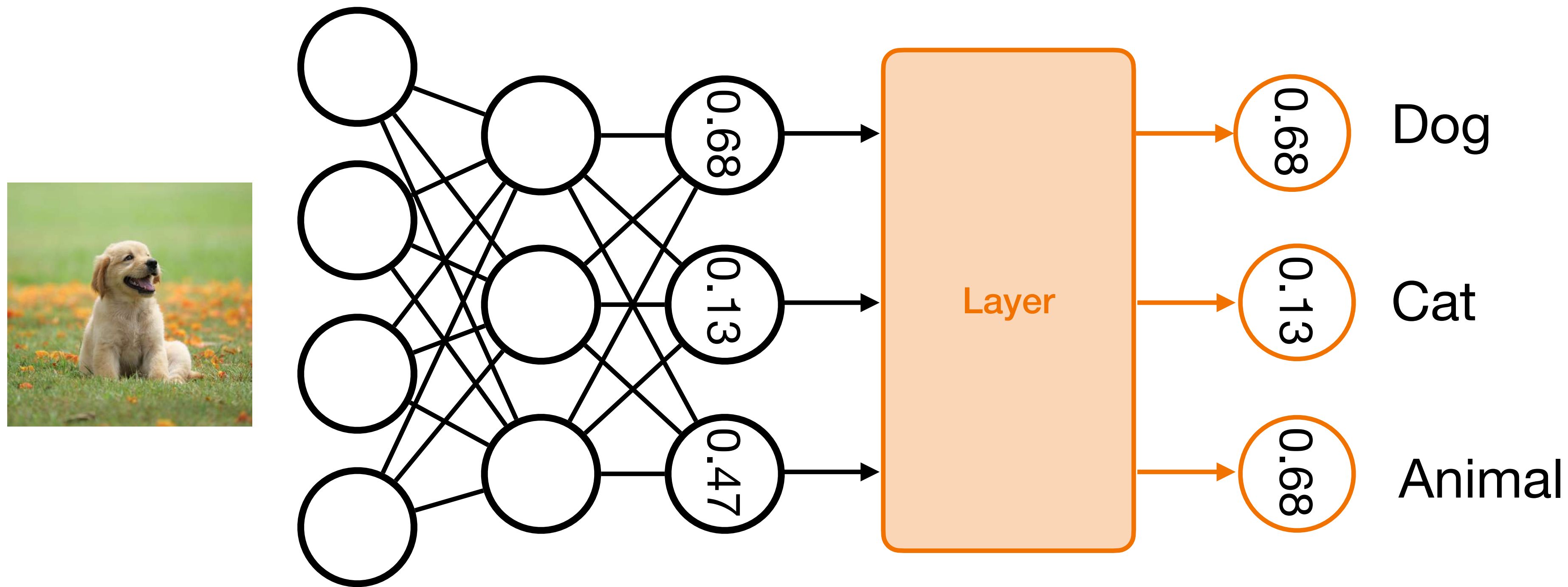
Cat

Animal



What does it mean to compile into the topology?

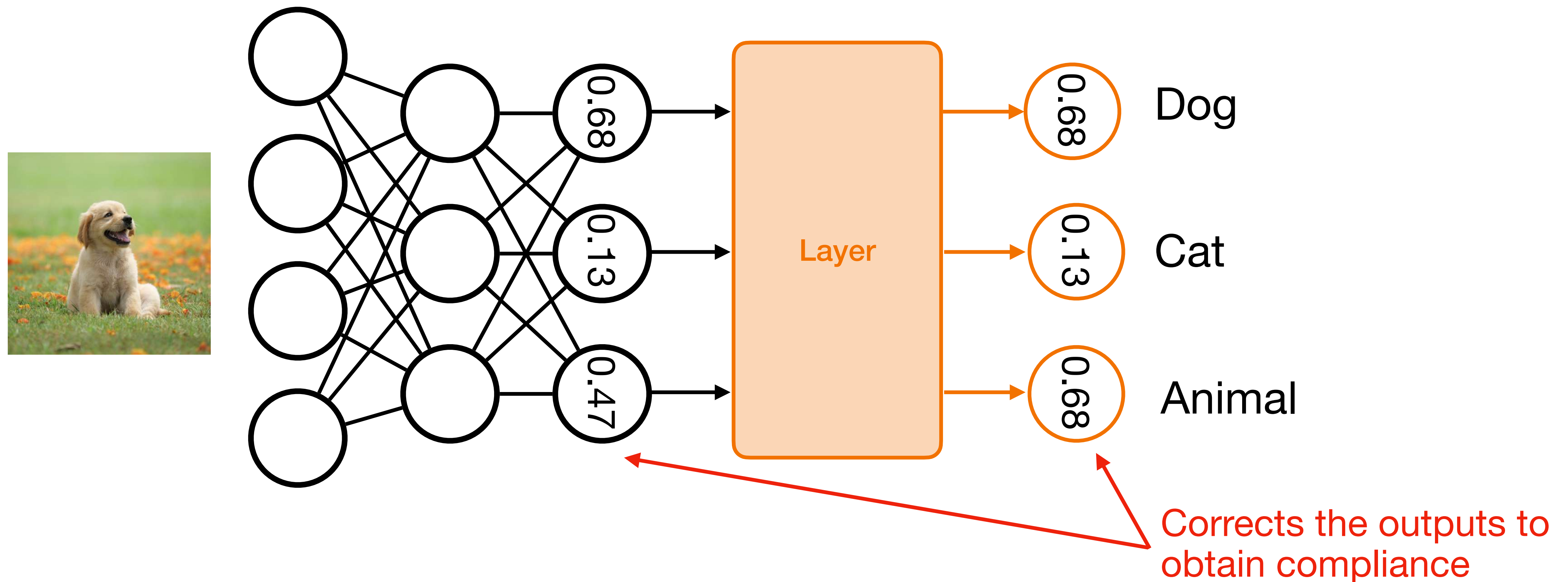
$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$





What does it mean to compile into the topology?

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$





How?

Again... we can take two routes → **Fuzzy**

CCN⁺: A neuro-symbolic framework for deep learning with requirements

Eleonora Giunchiglia^{a,*}, Alex Tatomir^b, Mihaela Cătălina Stoian^b,
Thomas Lukasiewicz^{a,b}

^a *Institute of Logic and Computation, TU Wien, Vienna, Austria*

^b *Department of Computer Science, University of Oxford, Oxford, UK*

↓
Probabilistic

Semantic Probabilistic Layers for Neuro-Symbolic Learning

Kareem Ahmed

Stefano Teso

Kai-Wei Chang

Guy Van den Broeck

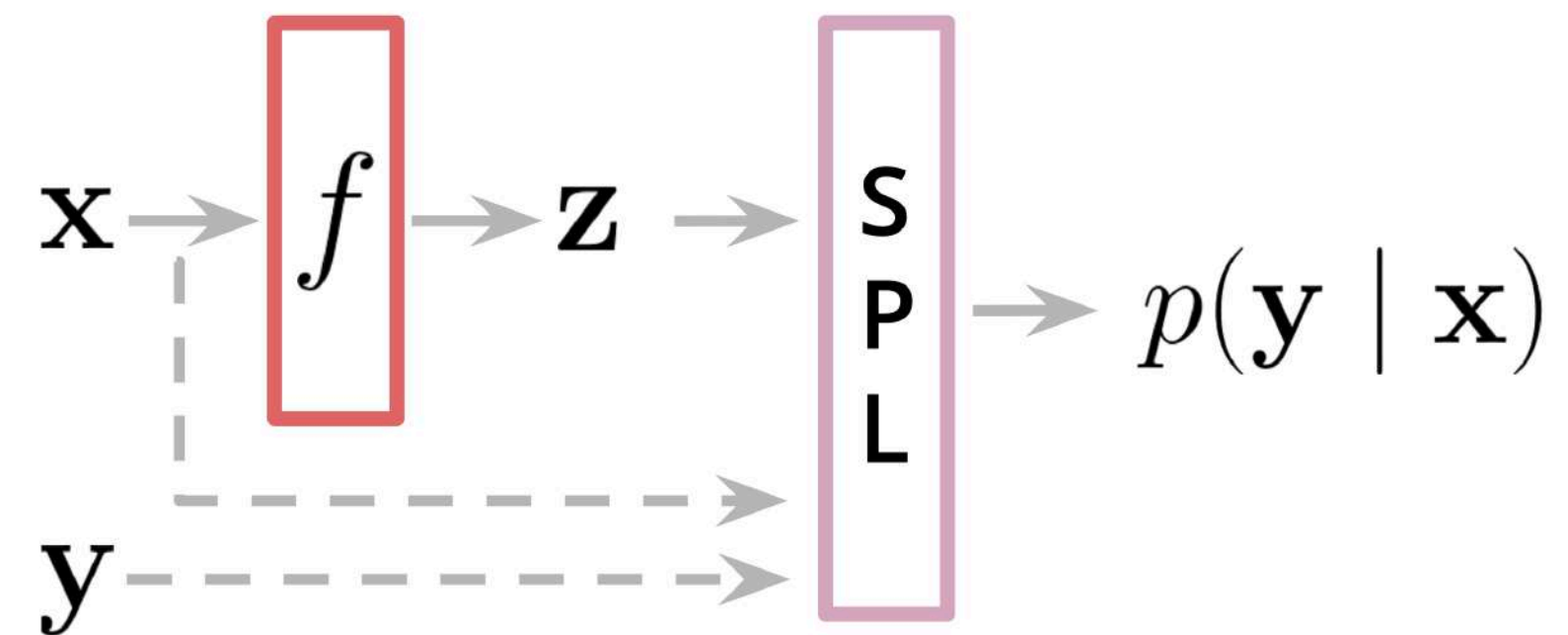
Antonio Vergari

The probabilistic route: Semantic Probabilistic Layer (SPL)



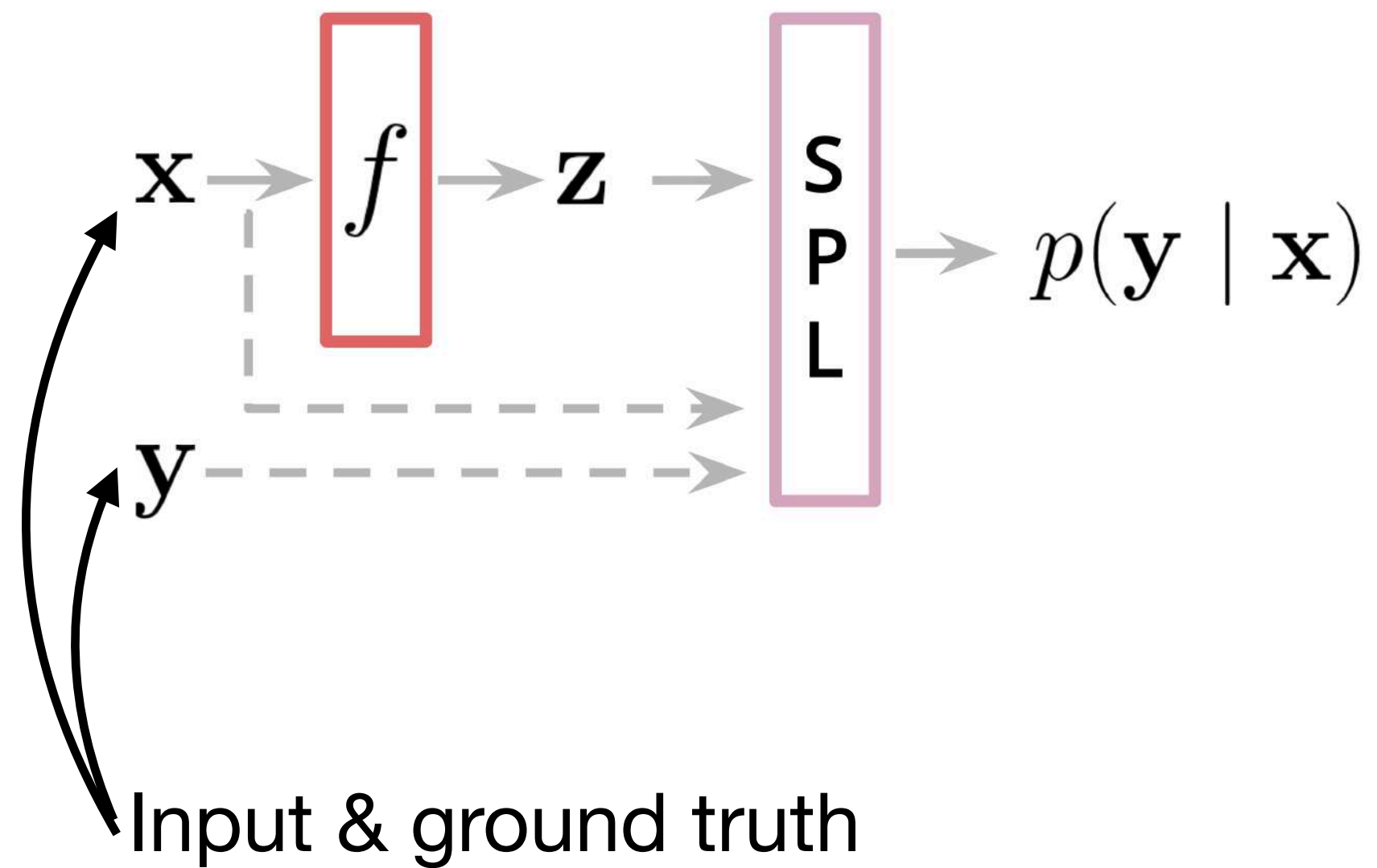


The probabilistic route: Semantic Probabilistic Layer (SPL)



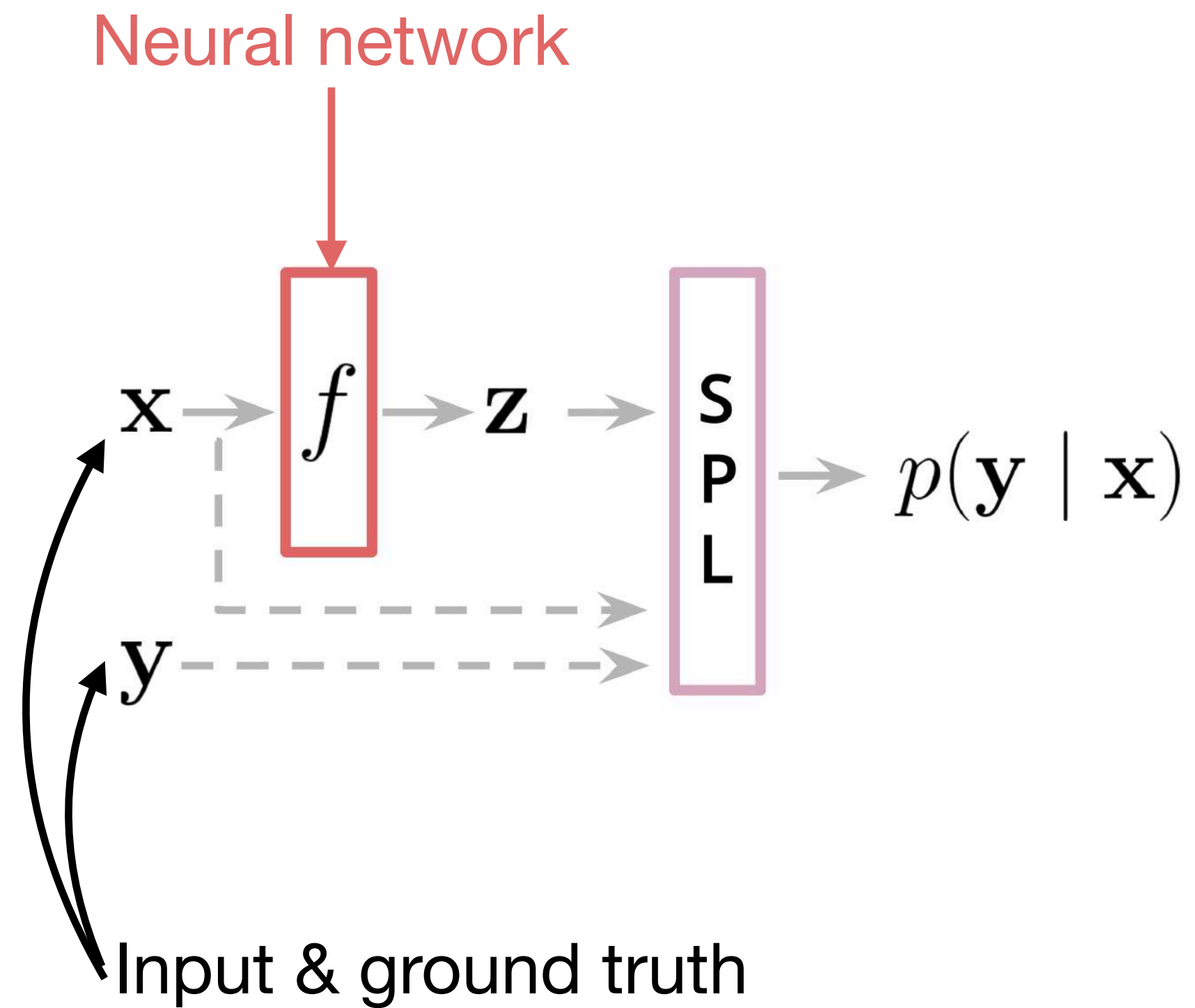


The probabilistic route: Semantic Probabilistic Layer (SPL)



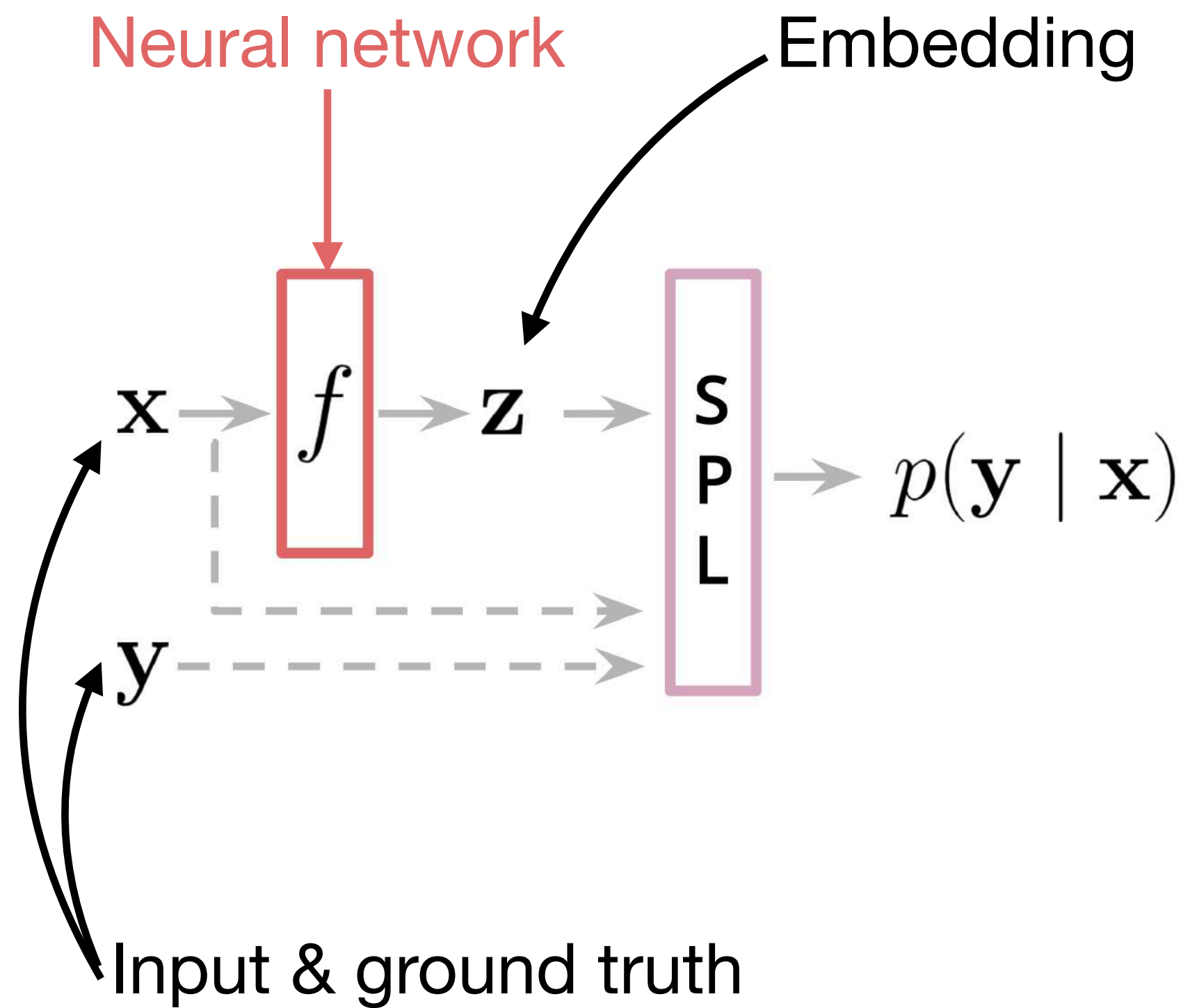


The probabilistic route: Semantic Probabilistic Layer (SPL)



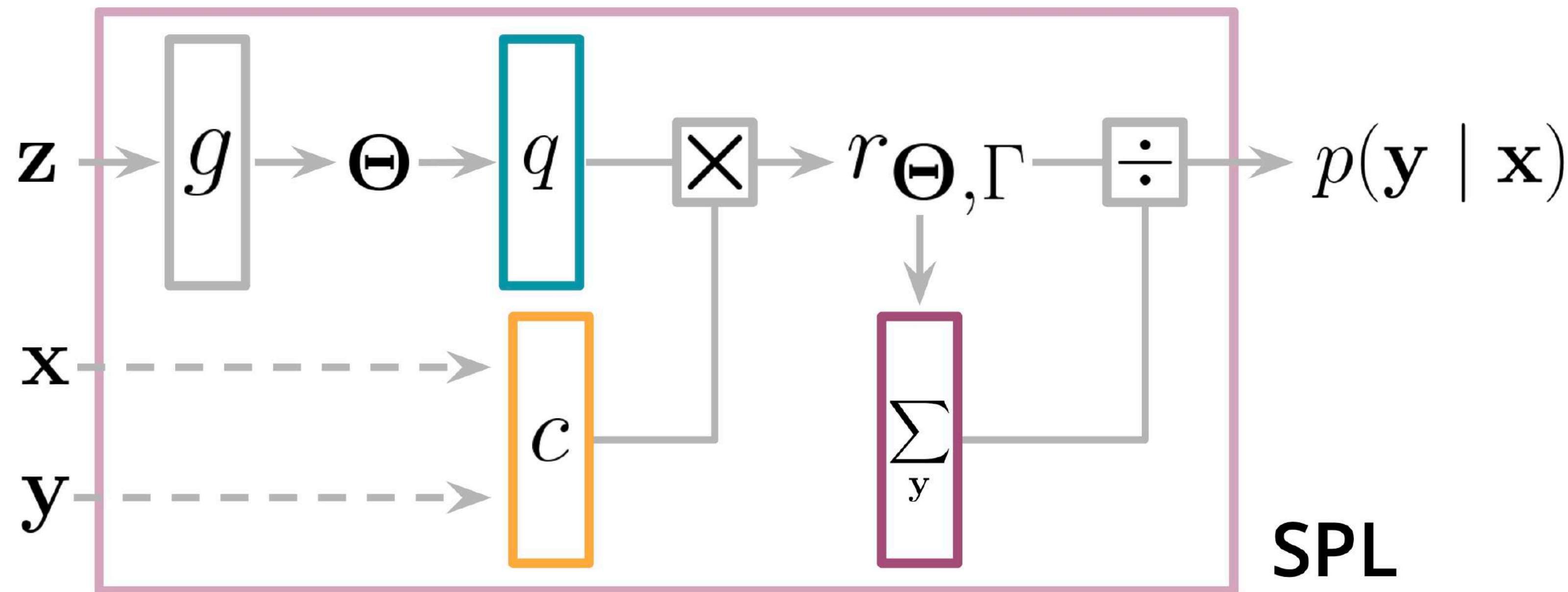
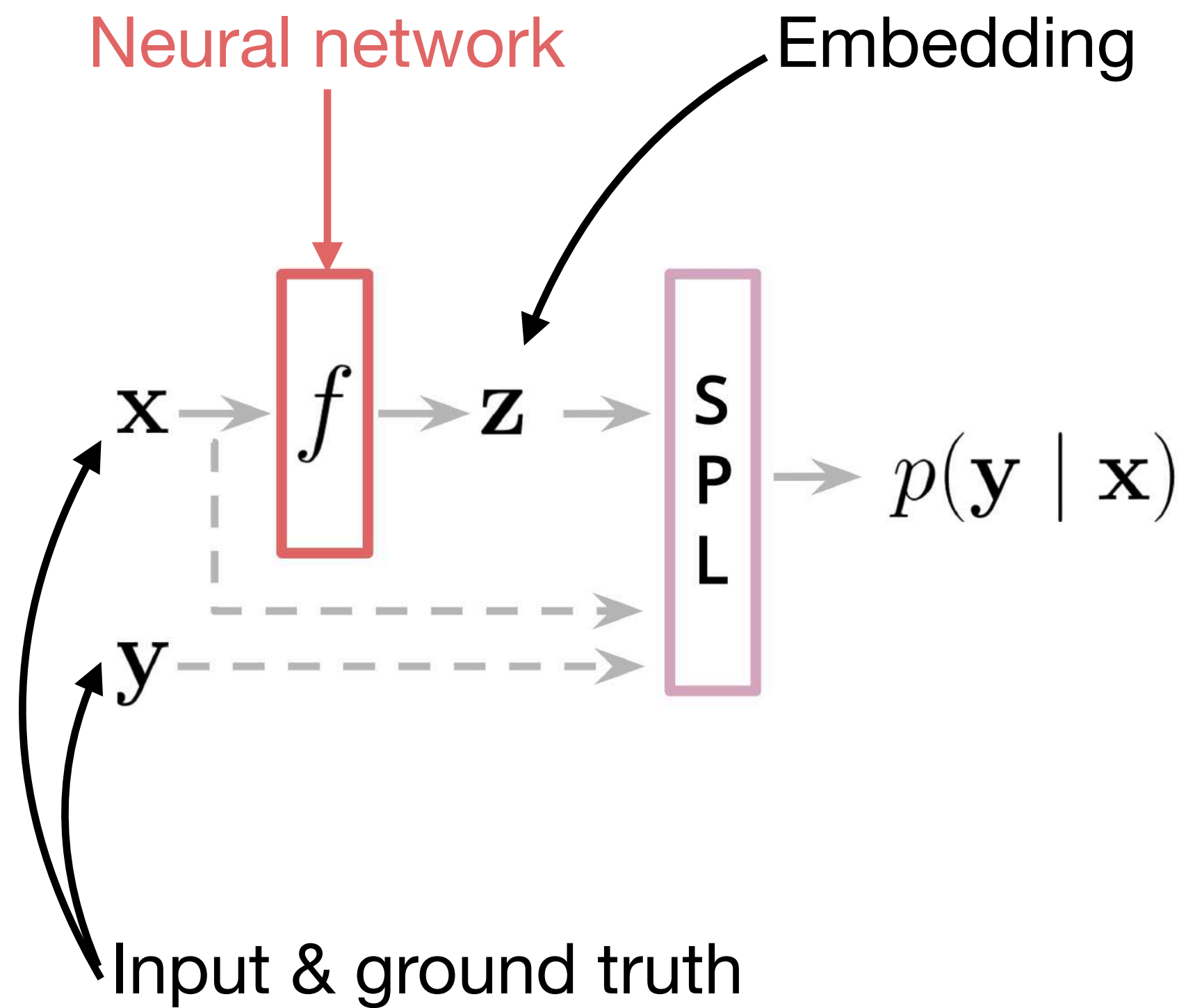


The probabilistic route: Semantic Probabilistic Layer (SPL)



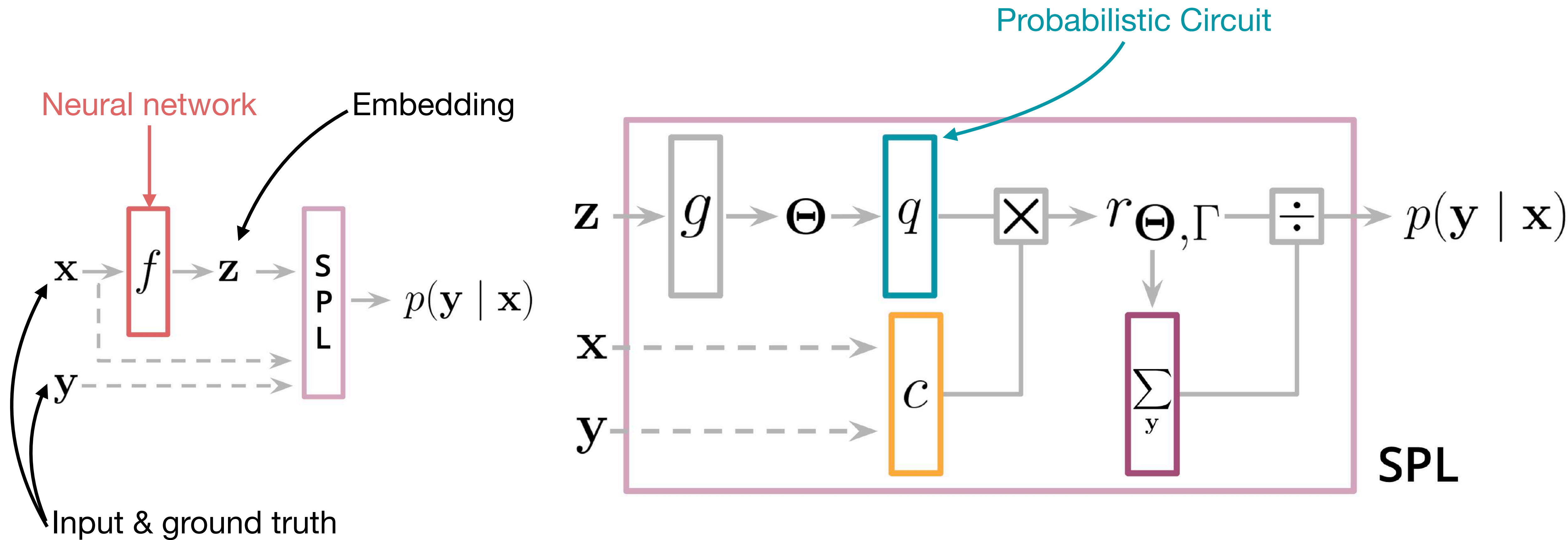


The probabilistic route: Semantic Probabilistic Layer (SPL)



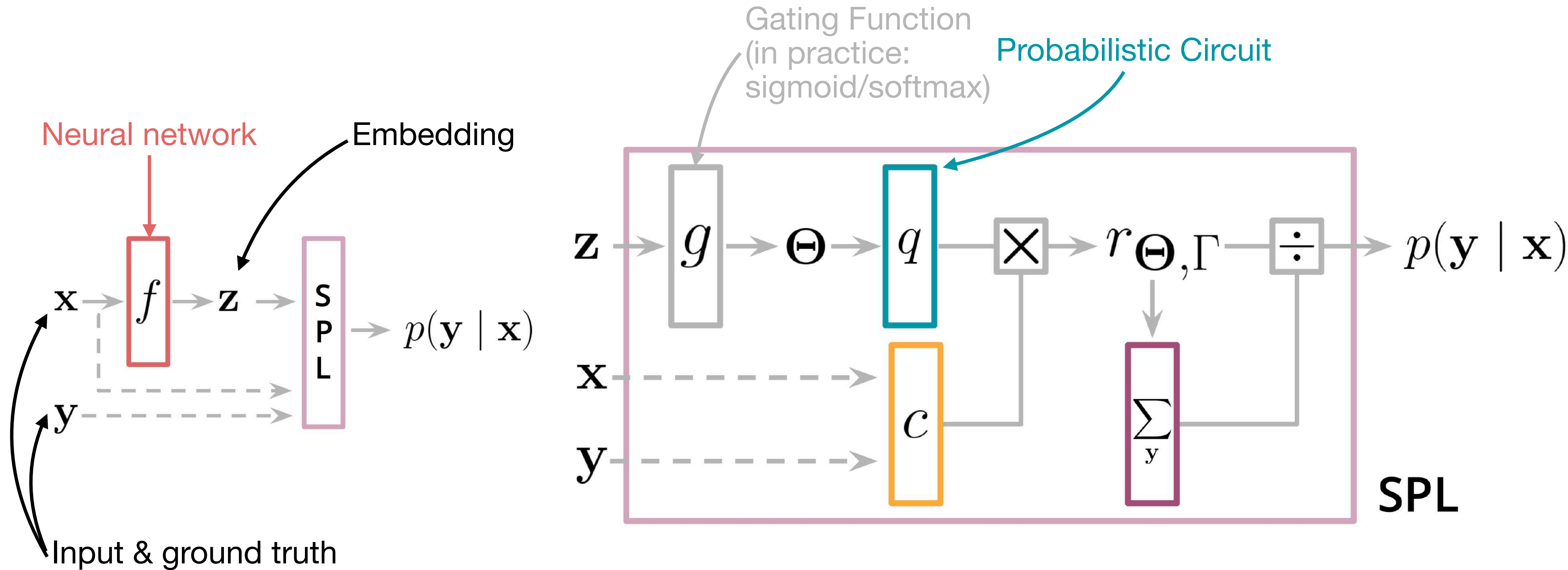


The probabilistic route: Semantic Probabilistic Layer (SPL)



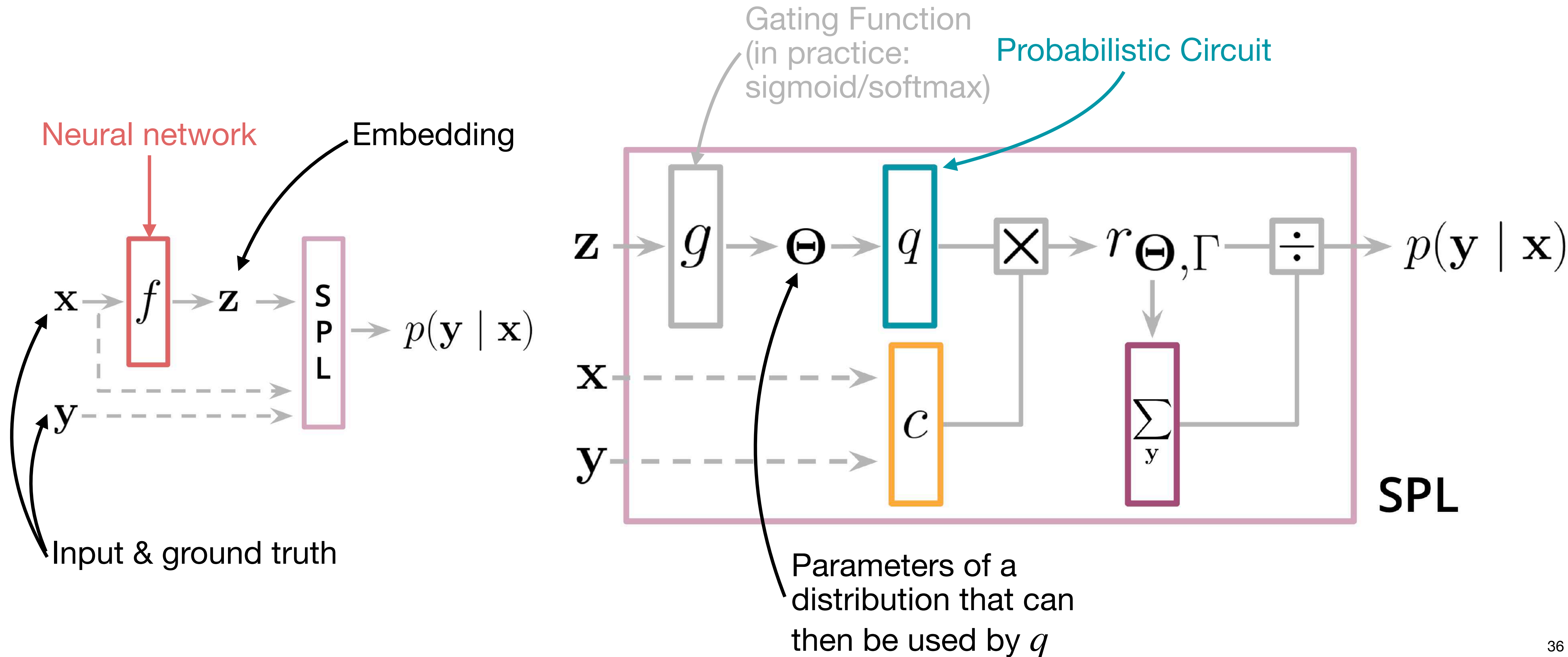


The probabilistic route: Semantic Probabilistic Layer (SPL)



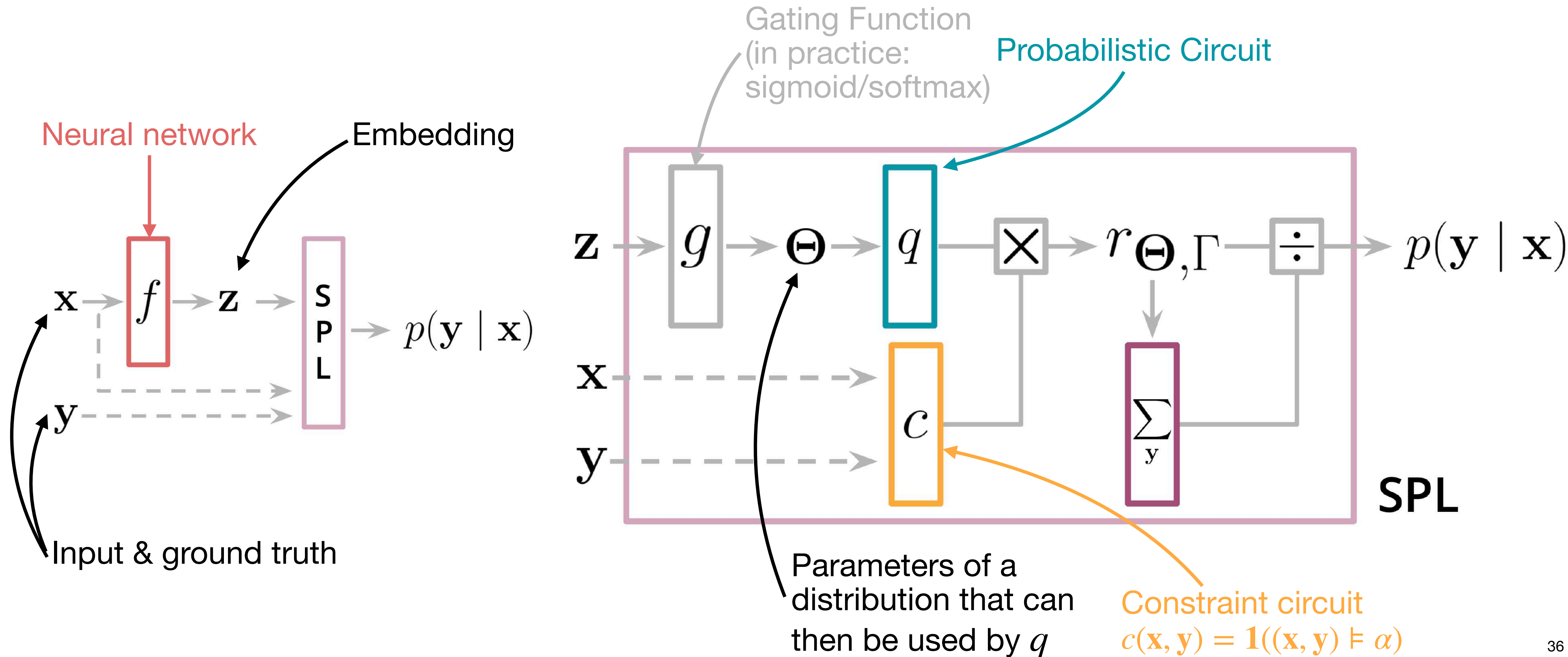


The probabilistic route: Semantic Probabilistic Layer (SPL)



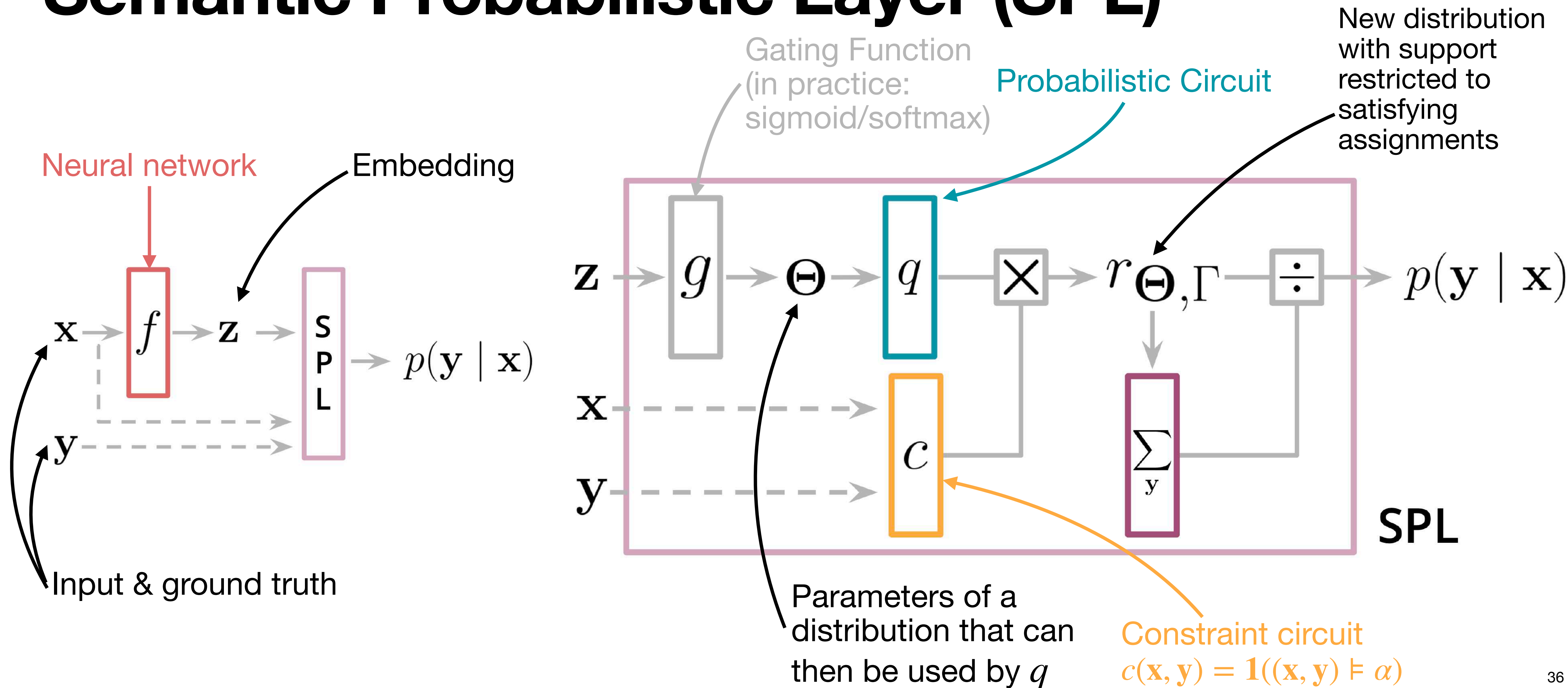


The probabilistic route: Semantic Probabilistic Layer (SPL)



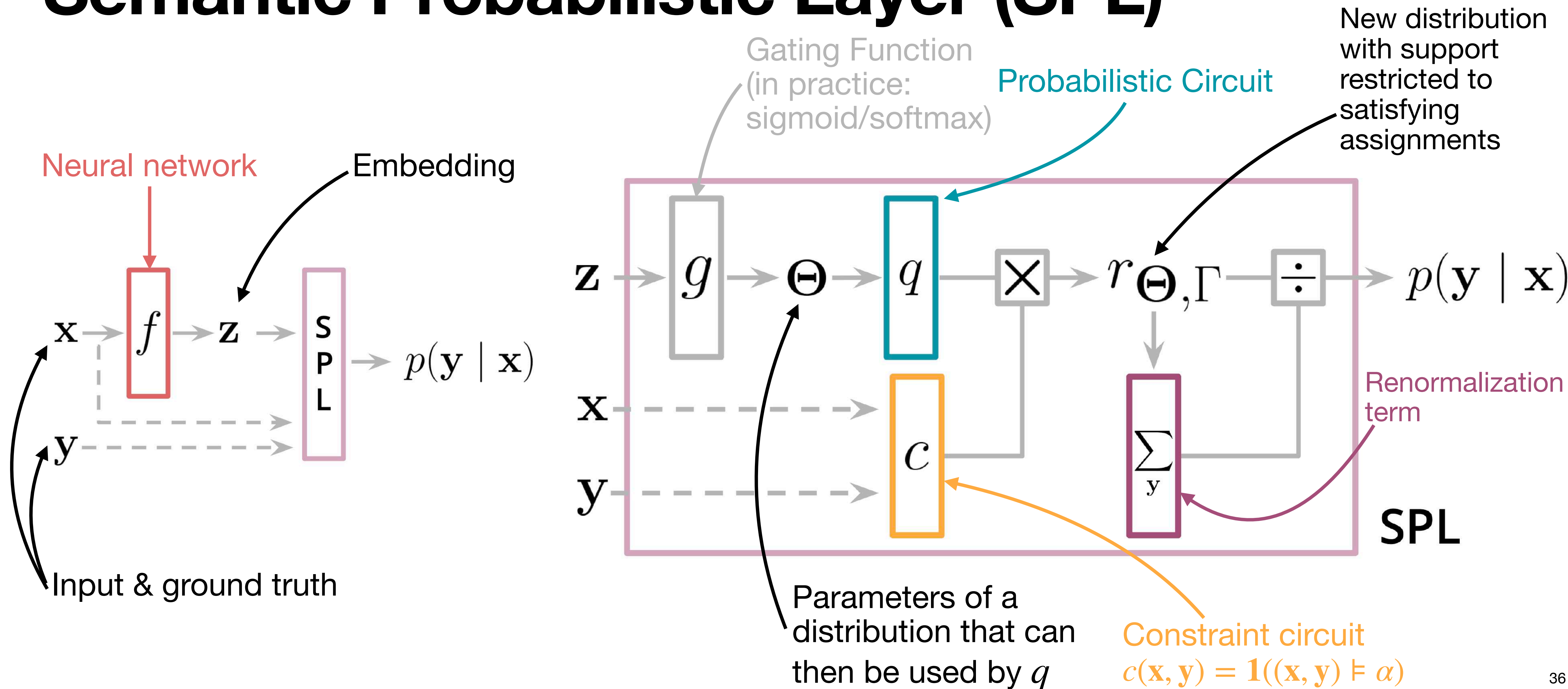


The probabilistic route: Semantic Probabilistic Layer (SPL)



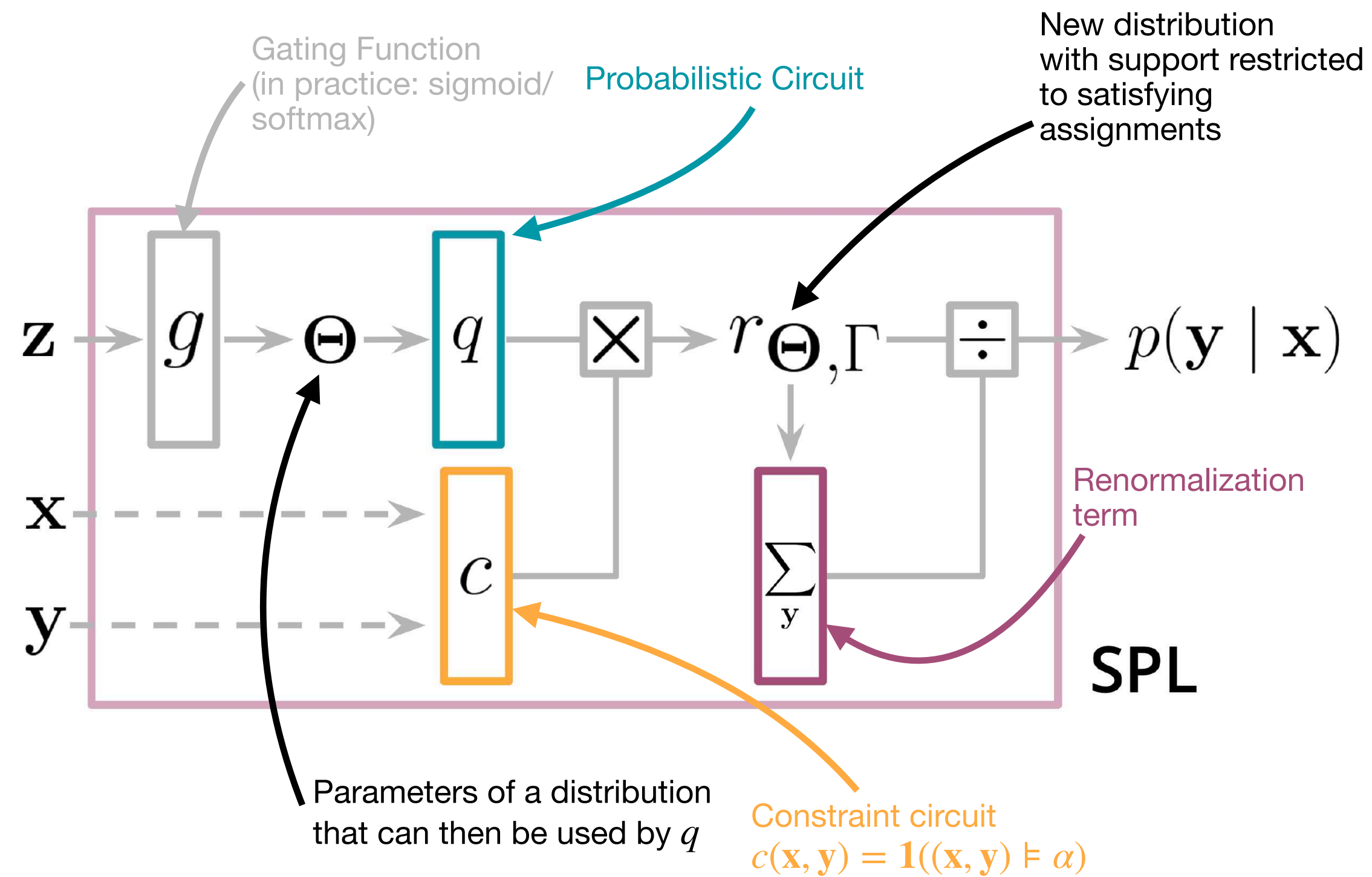


The probabilistic route: Semantic Probabilistic Layer (SPL)





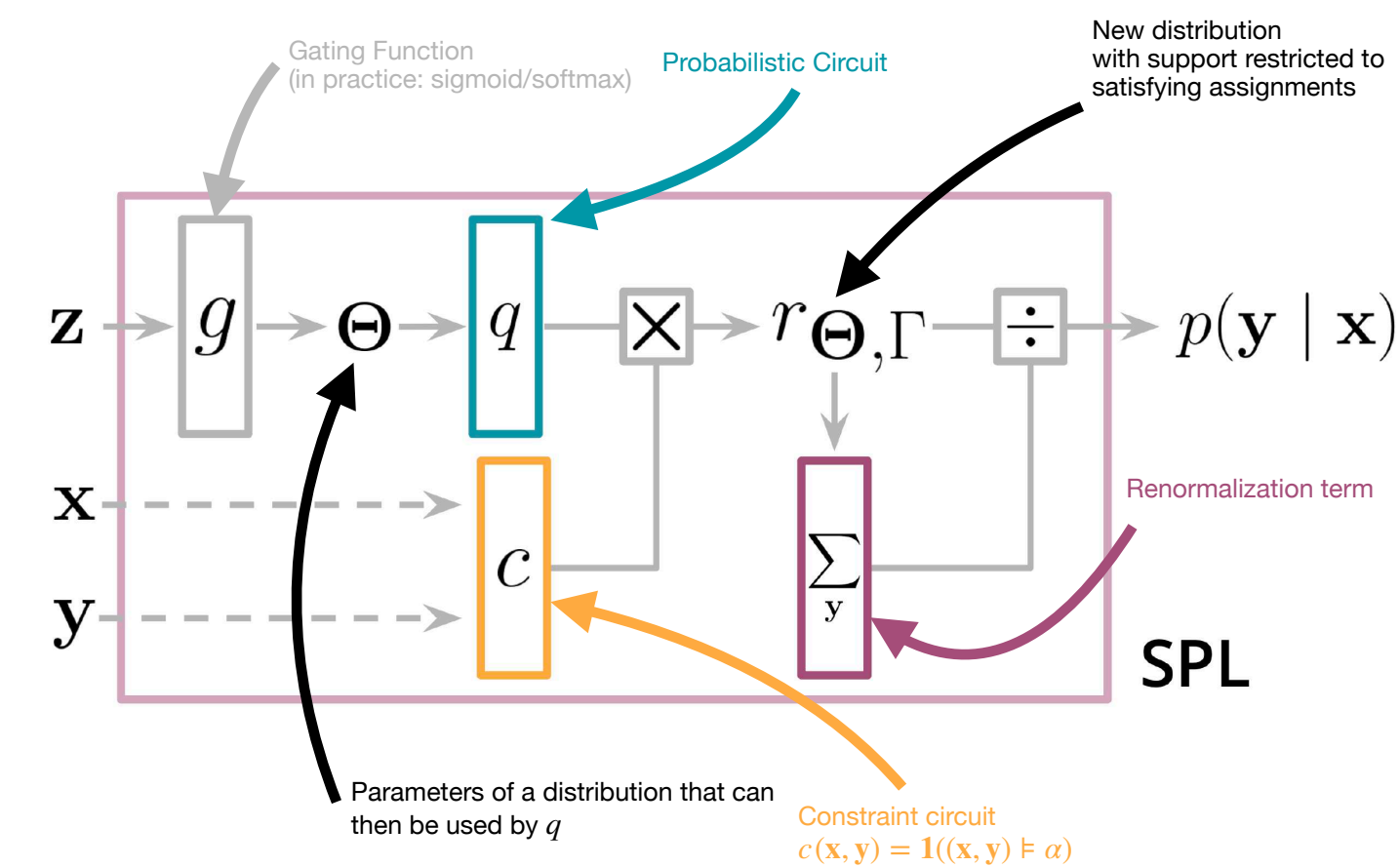
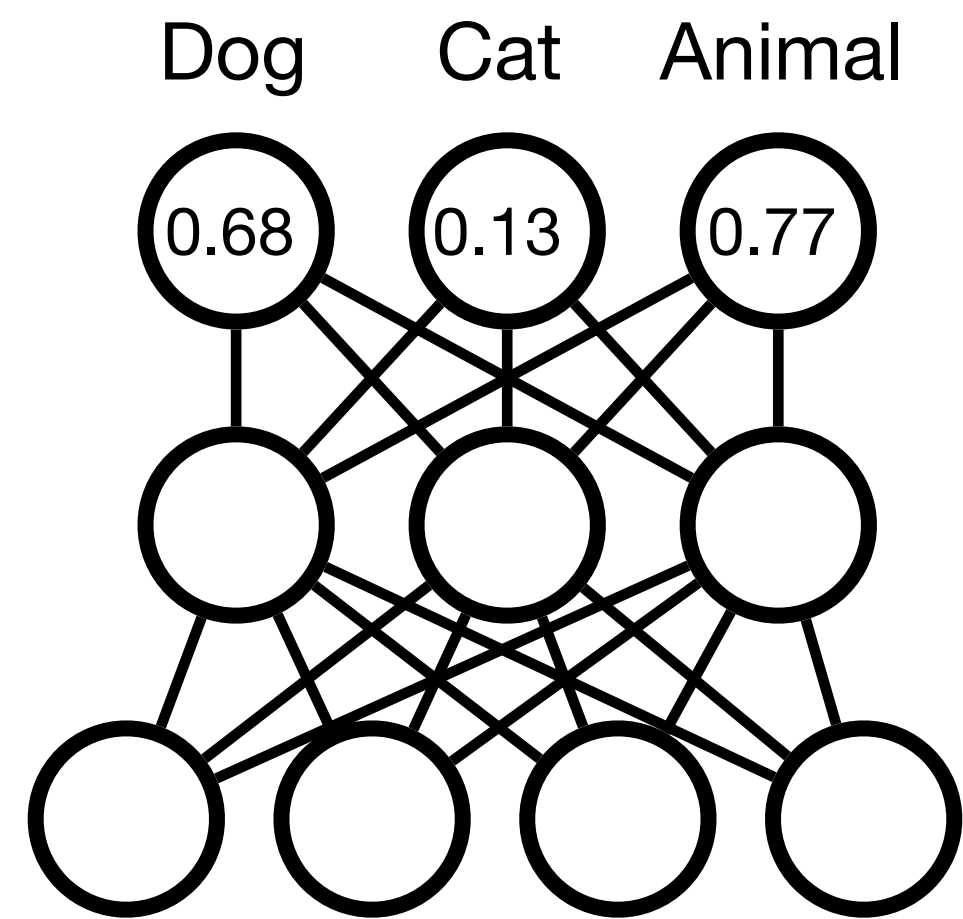
Semantic Probabilistic Layer (SPL)



$$p(y | \mathbf{x}) = \frac{q_{\theta}(y | g(f(\mathbf{x}))) \cdot c(\mathbf{x}, y)}{\sum_y q_{\theta}(y | g(f(\mathbf{x}))) \cdot c(\mathbf{x}, y)}$$

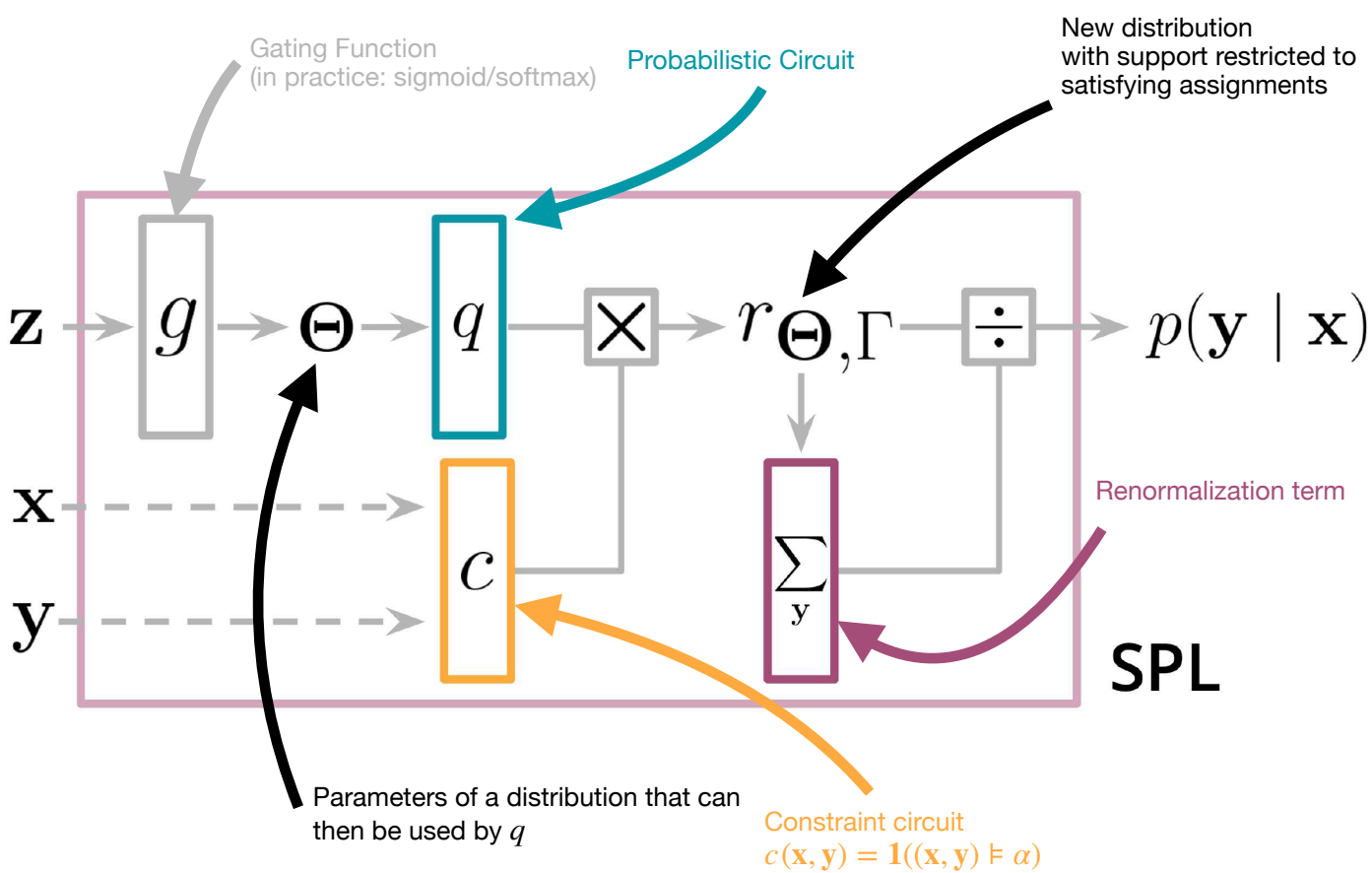
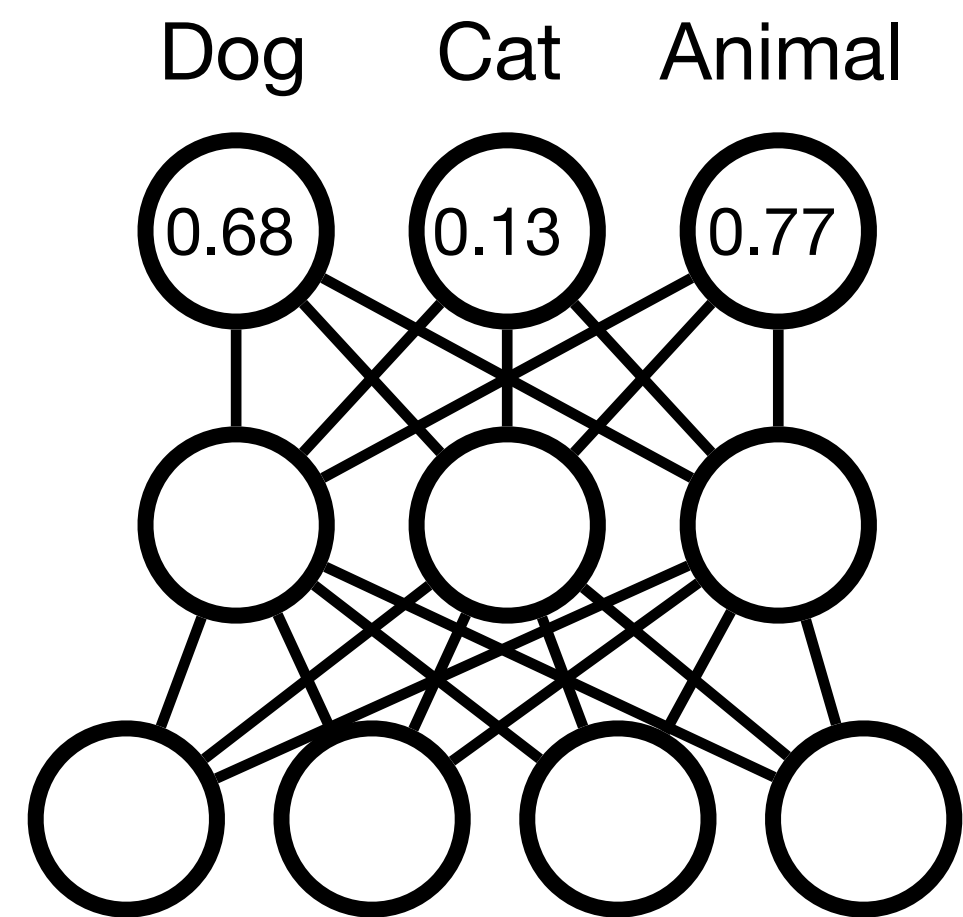
Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



Let's go back to our example...

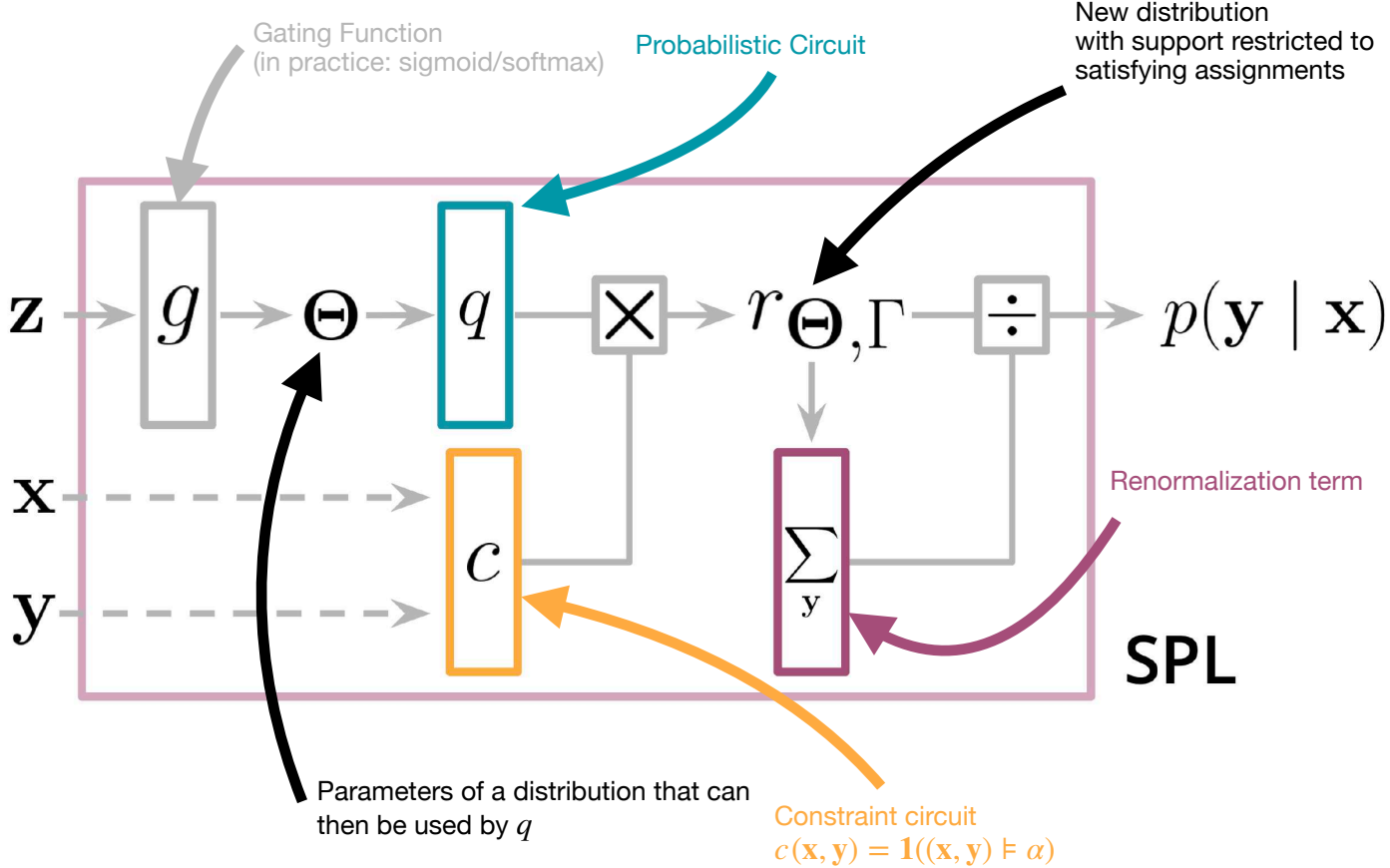
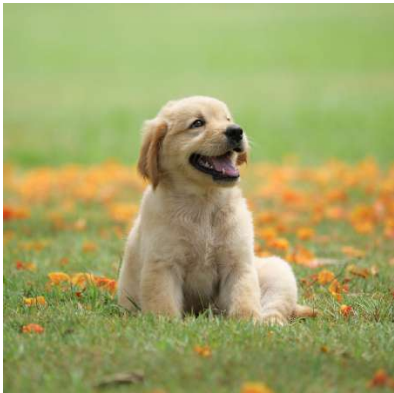
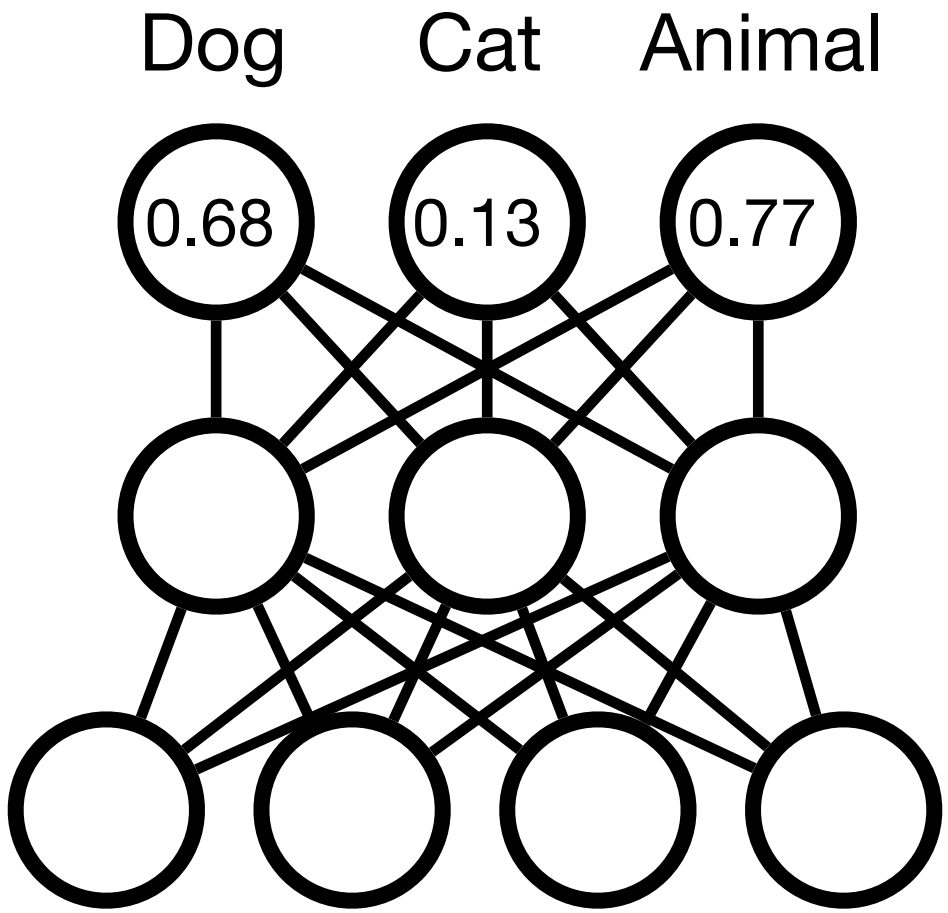
$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



Let us try to see how c would look like in this example.

Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$

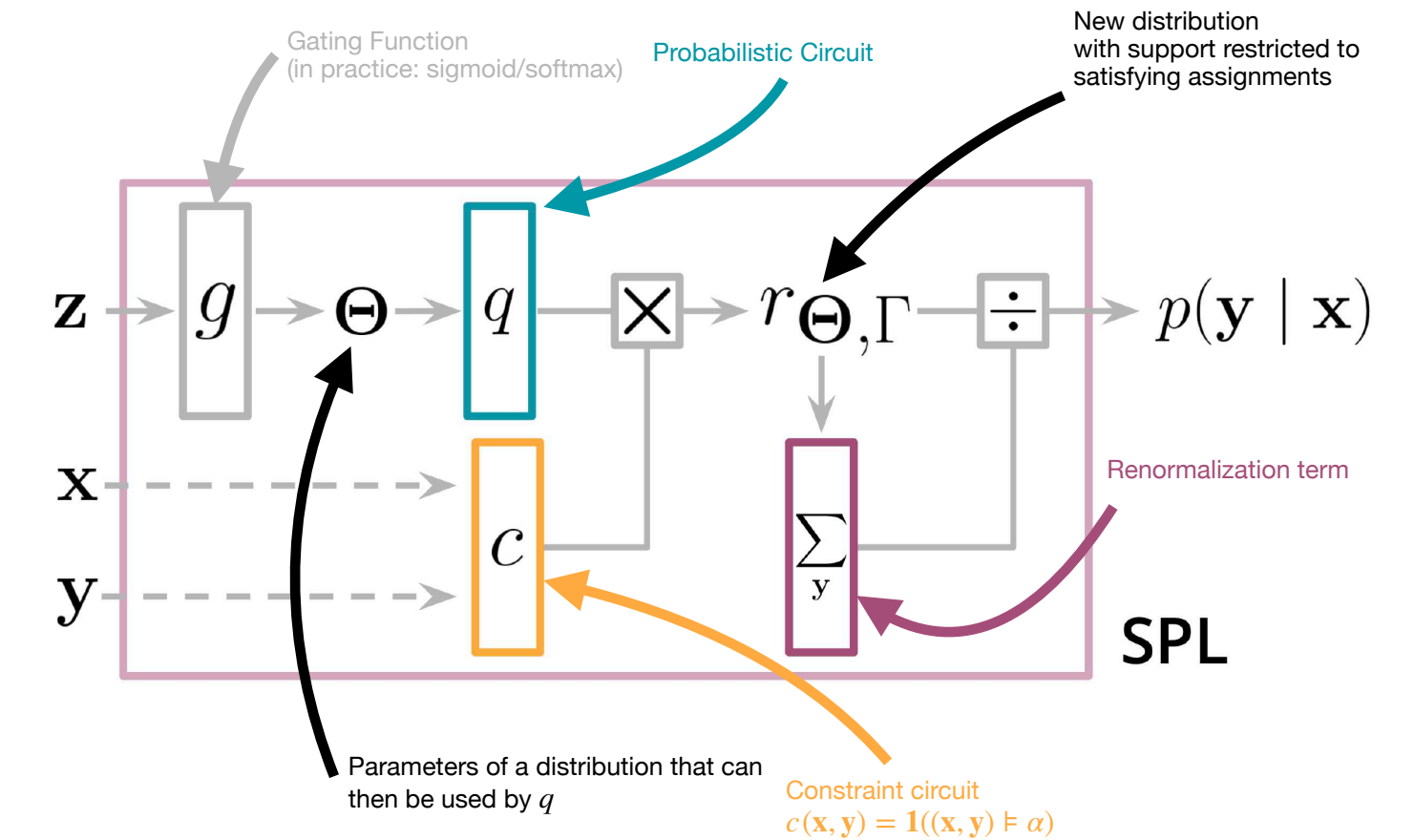
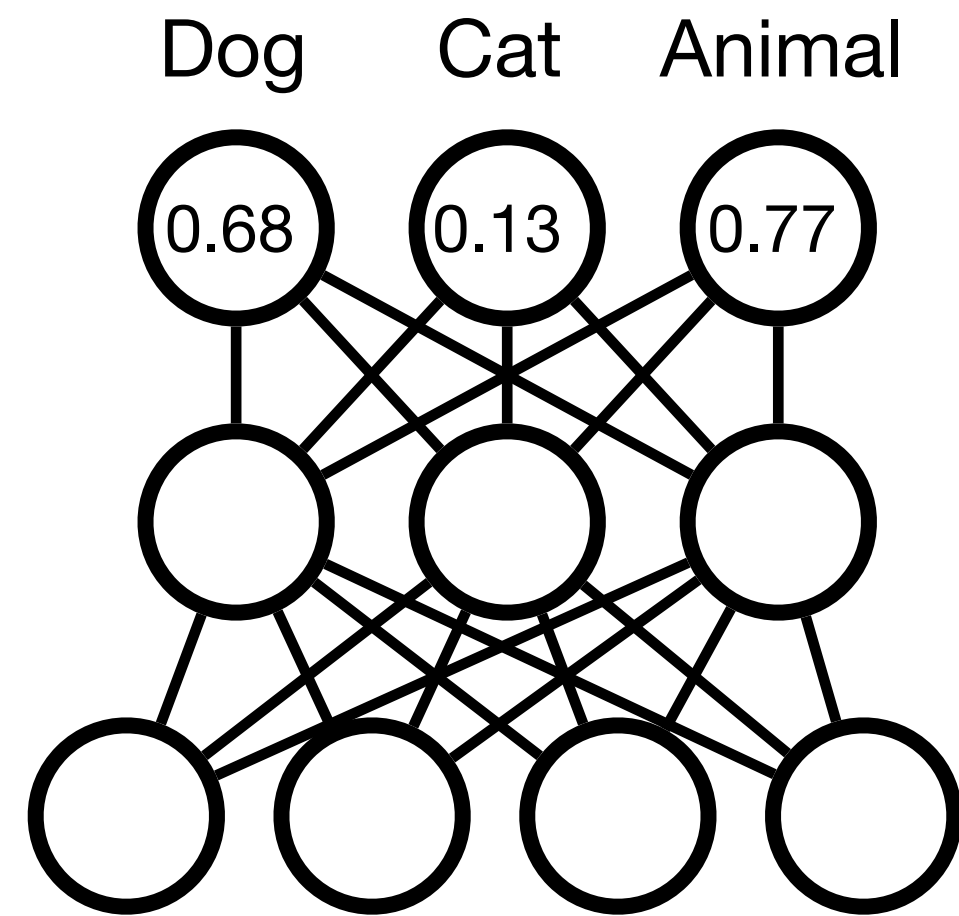


Let us try to see how c would look like in this example.

In general it will be a *circuit*, i.e., a computational graph made of 3 computational units:

Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



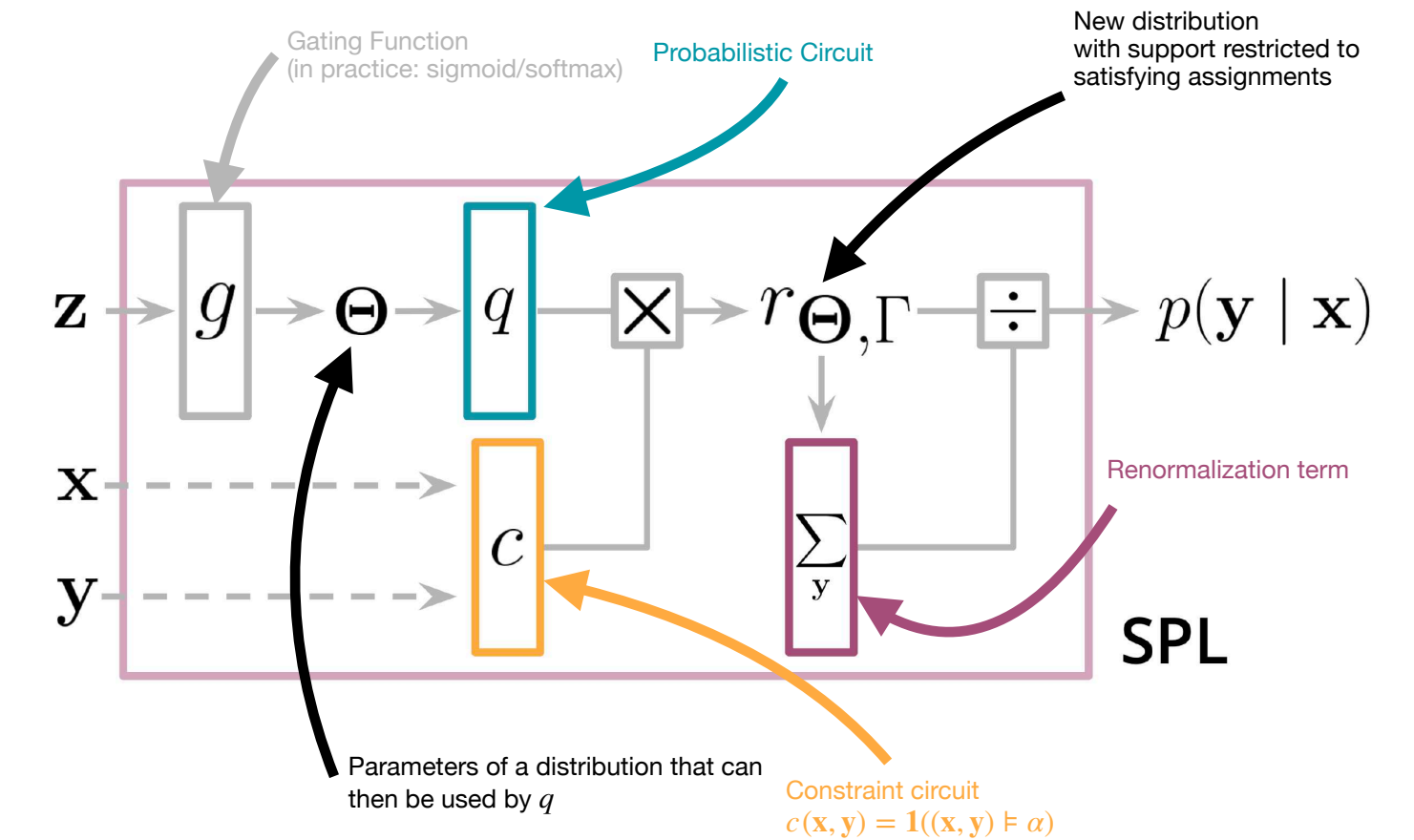
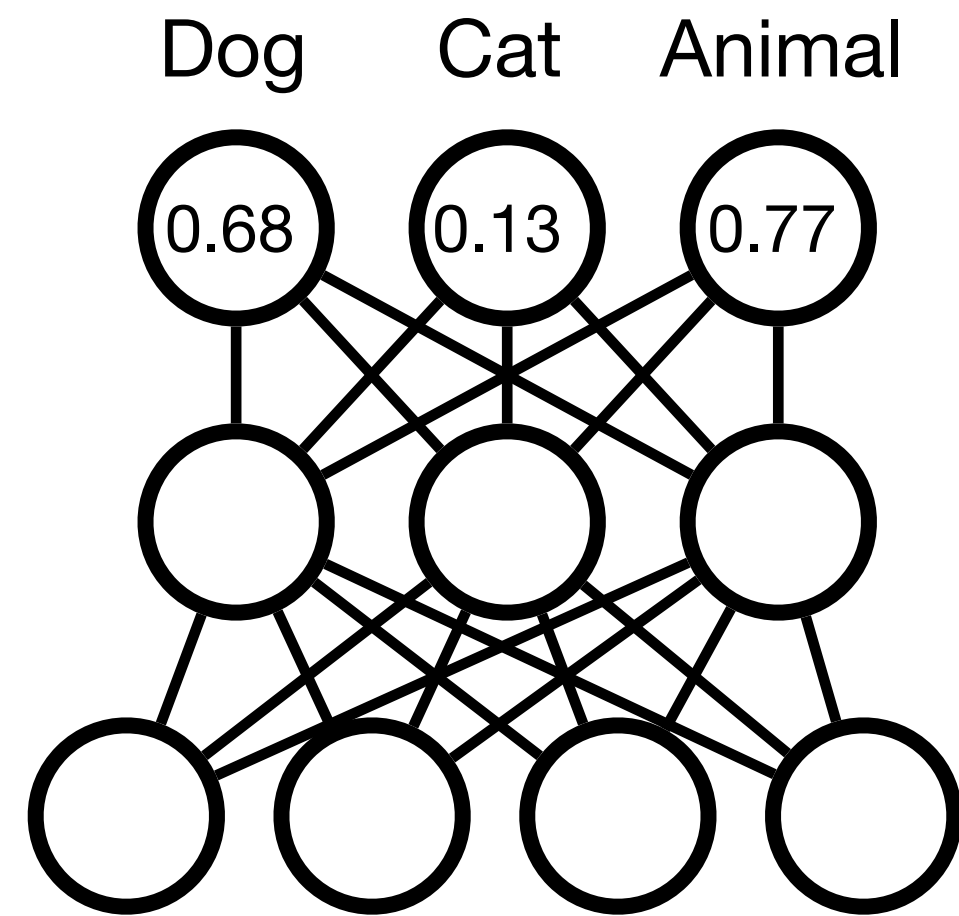
Let us try to see how c would look like in this example.

In general it will be a *circuit*, i.e., a computational graph made of 3 computational units:

- Input units

Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



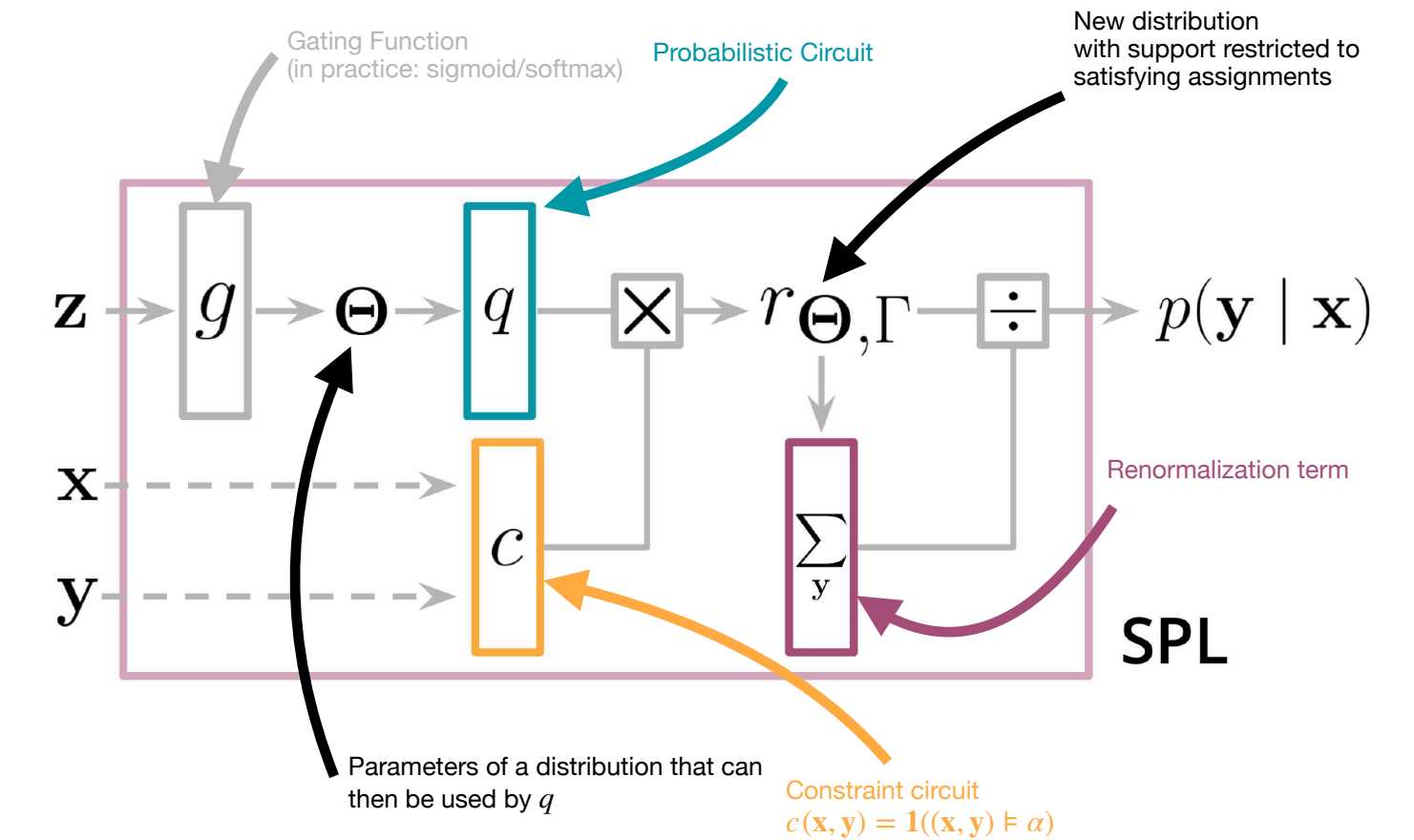
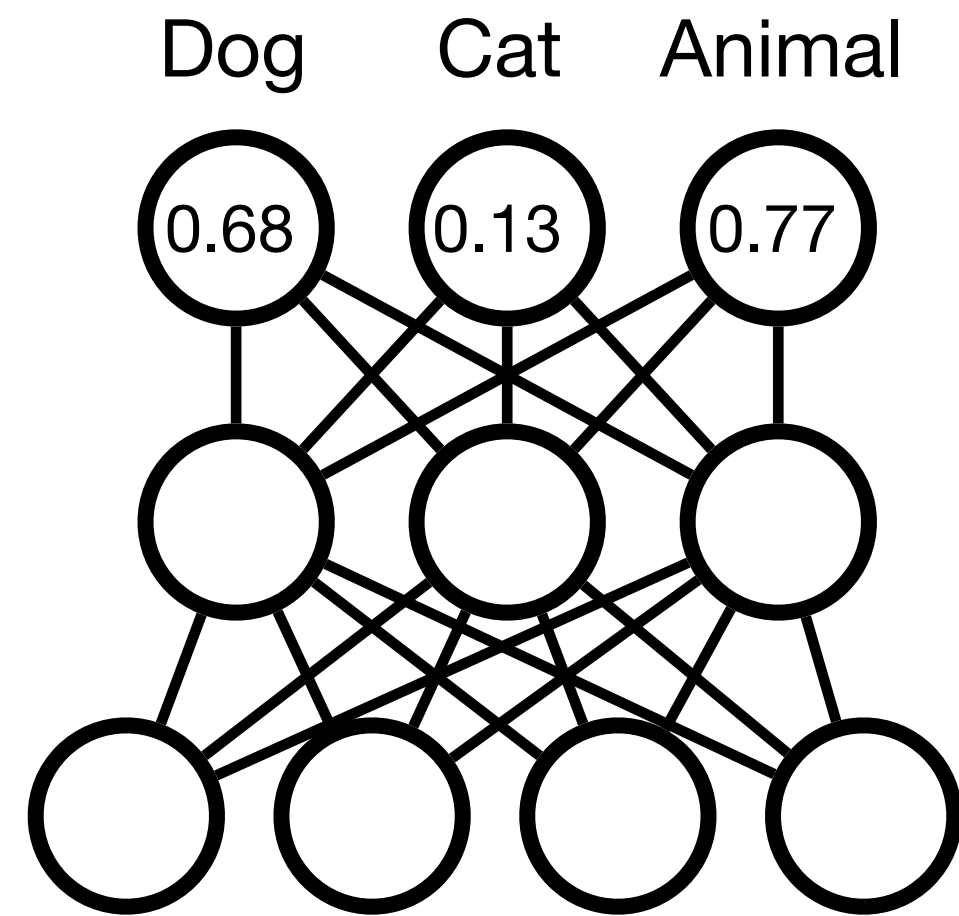
Let us try to see how c would look like in this example.

In general it will be a *circuit*, i.e., a computational graph made of 3 computational units:

- Input units
- Sum units

Let's go back to our example...

$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



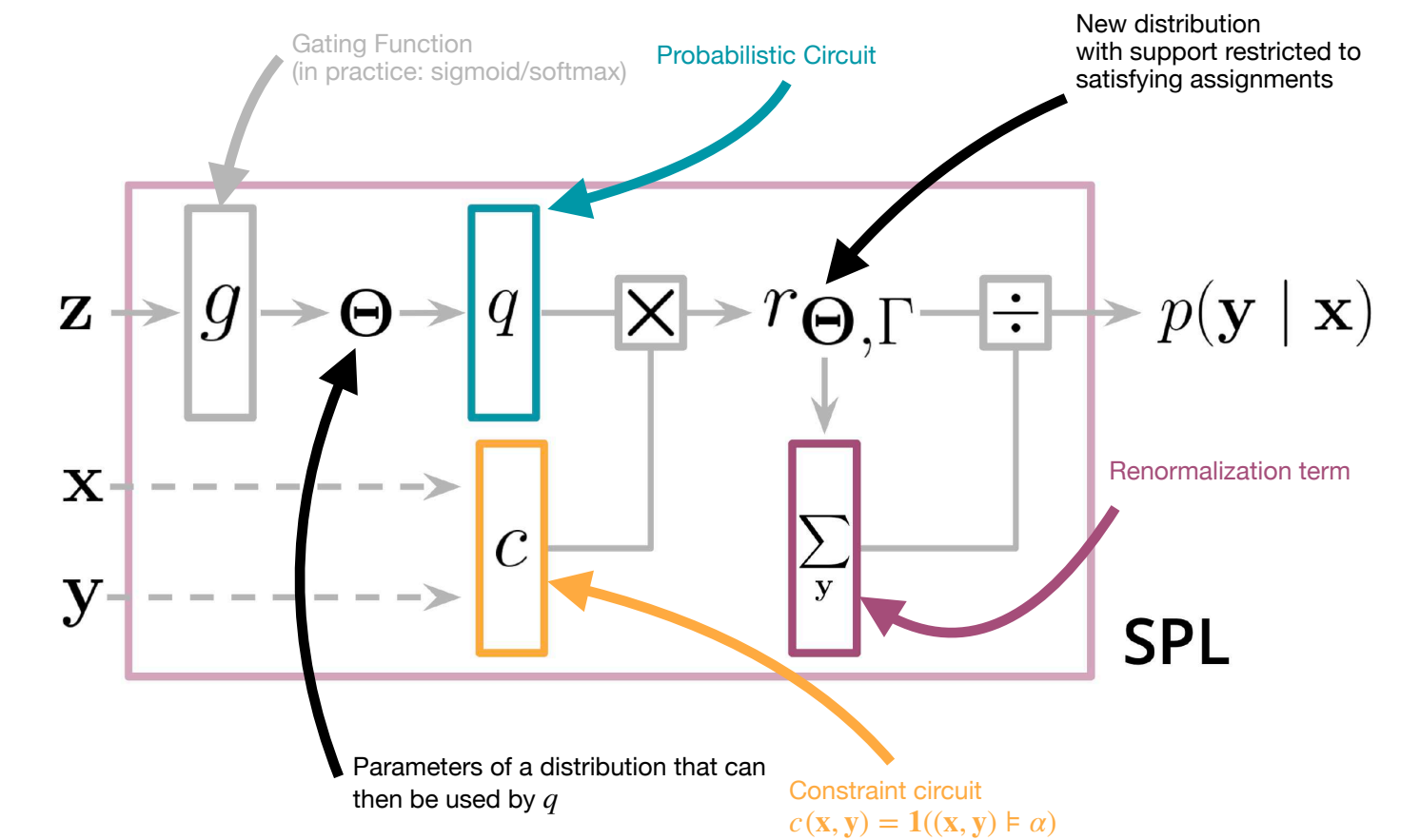
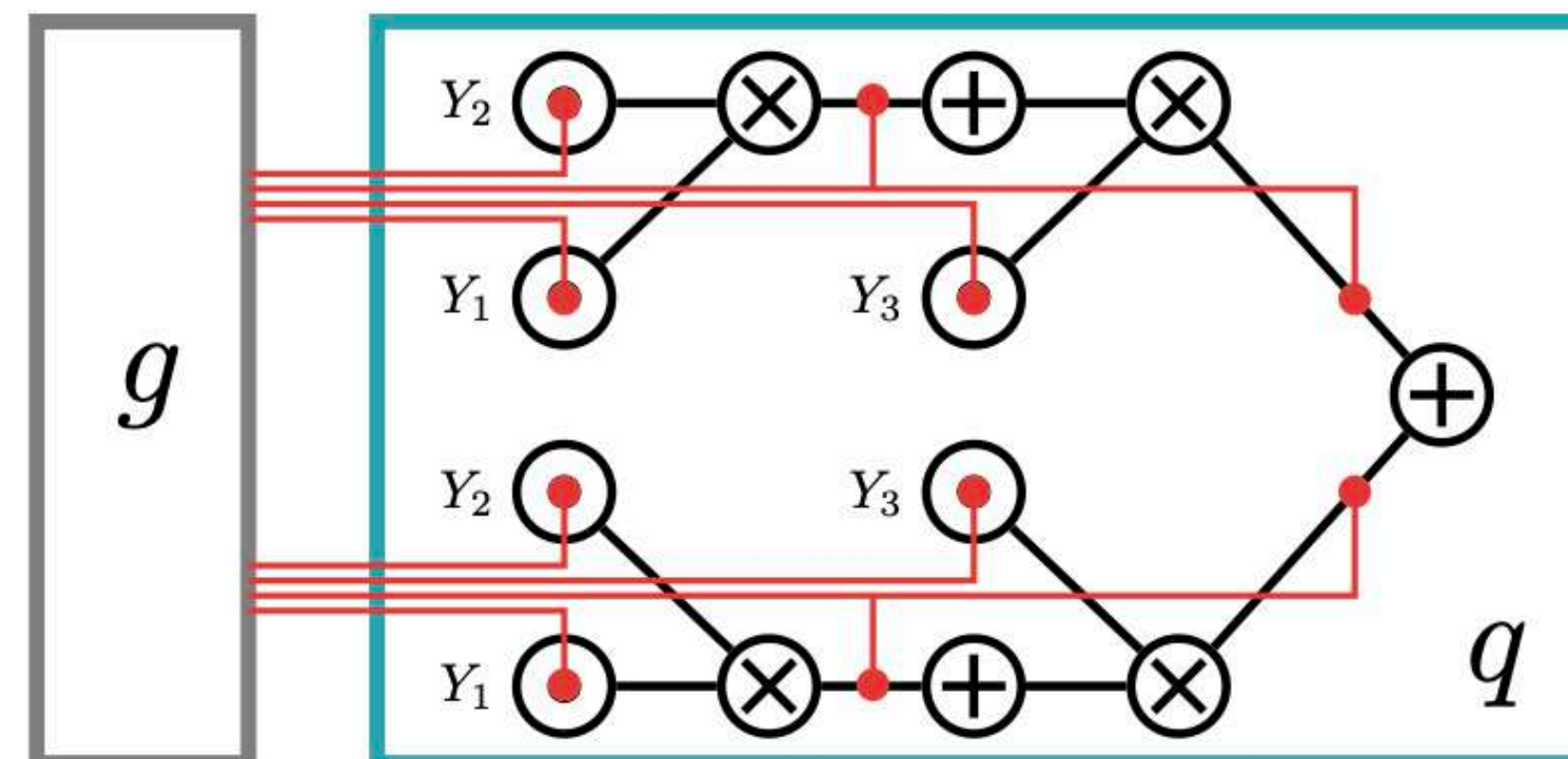
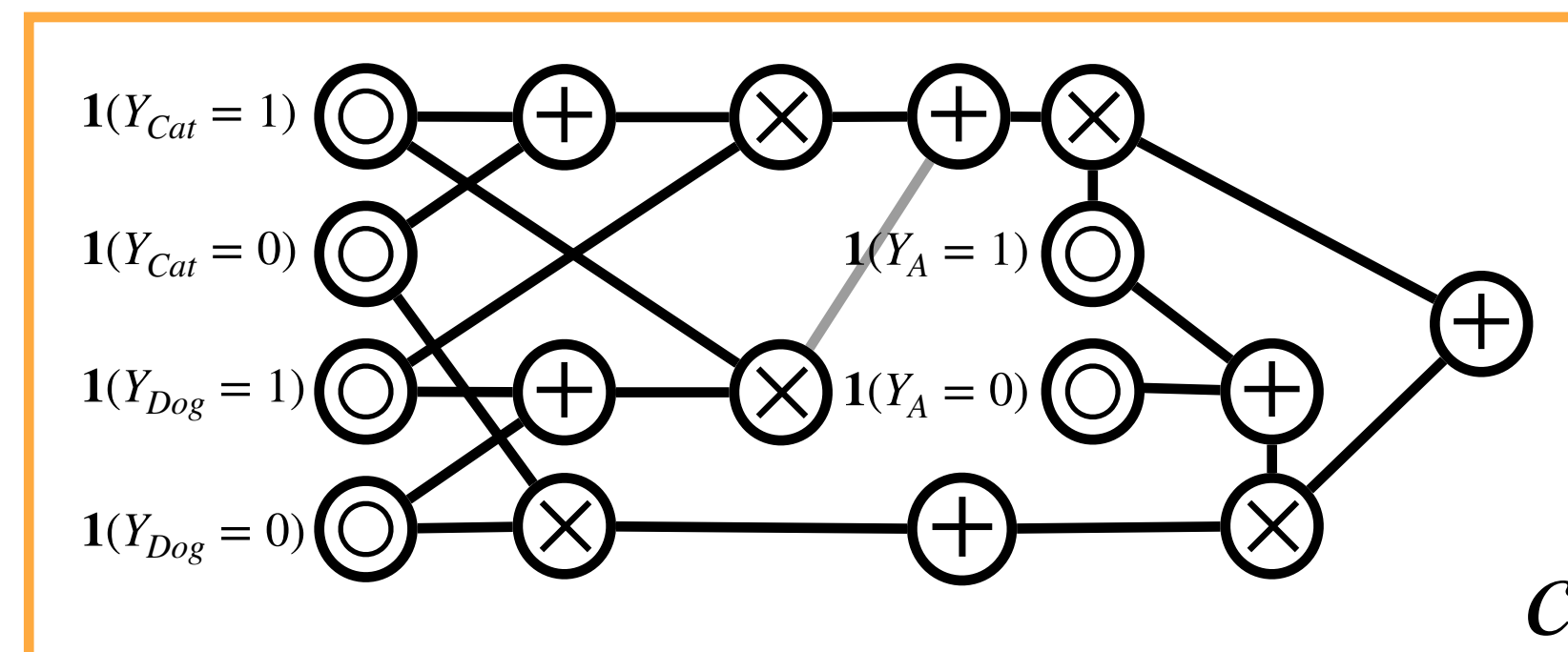
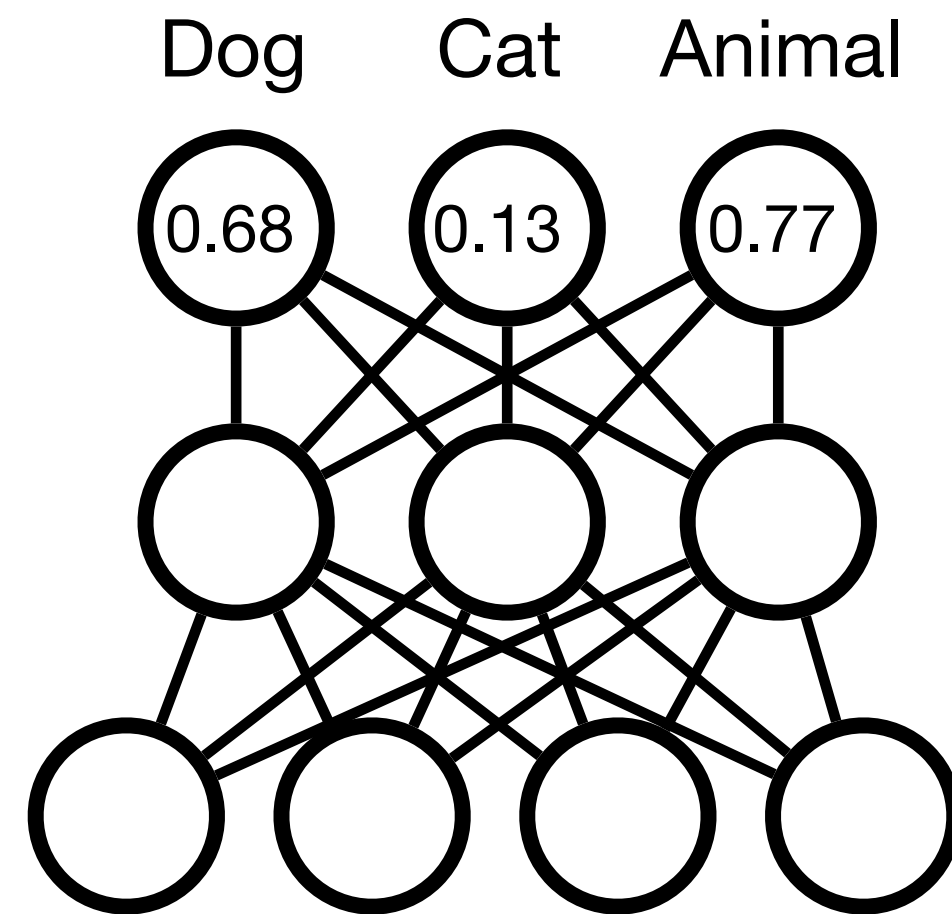
Let us try to see how c would look like in this example.

In general it will be a *circuit*, i.e., a computational graph made of 3 computational units:

- Input units
- Sum units
- Product units

What about q_θ ?

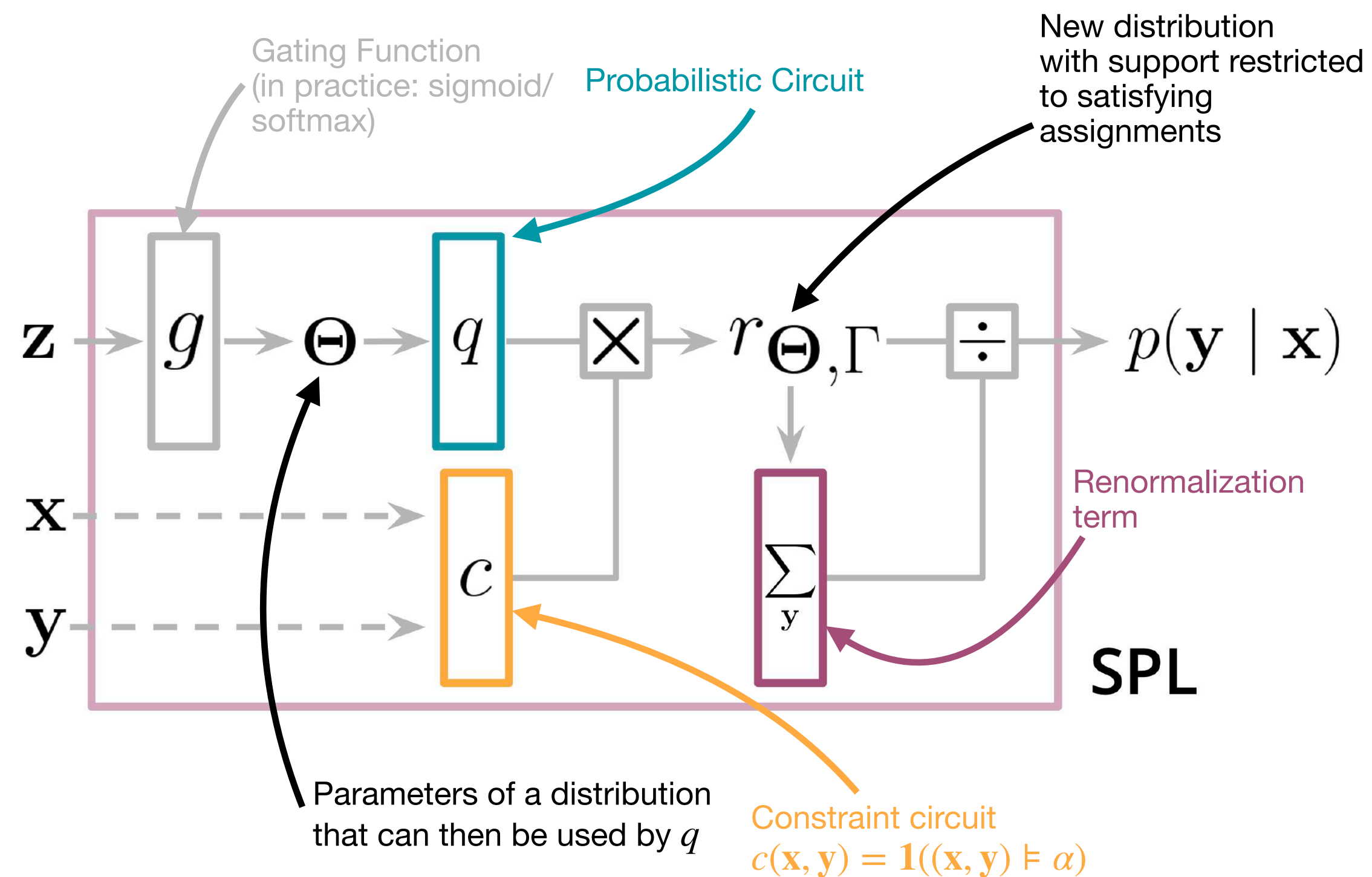
$$\alpha = (\text{Dog} \rightarrow \text{Animal}) \wedge (\text{Cat} \rightarrow \text{Animal})$$



c is just a particular PC with all sum unit parameters equal to 1 and input functionals that are indicator functions over their scope!



Semantic Probabilistic Layer (SPL)



$$p(\mathbf{y} \mid \mathbf{x}) = \frac{q_{\theta}(\mathbf{y} \mid g(f(\mathbf{x}))) \cdot c(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{y}} q_{\theta}(\mathbf{y} \mid g(f(\mathbf{x}))) \cdot c(\mathbf{x}, \mathbf{y})}$$

Pros and Cons



Pros

Cons

- Beautiful probabilistic semantics
- ***Safe wrt to sampling***

- Slower than other methods (*)
- If datapoint breaks the constraints
gradients explode - network is untrainable

Fuzzy

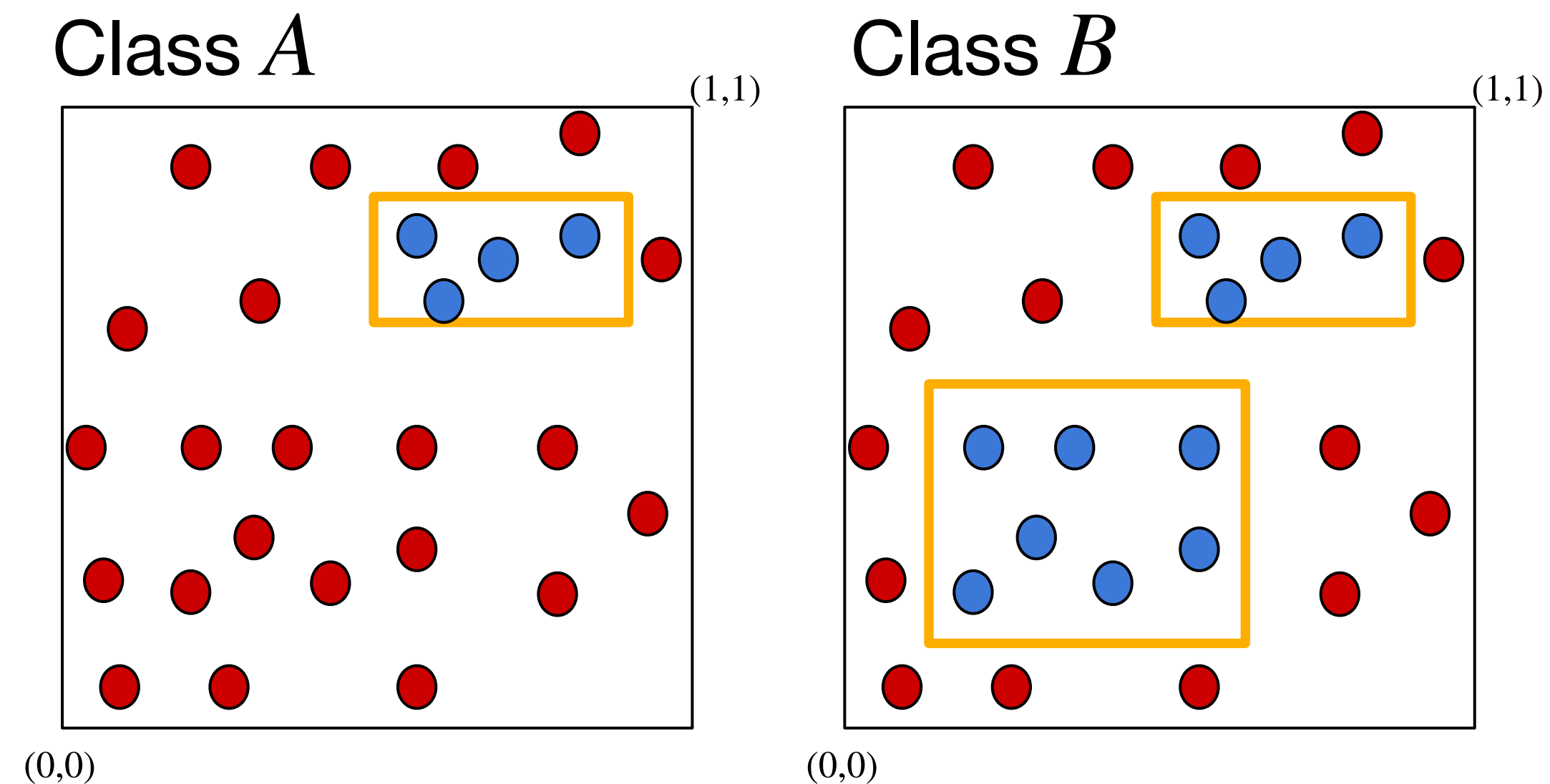
Probabilistic



How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

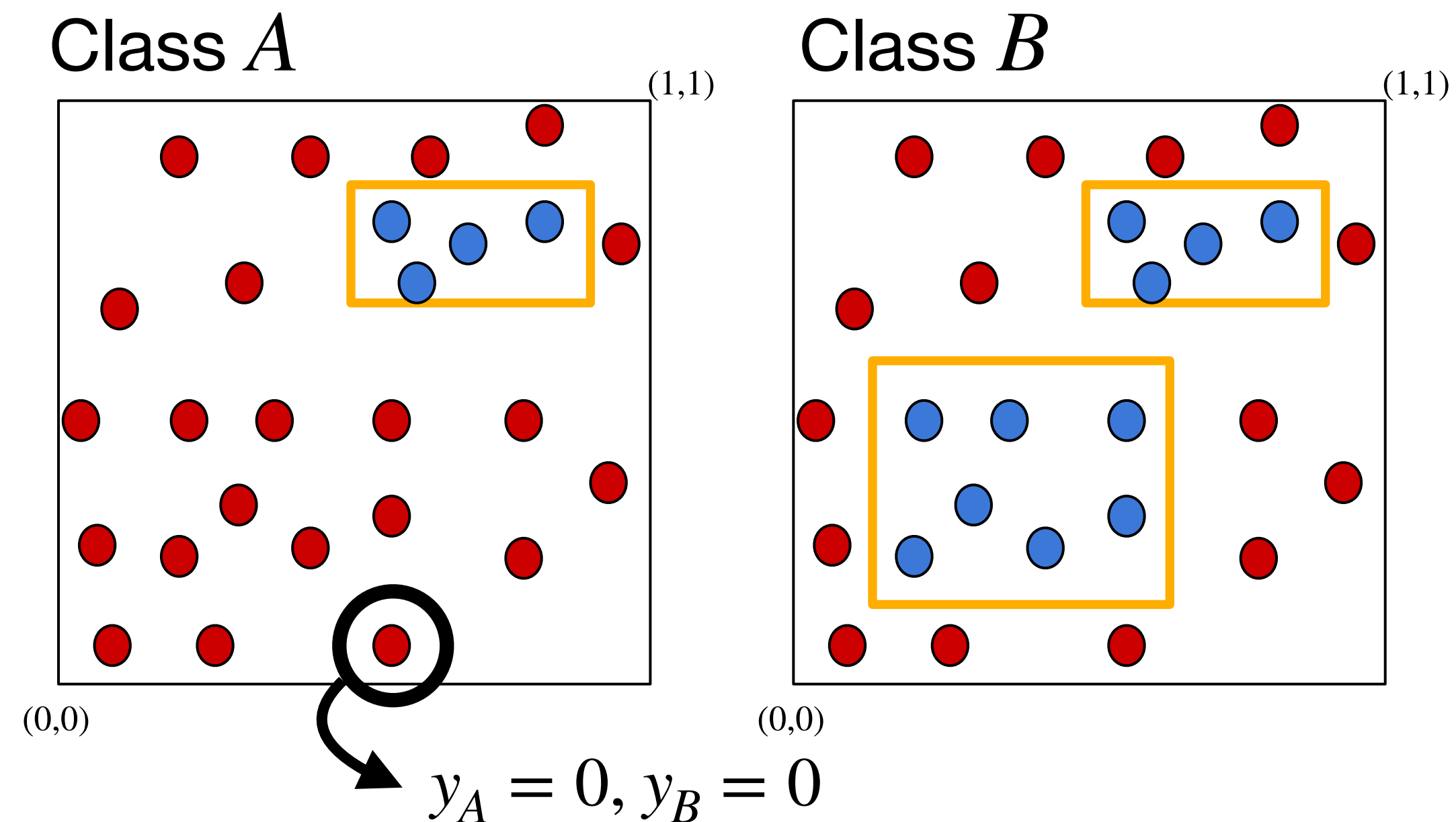




How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

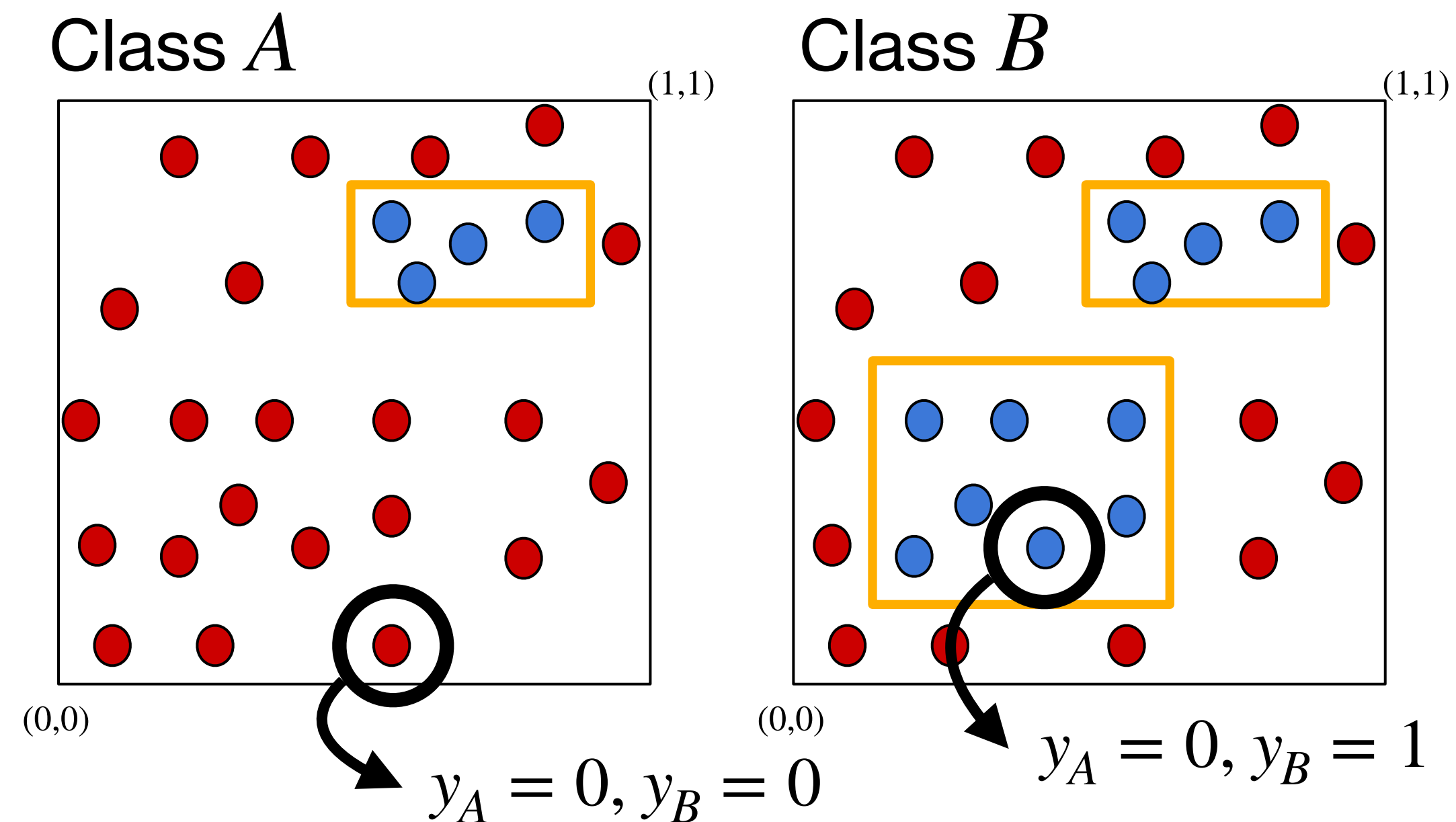




How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

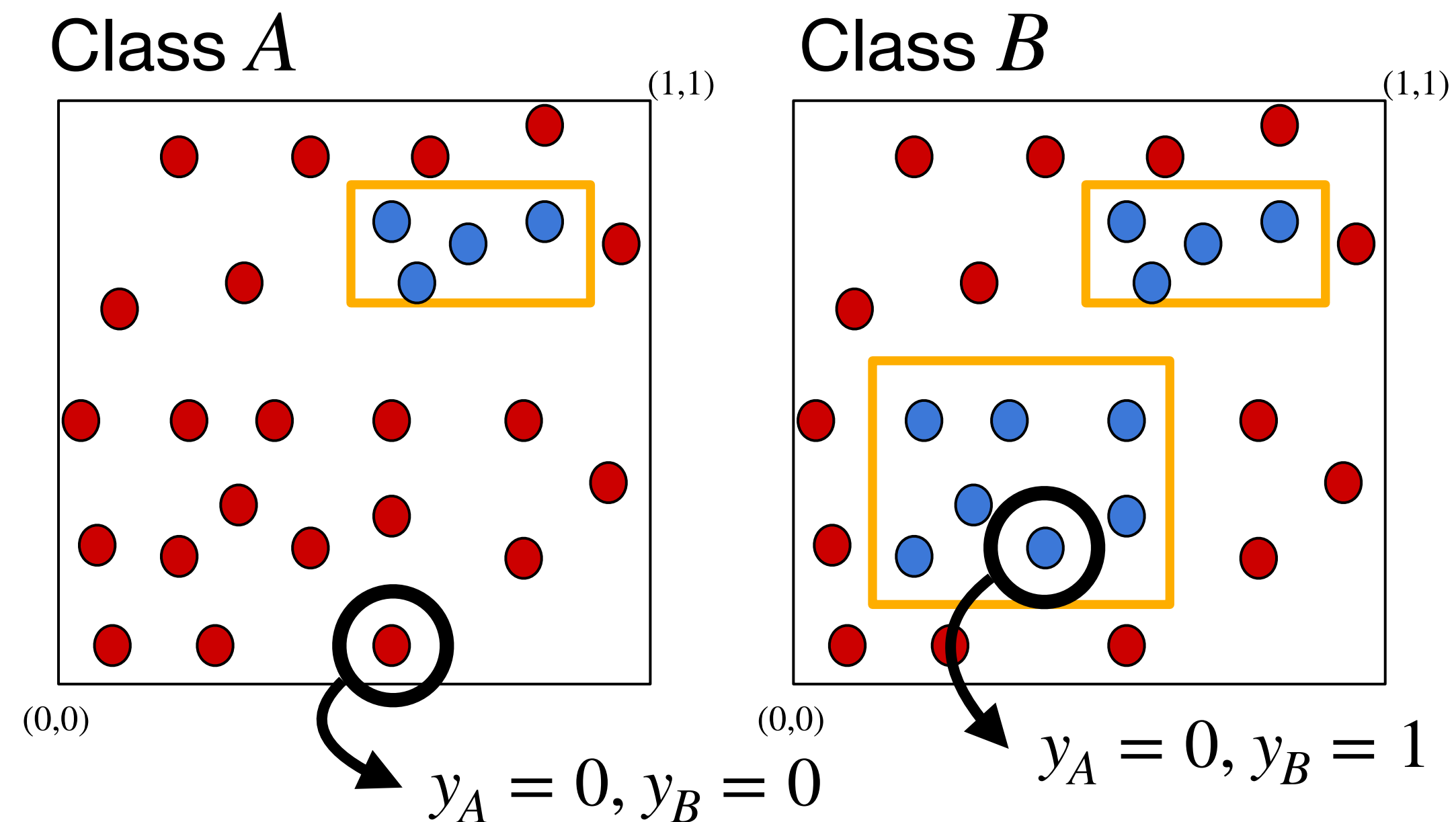




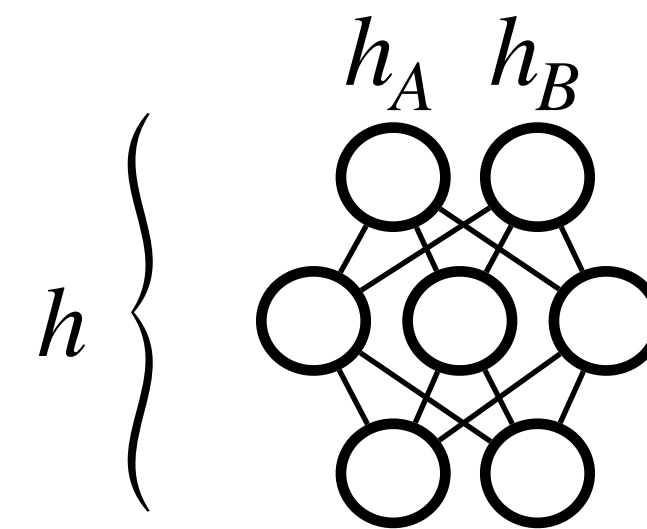
How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Network:

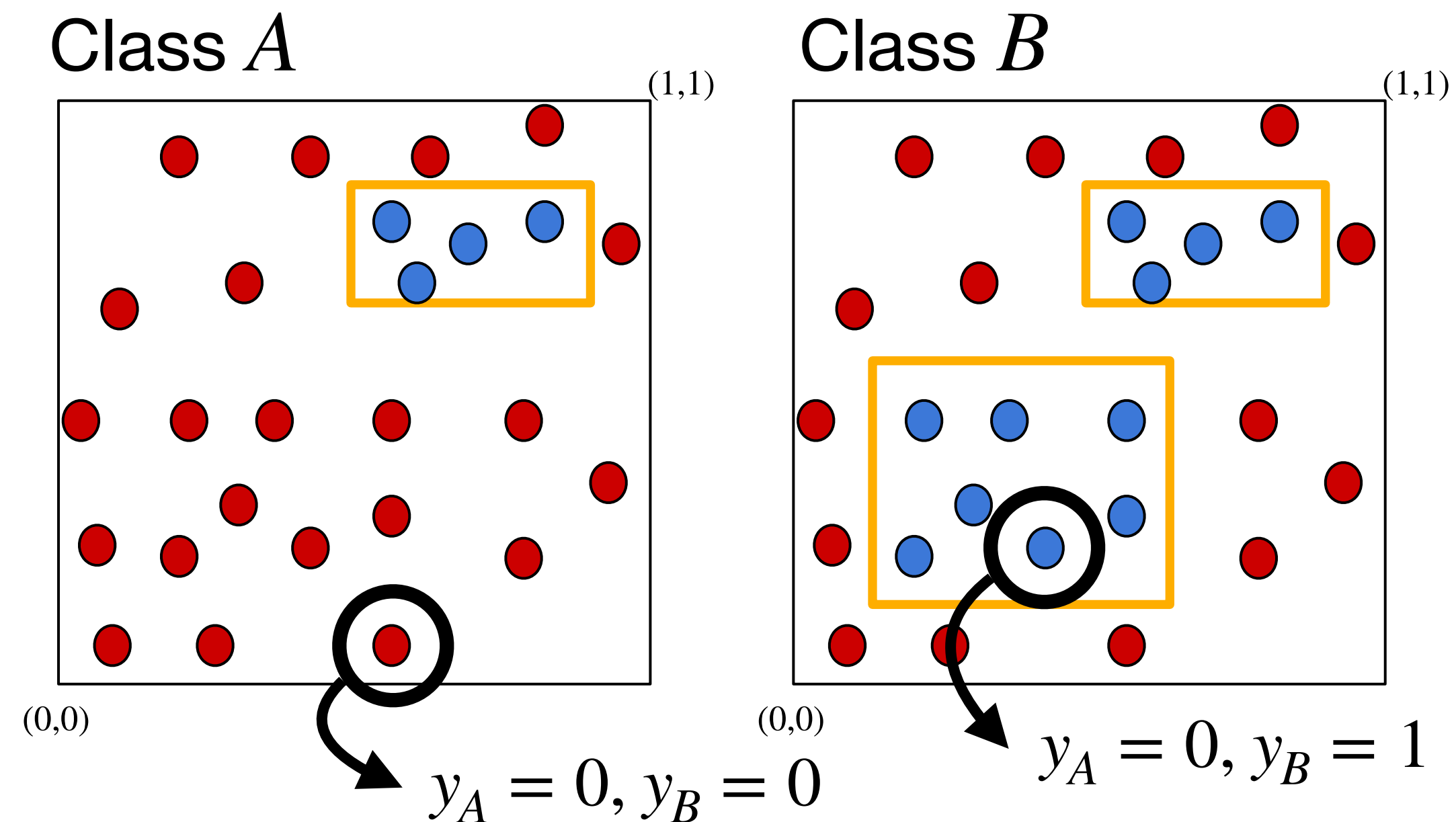




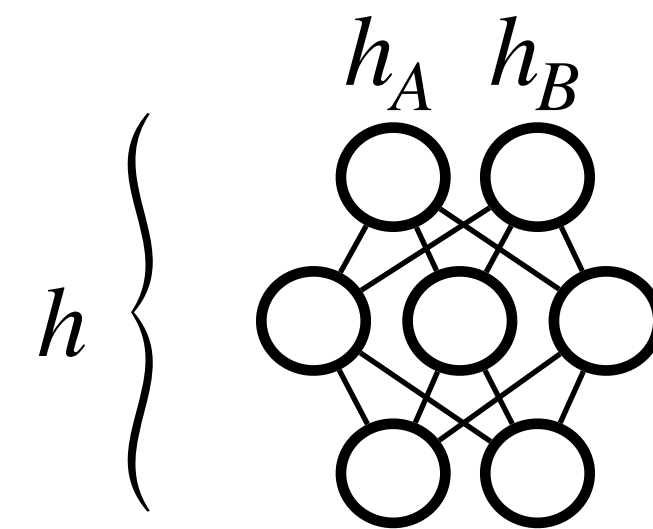
How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Network:



Loss:

$$\mathcal{L} = \mathcal{L}_A + \mathcal{L}_B$$

$$\mathcal{L}_A = -y_A \ln(h_A) - (1 - y_A) \ln(1 - h_A)$$

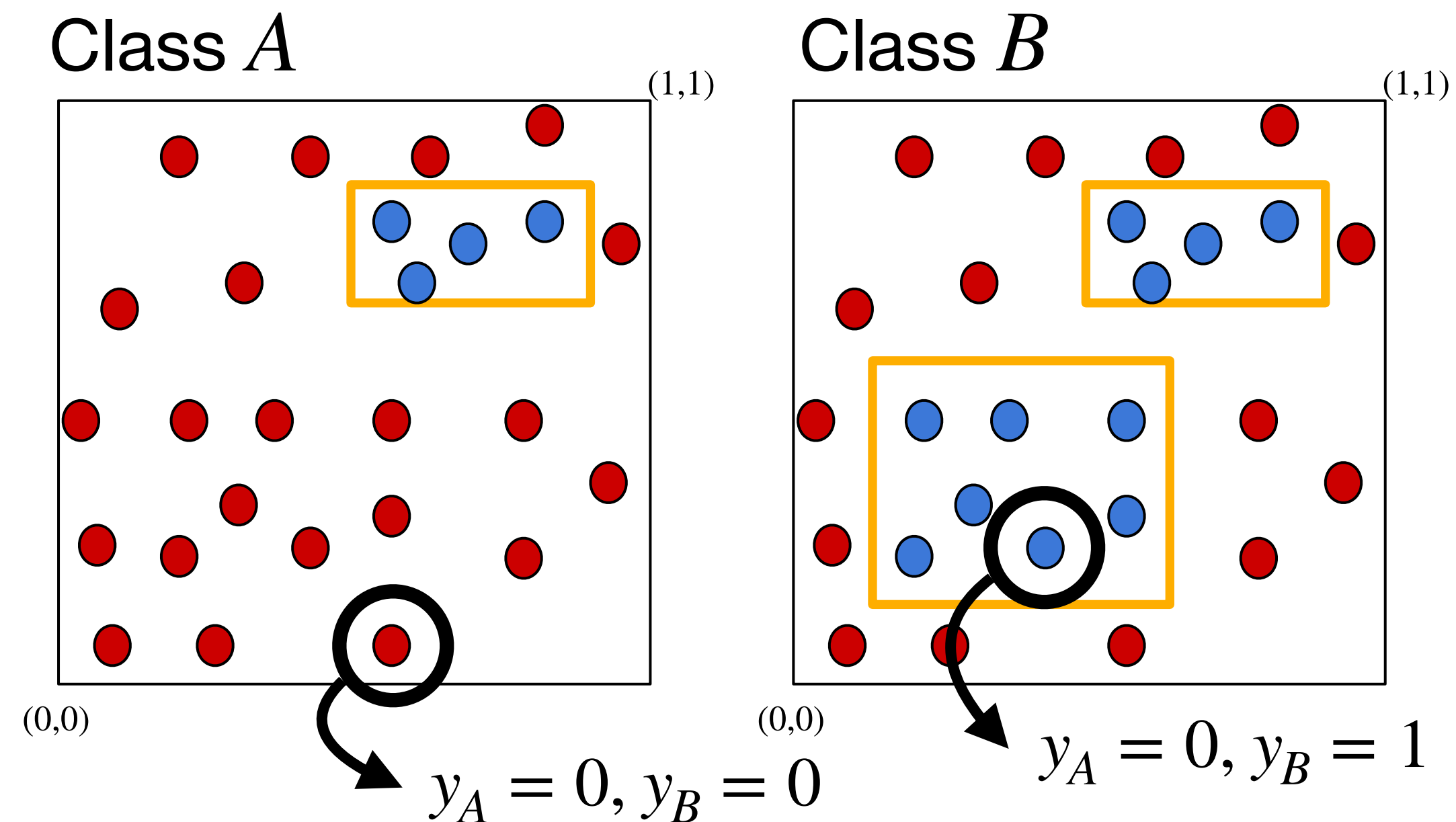
$$\mathcal{L}_B = -y_B \ln(h_B) - (1 - y_B) \ln(1 - h_B)$$



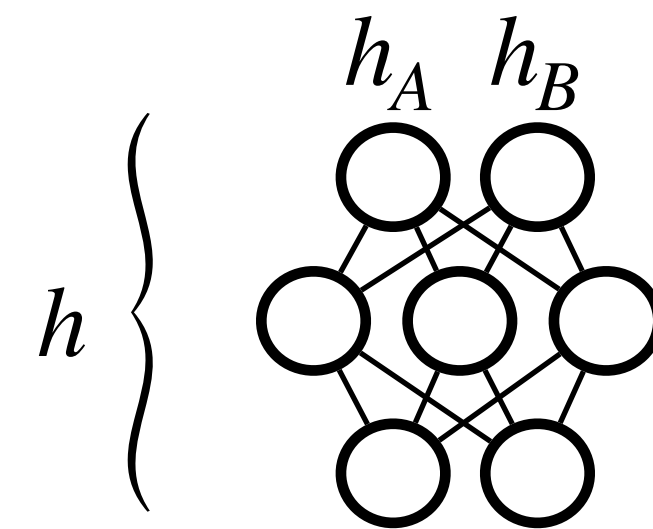
How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Network:

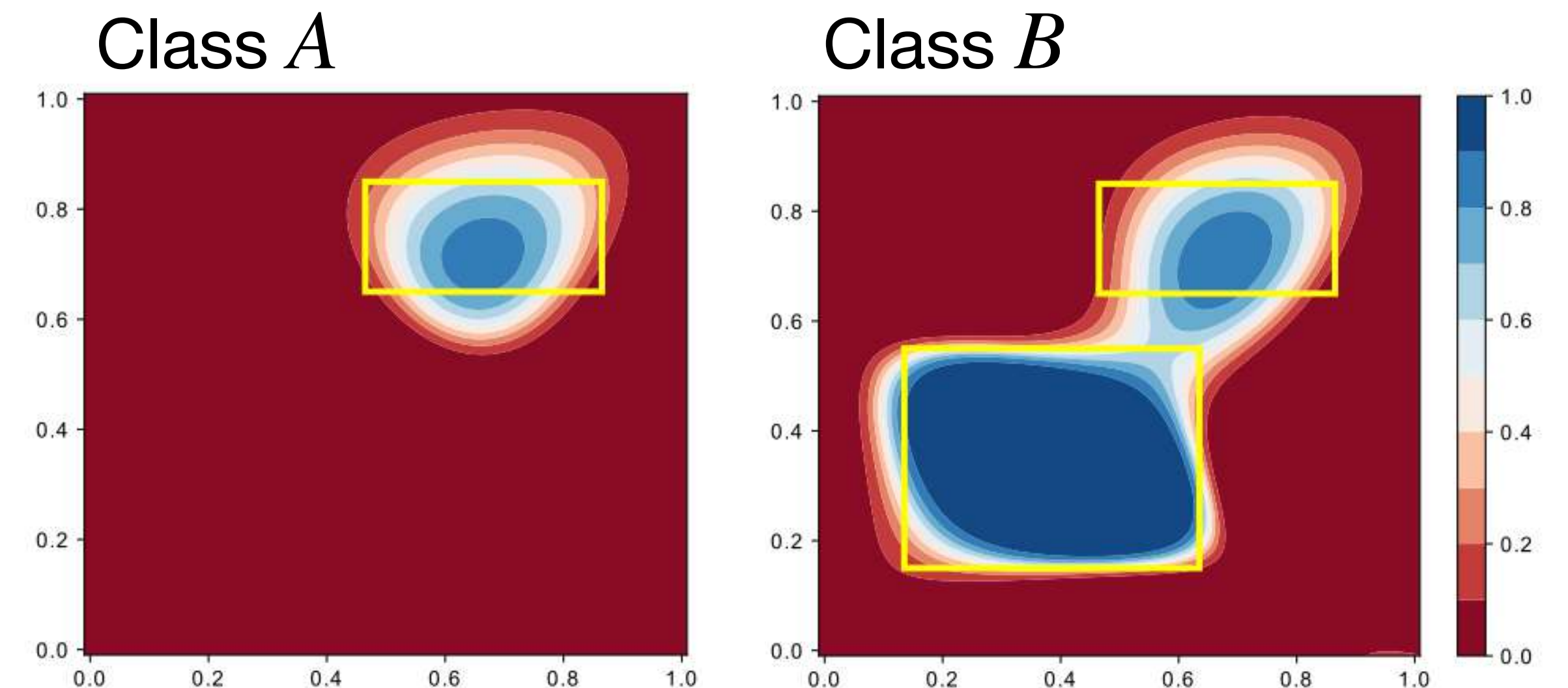


Loss:

$$\mathcal{L} = \mathcal{L}_A + \mathcal{L}_B$$

$$\mathcal{L}_A = -y_A \ln(h_A) - (1 - y_A) \ln(1 - h_A)$$

$$\mathcal{L}_B = -y_B \ln(h_B) - (1 - y_B) \ln(1 - h_B)$$

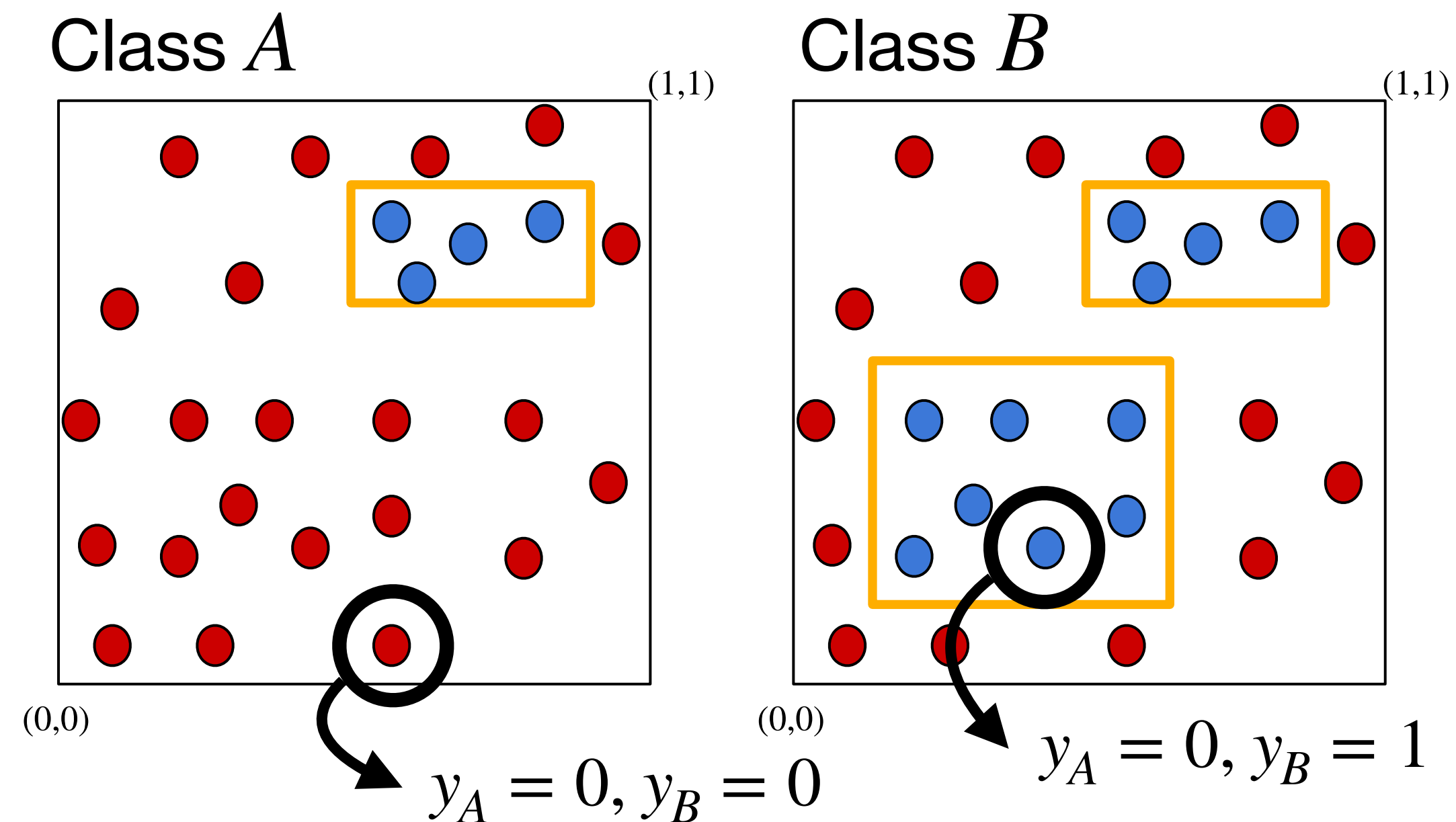




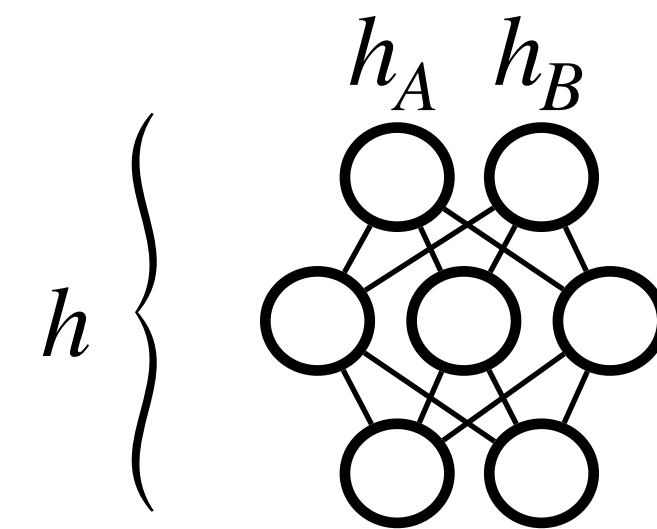
How can we design a fuzzy layer?

A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Network:

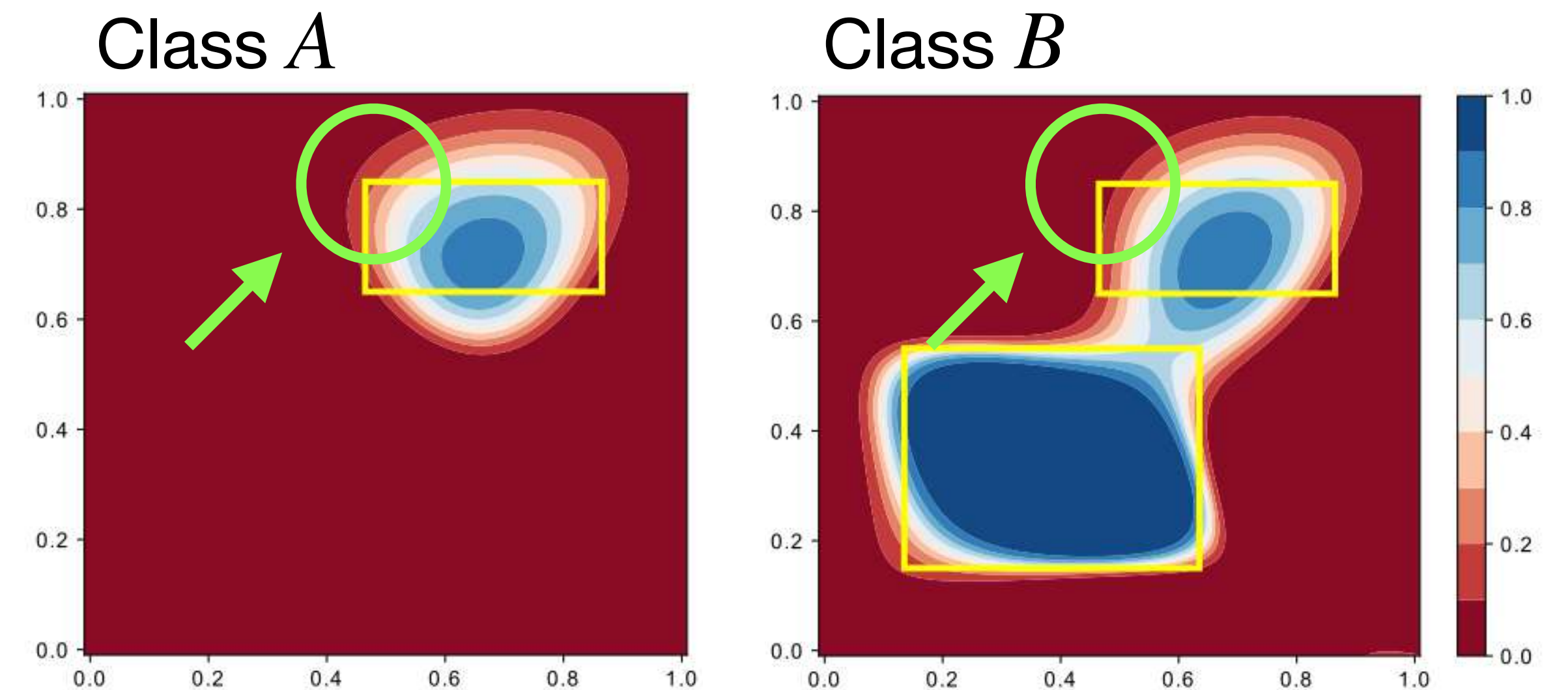


Loss:

$$\mathcal{L} = \mathcal{L}_A + \mathcal{L}_B$$

$$\mathcal{L}_A = -y_A \ln(h_A) - (1 - y_A) \ln(1 - h_A)$$

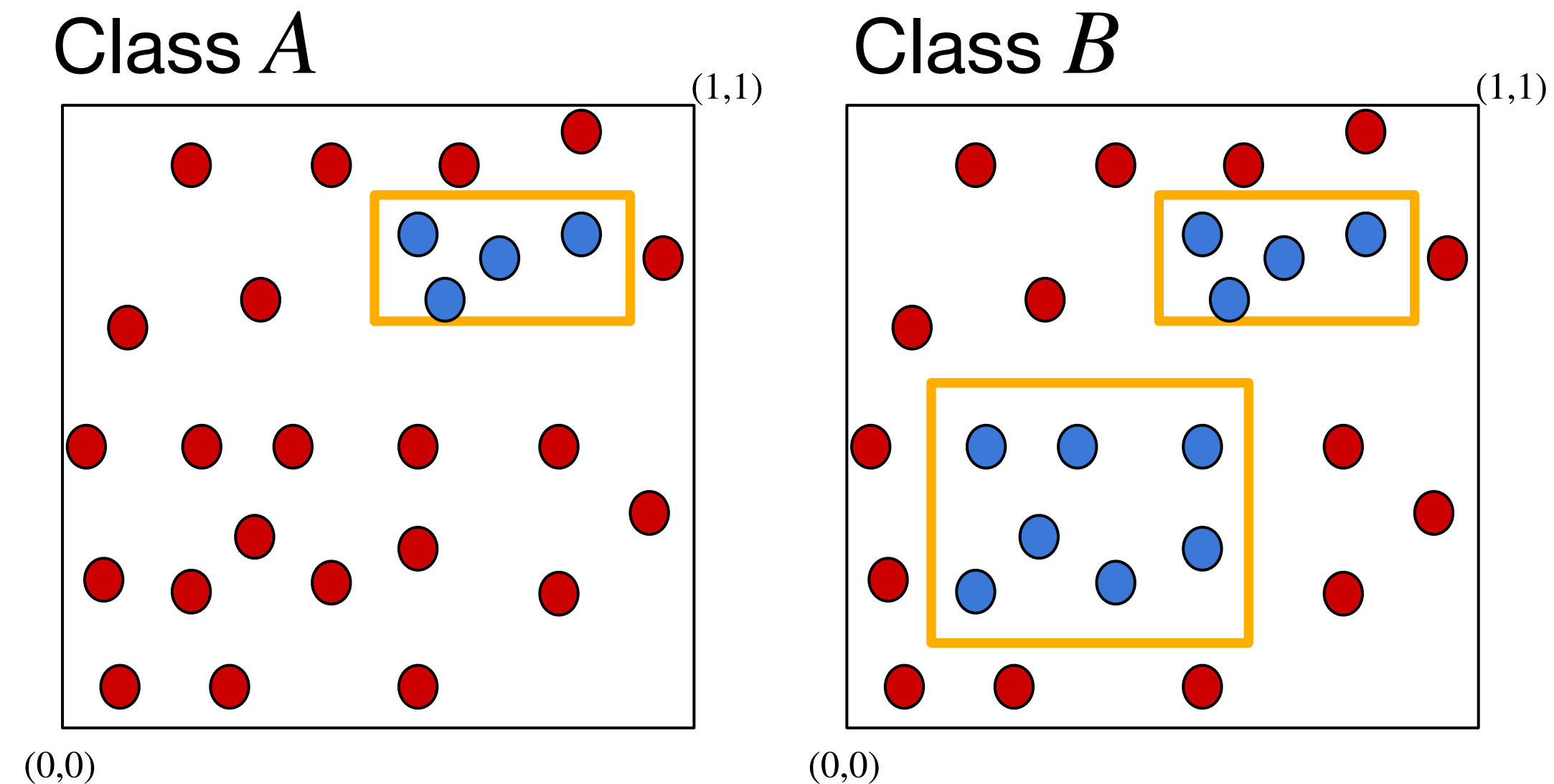
$$\mathcal{L}_B = -y_B \ln(h_B) - (1 - y_B) \ln(1 - h_B)$$



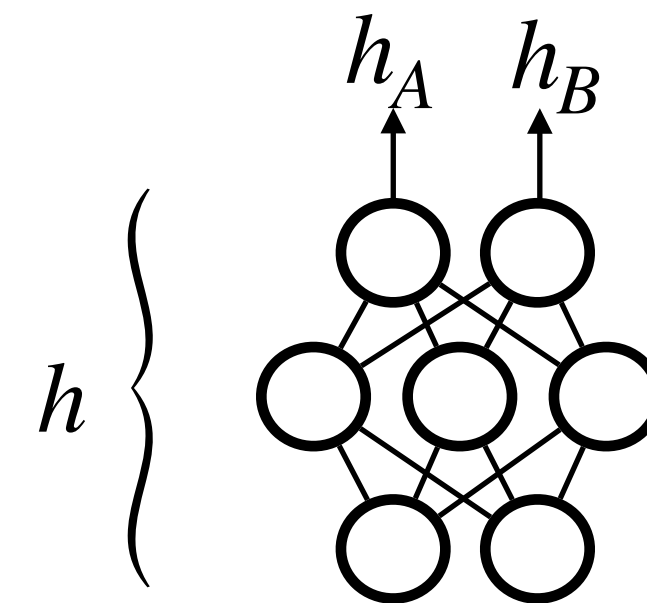


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



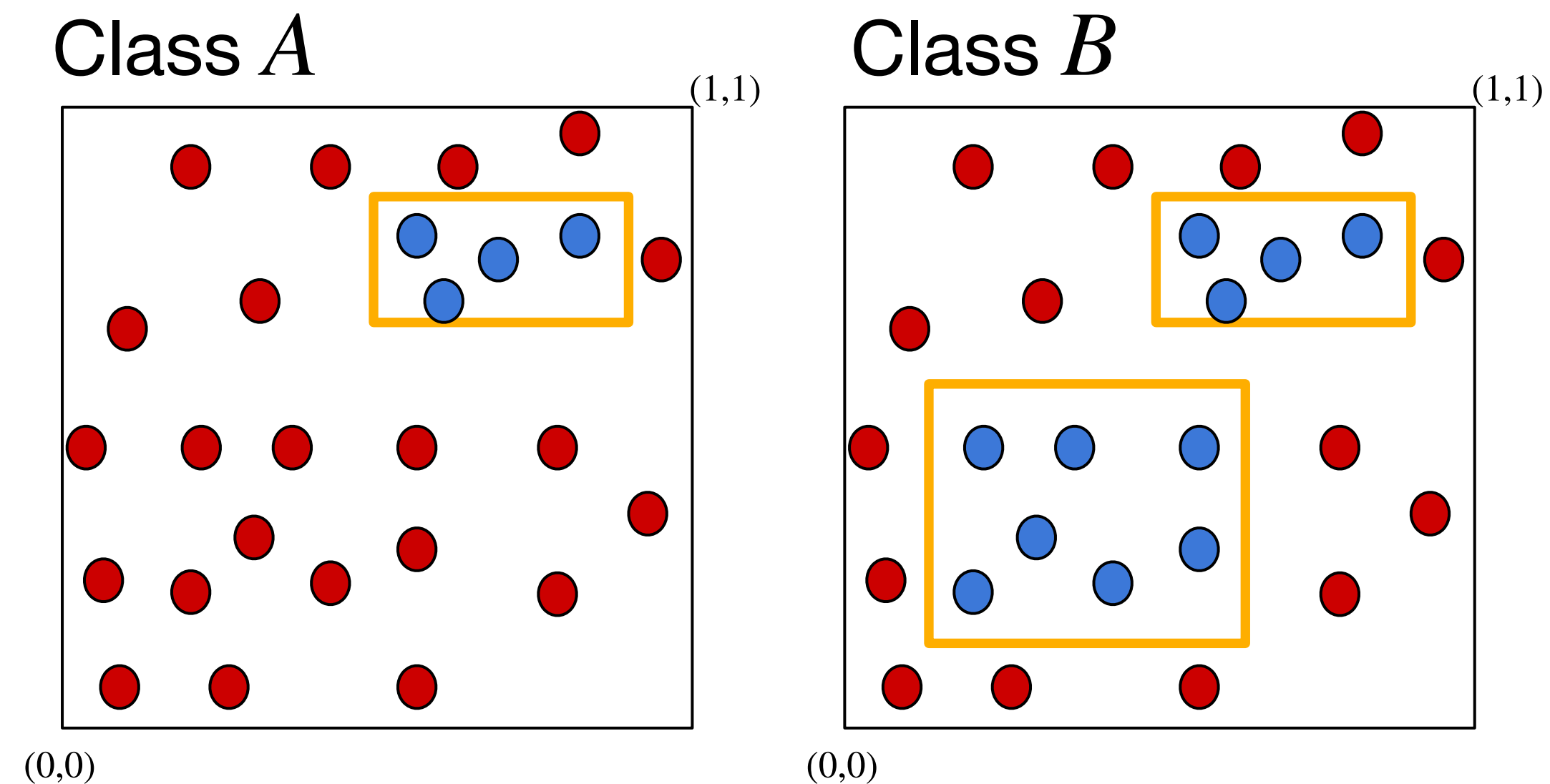
Network:



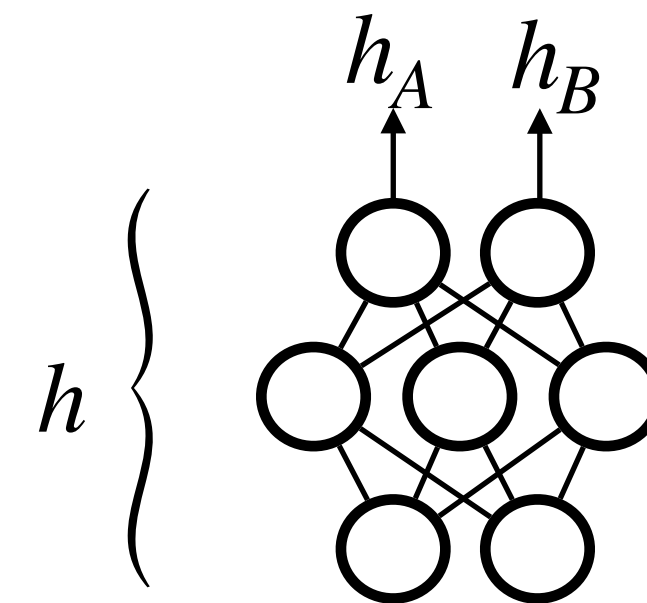


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Network:



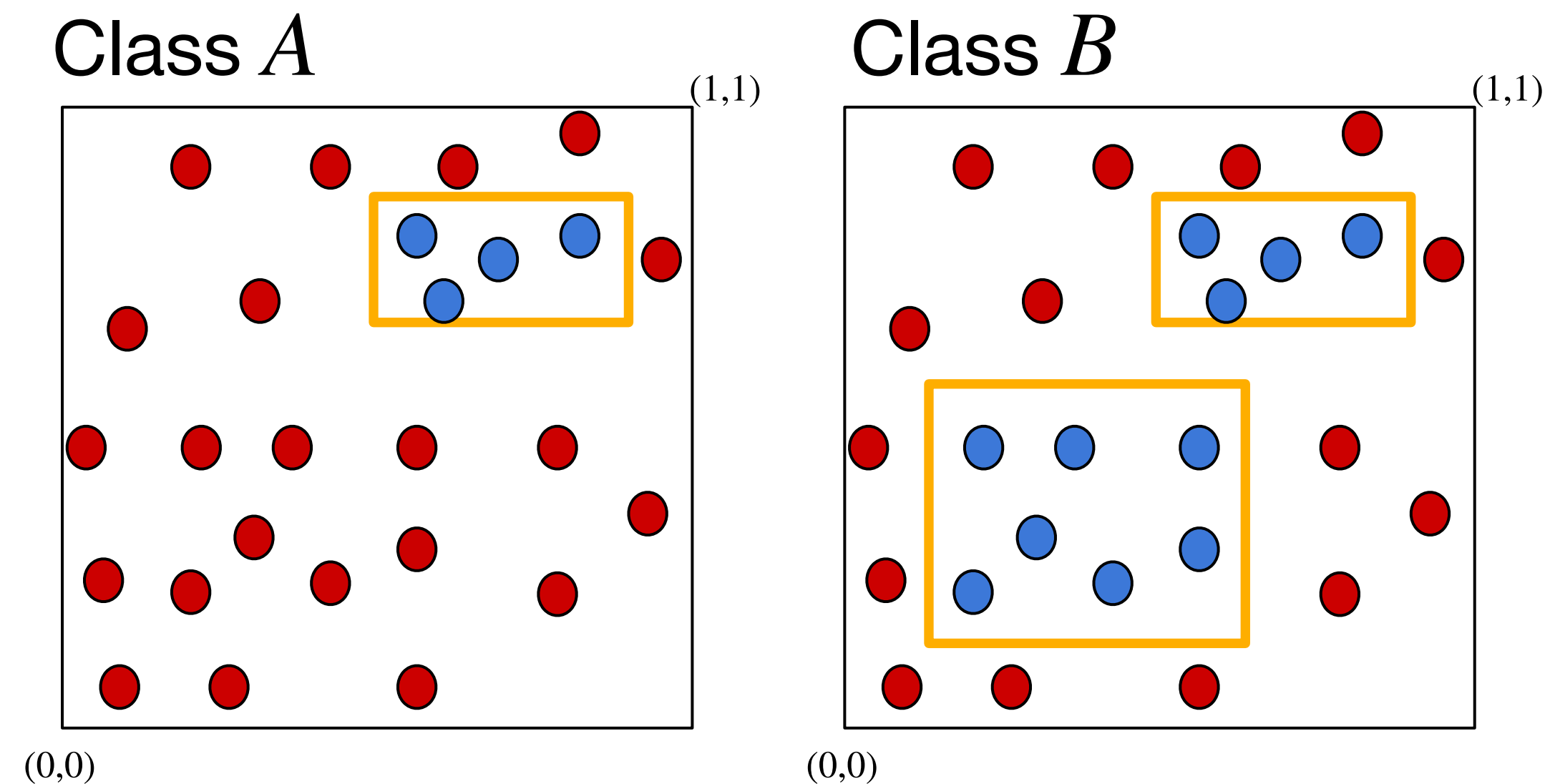
Intuition:

The requirement defines the *safe output space* of the model

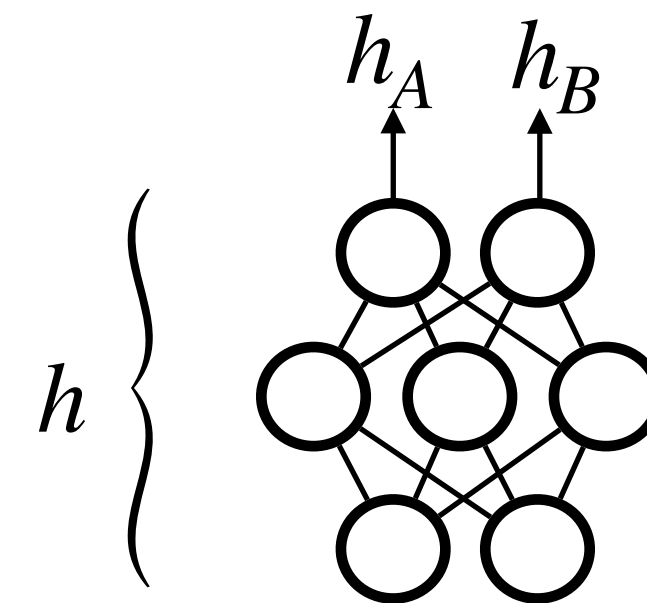


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

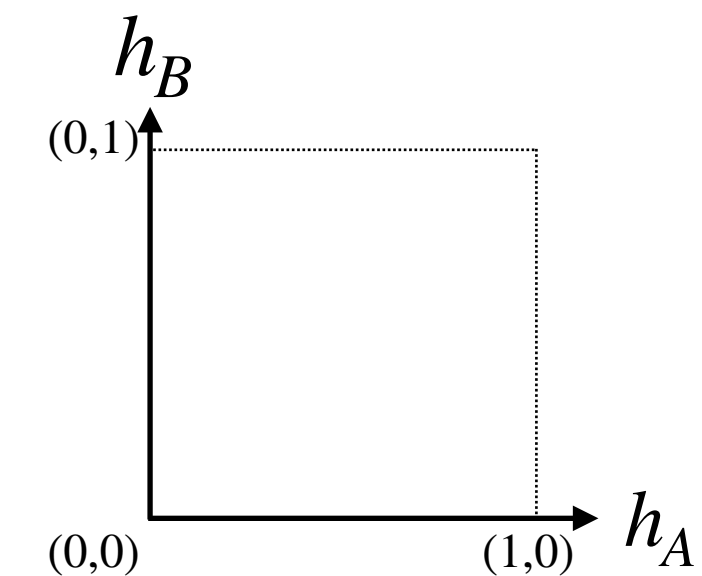


Network:



Intuition:

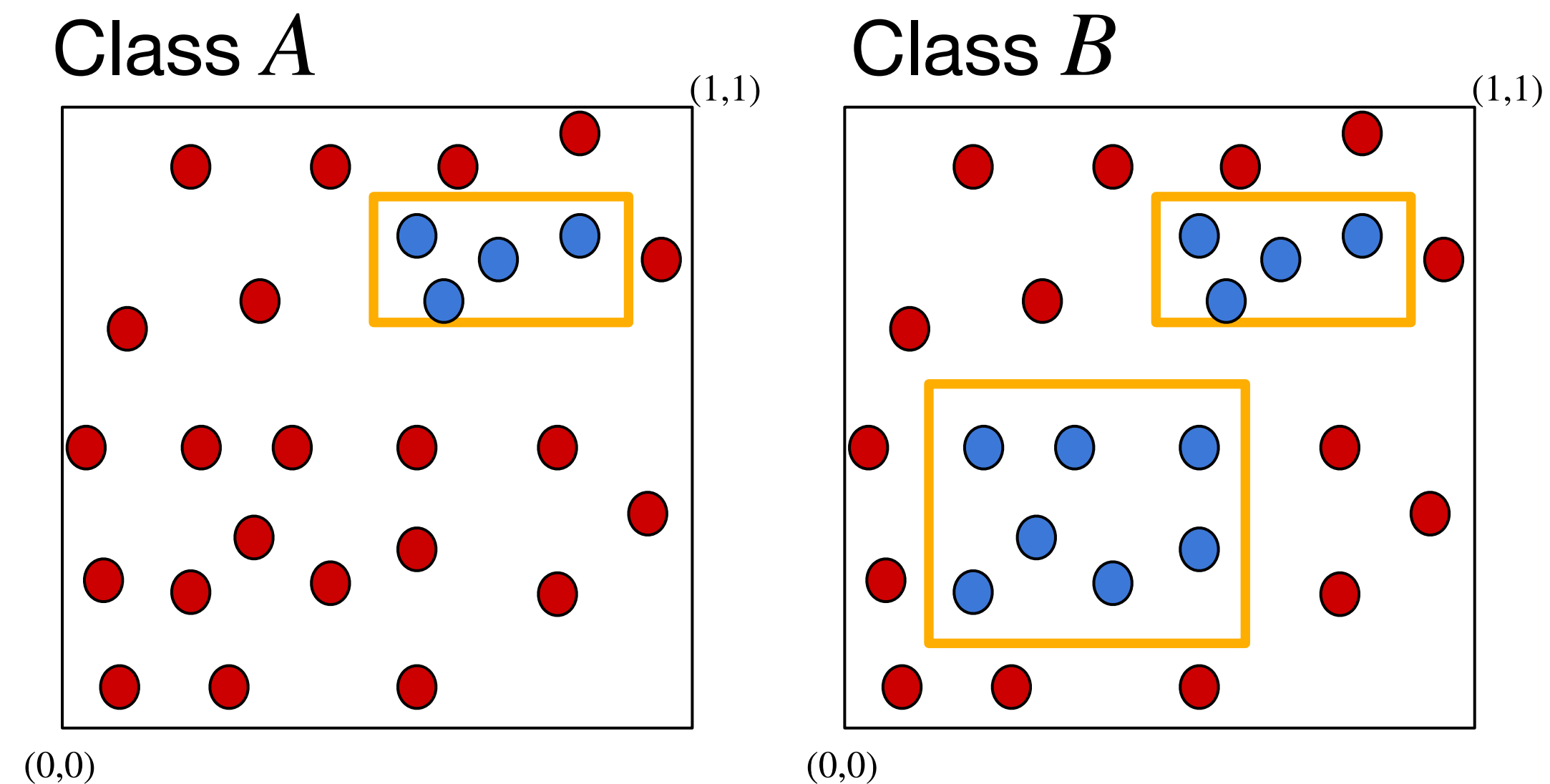
The requirement defines the *safe output space* of the model



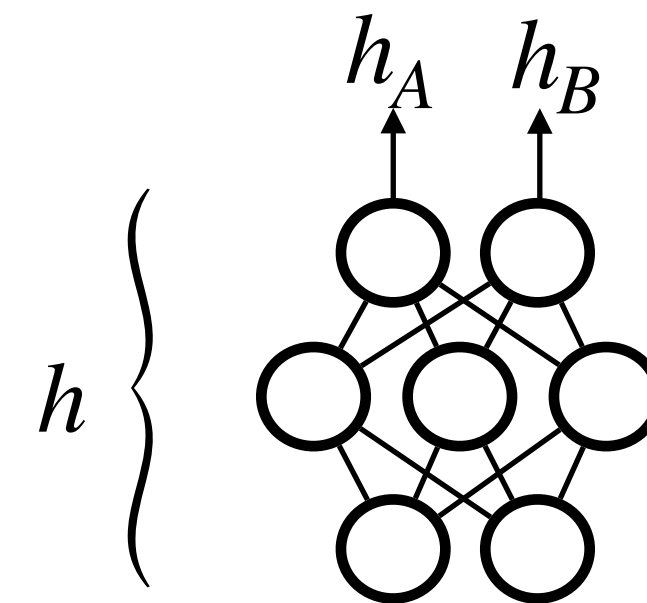


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

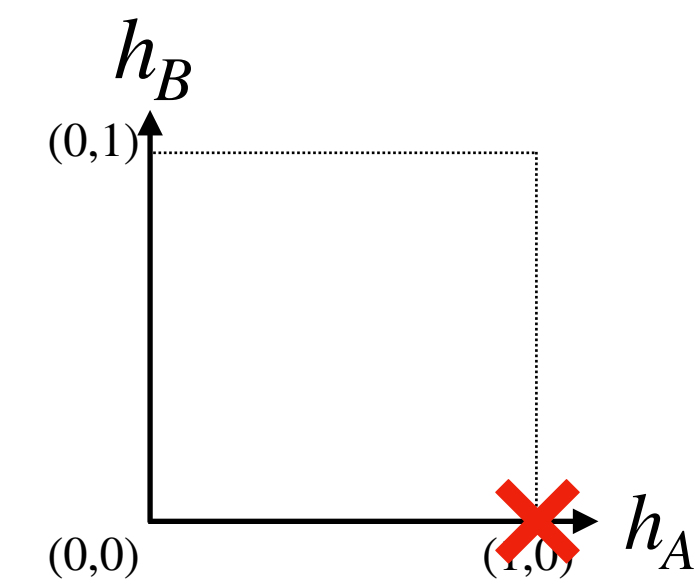


Network:



Intuition:

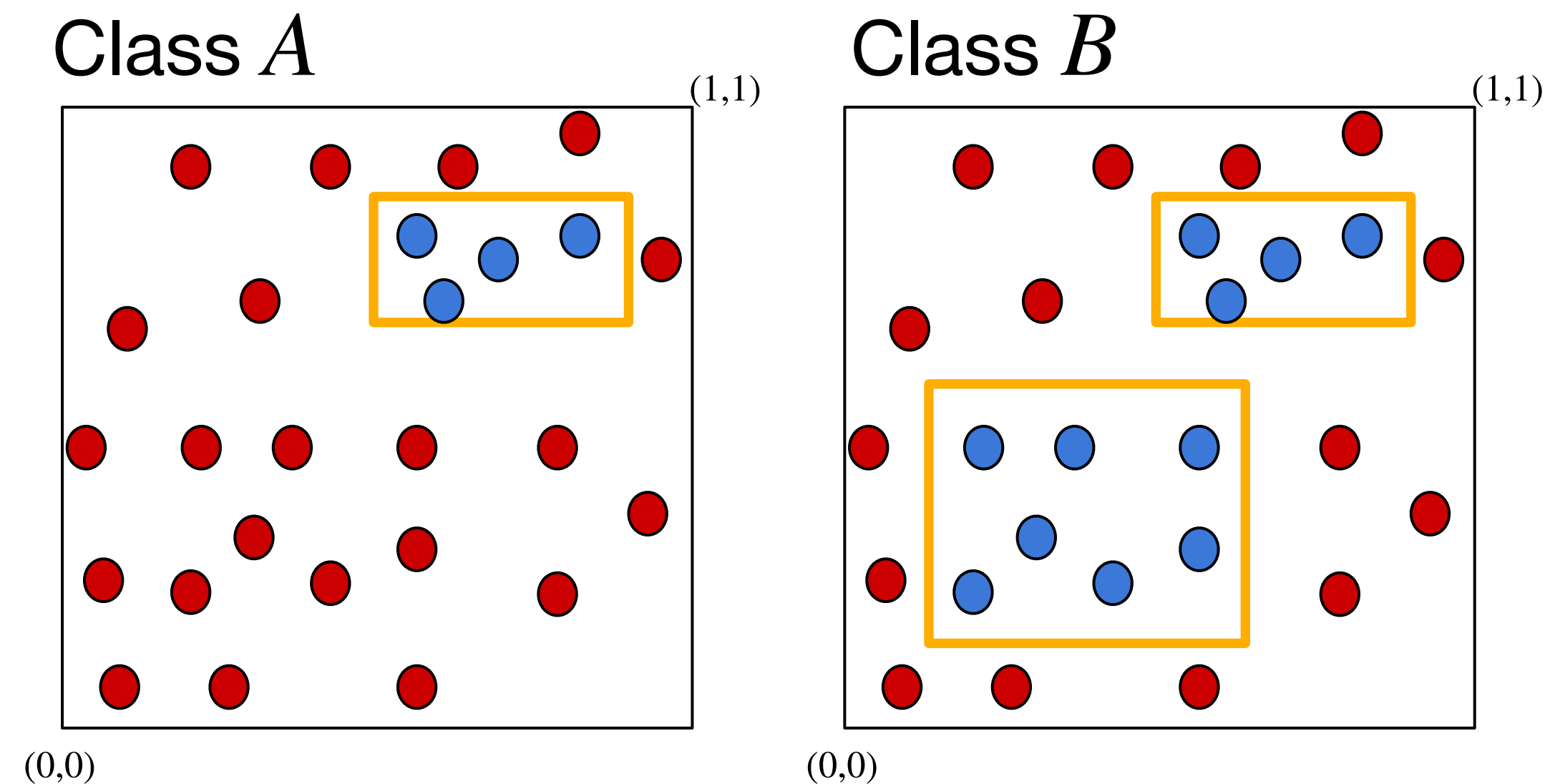
The requirement defines the *safe output space* of the model



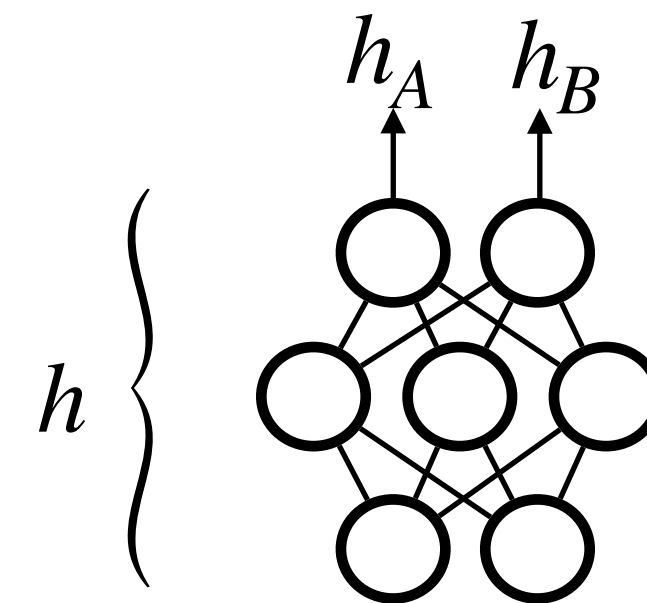


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

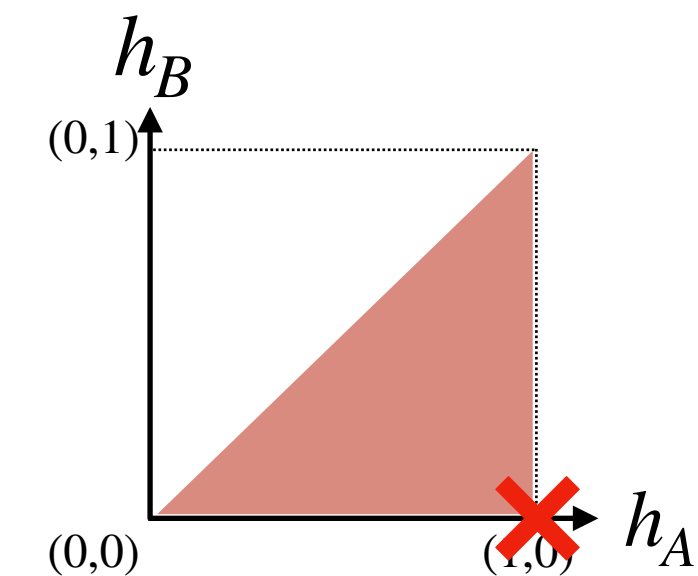


Network:



Intuition:

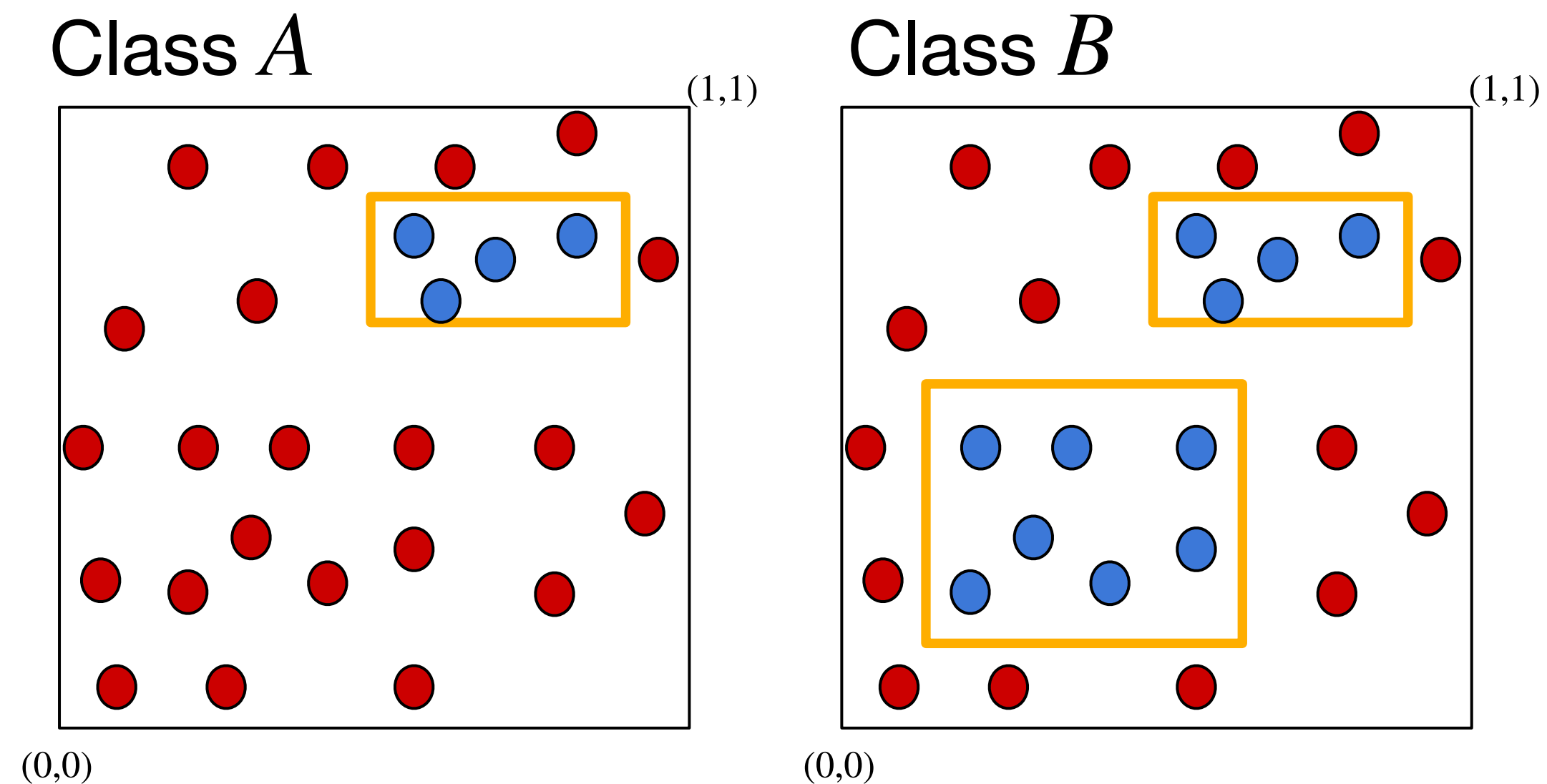
The requirement defines the *safe output space* of the model



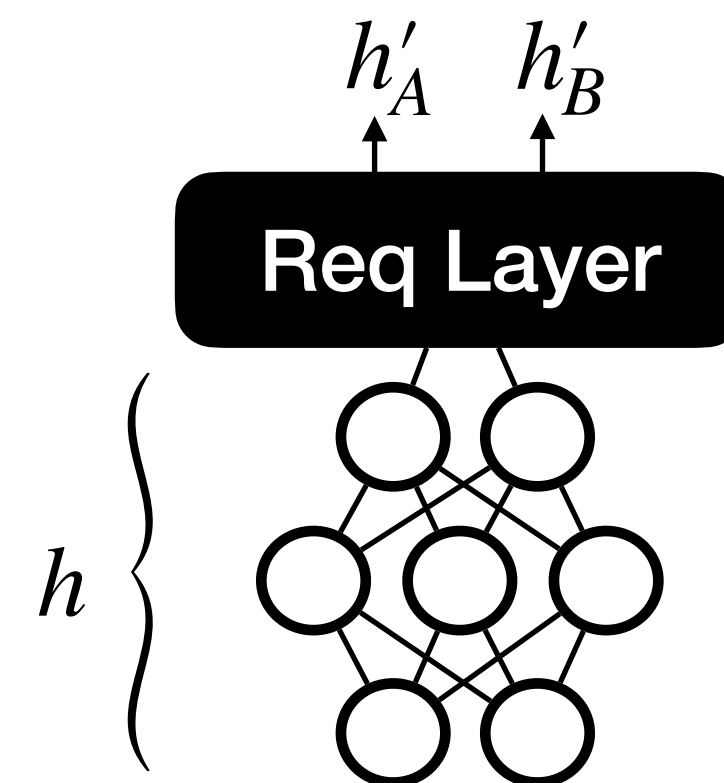


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

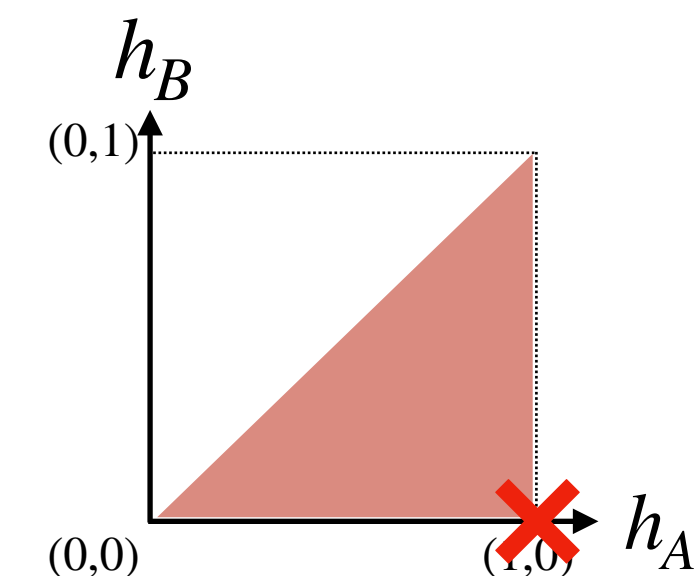


Network:



Intuition:

The requirement defines the *safe output space* of the model



Layer:

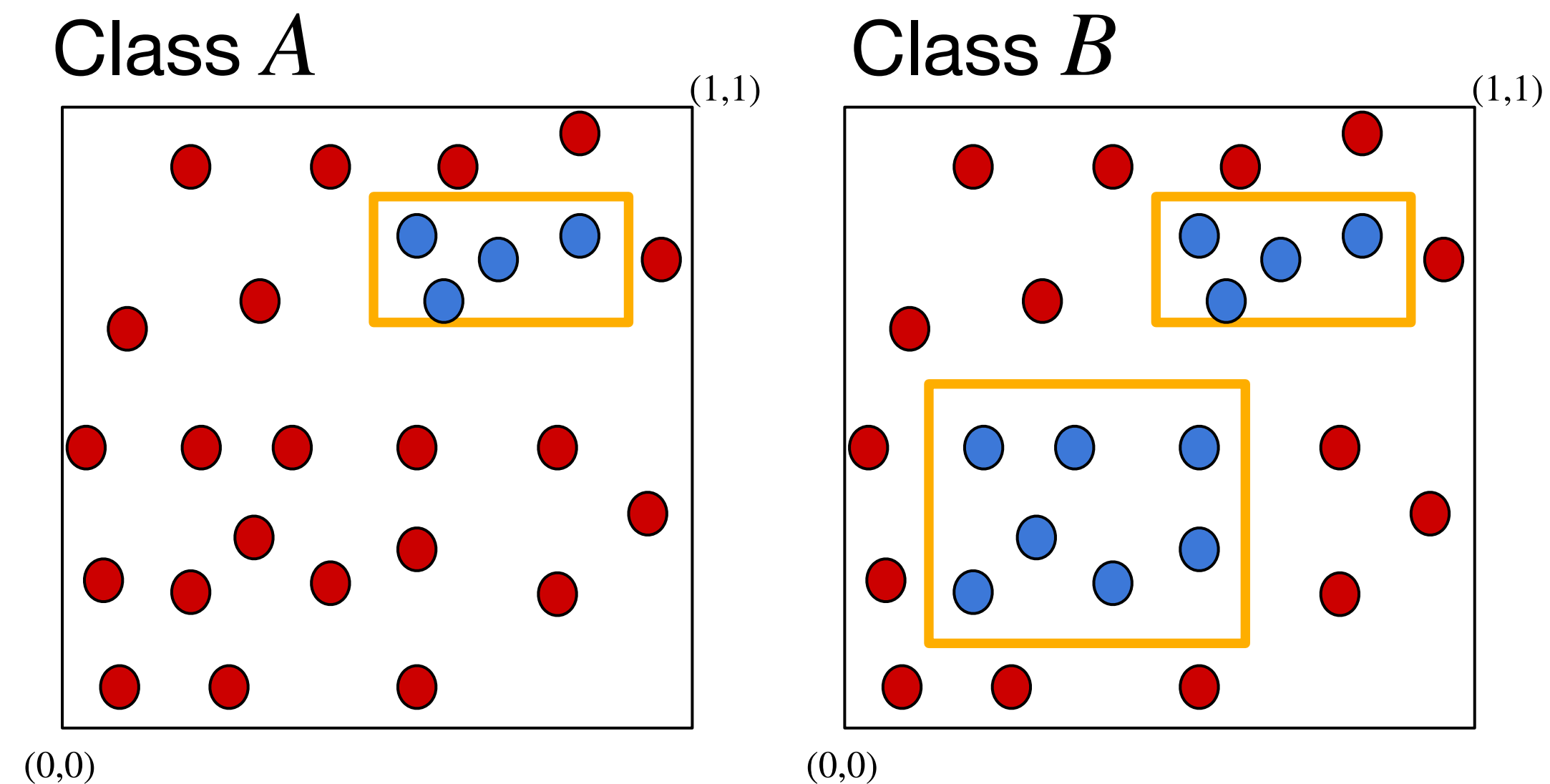
$$h'_A = h_A$$

$$h'_B = \max(h_A, h_B)$$

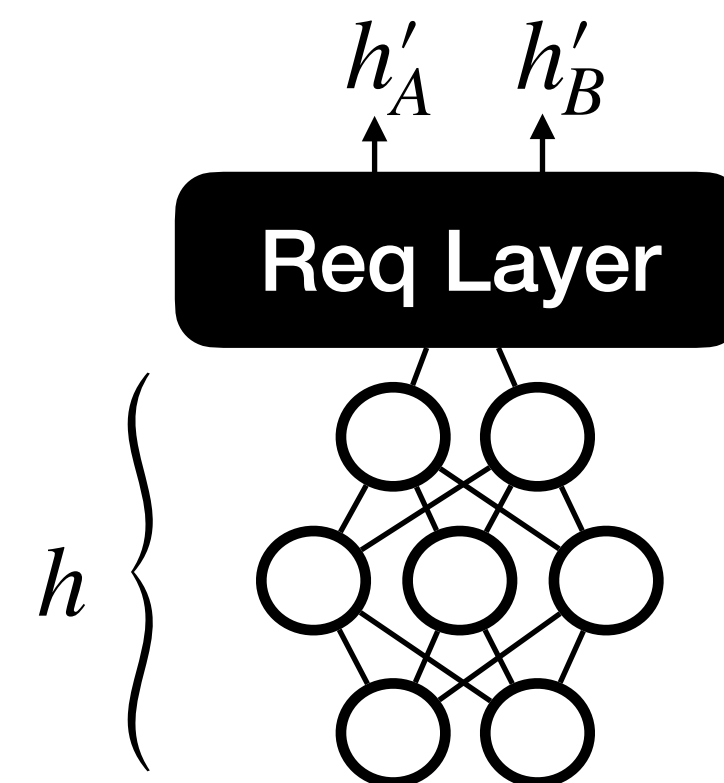


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

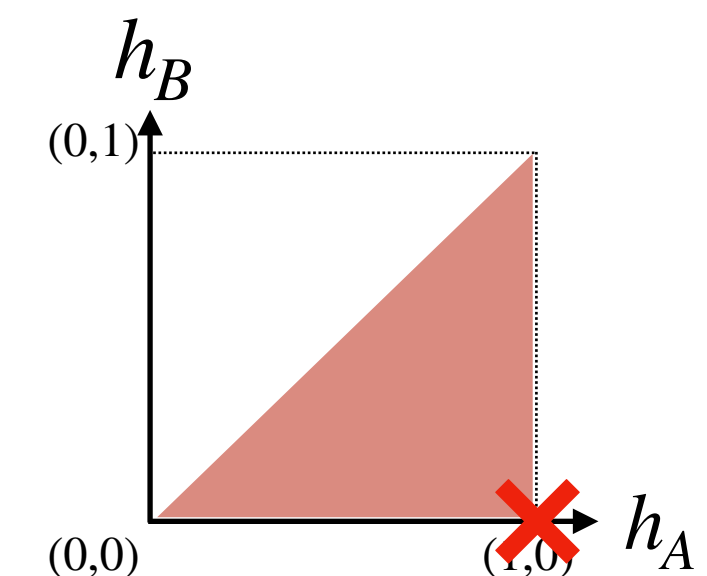


Network:



Intuition:

The requirement defines the *safe output space* of the model



Layer:

$$h'_A = h_A$$

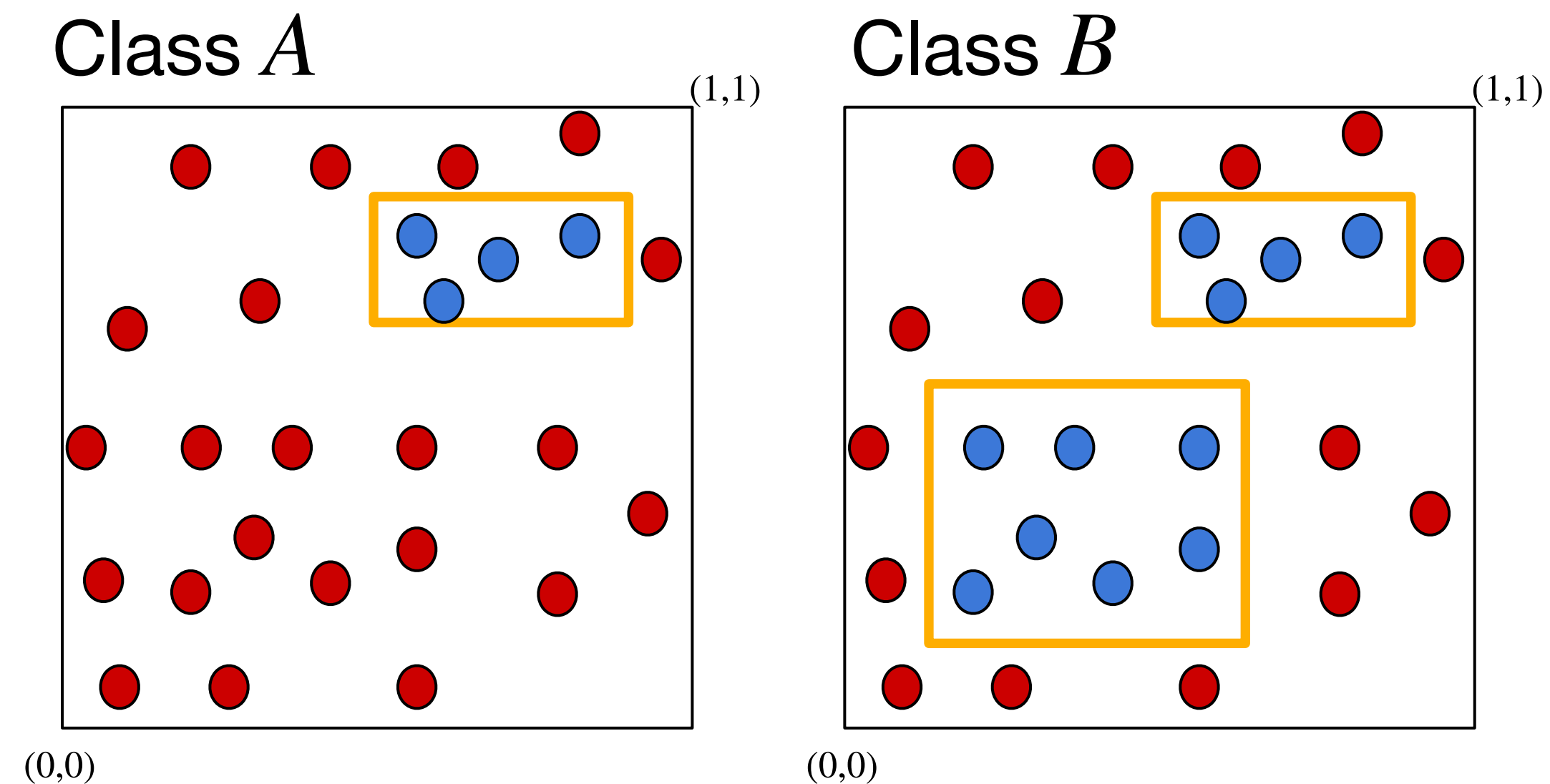
$$h'_B = \max(h_A, h_B)$$

The final output is now
guaranteed to be compliant
with $A \rightarrow B$

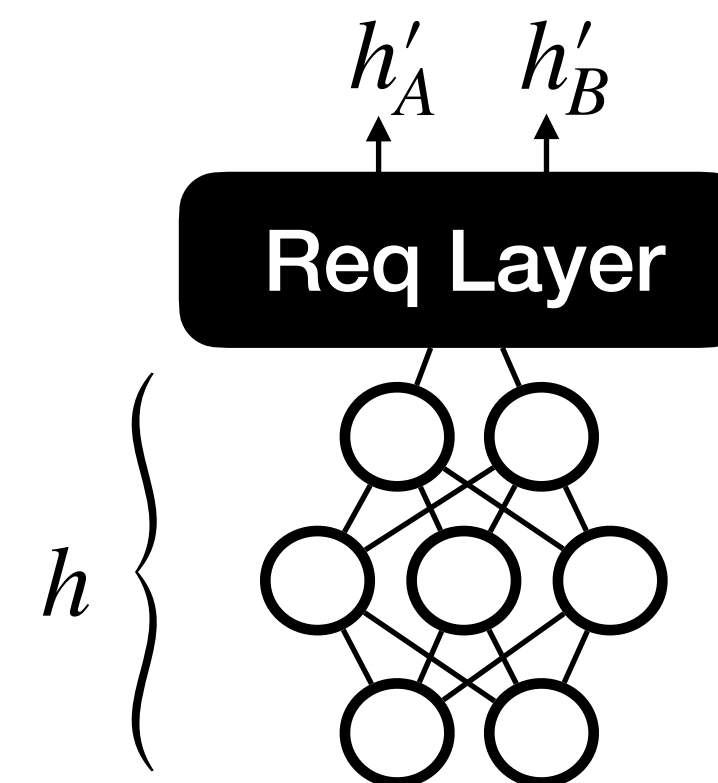


How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:

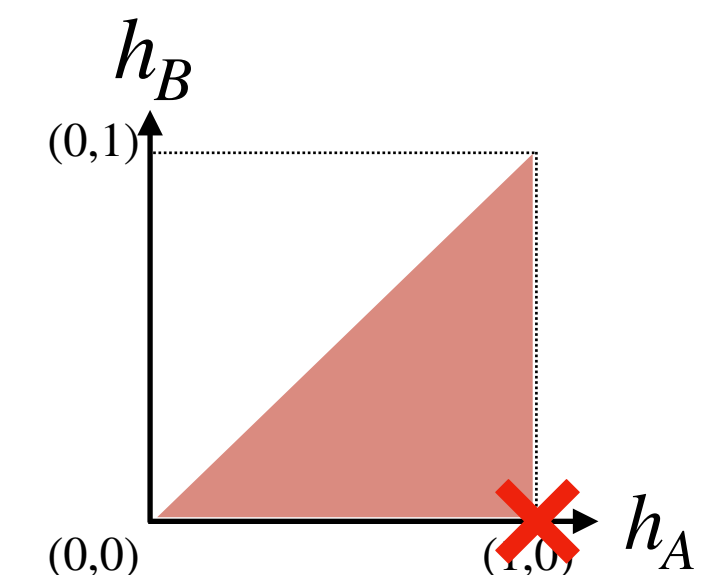


Network:



Intuition:

The requirement defines the *safe output space* of the model



Layer:

$$h'_A = h_A$$

$$h'_B = \max(h_A, h_B)$$

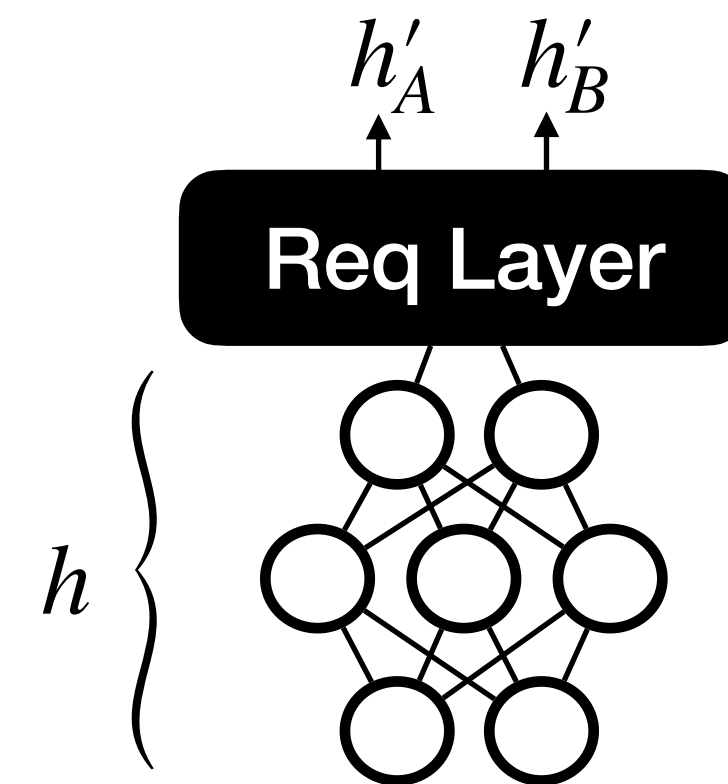
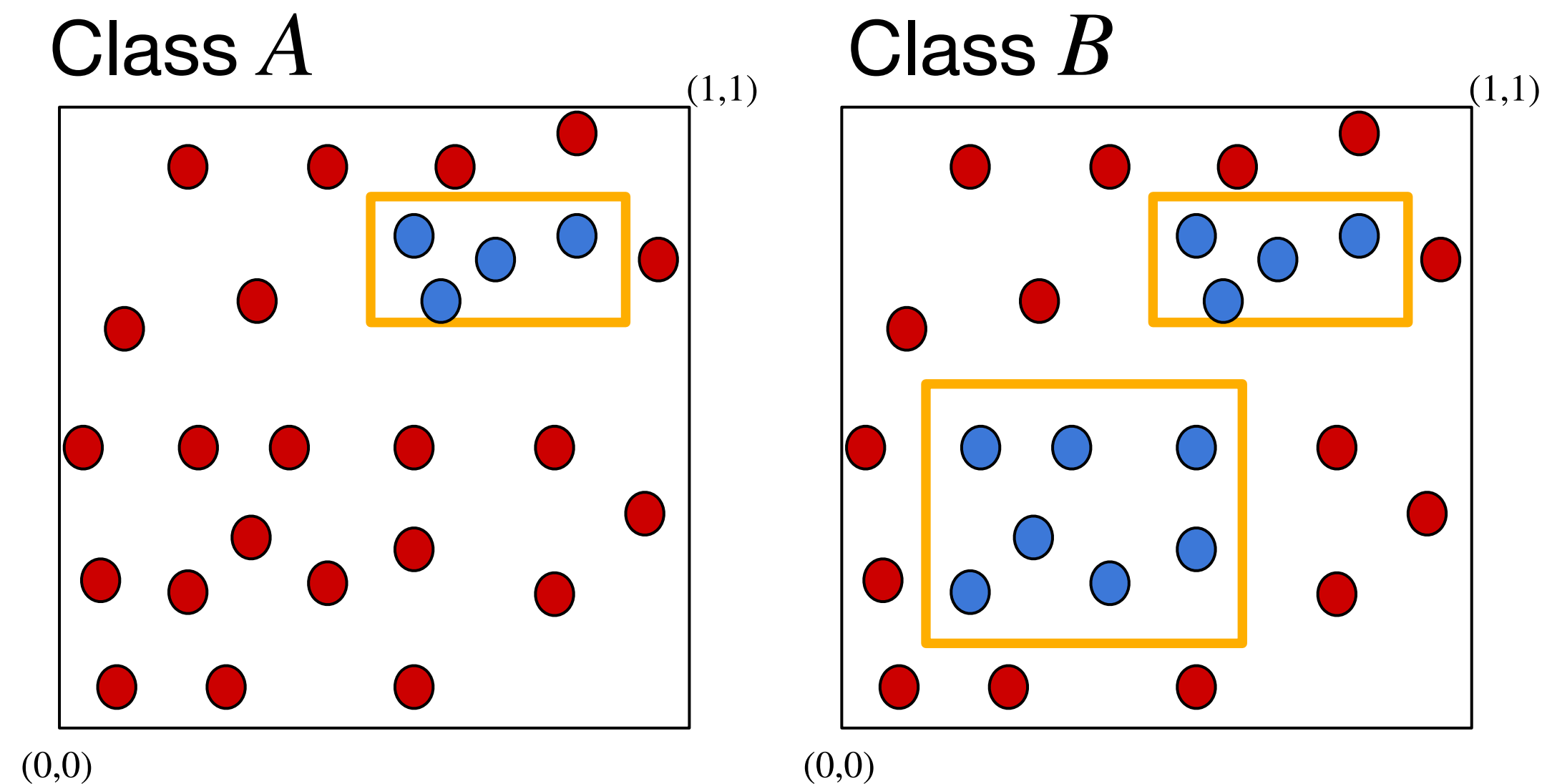
The final output is now
guaranteed to be compliant
with $A \rightarrow B$

The network is now aware
that A is a sufficient
condition to conclude B



How? A simple example

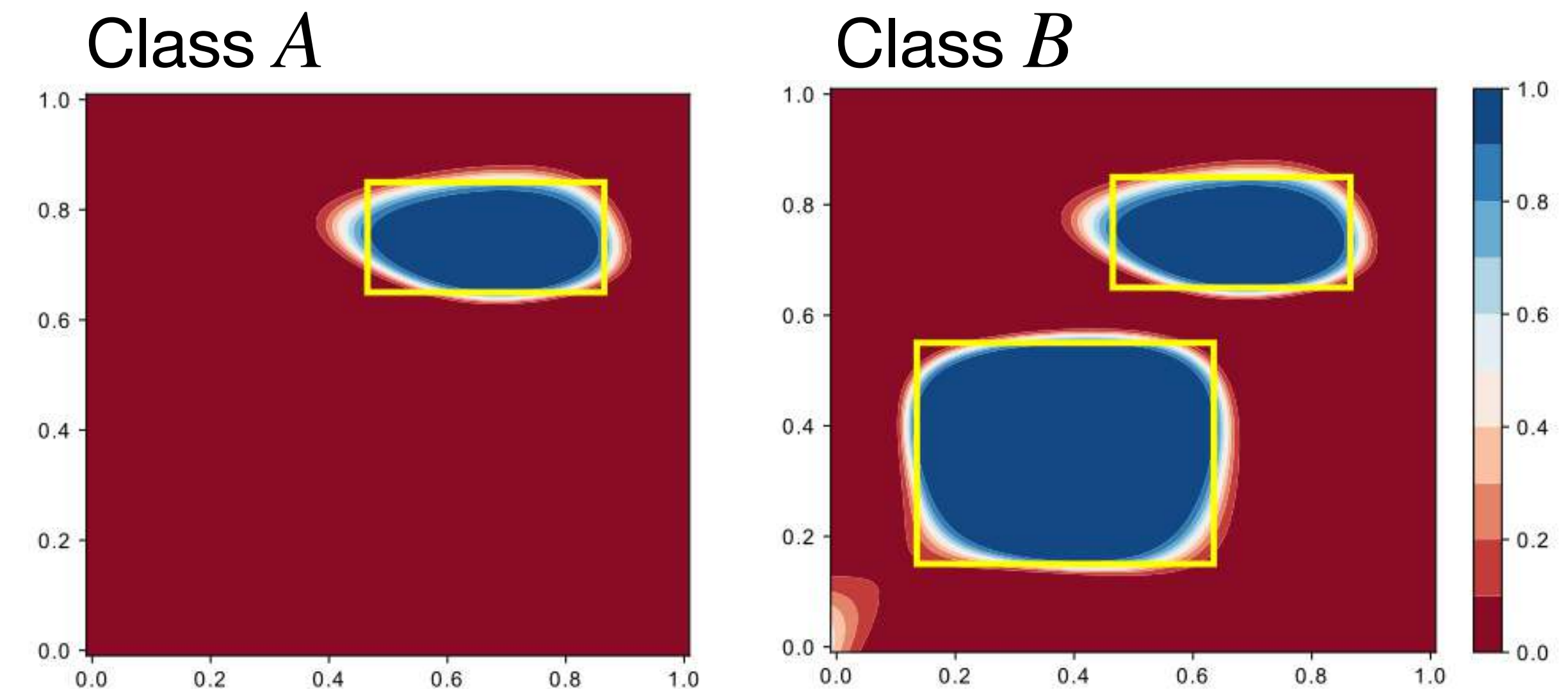
- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Layer:

$$h'_A = h_A$$

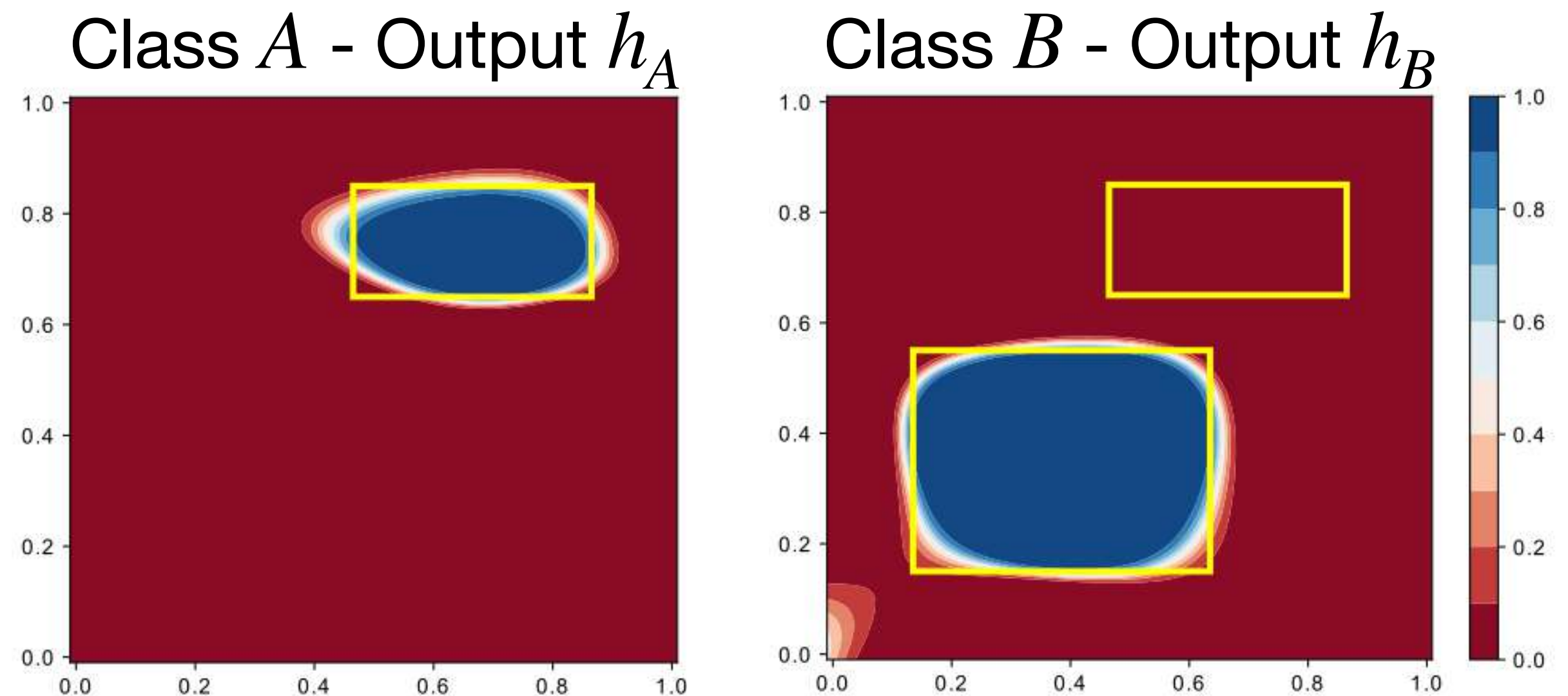
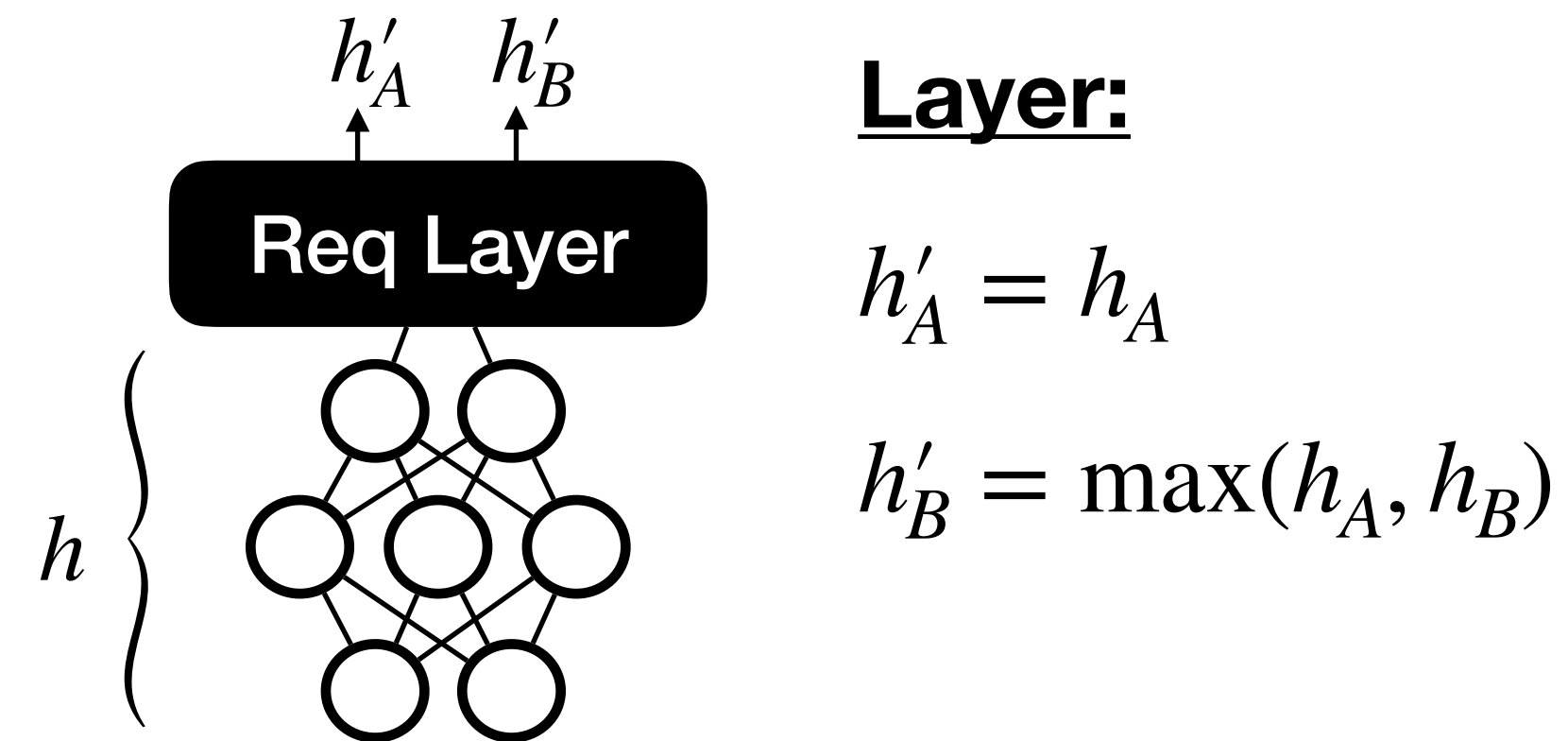
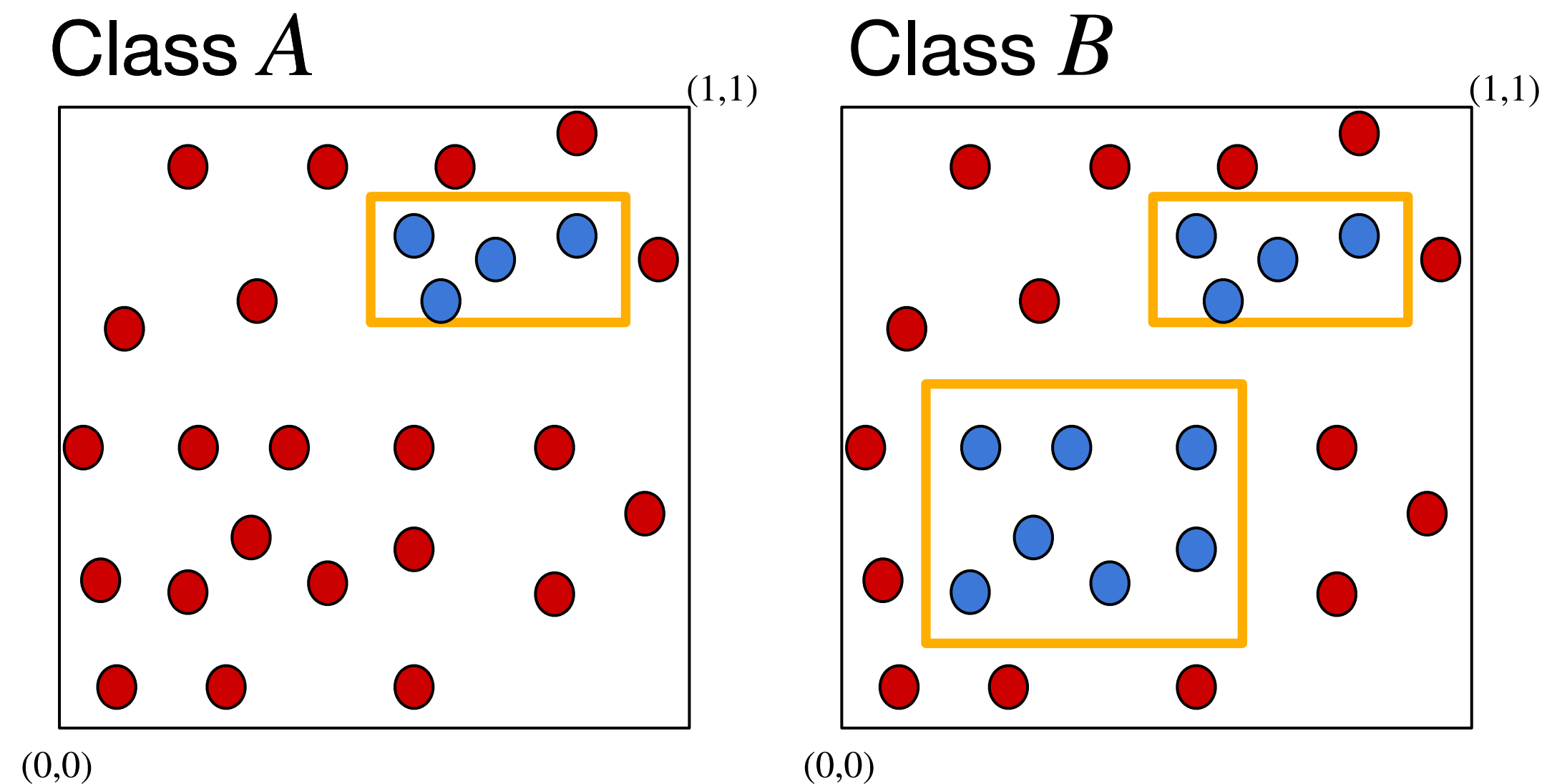
$$h'_B = \max(h_A, h_B)$$





How? A simple example

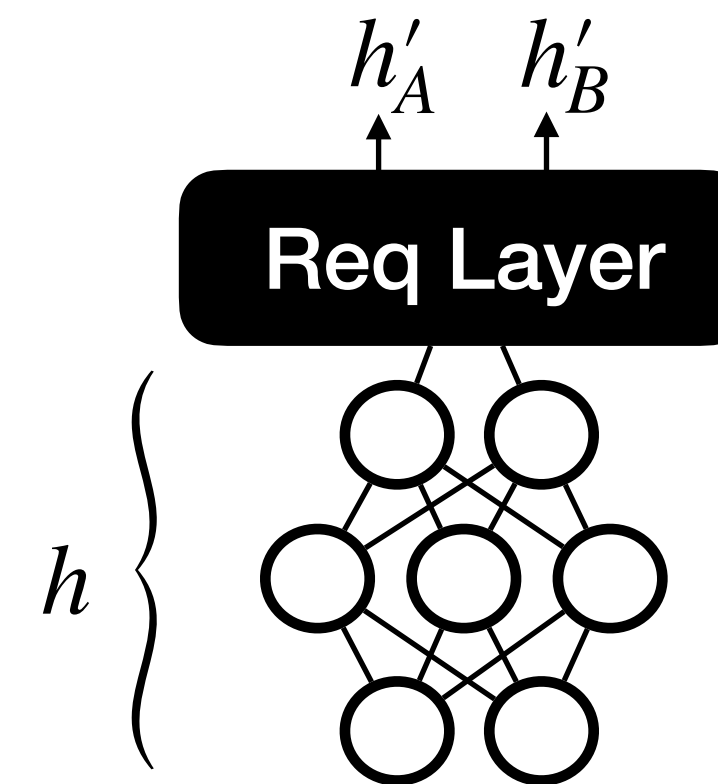
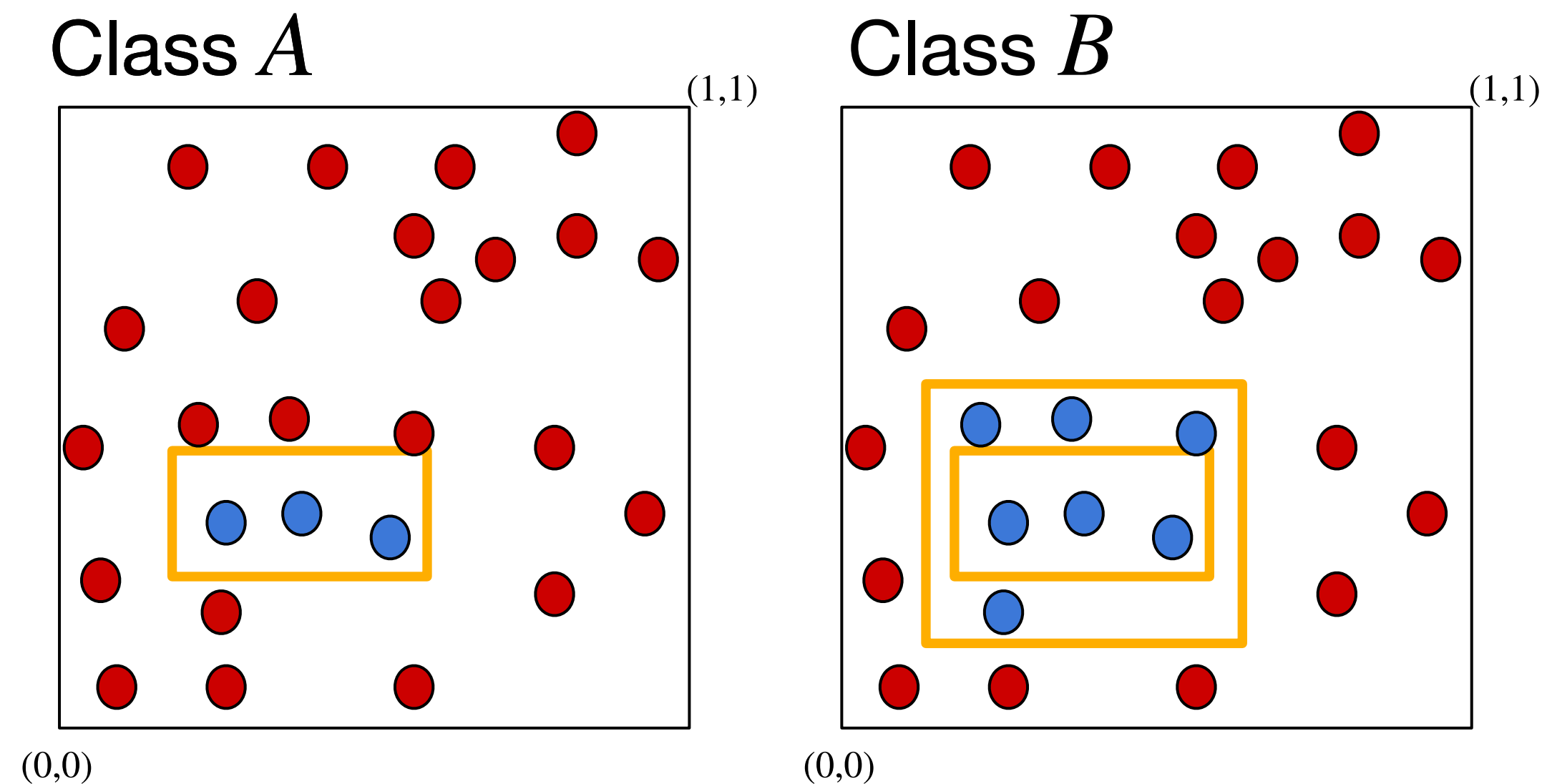
- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:





How? A simple example

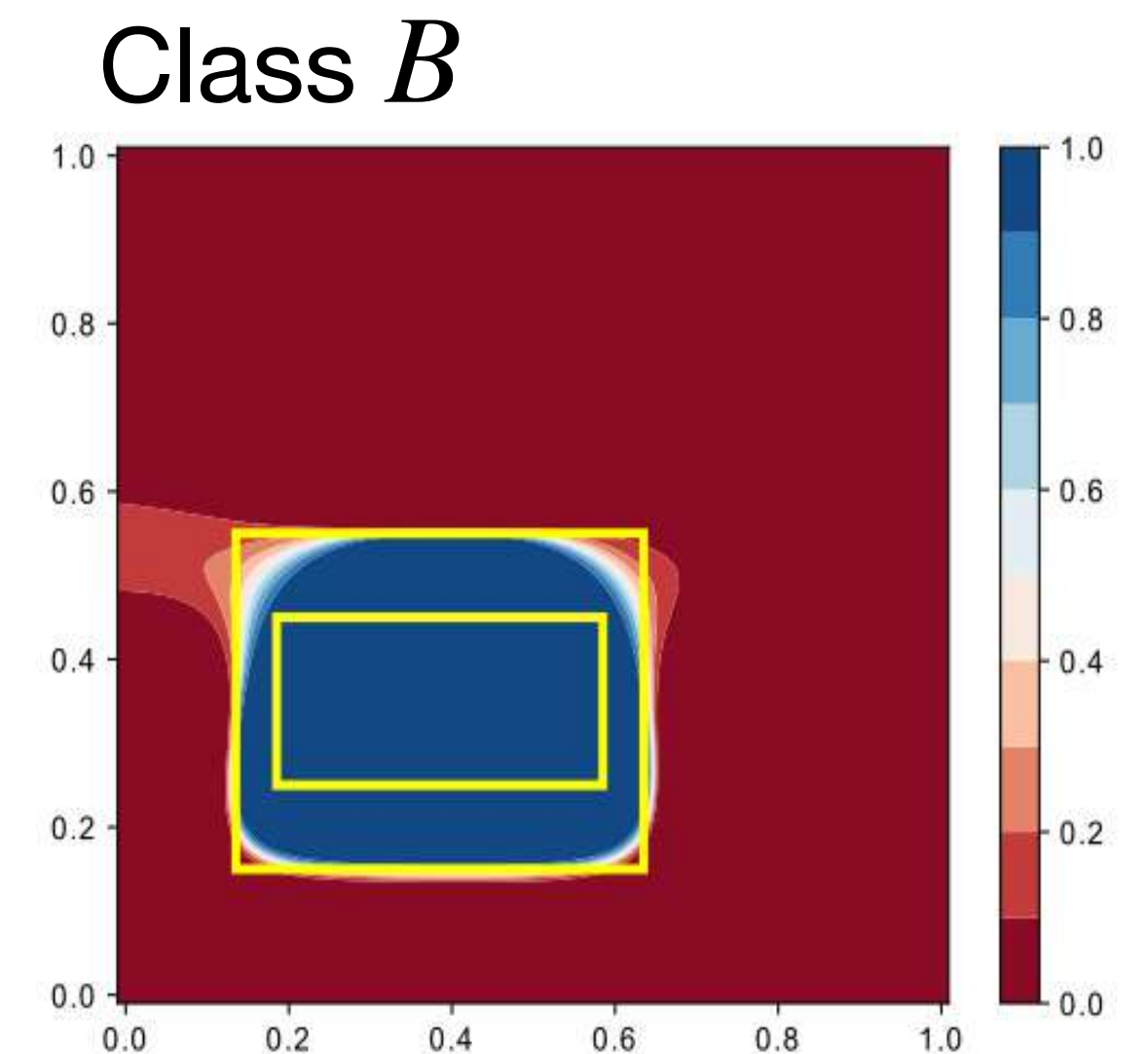
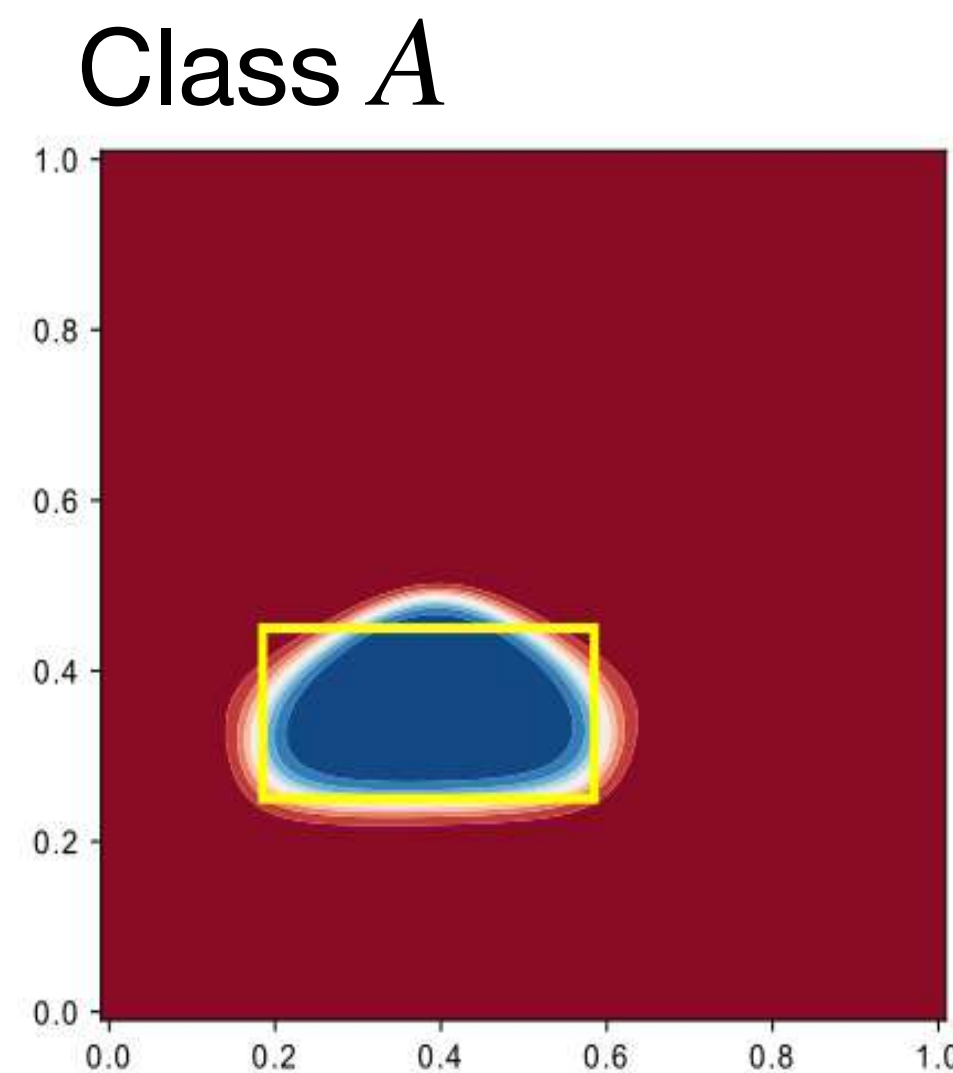
- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:



Layer:

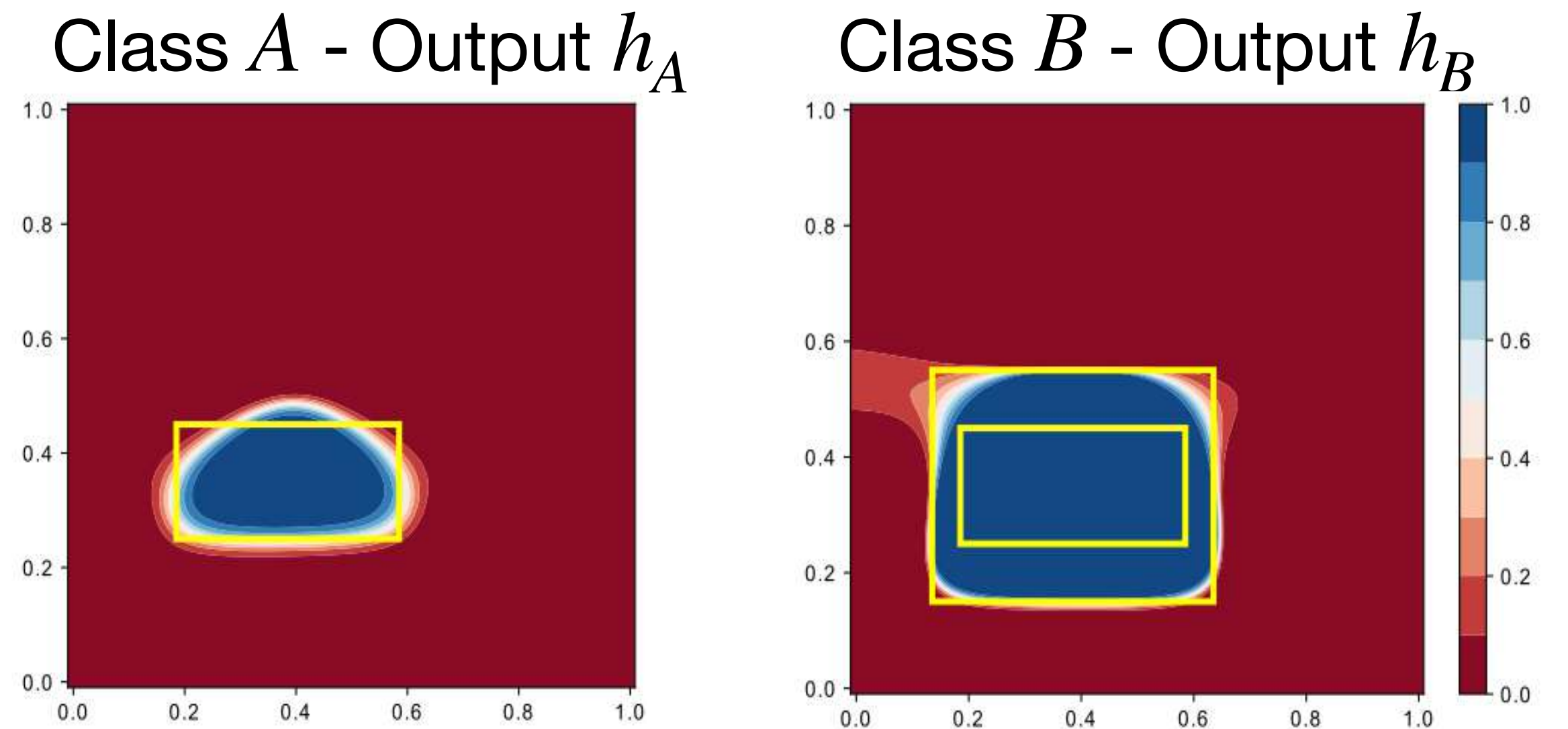
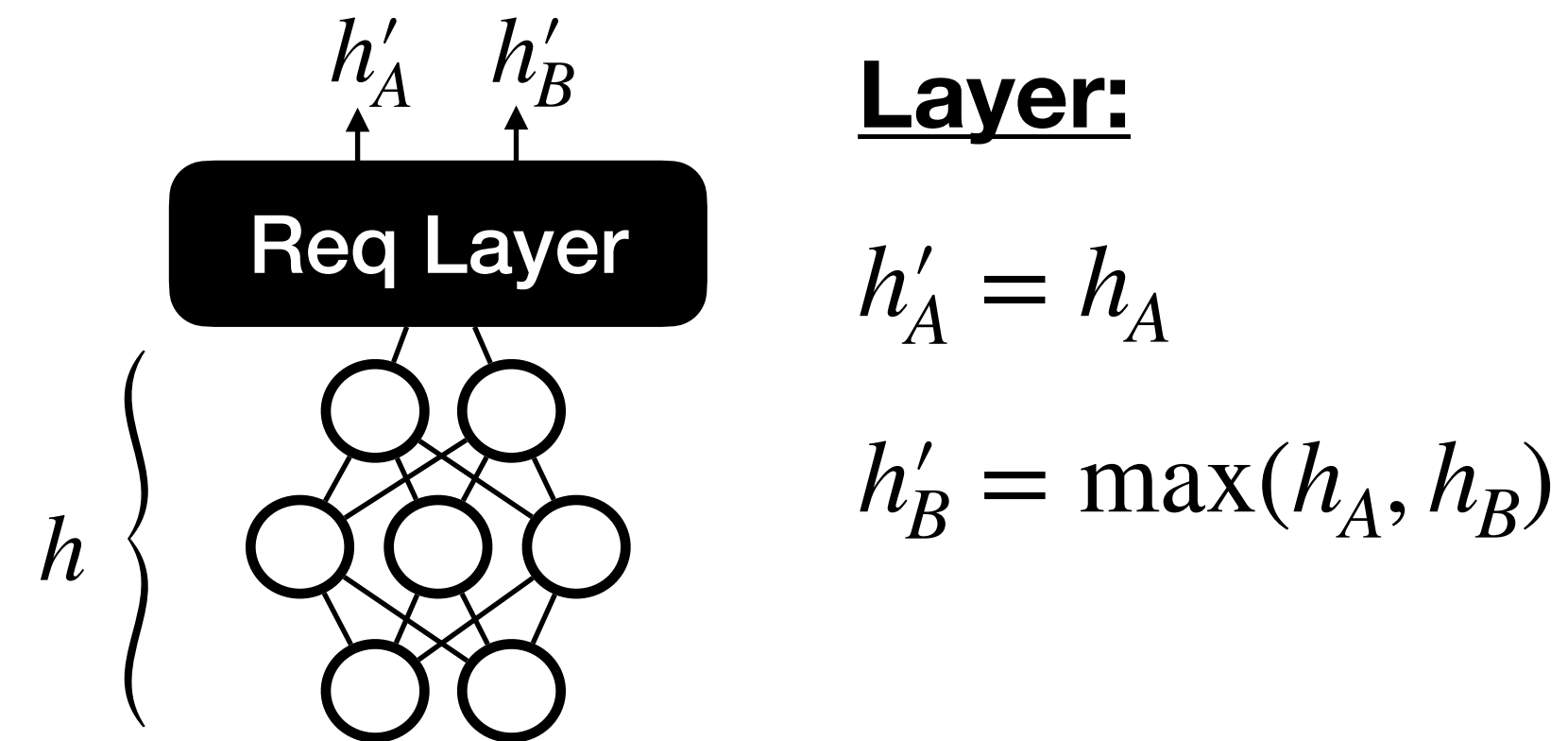
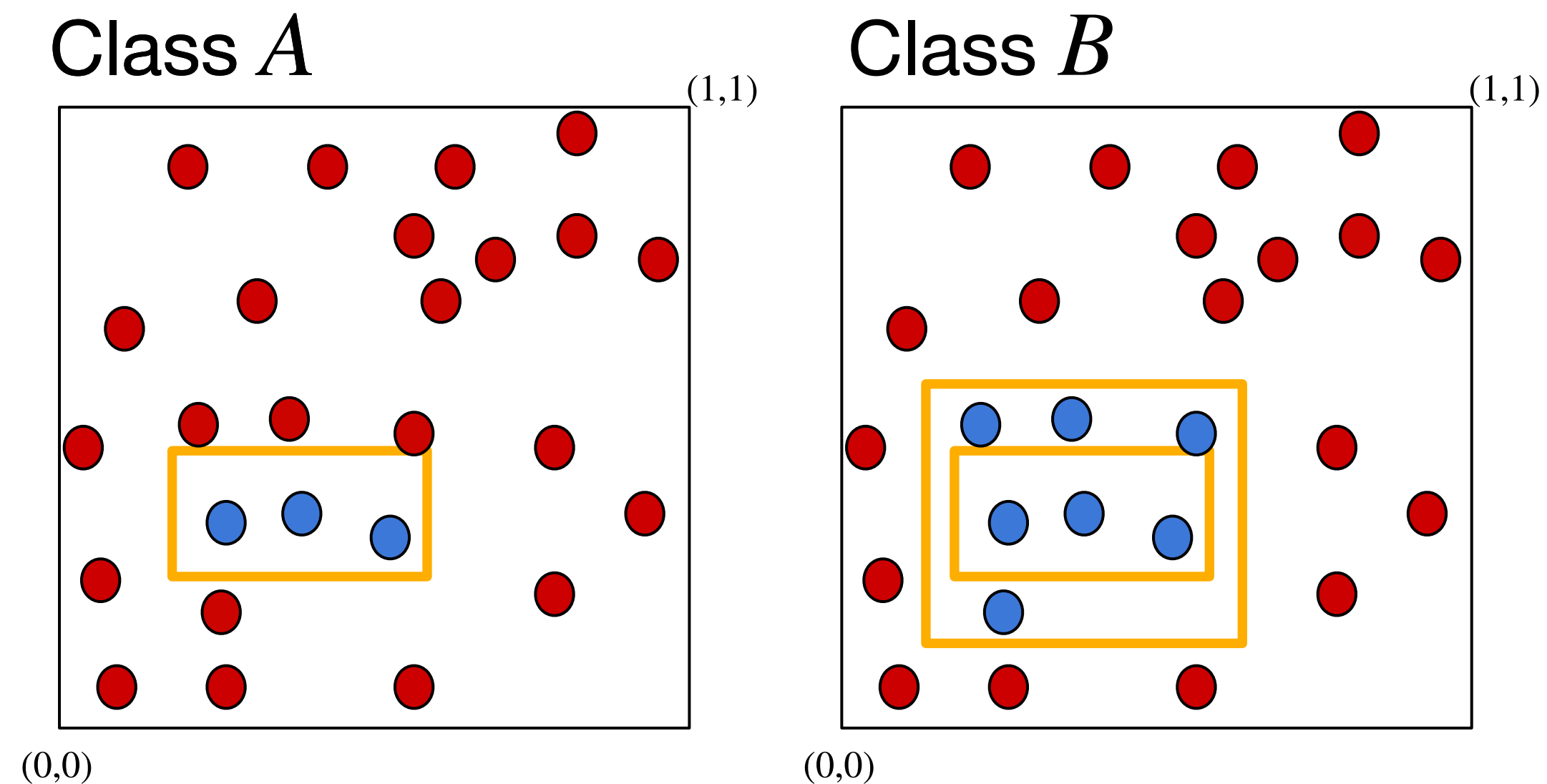
$$h'_A = h_A$$

$$h'_B = \max(h_A, h_B)$$



How? A simple example

- Two classes: A and B
- Requirement: $A \rightarrow B$
- Sample space: $[0,1]^2$
- Dataset:





Beyond Hierarchical Requirements

Intuition:

Single requirement of form: $A_1 \wedge \dots \wedge A_k \wedge \neg A_{k+1} \wedge \dots \wedge \neg A_n \rightarrow A$

It defines a lower bound for the class A equal to

$$lb_A = \min(h_{A_1}, \dots, h_{A_k}, 1 - h_{A_{k+1}}, \dots, 1 - h_{A_n})$$



Beyond Hierarchical Requirements

Intuition:

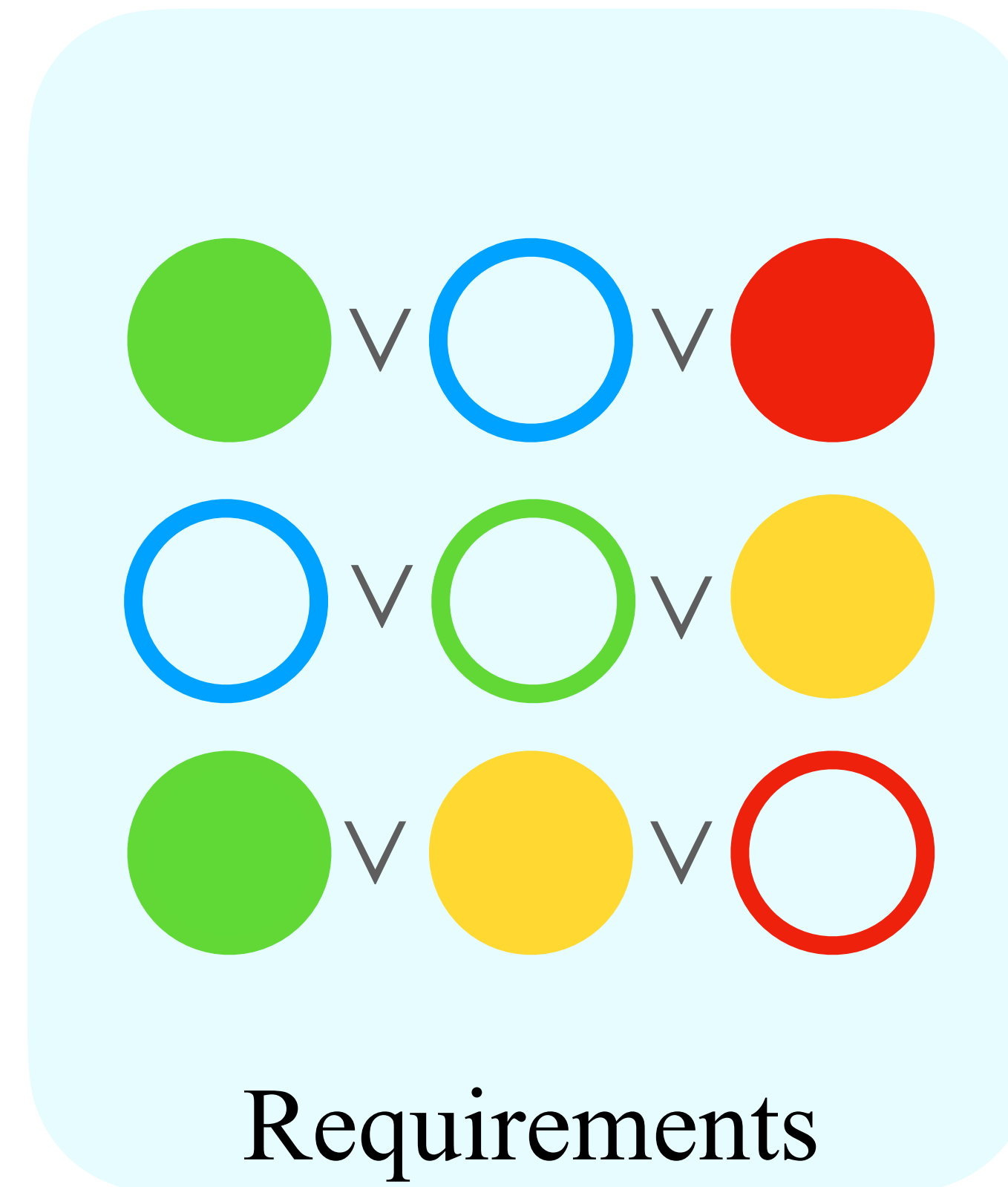
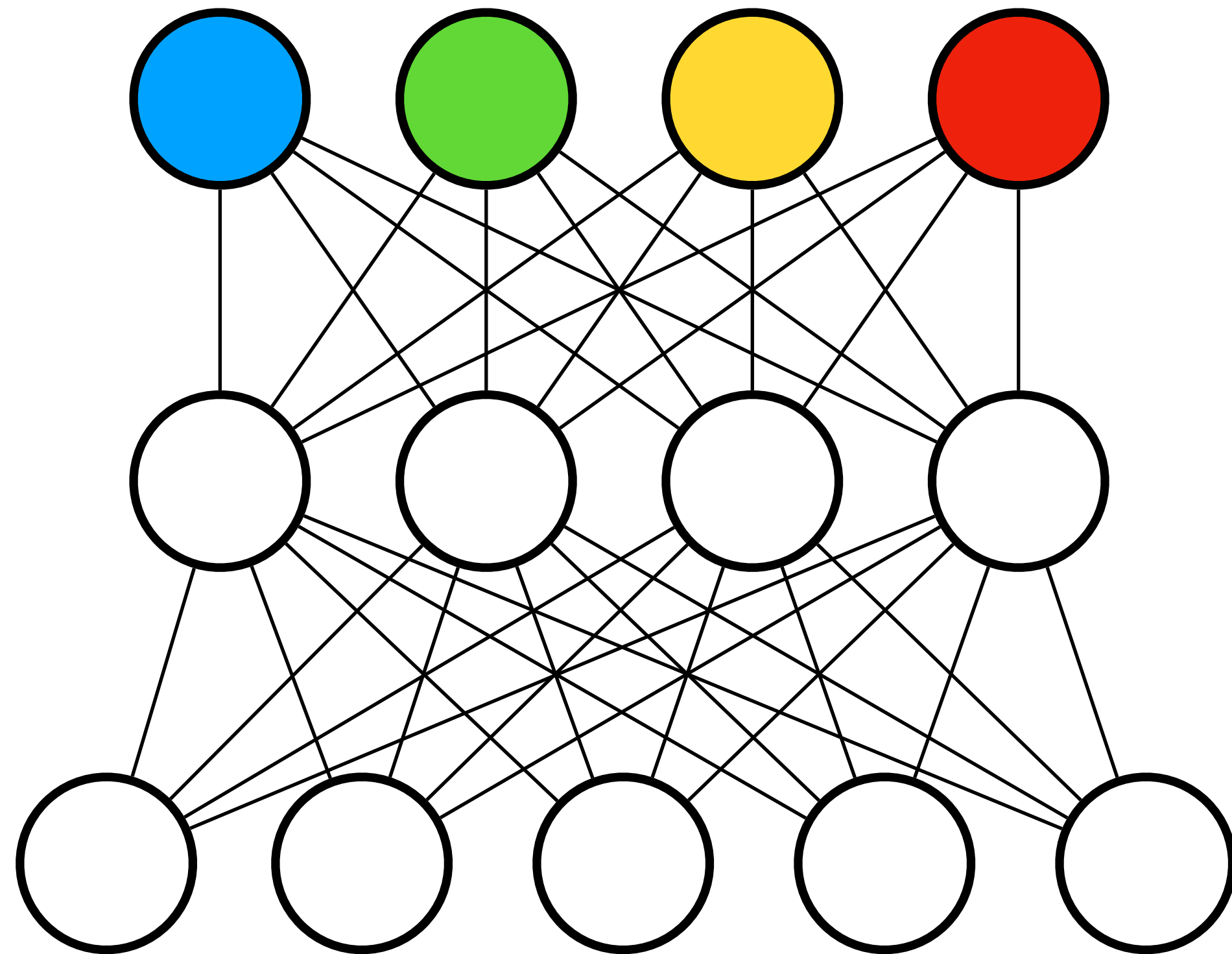
Single requirement of form: $A_1 \wedge \dots \wedge A_k \wedge \neg A_{k+1} \wedge \dots \wedge \neg A_n \rightarrow \neg A$

It defines a **upper** bound for the class A equal to

$$ub_A = 1 - \min(h_{A_1}, \dots, h_{A_k}, 1 - h_{A_{k+1}}, \dots, 1 - h_{A_n})$$

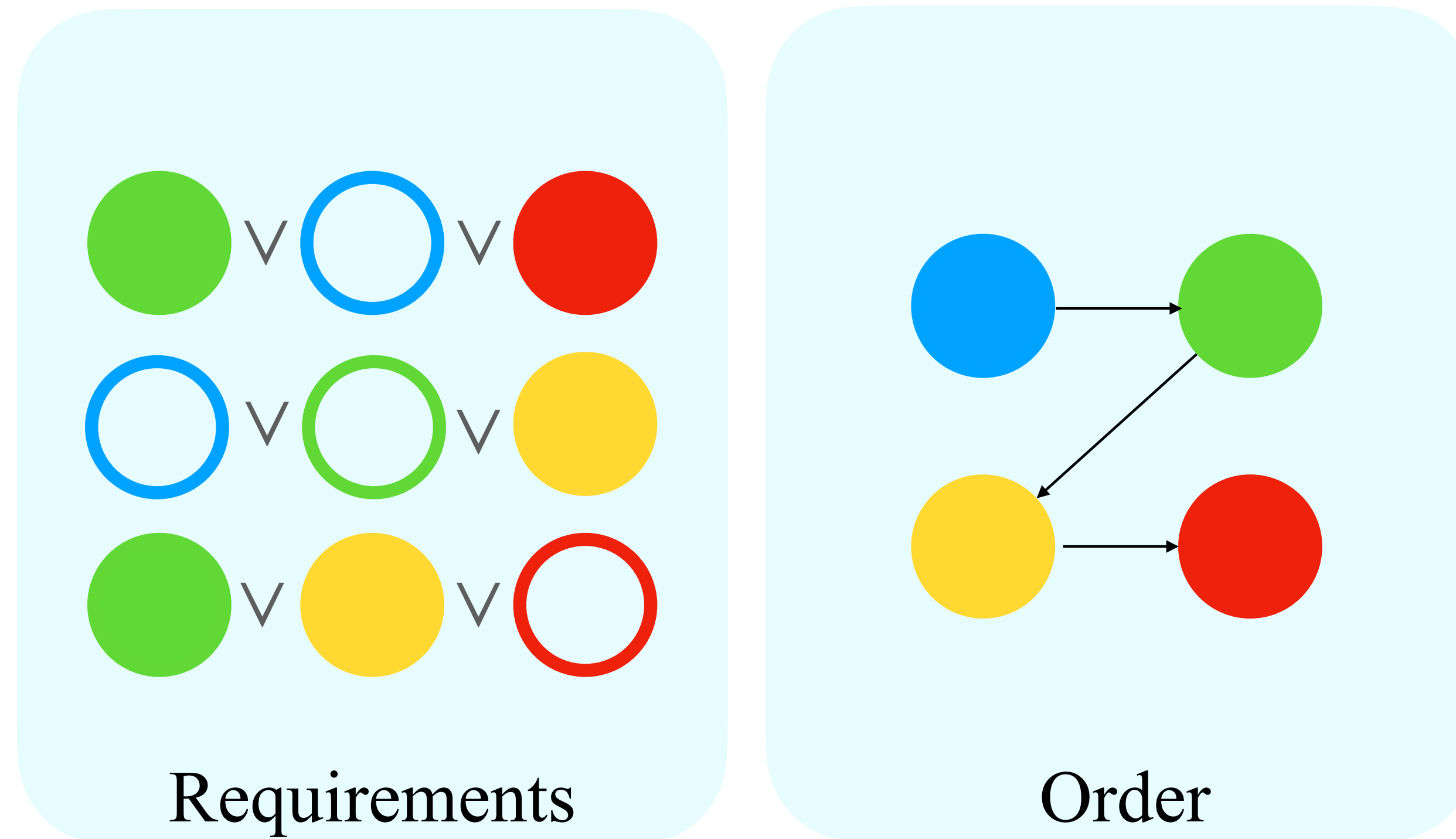


Propositional Logic Requirements



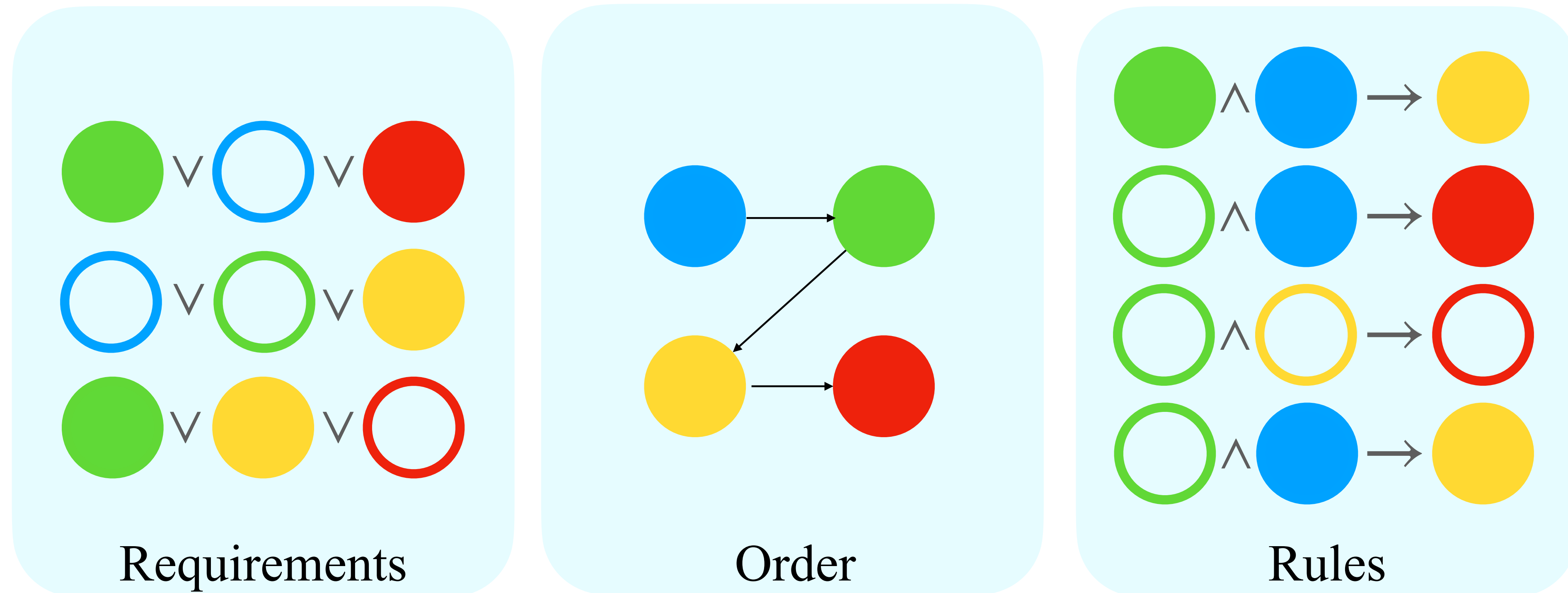


From clauses to rules



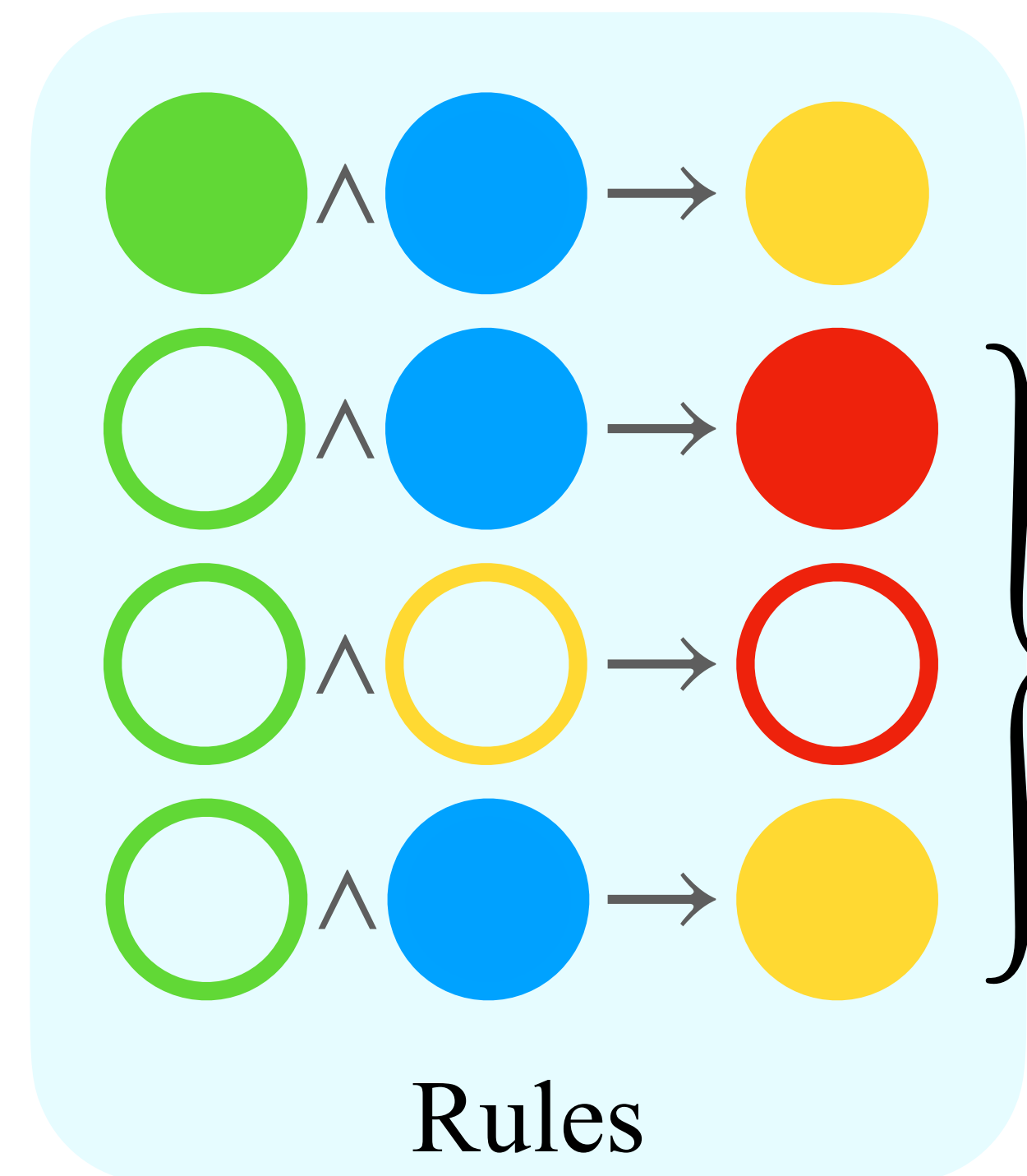
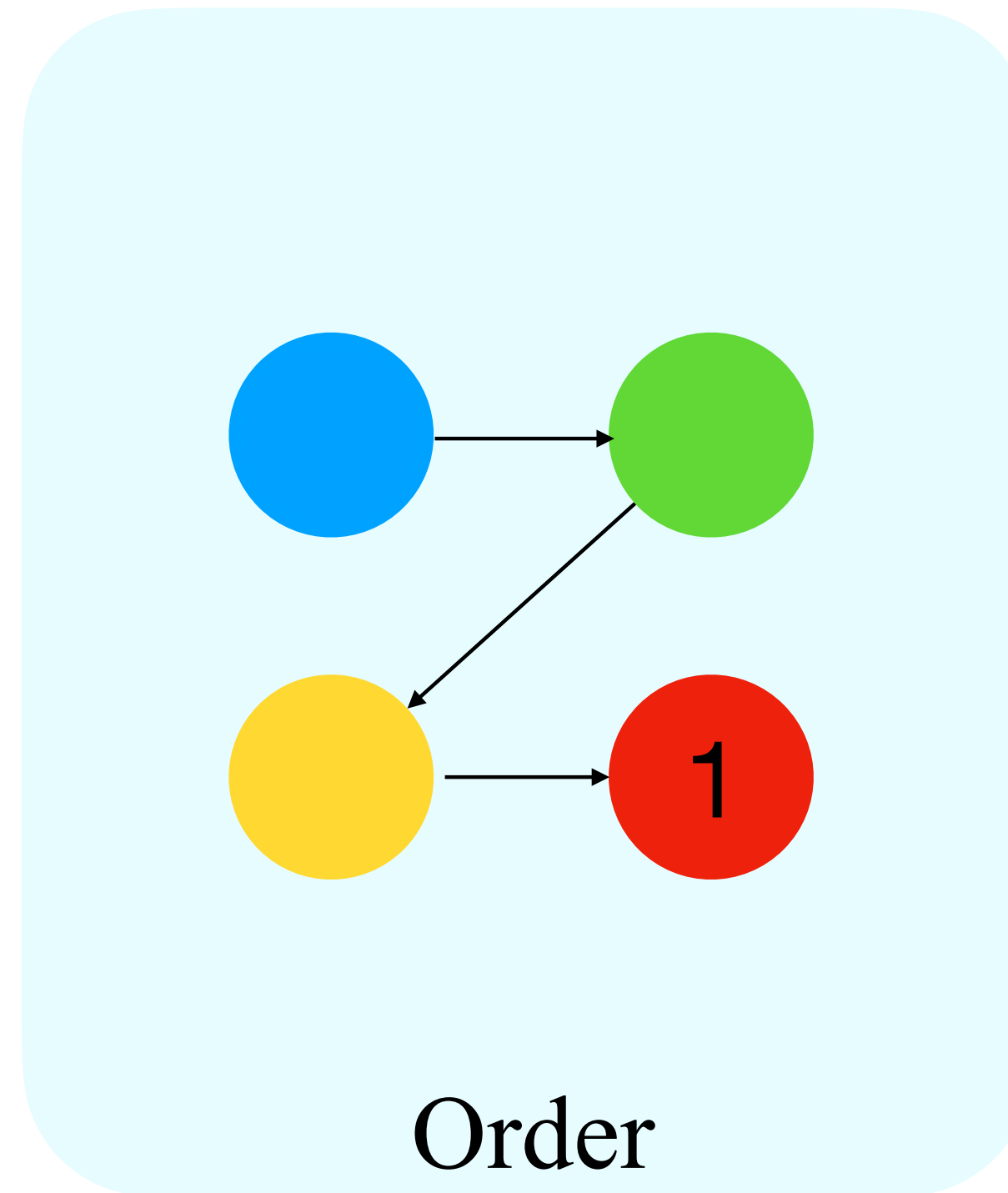
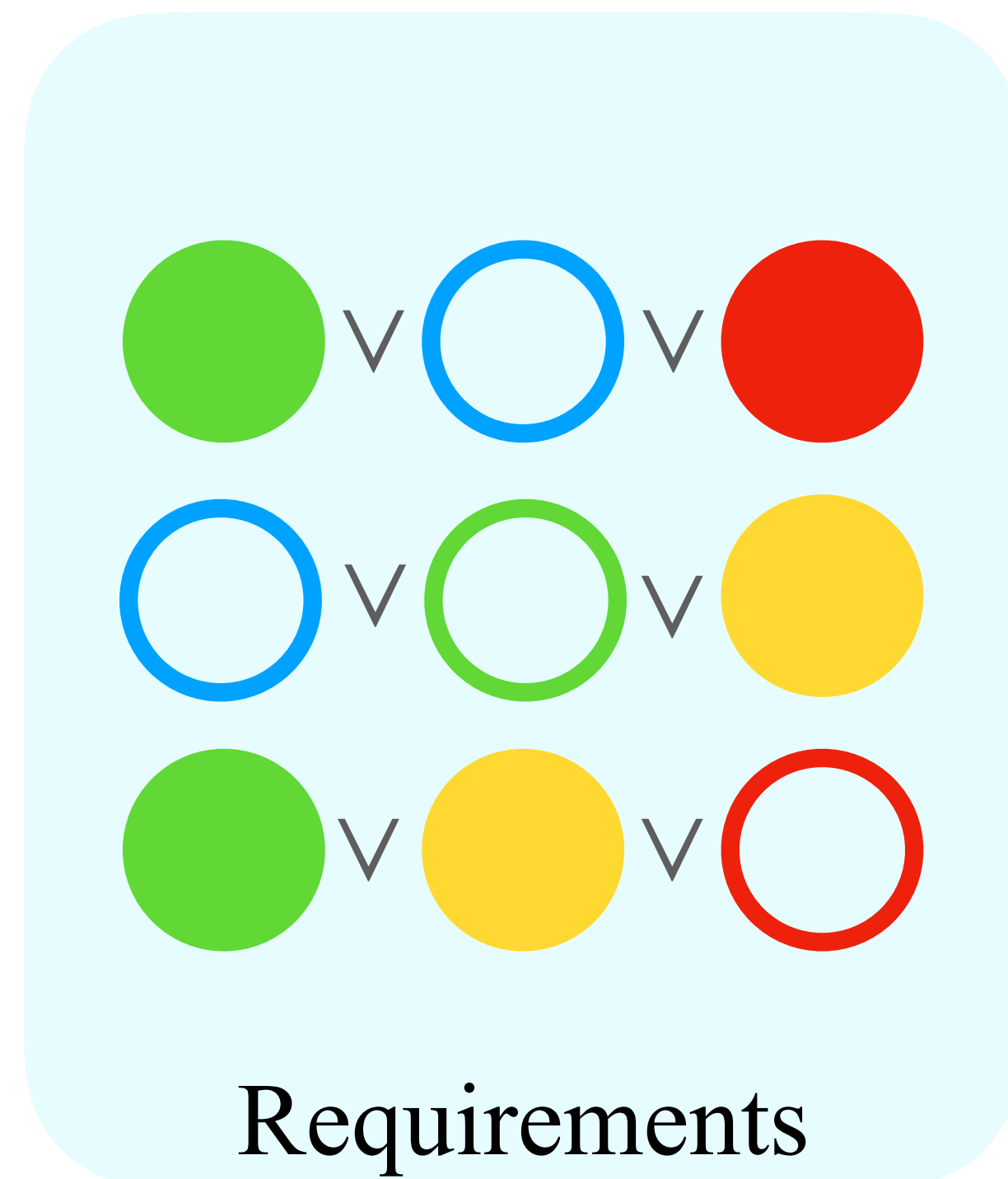


From clauses to rules



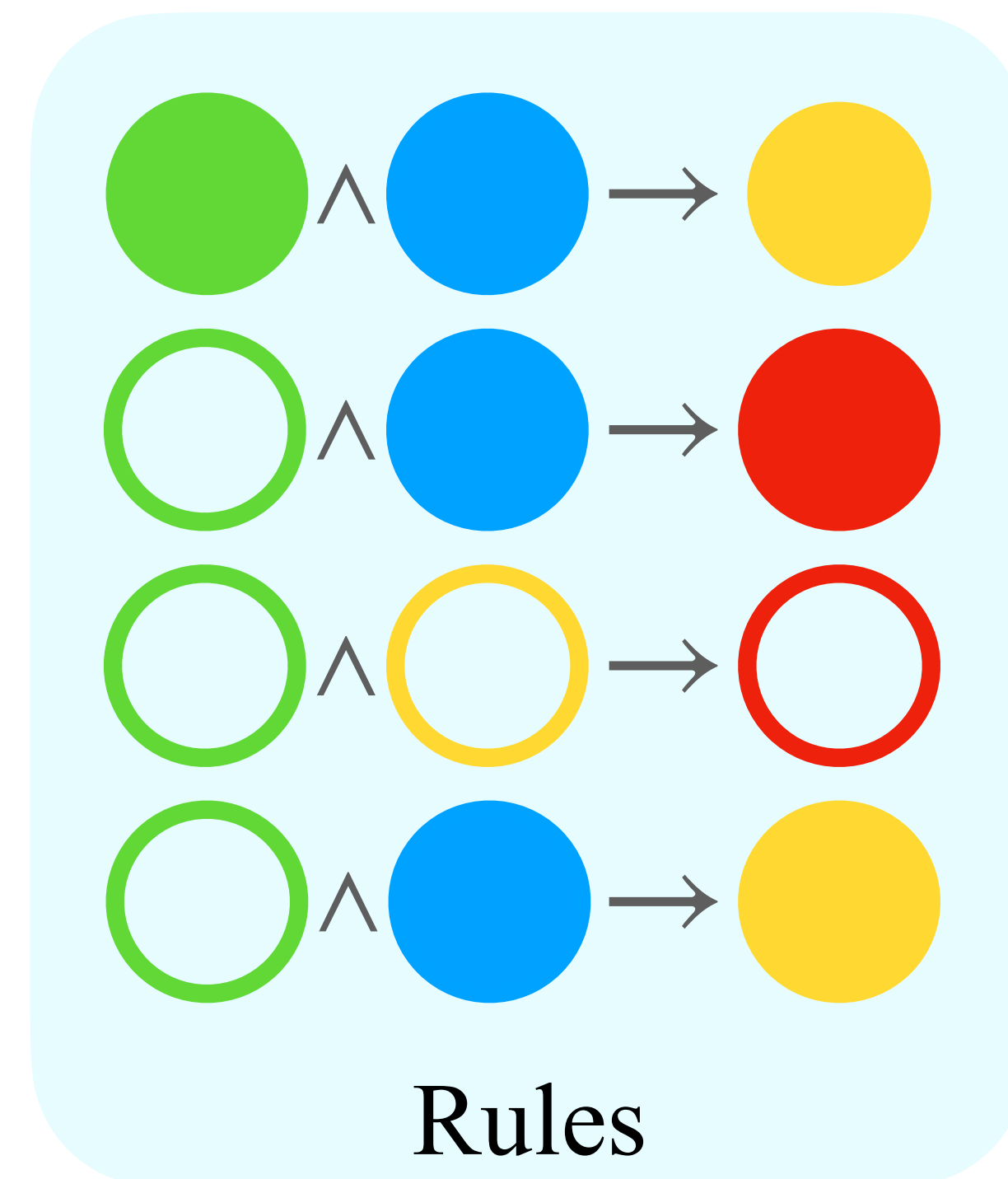
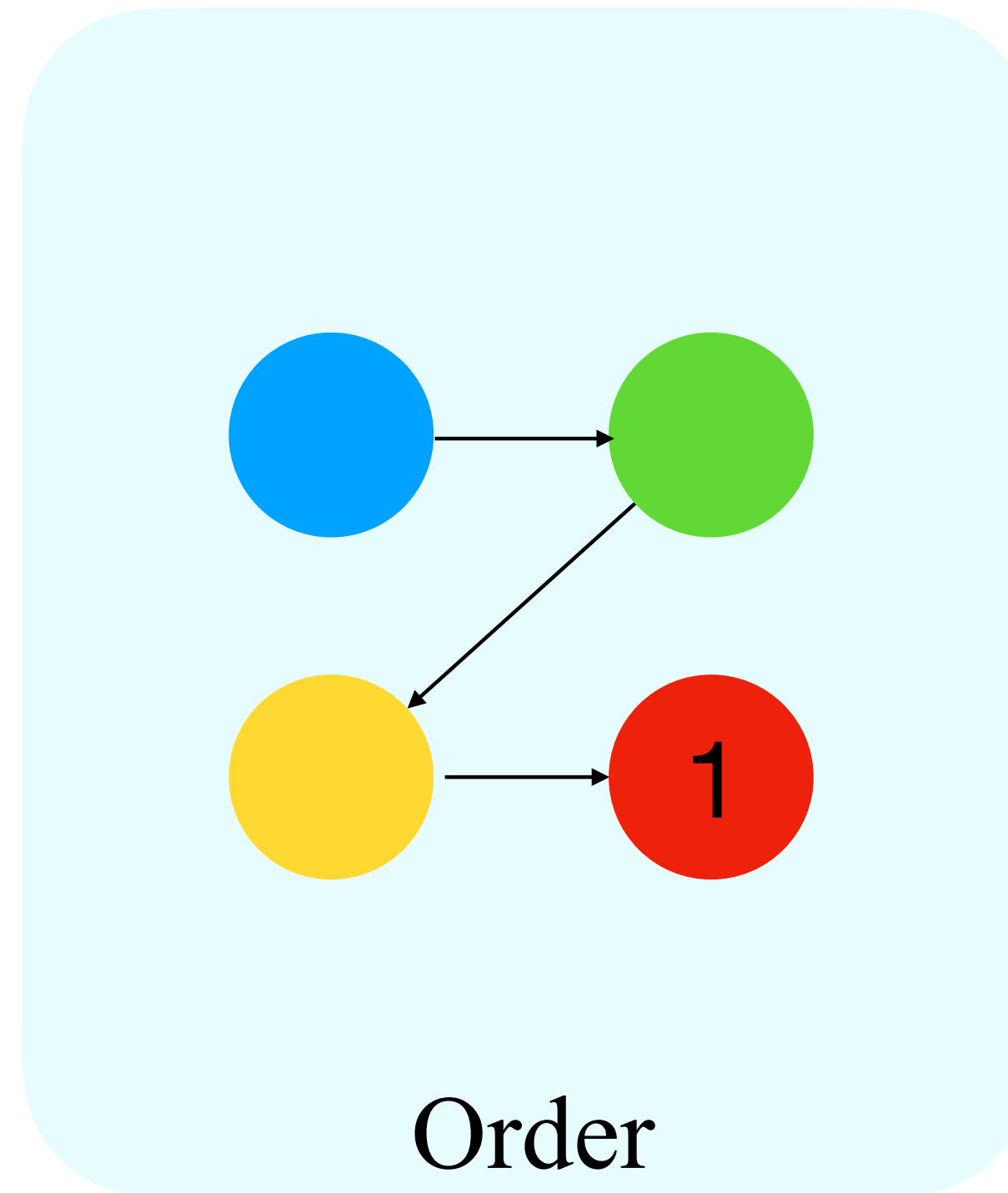
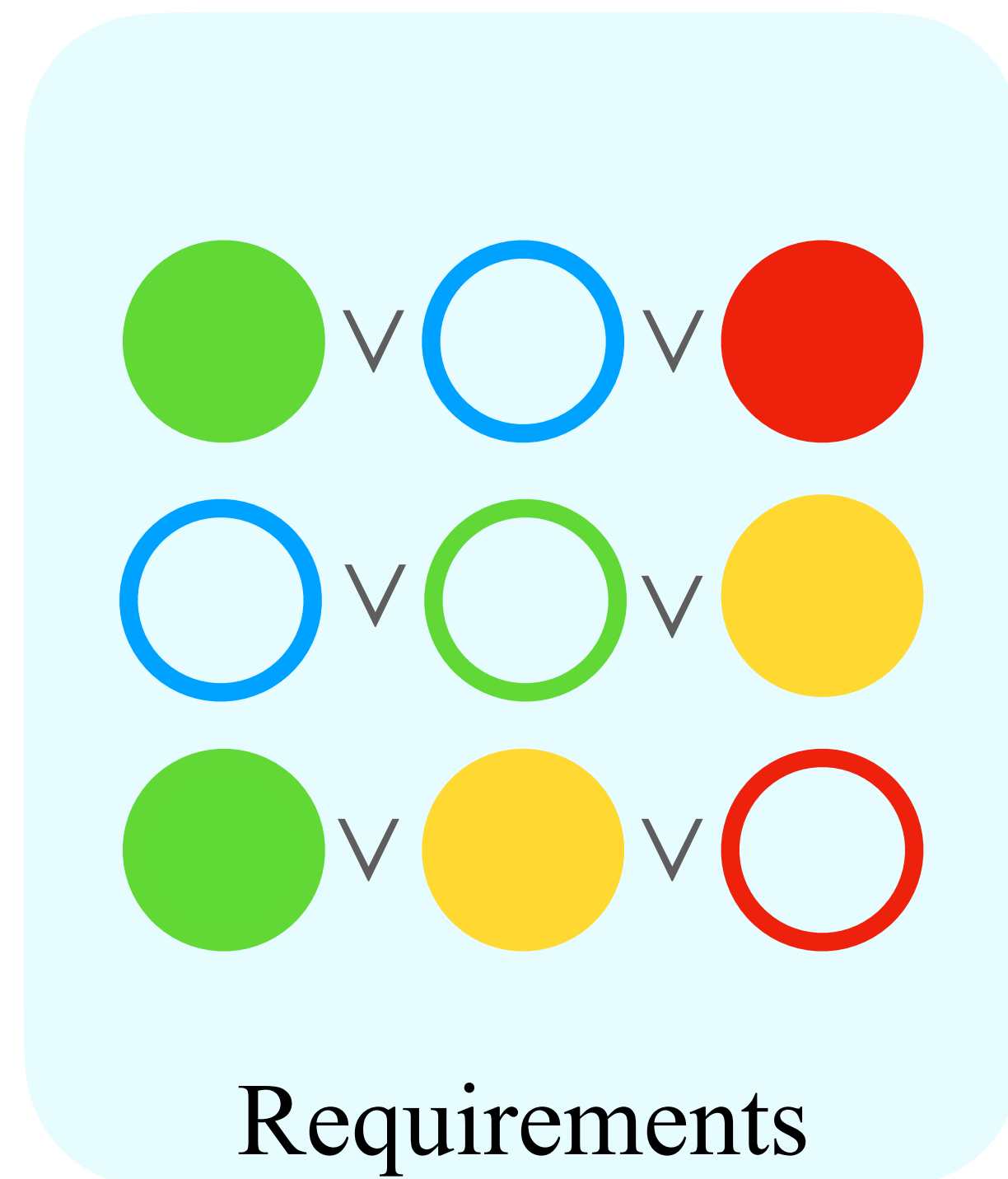


From clauses to rules



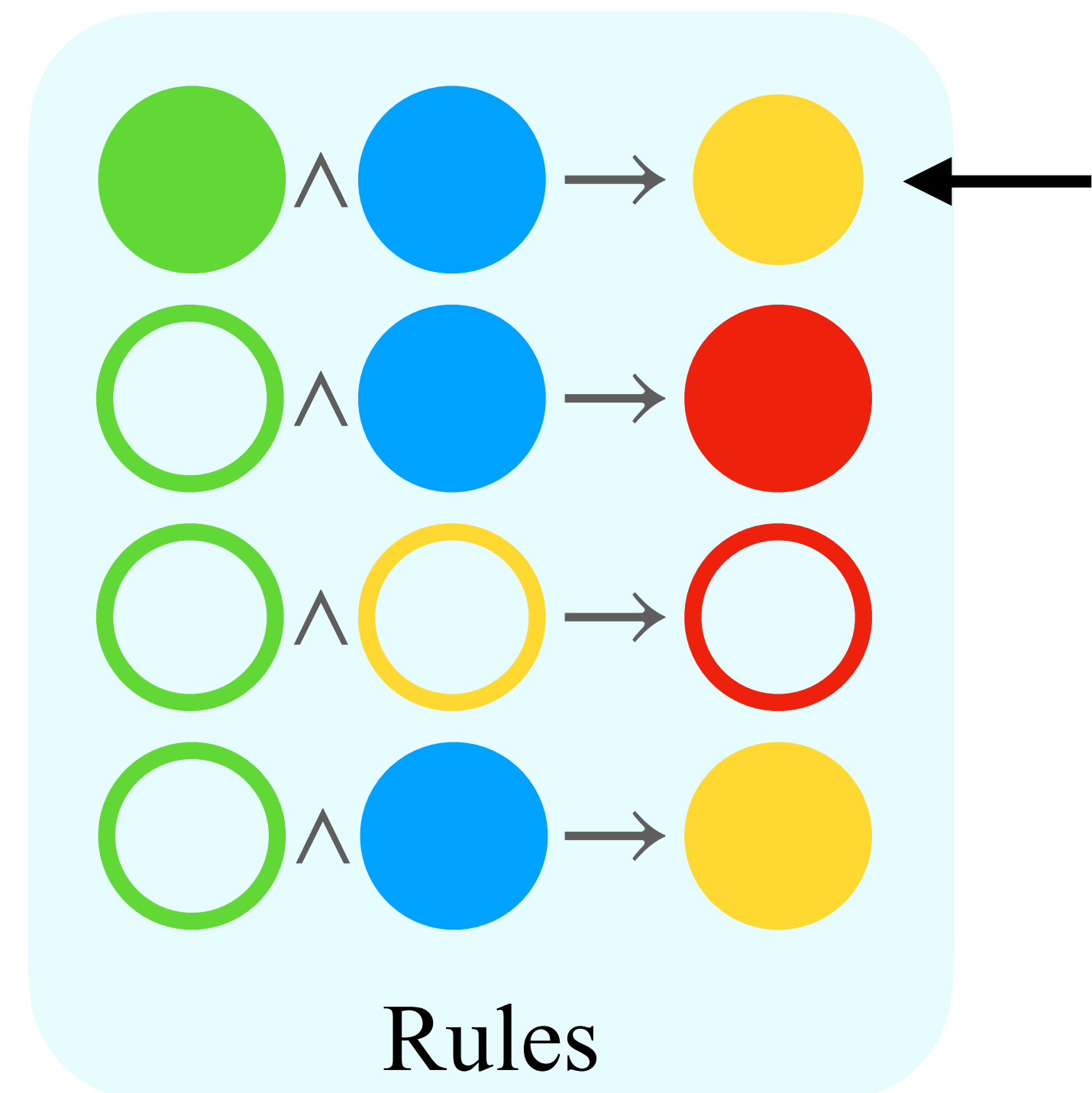
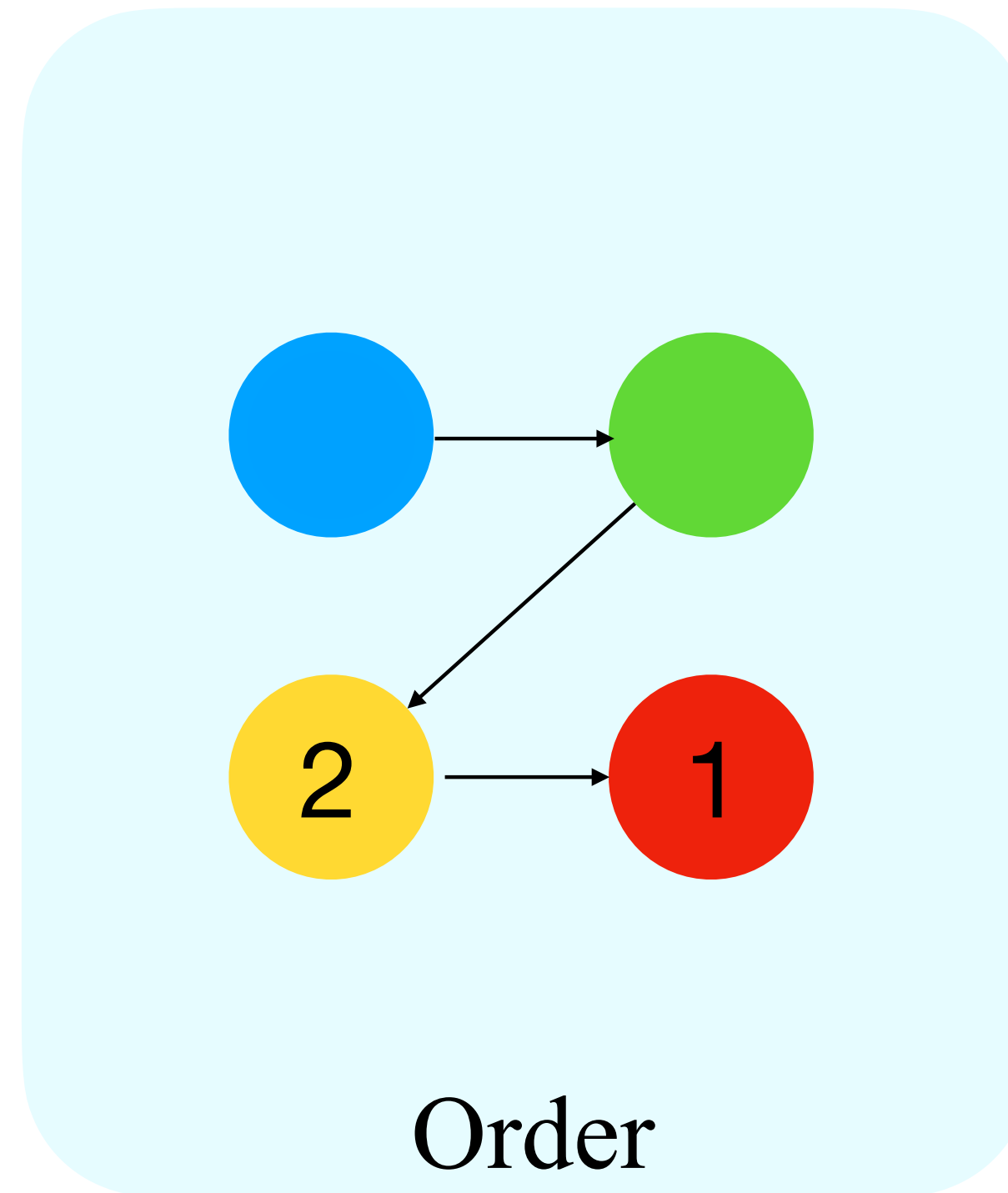
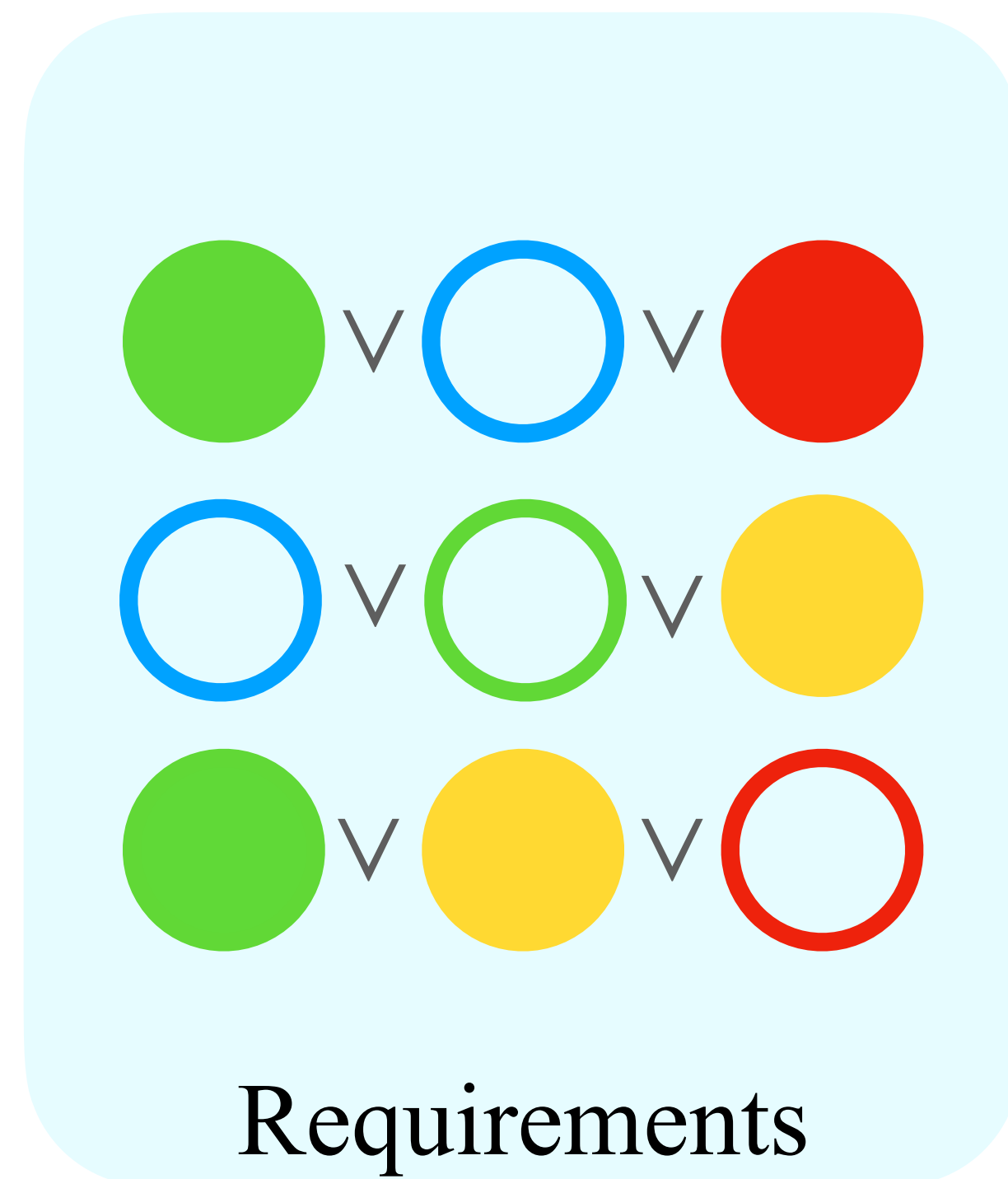


From clauses to rules



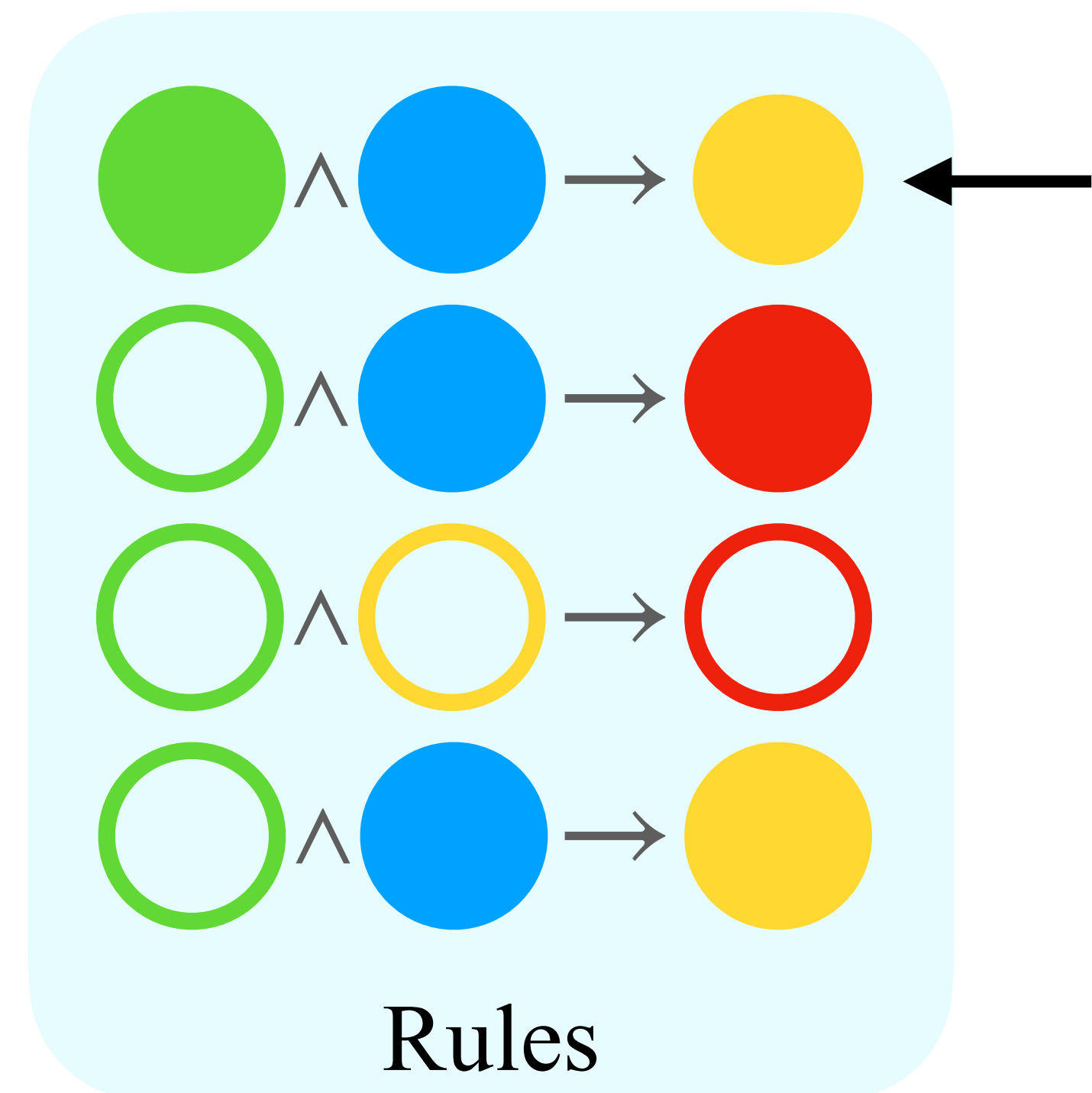
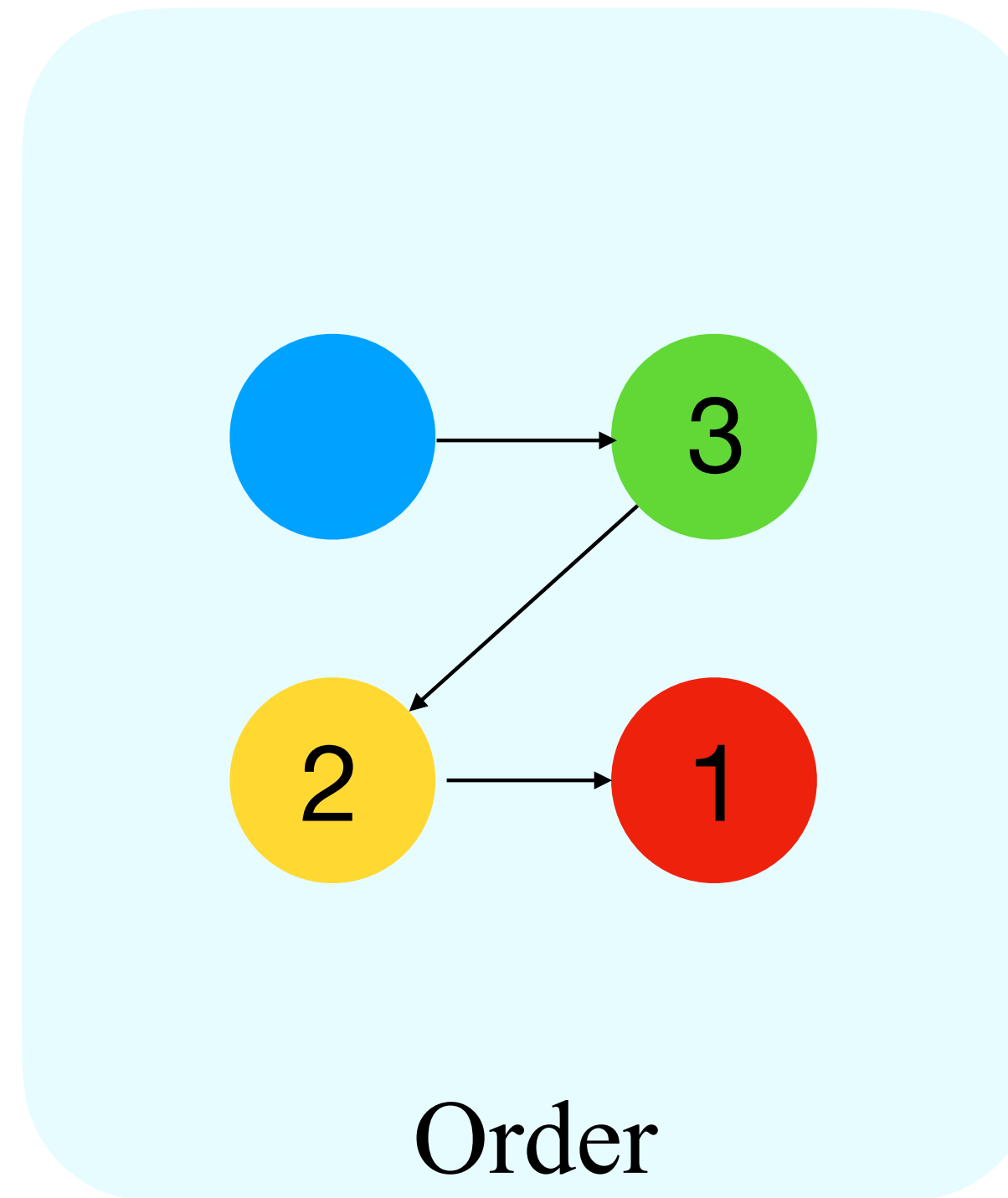
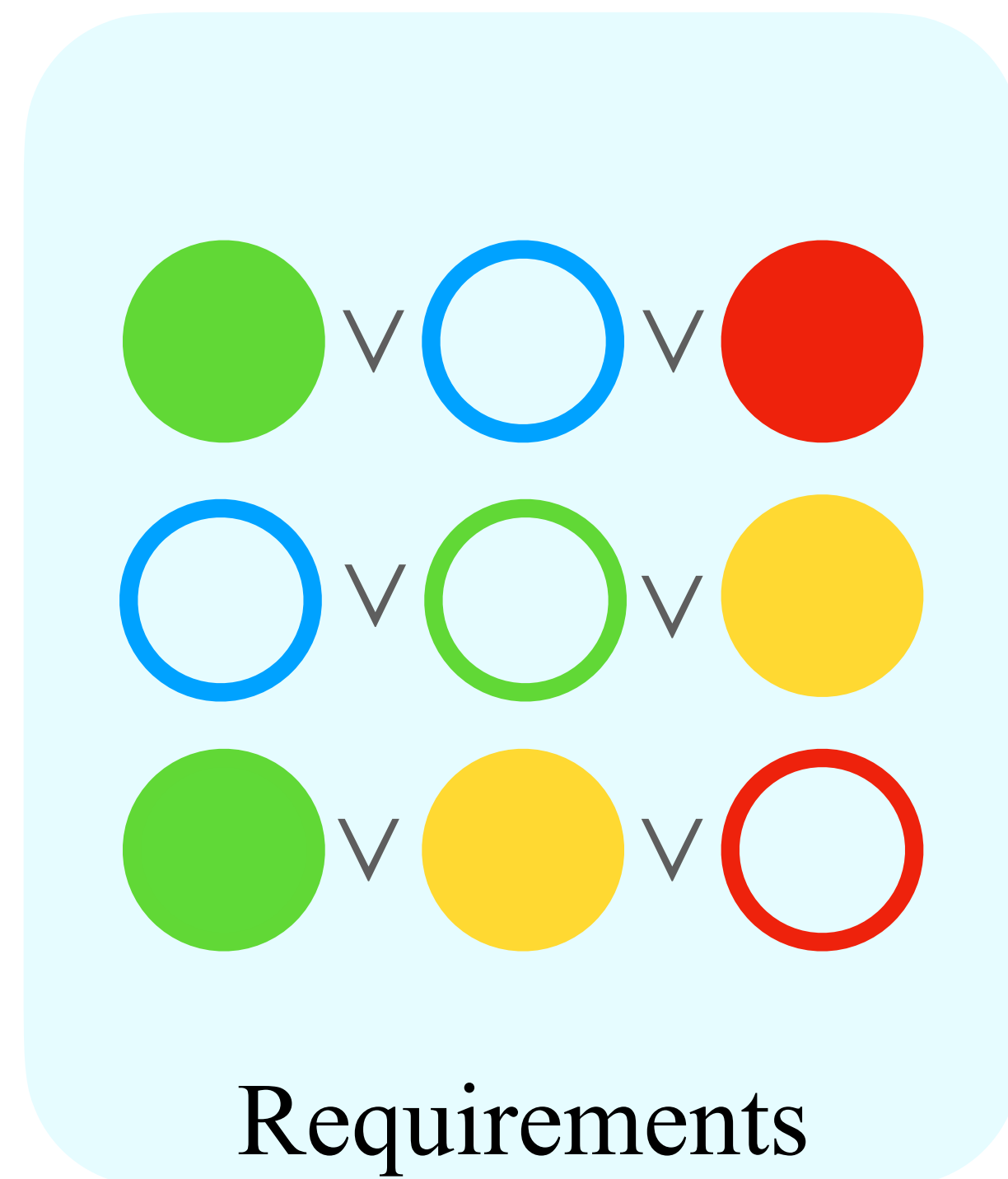


From clauses to rules



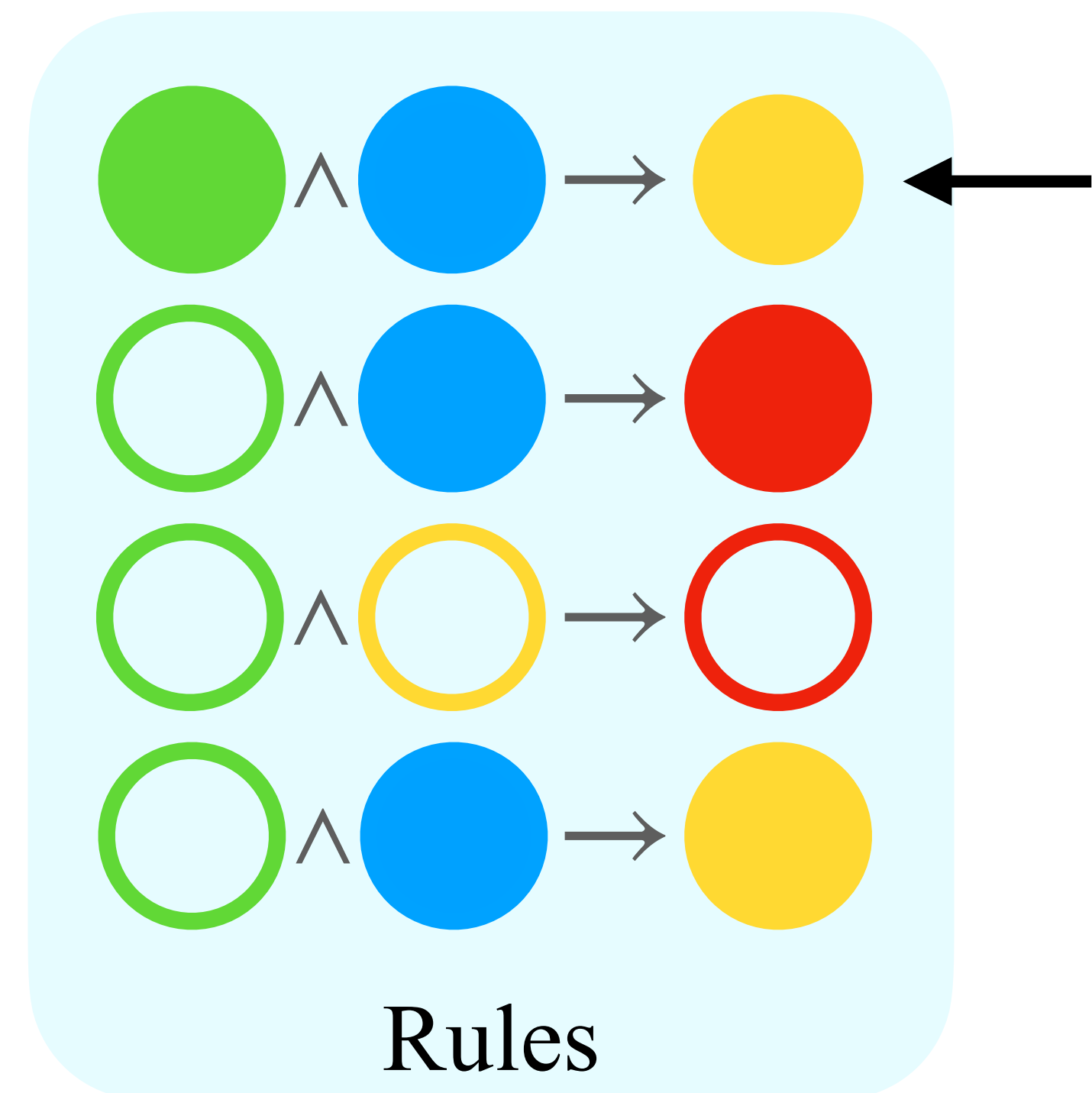
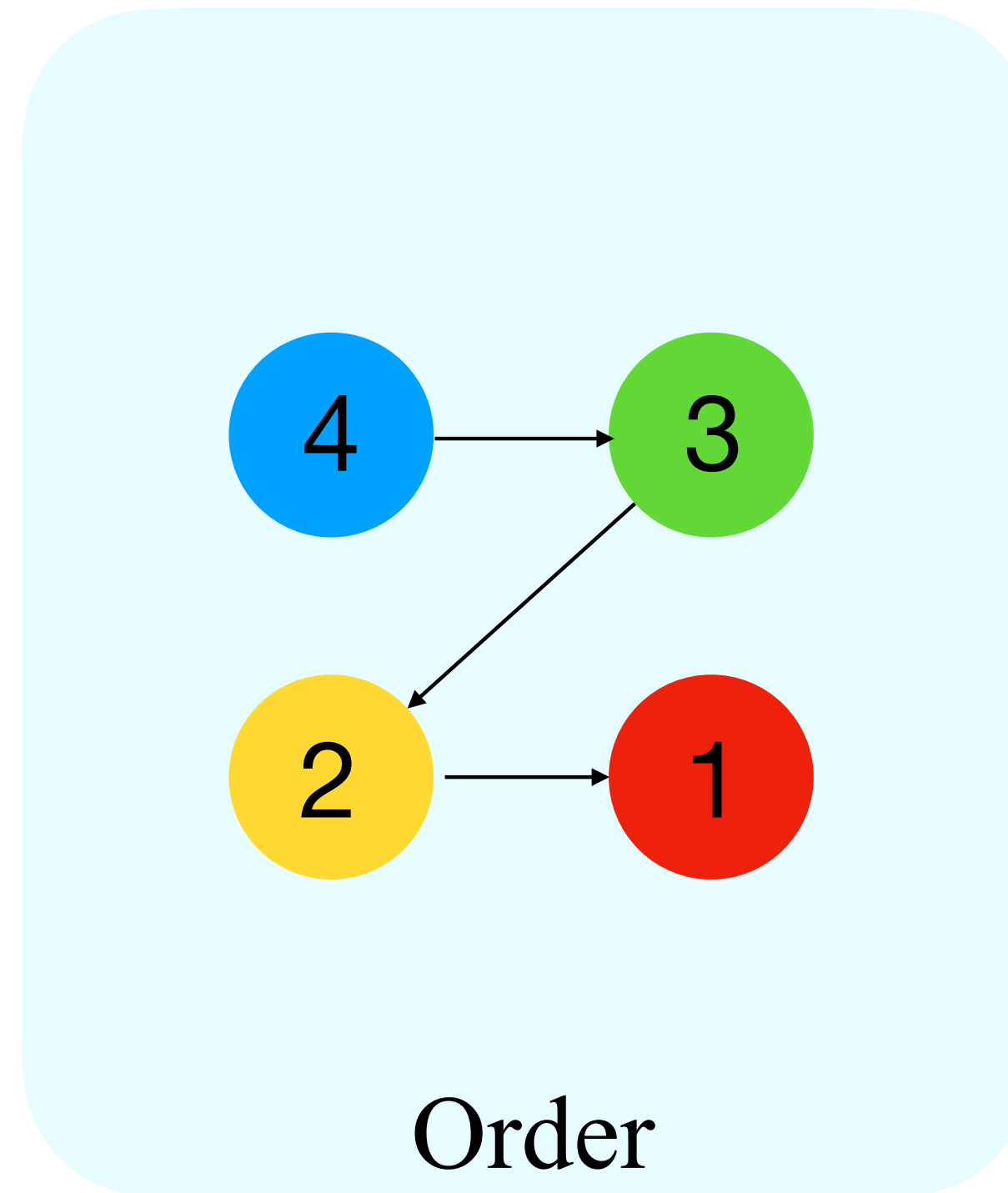
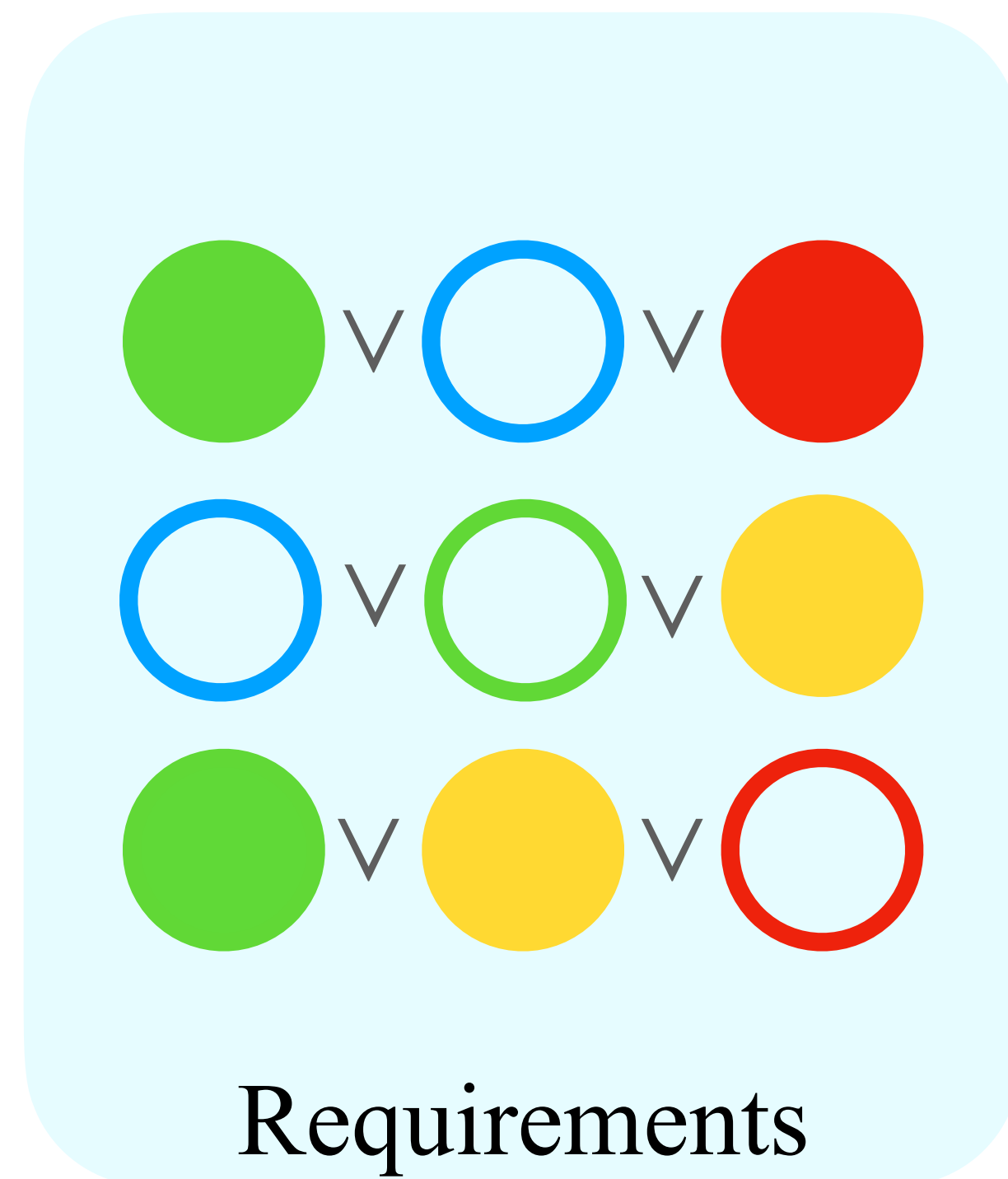


From clauses to rules

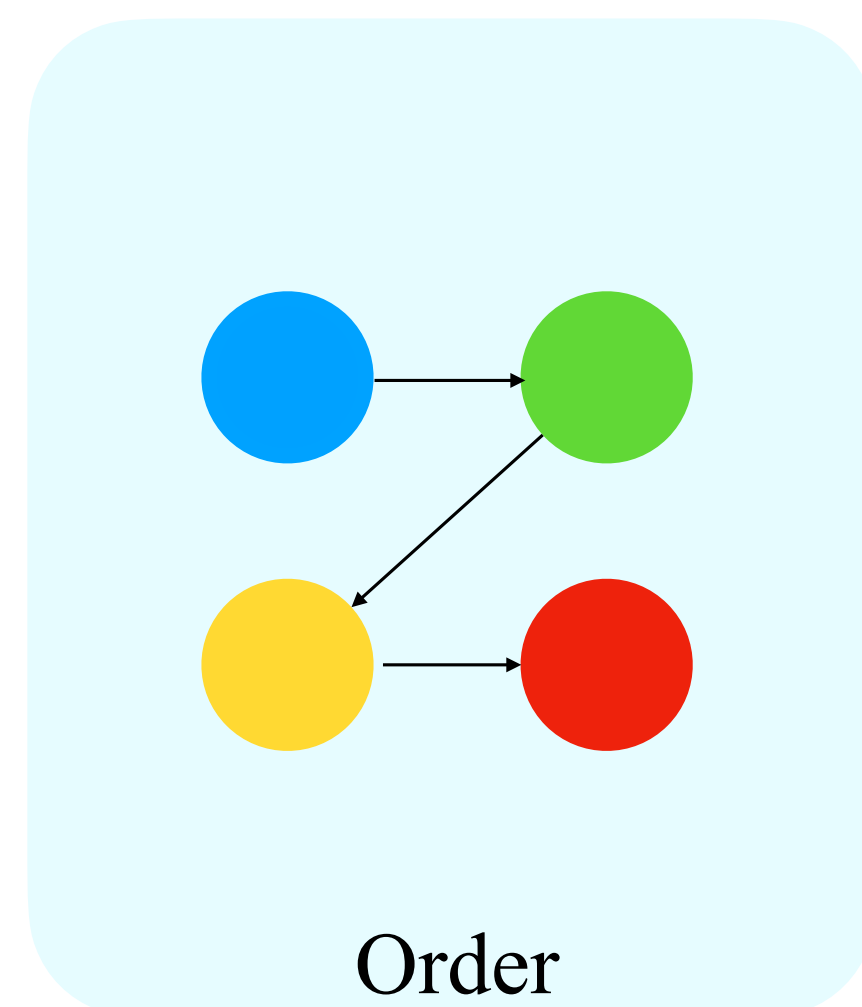
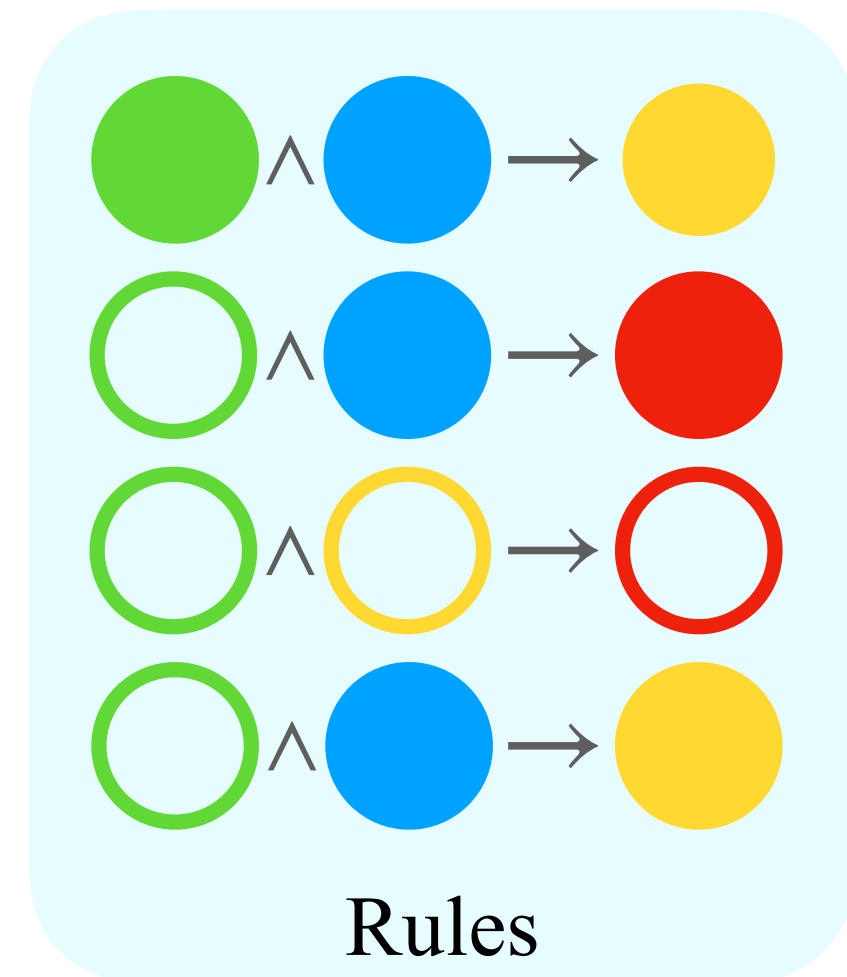




From clauses to rules



We can use the rules to compute bounds!

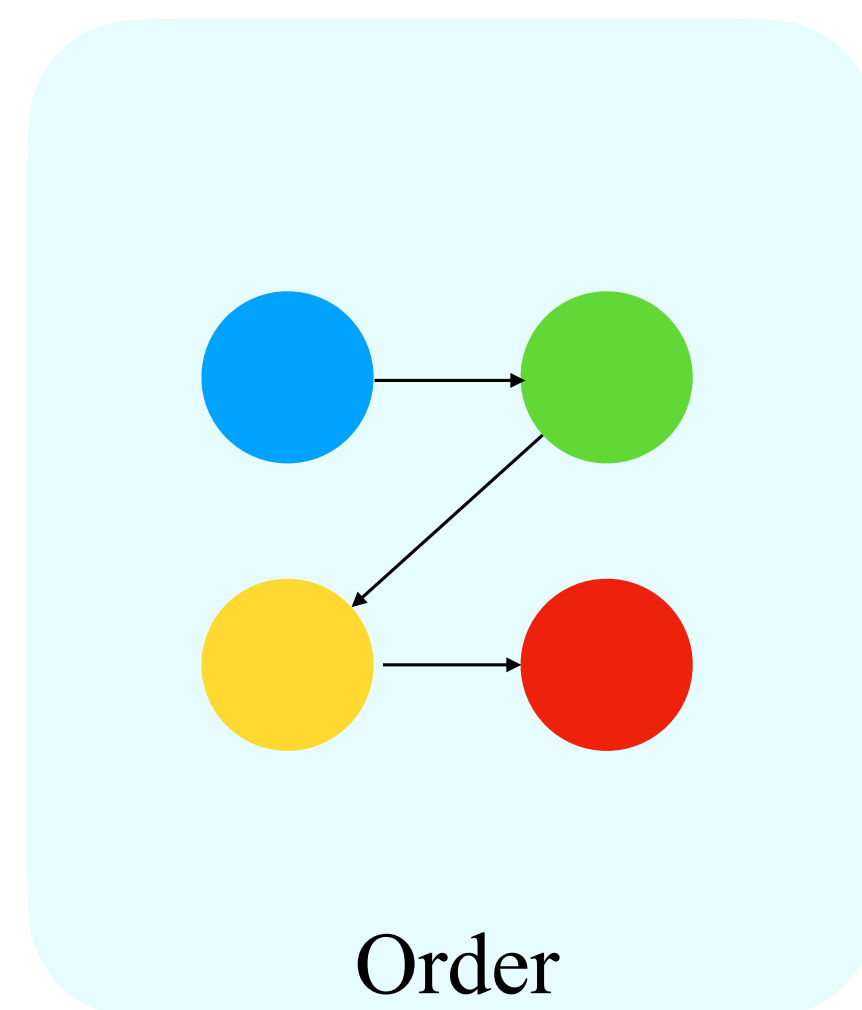
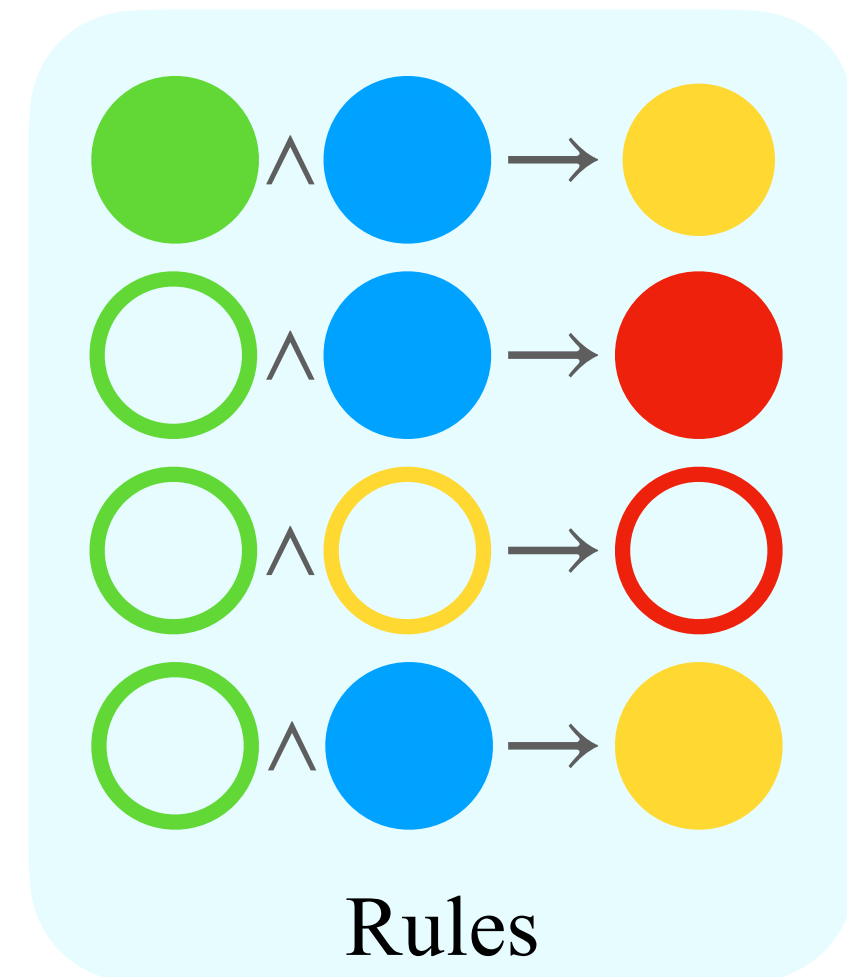




We can use the rules to compute bounds!

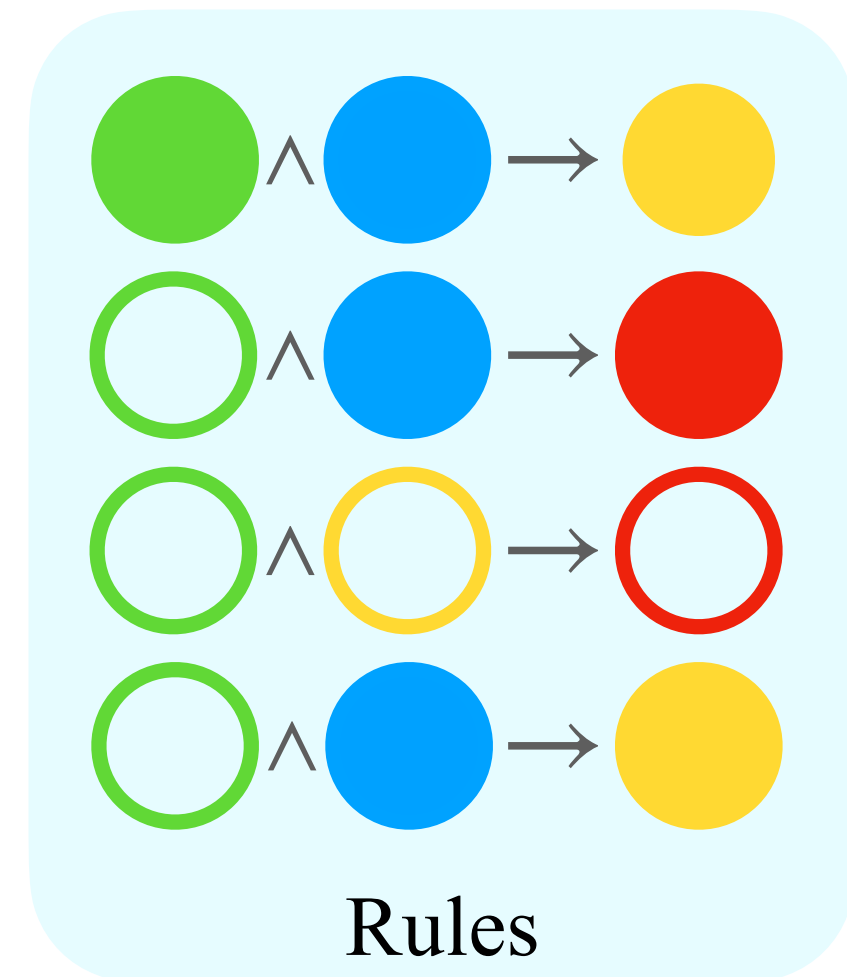
Consider **Blue**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$





We can use the rules to compute bounds!

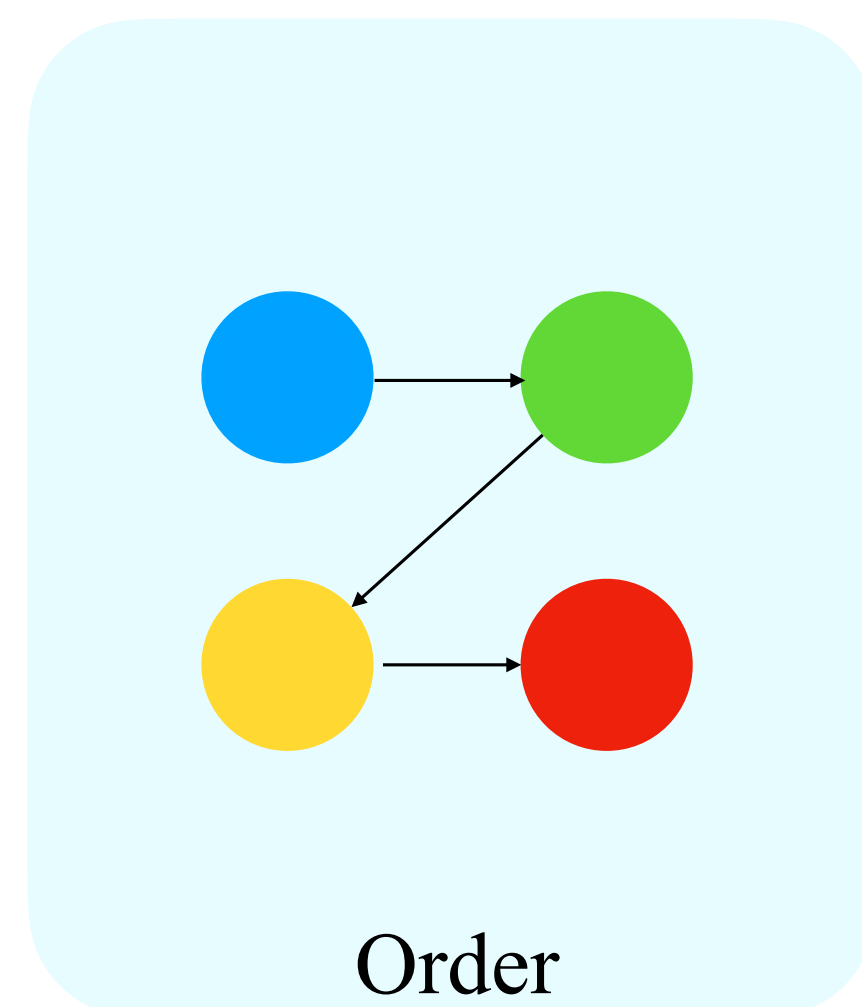


Consider **Blue**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$

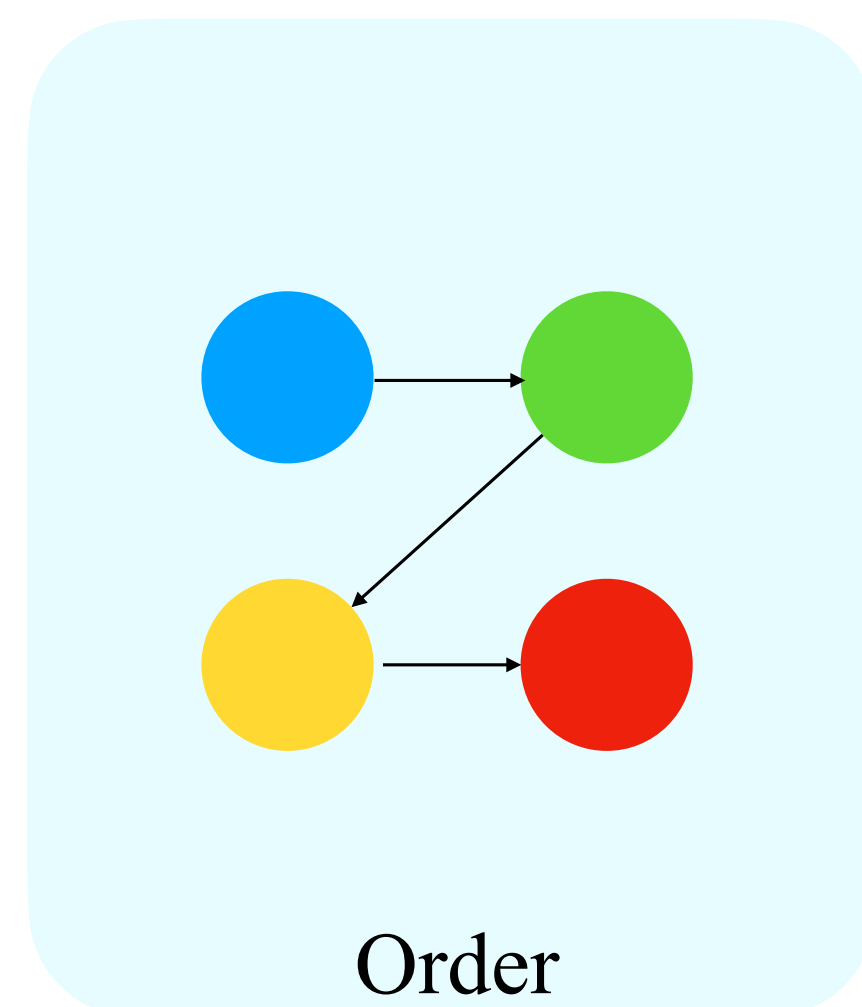
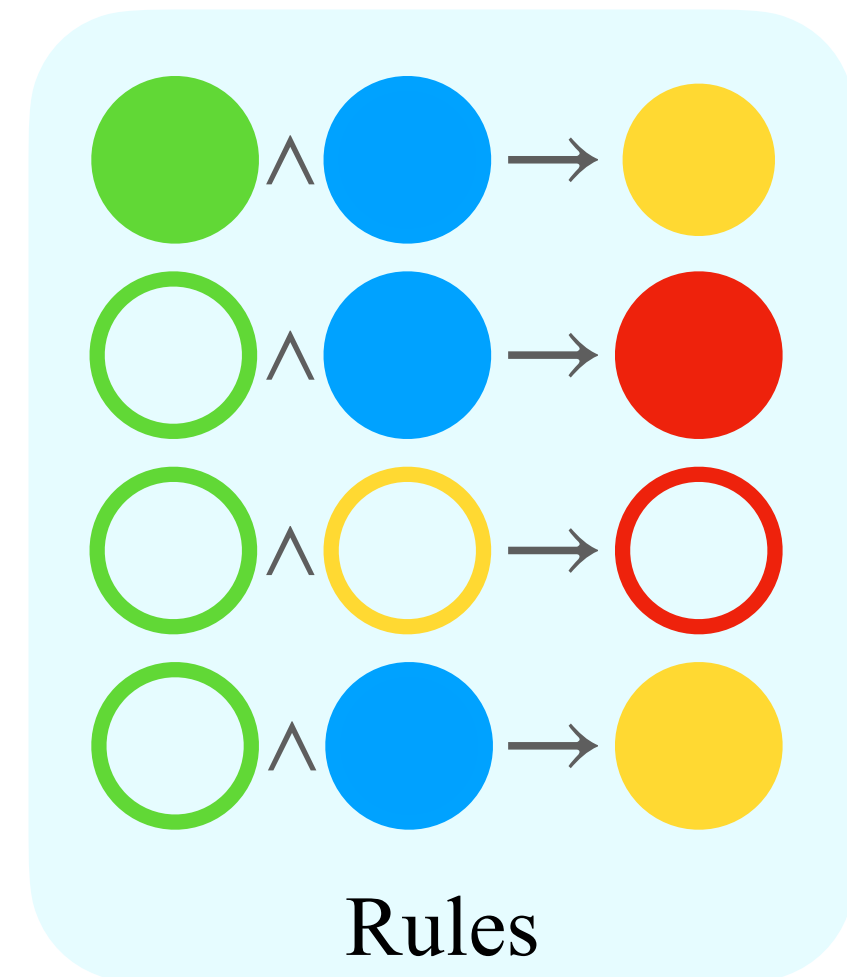
Consider **Green**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$





We can use the rules to compute bounds!



Consider **Blue**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$

Consider **Green**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$

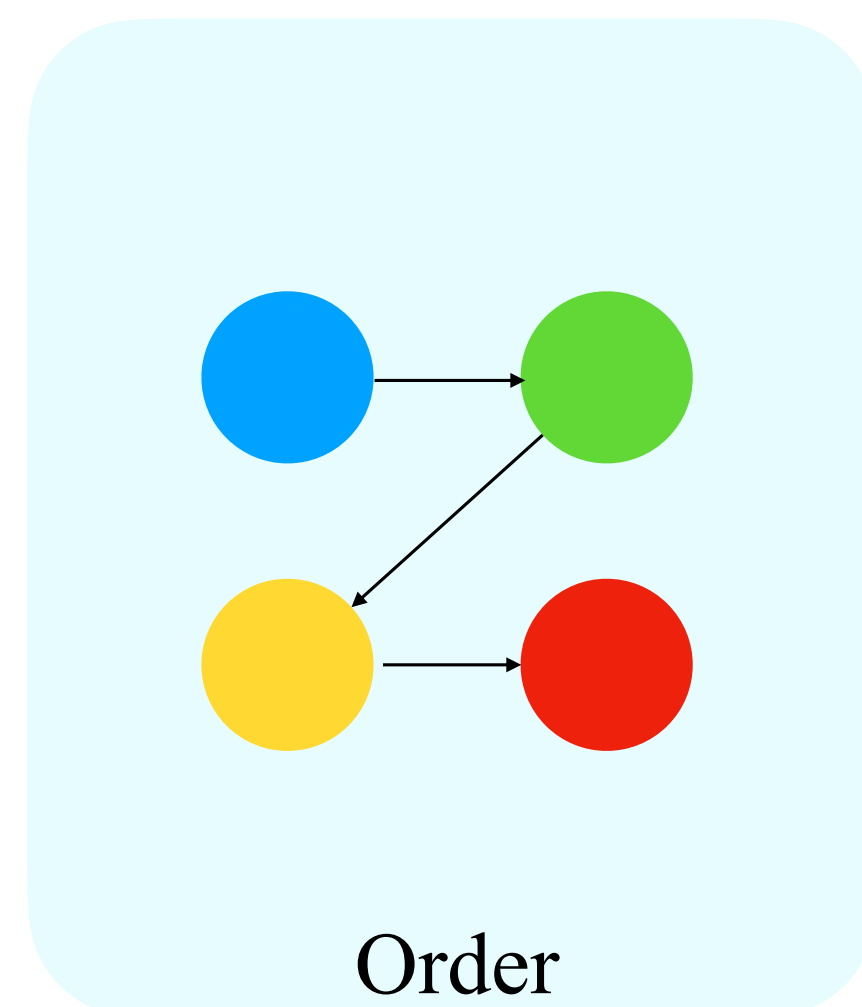
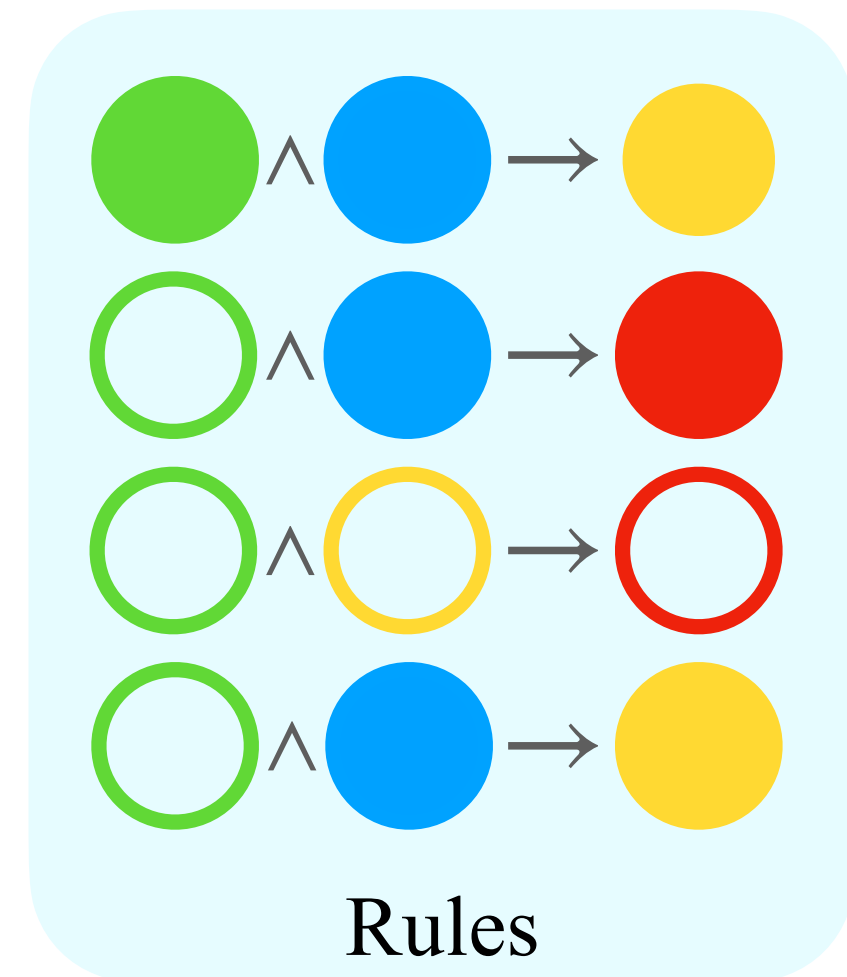
Consider **Yellow**:

$$lb_{\bullet} = \max(\min(h'_{\bullet}, h'_{\bullet}), \min(h'_{\bullet}, 1 - h'_{\bullet}))$$

$$ub_{\bullet} = 1$$



We can use the rules to compute bounds!



Consider **Blue**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$

Consider **Green**:

$$lb_{\bullet} = 0, \quad ub_{\bullet} = 1$$

Consider **Yellow**:

$$lb_{\bullet} = \max(\min(h'_{\bullet}, h'_{\bullet}), \min(h'_{\bullet}, 1 - h'_{\bullet}))$$

$$ub_{\bullet} = 1$$

Consider **Red**:

$$lb_{\bullet} = \min(h'_{\bullet}, 1 - h'_{\bullet})$$

$$ub_{\bullet} = 1 - \min(1 - h'_{\bullet}, 1 - h'_{\bullet})$$



Safe output space

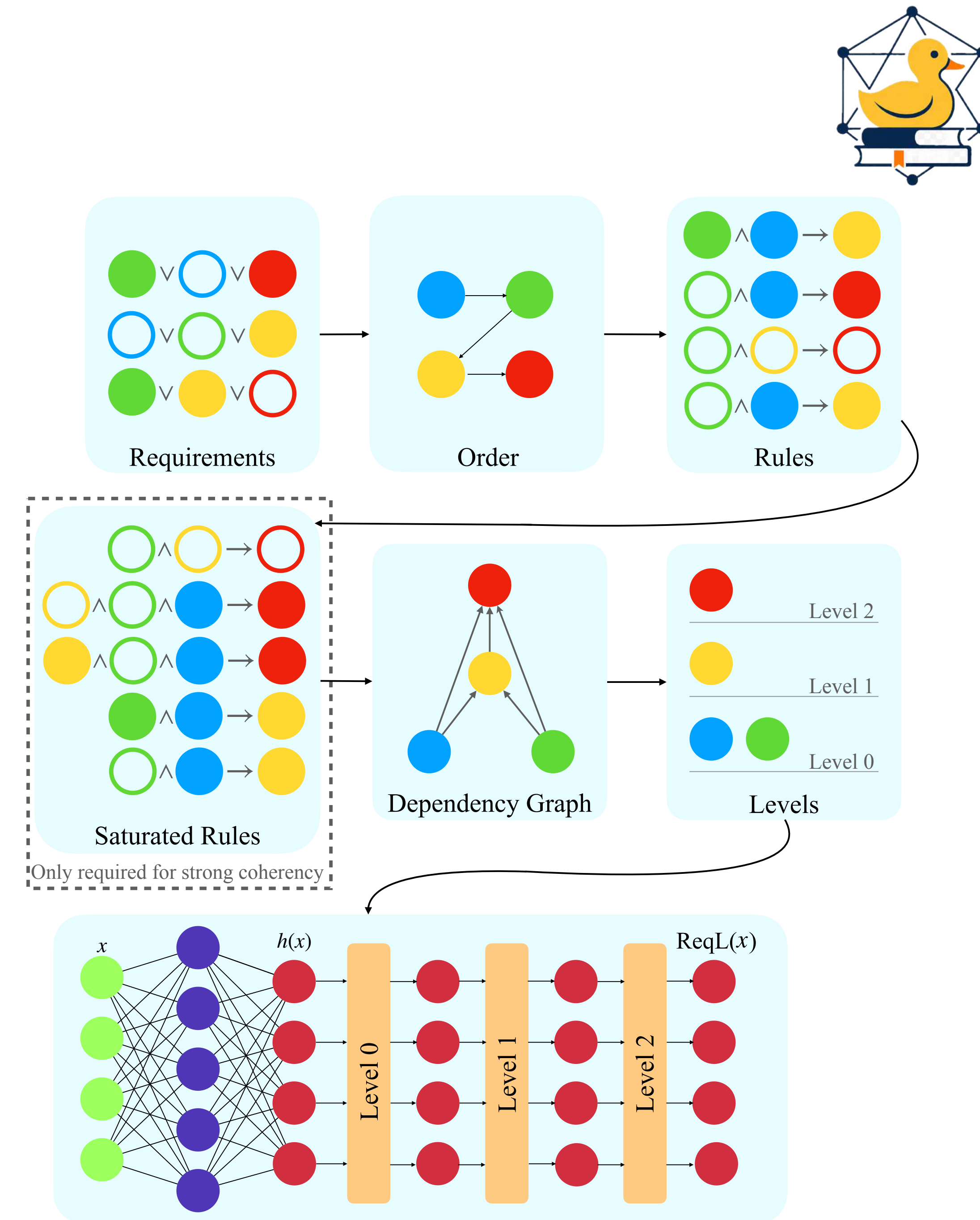
For every class A we can then compute its final value as:

$$h'_A = \min(ub_A, \max(h_A, lb_A))$$

Safe output space

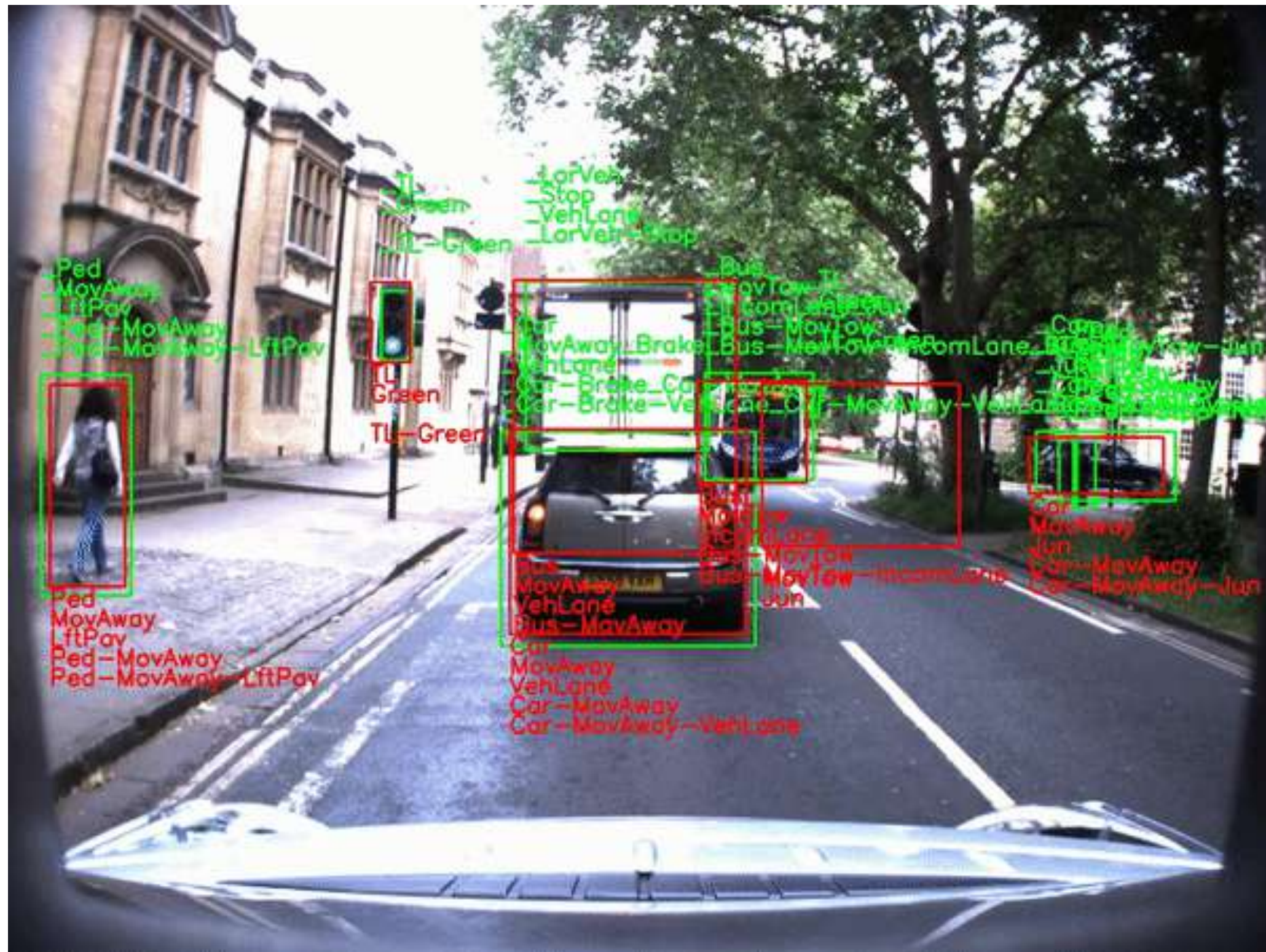
For every class A we can then compute its final value as:

$$h'_A = \min(ub_A, \max(h_A, lb_A))$$





The Autonomous Driving Case



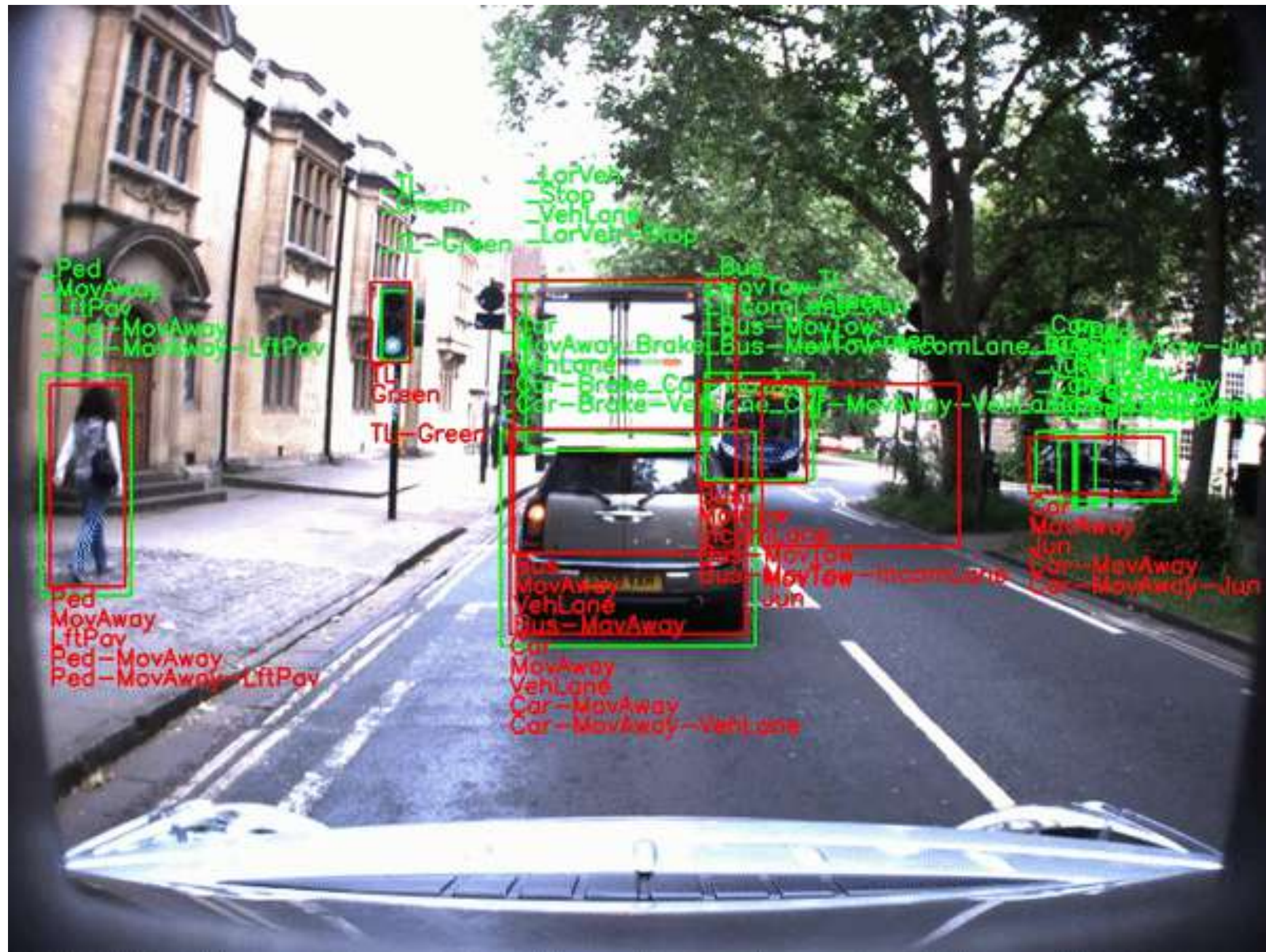
243 Requirements

Example:

$$\neg \text{RedTL} \vee \neg \text{GreenTL}$$



The Autonomous Driving Case



243 Requirements

Example:

$$\neg \text{RedTL} \vee \neg \text{GreenTL}$$



The Autonomous Driving Case

Standard Model





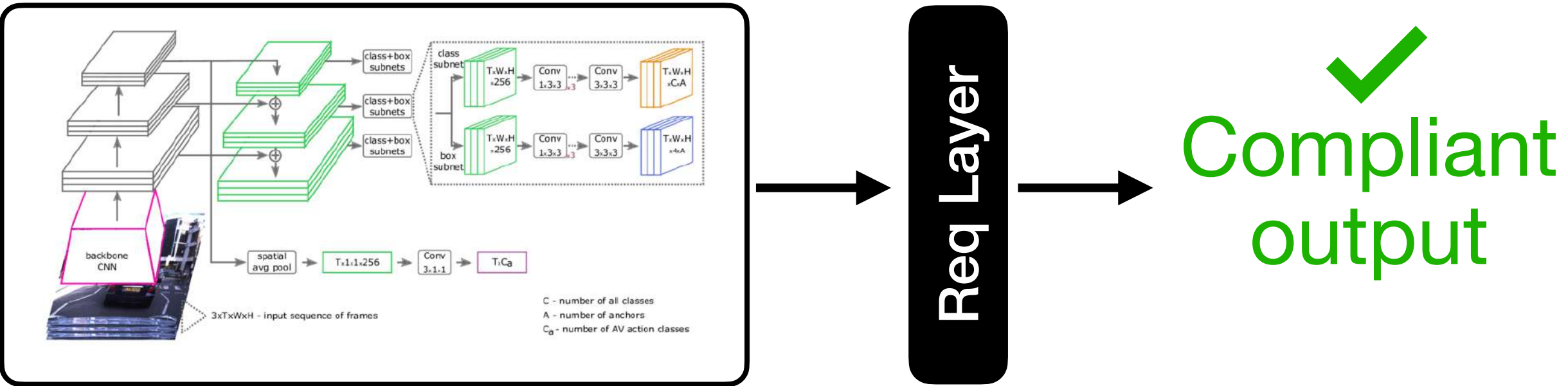
The Autonomous Driving Case

Standard Model





The Autonomous Driving Case

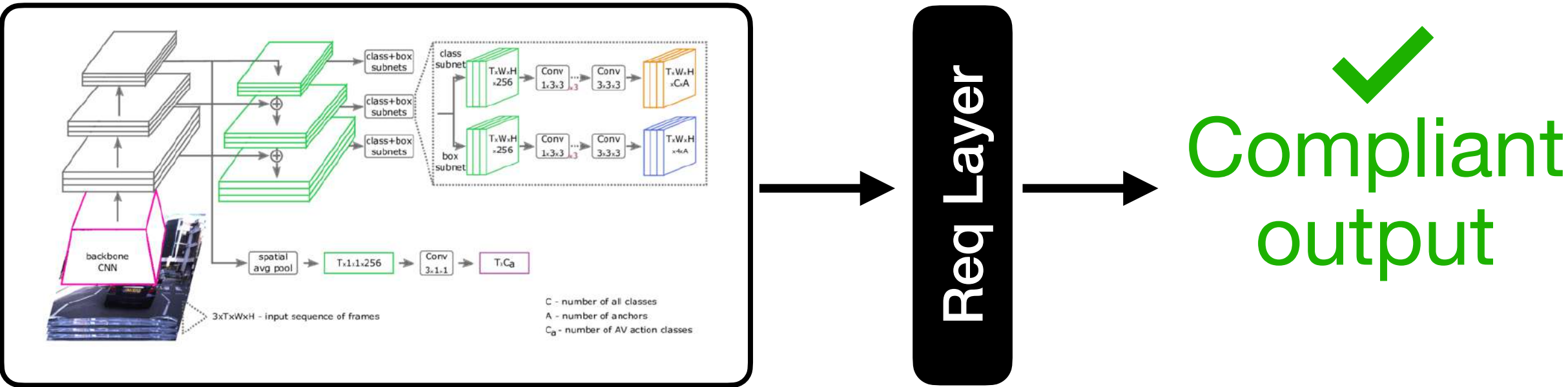


Standard Model





The Autonomous Driving Case



Standard Model

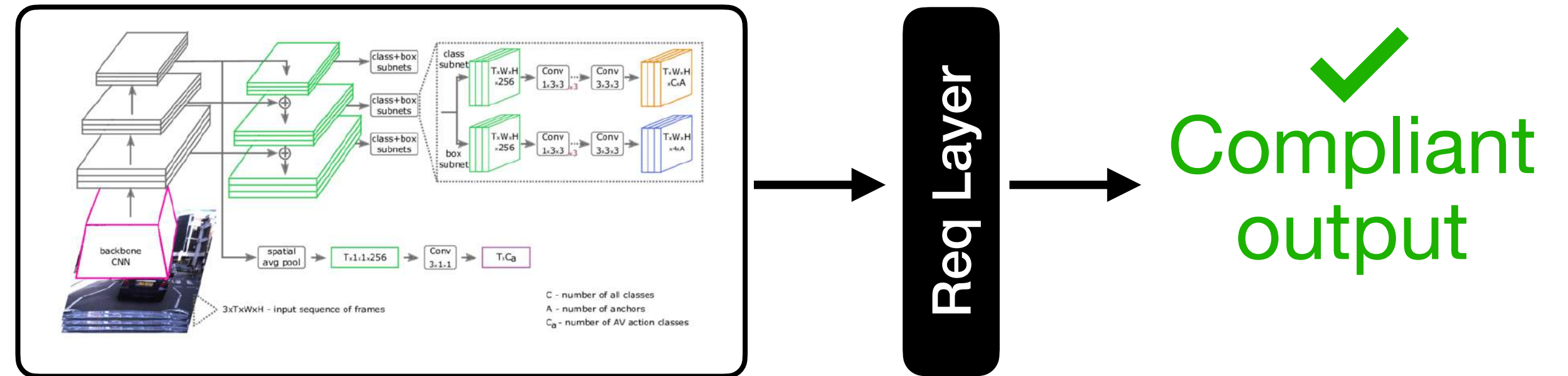


Same Model + Req Layer





The Autonomous Driving Case



Standard Model

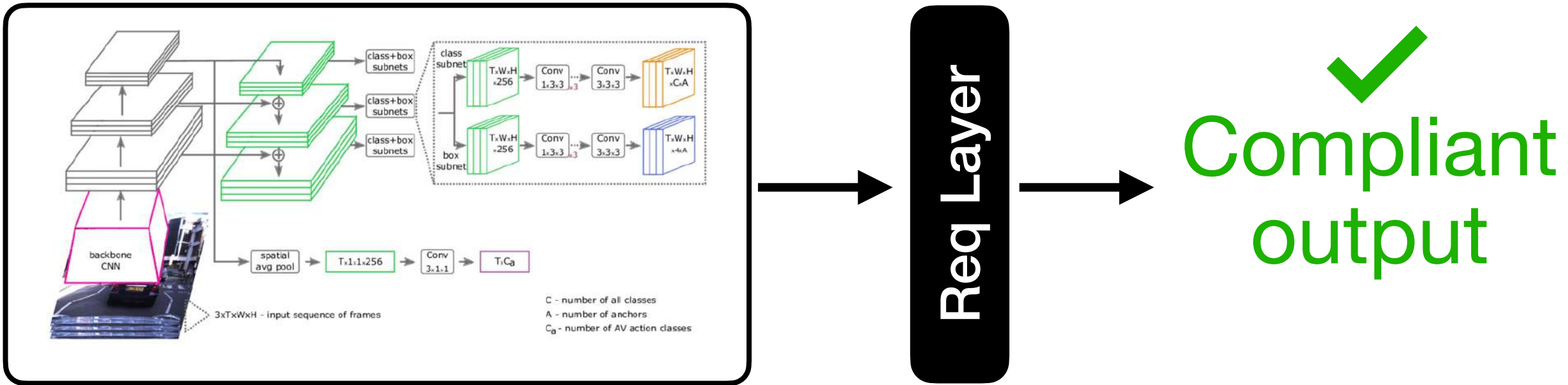


Same Model + Req Layer





The Autonomous Driving Case



Standard Model



Same Model + Req Layer



Pros and Cons



Pros

- **Faster than the other methods (*)**
- Safe wrt to thresholding

Cons

- Not safe wrt to sampling
- Performance depend on ordering

Fuzzy

- Beautiful probabilistic semantics
- ***Safe wrt to sampling***

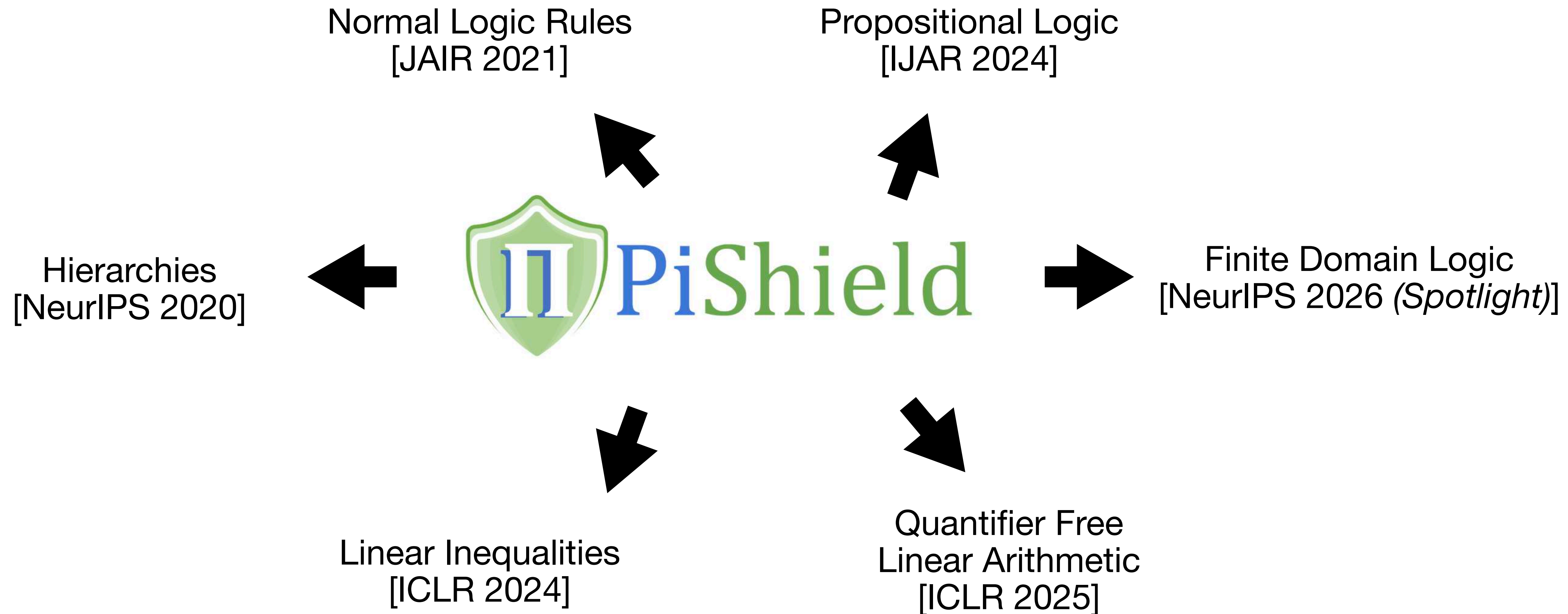
- Slower than other methods (*)
- If datapoint breaks the constraints
gradients explode - network is untrainable

Probabilistic



PiShield 2.0

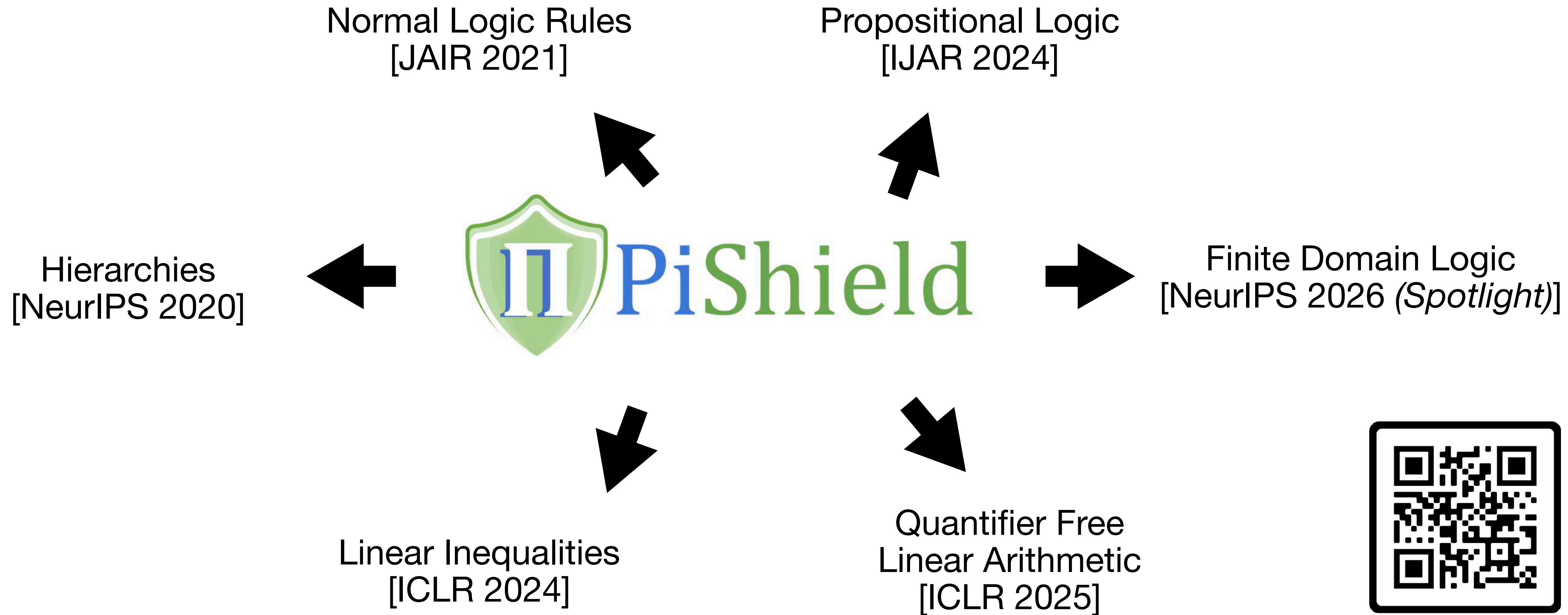
Many constraints, a single package





PiShield 2.0

Many constraints, a single package





PiShield 2.0

Many constraints, a single package

Pip installable package

```
from pishield.shield_layer import build_shield_layer  
layer = build_shield_layer(num_variables, "reqs.txt")  
corrected = layer(predictions)
```





PiShield 2.0

Many constraints, a single package

Pip installable package

```
from pishield.shield_layer import build_shield_layer  
layer = build_shield_layer(num_variables, "reqs.txt")  
corrected = layer(predictions)
```

*You can simply write your constraints in a txt file
and we give you the PyTorch layer*





PiShield 2.0

Many constraints, a single package

Pip installable package

```
from pishield.shield_layer import build_shield_layer  
layer = build_shield_layer(num_variables, "reqs.txt")  
corrected = layer(predictions)
```

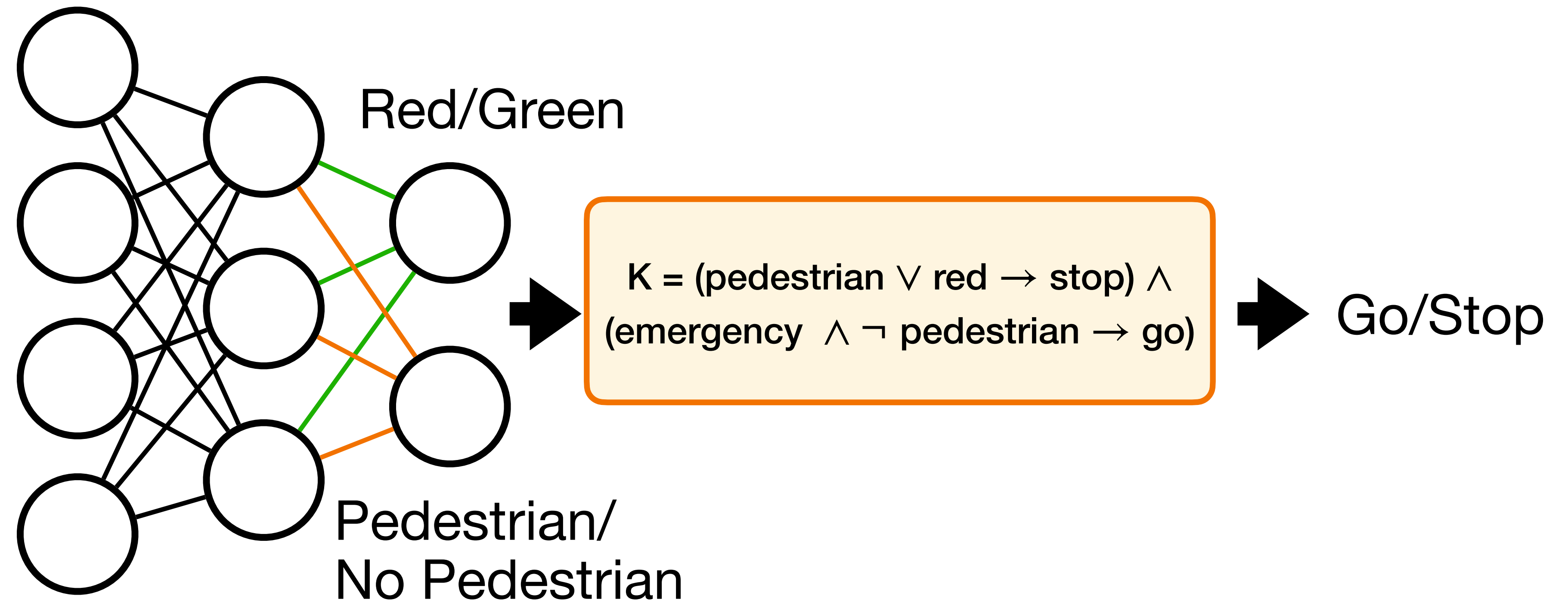
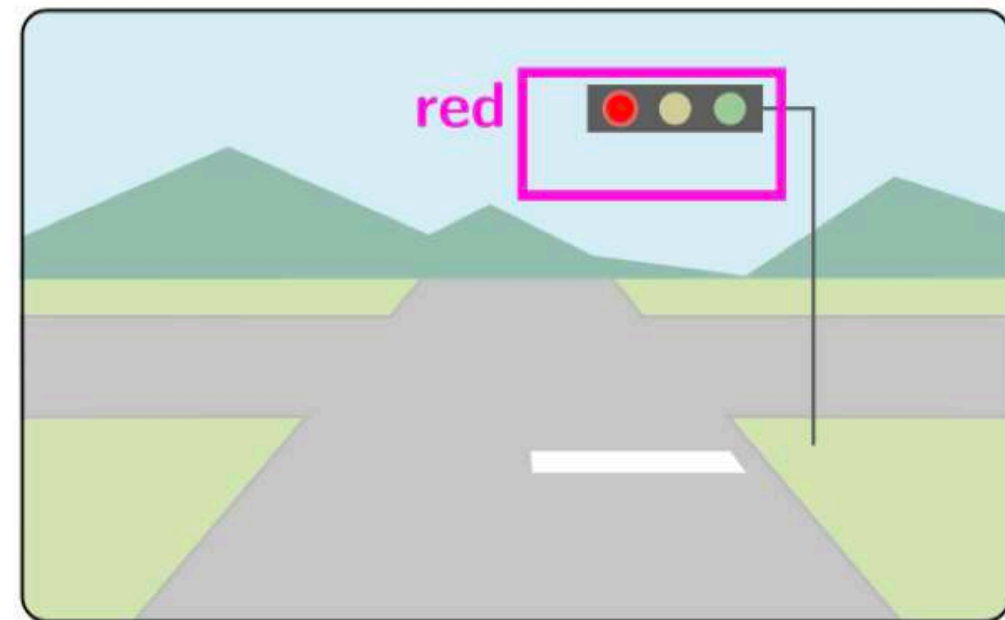
*You can simply write your constraints in a txt file
and we give you the PyTorch layer*

Use the layer as you want!



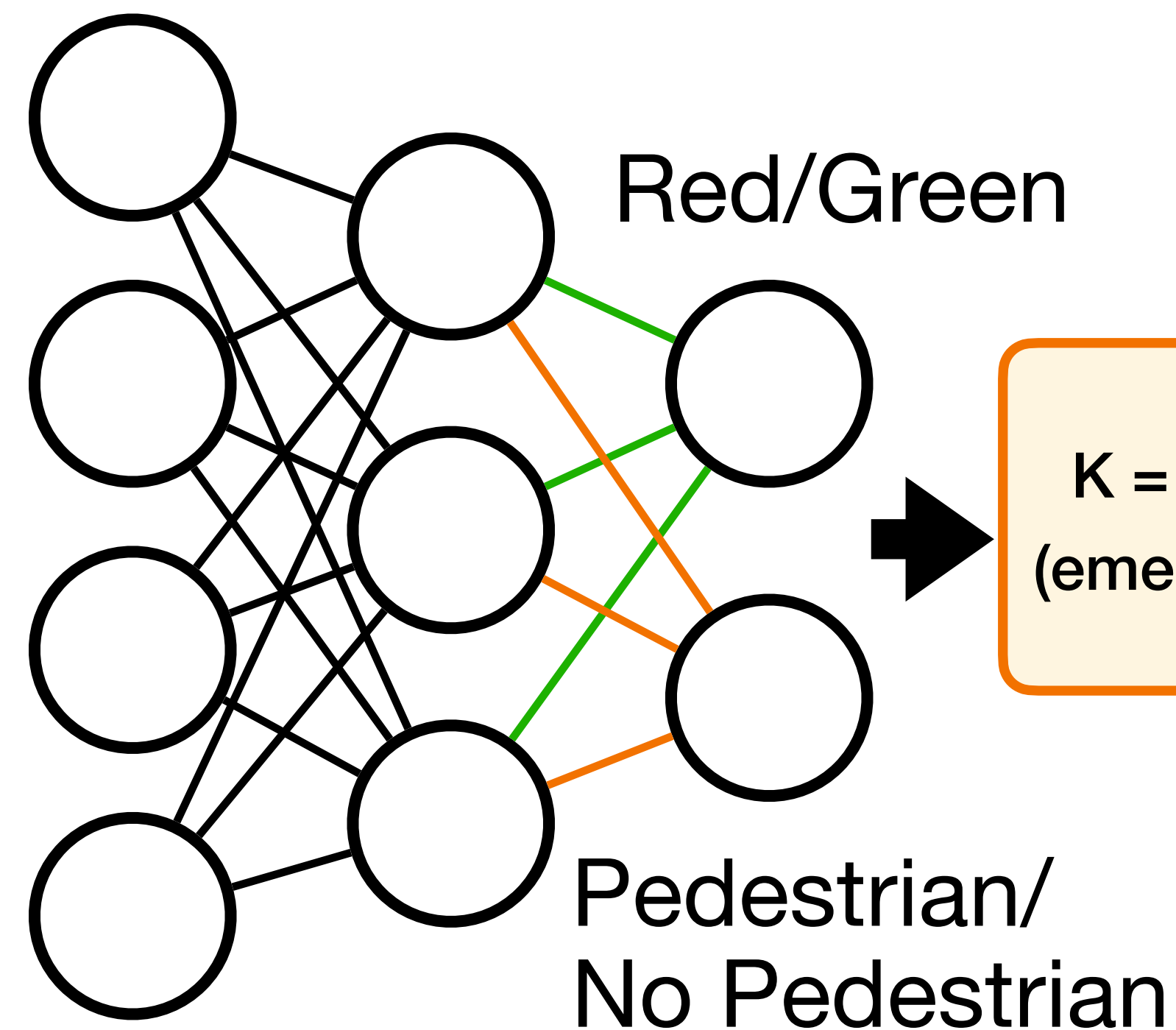
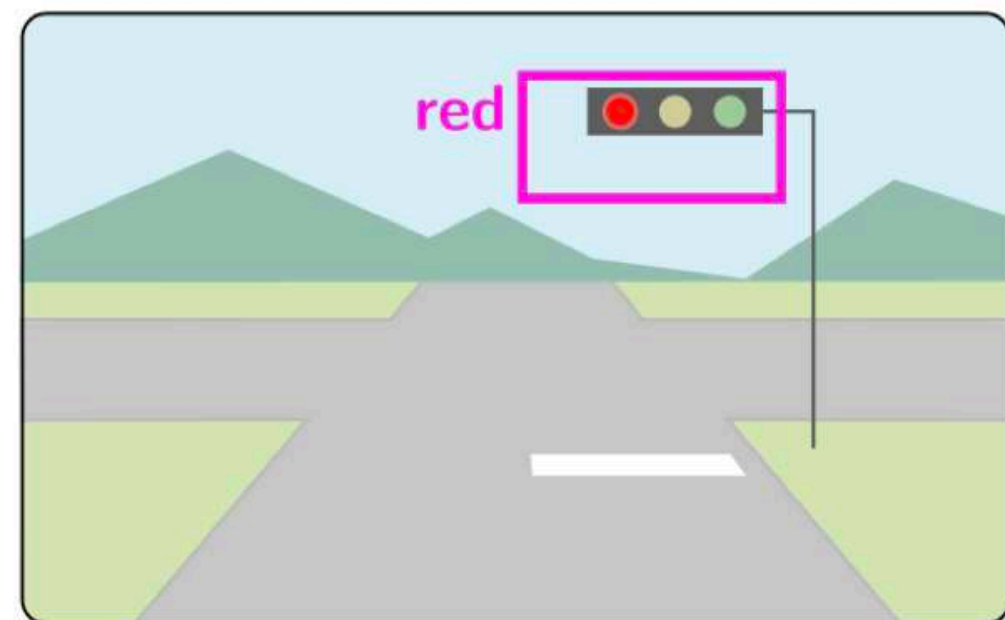


What if we have distant supervision?





What if we have distant supervision?



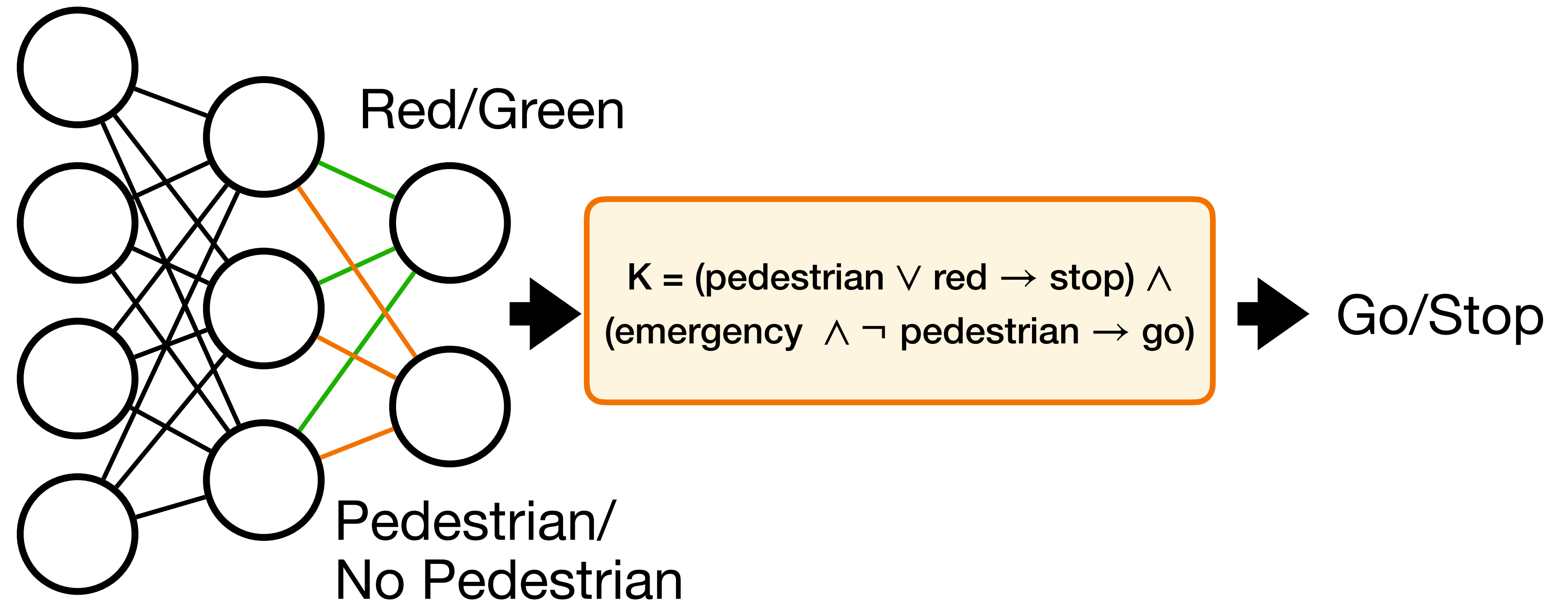
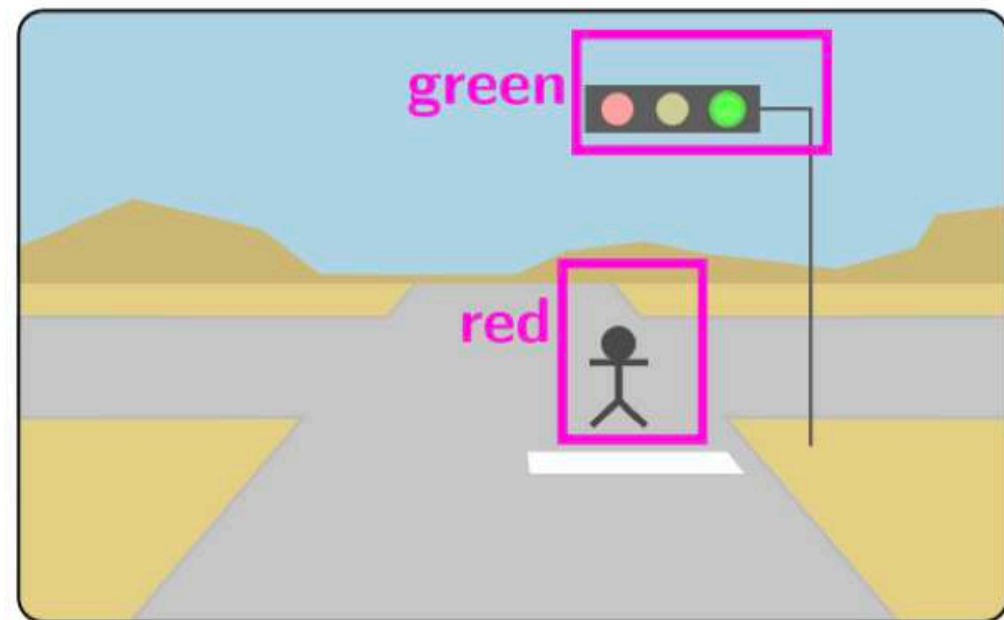
$K = (\text{pedestrian} \vee \text{red} \rightarrow \text{stop}) \wedge$
 $(\text{emergency} \wedge \neg \text{pedestrian} \rightarrow \text{go})$

Go/Stop

*Supervisions
happen only here*

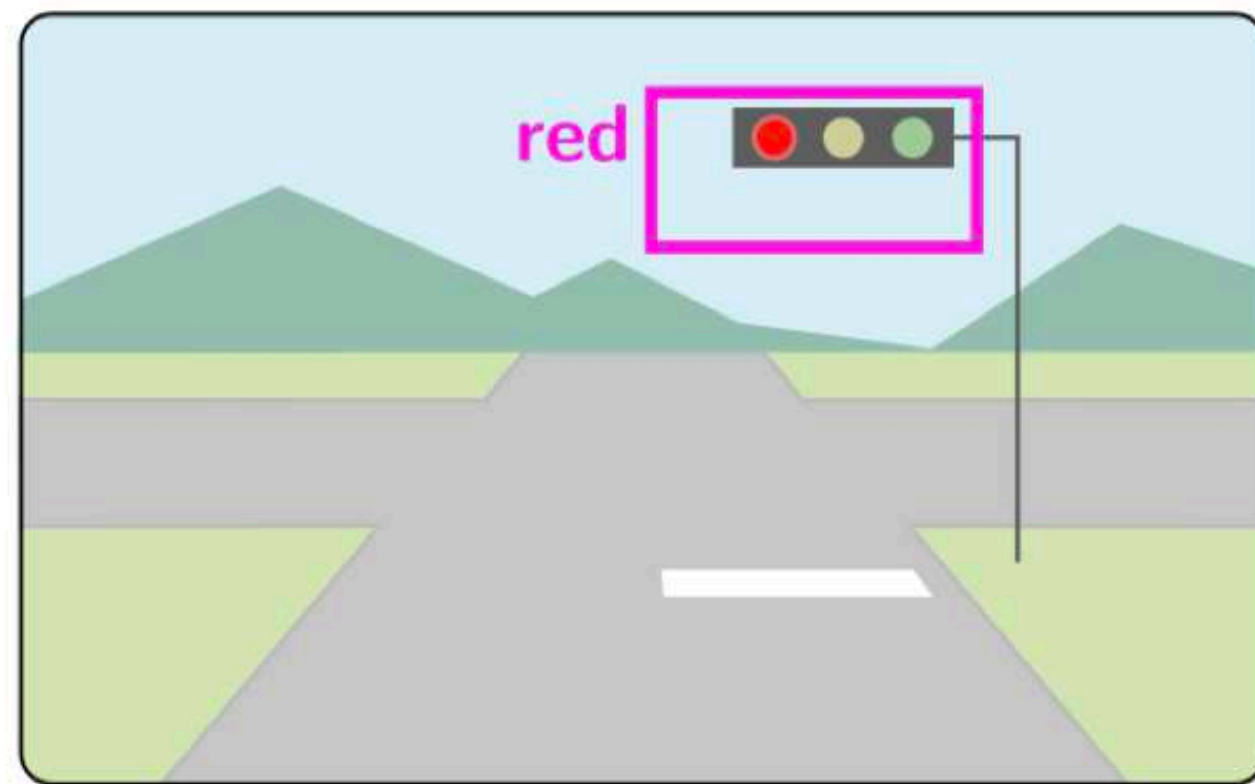


What if we have distant supervision?

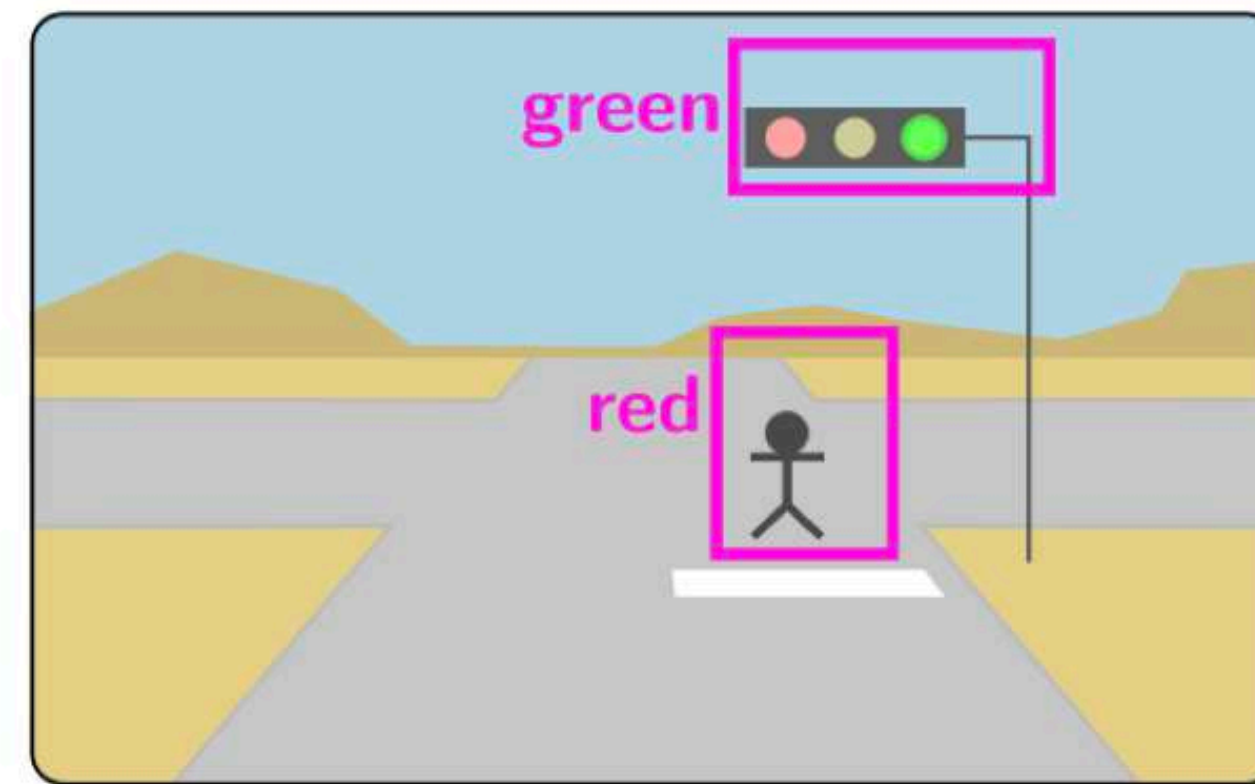




We might learn a shortcut!



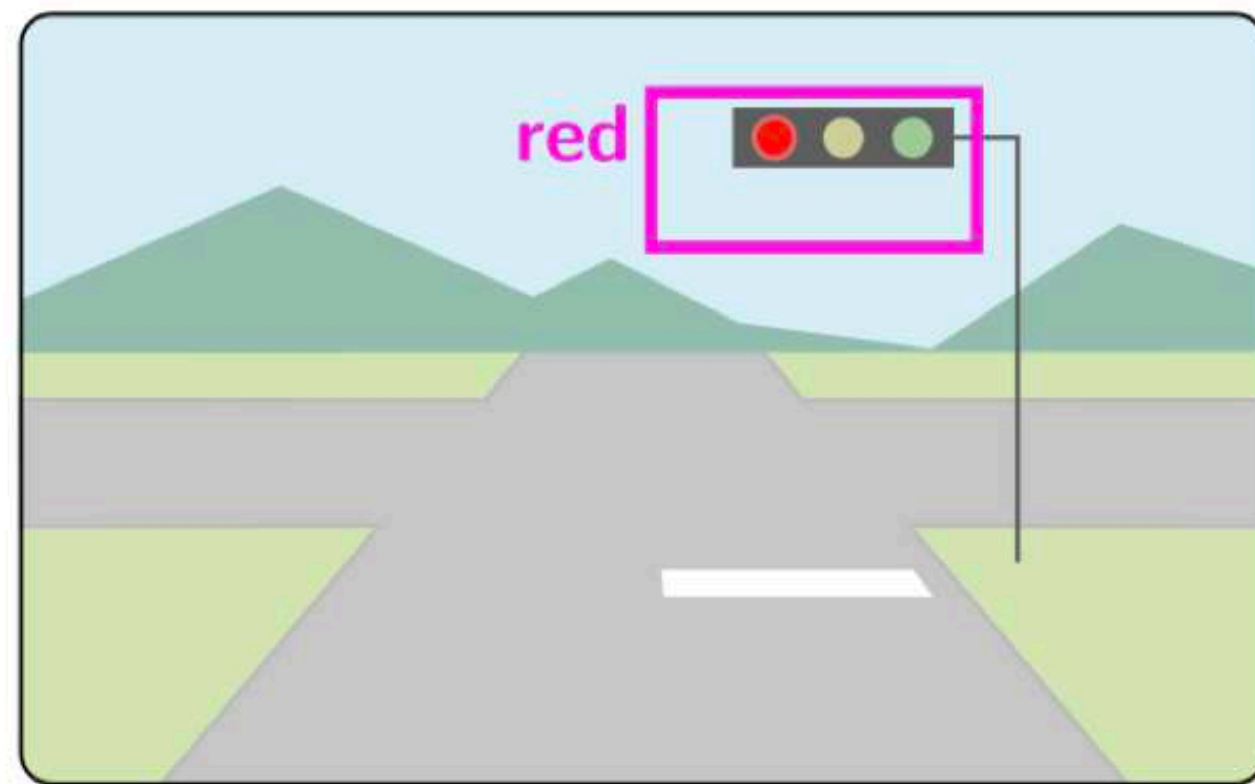
$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓



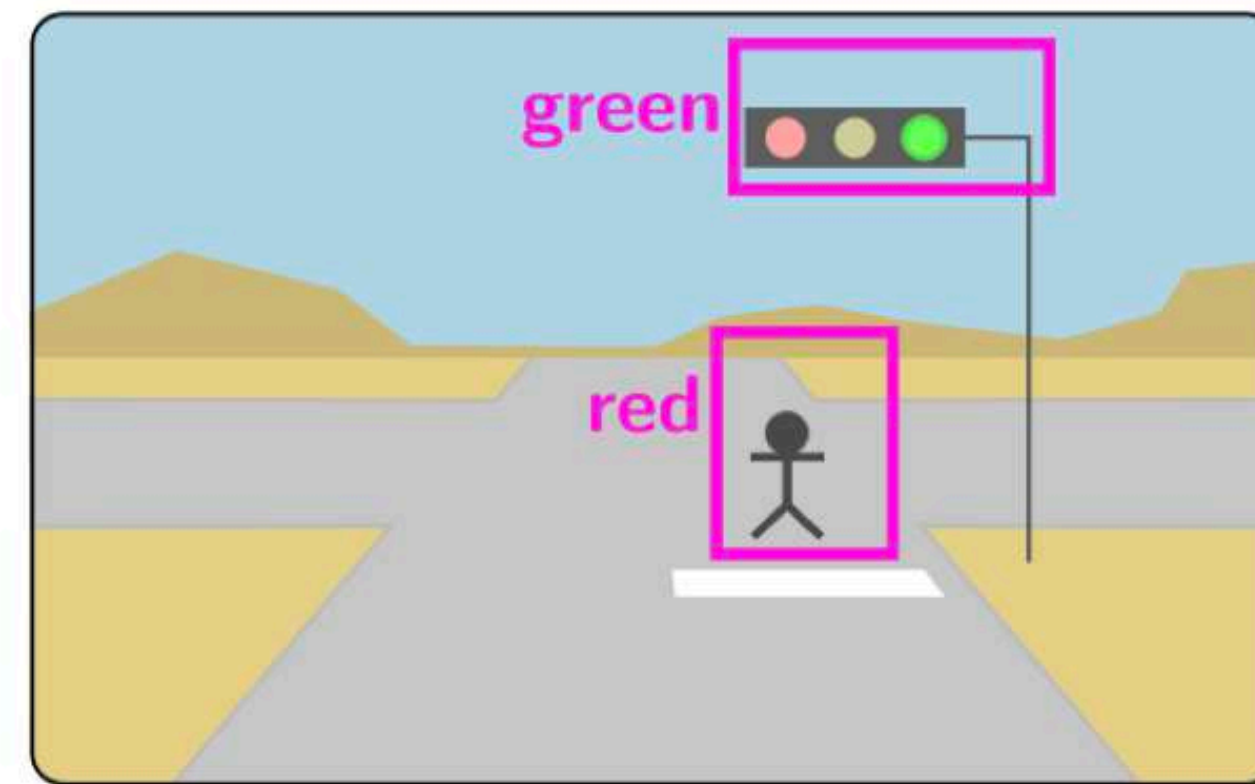
$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓



We might learn a shortcut!

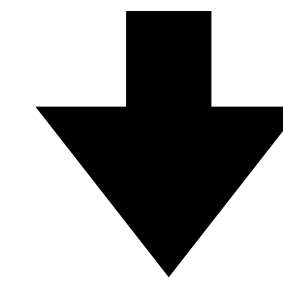


$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓



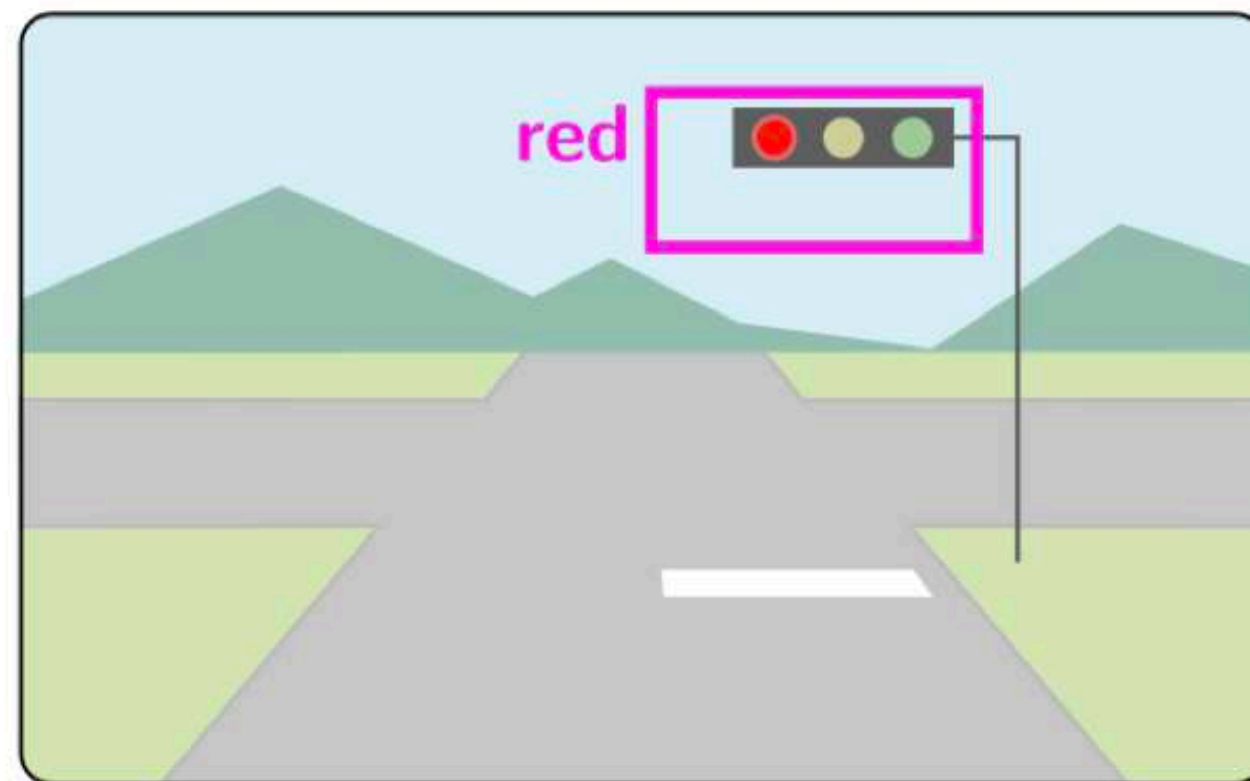
$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓

Predicting pedestrian or red
leads to the same action "Stop"

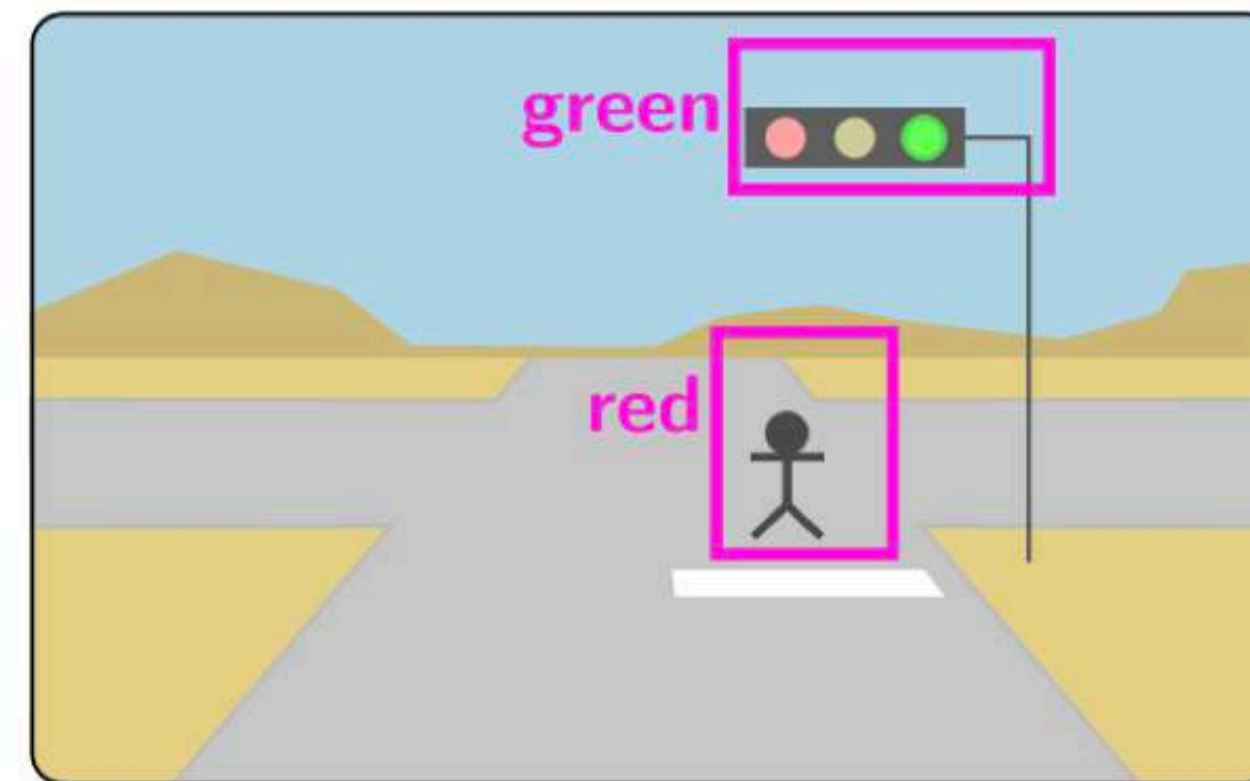


*The model can learn a reasoning
shortcut: predicting red for both
pedestrians and red traffic lights*

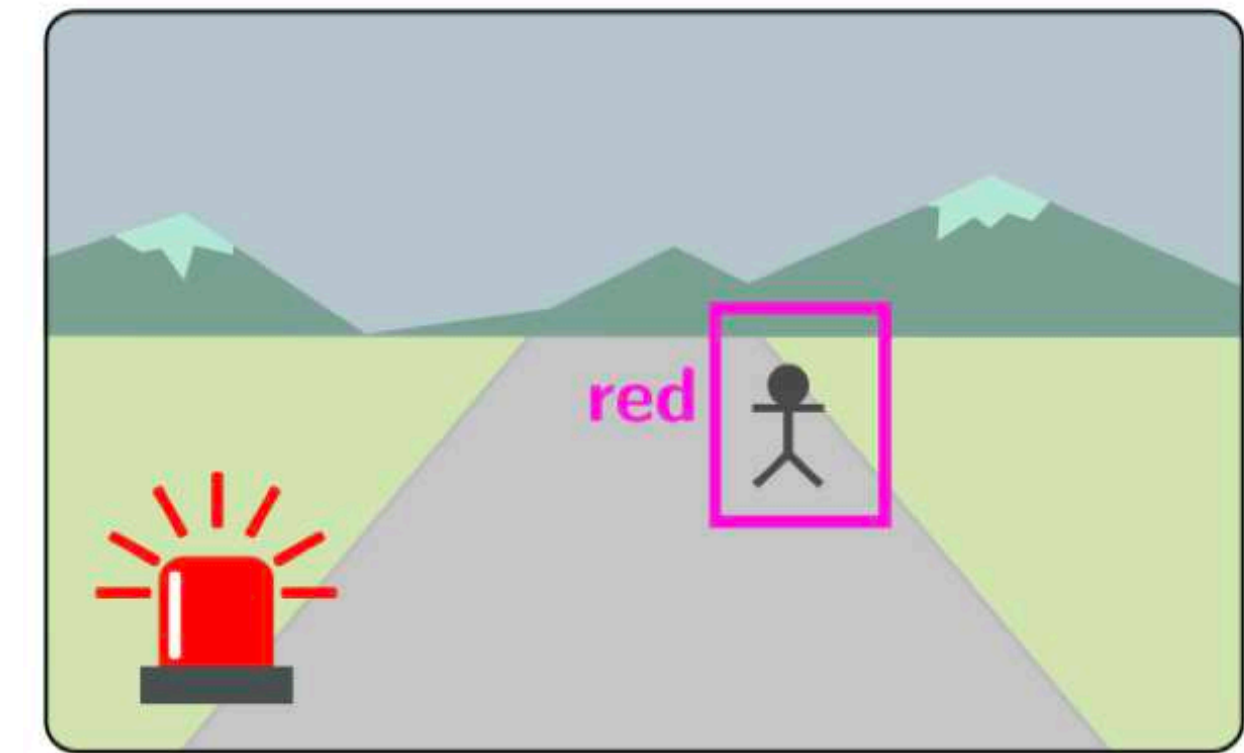
What if we never see an emergency during training?



$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓



$y = \text{stop}$ $\hat{y} = \text{stop}$ ✓

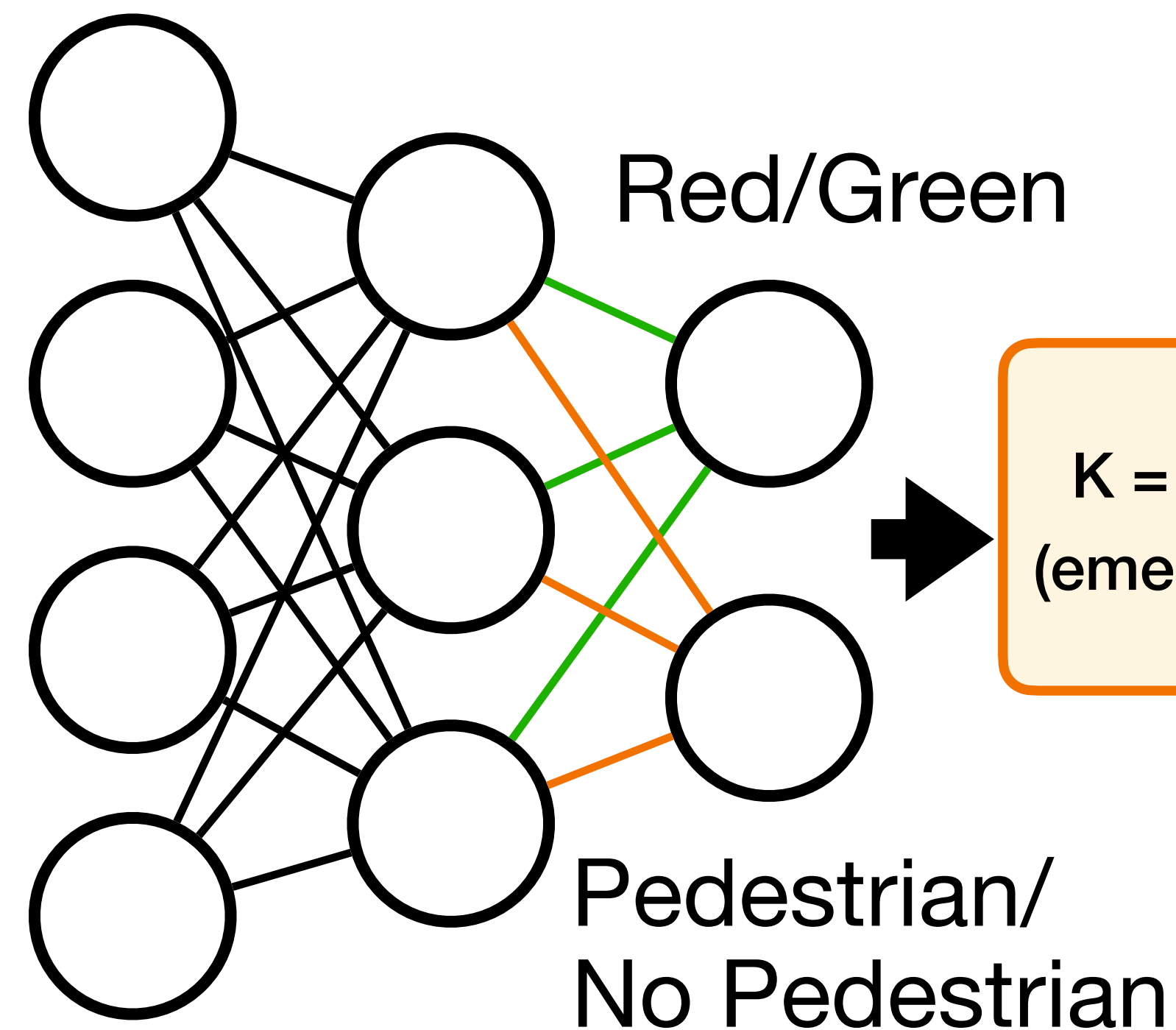
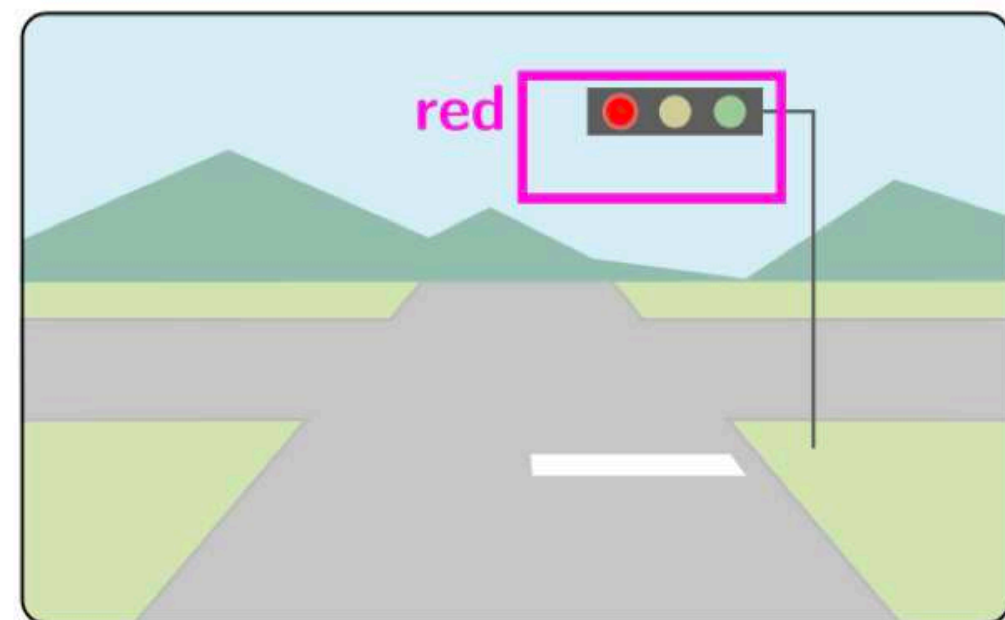


$\hat{y} = \text{go}$ ✗

$K = (\text{pedestrian} \vee \text{red} \rightarrow \text{stop}) \wedge$
 $(\text{emergency} \wedge \neg \text{pedestrian} \rightarrow \text{go})$

The car would just keep going! 😱

What if we have very few intermediate supervisions?

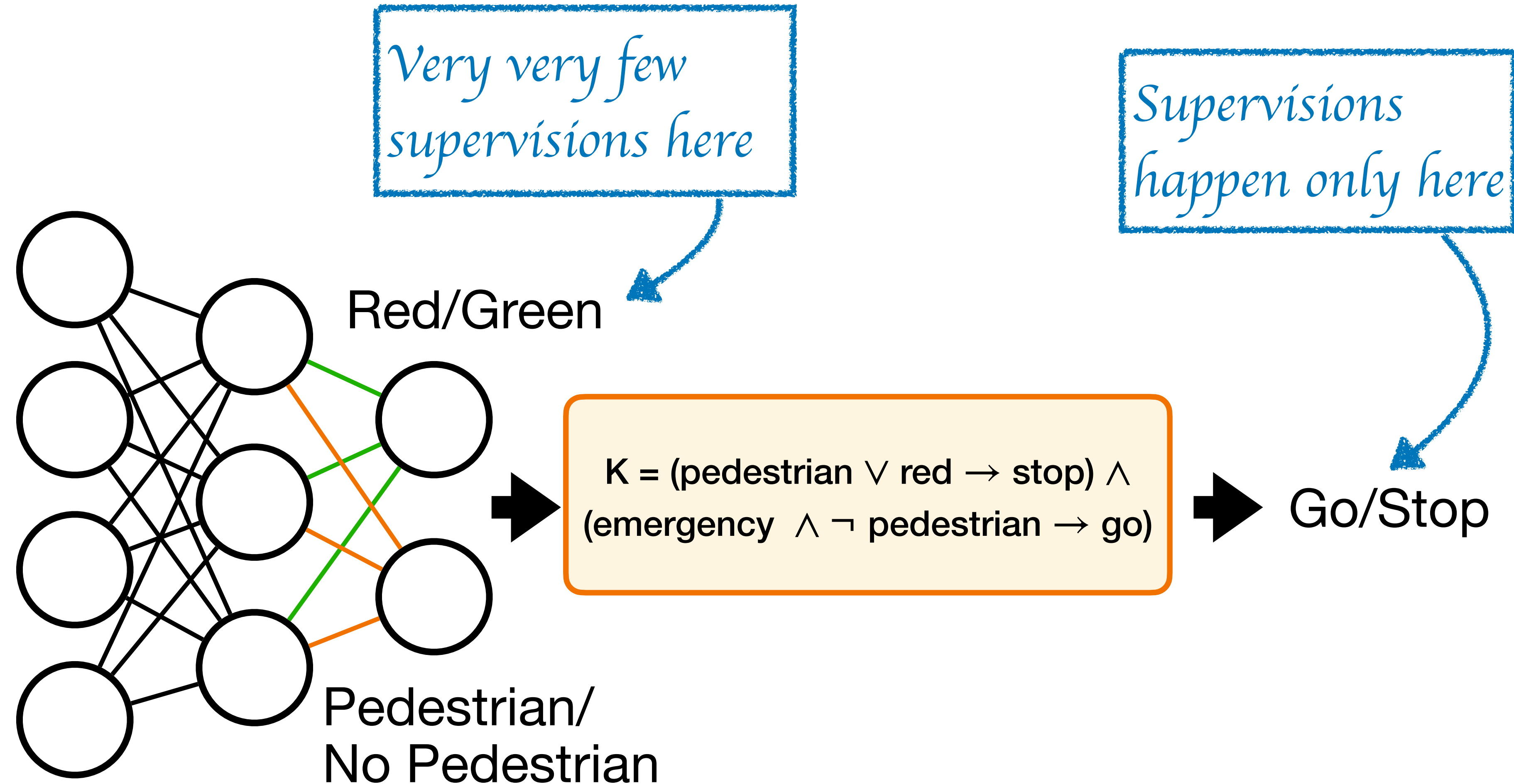
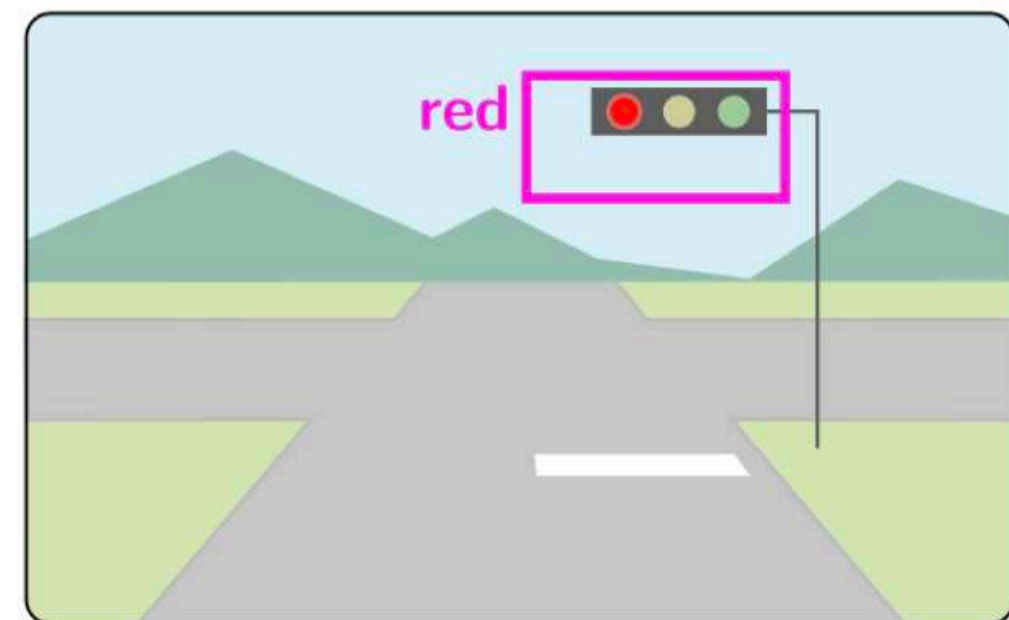


$K = (\text{pedestrian} \vee \text{red} \rightarrow \text{stop}) \wedge$
 $(\text{emergency} \wedge \neg \text{pedestrian} \rightarrow \text{go})$

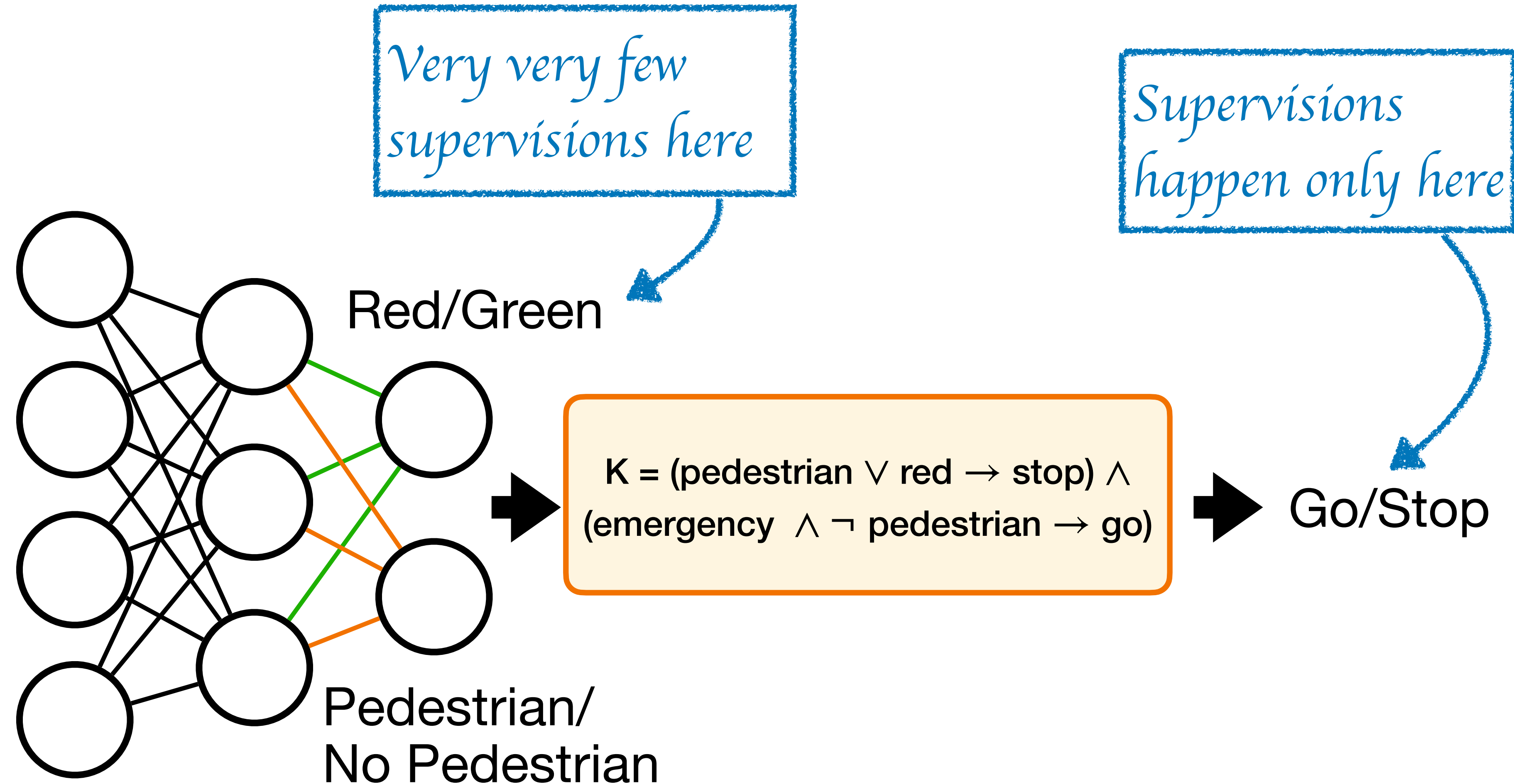
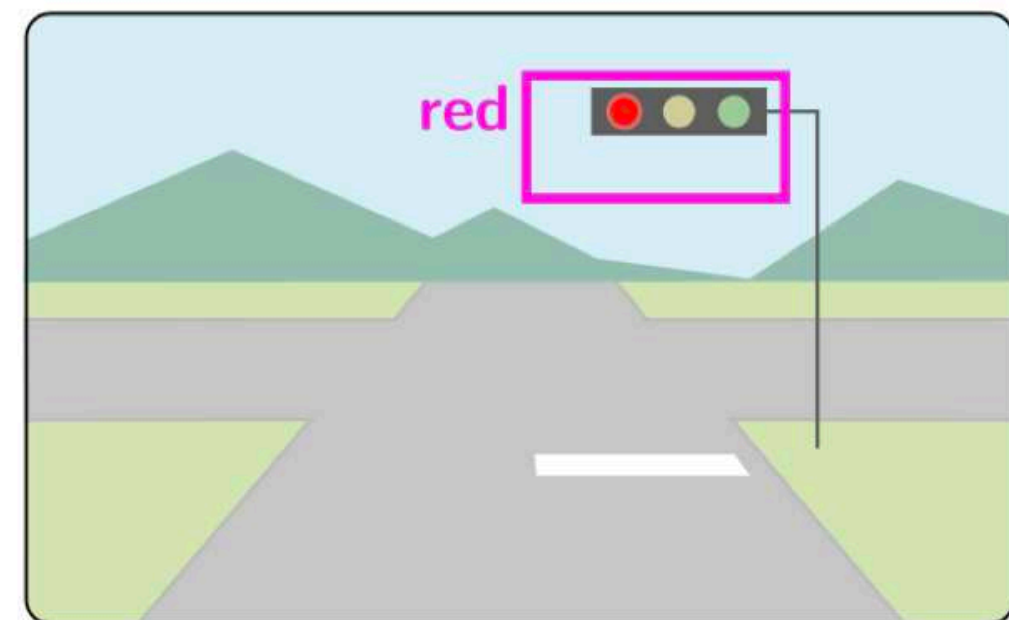
Go/Stop

*Supervisions
happen only here*

What if we have very few intermediate supervisions?



What if we have very few intermediate supervisions?

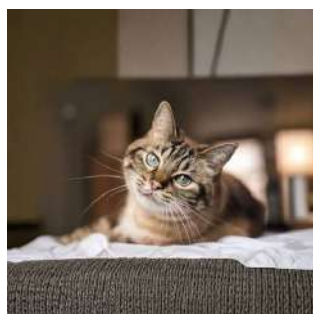
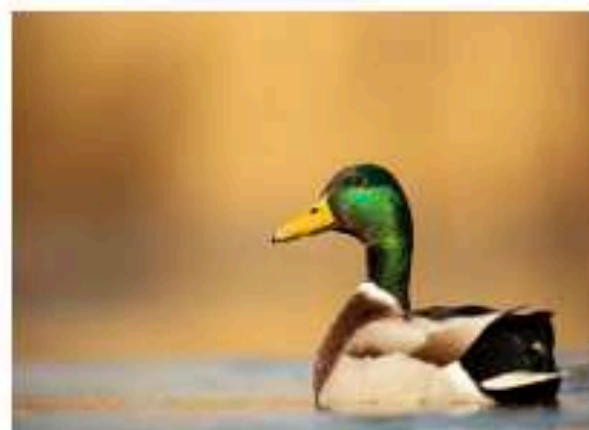
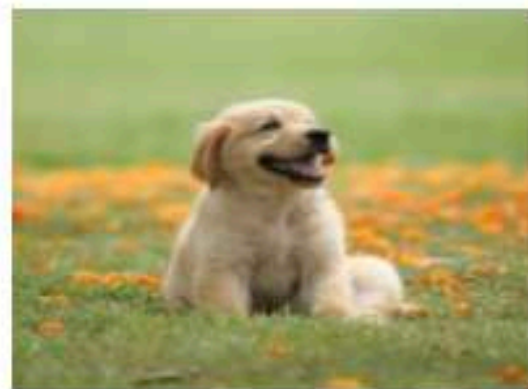


Then we "know" what a pedestrian and a red traffic light look like!



Example

$K = \{\text{Shark} \rightarrow \text{Fish}, \text{Dog} \rightarrow \text{Mammal}, \text{Cat} \rightarrow \text{Mammal}, \text{Duck} \rightarrow \text{Mammal}\}$





Example

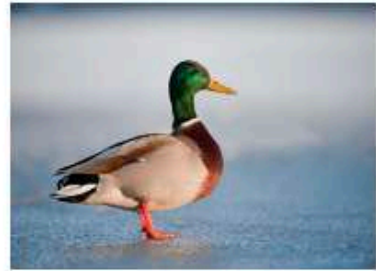
$K = \{ \text{Shark} \rightarrow \text{Fish}, \text{Dog} \rightarrow \text{Mammal}, \text{Cat} \rightarrow \text{Mammal}, \text{Duck} \rightarrow \text{Mammal} \}$



Shark,
Fish



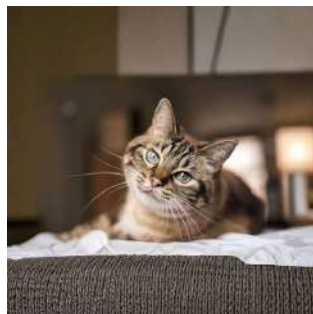
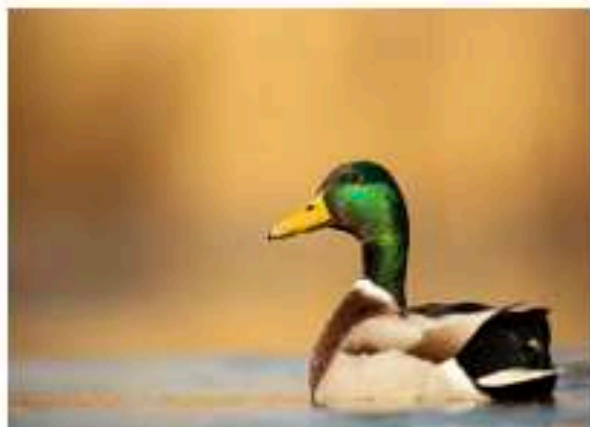
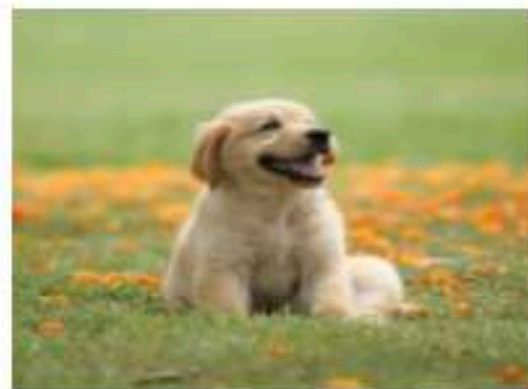
Dog,
Mammal



Duck
Bird

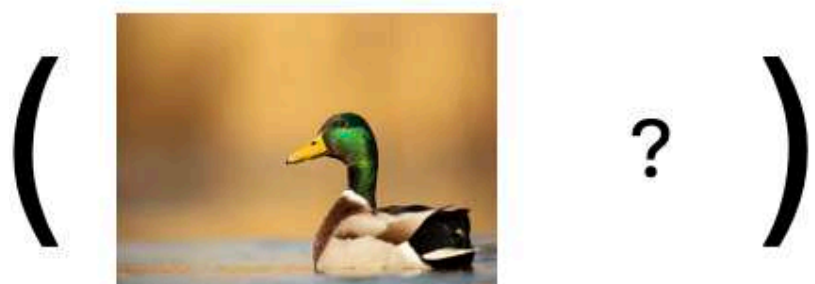
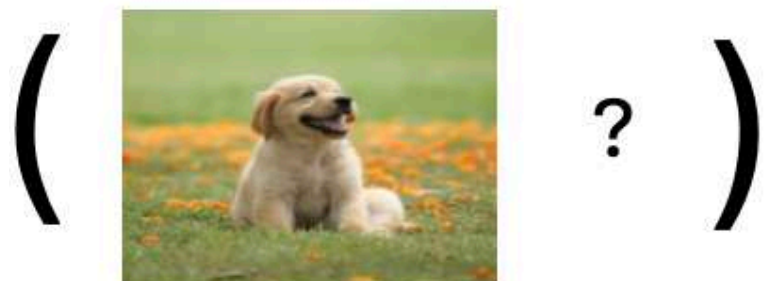
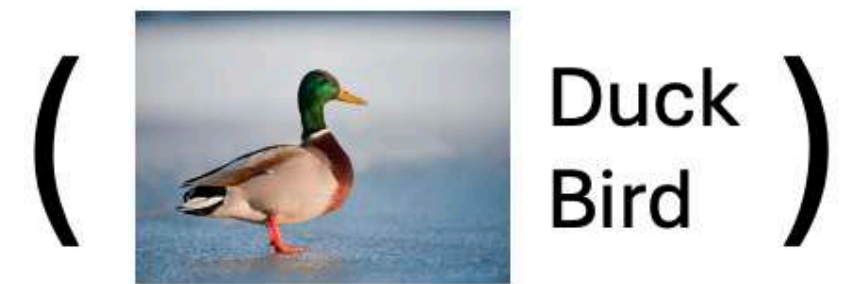
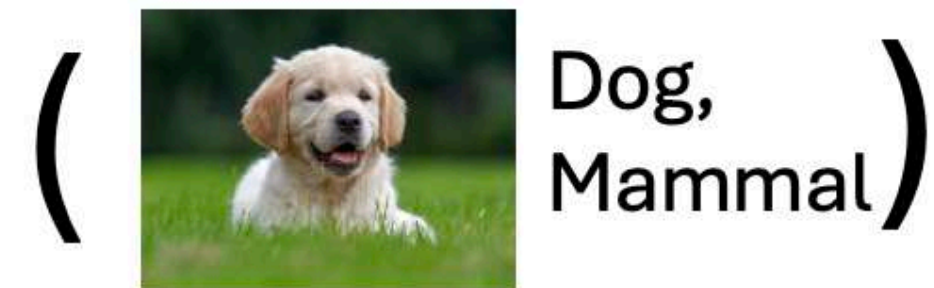


Cat
Mammal





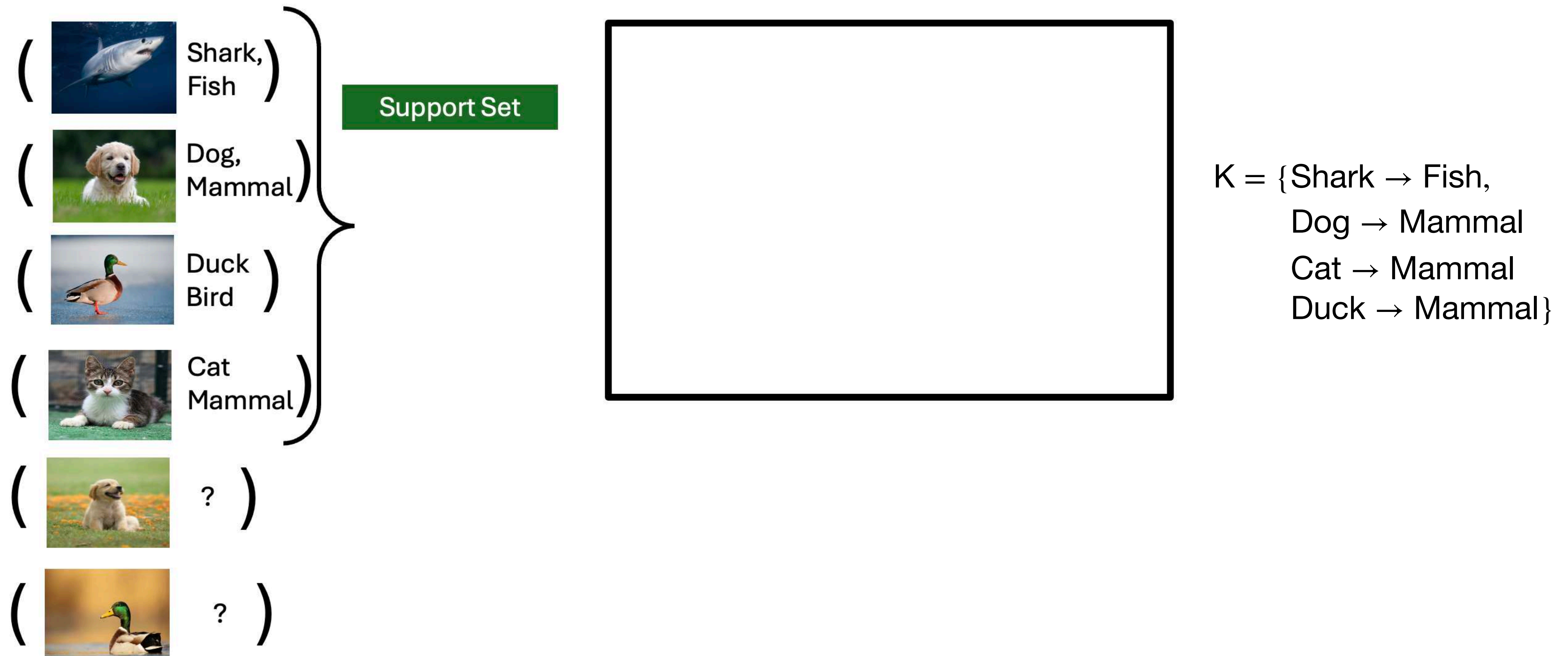
Prototypical networks are the answer!



$K = \{ \text{Shark} \rightarrow \text{Fish},$
 $\text{Dog} \rightarrow \text{Mammal}$
 $\text{Cat} \rightarrow \text{Mammal}$
 $\text{Duck} \rightarrow \text{Mammal} \}$

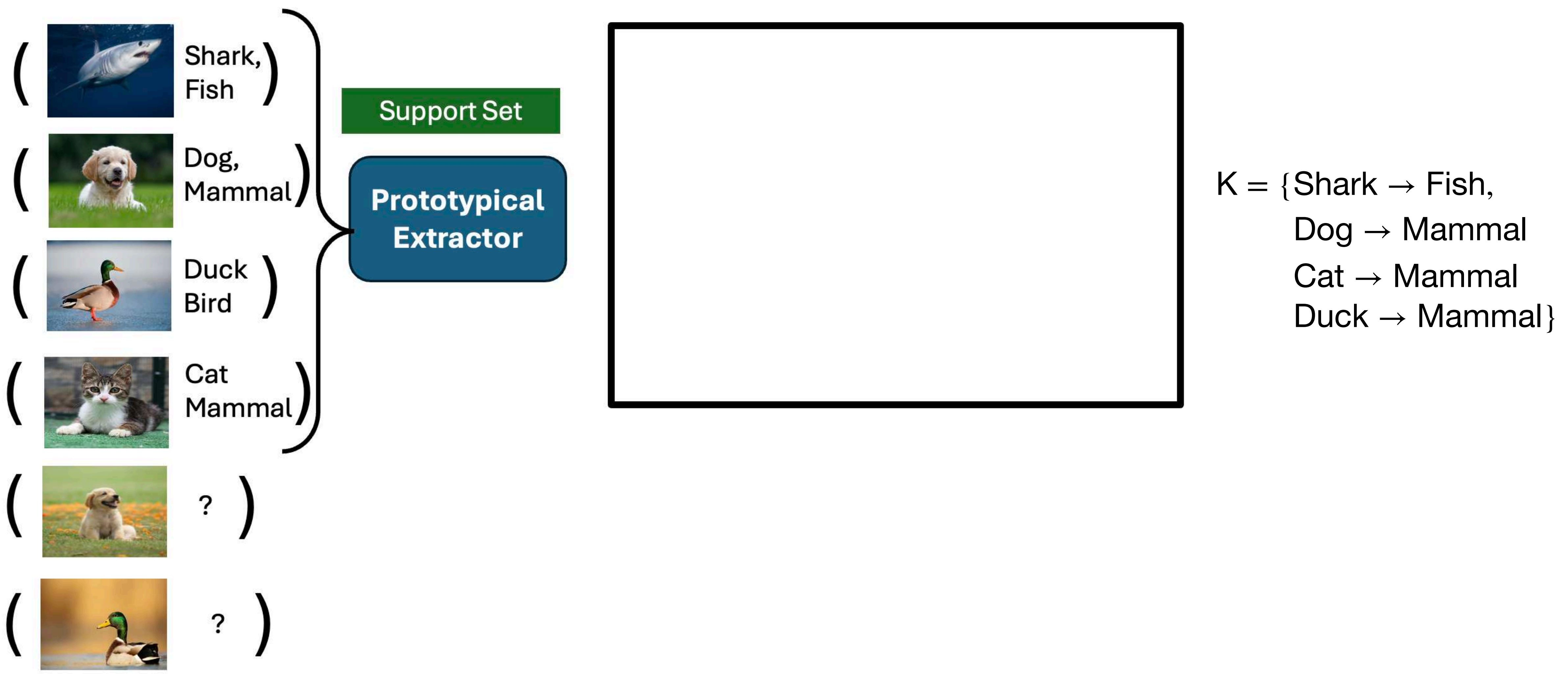


Prototypical networks are the answer!



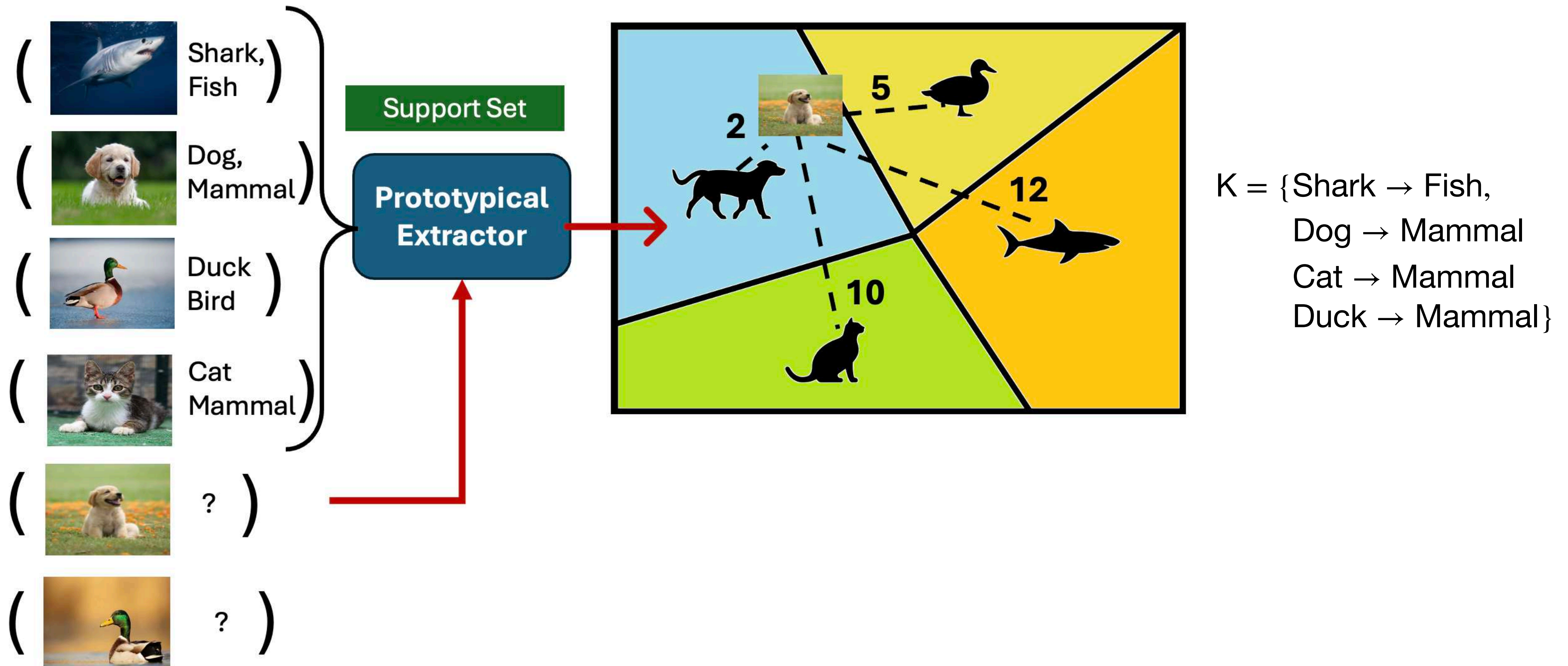


Prototypical networks are the answer!



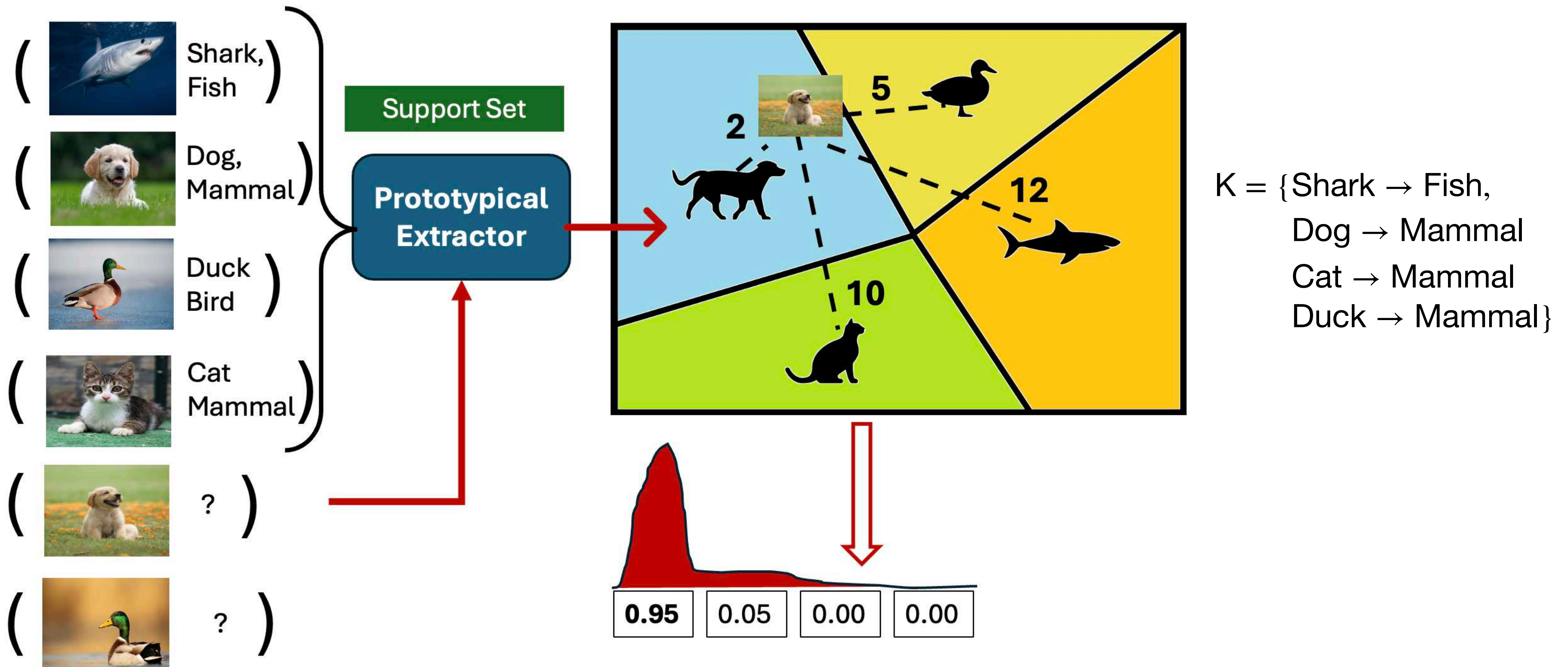


Prototypical networks are the answer!



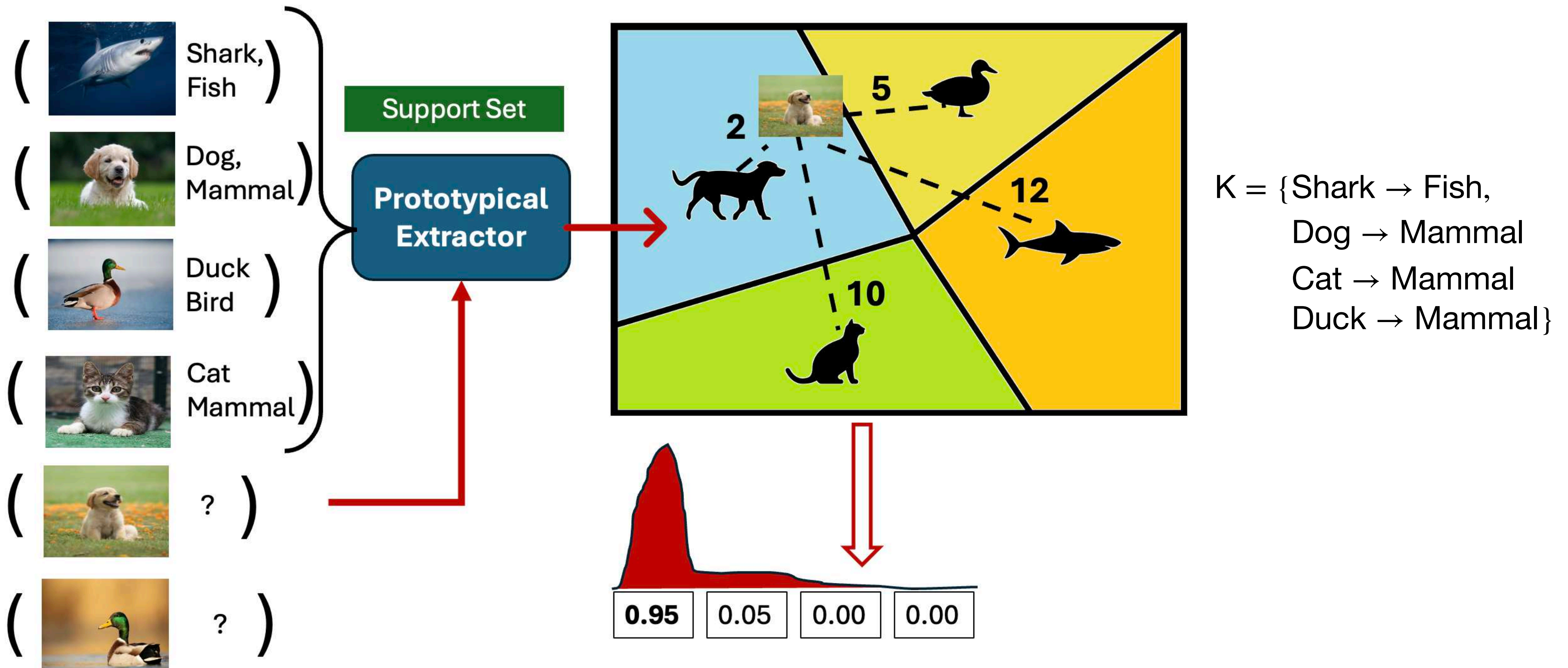


Prototypical networks are the answer!



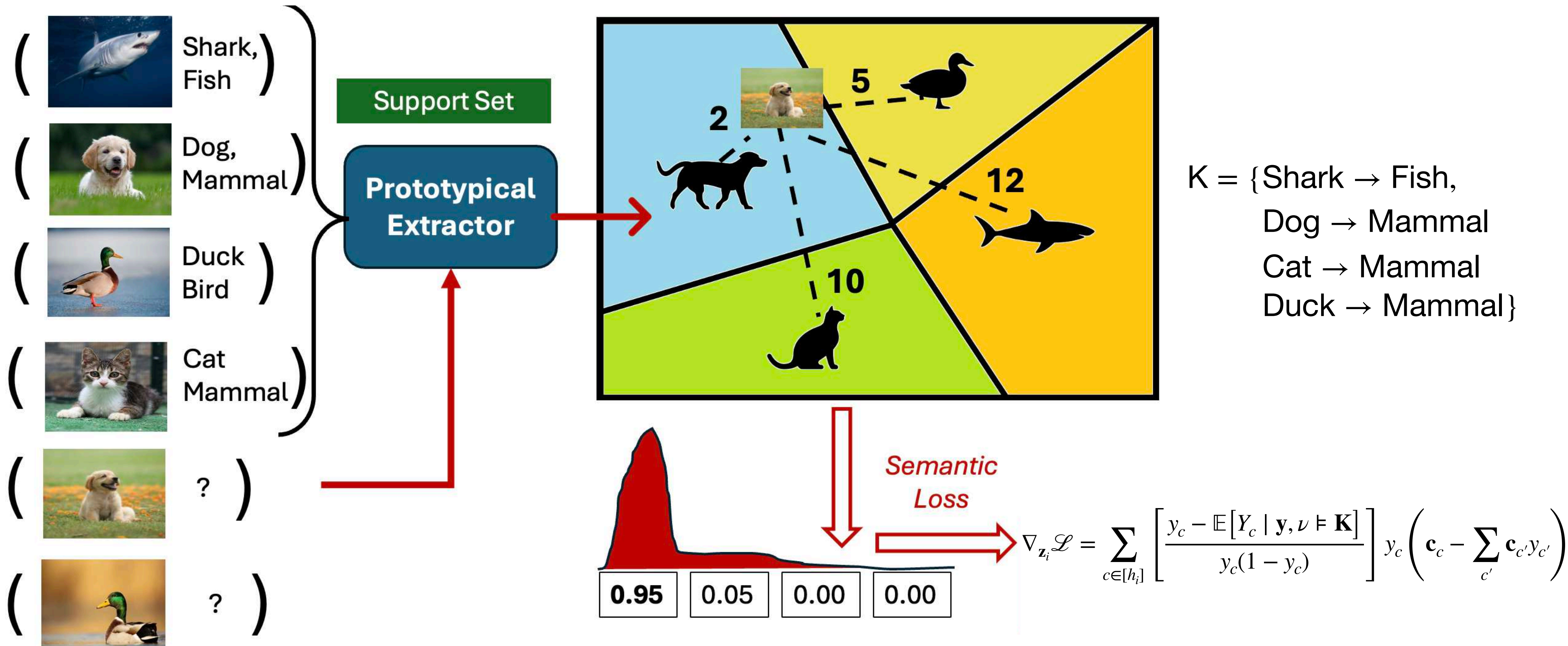


Prototypical networks are the answer!



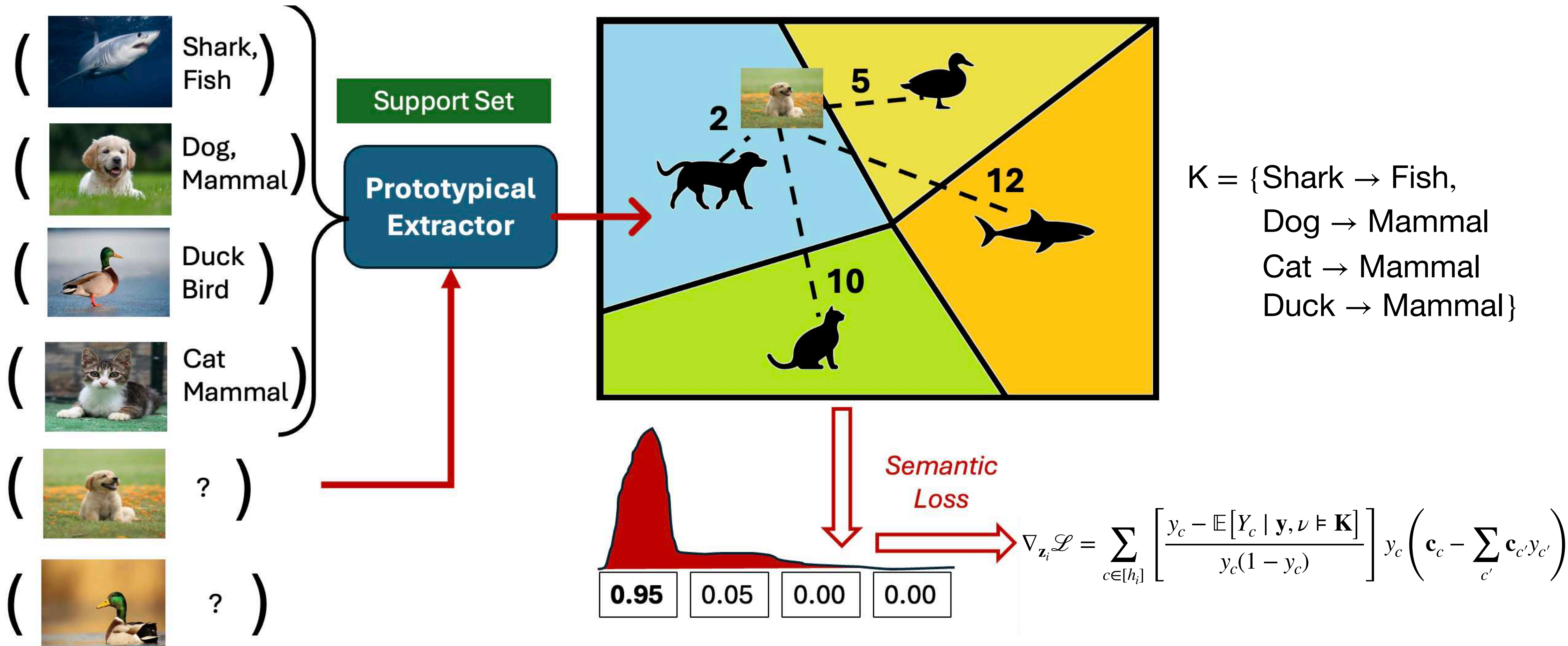


Prototypical networks are the answer!





Prototypical networks are the answer!





Error Analysis

Prototypical Networks vs Standard Networks

Suppose we use the **Semantic Loss** to train our network

Standard Networks

$$\frac{\partial \mathcal{L}^{\text{NeSy}}(\mathbf{y}, \mathbf{K})}{\partial y_c} = \frac{y_c - \mathbb{E}[Y_c \mid \nu \models \mathbf{K}, \mathbf{y}]}{y_c(1 - y_c)}$$

Prototypical Networks

$$\nabla_{\mathbf{z}_i} \mathcal{L} = \sum_{c \in [h_i]} \left[\frac{y_c - \mathbb{E}[Y_c \mid \mathbf{y}, \nu \models \mathbf{K}]}{y_c(1 - y_c)} \right] y_c \left(\mathbf{c}_c - \sum_{c'} \mathbf{c}_{c'} y_{c'} \right)$$



Error Analysis

Prototypical Networks vs Standard Networks

Suppose we use the **Semantic Loss** to train our network

Standard Networks

$$\frac{\partial \mathcal{L}^{\text{NeSy}}(\mathbf{y}, \mathbf{K})}{\partial y_c} = \frac{y_c - \mathbb{E}[Y_c \mid \nu \models \mathbf{K}, \mathbf{y}]}{y_c(1 - y_c)}$$

The error is determined only by the prediction and the symbolic constraint.

Prototypical Networks

$$\nabla_{\mathbf{z}_i} \mathcal{L} = \sum_{c \in [h_i]} \left[\frac{y_c - \mathbb{E}[Y_c \mid \mathbf{y}, \nu \models \mathbf{K}]}{y_c(1 - y_c)} \right] y_c \left(\mathbf{c}_c - \sum_{c'} \mathbf{c}_{c'} y_{c'} \right)$$



Error Analysis

Prototypical Networks vs Standard Networks

Suppose we use the **Semantic Loss** to train our network

Standard Networks

$$\frac{\partial \mathcal{L}^{\text{NeSy}}(\mathbf{y}, \mathbf{K})}{\partial y_c} = \frac{y_c - \mathbb{E}[Y_c \mid \nu \models \mathbf{K}, \mathbf{y}]}{y_c(1 - y_c)}$$

The error is determined only by the prediction and the symbolic constraint.

Prototypical Networks

$$\nabla_{\mathbf{z}_i} \mathcal{L} = \sum_{c \in [h_i]} \left[\frac{y_c - \mathbb{E}[Y_c \mid \mathbf{y}, \nu \models \mathbf{K}]}{y_c(1 - y_c)} \right] y_c \left(\mathbf{c}_c - \sum_{c'} \mathbf{c}_{c'} y_{c'} \right)$$



Error Analysis

Prototypical Networks vs Standard Networks

Suppose we use the **Semantic Loss** to train our network

Standard Networks

$$\frac{\partial \mathcal{L}^{\text{NeSy}}(\mathbf{y}, \mathbf{K})}{\partial y_c} = \frac{y_c - \mathbb{E}[Y_c \mid \nu \models \mathbf{K}, \mathbf{y}]}{y_c(1 - y_c)}$$

The error is determined only by the prediction and the symbolic constraint.

Prototypical Networks

$$\nabla_{\mathbf{z}_i} \mathcal{L} = \sum_{c \in [h_i]} \left[\frac{y_c - \mathbb{E}[Y_c \mid \mathbf{y}, \nu \models \mathbf{K}]}{y_c(1 - y_c)} \right] y_c \left(\mathbf{c}_c - \sum_{c'} \mathbf{c}_{c'} y_{c'} \right)$$

The same constraint error is now mediated by the geometry of the prototypes.



RSBench: a controlled benchmark

Mnist-EvenOdd

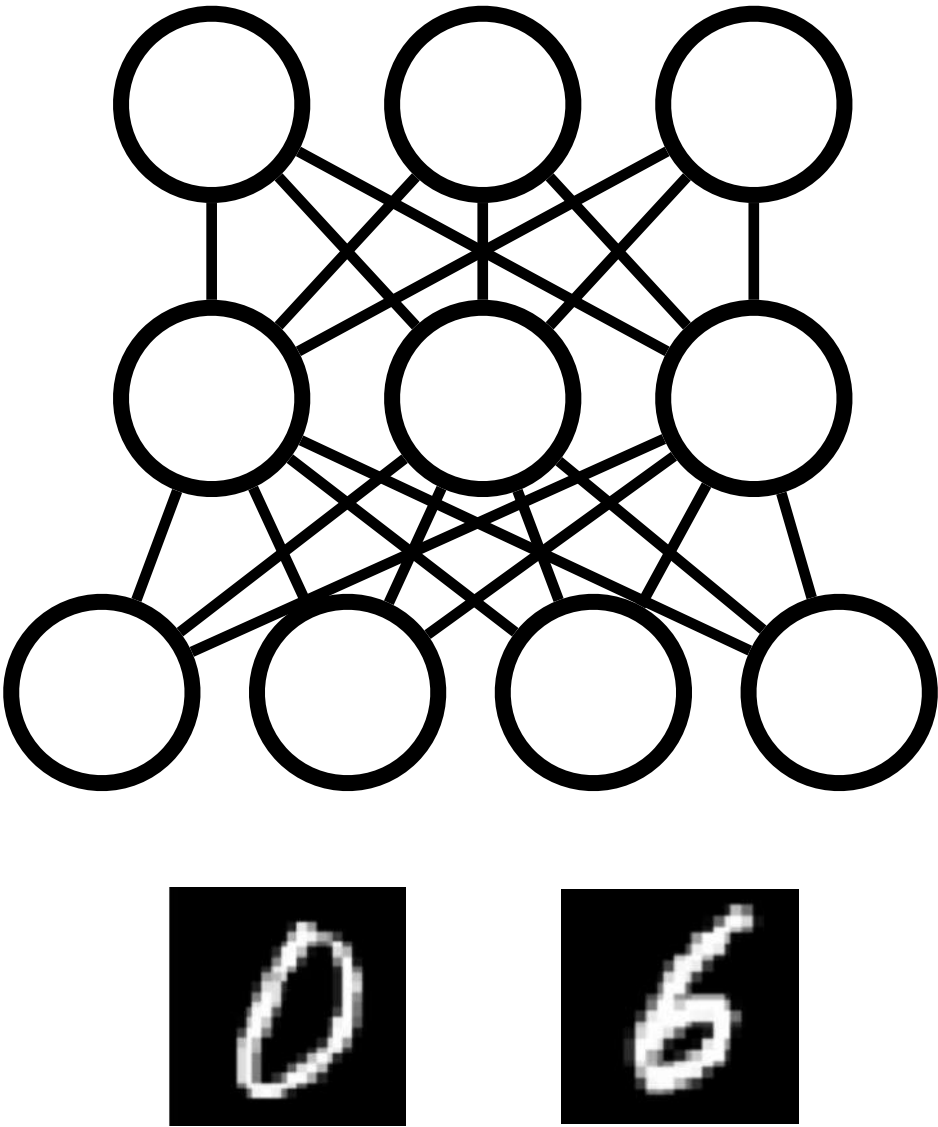
Example Pair	Label	Example Pair	Label
( , )	6	( , )	6
( , )	10	( , )	10
( , )	10	( , )	10
( , )	12	( , )	12



RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
( , )	6	( , )	6
( , )	10	( , )	10
( , )	10	( , )	10
( , )	12	( , )	12

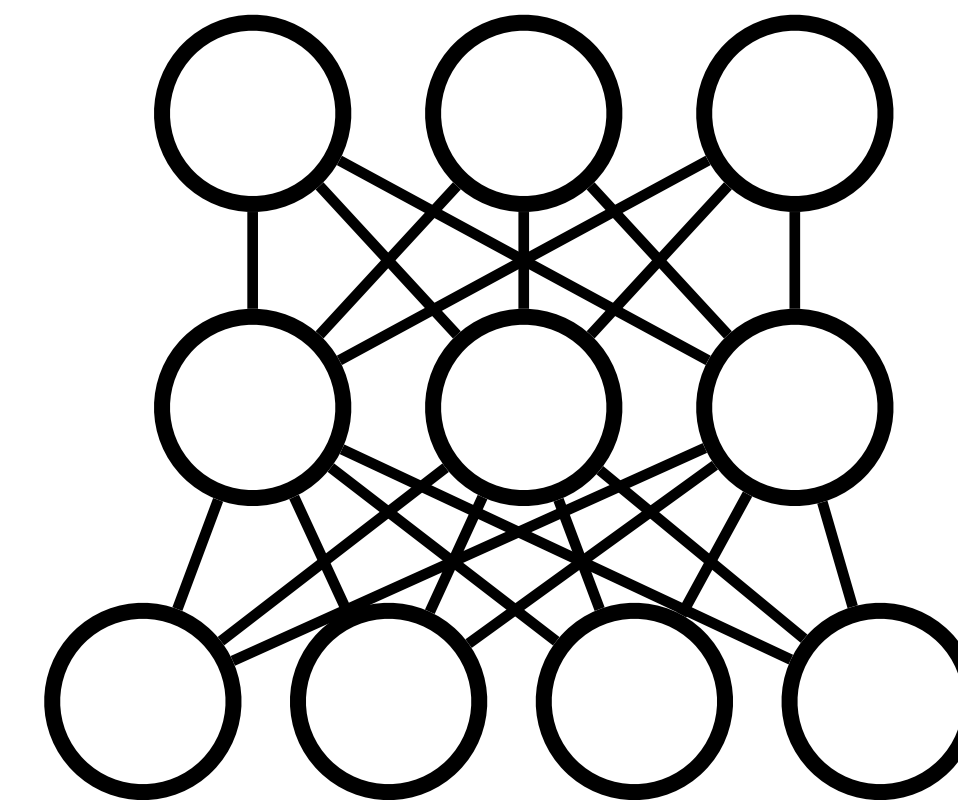
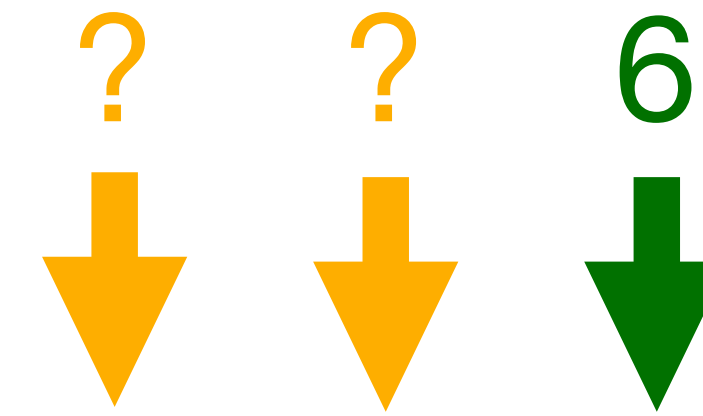




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
( , )	6	( , )	6
( , )	10	( , )	10
( , )	10	( , )	10
( , )	12	( , )	12

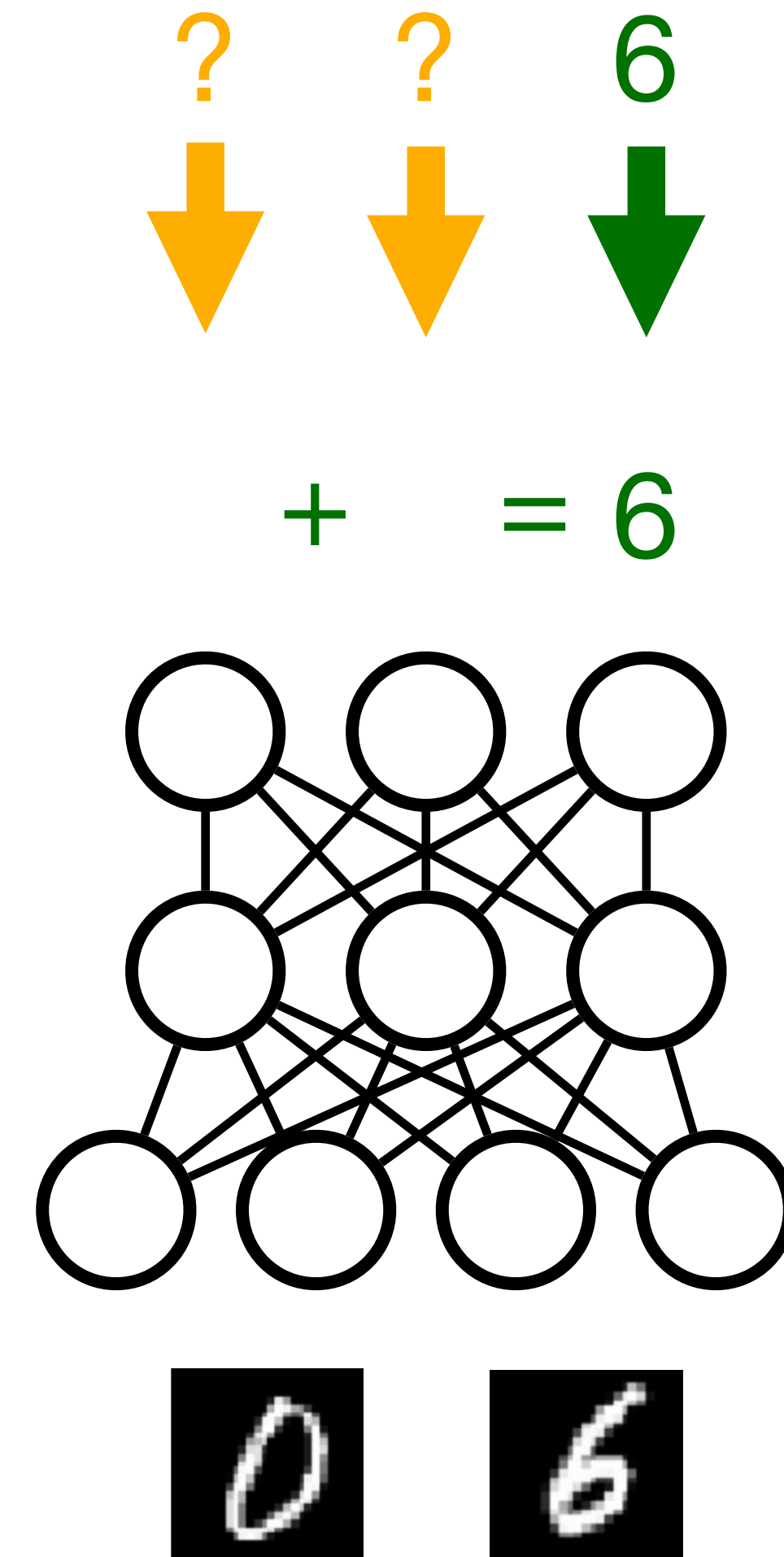




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
(0, 6)	6	(1, 5)	6
(2, 8)	10	(3, 7)	10
(4, 6)	10	(1, 9)	10
(4, 8)	12	(3, 9)	12

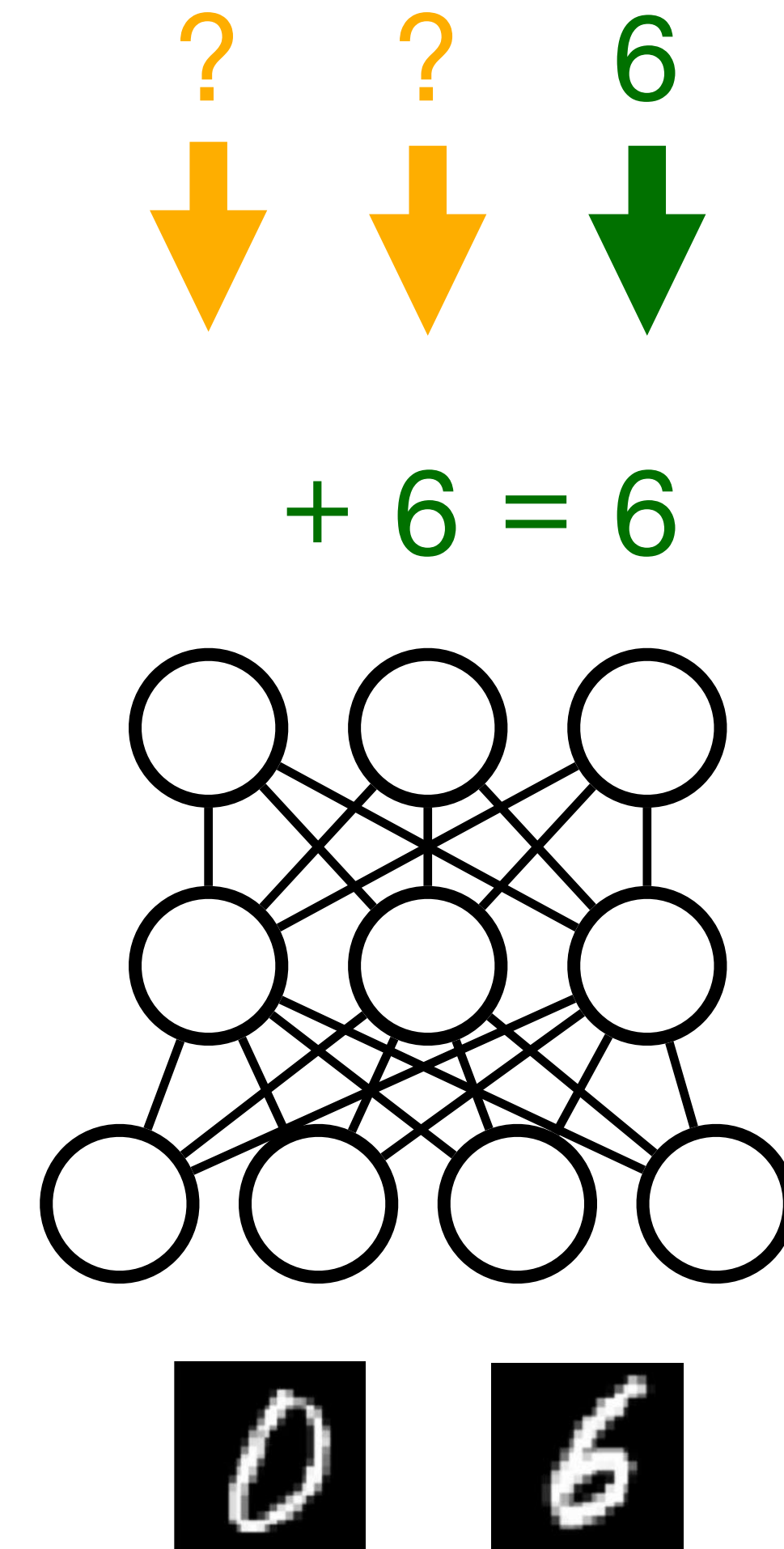




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
(0, 6)	6	(1, 5)	6
(2, 8)	10	(3, 7)	10
(4, 6)	10	(1, 9)	10
(4, 8)	12	(3, 9)	12

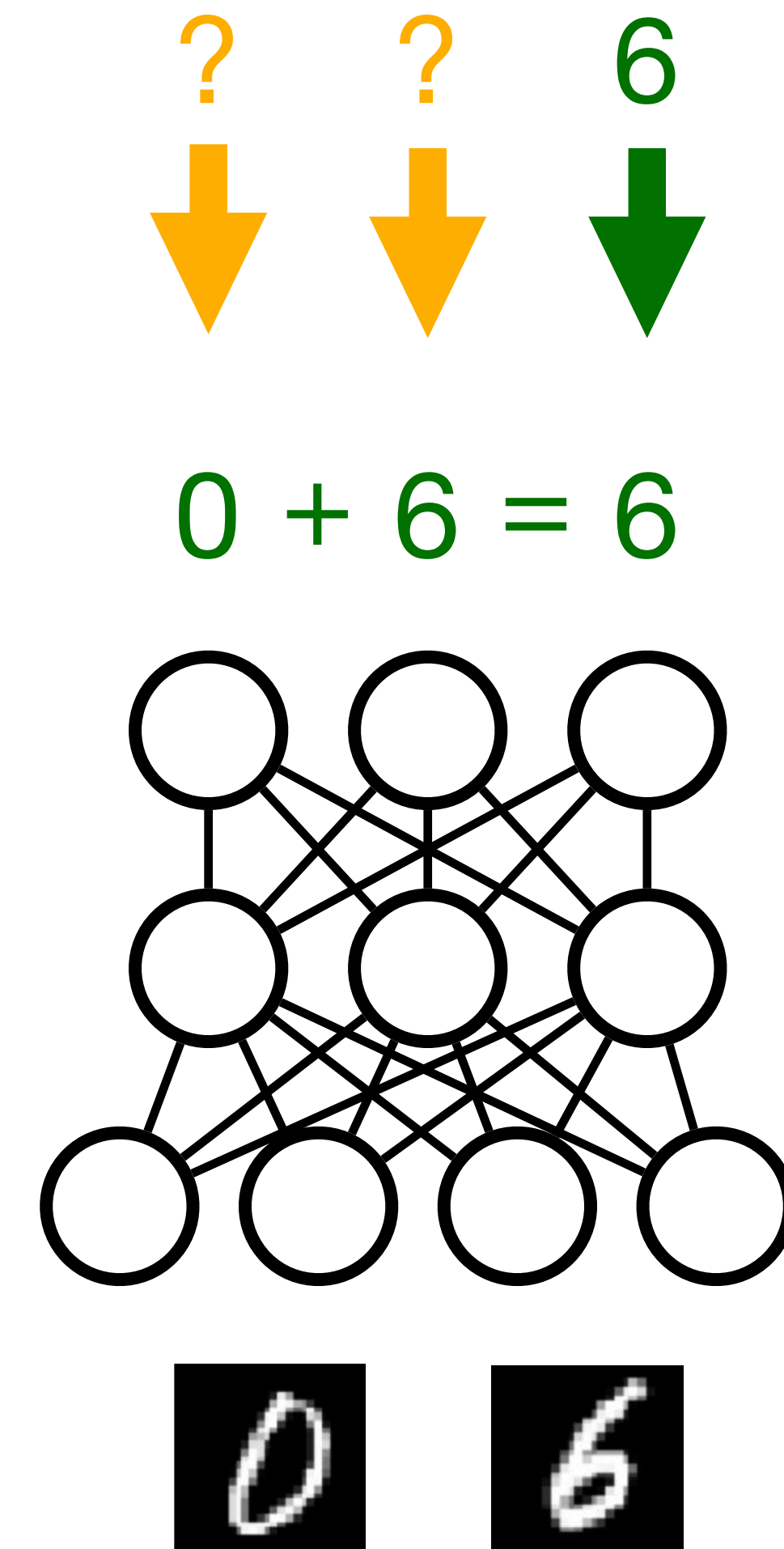




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
( , )	6	( , )	6
( , )	10	( , )	10
( , )	10	( , )	10
( , )	12	( , )	12

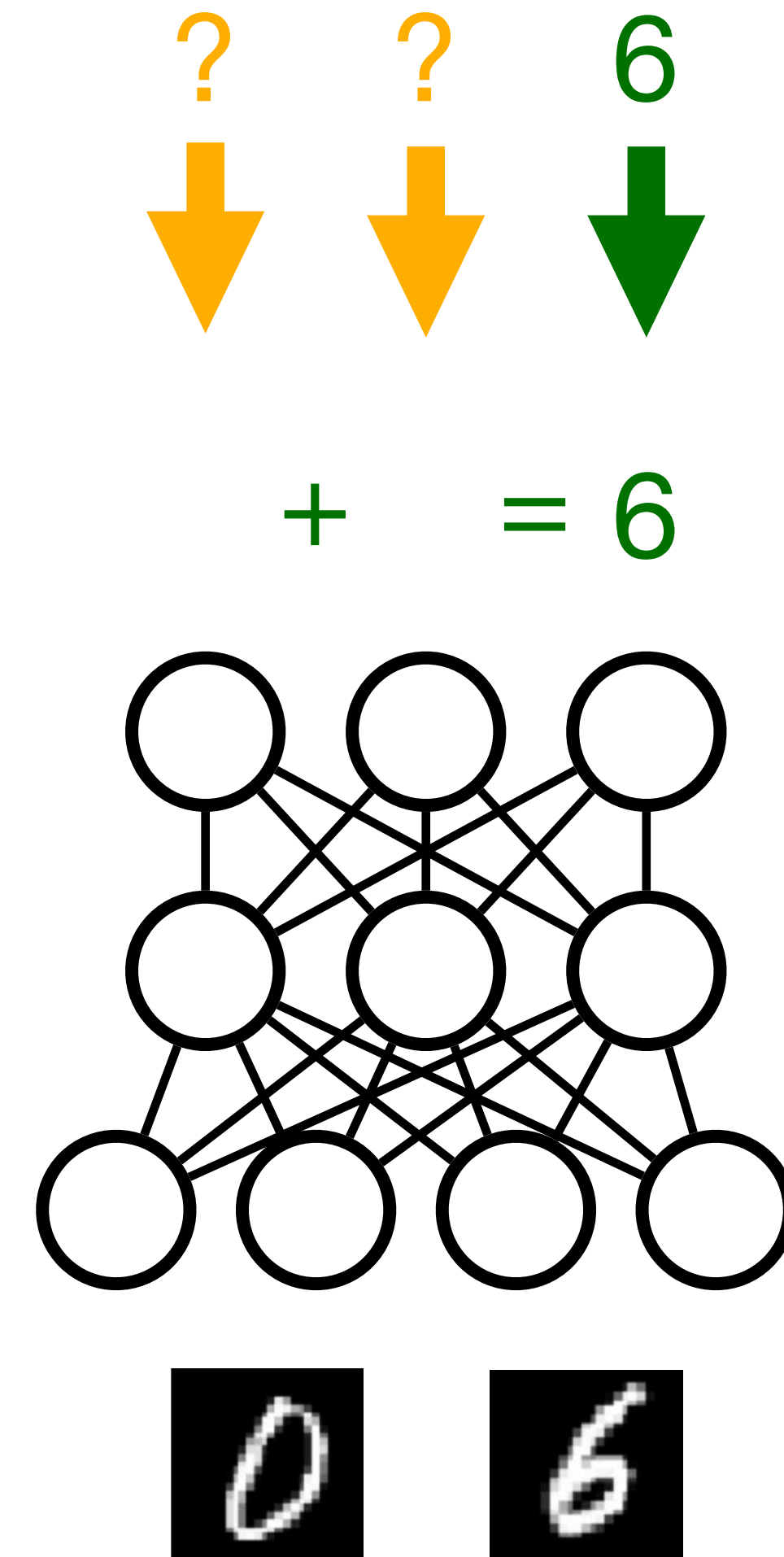




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
(0, 6)	6	(1, 5)	6
(2, 8)	10	(3, 7)	10
(4, 6)	10	(1, 9)	10
(4, 8)	12	(3, 9)	12

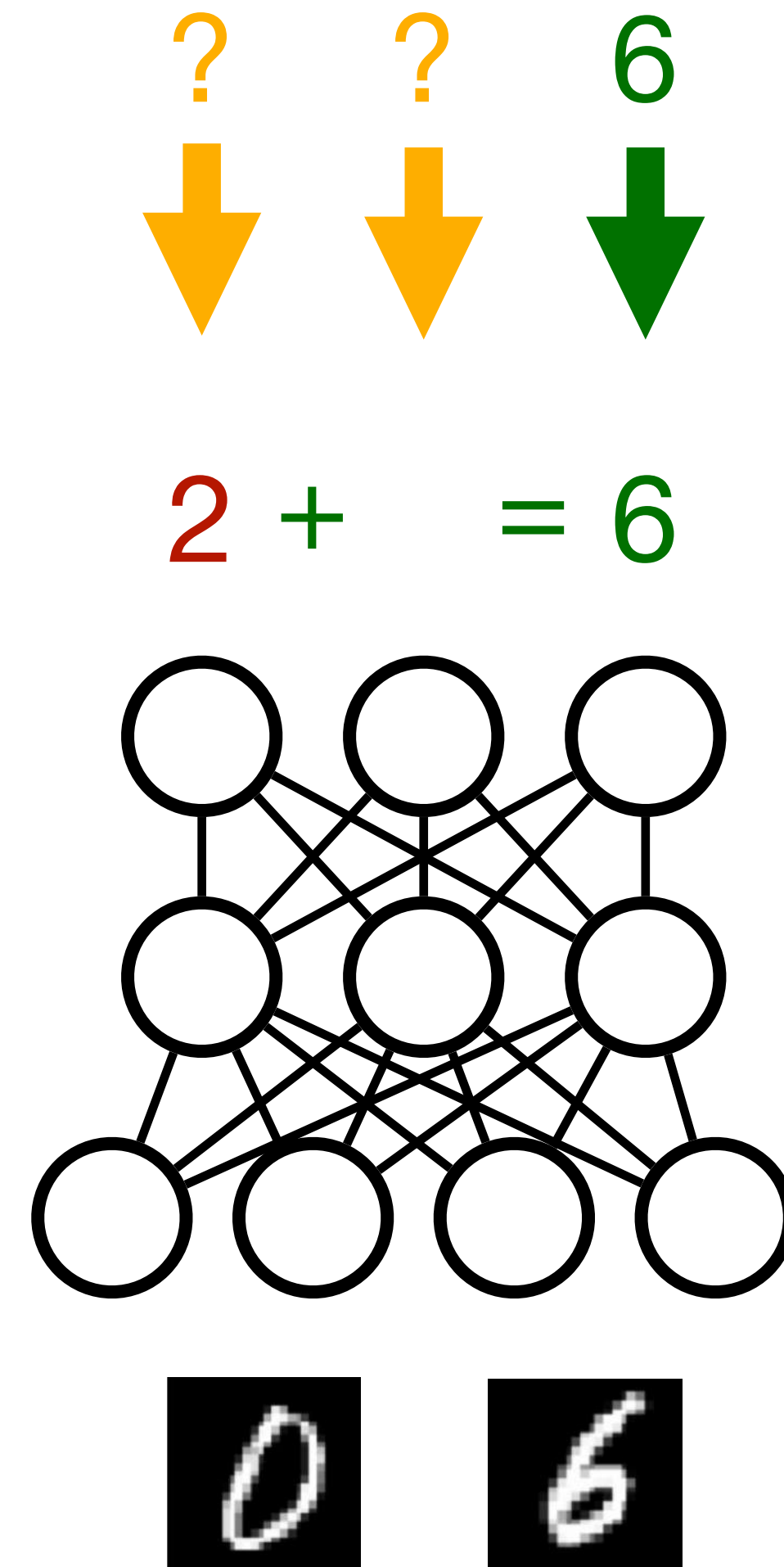




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
(0, 6)	6	(1, 5)	6
(2, 8)	10	(3, 7)	10
(4, 6)	10	(1, 9)	10
(4, 8)	12	(3, 9)	12

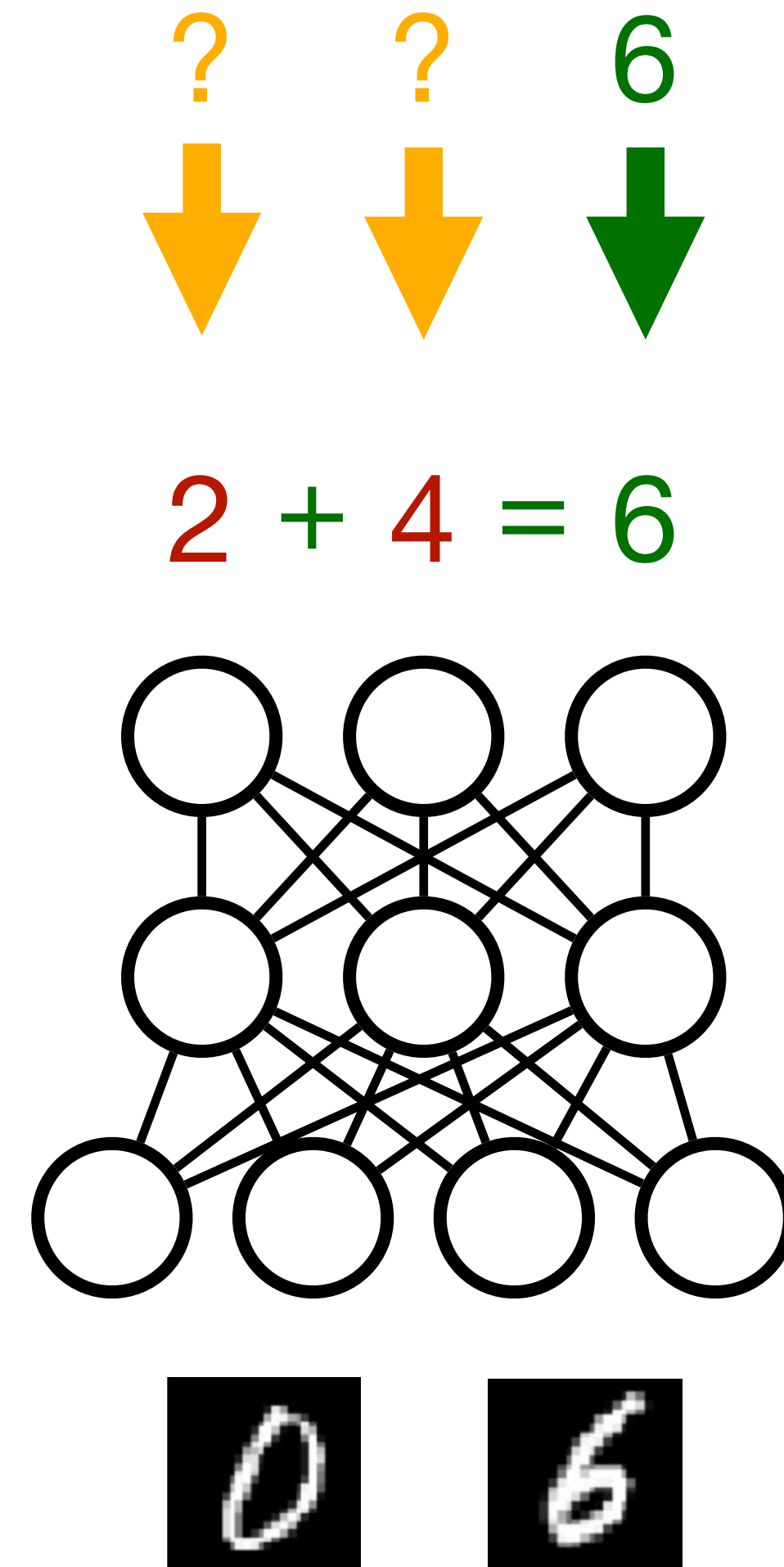




RSBench: a controlled benchmark

Mnist-EvenOdd

Example Pair	Label	Example Pair	Label
( , )	6	( , )	6
( , )	10	( , )	10
( , )	10	( , )	10
( , )	12	( , )	12

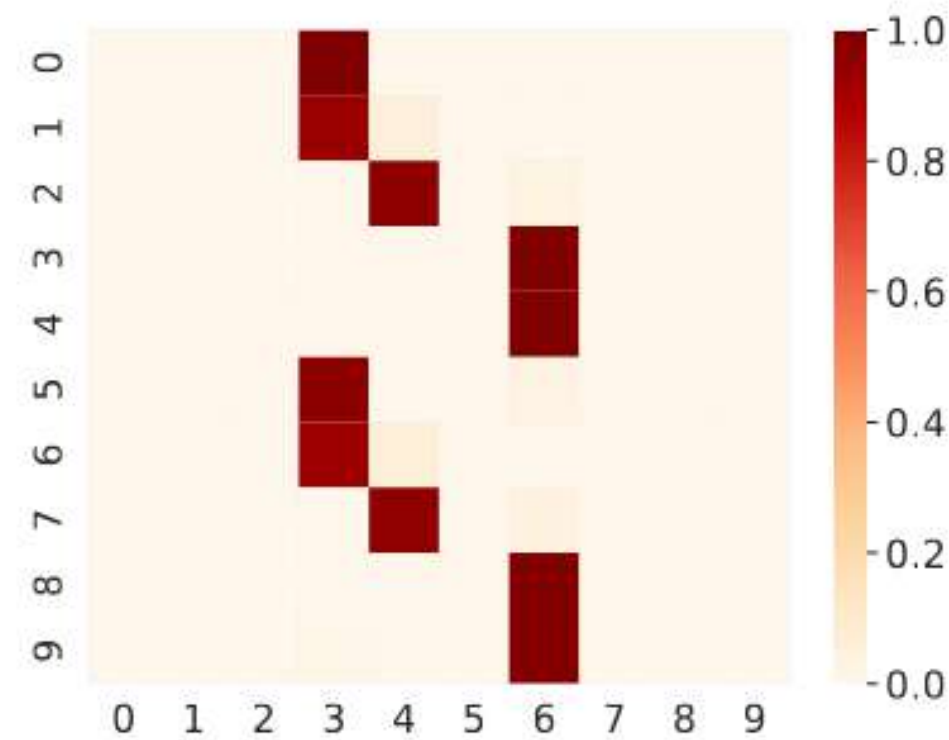




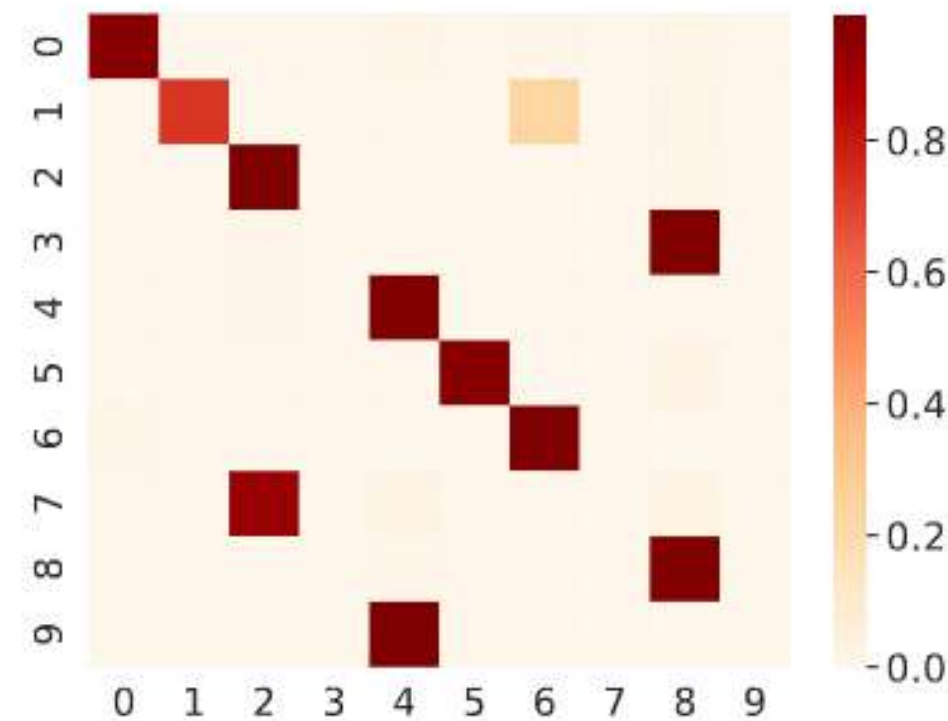
Results on MNISTEvenOdd

No more reasoning shortcuts!

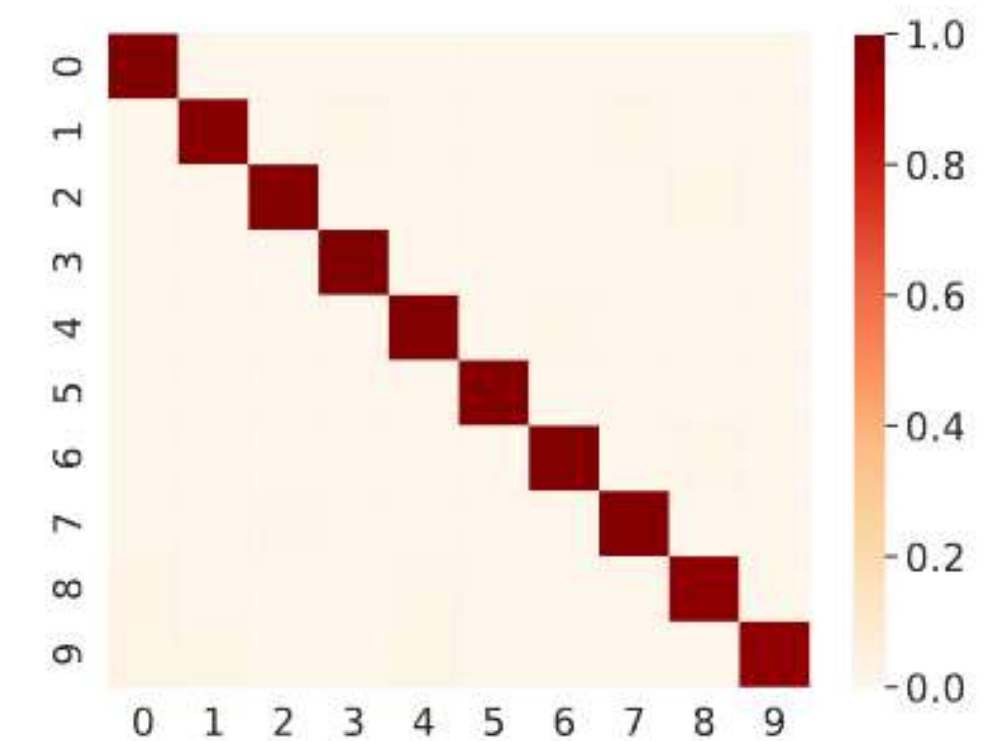
Example Pair	Label	Example Pair	Label
(0, 6)	6	(1, 5)	6
(2, 8)	10	(3, 7)	10
(4, 6)	10	(1, 9)	10
(4, 8)	12	(3, 9)	12



(a) SL



(b) SL+Pre^{*}



(c) SL+PNet



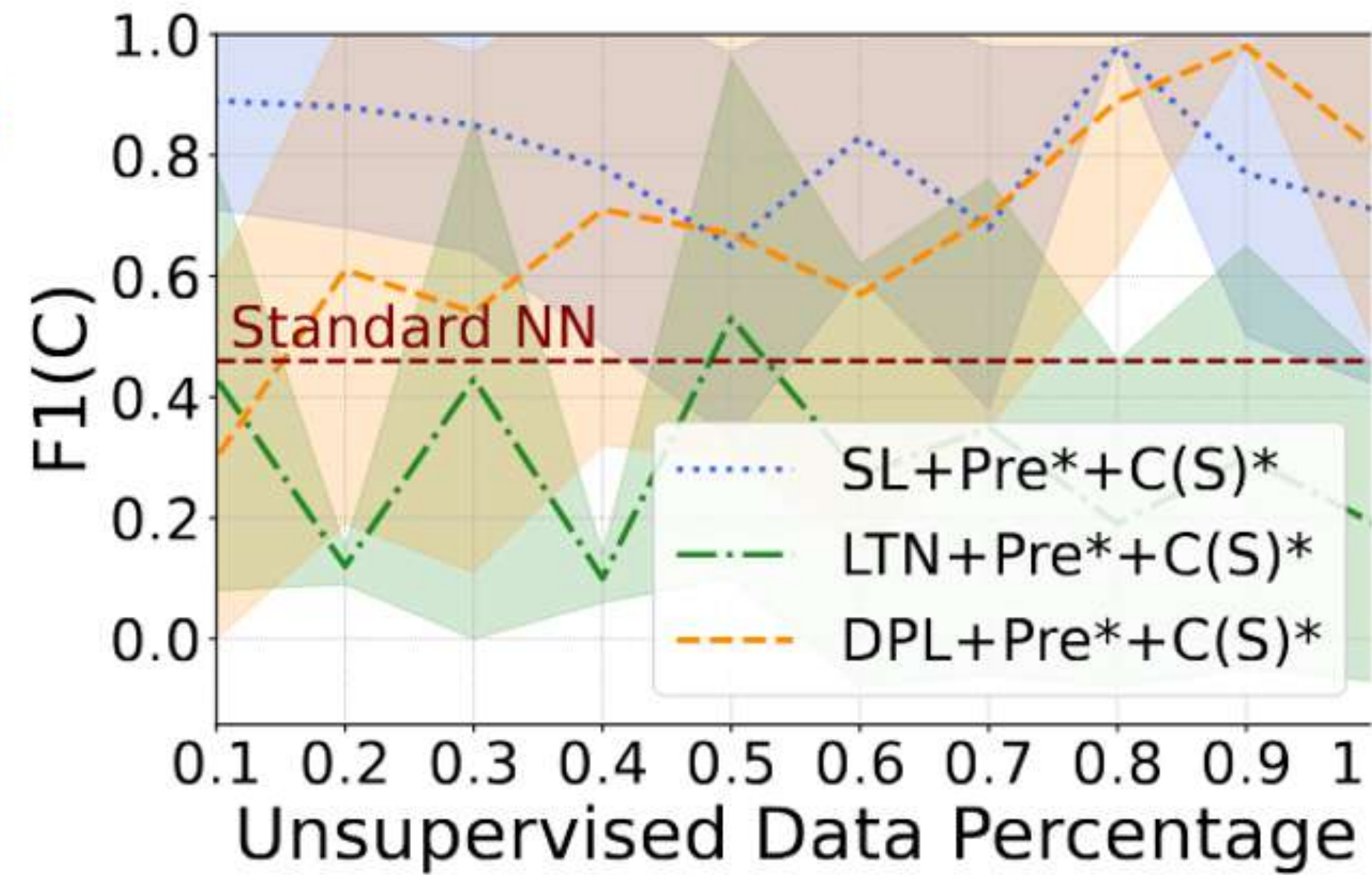
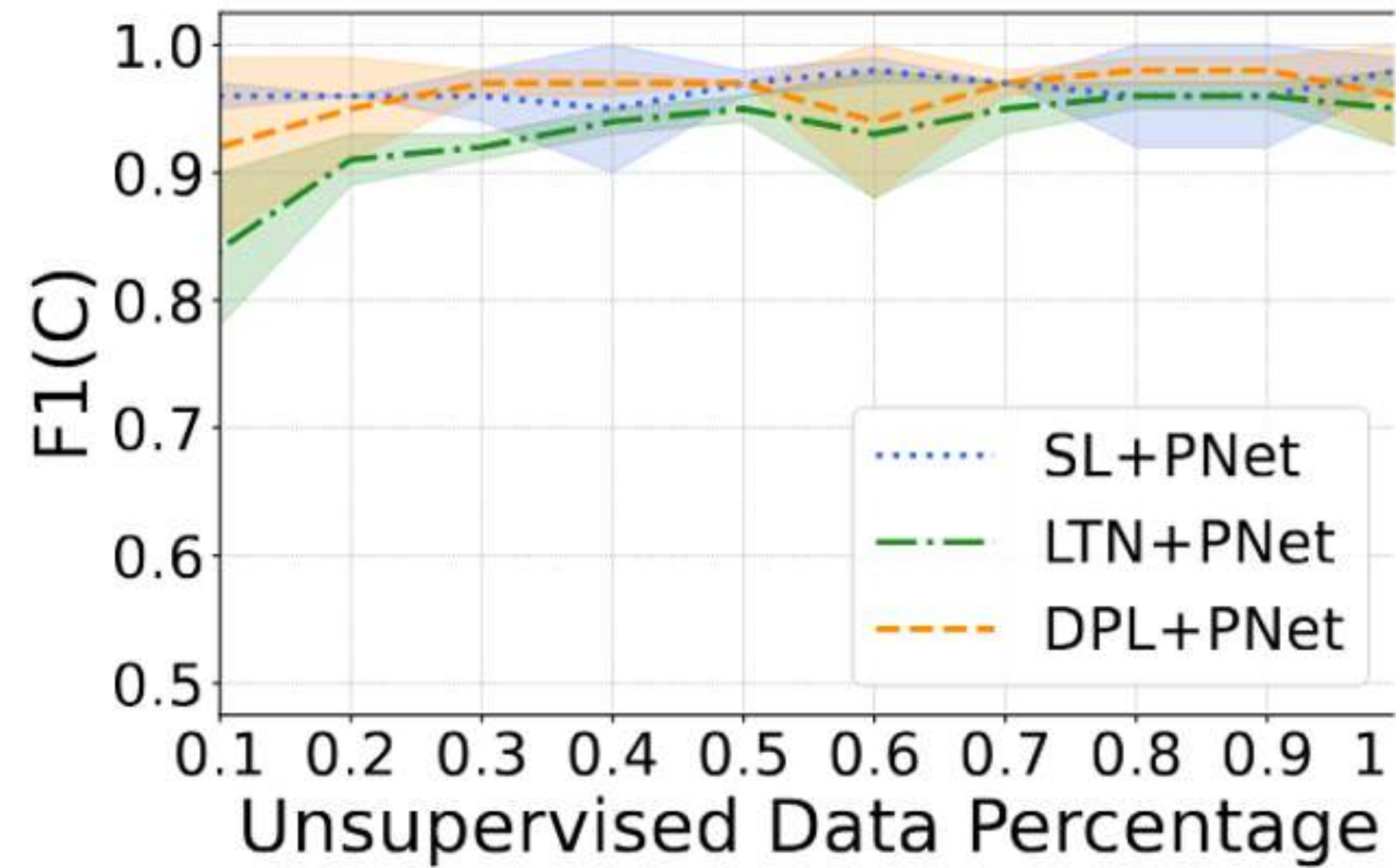
Results on MNISTEvenOdd

No more reasoning shortcuts!

	Acc(C)(↑)	F1(C)(↑)	Cls(C)(↓)
LTN	0.13±0.07	0.05±0.04	0.79±0.08
LTN+Pre [*]	0.51±0.22	0.39±0.25	0.45±0.23
LTN+PNet	0.95 ±0.03	0.95 ±0.03	0.00 ±0.00
SL	0.08±0.12	0.04±0.04	0.68±0.04
SL+Pre [*]	0.52±0.25	0.42±0.28	0.40±0.24
SL+PNet	0.98 ±0.01	0.98 ±0.01	0.00 ±0.00
DPL	0.08±0.10	0.04±0.05	0.67±0.05
DPL+Pre [*]	0.64±0.22	0.55±0.28	0.32±0.24
DPL+PNet	0.97 ±0.03	0.96 ±0.04	0.00 ±0.00
CCN ⁺	0.03±0.07	0.01±0.03	0.73±0.05
CCN ⁺ +Pre [*]	0.33±0.11	0.22±0.10	0.57±0.09
CCN⁺+PNet	0.96 ±0.04	0.95 ±0.05	0.00 ±0.00



And the networks are more robust!





A comprehensive survey on the topic

Symbol Grounding in Neuro-Symbolic AI: A Gentle Introduction to Reasoning Shortcuts

EMANUELE MARCONATO^{*†}, University of Trento, Italy

SAMUELE BORTOLOTTI[†], University of Trento, Italy

EMILE VAN KRIEKEN[†], Vrije Universiteit Amsterdam, The Netherlands

PAOLO MORETTIN, University of Trento, Italy

ELENA UMILI, Sapienza University of Rome, Italy

ANTONIO VERGARI, University of Edinburgh, United Kingdom

EFTHYMIA TSAMOURA, Huawei Labs, United Kingdom

ANDREA PASSERINI, University of Trento, Italy

STEFANO TESO, CIMEC & DISI, University of Trento, Italy

Neuro-symbolic (NeSy) AI aims to develop deep neural networks whose predictions comply with *prior knowledge* encoding, e.g., safety or structural constraints. As such, it represents one of the most promising avenues for reliable and trustworthy AI. The core idea behind NeSy AI is to combine neural and symbolic steps: neural networks are typically responsible for mapping low-level inputs into high-level symbolic concepts, while symbolic reasoning infers predictions compatible with the extracted concepts and the prior knowledge. Despite their promise, it was recently shown that – whenever the concepts are not supervised directly – NeSy models can be affected by **Reasoning Shortcuts (RSs)**. That is, they can achieve high label accuracy by grounding the concepts incorrectly. RSs can compromise the interpretability of the model’s explanations, performance in out-of-distribution scenarios, and therefore reliability. At the same time, RSs are difficult to detect and prevent unless concept supervision is available, which is typically not the case. However, the literature on RSs is scattered, making it difficult for researchers and practitioners to understand and tackle this challenging problem. This overview addresses this issue by providing a gentle introduction to RSs, discussing their causes and consequences in intuitive terms. It also reviews and elucidates existing theoretical characterizations of this phenomenon. Finally, it details methods for dealing with RSs, including mitigation and awareness strategies, and maps their benefits and limitations. By reformulating advanced material in a digestible form, this overview aims to provide a unifying perspective on RSs to lower the bar to entry for tackling them. Ultimately, we hope this overview contributes to the development of reliable NeSy and trustworthy AI models.

JAIR Track: Integration of Logical Constraints in Deep Learning



Conclusions

- Today we have learnt how to integrate constraints into neural networks
- In general, if we want to use the background knowledge to train on unsupervised datapoints we should incorporate the background knowledge in the loss function, while in order to get the guarantees we should use a layer
- A fuzzy vs probabilistic interpretation of the requirements leads to different advantages and disadvantages

Additional Papers

Background knowledge integration:

- DL2: Training and Querying Neural Networks with Logic
- Semantic-based regularization for learning and inference
- Integrating Prior Knowledge into Deep Learning
- Deep Learning with Logical Constraints (survey)
- MultiplexNet: Towards fully satisfied logical constraints in neural networks

Constraints over continuous outputs:

- How Realistic Is Your Synthetic Data? Constraining Deep Generative Models for Tabular Data
- Beyond the convexity assumption: Realistic tabular data generation under quantifier-free real linear constraints
- A Probabilistic Neuro-symbolic Layer for Algebraic Constraint Satisfaction
- DC3: A learning method for optimization with hard constraints

Learning and reasoning:

- SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver
- DeepProbLog: Neural Probabilistic Logic Programming
- NeurASP: Embracing Neural Networks into Answer Set Programming
- NeuPSL: Neural Probabilistic Soft Logic

Reasoning shortcuts:

- BEARS Make Neuro-Symbolic Models Aware of their Reasoning Shortcuts
- Shortcuts and identifiability in concept-based models from a neuro-symbolic lens

Constraining Large Language Models:

- NeuroLogic Decoding: (Un)supervised Neural Text Generation with Predicate Logic Constraints
- Controllable Text Generation with Neurally-Decomposed Oracle
- Tractable Control for Autoregressive Language Generation
- Stay Within Your Bounds: Distance-Guided Decoding for Guaranteed Context-Free Grammar Compliance

Any questions?

