Semantic objective functions

Unifying some types of loss functions

Vaishak Belle, University of Edinburgh

Agenda

- MultiplexNet - Towards Fully Satisfied Logical Constraints in Neural Networks.
- Some ideas for unifying strategies

Hoernle et al 2021; Mendez-Lucero et al. 2023

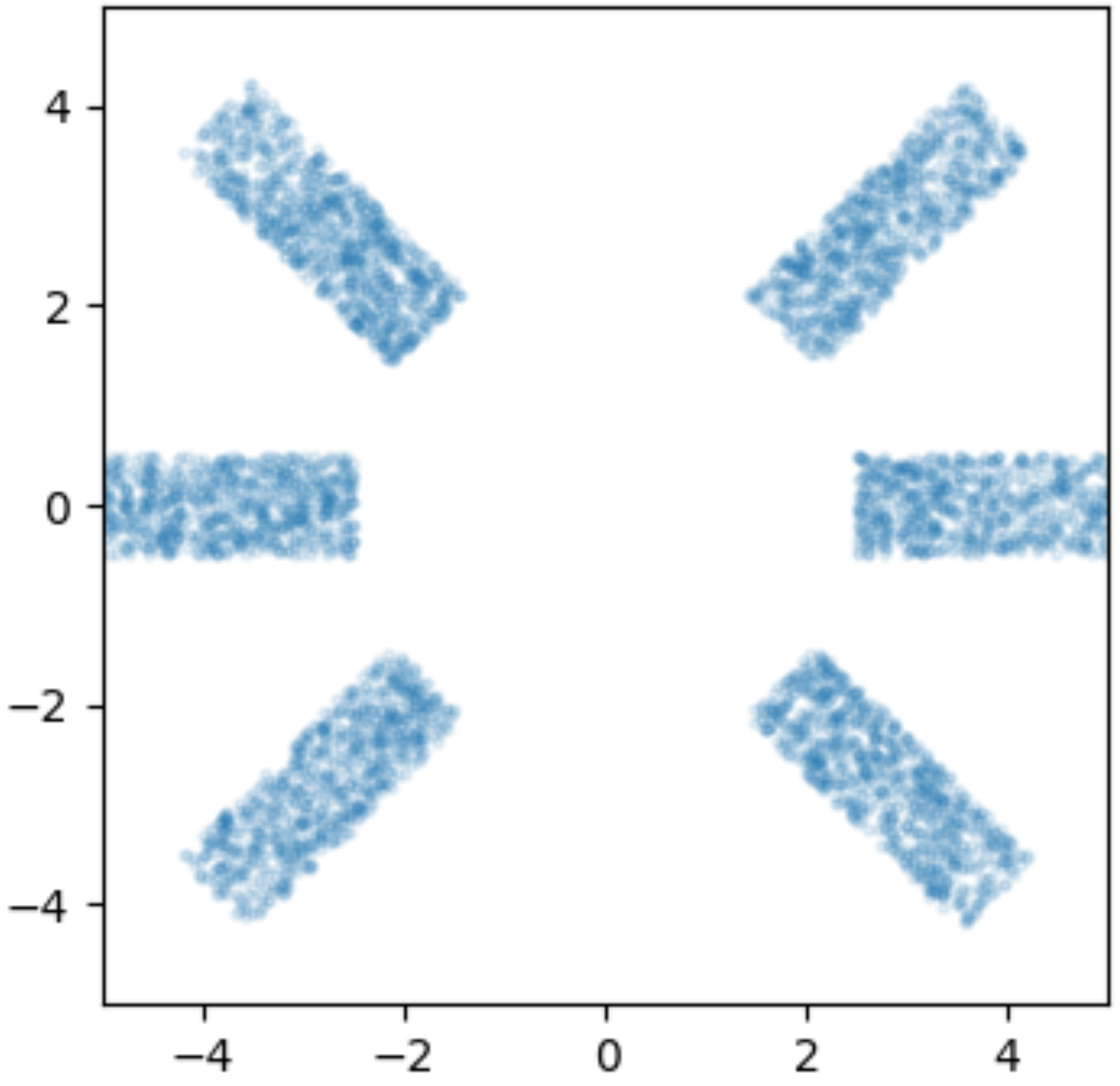
Overview

- Incorporation of **expert knowledge** into the **training of deep neural networks**.
- Domain knowledge represented as a **quantifier-free logical formula** in **disjunctive normal form (DNF)**.
- **Latent Categorical variable** that learns to choose which constraint term optimizes the error function.
- Approach guarantees **100% constraint satisfaction** in a network's output.

Results:

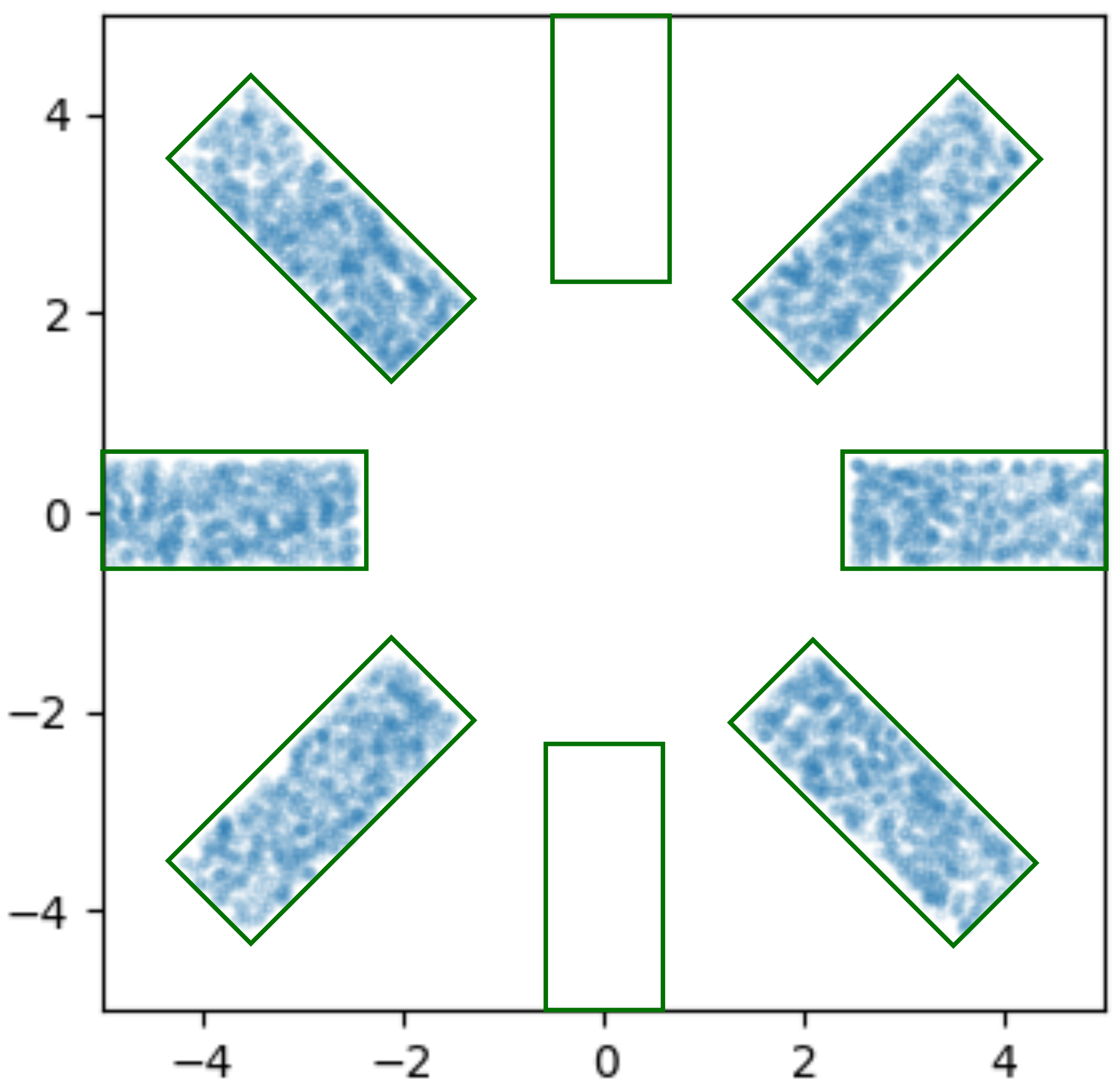
- Approximates unknown distributions well, requiring **fewer data samples** than the alternative approaches.
- Shown to be both **efficient and** general.

Motivating Example - Density Estimation Task



Generated Data

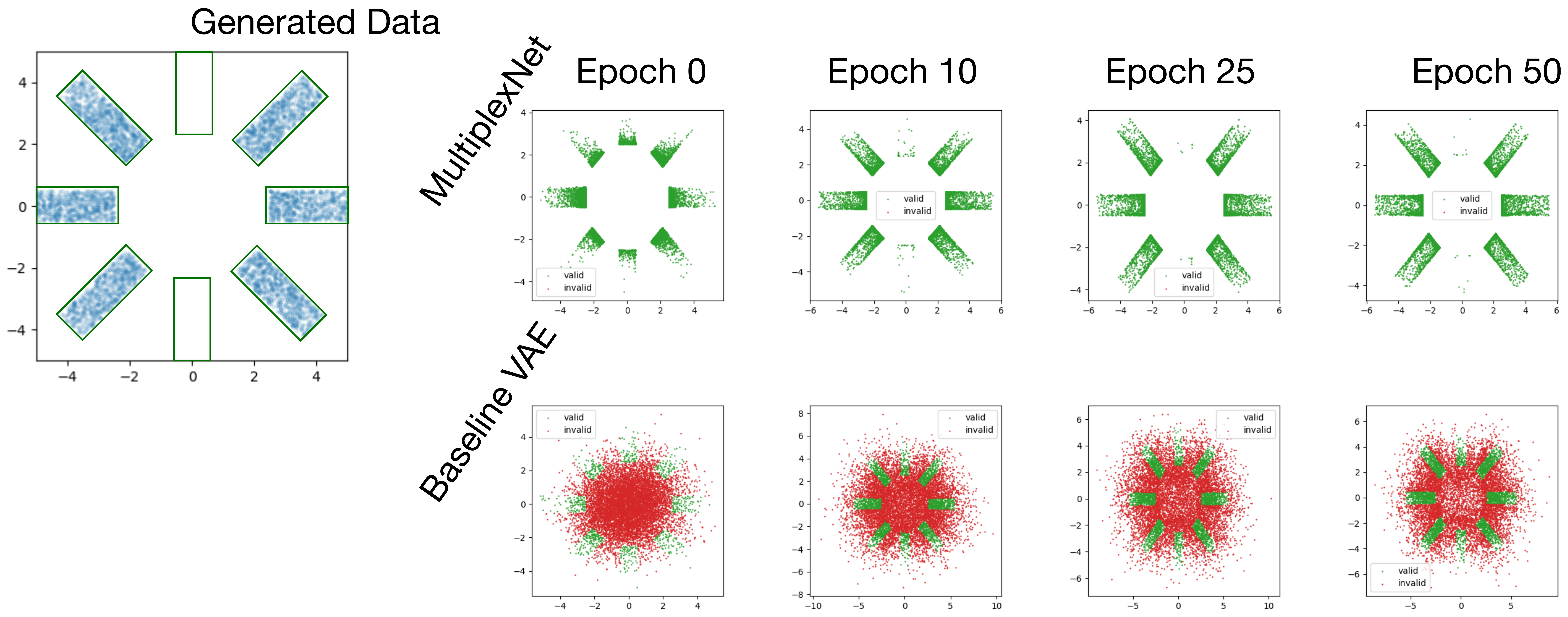
Motivating Example - How to use Constraints in Training?



Generated Data

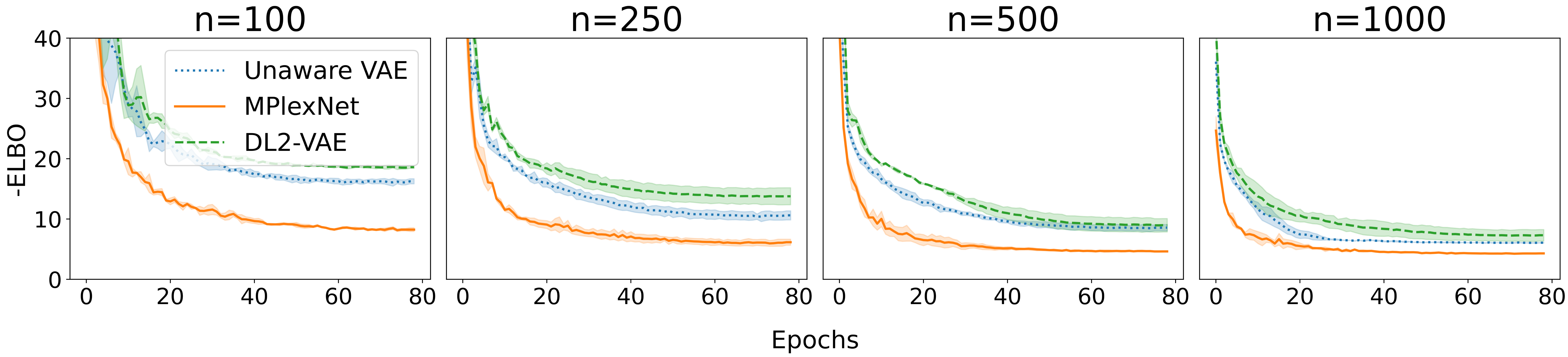
$$\Phi = (x_1 > -.5 \wedge x_1 < .5 \wedge x_2 > .5 \wedge x_2 < 4) \vee \ldots$$
$$\ldots \vee (x_1 + x_2 > -.5 \wedge x_1 + x_2 < .5 \wedge x_1 - x_2 > .5 \wedge x_1 - x_2 < 4)$$

Motivating Example - Force Constraint Satisfaction

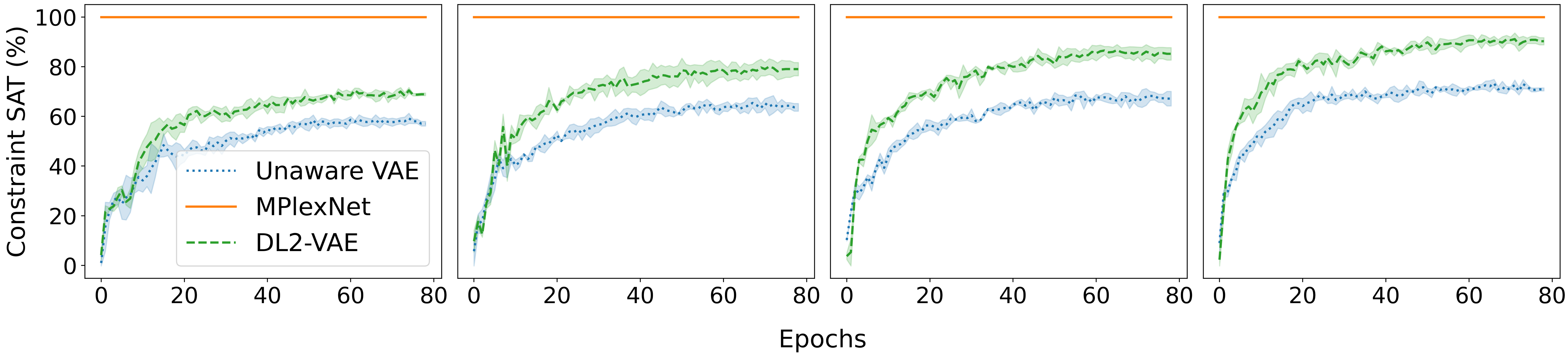


Motivating Example - Desiderata

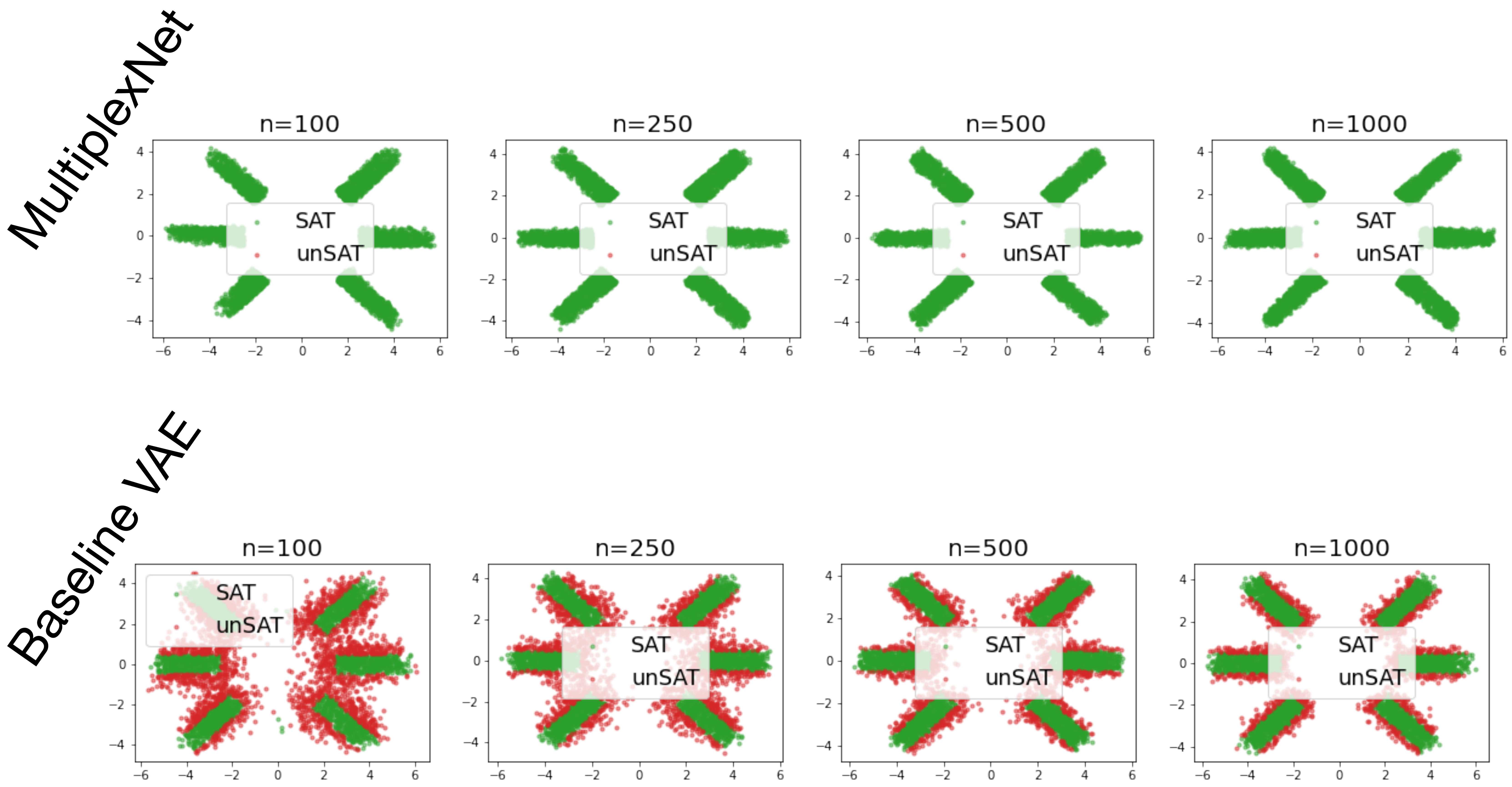
(1) Data Efficiency



(2) Predictability (safety critical systems)



Motivating Example - Posterior Samples

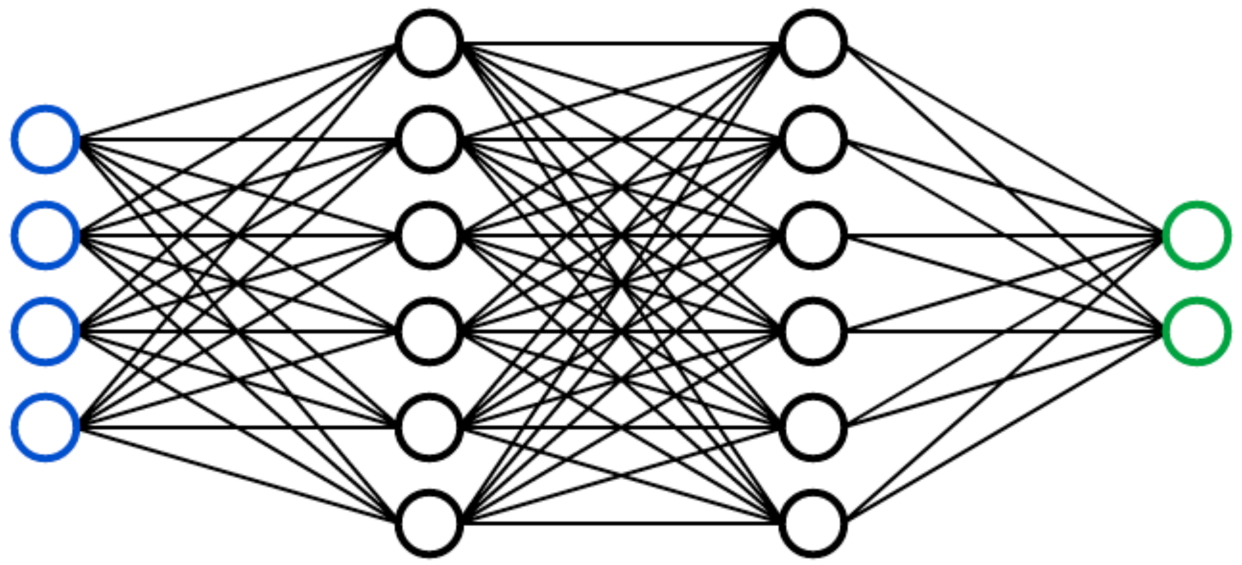


Constraining Probabilistic Models

(1) Given a dataset from unknown density p* but known to entail Φ:

$$X = \{x^{(0)}, \ldots, x^{(N)} \mid x^{(i)} \sim^{iid} p^*(x), x^{(i)} \models \Phi\}$$

Constraining Probabilistic Models - Standard Training



$p_\theta(x)$

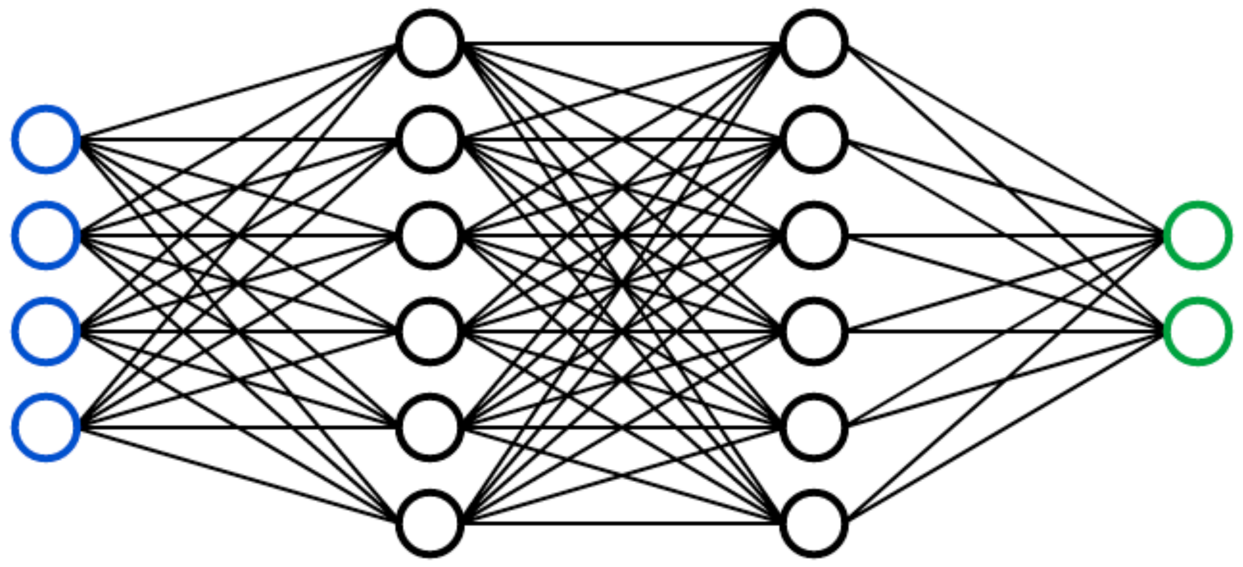
(1) Given a dataset from unknown density p* but known to entail Φ:

$$X = \{x^{(0)}, \ldots, x^{(N)} \mid x^{(i)} \sim^{iid} p^*(x), x^{(i)} \models \Phi\}$$

(2) Train a parameterised model to maximise the likelihood of the data:

Design: $p_\theta(x)$

Train: $p_{\theta*}(x) = \arg\max_\theta(\log p_\theta(X))$



$p_\theta(x)$

(1) Given a dataset from unknown density p* but known to entail Φ:

$$X = \{x^{(0)}, \ldots, x^{(N)} \mid x^{(i)} \sim^{iid} p^*(x), x^{(i)} \models \Phi\}$$

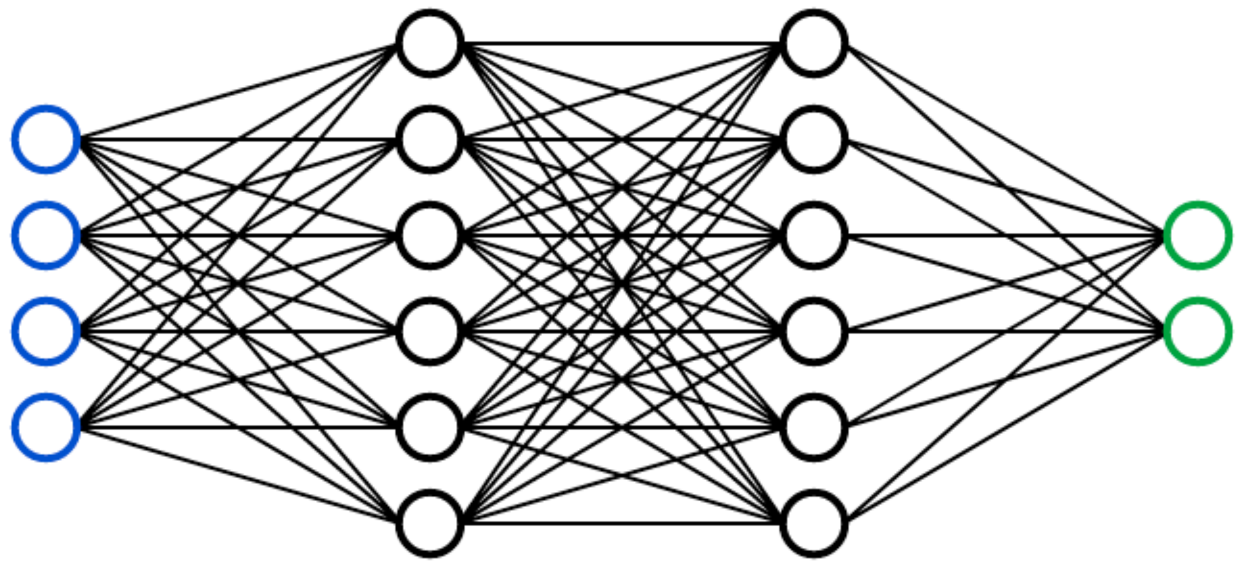
(2) Train a parameterised model to maximise the likelihood of the data:

Design: $p_\theta(x)$

Train: $p_{\theta*}(x) = \arg\max_\theta(\log p_\theta(X))$

But what about Φ?

Constraining Probabilistic Models - Solutions to Include Φ



$p_\theta(x)$

(1) Append a loss term to training:

Train: $p_{\theta^*}(x) = \arg\max_\theta [\log p_\theta(X) + L_\Phi(X)]$

(2) Reparameterise output of network:

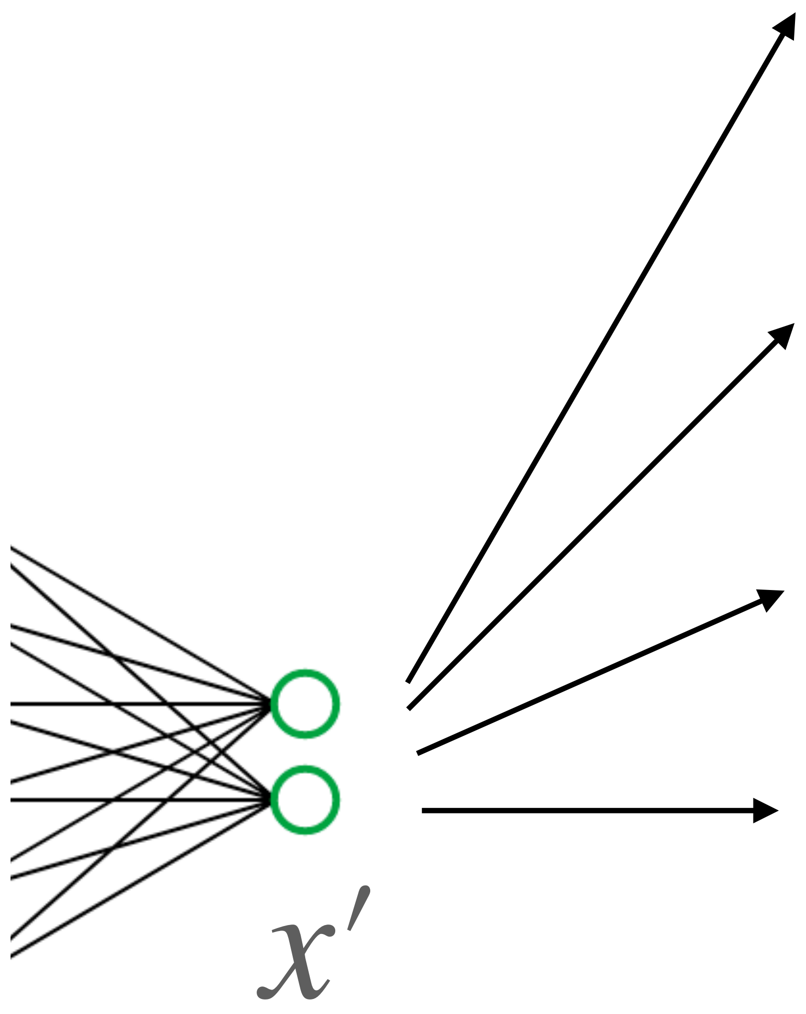
Design: $p_\theta(x)$ *such that the output of the network follows Φ by construction.*

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C. and Vechev, M., 2019, May. DI2: Training and querying neural networks with logic. ICML

Xu, J., Zhang, Z., Friedman, T., Liang, Y. and Broeck, G., 2018, July. A semantic loss function for deep learning with symbolic knowledge. ICML

Innes, C. and Ramamoorthy, S., 2020. Elaborating on learned demonstrations with temporal logic specifications.

Network Output Non-Linearities - Standard Transformations can Restrict Output



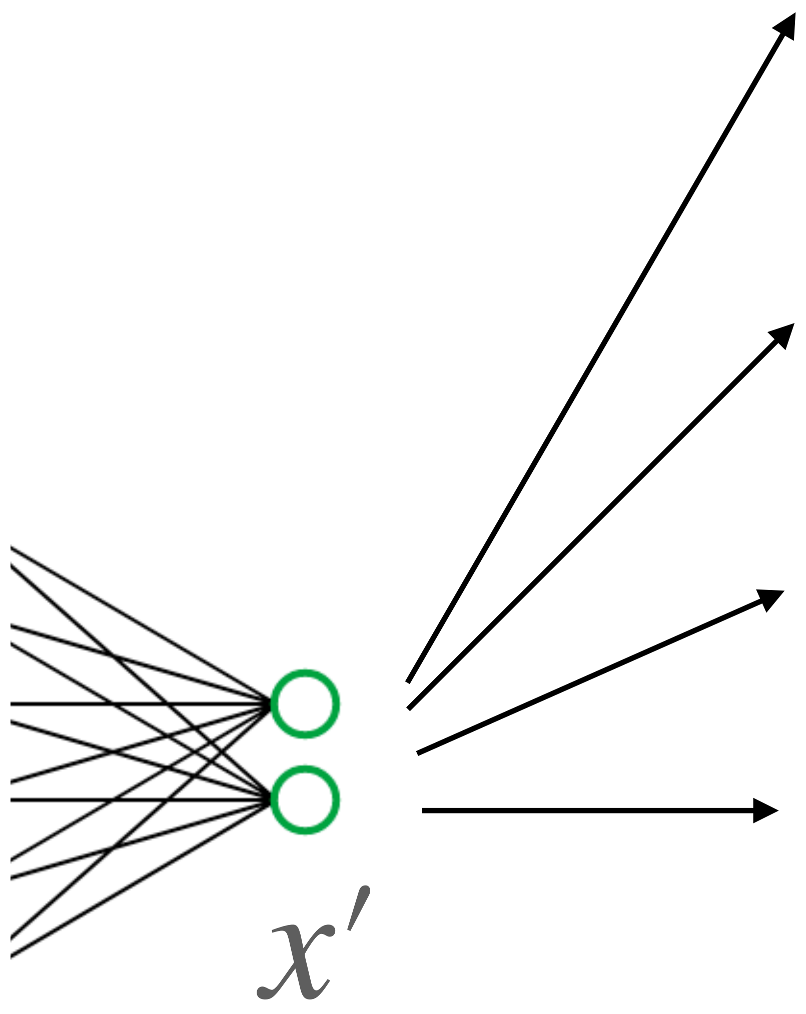
Identity. E.g. a regression network trained on MSE

Softplus. Constrains output to be element wise positive.

Sigmoid. Output is $\in (0,1)$.

ReLU. Constrains output to be element wise ≥ 0.

Network Output Non-Linearities



Identity. E.g. a regression network trained on MSE

Softplus. Constrain output to be element wise positive.

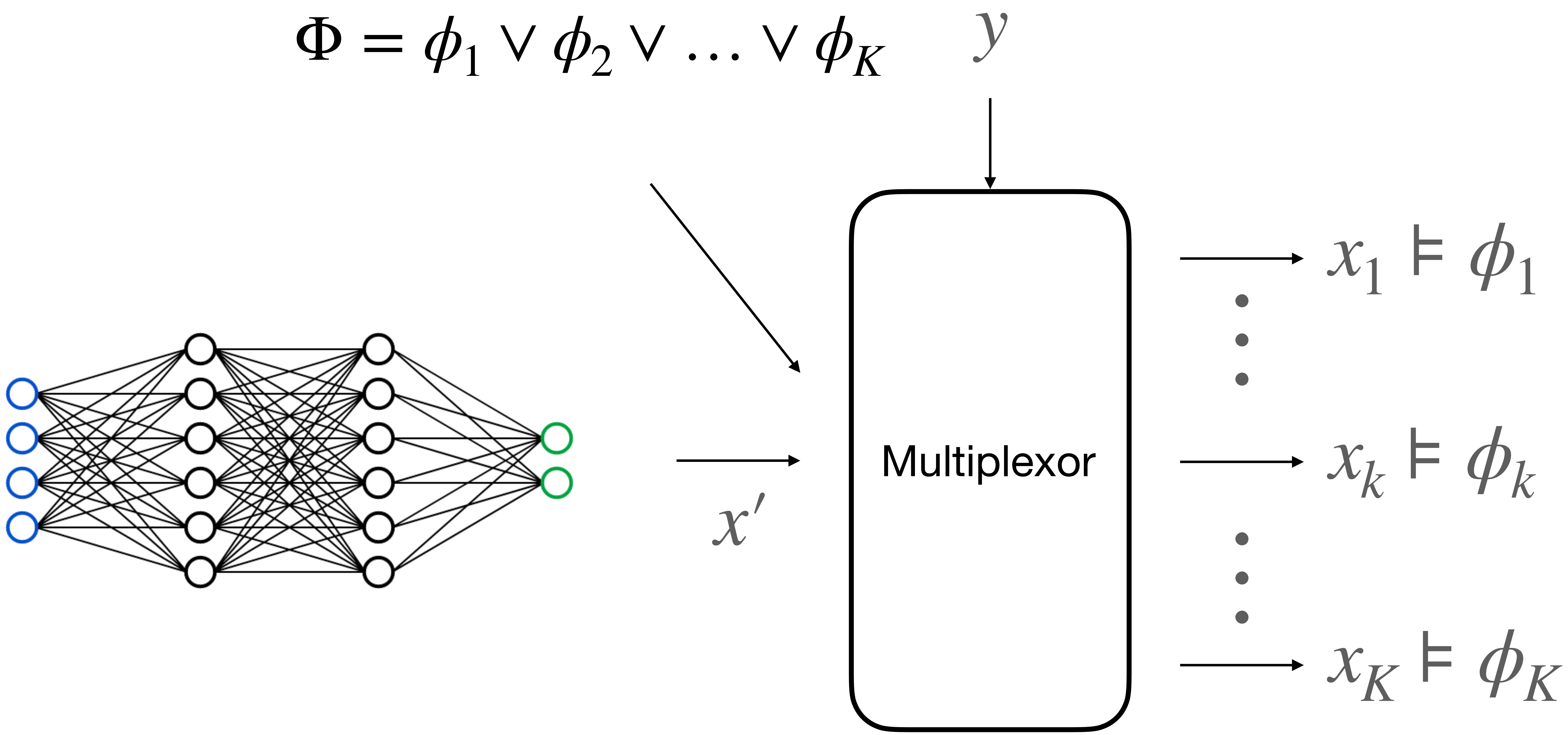
Sigmoid. Output is $\in (0,1)$.

ReLU. Constrain output to be element wise ≥ 0.

$$\Phi = \phi_1 \vee \phi_2 \vee \ldots \vee \phi_K$$

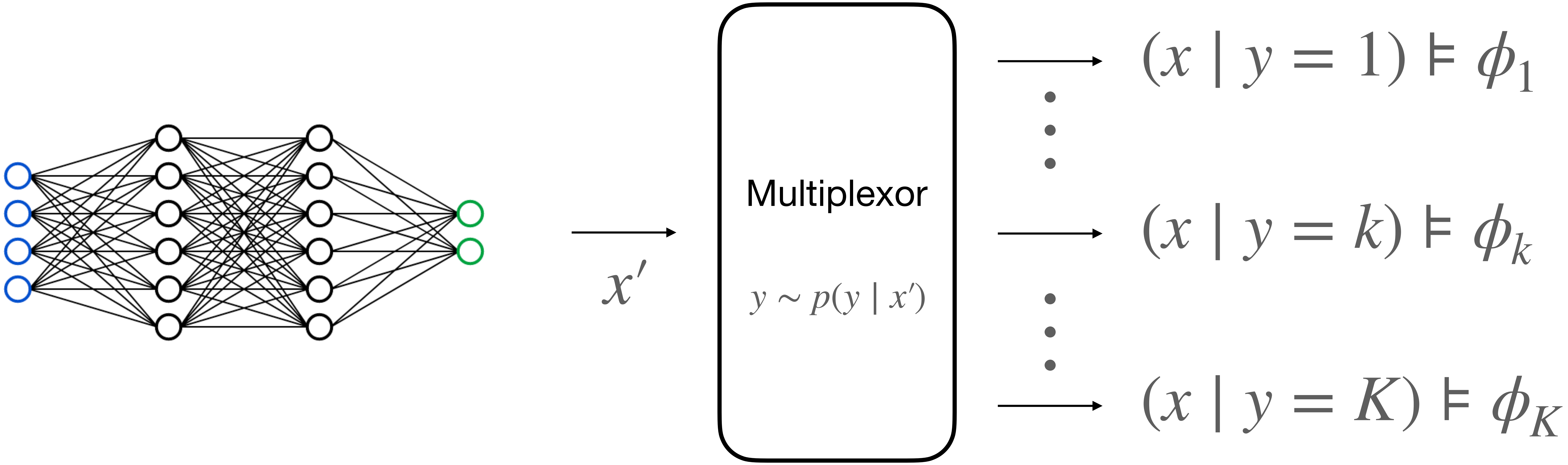
Idea: If Φ is given in DNF, each term ϕ_k in Φ can be suitably represented by a combination of affine transformations and the operators above.

MultiplexNet Architecture



MultiplexNet Architecture

$$\Phi = \phi_1 \vee \phi_2 \vee \ldots \vee \phi_K$$

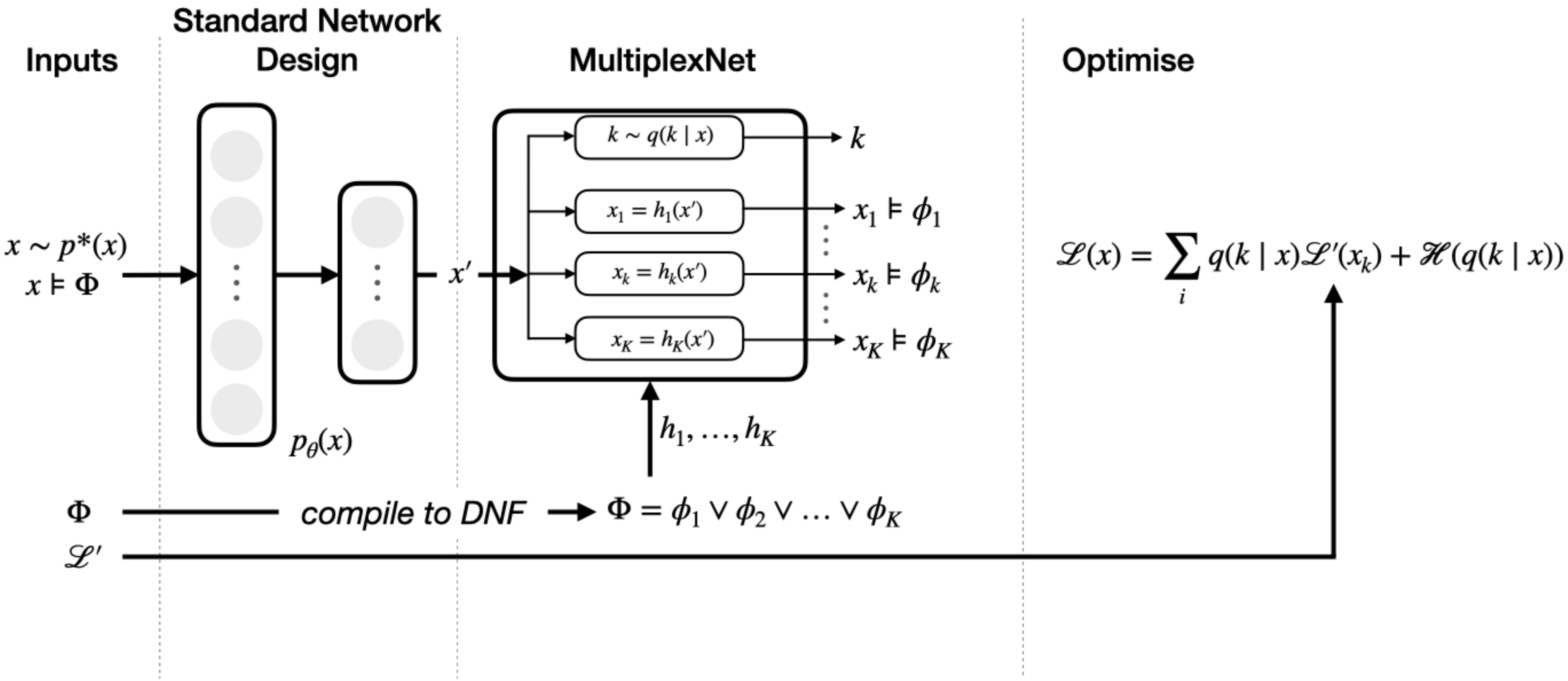


Kingma, D.P., Rezende, D.J., Mohamed, S. and Welling, M., 2014. Semi-supervised learning with deep generative models.

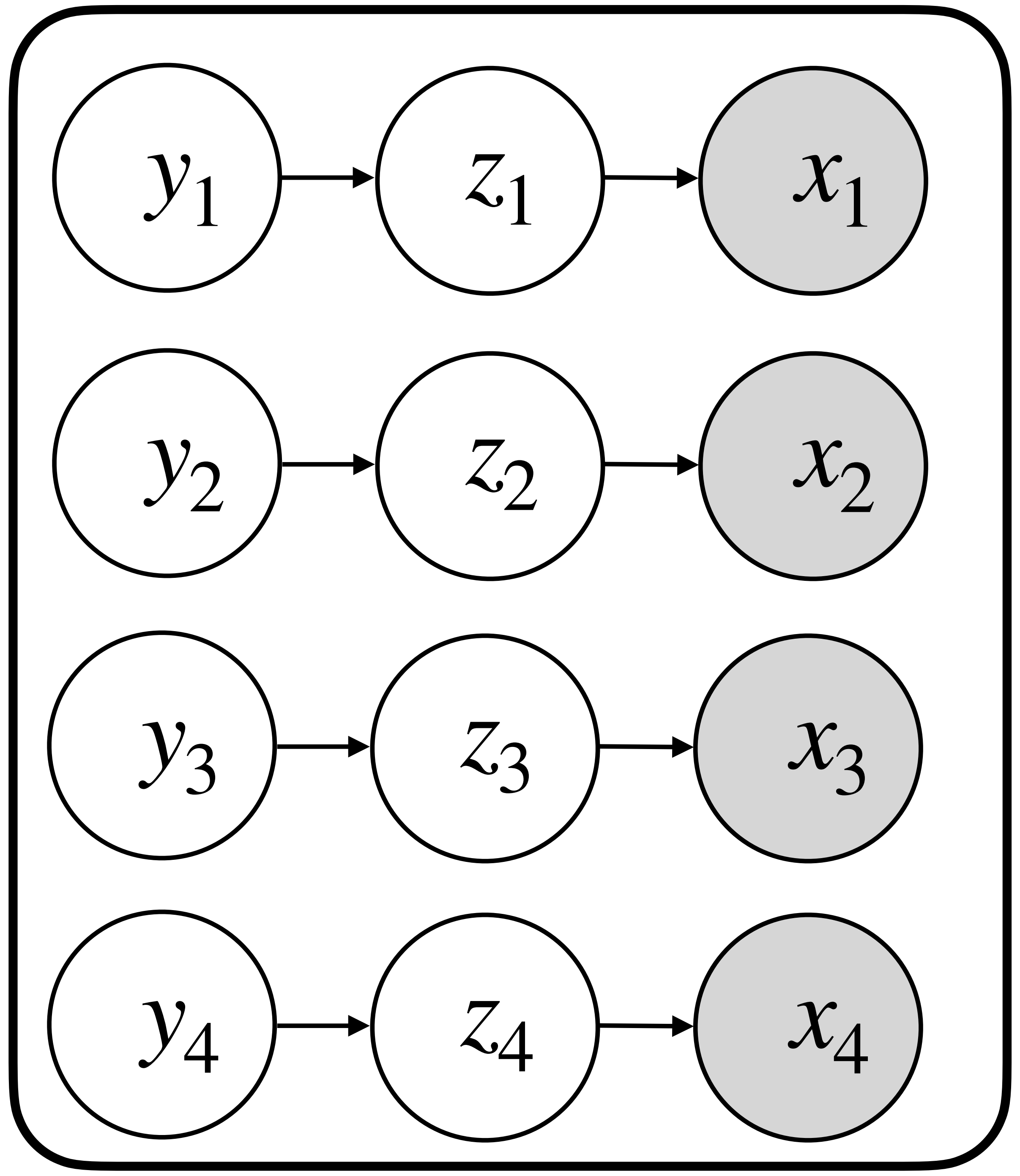
Jang, E., Gu, S. and Poole, B., 2016. Categorical reparameterization with gumbel-softmax.

Maddison, C.J., Mnih, A. and Teh, Y.W., 2016. The concrete distribution: A continuous relaxation of discrete random variables.

Architecture Overview



Example MNIST Label Free Self-Supervision



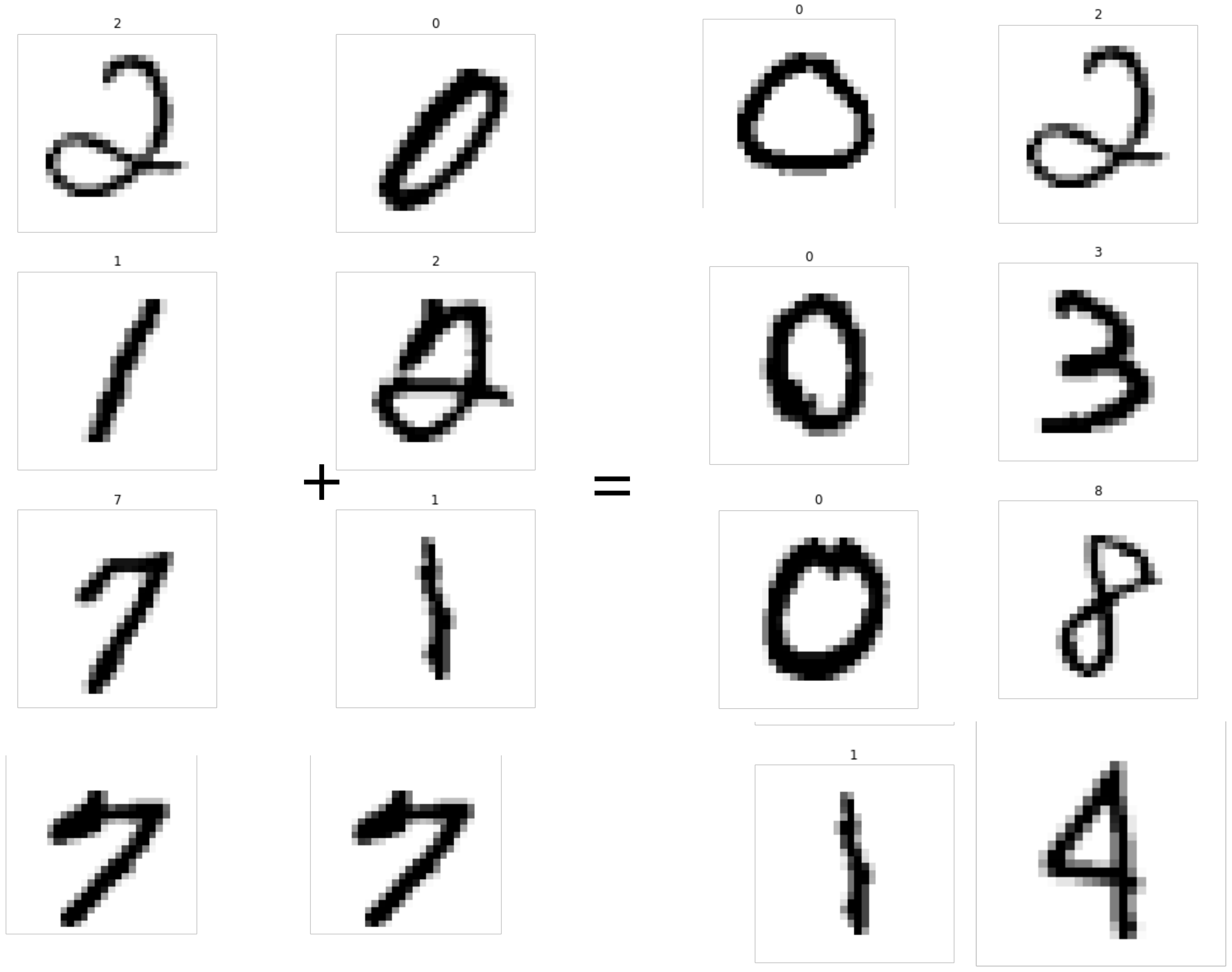
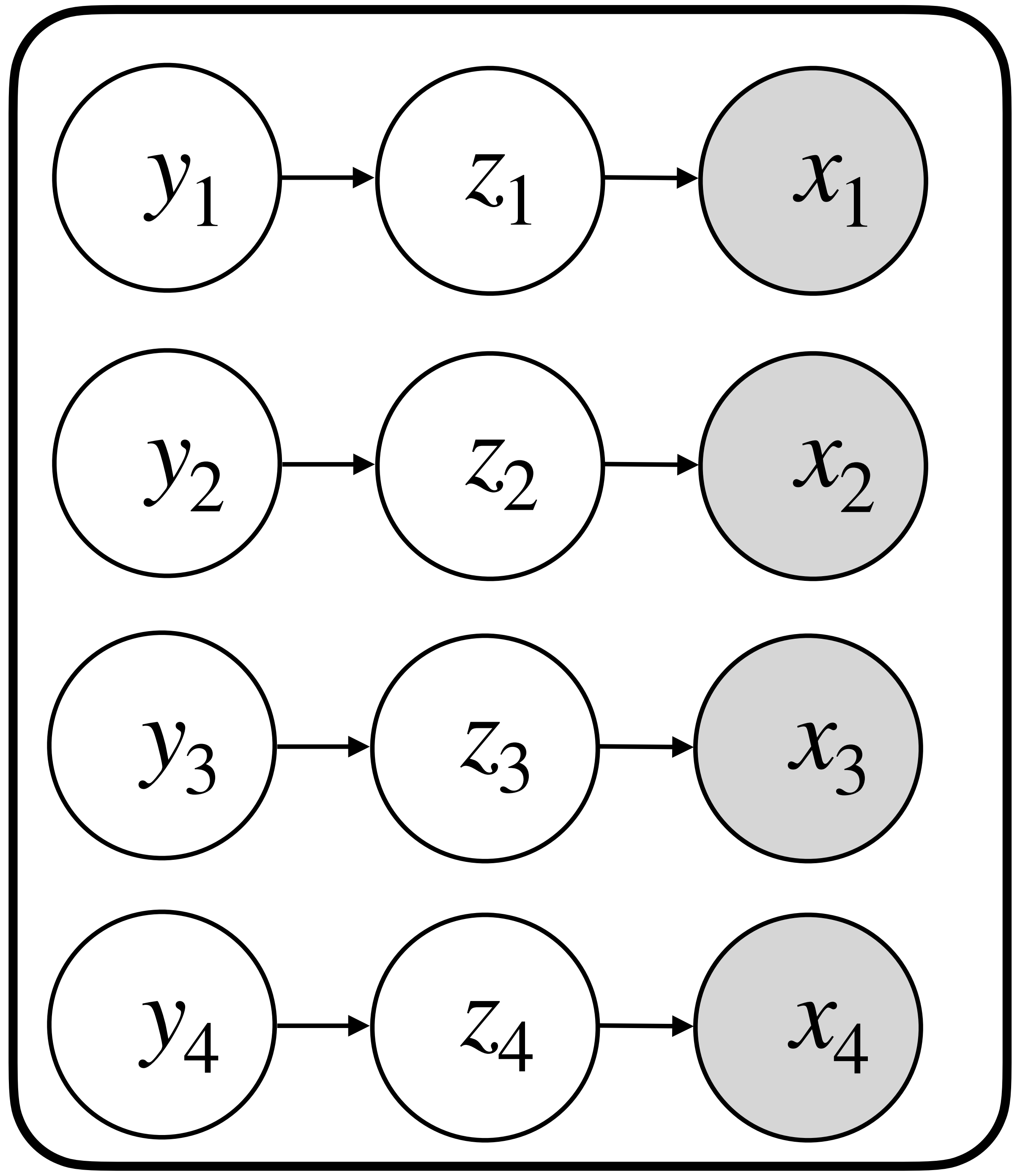


Image 1 Image 2 Image 3 Image 4

Example MNIST Label Free Self-Supervision

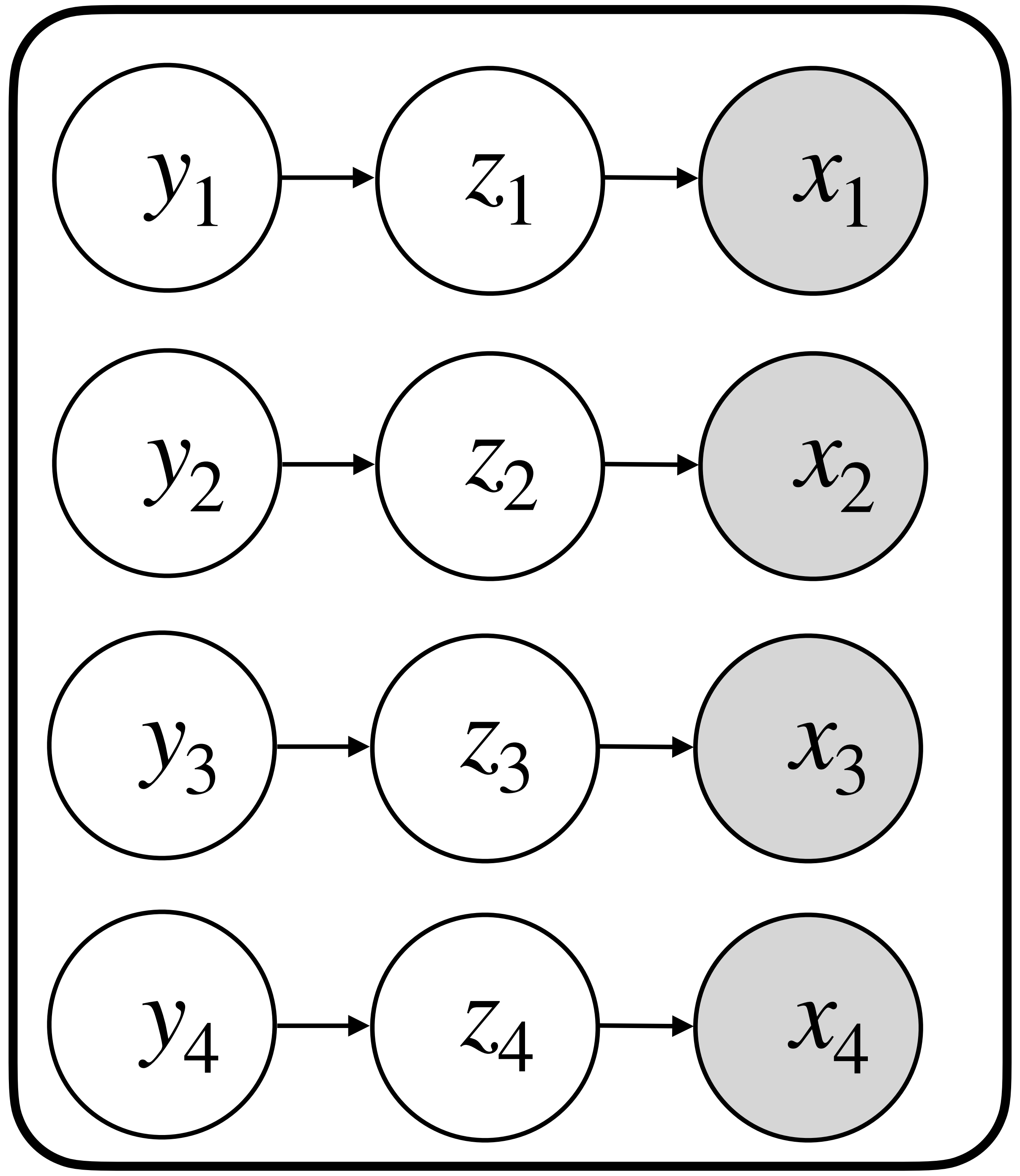


$$\Phi = (y_1 = 0 \wedge y_2 = 0 \wedge y_3 = 0 \wedge y_4 = 0)$$

$$\vee (y_1 = 0 \wedge y_2 = 1 \wedge y_3 = 0 \wedge y_4 = 1) \vee \ldots$$

$$\ldots \vee (y_1 = 9 \wedge y_2 = 9 \wedge y_3 = 1 \wedge y_4 = 8)$$

Example MNIST Label Free Self-Supervision



Conclusions: Part 1

- Incorporation of **logical knowledge** (as QFDNF) into the **training of deep neural networks**.
- Approach guarantees **100% constraint satisfaction** in a network's output.
- Shown to be both **efficient and** general.

Lineage

Semantic loss

$$L(\alpha, p) \propto -log \sum_{M \models \alpha} \prod_{M \models l_i} p_i$$

Xu, Van Den Broeck et al (2018)

What kind of foundations are emerging?

- Given a loss function L and a regularizing term L', the regularized loss function is a convex combination $(1-\lambda)L + \lambda L'$, where $\lambda \in [0,1]$.
- For any propositional formula ϕ, define the probability for interpretation m as:
  - $1/|\mathscr{M}_\phi|$ if $m \in \mathscr{M}_\phi$
  - 0 otherwise

The notion of a constraint distribution

- Given constraint distribution $c \in \mathscr{D}$, we define regularizer L_c for $p \in \mathscr{D}$ as:
  - $L_c(p) = dist_{\mathscr{D}}(p, c)$
- For example, given events $E = \{e_1, \ldots, e_n\}$,

$$dist_{\mathscr{D}}(p, q) \propto \sum_{e \in E} \sqrt{p(e)} \times \sqrt{q(e)}$$

Which means logically:

$$L_\phi(p) \propto \sum_{e \in \mathscr{M}_\phi} p(e) \times \frac{1}{|\mathscr{M}_\phi|}$$

Compare to semantic loss:

$$L(\alpha, p) \propto -log \sum_{M \models \alpha} \prod_{M \models l_i} p_i$$

There seems to be **principled foundation for constrained distributions**

Conclusions

- **Interesting challenge:** get distributions to obey constraints
- Use geometric interpretation to establish common grounds
- Can we push expressiveness of constraints?

Are regularisers worth it?

- Whether to use logic-based regularizers in deep learning depends on the specific application and the trade-offs between accuracy and computational efficiency
- Can improve performance, but their necessity may differ in certain applications or may not be worth the added computational cost
- What about expressiveness?
- Hybrid approach of external predicates
- Symbolic execution engine allows for increased modularity?