Magnitude Invariant Parametrizations Improve Hypernetwork Learning

ICLR 2024

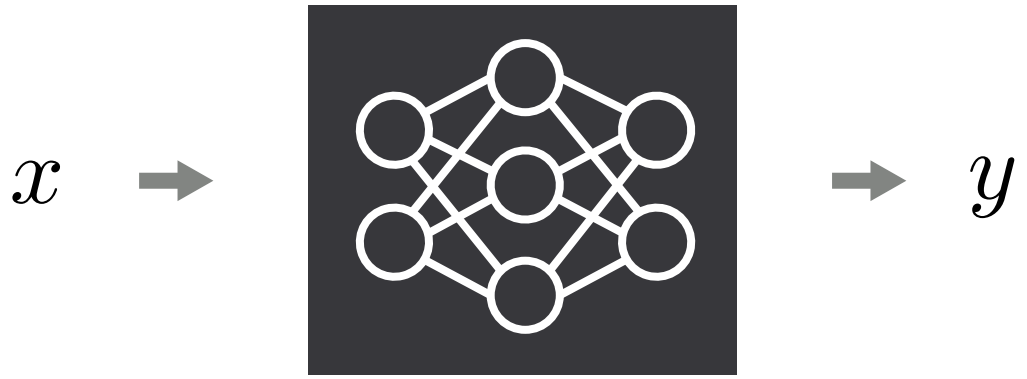
Jose Javier
Gonzalez Ortiz

Work done in collaboration with John Guttag and Adrian Dalca

Magnitude Invariant Parametrizations Improve Hypernetwork Learning – arXiv:2304.07645

What is a hypernetwork?

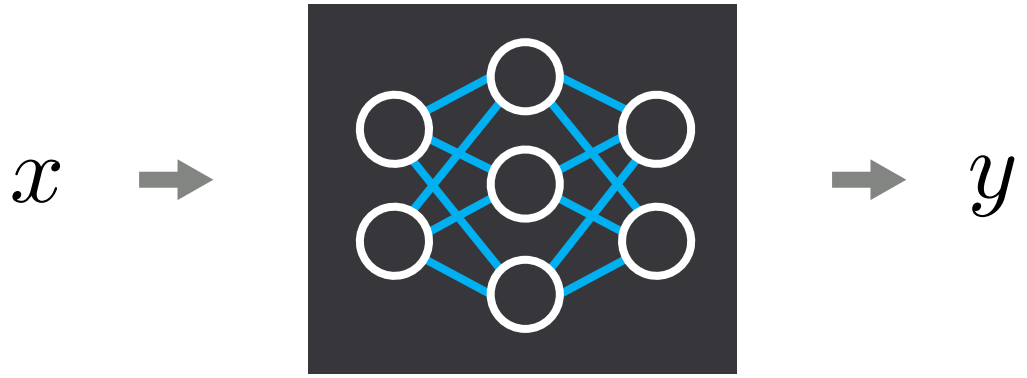
In a regular neural network, we make a prediction from a given input using a set of learnable parameters θ



$$f(x;\theta) \rightarrow y$$

What is a hypernetwork?

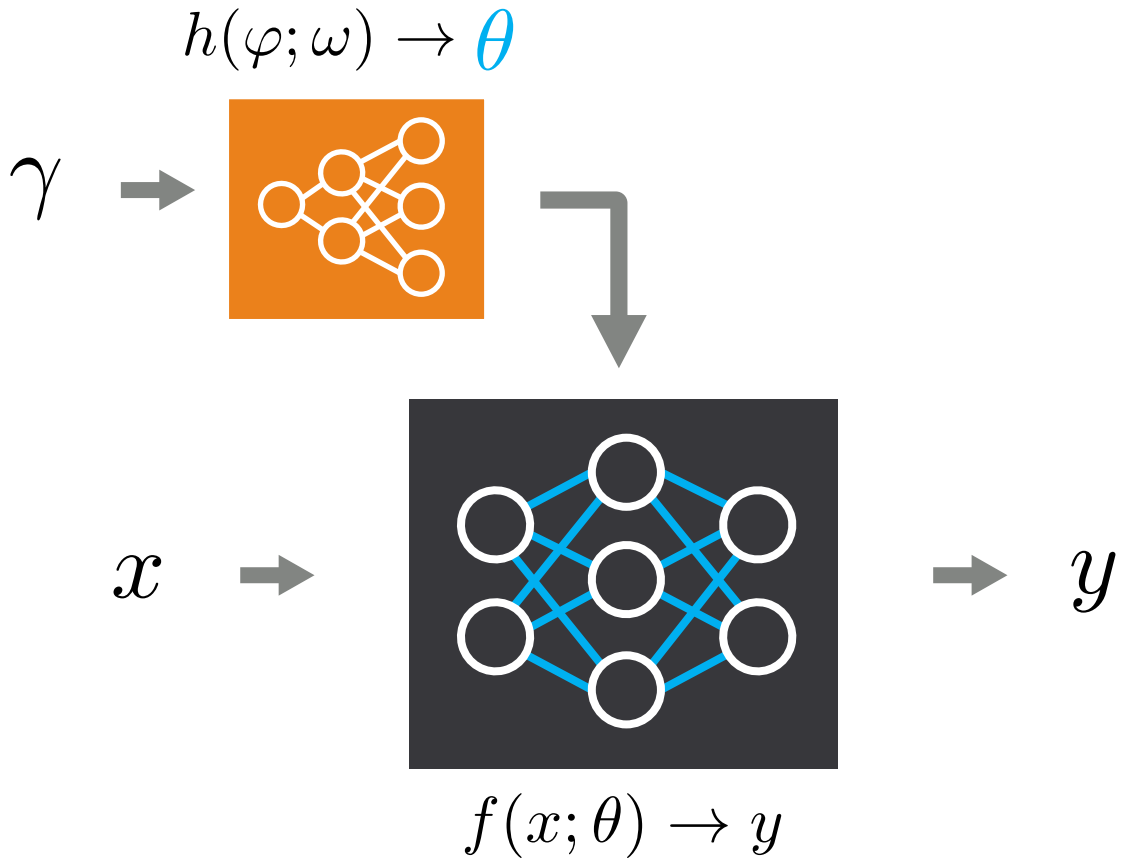
In a regular neural network, we make a prediction from a given input using a set of learnable parameters θ



$$f(x;\theta) \rightarrow y$$

What is a hypernetwork?

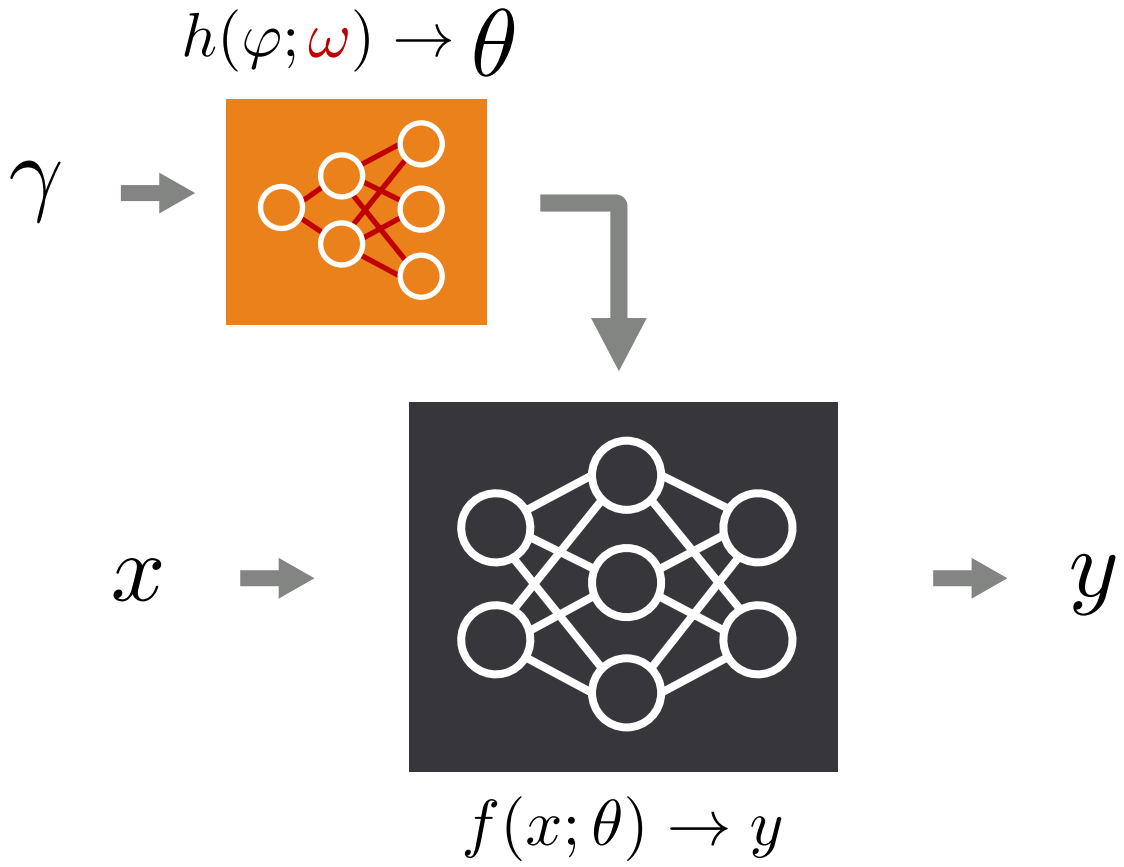
We can use another neural network – a hypernetwork[5] – to generate the parameters of our primary neural network



[5] HyperNetworks - David Ha, Andrew Dai, Quoc V. Le – *ICLR 2017*

What is a hypernetwork?

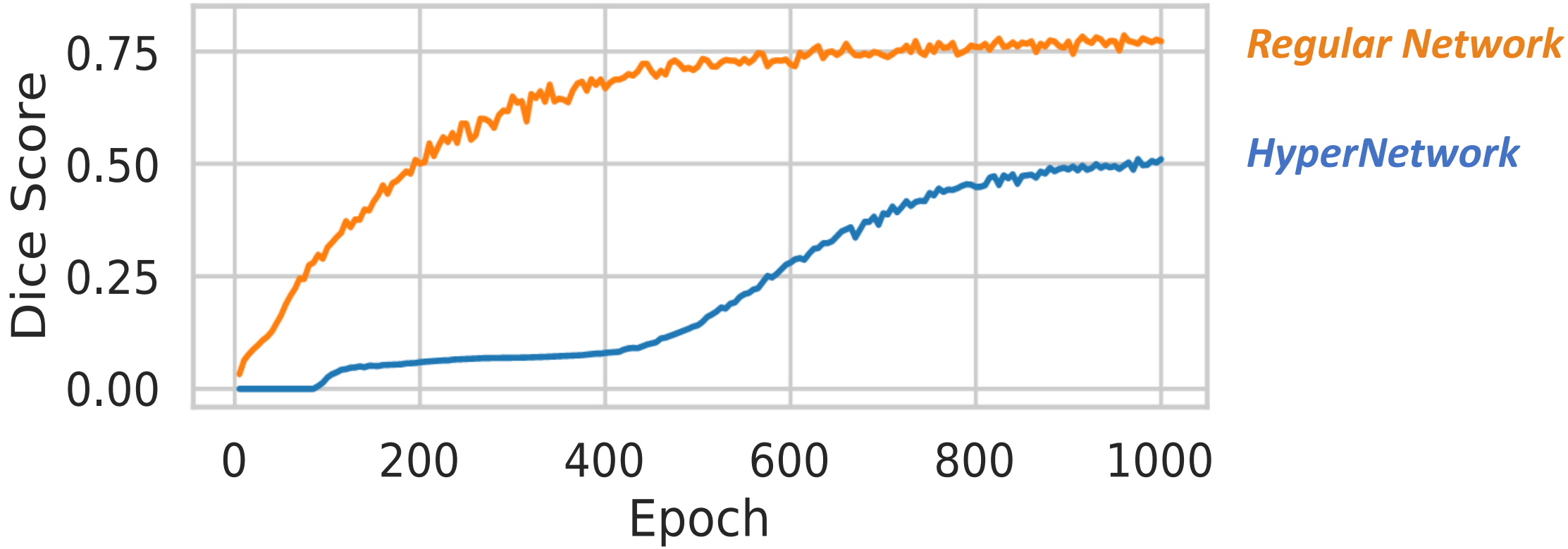
We want to learn the hypernetwork weights



[5] HyperNetworks - David Ha, Andrew Dai, Quoc V. Le – *ICLR 2017*

Problems with hypernetwork training

Hypernetworks are more challenging to train than regular networks



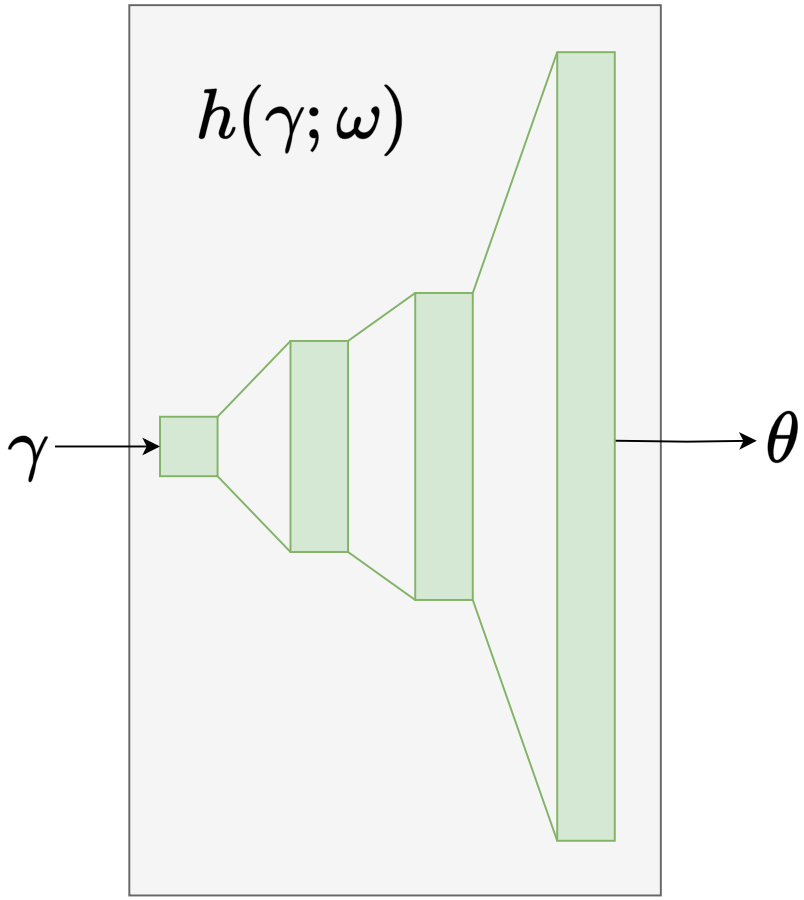
Problems with hypernetwork training

Hypernetworks are more challenging to train than regular networks

Why does this happen?
Can we do better?

HyperNetwork Proportionality

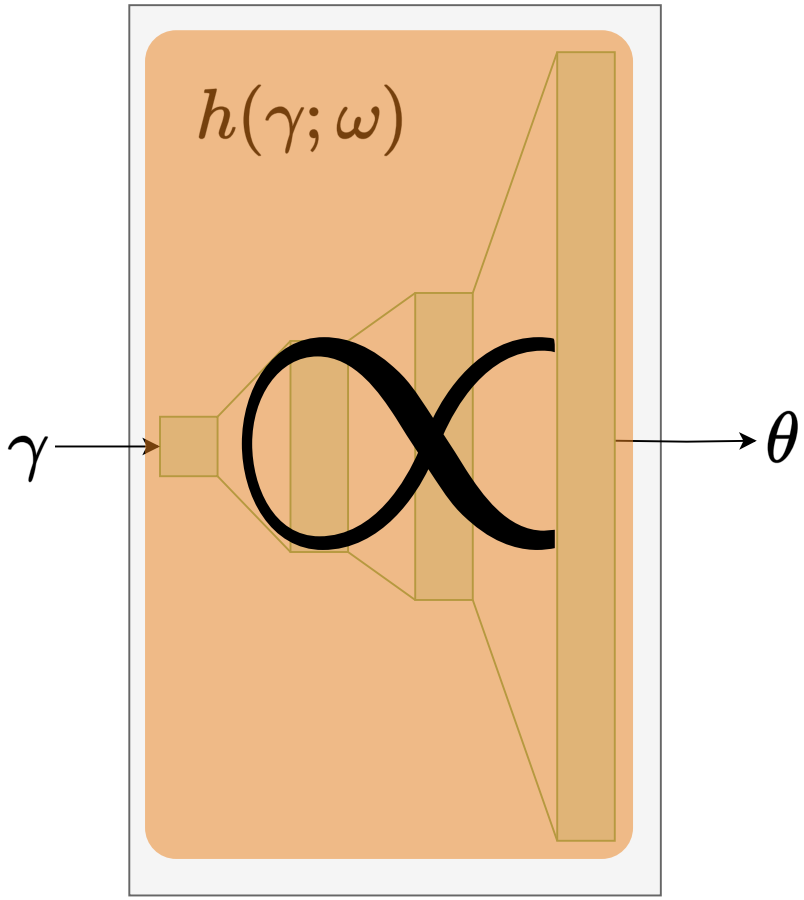
Common hypernetwork formulations use recommendations designed for regular neural networks



$$h(\gamma;\omega) = W^{(n)}x^{(n)} + b^{(n)}$$
$$x^{(k+1)} = \phi(W^{(k)}x^{(k)} + b^{(k)})$$
$$x^{(1)} = \gamma$$

HyperNetwork Proportionality

For common hypernetwork formulations, there exists a **proportionality dependency** between inputs and outputs



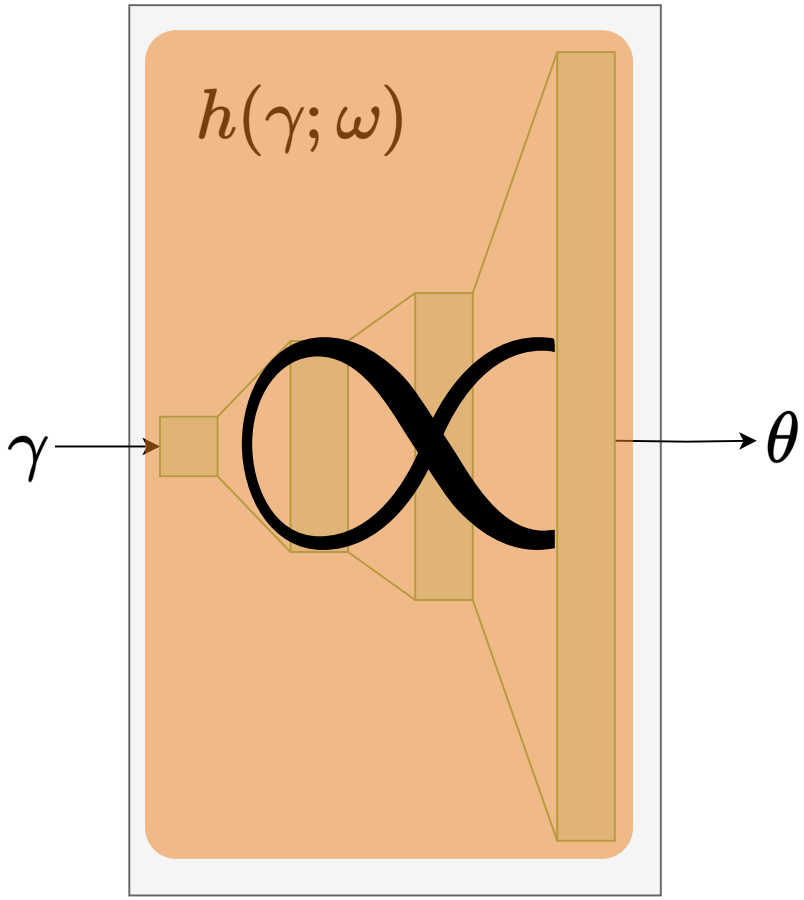
$$x_i^{(1)} = \gamma$$

$$\begin{aligned} x_i^{(2)} &= \phi(W_i^{(1)}\gamma + b^{(1)}) \\ &= \gamma\phi(W_i^{(1)}) \propto \gamma \end{aligned}$$

$$\begin{aligned} x_i^{(k+1)} &= \phi\left(b_i^{(k)} + \textstyle\sum_j W_{ij}^{(k)} x_j^{(k)}\right) \\ &= \gamma\,\phi\left(\textstyle\sum_j W_{ij}^{(k)} \alpha_j^{(k)}\right) \propto \gamma \end{aligned}$$

HyperNetwork Proportionality

Weight variance and gradient magnitude are proportional to hypernetwork input value, and this is **detrimental to optimization**



$$||\theta||_2 \propto ||\gamma||$$

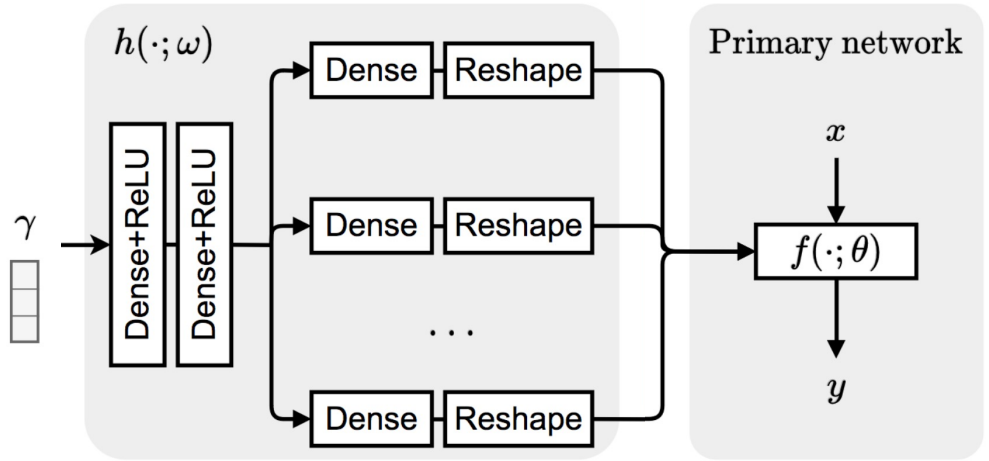
$$\mathrm{Var}(\theta) \propto ||\gamma||^2$$

$$\mathrm{Var}(\nabla_\theta \mathcal{L}) \propto ||\gamma||^2$$

Magnitude Invariant Parametrizations

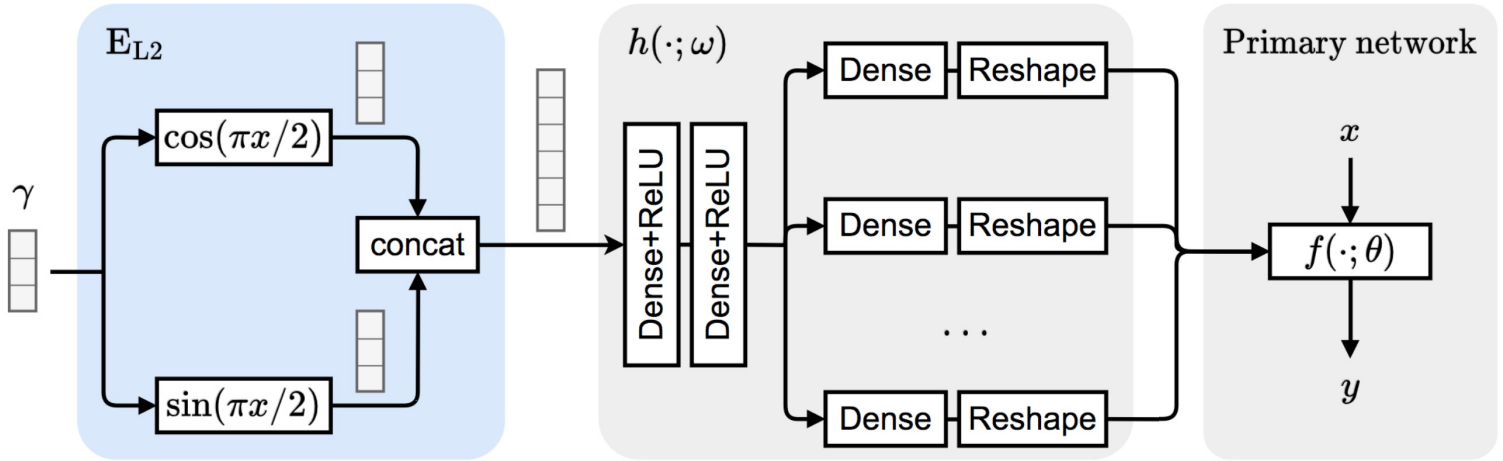
Magnitude Invariant Parametrizations

In the **default** setting the input is mapped directly to the primary network weights



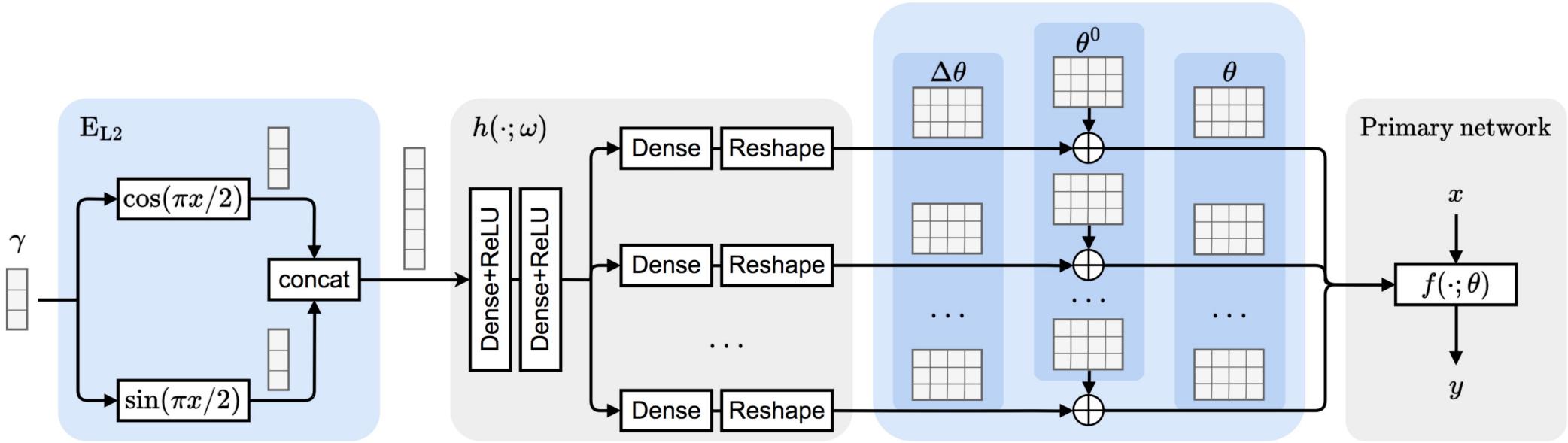
Magnitude Invariant Parametrizations

Before the hypernetwork, we process inputs using a constant norm **input encoding**



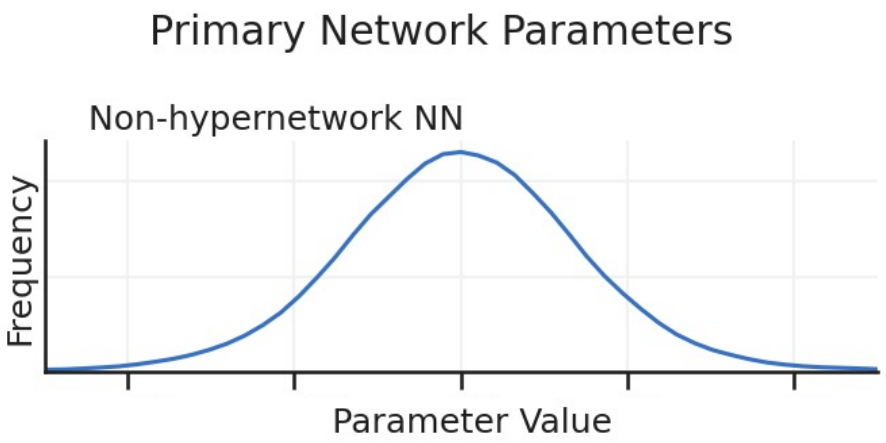
Magnitude Invariant Parametrizations

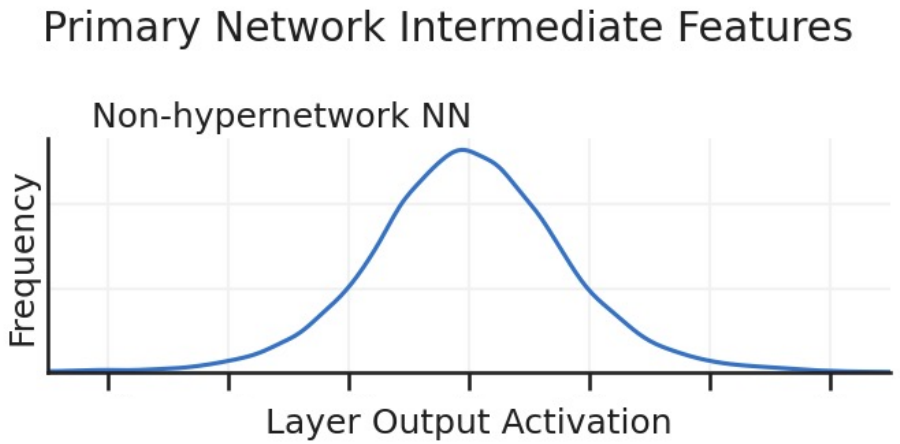
We reframe the hypernetwork predictions as **parameter changes**, instead of predicting parameters directly



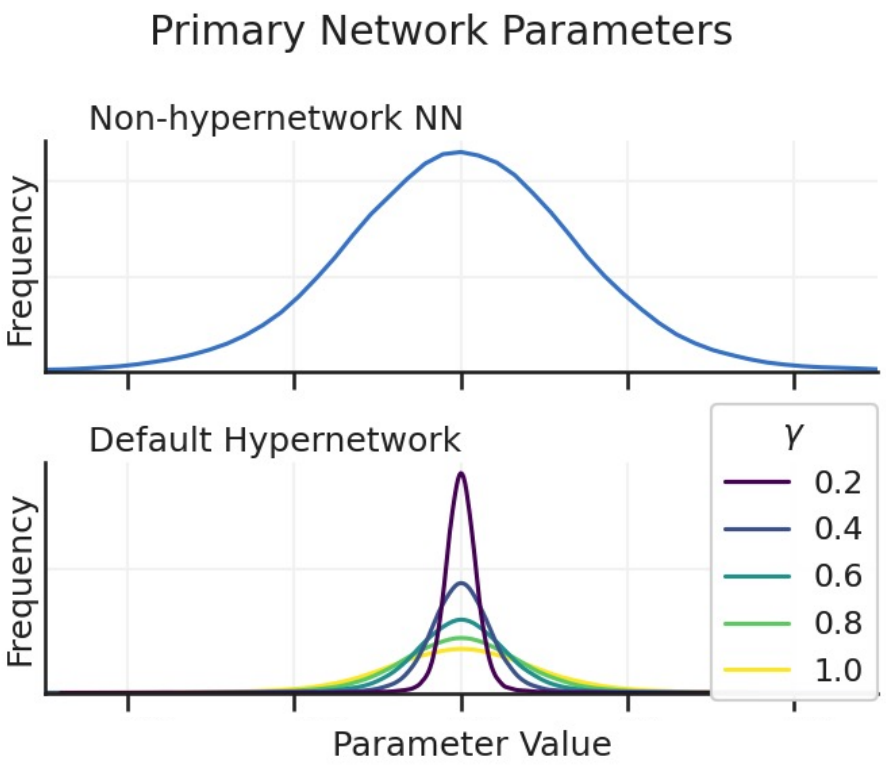
Experiments

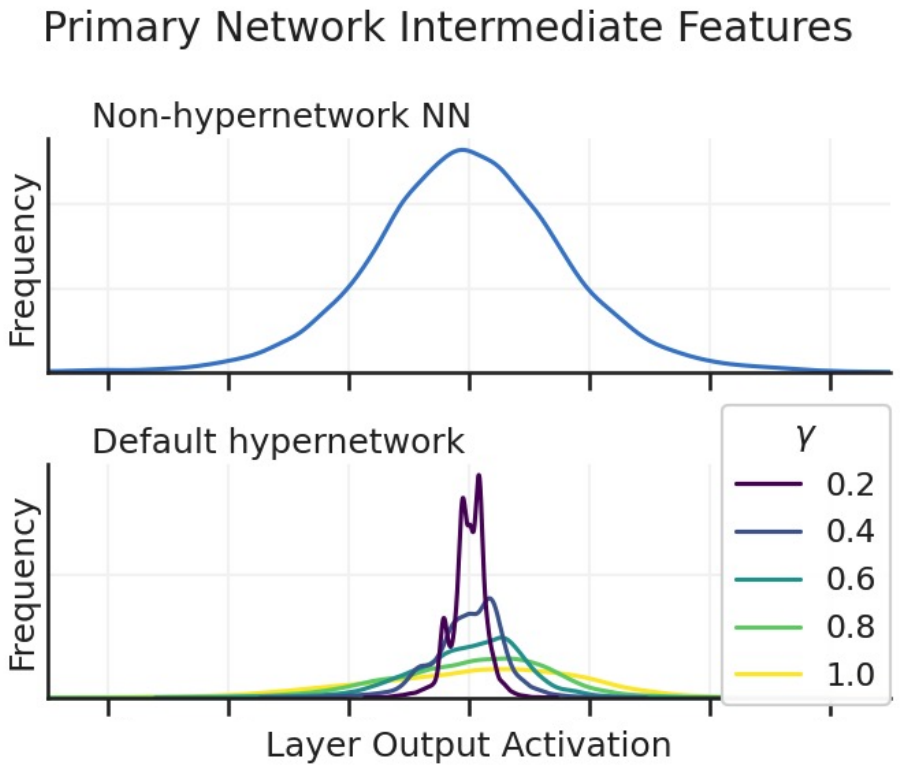
Studying Parameter & Feature Distributions



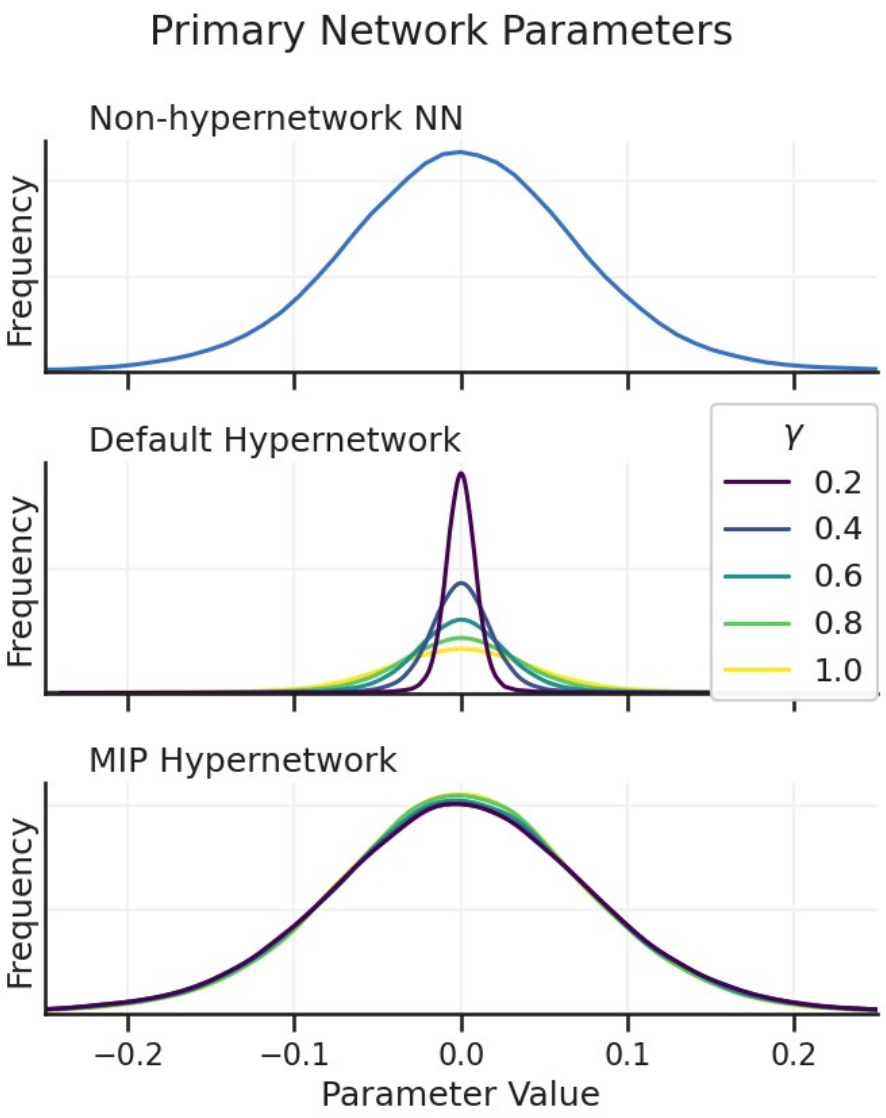


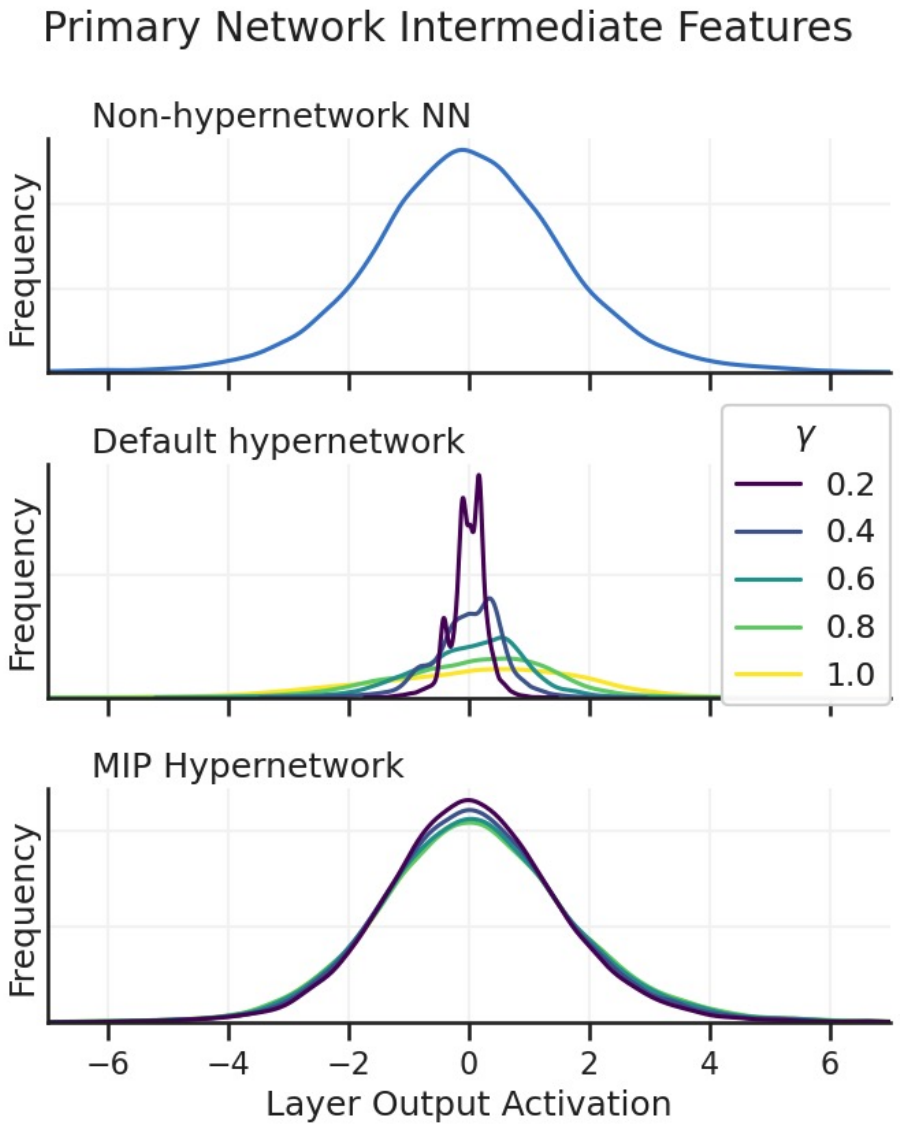
Studying Parameter & Feature Distributions





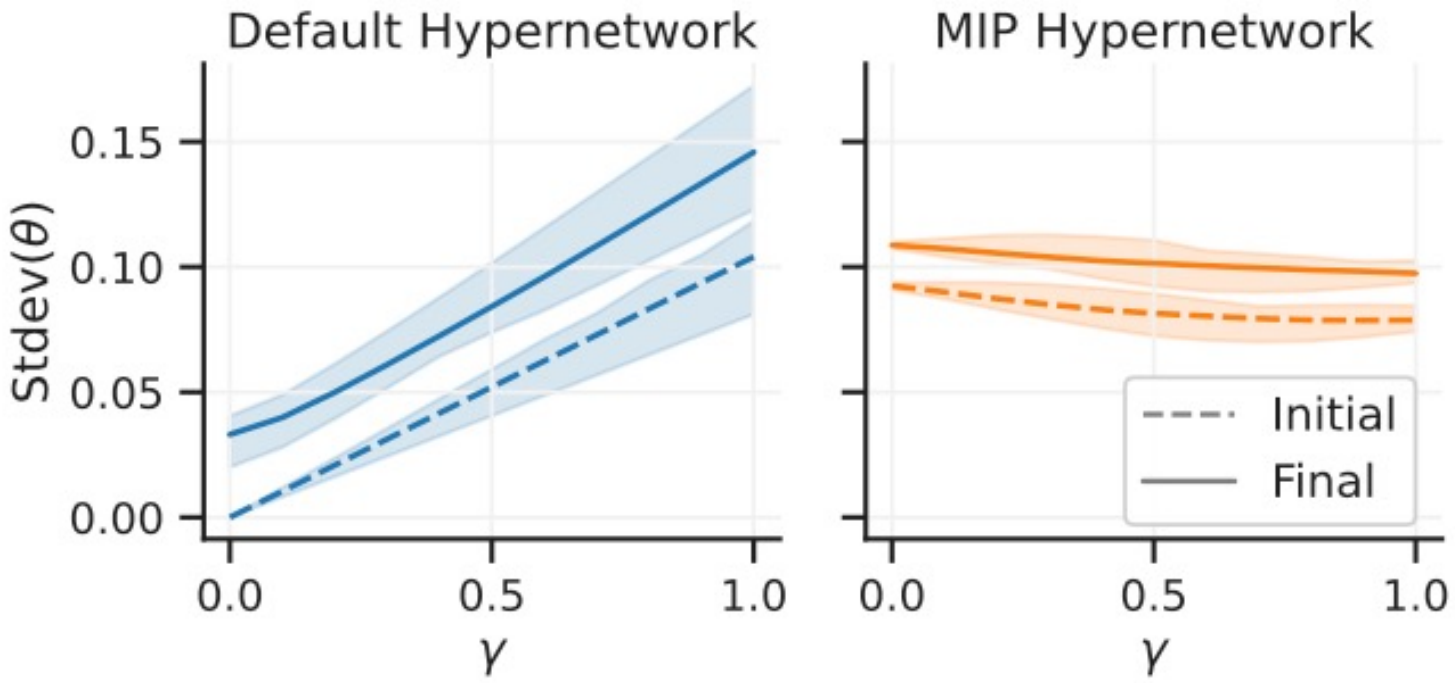
Studying Parameter & Feature Distributions





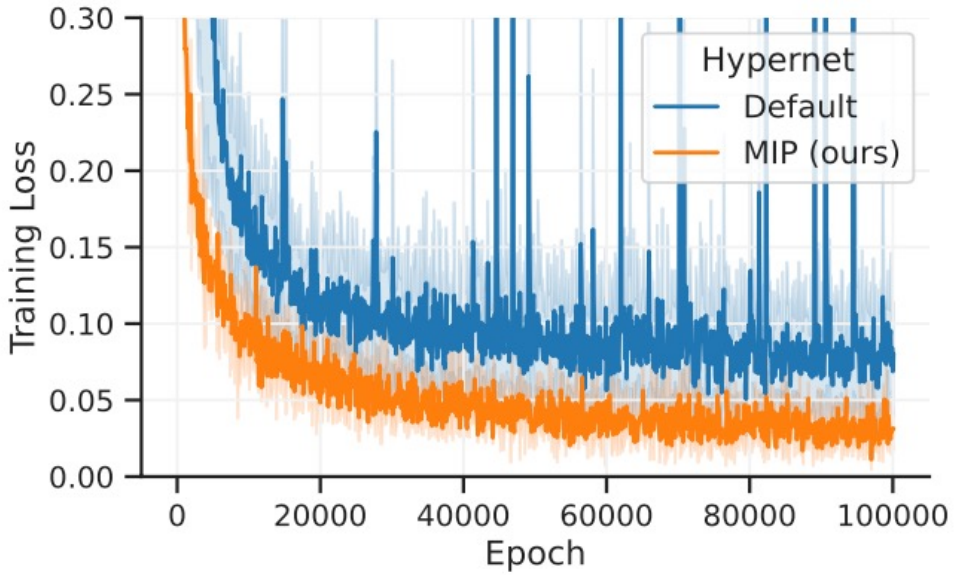
Improving Hypernetwork training

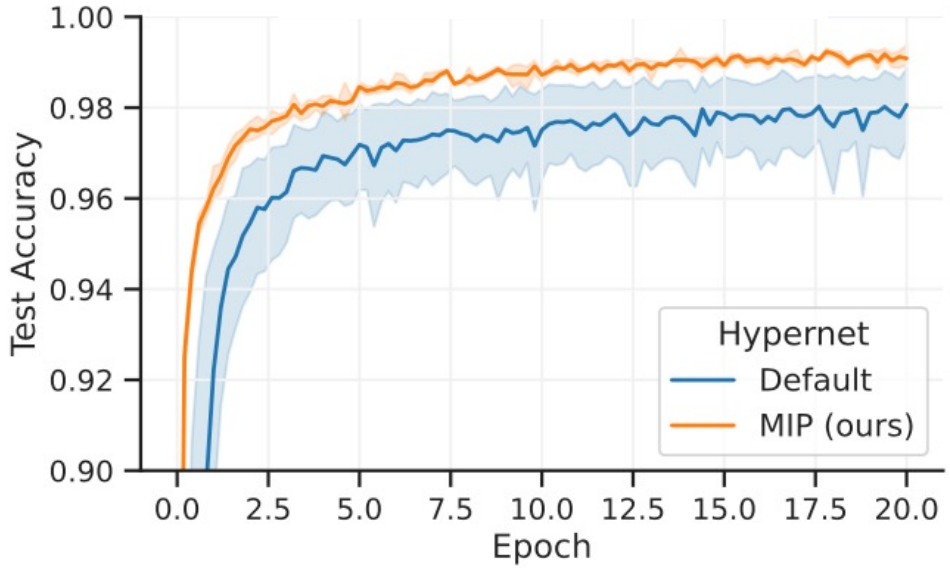
Empirical measurements reveal that this proportionality persists into training, leading to unstable optimization



Model Training Improvements

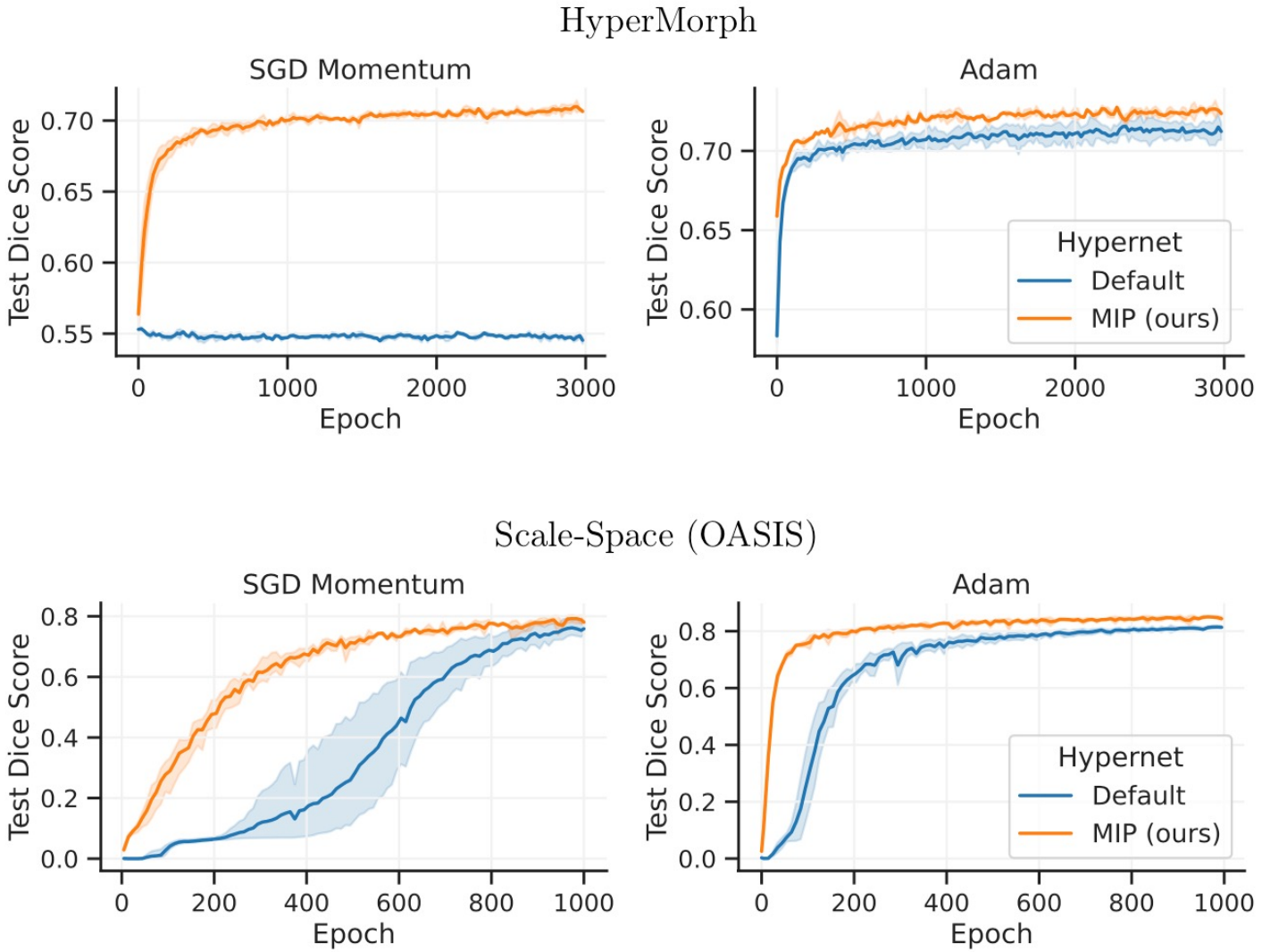
Our formulation stabilizes training and achieves faster convergence





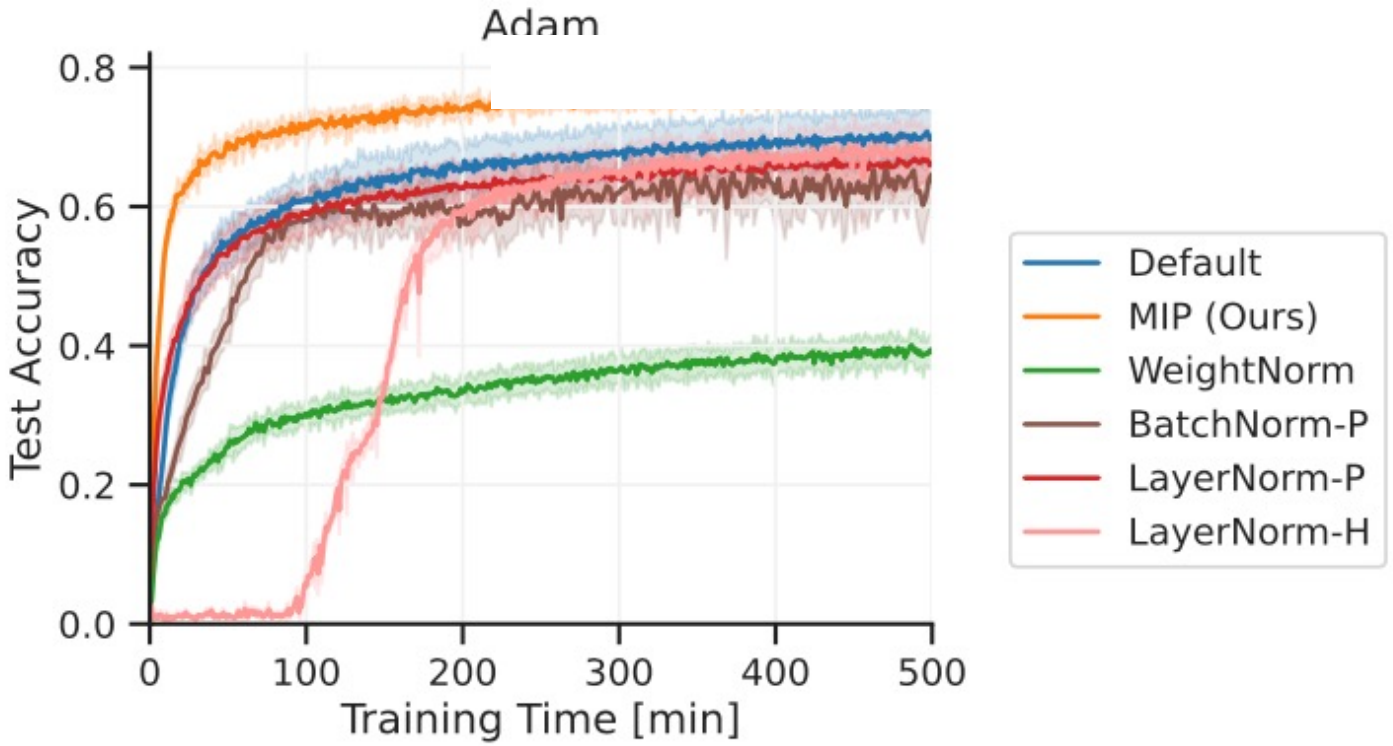
Model Training Improvements

Our parametrization dramatically improves **model convergence** and training stability, under all considered settings



Comparison to normalization strategies

Applying **common normalization strategies fails** to address the proportionality issue and does not improve training like MIP does.



Summary

Common hypernetwork formulations are susceptible to a **training instability**, caused by a proportionality between hypernetwork inputs and outputs.

Incorporating **Magnitude Invariant Parametrizations** leads to substantial improvements in convergence and training stability across multiple tasks.