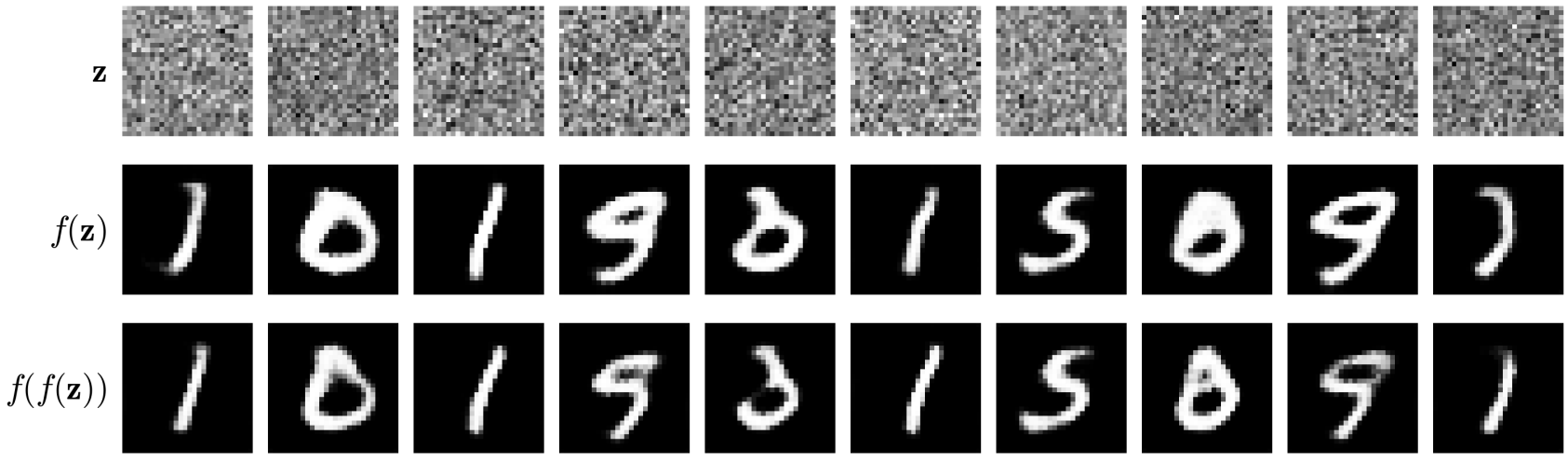
Enforcing Idempotency in Neural Networks

Nikolaj Banke Jensen[1] Jamie Vicary[2]

[1]University of Oxford
[2]University of Cambridge



Presented at ICML 2025

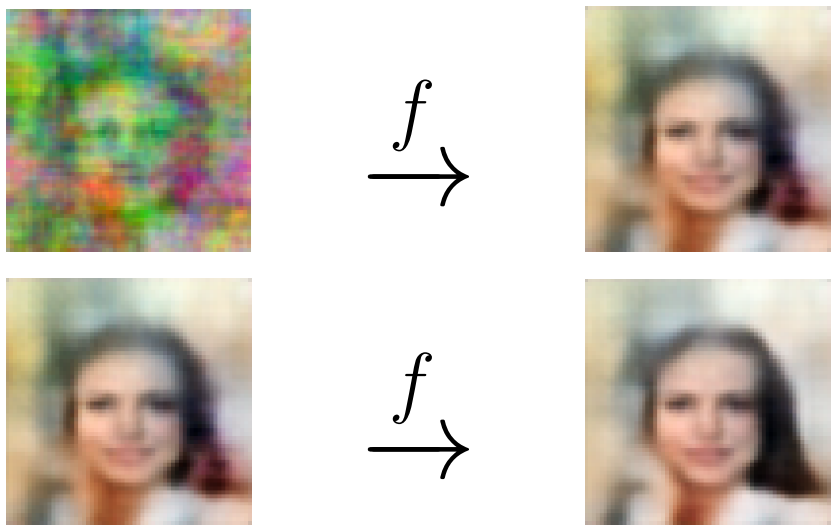
Why should I care about idempotency?

$f : X \to X$ is idempotent if it can be applied multiple times with no effect beyond the first application.

$$f(x) = f(f(x))$$

Some examples:

- Image generation
- Sorting algorithms
- Denoising
- Data compression
- ...



What's the normal way?

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be the network, then for a sample $\mathbf{x} \in \mathbb{R}^{m \times n}$, we can minimize the quantity

$$\mathcal{L}_{\text{idem}}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{m} \left(f(f(\mathbf{x}_i)) - f(\mathbf{x}_i) \right)^2$$

Some problems with this:

- Even for tiny networks we get relatively poor improvements.
- For deep networks, $\nabla_{\mathbf{w}} \mathcal{L}_{\text{idem}}$ becomes unwieldy if memoization is not employed.

An idea from Perturbation Theory

Let $\mathbf{K}$ be an "almost" idempotent matrix.

Up to some power j we define the ansatz

$$\mathbf{K}' = \alpha_1 \mathbf{K} + \alpha_2 \mathbf{K}^2 + \cdots + \alpha_j \mathbf{K}^j$$

If we demand $\mathbf{K}'$ satisfies $(\mathbf{K}')^2 - \mathbf{K}' = \mathbf{0}$ (i.e., it is idempotent), then we can solve for α_i such that $\mathbf{K}' = g(\mathbf{K})$.

We impose that g must "make $\mathbf{K}$ idempotent".

NB: Assume that $\boldsymbol{\varepsilon}^2 = \mathbf{0}$, $\mathbf{P}^2 = \mathbf{P}$, and $\boldsymbol{\varepsilon}\mathbf{M}\boldsymbol{\varepsilon} = \mathbf{0}$ for all $\mathbf{M}$, reducing the number of terms.

The solution

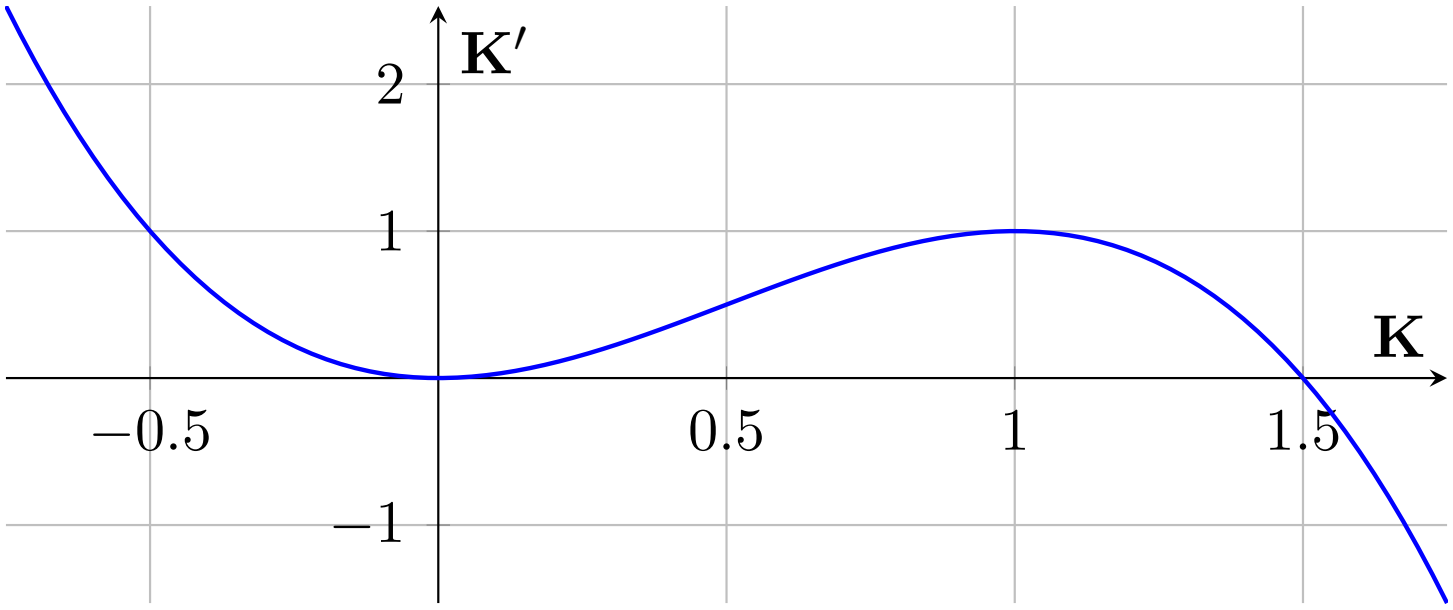


Figure: Plot of $\mathbf{K}' = 3\mathbf{K}^2 - 2\mathbf{K}^3$ in the case $\mathbf{K}$ is a scalar.

This has nice properties:

1. All idempotent matrices are solutions,
2. *Only* idempotent matrices are attracting points,
3. Wide area of attraction around idempotent points.

A general training scheme

Our iterator describes a desired change in the output $\mathbf{y} = f(\mathbf{x})$ of the network:

$$\mathbf{y}' = 3f(\mathbf{y}) - 2f(f(\mathbf{y}))$$
$$\Delta f(\mathbf{x}) = \mathbf{y}' - \mathbf{y}$$

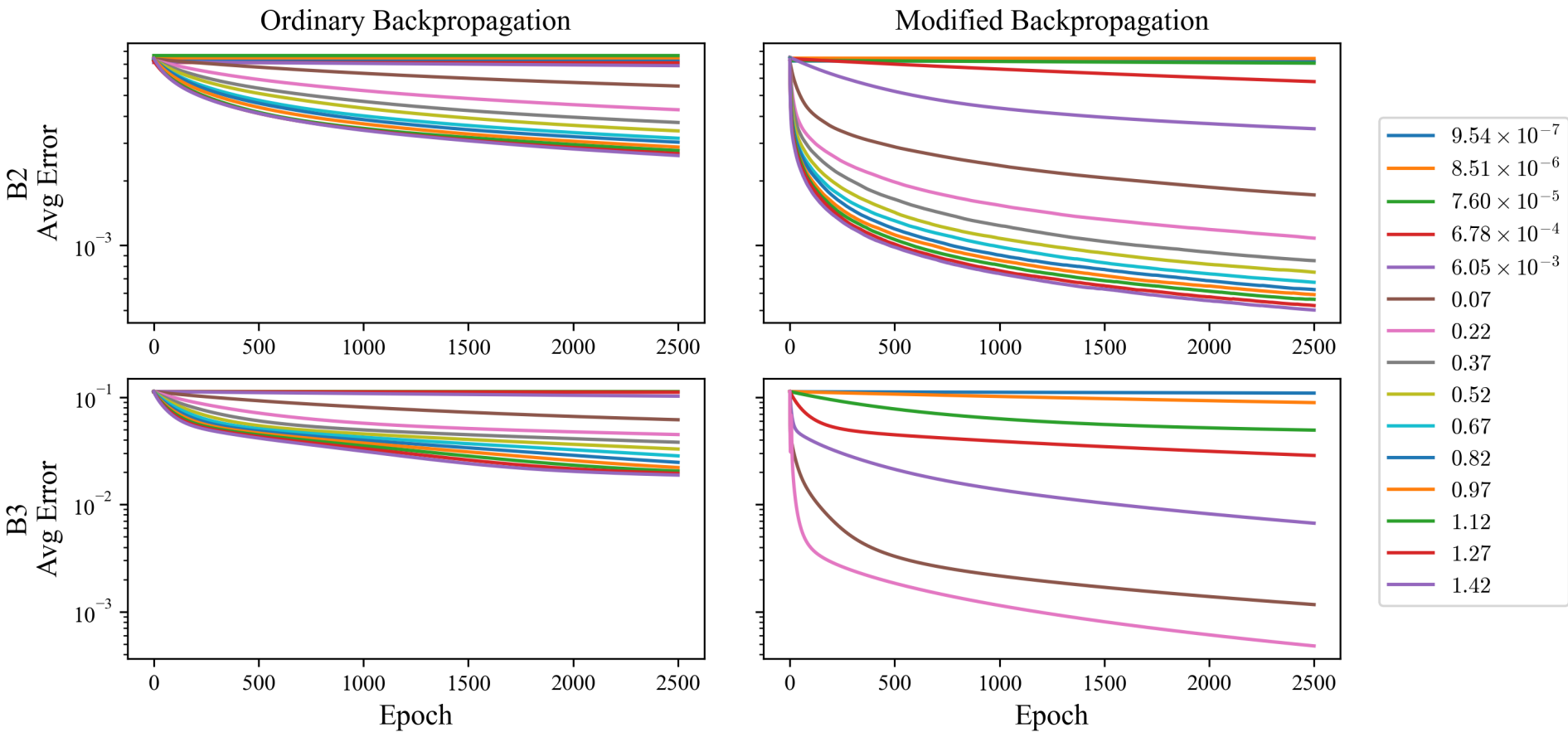
A simple training scheme therefore takes

$$\frac{\partial(-\mathcal{L}(\mathbf{y}))}{\partial \mathbf{y}} \equiv \Delta f(\mathbf{x})$$

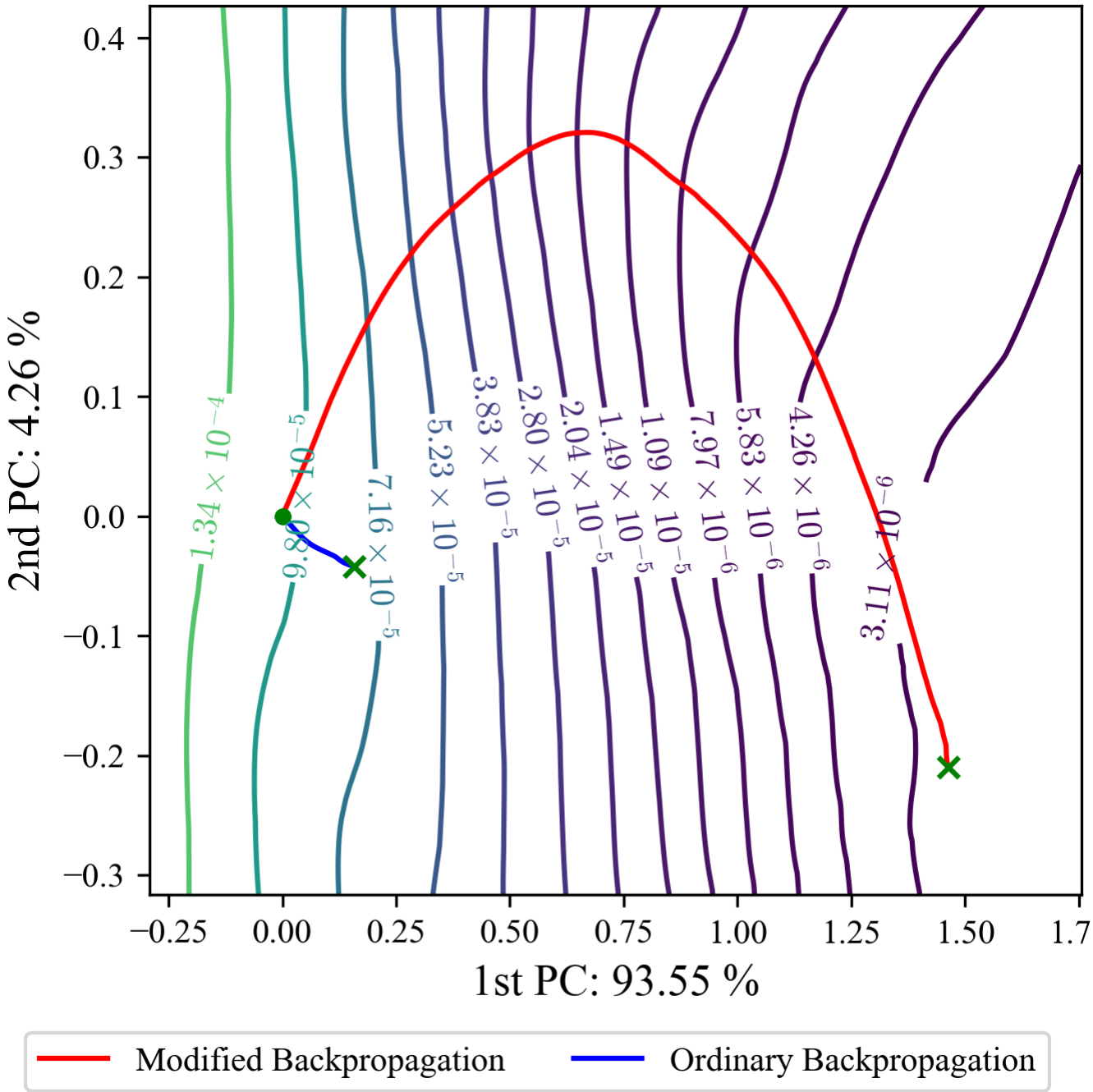
throughout the computational graph.

This is architecture agnostic.

Improved efficacy



Optimisation trajectory



Generative results

Replicating result of Shocher et al. 2023
"*Idempotent Generative Network*"