# Enforcing idempotency in neural networks

**Nikolaj Banke Jensen**
Department of Computer Science
University of Oxford
nikolaj.jensen@cs.ox.ac.uk

**Jamie Vicary**
Department of Computer Science and Technology
University of Cambridge
jamie.vicary@cl.cam.ac.uk

## Abstract

In this work we propose a new architecture-agnostic method for training idempotent neural networks. An idempotent operator satisfies $f(x) = f(f(x))$, meaning it can be applied iteratively with no effect beyond the first application. Neural networks used in image generation, sorting algorithms, data compression, and many other areas represent non-linear idempotent projections. Using methods from perturbation theory we derive the recurrence relation $\mathbf{M}' \leftarrow 3\mathbf{M}^2 + 2\mathbf{M}^3$ for iteratively projecting a real-valued matrix $\mathbf{M}$ and show that it has 1) idempotent fixed points, and 2) is attracting only around idempotent points. We give an extension to non-linear networks by considering our approach as a de-facto gradient for the loss function, achieving an architecture-agnostic training scheme. We provide experimental results for a variety of architectures demonstrating more than 10x improvement in idempotent error over the canonical gradient-based approach. Finally, we demonstrate scalability as we train a generative network successfully using only a simple reconstruction loss paired with our method.

## 1 Introduction

Using neural networks as data augmentation tools is becoming more widespread in areas such as signal processing and generative artificial intelligence. In particular, networks of the form $f : X \to X$, mapping data within the same space $X$, are frequently used in image augmentation, video generation, sorting algorithms, normalization algorithms, image denoising, and image generation, among others. While there is a variety of such transformation tasks, some of these can be considered *naturally idempotent* in the sense that the operation they are designed to carry out is idempotent. An idempotent operation is one which can be applied iteratively with no effect beyond the first application. For instance, it is intuitive to expect sorting algorithms to be naturally idempotent as sorting an already sorted data structure is needless. On the other hand, image augmentation is an example of a task which is not always naturally idempotent: rotating an image by 90° twice is not the same as rotating it once, for instance. As such, some tasks have *only* idempotent solutions and others have *no* idempotent solutions. This work, however, is concerned with actively enforcing idempotency as a component of the loss function, hence we focus on tasks which have both idempotent and non-idempotent solutions. In section 3 we study idempotency in generative networks, where it is the formal requirement of one-step inference, but also denoising and image augmentation (e.g., application of effect filters) are examples of tasks where idempotent solutions may be advantageous.

In this paper we are primarily concerned with networks $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$, where $\boldsymbol{\theta}$ is a collection of weight parameters. The condition that $f_\theta$ is idempotent is

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\mathbf{x})) \tag{1}$$

for all $\mathbf{x} \in \mathbb{R}^n$. If $f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}\mathbf{x}$ (a single-layer, fully-connected network with no bias and the identity activation function) where $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the weight matrix, then we recover the familiar notion from linear algebra where $\mathbf{W}^2 = \mathbf{W}$ and eigenvalues of $\mathbf{W}$ are either 0 or 1. Condition 1 gives the

Preprint. Under review.

correct notion for non-linear networks acting as idempotent projections, and can be optimized using a simple mean-squared error loss, $\mathcal{L}_{\text{idem}} = \frac{1}{m}\sum_{i=1}^{m}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) - f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))^2$, where $\mathbf{x} \in \mathbb{R}^{n\times m}$. As we show in section 3, minimizing this loss using canonical gradient descent can yield relatively poor improvement in the idempotent loss. Additionally, due to the higher-order application of $f_{\boldsymbol{\theta}}$ the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{idem}}$ can become unwieldy for deep networks.

In this work, we propose an alternative method for training neural networks to satisfy condition 1. Using ideas from Perturbation Theory we derive a function $g$ which solves $\mathbf{M}' = g(\mathbf{M})$ such that if $\mathbf{M}$ is an "almost" idempotent matrix, then $\mathbf{M}'$ is perfectly idempotent (i.e., $(\mathbf{M}')^2 = \mathbf{M}$). In this work, we focus on one such function:

$$g(\mathbf{M}) = 3\mathbf{M}^2 - 2\mathbf{M}^3 \tag{2}$$

Although we assume $\mathbf{M}$ is close to idempotent, we show that in practice $g$ can be used to derive matrices which are within machine precision of perfect idempotency even when the input matrix $\mathbf{M}$ is relatively far from idempotent. At a high level, this process is based on a recurrence relation $\mathbf{M}' \leftarrow (1-\gamma)\mathbf{M} + \gamma g(\mathbf{M})$, taking small $\gamma$-sized steps in the direction of $g(\mathbf{M})$. While this recurrence relation derives idempotent matrices – and can therefore be used to train single-layer networks with identity activations to be idempotent – we also give a more general application of Eq. 2 as a minor modification of the backpropagation algorithm, yielding an architecture agnostic and efficient algorithm for finding idempotent networks.

In section 2 we give a detailed description of the method used to derive Eq. 2 and alternative solutions. We also show that while there exists non-idempotent fixed points to Eq. 2, these points are repelling under the recurrence relation $\mathbf{M}' \leftarrow (1-\gamma)\mathbf{M} + \gamma g(\mathbf{M})$ for $0 \leq \gamma < 1$, giving credence to the use of such a recurrence relation in practice. Finally, in section 2 we derive a full training scheme for training arbitrary neural network architectures of the form $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$. In section 3, we present experimental data collected for a variety of fully-connected network architectures, showing that our method outperforms ordinary backpropagation under ideal conditions. We also replicate the results of Shocher et al. by applying our method on a U-net style DCGAN network to successfully create a generative network for MNIST and CelebA datasets. Lastly, sections 4 and 5 discuss how our method distinguishes itself from related approaches as well as practical limitations of our method.

## 2 Method

*Outline the perturbation analysis which yielded the update rule. Jordan Normal form justification for looking at eigenvalues and for fixed-points. Stability analysis (and nice fractals) to argue that specific fixed-points are reached. Compare ordinary and modified autodiff by looking at derivative on a single-layer, non-bias network.*

### 2.1 An idea from Perturbation Analysis

### 2.2 Fixed Points and Stability Analysis

### 2.3 Deriving a Training Scheme

## 3 Experimental Results

*On experimental networks we give line graphs comparing absolute error at varying learning rates. We also give line graphs for LRs over many epochs. Takeaway message is that for deeper/wider networks we outperform ordinary autodiff by a lot.*

*We replicate the results of IGN on MNIST and CelebA. Latent space analysis. Takeaway is that we show application to a U-net GAN architecture based on Conv layers.*

## 4 Related Work

*Review Idempotent Generative Networks, contrasting our work on a gradient-free approach. Have others applied perturbation theory to ML? We suffer same problems as GANs: mode collapse.*

## 5 Conclusion

*Give a conclusion of central idea: the use of perturbation analysis to find an iterator which we have successfully applied to a range of toy examples and GAN scenarios.*