一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

Banachの不動点定理に基づいた深層平衡モデルの高速化

佐藤尚樹†　飯塚秀明†

† 明治大学
E-mail: †naoki310303@gmail.com, ††iiduka@cs.meiji.ac.jp

あらまし　深層平衡モデル（DEQ）は，ニューラルネットワークの層の変換の不動点を探索することで，層を積み重ねることなく無限に深いネットワーク表現を実現する．DEQ は従来のモデルと比べて大幅に少ないメモリしか必要としないにもかかわらず，多くの大規模な数値実験において最先端のモデルに匹敵する性能を誇る．しかし，入力を受け取るたびに収束の保証のない不動点反復を実行するため，従来のモデルに比べて訓練と推論に膨大な計算時間を要するという課題に直面している．本稿では，不動点収束が保証されるようにモデルアーキテクチャを再構築することで，不動点収束を改善し，結果として計算時間を削減するアプローチを紹介する．我々が提案する画像分類のための Lipschitz MDEQ は，ハイパーパラメータの調整によってフォワードパスとバックワードパスの両方で理論的に不動点収束が保証される．CIFAR-10 データセットを使った数値実験では，わずかに精度が低下する代わりに最大 4.75 倍の高速化を達成した．
キーワード　深層平衡モデル，Banach の不動点定理，縮小写像

Naoki SATO† and Hideaki IIDUKA†

† Meiji University
E-mail: †naoki310303@gmail.com, ††iiduka@cs.meiji.ac.jp

Abstract　The Deep Equilibrium Model (DEQ) achieves infinitely deep network representations without stacking layers by exploring fixed points of layer transformations in neural networks. Despite requiring significantly less memory than conventional models, DEQ demonstrates performance comparable to state-of-the-art models in numerous large-scale numerical experiments. However, it faces the challenge of requiring enormous computational time for both training and inference compared to conventional models, as it performs fixed-point iterations with no convergence guarantee each time it receives input. This paper introduces an approach that improves fixed-point convergence and consequently reduces computational time by reconstructing the model architecture to guarantee fixed-point convergence. Our proposed Lipschitz MDEQ for image classification theoretically guarantees fixed-point convergence in both the forward and backward passes through hyperparameter tuning. Numerical experiments on the CIFAR-10 dataset achieved up to 4.75x faster speedup at the cost of a slight accuracy drop.
Key words　Deep equilibrium models, Banach fixed point theorem, contractive mapping

1. はじめに

近年，機械学習は著しい進歩を遂げており，深層ニューラルネットワークはますます巨大になっている．ResNet や Transformer といった一般的なモデルは，層が増えれば増えるほど性能が向上することが多くのタスクで実証されている．一方で，現代の深層学習では経験損失関数の最小化に勾配情報を利用する手法が主流であり，勾配を入手するためには誤差逆伝播法に頼らざるを得ない．誤差逆伝播法は全ての中間層の出力を保存しておく必要があるため，モデルの層が増えれば増えるほど膨大なメモリが必要となる．この問題を回避するために，必要なメモリを少なく抑えつつ，優れた性能を発揮できるモデルの研究が進められてきた．

深層平衡モデル（Deep Equilibrium Model, DEQ）[3], [5] は，従来の層を積み上げるアプローチとは根本的に異なる．従来のモデルはいくつもの層による変換を順番に適用して出力を得るが，DEQ は 1 層の変換を不動点に達するまで何度も適用し，得られた不動点を出力とすることで，実質的に無限の深さのニューラルネットワークを実現している．この暗黙的な定式化には，訓練時のメモリ使用量が劇的に削減されるという重大な利点がある．従来の N 層のモデルが誤差逆伝播法のために $O(N)$ のメモリを必要とするのに対し，DEQ が必要とするメモリは $O(1)$ で一定であり，モデルの深さとメモリコストを切り離すことができる．これにより，従来の深層ネットワーク

を悩ませてきたメモリの制約なしに，理論上無限に深いネットワークの表現が可能となる．驚くべきことに，言語処理タスク用の DEQ-Transformer [3] と画像処理タスク用の Multiscale DEQ（MDEQ）[5] の両方が，この優れたメモリ効率を達成しつつ，最先端モデルに匹敵する性能を発揮している．

このような長所があるにもかかわらず，訓練と推論に膨大な時間を要するという致命的な欠陥によって，DEQ は主流のアーキテクチャにはなっていない．先行研究の実験結果によれば，DEQ は従来のモデルに比べて 2.5 倍から 3 倍遅いことが示されている [3],[5]．我々は，この計算の遅さの原因は，DEQ が解く不動点問題に解が一意に存在する保証がないことにあると主張する．DEQ は入力を受け取るたびに適当な不動点ソルバーを使って以下の不動点問題を解く．

$$\text{Find } z \in \mathrm{Fix}(f_\theta) := \left\{ z \in \mathbb{R}^d \colon f_\theta(z; \boldsymbol{x}) = z \right\}, \tag{1}$$

ここで，z は隠れ状態，$\boldsymbol{x}$ は入力，$\boldsymbol{\theta}$ は重みパラメータ，$f_\theta \colon \mathbb{R}^d \to \mathbb{R}^d$ は単一の反復変換である．Banach の不動点定理 [7],[18] によれば，この不動点問題に一意に定まる解が存在するのは，写像 f_θ が縮小写像である場合に限られる．ここで，写像 $f \colon \mathbb{R}^d \to \mathbb{R}^d$ が縮小写像であるとは，$r \in (0,1)$ が存在し，任意の $z_1, z_2 \in \mathbb{R}^d$ に対して $\|f(z_1) - f(z_2)\| \le r\|z_1 - z_2\|$ が成り立つことをいう．言い換えれば，写像 f が $L < 1$ に対して L-Lipschitz 写像であることと同値である．写像 f が縮小写像である場合，Banach の不動点近似法によって生成される点列 $z_{t+1} := f(z_t)$ は f の唯一の不動点に収束することが知られている．しかし，f_θ は畳み込み，正則化，活性化関数などの複雑な演算の合成写像であるため，一般に縮小性を満たさない．その結果，不動点ソルバーは入力を受け取るたびに収束保証のない反復を余儀なくされ，訓練と推論の両方で膨大な計算コストが発生する．

そこで本稿では，不動点問題 (1) に唯一の解が存在するようにモデルアーキテクチャ（すなわち f_θ）を再設計するアプローチを紹介する．それによって不動点収束を理論的に保証し，収束を加速させることで DEQ の訓練と推論に必要な膨大な計算時間を削減できる．本稿では画像処理タスク向けの MDEQ の改良に焦点を絞る．

2. 背景と関連研究

2.1 重み共有モデル

DEQ の源流は，TrellisNet [4] や Universal Transformer [10] といった，全ての層で重みを共有するモデルにある．これらのモデルは層数に比例してパラメータ数が増加しないという意味で定数メモリを実現している．驚くべきことに，全ての層で重みが共有されているにもかかわらず，従来のモデルと同等かそれ以上の性能が得られることが実証されている．しかし，誤差逆伝播のためにメモリが層数に比例するという欠点は依然として解決されていなかった．Bai らは，層数が増加するにつれて，これらのモデルの中間層の出力，すなわち隠れ状態 z が変換 f_θ の不動点に収束することを実験的に観察した [3]．ここで，f_θ は各層による変換を表し，全ての層で重み $\boldsymbol{\theta}$ が共有されていることから，各層が施す変換は全て f_θ で表現できることに注意されたい．この実験的観察に基づいて，層を増やして不動点を得るのではなく，層の変換 f_θ の不動点 $z^\star$ を直接求めるアプローチ，深層平衡モデルが提案された．この手法は Transformer に組み込まれ，大規模な言語モデリングタスクにおいて極めて強力な性能を発揮した [3]．その後すぐに，画像処理タスクへの適用のためにマルチスケール機構を組み込んだ Multiscale DEQ (MDEQ) が提案され，画像分類とセマンティックセグメンテーションの両方において同様に優れた性能を発揮することが実証された [5].

2.2 深層平衡モデル

DEQ は入力を受け取ると，フォワードパスにおいて何らかの不動点ソルバーを使って式 (1) の不動点問題を解く．ソルバーの反復回数は DEQ の層数に対応する．不動点 $z^\star$ が求まると，これを使って経験損失関数 $\ell_{z^\star}(\boldsymbol{\theta})$ を定義し，確率的勾配降下法や Adam などの標準的な最適化アルゴリズムで損失最小化を進める．最適化アルゴリズムが必要とする勾配情報は，連鎖律と陰関数定理を適用することで，次のように得ることができる [17]:

$$\frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \underbrace{\frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial z^\star}\left(I - J_{f_\theta}(z^\star)\right)^{-1}}_{=:A} \frac{\partial f_\theta(z^\star; \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \tag{2}$$

ここで，J_{f_θ} は $z^\star$ における f_θ のヤコビ行列である．計算コストの高いヤコビ行列の逆行列の計算を回避するため，項 A はバックワードパスにおいて以下の式を解くことで近似される：

$$\mathbf{x}\left(I - J_{f_\theta}(z^\star)\right) - \frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial z^\star} = \mathbf{0}. \tag{3}$$

最初の項は，PyTorch [20] などの自動微分パッケージを用いて，任意の $\mathbf{x}$ に対して効率的に計算できる．線形方程式 (3) は，以下の条件を満たす不動点を求める問題と同値である：

$$\mathbf{x} = T(\mathbf{x}) \text{ under } T(\mathbf{x}) := \mathbf{x} J_{f_\theta}(z^\star) + \frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial z^\star} \tag{4}$$

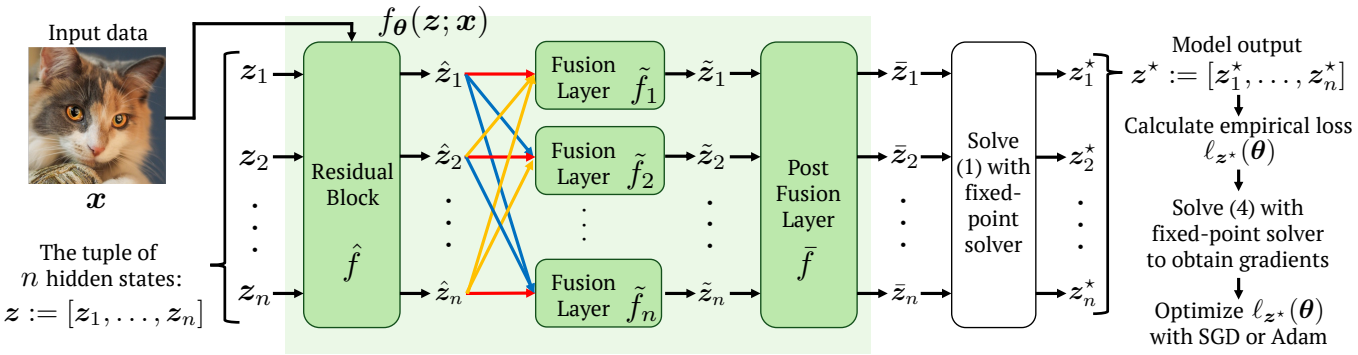
つまり，DEQ はフォワードパスとバックワードパスで 2 つの異なる不動点問題を解く．どちらの不動点問題でも，近年の研究では Anderson 加速法 [2] が不動点ソルバーとして採用されている．

2.3 DEQ の不動点収束の理論保証に関する関連研究

一般的に，DEQ が解く不動点問題において不動点の存在と一意性は保証されない．これまでに，不動点収束を保証する DEQ の変種が複数の先行研究で提案されている．Winston らは monotone DEQ (monDEQ) を導入し，層を単調作用素として設計することで不動点の存在を保証した [28]．最近では，Sittoni らは SubDEQ を提案し，写像 f_θ に準単調性を課すことで有界解の存在性と一意性を保証している [24]．さらに，Gabor らは positive concave DEQ (pcDEQ) を導入し，写像 f_θ を縮小写像と非拡大写像に分解することで一意な不動点を保証している．

2.4 DEQ の高速化に関する関連研究

DEQ は，訓練と推論が極めて遅いことでも知られている．この問題に対処するため，先行研究では近似勾配を利用する



MDEQ Architecture [Bai et al., 2020]

$$\begin{cases} \hat{z}_i = \hat{f}_i(z_i) := \mathrm{GN}(\mathrm{ReLU}(z_i + \mathrm{GN}(\mathrm{Drop}(\mathrm{Conv}(\mathrm{ReLU}(\mathrm{GN}(\mathrm{Conv}(z_i)))))))) & \text{(M1)} \\ \tilde{z}_i = \tilde{f}_i(\hat{z}_i) := \hat{z}_i + \sum_{j=1}^{i-1} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} \mathrm{U}_{i,j}(\hat{z}_i) & \text{(M2)} \\ \bar{z}_i = \bar{f}_i(\tilde{z}_i) := \mathrm{GN}(\mathrm{Conv}(\mathrm{ReLU}(\tilde{z}_i))) & \text{(M3)} \end{cases}$$

Lipschitz-MDEQ Architecture (ours)

$$\begin{cases} \hat{z}_i = \hat{f}_i(z_i) := \mathrm{MGN}(\mathrm{SReLU}((1-\alpha_1)z_i + \alpha_1 \mathrm{MGN}(\mathrm{Drop}(\mathrm{Conv}^\star(\mathrm{SReLU}(\mathrm{MGN}(\mathrm{Conv}^\star(z_i)))))))) & \text{(L1)} \\ \tilde{z}_i = \tilde{f}_i(\hat{z}_i) := (1-\alpha_2)\hat{z}_i + \alpha_2 \left(\sum_{j=1}^{i-1} w_{ij} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} w_{ij} \mathrm{U}_{i,j}(\hat{z}_i) \right) & \text{(L2)} \\ \bar{z}_i = \bar{f}_i(\tilde{z}_i) := \mathrm{MGN}(\mathrm{Conv}^\star(\mathrm{SReLU}(\tilde{z}_i))) & \text{(L3)} \end{cases}$$

図 1　MDEQ および Lipschitz MDEQ のアーキテクチャ．Lipschitz MDEQ は MDEQ の構造を継承し，各演算は Lipschitz 定数が大きくなりすぎるのを防ぐように修正されている．図のレイアウトと表記は先行研究 [5], Figure 1 から引用したものであり，結果の説明を明確化するためわずかな修正を加えている．サンプル画像は AFHQ データセット [8] から引用された．

ことで逆伝播 (2) の計算コストを削減することに成功している [12],[14],[19],[22]．特に，Fung らによる Jacobian Free Backpropagation（JFB）は，$(I - J_{f_\theta}(z^\star))^{-1}$ を単位行列で置き換えることで高速化を実現している．一方，Geng らは，計算が容易な代理損失 $\frac{1}{2}\|f_\theta(z^\star) - z^\star\|^2$ の勾配，ファントム勾配を用いることで高速化を達成した [14]．DEQ の高速化という観点では，我々のアプローチに最も近い先行研究はヤコビアン正則化 [6] である．ヤコビアン正則化は，ヤコビ行列のノルム $\|J_{f_\theta}(z^\star)\|_\mathrm{F}$ をバックワードパスにおける罰金項として用いることで学習を安定させる．ヤコビアン正則化には理論的な収束保証はないものの，性能低下が最小限に抑えられるという実用上の利点がある．不動点収束を安定化させ，少ない反復回数で高精度を達成できるようにする手法だが，不動点収束を保証したり加速したりするものではないことに注意されたい．

その他の関連する研究については，文献 [23] の Section 2 と Appendix A を参照されたい．

3. 主　結　果

3.1　記法と数学的準備

$\mathbb{N}$ を非負整数の集合とする．$n \in \mathbb{N}$ に対して，$[n] := \{1, 2, \dots, n\}$ と定義する．$\mathbb{R}^d$ を内積 $\langle \cdot, \cdot \rangle$ を備えた d 次元ユークリッド空間とし，これによりノルム $\|\cdot\|$ が誘導される．$f \circ g$ を写像 f と g の合成写像，すなわち $(f \circ g)(\cdot) := f(g(\cdot))$ とする．ここで，f^k は写像 f を k 回適用することを表し，f^0 は恒等写像を表す．要素ごとの積（Hadamard 積とも呼ばれる）を表すために $\odot$ を使用する．

［定義 1］(L-Lipschitz 写像)　任意の $x, y \in \mathbb{R}^d$ に対して $\|f(x) - f(y)\| \le L_f \|x - y\|$ が成り立つとき，写像 $f\colon \mathbb{R}^d \to \mathbb{R}^d$ は L_f-Lipschitz 写像であるという．また，L_f は写像 f の Lipschitz 定数と呼ぶ．

定義 1 にあるように，写像 f の Lipschitz 定数を L_f で表す．特定の演算については，L_Conv や L_ReLU などの簡略版を用いる．以下の補題は，MDEQ のフォワードパスにおける Lipschitz 定数を導出するために頻繁に使用される．

［補題 1］　写像 $f\colon \mathbb{R}^d \to \mathbb{R}^d$ と写像 $g\colon \mathbb{R}^d \to \mathbb{R}^d$ は，それぞれ L_f-Lipschitz 写像と L_g-Lipschitz 写像であるとする．このとき，写像 $f + g$ は $(L_f + L_g)$-Lipschitz 写像である．また，写像 $f \circ g$ は $L_f L_g$-Lipschitz 写像である．

まず，MDEQ に登場する不動点写像 f_θ の定義を明確化する．MDEQ では，各解像度ごとに隠れ状態が定義され，これらは並列処理されて互いに影響し合う．解像度の数を $n \in \mathbb{N}$ とし，最高解像度と最低解像度における隠れ状態をそれぞれ z_1 と z_n で表す．入力画像は z_1 にのみ直接入力され，他の隠れ状態は間接的にのみ入力を受ける．ただし，隠れ状態 $z := [z_1, \dots, z_n]$ に対する f_θ の Lipschitz 定数を検討しているため，入力の処理は無視できることに注意されたい．f_θ は，式 (M1) の Residual Block $\hat{f}\colon z \mapsto \hat{z}$，式 (M2) の Fusion Layer $\tilde{f}\colon \hat{z} \mapsto \tilde{z}$，式 (M3) の Post Fusion Layer $\bar{f}\colon \tilde{z} \mapsto \bar{z}$ の 3 つのブロックから構成される（図 1 参照）．なお，$f_\theta = \bar{f} \circ \tilde{f} \circ \hat{f}$ である．これらを構成する演算については，次の章で我々の修正とともに詳述する．

3.2　Lipschitz MDEQ のアーキテクチャ

この章では，MDEQ に現れる各演算を検討し，その Lipschitz

定数を導出する．次に，写像 f_θ の Lipschitz 定数が 1 未満となるよう各演算を修正し，それによって Lipschitz MDEQ を構築する．各演算の Lipschitz 定数は紹介にとどめ，その導出は文献 [23] の Appendix B に譲る．

a） 正規化演算

MDEQ は正規化演算にグループ正規化 (GN)[29] を採用しており，これはチャンネルのグループ内で特徴量を正規化する．入力特徴量 $\boldsymbol{z}$ に対する標準的な GN 演算は以下のように定義される．

$$\mathrm{GN}(\boldsymbol{z})_i = \gamma \frac{z_i - \mu_{\boldsymbol{z}}}{\sqrt{\sigma_{\boldsymbol{z}}^2 + \epsilon}} + \beta,$$

ここで，$\mu_{\boldsymbol{z}} \in \mathbb{R}$ と $\sigma_{\boldsymbol{z}}^2 \in \mathbb{R}$ は平均と分散であり，$\gamma, \beta \in \mathbb{R}$ は学習可能なアフィンパラメータである．この演算の Lipschitz 定数は，$L_{\mathrm{GN}} \le \frac{|\gamma|}{\sqrt{\epsilon}}$ である．ϵ はゼロ除算を防ぐための非常に小さい定数（10^{-5} など）であるため，分散がゼロに近づく最悪の場合，Lipschitz 定数が過度に大きくなる可能性がある．この Lipschitz 定数の爆発の可能性は，不動点反復の安定性にとって重大な問題となる．この問題を緩和するため，我々の Lipschitz MDEQ はより単純な正規化方式である平均グループ正規化（MGN）を採用する．この演算は分散による除算を省略し，平均を引くことでのみ特徴量を正規化する．

$$\mathrm{MGN}(\boldsymbol{z})_i = \gamma(z_i - \mu_{\boldsymbol{z}}) + \beta.$$

この一見小さな変更は，Lipschitz 定数に重大な影響を与える．MGN の Lipschitz 定数は $L_{\mathrm{MGN}} \le |\gamma|$ で与えられる．$1/\sqrt{\epsilon}$ 項を除去することで，Lipschitz 定数は学習可能パラメータ γ のみに直接依存するようになり，大幅に小さくなり，かつ制御しやすい．γ の上界 $\bar{\gamma}$ を別の調整可能なハイパーパラメータとして扱うことで，正規化演算の Lipschitz 定数を完全に制御することができる．実装において，学習可能パラメータ γ および β の使用は任意である．MDEQ の実装では，Residual Block および Fusion Layer に現れる GN 演算がこのオプションを利用しており，Lipschitz MDEQ もこの手法を採用している．これらを使用しない場合，$\gamma = 1$ と $\beta = 0$ が設定されるため，L_{MGN} は常に 1 となる．逆にこれらを使用する場合，$\bar{\gamma}$ を任意に決定することで L_{MGN} を制御できる．

b） 活性化関数

MDEQ は活性化関数に ReLU 関数を採用している．ReLU 関数は入力ベクトル $\boldsymbol{z}$ の各要素に対して施され，$\mathrm{ReLU}(\boldsymbol{z})_i := \max\{0, z_i\}$ と定義される．この演算の Lipschitz 定数は明らかに 1 である．全体の Lipschitz 定数をさらに小さく保つため，我々の Lipschitz MDEQ は Scaled-ReLU（SReLU）関数 [1] を採用する．SReLU は $\mathrm{SReLU}(\boldsymbol{z})_i := \max\{0, a z_i\}$ と定義される．ここで $a \in (0, 1]$ はハイパーパラメータであり，Lipschitz 定数は $L_{\mathrm{SReLU}} = a$ となる．

c） ドロップアウト

ドロップアウト演算 [25] は，過学習を抑制し汎化性能を向上させるための重要な手法である．MDEQ は変分ドロップアウト [13] を採用しており，$\mathrm{Drop}(\boldsymbol{z}) := \frac{1}{1-p}\boldsymbol{m} \odot \boldsymbol{z}$ で定義される．ここで $p \in (0, 1)$ はドロップアウト率，$\boldsymbol{m} \sim \mathrm{Bernoulli}(p)$ はマスクである．この操作の Lipschitz 定数は $L_{\mathrm{Drop}} = 1/(1-p)$ である．実装では，ドロップアウト率 p は通常 0 から 0.3 の間の小さな値であるため，ドロップアウト操作の Lipschitz 定数が過度に大きくなることはない．したがって，我々の Lipschitz MDEQ も変分ドロップアウトを採用する．

d） 畳み込み演算

標準的な畳み込み層は，入力特徴量 $\boldsymbol{z}$ に対してアフィン変換を実行する．この操作は $\mathrm{Conv}(\boldsymbol{z}) = W\boldsymbol{z} + \boldsymbol{b}$ と表すことができる．ここで W は畳み込みカーネルから導出される線形演算子であり，$\boldsymbol{b}$ はバイアス項である．この演算の Lipschitz 定数は $L_{\mathrm{Conv}} = \|W\|_2$ である．我々の Lipschitz MDEQ では，このスペクトルノルムを直接制御することで畳み込み層の Lipschitz 定数を調整する．具体的には，各最適化ステップ後に重みを射影することで制約 $\|W\|_2 \le c$ を課す．ここで $c > 0$ は調整可能なハイパーパラメータである．制約付きの畳み込み演算を $\mathrm{Conv}^\star$ と表記し，したがって $L_{\mathrm{Conv}^\star} = c$ となる．

e） 残差接続

MDEQ の Residual Block では，入力が単純に加算される残差接続が採用されている．例えば $h(x) = x + g(x)$ である．これは文献 [16] で導入された極めて効果的なメカニズムである．しかし，恒等写像の Lipschitz 定数は 1 であるため，補題 1 によれば $L_h = 1 + L_g$ となり，Lipschitz 定数の全体的な増加に寄与する．これを緩和する最も単純なアプローチは，$\alpha \in (0, 1)$ に対する凸結合を用いることであり，例えば $h'(x) = (1-\alpha)x + \alpha g(x)$ となる．すると $L_{h'} = (1-\alpha) + \alpha L_g$ となり，単純な残差接続よりも小さい Lipschitz 定数が得られる．したがって，Lipschitz MDEQ では全ての残差接続を凸結合に変更し，係数のパラメータをハイパーパラメータとして扱う．式 (L1) にあるように，Residual Block 内の凸結合パラメータは α_1 で表される．この凸結合構造はハイウェイネットワーク [26], [27] で用いられており，ゲート付き再帰ユニット（GRU）[9] や最近のゲート付き線形 RNN[11], [15], [21] におけるゲート機構の基本形式として広く採用されていることに留意されたい．

f） ダウンサンプリング

MDEQ の Fusion Layer において，高解像度から低解像度へ隠れ状態を渡す際，解像度の差に比例する畳み込み演算が適用される（図 1 の青矢印）．具体的には，$h_1(\boldsymbol{z}) := \mathrm{GN}(\mathrm{CONV}(\boldsymbol{z}))$ および $h_2(\boldsymbol{z}) := \mathrm{ReLU}(\mathrm{GN}(\mathrm{CONV}(\boldsymbol{z})))$ とすると，演算 $\mathrm{D}_{i,j}$ は $\mathrm{D}_{i,j} := h_1 \circ h_2^{i-j-1}$ と定義される．したがって，この演算の Lipschitz 定数は $L_{\mathrm{Down}}^{i,j} = L_{h_1} L_{h_2}^{i-j-1}$ となり，活性化関数，正規化演算，畳み込み演算の Lipschitz 定数に依存する．我々の Lipschitz MDEQ は，その形式を変更せずにダウンサンプリング演算を実装する．したがって，Lipschitz MDEQ においてこの操作の Lipschitz 定数は $L_{\mathrm{Down}}^{i,j} = L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} (L_{\mathrm{SReLU}} L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star})^{i-j-1}$ となる．

g） アップサンプリング

MDEQ の Fusion Layer において，低解像度から高解像度へ隠れ状態を渡す際，足りない次元は最近傍補間法を用いて補完される（図 1 の黄色矢印）．この演算 $\mathrm{U}_{i,j}$ は低解像度入力

$z_j \in \mathbb{R}^{H\times W}$ を受け取り，高解像度出力 $z_{j\to i} \in \mathbb{R}^{sH\times tW}$ を生成する．ここで $s,t \in \mathbb{N}$ はそれぞれ垂直方向と水平方向のスケール係数である．MDEQ では，両方のスケール係数は $2^{\text{level\_diff}}$ と定義される．ここで level_diff は隠れ状態のインデックス間の差である．例えば，z_1 と z_3 の間の level_diff は 2 である．この演算の Lipschitz 定数は $L_{\text{up}} = \sqrt{st}$，すなわち $2^{\text{level\_diff}}$ である．実装では解像度の数 n は 2 または 4 のいずれかであるため，Lipschitz 定数は最大で $2^3 = 8$ に達し，これが全体の Lipschitz 定数を爆発させる要因となりうる．Lipschitz MDEQ もこの操作を継承している．

h） Fusion Layer

式 (M2) にあるように，MDEQ の Fusion Layer において，$\tilde{z}_i$ はそれ自体 $\hat{z}_i$ と，ダウンサンプリングおよびアップサンプリングを経た全ての $\hat{z}_j$ $(i \neq j)$ の和である．しかし，特にアップサンプリング演算の Lipschitz 定数は解像度のレベル差に依存して大きくなる可能性があるため，単純に和をとると Fusion Layer において Lipschitz 定数が爆発する恐れがある．したがって，Lipschitz MDEQ はまず，自身 $\hat{z}_i$ と他のアップサンプリングまたはダウンサンプリングされた解像度 $\hat{z}_j$（$i \neq j$）との間で凸結合をとる．その後，アップサンプリングとダウンサンプリングに対して，重み付き平均を計算する．この際，level_diff が増加するにつれてより小さな重みを適用する．すなわち，

$$\tilde{z}_i = (1-\alpha_2)\hat{z}_i + \alpha_2\left(\sum_{j\neq i} w_{ij}\text{Fuse}_{i,j}(\hat{z}_i)\right),$$

ここで $\alpha_2 \in (0,1)$，$\text{Fuse}_{i,j}$ は $j < i$ の場合 $\text{D}_{i,j}$，$j > i$ の場合 $\text{U}_{i,j}$ であり，重み w_{ij} はアップサンプリング演算の level_diff に比例するペナルティに対するソフトマックス関数を通じて動的に計算される：

$$w_{ij} = \frac{\exp(-p_{ij})}{\sum_{k\neq i}\exp(-p_{ik})}, \text{where } p_{ij} = \begin{cases} j-i & \text{if } j > i \\ 0 & \text{if } j < i \end{cases}.$$

このメカニズムにより，level_diff が大きく，Lipschitz 定数が大きいアップサンプリング演算には，指数関数的に小さい重みが割り当てられる．このように重み付き平均をとることで，Fusion Layer の Lipschitz 定数を抑制する．

以上で MDEQ および Lipschitz MDEQ を構成する全ての演算の Lipschitz 定数の導出が完了した．我々の目的はこれらの演算で構成される全体の写像 f_θ の Lipschitz 定数を 1 未満に抑えることであった．f_θ の Lipschitz 定数を導出していこう．

3.3 Lipschitz MDEQ の Lipschitz 定数の導出

MDEQ の隠れ状態は，n 個の特徴量からなるタプル $z = [z_1,\dots,z_n]$ で構成される．ここで各 z_i は，d_i 次元ユークリッド空間 $\mathbb{R}^{d_i}$ 内のベクトルと見なすことができる．したがって，全体の隠れ状態 z は積空間 $\mathcal{Z} = \mathbb{R}^{d_1}\times\cdots\times\mathbb{R}^{d_n}$ に存在する．写像 f_θ は，隠れ状態の組 $z \in \mathcal{Z}$ を受け取り，$\bar{z} = f_\theta(z) \in \mathcal{Z}$ を出力する．我々は以下のように定義される $\mathcal{Z}$ 上のノルムに対する写像 f_θ の Lipschitz 定数を導出する．

$$\|z\|_{\mathcal{Z}} := \sqrt{\sum_{i=1}^{n}\|z_i\|_2^2},$$

ここで $\|\cdot\|_2$ はユークリッドノルムを表す．このノルムは，全ての隠れ状態 $z_i \in \mathbb{R}^{d_i}$ を連結して形成されるベクトルの L_2 ノルムと同値であり，ノルムの公理を自然に満たす．

a） Residual Block と Post Fusion Layer の Lipschitz 定数

Residual Block において，写像 $\hat{f}\colon \mathcal{Z}\to\mathcal{Z}$ は各解像度ブランチに対して独立に作用する．したがって，これを n 個の写像 $\hat{f}_i : \mathbb{R}^{d_i}\to\mathbb{R}^{d_i}$ に分解できる．入力 $z = [z_1,\dots,z_n]$ に対して，出力は単純に $\hat{f}(z) = [\hat{f}_1(z_1),\dots,\hat{f}_n(z_n)]$ となる．これらの関数は入力と出力の次元が異なるものの，それらを構成するモジュールは同一であることに注意されたい．すなわち，$\hat{f}_i$ は任意の $i \in [n]$ に対して同じ Lipschitz 定数 $\hat{L}$ を持ち，これを $\hat{L}$ と表記する．すると，次の式が成り立つ．

$$\begin{aligned}\|\hat{f}(z)-\hat{f}(z')\|_{\mathcal{Z}}^2 &= \sum_{i\in[n]}\|\hat{f}_i(z_i)-\hat{f}_i(z_i')\|_2^2 \\ &\le \sum_{i\in[n]}\hat{L}^2\|z_i-z_i'\|_2^2 \\ &= \hat{L}^2\|z-z'\|_{\mathcal{Z}}^2.\end{aligned}$$

したがって，$\hat{f}$ も同じ Lipschitz 定数 $\hat{L}$ を持つ．$\hat{f}_i$ の定義（式 (L1) を 参照）と補題 1 から，以下が示される．

$$\hat{L} = (1-\alpha_1)L_{\text{MGN}}L_{\text{SReLU}} + \alpha_1 L_{\text{MGN}}^3 L_{\text{Conv}\star}^2 L_{\text{SReLU}}^2 L_{\text{Drop}}. \quad (5)$$

同様に，Post Fusion Layer $\bar{f}$ における各ブランチの処理は完全に独立しているため，各ブランチの処理を $\bar{f}_i$ と表し，これらに対する Lipschitz 定数を $\bar{L}$ とすると，$\bar{f}$ に対する Lipschitz 定数も $\bar{L}$ となる．$\bar{f}_i$ の定義（式 (L3) を参照）から，次の式が成り立つ．

$$\bar{L} = L_{\text{MGN}}L_{\text{Conv}\star}L_{\text{SReLU}}. \quad (6)$$

b） Fusion Layer の Lipschitz 定数

Fusion Layer では隠れ状態が互いに影響し合うため，各ブランチの処理は独立していない．そこで，関数 $\tilde{f}_i\colon \mathcal{Z}\to\mathbb{R}^{d_i}$ を定義する．これは n 個の隠れ状態の組 $\hat{z} \in \mathcal{Z}$ を入力とし，単一の隠れ状態 $\tilde{z}_i \in \mathbb{R}^{d_i}$ を出力する．すなわち $\tilde{f}(\hat{z}) = [\tilde{f}_1(\hat{z}),\dots,\tilde{f}_n(\hat{z})]$ とする．$\tilde{f}_i$ が $\tilde{L}_i$-Lipschitz $(i \in [n])$ であるとする．すなわち，$\|\tilde{f}_i(\hat{z})-\tilde{f}_i(\hat{z}')\|_2 \le \tilde{L}_i\|\hat{z}-\hat{z}'\|_{\mathcal{Z}}$ とする．すると，次の式が成り立つ．

$$\begin{aligned}\|\tilde{f}(\hat{z})-\tilde{f}(\hat{z}')\|_{\mathcal{Z}}^2 &= \sum_{i\in[n]}\|\tilde{f}_i(\hat{z})-\tilde{f}_i(\hat{z}')\|_2^2 \\ &\le \sum_{i\in[n]}\tilde{L}_i^2\|\hat{z}-\hat{z}'\|_{\mathcal{Z}}^2.\end{aligned}$$

したがって，$\tilde{f}$ は $\sqrt{\sum_{i\in[n]}\tilde{L}_i^2}$-Lipschitz である．$\tilde{f}_i$ の定義（式 (L2) を参照）から，次の式が成り立つ．

$$\tilde{L}_i = \sqrt{(1-\alpha_2)^2 + \alpha_2^2\sum_{j\neq i}(w_{ij}L_{\text{fuse}}^{i,j})^2}. \quad (7)$$

f_θ はこれら 3 つのブロックの合成写像であるため，その Lipschitz 定数はそれらの積によって決定される．

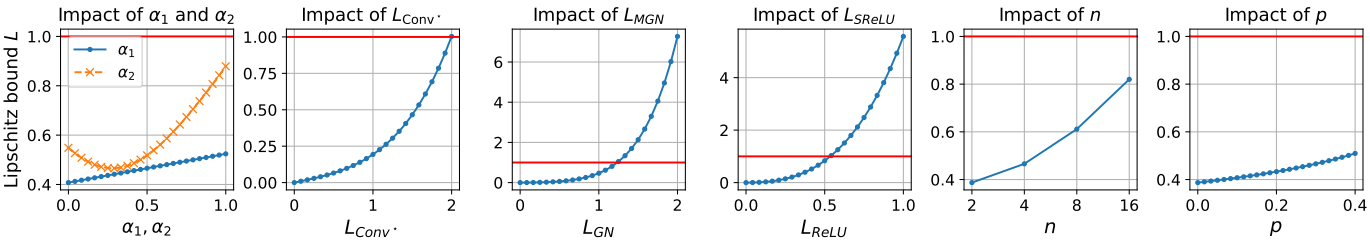


図2　$f_{\boldsymbol{\theta}}$ を構成する各ハイパーパラメータが Lipschitz 定数 L に及ぼす影響のグラフ．各グラフは，他のパラメータを一定に保ちつつ，1 つのパラメータのみを変化させてプロットしたものである．固定パラメータは $\alpha_1 = 0.5, \alpha_2 = 0.3, L_{\mathrm{Conv}^\star} = 1.5, L_{\mathrm{MGN}} = 1.0, L_{\mathrm{SReLU}} = 0.4, n = 4,$ および $p = 0.3$ に設定されている．固定パラメータの値を変更するとプロットも変化することに注意されたい．

$$L = \hat{L}\sqrt{\sum_{i\in[n]} \tilde{L}_i^2}\bar{L}, \tag{8}$$

ここで，$\hat{L}, \bar{L}, \tilde{L}_i$ はそれぞれ式 (5), 式 (6), 式 (7) で定義される．

c）　各ハイパーパラメータの感度

フォワードパスの Lipschitz 定数は，式 (8) で与えられるように，$f_{\boldsymbol{\theta}}$ を構成する各演算の Lipschitz 定数によって決定され，各演算の Lipschitz 定数はハイパーパラメータである．具体的には，2 つの凸結合パラメータ α_1 と α_2，畳み込み演算におけるノルムの上限 $L_{\mathrm{Conv}^\star} = c$，正規化演算におけるアフィンパラメータ $L_{\mathrm{MGN}} = \bar{\gamma}$，活性化関数の傾き $L_{\mathrm{SReLU}} = a$，解像度の数 n，ドロップアウト率 p によって決定される．図 2 は，各パラメータがフォワードパス全体における Lipschitz 定数 L に与える影響の大きさを示している．影響の大きさは L に現れる各演算の Lipschitz 定数の次数に依存し，特に L_{SReLU}，L_{MGN}，$L_{\mathrm{Conv}^\star}$ が L に強く影響を与えることがわかる．さらに，特にこれらの値が大きい場合，小さな凸結合パラメータ α_1, α_2 が L を小さく保つ上で重要な役割を果たす．例えば，$L_{\mathrm{SReLU}} = 0.35, L_{\mathrm{Conv}^\star} = 2.0, L_{\mathrm{MGN}} = 1.0, \alpha_1 = 0.5, \alpha_2 = 0.3, n = 4, p = 0.3$ の場合，フォワードパスの Lipschitz 定数は $L = 0.86$ となり，縮小性を満たす．

3.4　バックワードパスにおける副次的効果

我々はモデル構造を再構築し，フォワードパス（式 1）に現れる不動点写像 $f_{\boldsymbol{\theta}}$ を縮小写像とした．すなわち，Lipschitz 定数 $L = \sup_{z\in\mathcal{Z}} \|J_{f_{\boldsymbol{\theta}}}(z)\|_2$ が 1 未満となることを保証した．ここで $J_{f_{\boldsymbol{\theta}}}$ は $f_{\boldsymbol{\theta}}$ のヤコビ行列である．これによりフォワードパスにおける不動点収束が保証され，フォワードパスの高速化も期待される．一方，バックワードパスにおいても有益な効果が見られる．バックワードパス（式 4）に登場する不動点写像の Lipschitz 定数は，$\|J_{f_{\boldsymbol{\theta}}}(z^\star)\|_2$ である．ここで，$z^\star \in \mathcal{Z}$ はフォワードパスで得られた不動点である．また，$\|J_{f_{\boldsymbol{\theta}}}(z^\star)\|_2 \le \sup_{z\in\mathcal{Z}} \|J_{f_{\boldsymbol{\theta}}}(z)\|_2$ であるため，バックワードパスの Lipschitz 定数は常にフォワードパスのそれよりも小さい．したがって，フォワードパスにおける Lipschitz 定数を 1 未満に強制することは，バックワードパスにおける Lipschitz 定数も 1 未満に抑制することを意味する．これによりバックワードパスにおいても不動点収束が保証され，バックワードパスの高速化も期待される．

4.　数 値 実 験

全ての実験において，Lipschitz 定数 L は $L_{\mathrm{MGN}} = 1.0$, $L_{\mathrm{Conv}^\star} = 2.0$, $\alpha_1 = 0.5$, $\alpha_2 = 0.3$, $n = 4$, $p = 0.3$ を固定し，$L_{\mathrm{SReLU}} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ を変化させた．具体的には，$L_{\mathrm{SReLU}} = \{0.1, 0.4, 1.0\}$ の場合，それぞれ $L = \{0.03, 1.0, 14.43\}$ が得られる．

a）　不動点収束の比較

Lipschitz MDEQ では，フォワードパスにおける Lipschitz 定数を 1 未満に強制することで，一意の不動点の存在が保証される．図 3 は CIFAR-10 分類における不動点収束を示している．不動点反復回数はフォワードパスで 18 回，バックワードパスで 20 回に固定した．不動点収束の評価指標は相対残差 $\|f_{\boldsymbol{\theta}}(z;\boldsymbol{x}) - z\|/\|f_{\boldsymbol{\theta}}(z;\boldsymbol{x})\|$ である．訓練全体で Lipschitz MDEQ は圧倒的な不動点収束を達成し，理論的に収束が保証されない場合（すなわち $L \ge 1$）でも既存手法よりも優れた収束性を示した．

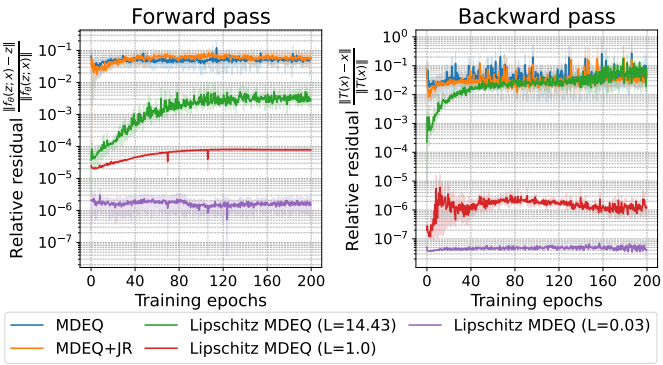


図3　訓練時のフォーワードパスとバックワードパスにおける不動点収束の比較

b）　画像分類タスクにおける高速化

表 1 は，CIFAR-10 分類における MDEQ と複数の高速化手法，および我々の Lipschitz MDEQ の結果をまとめたものである．全ての不動点反復の評価指標には相対残差 $\|f_{\boldsymbol{\theta}}(z;\boldsymbol{x}) - z\|/\|f_{\boldsymbol{\theta}}(z;\boldsymbol{x})\|$ を使用し，閾値は 1e-3 とした．最大反復回数はフォワードパスで 18，バックワードパスで 20 に設定した．訓練時およびテスト時のフォワードパス，ならびに訓練時のバックワード

表 1　CIFAR-10 における MDEQ の各種高速化手法の比較．精度（Acc），訓練と推論を合わせた全ての計算時間の高速化度合い（Speed-Up），不動点反復の平均回数（NFE），およびフォワード・バックワードパスの実行時間を報告する．各列の最良の結果は**太字**で強調表示している．JR: ヤコビアン正則化 [6]; PG: ファントム勾配 [14]; JFB: Jacobian Free Backpropagation [12]．†：最大反復回数が 7 および 8 に設定されているのは JR のみ．

Model and method	Size	Speed-up	Acc.	Train forward		Train backward		Test forward	
				NFEs	Time (ms)	NFEs	Time (ms)	NFEs	Time (ms)
MDEQ	10M	1.0×	93.23	17.9	217	20	291	18.0	63.9
MDEQ + JR	10M	2.22×	91.86	7.0†	74.8	8.0†	97.8	7.0†	39.2
MDEQ + PG	10M	1.41×	92.92	15.6	220	–	87.6	18.0	63.9
MDEQ + JFB	10M	2.84×	88.13	12.3	142	–	**30.6**	11.3	44.3
Lipschitz MDEQ ($L = 14.43$)	10M	1.28×	**93.73**	16.0	187	19.9	193	15.5	54.1
Lipschitz MDEQ ($L = 1.0$)	10M	3.33×	91.44	5.4	75.7	2.9	52.9	4.8	22.6
Lipschitz MDEQ ($L = 0.03$)	10M	**4.75**×	90.47	**2.0**	**44.0**	**2.0**	42.1	**2.0**	**13.3**
Lipschitz MDEQ ($L = 14.43$) + JFB	10M	3.66×	88.66	7.6	90.18	-	24.63	7.11	26.51
Lipschitz MDEQ ($L = 1.0$) + JFB	10M	4.98×	87.53	4.2	55.34	-	24.64	4.0	17.57
Lipschitz MDEQ ($L = 0.03$) + JFB	10M	6.43×	87.26	2.0	34.98	-	24.67	2.0	11.3

パスのいずれにおいても，Lipschitz MDEQ は既存手法よりも高速な不動点収束を達成し，実行時間を短縮できた．ただし，Lipschitz 定数の制約が厳しくなるほどテスト精度が低下する傾向が見られた．さらに理論上，Lipschitz 定数が小さいほど JFB の効果は高まる．そこで，表 1 では JFB と Lipschitz MDEQ の併用結果も報告する．この組み合わせによりさらなる速度向上が実現されるが，JFB なしの Lipschitz MDEQ と比較すると精度の悪化がより顕著である．

c）　精度と実行時間のトレードオフ

図 4 は，Lipschitz 定数を変化させた場合の精度と速度をプロットしたものである．Lipschitz 定数を制御することで精度と速度のトレードオフを管理できることが分かる．さらに，理論保証が得られない場合（$L \geq 1$）でも，既存手法と同等以上の十分な高速化を達成した．加えて，理論保証なしに Lipschitz 定数を大きく設定することで，Lipschitz MDEQ は MDEQ と同等以上のテスト精度を達成するとともに，わずかな速度向上も実現している．

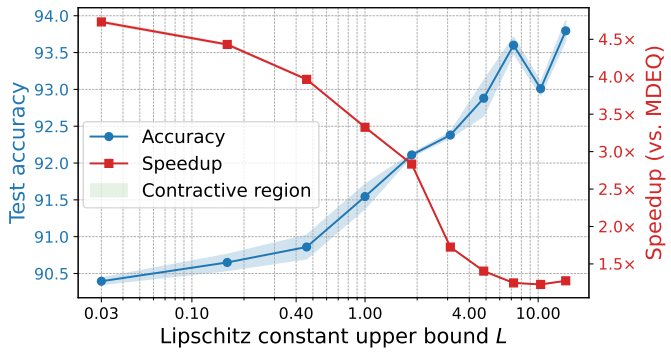


図 4　Lipschitz MDEQ における精度と速度のトレードオフ．収束が保証される領域 $L < 1$ は緑色で塗りつぶされている．

5. ま と め

本稿では，MDEQ の改良アーキテクチャである Lipschitz MDEQ を紹介した．このモデルでは，フォワードパスにおける不動点写像の Lipschitz 定数をユーザーが自由に決定可能であり，写像を縮小写像にすることで不動点収束を保証する．これによりフォワードパスにおける不動点収束が劇的に改善される．さらに，バックワードパスにおける不動点写像の Lipschitz 定数はフォワードパスよりも小さいため，不動点収束の高速化効果はバックワードにも波及する．結果として，提案アーキテクチャは精度がわずかに低下する代わりに，既存モデルや高速化手法と比較して訓練と推論に必要な時間を劇的に短縮できることを実証した．

文　献

[1] Ramy E. Ali, Jinhyun So, and A. Salman Avestimehr. On polynomial approximations for privacy-preserving and verifiable ReLU networks. In *Neural Information Processing Systems Workshop: Privacy Preserving Machine Learning*, 2020.

[2] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, pages 688–699, 2019.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. In *International Conference on Representation Learning*, 2019.

[5] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 33, pages 5238–5250, 2020.

[6] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing equilibrium models by jacobian regularization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 554–565, 2021.

[7] Stefan Banach. Sur les opérations dans les ensembles abstraits et

leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

[8] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *Proceeedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194, 2020.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS 2014 Deep Learning and Representation Learning Workshop*, volume `https://arxiv.org/abs/1412.3555`, 2014.

[10] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Representation Learning*, 2019.

[11] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs all we needed? *arXiv*, `https://arxiv.org/abs/2410.01201`, 2024.

[12] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley J. Osher, and Wotao Yin. JFB: jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6648–6656, 2022.

[13] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 1019–1027, 2016.

[14] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *Advances in Neural Information Processing Systems*, volume 34, pages 24247–24260, 2021.

[15] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the Conference on Language Modeling*, 2024.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[17] Steven G. Krantz and Harold R. Parks. *The Implicit Function Theorem: History, Theory, and Applications*. Birkhäuser Boston, 2012.

[18] Erwin Kreyszig. *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.

[19] Bac Nguyen and Lukas Mauch. Efficient training of deep equilibrium models. In *Hardware Aware Efficient Training ICML workshop*, 2022.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035, 2019.

[21] Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Advances in Neural Information Processing Systems*, volume 36, pages 33202–33221, 2023.

[22] Zaccharie Ramzi, Florian Mannel, Shaojie Bai, Jean-Luc Starck, Philippe Ciuciu, and Thomas Moreau. SHINE: sharing the inverse estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Representation Learning*, 2022.

[23] Naoki Sato and Hideaki Iiduka. Lipschitz multiscale deep equilibrium models: A theoretically guaranteed and accelerated approach. In *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300, 2026.

[24] Pietro Sittoni and Francesco Tudisco. Subhomogeneous deep equilibrium models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 45794–45812, 2024.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[26] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. In *ICML 2015 Deep Learning workshop*, volume `https://arxiv.org/abs/1505.00387`, 2015.

[27] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2377–2385, 2015.

[28] Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 10718–10728, 2020.

[29] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.