Banachの不動点定理に基づいた深層平衡モデルの高速化

情報論的学習理論と機械学習研究会(IBISML)
3/25(水) 11:20〜11:40

明治大学　佐藤尚樹
明治大学　飯塚秀明

準備：不動点と縮小写像

▷$T(z) = z$を満たすzを写像Tの**不動点**(fixed point)という．

ex.) $T(z) = 2z$の不動点は，$z = 0$．

▷写像$T\colon \mathbb{R}^d \to \mathbb{R}^d$が**Lipschitz連続写像**であるとは，

$$\exists L > 0, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d : \|T(\boldsymbol{x}) - T(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

が成り立つことをいう．Lを**Lipschitz定数**という．

▷写像$T\colon \mathbb{R}^d \to \mathbb{R}^d$が**縮小写像**であるとは，

$$\exists L \in (0, 1), \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d : \|T(\boldsymbol{x}) - T(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

が成り立つことをいう．

ex.) $T(z) = \frac{1}{2}z$は縮小写像．　$\because \|T(\boldsymbol{x}) - T(\boldsymbol{y})\| = \left\|\frac{1}{2}\boldsymbol{x} - \frac{1}{2}\boldsymbol{y}\right\| = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|$ □

▷Banachの不動点定理によると，

“写像Tが縮小写像であるならば，Tは**ただ一つの不動点を持つ**”．

背景：DNNの挙動 - forward pass


入力画像1

$\begin{pmatrix}1\\0\\0\end{pmatrix}$ 正解ラベル1


入力画像2

$\begin{pmatrix}0\\1\\0\end{pmatrix}$ 正解ラベル2


入力画像3

$\begin{pmatrix}0\\0\\1\end{pmatrix}$ 正解ラベル3

・
・
・

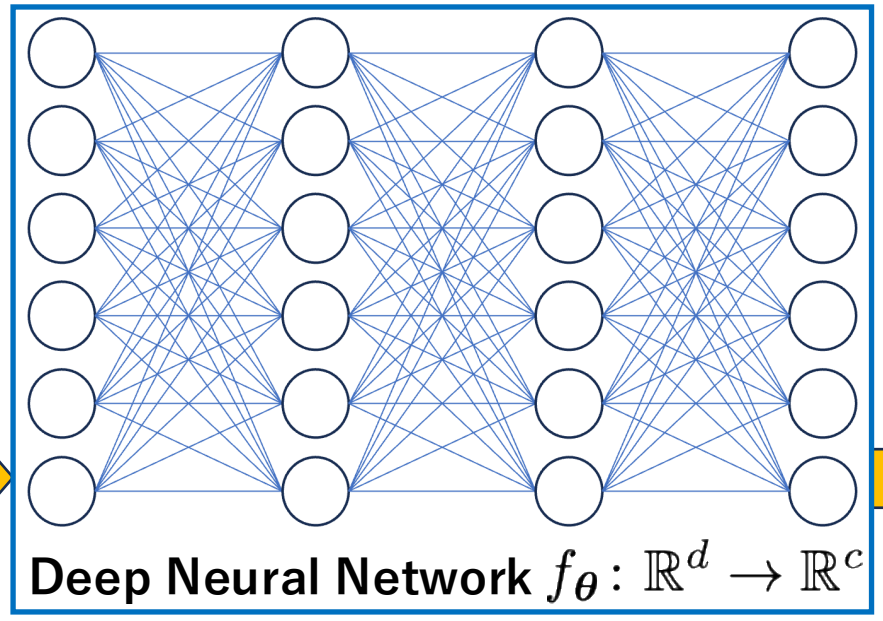
データセットN組
（N=数千～数百万）

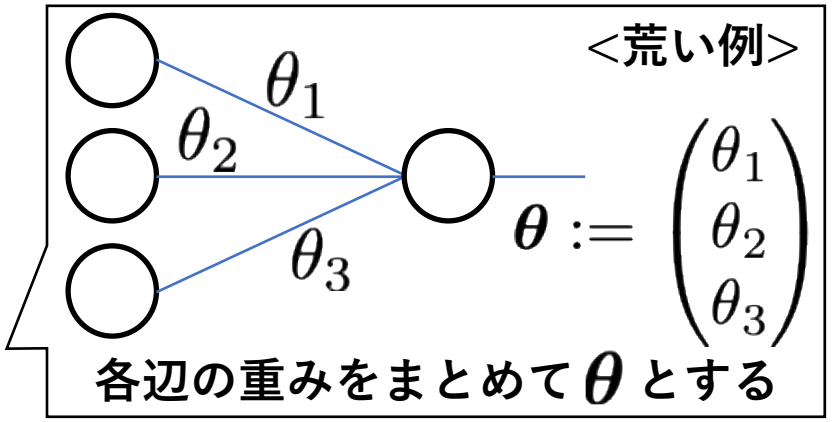
Step 1. 訓練データを用意する

Step 2. モデルを用意して出力を得る


入力画像1

$\boldsymbol{x}_1 \in \mathbb{R}^d$





$\begin{pmatrix}0.5\\0.3\\0.2\end{pmatrix}$
←画像1が“猫”である確率
←画像1が“狐”である確率
←画像1が“犬”である確率

モデルの出力1
$f_{\boldsymbol{\theta}}(\boldsymbol{x}_1) \in \mathbb{R}^c$

Step 3. 経験損失関数を計算する

$\begin{pmatrix}0.5\\0.3\\0.2\end{pmatrix}$
モデルの出力1
$f_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$

$\begin{pmatrix}1\\0\\0\end{pmatrix}$
正解ラベル1
$\boldsymbol{y}_1$

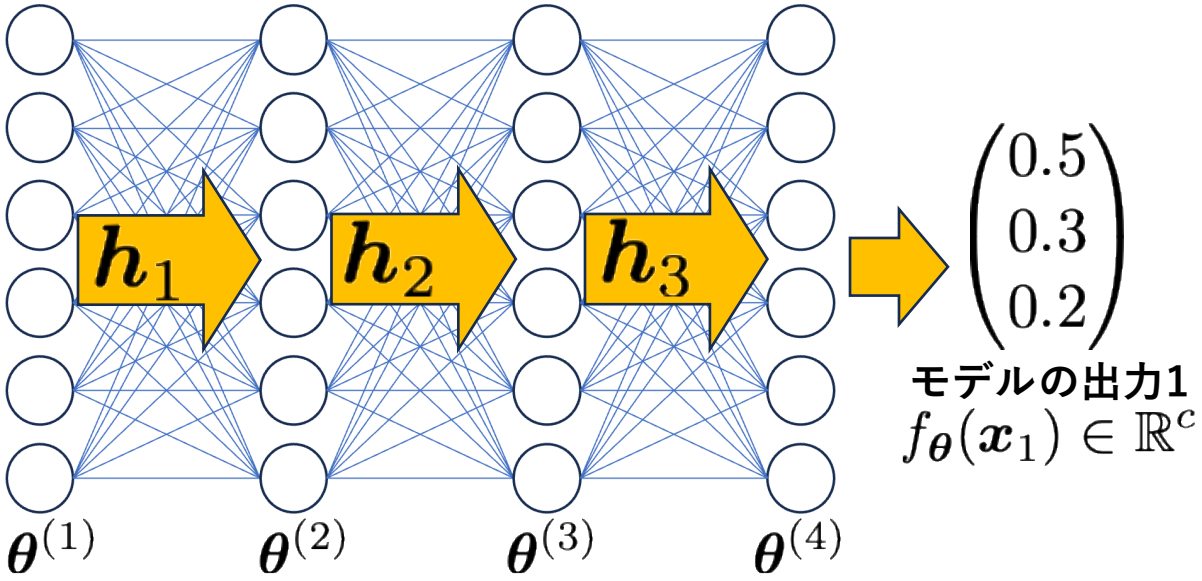
▷最小二乗誤差 $\ell(\boldsymbol{\theta}, \boldsymbol{x}) := \dfrac{1}{N}\sum_{i=1}^{N} \|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i\|^2$

▷クロスエントロピー誤差

$$\ell(\boldsymbol{\theta}, \boldsymbol{x}) := -\frac{1}{N}\sum_{i=1}^{N} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_{c_i}$$

背景：DNNの挙動 - backward pass

Step 4. 勾配を入手する



$$\begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

モデルの出力1

$f_\theta(x_1) \in \mathbb{R}^c$

▷経験損失関数の微分情報が必要

$$\ell(\theta, x) := \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(x_i) - y_i \|^2$$

<重要>

誤差逆伝播のためには，全ての層の出力を保存しておく必要がある

▷連鎖律を使う(誤差逆伝播法)

$$\frac{\partial \ell(\theta, x_1)}{\partial \theta^{(1)}} = \frac{\partial \ell(\theta, x_1)}{\partial f_\theta(x_1)} \cdot \frac{\partial f_\theta(x_1)}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial \theta^{(1)}}$$

Step 5. 経験損失関数の最適化(非凸最適化問題)

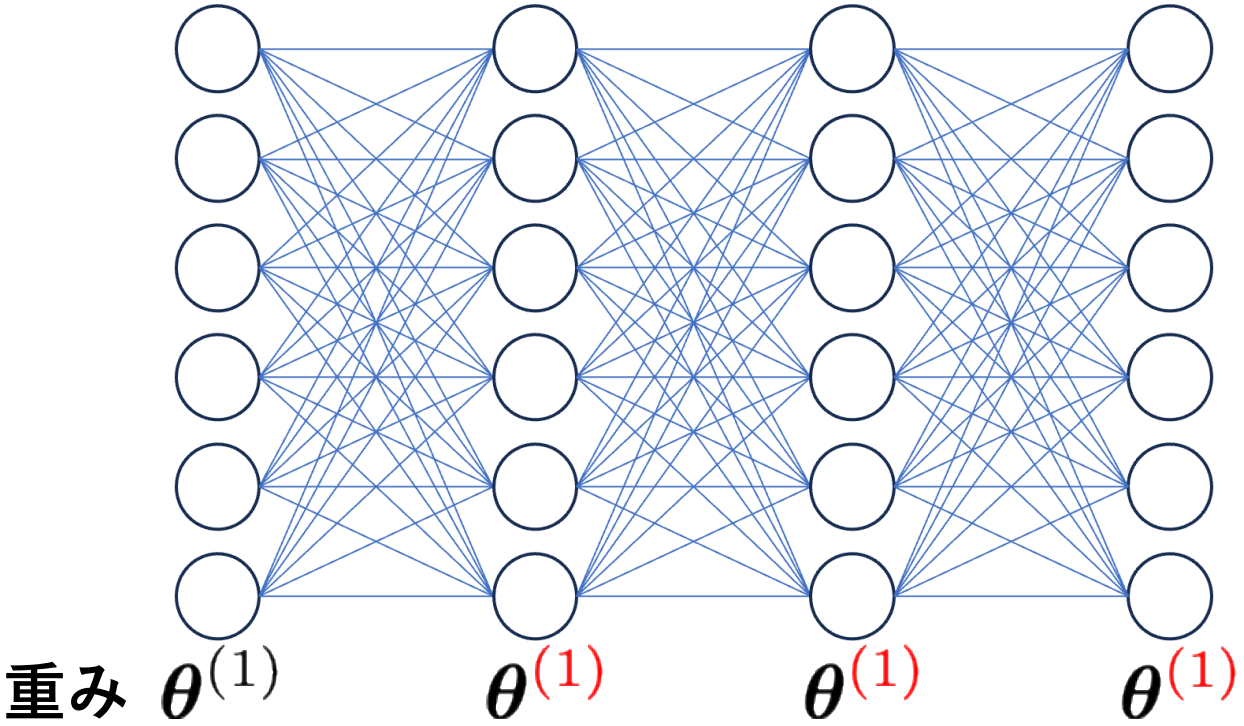
▷確率的勾配降下法(SGD)

▷Adam, AdamW等の適応的手法

$$\theta_{t+1}^{(1)} := \theta_t^{(1)} - \eta \frac{\partial \ell(\theta_t, x)}{\partial \theta_t^{(1)}}$$

背景：weight-tied models

▷“deeper is better”．無限層のnetworkを実現するためには？



▷全層で共通の重みを使うモデル

▷Trellis Network [1]

[1] S. Bai, J. Z. Kolter, and V. Koltun, “Trellis networks for sequence modeling”, ICLR2019

▷Universal Transformer [2]

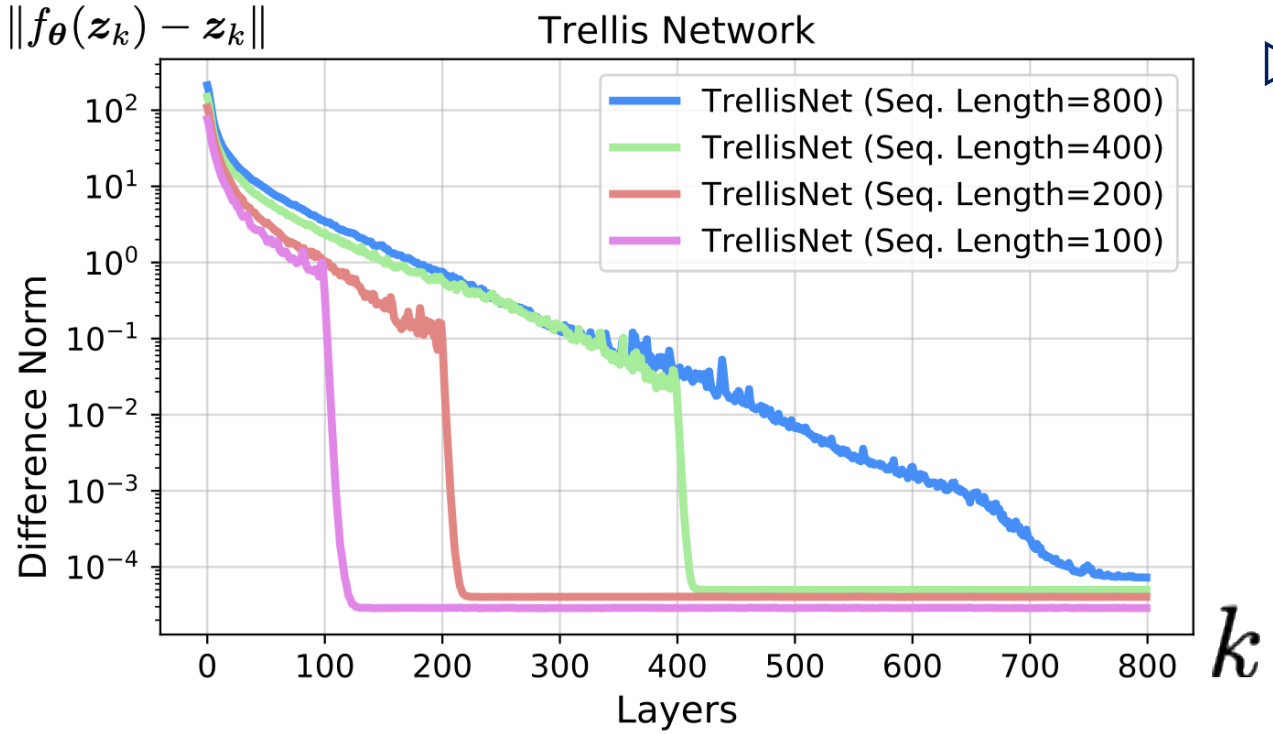
[2] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, “Universal Transformers”, ICLR2019

▷モデルサイズの大幅な削減に成功

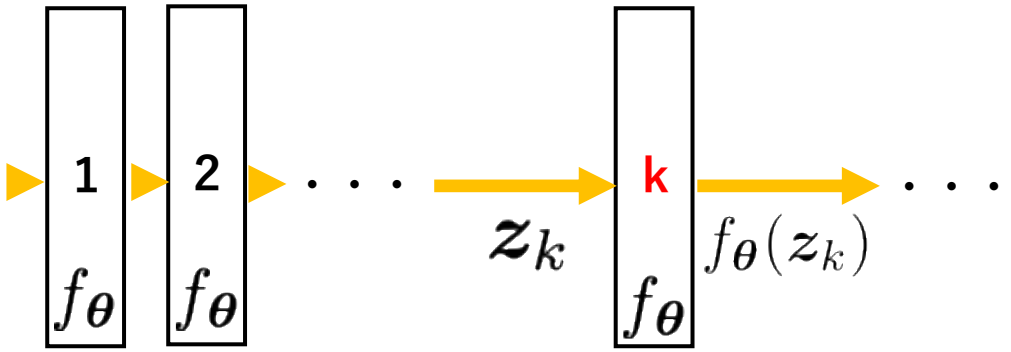
▷1層分のパラメータなのに性能は変わらなかった.

▷無限層にはできない＝誤差逆伝播の呪縛からは逃れられていない

背景：Deep Equilibrium Models(DEQ)



▷第k層の入力と出力の距離を観測した結果.



$$\therefore \|f_\theta(z_k) - z_k\| \to 0$$
$$\text{i.e., } z_k \to z^\star \text{ s.t. } z^\star = f_\theta(z^\star)$$

[3] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models", NeurIPS2019 (Figure 4)

▷層の数を大きくすると，モデルの出力は層の変換の**不動点に収束**する.

層を積み重ねて不動点を手に入れるアプローチ

↓

層の変換の不動点を直接求めるアプローチ

背景：DEQの挙動 - forward pass


入力画像1

$\begin{pmatrix}1\\0\\0\end{pmatrix}$ 正解ラベル1


入力画像2

$\begin{pmatrix}0\\1\\0\end{pmatrix}$ 正解ラベル2


入力画像3

$\begin{pmatrix}0\\0\\1\end{pmatrix}$ 正解ラベル3

・
・
・

データセットN枚
（N=数千～数百万）

Step 1. 訓練データを用意する

Step 2. モデルを用意して出力を得る


入力画像1

$\boldsymbol{x}_1 \in \mathbb{R}^d$

不動点問題

$$z = f_\theta(z; x)$$

を不動点近似法で解く．
指標 $\|f_\theta(z;x) - z\|$ が
閾値1e-3を達成するまで．
最大反復回数は10~30回．

Deep Equilibrium Model

Banachの不動点近似法

$$z_{k+1} := f_\theta(z_k; x)$$

$\begin{pmatrix}0.5\\0.3\\0.2\end{pmatrix}$ ←画像1が“猫”である確率
←画像1が“狐”である確率
←画像1が“犬”である確率

モデルの出力1
$f_\theta(\boldsymbol{x}_1) \in \mathbb{R}^c$

Step 3. 経験損失関数を計算する

$\begin{pmatrix}0.5\\0.3\\0.2\end{pmatrix}$ モデルの出力1 $f_\theta(\boldsymbol{x}_1)$

$\begin{pmatrix}1\\0\\0\end{pmatrix}$ 正解ラベル1 $\boldsymbol{y}_1$

▷最小二乗誤差 $\ell_{z^\star}(\boldsymbol{\theta}, \boldsymbol{x}) := \dfrac{1}{N}\sum_{i=1}^{N} \|f_\theta(\boldsymbol{x}_i) - \boldsymbol{y}_i\|^2$

▷クロスエントロピー誤差

$$\ell_{z^\star}(\boldsymbol{\theta}, \boldsymbol{x}) := -\frac{1}{N}\sum_{i=1}^{N} \log f_\theta(\boldsymbol{x}_i)_{c_i}$$

背景：DEQの挙動 - backward pass

Step 4. 勾配を入手する(誤差逆伝播は使わない/使えない)

陰関数の定理と連鎖律から,

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

が成り立つ．そのまま計算するのは高価なので，項Aは次の方程式を解いて求める．

$$\mathbf{x}\,(I - J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)) - \frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{z}^\star} = \mathbf{0}$$

この方程式は次の不動点問題と等価である．

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

∴DEQはforward passとbackward passで異なる2つの不動点問題を解いている！

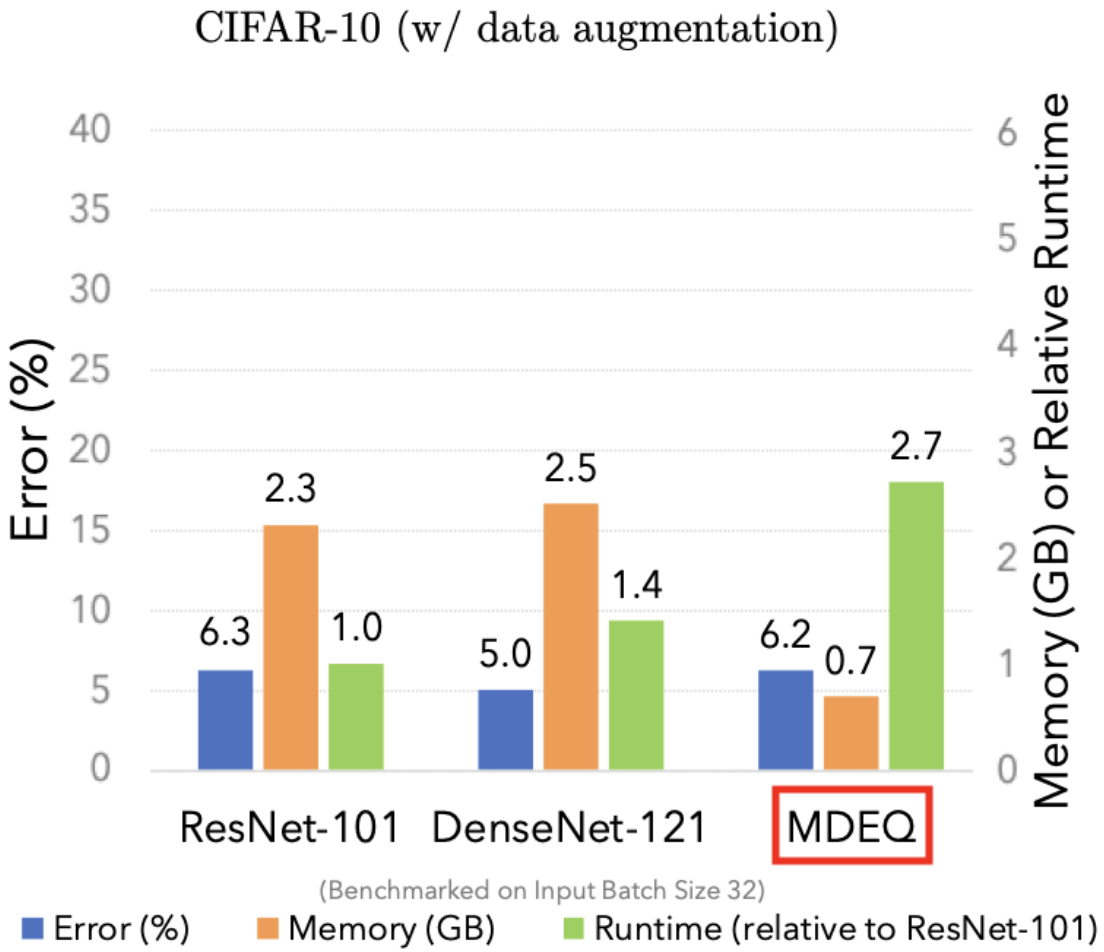
Step 5. 経験損失関数の最適化(非凸最適化問題)

▷確率的勾配降下法(SGD)

▷Adam, AdamW等の適応的手法

$$\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \eta \frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta}_t, \boldsymbol{x})}{\partial \boldsymbol{\theta}_t}$$

背景：DEQの特徴



テスト
エラー率

必要な
メモリ

実行時間

DEQは層を積み重ねることなく，無限層のnetworkを実現している．

▷Explicitな既存モデルと比較して，

高性能・省メモリ・超低速

省メモリな理由

1. 使うメモリは1層のパラメータ分だけ
2. 誤差逆伝播しないので，中間層の出力を保持しておく必要もない

超低速な理由

1. Backward passが遅い
2. 不動点反復が収束しない

[4] S. Bai, V. Koltun and J. Z. Kolter, “Multiscale Deep Equilibrium Models”, NeurIPS2020 (Figure 4(b))

研究の動機とアプローチ

▷DEQの実用化への課題：訓練も推論も超低速

DEQ / 18-layer Transformer		DEQ / 70-layer TrellisNet	
Training	Inference	Training	Inference
2.82×	1.76×	2.40×	1.64×

[3] S. Bai, J. Z. Kolter, and V. Koltun, “Deep equilibrium models”, NeurIPS2019 (Table 4)

超低速な理由

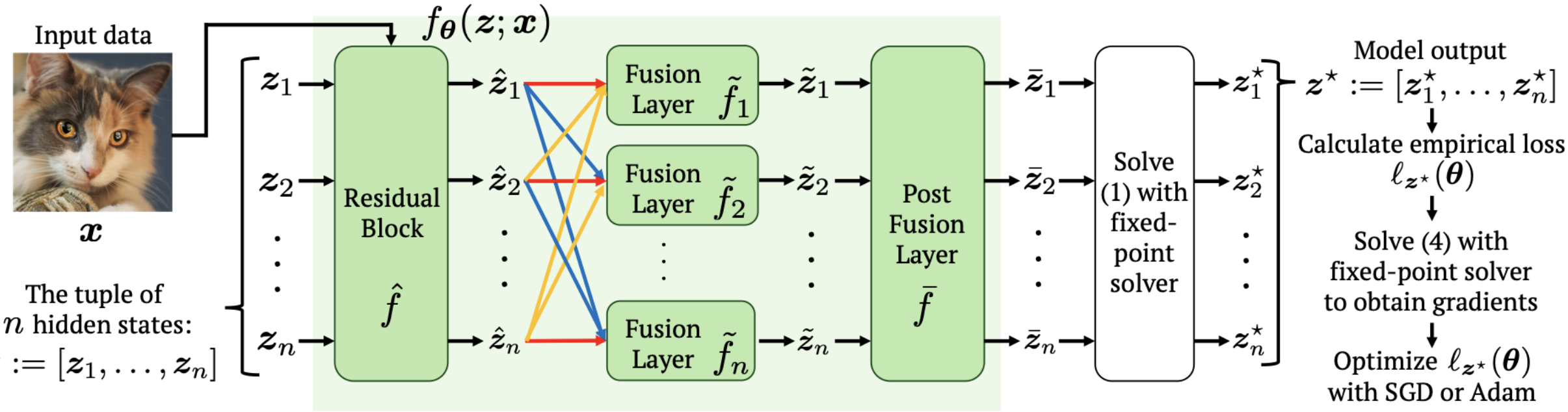
1. （誤差逆伝播と比較して）Backward passが遅い
   ⇒いくつか先行研究がある．
2. 不動点反復が収束しない <こちらの方が重大>
   ⇒経験損失のoptimizerの反復回数T_1，不動点solverの反復回数T_2に対して，計算量は$\mathcal{O}(T_1 T_2)$となる．
   ⇒forward passの不動点反復が収束するように，モデルアーキテクチャを改造すれば，高速化できるのでは？

Forward passの不動点問題

$$z = f_\theta(z; x)$$

f_θが縮小写像なら収束する

Multiscale DEQ(MDEQ)



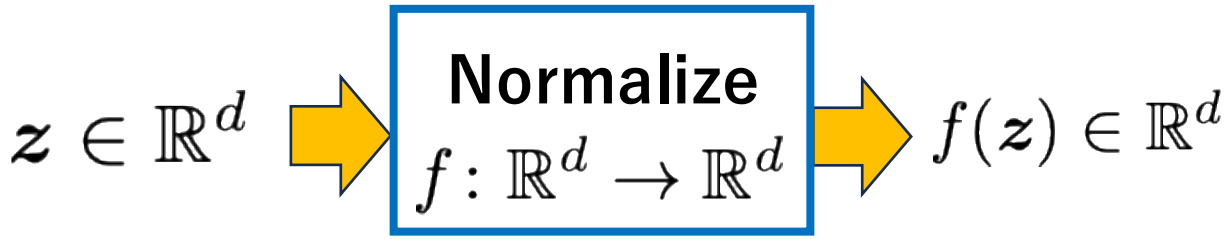
MDEQ Architecture [Bai et al., 2020]

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

$$f_\theta := \bar{f} \circ \tilde{f} \circ \hat{f}$$

各演算のLipschitz定数 - 正規化

▷スケールを統一し，学習を安定させるために多用される．



Group Normalization

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

Mean-Only Group Normalization

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

ただし，$\mu_{\boldsymbol{z}} := \frac{1}{d}\sum_{i\in[d]} z_i$, $\sigma_{\boldsymbol{z}}^2 := \frac{1}{d}\sum_{i\in[d]}(z_i - \mu_{\boldsymbol{z}})^2$ で，$\gamma, \beta \in \mathbb{R}$ は学習可能なパラメータ．

Lipschitz定数は，

$$L_{\mathrm{GN}} \leq \frac{|\gamma|}{\sqrt{\epsilon}}$$

Lipschitz定数は，

$$L_{\mathrm{MGN}} \leq |\gamma|$$

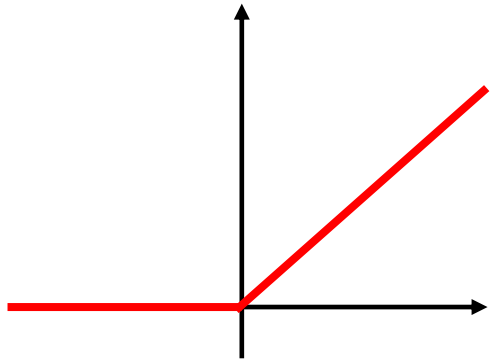
各演算のLipschitz定数 - 活性化関数

▷非線形性を加えて，表現力を向上させるためにある．

$\boldsymbol{z} \in \mathbb{R}^d$ → **Activation** $f : \mathbb{R}^d \to \mathbb{R}^d$ → $f(\boldsymbol{z}) \in \mathbb{R}^d$

ReLU

$$\mathrm{ReLU}(\boldsymbol{z})_i := \max\{0, z_i\}$$



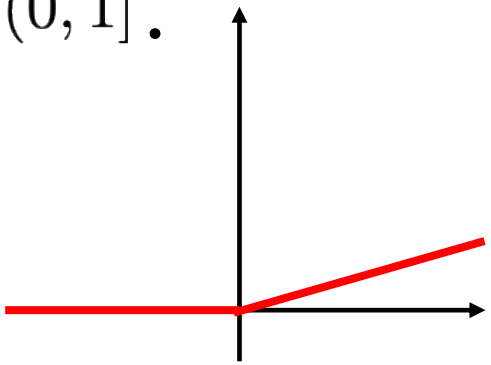
Lipschitz定数は，

$$L_{\mathrm{ReLU}} = 1$$

Scaled-ReLU

$$\mathrm{SReLU}(\boldsymbol{z})_i := \max\{0, az_i\}$$

ただし，$a \in (0, 1]$．

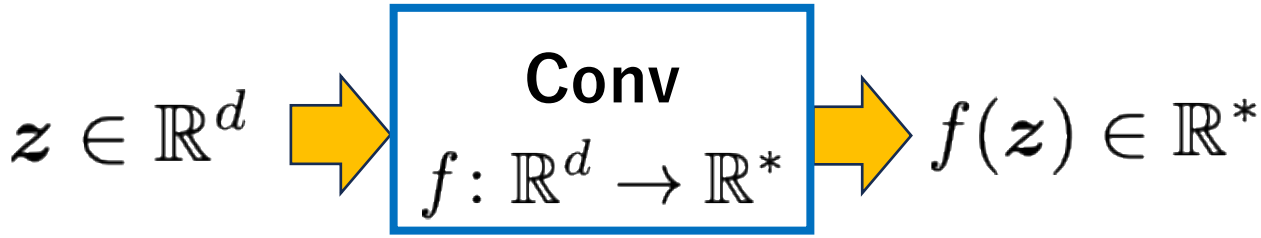


Lipschitz定数は，

$$L_{\mathrm{SReLU}} = a$$

各演算のLipschitz定数 - 畳み込み

▷画像データから効率的に特徴を抽出するためにある.



Convolution

$$\mathrm{Conv}(z) = Wz + b$$

ただし, $W \in \mathbb{R}^{*\times d}$ は畳み込みカーネルで, $b \in \mathbb{R}^*$ はバイアス項.

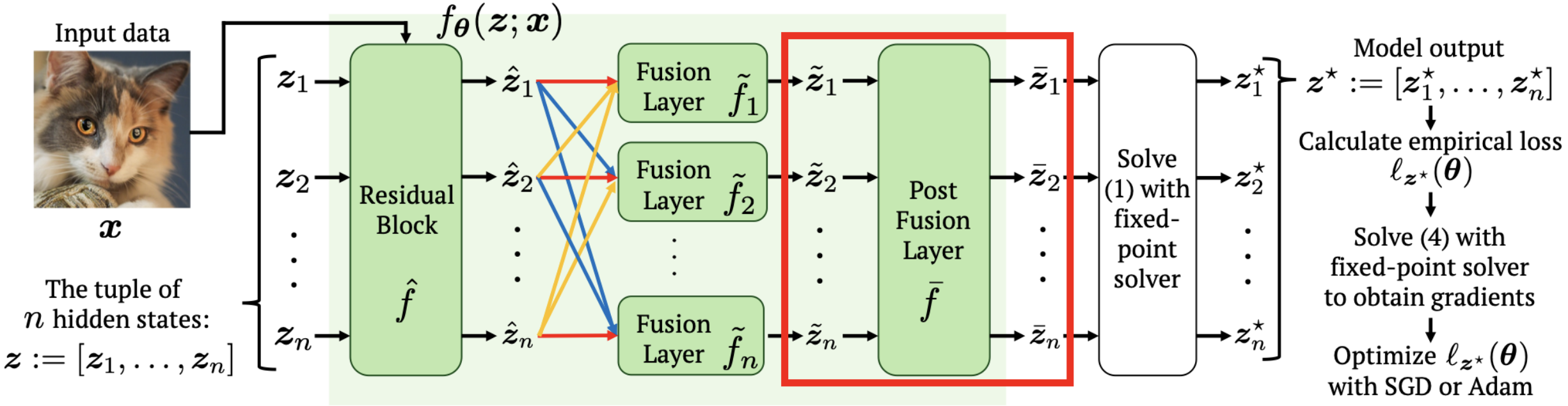
Lipschitz定数は,

$$L_{\mathrm{Conv}} = \|W\|_2$$

したがって，Wのノルムをパラメータcで制約すれば,

$$L_{\mathrm{Conv}^\star} = c$$

各演算のLipschitz定数 - PostFusion



<MDEQの式>

$$\bar{z}_i = \bar{f}_i(\tilde{z}_i) := \mathrm{GN}(\mathrm{Conv}(\mathrm{ReLU}(\tilde{z}_i)))$$

<改良後の式>

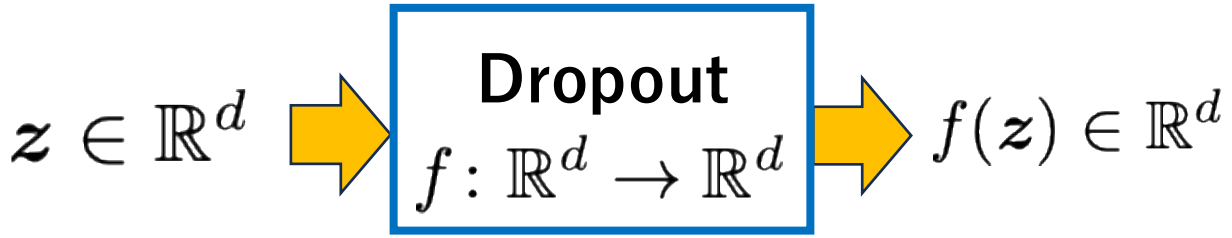
$$\bar{z}_i = \bar{f}_i(\tilde{z}_i) := \mathrm{MGN}(\mathrm{Conv}^\star(\mathrm{SReLU}(\tilde{z}_i)))$$

Lipschitz定数は,

$$\bar{L} = L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} L_{\mathrm{SReLU}}$$

各演算のLipschitz定数 - dropout

▷過学習を防ぐために使われる.



Variational Dropout [5]

$$\mathrm{Drop}(\boldsymbol{z}) := \frac{1}{1-p}\boldsymbol{m} \odot \boldsymbol{z}$$

ただし，$p \in (0, 1)$ はdropout rateで，0.05や0.3などが使われる.

$\boldsymbol{m} \sim \mathrm{Bernoulli}(p)$ はマスクで，各要素は確率pで0，確率1-pで1.

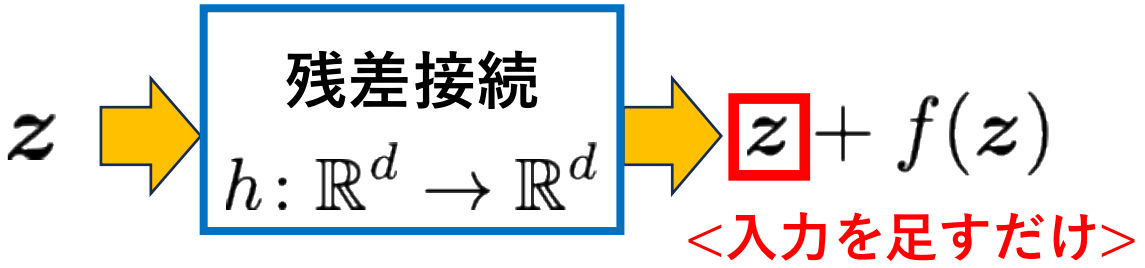
Lipschitz定数は,

$$L_{\mathrm{Drop}} = \frac{1}{1-p}$$

[5] Y. Gal and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks”, NeurIPS2016

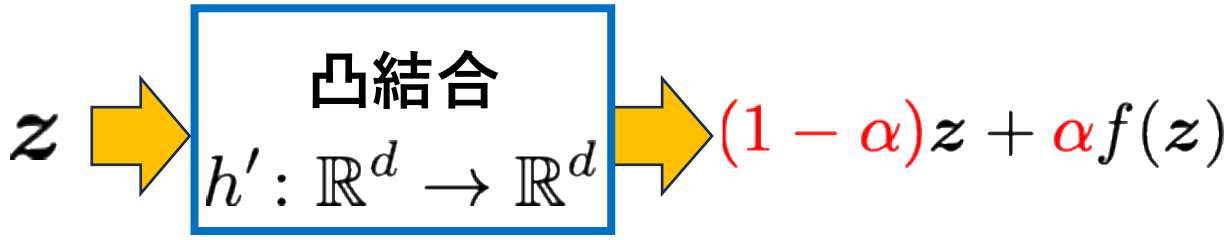
各演算のLipschitz定数 - 残差接続

▷勾配の消失を防ぎ，層を積んでも性能が悪化しないようにするための革新的なテクニック[6].



Lipschitz定数は，

$$L_h = 1 + L_f$$



ただし，$\alpha \in (0, 1)$.

Lipschitz定数は，

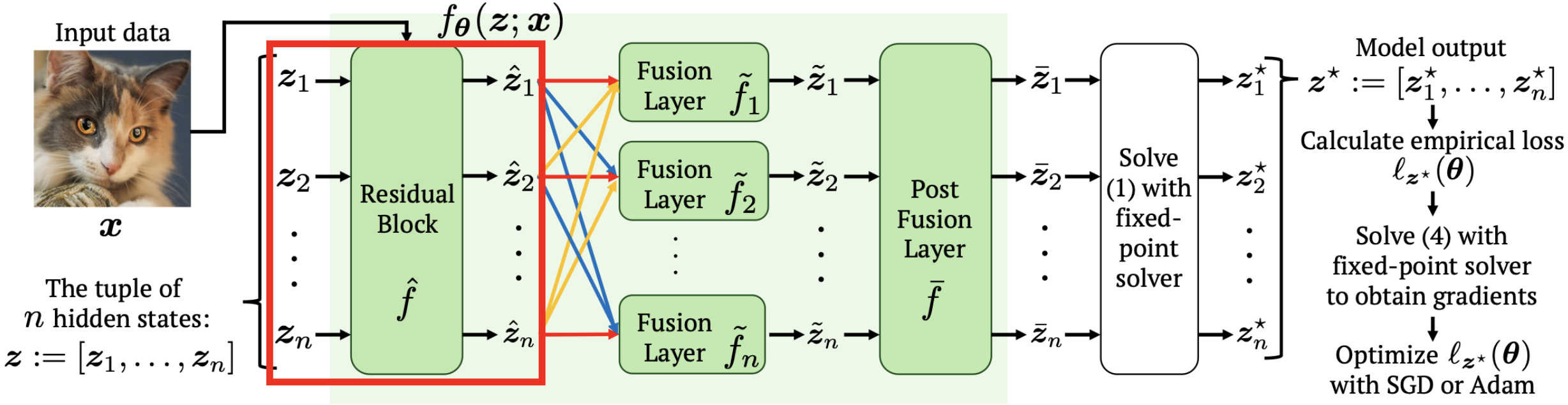
$$L_{h'} = (1 - \alpha) + \alpha L_f$$

例えば，α =0.5のとき，

$$L_{h'} = 0.5 + 0.5 L_f$$

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, CVPR2016

各演算のLipschitz定数 - Residual Block



<MDEQの式>

$$\hat{z}_i = \hat{f}_i(z_i) := \mathrm{GN}(\mathrm{ReLU}(z_i + \mathrm{GN}(\mathrm{Drop}(\mathrm{Conv}(\mathrm{ReLU}(\mathrm{GN}(\mathrm{Conv}(z_i)))))))))$$

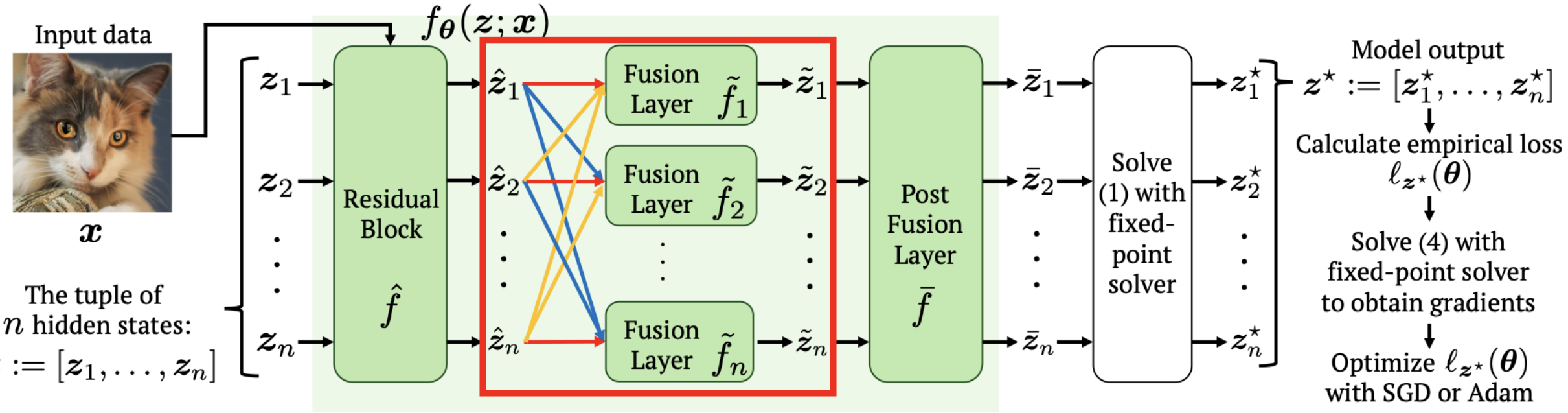
<改良後の式>

$$\hat{z}_i = \hat{f}_i(z_i) := \mathrm{MGN}(\mathrm{SReLU}((1-\alpha_1)z_i + \alpha_1 \mathrm{MGN}(\mathrm{Drop}(\mathrm{Conv}^\star(\mathrm{SReLU}(\mathrm{MGN}(\mathrm{Conv}^\star(z_i)))))))))$$

Lipschitz定数は,

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

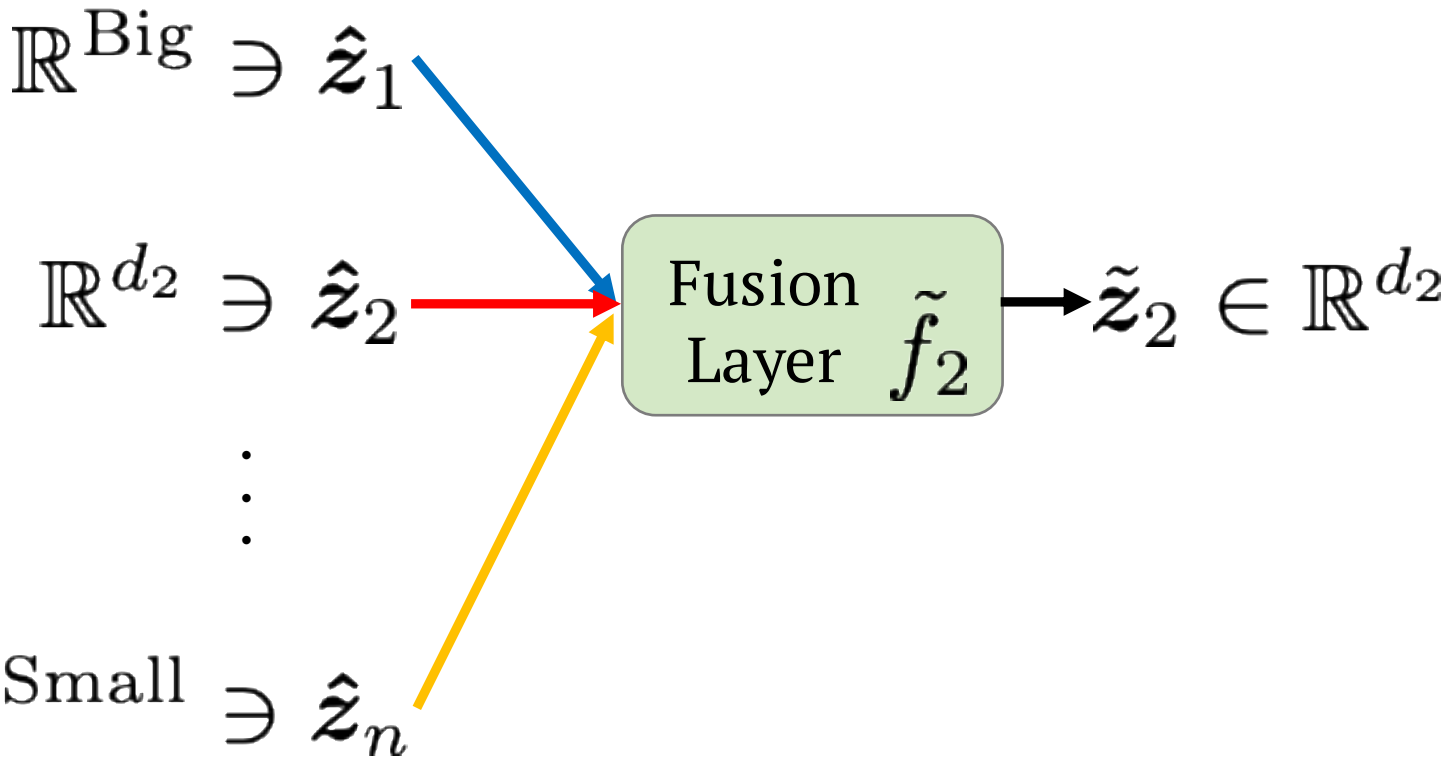
各演算のLipschitz定数 - fusion layer



MDEQ Architecture [Bai et al., 2020]

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

各演算のLipschitz定数 - fusion layer



$$\tilde{z}_i = \tilde{f}_i(\hat{z}_i) := \hat{z}_i + \sum_{j=1}^{i-1} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} \mathrm{U}_{i,j}(\hat{z}_i)$$

そのまま　downsample　upsample

各演算のLipschitz定数 - downsample

▷大きい次元から小さい次元へ圧縮する演算.

$$h_1(\boldsymbol{z}) := \mathrm{GN}(\mathrm{Conv}(\boldsymbol{z})),\ h_2(\boldsymbol{z}) := \mathrm{ReLU}(\mathrm{GN}(\mathrm{Conv}(\boldsymbol{z})))$$

とおくと,

$$D_{i,j} := h_1 \circ h_2^{i-j-1}$$

でかける. したがって, Lipschitz MDEQにおいて, Lipschitz定数は,

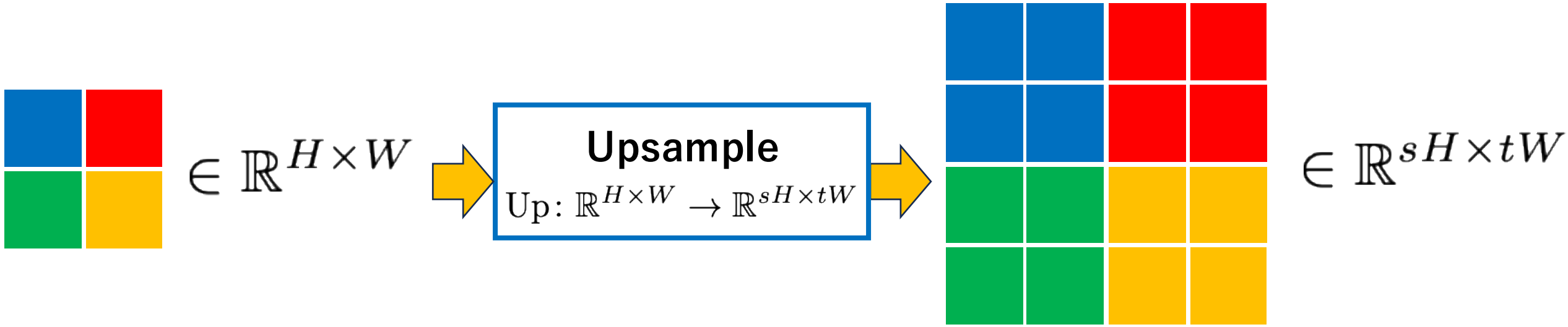
$$L_{\mathrm{Down}}^{i,j} = L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} \left(L_{\mathrm{SReLU}} L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} \right)^{i-j-1}$$

とできる.

各演算のLipschitz定数 - upsample

▷小さい次元から大きい次元へ拡張する演算.

▷最近傍補間(Nearest neighbor interpolation)



ただし，$s, t \in \mathbb{N}$ はスケールファクターで，MDEQにおいてはどちらも $2^{\text{level\_diff}}$.
例えば, $z_3 \to z_1$ のとき，$\text{level\_diff} = 2$.
したがって，Lipschitz定数は,

$$L_{\mathrm{Up}}^{i,j} = \sqrt{st} = 2^{\text{level\_diff}} = 2^{j-i}$$

各演算のLipschitz定数 - fusion layer

$$\tilde{z}_i = \tilde{f}_i(\hat{z}_i) := \hat{z}_i + \sum_{j=1}^{i-1} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} \mathrm{U}_{i,j}(\hat{z}_i)$$

<MDEQの式>

残差接続を凸結合に置き換えて,
Lipschitz定数が大きくなりすぎないようにする.

$$\tilde{z}_i = \tilde{f}_i(\hat{z}_i) := (1-\alpha_2)\hat{z}_i + \alpha_2 \left(\sum_{j=1}^{i-1} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} \mathrm{U}_{i,j}(\hat{z}_i) \right)$$

Upsample演算のLipschitz定数は最大で8.
大きい時ほど重みが小さくなるような重み付き平均をとって対応する.

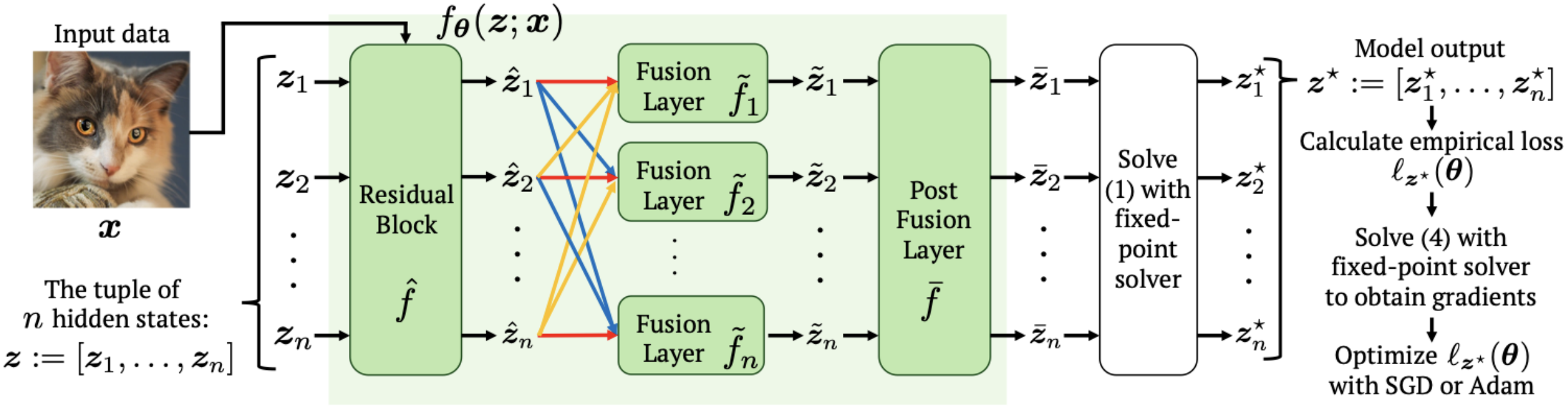
$$\tilde{z}_i = \tilde{f}_i(\hat{z}_i) := (1-\alpha_2)\hat{z}_i + \alpha_2 \left(\sum_{j=1}^{i-1} w_{ij} \mathrm{D}_{i,j}(\hat{z}_i) + \sum_{j=i+1}^{n} w_{ij} \mathrm{U}_{i,j}(\hat{z}_i) \right)$$

<改良後の式>

Lipschitz定数は,

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

Lipschitz MDEQ



Lipschitz-MDEQ Architecture (ours)

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

Lipschitz定数の導出

▷Forward passの不動点写像 f_θ は，3つのブロックの合成写像なので，f_θ のLipschitz定数は，

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

とできる．各ブロックのLipschitz定数は以下で定義されている．

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

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

$$\bar{L} = L_{\mathrm{MGN}}L_{\mathrm{Conv}^\star}L_{\mathrm{SReLU}}$$

▷例えば，

$$L_{\mathrm{SReLU}} = 0.35, L_{\mathrm{Conv}^\star} = 2.0, L_{\mathrm{MGN}} = 1.0, \alpha_1 = 0.5, \alpha_2 = 0.3, n = 4, p = 0.3$$

に設定すれば，L=0.86となって，f_θ は縮小写像となる．

Backward passの不動点収束

▷forward passに現れる写像 f_θ を縮小写像にすることができた.

▷すなわち, f_θ のLipschitz定数 $L_{\text{forward}} = \sup_{z \in \mathcal{Z}} \|J_{f_\theta}(z)\|_2$ は1未満.

▷一方, backward passで解かれる不動点問題は,

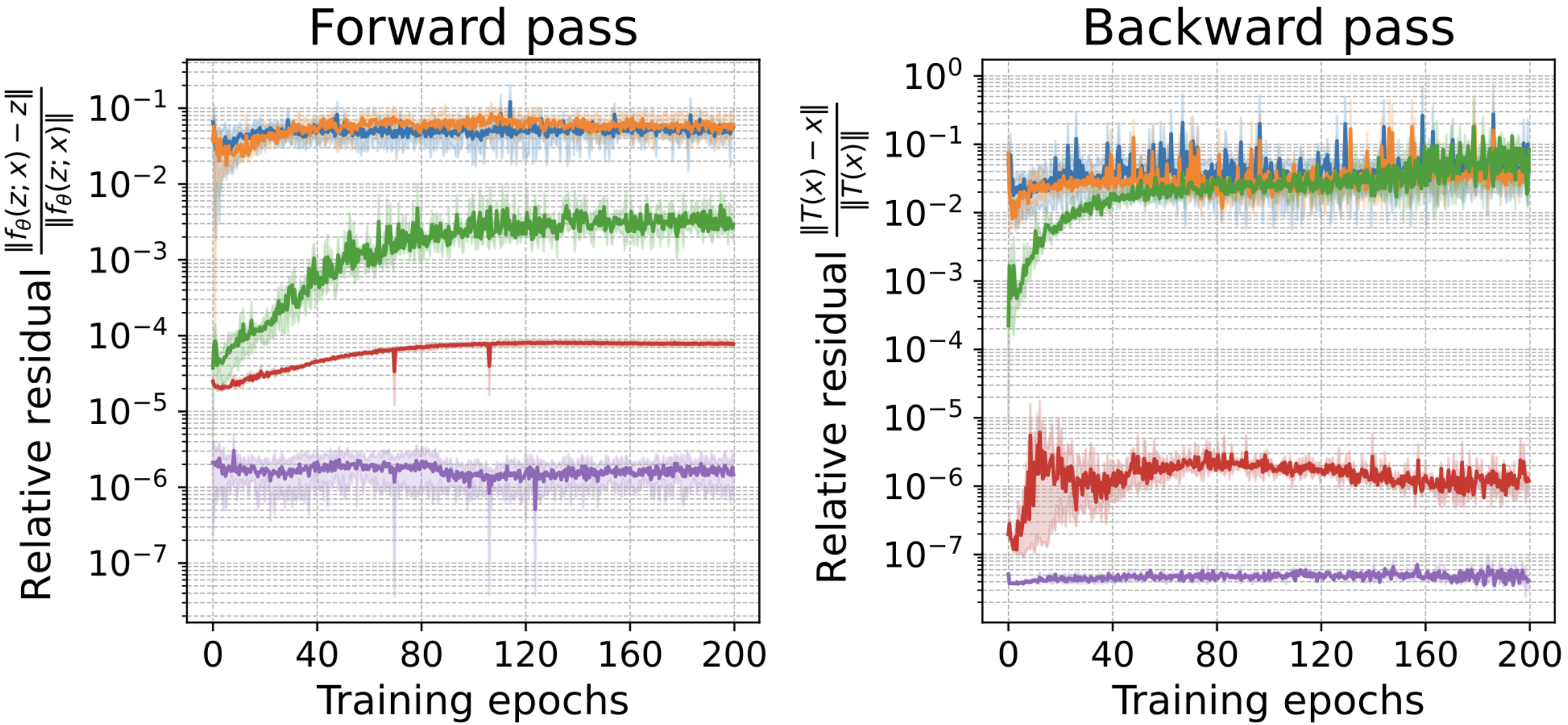
$$\mathbf{x} = T(\mathbf{x}) \text{ under } T(\mathbf{x}) := \mathbf{x} J_{f_\theta}(z^\star) + \frac{\partial \ell_{z^\star}(\theta)}{\partial z^\star}$$

写像TのLipschitz定数は, $L_{\text{back}} = \|J_{f_\theta}(z^\star)\|_2$ となる.

$$L_{\text{back}} = \|J_{f_\theta}(z^\star)\|_2 \leq \sup_{z \in \mathcal{Z}} \|J_{f_\theta}(z)\|_2 = L_{\text{forward}}$$

から, forward passの写像が縮小写像ならば, backward passの写像も縮小写像になる.

数値実験 - 不動点収束の優位性(訓練)



数値実験 - 高速化効果

Model and method	Size	Speed	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**

▷付録：Backward passを改善して高速化する先行研究

▷Phantom Gradient [Geng et al., (NeurIPS 2021)]

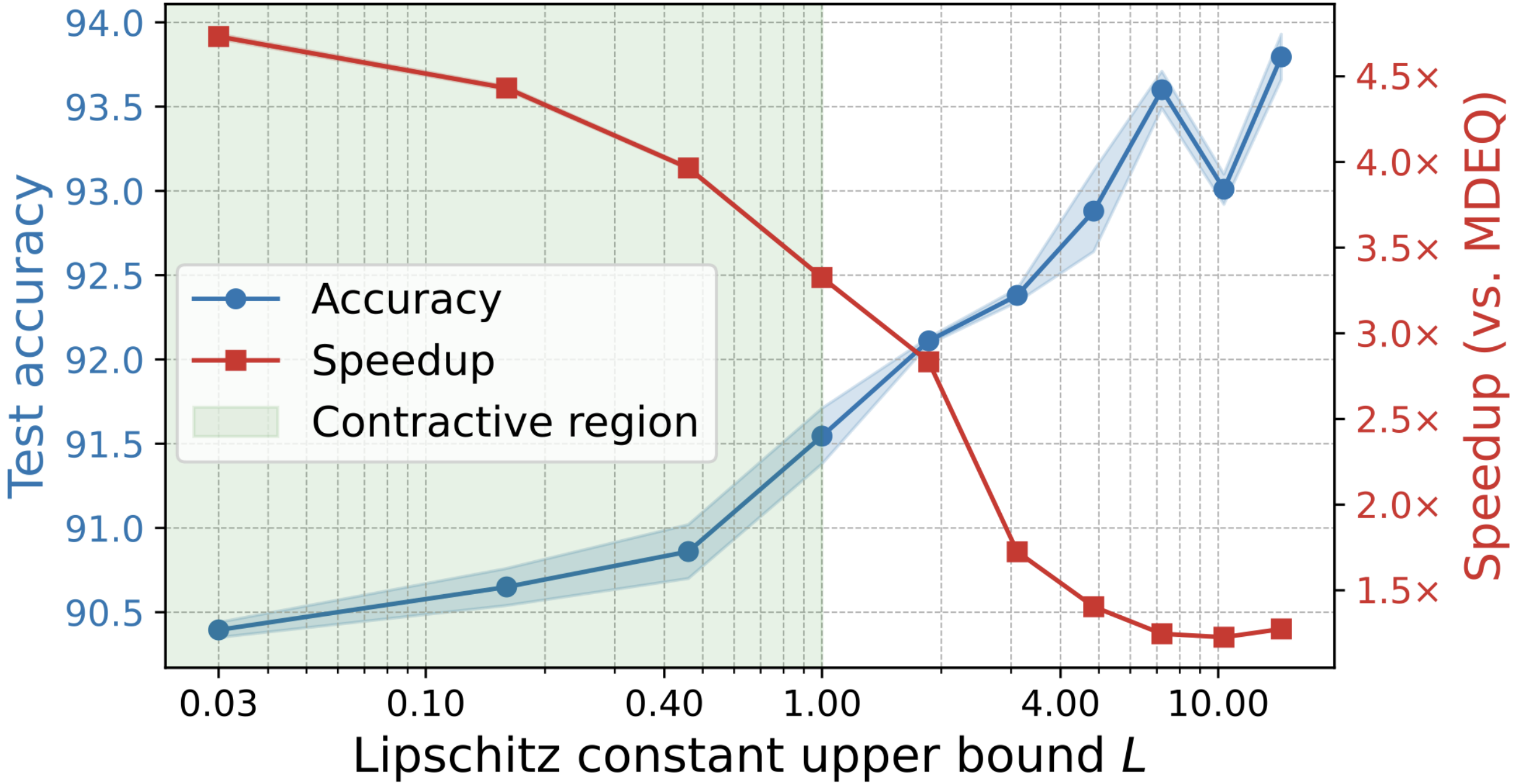
より計算しやすい補助関数 $\frac{1}{2}\|f_\theta(z^\star) - z^\star\|^2$ の勾配で代替する手法.

▷Jacobian Free Backpropagation [Fung et al., (AAAI 2022)]

$$\frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \ell_{z^\star}(\boldsymbol{\theta})}{\partial z^\star} \boxed{\left(I - J_{f_\theta}(z^\star)\right)^{-1}} \frac{\partial f_\theta(z^\star; x)}{\partial \boldsymbol{\theta}}$$

この部分を恒等写像Iで置換する大胆な近似.

数値実験 - 精度と時間のトレードオフ



まとめ

- ▷不動点を利用するモデル：Deep Equilibrium Modelsを紹介した.
  - ▷1層分のパラメータで無限層のnetworkを実現できる.
  - ▷誤差逆伝播を使わない.
  - ▷長所：高性能・省メモリ
  - ▷短所：超低速
- ▷Multiscale DEQの改良版：Lipschitz Multiscale DEQを提案した.
  - ▷アーキテクチャを構成する各演算を改良して縮小写像化するアプローチ.
  - ▷Forward passとbackward passの両方の不動点収束を理論的に保証できる.
  - ▷不動点収束が改善され，最大4.75倍の高速化を達成した.
  - ▷Lipschitz定数はハイパーパラメータの組み合わせで調整可能.
  - ▷Lipschitz定数を変化させると，精度と実行時間のトレードオフが見られる.

▷高性能・省メモリ・高速(同速) なモデルの実現へ！

付録：モデルの分類

Transformer
(NeurIPS2017)

SSM
(ICLR2022)

DEQ-Transformer
(NeurIPS2019)

Implicit SSM
(ICML2025)

DEQ
(不動点問題の解を使う)

<言語処理タスク>

HRNet
(TPAMI2020)

ResNet
(CVPR2016)

Multiscale-DEQ
(NeurIPS2020)

＋逆問題を中心に幅広い応用
(e.g. 画像の復元/ノイズ除去)

<画像処理タスク>

Neural ODE
(NeurIPS2018)

(微分方程式の解を使う)

GNN
(IEEE Trans. Neural Netw. 2009)

Implicit GNN
(NeurIPS2020)

OptNet
(ICML2017)

Differentiable optimization layers

<その他>

Explicit models
(ふつうのモデル)

Implicit models / Implicit layers
(ある最適化問題の解を出力にするモデル)

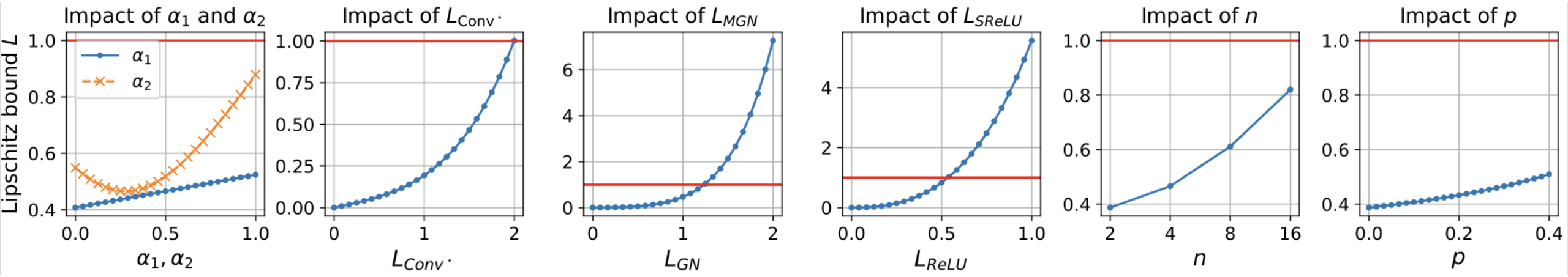
付録：Ablation study

Model and method	Size	Speed	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
Lipschitz MDEQ ($L = 1.0$)	10M	3.33×	91.44	5.4	75.7	2.9	52.9	4.8	22.6
(S1) Lipschitz MDEQ - MGN clip	10M	2.81×	91.99	6.4	90.5	4.7	65.3	5.7	25.2
(S2) Lipschitz MDEQ - MGN	10M	0.95×	92.78	18.0	225	20.0	296	18.0	72.9
(S3) Lipschitz MDEQ - Conv*	10M	**3.42**×	91.94	**3.8**	**61.4**	5.6	73.0	**3.7**	**19.0**
(S4) Lipschitz MDEQ - softmax	10M	3.41×	91.29	5.4	76.5	**2.8**	**48.2**	4.9	20.6
(S5) Lipschitz MDEQ - α_1	10M	2.72×	92.35	6.6	90.9	5.5	71.9	6.0	25.9
(S6) Lipschitz MDEQ - α_2	10M	1.64×	92.73	9.7	120	18.9	179	9.8	33.2
(S7) Lipschitz MDEQ - α_1, α_2	10M	1.37×	**93.35**	14.5	169	19.6	184	14.5	45.4

付録：各パラメータの感度

▷ f_θ のLipschitz定数Lは，各演算のLipschitz定数に依存している．

▷Lの式に登場する次数が大きいほど，大きな影響を与える．



▷各演算のLipschitz定数はユーザーが自由に決められる．

▷例えば，

$$L_{\mathrm{SReLU}} = 0.35, L_{\mathrm{Conv}^\star} = 2.0, L_{\mathrm{MGN}} = 1.0, \alpha_1 = 0.5, \alpha_2 = 0.3, n = 4, p = 0.3$$

に設定すれば，L=0.86となって，f_θ は縮小写像となる．

付録：不動点収束の優位性(推論)