図書館情報メディア研究科修士論文

深層ニューラルネットワークの適応的最適化手法における汎化性・頑健性の分析

2021年03月

201921649

本川 哲哉

深層ニューラルネットワークの適応的最適化手法における汎化性・頑健性の分析

An Investigation of Generalization and Robustness in Adaptive Optimization Methods for Deep Neural Networks

学籍番号：201921649
氏名：本川 哲哉
Motokawa Tetsuya

深層ニューラルネットワーク（Deep Neural Network，DNN）の学習において，Adam をはじめとする適応的最適化手法は確率的勾配降下法（Stochactic Gradient Descent，SGD）よりも高速に収束することで知られており，近年様々な深層学習タスクでよく利用される．その反面，特に画像分類タスクにおいて，SGD に比べて最終的に得られる DNN モデルの汎化性が確保されにくいという報告も見られる．しかしながらその原因解明はまだ進んでいない．

近年ヘッセ行列の固有値の分析をすることで，DNN モデルにおける学習ダイナミクスを解釈する研究は盛んに行われているが，この分析を用いて適応的最適化手法の特性を分析した研究は行われていない．そこで本研究では損失関数におけるヘッセ行列の固有値を分析することで，異なる最適化手法によって学習された DNN モデルの重みパラメータの汎化性，頑健性を分析した．

具体的には，いくつかの DNN モデルと画像分類タスクを用いて大きく 3 つの分析を行った．

まず学習後の DNN モデルの重みパラメータ空間において，損失関数のヘッセ行列の固有値分布を解析した．全ての重みパラメータを用いた損失関数形状の比較には，比較的小さな重みパラメータ数の DNN モデルである C3，LeNet を使用した．プラクティカルな DNN モデルとしてよく利用される ResNet-34，DenseNet-121 に対しては最終層の重みパラメータのみを用いた．その結果，適応的最適化手法の方が SGD に比べて局所的に急峻な形状に収束することを確認した．

汎化性を測る指標の 1 つとして，ヘッセ行列と勾配共分散行列を用いて計算される竹内情報量基準（Takeuchi Information Criterion，TIC）を用いて，SGD と適応的最適化手法における汎化性の比較を行った．実験の結果，今回用いた 2 つのデータセット（CIFAR-10，Fashion-MNIST），モデルについては SGD に比べて適応的最適化手法によって得られる重みパラメータに関するヘッセ行列の固有値の方が絶対値の大きな値に分布しやすく，また pseudo TIC の値が大きくなることを確認した．

さらに，Fast Gradient Sign Method (FGSM) によって生成された敵対的摂動を用いて，SGD と適応的最適化手法によって学習されたモデルの騙されやすさ（頑健性）の比較も行った．C3 on CIFAR-10 においては FGSM によって生成された敵対的摂動に対して，SGD に比べて適応的最適化手法によって学習されたモデルの方が頑健性が確保されにくいことを確認した．

以上の実験結果より，今回用いた画像分類タスクにおける DNN モデルの学習結果において，SGD に比べて適応的最適化手法によって得られる最終的な重みパラメータの汎化性，頑健性は確保されにくいことが示された．今後の課題としては，自然言語処理や音声処理などの画像分類以外の深層学習タスクにおいても同様の分析を行う必要がある．また本研究で得られた最適化手法ごとのヘッセ行列の固有値に関する知見を利用して，汎化性・頑健性を考慮した最適化手法の開発も行う予定である．

研究指導教員：若林 啓
副研究指導教員：加藤 誠

深層ニューラルネットワークの適応的最適化手法における汎化性・頑健性の分析

筑波大学
図書館情報メディア研究科

2021年03月

本川 哲哉

目 次

図 目 次

第1章　序章

1.1　深層学習の広がり

近年，深層ニューラルネットワーク（Deep Neaural Network，以下 DNN）を用いる深層学習（Deep Learning）が画像処理や自然言語処理，音声処理など多くの機械学習タスクで成功を収めている．ニューラルネットワークのアイディア自体は，機械学習の歴史で見てもかなり初期の頃から存在する．GPU などの計算処理技術の大幅な向上や，インターネットの普及によるビッグデータの充実化に伴い，DNN の学習を現実的に行うことが可能になった．また実験的研究に基づく多くのヒューリスティクスによって，様々なタスクで他の機械学習モデルに比べて圧倒的な DNN の認識性能を引き出せるようになった．DNN は従来の統計的機械学習モデルとは異なり，モデルの持つ重みパラメータの次元数が非常に大きい．従来の統計モデルでは"オッカムの剃刀"と呼ばれる思想に基づき，重みパラメータの次元数（モデルの複雑さ）をデータ次元数よりも小さくすることが重要視されていた．しかしながら画像処理分野でよく用いられる ResNet[14] や VGGNet[36] の重みパラメータ数は数百万から数億にまでのぼり，一般にデータ次元数に対して遥かに大規模である．また近年，自然言語処理分野でも BERT[7] や GPT-3[4] のような数千億個の重みパラメータを持つ非常に大規模なモデルも登場している．

非常に大規模な DNN がなぜ過学習せずに未知のデータに対しても高い精度で予測できるのかについて，まだ完全には明らかになっていない．

1.2　汎化性と頑健性

一般に機械学習において，モデルの学習がうまく行われたかどうかは訓練データだけでなく，モデルの評価のために用いるテストデータ（未知のデータ）に対しても正しく予測できるかによって確認する．テストデータに対してもうまく予測が行えるときにモデルは"汎化している"と言い，テストデータに対してもうまく予測を行うことができる性質を「汎化性」と呼ぶ．逆に訓練データに対してのみうまく予測可能で，テストデータに対する予測がうまくいかないような状態を「過学習」と呼ぶ．

また学習済みモデルにおいて，重みパラメータ，あるいは入力に対して摂動を加えた時に予測結果が変わりにくいという性質を「頑健性」と呼ぶ．

1.3　汎化性と損失関数形状の関係性

DNN の学習では訓練データを用いて計算される訓練損失を最小化することで重みパラメータの探索を行う．ここで一般的に，学習に用いる訓練データと評価に用いるテストデータは独立同分布から生成されると仮定するため，訓練損失とテスト損失が似た形状になることが予想される（図 1.1）．このような場合，勾配の変化が緩やかで平坦な局所解付近では，訓練損失とテスト損失のずれが小さくなる．一方で，勾配の変化が急峻な局所解付近

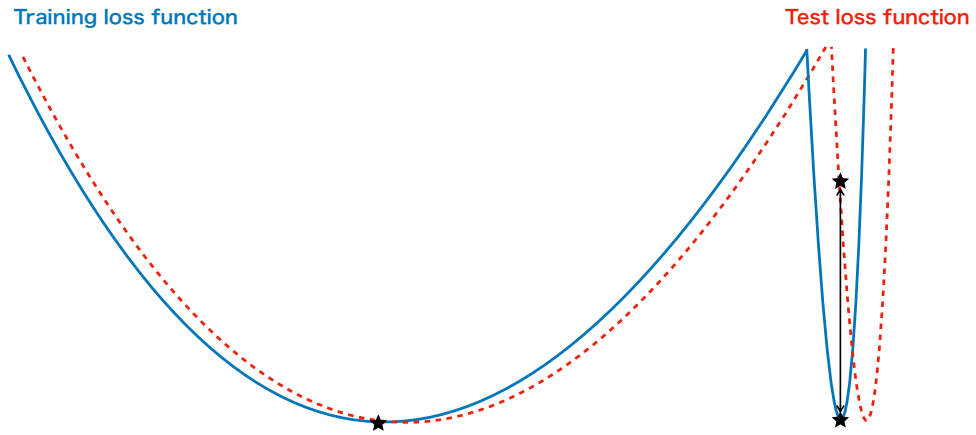


図 1.1: 訓練損失（青の実線）とテスト損失（赤の点線）の形状の違い．横軸は重みパラメータ，縦軸は損失関数の値を表す．

では訓練損失とテスト損失のずれが大きくなり，平坦な局所解と比べて汎化性が確保されにくいことが知られている [15, 17]．

1.4 本研究の貢献

通常 DNN の学習では損失関数の重みパラメータに関する一次微分で構成される勾配ベクトルを用いて，重みパラメータの最適化が行われる．このような最適化手法は勾配降下法と呼ばれ，深層学習では学習データの一部（ミニバッチ）を用いて繰り返し最適化を行う確率的勾配降下法（Stochactic Gradient Descent，以下 SGD）が主流である．DNN の最適化手法において，学習を行う前に人手で決めなければならないパラメータのことを重みパラメータとは区別してハイパーパラメータと呼ぶ．SGD ではハイパーパラメータの 1 つとして，学習時に勾配方向にどの程度進むかを表す学習率を設定する必要がある．学習率によっては全く最適化が進まなかったり，損失関数の値が発散してしまうことがある．近年では学習率を DNN の学習中に適応的に調節する適応的最適化手法が多く提案されている．適応的最適化手法は SGD よりも早く収束することで知られており，様々な深層学習タスクでよく利用される．その反面，特に画像分類タスクにおいては SGD よりも最終的に得られたモデルの汎化性が確保されにくいという報告も見られる [41]．

本研究ではこの問題に対するアプローチとして大きく以下の 3 つの分析を行い，画像分類タスクにおける適応的最適化手法に対する SGD の汎化性・頑健性に関して実験的に解釈を与える．

- 損失関数における重みパラメータに関するヘッセ行列の固有値を用いた損失関数形状の分析
- 竹内情報量規準（TIC）を用いた汎化性の分析
- 敵対的摂動を用いた頑健性の分析

1.5　本論文の構成

まず，2 章ではニューラルネットワークの定式化と本研究で扱う数学的な準備を行い，3 章では本研究と関連の深い研究についていくつか紹介する．4 章では本研究で行う実験の分析手法について詳しくレビューし，5 章で実験結果について報告する．6 章で本論文をまとめる．

第2章　準備

2.1　ニューラルネットワーク

本章ではニューラルネットワークの定式化を行い，数学的な準備を行う．

2.1.1　全結合ニューラルネットワーク

本節では，ニューラルネットワークモデルの中でも最も基本的な構成の全結合ニューラルネットワーク（Fully-Connected Neural network）を定式化する．訓練データ集合を $\mathbb{D}^{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}, \boldsymbol{x}_i \in \mathbb{R}^{d^{(0)}}, \boldsymbol{y}_i \in \mathbb{R}^{C}$ とする．本研究ではクラス分類タスクを扱うため，出力次元は分類クラス数の C とする．$\boldsymbol{y}_i$ は分類クラスに対応するインデックスの成分のみが 1 となり，その他の成分は全て 0 となるようなベクトルである．このようなベクトルを One-hot ベクトルと呼ぶ．例えば犬と猫の二値分類の場合，犬を 1 番目の要素，猫を 2 番目の要素とすると，犬を表す $\boldsymbol{y}_i$ は $[1, 0]$，猫を表す $\boldsymbol{y}_i$ は $[0, 1]$ となる．

全結合ニューラルネットワークの層数は K とする．k 層目の重みパラメータを $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d^{(k+1)}} \times \mathbb{R}^{d^{(k)}}, k = 0, \cdots, K-1$（$d^{(k)}$ は k 層目におけるノード数，また $d^{(K)} = C$），活性化関数を $\phi(\cdot)$ とする．このとき全結合ニューラルネットワークの各層で以下の変換を行う．

$$
\begin{aligned}
\boldsymbol{z}^{(0)} &= \boldsymbol{x}_i \\
\boldsymbol{a}^{(0)} = \boldsymbol{W}^{(0)}\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)} &= \phi(\boldsymbol{a}^{(0)}) \\
&\vdots \\
\boldsymbol{a}^{(K-2)} = \boldsymbol{W}^{(K-2)}\boldsymbol{z}^{(K-2)}, \boldsymbol{z}^{(K-1)} &= \phi(\boldsymbol{a}^{(K-2)}) \\
\boldsymbol{a}^{(K-1)} = \boldsymbol{W}^{(K-1)}\boldsymbol{z}^{(K-1)}, \boldsymbol{z}^{(K)} &= \phi(\boldsymbol{a}^{(K-1)})
\end{aligned}
\tag{2.1}
$$

本研究では最終層以外の活性化関数 $\phi(\cdot)$ としては，入力として与えられるベクトルの成分ごとに以下で定義される ReLU 関数を仮定する [27].

$$
\left[\mathrm{ReLU}(\boldsymbol{a}^{(k)})\right]_j = \max(0, \left[\boldsymbol{a}^{(k)}\right]_j), \ j = 1, \cdots, d^{(k+1)}, \ k = 0, \cdots, K-2 \tag{2.2}
$$

またクラス分類タスクを扱うため，ニューラルネットワークの最終層における活性化関数 $\phi(\cdot)$ には，入力 $\boldsymbol{a}^{(K-1)}$ の成分ごとに以下で定義される Softmax 関数を仮定する．

$$
\left[\mathrm{Softmax}(\boldsymbol{a}^{(K-1)})\right]_j = \frac{\exp(\left[\boldsymbol{a}^{(K-1)}\right]_j)}{\sum_{c=1}^{C}\exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)}, \ j = 1, \cdots, C \tag{2.3}
$$

ここでベクトル $\boldsymbol{v}$ において，$[\boldsymbol{v}]_j$ は $\boldsymbol{v}$ の j 番目の要素を表す．式（2.3）より最終出力 $\boldsymbol{z}^{(K)}$ の各要素の和が 1 となるため，各要素が各クラスに相当する確率として解釈が可能になる．

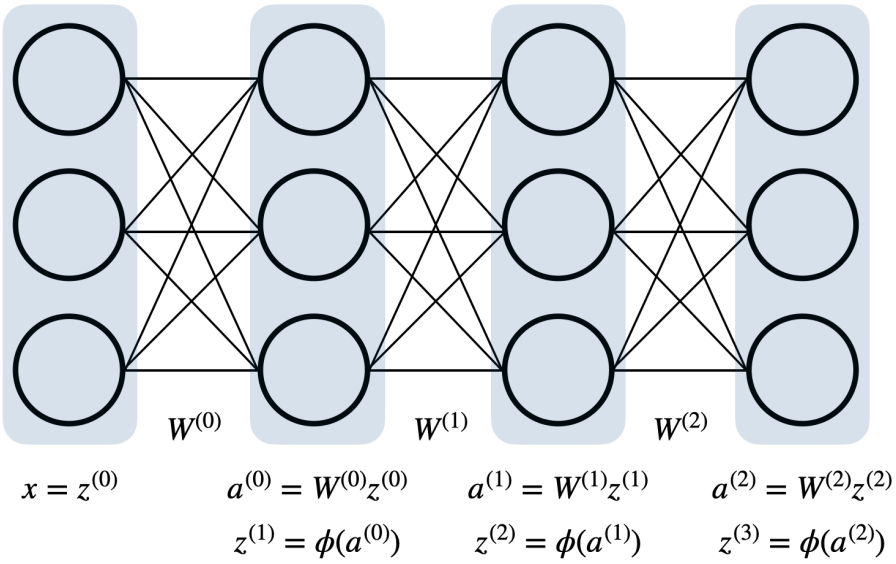


図 2.1: 全結合ニューラルネットワークの概略図.

2.1.2 損失関数

訓練損失最小化

ニューラルネットワークの重みパラメータをまとめて $\boldsymbol{\theta} \in \mathbb{R}^D$（$D = \sum_{k=0}^{K-1} d^{(k+1)} \times d^{(k)}$）とし，ある 1 つのデータポイント $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ に対するニューラルネットワークの出力を $f(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)$ とする．クラス分類タスクのため，データポイント毎の損失関数としては以下で定義される多クラス交差エントロピー損失を用いる．

$$l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) = -\sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log [f(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)]_j \tag{2.4}$$

学習の目的は以下で定義される，訓練データ全体で計算される訓練損失 $L(\boldsymbol{\theta})$ を最小化することである．

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.5}$$

勾配ベクトル

まずデータポイントごとの損失関数 $l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)$ における重みパラメータ $\boldsymbol{\theta} \in \mathbb{R}^D$ に関する一次微分で構成される勾配ベクトルを定義する．勾配ベクトル $\boldsymbol{g}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^D$ の各要素は以下で定義される．

$$[\boldsymbol{g}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)]_m = \frac{\partial l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)}{\partial \boldsymbol{\theta}_m}, \quad m = 1, \cdots, D \tag{2.6}$$

以下，勾配ベクトルのことを単に勾配と呼ぶ．訓練データ全体で計算される訓練損失 $L(\boldsymbol{\theta})$ の重みパラメータに関する勾配 $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ は，データポイントごとの損失関数の勾配を用い

て以下で定義する．

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{g}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.7}$$

勾配の各成分は重みパラメータの各成分の微小な変化に対する損失関数の変化を表すため，定性的には損失関数の「傾き具合」を表す．

ヘッセ行列

まずデータポイントごとの損失関数 $l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)$ における重みパラメータ $\boldsymbol{\theta} \in \mathbb{R}^D$ に関する二次微分で構成されるヘッセ行列を定義する．ヘッセ行列 $\boldsymbol{H}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^{D\times D}$ の各要素は以下で定義される．

$$[\boldsymbol{H}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)]_{(m,n)} = \frac{\partial^2 l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)}{\partial [\boldsymbol{\theta}]_m \partial [\boldsymbol{\theta}]_n}, \quad m = 1, \cdots, D \quad n = 1, \cdots, D \tag{2.8}$$

ここで行列 $\boldsymbol{A}$ において，$[\boldsymbol{A}]_{(m,n)}$ は $\boldsymbol{A}$ の m 行 n 列目の要素を表す．訓練データ全体で計算される訓練損失 $L(\boldsymbol{\theta})$ の重みパラメータに関するヘッセ行列は，データポイントごとの損失関数のヘッセ行列を用いて以下で定義する．

$$\boldsymbol{H}(\boldsymbol{\theta}) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{H}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.9}$$

ヘッセ行列の各成分は重みパラメータの各成分の微小な変化に対する勾配の変化を表すため，定性的には損失関数の「曲がり具合」を表す．

汎化ギャップ

従来の統計的機械学習モデルでは，重みパラメータの次元数がデータ次元数よりも大きいとき，式（2.5）に基づいて訓練損失を最小化すると過学習を引き起こすことが知られている．本来，機械学習で真に解きたい問題は以下の期待損失を最小化することである．$p(\boldsymbol{x}, \boldsymbol{y})$ は真のデータ生成分布とする．

$$\hat{L}(\boldsymbol{\theta}) = E_{p(\boldsymbol{x},\boldsymbol{y})}\left[l(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}))\right] \tag{2.10}$$

式（2.5）と式（2.10）との差 $|L - \hat{L}|$ は汎化ギャップ（Generalization Gap）と呼ばれる．訓練損失最小化後に汎化ギャップが小さいときは，訓練損失だけでなく期待損失も小さくなっているため，汎化しているモデルと言える．逆に汎化ギャップが大きいときは過学習を引き起こしているため，汎化性は確保されにくい．

真のデータ生成分布は一般に未知のため，実際に期待損失を計算するときは有限個のテストデータ集合 $\mathbb{D}^{\text{test}} = \{(\boldsymbol{x}_i^{\text{test}}, \boldsymbol{y}_i^{\text{test}})\}_{i=1}^{N^{\text{test}}}, \boldsymbol{x}_i^{\text{test}} \in \mathbb{R}^{d^{(0)}}, \boldsymbol{y}_i^{\text{test}} \in \mathbb{R}^C$ を用いて近似する．

$$\check{L}(\boldsymbol{\theta}) = \frac{1}{N^{\text{test}}}\sum_{i=1}^{N^{\text{test}}} l(\boldsymbol{\theta}; \boldsymbol{x}_i^{\text{test}}, \boldsymbol{y}_i^{\text{test}}) \tag{2.11}$$

$\check{L}(\boldsymbol{\theta})$ のことをテスト損失と呼ぶ．

つまり，汎化ギャップ G は実際には訓練損失とテスト損失の誤差の絶対値を用いて計算される．

$$G = |L(\boldsymbol{\theta}) - \check{L}(\boldsymbol{\theta})| \tag{2.12}$$

古典的な統計モデルとは大きく挙動の異なる DNN の汎化性に対しては様々なアプローチで研究が行われている．

2.1.3 誤差逆伝播法

本節では，ニューラルネットワークについて，単一のデータポイントごとの損失関数として定義される多クラス交差エントロピー $l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) = -\sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log [f(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)]_j$ の勾配 $g(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)$ を効率的に計算するアルゴリズムを導入する．これはニューラルネットワークの推論（予測）が順方向への演算（forward-propagation）であることに対応して，誤差逆伝播法（back-propagation）として知られている．

2.1.1 節の定式化に従って，任意の層の重みパラメータ $\boldsymbol{W}^{(k)}$ に関する多クラス交差エントロピー損失 $l(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)$（以下，単に l と表記する）の勾配を求める．今，出力層の活性化関数としては Softmax 関数（式（2.3））を用いているため，多クラス交差エントロピー損失は以下のように変形できる．

$$
\begin{aligned}
l &= -\sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log [f(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i)]_j \\
&= -\sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log \frac{\exp(\left[\boldsymbol{a}^{(K-1)}\right]_j)}{\sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)} \\
&= -\sum_{j=1}^{C} [\boldsymbol{y}_i]_j \left(\left[\boldsymbol{a}^{(K-1)}\right]_j - \log \sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)\right) \\
&= \sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log \sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c) - \sum_{j=1}^{C} [\boldsymbol{y}_i]_j \left[\boldsymbol{a}^{(K-1)}\right]_j
\end{aligned} \tag{2.13}
$$

式（2.13）最終行の第 2 項における，$\left[\boldsymbol{a}^{(K-1)}\right]_{j'}$ に関する一次微分は $-\frac{\partial \sum_{j=1}^{C} [\boldsymbol{y}_i]_j \left[\boldsymbol{a}^{(K-1)}\right]_j}{\partial \left[\boldsymbol{a}^{(K-1)}\right]_{j'}} = -[\boldsymbol{y}_i]_{j'}$ となる．式 (2.13) 最終行の第 1 項における，$\left[\boldsymbol{a}^{(K-1)}\right]_{j'}$ に関する一次微分は，$\sum_{j=1}^{C} [\boldsymbol{y}_i]_j = 1$ より以下のように与えられる．

$$
\begin{aligned}
\frac{\partial \sum_{j=1}^{C} [\boldsymbol{y}_i]_j \log \sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)}{\partial \left[\boldsymbol{a}^{(K-1)}\right]_{j'}} &= \frac{\partial \log \sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)}{\partial \left[\boldsymbol{a}^{(K-1)}\right]_{j'}} \\
&= \frac{\exp(\left[\boldsymbol{a}^{(K-1)}\right]_{j'})}{\sum_{c=1}^{C} \exp(\left[\boldsymbol{a}^{(K-1)}\right]_c)} \\
&= \left[\boldsymbol{z}^{(K)}\right]_{j'}
\end{aligned} \tag{2.14}
$$

よって多クラス交差エントロピー（式（2.13））における $\left[\boldsymbol{a}^{(K-1)}\right]_{j'}$ に関する一次微分は $\left[\boldsymbol{z}^{(K)}\right]_{j'} - [\boldsymbol{y}_i]_{j'}$ となる．従って，多クラス交差エントロピーにおける最終層の入力 $\boldsymbol{a}^{(K-1)}$ に関する勾配は，これを $j' = 1, \cdots, C$ まで成分ごとに並べることで以下によって与えられる．

$$
\frac{\partial l}{\partial \boldsymbol{a}^{(K-1)}} = \boldsymbol{z}^{(K)} - \boldsymbol{y}_i \tag{2.15}
$$

$\boldsymbol{z}^{(K)}$ は最終層の出力，つまりニューラルネットワークの予測値であり，$\boldsymbol{y}_i$ は正解ラベルなので，式（2.15）は単に予測値と正解値の誤差を表している．

次に，多クラス交差エントロピー l における最終層の重みパラメータ $\boldsymbol{W}^{(K-1)}$ に関する勾配は微分の連鎖律より以下で与えられる．

$$
\frac{\partial l}{\partial \boldsymbol{W}^{(K-1)}} = \frac{\partial l}{\partial \boldsymbol{a}^{(K-1)}} \frac{\partial \boldsymbol{a}^{(K-1)}}{\partial \boldsymbol{W}^{(K-1)}} = (\boldsymbol{z}^{(K)} - \boldsymbol{y}_i)(\boldsymbol{z}^{(K-1)})^T
$$

ここで，最終層以外の k 層目における活性化関数自身の微分で構成される対角行列を $\boldsymbol{D}^{(k)} = \mathrm{diag}(\left[\phi'(\boldsymbol{a}^{(k-1)})\right]_1, \cdots \left[\phi'(\boldsymbol{a}^{(k-1)})\right]_{d^{(k)}})$ とする．すると，$K-2$ 層目の重みパラメータ $\boldsymbol{W}^{(K-2)}$ に関する勾配も微分の連鎖律より以下で与えられる．

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{W}^{(K-2)}} &= \frac{\partial l}{\partial \boldsymbol{a}^{(K-1)}}\frac{\partial \boldsymbol{a}^{(K-1)}}{\partial \boldsymbol{z}^{(K-1)}}\frac{\partial \boldsymbol{z}^{(K-1)}}{\partial \boldsymbol{a}^{(K-2)}}\frac{\partial \boldsymbol{a}^{(K-2)}}{\partial \boldsymbol{W}^{(K-2)}} \\ &= (\boldsymbol{W}^{(K-1)}\boldsymbol{D}^{(K-1)})^T(\boldsymbol{z}^{(K)} - \boldsymbol{y}_i)(\boldsymbol{z}^{(K-2)})^T\end{aligned}$$

以上を再帰的に繰り返すことで，任意の k 層目における重みパラメータ $\boldsymbol{W}^{(k)}$ に関する多クラス交差エントロピー損失 l の勾配が与えられる．

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{W}^{(k)}} &= \frac{\partial l}{\partial \boldsymbol{a}^{(K-1)}}\frac{\partial \boldsymbol{a}^{(K-1)}}{\partial \boldsymbol{z}^{(K-1)}}\frac{\partial \boldsymbol{z}^{(K-1)}}{\partial \boldsymbol{a}^{(K-2)}}\frac{\partial \boldsymbol{a}^{(K-2)}}{\partial \boldsymbol{z}^{(K-2)}}\cdots\frac{\partial \boldsymbol{a}^{(k)}}{\partial \boldsymbol{W}^{(k)}} \\ &= (\boldsymbol{W}^{(K-1)}\boldsymbol{D}^{(K-1)}\boldsymbol{W}^{(K-2)}\boldsymbol{D}^{(K-2)}\cdots\boldsymbol{W}^{(k+1)}\boldsymbol{D}^{(k+1)})^T(\boldsymbol{z}^{(K)} - \boldsymbol{y}_i)(\boldsymbol{z}^{(k)})^T\end{aligned}$$

ここで，$\boldsymbol{e}^{(k)} = \frac{\partial l}{\partial \boldsymbol{a}^{(k)}}$ とすると，誤差逆伝播法の手順は以下のように書ける．

$$\begin{aligned}\boldsymbol{e}^{(K)} &= \boldsymbol{z}^{(K)} - \boldsymbol{y}_i \\ \boldsymbol{e}^{(K-1)} &= (\boldsymbol{W}^{(K-1)}\boldsymbol{D}^{(K-1)})^T\boldsymbol{e}^{(K)} \\ &\vdots \\ \boldsymbol{e}^{(1)} &= (\boldsymbol{W}^{(1)}\boldsymbol{D}^{(1)})^T\boldsymbol{e}^{(2)}\end{aligned}$$

従って $\frac{\partial l}{\partial \boldsymbol{W}^{(k)}} = \boldsymbol{e}^{(k+1)}(\boldsymbol{z}^{(k)})^T, k = 0, \cdots, K-1$ によって，損失関数における任意の層の重みパラメータ $\boldsymbol{W}^{(k)}$ に関する勾配を求めることができる．これは，式（2.1）における順伝播計算の際に，各層の活性化関数の出力 $\boldsymbol{z}^{(k)} = \phi(\boldsymbol{a}^{(k-1)})$ と活性化関数自身の微分 $\frac{\partial \phi(\boldsymbol{a}^{(k-1)})}{\partial \boldsymbol{a}^{(k-1)}}$ さえ保持していればよく，軽量かつ高速に動作することが期待できる．

2.1.4 勾配降下法

本節では DNN の訓練損失最小化のために用いられる最適化手法の中でも，最も基本的なアルゴリズムである勾配降下法について整理する．以下，更新前の重みパラメータを $\boldsymbol{\theta}_t$，更新後の重みパラメータを $\boldsymbol{\theta}_{t+1}$ とする．また，η は学習率である．

フルバッチ勾配降下法

フルバッチ勾配降下法（Full-Batch Gradient Descent，以下単に GD とする）とは，一度に訓練データ全体を用いて勾配の平均値を計算し，重みパラメータを更新する方法である．重みパラメータ更新式は以下のようになる．

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta\nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t)\end{aligned} \tag{2.16}$$

GD では一度の重みパラメータ更新に全ての訓練データを使用するので収束が遅く，一度に全ての訓練データがメモリに載らないほど大きなときには使用できない．

確率的勾配降下法

確率的勾配降下法（Stochactic Gradient Descent，以下SGDとする）は，ランダムにシャッフルされた訓練データ集合から順番に1つの訓練データを取り出し，そのデータポイントにおける勾配 $\boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i)$ を用いて重みパラメータの更新を行う．

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.17}$$

GDでは各ステップ t ごとに毎回全ての訓練データを用いて勾配を計算するのに対して，SGDではある1つのデータポイントのみの勾配を用いるので各ステップでの計算時間の削減に繋がる．またSGDの更新式（2.17）を以下のように変形すると，GDの更新式（2.16）に対して $\eta(\nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) - \boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i))$ というノイズが加わっているとみなすことができる（図2.2）．

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \eta(-\nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) + \nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) - \boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i)) \\ &= \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) + \eta(\nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) - \boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_i, \boldsymbol{y}_i)) \end{aligned} \tag{2.18}$$

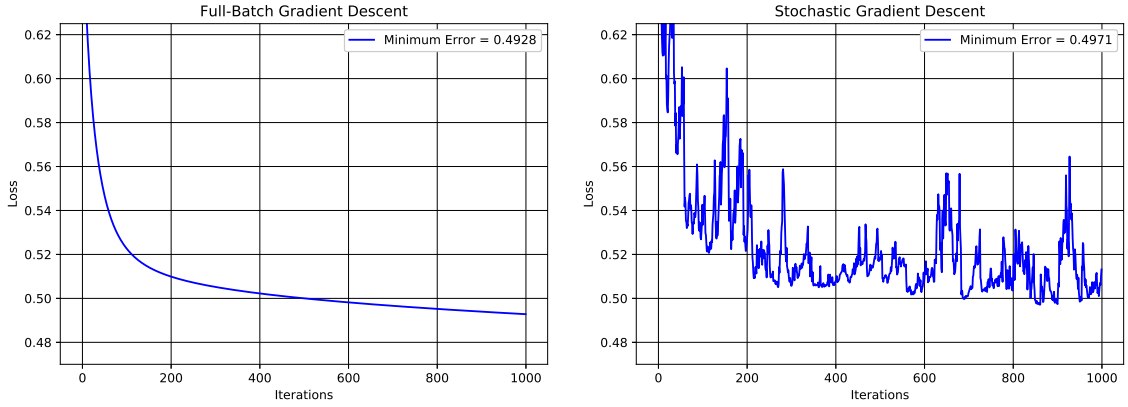


図 2.2: GDとSGDの学習曲線における挙動の違い．SGDは単一のデータポイントにおける勾配を用いるのでノイズが乗るため分散が大きい．

ミニバッチSGD

ミニバッチSGDは，GDとSGDの中間的な手法に相当する．GDが全ての訓練データ，SGDが1つのデータポイントのみを使用するのに対して，ミニバッチSGDではランダムにシャッフルされた訓練データ集合から，順番に M 個の訓練データを取り出して勾配の平均値を計算し，重みパラメータ更新を行う．

$$\begin{aligned} \{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^{M} &\sim \mathbb{D}^{\text{train}} \quad \text{(訓練データ集合からM個のデータを取り出す)} \\ \nabla_{\boldsymbol{\theta}_t} L_M(\boldsymbol{\theta}_t) &= \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{g}(\boldsymbol{\theta}_t; \boldsymbol{x}_m, \boldsymbol{y}_m) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} L_M(\boldsymbol{\theta}_t) \end{aligned} \tag{2.19}$$

ミニバッチSGDは $M = 1$ のときSGDに一致し，$M = N$ のときにGDに一致する．

DNNの学習ではSGDの軽量な動作とGDの安定した収束を得るために，適当なバッチサイズのミニバッチSGDを使用することが多い．深層学習の文脈では用語としてミニバッ

チ SGD のことを単に SGD と呼ぶことも多く，その慣例に従い本論文では以下 SGD と書かれた場合，特に断りのない限りミニバッチ SGD のことを指すことにする．

また，深層学習ではモデルの学習における 1 回の繰り返し（iteration）を表す用語として「エポック（epoch）」を用いることが多い．1 エポックの間にモデルは全訓練データを用いて重みパラメータを更新する．GD の場合，1 エポックの間に 1 回の重みパラメータ更新が行われ，ミニバッチサイズ M の SGD の場合，1 エポックの間に N/M 回の重みパラメータ更新が行われる．N/M が割り切れない場合，最後の余った $N \mathrm{mod} M$ サイズのミニバッチ SGD を実行する．

第3章　関連研究

3.1　ニューラルネットワークにおける損失関数の二次情報

近年，ニューラルネットワークの学習ダイナミクスの理解や最適化効率の改善のために，損失関数の二次情報を利用する研究が盛んに行われている．ニューラルネットワークの学習後半は訓練損失の重みパラメータに関する勾配のノルムはほぼ 0 に収束している（図 3.1）．そのため訓練損失の重みパラメータに関するヘッセ行列を用いて，損失関数の曲がり具合に着目する．

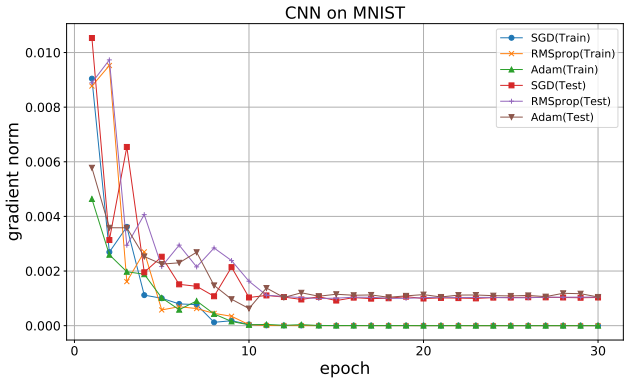


図 3.1: 異なる最適化手法毎に MNIST 手書き数字分類データを学習した Convolutional Neural Network（CNN）における勾配のノルムの変化．学習後半はどの手法でも訓練損失の重みパラメータに関する勾配のノルムはほぼ 0 に収束している．

3.1.1　ヘッセ行列の固有値に関する研究

LeCun（1998）は最初にニューラルネットワーク学習後に損失関数の重みパラメータに関するヘッセ行列の固有値分布を分析した人物の 1 人である [23]．彼らはヘッセ行列における絶対値の大きな固有値に対応する重みパラメータが学習に悪影響を与えるのではないかと指摘した．Sagun ら（2016）は中間層が 1 層のニューラルネットワークにおけるヘッセ行列の固有値分布を分析し，絶対値の大きな固有値の数は限られていることを実験的に示した [33]（図 3.2）．彼らはヘッセ行列における絶対値の大きな固有値集合を outlier eigenvalues と呼び，残りの固有値を bulk と呼んだ．また彼らは outlier eigenvalues の個数がデータセットとニューラルネットワークのモデル構造に依存することを確認した．

Chaudhari ら（2017）は Entropy-SGD という名の新たな最適化手法を提案した [5]．これは損失関数の局所形状を考慮して，より平坦な空間を目指すような最適化手法である．この手法では，汎化ギャップの小さな局所解ではヘッセ行列における絶対値の大きな固有値がわずかしか存在しないという実験的な観察に基づいている．

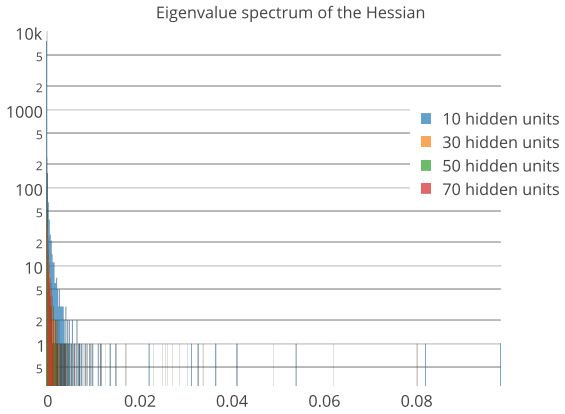


図 3.2: MNIST 手書き数字分類データ学習後の中間層が 1 層のニューラルネットワークにおけるヘッセ行列の固有値分布（図は Sagun ら（2016）[33] から引用，横軸は固有値の値，縦軸は頻度を表す.）0.02 よりも大きい固有値集合（outlier eigenvalues）は極端に少なく，残りの 0 に近い固有値（bulk）がほとんどを占めていることがわかる．

SGD のバッチサイズがヘッセ行列の固有値分布に与える影響を調べた研究もいくつか存在する．一般にニューラルネットワークの学習ではバッチサイズを大きくすることで，並列化による計算高速化を期待する．しかしながらバッチサイズを大きくすると，汎化性が確保されにくくなることで知られている．その原因としては主に SGD のノイズ効果が薄くなるためという説と，大きなバッチサイズの場合は小さなバッチサイズの時と比べて，同じエポック数では重みパラメータ更新の回数が減少するためではないかという説が提唱されている [17]．Yao ら（2018）はヘッセ行列の固有値を用いて SGD によって学習された DNN モデルの重みパラメータ空間における損失関数の平坦性を確認した [44]．図 3.3 に示すように，彼らはバッチサイズが大きい場合にヘッセ行列の固有値の絶対値が大きな（急峻な）局所解に収束し，バッチサイズが小さい場合にヘッセ行列の固有値の絶対値が小さな（平坦な）局所解に収束するということを実験的に示した．また彼らはバッチサイズが大きいほど，敵対的摂動に対する頑健性が劣化することを実験的に確認した．McCandlish らはヘッ

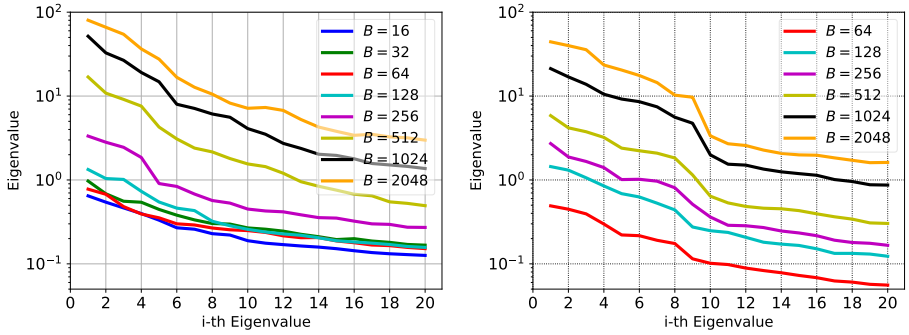


図 3.3: 異なるバッチサイズで学習した CNN on CIFAR-10 におけるヘッセ行列の上位 20 個の固有値分布（図は Yao ら（2018）[44] から引用，横軸は値が大きい順に並べた際の固有値のインデックス，縦軸は固有値の値を表す.）左右でモデルの重みパラメータの総数が異なる．左の方が大きなパラメータサイズのモデルにおける結果である．どちらのモデルにおいても，バッチサイズが大きいほど固有値全体が大きな値に分布していることがわかる．

セ行列を用いて計算量削減に繋がる限界のバッチサイズ（Critical Batch Size）を測る指標を提案した [25].

Ghorbani ら（2019）はヘッセ行列の全ての固有値を計算するコストを改善するために，新たな近似方法を提案した [10]．また彼らはこの近似方法を利用して outlier eigenvalues における最大固有値 $\lambda_{\max}$ と最小固有値 $\lambda_{\min}$ の比（$\lambda_{\max}/\lambda_{\min}$）のダイナミクスも調査した．Gur-Ari らは大規模な DNN を学習すると，勾配が非常に小さな部分空間に収束することを実験的に示した．部分空間はヘッセ行列の上位の固有値に対応する固有ベクトルで張られており，勾配降下法がこの部分空間で実行されることを示唆している [13].

また Maddox ら（2020）は人工データを用いた二値分類問題において，重みパラメータ空間におけるヘッセ行列の上位の固有値に対応する固有ベクトル方向に対して摂動を加えた時に，決定境界が大きく変化して分類精度が悪化することを示した [24]．つまりヘッセ行列の絶対値の大きな固有値に対応する固有ベクトル方向に摂動を加えることが，モデルの頑健性に影響を与えることを実験的に確認した．

以上のように，ヘッセ行列の固有値を用いて学習ダイナミクスやモデルの汎化性，頑健性を分析する研究は行われてきているが，この分析方法を適応的最適化手法が持つ性質の分析に対して適用した研究は行われていない．

3.1.2 DNN におけるヘッセ行列の計算効率化に関する研究

実用的な DNN の重みパラメータ数は数百万から数億にまでのぼることもある．例えば，ImageNet コンペティションでブレークスルーを起こした AlexNet[20] の重みパラメータ数は 6200 万であり，画像処理における特徴量抽出器としてもよく使用される VGG-16[36] は 1 億 3800 万重みパラメータを持つ．このような大規模な DNN モデルにおけるヘッセ行列を Exact に計算するのはメモリ容量，計算時間の観点で現実的には困難である．

Hessian vector product

多くの場合ヘッセ行列そのものではなく，ヘッセ行列と任意のベクトルとの積（Hessian vector product，以下 Hvp）さえ計算出来れば十分であることが多い．例えばヘッセ行列の絶対値最大の固有値を求めたい場合，べき乗法（power iteration[21]）によって計算が可能である．べき乗法ではヘッセ行列 $\boldsymbol{H}$ の絶対値最大の固有値を求めたいときに，ランダムベクトル $\boldsymbol{v_0} \neq \boldsymbol{0}$ を初期ベクトルとして，逐次的に以下の計算を繰り返す．

$$\boldsymbol{v}_k = \frac{\boldsymbol{H}\boldsymbol{v}_{k-1}}{\|\boldsymbol{H}\boldsymbol{v}_{k-1}\|} \tag{3.1}$$

k が十分大きいとき，$\boldsymbol{v}_k$ がヘッセ行列 $\boldsymbol{H}$ の絶対値最大の固有値に対応する固有ベクトルの方向に収束していくことを利用して，絶対値最大の固有値を計算する．実際にべき乗法を用いる場合，次式のように $\boldsymbol{v}_k$ と $\boldsymbol{v}_{k-1}$ の差のノルムを計算し，許容誤差 ϵ を用いて収束判定を行う．

$$\|\boldsymbol{v}_k - \boldsymbol{v}_{k-1}\| < \epsilon$$

ヘッセ行列のトレースの近似計算法としては Hutchinson Method[2] と呼ばれるアルゴリズムがある．Hutchinson Method ではヘッセ行列とラデマッハランダムベクトル（各要素

が $\frac{1}{2}$ の確率で 1 or -1 を取る）との二次形式の期待値を近似的に計算する．$\boldsymbol{I}$ は単位行列を表す．

$$tr(\boldsymbol{H}) = tr(\boldsymbol{H}\boldsymbol{I}) = tr(\boldsymbol{H}E[\boldsymbol{v}\boldsymbol{v}^T]) = E[tr(\boldsymbol{H}\boldsymbol{v}\boldsymbol{v}^T)] = E[\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v}] \tag{3.2}$$

さらに，Ghorbani ら（2019）によってヘッセ行列の固有値密度を Hvp を用いて近似的に計算するアルゴリズムも開発されている [10]．彼らはこのアルゴリズムを Stochastic Lanczos Quadrature（以下，SLQ）と呼んでいる．ヘッセ行列の固有値密度は以下で定義される．

$$\phi(t) = \frac{1}{m}\sum_{i=1}^{m}\delta(t-\lambda_i) \tag{3.3}$$

ここで，$\delta(\cdot)$ はディラックのデルタ関数，λ_i はヘッセ行列における i 番目の固有値，m は固有値の総数を表す．式（3.3）は固有値の個数の割合を表す．重みパラメータ次元数の大きな DNN のヘッセ行列において，式（3.3）を直接計算することは難しいため，SLQ ではまず次元の小さな三重対角行列に近似する．この近似計算の際に Hvp を利用する．次元数が十分小さい三重対角行列の固有値・固有ベクトルを全て計算して，それらを用いて元のヘッセ行列の固有値密度を推定する．固有値密度の推定にはガウシアンカーネルを用いるため，1 を超える値も取り得ることに注意する．

Hvp は，近似的には数値微分によって求めることができる．

$$H(\boldsymbol{\theta})v \sim \frac{g(\boldsymbol{\theta}+r\boldsymbol{v}) - g(\boldsymbol{\theta})}{r} \tag{3.4}$$

Hvp を Exact に計算するために，Pearlmutter（1994）によってニューラルネットワークにおける Hvp を誤差逆伝播中に高速に計算するアルゴリズムが提案されている [31]．本研究では Pearlmutter の提案したアルゴリズムが実装されている Pytorch[30] と呼ばれるフレームワークを用いて Exact な Hvp を求めている．また Hvp を利用してべき乗法，Hutchinson Method，SLQ のアルゴリズムが実装された PyHessian[43] と呼ばれるライブラリを用いてヘッセ行列の固有値に関する計算を行っている．

The Generalized Gauss-Newton Matrix

Hessian vector product は陽な形でヘッセ行列を計算せずに，ヘッセ行列と任意のベクトルの積を効率的に計算する方法である．ここでは，DNN のヘッセ行列の近似としてよく利用されている Schraudolph (2002) が導入した The Generalized Gauss-Newton Matrix（以下 GGN 行列）について議論する [35]．古典的には Gauss-Newton 近似と呼ばれるヘッセ行列の近似手法が存在しており，以下で定義される行列を Gauss-Newton 行列と呼び，準二次最適化手法としても広く利用されてきた．例えば Ortega and Rheinboldt（2000）などが詳しい [29]．

$$\tilde{\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i))^T\boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)) \tag{3.5}$$

ここで，$\boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i))$ はニューラルネットワーク出力 $f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)$ の重みパラメータ $\boldsymbol{\theta}$ に関するヤコビ行列を表す．すなわち，ヤコビ行列 $\boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i))$ の各要素は以下で表される．

$$\left[\boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i))\right]_{(m,n)} = \frac{\partial\left[f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)\right]_m}{\partial\left[\boldsymbol{\theta}\right]_n},\quad m=1,\cdots,C\quad n=1,\cdots,D \tag{3.6}$$

Schraudolph は Gauss-Newton 行列をより一般的に拡張し，以下のような近似行列を与えた．

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i))^T \nabla^2_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)} l(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i) \boldsymbol{J}_{\boldsymbol{\theta}}(f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)) \tag{3.7}$$

ここで，$\nabla^2_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)} l(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)$ はデータポイントごとの損失関数 $l(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)$ における最終出力 $f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)$ に関するヘッセ行列である．以下，$\nabla^2_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)} l(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)$ のことを $\boldsymbol{H}_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)}$ と書く．$\boldsymbol{H}_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)}$ の各成分は以下で表される．

$$\left[\boldsymbol{H}_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)}\right]_{(m,n)} = \frac{\partial l(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)}{\partial \left[f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)\right]_m \partial \left[f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)\right]_n}, \quad m = 1,\cdots,C \quad n = 1,\cdots,C \tag{3.8}$$

特に $\boldsymbol{H}_{f(\boldsymbol{\theta};\boldsymbol{x}_i,\boldsymbol{y}_i)}$ が単位行列として表せる場合，式（3.7）は古典的な Gauss-Newton 行列（式（3.5））に一致することがわかる．また活性化関数として ReLU 関数を使用する場合，活性化関数自身の二次微分の値が 0 になることを利用してヘッセ行列と GGN 行列が一致することが知られている [3]．現実的な DNN の設定ではほとんどの場合，活性化関数として ReLU 関数を用いることが多いため，GGN 行列を計算すれば十分であることがわかる．

本研究の実験で用いた BackPACK[6] と呼ばれるヘッセ行列計算のためのライブラリでは，GGN 行列を誤差逆伝播中に効率的に計算・保持するように設計されている．

3.2 適応的最適化手法

通常 DNN の学習には SGD，あるいはその発展手法である適応的最適化手法を用いる．Momentum SGD[32, 37] と Nesterov Accelerate Gradient[28] は SGD のシンプルな拡張であり，その実装の簡単さや理論的な明快さのため広く実用されている．

通常の SGD は全ての重みパラメータ更新に同じ学習率をもちいるため，収束が遅いことで知られている．近年この問題を克服するために，過去ステップにおける勾配情報を用いて学習率を適応的に決めながら最適化する手法が多く提案されている．そのような手法の中でも特によく用いられる手法として，Adagrad，Adadelta，RMSprop，Adam などが存在する [9, 45, 39, 18]．

以下では DNN の学習に用いられる最適化手法の中でも特に代表的な Momentum SGD，RMSprop，Adam についてレビューする．各手法ともランダムにシャッフルされた訓練データ集合から順番に M 個のミニバッチデータ $\{(\boldsymbol{x}_m,\boldsymbol{y}_m)\}_{m=1}^{M} \sim \mathbb{D}^{\text{train}}$ を取り出して勾配の推定に用いる．

3.2.1 Momentum SGD

Momenutm SGD[32] は DNN の学習で最も利用されている非適応的な最適化手法の 1 つで，1999 年に Qian によって提案された．Momentum SGD では過去ステップでの勾配の一次モーメントと現在の勾配との指数移動平均を用いて以下のような重みパラメータ更新を行う．

$$\begin{aligned}
\boldsymbol{g}_t &= \frac{1}{M}\sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}_t} l(\boldsymbol{\theta}_t;\boldsymbol{x}_m,\boldsymbol{y}_m) \\
\boldsymbol{m}_t &= \beta \boldsymbol{m}_{t-1} + (1-\beta)\boldsymbol{g}_t \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta \boldsymbol{m}_t
\end{aligned} \tag{3.9}$$

ここで β は指数移動平均の重みを表すハイパーパラメータであり，$\beta \in [0,1)$ である．多くの研究で実験的に SGD よりも収束が速いことで知られており，また理論的にも特定の凸二次制約のもとで SGD に比べて収束性能がよいことが証明されている [26].

3.2.2 RMSprop

RMSprop[39] は 2012 年に Hinton らによって提案された．Momentum SGD が勾配の一次モーメントを利用するのに対して，RMSprop では勾配の二次モーメントと現在の勾配との指数移動平均を用いて以下のような重みパラメータ更新を行う．

$$\begin{aligned} \boldsymbol{g}_t &= \frac{1}{M}\sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}_t} l(\boldsymbol{\theta}_t; \boldsymbol{x}_m, \boldsymbol{y}_m) \\ \boldsymbol{v}_t &= \beta \boldsymbol{v}_{t-1} + (1-\beta)\boldsymbol{g}_t^2 \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\boldsymbol{v}_t + \epsilon}} \odot \boldsymbol{g}_t \end{aligned} \tag{3.10}$$

β は指数移動平均の重みを表すハイパーパラメータであり，$\beta \in [0,1)$ である．ここで，$\frac{\eta}{\sqrt{\boldsymbol{v}_t+\epsilon}}$ は $\boldsymbol{v}_t$ の成分ごとに平方根を求め ϵ を加えて η を割ることで得られるベクトルであり，$\odot$ はベクトルの成分ごとの積（アダマール積）を表す．

3.2.3 Adam

Adam[18] は 2014 年に Kingma らによって提案され，以後 Momentum SGD と並んで実用的にも DNN の学習で非常によく使用されている．Adam は Momentum と RMSprop を組み合わせたような手法となっており，勾配の一次モーメントと二次モーメント両方の指数移動平均を使用して重みパラメータを更新する．

$$\begin{aligned} \boldsymbol{g}_t &= \frac{1}{M}\sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}_t} l(\boldsymbol{\theta}_t; \boldsymbol{x}_m, \boldsymbol{y}_m) \\ \boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\ \boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\boldsymbol{v}_t + \epsilon}} \odot \boldsymbol{m}_t \end{aligned} \tag{3.11}$$

β_1, β_2 は指数移動平均の重みを表すハイパーパラメータであり，$\beta_1 \in [0,1), \beta_2 \in [0,1)$ である．

3.2.4 適応的最適化手法の汎化性

適応的最適化手法を使用する主なメリットは，過去の勾配情報に基づきより広い重みパラメータ空間の情報を活用できることにある．損失関数における過去の勾配の一次モーメント，二次モーメントを考慮することでより大域的に損失関数を評価できる可能性が高くなる．一般に高次元の重みパラメータ空間において，勾配に基づく最適化手法では鞍点，あるいは局所最適解に陥る可能性が大きい．現在のところ適応的最適化手法に対する理論的な分析はあまり進んでいないが，様々な深層学習タスクで有効なため人気がある．Wilson らは単純な二値分類タスクにおいて，適応的最適化手法には限界があることを示した [41]. また彼らは画像分類タスクにおいて，SGD によって最終的に学習される重みパラメータの

方が適応的最適化手法よりも汎化することを実験的に示した．本研究でも，彼らの指摘に基づき画像分類タスクに焦点を置いている．Zhou ら（2020）は SGD と Adam の汎化性の違いを，Radon measure に基づく平坦性の観点から理論的に解析した [46]．彼らの研究では SGD と Adam におけるノイズ構造の違いを Lévy-driven 確率微分方程式を用いて定式化し，Adam のノイズ分布が SGD のノイズ分布に比べてなだらかになることを示した．本研究の結果は，ヘッセ行列の固有値分布を用いて実験的に彼らの結果を支持している．

第4章　分析手法

本研究の実験部分で用いる分析手法に関して詳しくレビューを行う．

4.1　ヘッセ行列を用いた損失関数形状の分析

本研究は異なる最適化手法によって学習されたDNNの重みパラメータの違いを比較することが目的である．どのような最適化手法を用いても，学習後の重みパラメータに関する損失関数の勾配のノルムはほぼ0に収束する．従って学習後の重みパラメータ付近における損失関数の局所形状を評価するには，勾配以外の評価基準が必要になる．その評価基準として，本研究では損失関数をテイラー展開した場合の二次の項の情報を含むヘッセ行列を用いることにする．

4.1.1　ヘッセ行列の固有値による曲がり具合の評価

学習後の重みパラメータを $\boldsymbol{\theta}_0 \in \mathbb{R}^D$ とする．$\boldsymbol{\theta}_0$ に摂動ベクトル $\delta\boldsymbol{\theta} \in \mathbb{R}^D, \|\delta\boldsymbol{\theta}\| = 1$ を加えた時の損失関数の変化 δL は，二次の項までのテイラー展開によって以下で表される（$\|\delta\boldsymbol{\theta}\| = 1$ は摂動ベクトルに対する規格化条件）．

$$\begin{aligned} \delta L &= L(\boldsymbol{\theta}_0 + \delta\boldsymbol{\theta}) - L(\boldsymbol{\theta}_0) \\ &\approx \nabla L(\boldsymbol{\theta}_0)^T \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T \boldsymbol{H}(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta} \end{aligned} \tag{4.1}$$

ここで，学習後の重みパラメータに関する損失関数の勾配がほぼ $\mathbf{0}$ と仮定すると，δL は $\delta L \approx \frac{1}{2}\delta\boldsymbol{\theta}^T H(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta}$ となる．今，損失関数の変化 δL を最大にする摂動ベクトルを求める最適化問題を考える．

$$\max_{\delta\boldsymbol{\theta}} \frac{1}{2}\delta\boldsymbol{\theta}^T \boldsymbol{H}(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta}, \quad \text{s.t.} \quad \delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta} = 1 \tag{4.2}$$

等式制約付きの最適化問題（4.2）をラグランジュの未定乗数法で解く．λ をラグランジュ未定乗数とすると，ラグランジュ関数 $\mathcal{L}(\delta\boldsymbol{\theta}, \lambda)$ は以下で定義される．

$$\mathcal{L}(\delta\boldsymbol{\theta}, \lambda) = \frac{1}{2}\delta\boldsymbol{\theta}^T \boldsymbol{H}(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta} - \lambda(\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta} - 1) \tag{4.3}$$

式（4.3）を $\delta\boldsymbol{\theta}$ に関して微分して0とおくことで以下を得る．

$$\boldsymbol{H}(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta} = \lambda\delta\boldsymbol{\theta} \tag{4.4}$$

式（4.4）は未定乗数 λ がヘッセ行列 $\boldsymbol{H}(\boldsymbol{\theta}_0)$ の固有値であり，摂動ベクトル $\delta\boldsymbol{\theta}$ が $\boldsymbol{H}(\boldsymbol{\theta}_0)$ の固有ベクトルであることを意味する．$\boldsymbol{H}(\boldsymbol{\theta}_0)$ の規格化された固有ベクトルを $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_D$，固有値を $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_D|$ とすると，$\delta\boldsymbol{\theta}$ に $\boldsymbol{u}_i$ を代入し，$\boldsymbol{u}_i$ に関する規格化条件を利用することで以下を得る．

$$\delta L \approx \frac{1}{2}\boldsymbol{u}_i^T \boldsymbol{H}(\boldsymbol{\theta}_0)\boldsymbol{u}_i = \frac{1}{2}\boldsymbol{u}_i^T \lambda_i \boldsymbol{u}_i = \frac{1}{2}\lambda_i \tag{4.5}$$

式（4.5）が最大となるのは $\lambda_i = \lambda_1$，つまり摂動ベクトル $\delta\boldsymbol{\theta}$ として絶対値最大の固有値 λ_1 に対応する固有ベクトル $\boldsymbol{u}_i$ を選んだときである．同様に $\boldsymbol{u}_1$ の補空間においてラグランジュ未定乗数法を行うことで $\boldsymbol{u}_2$ が損失関数の変化 δL を最大化することが示せる．

式（4.5）よりヘッセ行列の固有値の絶対値が大きいほど，重みパラメータに対して固有ベクトルを加えた時の損失関数の変化が大きくなることから，ヘッセ行列の固有値を損失関数における「曲がり具合」の評価指標として用いる．またヘッセ行列は対称行列のため，固有値は実数である．固有値が正の場合は式（4.5）より $\delta L > 0$，すなわち固有ベクトル方向への摂動は損失関数を増加させる．逆に固有値が負の場合は固有ベクトル方向への摂動は損失関数を減少させる．

4.1.2 最終層におけるヘッセ行列の固有値分布の分析

本研究では，ニューラルネットワークの最終層の重みパラメータのみを用いたヘッセ行列の固有値分布の分析も行った．実用的な DNN モデルでは高速な Hessian vector product 計算を使用しても，膨大なメモリ容量と計算時間を必要とする．そこで，本研究では最終層の重みパラメータのみを用いることで大幅なパラメータ次元数の削減を行った．

最終層のみに着目した理由としては，LeCun ら（1998）や Yao ら（2018, 2020）の行った研究で出力に近い層における重みパラメータほど二次微分の値が相対的に大きいという実験的事実に基づいている [23, 44, 43]．つまりヘッセ行列の固有値分布を用いて損失関数形状を相対的に比較・評価する場合，全ての重みパラメータに着目する必要はなく，最終層の重みパラメータにのみ着目すれば十分ではないかという考えに基づいている．

K 層の DNN の場合，式（2.1）より訓練損失の最終層の重みパラメータに関するヘッセ行列 $\nabla^2_{\boldsymbol{W}^{(K-1)}} L(\boldsymbol{\theta})$ は出力層の前の重みパラメータ $\boldsymbol{W}^{(K-1)}$ を用いて以下で定義される．

$$\nabla^2_{\boldsymbol{W}^{(K-1)}} L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial^2}{\partial \boldsymbol{W}^{(K-1)} \partial (\boldsymbol{W}^{(K-1)})^T} l(\boldsymbol{\theta}; x_i, y_i) \tag{4.6}$$

最終層における重みパラメータ次元数を $d^{\text{penultimate}} = d^{(K)} \times d^{(K-1)}$ とすると最終層におけるヘッセ行列の次元数は $d^{\text{penultimate}} \times d^{\text{penultimate}}$ になるため，全重みパラメータ次元数 $D = \sum_{k=0}^{K-1} d^{(k+1)} \times d^{(k)}$ を用いた $D \times D$ 次元のヘッセ行列に比べて大幅に小さくなる．

以下本論文では，モデルの全重みパラメータを用いて計算されたヘッセ行列の固有値分布のことを Full Hessian Spectrum と呼び，最終層の重みパラメータのみを用いて計算されたヘッセ行列の固有値分布のことを Penultimate Hessian Spectrum と呼ぶ．

4.2 竹内情報量規準を用いた汎化性の分析

ヘッセ行列を用いて損失関数形状を分析することで，重みパラメータ空間における訓練損失とテスト損失の不一致度を見ることができる．しかし一般に，汎化ギャップは入力空間における訓練損失とテスト損失における不一致度を測るために用いる．そこで入力の変化に対する勾配の感度を表す勾配共分散行列を用いて定義される竹内情報量基準を導入する．

竹内情報量規準（Takeuchi Information Criterion，以下 TIC）はモデル選択に用いられる情報量規準の 1 つで，竹内（1976）によって与えられた [38]．TIC は以下で定義される．

$$\hat{\mathcal{G}} = tr(\boldsymbol{H}(\boldsymbol{\theta}^*)^{-1} \boldsymbol{C}(\boldsymbol{\theta}^*)) \tag{4.7}$$

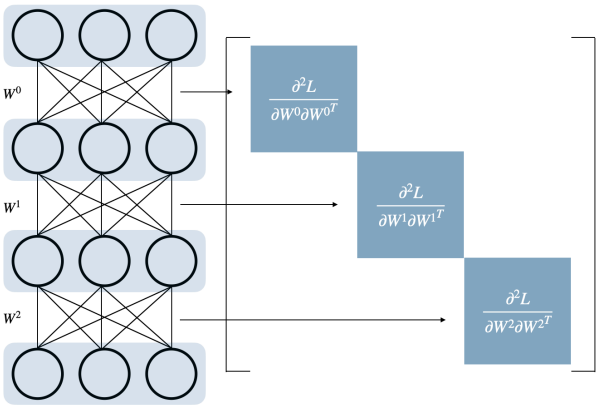


図 4.1: 各層ごとの重みパラメータに関するヘッセ行列．ここでは，p を ζ 層目におけるノード、q を ξ 層目におけるノードとして，$[\boldsymbol{H}(\boldsymbol{\theta})]_{(p,q)} = 0$ とみなすことで，ブロック対角行列を用いて訓練損失の重みパラメータに関するヘッセ行列の近似を行っている．この図の場合，最終層のみの重みパラメータに関するヘッセ行列は右下のブロックに相当する．

ここで，$\boldsymbol{\theta}^*$ は期待損失（2.10）を最小化する重みパラメータ，すなわち真の重みパラメータとする．

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \hat{L}(\boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} E_{p(\boldsymbol{x},\boldsymbol{y})}\left[l(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}))\right] \end{aligned} \tag{4.8}$$

真の重みパラメータに関する期待損失のヘッセ行列 $\boldsymbol{H}(\boldsymbol{\theta}^*)$ と，真の重みパラメータに関する期待損失の勾配共分散行列 $\boldsymbol{C}(\boldsymbol{\theta}^*)$ の各要素は以下で定義される．

$$[\boldsymbol{H}(\boldsymbol{\theta}^*)]_{(m,n)} = \frac{\partial^2 E_{p(\boldsymbol{x},\boldsymbol{y})}\left[l(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{y}))\right]}{\partial[\boldsymbol{\theta}^*]_m \partial[\boldsymbol{\theta}^*]_n} \tag{4.9}$$

$$[\boldsymbol{C}(\boldsymbol{\theta}^*)]_{(m,n)} = \frac{\partial E_{p(\boldsymbol{x},\boldsymbol{y})}\left[l(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{y}))\right]}{\partial[\boldsymbol{\theta}^*]_m} \frac{\partial E_{p(\boldsymbol{x},\boldsymbol{y})}\left[l(\boldsymbol{\theta}^*;\boldsymbol{x},\boldsymbol{y}))\right]}{\partial[\boldsymbol{\theta}^*]_n} \tag{4.10}$$

しかしながら，一般に真の重みパラメータ $\boldsymbol{\theta}^*$ と真のデータ生成分布 $p(\boldsymbol{x},\boldsymbol{y})$ は分からないため，代わりに学習後の重みパラメータ $\boldsymbol{\theta}_0$ とテストデータ集合 $\mathbb{D}^{\text{test}}$ を用いて以下のように計算される．

$$\check{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^{N^{\text{test}}} tr(\boldsymbol{H}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})^{-1}\boldsymbol{C}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})) \tag{4.11}$$

ここで，テストデータを用いたヘッセ行列 $\boldsymbol{H}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})$ と，テストデータを用いた勾配共分散行列 $\boldsymbol{C}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})$ の各要素は以下で定義される．

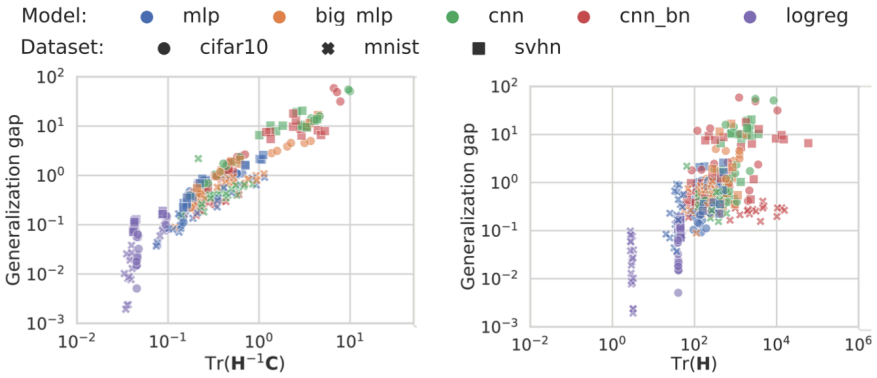
$$\left[\boldsymbol{H}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})\right]_{(m,n)} = \frac{\partial^2 l(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})}{\partial[\boldsymbol{\theta}_0]_m \partial[\boldsymbol{\theta}_0]_n}, \quad m = 1,\cdots,D \quad n = 1,\cdots,D$$

$$\left[\boldsymbol{C}(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})\right]_{(m,n)} = \frac{\partial l(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})}{\partial\left[\boldsymbol{\theta}_0\right]_m} \frac{\partial l(\boldsymbol{\theta}_0;\boldsymbol{x}_i^{\text{test}},\boldsymbol{y}_i^{\text{test}})}{\partial\left[\boldsymbol{\theta}_0\right]_n}, \quad m = 1,\cdots,D \quad n = 1,\cdots,D$$

TIC は重みパラメータの事後分布が正規分布で近似可能（正則モデル）なときに，漸近的に汎化ギャップと一致するような値である．ニューラルネットワークは正則モデルではないため，一般に重みパラメータの事後分布を正規分布で近似することはできない．しかしながら近年，学習後のニューラルネットワークにおける TIC を見ることで汎化ギャップと高い相関を持つことが実験的に確認されている [40]．つまりニューラルネットワークの重みパラメータの事後分布が正規分布で近似できることを数理的に保証するのは難しいが，実験的には正規分布で十分に近似できる可能性が高い．

また，TIC は式（4.11）の通りテストデータに対して計算される値である．それに対して汎化ギャップ（式（2.12））の計算には訓練データとテストデータの両方が必要となる．つまり学習後のモデルの汎化性を評価する際に，TIC は汎化ギャップよりも少ない情報量で求めることが可能である．

本研究では，異なる最適化手法によって学習されたニューラルネットワークに対して TIC を計算することでそれぞれの汎化性について確認する．



(a) Gap vs. TIC.　　(b) Gap vs. flatness.

図 4.2: 様々なニューラルネットワークモデルにおける TIC と汎化ギャップとの高い相関を表す（(a)：TIC vs 汎化ギャップ，(b)：flatness（ヘッセ行列のトレース） vs 汎化ギャップ．図は Thomas ら（2020）[40] から引用）．図 (a) 中の $Tr(\boldsymbol{H}^{-1}\boldsymbol{C})$ は式（4.11）を計算して求めている．図 (b) ではヘッセ行列のトレース，つまり全固有値の和を用いて flatness を定義している．ヘッセ行列のトレースを用いた flatness では汎化ギャップを捉えきれないことがわかる．また，学習に用いるハイパーパラメータを変えた様々な結果をプロットしているため，点の数はモデルとデータの組み合わせ数よりも多くなる．

4.3　敵対的摂動を用いた頑健性の分析

敵対的摂動（Adversarial perturbation）とは，学習済みの機械学習モデルの出力を意図的に誤らせるために入力に加えるノイズ（摂動）のことである．元の入力 $\boldsymbol{x}_{\text{origin}}$ に対して敵対的摂動 η を加えた入力を敵対的サンプル（Adversarial Examples）$\boldsymbol{x}_{\text{adv}}$ と呼ぶ．

$$\boldsymbol{x}_{\text{adv}} = \boldsymbol{x}_{\text{origin}} + \epsilon\boldsymbol{\eta} \tag{4.12}$$

多くの場合，人間の目では元の入力 $\boldsymbol{x}_{\text{origin}}$ と敵対的サンプル $\boldsymbol{x}_{\text{adv}}$ との区別を付けることが難しいため，機械学習モデルを運用する際の脆弱性として問題となる．本研究では敵対

的摂動の生成アルゴリズムとして，Goodfellow ら（2014）が提案した Fast Gradient Sign Method (FGSM[11]) を使用して，異なる最適化手法によって学習された DNN モデルに対する頑健性について確認した．

Fast Gradient Sign Method

FGSM はホワイトボックス攻撃手法の 1 つである．ホワイトボックス攻撃とは，攻撃者が学習済みの DNN モデルの情報に完全にアクセスできるという前提のもとでの攻撃手法である．

FGSM では，敵対的摂動を以下の式で生成する．$\boldsymbol{y}_{\text{origin}}$ は元の入力 $\boldsymbol{x}_{\text{origin}}$ に対するラベルを表す．

$$\boldsymbol{\eta} = \text{sign}(\nabla_{\boldsymbol{x}_{\text{origin}}} L(\boldsymbol{\theta}; \boldsymbol{x}_{\text{origin}}, \boldsymbol{y}_{\text{origin}})) \tag{4.13}$$

式（4.13）からわかるように，敵対的摂動は元の入力に対する損失関数の勾配の符号情報を用いる．例えば入力を画像とすると，画像の各ピクセルをどちらの方向に動かせば損失が大きくなるかを表している．損失が大きい場合，誤分類する確率が高いためモデルが騙されやすい入力となる．DNN の場合モデルの重みパラメータは一定のまま，誤差逆伝播法によって入力に関する勾配を求めることができるため，FGSM は高速に動作する．

摂動は符号情報のみなので，式（4.12）における ϵ を変化させることで敵対的摂動の強弱を調整する．ϵ が大きいほどモデルが騙されやすくなるが，摂動の影響もわかりやすくなる．本研究では，各最適化手法によって学習されたモデルが ϵ の変化によってどの程度騙されるようになるかを比較する．

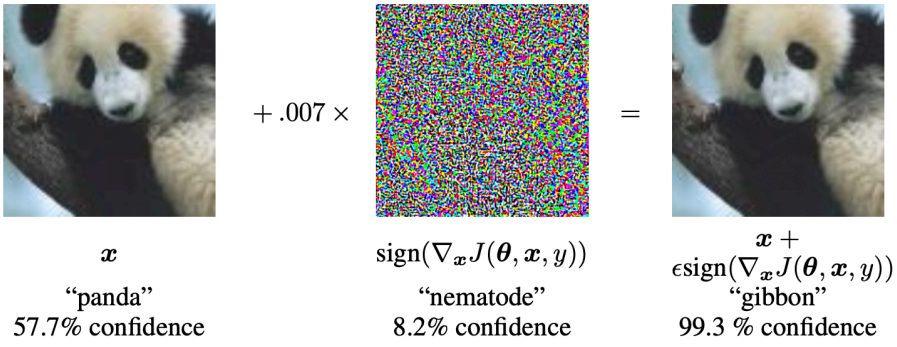


図 4.3: パンダの画像に敵対的摂動を乗せることで高い確信度でテナガザル（"gibbon"）と予測させる（図は Goodfellow ら（2014）[11] から引用）．人間の目には分からない摂動なのでこのような入力を除去するのは難しい．

第5章　実験

5.1　実験環境

5.1.1　深層学習フレームワーク

深層学習の研究が活発になると同時に，ニューラルネットワークを用いた学習，予測を行うためのフレームワークやベンチマークとなるデータセットが多数公開されている．代表的な深層学習フレームワークとしては，Pytorch[30] や Tensorflow[1]，などが挙げられる．本研究では基本的に Pytorch を用いているが，一部 Tensorflow も併用して実験を行った．

5.1.2　モデルとデータセット

本研究では 4 章でレビューした 3 つの分析を行うための DNN モデルとして，Yao らの研究（2018）における C3[44] と LeNet[22] を用いる（表 5.1）．先行研究（[44, 41, 40]）に倣って画像分類タスクにおける分析に焦点を当てた．また，最終層のみのパラメータに関するヘッセ行列の固有値分析を行うモデルとして，ResNet-34[14] と DenseNet-121[16] を用いる．どちらのモデルもプラクティカルな問題設定で高い認識性能を出すモデルとして，広く利用されている．Dinh ら（2017）の指摘する Reparametrization に対するヘッセ行列の固有値の脆弱性 [8] への対策として，Batch-Normalization 層は使用しない．

データセットとしては画像分類のベンチマークとして広く利用されている Fashion-MNIST[42] と CIFAR-10[19] を用いる．

- Fashion-MNIST: Fashion-MNIST は Zalando（https://www.zalando.com）内のファッション画像分類のためのデータセットで，60,000 の訓練データと 10,000 のテストデータから構成される．各サンプルは 28x28 のグレースケール画像で，10 クラスのラベルに分類される．
- CIFAR-10: CIFAR-10 データセットは一般画像分類のためのデータセットで，50,000 の訓練データと 10,000 のテストデータから構成される．各サンプルは 32x32 のカラー画像で，10 クラスのラベルに分類される．

5.1.3　ヘッセ行列の計算ライブラリ

上記で挙げたような深層学習フレームワークは，基本的にニューラルネットワークによる学習，予測をするという点で効率的な実装がなされている．したがって，ニューラルネットワークモデルの勾配情報には直感的なインターフェースでアクセス可能である．しかしながら，本研究では勾配だけでなくヘッセ行列の計算にも興味がある．これらのフレームワークでは学習時には計算効率化のため，重みパラメータ更新で勾配情報を一度用いると，その後は必要ないため捨ててしまう．誤差逆伝播法を利用して Hessian vector product を計算する場合，勾配情報が複数回必要になる．そのため近年では，明示的にヘッセ行列に関す

model	structure	num of params
C3（for CIFAR-10）	Conv2d(3,3,64)-Conv2d(3,3,64)-MaxPool2d -Conv2d(3,3,128)-Conv2d(3,3,128)-MaxPool2d -FC(256)-FC(256)-Softmax(10)	1,147,978
LeNet（for Fashion-MNIST）	Conv2d(5,5,6)-Conv2d(5,5,16)-MaxPool2d -FC(120)-FC(80)-Softmax(10)	43,902

表 5.1: 実験で使用した比較的小さな重みパラメータ数の DNN モデルの構造と，モデルの持つ重みパラメータの総数.

る情報を計算・保持するためのライブラリがいくつか研究開発されている [6, 43, 12]．本研究では，ヘッセ行列の固有値に関する計算には PyHessian[43] を用いており，勾配共分散行列の計算には BackPACK[6] と呼ばれるライブラリを用いた.

5.2 実験結果

基本的な実験の流れは以下の通りである.

1. DNN モデルの学習を行い，最終エポックとバリデーションデータに対するベストエポックのモデルの重みパラメータを保存する.
2. 最終エポック or ベストエポックにおけるモデルの重みパラメータを読み込み，4 章の各分析を行う.

ここでバリデーションデータとは訓練データを適当な比率で分割して得られる，学習時に評価用としてのみ用いるデータ集合のことである．またバリデーションデータに対するベストエポックとは，バリデーションデータによって計算される Accuracy（与えられた全データのうち，モデルが正解クラスを正しく予測した割合）が最も高いエポックのことである．本研究では，CIFAR-10，Fashion-MNIST 共に全訓練データの 20% をバリデーションデータとして用いた．以下では SGD，RMSprop，Adam の各最適化手法によって学習されたモデルにおける重みパラメータのことをそれぞれ $\boldsymbol{\theta}_{\mathrm{SGD}}$，$\boldsymbol{\theta}_{\mathrm{RMSprop}}$，$\boldsymbol{\theta}_{\mathrm{Adam}}$ と呼ぶことにする．全ての実験で DNN 学習のランダム性を考慮して，重みパラメータの初期値が異なる学習結果の平均した値を用いている．また最終エポックは C3 on CIFAR-10 で 50，LeNet on Fashion-MNIST で 100 とした．ベストエポックは各最適化手法で異なり，4 回の試行の平均ベストエポックを表 5.3 に示す.

学習に用いた最適化手法ごとのハイパーパラメータを表 5.2 に示す．これらのハイパーパラメータは使用したライブラリのデフォルトの値に基づいている．式（2.17），(3.10），(3.11）より，SGD のハイパーパラメータは学習率 η，RMSprop のハイパーパラメータは学習率 η，勾配の二次モーメントにおける指数移動平均の重み β，ゼロ除算を防ぐ ϵ，Adam のハイパーパラメータは学習率 η，勾配の一次モーメント，二次モーメントにおける指数移動平均の重み β_1, β_2，ゼロ除算を防ぐ ϵ である.

また，バッチサイズ M は汎化性に影響のあるハイパーパラメータとして知られているため，あるモデルを学習するときには全ての最適化手法で同一の値を用いている．比較的小さな DNN モデルに関しては全ての手法で $M = 128$ とし，プラクティカルな DNN モデルに関しては全ての手法で $M = 256$ としている.

optimizer	hyper parameters
SGD	η: 0.1
RMSprop	η: 0.01, β: 0.99, ϵ=1e-8
Adam	η: 0.001, β_1: 0.9, β_2: 0.999, ϵ=1e-8

表 5.2: 学習に用いた最適化手法ごとのハイパーパラメータ.

model\optimizer	SGD	RMSprop	Adam
C3 on CIFAR-10	46	44	38
LeNet on Fashion-MNIST	32	26	32

表 5.3: 各最適化手法における平均ベストエポック.

5.2.1 比較的小さな DNN モデルでの結果

まず，重みパラメータの次元数が比較的小さな DNN モデル（表 5.1）での実験結果を示す．C3 on CIFAR-10 に対して SGD，RMSprop，Adam を用いてそれぞれ学習し，4 章の各分析を行った．また，LeNet on Fashion-MNIST に対しては Full Hessian Spectrum と TIC の比較を行った．各モデル，データセットでも FGSM，TIC の計算にはベストエポックの重みパラメータのみを用いている．

Full Hessian Spectrum の比較

最終エポックにおけるモデルパラメータと，ベストエポックにおけるモデルパラメータを用いて Full Hessian Spectrum を計算，比較した結果を図 5.1（C3 on CIFAR-10）と図 5.2（LeNet on Fashion-MNIST）に示す．どちらの図においても，横軸は固有値，縦軸は固有値密度（式（3.3））の SLQ アルゴリズムによる推定値を表している．図 5.1 と図 5.2 からわかるように，どちらのエポックにおいても $\boldsymbol{\theta}_{\mathrm{SGD}}$ に比べて，$\boldsymbol{\theta}_{\mathrm{RMSprop}}$，$\boldsymbol{\theta}_{\mathrm{Adam}}$ におけるヘッセ行列の固有値全体が絶対値の大きな値に分布している．つまり適応的最適化手法の方がモデルの重みパラメータ空間において，SGD に比べて相対的に急峻な損失形状に到達しているということがわかる．

pseudo TIC の比較

ここでは各最適化手法で学習されたベストエポックにおけるモデルの全重みパラメータを用いて TIC を計算し，比較する．TIC は式（4.7）の通り，ヘッセ行列の逆行列 $\boldsymbol{H}^{-1}$ と勾配共分散行列 $\boldsymbol{C}$ の積のトレースが必要になる．ヘッセ行列の成分数は重みパラメータ次元数の二乗の個数になり，DNN においてその逆行列は現実的に計算不可能である．そこで，Thomas ら [40] は式（4.7）の近似として $tr(\boldsymbol{C})/tr(\boldsymbol{H})$ で TIC を計算した．$tr(\boldsymbol{H})$ は 3 章で説明した Hutchinson Method を用いることで現実的に計算可能であり，$tr(\boldsymbol{C})$ は勾配ベクトルの各要素を二乗した値の和と等しいのでこちらも現実的に計算可能である．本研究でも Thomas らによる近似式を用いて TIC の計算を行った．以下，Exact な TIC とは区別するために本研究では pseudo TIC（擬似 TIC）と呼ぶ．また本来 TIC は期待損失を最小にする真の重みパラメータ $\boldsymbol{\theta}^*$ に対して計算される値のため，ここではその近似として学習中の

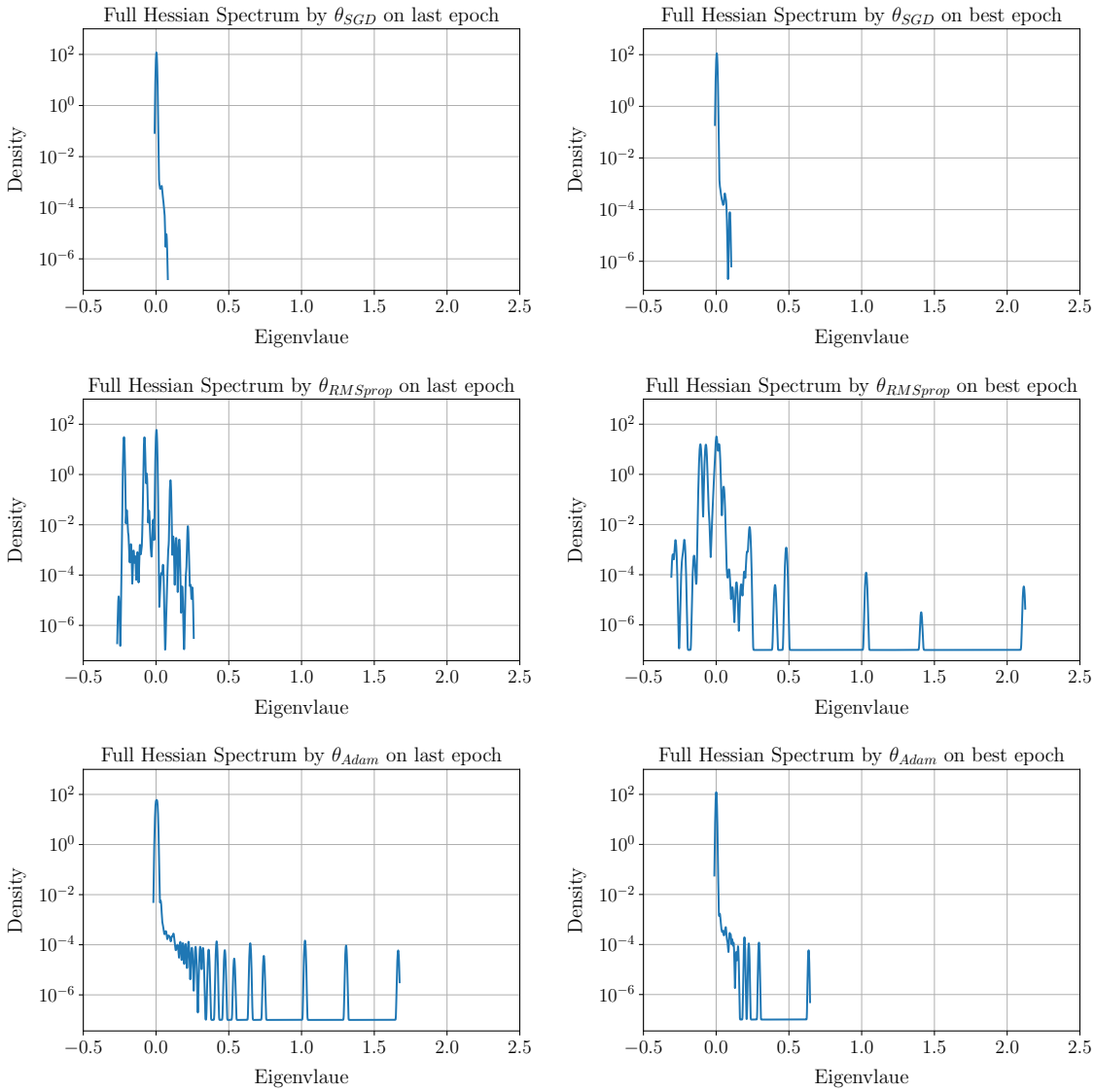


図 5.1: C3 on CIFAR-10 における SGD，RMSprop，Adam の各最適化手法によって学習された重みパラメータに関する Full Hessian Spectrum（左：最終エポック，右：ベストエポック）．横軸の範囲外には固有値は存在しない．結果は 4 回の試行の平均値となっている．

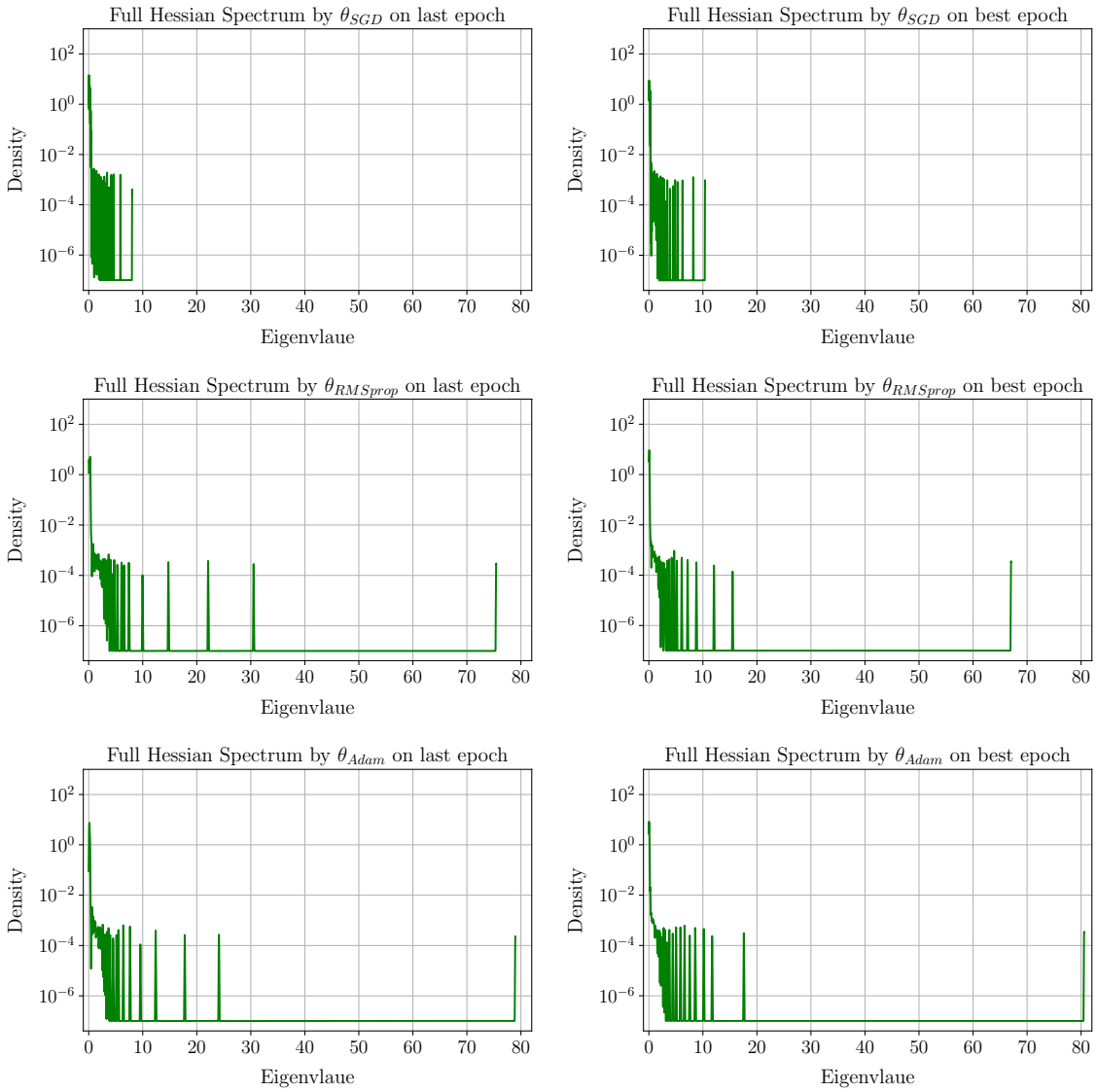


図 5.2: LeNet on Fashion-MNIST における SGD，RMSprop，Adam の各最適化手法によって学習されたパラメータに関する Full Hessian Spectrum（左：最終エポック，右：ベストエポック）．横軸の範囲外には固有値は存在しない．結果は 4 回の試行の平均値となっている．

pseudo TIC	$\boldsymbol{\theta}_{\mathrm{SGD}}$	$\boldsymbol{\theta}_{\mathrm{RMSprop}}$	$\boldsymbol{\theta}_{\mathrm{Adam}}$
$\tilde{\mathcal{G}}$（$\times 10^{-4}$）	**0.07**	3.63	1.06

(a) C3 on CIFAR-10

pseudo TIC	$\boldsymbol{\theta}_{\mathrm{SGD}}$	$\boldsymbol{\theta}_{\mathrm{RMSprop}}$	$\boldsymbol{\theta}_{\mathrm{Adam}}$
$\tilde{\mathcal{G}}$（$\times 10^{-1}$）	**6.24**	7.34	7.74

(b) LeNet on Fashion-MNIST

表 5.4: SGD，RMSprop，Adam の各最適化手法によって学習されたベストエポックにおける pseudo TIC．結果は 4 回の試行の平均値となっている．

ベストエポック，つまりバリデーションデータにおける最適な重みパラメータ $\boldsymbol{\theta}^{\mathrm{best}}$ に対して計算を行った．また式（4.11）と同様に，pseudo TIC $\tilde{\mathcal{G}}$ はテストデータ集合 $\mathbb{D}^{\mathrm{test}}$ を用いて以下で計算される．

$$\tilde{\mathcal{G}}(\boldsymbol{\theta}^{\mathrm{best}}) = \frac{1}{N^{\mathrm{test}}} \sum_{i=1}^{N^{\mathrm{test}}} \frac{tr(\boldsymbol{C}(\boldsymbol{\theta}^{\mathrm{best}}; \boldsymbol{x}_i^{\mathrm{test}}, \boldsymbol{y}_i^{\mathrm{test}}))}{tr(\boldsymbol{H}(\boldsymbol{\theta}^{\mathrm{best}}; \boldsymbol{x}_i^{\mathrm{test}}, \boldsymbol{y}_i^{\mathrm{test}}))} \tag{5.1}$$

pseudo TIC の比較結果を表 5.4 に示す．C3 on CIFAR-10 と LeNet on Fashion-MNIST のどちらの実験設定においても，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{SGD}})$ が最も小さい値を取っている．

FGSM に対する頑健性比較

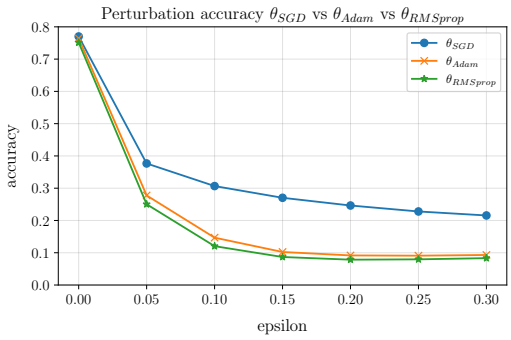
ここでは C3 on CIFAR-10 に対して FGSM を用いて生成した敵対的摂動 $\boldsymbol{\eta}$ を CIFAR-10 のテストデータ集合に対して適用する．敵対的摂動を加えたテストデータ集合を，敵対的テストデータ集合 $\mathbb{D}^{\mathrm{adv}} = \{(\boldsymbol{x}_i^{\mathrm{test}} + \epsilon\boldsymbol{\eta}, \boldsymbol{y}_i^{\mathrm{test}} + \epsilon\boldsymbol{\eta})\}_{i=1}^{N^{\mathrm{test}}}$ として定義する．ϵ を 0.05 から 0.3 まで 0.05 刻みで変化させたときの $\mathbb{D}^{\mathrm{adv}}$ に対する，$\boldsymbol{\theta}_{\mathrm{SGD}}$ と $\boldsymbol{\theta}_{\mathrm{Adam}}$ と $\boldsymbol{\theta}_{\mathrm{RMSprop}}$ の Accuracy に与える影響を比較する（図 5.3a）．$\mathbb{D}^{\mathrm{adv}}$ に対するモデルの Accuracy のことを Perturbation accuracy と呼ぶ．

図 5.3a からわかるように，Adam，RMSprop で学習されたモデルの方が SGD で学習されたモデルに比べて FGSM によって生成された摂動に対する Accuracy の劣化の影響が大きい．Adam，RMSprop 間では性能劣化の変化度合いはほぼ変わらない．つまり C3 on CIFAR-10 においては，$\boldsymbol{\theta}_{\mathrm{Adam}}$，$\boldsymbol{\theta}_{\mathrm{RMSprop}}$ の方が $\boldsymbol{\theta}_{\mathrm{SGD}}$ よりも入力への敵対的摂動に対する頑健性の面で悪化しやすいことがわかる．

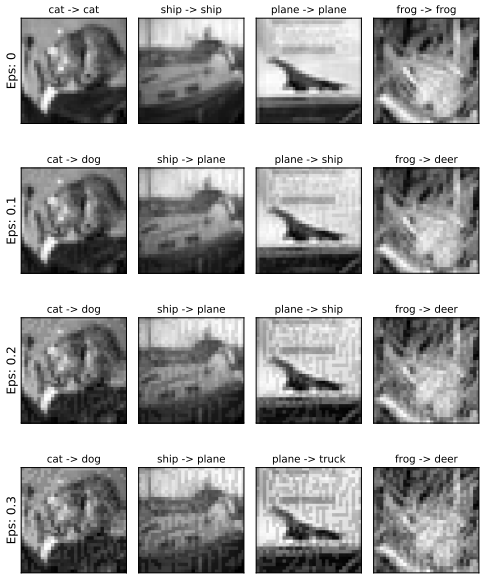
また，図 5.3b に実際に生成された敵対的サンプルの例を示す．ϵ が大きいほど摂動の影響が大きくなっていることがわかる．

5.2.2 プラクティカルな DNN モデルにおける結果

最後に，プラィティカルな問題設定でもよく利用されている DNN モデル（表 5.5）を用いて行った実験結果を示す．どちらのモデルにおいても SGD，Adam ともに 100 エポック学習を行った後の重みパラメータを用いた．表 5.5 に示すように最終層の重みパラメータのみを用いることで，全重みパラメータの次元数から，ResNet-34 については約 1/2300，DenseNet-121 については約 1/800 の次元数まで削減している．



(a) FGSM Attack（式（4.13））における ϵ が認識精度の劣化に与える影響．C3 on CIFAR-10 における $\boldsymbol{\theta}_{\mathrm{SGD}}$ と $\boldsymbol{\theta}_{\mathrm{Adam}}$，$\boldsymbol{\theta}_{\mathrm{RMSprop}}$ を比較した．



(b) 実際に生成された敵対的サンプルの例．各図の上部におけるラベルは，(正解クラス)→(予測クラス) を表している．摂動の影響がわかりやすいようにグレースケールで表示している．

図 5.3: ϵ の変化に対する Perturbation accuracy の変化と実際に生成された敵対的サンプルの例．

model	num of full-params	num of penultimate-params
ResNet-34	23,377,922	10,000
DenseNet-121	8,078,658	10,000

表 5.5: 実験で使用した ResNet-34 と DenseNet-121 の全重みパラメータの次元数と，最終層における重みパラメータの次元数．

Penultimate Hessian Spectrum の上位 20 固有値の比較

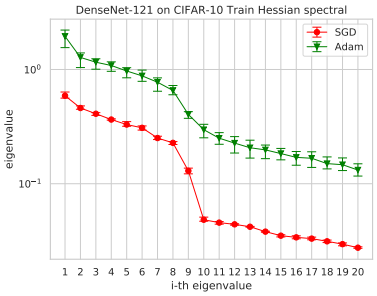
最終層の重みパラメータのみを用いたヘッセ行列の上位 20 固有値を比較する．上位 20 個のみを用いているのは，クラス分類の場合はヘッセ行列の outlier eigenvalues の個数が限られているという実験的観察 [34, 44] に基づく．図 5.4a, 5.4b から，プラクティカルな DNN においては $\boldsymbol{\theta}_{\text{Adam}}$ の方が，$\boldsymbol{\theta}_{\text{SGD}}$ に比べて最終層の重みパラメータに関するヘッセ行列の固有値の絶対値が大きいということがわかる．

また，最大固有値と第二固有値に対応する固有ベクトル（$\boldsymbol{u}_1, \boldsymbol{u}_2$）を用いて，重みパラメータ空間を $\boldsymbol{u}_1, \boldsymbol{u}_2$ が張る空間に射影し，損失関数の局所形状の可視化を行った（図 5.4c, 図 5.4d, 図 5.4e, 図 5.4f ）．図は学習済みモデルにおける最終層の重みパラメータを中心としたグリッドポイントにおける訓練損失を計算して描いた．つまり，最終層の重みパラメータに摂動ベクトルを加えたときの局所的な訓練損失形状を表している．摂動ベクトルは $\boldsymbol{u}_1, \boldsymbol{u}_2$ の重み付けされた和である．学習後の最終層の重みパラメータ $\boldsymbol{W}_0^{K-1} \in \mathbb{R}^{d^{(K)}} \times \mathbb{R}^{d^{(K-1)}}$ を重みパラメータベクトル $\boldsymbol{w}_0^{K-1} \in \mathbb{R}^{d^{(K)} \times d^{(K-1)}}$ に変換し、摂動ベクトルを加えた損失は次のように表せる．

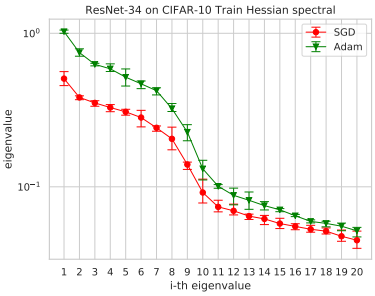
$$L(\boldsymbol{w}_0^{K-1} + a_1 \boldsymbol{u}_1 + a_2 \boldsymbol{u}_2) \tag{5.2}$$

ここで，a_1, a_2 はグリッドポイントを描画するために範囲を決めてインクリメントする値である．本実験では a_1, a_2 は -0.5 から 0.5 まで 0.05 刻みで変化させた．

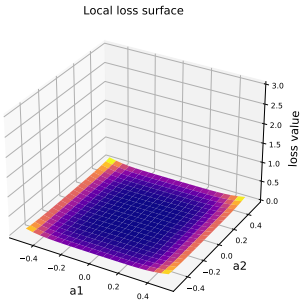
また図 5.5 と図 5.6 に各モデルにおける最終層の重みパラメータの要素分布と，摂動として加えた最大固有値に対応する固有ベクトルの要素分布を示す．これらの図から重みパラメータに対して十分小さな摂動ベクトルになっていることが確認できる．



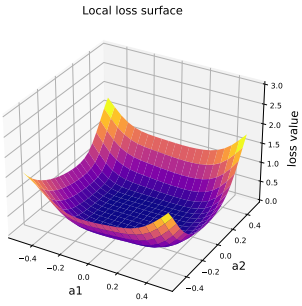
(a) DenseNet-121 on CIFAR-10 の上位 20 固有値.



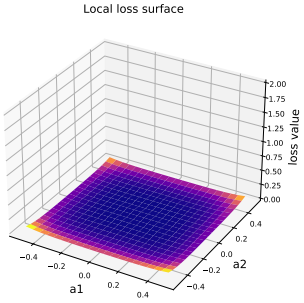
(b) ResNet-34 on CIFAR-10 の上位 20 固有値.



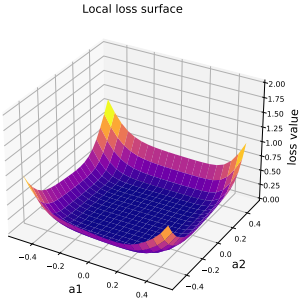
(c) DenseNet-121 の $\boldsymbol{\theta}_{\mathrm{SGD}}$ における損失関数形状



(d) DenseNet-121 の $\boldsymbol{\theta}_{\mathrm{Adam}}$ における損失関数形状

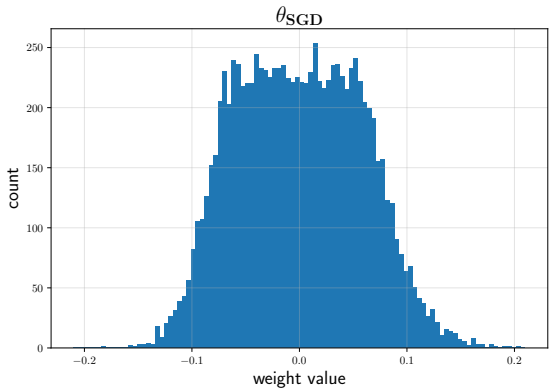


(e) ResNet-34 の $\boldsymbol{\theta}_{\mathrm{SGD}}$ における損失関数の局所形状

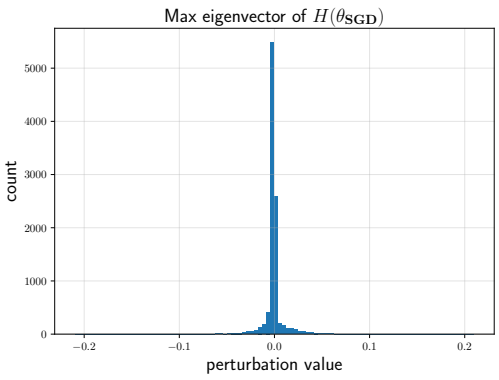


(f) ResNet-34 の $\boldsymbol{\theta}_{\mathrm{Adam}}$ における損失関数の局所形状

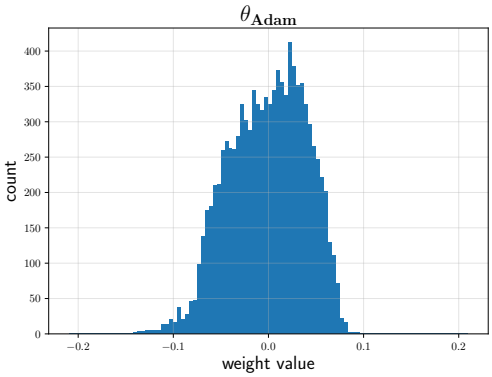
図 5.4: プラクティカルな DNN モデルに関して，SGD と Adam によって学習したモデルの訓練損失における Penultimate Hessian Spectrum（上位 20 固有値のみ）の比較と訓練損失の局所形状の可視化．結果は 4 回の試行の平均値となっており，Penultimate Hessian Spectrum のエラーバーは平均からの最大値と最小値との誤差を表す．また，局所形状の図における z 軸は訓練データに対して $K-1$ 層目における重みパラメータ以外は固定したときの $L(\boldsymbol{w}_0^{K-1} + a_1\boldsymbol{u}_1 + a_2\boldsymbol{u}_2)$ を計算した値である．



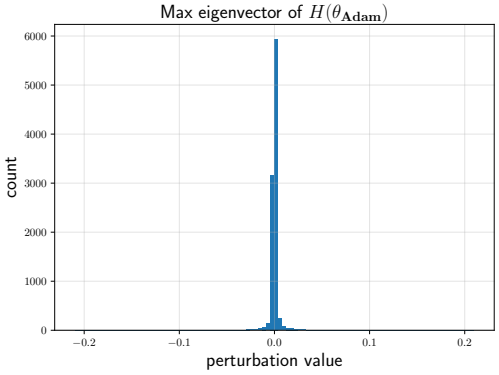
(a) $\boldsymbol{\theta}_{\mathrm{SGD}}$ における最終層の重みパラメータの要素分布



(b) $\boldsymbol{\theta}_{\mathrm{SGD}}$ における最終層のヘッセ行列の最大固有ベクトルの要素分布

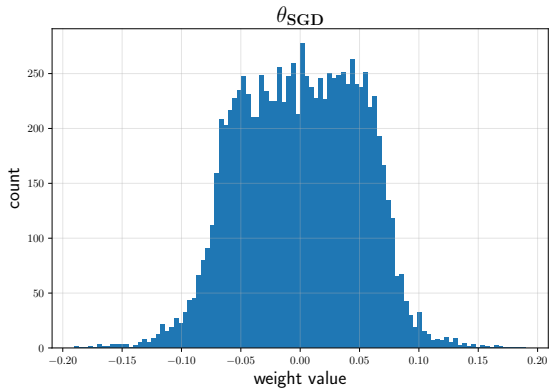


(c) $\boldsymbol{\theta}_{\mathrm{Adam}}$ における最終層の重みパラメータの要素分布

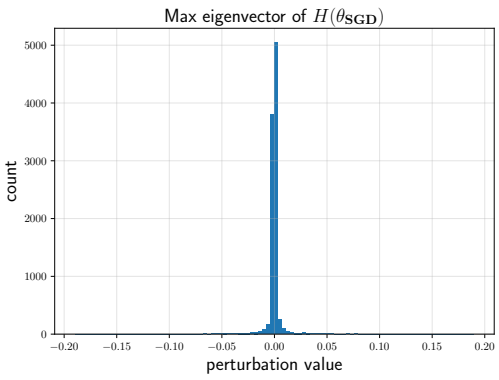


(d) $\boldsymbol{\theta}_{\mathrm{Adam}}$ における最終層のヘッセ行列の最大固有ベクトルの要素分布

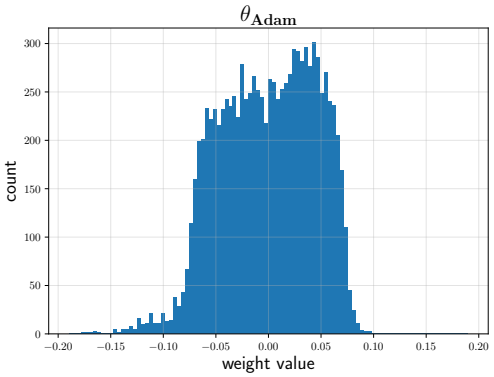
図 5.5: DenseNet-121 on CIFAR-10 における重みパラメータと摂動ベクトルの要素分布.



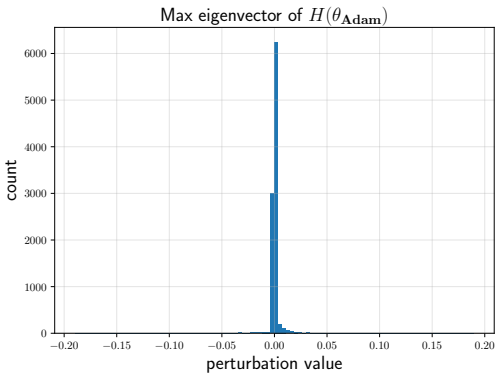
(a) $\boldsymbol{\theta}_{\mathrm{SGD}}$ における最終層の重みパラメータの要素分布



(b) $\boldsymbol{\theta}_{\mathrm{SGD}}$ における最終層のヘッセ行列の最大固有ベクトルの要素分布



(c) $\boldsymbol{\theta}_{\mathrm{Adam}}$ における最終層の重みパラメータの要素分布



(d) $\boldsymbol{\theta}_{\mathrm{Adam}}$ における最終層のヘッセ行列の最大固有ベクトルの要素分布

図 5.6: ResNet-34 on CIFAR-10 における重みパラメータと摂動ベクトルの要素分布.

第6章　結論

6.1　考察

本研究では CIFAR-10 と Fashion-MNIST の 2 種類の画像分類データセットに対して，異なる最適化手法によって学習された DNN モデルについて，大きく 3 つの分析を行い汎化性・頑健性に関する実験的な検証を行った．

まず，損失関数におけるヘッセ行列の固有値分布を調査することで学習後の DNN の重みパラメータ空間における平坦性について実験的に確認した．今回用いたデータセットとモデルにおいては，適応的最適化手法は SGD に比べて，ヘッセ行列の固有値の絶対値がより大きな重みパラメータを持つ，つまり相対的に急峻な損失形状へ収束することがわかった．これは，Zhou らの行った確率微分方程式を用いた連続的な重みパラメータに関する理論解析 [46] の結果を支持するものとなっている．

また，各最適化手法での学習後のベストエポックにおける pesudo TIC $\tilde{\mathcal{G}}$ を比較し，SGD に比べて適応的最適化手法で学習されたモデルの方が $\tilde{\mathcal{G}}$ の値が大きいことを確認した．具体的には C3 on CIFAR-10 において，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{SGD}}) = 0.07 \cdot 10^{-4}$，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{RMSprop}}) = 3.63 \cdot 10^{-4}$，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{Adam}}) = 1.06 \cdot 10^{-4}$ となり，LeNet on Fashion-MNIST において，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{SGD}}) = 6.24 \cdot 10^{-1}$，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{RMSprop}}) = 7.34 \cdot 10^{-1}$，$\tilde{\mathcal{G}}(\boldsymbol{\theta}_{\mathrm{Adam}}) = 7.74 \cdot 10^{-1}$ という結果となった．TIC の値は漸近的に汎化ギャップと一致する値であり，以上 2 つの結果より DNN を用いた画像分類タスクにおいて SGD に比べて適応的最適化手法で学習されたモデルの方が汎化性が確保されにくいことが示唆される．

さらに C3 on CIFAR-10 においては，FGSM を用いた敵対的サンプルに対して，SGD に比べて適応的最適化手法で学習されたモデルの方が頑健性が確保されにくいということも確認した．

最後に，近年よく用いられる 2 つのプラクティカルな DNN モデル（ResNet-34, DenseNet-121）に対しては，SGD と Adam によって学習されたモデルの最終層の重みパラメータに関するヘッセ行列を用いて，上位 20 個の固有値分布の比較と固有ベクトルを摂動として加えたときの訓練損失の局所形状の可視化を行った．こちらの実験においても，SGD と比べて Adam で学習されたモデルの方が最終層の重みパラメータ空間において急峻な損失形状に収束する可能性が高いことがわかった．つまりプラクティカルな DNN モデルにおいて，最終層の重みパラメータに関しては比較的小さな DNN モデルでの実験と整合性のある結果となった．

本実験の結果より，特に DNN を用いた画像分類タスクの場合，摂動に対してより汎化性，頑健性を担保したい場合には適応的最適化手法ではなく非適応的な最適化手法を使用することを勧める．

6.2　今後の課題

本研究の実験は，画像分類データセットに対してのみの結果となっている．DNN の発展は画像処理以外にも大きな影響を与えており，自然言語処理や音声処理など，より多くの

データセットについても同様の分析を行う必要がある．

また今後は，ヘッセ行列から得られる情報（固有値分布，トレース，TIC など）を利用し，適応的最適化と非適応的最適化を交互に行う動的な最適化手法を開発する予定である．このような方法は適応的最適化手法の高速な収束性と SGD の持つ汎化性，頑健性の恩恵を受けることが期待できる．

謝辞

本研究に取り組むに当たって，ご指導してくださった筑波大学図書館情報メディア系手塚太郎准教授に心より感謝いたします．日頃から様々な観点からアドバイスを頂き，本論文を執筆することができました．また，筑波大学図書館情報メディア系若林啓准教授には手塚若林研の合同ゼミや合同合宿でアドバイスを頂き，大変参考になりました．本当にありがとうございます．幸運なことに手塚・若林合同研究室の友人や先輩・後輩にも恵まれました．日々の友人や先輩・後輩との議論によって，大変有意義な研究生活を送ることができました．本当にありがとうございました．

参考文献

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2):1–34, 2011.

[3] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 557–565. PMLR, 2017.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

[5] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2017.

[6] Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *8th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, NAACL-HLT, pages 4171–4186. Association for Computational Linguistics, 2019.

[8] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR, 2017.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

[10] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2232–2241. PMLR, 2019.

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[12] Diego Granziol, Xingchen Wan, and Timur Garipov. Deep curvature suite. arXiv, 2020.

[13] Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. arXiv, 2018.

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

[15] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, January 1997.

[16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 2261–2269, 2017.

[17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25, NeurIPS*, pages 1106–1114, 2012.

[21] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau Standers*, 45:255–282, 1950.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, page 9–50, Berlin, Heidelberg, 1998. Springer-Verlag.

[24] Wesley J. Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. arXiv, 2020.

[25] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. arXiv, 2018.

[26] Ioannis Mitliagkas. Ift 6085 - lecture 5 accelerated methods - polyak ' s momentum (heavy ball method). Theoretical principles for deep learning, 2019.

[27] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.

[28] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). *Doklady AN USSR*, 269:543–547, 1983.

[29] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, 2000.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, 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 32, NeurIPS*, pages 8024–8035, 2019.

[31] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994.

[32] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[33] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. arXiv, 2016.

[34] Levent Sagun, Utku Evci, V. Ugur Güney, Yann N. Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. In *6th International Conference on Learning Representations, ICLR, Workshop Track Proceedings*, 2018.

[35] Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[37] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1139–1147. JMLR.org, 2013.

[38] K. Takeuchi. The distribution of information statistics and the criterion of goodness of fit of models. *Mathematical Science*, 153:12–18, 1976.

[39] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[40] Thomas Valentin, Pedregosa Fabian, van Merriënboer Bart, Manzagol Pierre-Antoine, Bengio Yoshua, and Roux Nicolas Le. On the interplay between noise and curvature and its effect on optimization and generalization. In *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3503–3513. PMLR, 2020.

[41] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems 30, NeurIPS*, pages 4148–4158, 2017.

[42] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv, 2017.

[43] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *ICML workshop on Beyond First-Order Optimization Methods in Machine Learning*, 2020.

[44] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems 31, NeurIPS*, pages 4954–4964, 2018.

[45] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. arXiv, 2012.

[46] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu-Hong Hoi, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.