【総説・解説論文】

Forward-Forward の MNIST 追試とハイパーパラメータ探索

Replication of the Forward-Forward Algorithm on MNIST and a Log of Hyperparameter Search

白濵　成希[1]　　中谷　直史[2]　　渡邉　志[3]

Naruki　Shirahama[1]　　Naofumi　Nakaya[2]　　Satoshi　Watanabe[3]

[1]下関市立大学　　[2]順天堂大学　　[3]静岡理工科大学

[1]Shimonoseki City University

[2]Juntendo University

[3]Shizuoka Institute of Science and Technology

要旨

本稿は，Forward-Forward（FF）アルゴリズムの MNIST 教師あり学習を対象に，追試実験ログを整理した解説論文である．実装仕様と数式の対応を確認したうえで，ハイパーパラメータ探索を「仮説→操作→結果→解釈→次の一手」の形で記録し，ログを横断集計した表・図として提示する．追試では，ラベルを総当たりで埋め込んで評価する推論（label-search）による最良のテスト誤り率は，平行移動拡張（jitter；±2 ピクセル）を用いた 500 エポック学習で 2.12%であった．

キーワード：　Forward-Forward，MNIST，局所学習，ハイパーパラメータ探索，再現実験

Abstract

This paper is a Japanese tutorial article that summarizes our replication of the Forward-Forward (FF) algorithm on MNIST in the supervised label-in-input setting. We verify the correspondence between the implementation and the mathematical formulation and organize our hyperparameter exploration in a hypothesis-intervention-result-interpretation-next-step format, providing tables and figures aggregated across runs from the experiment logs. In our replication, the best test error under label-search inference was 2.12% with jitter augmentation (± 2 pixels) after 500 epochs.

Keywords: Forward-Forward, MNIST, local learning, hyperparameter search, replication study

1.はじめに

深層学習で広く用いられるバックプロパゲーション[1]は，誤差微分を逆伝播させるために，順伝播（forward pass）で行われる計算の詳細が既知であることや，中間活動を保存したうえで逆伝播計算（backward pass）を行うことを前提とする．Hinton[2]は，この前提が大脳皮質における学習を説明するモデルとしては強い制約であることに加え，順伝播の途中に未知の非線形や確率的変換を含むブラックボックスが挿入されると，正しい微分を計算できないことを指摘した．さらに，同論文は，学習を二度の順伝播のみで進める FF が，低消費電力アナログハードウェアと親和的である可能性も論じている．

FF は，各層が「正例では goodness（良さの指標）を高く，負例では低くする」という局所目的を持ち,正例データと負例データに対する二度の順伝播によって学習更新を行う．原論文[2]は，FF の有効性をまず比較的小規模な課題で検証する予備的研究として位置づけており，本稿もそのうち Hinton[2]が提示した MNIST の教師あり入力ラベル埋め込み（label-in-input）設定を対象とする．本稿では，追試実装と実験ログを材料として，(i) アルゴリズムの要点（goodness，負例生成，評価手順）を学習者向けに整理し，(ii) ハイパーパラメータ探索の意思決定を追跡可能な形で記述し，(iii) 原論文の報告値と追試結果の一致点・差異を議論する．

本稿の構成は次の通りである．第 2 節で FF の多層化に必要となる正規化の位置づけと近年の関連研究を述べ，第 3 節で追試実装における設計（goodness，正規化，負例生成，推論，ピア正規化（peer normalization））を実装仕様に対応付けて整理する．第 4 節で実験設定をまとめ，第 5 節で原論文の報告値と追試ログ集計を提示し，第 6 節で探索の意思決定と差異要因を考察し，第 7 節で結論と今後の課題を述べる．

2. 理論的背景と関連研究

FF の多層学習では，各層の goodness の情報，すなわち活動ベクトルの長さをそのまま次層へ渡すと，次層が長さだけで正例／負例を分離できてしまい，意味のある表現学習になりにくい．Hinton[2]はこれを避けるため，各層の ReLU（整流線形ユニット）活動ベクトルに対し，平均を差し引かずベクトル長で割る簡約形の層正規化を施し，次層には活動の大きさではなく相対的な活動パターン，すなわち向きのみを渡す層間表現の受け渡しを採用している[3]．本稿の追試実装でも同様に，各層の ReLU 出力を L2 正規化して次層入力とする．

FF の応用と拡張に関する研究も近年進んでいる．Scodellaro らは，FF を畳み込みニューラルネットワークへ拡張し，ラベルを画像全体へ空間的に埋め込む手法を提案した[4]．Zhao らは，負例生成を要しない Cascaded Forward（CaFo）を提案し，各カスケードブロ

ックが局所的にラベル分布を出力する構成によって，学習と推論の効率化を図っている．Gandhi らは，MNIST 追試に加えて IMDb 感情分類へ適用範囲を拡張し，FF 固有のしきい値に対する pyramidal optimization の効果を報告している[6]．これらに対し本稿は，新規アルゴリズムの提案ではなく，Hinton[2]の label-in-input 設定を対象に，実装仕様と探索ログを対応づけて整理することを目的とする．

3.追試実装の仕様

本節では，原論文の教師あり「入力にラベルを埋め込む方式（label-in-input）」設定[2]を追試した実装仕様を，数式と対応付けて解説する．

3.1. Goodness と局所目的

層ℓの線形出力をa_ℓ，ReLU 出力を$y_\ell = max(0, a_\ell)$とする．本追試では goodness を

$$g_\ell(x) = \sum_j y\ell_j(x)^2 \tag{1}$$

（ReLU 活動の二乗和）として定義する．原論文では，各層が正例/負例を判別できるように

$$p_\ell(positive|x) = \sigma(g_\ell(x) - \theta) \tag{2}$$

（σはシグモイド，θはしきい値）と定式化される[2]．ただし，原論文は具体的な損失関数の形が明示されていないため，本追試実装では式(2)と整合する単純な二値交差エントロピー（$logit = g_\ell - \theta$; $logit$はシグモイドの入力）を各層で用い，層平均した損失を用いて確率的勾配降下法（SGD）により更新する．

3.2. 層間の正規化と勾配遮断

goodness は活動ベクトルの長さに依存するため，多層化の際は長さ情報を次層へ渡さない工夫が必要である[2,3]．本追試実装では，各層の ReLU 出力y_ℓを L2 正規化して次層入力とし，さらに学習ステップでは層間で勾配を遮断（PyTorch の detach 操作）する．これにより，各層が「自分の局所目的」で表現を学ぶ層別学習（greedy layer-wise learning）を実装する．

3.3. 教師あり：入力ラベル埋め込み（label-in-input）の正例/負例

MNIST 教師あり例では，入力$x \in \mathbb{R}^{784}$の先頭$K = 10$次元をラベル表現として用いる（入力ラベル埋め込み；label-in-input）[2]．正例は「正しいラベルを埋め込んだ入力」，負例は「誤ったラベルを埋め込んだ入力」である．簡略化のため，ラベル埋め込み操作を$Inject(x, y)$（クラスcの one-hot（one-of-K）を先頭に書き込む）と書くと，

$$x^{(+)} = Inject(x, y), \tag{3}$$

$$x^{(-)} = Inject(x, \tilde{y}), \tilde{y} \neq y \tag{4}$$

である．原論文の議論に従い，負例ラベル$\tilde{y}$は一様に誤ラベルをサンプルする方式（ログ上の設定名：randneg）だけでなく，hard negative（紛らわしい負例；誤ラベルの中で goodness が高いもの）を選ぶことで学習を加速できる[2]．本追試実装では，hard negative の選び方として，(i) ラベル総当たり（label-search）による厳密選択（計算量が大きい），(ii) one-pass softmax による近似選択（高速）を切り替え可能にしている．これらの呼称は，追試実装の設定名として config.yaml にも記録される．具体的には，(i) は 3.4 節で定義するスコア$S(c)$を用いて誤ラベル集合の中で$S(c)$が最大となるラベルを選び，(ii) はニュートラルラベルによる 1 回の順伝播から得た特徴に対して学習した softmax head の予測に基づき，最も紛らわしい誤ラベルを選ぶ[2]．

3.4. 推論：ラベル総当たり（label-search）と 1 回推論（one-pass）

原論文では，テスト時に各ラベルcを入力に埋め込み，層$\ell = 2, \cdots, L$の goodness を足し合わせたスコア

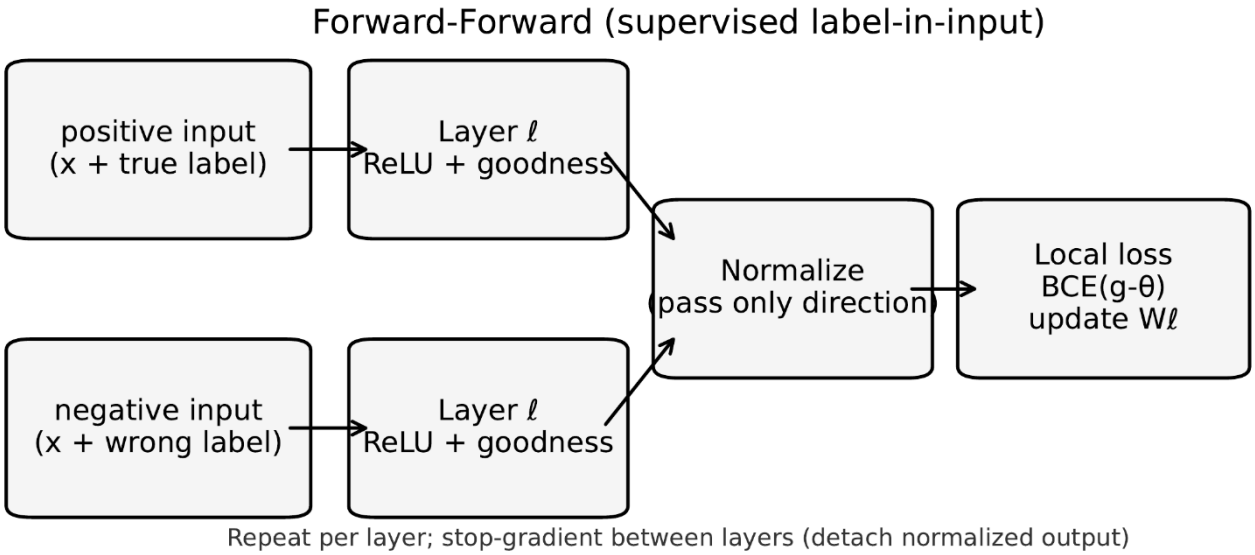
$$S(c) = \sum_{\ell=2}^{L} g_\ell\big(Inject(x, c)\big) \tag{5}$$

を最大化するラベル$\hat{y} = arg\ max_c S(c)$を採用する[2]（ラベル総当たり；label-search）．これは高精度だが，クラス数（MNIST では 10）回の順伝播が必要になる．

一方，原論文は高速だが精度が劣る近似として，ニュートラルラベル（先頭 10 次元をすべて 0.1）で 1 回だけ順伝播し，中間層表現から softmax 分類器を学習する方法（1 回推論；one-pass）も述べている[2]．本追試のログでも，one-pass は補助指標として毎エポック記録している．

3.5. peer normalization（ピア正規化）と dead ReLU 対策

原論文の脚注の議論に従い[2]，ユニットが極端に活動し続けたり，完全に沈黙（dead ReLU：ReLU が常に 0 となり更新が止まる状態）したりすることを抑えるために，peer normalization（層内ユニットの平均活動を揃える正則化）を実装した．これは，各ユニットの平均活動が，層内平均との差の移動平均へ回帰するように正則化するものであり，実装では ReLU が off のときも擬似的に勾配を流す straight-through（導関数を 1 とみなす近似）を併用する．本追試実装における教師あり FF（入力ラベル埋め込み；label-in-input）の学習手順（正例/負例の 2 回順伝播と局所損失）を図 1 に示す．



出典：筆者の実験結果に基づき作成

図 1　追試実装における教師あり FF（入力ラベル埋め込み；label-in-input）

4.実験設定

4.1. データと分割

MNIST[7]の公式訓練データ 60,000 枚のうち 50,000 枚を学習に，10,000 枚を検証（validation；val）に用い，公式テストデータ 10,000 枚で最終評価する[2]．平行移動によるデータ拡張（jitter augmentation）は学習時のみ適用し，本追試実装ではpピクセルのパディング（padding）を付けたランダム切り出し（random crop）による平行移動$(\pm p)$として実装している（原論文では 25 シフトの拡張を記述）[2].

4.2. モデル

原論文の設定に合わせ，入力次元 784，隠れ層 4 層（各 2000 ReLU）の全結合 MLP（多層パーセプトロン）を用いる[2]．各層は ReLU 出力の L2 正規化を次層へ渡す．バイアス項の有無は実験条件として切り替えた．

4.3 学習と評価

本追試で用いたソースコードは GitHub のリポジトリとして公開している（以下の URL からアクセス可能 https://github.com/nshrhm/forward-forward-algorithm ）．本稿では以降この公開リポジトリの理解を促進するための，変数名やファイル名と合わせて説明する．

最適化には確率的勾配降下法（SGD）を用い，ミニバッチサイズ 256 とした．ハイパーパラメータとしてθ，学習率，重み減衰（weight decay），モメンタム（momentum），peer

normalization の強度，hard negative の方式（label-search / one-pass）および warmup（切替までのエポック数），epochs（学習エポック数），jitter（平行移動拡張の幅）を探索した．

評価は原論文の推奨に従い label-search の誤り率（error rate）を主指標とし[2]，one-pass の誤り率は補助的に記録した．ここで label-search / one-pass，val / test といった表記は，追試実装が出力するログ（metrics.csv 等）の列名・設定名に対応する．

ログは実験実行ごとに config.yaml（設定ファイル），metrics.csv/jsonl（エポックごとの検証（val）と最終テスト（test））， training_curves.png に保存される．本稿では scripts/collect_runs.py により全 run を走査し tab/summary_runs.csv と図表を生成した（再現可能な集計）．以降，本稿では可読性のため，run_id は先頭の日時（YYYYMMDD_HHMMSS）で略記する．

5. 実験結果

5.1 原論文の報告値

原論文の教師あり（入力ラベル埋め込み；label-in-input）では，4×2000 ReLU の全結合 MLP で 60 エポックでテスト誤り率 1.36%が報告されている．また jitter（±2 ピクセル，25 シフト）を用いて 500 エポック学習するとテスト誤り率 0.64%が得られたと報告されている[2].

5.2 追試ログの集計（全 run）

本追試リポジトリから得られた実行結果（results/*）を集計した結果を表 1 および表 2 に示す．ここで val は検証（validation）データの誤り率，test はテストデータの誤り率を表し（いずれも追試実装のログ列名に対応），test は最終エポック終了後のモデルで評価した値である．

表 1　系列（epochs・jitter）ごとの best run（label-search の test error 最小）

series	run（日時）	θ	lr	val(%)	test(%)
ep100_nojitter	20260109_064848	112	0.04	2.52	2.53
ep120_nojitter	20260109_070300	112	0.04	2.40	2.43
ep200_jitter2	20260109_091335	112	0.04	2.05	2.35
ep500_jitter2	20260109_102355	112	0.04	1.77	2.12
ep60_nojitter	20260109_055305	116	0.04	2.84	2.55
ep80_nojitter	20260109_063756	112	0.04	2.63	2.58

出典：筆者の実験結果に基づき作成

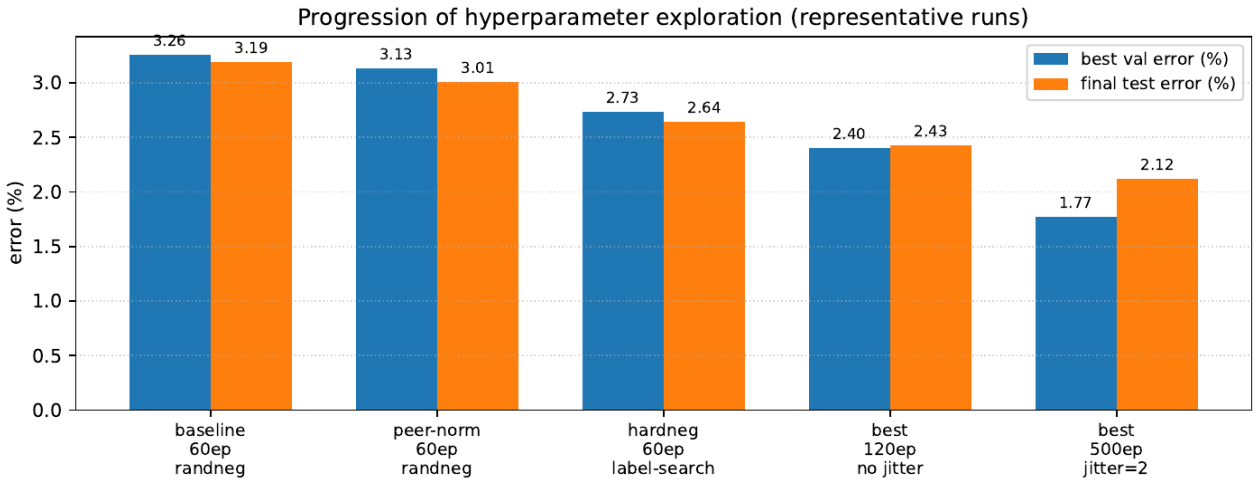
表 2　実験ログから集計した上位 run（label-search の test error が小さい順）

run（日時）	ep	jit	θ	lr	val(%)	test(%)
20260109_102355	500	2	112	0.04	1.77	2.12
20260109_091335	200	2	112	0.04	2.05	2.35
20260109_070300	120	0	112	0.04	2.40	2.43
20260109_073239	120	0	112	0.04	2.53	2.45
20260109_234645	500	2	112	0.04	2.42	2.52
20260109_064848	100	0	112	0.04	2.52	2.53
20260109_055305	60	0	116	0.04	2.84	2.55
20260109_052353	60	0	112	0.05	2.91	2.57
20260109_050613	60	0	112	0.035	2.87	2.58
20260109_063756	80	0	112	0.04	2.63	2.58

出典：筆者の実験結果に基づき作成

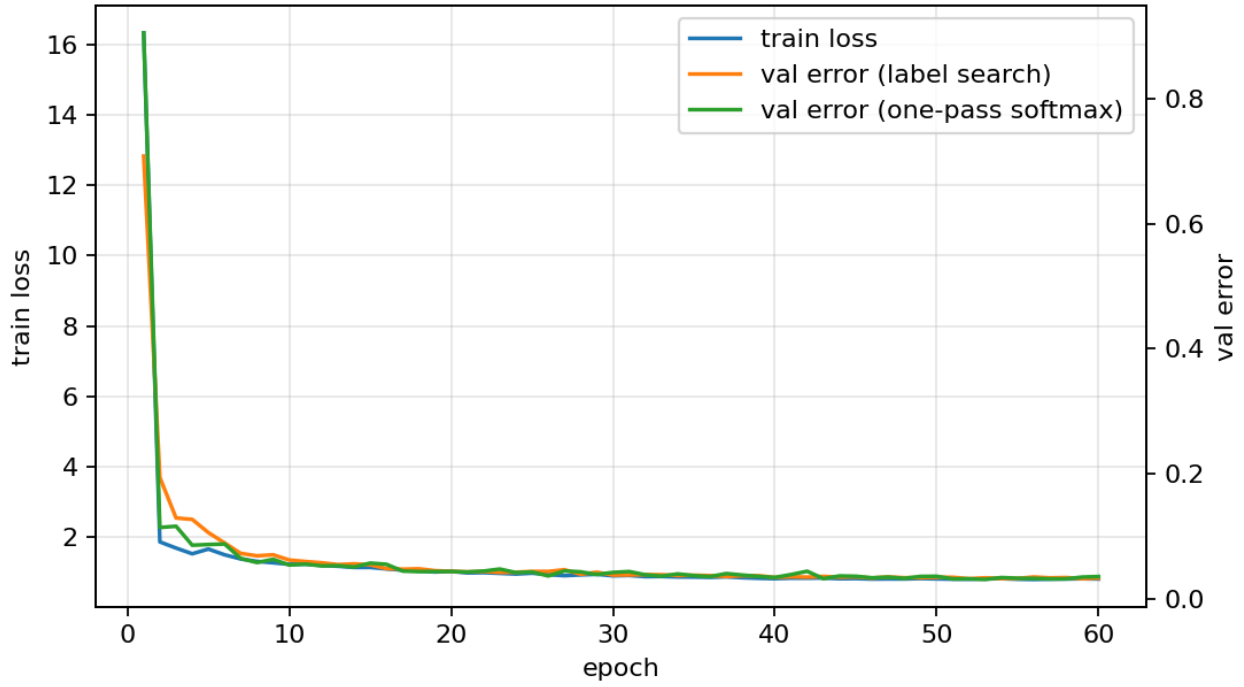
5.3 学習曲線と代表例

図 2 は，探索の代表的な節目（ベースライン，peer normalization（ピア正規化），hard negative（紛らわしい負例），長期学習，jitter（平行移動拡張））を，検証（val）の最良値と最終テスト（test）の誤り率で比較したものである．



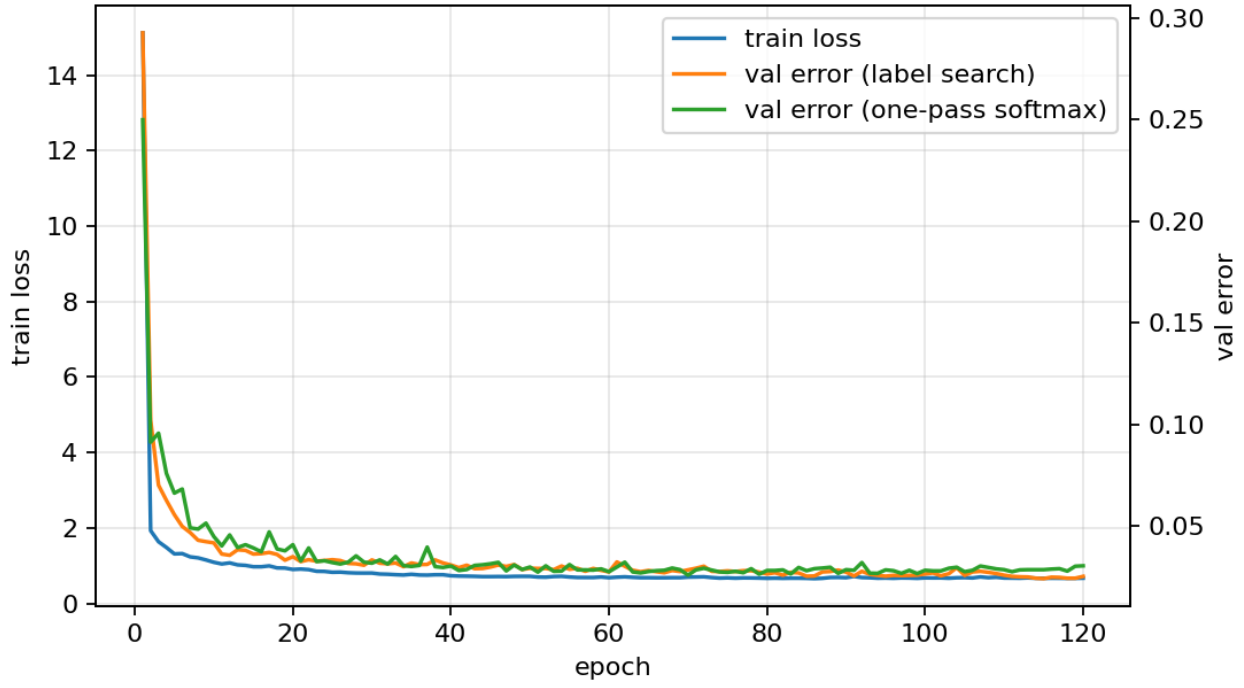
出典：筆者の実験結果に基づき作成

図 2　探索の節目に対応する代表実行の比較（検証誤り率の最良値と最終テスト誤り率）



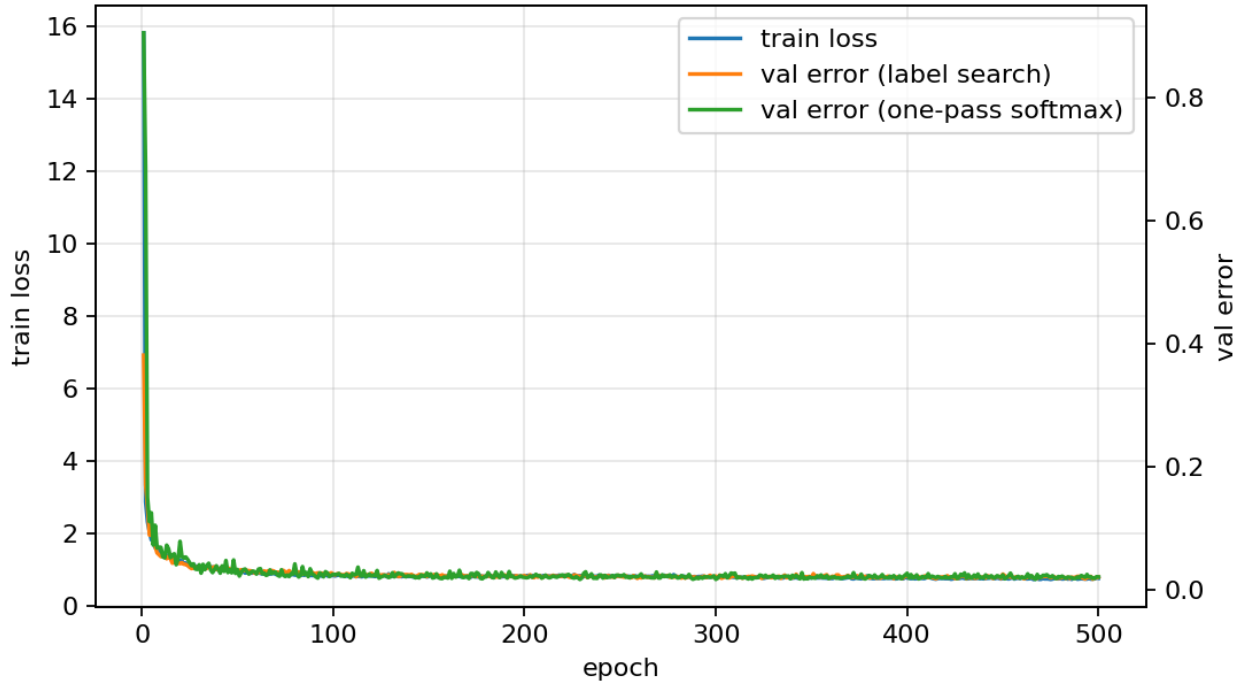
出典：筆者の実験結果に基づき作成

図 3　ベースライン(60 エポック，一様誤ラベル; randneg)の学習曲線（run: 20260107_062559）



出典：筆者の実験結果に基づき作成

図 4　jitter なしの最良例（120 エポック）の学習曲線（run（日時）：20260109_070300）



出典：筆者の実験結果に基づき作成

図 5　jitter ありの最良例（500 エポック，jitter=2（ピクセル））の学習曲線（run（日時）:20260109_102355）

6. 考察

6.1 探索の方針：検証（val）を主指標として段階的に固定する

本追試では，原論文がθを具体的に与えていない点[2]を出発点として，(i) θ，(ii) 学習率，(iii) weight decay（重み減衰），(iv) momentum（モメンタム），(v) peer normalization，(vi) hard negative，(vii) epochs（学習エポック数），(viii) jitter augmentation（平行移動拡張）の順に，一度に変える要素を 1 つに近づける方針で探索を進めた（探索手順の一次情報は notes/todo_experiments.md）．ハイパーパラメータの選択には検証（val）の誤り率を用い，test は最後に参照する．

6.2 θと最適化ハイパーパラメータ：「学習が進行する」領域を作る

$\boldsymbol{\theta}$は式(2)の logit のしきい値として働くため，活動スケールや正規化の仕方と強く相互作用する．初期段階では$\boldsymbol{\theta}$と学習率を粗く探索し，学習が安定して val が低下する領域（本ログでは主として$\theta = 128$前後，$lr = 0.04$を前後）を確保した．その後，weight decay と momentum を掃引して，性能劣化が小さい範囲を固定した．

6.3 ピア正規化：不活性ユニットを避ける正則化の効果

ピア正規化（peer normalization）を有効化すると，ベースライン（一様誤ラベル；randneg）に対してテスト誤り率（test error）が改善した（例：60 エポックで約 3.19%から 3.01%，図 2）．これは，層内のユニットが「極端に活動または沈黙」する状態を抑え，表現の利用効率が上がった可能性と整合的である[2].

6.4 hard negative：計算量と学習の難しさのトレードオフ

hard negative は，負例ラベルを「最も紛らわしい誤ラベル」に寄せることで学習信号を強め，必要エポック数を減らしうる[2]．ログでは，一様誤ラベル（randneg）から hard negative（label-search）へ切り替えることで誤り率が一段階改善し，さらにエポック数を 80，100，120 と延長することでテスト誤り率が 2.58%，2.53%，2.43%と順に改善した（表 2）.

一方で，hard negative を one-pass で近似して高速化する試みでは，one-pass 指標自体は改善しても，label-search 指標（主指標）が悪化するケースが観測された（notes/todo_experiments.md の Step 16）．one-pass と label-search は推論手順が異なるため，one-pass を最適化しすぎると「label-search にとっての望ましい表現」とずれる可能性がある.

6.5 jitter augmentation（平行移動拡張）：原論文との差をどう解釈するか

原論文は平行移動拡張（jitter）を用いて 500 エポック学習した場合にテスト誤り率 0.64%を報告する[2]が，本追試ログの最良は 2.12%に留まった（表 2）．差異の要因として，少なくとも以下が考えられる.

- データ拡張の実装差：原論文は 25 シフトの拡張を記述するのに対し，本追試は random crop（ランダム切り出し）によるランダム平行移動である.
- hard negative の生成差：原論文はニュートラルラベルの順伝播を用いた hard negative 選択を述べる[2]のに対し，本追試の最良の実験実行は精度を優先して label-search による hard negative を用いている.
- 局所損失の設計差：原論文は確率的定式化（式(2)）を与えるが，具体の損失関数の選び方にはなお余地がある.

したがって，「jitter を入れれば 0.64%に近づく」という単純な期待ではなく，データ拡張，hard negative の生成，局所損失の設計を含めた追試差分の洗い出しが必要である.

7. 結論

本稿では，FF の教師あり入力ラベル埋め込み（label-in-input）設定[2]を対象に，追試実装の設計（goodness，局所損失，正規化，推論手順）を整理し，実験ログに基づいてハ

イパーパラメータ探索の意思決定を段階的に記述した．ログ集計の結果，ラベル総当たり推論（label-search）による最良のテスト誤り率は，平行移動拡張（jitter；±2 ピクセル）を用いた 500 エポック学習で 2.12%であった．これは原論文[2]の 0.64%には未到達であり，その要因としては，25 シフト拡張と random crop ベース拡張の差，hard negative の生成法の差，および局所損失の具体的な定義の差が考えられる．今後は，これらの差分を個別に切り分ける追加実験を通じて，原論文との差異をより厳密に特定する必要がある．

引用・参考文献

[1] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams (1986) "*Learning representations by back-propagating errors*". Nature 323.6088, pp. 533–536. doi: 10.1038/323533a0

[2] Geoffrey E. Hinton (2022) "*The Forward-Forward Algorithm: Some Preliminary Investigations*" arXiv: 2212.13345 [cs.LG]. doi: 10.48550/arXiv.2212.13345

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016), "*Layer Normalization*", arXiv: 1607.06450 [stat.ML]. doi: 10.48550/arXiv.1607.06450

[4] Riccardo Scodellaro, Ajinkya Kulkarni, Frauke Alves, and Matthias Schröter (2025) "*Training convolutional neural networks with the Forward–Forward Algorithm*". Scientific Reports 15, 38461. doi: 10.1038/s41598-025-26235-2

[5] Gongpei Zhao, Tao Wang, Yi Jin, Congyan Lang, Yidong Li, and Haibin Ling (2025) "*The Cascaded Forward algorithm for neural network training*". Pattern Recognition 161, 111292. doi: 10.1016/j.patcog.2024.111292.

[6] Saumya Gandhi, Ritu Gala, Jonah Kornberg, and Advaith Sridhar (2023) "*Extending the Forward Forward Algorithm*". arXiv:2307.04205 [cs.LG]. doi: 10.48550/arXiv.2307.04205.

[7] Yann LeCun et al. "*Gradient-Based Learning Applied to Document Recognition*". In: Proceedings of the IEEE 86.11 (1998), pp. 2278–2324. doi: 10.1109/5.726791