斟酌学習　レポート

本田 康祐

2020/05/25

1　「ナイーブな方法」の実装

1. training accuracy が 50,60,75,80%になったところで学習を止める
2. validation accuracy が 50,60,75,80% になったところで学習を止める

という 8 つの条件でそれぞれ 5 回試行し、test accuracy がどういう値になるのか、実験してみた。

今回データは MNIST を利用した。MNIST には訓練データ 60000 個と検証データ 10000 個があるが、バリデーションデータは用意されてないので、訓練データから訓練データ 48000 個、バリデーションデータ 12000 個を取り出した。

モデルは以下に示すように、全結合層 (fc1, fc2)2 個を用いた簡単な DNN で試した。ドロップアウト層を付けると train acc と val acc で大きく差が出るので、今回は付けていない。

```
Net(
  (fc1): Linear(in_features=784, out_features=1000, bias=True)
  (fc2): Linear(in_features=1000, out_features=10, bias=True)
)
```

Table 1: その他のパラメータ

パラメータの種類	使用したパラメータ
fc1 と fc2 の活性化関数	ReLU
出力関数	Softmax
損失関数	Cross Entropy Error
最適化アルゴリズム	SGD

また、各条件に応じて学習率を変更した。

Table 2: 各条件に応じた学習率の設定

train および val acc(%)	学習率
50	10^{-4}
60	10^{-3}
75	10^{-2}
80	10^{-2}

2 実験結果

2.1 8 つの各条件に対する test accuracy の結果

8 つの各条件に対する test accuracy は以下のようになった。

Table 3: 8 つの各条件に対する test accuracy の結果

条件	平均 (%)	分散
train acc=50%	54.0	4.64
val acc=50%	52.9	4.35
train acc=60%	70.2	35.38
val acc=60%	60.9	5.06
train acc=75%	85.2	10.48
val acc=75%	76.9	55.86
train acc=80%	89.4	16.22
val acc=80%	88.1	35.65

この表より、次の 2 つのことがわかった。

- train acc より val acc を指定した方が test acc が制御できた
- 基本的に acc が 80%より 50%の方が test acc の分散が小さい結果となった

また、val acc が 75, 80%のときに悪い結果となっているが、その原因としてわかっていることが 2 つある。

2.1.1　学習終了時の accuracy の値が指定した値をオーバーした

1 つ目は val acc が 75,80%の時の学習率を大きくしていたからか、突発的に指定した acc の値を大きく超えてしまったケースが表れてしまったからである。

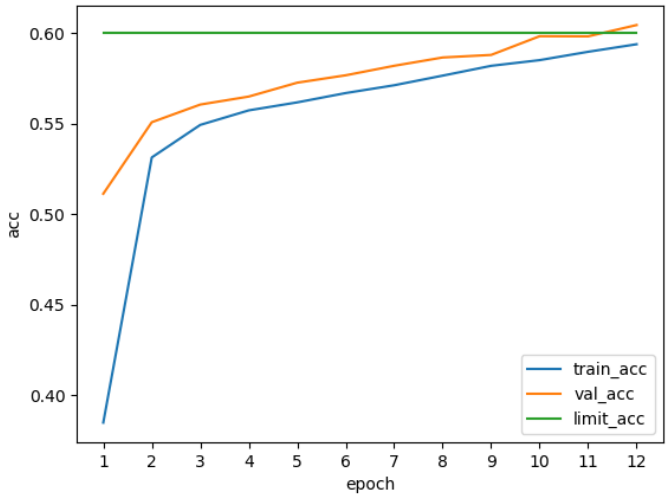

Figure 1: val acc が 60%の時の 1 回目の試行の学習曲線

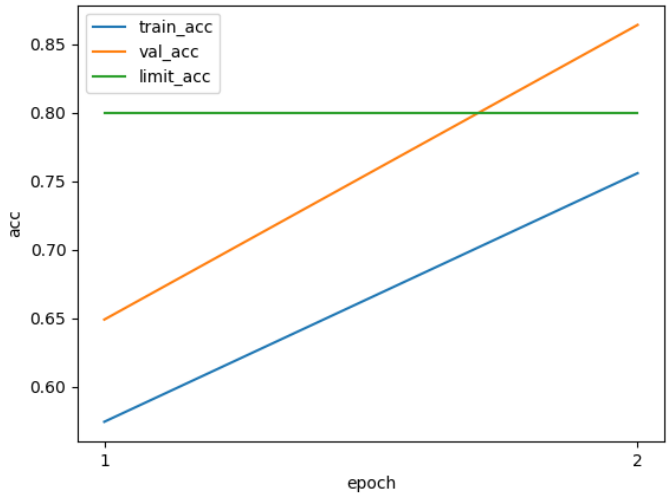

Figure 2: val acc が 80%の時の 4 回目の試行の学習曲線

図 1 は val acc が 60%の時の 1 回目の試行の学習曲線を取り出したものである。epoch は 12 回で終わり、test acc は 59.9%という良い結果が得られた。

一方、図 2 は val acc が 80%の時の 4 回目の試行の学習曲線を取り出したものである。epoch は 2 回で終わったためこのような直線になっているが、最終的な val acc の値が 80%を大きく越しており、test acc は 86.44%と、80%を越す結果が得られた。

2.1.2 学習 epoch 数が上限に達した

2 つ目は定めた acc に達しないまま epoch 数が上限を超えてしまったことがあったからである。今回、上限の epoch 数を 50 回と定めていたが、val acc が 75%のときに 1 回だけ、80%のときに 1 回だけ acc の目標値に達しないまま上限の epoch 数に達してしまい、その条件のときの test acc が悪くなってしまった。その 2 つの条件で分散が異様に大きいのはそのためである。

2.2 誤検出/正検出が多かったデータ

2.2.1 訓練データ・バリデーションデータ・検証データで比較

8 つの条件全体から誤検出/正検出の数を総計し、訓練データ・バリデーションデータ・検証データで上位 30 枚のデータを見比べた。(バリデーションデー

タ、テストデータでは全て誤検出したデータが 30 枚以上あったが ID 順で上位 30 枚を決めている。また、各文字画像のタイトルは「MNIST の ID(正検出の数)」を表す。)

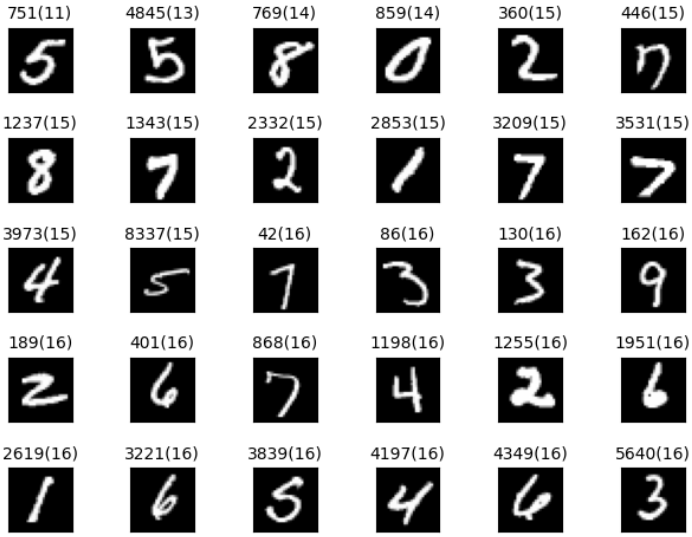

Figure 3: 8 つの条件全体で誤検出した訓練データ上位 30 枚

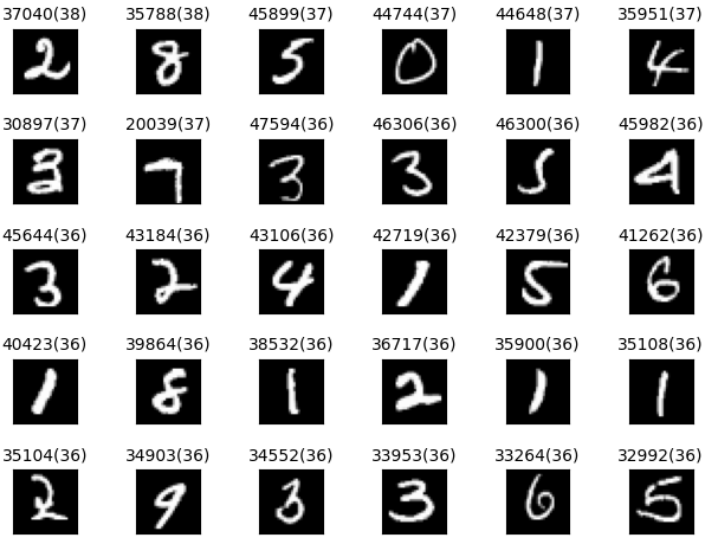

Figure 4: 8 つの条件全体で正検出した訓練データ上位 30 枚

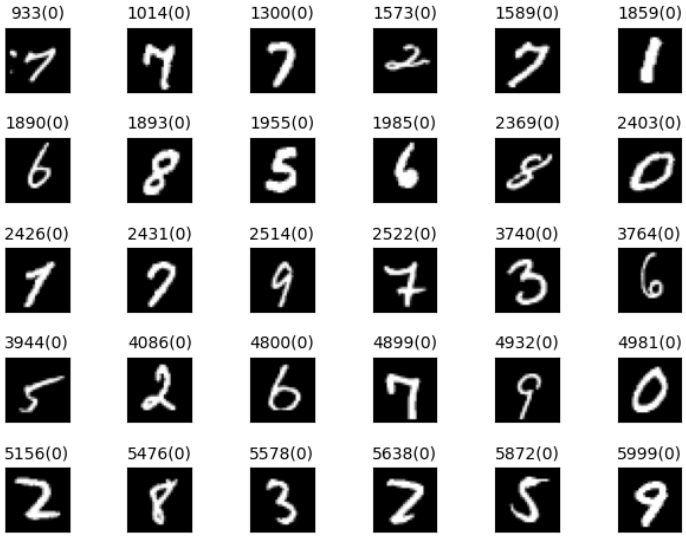

Figure 5: 8 つの条件全体で誤検出したバリデーションデータ上位 30 枚

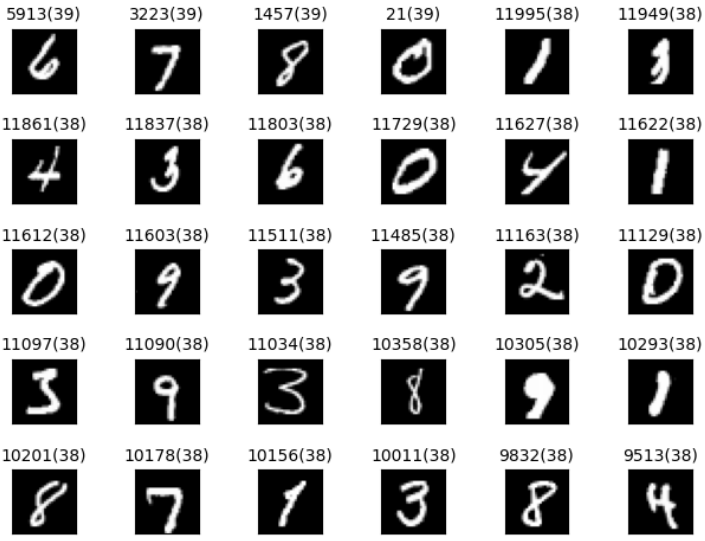

Figure 6: 8 つの条件全体で正検出したバリデーションデータ上位 30 枚

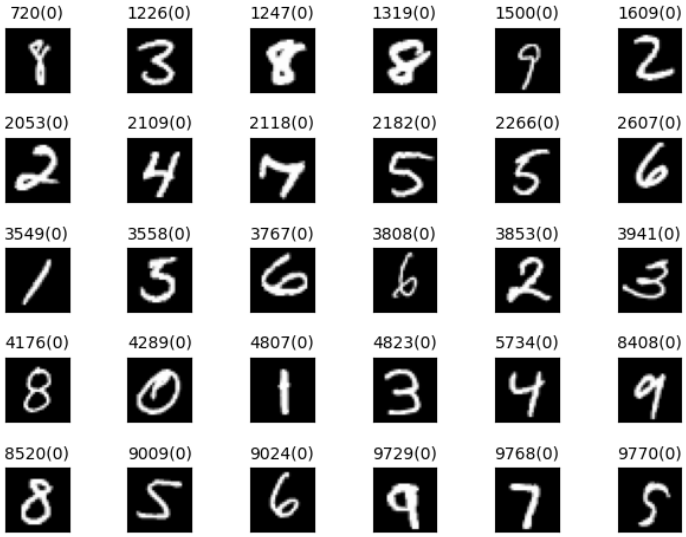

Figure 7: 8 つの条件全体で誤検出したテストデータ上位 30 枚

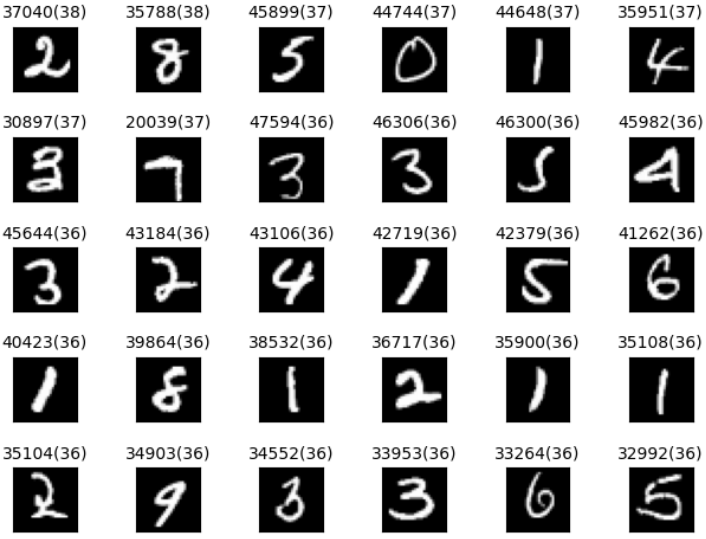

Figure 8: 8 つの条件全体で正検出したテストデータ上位 30 枚

これらの図より、次の 2 つのことがわかった。

- val, test は 40 個の結果全てにおいて誤検出していたものがあったが、train は多くても 29 個誤検出くらいであった。そのため、val, test で誤検出になるデータと、train で誤検出になるデータに相関があるのかがここではわからなかった。

- 見た感じ、誤検出/正検出のデータを見比べても誤検出データのほうが難しいものが集まっている、というわけではなかった。

2.2.2 train および validation accuracy が 50%の時と 80%の時で比較

train および validation accuracy が 50%の時と 80%の時で、誤検出/正検出の数の上位 30 枚を比較した。

結論からいうと、あまり明確な違いが見られなかった。

ここでは validation accuracy が 50%の時と 80%の時の、訓練データ・バリデーションデータ・検証データの誤検出上位 30 枚を示す。

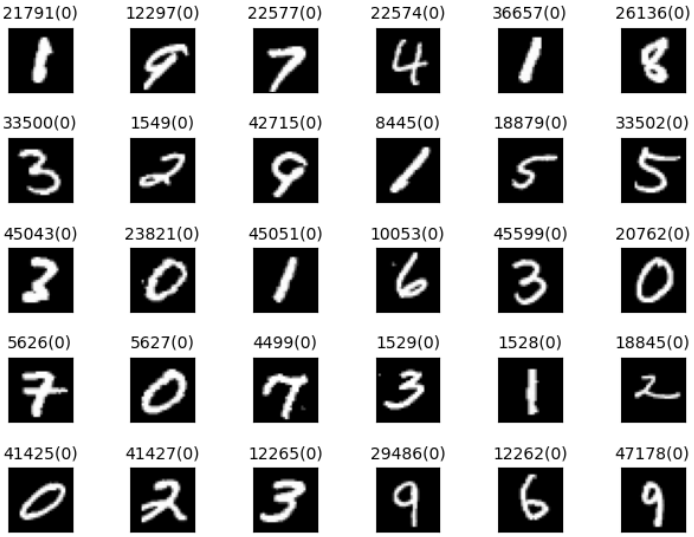

Figure 9: validation accuracy が 50%の時に誤検出した訓練データ上位 30 枚

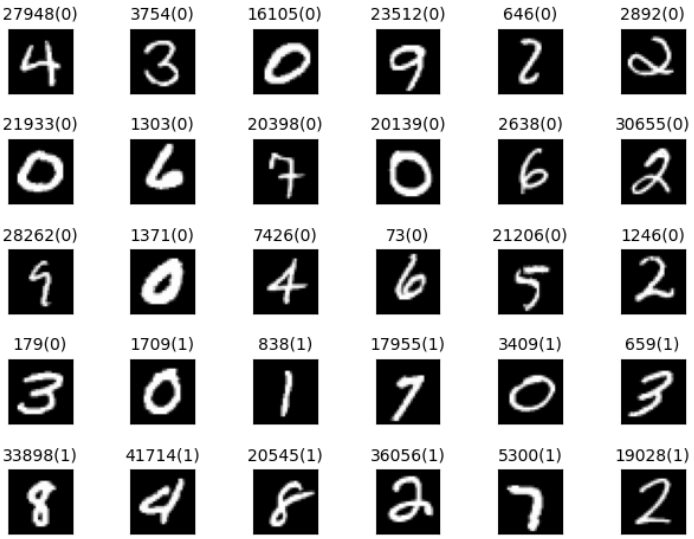

Figure 10: validation accuracy が 80%の時に誤検出した訓練データ上位 30 枚

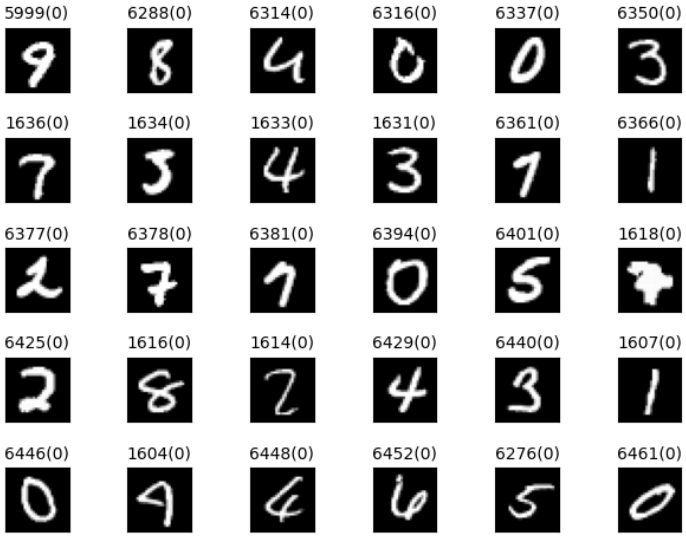

Figure 11: validation accuracy が 50%の時に誤検出したバリデーションデータ上位 30 枚

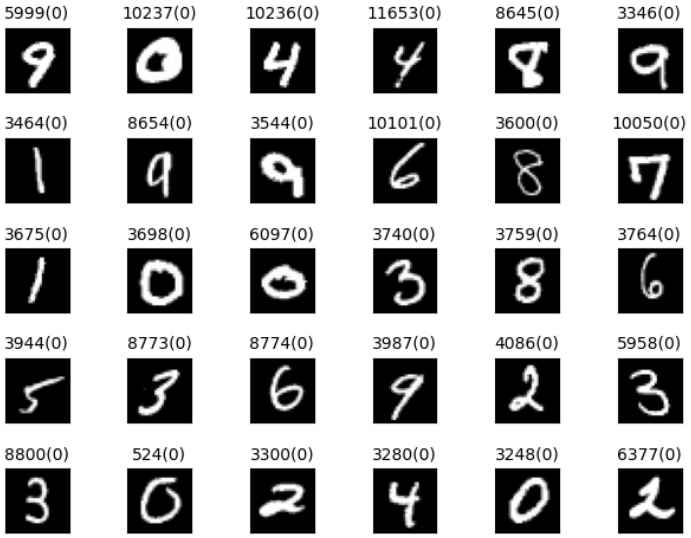

Figure 12: validation accuracy が 80%の時に誤検出したバリデーションデータ上位 30 枚

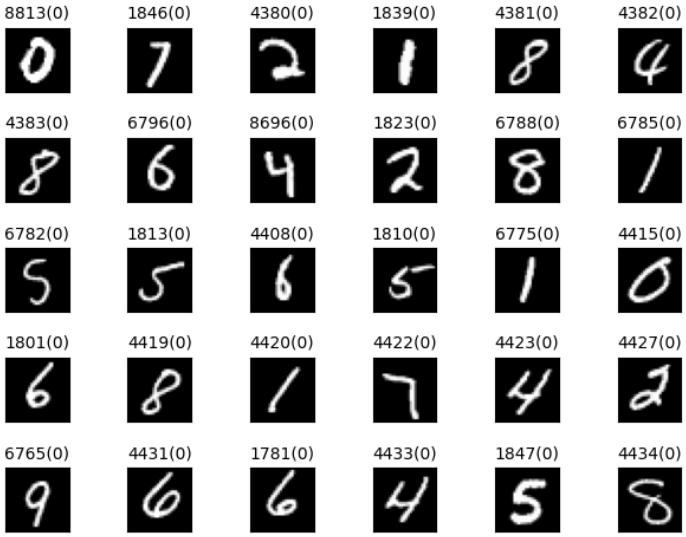

Figure 13: validation accuracy が 50%の時に誤検出した検証データ上位 30 枚

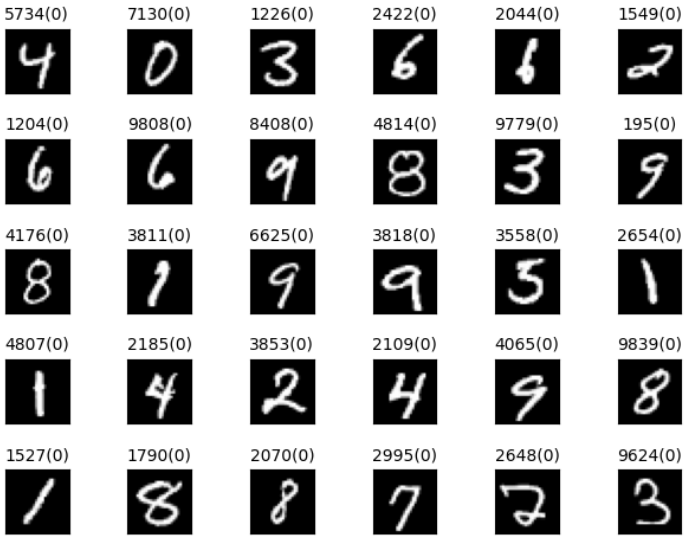

Figure 14: validation accuracy が 80%の時に誤検出した検証データ上位 30 枚

3　結論

MNIST 程度のデータであれば、学習率をうまく調整し目標の accuracy で学習が止まるようにすればある程度 test accuracy の値が調整できることが分かった。

ただ、MNIST だとどのデータが「難しい」データなのかがわからなかったので、識別器が「難しい」データを識別できなくなっていたのかはこの実験ではわからなかった。

そのため、末廣先生に提案していただいたようなシンプルなデータでもう一度実験したい。