



ネットワークの試行錯誤による精度向上

試行錯誤の結果発表

グループワーク: 試行錯誤の結果まとめ

Best Accuracy、パラメタと共に各パラメタの効果とその考察を議論

- ①各メンバーのベストの“Accuracy”を記載
(グループでベストの“Accuracy”は自動的に記載)
- ②ベストのAccuracyの設定(標準から変えたもののみ)
- 各メンバーの変化させたパラメタと“Accuracy”との関係を議論
 - ➔ ③各パラメタと“Accuracy”の関係(傾向)や議論した内容を考察としてまとめる

確認事項1: 分析データセット(MNIST)

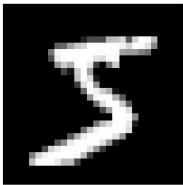
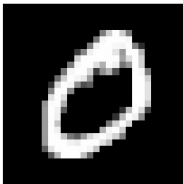
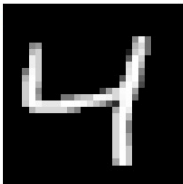
Home 編集 学習 評価 CNN-0_9 データセット 詳細設定

Training ...
mnist.mnist_training
Num Data 60000
Num Column 2
Shuffle true
Cache true
Normalize true
Main

Validation ...
mnist.mnist_test
Num Data 10000
Num Column 2
Shuffle false
Cache true
Normalize true

データセットの選択 mnist.mnist_training

☒ Main ☒ Shuffle ☒ Image Normalization(1.0/255.0) 1 / 6000

Index	x:image	y:label
1	1,28,28 	5
2	1,28,28 	0
	1,28,28 	4

- Training
 - mnist.mnist_training
 - Num Data: 60000
- Validation
 - mnist.mnist_test
 - Num Data: 10000

異なる2つのデータセットを設定

確認事項2: 学習反復世代数(Epoch数)

Home 編集 学習 評価 test09 データセット 詳細設定

詳細設定 ...

Global Config Global
Max Epoch = 10
Batch Size = 64

Optimizer Optimizer
Network = Main
Dataset = Training
Updater = Adam

train_error Monitor
Network = MainValidation
Dataset = Training

valid_error
Network = MainValidation
Dataset = Validation

Executor Executor
Network = MainRuntime
Dataset = Validation

プロジェクト説明:

プロジェクトタグ: 編集

学習反復世代数: 10 ☒ 最高精度のモデルを保存

バッチサイズ: 64

精度: Float

学習反復世代数: 10 ☒ 最高精度のモデルを保存

構造自動探索: ☐ 有効

検索範囲: Min Max

Validation

4

- 学習反復世代数(Epoch数): 10

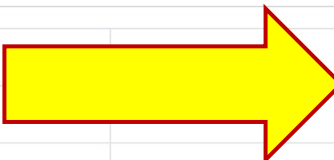
確認事項3:モデルの評価 (混同行列:Accuracy)

経過時間 00:00:00:21 残り時間 --:--:-- 合計時間 00:00:00:21 リソース CPU x 1 Data 10000/10000

○ 評価結果 ● 混同行列 $y-y'$ ○ その他

○ 分類結果 $y-y'$ ○ 分類マトリックス $y-y'$: Recall ○ 尤度グラフ $y-y'$

Accuracy	0.9914
Avg.Precision	0.9914
Avg.Recall	0.9913
Avg.F-Measures	0.9914



Accuracy	0.9914
Avg.Precision	0.9914
Avg.Recall	0.9913
Avg.F-Measures	0.9914

	Recall	y'_0	y'_1	y'_2								
Precision		0.9929	0.9947	0.9866								
F-Measures		0.9954	0.9943	0.9913								
$y:\text{label}=0$	0.998	978	0	0	1	0	1	0	0	0	0	
$y:\text{label}=1$	0.9938	0	1128	2	2	0	0	1	0	2	0	
$y:\text{label}=2$												
$y:\text{label}=3$												
$y:\text{label}=4$	0.9929	0	0	0	0	975	0	2	0	0	5	
$y:\text{label}=5$	0.9877	2	0	0	6	0	881	1	0	1	1	
$y:\text{label}=6$	0.9906	2	2	1	0	1	2	949	0	1	0	
$y:\text{label}=7$	0.9844	1	2	8	0	0	0	0	1012	1	4	
$y:\text{label}=8$	0.9867	2	0	1	2	0	1	0	1	961	6	

□Accuracyを比較

試行錯誤の結果発表

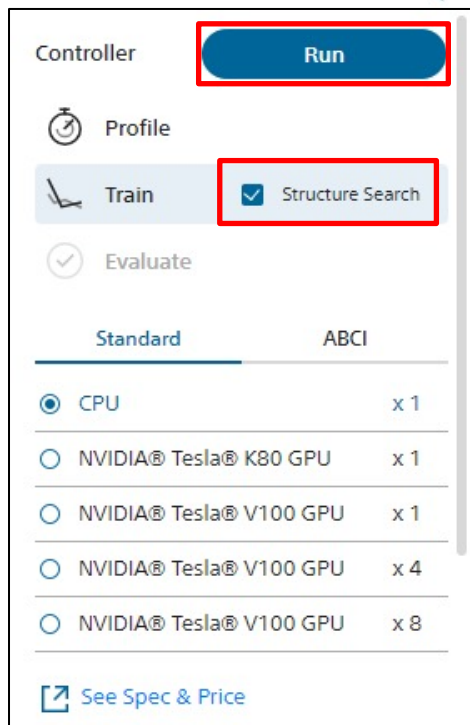
グループワーク発表

- 試行錯誤の結果を発表
 - 各グループのリーダーが説明(1分/グループ)
1. グループでベストの“Accuracy”
 2. 各パラメタと“Accuracy”の関係について「分かったこと」
 3. 議論した内容(考察)

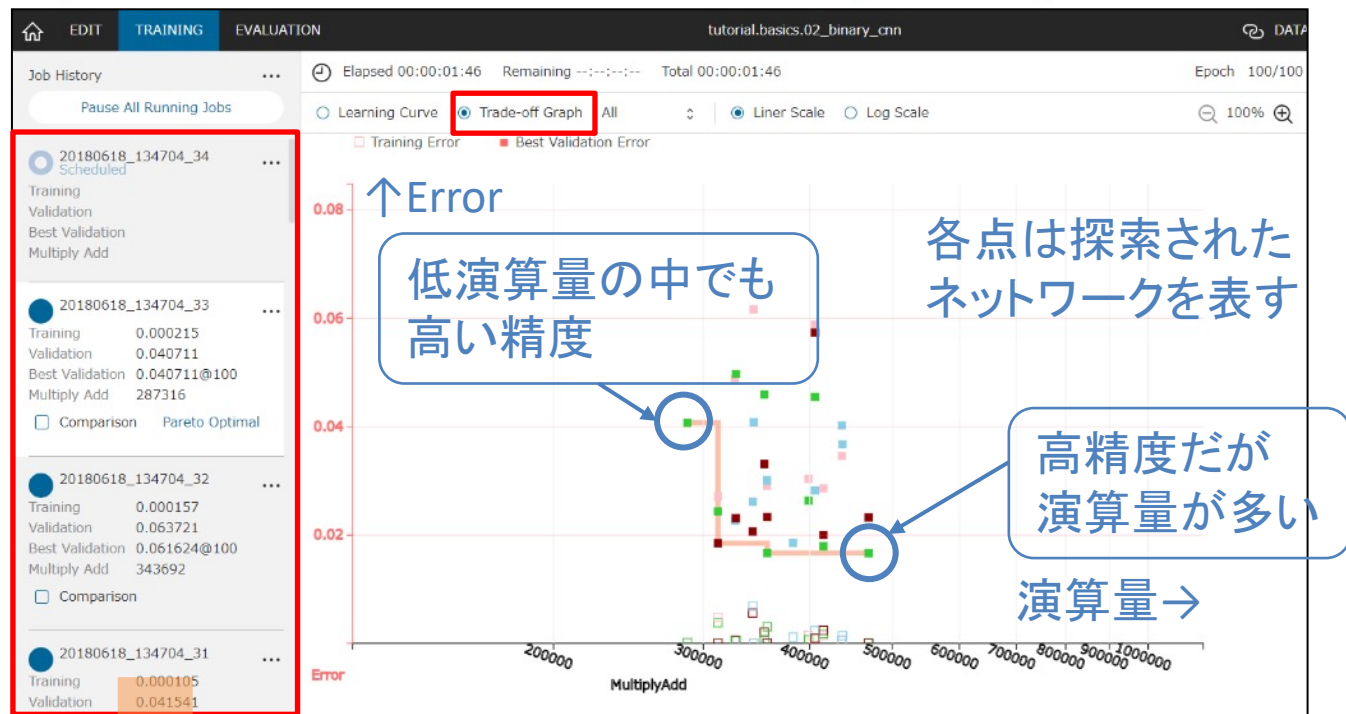
おまけ: 構造自動探索機能 最適なネットワークはデータに依存します

ネットワークの編集と学習が自動で繰り返され、ネットワーク作成の試行錯誤を自動的に行う。
EDITページのStructure Searchのチェックボックスを選択し、Runボタンで評価が実行される。
各モデルの結果はTRAININGページのTrade-off Graphで比較することが可能である。

構造探索機能の実行の方法



構造探索機能の結果



ネットワークを自動生成

おまけ: 自作手書き文字の認識

- 参考資料: 自作の手書き数字を使った分析
 - Windows「ペイント」を使って手書き文字画像を作成して分析する方法
 - 手書き文字の重心やサイズが大きく影響
 - MNISTのデータ:
 - アスペクト比を保ったまま20x20ピクセルに変換
 - 画像の重心を28x28ピクセルの画像の中心に重なるように配置
 - 前処理の影響がわかるサイト
 - https://kaityo256.github.io/mnist_check/
- ➡データの**前処理**って、実はとても大事なんです！
- 分析の7割、8割という分析者もいます。