ネットワークの試行錯誤による精度向上

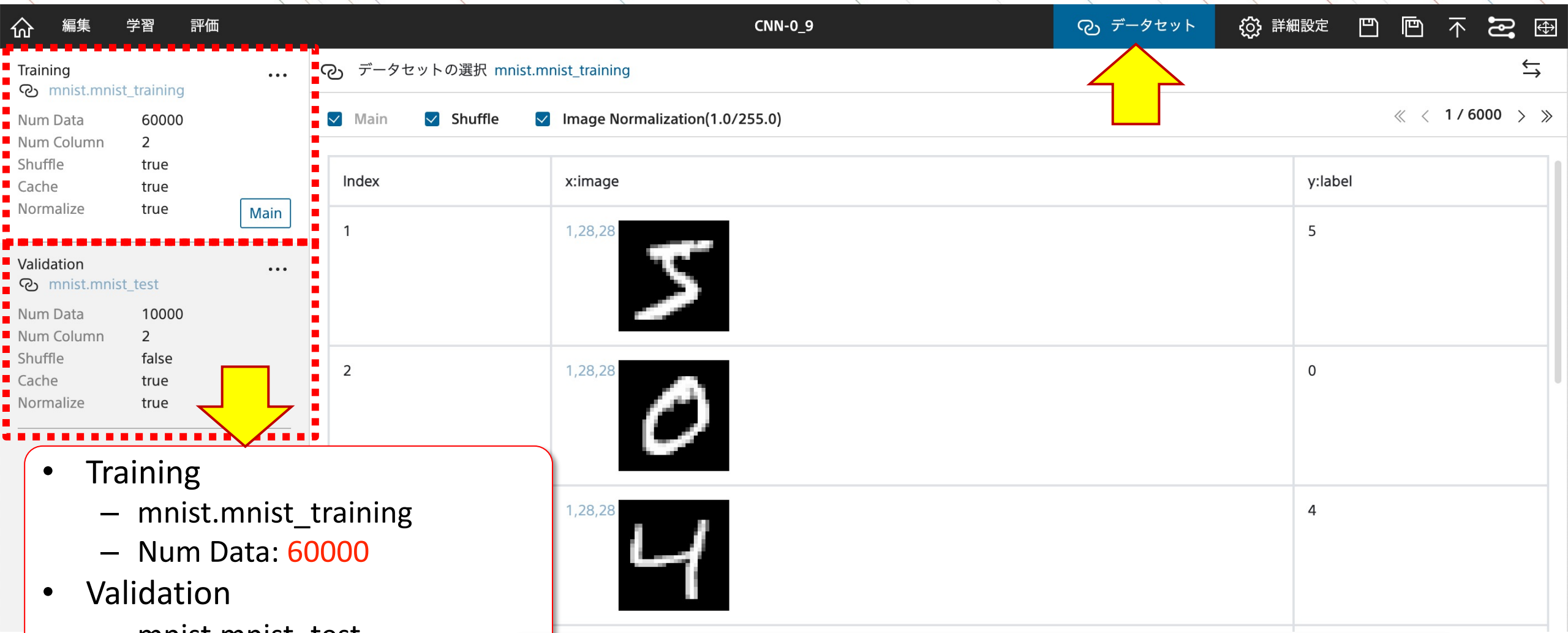
試行錯誤の結果発表

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

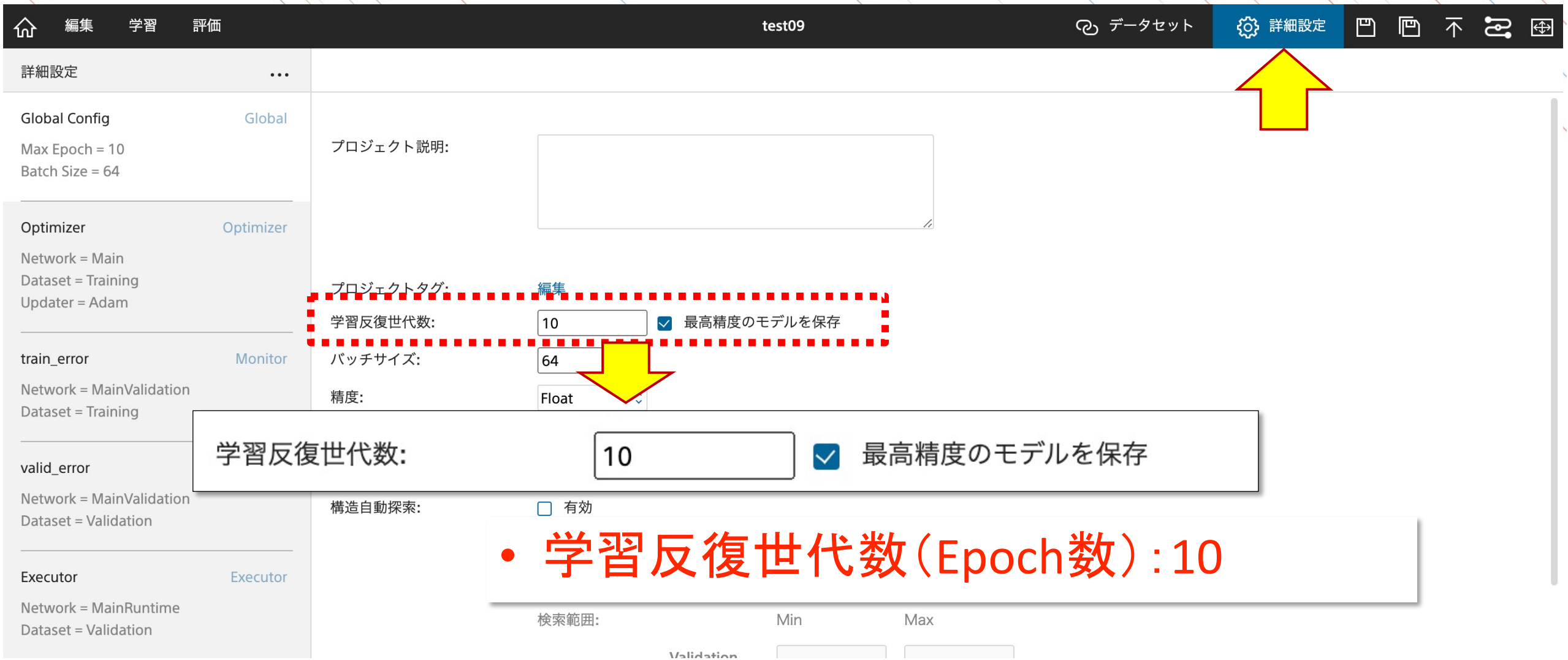
確認事項1：分析データセット（MNIST）



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

口異なる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)



- 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）

□Accuracyを比較

試行錯誤の結果発表

グループワーク発表

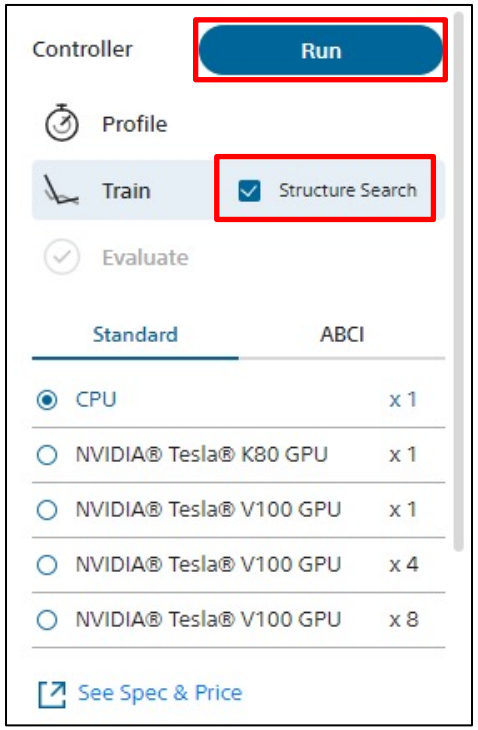
- 試行錯誤の結果を発表
  - 各グループのリーダーが説明（1分/グループ）

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）

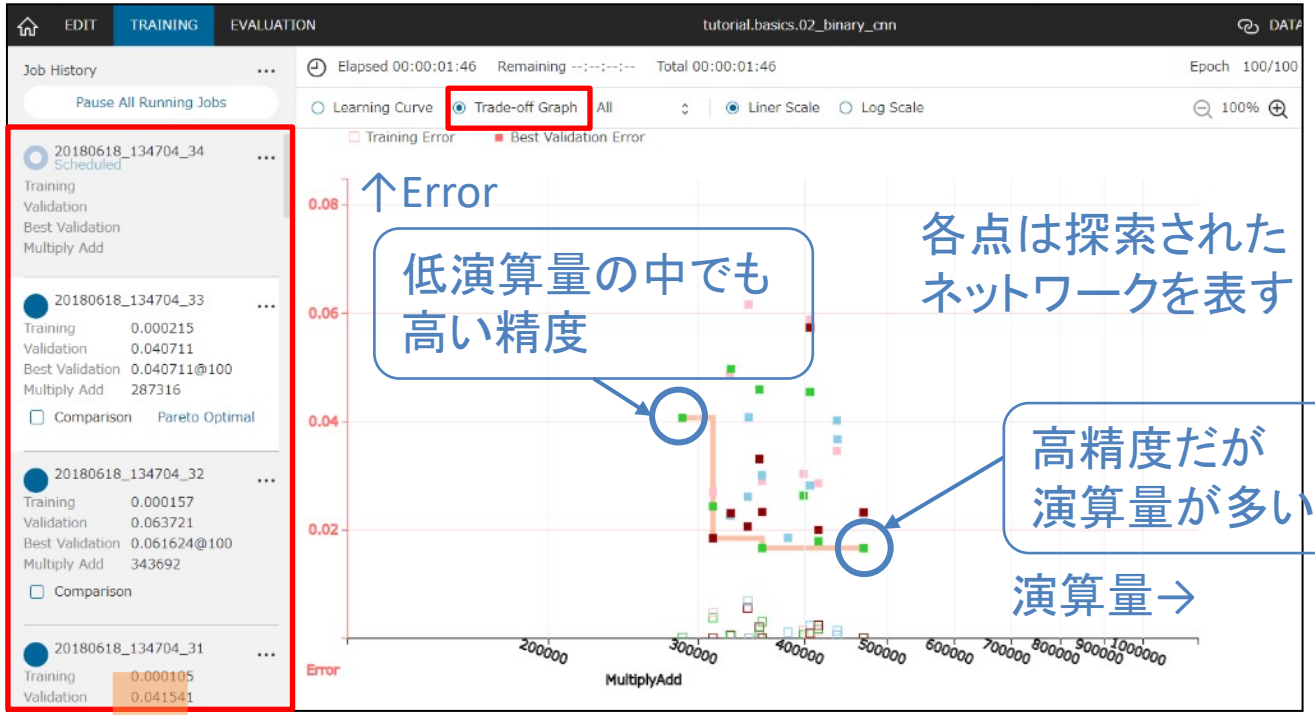
おまけ： 構造自動探索機能　最適なネットワークはデータに依存します

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.