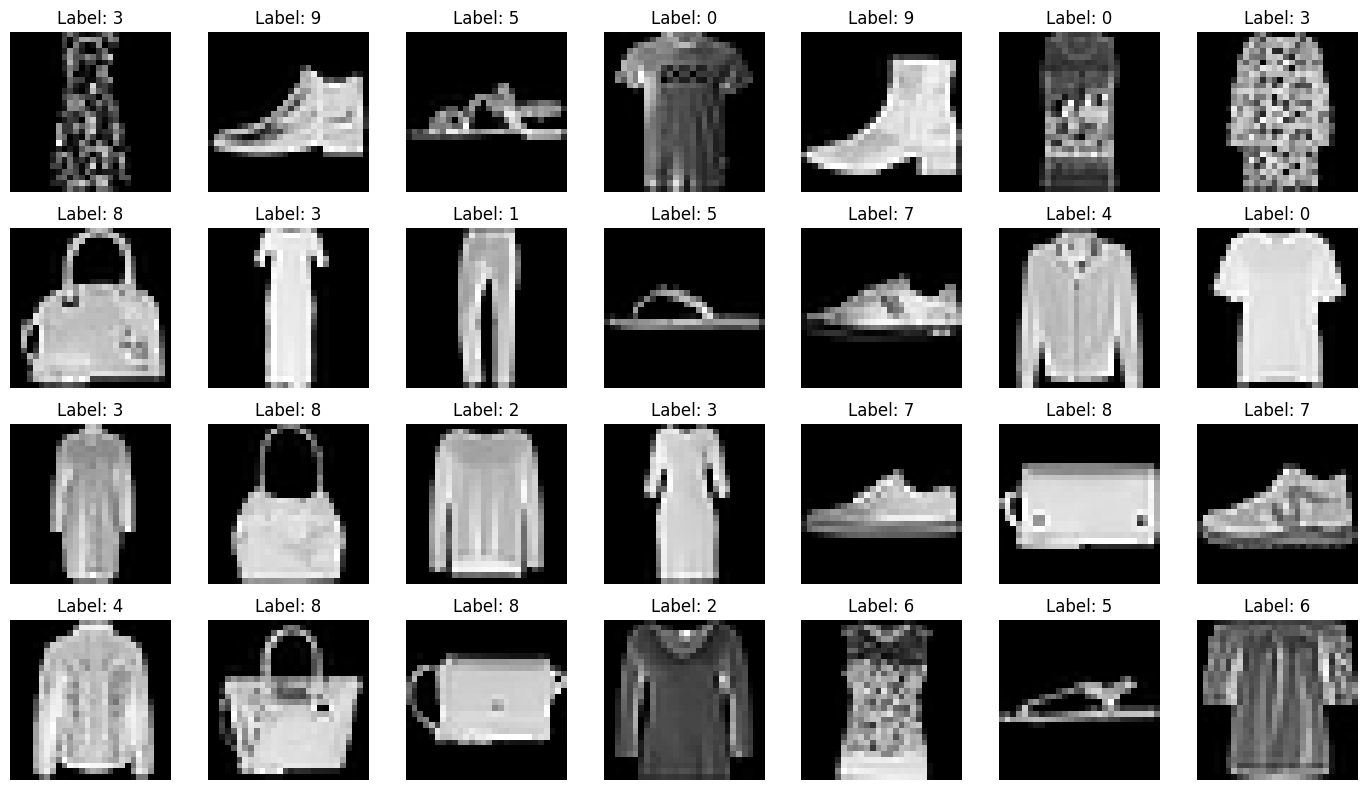
## 概要



FashionMNIST

目標：FashionMNISTを学習し、洋服や靴の種類を認識する深層学習モデルの開発

詳細：深層学習ライブラリPyTorchを利用して、モデル構造を自由に改善しましょう。モデルの定義を変更して実行すると、学習過程が表示されます。入力は28×28（画像サイズ）で、出力は10（ラベルの種類）なので、最初と最後は固定です。

# モデルの定義

model = nn.Sequential(

nn.Linear(28\*28, 64),

nn.Linear(64, 10)

)

# モデルを訓練して、テストする

train\_and\_test(model)

コンペの取り組み方：ノートブック [base\_code.ipynb](https://colab.research.google.com/drive/1Y_af_EmqqgPjVGNNWVFX1mb0hVc2nupC?usp=drive_link) のコピーを作成して、上記のコードを編集しましょう。モデル例は [model-example.ipynb](https://colab.research.google.com/drive/1PsQMYOSSBVg6FAURCzsgD4IPUNK_ThRA?usp=sharing) に記載されていますので、参考にすることがおすすめです。利用できる層の種類は次のページに記載されています。精度を高めたい方は、層の処理内容を理解することがコツです。

## モデル構築の方法

深層学習モデルは、層（Layer）を複数重ねることで精度を高められます。その組み合わせは自由ですが、今回のコンペでは下記の層に限定して取り組みましょう。

1. 線形層  
   特徴　：最も基本的な層で、線形変換を行います  
   書き方：nn.Linear(入力の数, 出力の数)  
   “”” 例えば  
    nn.Linear(28\*28, 256),  
    nn.Linear(256, 128),  
    nn.Linear(128, 64),  
    nn.Linear(64, 10),  
   “””  
   のように、入力の数は前の層の値を使う  
   線形層の最初は入力（画像サイズ）なので、28\*28  
   線形層の最後は出力（クラス数）なので、10
2. 正規化  
   特徴　：平均を0, 分散を1にすることでデータを整える  
   書き方：nn.BatchNorm1d(入力の数)  
   “”” 例えば  
    nn.Linear(28\*28, 256),  
    nn.BatchNorm1d(256),  
    nn.Linear(256, 128),  
    nn.BatchNorm1d(128),  
   “””
3. 活性化関数  
   特徴　：平均を0, 分散を1にすることでデータを整える  
   書き方：nn.Sigmoid() ※ シグモイド関数の場合  
   “”” 例えば  
    nn.Linear(28\*28, 256),  
    nn.Sigmoid(),  
    nn.Linear(256, 128),  
    nn.ReLU(),  
   “””  
   活性化関数は下記の3つから選んでください！  
    nn.Tanh()  
    nn.Sigmoid()  
    nn.ReLU()

## ルール

1. 「モデル構築の方法」で紹介された層のみを利用すること
2. [base\_code.ipynb](https://colab.research.google.com/drive/1Y_af_EmqqgPjVGNNWVFX1mb0hVc2nupC?usp=drive_link) を元にしてモデルの定義と、学習を実行すること
3. 学習の1エポック実行時間は20秒以下であること（超えると学習が停止されます）  
   学習ログに経過時間が表示されるので、調節してみましょう
4. モデルの構築例をネットで検索することはOKですが、今回のデータセットFashionMNISTを学習対象にしたコードは閲覧しないこと

学習がうまくいったコードと、出力結果は残しておくか、写真を撮っておきましょう。コンペ終了時にどのチームが高精度かを判断します。