実験：畳み込みありとなしの比較

* 概要
  + まず、現在使っているモデルは全結合層のみで畳み込み層を含んでいない。そこで、モデルに畳み込み層を実装してみてみたい。
* 目標
  + 畳み込み層の実装
  + 汎化性能にどのような影響があるのかを確認 <- 畳み込み層実装前のモデルのデータと比較する必要がある。
* それぞれの条件
  + 共通条件
    - シード
      * 20201218 #int
    - 学習用データ
      * 129
    - ニューラルネットワーク
      * 入力層
        + width:80 x height:45 x framesize:30 x COLOR:RGB=3
      * 出力層
        + サイズ：４
        + 活性化関数：softmax
    - 最適化
      * 最適化アルゴリズム
        + adam

lr=1e-7

beta\_1=0.9

beta\_2=0.999

epsilon=None

decay=0.0

amsgrad=False

* + - 訓練方法
      * バッチサイズ ：16
      * エポック数：512
  + 畳み込み**なし**モデルの条件
    - ニューラルネットワーク
      * 中間層
        + dence0

ユニットサイズ：1024

* + 畳み込み**あり**モデルの条件
    - ニューラルネットワーク
      * 中間層
        + conv0

フィルタ数：32

カーネルサイズ：(3,3,3), (2,3,3)

ストライド：(1,1,1), (2,3,1)

padding : “same”

* + - * + pool0

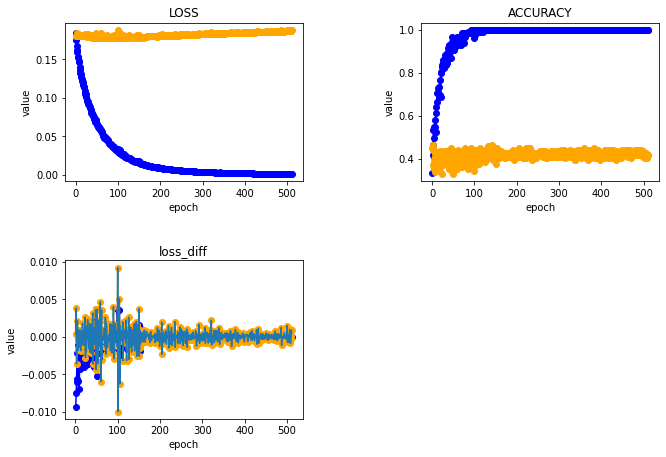
プーリング方法：maxpooling

サイズ：(3,3,3), (2,3,3)

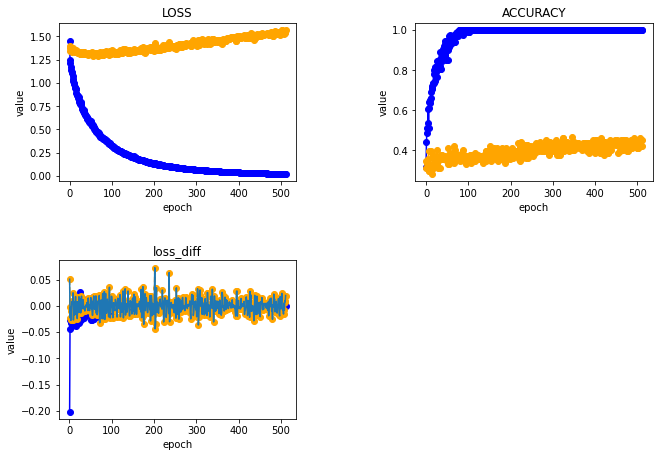
* + - * + dence0

ユニットサイズ:1024

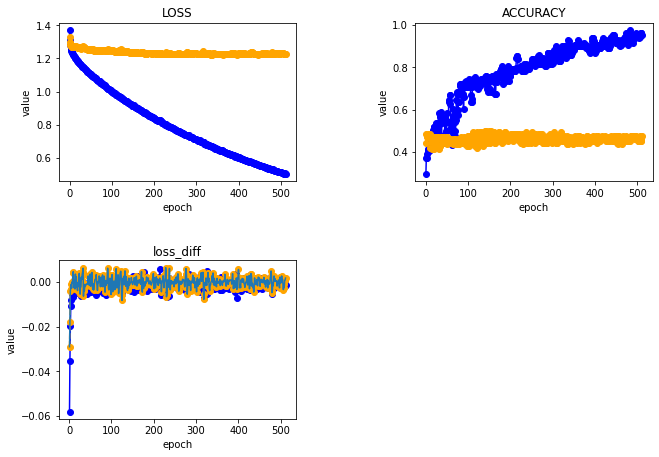
* 結果の評価
  + 畳み込み**なし**モデルの結果
    - loss = mse
      * 図



* + - * 評価
        + ヒューマンエラーはなさそう
    - loss = cee
      * 図



* + - * 評価
        + val\_lossが上がっている
        + 序盤のloss\_diffが高い
  + 畳み込み**あり**モデルの結果
    - loss=cee
      * 図



* + - * 評価
        + なしのモデルに比べて、ほんの少しだけval\_accの収束値が高い
        + loss, acc が収束していない
        + プロットの点が大きい
* 考察
  + 畳み込み**なし**モデル
    - lossの収束を踏まえ、acc, val\_accの収束値から、過学習していると判断できる
    - 誤差関数の違いが分かりづらい
  + 畳み込み**あり**モデル
    - こちらも過学習している。
    - val\_lossが下がっていること、lossが収束していないことから、まだ学習できる
* 結果
  + 畳み込み層はあったほうがいい。なぜなら、**ない**場合と比較して汎化性能が僅かに高いからだ。
  + 今回の条件（特にエポック数：512）において、学習完了していないと考えられる。なぜなら、loss値が収束してないからだ。->エポック数を1024とかにしてみる。
  + loss値の収束の基準を決めなければならない。
  + dropoutを実験してみたい。＼