Sansan

BIG DATA UNIVERSITY

データサイエンティスト実践セミナー
Python/Sparkで始める分析の実際

ディープラーニングで名刺を解析する

Sansan株式会社

高際睦起

2016-08-26

発表の内容と目的

- 開催中の画像認識コンペ紹介
  - 名刺に書かれた項目(名前、会社名、、、)を推定

- Deep Learningを使ったベンチマークコード公開中

  スコアボードのbenchmarkとは違います。おそらく0.015～0.02程度のスコア(未計測)

  https://github.com/takagiwa-ss/deepanalytics_compe26_benchmark

  最新の手法を出来るだけ平易に書いたつもりです

  ※弊社Sansanで実際に使っている手法とは異なります

- ベンチマークコードを例に画像認識の方法を解説

› コンペ内容の紹介

› データ

名刺画像

矩形情報・正解ラベルデータ

filename	left	top	right	bottom	company_	full_name	position_r	address	phone_nu	fax	mobile	email	url
2842.png	491	455	796	485	0	0	0	0	0	0	1	0	0
182.png	24	858	311	886	0	0	0	0	0	0	1	0	0
95.png	320	498	865	521	0	0	0	0	0	1	1	0	0

訓練用のデータ = 例題

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	filename	left	top	right	bottom	company	full_name	position_r	address	phone_nu	fax	mobile	email	url
2	2842.png	491	455	796	485	0	0	0	0	0	0	1	0	0
3	182.png	24	858	311	886	0	0	0	0	0	0	1	0	0
4	95.png	320	498	865	521	0	0	0	0	0	1	1	0	0
5	2491.png	65	39	497	118	1	0	0	0	0	0	0	0	0
6	3301.png	271	83	333	463	0	1	1	0	0	0	0	0	0
7	3240.png	441	45	985	96	1	0	0	0	0	0	0	0	0
8	1102.png	338	197	637	259	0	1	0	0	0	0	0	0	0
9	2141.png	344	817	588	843	0	0	0	0	1	0	0	0	0
10	3595.png	354	262	674	326	0	1	0	0	0	0	0	0	0
11	1787.png	156	224	389	249	0	0	0	0	0	1	0	0	0
12	2022.png	58	43	813	103	1	0	0	0	0	0	0	0	0
13	3577.png	517	266	789	329	0	1	0	0	0	0	0	0	0

評価

	A	B	C	D	E	F	G
1		filename	left	top	right	bottom	
2	0	1942.png	57	114	361	173	
3	1	1128.png	58	373	519	422	
4	2	2719.png	62	289	297	314	
5	3	641.png	58	668	416	747	
6	4	2529.png	42				

評価用データは、画像と矩形座標のみ、項目ラベルは無し

コンペ参加者には項目ラベルを予測して頂きます

9種の項目それぞれ0～1の実数

Deep Learningを使った名刺の画像認識の方法
と
ベンチマークコードの解説

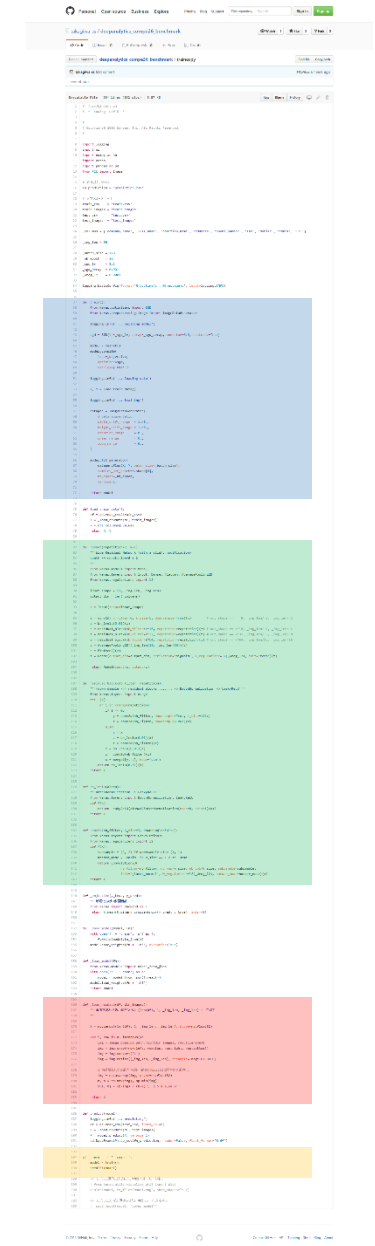
› ベンチマークコードの処理概要

1. 訓練用のcsv, png画像を読み込み、
   訓練データとして格納
   - csvからpngファイル名と矩形座標、及び項目ラベルを得る
   - 訓練に使えるように画像を加工

2. 学習に使うDeep Learningモデルを定義し、
   訓練データでモデルを訓練
   - ライブラリkerasを使う

3. 評価用のcsv, png画像を読み込み、予測を出力

ソースコード概観



訓練

Deep Learningモデルの定義

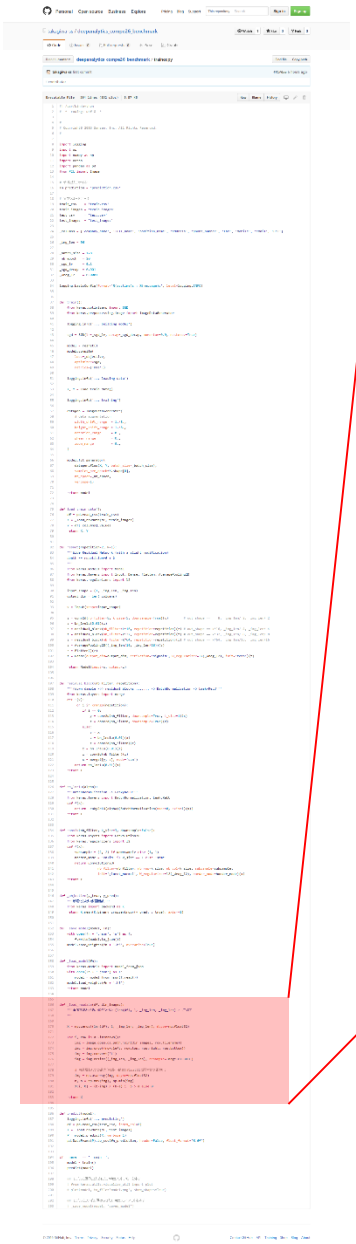
画像の読み込み加工

ベンチマークコードの処理概要

1. 訓練用のcsv, png画像を読み込み、
   訓練データとして格納
   - csvからpngファイル名と矩形座標、及び項目ラベルを得る
   - 訓練に使えるように画像を加工

2. 学習に使うDeep Learningモデルを定義し、
   訓練データでモデルを訓練
   - ライブラリkerasを使う

3. 評価用のcsv, png画像を読み込み、予測を出力



```python
def _load_rawdata(df, dir_images):
    '''画像を読み込み、4Dテンソル (len(df), 1, _img_len, _img_len) として返す
    '''

    X = np.zeros((len(df), 1, _img_len, _img_len), dtype=np.float32)

    for i, row in df.iterrows():
        img = Image.open(os.path.join(dir_images, row.filename))
        img = img.crop((row.left, row.top, row.right, row.bottom))
        img = img.convert('L')
        img = img.resize((_img_len, _img_len), resample=Image.BICUBIC)

        # 白黒反転しつつ最大値1最小値0のfloat32に画素値を正規化
        img = np.asarray(img, dtype=np.float32)
        b, a = np.max(img), np.min(img)
        X[i, 0] = (b-img) / (b-a) if b > a else 0

    return X
```

画像を扱う定番の方法

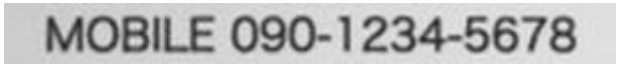
1. 本コンペのように、画像中の一部の矩形領域を使う場合、
   矩形内の画像を切り抜く

2. 画像サイズを合わせる
   矩形のサイズは**ばらばら**
   リサイズして大きさをそろえ、扱いやすくする

3. 値のレンジをそろえる
   輝度値(0～255のレンジ)を浮動小数(0～1)に変換
   Deep Learningの計算を安定化

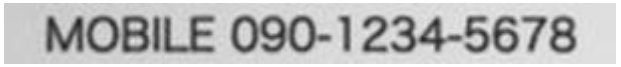
   このコードではやってないが、平均値を引く、分散でわる、こともよくある

画像→訓練データの具体例



画像→訓練データ(1)画像切り抜き

PILのcropで矩形座標を指定して画像切り抜き



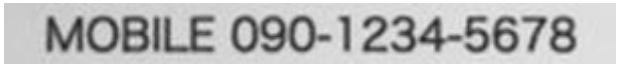
画像→訓練データ(2)サイズ変更

PILresizeで固定サイズに変換

長方形を無理やり正方形にする

Deep Learningでは正方形にするのはよくやる方法

歪み激しいが学習してみたら
そこそこ精度出たのでまぁいいか

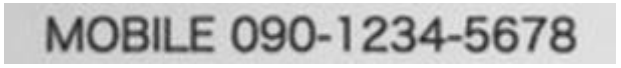


画像→訓練データ(3)レンジの正規化

pixel値の範囲(0～255)を
0～1の浮動小数に変換、白黒反転

float32を使うのはGPUの都合

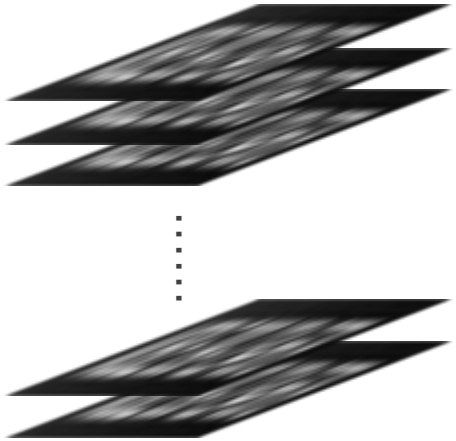
値のレンジをそろえないと
計算が発散して学習がうまく
うごかないことがある



画像→訓練データ

全画像を読み込んで、全画像(X)と項目ラベル(Y)を積み重ねて
学習データとする

$$X = \begin{matrix} \\ \\ \\ \vdots \\ \\ \\ \end{matrix}$$



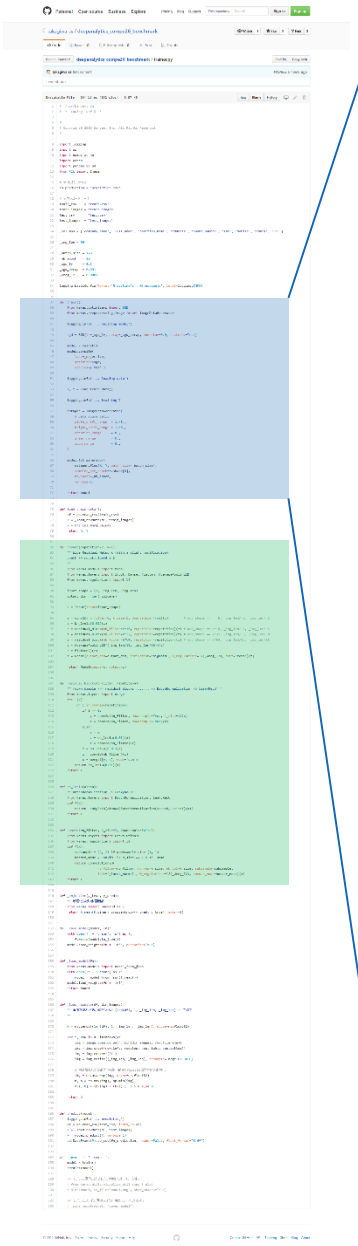
$$Y = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

› ベンチマークコードの処理概要

1. 訓練用のcsv, png画像を読み込み、
   訓練データとして格納
   - csvからpngファイル名と矩形座標、及び項目ラベルを得る
   - 訓練に使えるように画像を加工

2. 学習に使うDeep Learningモデルを定義し、
   訓練データでモデルを訓練
   - ライブラリkerasを使う

3. 評価用のcsv, png画像を読み込み、予測を出力

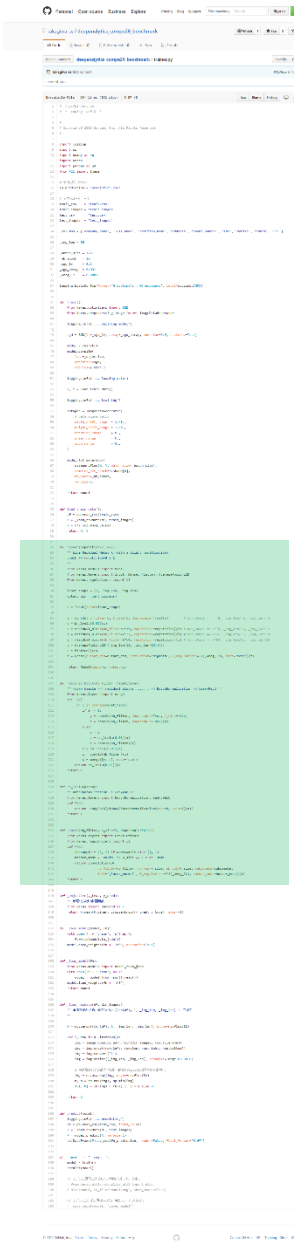


```python
def train():
    from keras.optimizers import SGD
    from keras.preprocessing.image import ImageDataGenerator

    logging.info('... building model')

    sgd = SGD(lr=_sgd_lr, decay=_sgd_decay, momentum=0.9, nesterov=True)

    model = resnet()
    model.compile(
        loss=_objective,
        optimizer=sgd,
        metrics=['mae'])

    logging.info('... loading data')

    X, Y = load_train_data()

    logging.info('... training')

    datagen = ImageDataGenerator(
        # data augmentation
        width_shift_range  = 1./8.,
        height_shift_range = 1./8.,
        rotation_range     = 0.,
        shear_range        = 0.,
        zoom_range         = 0.,
    )

    model.fit_generator(
        datagen.flow(X, Y, batch_size=_batch_size),
        samples_per_epoch=X.shape[0],
        nb_epoch=_nb_epoch,
        verbose=1)

    return model
```

Deep Learningのモデル
と最適化の方法を定義

“データ拡張”の機能を使う

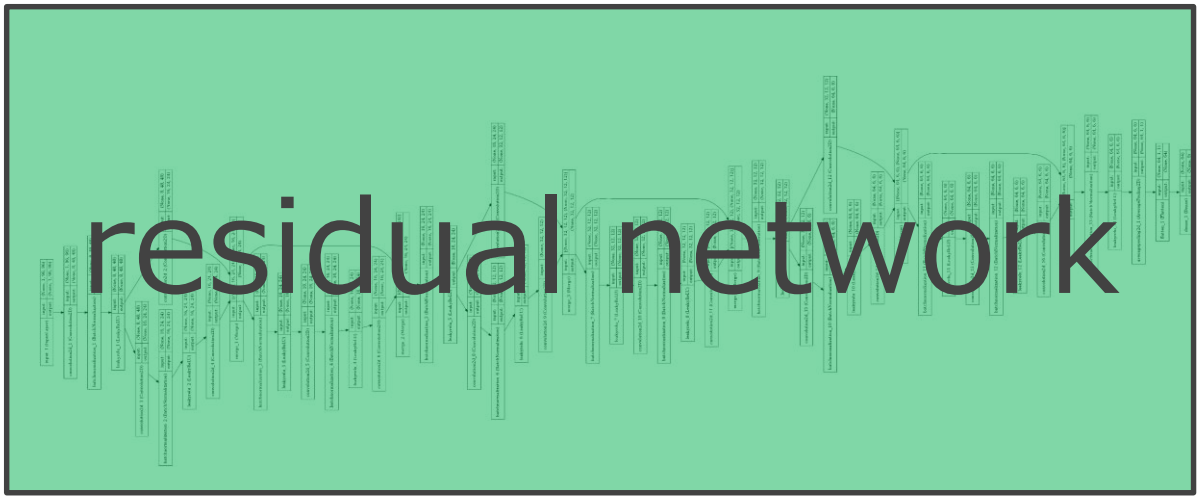
訓練を行う



Deep Learningのモデル定義

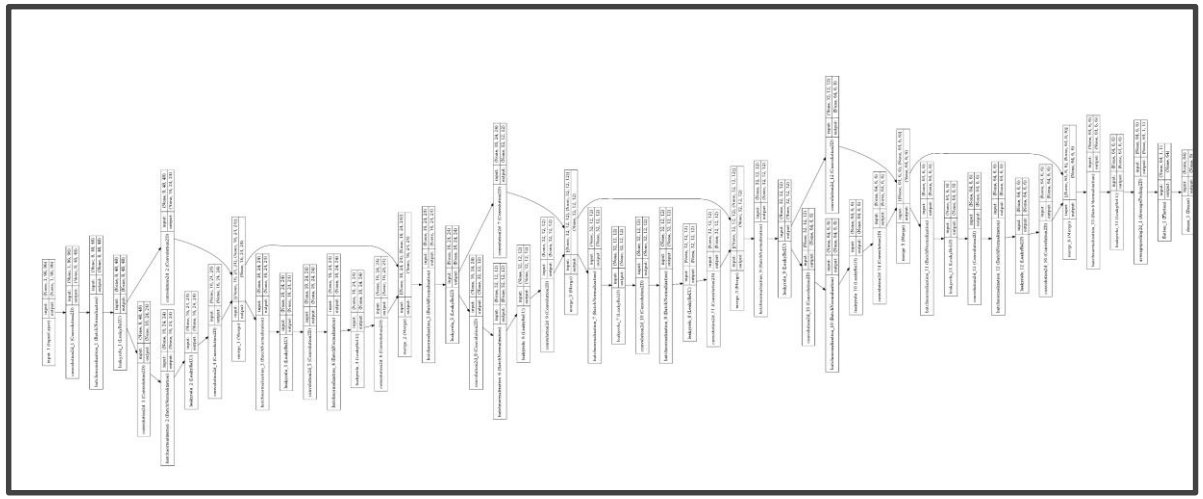
residual networkはDeep Learningの一種

2016年時点、画像認識でトップの性能をもつ



ライブラリkerasを使って、Deep Learningモデルを定義

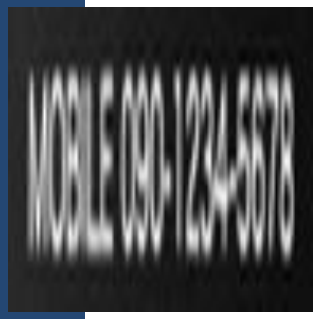
□と→の繋がり方を作る

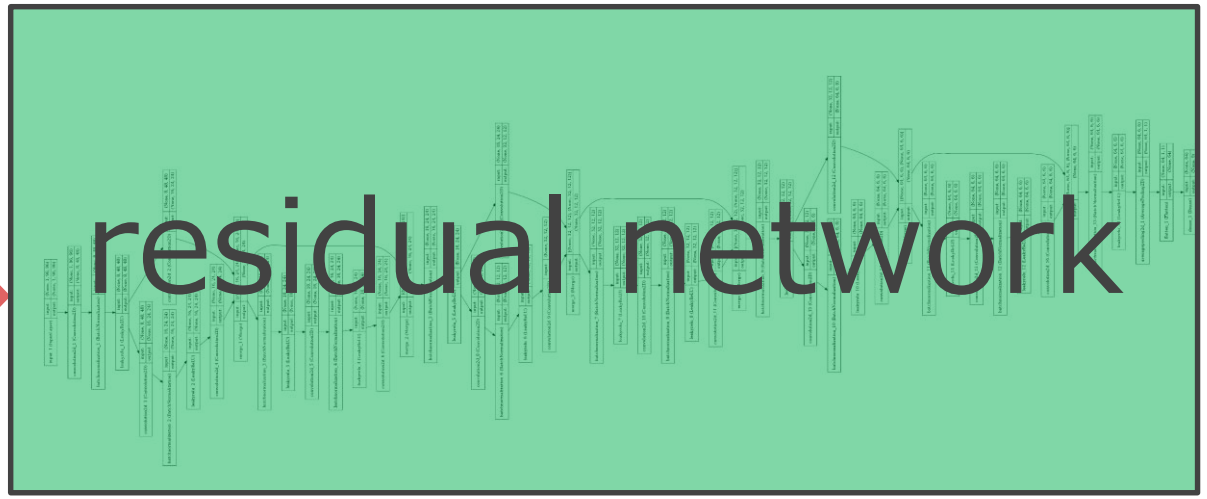


この発表では詳しい説明は省略

モデルの定義

固定サイズの行列＝画像(96x96, グレー画像)を入力すると

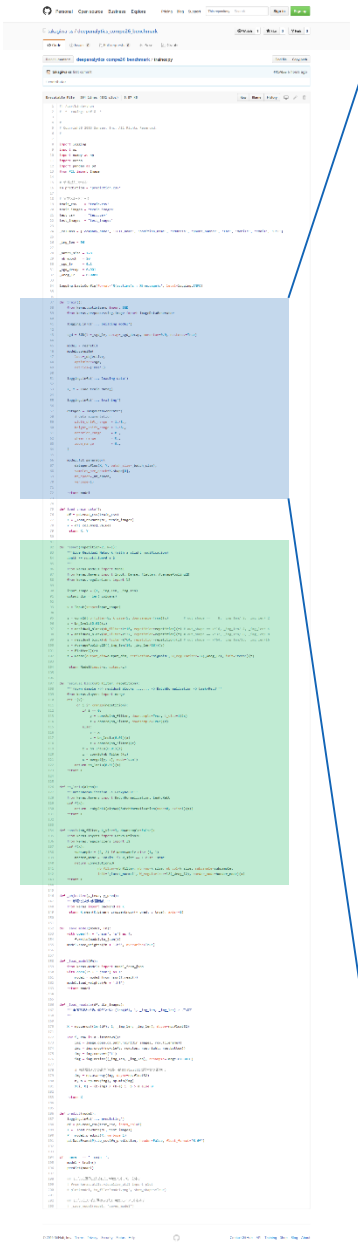




$$\begin{bmatrix} 0.068 \\ 0.146 \\ 0.128 \\ 0.070 \\ 0.044 \\ 0.100 \\ 0.937 \\ 0.064 \\ 0.231 \end{bmatrix}$$

項目(9種)ごと”らしさ”＝予測を9次元ベクトル0～1で出力

と、するようにresidual networkのモデルを作る



```
37  def train():
38      from keras.optimizers import SGD
39      from keras.preprocessing.image import ImageDataGenerator
40
41      logging.info('... building model')
42
43      sgd = SGD(lr=_sgd_lr, decay=_sgd_decay, momentum=0.9, nesterov=True)
44
45      model = resnet()
46      model.compile(
47          loss=_objective,
48          optimizer=sgd,
49          metrics=['mae'])
50
51      logging.info('... loading data')
52
53      X, Y = load_train_data()
54
55      logging.info('... training')
56
57      datagen = ImageDataGenerator(
58          # data augmentation
59          width_shift_range  = 1./8.,
60          height_shift_range = 1./8.,
61          rotation_range     = 0.,
62          shear_range        = 0.,
63          zoom_range         = 0.,
64      )
65
66      model.fit_generator(
67          datagen.flow(X, Y, batch_size=_batch_size),
68          samples_per_epoch=X.shape[0],
69          nb_epoch=_nb_epoch,
70          verbose=1)
71
72      return model
73
```

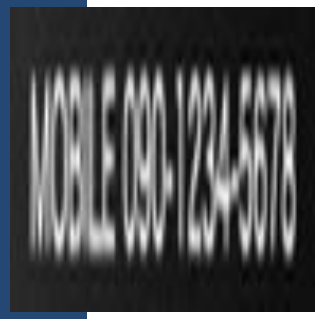
Deep Learningのモデルと最適化の方法を定義

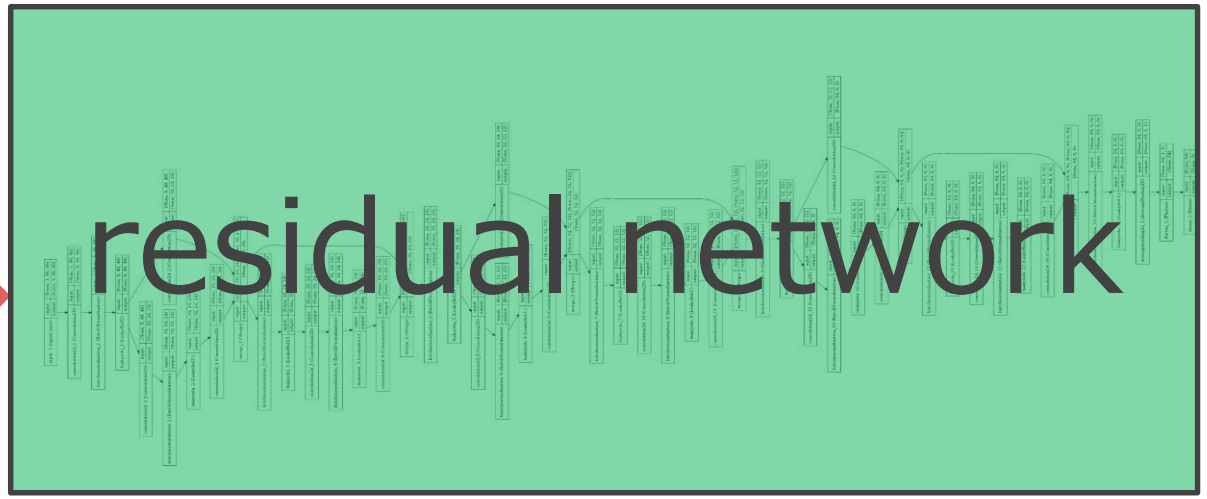
kerasの“データ拡張”の機能を使う

訓練を行う

学習

fit_generatorを呼んで、画像と項目ラベルの対応を覚えさせる = 学習





出力

$$\begin{bmatrix} 0.068 \\ 0.146 \\ 0.128 \\ 0.070 \\ 0.044 \\ 0.100 \\ 0.937 \\ 0.064 \\ 0.231 \end{bmatrix}$$

出力と正解ラベルの差を縮めるように
networkを更新する

正解ラベル

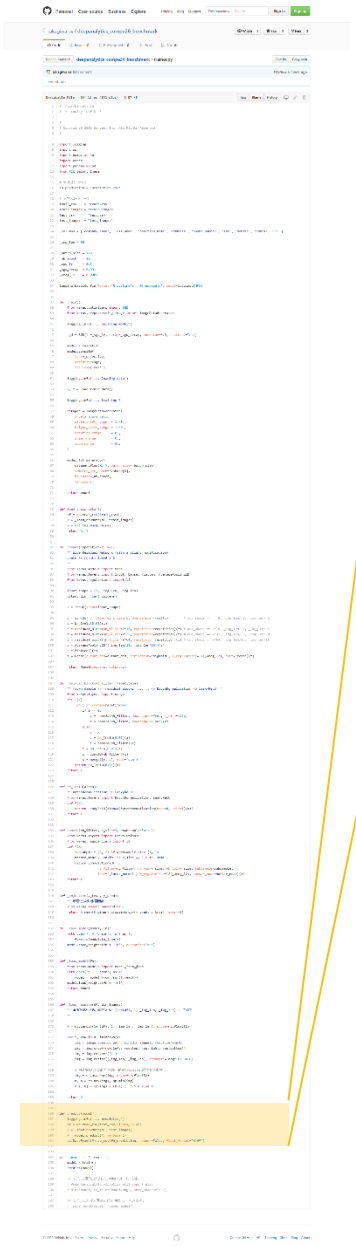
$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

› ベンチマークコードの処理概要

1. 訓練用のcsv, png画像を読み込み、
   訓練データとして格納
   - csvからpngファイル名と矩形座標、及び項目ラベルを得る
   - 訓練に使えるように画像を加工

2. 学習に使うDeep Learningモデルを定義し、
   訓練データでモデルを訓練
   - ライブラリkerasを使う

3. 評価用のcsv, png画像を読み込み、予測を出力



```
185
186 def predict(model):
187     logging.info('... predicting')
188     df = pd.read_csv(test_csv, index_col=0)
189     X = _load_rawdata(df, test_images)
190     P = model.predict(X, verbose=1)
191     pd.DataFrame(P).to_csv(fn_prediction, header=False, float_format='%.6f')
192
```

1. 訓練の時と同じように評価用の画像を加工
2. 学習したモデルで predict を呼ぶと全予測を行列として出力
3. 行列をcsvファイルに書き出し

› ベンチマークコードの改良ポイント

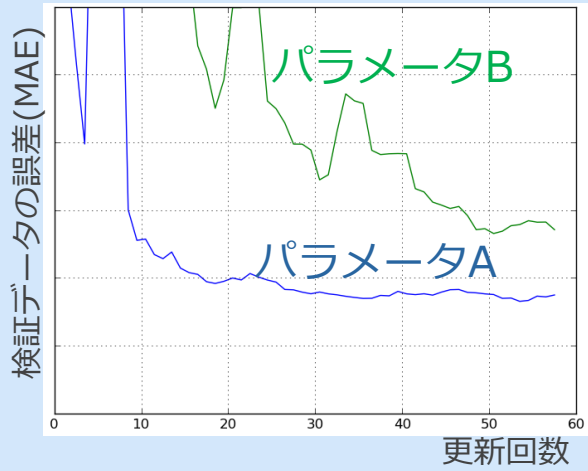
※公開したベンチマークコードコードで用いているパラメータ類は最適値ではありません※

- 訓練データを分割(80%:20%)し、
  訓練用(80%)と検証用(20%)として使えるようにする
  - 訓練に使っていない検証データで精度を評価する
  - 次ページ以降のようなパラメータをチューニングするときに、手元で精度評価して、パラメータの良し悪しを判断

› ベンチマークコードの改良ポイント

- 最大更新回数(_nb_epoch)を増やす
  - モデルを更新するループの回数
  - ベンチマークでは20回の更新で終了
    最適値で無い
  - ベストの値は他のパラメータや
    最適化方法にも依存

パラメータを変えた学習例



- 正規化画像サイズ(_img_len)を大きく
  - ベンチマークの96x96の画像では文字が小さい
  - 辺長128～512程度がよさそうかな?
    コードの都合上 16 の倍数

ベンチマークコードの改良ポイント

- Deep Leaningのモデルを深く、幅広く
  - 構造を定義するパラメータ２個
    (trainer.pyの45行目: model = resnet())
    - 深さパラメータ(repetition=2) □の数
    - 幅広さパラメータ(k=1) □の大きさ
  - repetition=2～5, k=2～8程度がよさそうかな?
  - 過適合を抑える
    - Dropout
    - データ拡張(trainer.pyの59～63行目: ImageDataGenerator)
      みかけの訓練データ数を増やす
  - Deep Learning職人とは、Deep Learningの気持ちを理解してパラメータチューニングがうまくできる人

Sansan 主催のデータサイエンティスト向け分析コンテスト @Datapalooza Tokyo

人工知能は名刺をどこまで解読できるのか

人工知能　名刺

手のひらサイズの小さな画像に特化した画像解析の限界に挑む

開催期間 ：2016年8月8日～9月30日

懸賞金：　1位 **30万** 2位 **20万** 3位 **10万**

Sansan　　日本アイ･ビー･エム株式会社　　OPT DataScienceLab

ご参加お待ちしております

Bibliography

- Keras https://keras.io/
  - https://github.com/fchollet/keras/tree/master/examples
    にある畳み込みニューラルネットワーク例
    - mnist_cnn.py 手書き数字0～9の画像認識
    - cifar10_cnn.py 一般物体10種(犬、猫、船、…)の画像認識
    - resnet_50.py 一般物体認識の学習済みモデルを使う例
- Deep Learning(Residual Network)関連
  - Deep Residual Learning for Image Recognition(2015)
    Identity Mappings in Deep Residual Networks(2016)
    Wide Residual Networks(2016)
    Weighted Residuals for Very Deep Networks(2016)
  - Residual Network作者Kaiming HeのICML2016での解説スライド
    http://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf
  - 産総研片岡さんの解説スライド
    http://www.slideshare.net/HirokatsuKataoka/deep-residual-learning-ilsvrc2015-winner