2019.10.08

# TCNを用いてFX予測してみる



こんにちは。次世代システム研究室のT.D.Qです。 最近、業務でGPUクラウド上にDeep Learning環境を構築したり、ビッグデータ解析で金融データを触ったりしていますので、この技術を使ってFX予測できないかと思い始めました。同僚がFXの予測をテーマとして複数のブログを書いていますが、時系列処理は再帰型ニューラルネットワーク(*RNN*)、特にLSTMが優れている印象を受けました。関連する技術を調査した結果、LSTMより精度が高いというTCNを発見しましたので、今回はTCNを用いてFXやってみたいと思います。

## 実行環境

今回はGPUクラウド by GMOの１GPUプランで下記のマシンで、nvidia-docker、Googleが提供するtensorflow:latest-gpu-jupyterイメージを使って機械学習の開発環境を構築しました。ちなみに、マシンのスペックは下記の通りで、少しハイスペックですね。

| | | |
|---|---|---|
| 基本仕様 | GPUカード | NVIDIA® Tesla® V100 SXM2（16GB） |
| | GPUカード搭載数 | 1 |
| | GPU搭載メモリ | 16GB |
| | 単精度浮動小数点数演算 | 約15.7TFLOPS |
| | 倍精度浮動小数点数演算 | 約7.8TFLOPS |
| | NVIDIA® Tensorコア数 | 640 |
| | NVIDIA® CUDA®コア数 | 5,120 |
| | 標準OS | Ubuntu 18.04 LTS/ CentOS 7.5 |
| | CPU | 16vCPU（Intel® Xeon® Gold 5122 4コア 3.60GHz ×2） |

実際にコマンドでGPUマシンの開発環境を確認してみましょう。

```
> nvidia-smi
Wed Sep 25 13:29:59 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.72       Driver Version: 410.72       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:00:05.0 Off |                    0 |
| N/A   47C    P0    54W / 300W |   8785MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

## TCNの概要

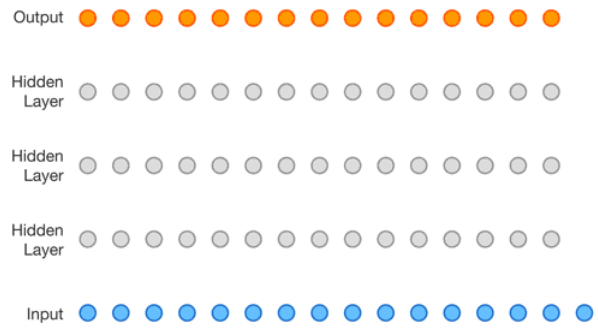
TCNが「**Temporal Convolutional Networks**」の略でAn Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modelingで紹介されました。畳み込みニューラルネットワーク(CNN)のアーキテクチャーにRNNの特徴を加えて時系列データを処理します。TCNの特徴には下記の２点があります。

1. どんな長さのInputでも同じ長さのOutputを作成できる。またOutputはInputより短くても可能。
2. 畳み込み(Convolutional)構造がCausalで、情報が未来から過去に”漏れる”（leakage）ことがない

1を保証するため1D fully-convolutional network (FCN) architectureを使います。これで、ネットワークのhidden layerもinput layerと同じ長さとなります。また、 (kernel size − 1)ゼロパディングという手法を使って、次のシーケンスをInputシーケンスの長さと等しくします。 また、 2を保証するため、時刻tのOutputはそれ以前のデータのみを利用します。これは「**causal convolutions**」と言います。つまり、「**TCN = 1D FCN + causal convolutions**」で、挙動は下記の画像を見た方が分かりやすいですね。



検索:
検索




### TCNの強い点

- 並列処理が可能：これがRNNと違うところです。TCNは同じFilterを各Layerに適用するため、長いInputシーケンスを同時に処理することができます。
- トレーニングの際にRNN（LSTM、GRUなど）より必要なメモリが少ない。これは各Layerに同じFilterを共有するためです。この特徴を持つことでメモリにあまり余裕がなくてとも長いInputを処理可能です。
- 時系列からパータンを学習する以外にも各Layerでデータから特徴的なパターンを発見可能。

## 学習の準備

### データ入手

Forexのデータはhistdata.comから[ドル円の2019年のレートデータ]を入手しました。無料で[複数のFormat]（MetaTrader, MetaStockなど）を提供しているので非常に便利です。今回は2019年1月から7月までのレートデータをダウンロードして集約しました。

```
tmp = []
for fname in sort(glob.glob('../data/DAT_ASCII_USDJPY_T_*.csv')):
    print(fname)
    tmp.append(pd.read_csv(fname,
                names=['datetime_stamp', 'bid_quote', 'ask_quote',
'volume']))

df = pd.concat(tmp)

../data/DAT_ASCII_USDJPY_T_201901.csv
../data/DAT_ASCII_USDJPY_T_201902.csv
../data/DAT_ASCII_USDJPY_T_201903.csv
../data/DAT_ASCII_USDJPY_T_201904.csv
../data/DAT_ASCII_USDJPY_T_201905.csv
../data/DAT_ASCII_USDJPY_T_201906.csv
../data/DAT_ASCII_USDJPY_T_201907.csv
```

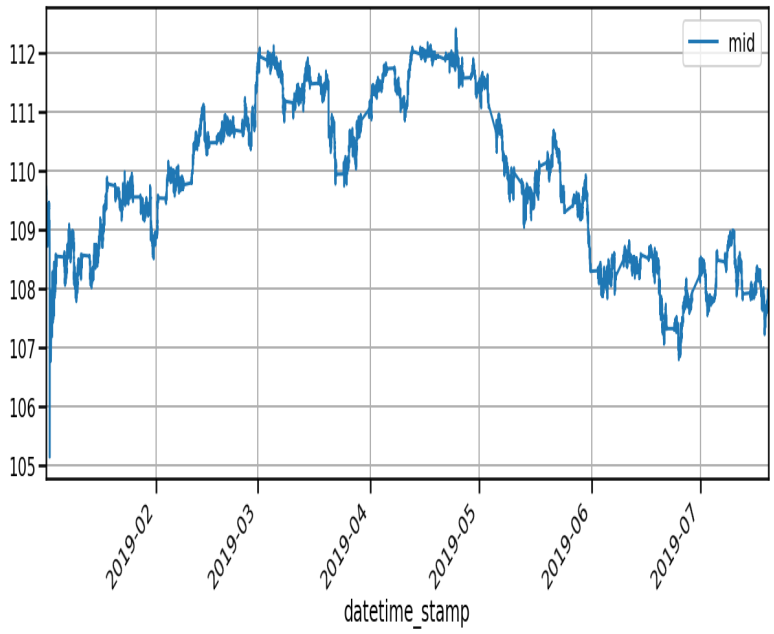
### データの前処理

・Tickデータとして入手しましたが、実際データを確認してみると欠損が多くてTickデータとしては使えなさそうですので、1分足のデータにDownSamplingを行いました。

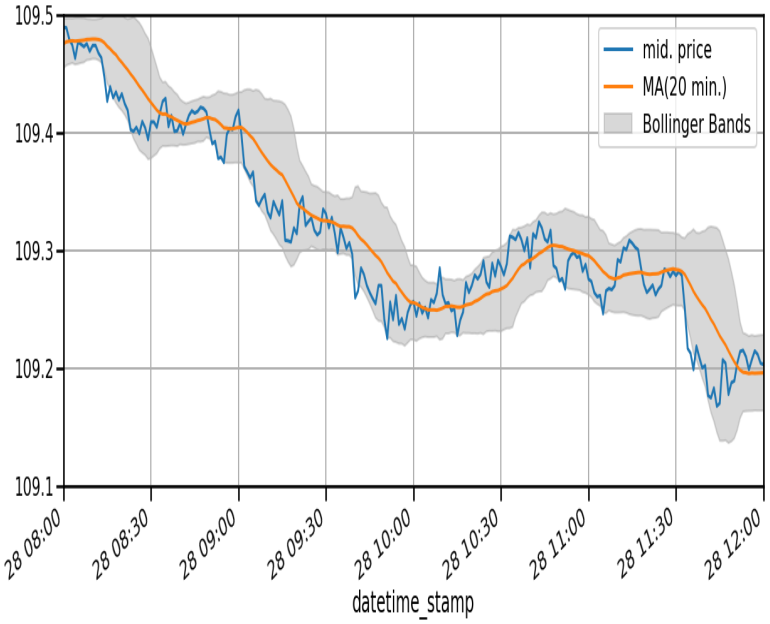
### 特徴エンジニアリング

レートデータはそのまま使えないので、下記のデータの前処理を行いました。

**BIDレートとASKレートからMIDレートを作る**




**MA、MACD、BollingerBandなどの特徴量を作る**




作った特徴量を確認しましょう。今回のMIDプライスを表現するため、21個の特徴量を使います。約２０万点の1分足のFXデータを用いて学習・検証を行います。

```
raw_forex_df.columns

Index(['datetime_stamp', 'mid', 'MA_5min', 'MA_8min', 'MA_13min',
'MA_1hour',
       'MA_1day', 'MA_5day', 'STD_5min', 'STD_8min', 'STD_13min',
'STD_1hour',
       'STD_1day', 'STD_5day', 'momentum', 'MA_20min', 'STD_20min',
       'upper_band', 'lower_band', 'ema', 'ema_12', 'ema_26', 'MACD',
       'MACD_signal'],
      dtype='object')


raw_forex_df.values.shape

(205590,21)
```

### トレーニングデータセットと検証データセットを分割する

```
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(raw_forex_df, train_size=0.9, test_siz
```

### Feature Scaling

データを確認しましょう。

```
In [18]: df_train.head(5)
```

Out[18]:

| datetime_stamp | mid | MA_5min | MA_8min | MA_13min | MA_1hour | STD_5min | STD_8min | STD_13min | STD_1hour | STD_1day | ... | momentum | MA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2019-01-01 18:00:00 | 109.681503 | 109.685402 | 109.684128 | 109.686729 | 109.668091 | 0.003362 | 0.003238 | 0.004549 | 0.015556 | 0.0 | ... | -0.0090 | 109. |
| 2019-01-01 18:01:00 | 109.678001 | 109.684196 | 109.683876 | 109.685616 | 109.668228 | 0.004764 | 0.003652 | 0.004788 | 0.015607 | 0.0 | ... | -0.0035 | 109. |
| 2019-01-01 18:02:00 | 109.666496 | 109.680199 | 109.681747 | 109.683578 | 109.668167 | 0.008927 | 0.007161 | 0.006658 | 0.015606 | 0.0 | ... | -0.0115 | 109. |
| 2019-01-01 18:03:00 | 109.657997 | 109.674896 | 109.678688 | 109.681076 | 109.667969 | 0.012774 | 0.011003 | 0.009385 | 0.015659 | 0.0 | ... | -0.0085 | 109. |
| 2019-01-01 18:04:00 | 109.658501 | 109.668503 | 109.675499 | 109.678963 | 109.667786 | 0.010880 | 0.012792 | 0.011121 | 0.015705 | 0.0 | ... | 0.0005 | 109. |

5 rows × 21 columns

データセットの特徴量によってスケールが異なるので揃える必要があります。今回はSklearnのMinMaxScalerを用いて正規化し、特徴量を[0, 1]の範囲に変換しました。

```
x_scaler = MinMaxScaler()

df_x_train_scaled = x_scaler.fit_transform(df_train)
df_x_train_scaled[0:5]

array([[0.62589499, 0.61497242, 0.6002301 , 0.58236372, 0.51455824,
        0.00253999, 0.0021764 , 0.0028962 , 0.01220264, 0.        ,
        0.        , 0.72659446, 0.56524956, 0.0055043 , 0.51399822,
        0.67436233, 0.62083891, 0.55097021, 0.51782655, 0.8427443 ,
        0.83256743],
       [0.6254129 , 0.61480089, 0.6001928 , 0.58219051, 0.51458329,
        0.00360003, 0.00245516, 0.00304832, 0.01224252, 0.        ,
        0.        , 0.72924187, 0.56531401, 0.00513634, 0.51387099,
        0.67454783, 0.62029185, 0.5508006 , 0.51780168, 0.84193072,
        0.83208187],
       [0.62382904, 0.61423204, 0.59987749, 0.58187377, 0.51457216,
        0.00674563, 0.00481407, 0.00423895, 0.01224236, 0.        ,
```

```
         0.        , 0.72539109, 0.56529046, 0.00535693, 0.51396402,
         0.67444758, 0.61903777, 0.55036191, 0.51762208, 0.84040926,
         0.83130946],
        [0.62265901, 0.61347754, 0.5994243 , 0.58148465, 0.51453598,
         0.00965206, 0.00739651, 0.00597524, 0.01228387, 0.        ,
         0.        , 0.72683513, 0.56517272, 0.00643653, 0.51441822,
         0.67395759, 0.61782847, 0.54977148, 0.517343  , 0.83857018,
         0.83022741],
        [0.62272833, 0.61256781, 0.59895189, 0.58115604, 0.51450259,
         0.00822098, 0.00859928, 0.00708083, 0.01232   , 0.        ,
         0.        , 0.73116726, 0.56501409, 0.0074225 , 0.51477803,
         0.67347134, 0.61747123, 0.54928433, 0.51709017, 0.83716339,
         0.82900676]])
```

## 目的変数

今回は、ドル円の過去のデータを学習して将来のMIDプライスを予測するので、datasetの「mid」が目的変数ですね。また、時系列データを予測する問題なので、タイムステップ（lookback window）が512でTrain, Testデータを作りました。過去の512分の観察を学習して次の1分のMIDプライスを予測するイメージです。TCNは長いスパンの過去データの特徴を覚えられるので、タイムステップを少々長めに設定しました。 データセットから説明変数、目的変数の時系列に変換します。

```python
from tqdm import tqdm_notebook
def build_timeseries(dataset, target_column_index, time_steps):
    # Total number of time-series samples would be len(dataset) - time_ste
    dim_0 = dataset.shape[0] - time_steps
    print(dim_0)
    dim_1 = dataset.shape[1]
    print(dim_1)

    # Init the x, y matrix
    x = np.zeros((dim_0, time_steps, dim_1))
    # Number of output variables is 1 for this mid price prediction
    y = np.zeros((dim_0, 1) )

    # fill data to x, y
    for i in tqdm_notebook(range(dim_0)):
        x[i] = dataset[i: i + time_steps]
        y[i] = dataset[i + time_steps, target_column_index]

    print("Lenght of time-series x, y", x.shape, y.shape)
    return x, y
```

```python
lookback_window = 512

x_train_scaled, y_train_scaled =
build_timeseries(df_x_train_scaled, target_column_index, lookback_window)

x_test_scaled, y_test_scaled  = build_timeseries(df_x_test_scaled, target_c
```

## モデル構築

TCN に関しては，深層学習フレームワーク Keras で実装されているTCN-tensorflowを TensorFlow のバックグラウンドのもと使用し、TCN-tensorflowのソースを参考してモデルを作成しました。

```python
i = Input(shape=(lookback_window, num_x_signals, ))
m = TCN(nb_stacks=1)(i)
m = Dense(num_y_signals, activation='linear')(m)
model = Model(inputs=[i], outputs=[m])
optimizer = RMSprop(lr=1e-4)
model.compile(optimizer=optimizer,
              loss='mse',
              metrics=['mae'])
model.summary()
```

```
Model: "model"

_________________________________________________________________
Layer (type)                 Output Shape              Param #     Connecte
=================================================================
input_1 (InputLayer)         [(None, 512, 21)]         0

_________________________________________________________________
conv1d (Conv1D)              (None, 512, 64)           1408        input_1[

_________________________________________________________________
conv1d_1 (Conv1D)            (None, 512, 64)           8256        conv1d[0

_________________________________________________________________
activation (Activation)      (None, 512, 64)           0           conv1d_1

_________________________________________________________________
spatial_dropout1d (SpatialDropo (None, 512, 64)        0           activati

_________________________________________________________________
conv1d_2 (Conv1D)            (None, 512, 64)           8256        spatial_

_________________________________________________________________
activation_1 (Activation)    (None, 512, 64)           0           conv1d_2

_________________________________________________________________
spatial_dropout1d_1 (SpatialDro (None, 512, 64)        0           activati

_________________________________________________________________
conv1d_3 (Conv1D)            (None, 512, 64)           4160        conv1d[0

_________________________________________________________________
add (Add)                    (None, 512, 64)           0           conv1d_3
                                                                   spatial_

_________________________________________________________________
activation_2 (Activation)    (None, 512, 64)           0           add[0][0

_________________________________________________________________
conv1d_4 (Conv1D)            (None, 512, 64)           8256        activati

_________________________________________________________________
activation_3 (Activation)    (None, 512, 64)           0           conv1d_4

_________________________________________________________________
spatial_dropout1d_2 (SpatialDro (None, 512, 64)        0           activati

_________________________________________________________________
conv1d_5 (Conv1D)            (None, 512, 64)           8256        spatial_

_________________________________________________________________
activation_4 (Activation)    (None, 512, 64)           0           conv1d_5

_________________________________________________________________
spatial_dropout1d_3 (SpatialDro (None, 512, 64)        0           activati

_________________________________________________________________
conv1d_6 (Conv1D)            (None, 512, 64)           4160        activati
```

```
__________________________________________________________________________________________________
add_1 (Add)                     (None, 512, 64)      0           conv1d_6
                                                                 spatial_
__________________________________________________________________________________________________
activation_5 (Activation)       (None, 512, 64)      0           add_1[0]
__________________________________________________________________________________________________
conv1d_7 (Conv1D)               (None, 512, 64)      8256        activati
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 512, 64)      0           conv1d_7
__________________________________________________________________________________________________
spatial_dropout1d_4 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_8 (Conv1D)               (None, 512, 64)      8256        spatial_
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 512, 64)      0           conv1d_8
__________________________________________________________________________________________________
spatial_dropout1d_5 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_9 (Conv1D)               (None, 512, 64)      4160        activati
__________________________________________________________________________________________________
add_2 (Add)                     (None, 512, 64)      0           conv1d_9
                                                                 spatial_
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 512, 64)      0           add_2[0]
__________________________________________________________________________________________________
conv1d_10 (Conv1D)              (None, 512, 64)      8256        activati
__________________________________________________________________________________________________
activation_9 (Activation)       (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_6 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_11 (Conv1D)              (None, 512, 64)      8256        spatial_
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_7 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_12 (Conv1D)              (None, 512, 64)      4160        activati
__________________________________________________________________________________________________
add_3 (Add)                     (None, 512, 64)      0           conv1d_1
                                                                 spatial_
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 512, 64)      0           add_3[0]
__________________________________________________________________________________________________
conv1d_13 (Conv1D)              (None, 512, 64)      8256        activati
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_8 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_14 (Conv1D)              (None, 512, 64)      8256        spatial_
__________________________________________________________________________________________________
activation_13 (Activation)      (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_9 (SpatialDro (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_15 (Conv1D)              (None, 512, 64)      4160        activati
__________________________________________________________________________________________________
add_4 (Add)                     (None, 512, 64)      0           conv1d_1
                                                                 spatial_
__________________________________________________________________________________________________
activation_14 (Activation)      (None, 512, 64)      0           add_4[0]
__________________________________________________________________________________________________
conv1d_16 (Conv1D)              (None, 512, 64)      8256        activati
__________________________________________________________________________________________________
activation_15 (Activation)      (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_10 (SpatialDr (None, 512, 64)      0           activati
__________________________________________________________________________________________________
conv1d_17 (Conv1D)              (None, 512, 64)      8256        spatial_
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 512, 64)      0           conv1d_1
__________________________________________________________________________________________________
spatial_dropout1d_11 (SpatialDr (None, 512, 64)      0           activati
__________________________________________________________________________________________________
add_6 (Add)                     (None, 512, 64)      0           spatial_
                                                                 spatial_
                                                                 spatial_
                                                                 spatial_
                                                                 spatial_
                                                                 spatial_
__________________________________________________________________________________________________
lambda (Lambda)                 (None, 64)           0           add_6[0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1)            65          lambda[0
==================================================================================================
Total params: 121,345
Trainable params: 121,345
Non-trainable params: 0
__________________________________________________________________________________________________
```

### コールバック関数の設定

効率的にトレーニングを行うため、KerasのCallback Functionを使いました。

**Checkpointを作成する**

トレーニングではロス関数の最小値が改善された場合、その時点でのモデルを一旦保存します。今回は、Validationの際の値で評価したいと思いますので、「val_loss」を使いました。

```
path_checkpoint = 'alpha_trader_checkpoint.keras'
callback_checkpoint = ModelCheckpoint(filepath=path_checkpoint,
                                      monitor='val_loss',
                                      verbose=1,
                                      save_weights_only=True,
                                      save_best_only=True)
```

**Early Stopping**

トレーニングを行ってもValidationのパフォーマンスが改善されない場合に学習を強制的に終了する機能です。Overfitting問題を回避する効果もあります。今回は5 epochs間で精度が改善しなければ終了するようにします。

```
callback_early_stopping = EarlyStopping(monitor='val_loss',
```

```
                                       patience=5, verbose=1)
```

**TensorBoard logに出力**

トレーニングの際にTensorboard logに出力することで、トレーニング後に色々なことを確認できるので、使いました。

```
from datetime import datetime
logdir="/tf/alpha-trader/source/logs/fit/" + datetime.today().strftime("%Y%
callback_tensorboard = TensorBoard(log_dir=logdir, histogram_freq=0, write_
```

**Learning Rateの自動調整**

トレーニングしている時にValidationのパフォーマンスが改善されない際にLearning Rateを自動的に調整するための設定です。 下記の設定では、patience=0なので今回のepochの評価にパフォーマンスが改善しなければすぐに次のepochにLearning Rateが１０倍（factor=0.1）減って、 トレーニングを行うことになります。また、Learning rateは1e-5を最小値として設定してありますので、この値よりは小さくなりません。

```
callback_reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                                       factor=0.1,
                                       min_lr=1e-5,
                                       patience=0,
                                       verbose=1)
```

# 学習

これまで作成したTCNモデルにトレーニングデータで30epochsくらい学習させましょう。念のため、Epochごとに精度の改善がなければ学習を終了するように設定しました。

```
%%time
model.fit(x_train_scaled,
          y_train_scaled,
          epochs=30,
          validation_split = 0.1,
          shuffle=True,
          callbacks=callbacks)
```

```
Train on 166067 samples, validate on 18452 samples
Epoch 1/30
165952/166067 [============================>.] - ETA: 0s - loss: 0.0035
- mean_absolute_error: 0.0430
Epoch 00001: val_loss improved from inf to 0.00035, saving model to
alpha_trader_checkpoint.keras
166067/166067 [==============================] - 80s 481us/sample -
loss: 0.0035 - mean_absolute_error: 0.0430 - val_loss: 3.4900e-04 -
val_mean_absolute_error: 0.0137
Epoch 2/30
166048/166067 [============================>.] - ETA: 0s - loss:
4.4827e-04 - mean_absolute_error: 0.0166
Epoch 00002: val_loss improved from 0.00035 to 0.00032, saving model to
alpha_trader_checkpoint.keras

Epoch 00002: ReduceLROnPlateau reducing learning rate to 1e-05.
166067/166067 [==============================] - 77s 465us/sample -
loss: 4.4827e-04 - mean_absolute_error: 0.0166 - val_loss: 3.1790e-04 -
val_mean_absolute_error: 0.0169
Epoch 3/30
166048/166067 [============================>.] - ETA: 0s - loss:
2.9954e-05 - mean_absolute_error: 0.0026
Epoch 00003: val_loss improved from 0.00032 to 0.00013, saving model to
alpha_trader_checkpoint.keras
166067/166067 [==============================] - 82s 495us/sample -
loss: 2.9951e-05 - mean_absolute_error: 0.0026 - val_loss: 1.2694e-04 -
val_mean_absolute_error: 0.0079
Epoch 4/30
...
```

```
Epoch 27/30
166016/166067 [============================>.] - ETA: 0s - loss:
5.5239e-06 - mean_absolute_error: 0.0017
Epoch 00027: val_loss did not improve from 0.00005
166067/166067 [==============================] - 78s 467us/sample -
loss: 5.5232e-06 - mean_absolute_error: 0.0017 - val_loss: 5.3255e-05 -
val_mean_absolute_error: 0.0050
Epoch 28/30
165984/166067 [============================>.] - ETA: 0s - loss:
5.5814e-06 - mean_absolute_error: 0.0017
Epoch 00028: val_loss did not improve from 0.00005
166067/166067 [==============================] - 76s 459us/sample -
loss: 5.5801e-06 - mean_absolute_error: 0.0017 - val_loss: 5.1092e-05 -
val_mean_absolute_error: 0.0050
Epoch 00028: early stopping
CPU times: user 39min 16s, sys: 1min 59s, total: 41min 15s
Wall time: 36min 6s
```

23 epochまで学習しましたね。それ以上、精度が改善されなかったので28 epoch目に学習が強制終了されました。良さそうです。

# 評価

最後に保存したモデルがベストモデルなので、それをロードして評価を行いましょう。

```
try:
    model.load_weights(path_checkpoint)
except Exception as error:
```

```
    print("Error trying to load checkpoint.")
    print(error)
```

```
result = model.evaluate(x_test_scaled, y_test_scaled)

20047/20047 [==============================] - 6s 298us/sample - loss: 3.02
```

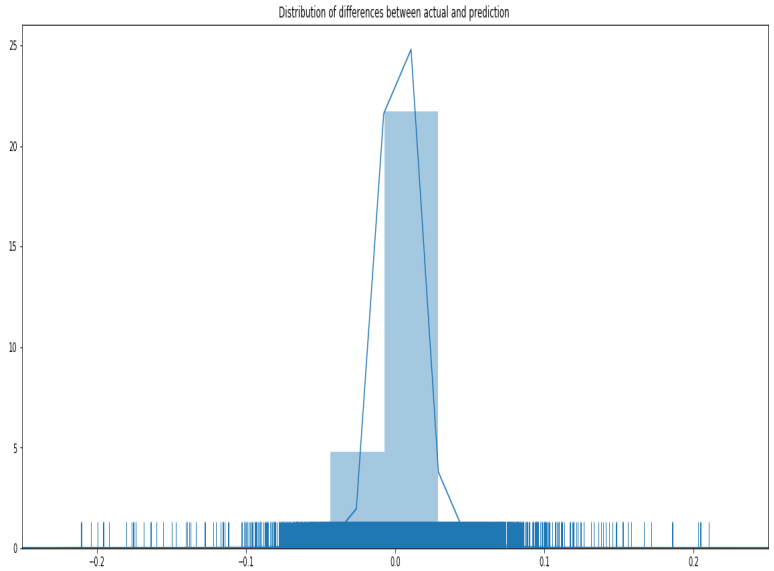
### 評価関数を実装

評価するため、実際のMidプライスと予測プライスの乖離はどのくらいあるか可視化する関数を実装しておきましょう。

```
import seaborn as sns
def plot_comparison(y_train, y_test, y_scaler_min, y_scaler_max, lookback_
    """
    Plot the predicted and true output-signals.

    :param start_idx: Start-index for the time-series.
    :param length: Sequence-length to process and plot.
    :param train: Boolean whether to use training- or test-set.
    """

    if train:
        # Use training-data.
        x = x_train_scaled
        y_true = y_train
    else:
        # Use test-data.
        x = x_test_scaled
        y_true = y_test

    # End-index for the sequences.
    end_idx = start_idx + length

    # Select the sequences from the given start-index and
    # of the given length.
    x = x[start_idx:end_idx]
    y_true = y_true[lookback_window+ start_idx:lookback_window+end_idx]

    # Input-signals for the model.
   # x = np.expand_dims(x, axis=0)

    # Use the model to predict the output-signals.
    y_pred = model.predict(x)

    # The output of the model is between 0 and 1.
    # Do an inverse map to get it back to the scale
    # of the original data-set.
    #y_pred_rescaled = y_scaler.inverse_transform(y_pred[0])
    y_pred_rescaled = y_pred*(y_scaler_max - y_scaler_min) + y_scaler_min

    # For each output-signal.
    for signal in range(len(target_names)):
        # Get the output-signal predicted by the model.
        signal_pred = y_pred_rescaled[:, signal]

        # Get the true output-signal from the data-set.
        #signal_true = y_true[:, signal]
        signal_true = y_true

        # Make the plotting-canvas bigger.
        plt.figure(figsize=(20,8))

        # Plot and compare the two signals.
        plt.plot(signal_true, label='true')
        plt.plot(signal_pred, label='pred')

        # Plot grey box for warmup-period.
        p = plt.axvspan(0, warmup_steps, facecolor='black', alpha=0.15)

        # Plot labels etc.
        plt.ylabel(target_names[signal])
        plt.legend()
        plt.show()

        # Some more benchmark
        diff = signal_pred - signal_true
        fig, ax = plt.subplots(figsize=(20, 8))
        sns.distplot(diff, kde=True, rug=True)
        ax.set_xlim(-0.25, 0.25)
        plt.title('Distribution of differences between actual and predicti
        plt.show()
```

### トレーニングセットで評価してみる

```
y_train_mid_min = np.min(df_train)["mid"]
y_train_mid_max = np.max(df_train)["mid"]

plot_comparison(y_train, y_test, y_train_mid_min, y_train_mid_max, lookback
```

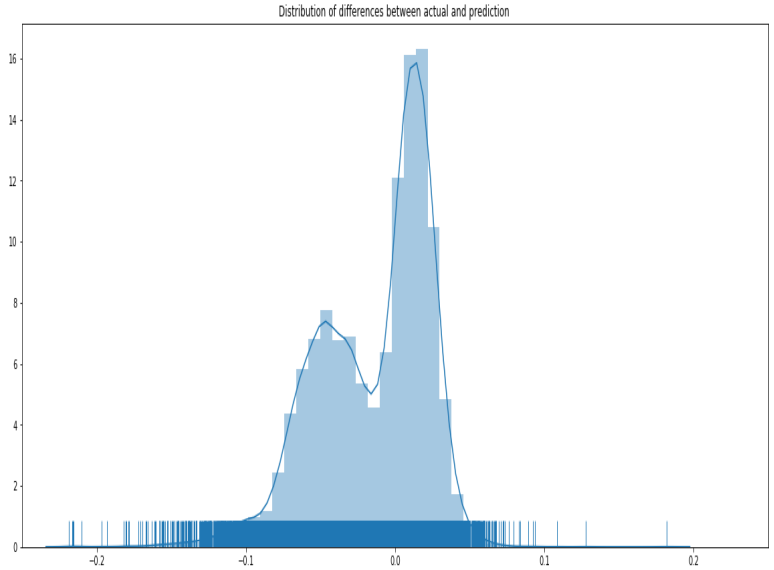




見事、トレーニングデータで評価結果は良さそうですね。Histogramから見ると予測値と実際値の乖離がありますが、基本的に0周辺に集中されていますね。

### テストデータで評価してみる

次はテストデータで予測結果はどうなるか確認しましょう。

```
1 plot_comparison(y_train, y_test, y_train_mid_min, y_train_mid_max, lookbac
```


うん、テストデータセットの予測と実際の数値との乖離はトレーニングデータセットの評価結果より大きですね。差分のHistogramは0周りに分布されていることも見えますが。とは言え、プライスのトレンドの再現性は良さそうですね。

## 所感

今回は時系列データを対応可能なTCNでFX予測をしてみました。検証結果は、プライスの予測はまだイマイチですが上下トレンドの予測は良さそうですね。今後、特徴量の見直しやHyper Parameterのチューニングをして予測精度を改善していきたいと思います。ではまた！

## 最後に

次世代システム研究室では、ビッグデータ解析プラットホームの設計・開発を行うアーキテクトとデータサイエンティストを募集しています。次世代システム研究室にご興味を持って頂ける方がいらっしゃいましたら、ぜひ 募集職種一覧 からご応募をお願いします。

皆さんのご応募をお待ちしています。