07_cnn_for_trading

September 29, 2021

1 CNN for Trading - Part 3: Training and Evaluating a CNN

To exploit the grid-like structure of time-series data, we can use CNN architectures for univariate and multivariate time series. In the latter case, we consider different time series as channels, similar to the different color signals.

An alternative approach converts a time series of alpha factors into a two-dimensional format to leverage the ability of CNNs to detect local patterns. Sezer and Ozbayoglu (2018) propose CNN-TA, which computes 15 technical indicators for different intervals and uses hierarchical clustering (see Chapter 13, Data-Driven Risk Factors and Asset Allocation with Unsupervised Learning) to locate indicators that behave similarly close to each other in a two-dimensional grid.

The authors train a CNN similar to the CIFAR-10 example we used earlier to predict whether to buy, hold, or sell an asset on a given day. They compare the CNN performance to "buy-and-hold" and other models and find that it outperforms all alternatives using daily price series for Dow 30 stocks and the nine most-traded ETFs over the 2007-2017 time period.

The section on *CNN for Trading* consists of three notebooks that experiment with this approach using daily US equity price data. They demonstrate 1. How to compute relevant financial features 2. How to convert a similar set of indicators into image format and cluster them by similarity 3. How to train a CNN to predict daily returns and evaluate a simple long-short strategy based on the resulting signals.

1.1 Creating and training a convolutional neural network

Now we are ready to design, train, and evaluate a CNN following the steps outlined in the previous section.

1.2 Imports

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
%matplotlib inline

from time import time
from pathlib import Path
import sys, os
```

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import MinMaxScaler
from scipy.stats import spearmanr

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,␣
↪MaxPooling2D

import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
if gpu_devices:
    print('Using GPU')
    tf.config.experimental.set_memory_growth(gpu_devices[0], True)
else:
    print('Using CPU')
```

```
Using CPU
```

[4]:
```
sys.path.insert(1, os.path.join(sys.path[0], '..'))
from utils import MultipleTimeSeriesCV, format_time
```

[5]:
```
sns.set_style('whitegrid')
np.random.seed(42)
```

[6]:
```
size = 15
lookahead = 1
```

[7]:
```
results_path = Path('results', 'cnn_for_trading')
if not results_path.exists():
    results_path.mkdir(parents=True)
```

1.3 Load Model Data

[8]:
```
with pd.HDFStore('data.h5') as store:
    features = store['img_data']
    targets = store['targets']
```

[9]:
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 2378728 entries, ('A', Timestamp('2001-01-02 00:00:00')) to ('ZTS',
Timestamp('2017-12-29 00:00:00'))
```

```
Columns: 225 entries, 01_CMO to 11_WMA
dtypes: float32(225)
memory usage: 2.0+ GB
```

[10]:
```
targets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 2378728 entries, ('A', Timestamp('2001-01-02 00:00:00')) to ('ZTS',
Timestamp('2017-12-29 00:00:00'))
Data columns (total 4 columns):
 #   Column      Dtype
---  ------      -----
 0   r01_fwd     float64
 1   r01dec_fwd  float64
 2   r05_fwd     float64
 3   r05dec_fwd  float64
dtypes: float64(4)
memory usage: 81.8+ MB
```

[11]:
```
outcome = f'r{lookahead:02}_fwd'
features = features.join(targets[[outcome]]).dropna()
target = features[outcome]
features = features.drop(outcome, axis=1)
```

1.4 Convolutional Neural Network

We again closely follow the authors in creating a CNN with 2 convolutional layers with kernel size 3 and 16 and 32 filters, respectively, followed by a max pooling layer of size 2.

We flatten the output of the last stack of filters and connect the resulting 1,568 outputs to a dense layer of size 32, applying 25 and 50 percent dropout probability to the incoming and outcoming connections to mitigate overfitting.

1.4.1 Model Architecture

[12]:
```
def make_model(filter1=16, act1='relu', filter2=32, act2='relu', do1=.25, do2=.
↪5, dense=32):
    input_shape = (size, size, 1)
    cnn = Sequential([
        Conv2D(filters=filter1,
               kernel_size=3,
               padding='same',
               activation=act1,
               input_shape=input_shape,
               name='CONV1'),
        Conv2D(filters=filter2,
               kernel_size=3,
               padding='same',
```

```
                activation=act2,
                name='CONV2'),
        MaxPooling2D(pool_size=2, name='POOL2'),
        Dropout(do1, name='DROP1'),
        Flatten(name='FLAT1'),
        Dense(dense, activation='relu', name='FC1'),
        Dropout(do2, name='DROP2'),
        Dense(1, activation='linear', name='FC2')
    ])
    cnn.compile(loss='mse',
                optimizer=tf.keras.optimizers.SGD(learning_rate=0.01,
                                                  momentum=0.9,
                                                  nesterov=False,
                                                  name='SGD'),
                metrics=[tf.keras.metrics.RootMeanSquaredError(name='rmse')])
    return cnn
```

[13]:
```
cnn = make_model()
cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
CONV1 (Conv2D)               (None, 15, 15, 16)        160
_________________________________________________________________
CONV2 (Conv2D)               (None, 15, 15, 32)        4640
_________________________________________________________________
POOL2 (MaxPooling2D)         (None, 7, 7, 32)          0
_________________________________________________________________
DROP1 (Dropout)              (None, 7, 7, 32)          0
_________________________________________________________________
FLAT1 (Flatten)              (None, 1568)              0
_________________________________________________________________
FC1 (Dense)                  (None, 32)                50208
_________________________________________________________________
DROP2 (Dropout)              (None, 32)                0
_________________________________________________________________
FC2 (Dense)                  (None, 1)                 33
=================================================================
Total params: 55,041
Trainable params: 55,041
Non-trainable params: 0
_________________________________________________________________
```

1.4.2 Train the Model

We cross-validate the model with the MutipleTimeSeriesCV train and validation set index generator introduced in Chapter 7, Linear Models – From Risk Factors to Return Forecasts. We provide 5 years of trading days during the training period in batches of 64 random samples and validate using the subsequent 3 months, covering the years 2014-2017.

[14]:
```
train_period_length = 5 * 12 * 21
test_period_length = 5 * 21
n_splits = 16
```

[15]:
```
cv = MultipleTimeSeriesCV(n_splits=n_splits,
                          train_period_length=train_period_length,
                          test_period_length=test_period_length,
                          lookahead=lookahead)
```

We scale the features to the range [-1, 1] and again use NumPy's .reshape() method to create the requisite format:

[16]:
```
def get_train_valid_data(X, y, train_idx, test_idx):
    x_train, y_train = X.iloc[train_idx, :], y.iloc[train_idx]
    x_val, y_val = X.iloc[test_idx, :], y.iloc[test_idx]
    scaler = MinMaxScaler(feature_range=(-1, 1))
    x_train = scaler.fit_transform(x_train)
    x_val = scaler.transform(x_val)
    return (x_train.reshape(-1, size, size, 1), y_train,
            x_val.reshape(-1, size, size, 1), y_val)
```

[17]:
```
batch_size = 64
```

[18]:
```
checkpoint_path = results_path / f'lookahead_{lookahead:02d}'
if not checkpoint_path.exists():
    checkpoint_path.mkdir(parents=True, exist_ok=True)
```

Training and validation follow the process laid out in Chapter 17, Deep Learning for Trading, relying on checkpointing to store weights after each epoch and generate predictions for the best-performing iterations without the need for costly retraining.

[20]:
```
start = time()
ic  = []
for fold, (train_idx, test_idx) in enumerate(cv.split(features)):
    X_train, y_train, X_val, y_val = get_train_valid_data(features, target,␣
 ↪train_idx, test_idx)
    preds = y_val.to_frame('actual')
    r = pd.DataFrame(index=y_val.index.unique(level='date')).sort_index()
    model = make_model(filter1=16, act1='relu', filter2=32,
                       act2='relu', do1=.25, do2=.5, dense=32)
    best_mean = best_median = -np.inf
```

```
    for epoch in range(25):
        model.fit(X_train, y_train,
                  batch_size=batch_size,
                  validation_data=(X_val, y_val),
                  epochs=epoch + 1,
                  initial_epoch=epoch,
                  verbose=0, shuffle=True)
        model.save_weights((checkpoint_path / f'ckpt_{fold}_{epoch}').
↪as_posix())
        preds[epoch] = model.predict(X_val).squeeze()
        r[epoch] = preds.groupby(level='date').apply(lambda x: spearmanr(x.
↪actual, x[epoch])[0]).to_frame(epoch)
        print(f'{format_time(time()-start)} | {fold + 1:02d} | {epoch + 1:02d}␣
↪| {r[epoch].mean():7.4f} | {r[epoch].median():7.4f}')
    ic.append(r.assign(fold=fold))
ic = pd.concat(ic)
ic.to_csv(checkpoint_path / 'ic.csv')
```

```
00:00:14 | 01 | 01 | -0.0043 | -0.0043
00:00:25 | 01 | 02 | -0.0070 | -0.0062
00:00:36 | 01 | 03 | -0.0076 | -0.0054
00:00:46 | 01 | 04 | -0.0072 |  0.0021
00:00:57 | 01 | 05 | -0.0074 | -0.0014
00:01:08 | 01 | 06 | -0.0072 | -0.0007
00:01:18 | 01 | 07 | -0.0059 |  0.0010
00:01:29 | 01 | 08 | -0.0054 | -0.0031
00:01:40 | 01 | 09 | -0.0085 | -0.0134
00:01:52 | 01 | 10 | -0.0083 | -0.0102
00:02:04 | 01 | 11 | -0.0075 | -0.0072
00:02:15 | 01 | 12 | -0.0064 | -0.0122
00:02:25 | 01 | 13 | -0.0071 | -0.0082
00:02:37 | 01 | 14 | -0.0069 | -0.0126
00:02:48 | 01 | 15 | -0.0065 | -0.0133
00:02:59 | 01 | 16 | -0.0070 | -0.0129
00:03:10 | 01 | 17 | -0.0071 | -0.0131
00:03:21 | 01 | 18 | -0.0065 | -0.0138
00:03:32 | 01 | 19 | -0.0065 | -0.0093
00:03:44 | 01 | 20 | -0.0063 | -0.0072
00:03:55 | 01 | 21 | -0.0069 | -0.0086
00:04:06 | 01 | 22 | -0.0065 | -0.0057
00:04:17 | 01 | 23 | -0.0061 | -0.0073
00:04:29 | 01 | 24 | -0.0060 | -0.0063
00:04:41 | 01 | 25 | -0.0061 | -0.0070
00:04:53 | 02 | 01 | -0.0069 | -0.0002
00:05:05 | 02 | 02 | -0.0066 | -0.0030
00:05:16 | 02 | 03 | -0.0066 | -0.0034
00:05:27 | 02 | 04 | -0.0065 |  0.0001
```

00:05:38 | 02 | 05 | -0.0069 | -0.0007
00:05:49 | 02 | 06 | -0.0069 | -0.0027
00:05:60 | 02 | 07 | -0.0069 | -0.0029
00:06:11 | 02 | 08 | -0.0064 | -0.0012
00:06:22 | 02 | 09 | -0.0062 | -0.0035
00:06:33 | 02 | 10 | -0.0056 | -0.0034
00:06:44 | 02 | 11 | -0.0056 | -0.0001
00:06:56 | 02 | 12 | -0.0055 | -0.0050
00:07:07 | 02 | 13 | -0.0052 | -0.0052
00:07:18 | 02 | 14 | -0.0053 | -0.0042
00:07:29 | 02 | 15 | -0.0054 | -0.0075
00:07:40 | 02 | 16 | -0.0051 | -0.0055
00:07:51 | 02 | 17 | -0.0054 | -0.0053
00:08:03 | 02 | 18 | -0.0056 | -0.0063
00:08:14 | 02 | 19 | -0.0056 | -0.0049
00:08:25 | 02 | 20 | -0.0055 | -0.0072
00:08:37 | 02 | 21 | -0.0054 | -0.0057
00:08:49 | 02 | 22 | -0.0056 | -0.0030
00:09:00 | 02 | 23 | -0.0058 | -0.0063
00:09:11 | 02 | 24 | -0.0057 | -0.0055
00:09:23 | 02 | 25 | -0.0053 | -0.0051
00:09:34 | 03 | 01 | 0.0018 | -0.0184
00:09:44 | 03 | 02 | 0.0060 | -0.0087
00:09:54 | 03 | 03 | 0.0062 | -0.0103
00:10:04 | 03 | 04 | 0.0068 | -0.0102
00:10:13 | 03 | 05 | 0.0061 | -0.0086
00:10:22 | 03 | 06 | 0.0032 | -0.0098
00:10:31 | 03 | 07 | 0.0060 | -0.0054
00:10:41 | 03 | 08 | 0.0069 | -0.0026
00:10:50 | 03 | 09 | 0.0074 | -0.0013
00:10:59 | 03 | 10 | 0.0078 | -0.0006
00:11:09 | 03 | 11 | 0.0047 | 0.0008
00:11:18 | 03 | 12 | 0.0084 | 0.0004
00:11:27 | 03 | 13 | 0.0069 | 0.0084
00:11:36 | 03 | 14 | 0.0092 | 0.0033
00:11:46 | 03 | 15 | 0.0095 | 0.0039
00:11:55 | 03 | 16 | 0.0103 | 0.0070
00:12:04 | 03 | 17 | 0.0101 | 0.0032
00:12:13 | 03 | 18 | 0.0107 | 0.0099
00:12:22 | 03 | 19 | 0.0106 | 0.0066
00:12:31 | 03 | 20 | 0.0114 | 0.0090
00:12:41 | 03 | 21 | 0.0118 | 0.0118
00:12:50 | 03 | 22 | 0.0119 | 0.0164
00:12:60 | 03 | 23 | 0.0124 | 0.0150
00:13:10 | 03 | 24 | 0.0130 | 0.0175
00:13:19 | 03 | 25 | 0.0136 | 0.0217
00:13:31 | 04 | 01 | 0.0303 | 0.0438
00:13:43 | 04 | 02 | 0.0288 | 0.0403

00:13:55 | 04 | 03 | 0.0303 | 0.0436
00:14:07 | 04 | 04 | 0.0316 | 0.0452
00:14:19 | 04 | 05 | 0.0314 | 0.0448
00:14:33 | 04 | 06 | 0.0314 | 0.0447
00:14:46 | 04 | 07 | 0.0307 | 0.0407
00:15:00 | 04 | 08 | 0.0313 | 0.0412
00:15:14 | 04 | 09 | 0.0308 | 0.0446
00:15:27 | 04 | 10 | 0.0300 | 0.0414
00:15:40 | 04 | 11 | 0.0295 | 0.0295
00:15:54 | 04 | 12 | -0.0300 | -0.0303
00:16:07 | 04 | 13 | 0.0305 | 0.0342
00:16:19 | 04 | 14 | 0.0306 | 0.0365
00:16:31 | 04 | 15 | 0.0304 | 0.0315
00:16:43 | 04 | 16 | 0.0307 | 0.0295
00:16:54 | 04 | 17 | 0.0306 | 0.0294
00:17:06 | 04 | 18 | -0.0306 | -0.0319
00:17:17 | 04 | 19 | 0.0305 | 0.0315
00:17:28 | 04 | 20 | 0.0305 | 0.0313
00:17:40 | 04 | 21 | 0.0303 | 0.0342
00:17:51 | 04 | 22 | 0.0305 | 0.0339
00:18:02 | 04 | 23 | 0.0302 | 0.0352
00:18:13 | 04 | 24 | 0.0306 | 0.0347
00:18:24 | 04 | 25 | 0.0310 | 0.0354
00:18:37 | 05 | 01 | 0.0003 | -0.0049
00:18:48 | 05 | 02 | -0.0091 | -0.0068
00:18:59 | 05 | 03 | 0.0122 | -0.0021
00:19:11 | 05 | 04 | 0.0052 | 0.0275
00:19:22 | 05 | 05 | -0.0022 | 0.0143
00:19:33 | 05 | 06 | -0.0099 | 0.0068
00:19:45 | 05 | 07 | 0.0100 | 0.0191
00:19:56 | 05 | 08 | 0.0036 | 0.0165
00:20:08 | 05 | 09 | -0.0092 | -0.0188
00:20:20 | 05 | 10 | 0.0085 | -0.0064
00:20:32 | 05 | 11 | -0.0063 | 0.0062
00:20:43 | 05 | 12 | -0.0120 | -0.0126
00:20:55 | 05 | 13 | -0.0014 | -0.0147
00:21:07 | 05 | 14 | 0.0040 | 0.0108
00:21:18 | 05 | 15 | 0.0022 | 0.0098
00:21:29 | 05 | 16 | 0.0089 | 0.0171
00:21:42 | 05 | 17 | 0.0037 | 0.0096
00:21:55 | 05 | 18 | -0.0044 | -0.0038
00:22:07 | 05 | 19 | 0.0116 | 0.0128
00:22:18 | 05 | 20 | 0.0092 | 0.0219
00:22:30 | 05 | 21 | -0.0114 | -0.0179
00:22:41 | 05 | 22 | 0.0043 | 0.0006
00:22:53 | 05 | 23 | 0.0099 | 0.0035
00:23:04 | 05 | 24 | 0.0098 | 0.0136
00:23:17 | 05 | 25 | 0.0048 | 0.0004

00:23:30 | 06 | 01 | -0.0098 | -0.0099
00:23:41 | 06 | 02 | -0.0109 | -0.0119
00:23:54 | 06 | 03 | -0.0111 | -0.0121
00:24:07 | 06 | 04 | -0.0126 | -0.0147
00:24:20 | 06 | 05 | -0.0096 | -0.0058
00:24:33 | 06 | 06 | -0.0104 | -0.0062
00:24:46 | 06 | 07 | -0.0118 | -0.0045
00:24:60 | 06 | 08 | -0.0116 | -0.0033
00:25:13 | 06 | 09 | -0.0116 | -0.0063
00:25:24 | 06 | 10 | -0.0118 | -0.0041
00:25:36 | 06 | 11 | -0.0114 | -0.0024
00:25:47 | 06 | 12 | -0.0103 | -0.0063
00:26:09 | 06 | 13 | -0.0094 | -0.0023
00:26:30 | 06 | 14 | -0.0118 | -0.0021
00:26:49 | 06 | 15 | -0.0104 | 0.0004
00:27:01 | 06 | 16 | -0.0122 | -0.0041
00:27:16 | 06 | 17 | -0.0114 | -0.0058
00:27:28 | 06 | 18 | -0.0114 | -0.0052
00:27:39 | 06 | 19 | -0.0110 | -0.0064
00:27:51 | 06 | 20 | -0.0129 | -0.0029
00:28:02 | 06 | 21 | -0.0128 | 0.0010
00:28:14 | 06 | 22 | -0.0121 | -0.0019
00:28:25 | 06 | 23 | -0.0126 | -0.0006
00:28:37 | 06 | 24 | -0.0119 | -0.0003
00:28:49 | 06 | 25 | -0.0125 | -0.0021
00:29:01 | 07 | 01 | -0.0007 | -0.0097
00:29:14 | 07 | 02 | -0.0037 | -0.0184
00:29:26 | 07 | 03 | -0.0009 | -0.0184
00:29:38 | 07 | 04 | 0.0081 | 0.0153
00:29:50 | 07 | 05 | 0.0259 | 0.0290
00:30:02 | 07 | 06 | 0.0402 | 0.0691
00:30:14 | 07 | 07 | 0.0754 | 0.0754
00:30:26 | 07 | 08 | 0.0754 | 0.0754
00:30:37 | 07 | 09 | 0.0754 | 0.0754
00:30:49 | 07 | 10 | 0.0754 | 0.0754
00:31:02 | 07 | 11 | 0.0754 | 0.0754
00:31:13 | 07 | 12 | 0.0754 | 0.0754
00:31:25 | 07 | 13 | nan | nan
00:31:37 | 07 | 14 | nan | nan
00:31:49 | 07 | 15 | nan | nan
00:32:01 | 07 | 16 | 0.0754 | 0.0754
00:32:12 | 07 | 17 | nan | nan
00:32:24 | 07 | 18 | nan | nan
00:32:35 | 07 | 19 | nan | nan
00:32:47 | 07 | 20 | nan | nan
00:32:59 | 07 | 21 | nan | nan
00:33:11 | 07 | 22 | nan | nan
00:33:24 | 07 | 23 | nan | nan

00:33:38	07	24	nan	nan
00:33:52	07	25	nan	nan
00:34:09	08	01	-0.0349	-0.0525
00:34:26	08	02	-0.0350	-0.0514
00:34:40	08	03	-0.0347	-0.0494
00:34:54	08	04	-0.0347	-0.0561
00:35:07	08	05	-0.0351	-0.0516
00:35:22	08	06	-0.0348	-0.0522
00:35:36	08	07	-0.0351	-0.0510
00:35:49	08	08	-0.0349	-0.0513
00:36:01	08	09	-0.0354	-0.0552
00:36:13	08	10	-0.0348	-0.0529
00:36:26	08	11	-0.0349	-0.0556
00:36:38	08	12	-0.0346	-0.0494
00:36:50	08	13	-0.0339	-0.0463
00:37:02	08	14	-0.0346	-0.0514
00:37:14	08	15	-0.0340	-0.0452
00:37:27	08	16	-0.0339	-0.0460
00:37:39	08	17	-0.0339	-0.0422
00:37:51	08	18	-0.0343	-0.0462
00:38:02	08	19	-0.0342	-0.0548
00:38:13	08	20	-0.0338	-0.0490
00:38:25	08	21	-0.0339	-0.0487
00:38:37	08	22	-0.0337	-0.0547
00:38:49	08	23	-0.0335	-0.0535
00:39:01	08	24	-0.0339	-0.0544
00:39:13	08	25	-0.0347	-0.0576
00:39:26	09	01	-0.0112	-0.0262
00:39:38	09	02	0.0122	0.0164
00:39:50	09	03	0.0134	0.0204
00:40:02	09	04	0.0140	0.0207
00:40:14	09	05	0.0150	0.0222
00:40:25	09	06	0.0167	0.0174
00:40:37	09	07	0.0173	0.0157
00:40:49	09	08	0.0177	0.0181
00:41:01	09	09	0.0182	0.0217
00:41:15	09	10	0.0184	0.0201
00:41:34	09	11	0.0189	0.0253
00:41:48	09	12	0.0194	0.0221
00:42:02	09	13	0.0199	0.0196
00:42:15	09	14	0.0199	0.0159
00:42:28	09	15	0.0205	0.0176
00:42:49	09	16	0.0210	0.0129
00:43:14	09	17	0.0211	0.0173
00:43:37	09	18	0.0214	0.0221
00:44:00	09	19	0.0216	0.0198
00:44:23	09	20	0.0222	0.0171
00:44:46	09	21	0.0224	0.0186

00:45:09 | 09 | 22 | 0.0227 | 0.0193
00:45:30 | 09 | 23 | 0.0230 | 0.0169
00:45:43 | 09 | 24 | 0.0233 | 0.0157
00:45:54 | 09 | 25 | 0.0235 | 0.0159
00:46:07 | 10 | 01 | -0.0063 | -0.0120
00:46:19 | 10 | 02 | -0.0063 | -0.0114
00:46:31 | 10 | 03 | -0.0065 | -0.0117
00:46:43 | 10 | 04 | -0.0068 | -0.0108
00:46:55 | 10 | 05 | 0.0063 | 0.0098
00:47:07 | 10 | 06 | -0.0072 | -0.0124
00:47:20 | 10 | 07 | -0.0070 | -0.0095
00:47:32 | 10 | 08 | -0.0065 | -0.0087
00:47:44 | 10 | 09 | 0.0060 | 0.0101
00:47:55 | 10 | 10 | -0.0061 | -0.0026
00:48:07 | 10 | 11 | 0.0066 | 0.0059
00:48:19 | 10 | 12 | -0.0039 | 0.0009
00:48:30 | 10 | 13 | -0.0024 | -0.0016
00:48:42 | 10 | 14 | -0.0030 | -0.0084
00:48:54 | 10 | 15 | 0.0046 | 0.0074
00:49:07 | 10 | 16 | -0.0074 | -0.0074
00:49:20 | 10 | 17 | 0.0043 | 0.0062
00:49:31 | 10 | 18 | -0.0015 | -0.0019
00:49:43 | 10 | 19 | -0.0028 | -0.0028
00:49:56 | 10 | 20 | -0.0026 | -0.0090
00:50:08 | 10 | 21 | 0.0084 | 0.0120
00:50:21 | 10 | 22 | 0.0092 | 0.0156
00:50:32 | 10 | 23 | 0.0087 | 0.0166
00:50:44 | 10 | 24 | 0.0076 | 0.0155
00:50:56 | 10 | 25 | 0.0022 | 0.0031
00:51:09 | 11 | 01 | 0.0008 | 0.0057
00:51:21 | 11 | 02 | 0.0077 | 0.0021
00:51:32 | 11 | 03 | 0.0039 | -0.0012
00:51:44 | 11 | 04 | 0.0044 | -0.0006
00:51:56 | 11 | 05 | 0.0035 | 0.0064
00:52:07 | 11 | 06 | 0.0010 | -0.0047
00:52:19 | 11 | 07 | 0.0012 | 0.0007
00:52:30 | 11 | 08 | -0.0010 | 0.0007
00:52:42 | 11 | 09 | 0.0021 | -0.0008
00:52:54 | 11 | 10 | 0.0039 | 0.0039
00:53:05 | 11 | 11 | 0.0043 | -0.0010
00:53:17 | 11 | 12 | 0.0044 | 0.0110
00:53:30 | 11 | 13 | 0.0030 | -0.0023
00:53:42 | 11 | 14 | -0.0014 | -0.0033
00:53:54 | 11 | 15 | 0.0041 | 0.0161
00:54:06 | 11 | 16 | 0.0046 | 0.0082
00:54:18 | 11 | 17 | 0.0054 | 0.0080
00:54:29 | 11 | 18 | 0.0059 | 0.0095
00:54:41 | 11 | 19 | 0.0081 | -0.0128

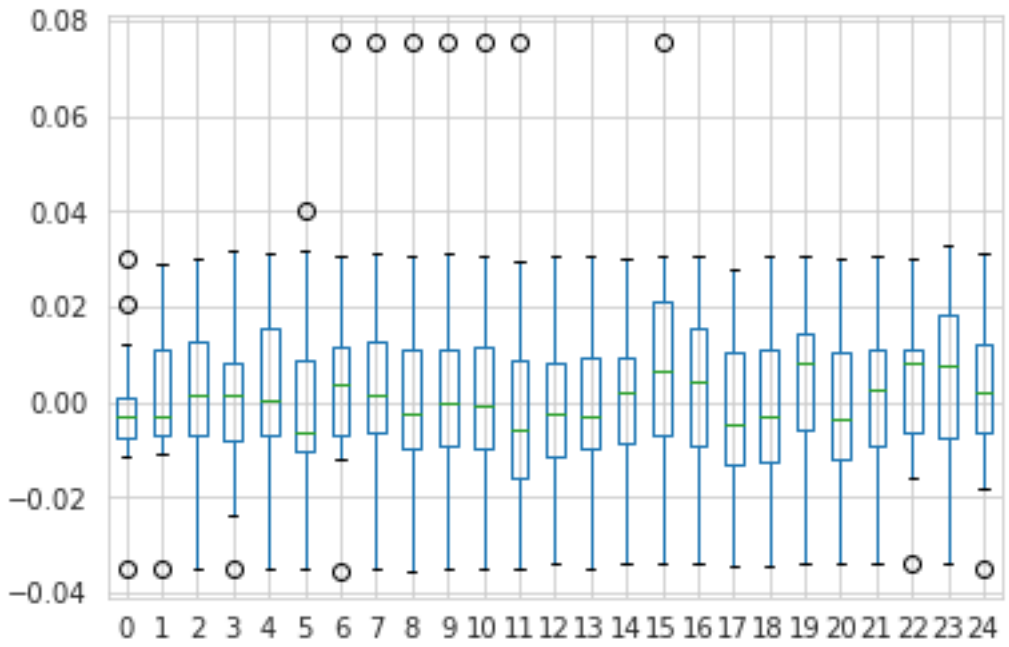
00:54:54 | 11 | 20 | 0.0081 | -0.0165
00:55:06 | 11 | 21 | 0.0091 | -0.0149
00:55:17 | 11 | 22 | 0.0026 | 0.0062
00:55:29 | 11 | 23 | 0.0057 | 0.0104
00:55:41 | 11 | 24 | 0.0031 | 0.0052
00:55:53 | 11 | 25 | 0.0059 | 0.0035
00:56:06 | 12 | 01 | -0.0101 | -0.0087
00:56:17 | 12 | 02 | -0.0034 | 0.0075
00:56:29 | 12 | 03 | -0.0074 | -0.0076
00:56:42 | 12 | 04 | -0.0106 | -0.0079
00:56:54 | 12 | 05 | -0.0044 | 0.0005
00:57:06 | 12 | 06 | -0.0116 | -0.0064
00:57:18 | 12 | 07 | 0.0076 | 0.0141
00:57:29 | 12 | 08 | -0.0029 | -0.0027
00:57:41 | 12 | 09 | -0.0062 | -0.0045
00:57:53 | 12 | 10 | -0.0038 | 0.0039
00:58:06 | 12 | 11 | -0.0088 | -0.0081
00:58:18 | 12 | 12 | -0.0212 | -0.0237
00:58:30 | 12 | 13 | -0.0131 | -0.0123
00:58:41 | 12 | 14 | -0.0072 | 0.0024
00:58:53 | 12 | 15 | -0.0025 | 0.0066
00:59:06 | 12 | 16 | -0.0048 | -0.0009
00:59:17 | 12 | 17 | -0.0154 | -0.0156
00:59:29 | 12 | 18 | -0.0143 | -0.0158
00:59:41 | 12 | 19 | -0.0145 | -0.0113
00:59:52 | 12 | 20 | -0.0048 | -0.0047
01:00:04 | 12 | 21 | -0.0037 | 0.0139
01:00:16 | 12 | 22 | 0.0020 | 0.0093
01:00:28 | 12 | 23 | -0.0064 | -0.0063
01:00:40 | 12 | 24 | -0.0090 | -0.0010
01:00:52 | 12 | 25 | -0.0064 | -0.0012
01:01:06 | 13 | 01 | -0.0014 | -0.0098
01:01:18 | 13 | 02 | 0.0107 | 0.0130
01:01:30 | 13 | 03 | 0.0078 | -0.0015
01:01:41 | 13 | 04 | 0.0091 | 0.0055
01:01:54 | 13 | 05 | 0.0171 | 0.0077
01:02:05 | 13 | 06 | 0.0061 | -0.0035
01:02:17 | 13 | 07 | 0.0078 | 0.0050
01:02:29 | 13 | 08 | 0.0083 | 0.0048
01:02:41 | 13 | 09 | 0.0090 | 0.0076
01:02:53 | 13 | 10 | 0.0086 | 0.0032
01:03:05 | 13 | 11 | 0.0092 | 0.0036
01:03:17 | 13 | 12 | 0.0094 | 0.0050
01:03:29 | 13 | 13 | 0.0095 | 0.0062
01:03:41 | 13 | 14 | 0.0093 | 0.0054
01:03:53 | 13 | 15 | 0.0099 | 0.0078
01:04:05 | 13 | 16 | 0.0103 | 0.0079
01:04:18 | 13 | 17 | 0.0112 | 0.0056

01:04:30 | 13 | 18 | 0.0099 | 0.0051
01:04:42 | 13 | 19 | 0.0109 | 0.0087
01:04:54 | 13 | 20 | 0.0099 | 0.0050
01:05:07 | 13 | 21 | 0.0096 | 0.0064
01:05:19 | 13 | 22 | 0.0098 | 0.0037
01:05:31 | 13 | 23 | 0.0103 | 0.0007
01:05:43 | 13 | 24 | 0.0106 | 0.0007
01:05:55 | 13 | 25 | 0.0103 | 0.0007
01:06:09 | 14 | 01 | -0.0067 | -0.0066
01:06:21 | 14 | 02 | -0.0025 | -0.0001
01:06:33 | 14 | 03 | -0.0037 | -0.0062
01:06:44 | 14 | 04 | -0.0012 | -0.0066
01:06:56 | 14 | 05 | -0.0041 | -0.0142
01:07:08 | 14 | 06 | -0.0052 | -0.0069
01:07:20 | 14 | 07 | -0.0080 | -0.0135
01:07:32 | 14 | 08 | -0.0066 | -0.0152
01:07:44 | 14 | 09 | -0.0107 | -0.0082
01:07:56 | 14 | 10 | -0.0119 | -0.0146
01:08:08 | 14 | 11 | -0.0183 | -0.0354
01:08:20 | 14 | 12 | -0.0136 | -0.0304
01:08:32 | 14 | 13 | -0.0153 | -0.0240
01:08:45 | 14 | 14 | -0.0152 | -0.0256
01:09:01 | 14 | 15 | -0.0188 | -0.0322
01:09:17 | 14 | 16 | -0.0161 | -0.0247
01:09:33 | 14 | 17 | -0.0144 | -0.0258
01:09:48 | 14 | 18 | -0.0185 | -0.0331
01:10:04 | 14 | 19 | -0.0143 | -0.0199
01:10:20 | 14 | 20 | -0.0163 | -0.0260
01:10:35 | 14 | 21 | -0.0142 | -0.0245
01:10:51 | 14 | 22 | -0.0144 | -0.0236
01:11:07 | 14 | 23 | -0.0157 | -0.0332
01:11:23 | 14 | 24 | -0.0134 | -0.0165
01:11:39 | 14 | 25 | -0.0183 | -0.0407
01:11:58 | 15 | 01 | 0.0120 | 0.0005
01:12:15 | 15 | 02 | 0.0278 | 0.0189
01:12:31 | 15 | 03 | 0.0256 | 0.0315
01:12:48 | 15 | 04 | 0.0313 | 0.0375
01:13:04 | 15 | 05 | 0.0299 | 0.0229
01:13:21 | 15 | 06 | 0.0321 | 0.0206
01:13:37 | 15 | 07 | 0.0286 | 0.0248
01:13:53 | 15 | 08 | 0.0298 | 0.0238
01:14:09 | 15 | 09 | 0.0290 | 0.0290
01:14:24 | 15 | 10 | 0.0311 | 0.0312
01:14:40 | 15 | 11 | 0.0308 | 0.0204
01:14:57 | 15 | 12 | 0.0295 | 0.0345
01:15:14 | 15 | 13 | 0.0283 | 0.0361
01:15:32 | 15 | 14 | 0.0302 | 0.0255
01:15:48 | 15 | 15 | 0.0282 | 0.0328

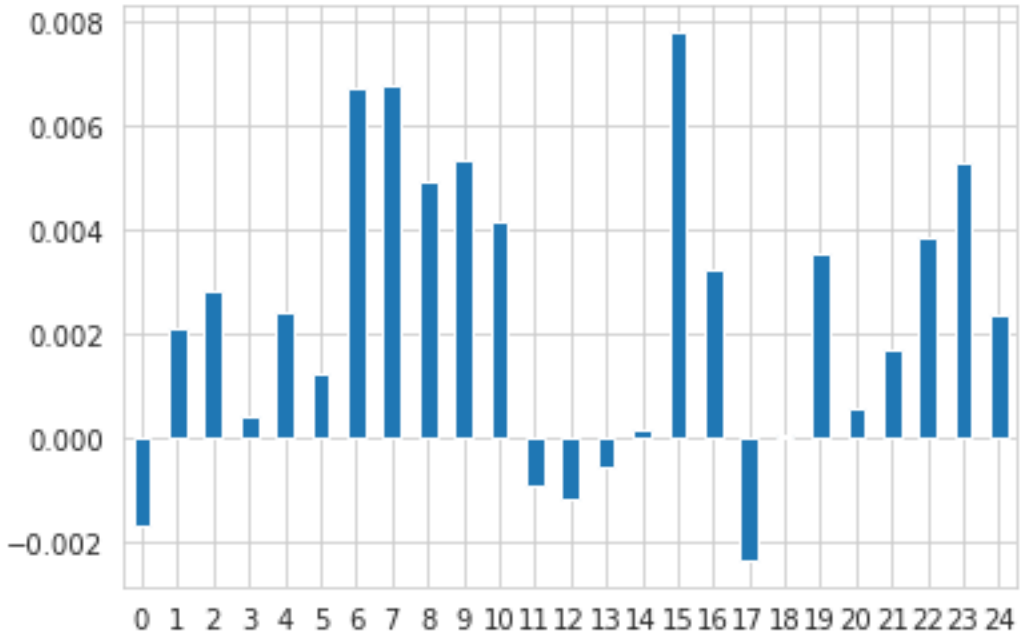
```
01:16:05 | 15 | 16 |  0.0286 |  0.0266
01:16:23 | 15 | 17 |  0.0299 |  0.0259
01:16:40 | 15 | 18 |  0.0277 |  0.0370
01:16:59 | 15 | 19 |  0.0290 |  0.0270
01:17:15 | 15 | 20 |  0.0273 |  0.0421
01:17:31 | 15 | 21 |  0.0296 |  0.0257
01:17:49 | 15 | 22 |  0.0310 |  0.0274
01:18:06 | 15 | 23 |  0.0295 |  0.0175
01:18:20 | 15 | 24 |  0.0282 |  0.0331
01:18:32 | 15 | 25 |  0.0285 |  0.0402
01:18:45 | 16 | 01 |  0.0204 |  0.0199
01:18:57 | 16 | 02 |  0.0253 |  0.0321
01:19:09 | 16 | 03 |  0.0244 |  0.0282
01:19:21 | 16 | 04 | -0.0238 | -0.0368
01:19:34 | 16 | 05 | -0.0269 | -0.0416
01:19:46 | 16 | 06 | -0.0172 | -0.0321
01:19:58 | 16 | 07 | -0.0020 |  0.0026
01:20:10 | 16 | 08 |  0.0109 |  0.0005
01:20:22 | 16 | 09 | -0.0110 | -0.0163
01:20:34 | 16 | 10 | -0.0155 | -0.0150
01:20:46 | 16 | 11 | -0.0199 | -0.0146
01:20:58 | 16 | 12 | -0.0231 | -0.0315
01:21:10 | 16 | 13 | -0.0277 | -0.0307
01:21:22 | 16 | 14 | -0.0256 | -0.0214
01:21:34 | 16 | 15 | -0.0289 | -0.0192
01:21:46 | 16 | 16 |  0.0215 |  0.0301
01:21:58 | 16 | 17 |  0.0199 |  0.0147
01:22:10 | 16 | 18 |  0.0166 |  0.0098
01:22:22 | 16 | 19 | -0.0321 | -0.0262
01:22:34 | 16 | 20 |  0.0170 |  0.0102
01:22:46 | 16 | 21 | -0.0242 | -0.0345
01:22:59 | 16 | 22 | -0.0258 | -0.0401
01:23:11 | 16 | 23 |  0.0084 |  0.0130
01:23:22 | 16 | 24 |  0.0330 |  0.0211
01:23:35 | 16 | 25 | -0.0004 |  0.0015
```

1.4.3 Evaluate results

[21]:
```
ic.groupby('fold').mean().boxplot();
```

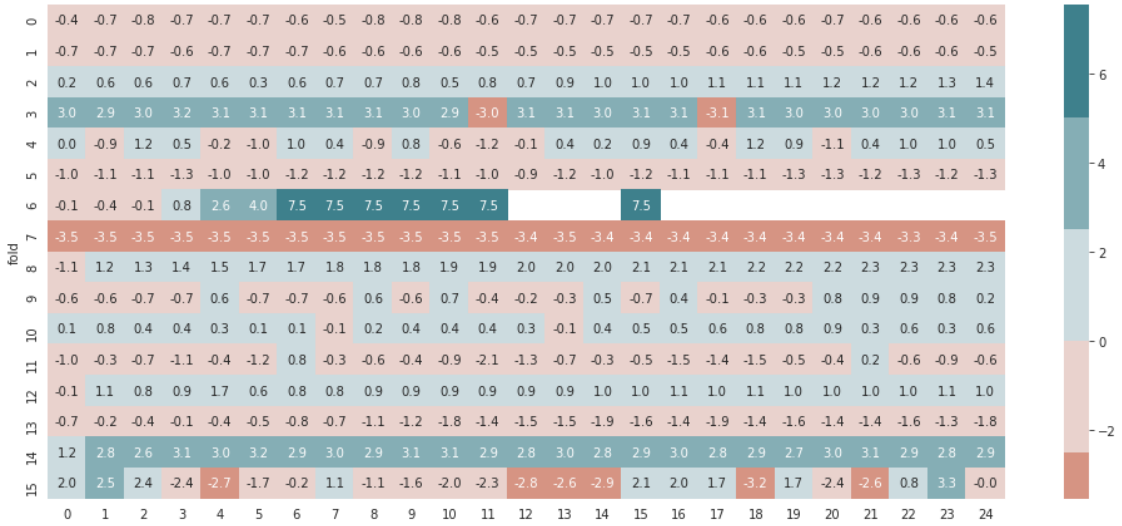


[22]:
```
ic.groupby('fold').mean().mean().sort_index().plot.bar(rot=0);
```



[23]:
```
cmap = sns.diverging_palette(h_neg=20, h_pos=210)
fig, ax = plt.subplots(figsize=(14, 6))
```

```
sns.heatmap(ic.groupby('fold').mean().mul(100), ax=ax, center=0, cmap=cmap,␣
↪annot=True, fmt='.1f')
fig.tight_layout()
```



1.5 Make Predictions

To evaluate the model's predictive accuracy, we compute the daily information coefficient (IC) for the validation set like so:

[24]:
```
def generate_predictions(epoch):
    predictions = []
    for fold, (train_idx, test_idx) in enumerate(cv.split(features)):
        X_train, y_train, X_val, y_val = get_train_valid_data(features, target,␣
↪train_idx, test_idx)
        preds = y_val.to_frame('actual')
        model = make_model(filter1=16, act1='relu', filter2=32,
                           act2='relu', do1=.25, do2=.5, dense=32)
        status = model.load_weights((checkpoint_path / f'ckpt_{fold}_{epoch}').
↪as_posix())
        status.expect_partial()
        predictions.append(pd.Series(model.predict(X_val).squeeze(),␣
↪index=y_val.index))
    return pd.concat(predictions)
```

[25]:
```
preds = {}
for i, epoch in enumerate(ic.drop('fold', axis=1).mean().nlargest(5).index):
    preds[i] = generate_predictions(epoch)
```

[26]:
```
with pd.HDFStore(results_path / 'predictions.h5') as store:
    store.put('predictions', pd.DataFrame(preds).sort_index())
```