Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions

Longhua Chow, Yuewei Gao, Xiangqing Wang, Dafu, Zhu

SOE & WISE, XMU

2024.05.29

Outline

Machine Learning in Financial Time Series Predictions

- Prediction tasks on financial time series are notably challenging
  - **High noise levels**
  - Widely accepted **semi-strong form of market efficiency**
  - **Numerous capital market anomalies**
- Traditional financial models struggle to capture complex nonlinear dependencies.
- Recent evidence shows that machine learning techniques can identify nonlinear structures in financial data.
  - Deep neural networks under-performed compared to gradient-boosted trees and random forests in forecasting S&P 500 constituents from 1992 to 2015 (*Huck (2009, 2010), Takeuchi and Lee (2013), Moritz and Zimmermann (2014), etc.*)
  - Potential for improvement through parameter configurations

Machine Learning in Financial Time Series Predictions

- Prediction tasks on financial time series are notably challenging
  - **High noise levels**
  - Widely accepted **semi-strong form of market efficiency**
  - **Numerous capital market anomalies**
- Traditional financial models struggle to capture complex nonlinear dependencies.
- Recent evidence shows that machine learning techniques can identify nonlinear structures in financial data.
  - Deep neural networks under-performed compared to gradient-boosted trees and random forests in forecasting S&P 500 constituents from 1992 to 2015 (*Huck (2009, 2010), Takeuchi and Lee (2013), Moritz and Zimmermann (2014), etc.*)
  - Potential for improvement through parameter configurations

LSTM for Financial Market Predictions

This paper: tries to investigate the application of LSTM networks in financial time series prediction, offering a detailed data science approach and a proposed trading strategy.

Main Findings

- LSTM networks **outperform** memory-free classification methods, (a random forest (RAF), a deep neural net (DNN), and a logistic regression classifier (LOG))
- Unveil sources of **profitability**, shedding light into the black box of artificial neural networks.
  - One common pattern among the stocks selected for trading –high volatility and a short-term reversal return profile.
- Formalize a **rules-based short-term reversal strategy** that yields 0.23 percent prior to transaction costs

Contribution

1. Provide the large-scale empirical application of LSTM networks to financial time series prediction tasks
2. Disentangle the black box "LSTM", and unveil common patterns of stocks that are selected for profitable trading
3. Devise a simplified rules-based trading strategy.
   - Short short-term winners and buy short-term losers, and hold the position for one day

Outline

Memory-Free Classification

- **Random Forest:** an ensemble learning method that makes predictions by combining multiple decision trees
- **Deep Net:** an artificial neural network structure consisting of multiple layers (input layers, hidden layers, output layers) of neurons
- **Logistic Regression:** a statistical learning method used for binary classification problems

From Statistical Models to RNN Models

- For an $AR(p)$ model, we predict the future using the following formula:

$$\hat{y_{t+1}} = E[y_{t+1}|y_t, y_{t-1}, ..., y_{t-p+1}]$$

- To fully utilize the information in the entire series, the desired prediction form is as follows:

$$\hat{y_{t+1}} = E[y_{t+1}|y_t, y_{t-1}, ..., y_1]$$

- This form can result in a massive number of parameters to estimate due to the increasing data volume.

From Statistical Models to RNN Models

- Consider defining a new variable h_t that can represent the information of the original variable up to time point t, and conduct predictions by this h_t:

$$\hat{y_{t+1}} = E[y_{t+1}|h_t]$$

- **Update** the value of h_t step by step through a certain function:

$$h_{t+1} = f(h_t, y_{t+1})$$

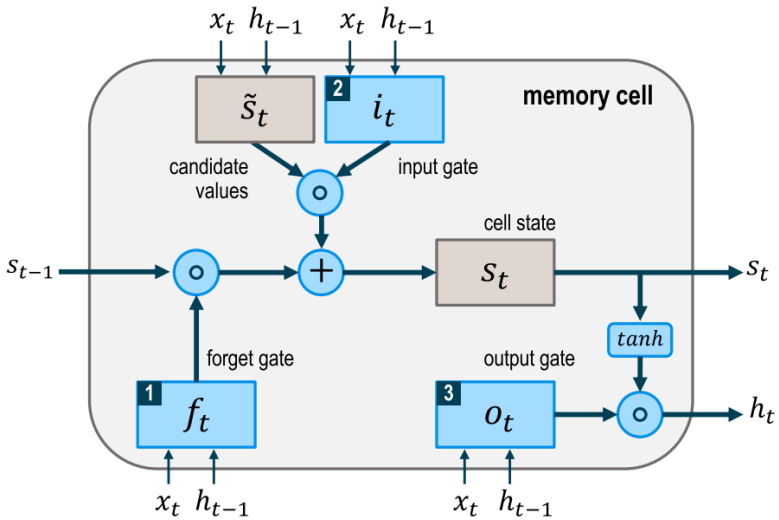
- The specific prediction parameters and the parameters for updating h_t can be learned through a neural network.

LSTM Networks

- LSTM networks are designed to tackle the challenges faced by traditional RNNs
  - Handle **long-term dependencies** more effectively
  - Mitigate issues related to vanishing or exploding **gradients**
- LSTM networks consist of an input layer, one or multiple hidden layers, and an output layer.
  - Neurons in the input layer $\rightarrow$ the explanatory variables (features)
  - output layer size matches the desired output space (e.g. two neurons indicating binary outcomes)

LSTM Networks

- The critical component of LSTM networks is the hidden layer(s) containing memory cells with three gates: forget gate (f_t), input gate (i_t), and output gate (o_t).



1 Forget gate:

Defines which information to remove from the memory (cell state)

2 Input gate:

Defines which information to add to the memory (cell state)

3 Output gate:

Defines which information from the memory (cell state) to use as output

Process: First Step

1. Determines which information should be removed from its previous cell states s_{t-1}

- Calculate the activation values f_t of the forget gates at t based on the current input x_t, the last outputs h_{t-1} of the memory cells and the bias terms b_f of the forget gates
- Sigmoid function to scale all activation values into the range between 0 (completely forget) and 1 (completely remember)

$$f_t = sigmoid(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f)$$

Process: Second Step

2. Determines which information should be added to the network's cell states (s_t)

- compute candidate values $\tilde{s}_t$ that could potentially be added to the cell states

$$\tilde{s}_t = tanh(W_{\tilde{s},x}x_t + W_{\tilde{s},h}h_{t-1} + b_{\tilde{s})}$$

- Calculate the activation values i_t of the input gates

$$i_t = sigmoid(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i)$$

Process: Third Step & Forth Step

3. Calculate the new cell states s_t based on the results of the previous two steps[1]

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}_t$$

4. Derive the output h_t of the memory cells

$$o_t = sigmoid(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o)$$

$$h_t = o_t \circ tanh(s_t)$$

# of Parameters to Train[2]:

$$4hi + 4h + 4h^2 = 4(hi + h + h^2) = 4(h(i+1) + h^2)$$

[1] $\circ$ denoting the Hadamard (elementwise) product

[2] h denote the number of hidden units of the LSTM layer, and i is the number of input features

Data

- **Data:** S&P 500 index constituents data from December 1989 to September 2015
- **Address survivor bias**
  - Obtain the month end constituent lists & consolidate the lists into one binary matrix
  - Daily total return indices for stocks to calculate returns accurately with adjustments for corporate actions and stock splits.

Summary Statistics

Equal-weighted portfolios per industry sector, monthly average summary statistics

Industry	No. of stocks	Mean return	Standard deviation	Skewness	Kurtosis
Industrials	80.6	0.99	5.36	−0.19	1.71
Consumer services	72.6	1.07	5.27	−0.20	2.59
Basic materials	35.2	0.90	6.31	−0.02	2.24
Telecommunications	10.7	0.92	6.50	0.34	4.76
Health care	41.3	1.33	4.40	−0.40	1.18
Technology	50.3	1.41	8.50	−0.06	1.11
Financials	78.0	1.13	6.17	−0.39	2.44
Consumer goods	65.2	1.04	4.53	−0.44	3.02
Oil and gas	31.2	1.00	6.89	−0.03	1.06
Utilities	34.6	0.85	4.54	−0.43	1.72
All	499.7	1.04	4.78	−0.49	2.01

Outline

Generation of Training and Trading sets

- "Study period" as a training-trading set, consisting of a training period of 750 days and a trading period of 250 days
  - The entire dataset from 1990 to 2015 is split into 23 of these study periods with non-overlapping trading periods

Notation:

- n_i: the number of stocks that are a S&P 500 constituent at the very last day of the training period in study period i,
  - n_i is very close to 500
  - All n_i stocks with the history they have available for the training set
  - The trading set is also composed of all n_i stocks.

Features

- $(P_t^s)_{t \in T}$: price process of stock s at time t
- $R_t^{m,s}$: simple return for a stock s over m periods

$$R_t^{m,s} = \frac{P_t^s}{P_t^{s-m}} - 1$$

- μ_{train}^m: mean
- σ_{train}^m: standard deviation

$$\tilde{R}_t^{m,s} = \frac{R_t^{m,s} - \mu_{train}^m}{\sigma_{train}^m}$$

- T_{study}: the number of days in the study period
- V: feature vector

$$V = n_i * T_{study}$$

Features

- Sequences are constructed in the format $\tilde{R}^{1,s}_{t-239}$, $\tilde{R}^{1,s}_{t-238}$, ..., $\tilde{R}^{1,s}_{t}$ for each stock s and each $t \geq 240$ of the study period
- In total, each study period comprises approximately 380,000 sequences, with around 255,000 used for in-sample training and about 125,000 utilized for out-of-sample predictions.

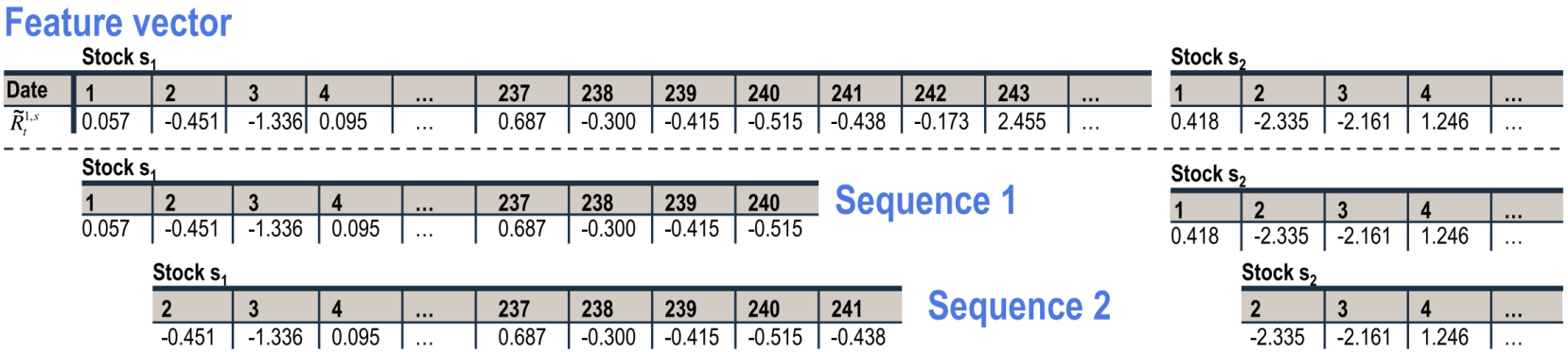


Fig. 1. Construction of input sequences for LSTM networks (both, feature vector and sequences, are shown transposed).

Targets

Define a binary classification problem

- Class 0: stocks with the one-period return $R_{t+1}^{1,s}$ smaller than the cross-sectional median return of all stocks in period $t+1$
- Class 1: stocks with the one-period return larger than or equal to the cross-sectional median.

Specified Topology

Model Setting

- Cross-entropy loss
- Input layer with 1 feature and 240 timesteps.
- LSTM layer with $h = 25$ hidden neurons and a dropout value of 0.1.
- Output layer (dense layer) with two neurons and softmax activation function.

```
#        lstm
model_lstm = Sequential()
#                          lstm                          (240, 1)
                             25              Dropout
#


model_lstm.add(LSTM(25, input_shape = (240, 1)))
model_lstm.add(Dropout(0.1))
model_lstm.add(Dense(2, activation = 'softmax'))

#
model_lstm.compile(optimizer = RMSprop(), loss = '
    sparse_categorical_crossentropy', metrics = ['accuracy'])

#
checkpoint = ModelCheckpoint('best_model.h5', monitor = 'val_loss',
     save_best_only = True, mode = 'min')
early_stopping = EarlyStopping(monitor = 'val_loss', patience = 10,
     restore_best_weights = True)
```

```
#                          loss
record_lstm = model_lstm.fit(X_train, y_train, epochs = 1000,
    batch_size = 64, validation_split = 0.2, callbacks = [
    checkpoint, early_stopping], verbose = 1)

#        loss
plt.figure(figsize = (21, 14))
plt.plot(record_lstm.history['loss'], label = 'Training Loss')
plt.plot(record_lstm.history['val_loss'], label = 'Validation Loss'
    )
plt.title('Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

#
y_pred_lstm = model_lstm.predict(X_test)
y_pred_lstm = np.argmax(y_pred_lstm, axis = 1)
pred_evaluate(y_test, y_pred_lstm)
```

Three advanced methods via keras

1. RMSprop, a mini-batch version of rprop, as an optimizer
2. Dropout (0.1) regularization within the recurrent layer
3. Further split the training data to training sets and validation set

Random forest

A state-of-the-art machine learning model that requires virtually no tuning and usually delivers good results

```
#  b a s e l i n e 2
from sklearn.ensemble import RandomForestClassifier

#
rf = RandomForestClassifier(n_estimators = 1000, max_depth = 20,
    random_state = 42)

#
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1])
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1])

#
rf.fit(X_train, y_train)

#
y_pred_rf = rf.predict(X_test)
pred_evaluate(y_test, y_pred_rf)
```

Deep Net

- A feed forward neural network with 31 input neurons, 31 neurons in the first, 10 in the second, 5 in the third hidden layer, and 2 neurons in the output layer
- Softmax in the output layer
- Dropout is set to 0.5, and
- L1 regularization with shrinkage 0.00001 is used

```
# baseline1 DNN
from keras.regularizers import l1
from keras.layers import Lambda
from keras import backend as K

#
    Maxout
def maxout(inputs, num_units):
    shape = K.shape(inputs)
    shape = [shape[0], shape[1] // num_units, num_units]
    outputs = K.max(K.reshape(inputs, shape), axis = -1)

    return outputs

#        DNN
model_dnn = Sequential()
model_dnn.add(Dense(31, input_dim = 240, activation = 'linear'))
model_dnn.add(Lambda(maxout, arguments = {'num_units': 31}))
model_dnn.add(Dropout(0.5))
model_dnn.add(Dense(10, activation = 'linear'))
model_dnn.add(Lambda(maxout, arguments={'num_units': 10}))
model_dnn.add(Dropout(0.5))
model_dnn.add(Dense(5, activation='linear'))
model_dnn.add(Lambda(maxout, arguments={'num_units': 5}))
model_dnn.add(Dropout(0.5))
model_dnn.add(Dense(2, activation = 'softmax', kernel_regularizer =
    l1(0.00001)))

#
model_dnn.compile(optimizer = 'adam', loss = '
    sparse_categorical_crossentropy', metrics = ['accuracy'])
```

```
#
model_dnn.compile(optimizer = 'adam', loss = '
    sparse_categorical_crossentropy', metrics = ['accuracy'])

#                        loss
record_dnn = model_dnn.fit(X_train, y_train, epochs = 1000,
    batch_size = 64, validation_split = 0.2, verbose = 1)

#        loss
plt.figure(figsize = (21, 14))
plt.plot(record_dnn.history['loss'], label = 'Training Loss')
plt.plot(record_dnn.history['val_loss'], label = 'Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

#
y_pred_dnn = model_dnn.predict(X_test)
y_pred_dnn = np.argmax(y_pred_dnn, axis = 1)
pred_evaluate(y_test, y_pred_dnn)
```

Logistic Regression

- Optimal L2 regularization is determined among 100 choices on a logarithmic scale between 0.0001 and 10,000 via 5-fold cross-validation on the respective training set and
- L-BFGS is deployed to find an optimum
- Maximum number of iterations to 100

```
#  b a s e l i n e 3 logistics
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

#        logistics
log_reg = LogisticRegression(solver = 'lbfgs', max_iter = 100)
param_grid = {'C': np.logspace(-4, 4, 100)}

#        GridSearchCV        5

grid_search = GridSearchCV(log_reg, param_grid, cv = 5, scoring = '
    accuracy')
grid_search.fit(X_train, y_train)

#
best_log_reg = grid_search.best_estimator_

#
y_pred_lr = best_log_reg.predict(X_test)
pred_evaluate(y_test, y_pred_lr)
```

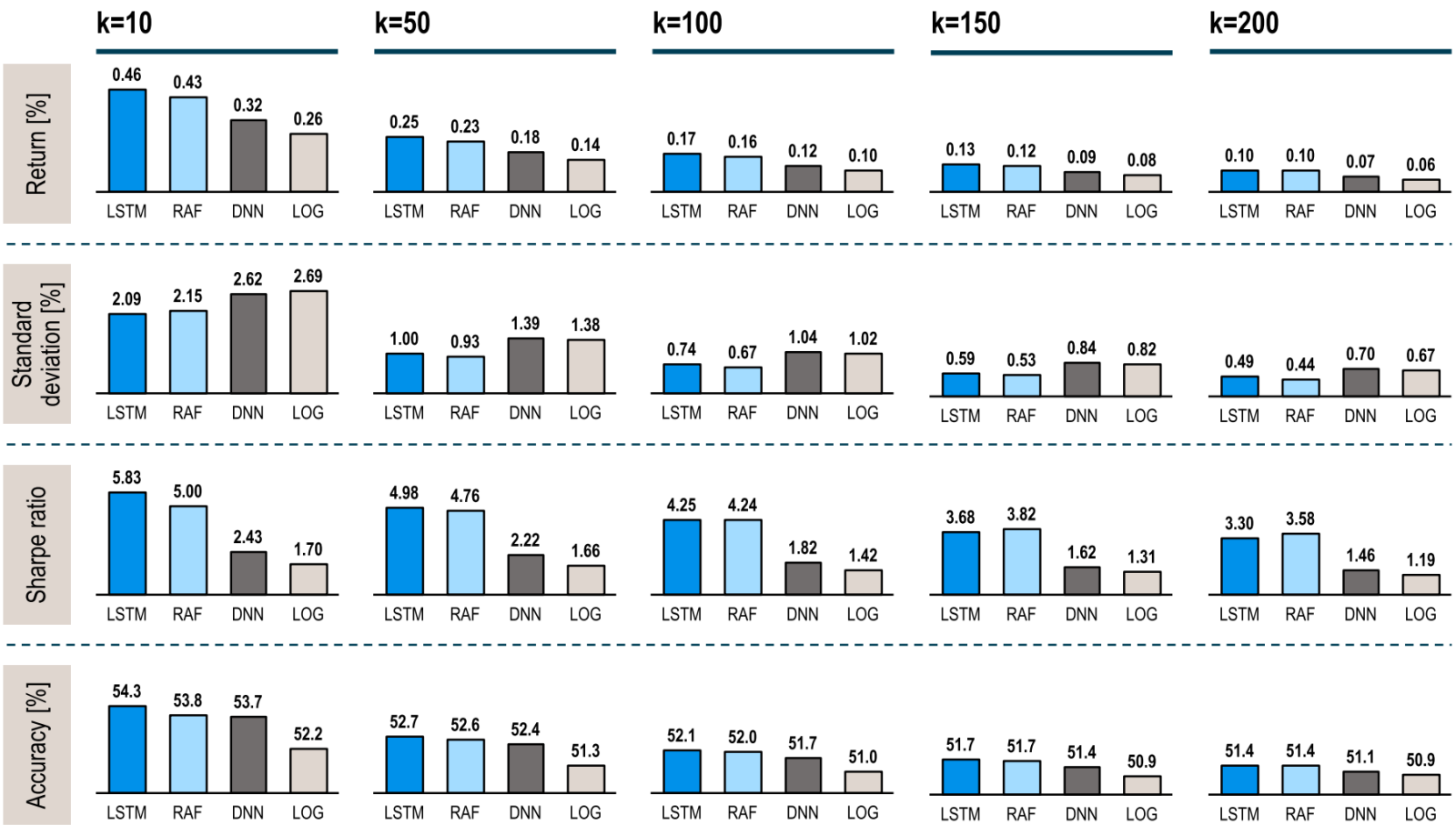
Forecasting, Ranking, and Trading

- Forecast the probability $\hat{P}^s_{t+1|t}$ for each stock s to out-/underperform the cross-sectional median in period $t+1$, making only use of information up until time t.
- Rank all stocks for each period t + 1 in descending order of this probability.
- Go long the top k and short the flop k stocks of each ranking, for a long-short portfolio consisting of $2k$ stocks

Outline

Overview

- LSTM has better overall performance on return, standard deviation, sharpe ratio and accuracy.



Details on Predictive Accuracy

- LSTM passed Diebold-Mariano(DM) test[3] and Pesaran-Timmermann(PT) test[4]

A: DM test					**B**: PT test	
i	j =	LSTM	RAF	DNN	LOG	Method	Result
LSTM		-	0.0143	0.0037	0.0000	LSTM	0.0000
RAF		0.9857	-	0.3180	0.0000	RAF	0.0000
DNN		0.9963	0.6820	-	0.0000	DNN	0.0000
LOG		1.0000	1.0000	1.0000	-	LOG	0.0000

- Provide a statistical estimate for the probability of the LSTM network having randomly achieved these results, reject null hypothesis that accuracy = 0.5
- Construct 100,000 random portfolios, plot the return distribution, reject the null hypothesis that portfolio given by LSTM is randomly constructed

[3] reject null hypothesis that it has inferior performance than the other three

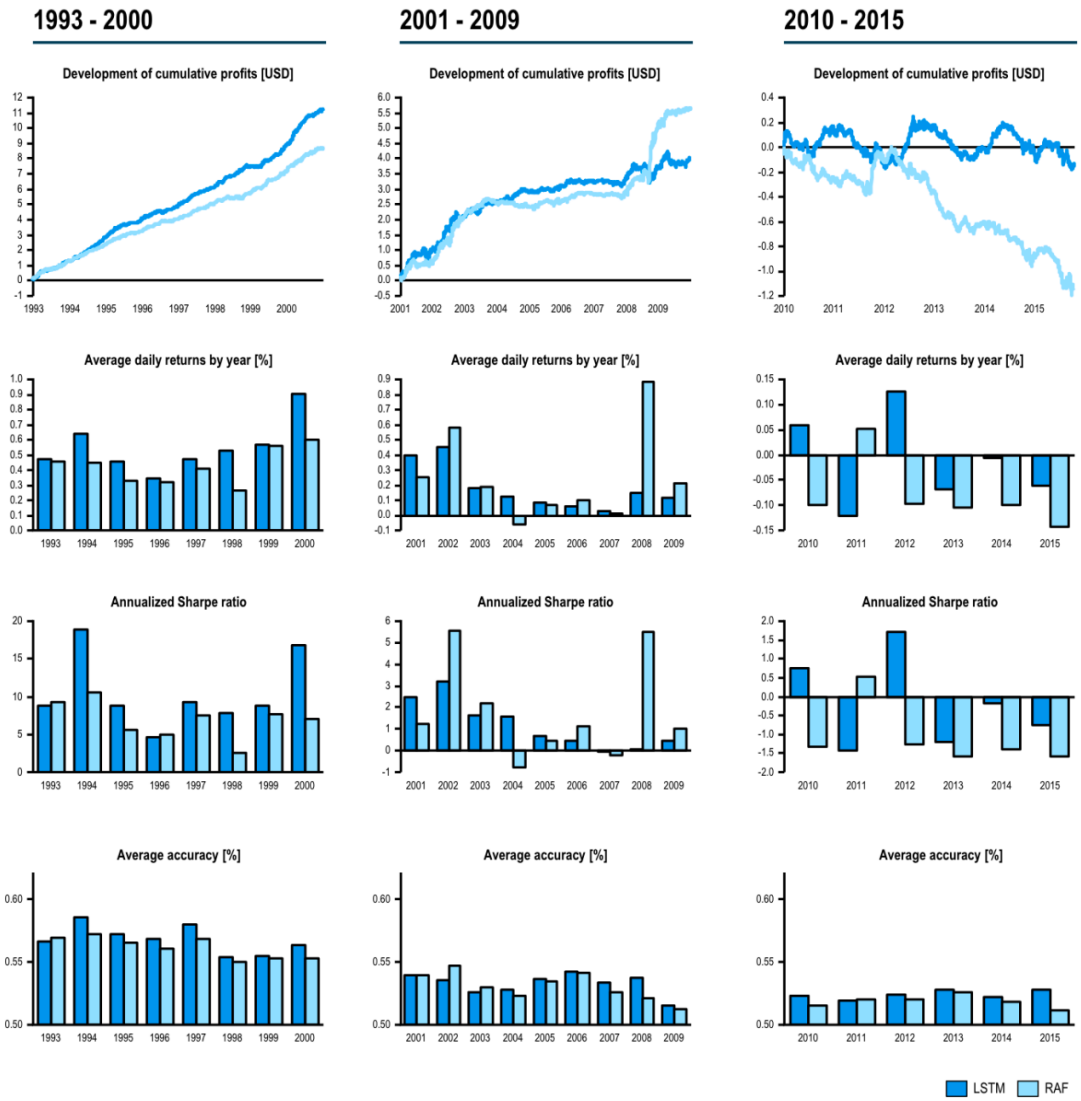
[4] reject the null hypothesis that prediction and response are independently distributed

Details on Financial Performance

		Before transaction costs				After transaction costs				
		LSTM	RAF	DNN	LOG	LSTM	RAF	DNN	LOG	MKT
A	Mean return (long)	0.0029	0.0030	0.0022	0.0021	0.0019	0.0020	0.0012	0.0011	-
	Mean return (short)	0.0017	0.0012	0.0010	0.0005	0.0007	0.0002	0.0000	-0.0005	-
	Mean return	0.0046	0.0043	0.0032	0.0026	0.0026	0.0023	0.0012	0.0006	0.0004
	Standard error	0.0003	0.0003	0.0004	0.0004	0.0003	0.0003	0.0004	0.0004	0.0001
	t-Statistic	16.9336	14.1136	8.9486	7.0006	9.5792	7.5217	3.3725	1.6666	2.8305
	Minimum	-0.2176	-0.2058	-0.1842	-0.1730	-0.2196	-0.2078	-0.1862	-0.1750	-0.0895
	Quartile 1	-0.0053	-0.0050	-0.0084	-0.0089	-0.0073	-0.0070	-0.0104	-0.0109	-0.0046
	Median	0.0040	0.0032	0.0025	0.0022	0.0020	0.0012	0.0005	0.0002	0.0008
	Quartile 3	0.0140	0.0124	0.0140	0.0133	0.0120	0.0104	0.0120	0.0113	0.0058
	Maximum	0.1837	0.3822	0.4284	0.4803	0.1817	0.3802	0.4264	0.4783	0.1135
	Share > 0	0.6148	0.6078	0.5616	0.5584	0.5574	0.5424	0.5146	0.5070	0.5426
	Standard dev.	0.0209	0.0215	0.0262	0.0269	0.0209	0.0215	0.0262	0.0269	0.0117
	Skewness	-0.1249	2.3052	1.2724	1.8336	-0.1249	2.3052	1.2724	1.8336	-0.1263
	Kurtosis	11.6967	40.2716	20.6760	30.2379	11.6967	40.2716	20.6760	30.2379	7.9791
B	VaR 1%	-0.0525	-0.0475	-0.0676	-0.0746	-0.0545	-0.0495	-0.0696	-0.0766	-0.0320
	CVaR 1%	-0.0801	-0.0735	-0.0957	-0.0995	-0.0821	-0.0755	-0.0977	-0.1015	-0.0461
	VaR 5%	-0.0245	-0.0225	-0.0333	-0.0341	-0.0265	-0.0245	-0.0353	-0.0361	-0.0179
	CVaR 5%	-0.0430	-0.0401	-0.0550	-0.0568	-0.0450	-0.0421	-0.0570	-0.0588	-0.0277
	Max. drawdown	0.4660	0.3187	0.5594	0.5595	0.5233	0.7334	0.9162	0.9884	0.5467
C	Return p.a.	2.0127	1.7749	1.0610	0.7721	0.8229	0.6787	0.2460	0.0711	0.0925
	Excess return p.a.	1.9360	1.7042	1.0085	0.7269	0.7764	0.6359	0.2142	0.0437	0.0646
	Standard dev. p.a.	0.3323	0.3408	0.4152	0.4266	0.3323	0.3408	0.4152	0.4266	0.1852
	Downside dev. p.a.	0.2008	0.1857	0.2524	0.2607	0.2137	0.1988	0.2667	0.2751	0.1307
	Sharpe ratio p.a.	5.8261	5.0001	2.4288	1.7038	2.3365	1.8657	0.5159	0.1024	0.3486
	Sortino ratio p.a.	10.0224	9.5594	4.2029	2.9614	3.8499	3.4135	0.9225	0.2583	0.7077

- Return characteristics: the highest mean return (0.46)
- Risk characteristics: the lowest maximum drawdown (52.33%)
- Annualized risk-return metrics: the highest annualized returns (82.29%)

A Critical Review of LSTM Profitability Over Time



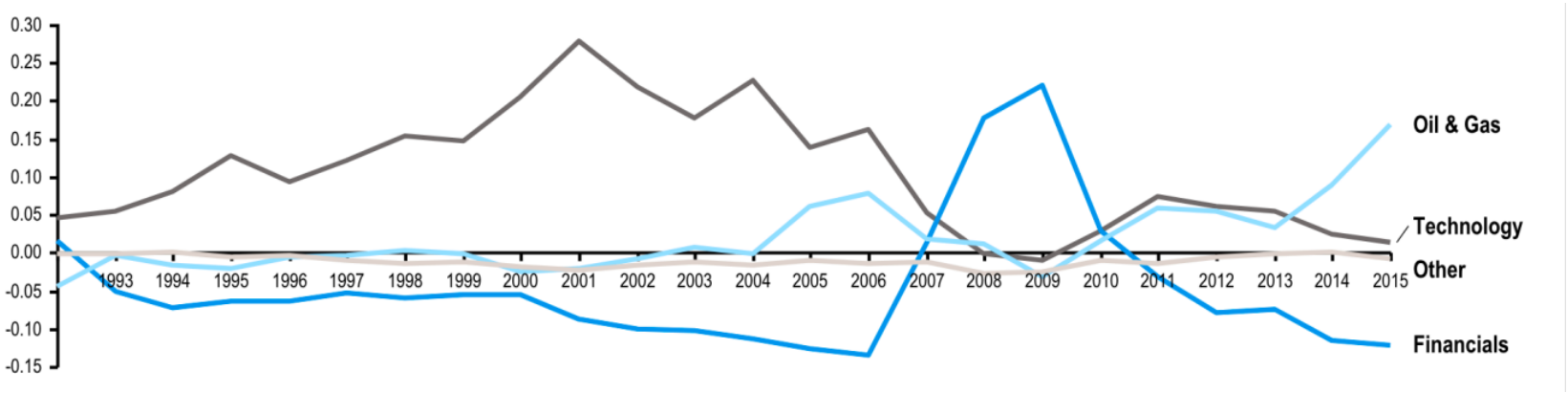
- 1993/01 to 2000/12: Benefits because LSTM is not introduced until 1997, and GPU computing does not emerge until 2000s.
- 2001/01 to 2009/12: A key advantage of these tree-based methods is their **robustness to noise and outliers** - which plays out during such volatile times.

  Profit driven by two factors: Benefit largely from short side, cause the economy is plunging; Unable to short sell, cause no one is buying, liquidity is limited.
- 2010/01 to 2015/10: LSTM still perform well at time of deterioration.

Industry Breakdown

The difference between the share of an industry in the k = 10 portfolio and the share of that industry in the SP 500 at that time.

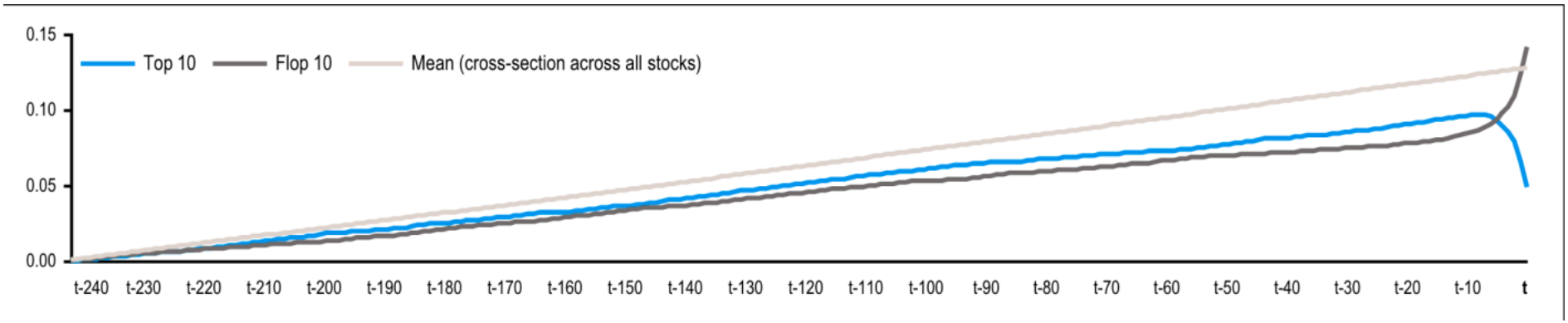
- A significant overweight of technology stocks building up end of the 90s - the growing dot-com bubble and its bust.
- A rise in financial stocks around the years 2008/2009 - the global financial crisis.
- Oil & gas stocks gain in weight as of 2014 - falling together with the recent oil glut and the significant drop in crude oil prices.



Sources of profitability - Common patterns in the top and flop stocks

Weak momentum, strong reverse

- The top stocks exhibit highly negative returns in the last days prior to the prediction, and the flop stocks highly positive returns. This behavior corresponds to short-term reversal strategies.



Outline

Main Findings

- LSTM networks **outperform** memory-free classification methods, (a random forest (RAF), a deep neural net (DNN), and a logistic regression classifier (LOG))
- Unveil sources of **profitability**, shedding light into the black box of artificial neural networks.
  - One common pattern among the stocks selected for trading –high volatility and a short-term reversal return profile.
- Formalize a **rules-based short-term reversal strategy** that yields 0.23 percent prior to transaction costs

Thanks!