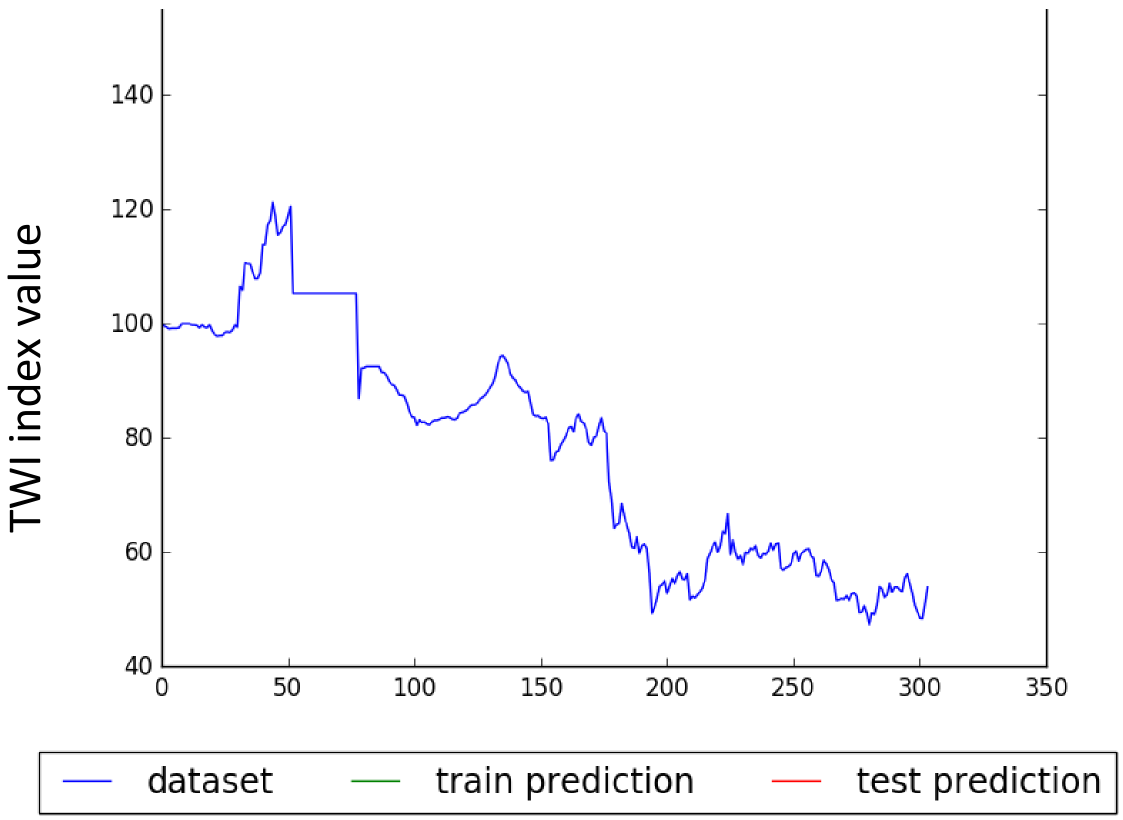
Forex Prediction using Deep Learning Neural Networks

By Abhishek Jain, Pradyumna Kaushik, Sreedhar Kumar

Dataset description



X-axis = time in months

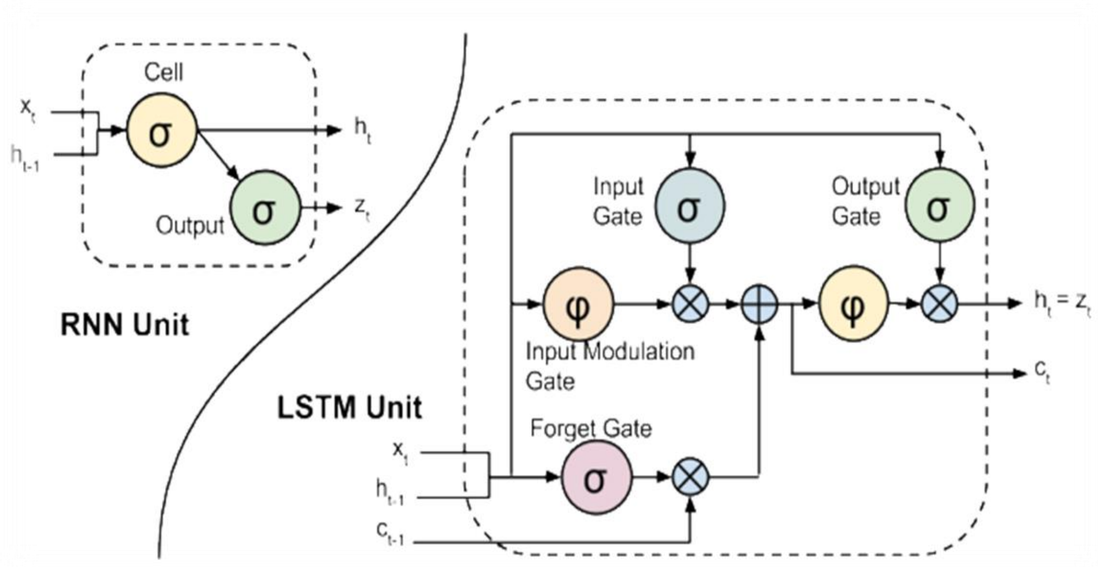
The **Trade Weighted Index** (**TWI**) is a weighted average of a basket of **currencies** that reflects the importance of the sum of USA's exports and imports of goods by country.

This dataset consists of 304 time points, each being the month ending value of the index, from May 1970 to August 1995.

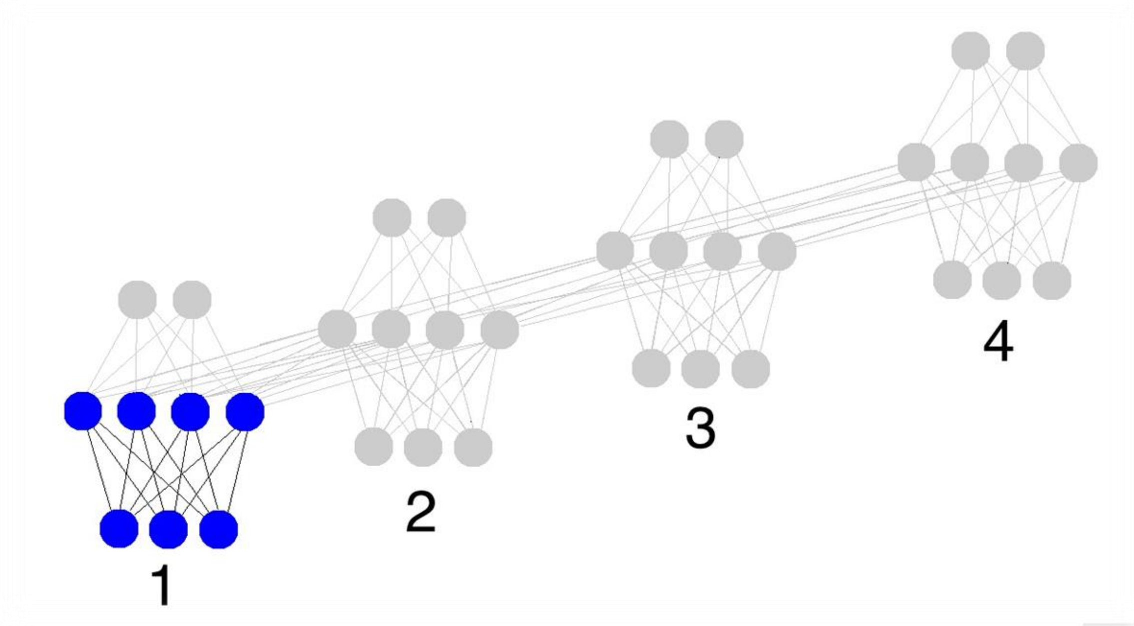
The interpretation of the effective exchange rate is that if the index rises, other things being equal, the purchasing power of that currency also rises.

https://datamarket.com/data/set/22tb/exchange-rate-twi-may-1970-aug-1995#!ds=22tb&display=line

LSTM Neural Network Architecture



Motivation:
Long Short-Term Memory is a recurrent neural network that is efficient in sequence learning tasks involving long term dependencies.

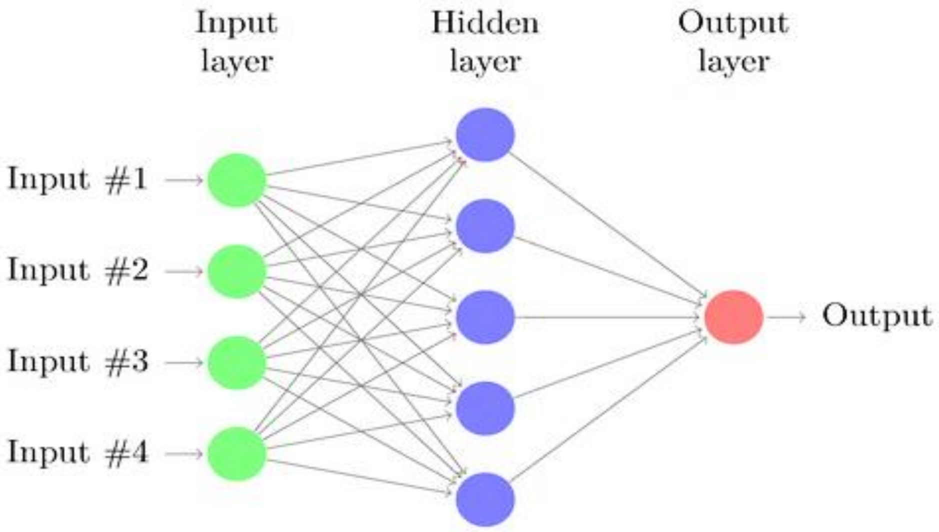


Forget Gate: conditionally decides what information to discard from the unit.

Input Gate: conditionally decides which values from the input to update the memory state.

Output Gate: conditionally decides what to output based on input and the memory of the unit

Technical terms



No: of layers

No: of computational units within a layer

Optimiser – *algorithm used to update and learn the correct weights. Variants include stochastic gradient descent, RMSprop, Adam etc.*

Learning rate - *This number controls the step size of the weight update and is critical to the convergence of the algorithm to the global minima of the objective function.*

Momentum - *Momentum simply adds a fraction m of the previous weight updates to the current one. When the gradient keeps pointing in the same direction, this will increase the size of the steps taken towards the minimum.*

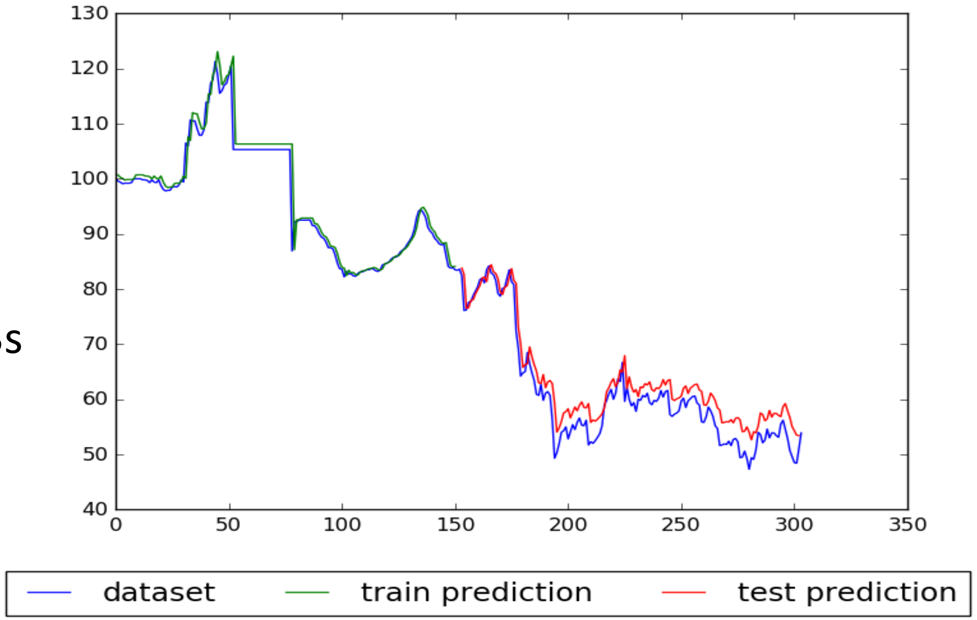
Epochs – *An epoch is one complete presentation of the data set to be learned by a learning machine. Learning machines like feedforward neural nets, that use iterative algorithms, often need many epochs during their learning phase.*

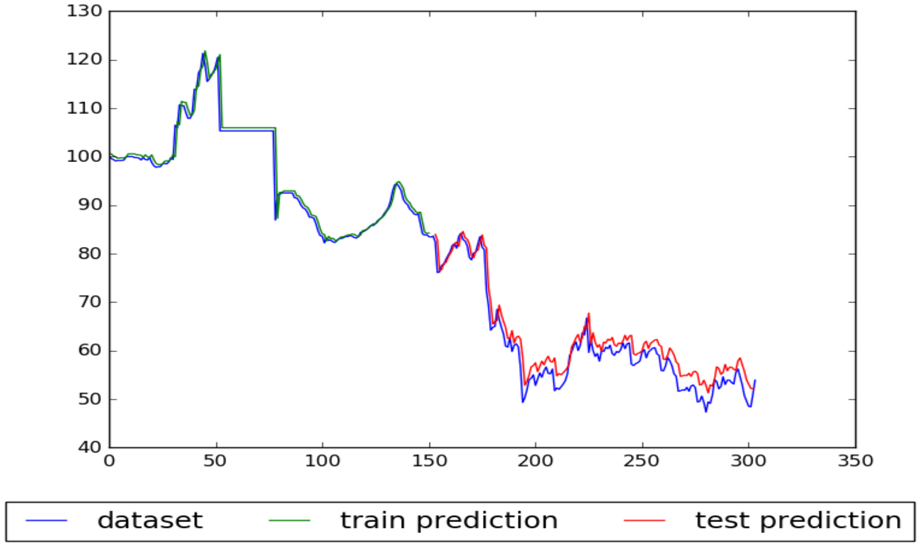
Network architecture

- ***LSTM with RMSprop*** - variation with number of layers

learning_rate=0.001, training_percent=0.5, epochs = 100

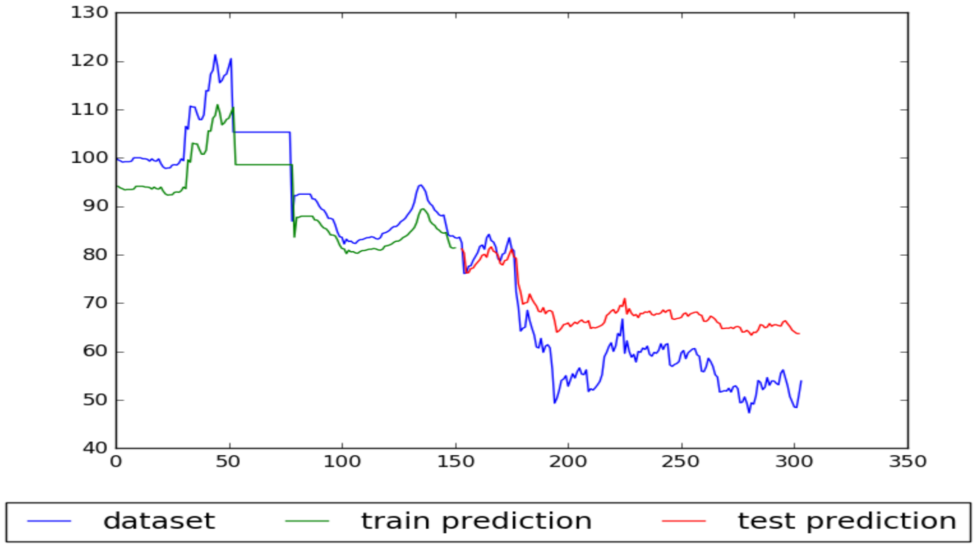
train-time = 39.463s
test-RMSE = 3.313
n_layers=3,
layer_dimensions
=[1, 60, 1]





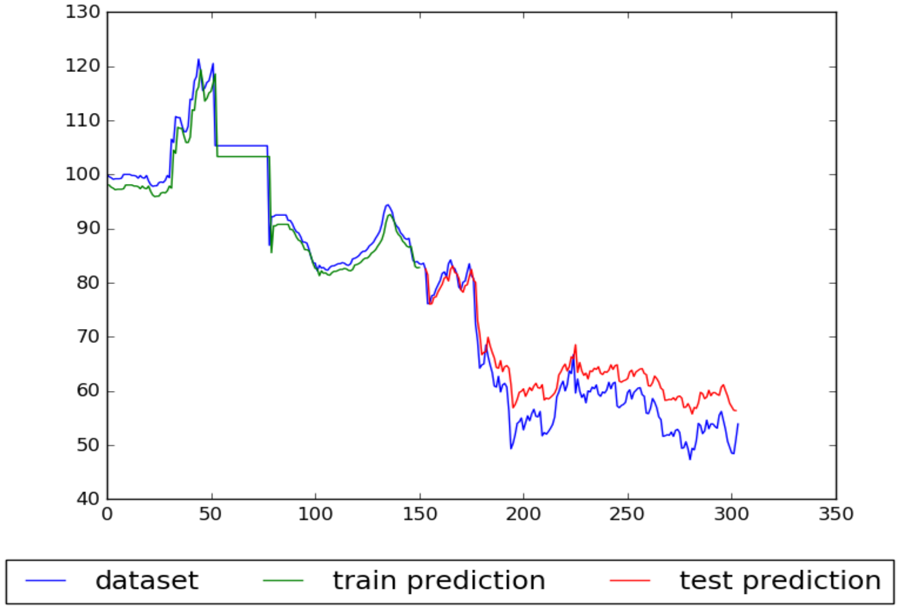
train-time = 57.046s
test-RMSE = 2.961
n_layers=4,
layer_dimensions=[1, 60, 60, 1]

train-time = 90.995s
test-RMSE = 9.600
n_layers=6,
layer_dimensions=
[1, 60, 60, 60, 60, 1]



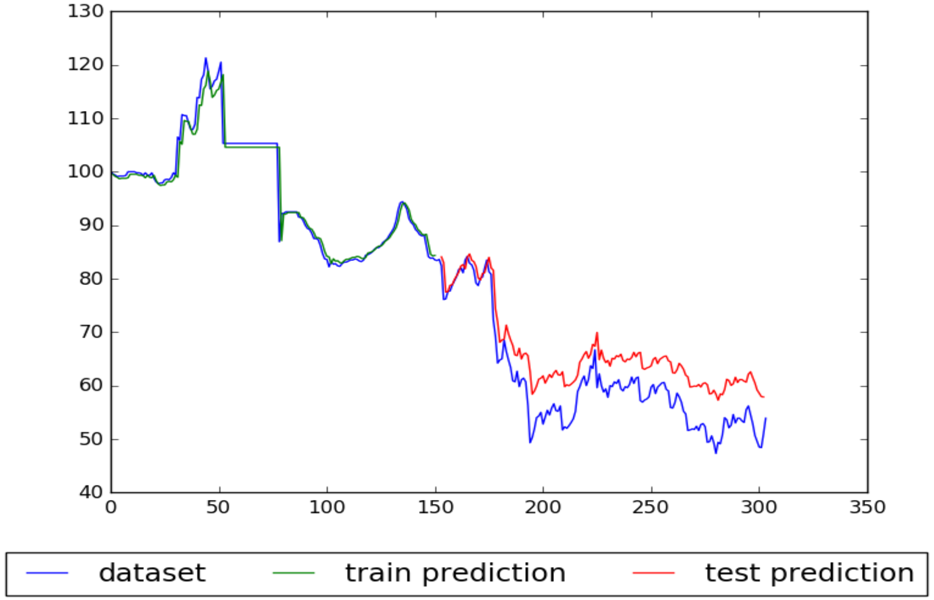
No: of units within a layer

LSTM with RMSprop - variation with number of units
learning_rate=0.001, training_percent=0.5, epochs = 100

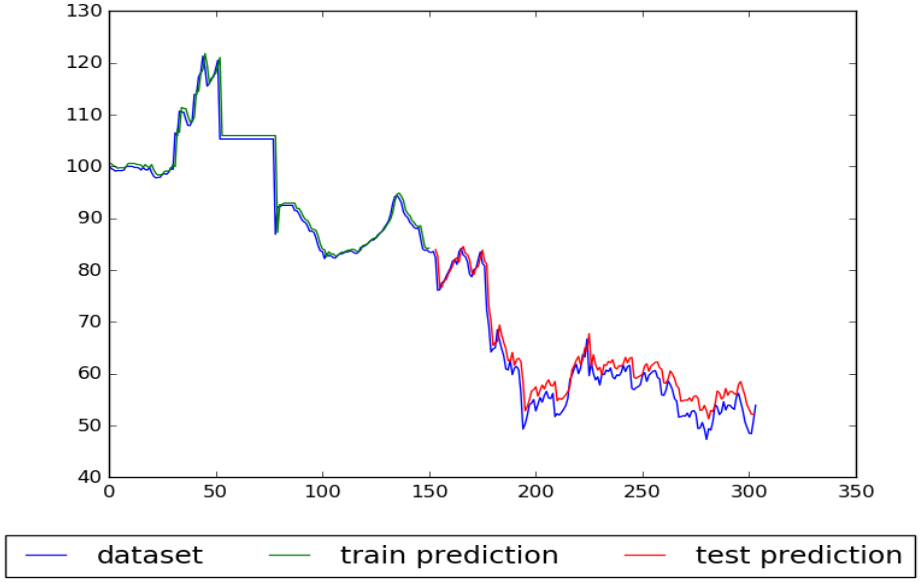


train-time = 53.131s
test-RMSE = 6.103
n_layers=4,
layer_dimensions=[1, 10, 10, 1]

train-time = 53.092s
test-RMSE = 5.044
n_layers=4,
layer_dimension=
[1, 20, 20, 1]



train-time = 57.047s
test-RMSE = 2.961
n_layers=4,
layer_dimension=
[1, 60, 60, 1]

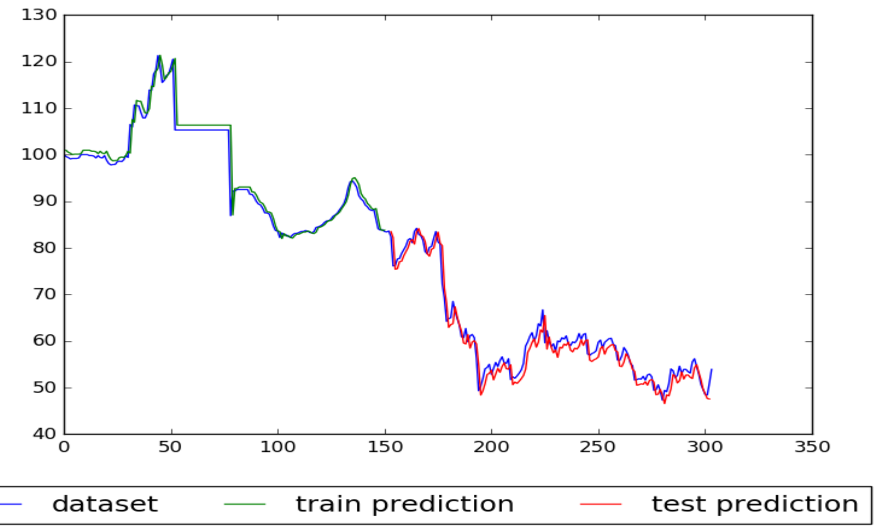


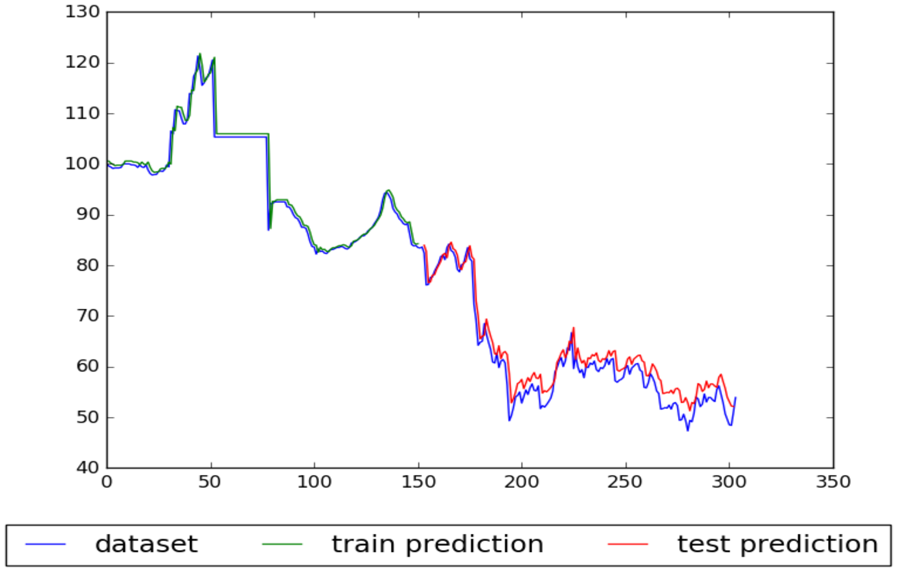
Optimsers

- ***Stochastic Gradient Descent*** - training-time = 37.251s, test-RMSE =27.53
- ***RMSprop*** - training-time = 57.047s, test-RMSE = 2.962
- ***Adam*** - training-time = 60.804s, test-RMSE = 2.419

n_layers=4, layer_dimension=[1, 60, 60, 1]

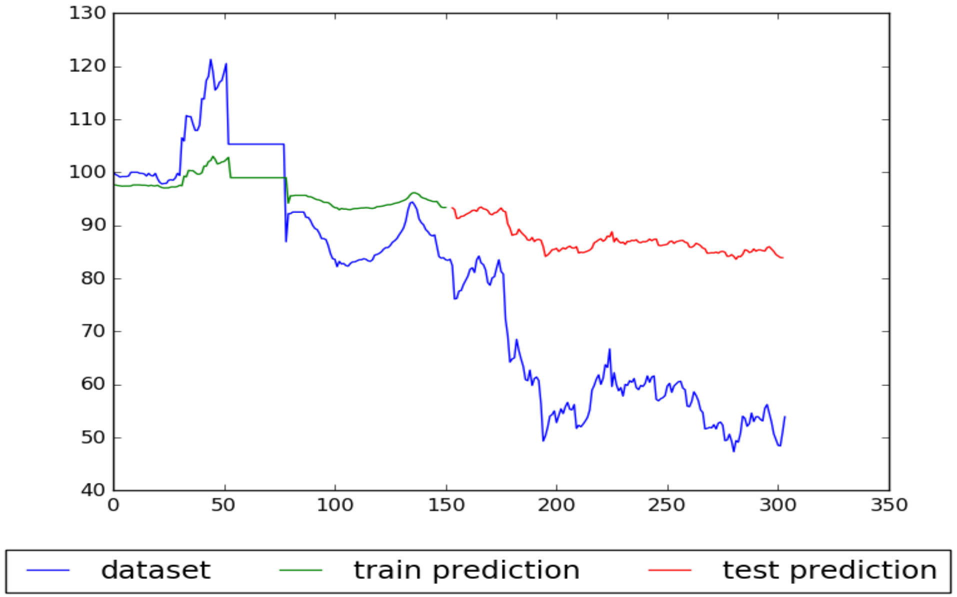
Adam





RMSprop

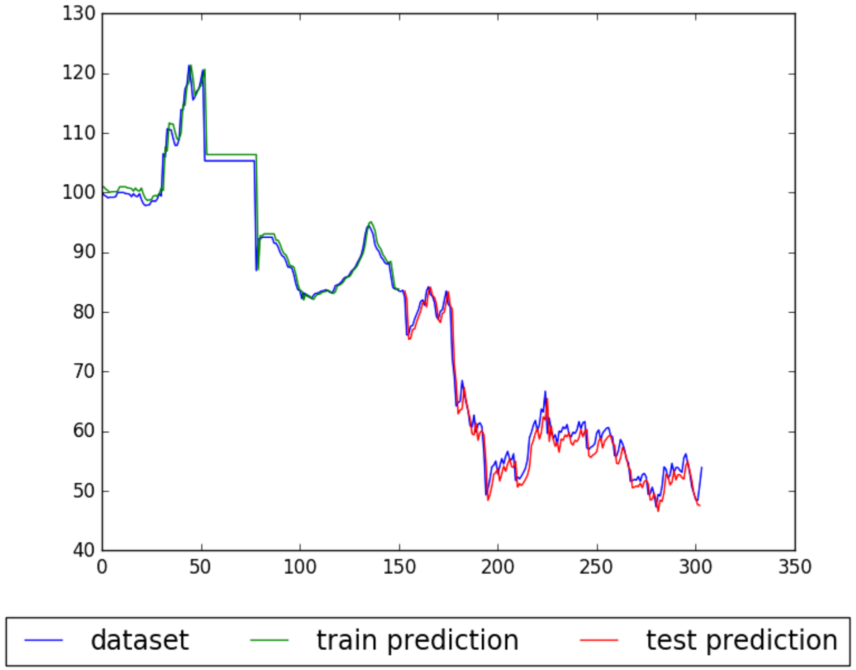
SGD



Learning rate

LSTM with Adam Optimiser

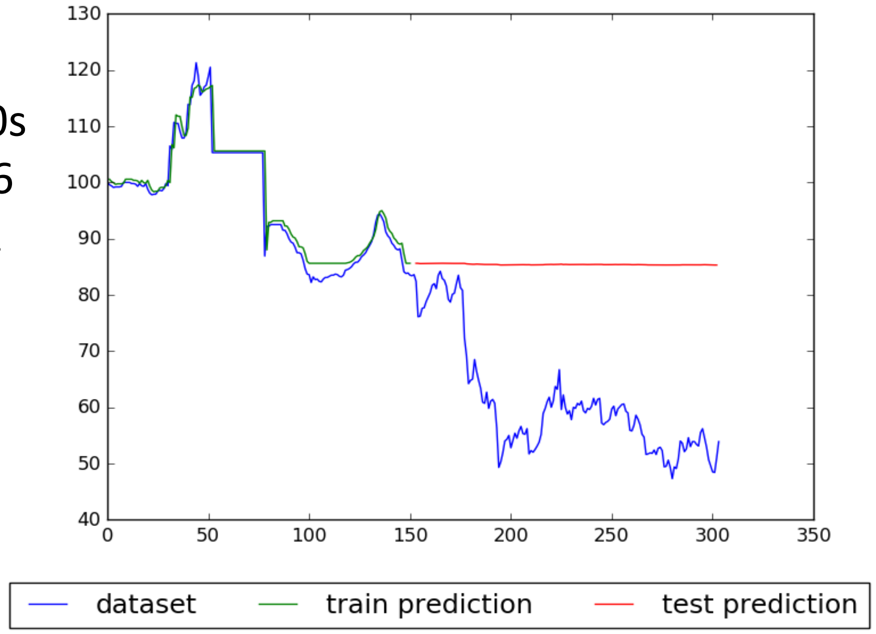
n_layers = 4, layer_dimension = [1, 60, 60, 1]



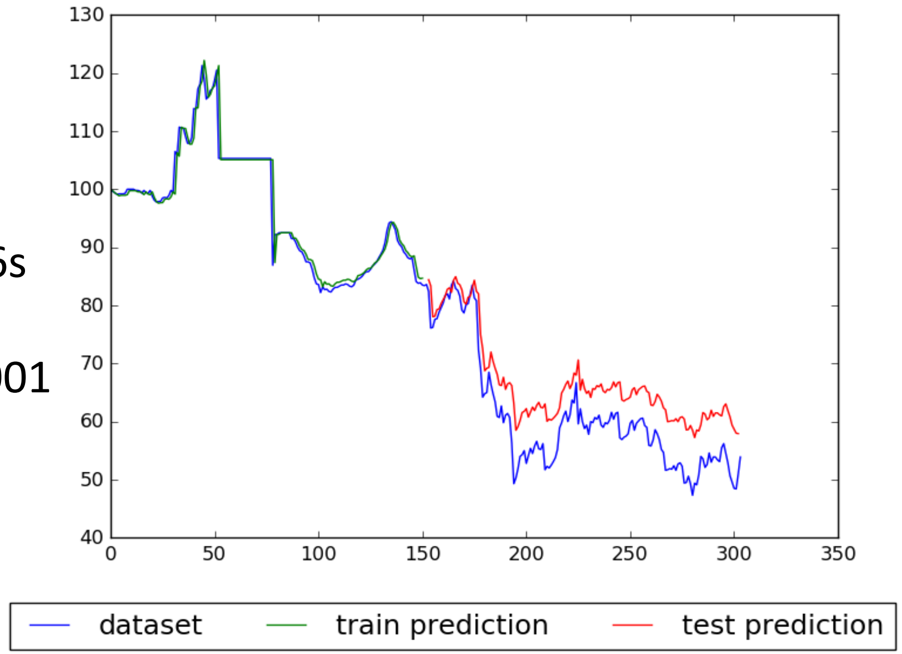
train-time = 62.031s
test-RMSE = 2.201
learning_rate=0.001

Note: No: of epochs is kept constant at 100

train-time = 64.890s
test-RMSE = 26.596
learning_rate=0.01



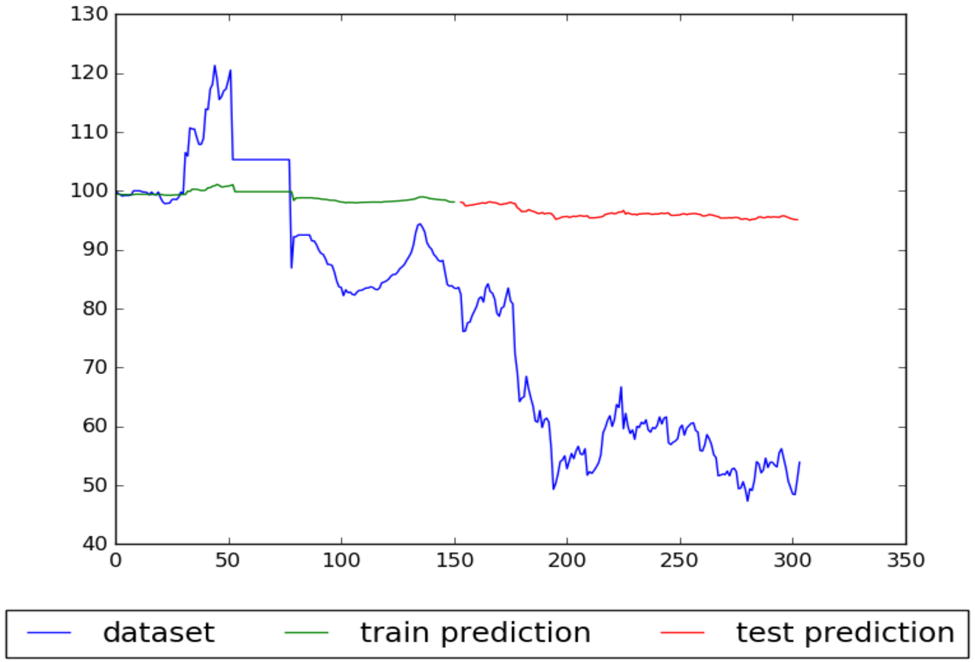
train-time = 63.186s
test-RMSE = 6.652
learning_rate=0.0001



Momentum

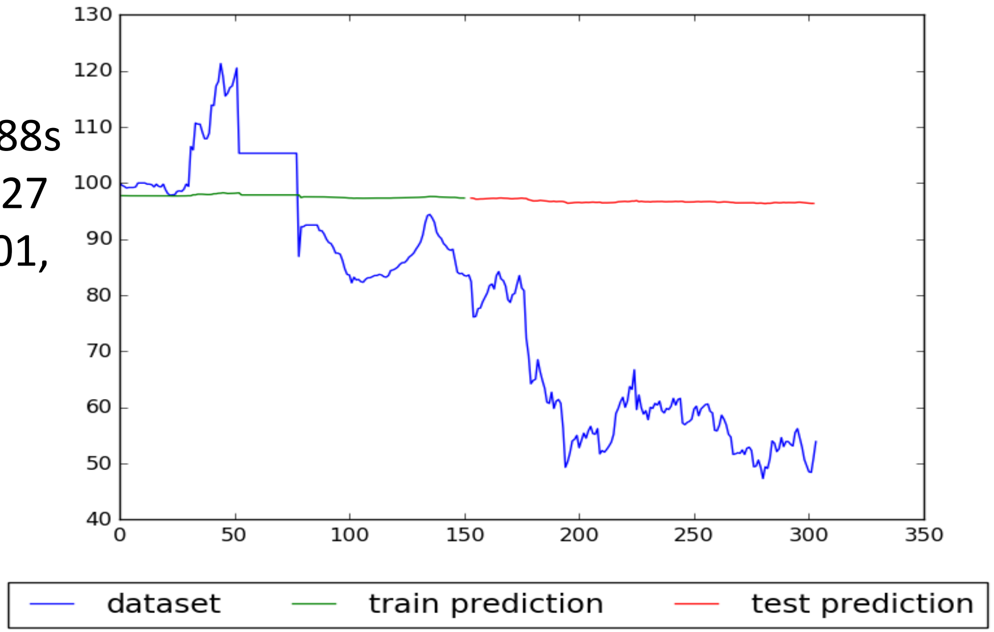
Momentum with SGD optimiser with
n_epochs = 100, training_percent=0.5

n_layers=4, layer_dimension=[1, 60, 60, 1]

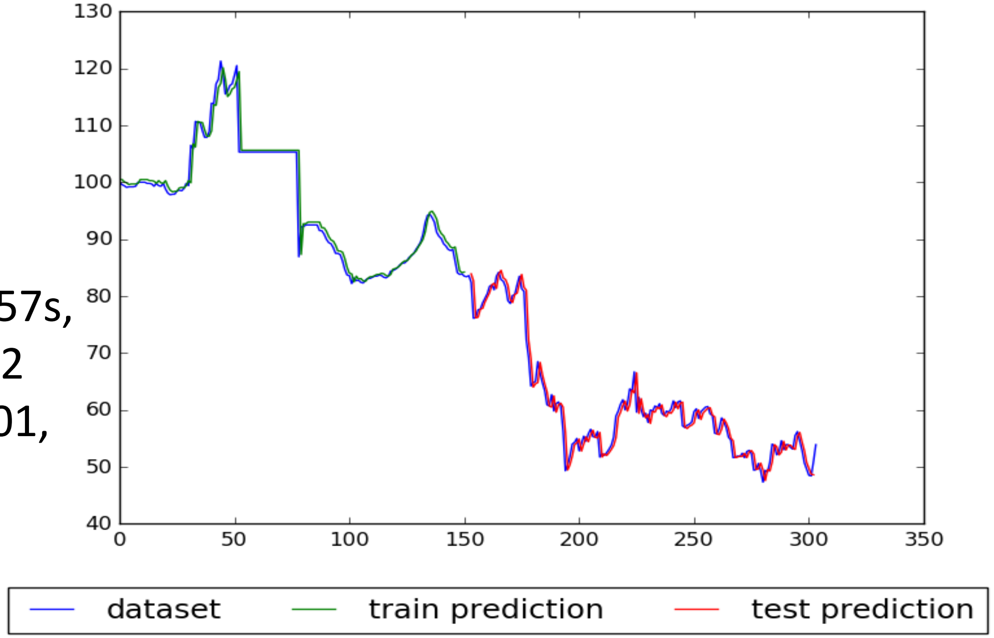


train-time = 52.038s
test-RMSE = 37.291
learning_rate=0.01, momentum=0.1

train-time = 52.988s
test-RMSE = 36.627
learning_rate=0.01,
momentum=0.6

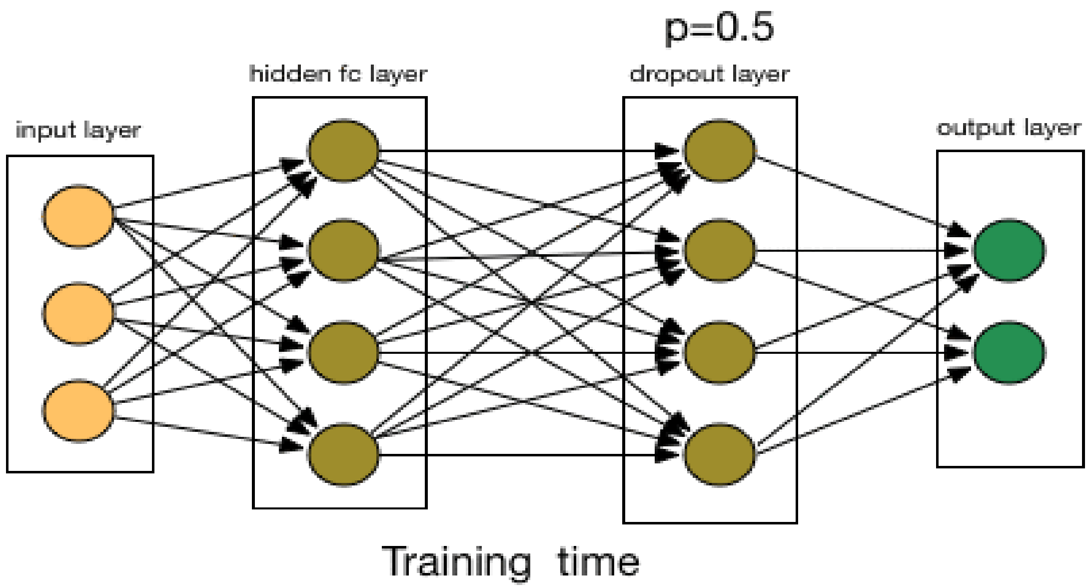


train-time = 52.557s,
test-RMSE = 1.932
learning_rate=0.01,
momentum=0.9

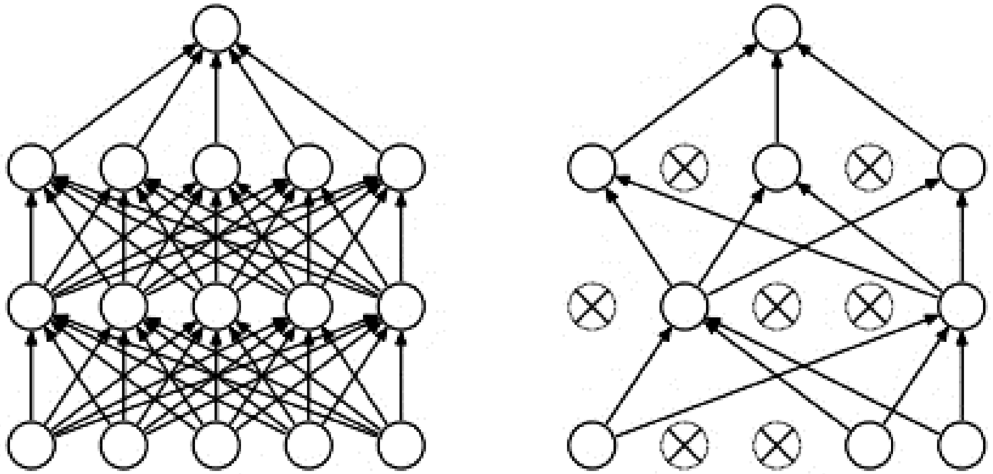


Dropout – a strategy for countering overfitting

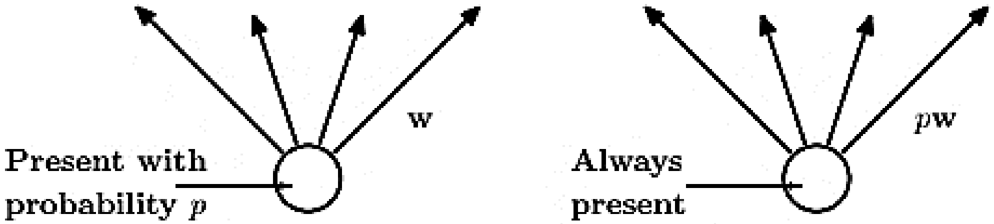
- Reduces overfitting in deep neural networks.
- Randomly drop computation units or nodes along with their connections from the network during training.
- This is equivalent to training a number of different “thinned” networks and then averaging their prediction to get the final output.
- In effect, this strategy gets the benefit of both bagging and boosting used in machine learning.



In the case of LSTM, dropout can be applied at two levels
1. Dropping inputs
2. Dropping recurrent connections


(a) Standard Neural Net (b) After applying dropout.

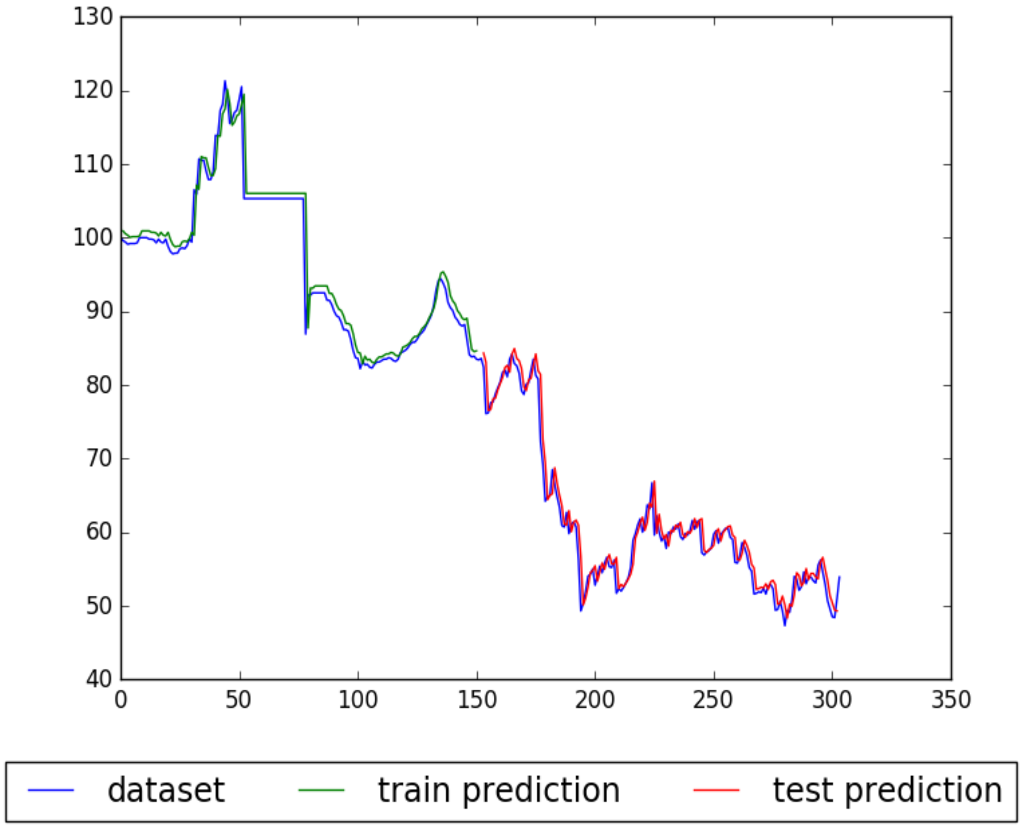
Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.



(a) At training time (b) At test time

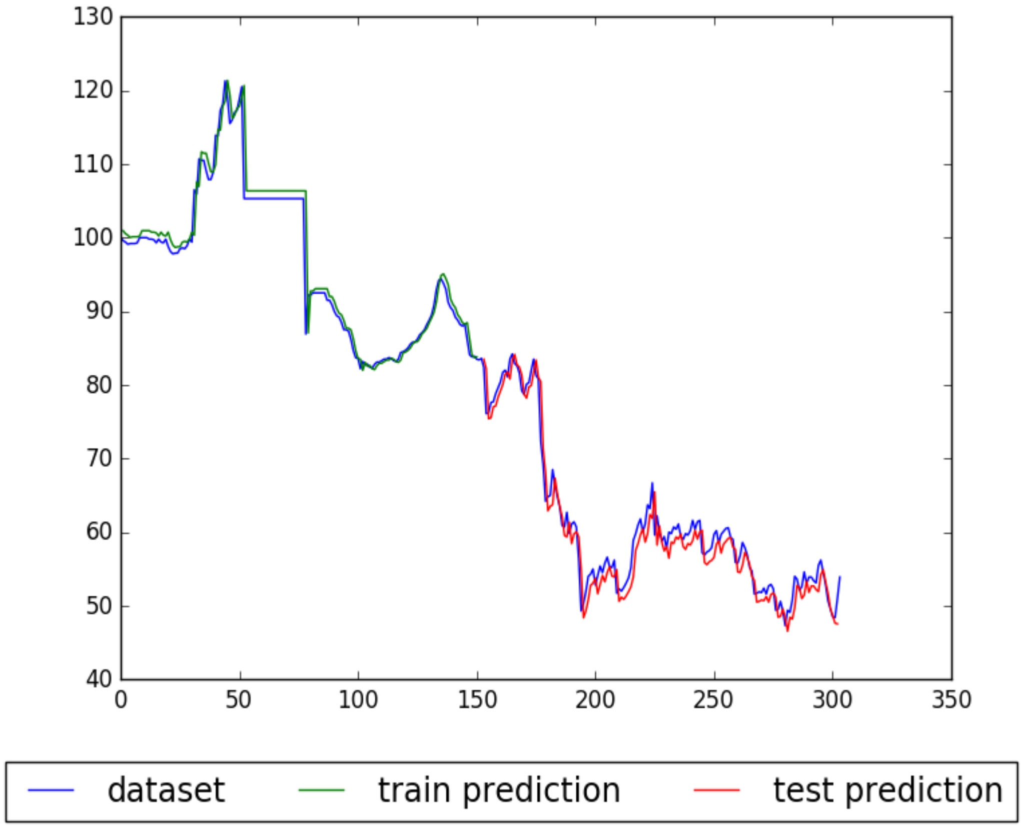
Left: A unit at training time that is present with probability p and is connected to units in the next layer with weights **w**. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

With dropout



Size = [1,60,60,1], train-time = 63.596s
test-RMSE = 2.071, Learning rate = 0.001
CV_error = 2.8 e-4
Dropout percent = 0.4, Optimiser = Adam

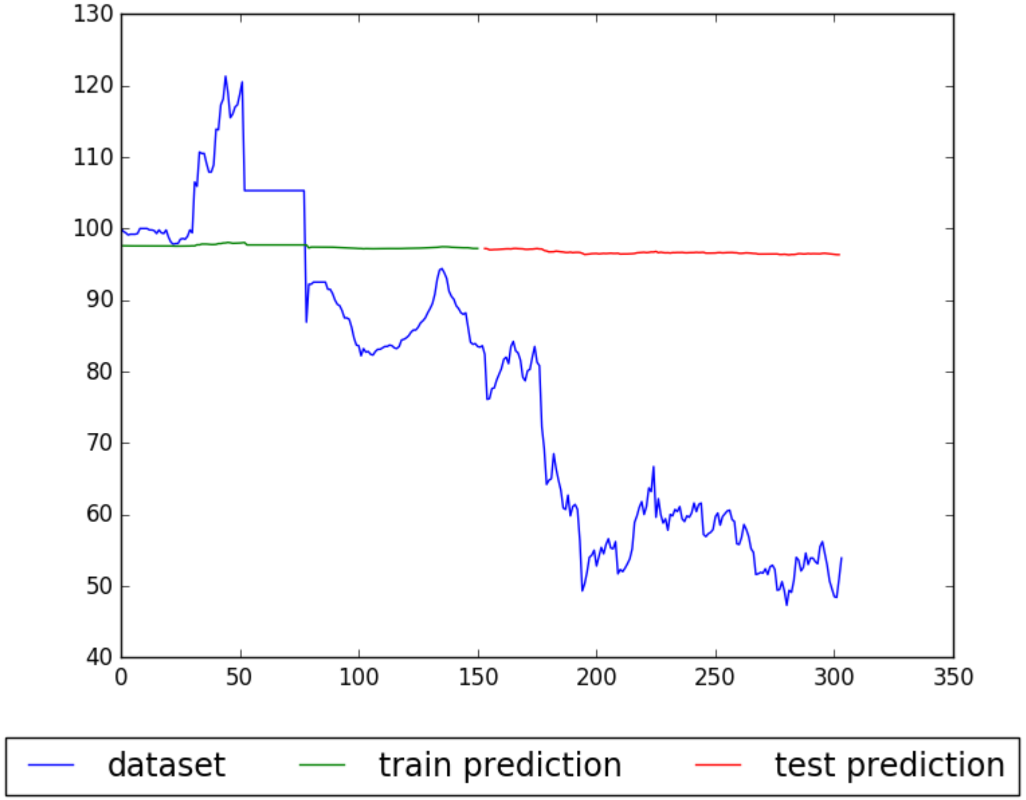
Without dropout



Size = [1,60,60,1], train-time = 60.804s
test-RMSE = 2.203, Learning rate=0.001
CV_error = 5.15 e-4
Optimiser = Adam

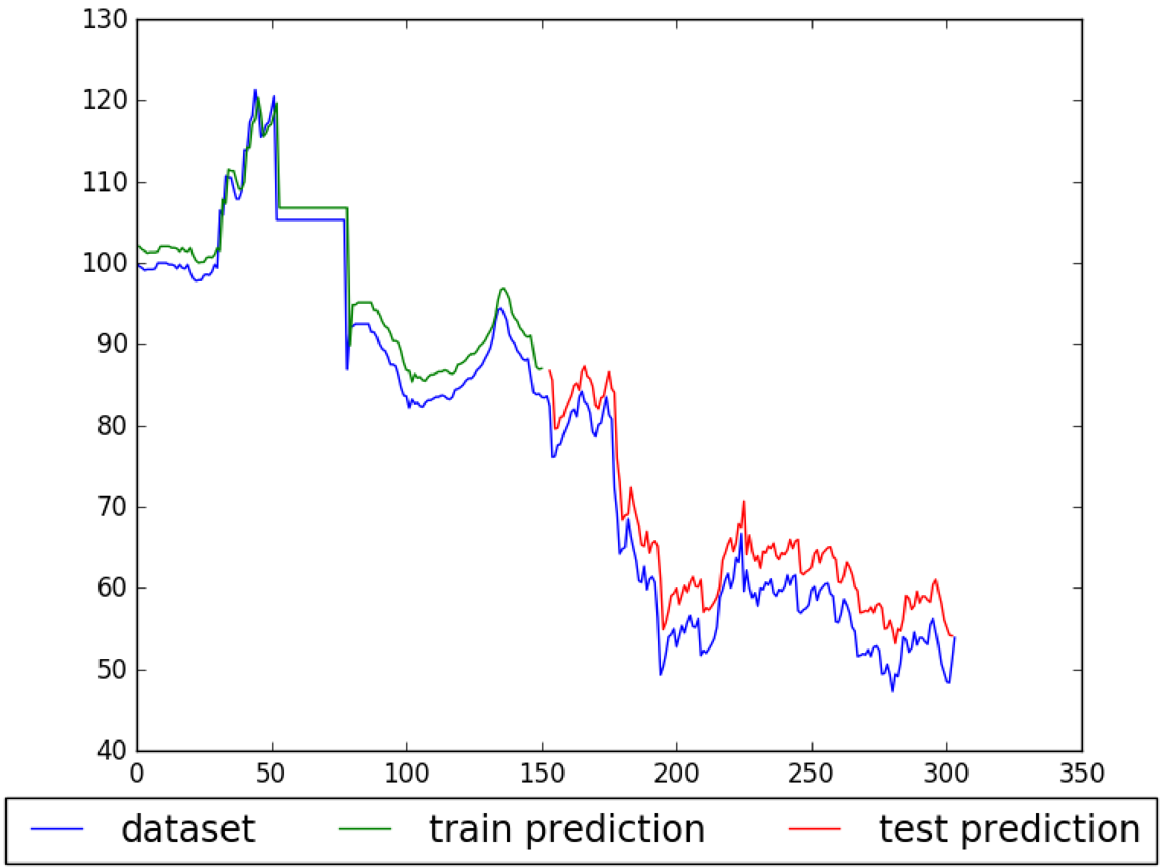
Is regularization always necessary ?

train-time = 51.674s, Optimiser = SGD
test-RMSE = 1.980
Learning Rate = 0.1, n_epochs = 100
Momentum = 0



train-time = 52.511s, Optimiser = SGD
test-RMSE = 37.243
Learning = 0.01, n_epochs = 100
Momentum = 0

Increasing number of epochs



train-time = 232.706s
train-RMSE = 7.382
test-RMSE = 26.015
optimizer=SGD
Learning = 0.01,
n_epochs = 700
Momentum = 0

Result and recommendations

- For the given dataset, assuming that future data points will be from the same data distribution, the optimal hyperparameters for the LSTM Deep Learning Neural Network are

1. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = Adam, Drop_out_recurrent = 0.4, Learning rate = 0.001, Momentum = 0. Test-RMSE = 2.071
2. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = SGD, Drop_out_recurrent = 0, Learning rate = 0.1, Momentum = 0. Test-RMSE = 1.980
3. No: of layers = 4, Layer dimensions = [1,60,60,1], Optimser = SGD, Drop_out_recurrent = 0, Learning rate = 0.01, Momentum = 0.9. Test-RMSE = 1.983

- Increasing the number of epochs, while computationally costly, will provide the algorithm more iterations over the data, enabling it to converge to the right weights even for a plain vanilla optimizer.

Deliverables

- A GUI is proved to enable experimentation with the hyperparameters.
- A report and readme describing the materials, methodology and results.

References

- G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," CoRR, vol. abs/1207.0580, 2012
- https://datamarket.com/data/set/22tb/exchange-rate-twi-may-1970-aug-1995#!ds=22tb&display=line
- Greff, Klaus, Srivastava, Rupesh Kumar, Koutn´ık, Jan, Steunebrink, Bas R, and Schmidhuber, Jurgen. Lstm: A search space odyssey. arXiv preprint arXiv:1503.04069, 2015.
- http://cs229.stanford.edu/proj2015/054_report.pdf
- Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.
- Schaul, Tom, Sixin Zhang, and Yann LeCun. "No more pesky learning rates." ICML (3) 28 (2013): 343-351.