Pleadml-Kera : CIFAR-10 Long Short Term Memory

MatConvNet

< Back to Alex Krizhevsky's home page

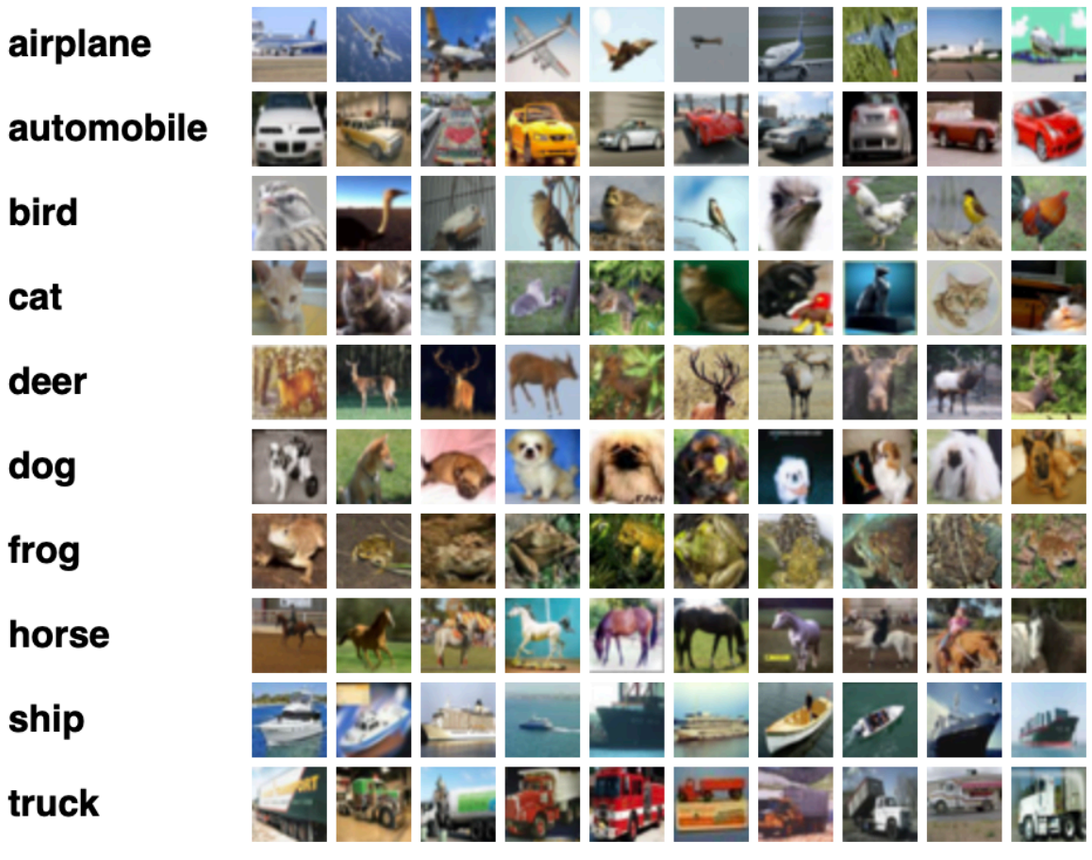
The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Learning Multiple Layers of Features from Tiny Images

Alex Krizhevsky

April 8, 2009

Train a simple deep CNN on the CIFAR10 small images dataset.

It gets to 75% validation accuracy in 25 epochs, and 79% after 50 epochs. (it's still underfitting at that point, though).

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import os

batch_size = 32
num_classes = 10
epochs = 100
data_augmentation = True
num_predictions = 20
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'
```

cifar10 [~/Desktop/py_code_2019/cifar10] - ~/Library/Preferences/PyCharmCE2019.1/scratches/scratch.py

Scratches 〉 scratch.py

scratch

Project

scratch.py

cifar10 ~/Desktop/py_code_2019/cifar1(
- venv
  - bin
  - lib
  - share
  - pyvenv.cfg
- External Libraries
- Scratches and Consoles

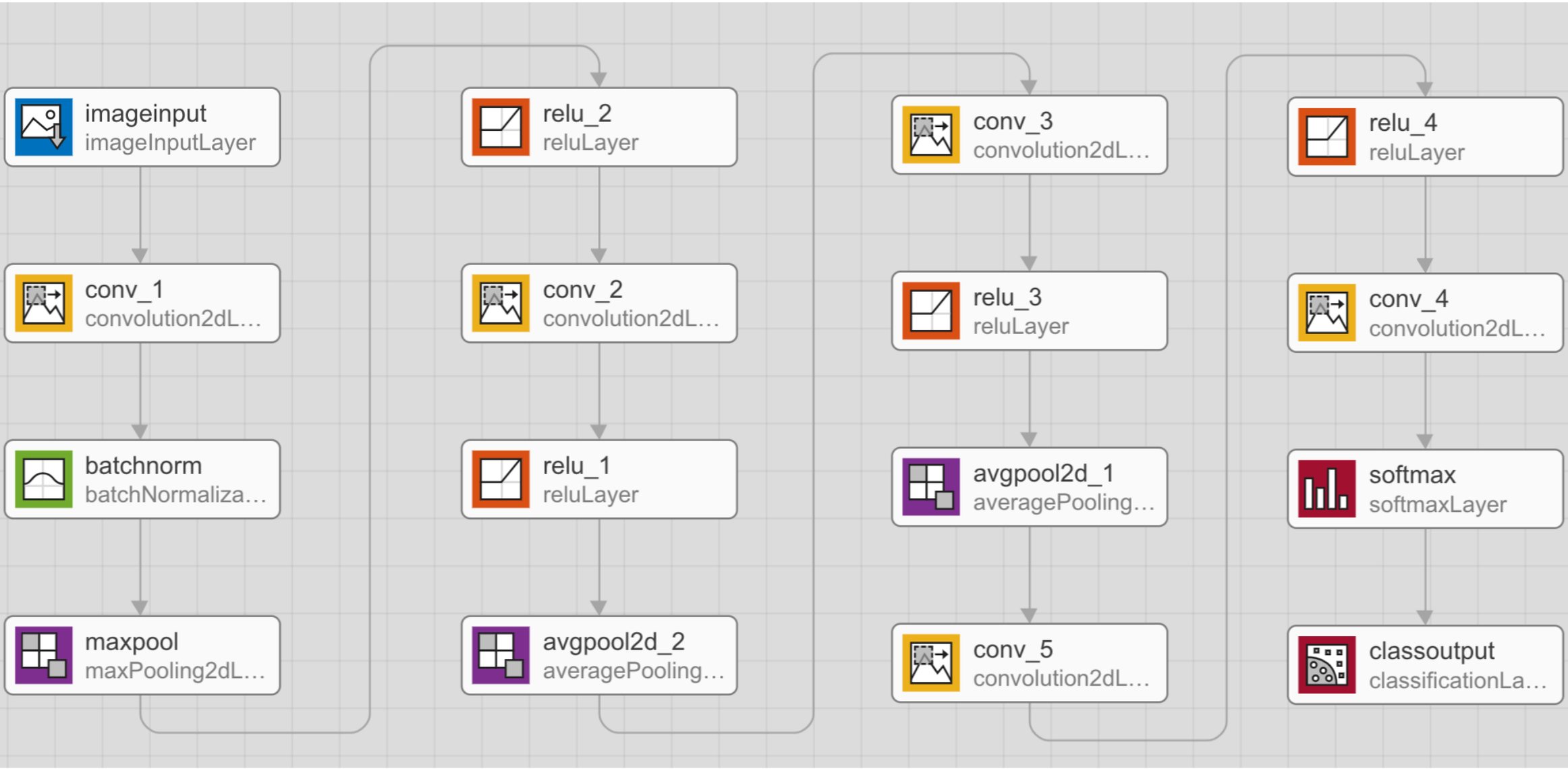
```
    # Fit the model on the batches generated by datagen.flow().
    model.fit_generator(datagen.flow(x_train, y_train,
                                     batch_size=batch_size),
                        epochs=epochs,
                        validation_data=(x_test, y_test),
                        workers=4)

# Save model and weights
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)

# Score trained model.
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

Run: scratch

```
1475/1563 [============================>..] - ETA: 5s - loss: 0.7945 - accuracy: 0.7325
1476/1563 [============================>..] - ETA: 5s - loss: 0.7947 - accuracy: 0.7324
1477/1563 [============================>..] - ETA: 5s - loss: 0.7948 - accuracy: 0.7324
1478/1563 [============================>..] - ETA: 5s - loss: 0.7952 - accuracy: 0.7323
1479/1563 [============================>..] - ETA: 5s - loss: 0.7951 - accuracy: 0.7323
1480/1563 [============================>..] - ETA: 5s - loss: 0.7953 - accuracy: 0.7323
1481/1563 [============================>..] - ETA: 5s - loss: 0.7954 - accuracy: 0.7322
1482/1563 [============================>..] - ETA: 5s - loss: 0.7954 - accuracy: 0.7322
1483/1563 [============================>..] - ETA: 5s - loss: 0.7953 - accuracy: 0.7322
1484/1563 [============================>..] - ETA: 5s - loss: 0.7952 - accuracy: 0.7323
1485/1563 [============================>..] - ETA: 5s - loss: 0.7951 - accuracy: 0.7323
1486/1563 [============================>..] - ETA: 4s - loss: 0.7950 - accuracy: 0.7323
1487/1563 [============================>..] - ETA: 4s - loss: 0.7950 - accuracy: 0.7323
1488/1563 [============================>..] - ETA: 4s - loss: 0.7950 - accuracy: 0.7323
```

1: Project 7: Structure 2: Favorites

4: Run 6: TODO Terminal Python Console Event Log


imageinput
imageInputLayer
conv_1
convolution2dL…
batchnorm
batchNormaliza…
maxpool
maxPooling2dL…
relu_2
reluLayer
conv_2
convolution2dL…
relu_1
reluLayer
avgpool2d_2
averagePooling…
conv_3
convolution2dL…
relu_3
reluLayer
avgpool2d_1
averagePooling…
conv_5
convolution2dL…
relu_4
reluLayer
conv_4
convolution2dL…
softmax
softmaxLayer
classoutput
classificationLa…

MATLAB R2019a - academic use

HOME | PLOTS | APPS | EDITOR | PUBLISH | VIEW

Search Documentation

New | Open | Save | Find Files | Compare | Print — FILE
Go To | Find — NAVIGATE
Insert | Comment | Indent — EDIT
Breakpoints — BREAKPOINTS
Run | Run and Advance | Run Section | Advance | Run and Time — RUN

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ code2019 ▸ MatConvNet ▸ elastic-matconvnet-1.0-beta25 ▸ matconvnet-1.0-beta25 ▸ examples ▸ mnist

Current Folder

Name
- data
- cnn_mnist.fig
- cnn_mnist.m
- cnn_mnist_experiments.m
- cnn_mnist_init.m

cnn_mnist.fig (Figure)

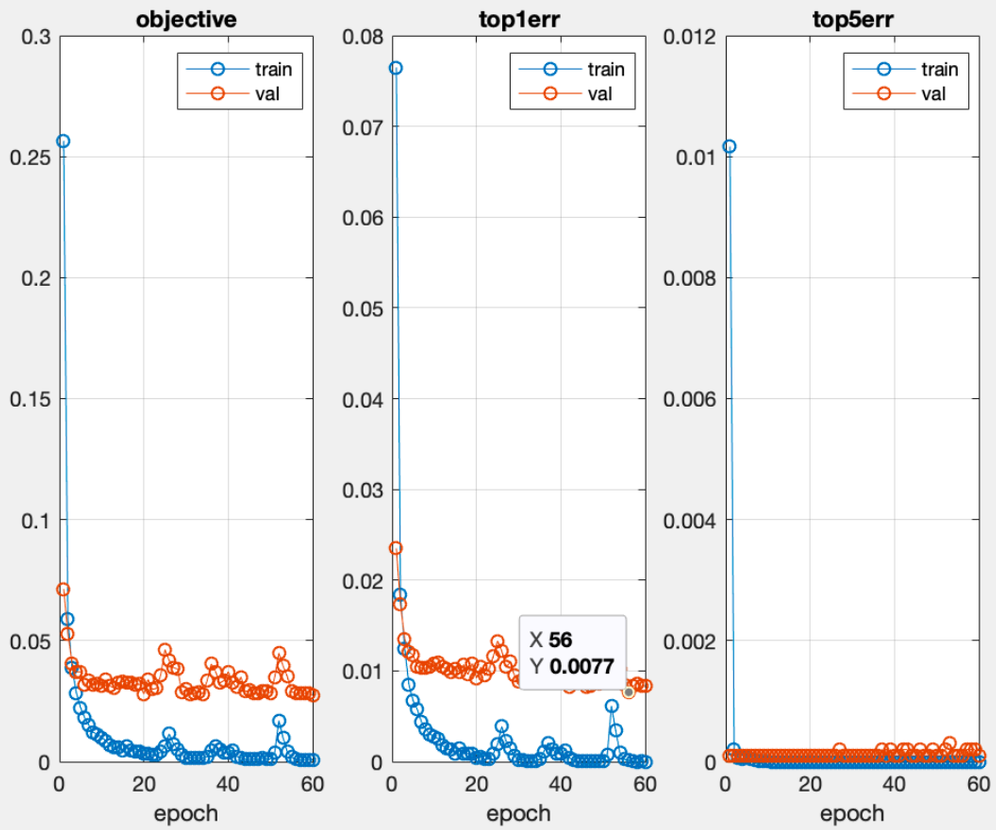
Editor - /Users/apple/Desktop/Jiann-Ming Wu/code2019/longShortTermMemory/lstm-matlab-master-2/matlab/lstm_trai...

+1 | lstm_train_TimeSeriesTask.m | my_seg_overlapping.m | updateLegendMenuToolbar.m | lstm_train_SinToyTask.m

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

Figure 2

File Edit View Insert Tools Desktop Window Help




```
s
 of charge, to any pe
ed documentation file
striction, including
erge, publish, distri
, and to permit perso
bject to the followin

s permission notice s
ons of the Software.
```

Comma

fx >>

Research Exercise:

Develop deep learning for classification based on MatConvNet

Develop iOS APP based on MatConvNet

Develop deep learning for regression based on MatConvNet

MATLAB R2019a - academic use

HOME | PLOTS | APPS | EDITOR | PUBLISH | VIEW

Search Documentation

New | Open | Save | Find Files | Compare | Print — FILE
Go To | Find — NAVIGATE
Insert | Comment | Indent — EDIT
Breakpoints — BREAKPOINTS
Run | Run and Advance | Run Section | Advance | Run and Time — RUN

/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ code2019 ▸ MatConvNet ▸ elastic-matconvnet-1.0-beta25 ▸ matconvnet-1.0-beta25 ▸ examples ▸ cifar ▸

Current Folder

Name

- data
- cnn_cifar.fig
- cnn_cifar.m
- cnn_cifar_init.m
- cnn_cifar_init_nin.m

cnn_cifar.fig (Figure)

Editor - /Users/apple/Desktop/Jiann-Ming Wu/code2019/longShortTermMemory/lstm-matlab-master-2/matlab/lstm_trai...

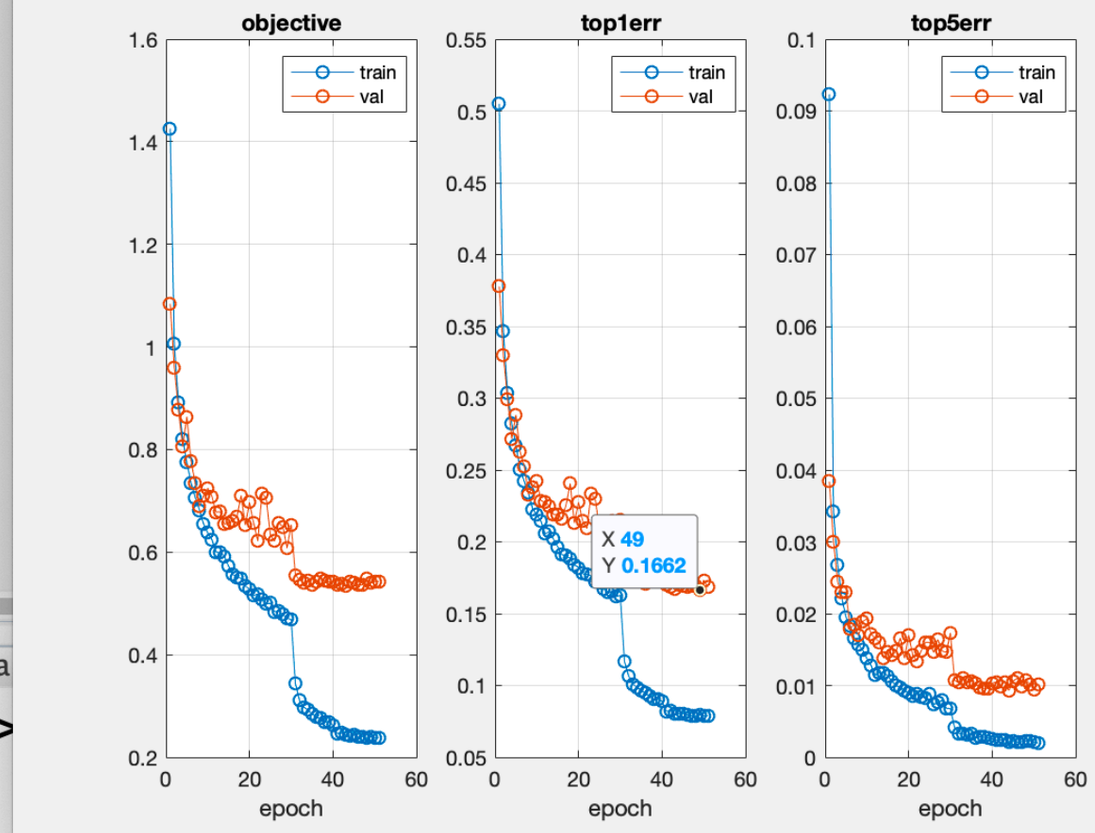
lstm_train_TimeSeriesTask.m | my_seg_overlapping.m | updateLegendMenuToolbar.m | lstm_train_SinToyTask.m

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
s
 of charge, to any pe
ed documentation file
striction, including
erge, publish, distri
, and to permit perso
bject to the followin

s permission notice s
ons of the Software.
```

Figure 1

File Edit View Insert Tools Desktop Window Help



Comma

fx >>

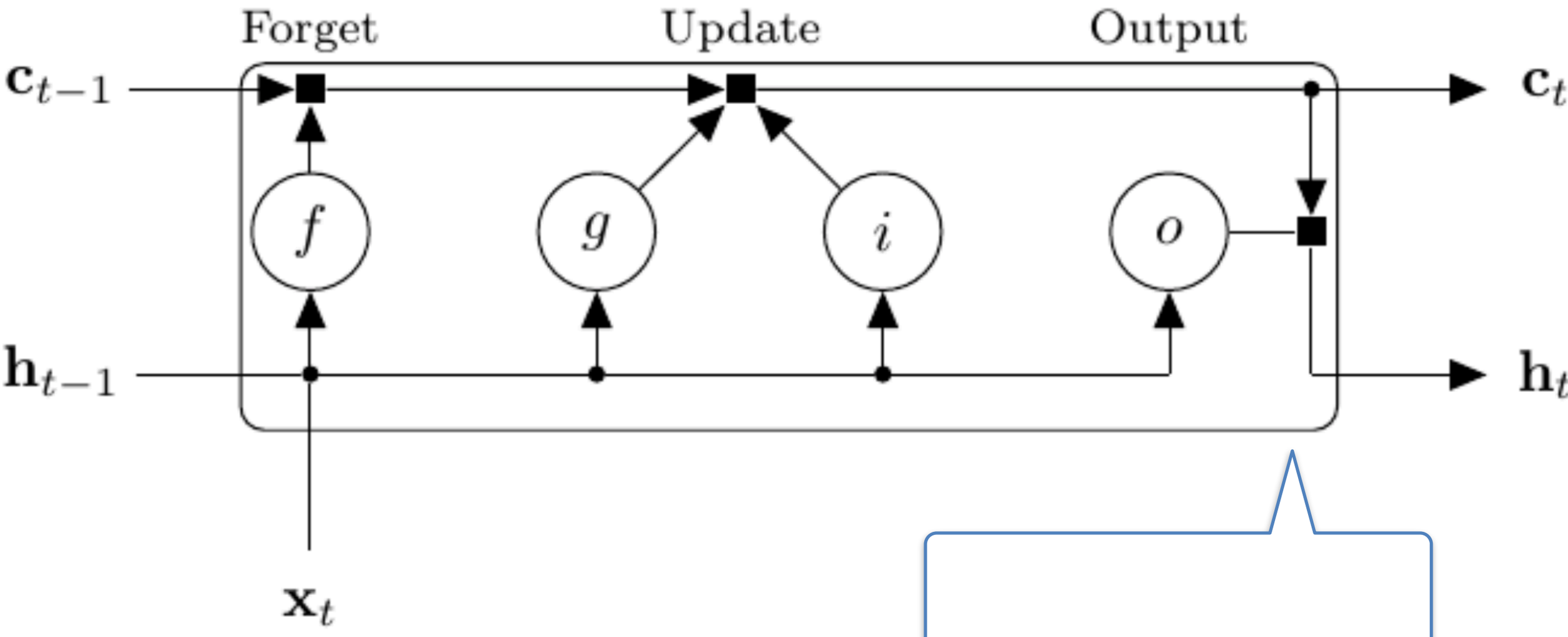
Ready

Component	Formula
Input gate	$i_t = \sigma_g(W_i \mathbf{x}_t + \mathrm{R}_i \mathbf{h}_{t-1} + b_i)$
Forget gate	$f_t = \sigma_g(W_f \mathbf{x}_t + \mathrm{R}_f \mathbf{h}_{t-1} + b_f)$
Cell candidate	$g_t = \sigma_c(W_g \mathbf{x}_t + \mathrm{R}_g \mathbf{h}_{t-1} + b_g)$
Output gate	$o_t = \sigma_g(W_o \mathbf{x}_t + \mathrm{R}_o \mathbf{h}_{t-1} + b_o)$

Sigmoid

σ_c is tanh

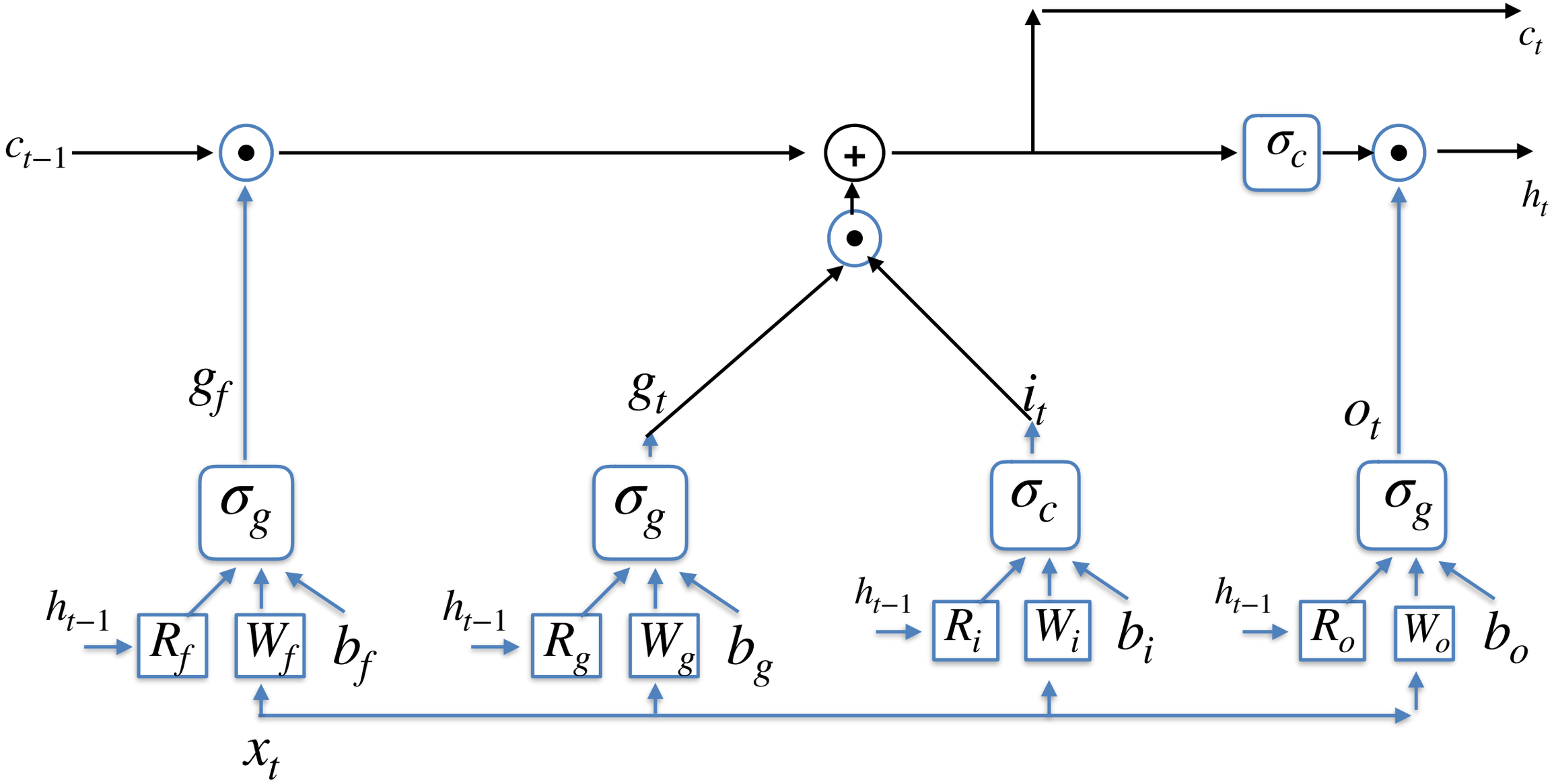
$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot g_t,$$

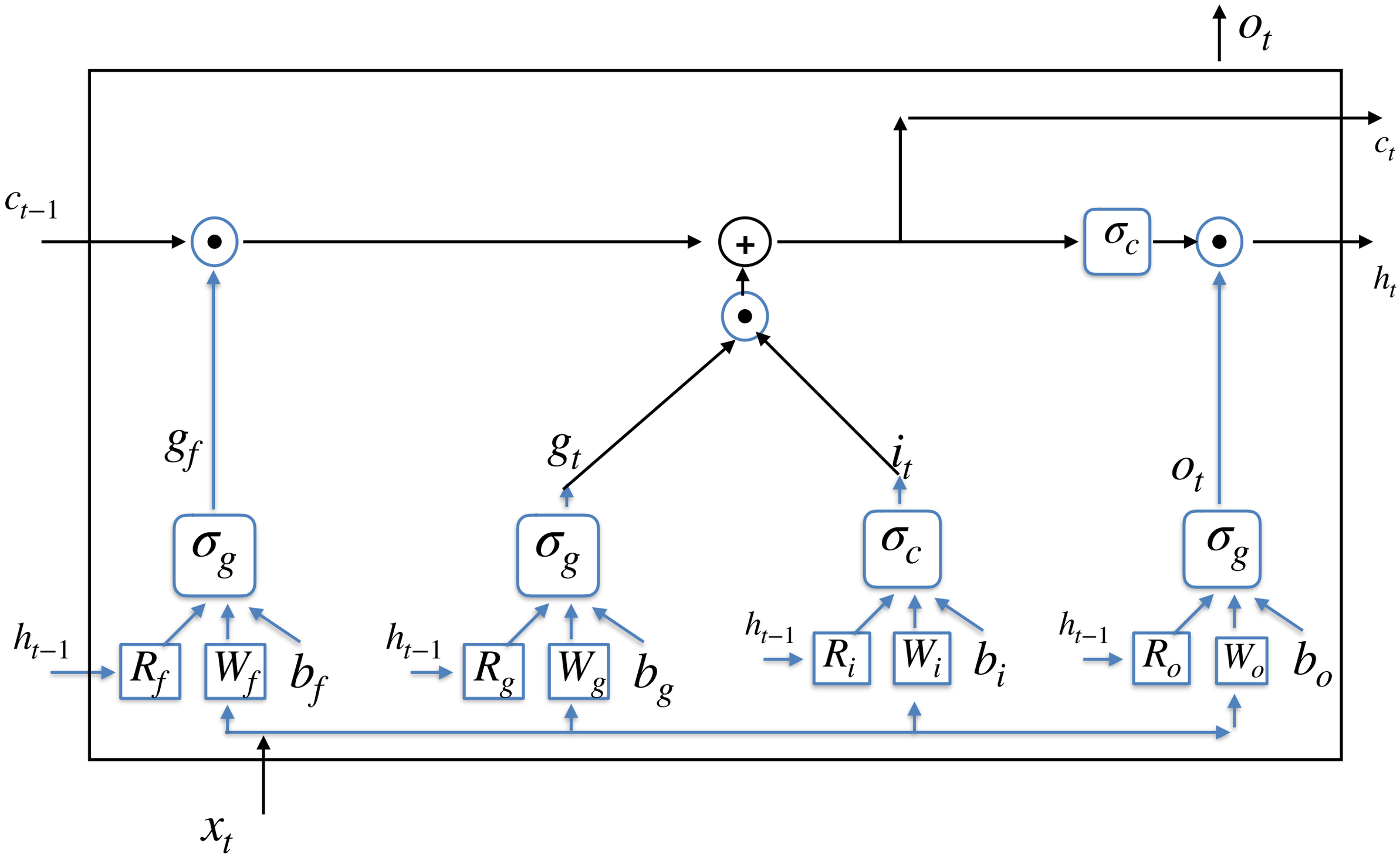


$$\mathbf{h}_t = o_t \odot \sigma_c(\mathbf{c}_t),$$

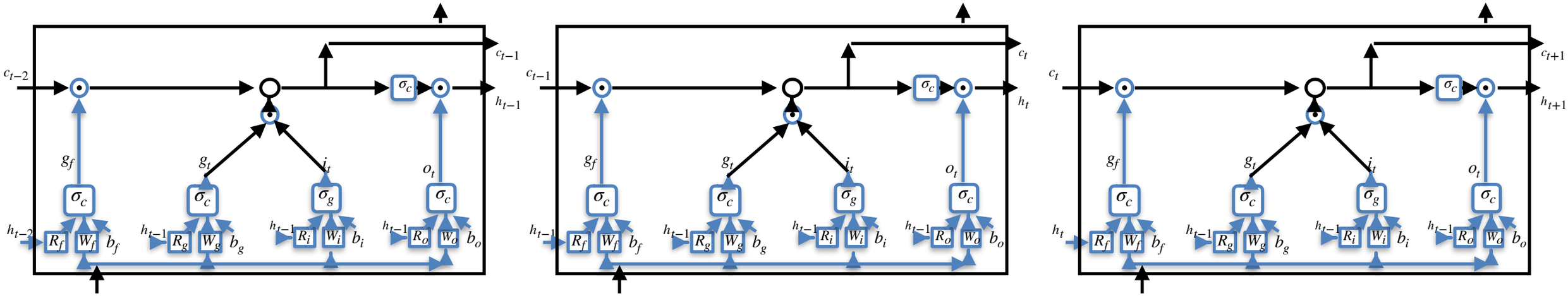
⊙ denotes the Hadamard product

element-wise multiplication of vectors


c_{t-1}
c_t
h_t
g_f
g_t
i_t
o_t
σ_g
σ_g
σ_c
σ_c
σ_g
h_{t-1}
R_f
W_f
b_f
h_{t-1}
R_g
W_g
b_g
h_{t-1}
R_i
W_i
b_i
h_{t-1}
R_o
W_o
b_o
x_t


o_t
c_t
c_{t-1}
σ_c
h_t
g_f
g_t
i_t
o_t
σ_g
σ_g
σ_c
σ_g
h_{t-1}
R_f
W_f
b_f
h_{t-1}
R_g
W_g
b_g
h_{t-1}
R_i
W_i
b_i
h_{t-1}
R_o
W_o
b_o
x_t

o_{t-1} o_t o_{t+1}



x_{t-1} x_t x_{t+1}

1

...19

LSTM-MATLAB is Long Short-term Memory (LSTM) in MATLAB, which is meant to be succinct, illustrative and for research purpose only. It is accompanied with a paper for reference: Revisit Long Short-Term Memory: An Optimization Perspective, NIPS deep learning workshop, 2014.

7 commits | **1** branch | **0** packages | **0** releases | **1** contributor

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

huashiyiqike Update README.md — Latest commit 4969467 on 28 Dec 2015

data	first commit.	5 years ago
dependence	first commit.	5 years ago
Main.m	first commit.	5 years ago
README.md	Update README.md	4 years ago
aStart.m	first commit.	5 years ago
batch_cell_lstm.m	first commit.	5 years ago
batch_equal_nomask_lstm.m	first commit.	5 years ago
clientLoadDataMinibatchNomask_ref.m	first commit.	5 years ago
gputype.m	first commit.	5 years ago
netInit.m	first commit.	5 years ago
runClient.m	first commit.	5 years ago
server_batch_cell_lstm.m	first commit.	5 years ago

Example script to generate text from Nietzsche's writings.

At least 20 epochs are required before the generated text starts sounding coherent.

It is recommended to run this script on GPU, as recurrent networks are quite computationally intensive.

If you try this script on new data, make sure your corpus has at least ~100k characters. ~1M is better.

```
from __future__ import print_function
from keras.callbacks import LambdaCallback
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.optimizers import RMSprop
from keras.utils.data_utils import get_file
import numpy as np
import random
import sys
import io

path = get_file(
    'nietzsche.txt',
    origin='https://s3.amazonaws.com/text-datasets/nietzsche.txt')
with io.open(path, encoding='utf-8') as f:
    text = f.read().lower()
print('corpus length:', len(text))
```

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

keras-team / keras

Used by 38k | Watch 2.1k | Star 45.9k | Fork 17.4k

Code | Issues 2,694 | Pull requests 40 | Actions | Projects 1 | Wiki | Security | Insights

Branch: master ▾ keras / examples / Create new file | Upload files | Find file | History

xemcerk and gabrieldemarmiesse Fix too many values to unpack error (#13511) ... ✓ Latest commit 7a39b6c on 6 Nov

..		
README.md	Add missing examples to examples/README.md (#10637)	last year
addition_rnn.py	Displayed some examples in the docs. (#11758)	11 months ago
antirectifier.py	Added MarkDown formatting to examples/antirectifier.py (#12294)	10 months ago
babi_memnn.py	Added MarkDown formatting support to examples/babi_memnn.py (#12221)	10 months ago
babi_rnn.py	Update babi_rnn.py (#13263)	3 months ago
cifar10_cnn.py	Update examples	3 months ago

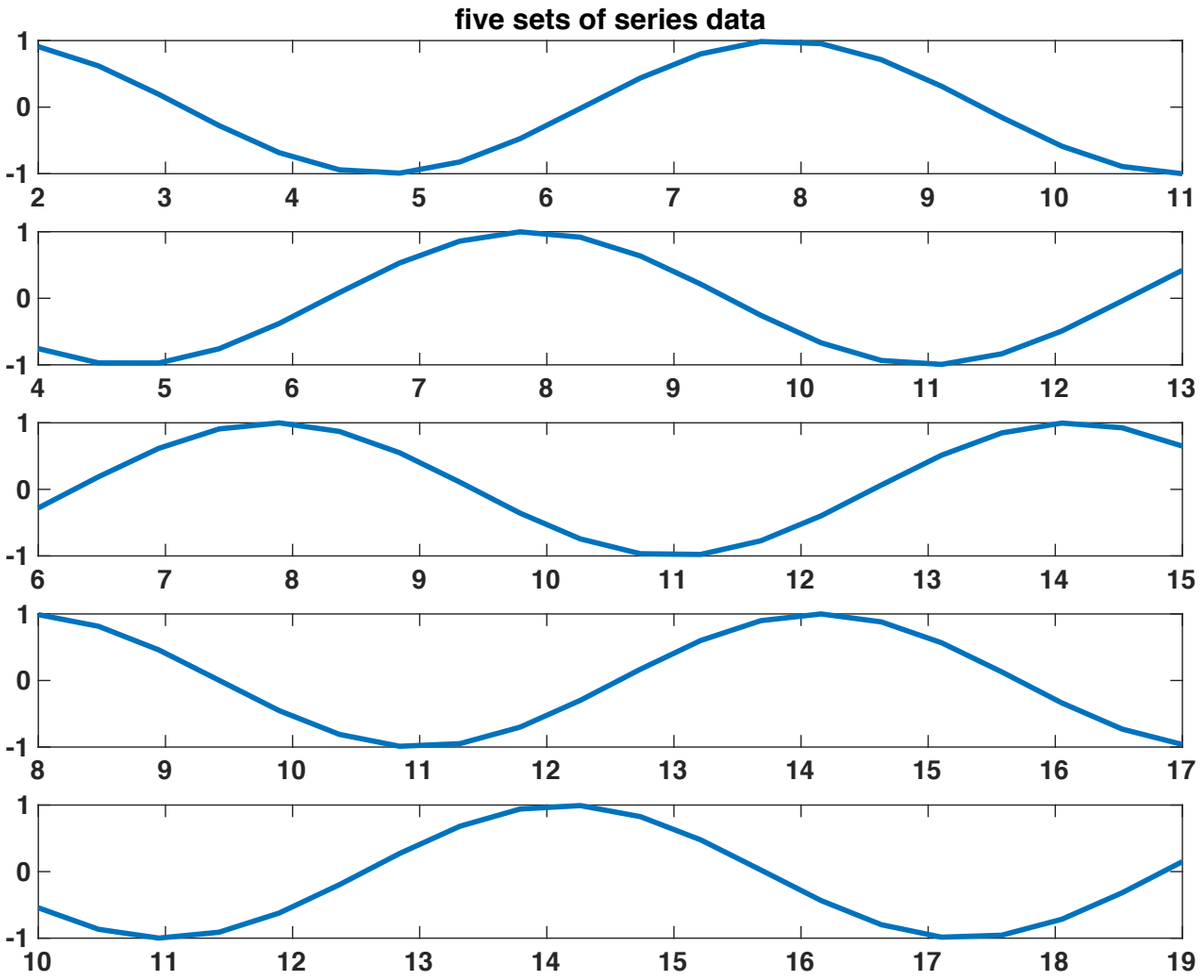
No description, website, or topics provided.

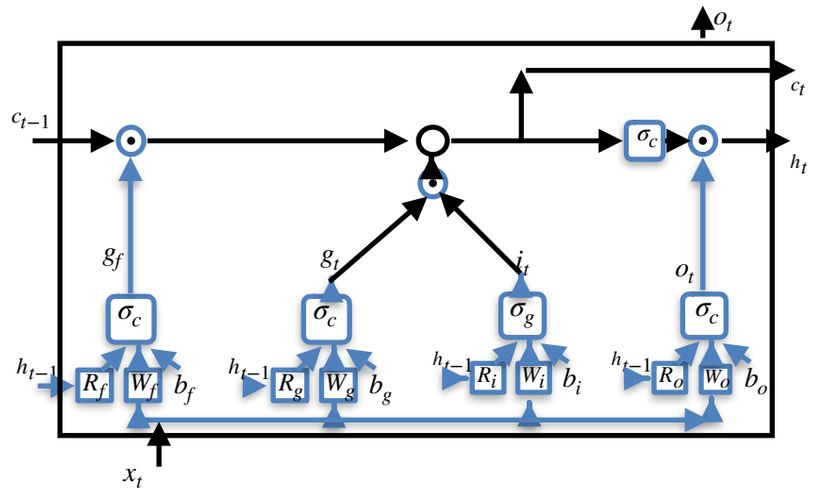
7 commits | 1 branch | 0 packages | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

BlendingInfinite Merge branch 'master' of https://github.com/MoritzN89/lstm-matlab — Latest commit 712dea2 on 21 Feb 2018

images	Initial commit.	2 years ago
matlab	Initial commit.	2 years ago
LSTMGradientsDerivations.pdf	Minor Update	2 years ago
README.md	Update README.md	2 years ago

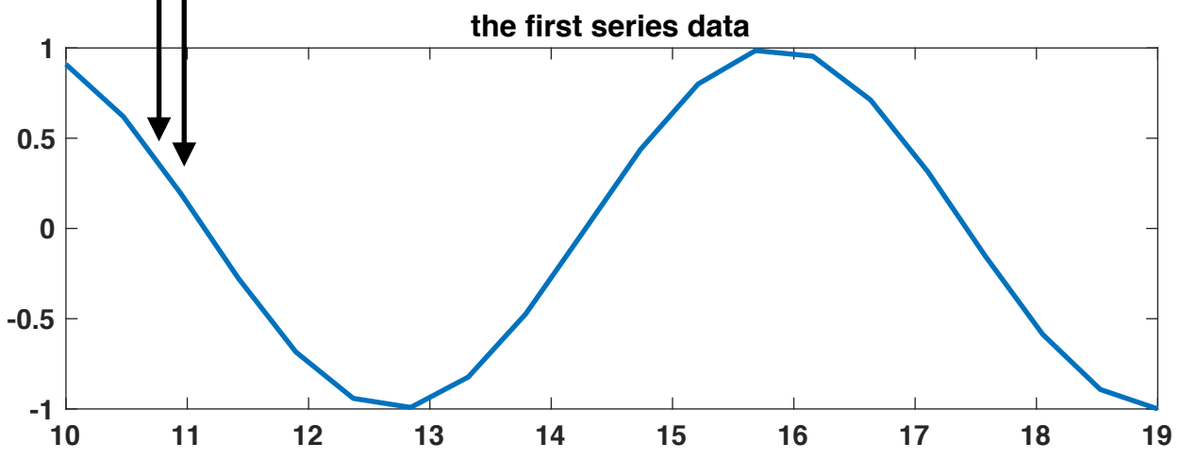




x_i = data(i)

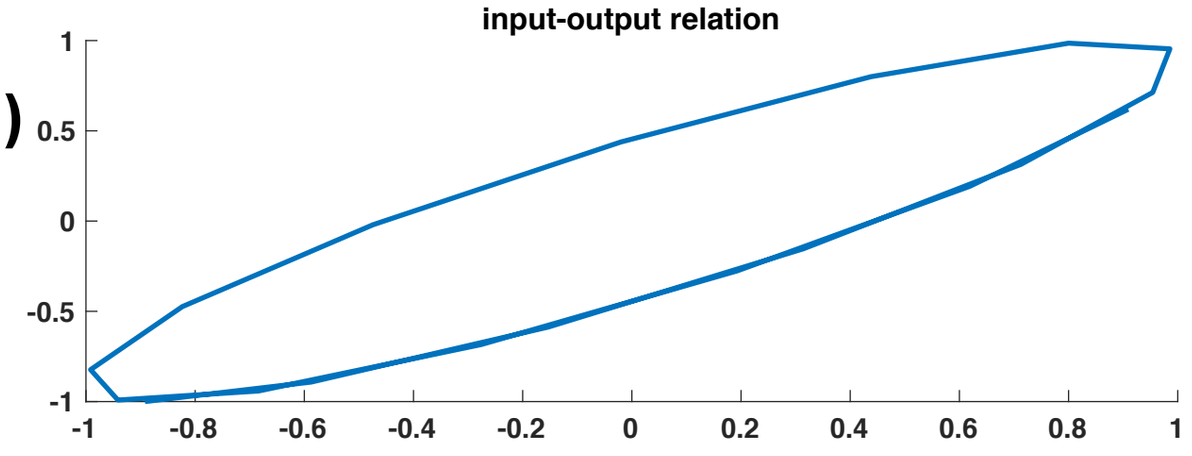
y_i = data(i+1)

data(1:20)



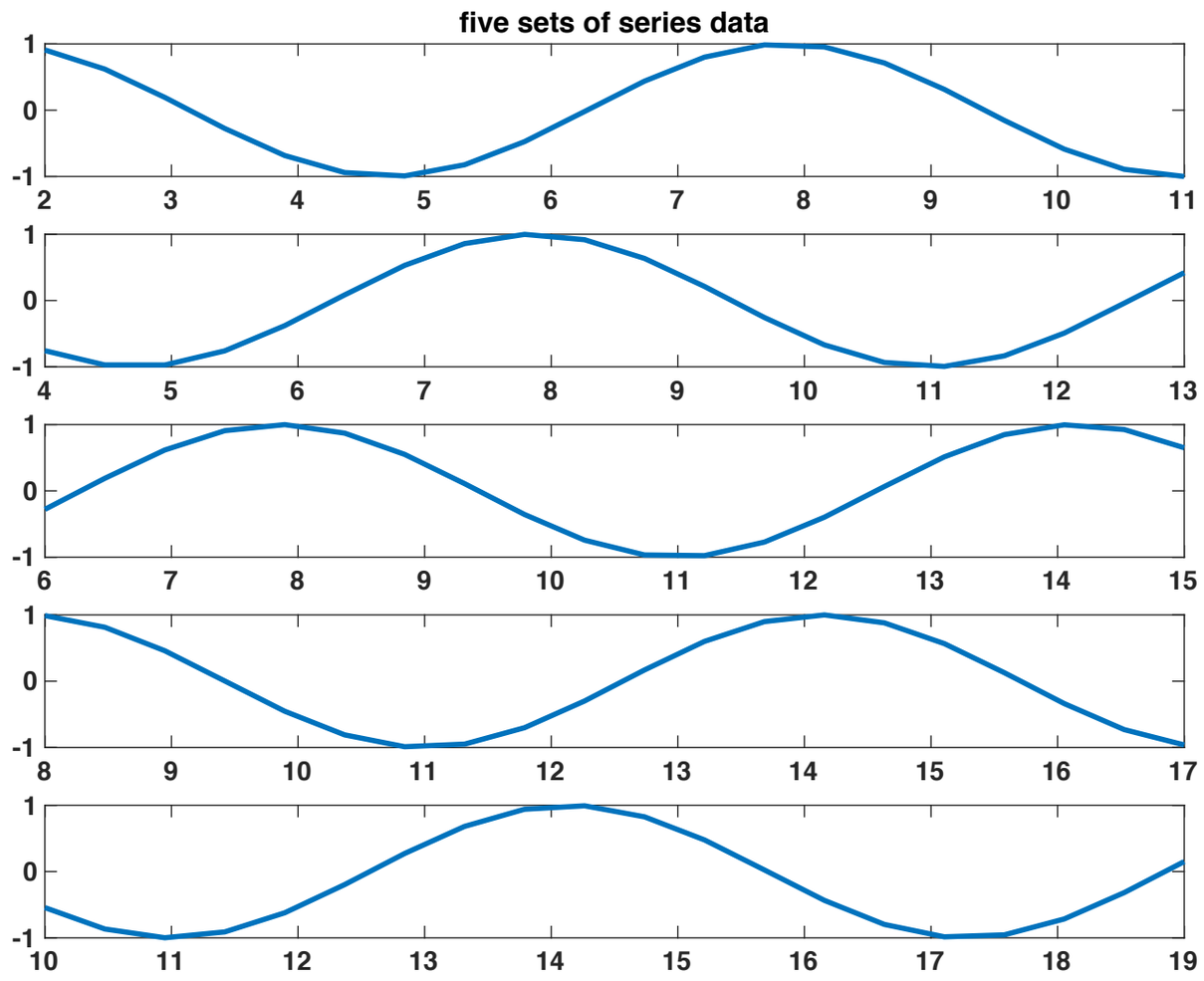
Set Valued Mapping

y_i = data(i+1)

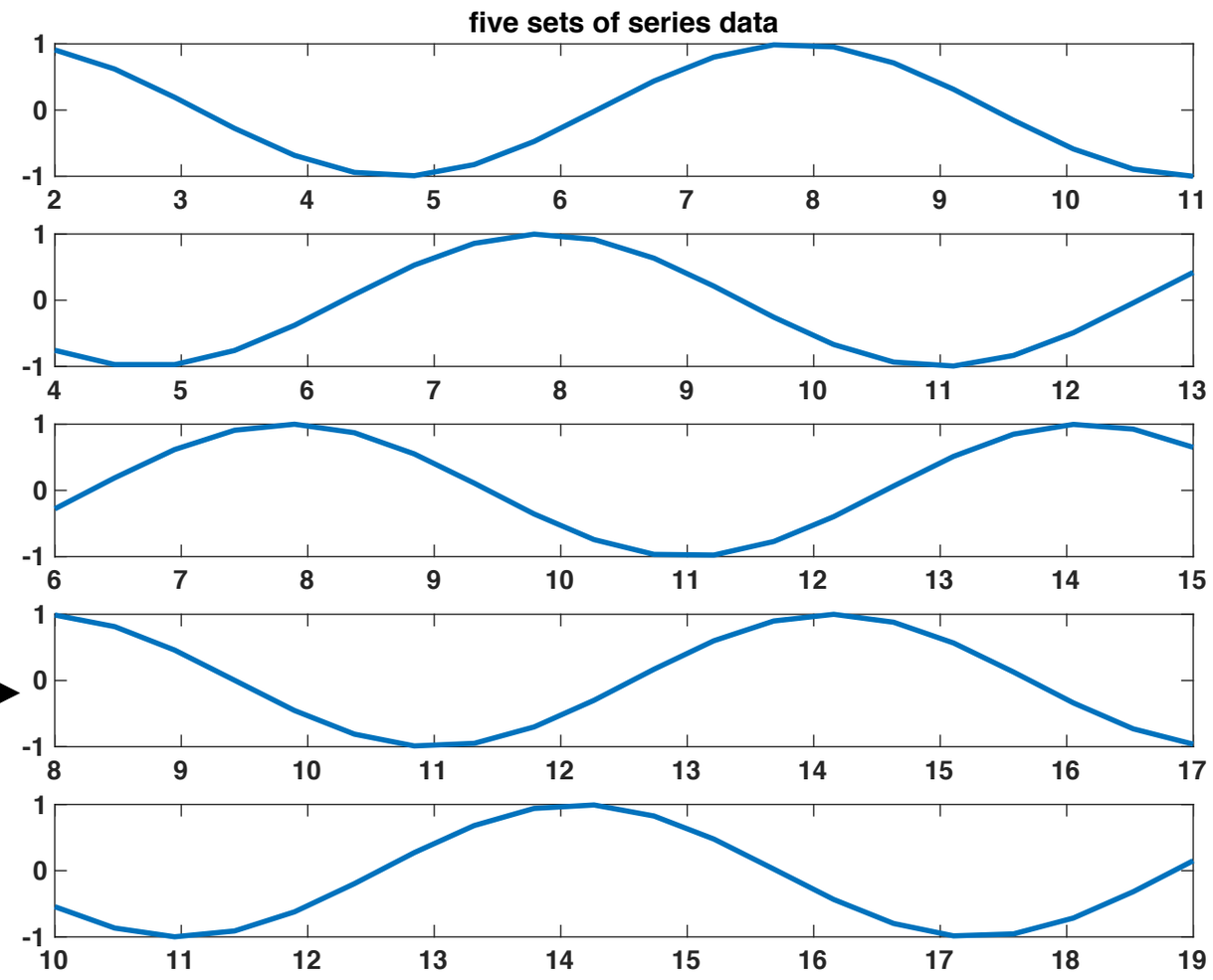


x_t = data(i)

Cross validation



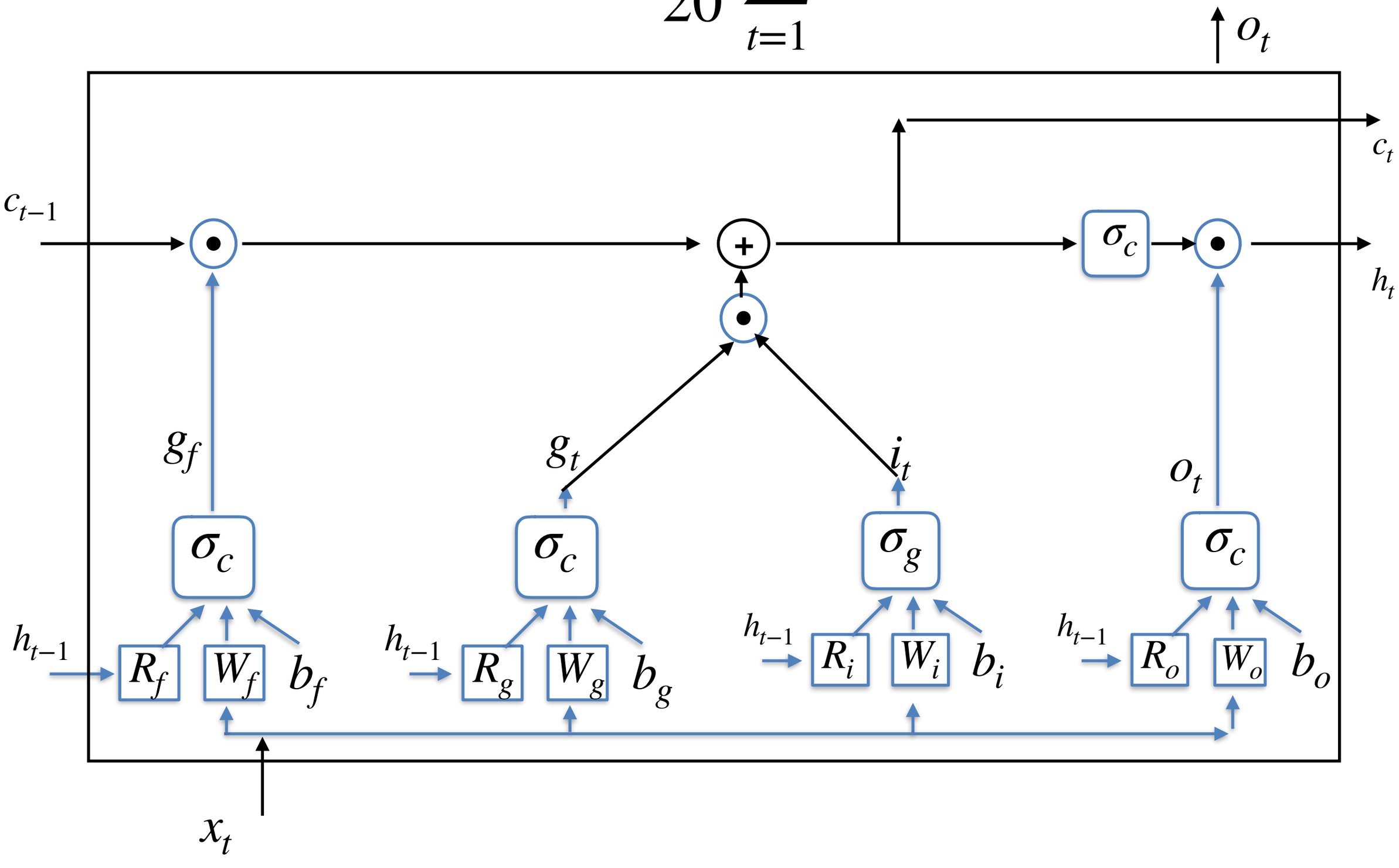
Cross validation



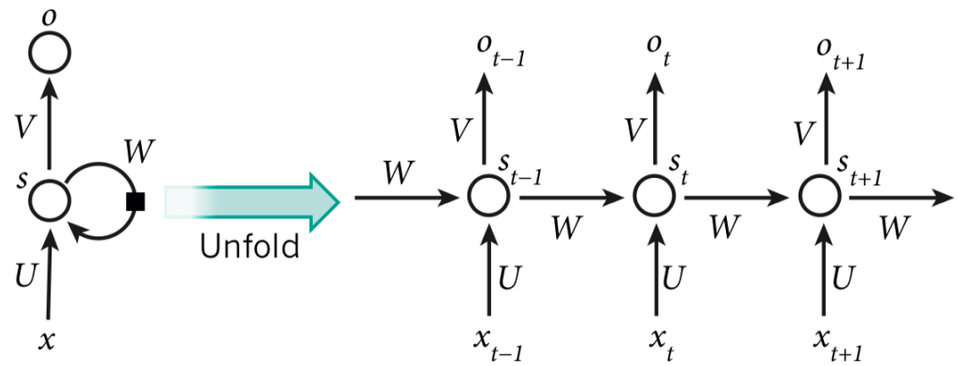
-

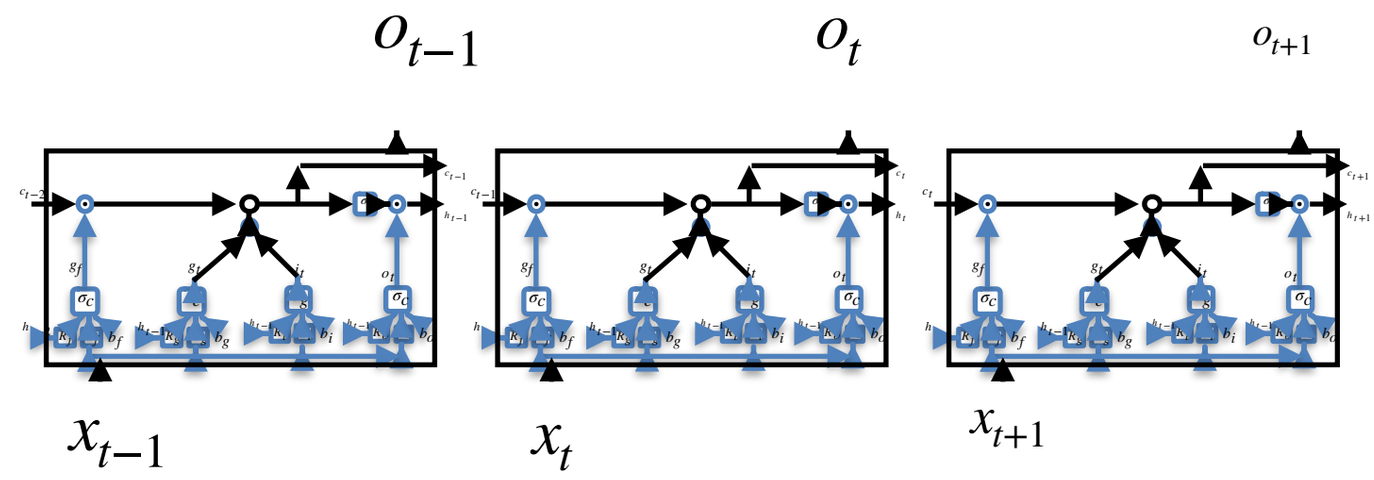
Testing error

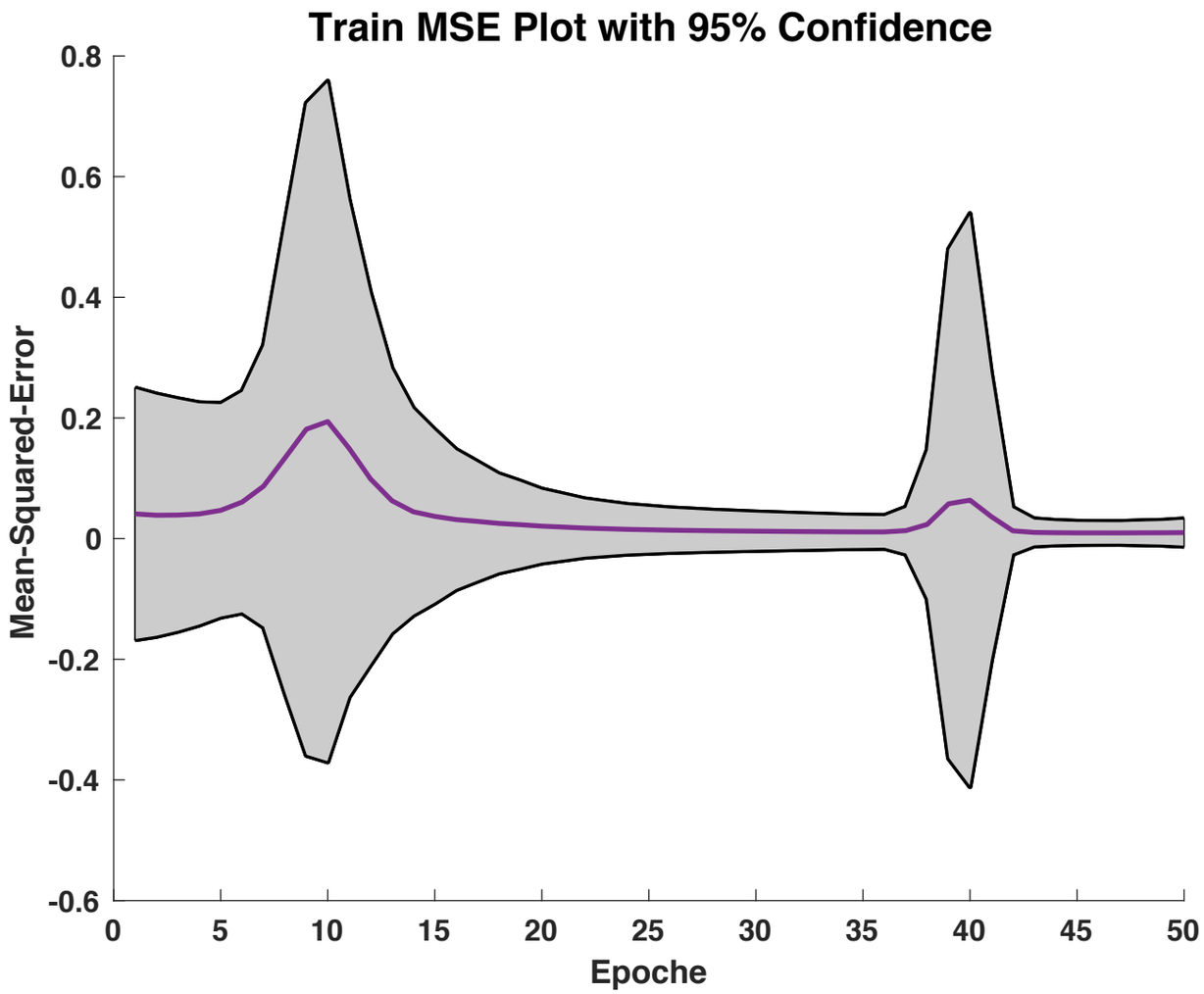
$$mse = \frac{1}{20}\sum_{t=1}^{20}(o_t - y_t)^2$$



```
% x: mxn -> m input_vector for timestep n
function [outputs] = forwardPropagation(obj, X)

    h = zeros(obj.hiddenDim,1);
    C = zeros(obj.hiddenDim,1);

    for t=1:obj.timesteps

        obj.layers{t}.last_h = h;
        obj.layers{t}.last_C = C;
        [y,h]=obj.layers{t}.activateLayer(X(:,t));

        obj.outputs(:,t) = y;
    end

    outputs = obj.outputs;
end
```







ans =

'Test Mean MSE: 2.827237e-02'

ans =

'Test Variance MSE: 8.406420e-04'

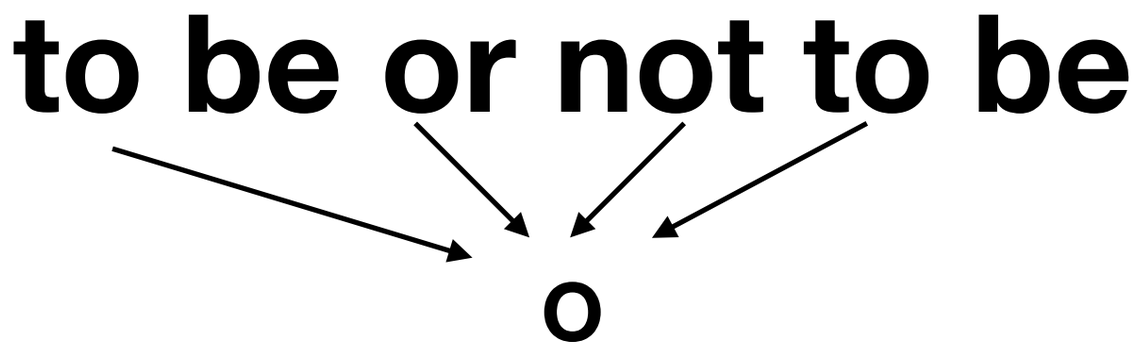
Cross validation error is different from test error

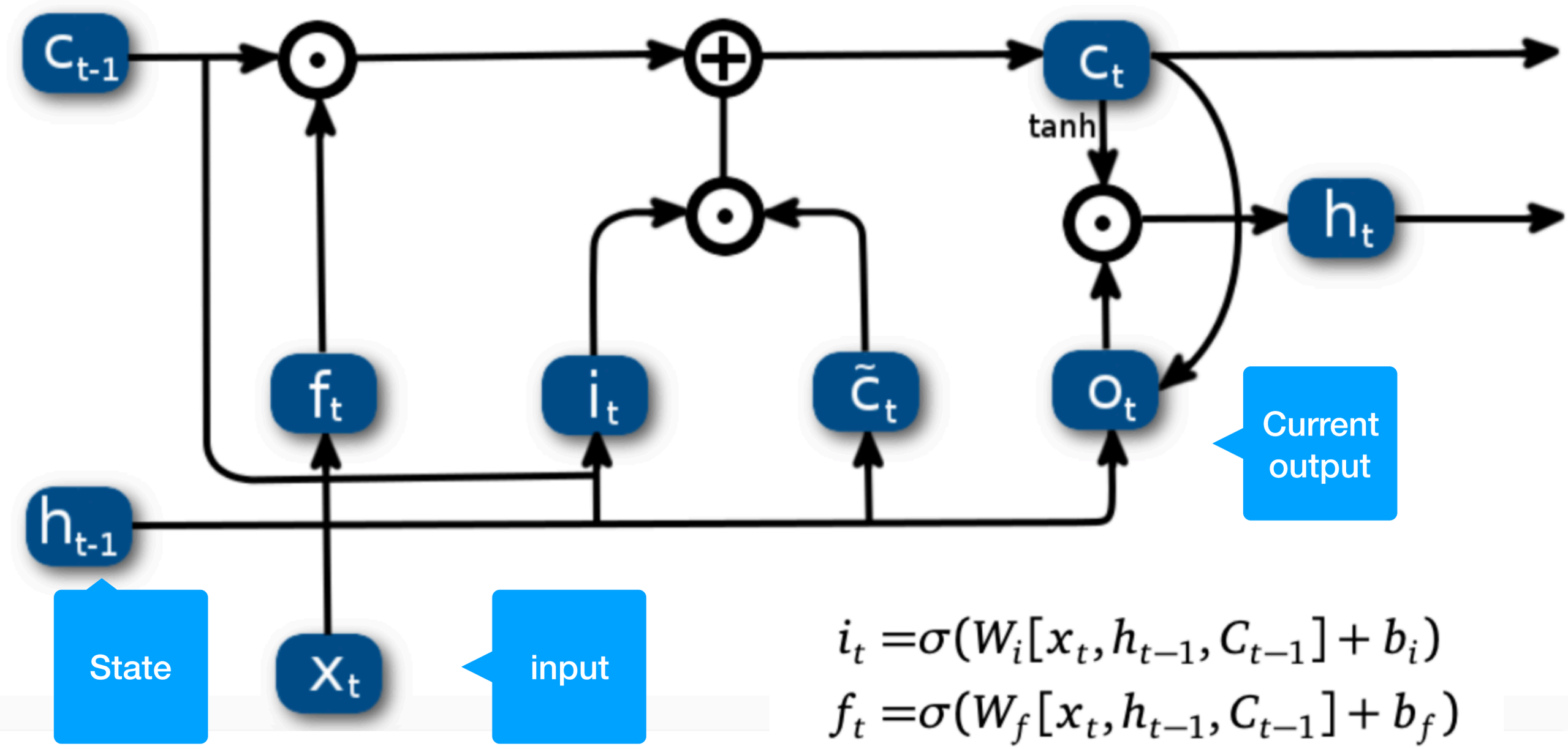
Derivations of Backpropagation Through Time Gradients for LSTM Neural Networks

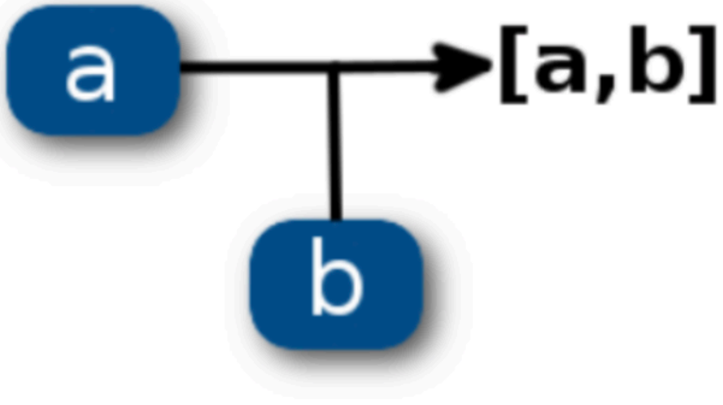
Part of a Honors Thesis Regarding LSTM Neural Networks

By Moritz Nakatenus

December 2017

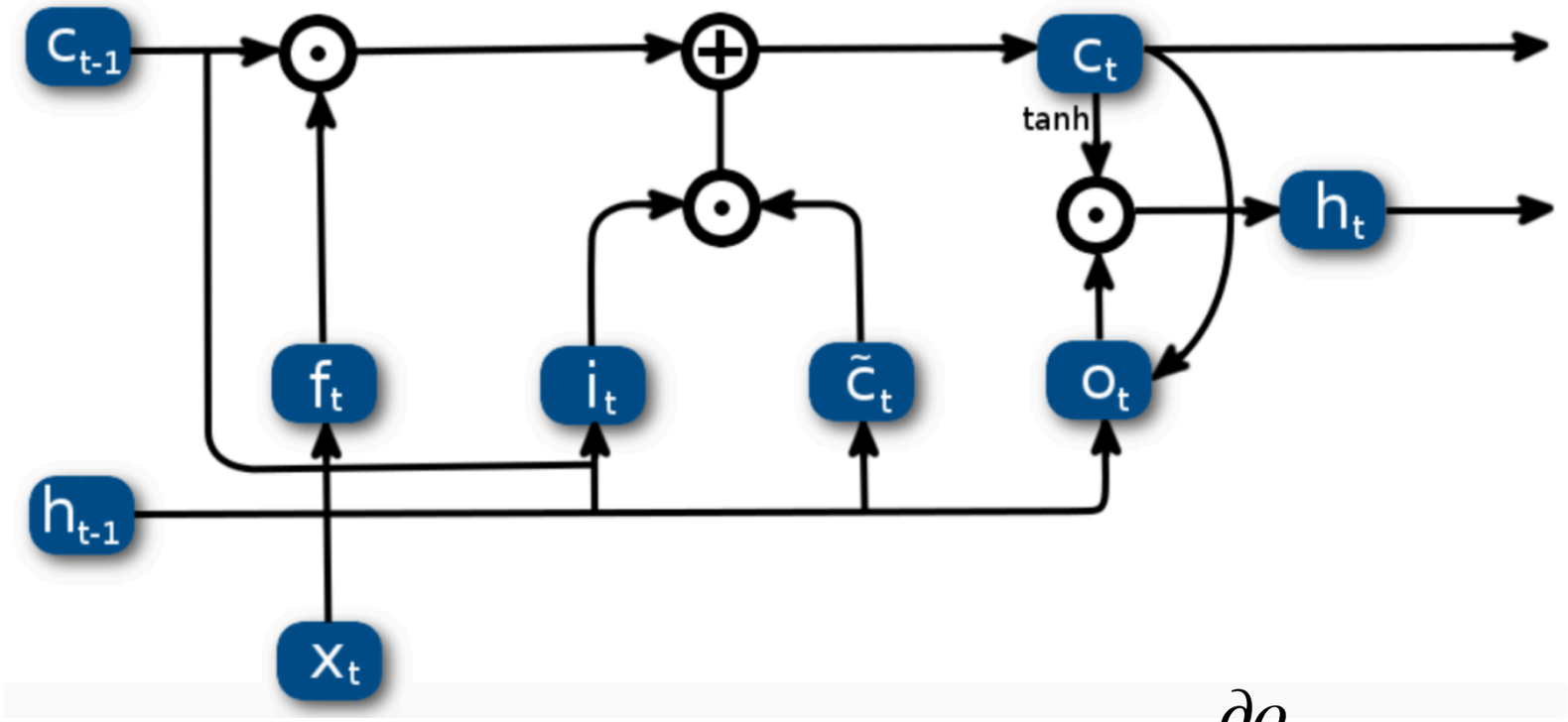






$$i_t = \sigma(W_i[x_t, h_{t-1}, C_{t-1}] + b_i)$$
$$f_t = \sigma(W_f[x_t, h_{t-1}, C_{t-1}] + b_f)$$
$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma(W_o[x_t, h_{t-1}, C_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

Forward propagation



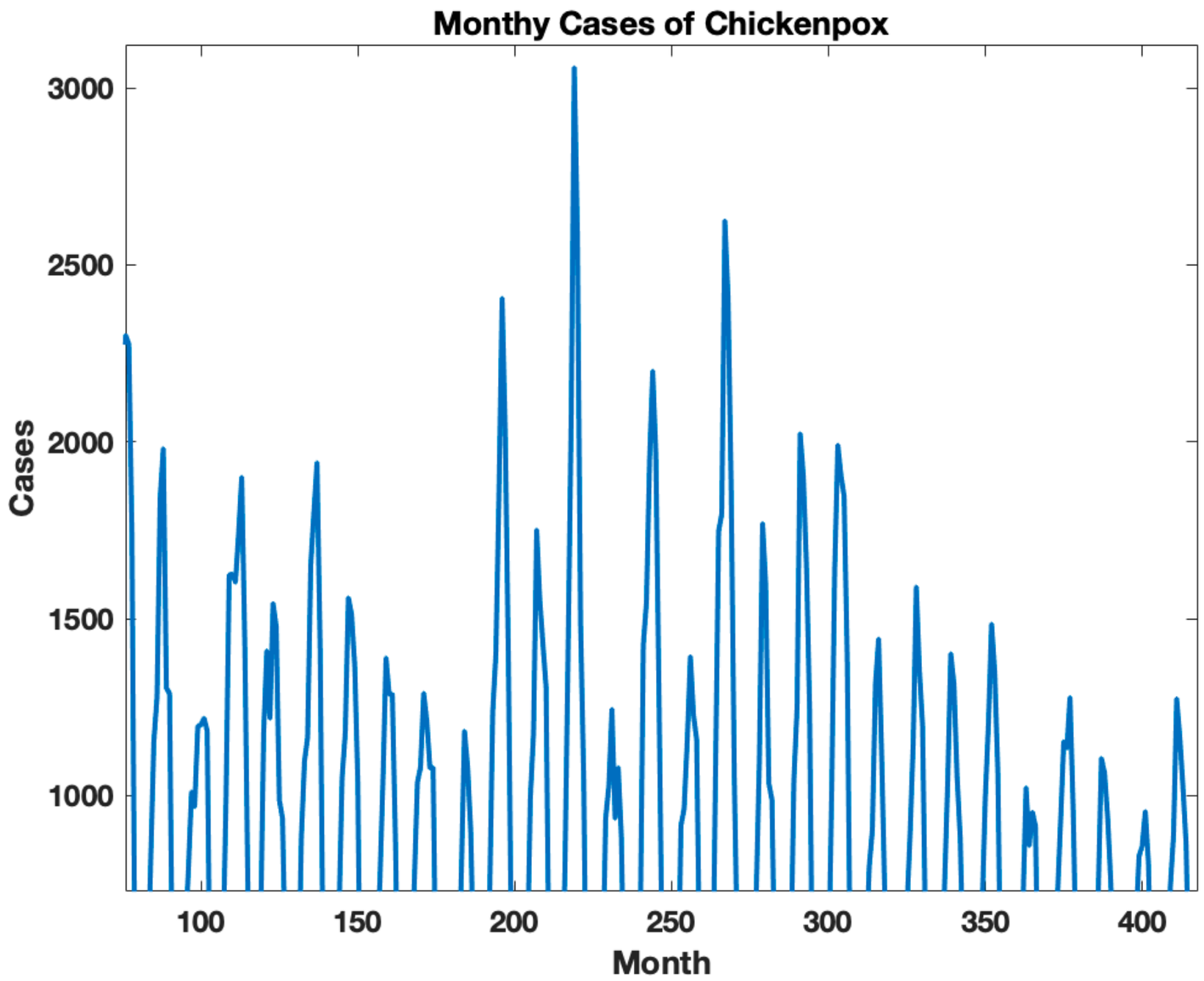
Flow chart of Calculating Gradients

$$\frac{\partial o_t}{\partial h_{t-1}}$$

$$\frac{\partial i_t}{\partial h_{t-1}} \quad \frac{\partial f_t}{\partial h_{t-1}} \quad \frac{\partial \tilde{C}_t}{\partial h_{t-1}} \rightarrow \frac{\partial C_t}{\partial h_{t-1}} \rightarrow \frac{\partial h_t}{\partial h_{t-1}}$$

```
dhNext_dh = nextLayer.weights.W_o(:,obj.o_hArgInd)' * dsigmoid(obj, nextLayer.o) *diag(tanh(nextLayer.C)) ...

+ (nextLayer.weights.W_f(:,obj.f_hArgInd)' * dsigmoid(obj, nextLayer.f) * diag(layer.C) ...

                + nextLayer.weights.W_i(:,obj.i_hArgInd)' * dsigmoid(obj, nextLayer.i) * diag(nextLayer.C_tild) ...

   + nextLayer.weights.W_c(:,obj.c_hArgInd)' * dtanh(obj, nextLayer.C_tild) * diag(nextLayer.i)) ...
* dtanh(obj, nextLayer.C) * diag(nextLayer.o);

delta_h = dhNext_dh * delta_h;
```

Matlab
Time series forecasting
Using deep learning


Monthy Cases of Chickenpox
3000
2500
2000
1500
1000
Cases
100
150
200
250
300
350
400
Month

Partition the training and test data

```
data = chickenpox_dataset;
data = [data{:}];

figure
plot(data)
xlabel("Month")
ylabel("Cases")
title("Monthy Cases of Chickenpox")
```

```
numTimeStepsTrain = floor(0.9*numel(data));

dataTrain = data(1:numTimeStepsTrain+1);
dataTest = data(numTimeStepsTrain+1:end);
```

Standardize Data

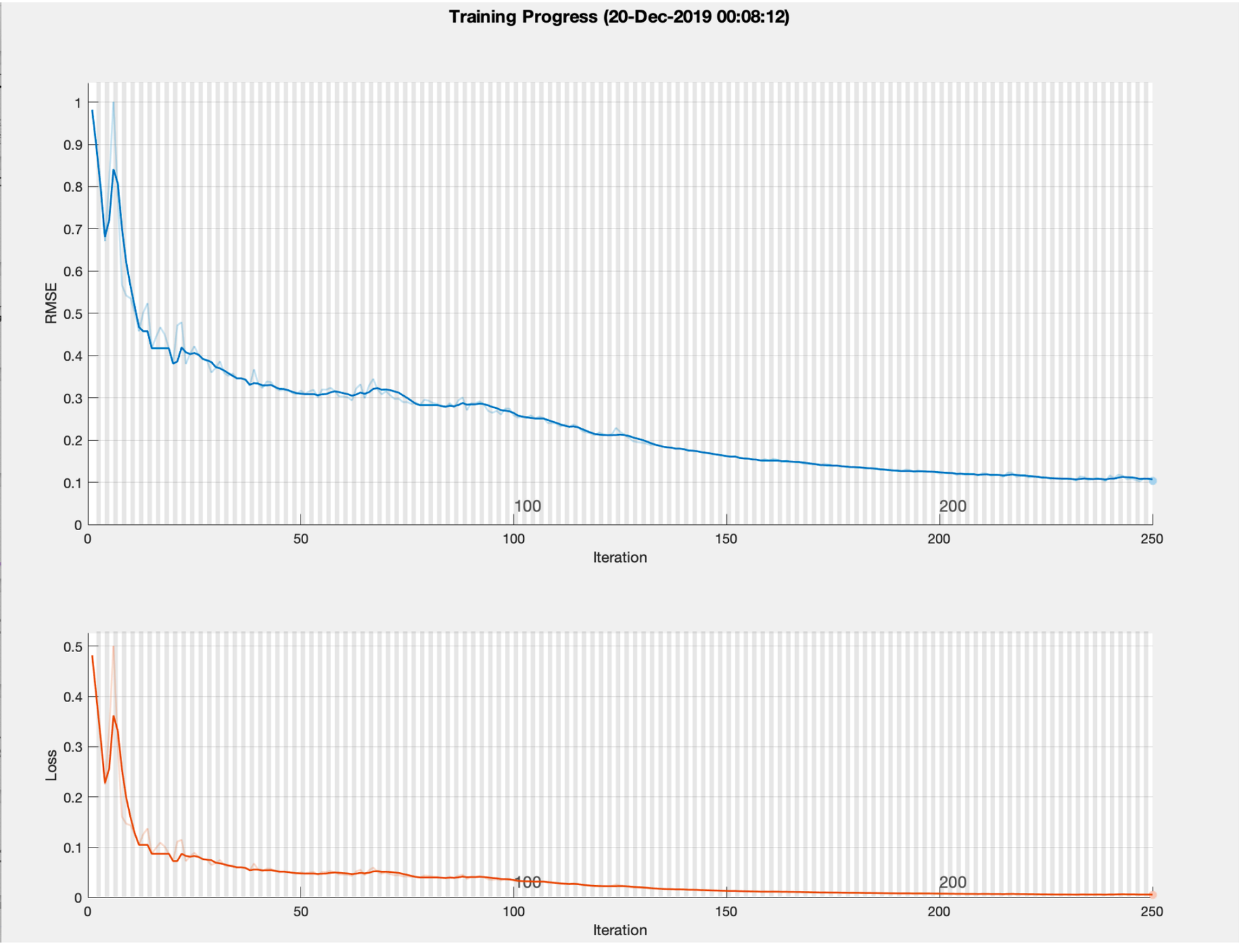
```
mu = mean(dataTrain);
sig = std(dataTrain);

dataTrainStandardized = (dataTrain - mu) / sig;
```

```
XTrain = dataTrainStandardized(1:end-1);
YTrain = dataTrainStandardized(2:end);
```

```
numFeatures = 1;
numResponses = 1;
numHiddenUnits = 200;

layers = [ ...
sequenceInputLayer(numFeatures)
lstmLayer(numHiddenUnits)
fullyConnectedLayer(numResponses)
regressionLayer];
```

```
options = trainingOptions('adam', ...
    'MaxEpochs',250, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.005, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',125, ...
    'LearnRateDropFactor',0.2, ...
    'Verbose',0, ...
    'Plots','training-progress');
```

```
net = trainNetwork(XTrain,YTrain,layers,options);
```



```
dataTestStandardized = (dataTest - mu) / sig;
XTest = dataTestStandardized(1:end-1);
```

```
net = predictAndUpdateState(net,XTrain);
[net,YPred] = predictAndUpdateState(net,YTrain(end));

numTimeStepsTest = numel(XTest);
for i = 2:numTimeStepsTest
    [net,YPred(:,i)] =
predictAndUpdateState(net,YPred(:,i-1),'ExecutionEnvironment','cpu');
end
```

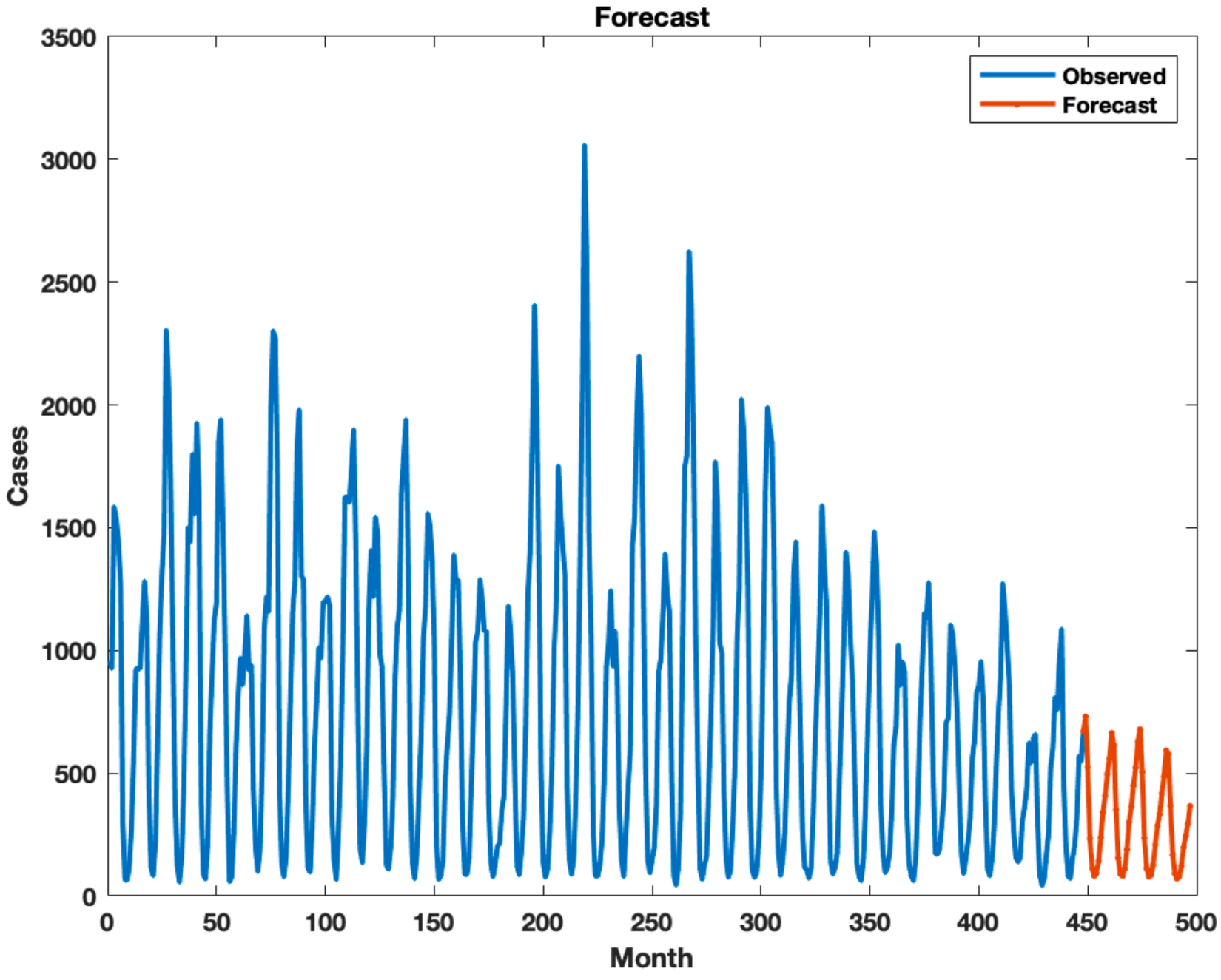
Unstandardize the predictions using the parameters calculated earlier.

```
YPred = sig*YPred + mu;
```

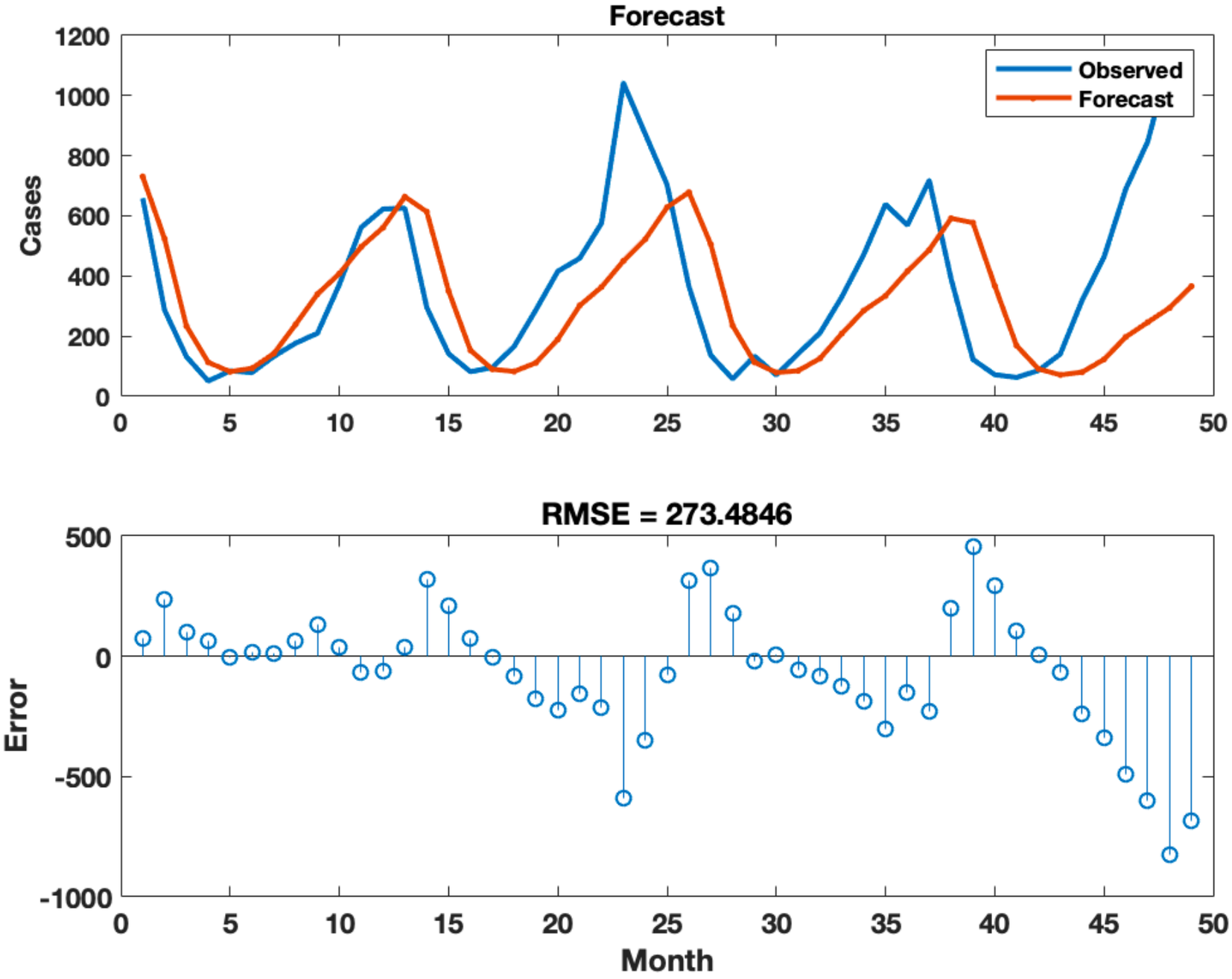
The training progress plot reports the root-mean-square error (RMSE) calculated from the standardized data. Calculate the RMSE from the unstandardized predictions.

```
YTest = dataTest(2:end);
rmse = sqrt(mean((YPred-YTest).^2))
rmse = single
    2.7348459e+02
```

Plot the training time series with the forecasted values.


Forecast
Observed
Forecast
Cases
Month
3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
400
450
500

Project:
Times series prediction using deep learning


Forecast
Observed
Forecast
Cases
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
40
45
50
RMSE = 273.4846
Error
500
0
-500
-1000
Month