Dealing with Longer Sequences

Advanced ML with TensorFlow on GCP

End-to-End Lab on Structured Data ML

Production ML Systems

Image Classification Models

Sequence Models

Recommendation Systems

Agenda

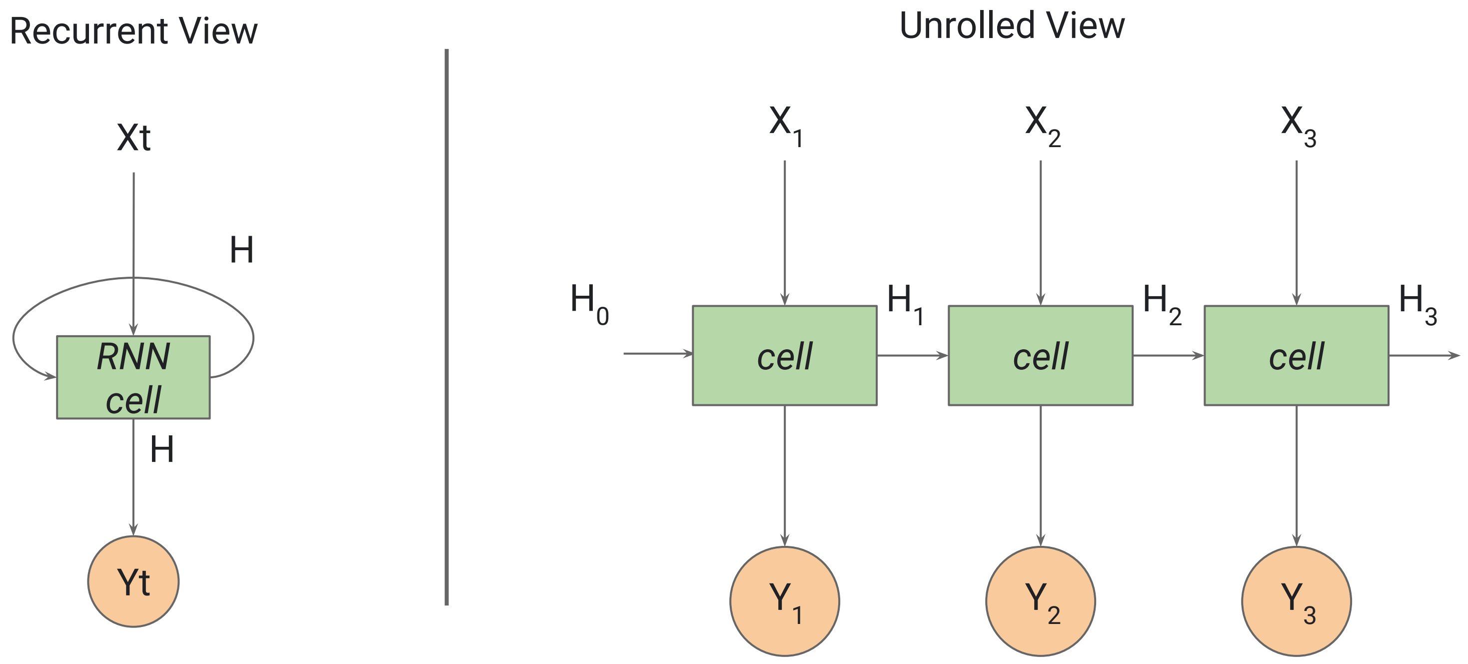
LSTMs and GRUs

Deep RNNs

Improving our loss function

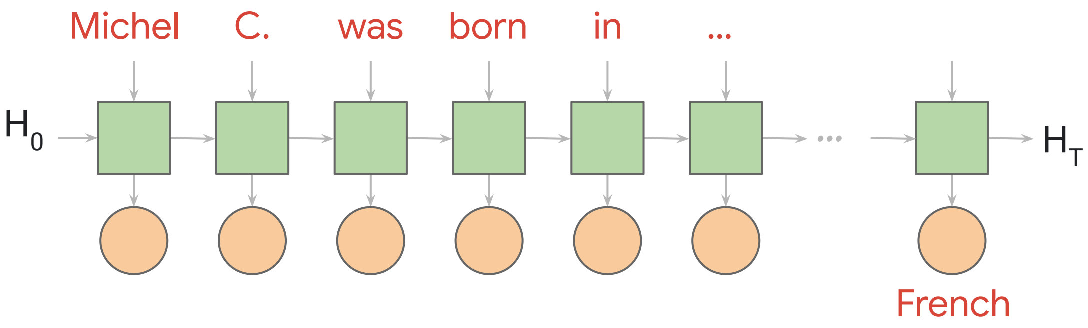
Working with real world data

Problem with RNNs: Long term dependencies



Problem with RNNs: Long term dependencies

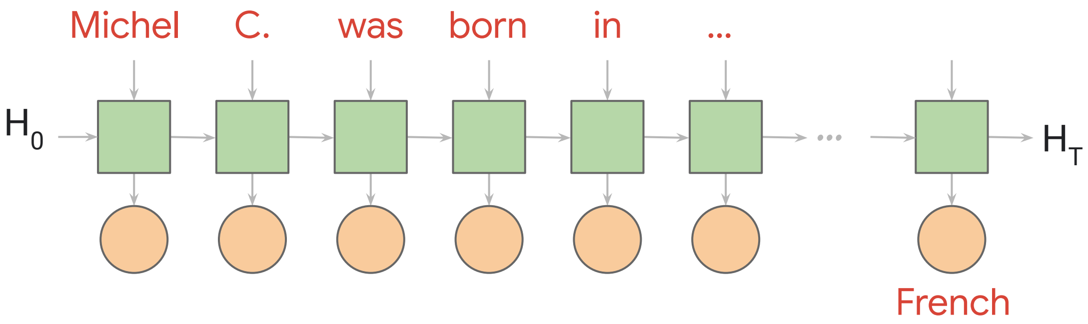
“Michel C. was born in Paris, France. His mother tongue is ***? ? ? ?”***



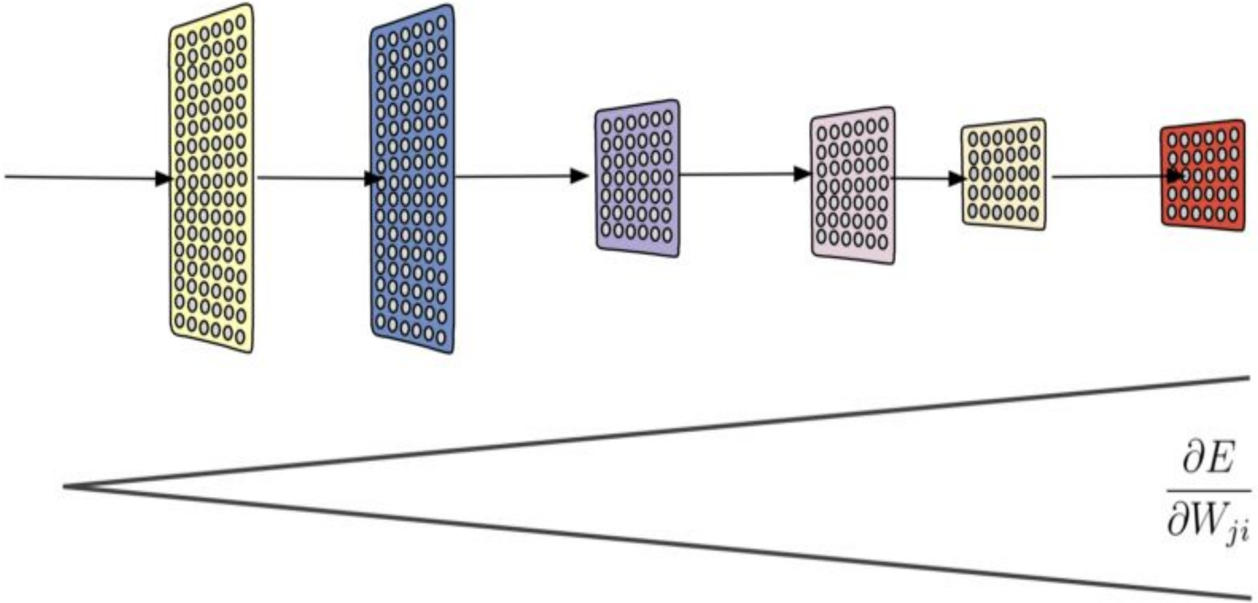
T = max sequence length

Problem with RNNs: Long term dependencies

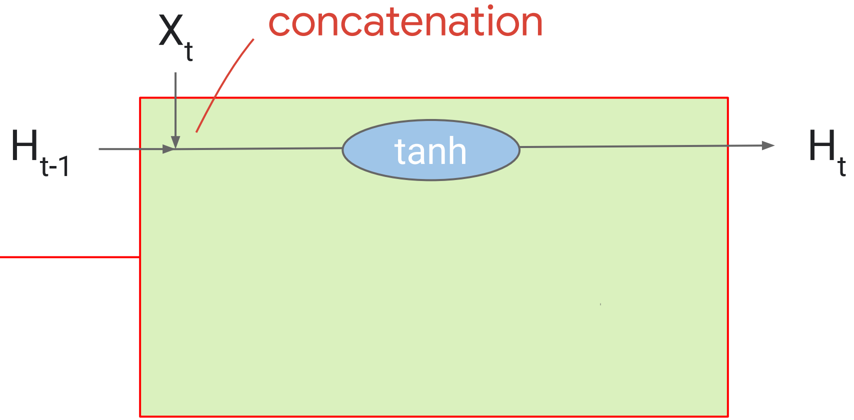
"Michel C. was born in Paris, France. He is married and has three children. He received a M.S. in neurosciences from the University Pierre & Marie Curie and the Ecole Normale Supérieure in 1987, and and then spent most of his career in Switzerland, at the Ecole Polytechnique de Lausanne. He specialized in child and adolescent psychiatry and his first field of research was severe mood disorders in adolescent, topic of his PhD in neurosciences (2002). His mother tongue is ***? ? ?***

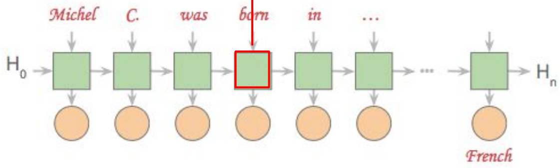


Problem with RNNs: Vanishing gradients



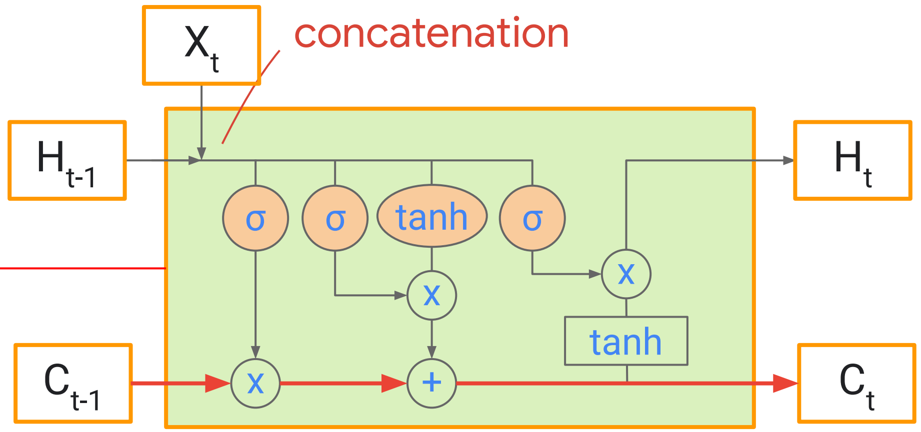
Simple RNN cell



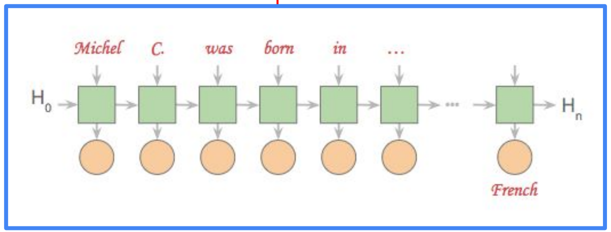


$$\text{tanh} = \tanh([X_t|H_{t-1}].W_H + b_H)$$

LSTM cell



X
t
concatenation
H
t-1
σ
σ
tanh
σ
x
x
tanh
H
t
C
t-1
x
+
C
t

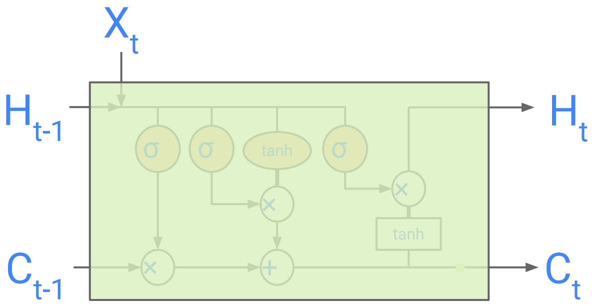


Michel
C.
was
born
in
...
H
0
...
H
n
French

LSTM cell

LSTM = Long Short Term Memory

		vector sizes
concatenate :	$X = X_t \mid H_{t-1}$	p+n
forget gate :	f = σ(X.Wf + bf)	n
update gate :	u = σ(X.Wu + bu)	n
output gate :	o = σ(X.Wr + br)	n
input :	X’ = tanh(X.Wc + bc)	n
new C :	Ct = f * Ct-1 + u * X’	n
new H :	Ht = o * tanh(Ct)	n



Our cell has been given control abilities

1. What to forget from the cell state.
2. What new data to store into the cell state.
3. What data from the cell state to expose to the hidden state.

Gated Recurrent Unit (GRU)

vector sizes

$X = X_t \mid H_{t-1}$ p+n

2 gates instead of 3
=> cheaper

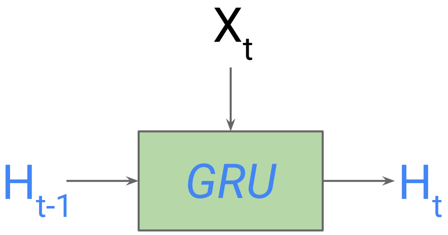
$z = \sigma(X.Wz + bz)$ n

$r = \sigma(X.Wr + br)$ n

$X' = X_t \mid r * H_{t-1}$ p+n

$X'' = \tanh(X'.Wc + bc)$ n

$Ht = (1-z) * Ht-1 + z * X''$ n



Quiz: How many weight matrices do a simple RNN, LSTM, and GRU cell have respectively?

1. 1, 4, 3
2. 1, 4, 4
3. 1,1,1
4. Depends on the number of time steps

Quiz: How many weight matrices do a simple RNN, LSTM, and GRU cell have respectively?

1. $\boxed{1, 4, 3}$
2. 1, 4, 4
3. 1,1,1
4. Depends on the number of time steps

Use an RNN in TensorFlow

```
CELL_SIZE = 32  # size of the cell's internal state

# 1. Choose RNN Cell type
cell = tf.nn.rnn_cell.GRUCell(CELL_SIZE) # or BasicLSTMCell or BasicRNNCell
# 2. Create RNN by passing cell and tensor of features (x)
outputs, state = tf.nn.dynamic_rnn(cell, x, dtype=tf.float32)

# x needs shape: [BATCH_SIZE, MAX_SEQUENCE_LENGTH, INPUT_DIM]
# outputs has shape: [BATCH_SIZE, MAX_SEQUENCE_LENGTH, CELL_SIZE]

# state has shape: [BATCH_SIZE, CELL_SIZE]
# 3. Pass rnn state through a DNN to get prediction
h1 = tf.layers.dense(state, DNN, activation=tf.nn.relu)
predictions = tf.layers.dense(h1, 1, activation=None) # (BATCH_SIZE, 1)
return predictions
```

Lab

Time series prediction: end-to-end (rnn)

In this lab we will continue with the sprinkler problem, this time implementing an RNN to predict the sprinkler height.

Lab Steps

1. Complete rnn_model function in model.py.

2. Run RNN model locally to make sure code works.

Agenda

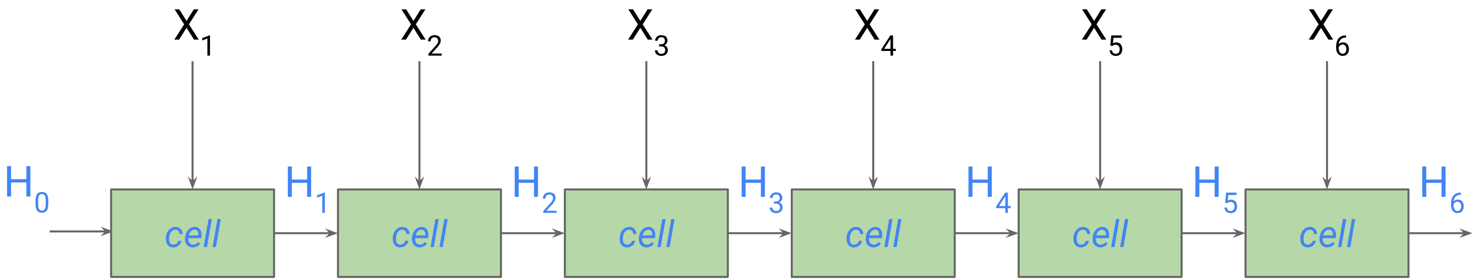
LSTMs and GRUs

Deep RNNs

Improving our loss function

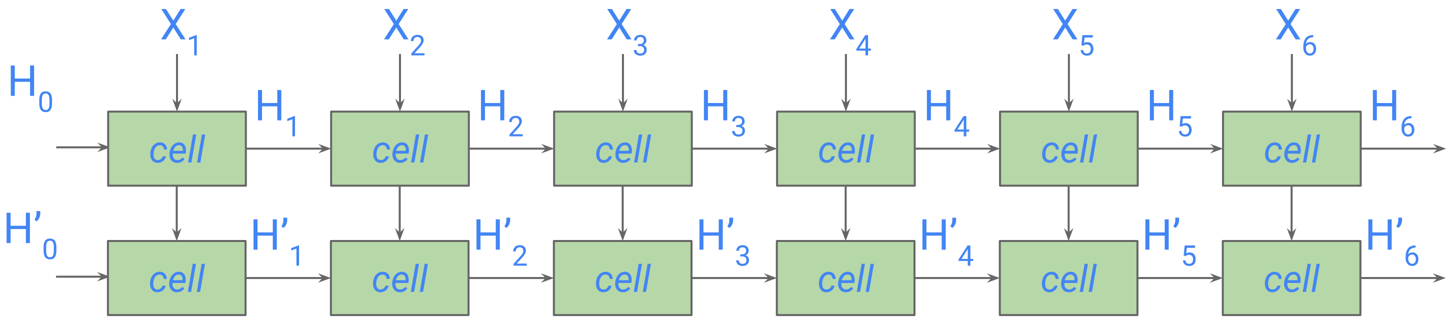
Working with real world data

Shallow RNN



The same weights and biases shared and updated in training loop (across iterations).

Two layer RNN



$$H_t = \tanh([X_t|H_{t-1}].W + b)$$

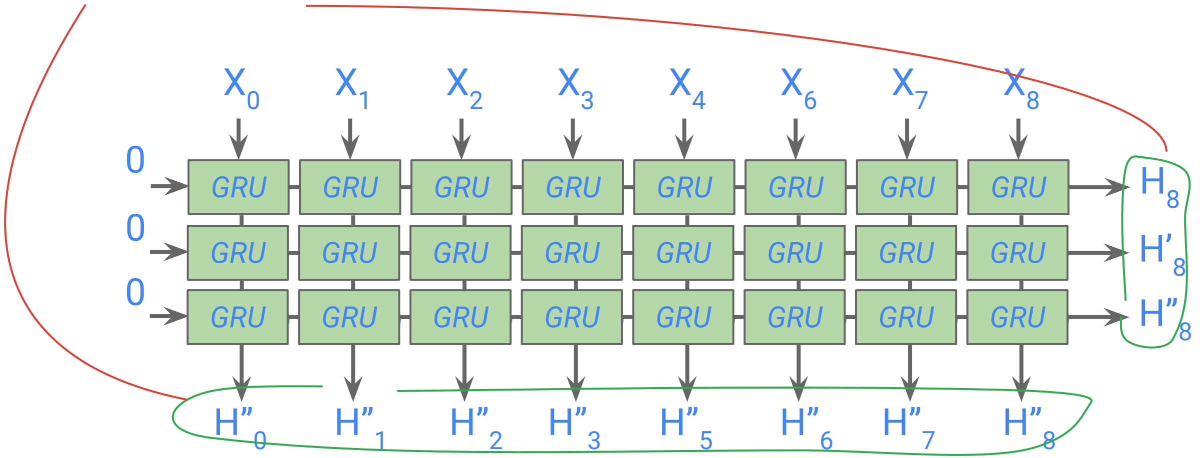
$$H'_t = \tanh([H_t|H'_{t-1}].W' + b')$$

Deep RNNs in Tensorflow

```
cells = [tf.nn.rnn_cell.GRUCell(CELLSIZE) for _ in range(NLAYERS)]

mcell = tf.nn.rnn_cell.MultiRNNCell(cells)

outputs, state = tf.nn.dynamic_rnn(mcell, X, dtype=tf.float32)
```



Lab

Time series prediction: end-to-end (rnn2)

In this lab, you will define a rnn2 model to find next value of a time series.

Lab Steps

1. Complete rnn_model function in model.py.

2. Run RNN model locally to make sure code works.

3. Run all models in in the cloud.

4. Compare results using Tensorboard.

Agenda

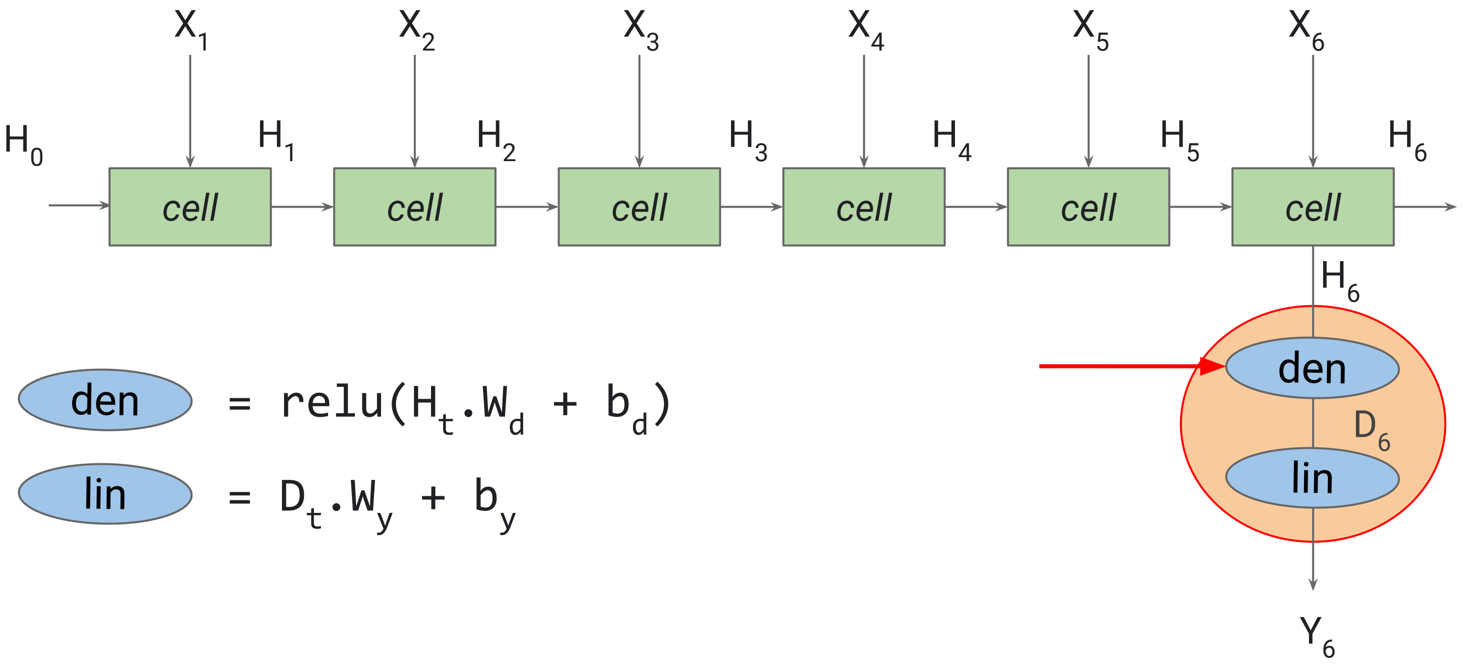
LSTMs and GRUs

Deep RNNs

Improving our loss function

Working with real world data

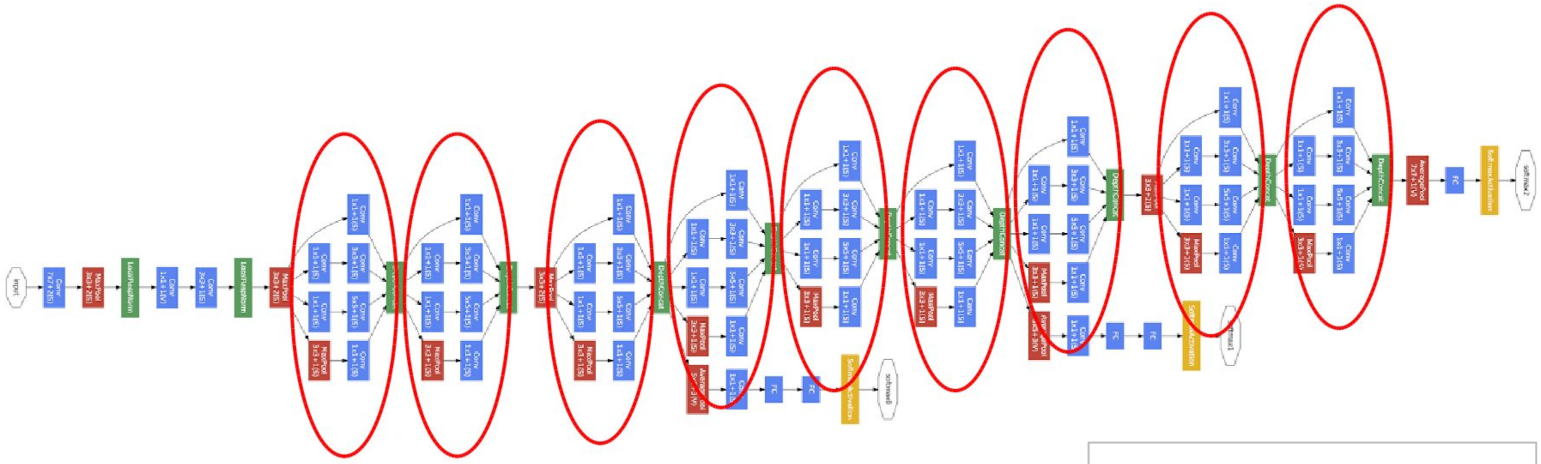
Improving our loss function



den $= \mathtt{relu}(H_t.W_d + b_d)$

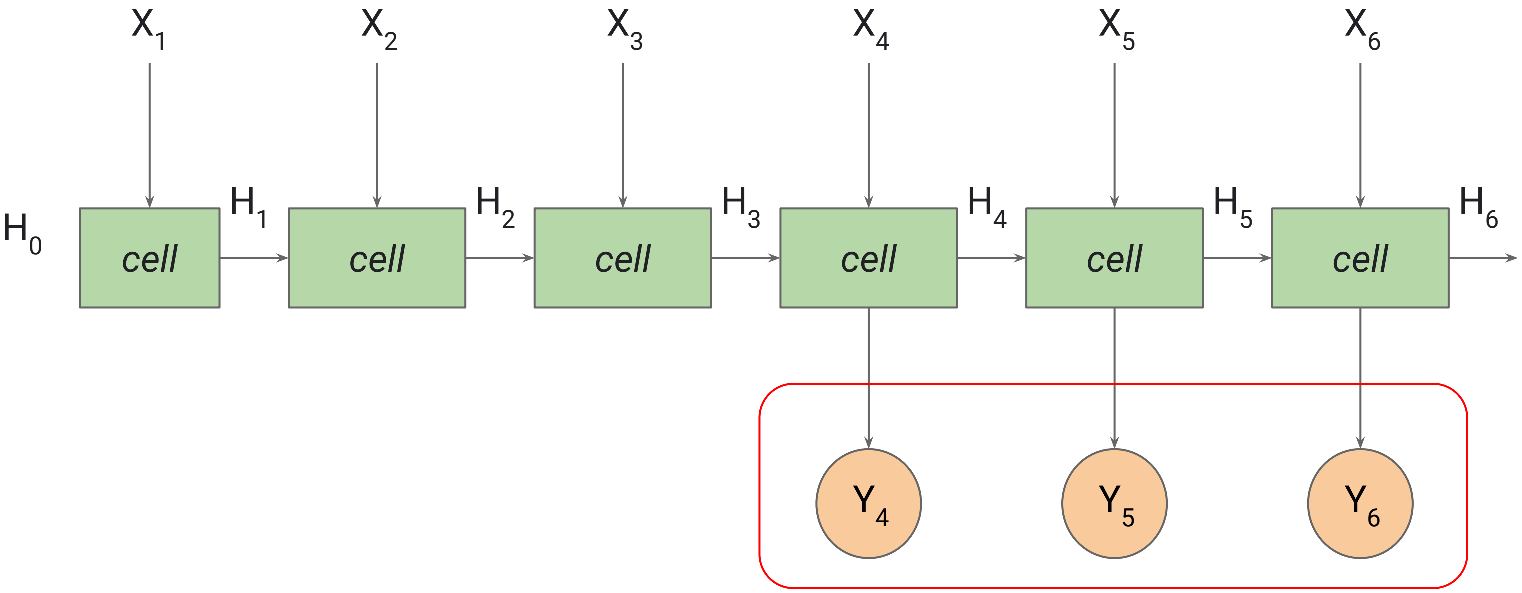
lin $= D_t.W_y + b_y$

Inception CNN taking intermediate outputs



Convolution
Pooling
Softmax
Concat/Normalize

Improving our loss function



Average loss over multiple predictions

Agenda

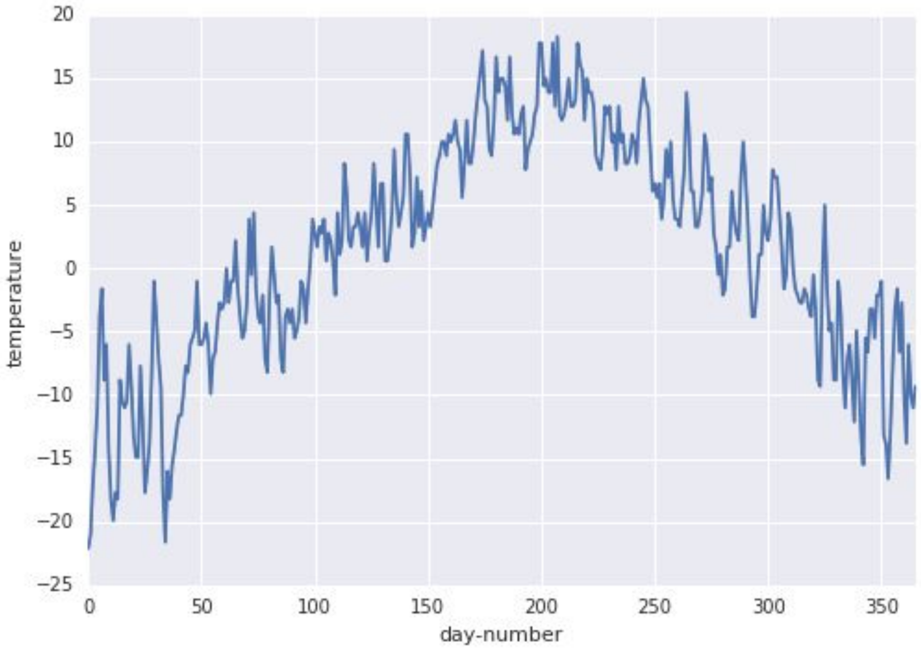
LSTMs and GRUs

Deep RNNs

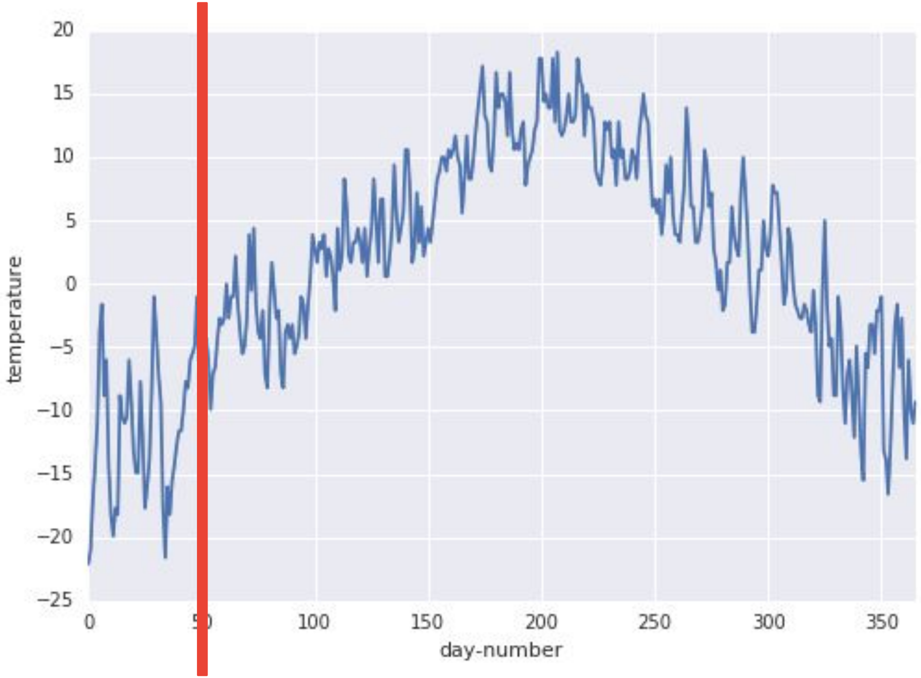
Improving our loss function

Working with real world data

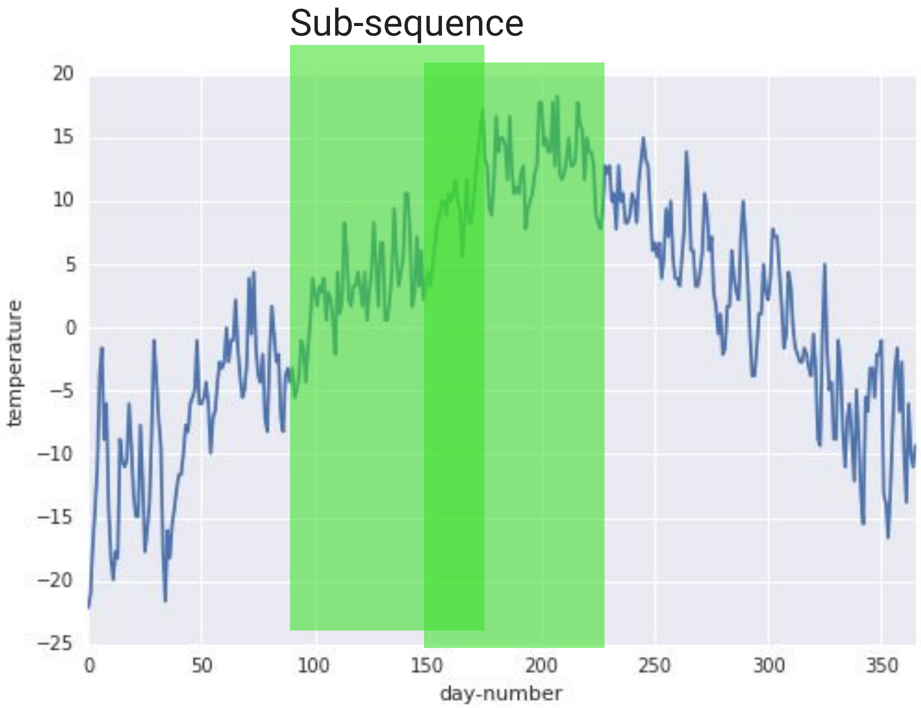
Working with real world sequential data



Splitting up sequences



Splitting up sequences

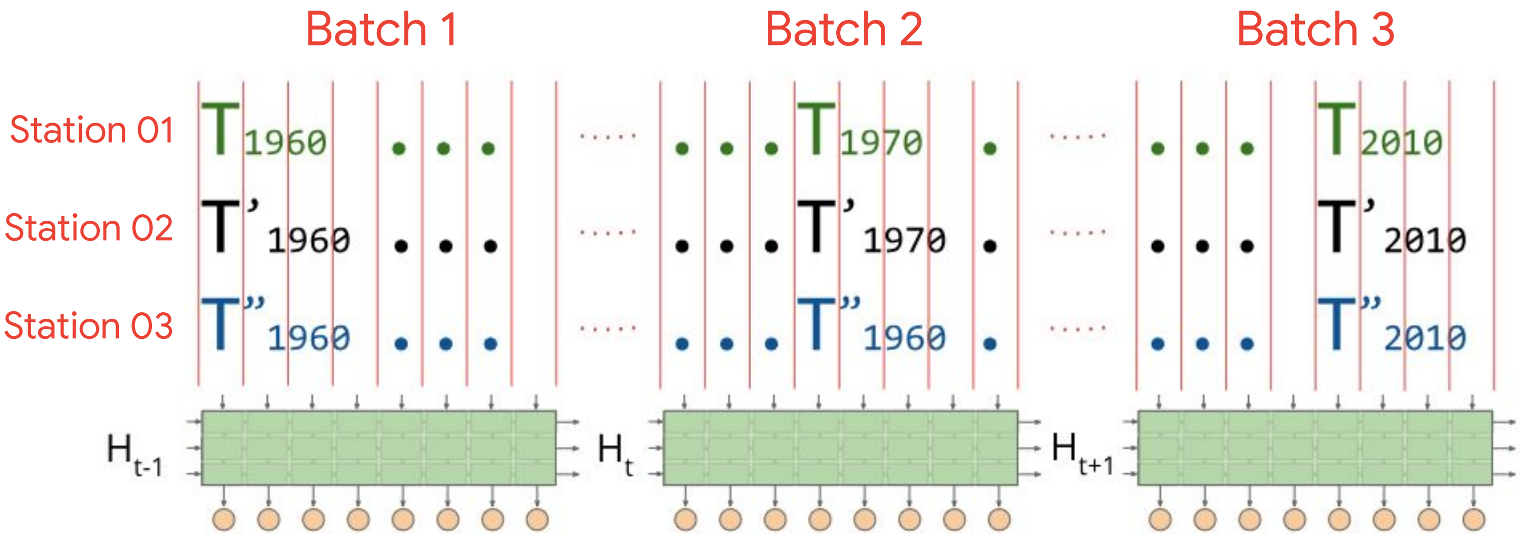


Splitting up sequences

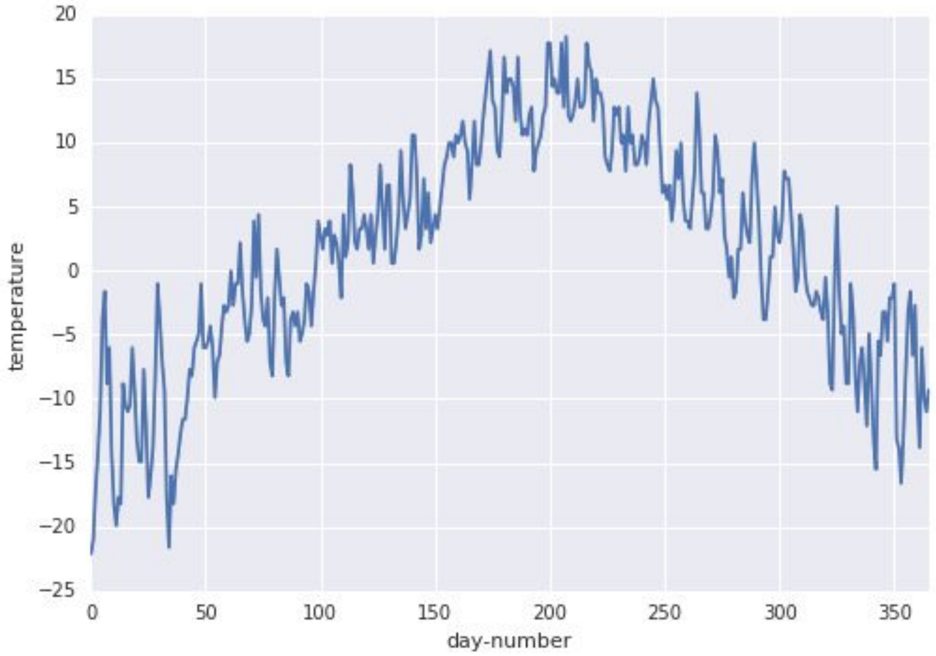
Original Sequence
1, 2, 3, 4, 5, 6

Non-overlapping	Sequence 1 1,2,3	Sequence 2 4, 5, 6
Overlapping	Sequence 3 3, 4, 5	Sequence 4 4, 5, 6

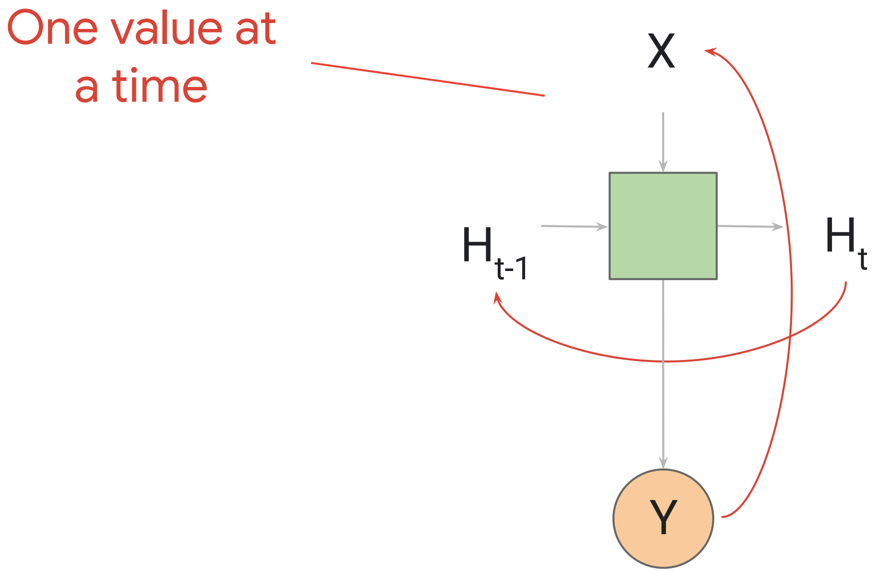
Propagating state between batches



Splitting up sequences



Predicting multiple time steps ahead



Other considerations

1. Resampling data.
2. One model versus multiple models.
3. Incorporating non-sequential data.

Resampling data

One versus multiple models

Model 1 | Model 2 | Model 3

Station #1 | Station #2 | Station #3

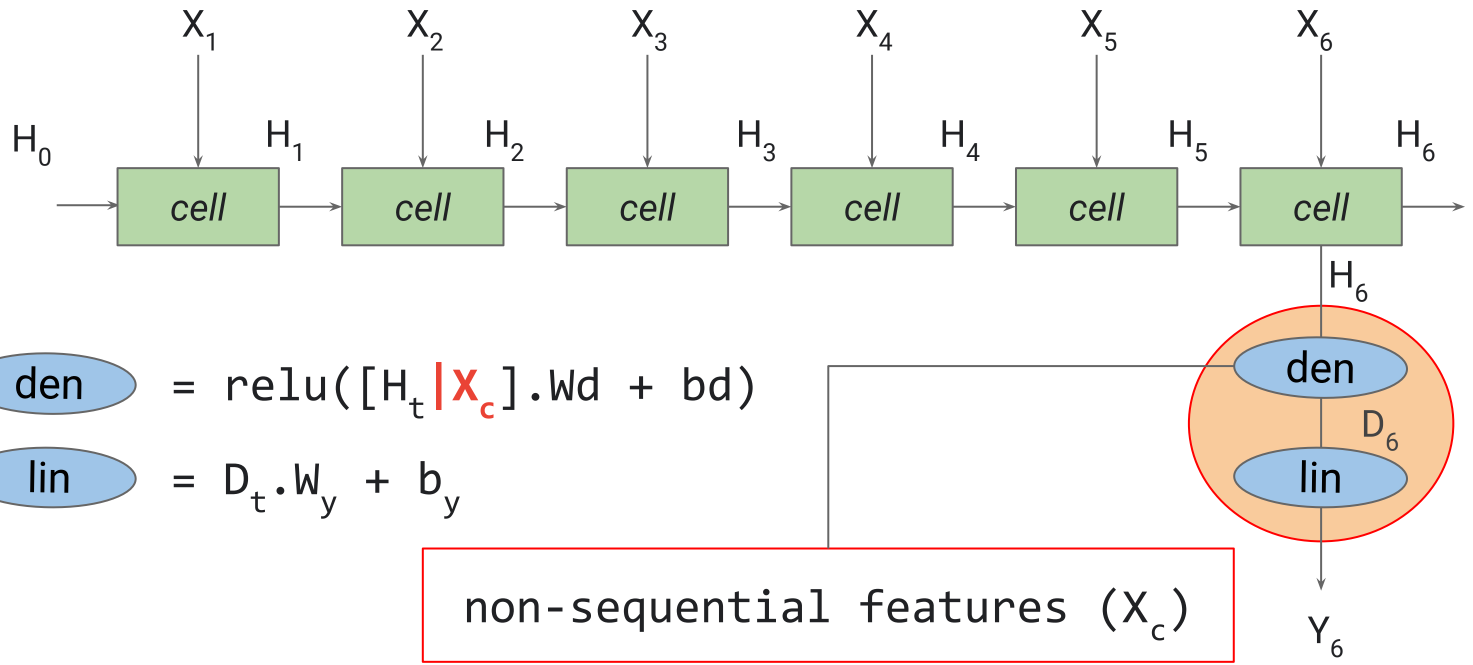
One Model

Station #1 | Station #2 | Station #3

One Model

Station #1 LAT LONG | Station #2 LAT LONG | Station #3 LAT LONG

Incorporating non-sequential data



den $= \mathrm{relu}([H_t | X_c].Wd + bd)$

lin $= D_t.W_y + b_y$

non-sequential features (X_c)

Lab

Time series prediction: Temperature from weather stations

In this lab you'll see how to resample data, split up sequences, generate a sequence of predictions, and more.

Lab Steps

1. Run the notebook as it is. Look at the data visualisations. Then look at the predictions at the end.

2. Play with the data to find good values for RESAMPLE_BY and SEQLEN in hyperparameters.

3. Predict N data points ahead instead of only 1 ahead.

4. Adjust hyperparameters and add regularization.

cloud.google.com