SAN DIEGO MACHINE LEARNING

2022 FEB 19

Chapter 15: Processing Sequences using RNNs (& CNNs)

Discussion led by
Steven Fouskarinis

AGENDA

What is a Recurring Neural Network (RNN)?

Issues

- Unstable Gradients
- Short Term Memory (Long Sequences)
- Trend and Seasonality

Performance Summary

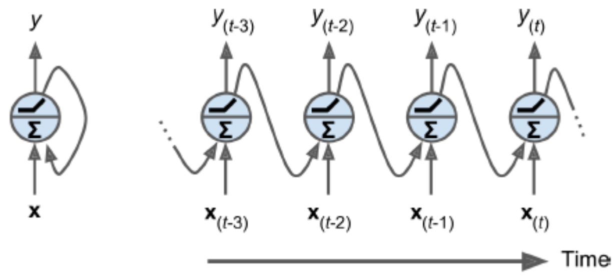
Other Models

- Weighted Moving Average
- ARIMA
- CNNs

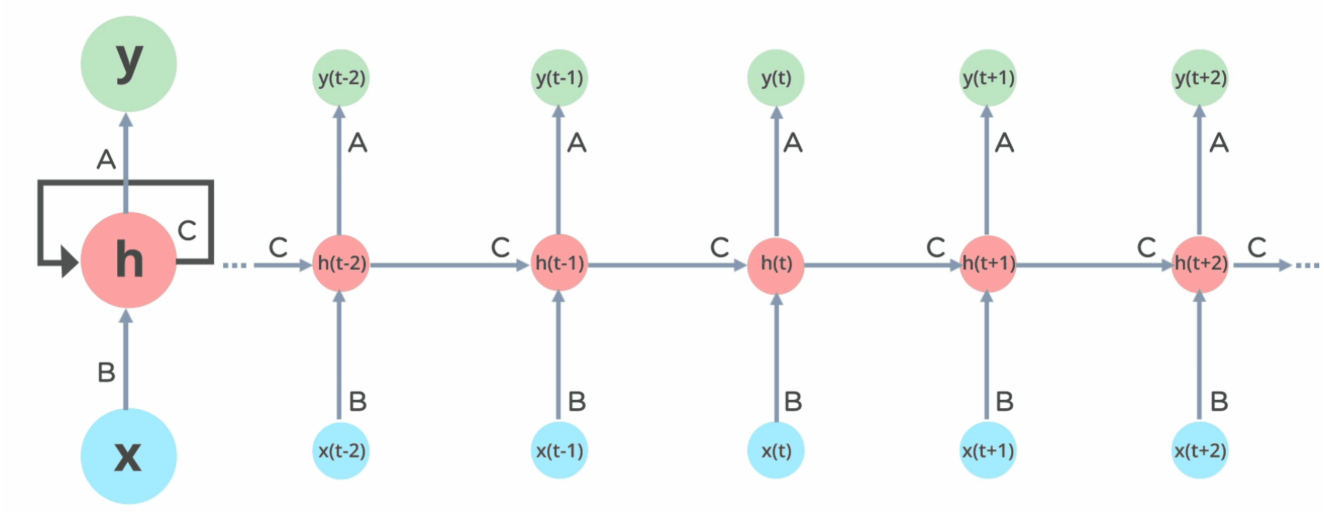
Applications

WHAT IS AN RNN?

Simple: feed forward neural network with output fed back into the input



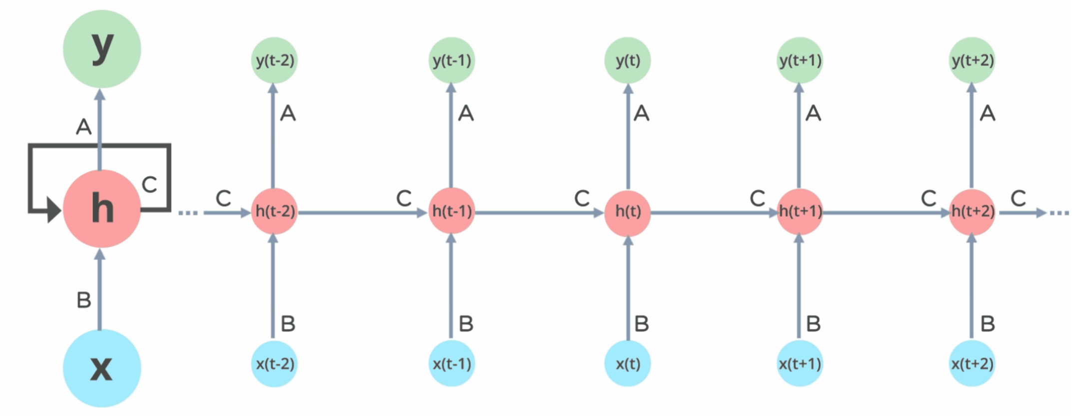
Unrolling in Time



WHAT IS AN RNN

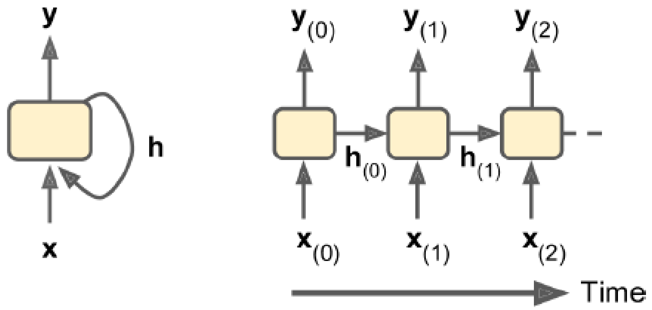
Stock Price example

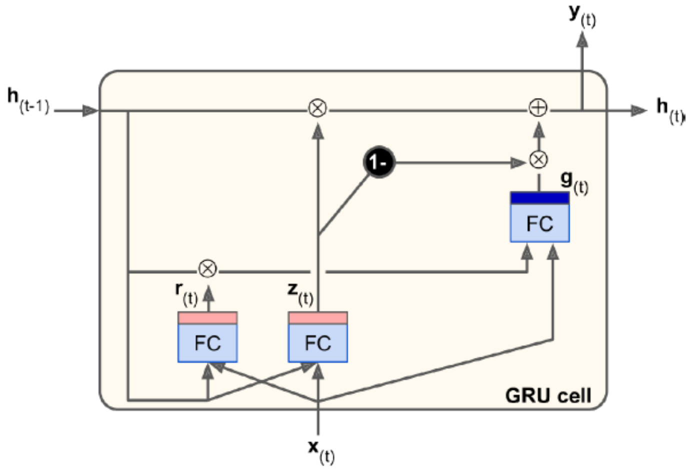
The Coca- Cola Company (KO)		
Date	Time Step	Closing Price
9-Feb	x(t-2)	$ 61.05
10-Feb	x(t-1)	$ 61.38
11-Feb	x(t)	$ 60.28
14-Feb	x(t+1)	$ 60.67
15-Feb	x(t+2)	$ 60.90

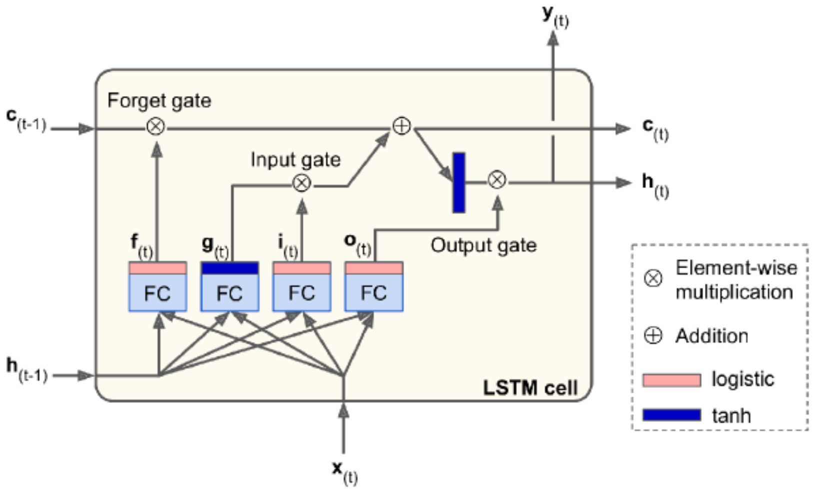


WHAT IS AN RNN?

More Accurate: hidden state ≠ output

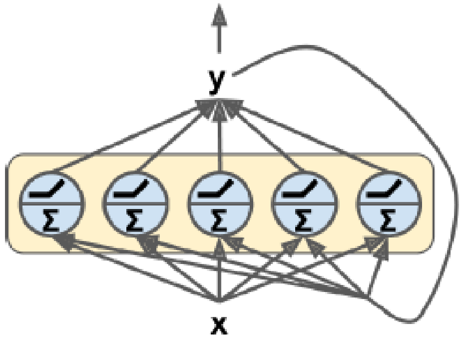


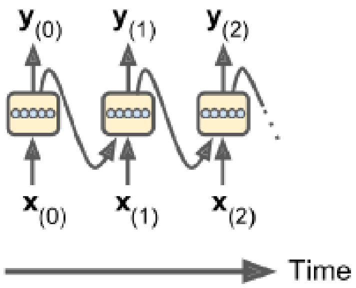




WHAT IS AN RNN?

More Complex: multiple neurons



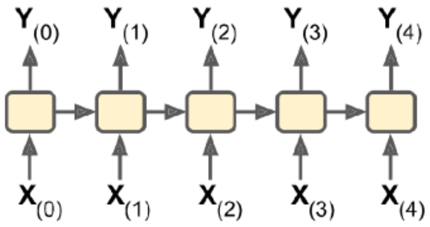


$$\mathbf{y}_{(t)} = \phi\left(\mathbf{W}_x{}^{\mathsf{T}}\mathbf{x}_{(t)} + \mathbf{W}_y{}^{\mathsf{T}}\mathbf{y}_{(t-1)} + \mathbf{b}\right)$$

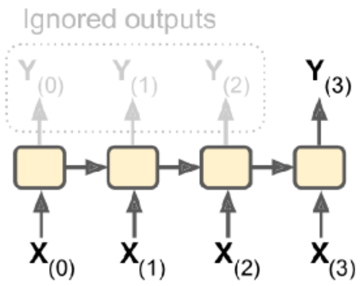
$$\begin{aligned} \mathbf{Y}_{(t)} &= \phi\left(\mathbf{X}_{(t)}\mathbf{W}_x + \mathbf{Y}_{(t-1)}\mathbf{W}_y + \mathbf{b}\right) \\ &= \phi\left(\left[\mathbf{X}_{(t)} \quad \mathbf{Y}_{(t-1)}\right]\mathbf{W} + \mathbf{b}\right) \text{ with } \mathbf{W} = \begin{bmatrix}\mathbf{W}_x \\ \mathbf{W}_y\end{bmatrix} \end{aligned}$$

RNN CONFIGURATIONS

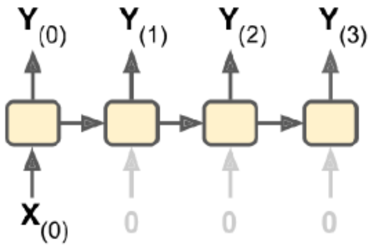
Sequence to Sequence
- Good for multiple stock price



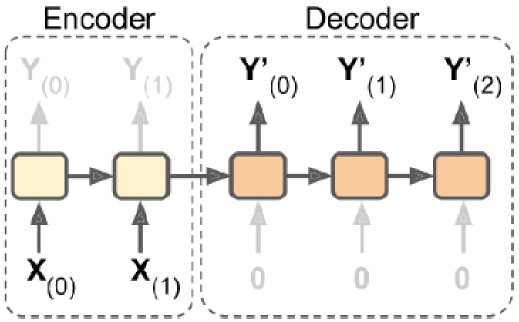
Sequence to Vector
- Good for single stock price



Vector to Sequence
- Good for music generation



Sequence to Vector (Encoder) –> Vector to Sequence (Decoder)
- Good for machine translation



KERAS API REFERENCE

https://keras.io/api/layers/recurrent_layers/

keras.layers.SimpleRNN()

keras.layers.RNN()

keras.layers.LSTM()

keras.layers.GRU()

```
model = keras.models.Sequential([
keras.layers.SimpleRNN(20, return_sequences=True, input_shape=[None, 1]),
keras.layers.SimpleRNN(20, return_sequences=True),
keras.layers.SimpleRNN(1)
])
```

ADVANTAGES & DISADVANTAGES

Pros

Can process input of arbitrary length

Model size does not increase with size of input

Computation takes into account historical information

Weights are shared across time

Cons

Computation slow

Short Memory

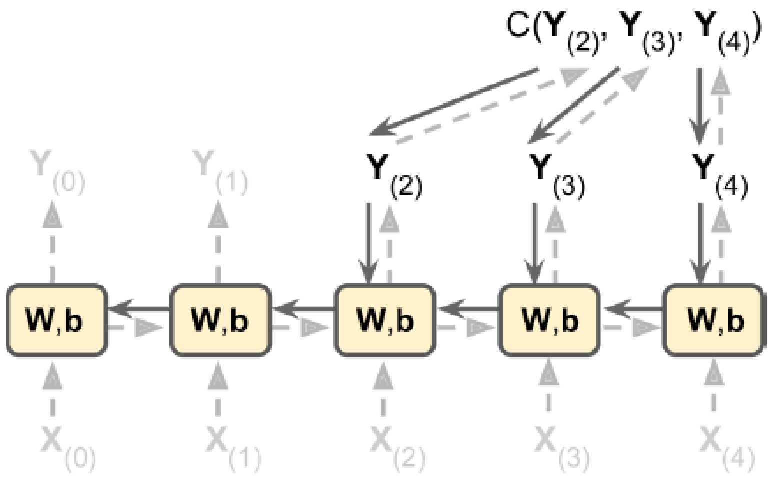
Cannot consider any future input for the current state

Weights are shared across time

TRAINING AN RNN

Unroll + Back propagation = Back Propagation Through Time (BPTT)

- For each Sequence
  - Process entire sequence $X_0 \ldots X_n$ -> Y_n
  - Calculate Loss Function for Y_n
  - Back Propagate Gradients for W_x & W_y
    - Repeat for Y_{n-1}, $Y_{n-2} \ldots Y_0$



Remember: applying the gradient to same W_x, W_y, b

ISSUES

Unstable Gradients

- Vanishing gradient
- Non convergence
- Runaway gradient

Short Term Memory (Long Sequences)

- Remembers back about 10 steps

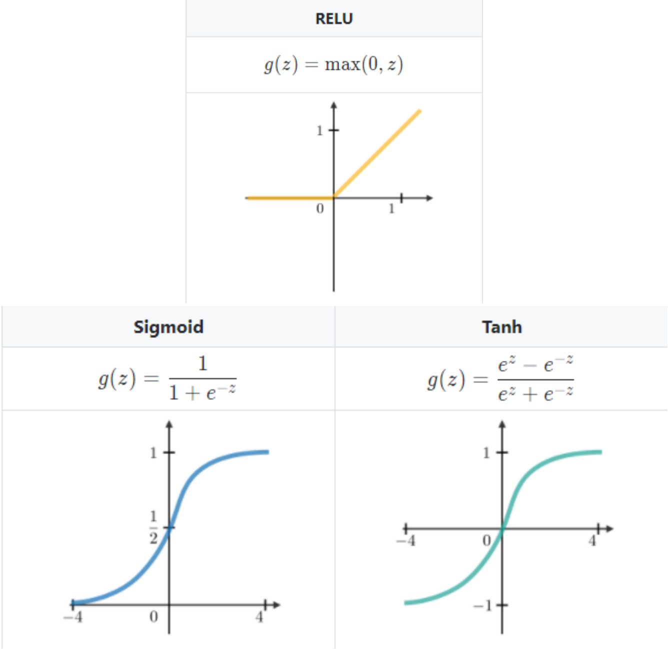
Trend and Seasonality

- Not necessary to remove but improves performance

UNSTABLE GRADIENTS

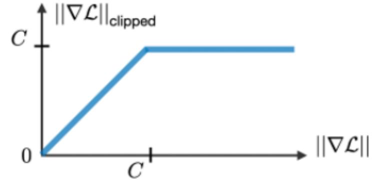
Vanishing Gradients & Non Convergence: usual DNN tactics

- Good parameter initialization
- Fast optimizers
- Dropout



RELU can make RNN more unstable

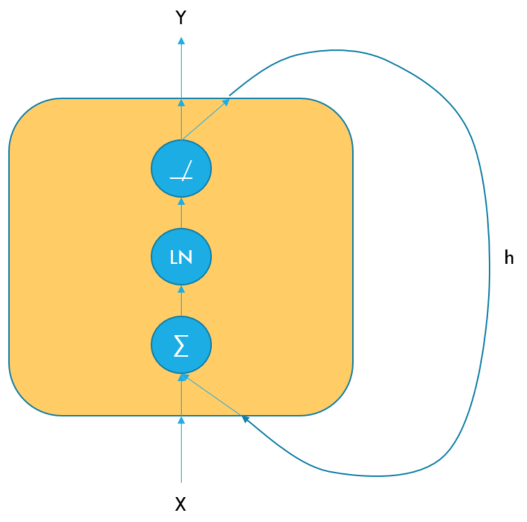
- Runaway weights
  - Use smaller learning rate, but may not fix
  - Saturating activation function: tanh, sigmoid
- Runaway gradients
  - Gradient clipping



UNSTABLE GRADIENTS

Can't use Batch Normalization

- Same normalization (parameters) applied at each time step
- Only slightly better than nothing when applied between layers[1]



Layer Normalization

- Normalize features (outputs) rather than batch
- Add after linear combination layer

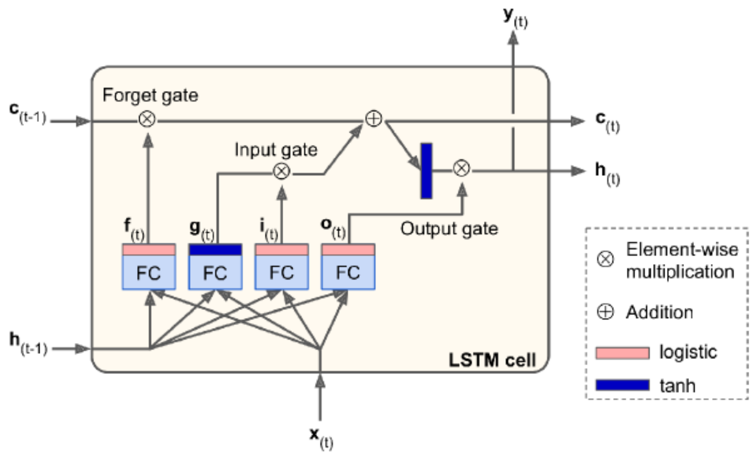
1 César Laurent et al., "Batch Normalized Recurrent Neural Networks," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (2016): 2657–2661.

SHORT TERM MEMORY

Simple RNN can only remember ~10 steps

Long Short-Term Memory (LSTM) Cell

- Better performance, faster convergence, detect long term dependencies
- $c_{(t)}$ long term memory, $h_{(t)}$ short term memory
- Forget gate $f_{(t)}$: which parts of long term state to erase
- Input gate $i_{(t)}$: which parts of prediction to add to long term state
- Output gate $o_{(t)}$: which parts of long term state to output to both $y_{(t)}$ and $h_{(t)}$
- Each gate is mini NN

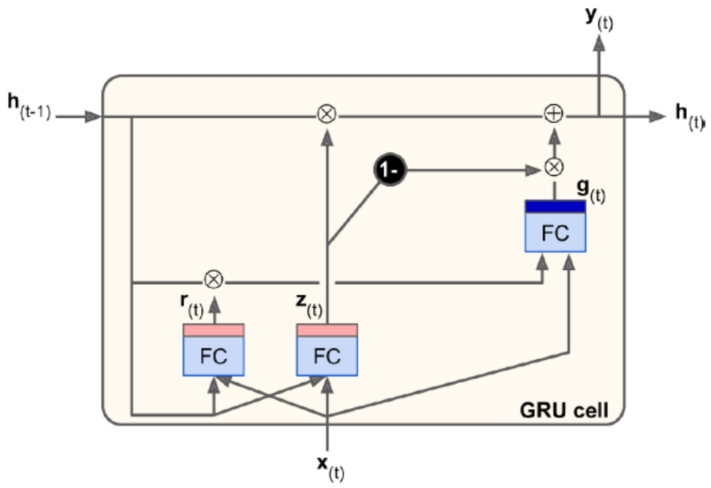


In Keras use LSTM layer, not RNN layer
with LSTM cell, to take advantage of GPU

SHORT TERM MEMORY

Gated Recurrent Unit (GRU) Cell

- Simplified LSTM but performs just as well
- Just a single hidden state
- Combines Forget Gate and Input Gate, No Output Gate
- New Remember gate $r_{(t)}$: which parts of previous state to show to Main Layer
- Forget + Input gate $z_{(t)}$: which parts of previous state to add to prediction, but erases storage location first

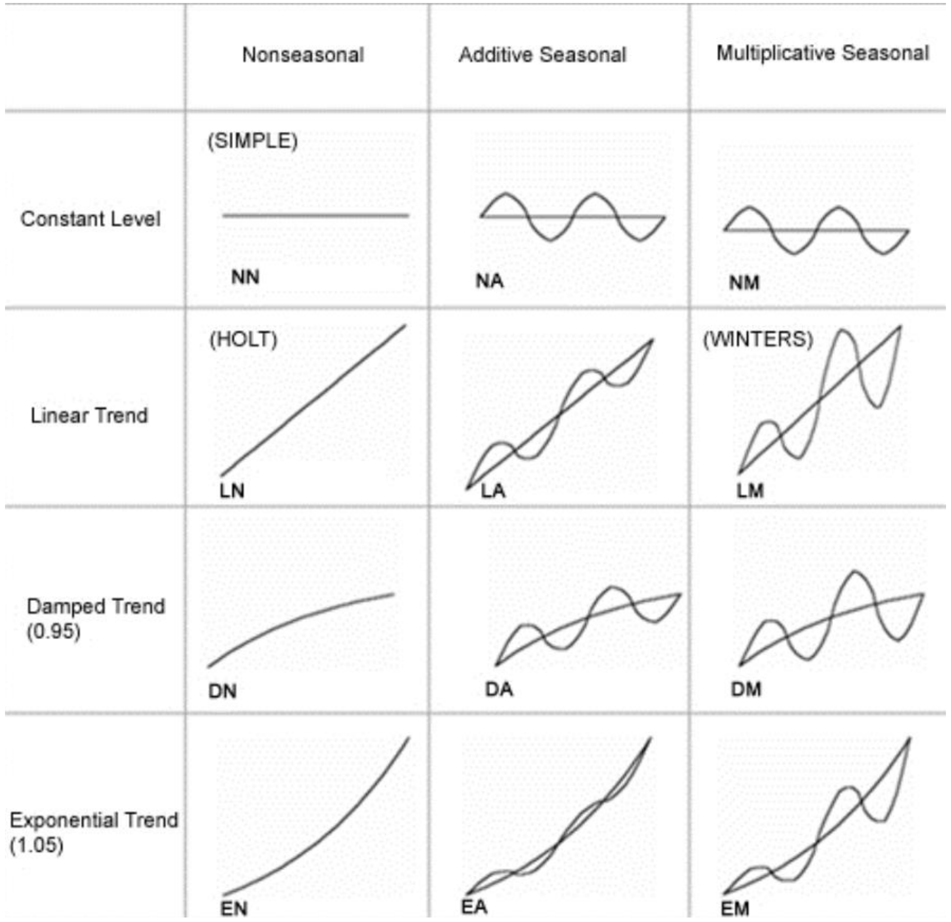


TREND AND SEASONALITY

Remove trend

Remove seasonality

General best practice, but you don’t have to – however will likely take longer to train and it *may not* improve accuracy



PERFORMANCE SUMMARY

Model	MSE
Naïve Forecasting	0.020
Linear Regression	0.004
Single Cell RNN	0.014
Simple RNN (2 Layer 20 Node)	0.003
WaveNet (3 * 10 layer 1D CNN)	< 0.003

In this case – WaveNet performed better than RNNs

OTHER MODELS

Weighted Moving Average (WMA)

Autoregressive Integrated Moving Average (ARIMA)

WaveNet – Convolutional Neural Network (CNN)

But field is evolving

- LSTM outperforms ARIMA for time series (2018): https://par.nsf.gov/servlets/purl/10186768
- Transformers outperform LSTM for time series (2021): https://medium.com/mlearning-ai/transformer-implementation-for-time-series-forecasting-a9db2db5c820

APPLICATIONS

Time Series

- Stock Price Predictions
- Speech Recognition
- Natural Language Processing
- Medical Event Prediction
- River Water Level Prediction (where influenced by tides)

Other Series

- Generating Image Captions
- Human Activity Recognition (aka moving posture recognition) (https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition)
- Chinese Handwriting Recognition (https://ieeexplore.ieee.org/document/7333746)

THANKS

SAN DIEGO MACHINE LEARNING

2022 FEB 19

Chapter 15: Processing Sequences using RNNs (& CNNs)

Discussion led by
Steven Fouskarinis