THE OHIO STATE UNIVERSITY

THE OHIO STATE UNIVERSITY

ISE 5194: Long Short Term Memory Neural Networks

Advisor: Theodore Allen

Outline

- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

Outline

- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

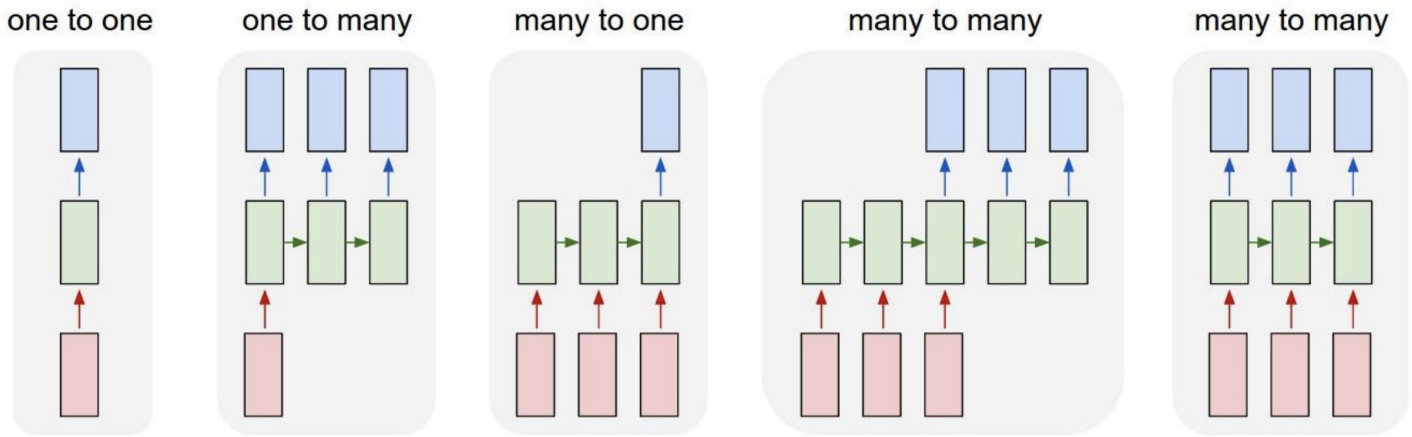
Recurrent Neural Networks (RNNs)

- What happens if we want to analyze dynamic data? What about videos, voice recognition or sequences of text?
- The most popular method of performing classification and other analysis on sequences of data is Recurrent Neural Networks (RNNs).
- RNNs were based on David Rumelhart's work in 1986.

Recurrent Neural Networks (RNNs)

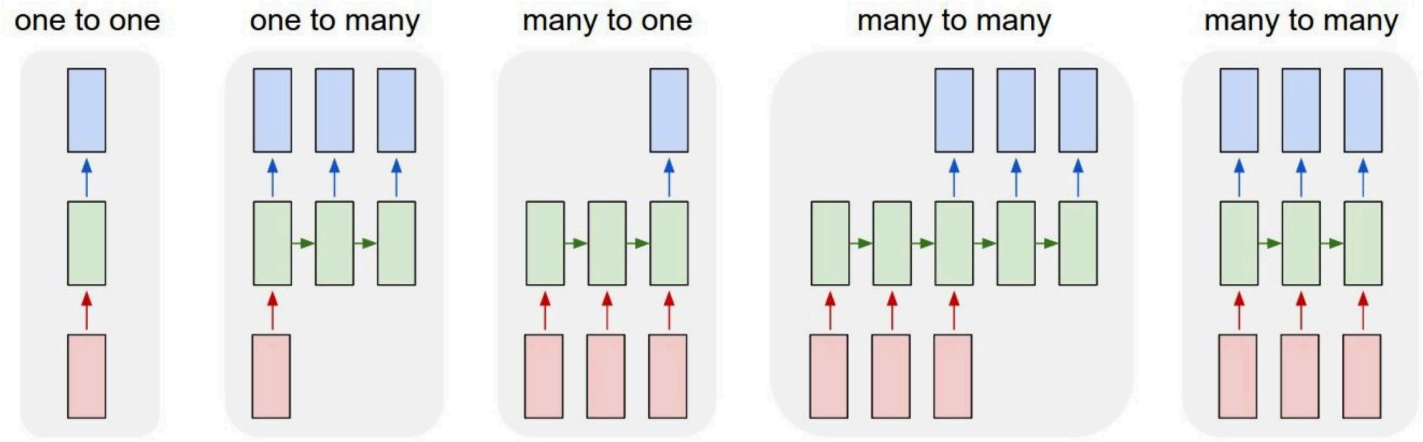
- RNNs can work on sequences of arbitrary lengths, rather than on fixed-sized inputs
- They can analyze time series data such as stock prices, and tell you when to buy or sell.
- In autonomous driving systems, they can anticipate car trajectories and help avoid accidents.
- They can take sentences, documents, or audio samples as input, making them extremely useful for natural language processing (NLP) systems such as automatic translation, speech-to-text, or sentiment analysis (e.g., reading movie reviews and extracting the rater's feeling about the movie).

Recurrent Neural Networks (RNNs)



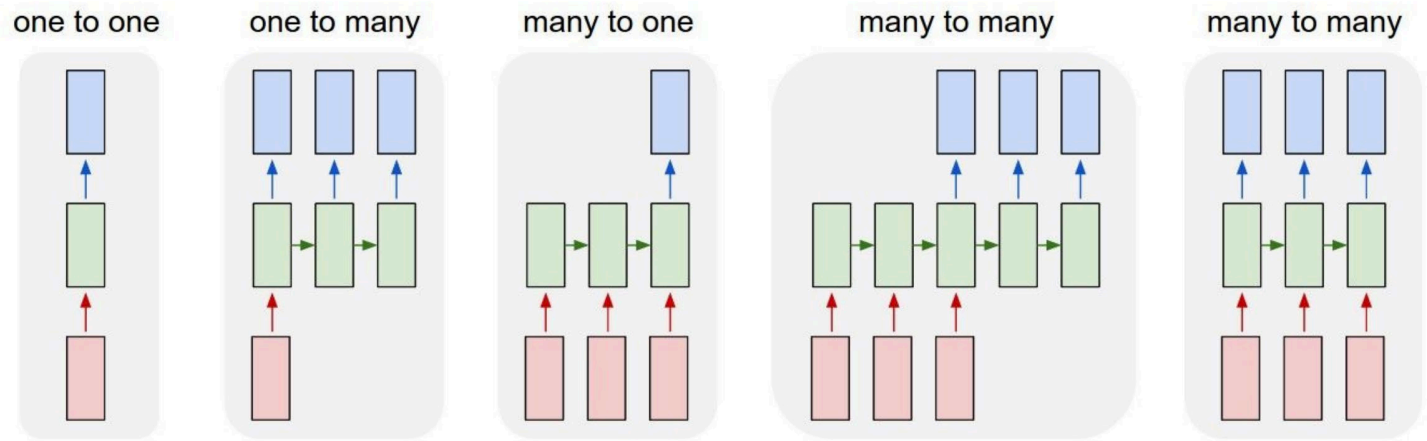
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks (RNNs)



e.g. **Sentiment Classification**
sequence of words -> sentiment

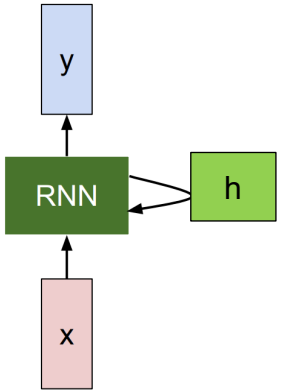
Recurrent Neural Networks (RNNs)



e.g. **Machine Translation**
seq of words -> seq of words

Elements of a Simple RNN

- Input layer: x with weights W_{hx}.
- Hidden, recursive layer (feeds back into itself): h with weights W_{hh}.
- Output layer: y with weight W_{hy}.



$$h_t = f_W(h_{t-1}, x_t)$$

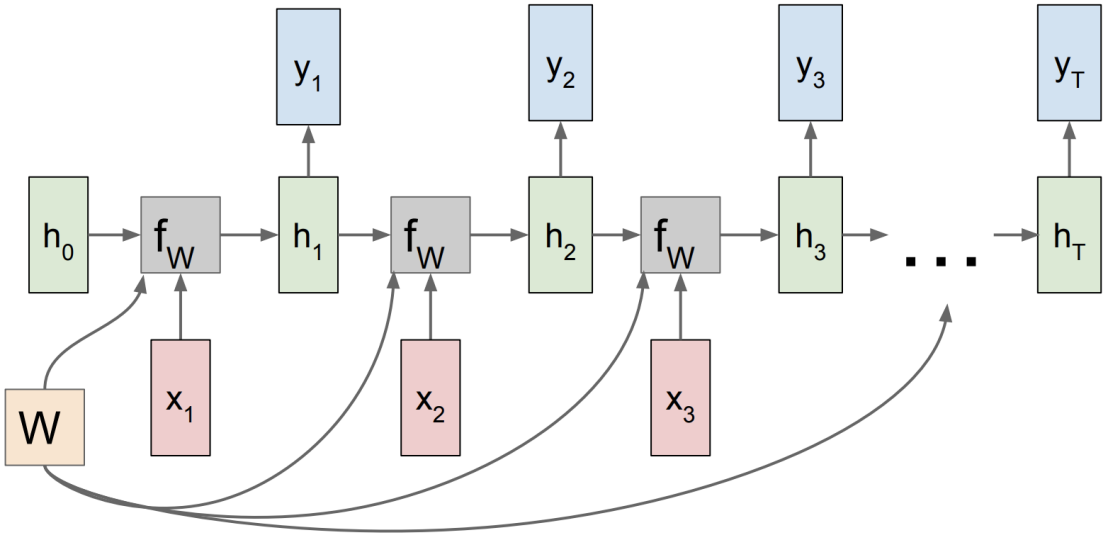
$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Unrolling the Recursion

- We can see how this is applicable to sequence data when unrolling the recursion:

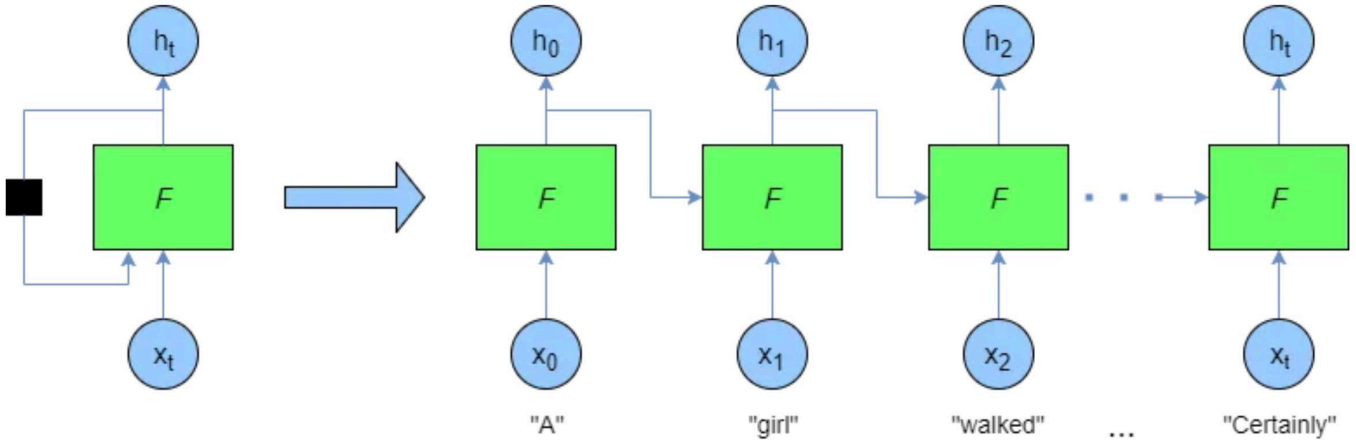


Example: predicting text sequences

- Consider the following text string: “A girl walked into a bar, and she said ‘Can I have a drink please?’. The bartender said ‘Certainly { }”.
- There are many options for what could fill in the { } symbol in the above string. For instance, “miss”, “ma’am” and so on. However, other words could also fit, such as “sir”, “Mister” etc.
- In order to get the correct gender of the noun, the neural network needs to “recall” that two previous words designating the likely gender (i.e. “girl” and “she”) were used.

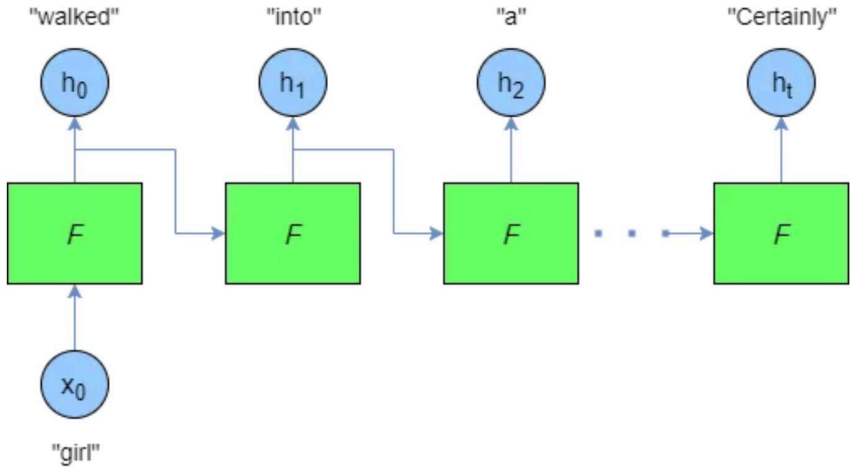
Example: predicting text sequences

- This type of flow of information through time (or sequence) in a recurrent neural network is shown in the diagram below, which unrolls the sequence:



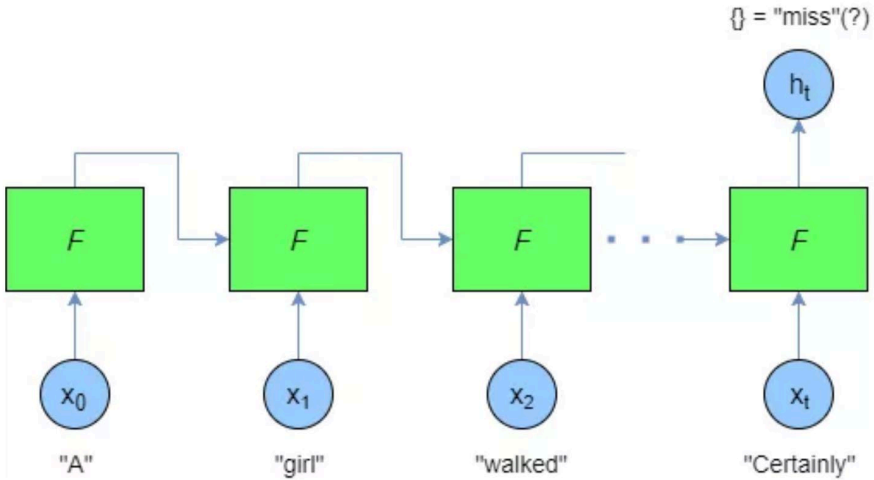
Example: predicting text sequences

- One-to-many model: supplying one input, say “girl” and predicting multiple outputs h_0 to h_t (i.e. trying to generate sentences based on a single starting word).



Example: predicting text sequences

- Many-to-one model: supplying many words as input, like the sentence “A girl walked into a bar, and she said ‘Can I have a drink please?’. The bartender said ‘Certainly { }” and predicting the next word i.e. { }.



Outline

- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

Vanishing / Exploding Gradient Problem

- Training the RNN means: Perform backpropagation to find optimal weights.
- Need to minimize the error (or loss) function E with regards to parameter θ.
- The backpropagation algorithm works by going from the output layer to the input layer, propagating the error gradient on the way.
- Once the algorithm has computed the gradient of the cost function with regards to each parameter in the network, it uses these gradients to update each parameter with a Gradient Descent step.

Vanishing / Exploding Gradient Problem

- Unfortunately, gradients often get smaller and smaller as the algorithm progresses down to the lower layers. As a result, the Gradient Descent update leaves the lower layer connection weights virtually unchanged, and training never converges to a good solution. This is called the vanishing gradients problem.
- In some cases, the opposite can happen: the gradients can grow bigger and bigger, so many layers get insanely large weight updates and the algorithm diverges. This is the exploding gradients problem.
- *Impact of vanishing gradients to RNN: Can't "remember" impacts of long sequences.*

Outline

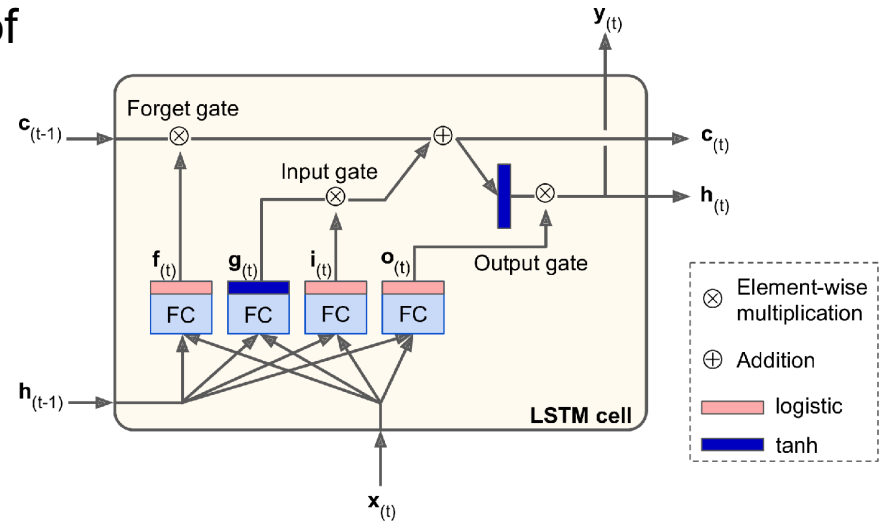
- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

Long Short Term Memory (LSTM)

- To reduce the vanishing (and exploding) gradient problem, and therefore allow deeper networks and recurrent neural networks to perform well in practical settings, there needs to be a way to reduce the multiplication of gradients which are less than zero.
- The Long Short-Term Memory (LSTM) cell was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber, and it was gradually improved over the years by several researchers, such as Alex Graves, Haşim Sak, Wojciech Zaremba, and many more.

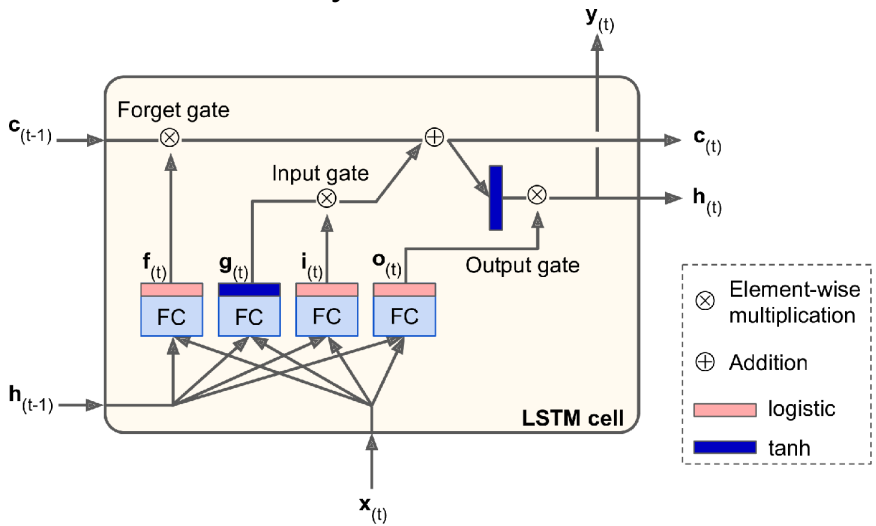
Long Short Term Memory

- Variant of RNNs that introduce a number of special, internal gates.
- Internal gates help with the problem of learning relationships between both long and short sequences in data.
- If you consider the LSTM cell as a black box, it can be used very much like a basic cell, except it will perform much better; training will converge faster and it will detect long-term dependencies in the data.



Long Short Term Memory

- You can think of $\mathbf{h}_{(t)}$ as the short-term state and $\mathbf{c}_{(t)}$ as the long-term state.
- The key idea is that the network can learn what to store in the long-term state, what to throw away, and what to read from it.



$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma\left(\mathbf{W}_{xi}{}^{\intercal}\mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{\intercal}\mathbf{h}_{(t-1)} + \mathbf{b}_i\right) \\
\mathbf{f}_{(t)} &= \sigma\left(\mathbf{W}_{xf}{}^{\intercal}\mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{\intercal}\mathbf{h}_{(t-1)} + \mathbf{b}_f\right) \\
\mathbf{o}_{(t)} &= \sigma\left(\mathbf{W}_{xo}{}^{\intercal}\mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{\intercal}\mathbf{h}_{(t-1)} + \mathbf{b}_o\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}{}^{\intercal}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{\intercal}\mathbf{h}_{(t-1)} + \mathbf{b}_g\right) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\left(\mathbf{c}_{(t)}\right)
\end{aligned}
$$

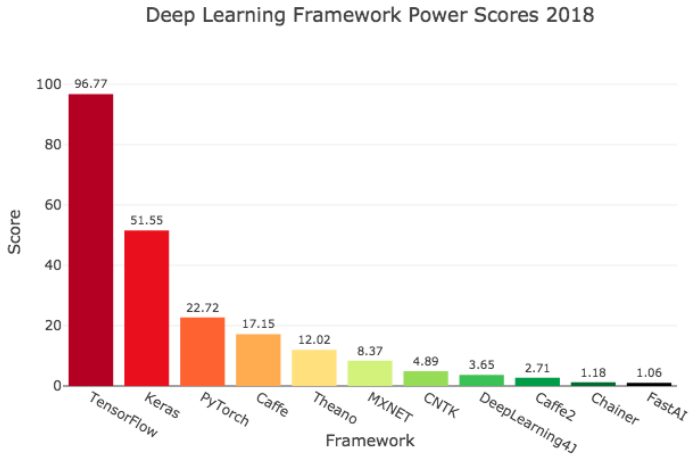
- More details: “Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow”.

Outline

- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

Keras: the Python Deep Learning Library

- Consistent and simple high-level APIs for Deep Learning in Python.
- Focus on getting stuff done w/o having to write tons of lines of code.
- Has abstraction layer for multiple Deep Learning backends: Tensorflow, CNTK, Theano.



Keras: Runs Everywhere

- On iOS, via Apple's CoreML (Keras support officially provided by Apple).
- On Android, via the TensorFlow Android runtime.
- In the browser, via GPU-accelerated JavaScript runtimes such as Keras.js and WebDNN.
- On Google Cloud, via TensorFlow-Serving.
- In a Python webapp backend (such as a Flask app).
- On the JVM, via DL4J model import provided by SkyMind.

- Keras documentation: https://keras.io/

Outline

- Recurrent Neural Networks (RNN)
- Vanishing / Exploding Gradient Problem
- Long Short Term Memory (LSTM)
- Keras
- Examples in Python

Examples in Python

- Before you run the code, you have to install Keras and Tensorflow. Please follow the steps in https://keras.io/