Encoder-Decoder Long Short-Term Memory Networks

by **Jason Brownlee** on August 23, 2017 in **Long Short-Term Memory Networks**

Gentle introduction to the Encoder-Decoder LSTMs for sequence-to-sequence prediction with example Python code.

The Encoder-Decoder LSTM is a recurrent neural network designed to address sequence-to-sequence problems, sometimes called seq2seq.

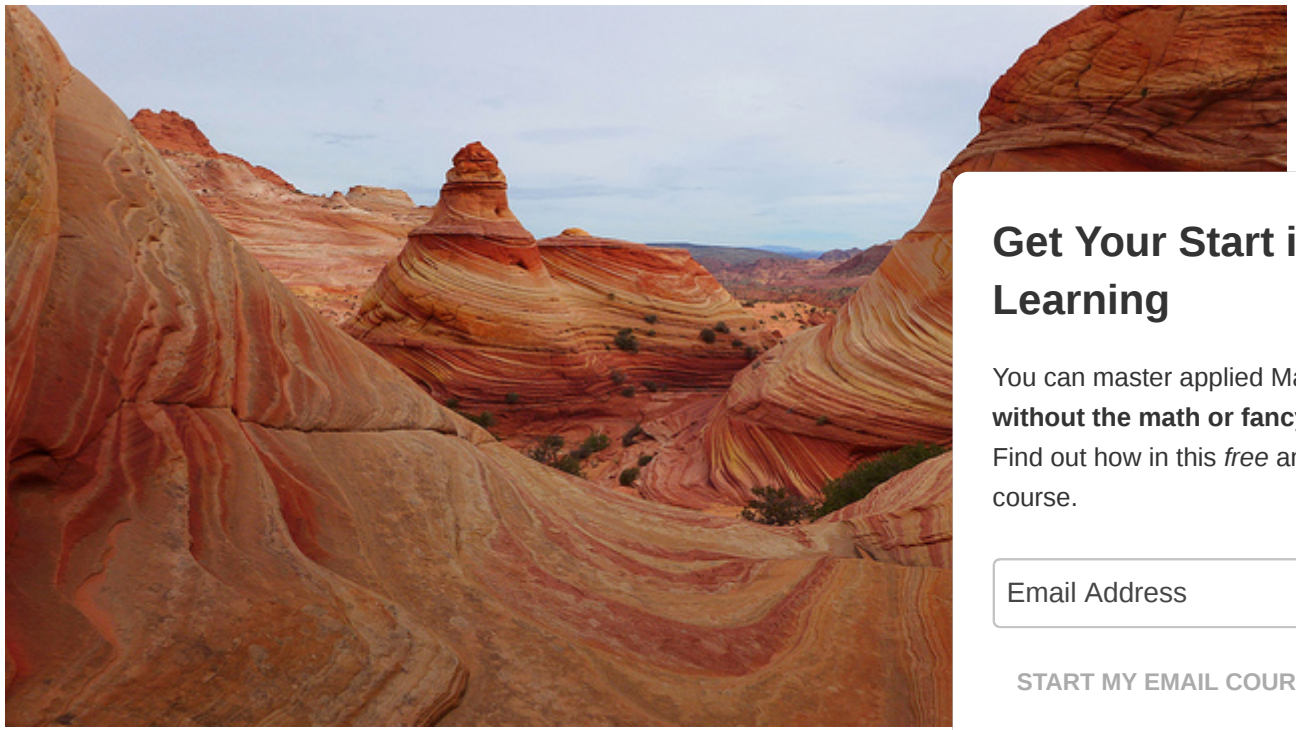
Sequence-to-sequence prediction problems are challenging because the number of items in the input and output sequences can vary. For example, text translation and learning to execute programs are examples of seq2seq problems.

In this post, you will discover the Encoder-Decoder LSTM architecture for sequence-to-sequence pr

After completing this post, you will know:

- The challenge of sequence-to-sequence prediction.
- The Encoder-Decoder architecture and the limitation in LSTMs that it was designed to address.
- How to implement the Encoder-Decoder LSTM model architecture in Python with Keras.

Let's get started.



Encoder-Decoder Long Short-Term Memory Networks
Photo by slashvee, some rights reserved.

Sequence-to-Sequence Prediction Problems

Sequence prediction often involves forecasting the next value in a real valued sequence or outputting a class label for an input sequence.

Get Your Start in Machine Learning

This is often framed as a sequence of one input time step to one output time step (e.g. one-to-one) or multiple input time steps to one output time step (many-to-one) type sequence prediction problem.

There is a more challenging type of sequence prediction problem that takes a sequence as input and requires a sequence prediction as output. These are called sequence-to-sequence prediction problems, or seq2seq for short.

One modeling concern that makes these problems challenging is that the length of the input and output sequences may vary. Given that there are multiple input time steps and multiple output time steps, this form of problem is referred to as many-to-many type sequence prediction problem.

Encoder-Decoder LSTM Architecture

One approach to seq2seq prediction problems that has proven very effective is called the Encoder-D

This architecture is comprised of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. The use of the models in concert gives the architecture its name of Encoder-Decoder LSTM designed specifically for seq2seq problems.

> *… RNN Encoder-Decoder, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence.*

— Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

The Encoder-Decoder LSTM was developed for natural language processing problems where it demonstrated state-of-the-art performance, specifically in the area of text translation called statistical machine translation.

The innovation of this architecture is the use of a fixed-sized internal representation in the heart of the model that input sequences are read to and output sequences are read from. For this reason, the method may be referred to as sequence embedding.

In one of the first applications of the architecture to English-to-French translation, the internal representation of the encoded English phrases was visualized. The plots revealed a qualitatively meaningful learned structure of the phrases harnessed for the translation task.

> *The proposed RNN Encoder-Decoder naturally generates a continuous-space representation of a phrase. [...] From the visualization it is clear that the RNN Encoder-Decoder captures both semantic and syntactic structures of the phras*

— Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translatio

On the task of translation, the model was found to be more effective when the input sequence was r effective even on very long input sequences.

> *We were able to do well on long sentences because we reversed the order of words in the so training and test set. By doing so, we introduced many short term dependencies that made th simple trick of reversing the words in the source sentence is one of the key technical contribu*

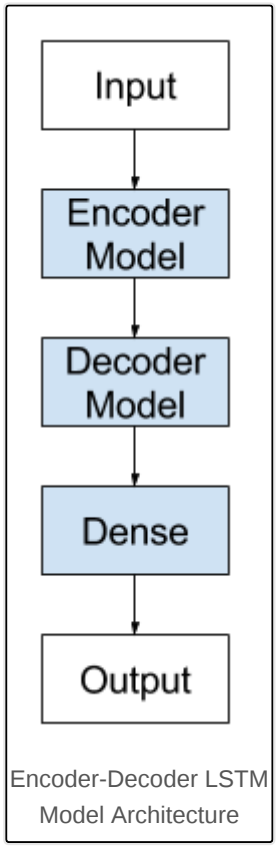
— Sequence to Sequence Learning with Neural Networks, 2014.

This approach has also been used with image inputs where a Convolutional Neural Network is used as a feature extractor on input images, which is then read by a decoder LSTM.

> *… we propose to follow this elegant recipe, replacing the encoder RNN by a deep convolution neural network (CNN). […] it is natural to use a CNN as an image encoder", by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences*

Get Your Start in Machine Learning

— Show and Tell: A Neural Image Caption Generator, 2014.



Encoder-Decoder LSTM Model Architecture

Applications of Encoder-Decoder LSTMs

The list below highlights some interesting applications of the Encoder-Decoder LSTM architecture.

- Machine Translation, e.g. English to French translation of phrases.
- Learning to Execute, e.g. calculate the outcome of small programs.
- Image Captioning, e.g. generating a text description for images.
- Conversational Modeling, e.g. generating answers to textual questions.
- Movement Classification, e.g. generating a sequence of commands from a sequence of gestures

Get Your Start in Machine Learning

Implement Encoder-Decoder LSTMs in Keras

The Encoder-Decoder LSTM can be implemented directly in the Keras deep learning library.

We can think of the model as being comprised of two key parts: the encoder and the decoder.

First, the input sequence is shown to the network one encoded character at a time. We need an encoding level to learn the relationship between the steps in the input sequence and develop an internal representation of these relationships.

One or more LSTM layers can be used to implement the encoder model. The output of this model is a fixed-size vector that represents the internal representation of the input sequence. The number of memory cells in this layer defines the length of this fixed-sized vector.

```
model = Sequential()
model.add(LSTM(..., input_shape=(...)))
```

The decoder must transform the learned internal representation of the input sequence into the corre

One or more LSTM layers can also be used to implement the decoder model. This model reads from

As with the Vanilla LSTM, a Dense layer is used as the output for the network. The same weights ca
sequence by wrapping the Dense layer in a TimeDistributed wrapper.

```
model.add(LSTM(..., return_sequences=True))
model.add(TimeDistributed(Dense(...)))
```

There's a problem though.

We must connect the encoder to the decoder, and they do not fit.

That is, the encoder will produce a 2-dimensional matrix of outputs, where the length is defined by the number of memory cells in the layer. The decoder is an LSTM layer that expects a 3D input of [samples, time steps, features] in order to produce a decoded sequence of some different length defined by the problem.

If you try to force these pieces together, you get an error indicating that the output of the decoder is 2D and 3D input to the decoder is required.

We can solve this using a RepeatVector layer. This layer simply repeats the provided 2D input multip

The RepeatVector layer can be used like an adapter to fit the encoder and decoder parts of the network together. We can configure the RepeatVector to repeat the fixed length vector one time for each time step in the output sequence.

```
model.add(RepeatVector(...))
```

Putting this together, we have:

```
model = Sequential()
model.add(LSTM(..., input_shape=(...)))
model.add(RepeatVector(...))
model.add(LSTM(..., return_sequences=True))
model.add(TimeDistributed(Dense(...)))
```

To summarize, the RepeatVector is used as an adapter to fit the fixed-sized 2D output of the encoder to the differing length and 3D input expected by the decoder. The TimeDistributed wrapper allows the same output layer to be reused for each element in the output sequence.

Further Reading

This section provides more resources on the topic if you are looking go deeper.

Papers

- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translati
- Sequence to Sequence Learning with Neural Networks, 2014.
- Show and Tell: A Neural Image Caption Generator, 2014.
- Learning to Execute, 2015.
- A Neural Conversational Model, 2015.

Keras API

- RepeatVector Keras API.
- TimeDistributed Keras API.

Posts

- How to use an Encoder-Decoder LSTM to Echo Sequences of Random Integers

- Learn to Add Numbers with an Encoder-Decoder LSTM Recurrent Neural Network
- Attention in Long Short-Term Memory Recurrent Neural Networks

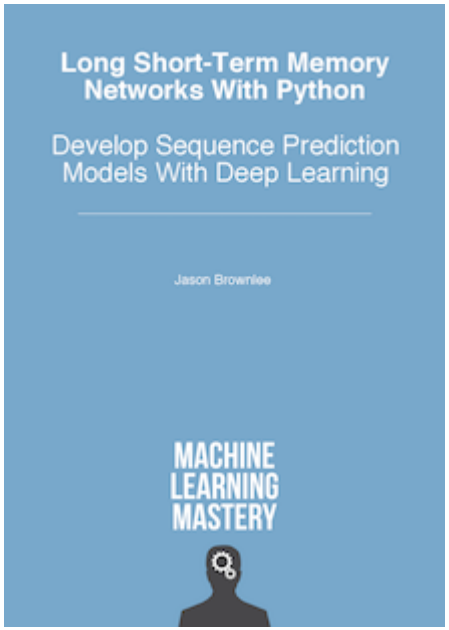
Summary

In this post, you discovered the Encoder-Decoder LSTM architecture for sequence-to-sequence prediction

Specifically, you learned:

- The challenge of sequence-to-sequence prediction.
- The Encoder-Decoder architecture and the limitation in LSTMs that it was designed to address.
- How to implement the Encoder-Decoder LSTM model architecture in Python with Keras.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.



About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional develo to helping developers get started and get good at applied machine learning. Learn more.
View all posts by Jason Brownlee →

‹ CNN Long Short-Term Memory Networks

Gentle Introduction to Generative Long Short-Term Memory Networks ›

6 Responses to *Encoder-Decoder Long Short-Term Memory Networks*

Valeriy August 25, 2017 at 6:40 am #

REPLY

Hi Jason

After applying model.add(TimeDistributed(Dense(…))) what kind of output we will receive?

Regards

Jason Brownlee August 25, 2017 at 6:46 am #

Great question!

You will get the output from the Dense at each time step that the wrapper receives as input.

Valeriy August 25, 2017 at 4:44 pm #

Many thanks

That means we will receive a fixed length output as well?
Given an input sequence of the same size but different meaning?

For example
I like this – input_1
and
I bought car – input_2

Will end up as 2 sequences of the same length in any case?

Jason Brownlee August 26, 2017 at 6:37 am #

REPLY

No, the input and output sequences can be different lengths.

The RepeatVector allows you to specify the fixed-length of the output vector, e.g. the number of times to repeat the fixed-length encoding of the input sequence.

Yoel August 25, 2017 at 7:26 pm #

REPLY

Hi Jason, I had a great time reading your post.

One question: a common method to implement seq2seq is to use the encoder's output as the initial valu
outputs is then fed back as input to the decoder.

In your method you do the other way around, where the encoder's output is fed as input at each time ste

I wonder if you tried both methods, and if you did – which one worked better for you?

Thanks

Jason Brownlee August 26, 2017 at 6:44 am #

Thanks Yoel. Great question!

I have tried the other method, and I had to contrive the implementation in Keras (e.g. it was not natu
attention. I hope to cover this in more detail in the future.

As for better, I'm not sure. The simple approach above can get you a long way for seq2seq applications in my experience.

Leave a Reply

Get Your Start in Machine Learning

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

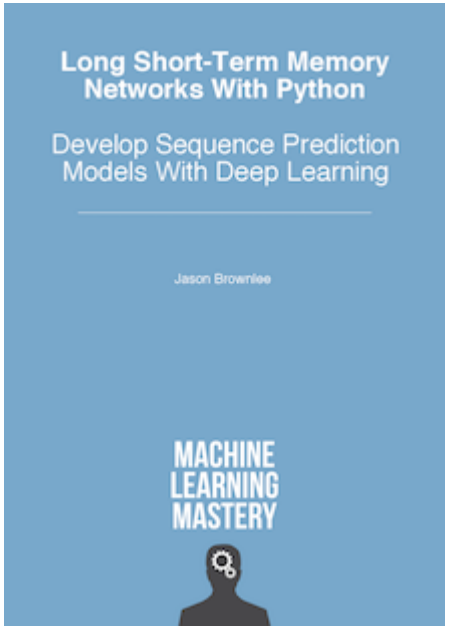
Read More

Deep Learning for Sequence Prediction

Cut through the math and research papers.
Discover 4 Models, 6 Architectures, and 14 Tutorials.

Get Your Start in Machine Learning

Get Started With LSTMs in Python Today!



POPULAR


Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
JULY 21, 2016


Your First Machine Learning Project in Python Step-By-Step
JUNE 10, 2016


Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016

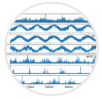

Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

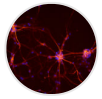
Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016



Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017



How to Implement the Backpropagation Algorithm From Scratch In Python
NOVEMBER 7, 2016

Privacy | Contact | About