What is Teacher Forcing for Recurrent Neural Networks?

by **Jason Brownlee** on December 6, 2017 in **Long Short-Term Memory Networks**

Last Updated on August 14, 2019

Teacher forcing is a method for quickly and efficiently training recurrent neural network models that use the ground truth from a prior time step as input.

It is a network training method critical to the development of deep learning language models used in machine translation, text summarization, and image captioning, among many other applications.

In this post, you will discover the teacher forcing as a method for training recurrent neural networks.

After reading this post, you will know:

- The problem with training recurrent neural networks that use output from prior time steps as input.
- The teacher forcing method for addressing slow convergence and instability when training these types of recurrent networks.
- Extensions to teacher forcing that allow trained models to better handle open loop applications of this type of network.

Kick-start your project with my new book Long Short-Term Memory Networks With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

What is Teacher Forcing for Recurrent Neural Networks?
Photo by Nathan Russell, some rights reserved.

Using Output as Input in Sequence Prediction

There are sequence prediction models that use the output from the last time step y(t-1) as input for the model at the current time step X(t).

This type of model is common in language models that output one word at a time and use the output word as input for generating the next word in the sequence.

For example, this type of language model is used in an Encoder-Decoder recurrent neural network architecture for sequence-to-sequence generation problems such as:

- Machine Translation
- Caption Generation
- Text Summarization

After the model is trained, a “start-of-sequence” token can be used to start the process and the generated word in the output sequence is used as input on the subsequent time step, perhaps along with other input like an image or a source text.

This same recursive output-as-input process can be used when training the model, but it can result in problems such as:

- Slow convergence.
- Model instability.
- Poor skill.

Teacher forcing is an approach to improve model skill and stability when training these types of models.

What is Teacher Forcing?

Teacher forcing is a strategy for training recurrent neural networks that uses model output from a prior time step as an input.

> *Models that have recurrent connections from their outputs leading back into the model may be trained with teacher forcing.*

— Page 372, Deep Learning, 2016.

The approach was originally described and developed as an alternative technique to backpropagation through time for training a recurrent neural network.

> *An interesting technique that is frequently used in dynamical supervised learning tasks is to replace the actual output y(t) of a unit by the teacher signal d(t) in subsequent computation of the behavior of the network, whenever such a value exists. We call this technique teacher forcing.*

— A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, 1989.

Teacher forcing works by using the actual or expected output from the training dataset at the current time step y(t) as input in the next time step X(t+1), rather than the output generated by the network.

> *Teacher forcing is a procedure […] in which during training the model receives the ground truth output y(t) as input at time t + 1.*

— Page 372, Deep Learning, 2016.

Worked Example

Let's make teacher forcing concrete with a short worked example.

Given the following input sequence:

```
1 Mary had a little lamb whose fleece was white as snow
```

Imagine we want to train a model to generate the next word in the sequence given the previous sequence of words.

First, we must add a token to signal the start of the sequence and another to signal the end of the sequence. We will use "*[START]*" and "*[END]*" respectively.

```
1 [START] Mary had a little lamb whose fleece was white as snow [END]
```

Next, we feed the model "*[START]*" and let the model generate the next word.

Imagine the model generates the word "*a*", but of course, we expected "*Mary*".

```
X,                  yhat
[START],            a
```

Naively, we could feed in "*a*" as part of the input to generate the subsequent word in the sequence.

```
X,                  yhat
[START], a,         ?
```

You can see that the model is off track and is going to get punished for every subsequent word it generates. This makes learning slower and the model unstable.

Instead, we can use teacher forcing.

In the first example when the model generated "*a*" as output, we can discard this output after calculating error and feed in "*Mary*" as part of the input on the subsequent time step.

```
X,                  yhat
[START], Mary,      ?
```

We can then repeat this process for each input-output pair of words.

```
X,                  yhat
[START],            ?
[START], Mary,      ?
[START], Mary, had, ?
[START], Mary, had, a, ?
...
```

The model will learn the correct sequence, or correct statistical properties for the sequence, quickly.

Extensions to Teacher Forcing

Teacher forcing is a fast and effective way to train a recurrent neural network that uses output from prior time steps as input to the model.

But, the approach can also result in models that may be fragile or limited when used in practice when the generated sequences vary from what was seen by the model during training.

This is common in most applications of this type of model as the outputs are probabilistic in nature. This type of application of the model is often called open loop.

> *Unfortunately, this procedure can result in problems in generation as small prediction error compound in the conditioning context. This can lead to poor prediction performance as the RNN's conditioning context (the sequence of previously generated samples) diverge from sequences seen during training.*

– Professor Forcing: A New Algorithm for Training Recurrent Networks, 2016.

There are a number of approaches to address this limitation, for example:

Search Candidate Output Sequences

One approach commonly used for models that predict a discrete value output, such as a word, is to perform a search across the predicted probabilities for each word to generate a number of likely candidate output sequences.

This approach is used on problems like machine translation to refine the translated output sequence.

A common search procedure for this post-hoc operation is the beam search.

> *This discrepancy can be mitigated by the use of a beam search heuristic maintaining several generated target sequences*

— Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, 2015.

Curriculum Learning

The beam search approach is only suitable for prediction problems with discrete output values and cannot be used for real-valued outputs.

A variation of forced learning is to introduce outputs generated from prior time steps during training to encourage the model to learn how to correct its own mistakes.

> *We propose to change the training process in order to gradually force the model to deal with its own mistakes, as it would have to during inference.*

— Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, 2015.

The approach is called curriculum learning and involves randomly choosing to use the ground truth output or the generated output from the previous time step as input for the current time step.

The curriculum changes over time in what is called scheduled sampling where the procedure starts at forced learning and slowly decreases the probability of a forced input over the training epochs.

There are also other extensions and variations of teacher forcing and I encourage you to explore them if you are interested.

Further Reading

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

Papers

- A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, 1989.
- Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, 2015.
- Professor Forcing: A New Algorithm for Tra

Books

- Section 10.2.1, Teacher Forcing and Networks with Output Recurrence, Deep Learning, 2016.

Summary

In this post, you discovered teacher forcing as a method for training recurrent neural networks that use output from a previous time step as input.

Specifically, you learned:

- The problem with training recurrent neural networks that use output from prior time steps as input.
- The teacher forcing method for addressing slow convergence and instability when training these types of recurrent networks.
- Extensions to teacher forcing that allow trained models to better handle open loop applications of this type of network.

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

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learni

View all posts by Jason Brownlee →

‹ How to Prepare News Articles for Text Summarization

Encoder-Decoder Models for Text Summarization in Keras ›

40 Responses to *What is Teacher Forcing for Recurrent Neural Networks?*

Huzefa Calcuttawala March 14, 2018 at 10:24 pm #

REPLY

Hi Jason,

Thanks for such as informative posts. Does current version of Keras support ‘teacher forcing’ ? I know recurrent shop can be used to do that but how to use that in Keras?

Jason Brownlee March 15, 2018 at 6:30 am #

REPLY

Yes, I give examples in a photo captioning example.

jundong May 3, 2018 at 5:25 am #

REPLY

HI Jason,

Thank you for your post!
Currently, I am learning CNN-LSTM, LSTM encoder-decoder according to the chapter 8 and 9 in your book “Long Short-Term Memory Networks with Python”.

I have a task mapping a sequence of 2D inputs to a sequence of classification. It requires a network CNN-LSTM-encoder-decoder, and I combine the two examples together as below:

```
def cnn_lstm(lmda1, lmda2):

model = Sequential()

# CNN module
model.add(TimeDistributed(Conv2D(filters = 8,
kernel_size = (2, 2),
padding = ‘same’,
activation=’relu’,
kernel_regularizer = regularizers.l1_l2(lmda1, lmda2),
name = ‘Conv_1′),
input_shape = (None, img_height, img_width, channels)))
model.add(TimeDistributed(BatchNormalization(axis=1, name=’BN_1’)))
model.add(TimeDistributed(MaxPooling2D(pool
```

```
model.add(TimeDistributed(Conv2D(filters = 16,
kernel_size = (2, 2),
padding = 'same',
activation='relu',
kernel_regularizer = regularizers.l1_l2(lmda1, lmda2),
name = 'Conv_2')))
model.add(TimeDistributed(BatchNormalization(name='BN_2')))
model.add(TimeDistributed(MaxPooling2D(pool_size = pool_size)))

# Flatten all features from CNN before inputing them into encoder-decoder LSTM
model.add(TimeDistributed(Flatten()))

# LSTM module
# encoder
model.add(LSTM(50, name = 'encoder'))
model.add(RepeatVector(n_out_seq_length))

# decoder
model.add(LSTM(50, return_sequences=True, name = 'decoder'))
model.add(TimeDistributed(Dense(nb_classes, activation='softmax')))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

return model
```

Do you think it is the correct way to do it? Thank you very much!

Jason Brownlee May 3, 2018 at 6:38 am #

REPLY

Perhaps try a few different approaches and see which results in best skill on your dataset.

Skye May 29, 2018 at 1:35 am #

REPLY

Hi Jason,

Thank a lot for your share!

I want to ask if the teacher forcing method performs bad on multistep forecasting problem? Because at predicting stage some of them cannot access the ground truth value of the previous value?

Jason Brownlee May 29, 2018 at 6:28 am #

REPLY

Teach forcing is only used during training.

Skye May 29, 2018 at 10:46 am #

REPLY

Oh I get it.

So how can I deal with it if we want to use it in practice? Does teach forcing not have practical meaning?

REPLY

Jason Brownlee May 29, 2018 at 2:52 pm #

Sorry, I don't follow. What is the problem you are having exactly?

Teacher forcing is used to help to keep the model on track during training.

REPLY

Skye May 29, 2018 at 3:34 pm #

I use teacher forcing to train a seq2seq time series problem and get a low loss on training and validation dataset, but get a poor result on test dataset. Is it normal to have a bad result on test dataset?

REPLY

Jason Brownlee May 30, 2018 at 6:31 am #

Ideally, you want good skill on train and test sets.

Poor skill on a test with good skill on the training set suggests overfitting.

REPLY

max June 24, 2018 at 9:14 pm #

https://arxiv.org/pdf/1409.3215.pdf is the paper from sutskever describing teacher forcing ? I think not or am I wrong, your implementation here is also with https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/ teacher forcing ?

REPLY

Jason Brownlee June 25, 2018 at 6:21 am #

Yes, I almost always use teacher forcing.

REPLY

Igor Aherne August 23, 2018 at 9:25 am #

Hey Jason, thank you for the post!

I have several questions:

1. for Curriculum Learning, do we decide to teacher-force once per entire episode, or at every timestep?

2. Assuming I were to use 100% pure teacher forcing while training my LSTM, how should I deal with the gradient that supposed to flow from 'Cell_{t+1}' to 'Cell_t' ? In other words, what is the gradient that arrives into Cell_t?

As I understood, teacher forcing made us plug-in the Cell values during fwd prop. During Backprop, do we use original value of Cell_t (pretending there never was a swap), and how is that possible to be combined with the gradient from Cell_{t+1}? Especially during Curriculum learning where at Cell_{t+1} or Cell_{t+2} we "played fair" and never swapped anything. (if Question 1 was true)

3. "Teacher forcing is a fast and effective way to train a recurrent neural network that uses output from prior time steps as input to the model. But, the approach can also result in models that may be fragile or limited when used in practice when the generated sequences vary from what was seen by the model during training."

When you used Curriculum Learning in the past, did you still get slightly fragile networks, or they were just as strong as your non-forced networks? In particular LSTM.

If I use Curriculum Learning, will I be safe while enjoying the speed-ups during training?

4. From experience, how much faster does the training go?

Thanks! 🙂

pierre November 2, 2018 at 1:09 pm #

REPLY

Thanks for the post.
I have trained my seq2seq model with teacher-forcing. My question is do I also have to compute the validation loss and ppl with the teacher-forcing?
I compute the validation loss without teacher-forcing and it remains almost the same (it decreases a bit until a certain point and stops and I am sure the point it stops is not an overfitting point) and it is generally much larger than my training loss (it is almost in the range of the training loss of the first epoch).

Jason Brownlee November 2, 2018 at 2:52 pm #

REPLY

This is an implementation detail that really depends on your code and how you've prepared your data.

What problem are you having exactly?

pierre November 2, 2018 at 6:34 pm #

REPLY

I am looking for the way to compute the validation loss. Is that necessary to compute the validation loss in an exact manner as we do in training?

Jason Brownlee November 3, 2018 at 7:01 am #

REPLY

Sure.

Nick March 20, 2019 at 2:45 pm #

REPLY

Hi Jason, is “teacher forcing” the same thing as the concept you showed in your article “How to Develop Word-Based Neural Language Models in Python with Keras” in the section “Model 2: Line-by-Line Sequence”?

https://machinelearningmastery.com/develop-word-based-neural-language-models-python-keras/

There, you trained an LSTM with training data that looked like:

1. X=(_, _, _, _, _, Jack), y=and
2. X=(_, _, _, _, Jack, and), y=Jill
3. X=(_, _, _, Jack, and, Jill), y=went
4. X=(_, _, Jack, and, Jill, went), y=up
5. X=(_, Jack, and, Jill, went, up), y=the
6. X=(Jack, and, Jill, went, up, the), y=hill
7. X=(and, Jill, went, up, the, hill), y=to

Thanks for any clarification.

Jason Brownlee March 21, 2019 at 7:58 am #

REPLY

Great question.

I use teacher forcing by default, it is just so effective. What is harder is using it sometimes and not others, and allow the model to correct an offtrack input sequence.

Nick March 22, 2019 at 8:38 am #

REPLY

Can you confirm that the approach in “Model 2: Line-by-Line Sequence” is the same as teacher forcing? I’m just trying to wrap my head around the terminology.

MultiK April 28, 2019 at 7:26 pm #

REPLY

‘Teacher forcing is a strategy for training recurrent neural networks that uses model output from a prior time step as an input.’

At first reading, I think your word ‘output from a prior time step’ is the output(not ground True) from time t-1.
Actually, It’s the ground True at time t-1 which is send to time t as input during training.Right?

REPLY

Jason Brownlee April 29, 2019 at 8:20 am #

Yes, ground truth.

REPLY

Ruzbeh June 6, 2019 at 2:03 am #

Great post, thank you!

How is the Scheduled Sampling/Curriculum learning actually implemented? I assume we need to write a custom Keras backend function?

Do you know of any pseudo-code to help implement this?

Thanks!

REPLY

Jason Brownlee June 6, 2019 at 6:36 am #

No, you provide real inputs during training rather than predicted inputs.

I have countless examples, perhaps start here:
https://machinelearningmastery.com/start-here/#lstm

and here:
https://machinelearningmastery.com/start-here/#deep_learning_time_series

REPLY

Ruzbeh June 6, 2019 at 6:51 am #

Thanks! Maybe I should clarify:

Doesn’t Scheduled Sampling, during training, gradually change from using “real inputs” to “predicted inputs”? This is my understanding from the paper.

REPLY

Jason Brownlee June 6, 2019 at 2:13 pm #

Yes, that is the ideal implementation.

To implement this in Keras requires customized code, I don’t have an example of the transition.

Ruzbeh June 6, 2019 at 11:06 pm #

Great! Thank you for your reply. I'll try to code one up!

Brando Miranda June 12, 2019 at 7:43 am #

REPLY

Hi Jason!

It might be good to mention this:

```
The first trick is using teacher forcing. This means that at some
probability, set by teacher_forcing_ratio, we use the current target word
as the decoder's next input rather than using the decoder's current guess.
This technique acts as training wheels for the decoder, aiding in more
efficient training. However, teacher forcing can lead to model instability
during inference, as the decoder may not have a sufficient chance to truly
craft its own output sequences during training. Thus, we must be mindful
of how we are setting the teacher_forcing_ratio, and not be fooled by fast
convergence.
```

Jason Brownlee June 12, 2019 at 8:08 am #

REPLY

Thanks.

Omar September 5, 2019 at 3:28 am #

REPLY

Hi Jason, thanks for the article!
I am curious about a point: in section "Using Output as Input in Sequence Prediction", you mention
"
This same recursive output-as-input process can be used when training the model, but it can result in problems such as:

Slow convergence.
Model instability.
Poor skill.
"
Can you provide a reference to this point? I could not really put my hand on a paper where they report trying this training scheme or studied its effect.
That will be much appreciated :))

Jason Brownlee September 5, 2019

REPLY

Not off hand, perhaps check some of the papers on teacher forcing and extensions?

Omar September 5, 2019 at 10:01 pm #

REPLY

I will. Thanks Jason 🙂

Ferdinando Insalata March 12, 2020 at 3:31 am #

REPLY

Hi Jason,
how would I go about implementing teacher forcing in an autoencoder?
Since the input of the decoder are embeddings produced by the autoencoder, how do I supply the target token ?

Thank you for the helpful resources.

Jason Brownlee March 12, 2020 at 8:54 am #

REPLY

It would be just like teacher forcing for any LSTM model. The autoencoder model does not make it different.

Johnathan July 4, 2020 at 6:35 am #

REPLY

there is a efficient way to do teacher forcing trainning but using yhat as input in lieu of ground true y?

Jason Brownlee July 5, 2020 at 6:46 am #

REPLY

Yes, but that is not teacher forcing.

You can do it sample by sample manually.

Darryl Fenwick July 11, 2020 at 4:59 am #

REPLY

Hi Jason,

Teacher forcing makes me think of NARX neural networks, a subject which I am interested in. I have yet to find an example of one with Keras. My question is whether you could use teacher forcing with multiple delays of outputs to create a NARX model, which normally would use the model outputs and not ground truth.

Jason Brownlee July 11, 2020 at 6:22 am #

REPLY

Not sure off hand, perhaps explore whether it is viable with some prototypes.

daniel August 17, 2020 at 2:38 pm #

REPLY

Hi Jason,
Have you ever tried using teacher forcing on the first half of the number of epochs and the last half do not ??. when I did that I noticed that when the first half of the epochs ended, my val_loss suddenly dropped sharply, the train_loss increased and I didn't understand why that happened. As far as I thought, when using teacher forcing, both train_loss and val_loss will decrease.

Jason Brownlee August 18, 2020 at 5:57 am #

REPLY

Nice experiment.

I would expect that once teacher forcing is removed that model performance would get worse.

It is a good idea to cycle teacher forcing on and off so the model can slowly learn how to correct its own mistakes.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Start Machine Learning

Welcome!

My name is *Jason Brownlee* PhD, and I **help developers** get results with **machine learning**.

Read more

Never miss a tutorial:

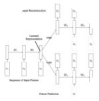
Picked for you:

How to Reshape Input Data for Long Short-Term Memory Networks in Keras

How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras

How to Develop an Encoder-Decoder Model with Attention in Keras

How to Use the TimeDistributed Layer in Keras



A Gentle Introduction to LSTM Autoencoders

Loving the Tutorials?

The LSTMs with Python EBook is where you can discover the ***Really Good*** stuff.

SEE WHAT'S INSIDE

Start Machine Learning