Phoneme Recognition Using the Encoder-decoder Framework

Israel Malkin
New York University
Center for Data Science
israelmalkin@nyu.edu

Peter Li
New York University
Center for Data Science
Peter.Li@nyu.edu

Abstract

Encoder-decoder models are a powerful class of models that let us learn mappings from variable length input sequences to variable length output sequences. In this report, we investigate the efficacy of Encoder-decoder systems for the task of phoneme recognition.

1 Introduction

Deep Neural Networks are a family of machine learning models that have demonstrated excellent results on a number of difficult tasks across a number of domains including computer vision (Krizhevsky et al., 2012), music information retrieval (Dieleman et al., 2011), and genetics (Leung et al., 2014). Usually, however, these models rely on inputs and outputs of fixed size. For example, input images for an object recognition system might always be 100 pixels $\times$ 100 pixels $\times$ 3 channels. Similarly, the output will always be a fixed length vector that might, for example, correspond to the probability of each class.

For many problems, however, we cannot naturally encode our input and output into data structures with a fixed size. For example, translation can be viewed as learning a mapping from one *arbitrary* length sequence (source sentence) into another *arbitrary* length sequence (target sentence). For these tasks, researchers have recently proposed a new class of models that can operate over variable length input and outputs (Cho et al., 2014a). In the so called Encoder-decoder family of models, one recurrent neural network (RNN) is trained to encode the input sequence and another RNN is trained to decode the output of the encoder.

For our project, we investigate the efficacy of Encoder-decoder systems for the task of phoneme recognition. In this task, we take spectrogram representations of speech audio as input to our model and output phoneme sequences. In designing our experiments, we hoped to understand the following:

- Does the attention mechanism improve model performance? If so, how much?
- What is the effect of the size of the encoder and decoder memory states?
- How do different representations affect model performance?

2 Model

The basic unit of our RNNs is a Gated Recurrent Unit (GRU) (Cho et al., 2014b). The memory state of a GRU is determined by the following set of equations:

$$z_t = \sigma(W_{z,h_{t-1}} h_{t-1} + W_{z,x} x_t + b_z) \tag{1}$$

$$r_t = \sigma(W_{r,h_{t-1}} h_{t-1} + W_{r,x} x_t + b_r) \tag{2}$$

$$\tilde{h_t} = \tanh(\tilde{W}_{r_t,h_{t-1}}[r_t * h_{t-1}] + \tilde{W}_x x_t + \tilde{b}) \tag{3}$$

$$h_t = (1 - z_t) * h_t + z_t * \tilde{h}_t \tag{4}$$

For an arbitrary length input sequence $X = [x_1, ..., x_T]$, an RNN with GRUs will compute $H = [h_1, ..., h_T]$

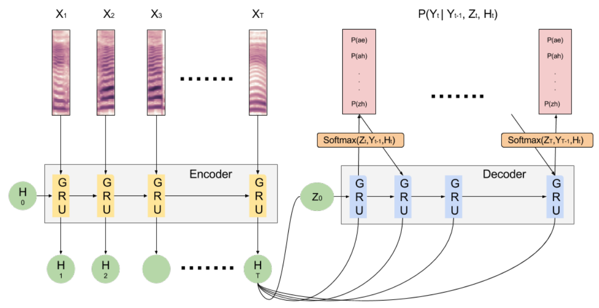

Figure 1: Schematic of Simple Encoder-decoder model. The decoder considers only the last memory state of the encoder.

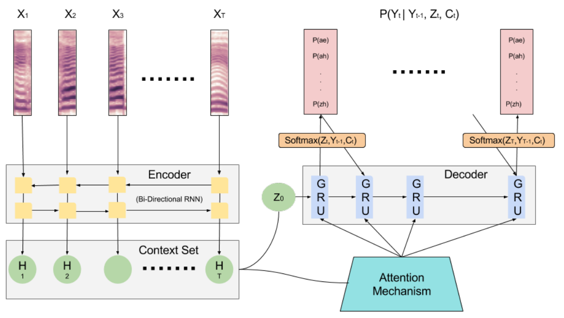

Figure 2: Schematic of Encoder-decoder model with attention. The decoder is connected to the entire memory state history via the attention mechanism.

2.1 Encoder-decoder

An Encoder-decoder model is an architecture that learns to **encode** a *variable*-length input sequence into a *fixed*-length vector representation (typically the last memory state, h_T) and then learns to **decode** the fixed length vector into a *variable*-length output sequence.

This model consists of two RNNs. One to encode the input sequence into a fixed length vector, C and another to decode C into the output sequence. The decoder learns a language model, but one that is additionally conditioned on C. To accomplish this, we use a slightly modified version of the GRU that lets us condition our outputs on C. Note that for the simpler model, the conditioning vector C is the last memory state of the encoder RNN(H_T in figure 1).

$$z_t = \sigma(W_{z,h_{t-1}} h_{t-1} + W_{z,x} x_t + W_{z,c} C + b_z) \quad (5)$$

$$r_t = \sigma(W_{r,h_{t-1}} h_{t-1} + W_{r,x} x_t + W_{r,c} C + b_r) \quad (6)$$

$$\tilde{h_t} = \tanh(\tilde{W}_{r_t,h_{t-1}} [r_t * h_{t-1}] + \tilde{W}_x x_t + \tilde{W}_c C + \tilde{b}) \quad (7)$$

$$h_t = (1 - z_t) * h_t + z_t * \tilde{h}_t \quad (8)$$

At each time step, an affine combination of the decoder output ,h_t, the previous phoneme, y_{t-1}, and the current context, C_t, are all passed to a softmax to obtain the the conditional probability distribution over the target vocabulary, $p(y_t | h_t, y_{t-1}, C_t)$.

We show a schematic of this model in Figure 1.

2.2 Attention

The simple Encoder-decoder model described above encodes a variable length input sequence into a *fixed*-length vector. Using this fixed-length vector, we can train a decoder as a conditional language model that expects this fixed-length vector. This convenience, however, may come with a cost. In (Cho et al., 2014a), researchers observed that the performance of simple Encoder-decoder models degrade as the length of the input sequence grows (Cho et al., 2015). Intuitively, this makes sense since the limited capacity of the fixed-length memory state can not store input from many time-steps back. A naive approach would be to increase the size of the memory state vector, but obviously this logic can only be taken so far until the memory state vector becomes too large.

The attention mechanism sidesteps the finite-memory issue by generating a context set that is comprised of all the memory states of the encoder RNN. At each time step during decoding, the attention mechanism will generate a glimpse,G_t , of the context set that is passed to the conditional language model as context. Compare this to the simple encoder-decoder model where the context for the conditional language model is always the last memory state of the encoder. Additionally, our attention model utilizes a bi-directional RNN to encode the input speech data.

This model is displayed in figure 2, along with equations 9-11.

Score

$$Score_{j,t} = W^T tanh(h_{t-1}, y_{t-1}, C_j) \quad (9)$$

Attention

$$a_{j,t} = softmax(score_{1,t}, \ldots, score_{DimEnc,t}) \quad (10)$$

Glimpse/Context for attention

$$G_t = A_t C \tag{11}$$

The model learns which parts of the context set to "pay attention to" depending on the current state of the decoder system. This enhanced access to the entire context/memory states of the encoder comes at the relatively cheap price of learning the scoring/attention weights, as opposed to attempting to encode the entire history of longer input sequences via the final state of an extremely large fixed-length vector. The one-hot phoneme input was passed through an embedding before being fed into the decoder GRU. The initial state of the encoder GRUs was set to a vector of zeros. In the simple model, the initial state of the decoder was learned as a function of the context vector. In the attention mode, the decoder was initialized with the average of the context set.

3 Data

To train our models, we used the TIMIT corpus which consists of 5.4 hours of audio recordings of 6300 read sentences. For each recording, TIMIT provides word and phoneme level annotations (Garofolo et al., 1993).

As input to our models, we used a time/frequency representation of the original signal. We computed the Mel-spectrograms using a 25ms asymmetric Hann window and 10ms hops. For better contrast, we took the logarithm to get log Mel-spectrograms. Additionally, we computed another representation that appends the first and second order differences. This captures the speed and acceleration of the frequency changes and is a commonly used feature in audio signal processing (especially in processing Mel-frequency Cepstral Coefficients).

4 Experiments

4.1 Experimental Setup

To answer the questions posed in the introduction, we ran experiments while varying the model architecture, input feature type, and size of memory states.

Specifically:

- Model Architecture
  - simple GRU Encoder/ GRU decoder
  - bi-directional GRU encoder/ GRU decoder
- Input
  - log Mel-spectrogram, 40 frequency bins
  - log Mel-spectrogram, 100 frequency bins
  - log Mel-spectrogram (40) + Δ + Δ^2 (120 dimensions total)
  - log Mel-spectrogram (100) + Δ + Δ^2 (300 dimensions total)
- memory State Size
  - Encoder: 200 and 500
  - Decoder: 200 and 500

Taking all possible combinations results in 32 different configurations.

4.2 Model Training

As is standard practice, we removed the SA sentences from the corpus and used the standard TIMIT train and test split. From the training data, we hold-out 500 samples for validation. To combat the issue of exploding gradients, the norm of the gradient was clipped to 2, and optimization proceeded using RMSPROP (Tieleman and Hinton, 2012). Running on a Titan X GPU, the smaller models without attention took approximately one day to converge while the larger architectures with attention took around 3 days to converge.

Final evaluation was based on the phoneme error rate (PER). To make the results comparable to past work, we converted the TIMIT phonemes from 61 to 39 broader categories (Fernández et al., 2008). The optimization problem of picking the most probable output sequence was approximated using beam search with a beam size of five.

We present the results of our experiments in Figures 3 and 4.

5 Discussion

In designing our experiments, we wanted to answer the following three questions:

Model Architecture (40)	Validation PER	Test PER
GRU Encoder/ GRU Decoder		
log Mel Spectrogram / log Mel Spectrogram + Δ + Δ^2		
(200,200), (200,500), (500,200), (500,500)	100%	100%
Bi-directional GRU Encoder/ GRU Decoder + Attention		
log Mel Spectrogram + Δ + Δ^2		
200,200	78.55%	79.26%
200,500	77.62%	78.99%
500,200	80.42%	81.20%
500,500	79.92%	79.46%
log Mel Spectrogram		
200,200	77.52%	81.62%
200,500	78.40%	81.72%
500,200	78.64%	80.98%
500,500	78.23%	81.23%

Figure 3: Experiment Results Using 40 Frequency Bins

Model Architecture(100)	Validation PER	Test PER
GRU Encoder/ GRU Decoder		
log Mel Spectrogram / log Mel Spectrogram + Δ + Δ^2		
(200,200), (200,500), (500,200), (500,500)	100%	100%
Bi-directional GRU Encoder/ GRU Decoder + Attention		
log Mel Spectrogram + Δ + Δ^2		
200,200	76.79%	78.02%
200,500	76.85%	78.07%
500,200	75.93%	77.72%
500,500	77.74%	79.01%
log Mel Spectrogram		
200,200	79.13%	80.51%
200,500	80.01%	80.95%
500,200	79.70%	80.01%
500,500	75.78%	81.02%

Figure 4: Experiment Results Using 100 Frequency Bins

1. Does the attention mechanism improve model performance? If so, how much?
2. What is the effect of the size of the encoder and decoder memory states?
3. How do different representations affect model performance?

Based on our results we would report that attention makes a tremendous difference, while the size of the memory states and the input representation do not affect performance too much. However, it is unclear if we can address any of these questions with a high degree of confidence. The simple Encoder-decoder models failed to learn anything and achieved 100% PER in all experiments. While it is not unexpected that the Encoder-decoder models would perform poorly, we did not expect 100% PER. Investigating further, we see that during training the optimization does not minimize the cost. The negative log-likelihood oscillates near the initial value and training ends due to early stopping. We plot the convergence of a typical simple Encoder-decoder model in Figure 5. To rule out optimization problems, we also ran the experiment with AdaDelta. Results were similar.

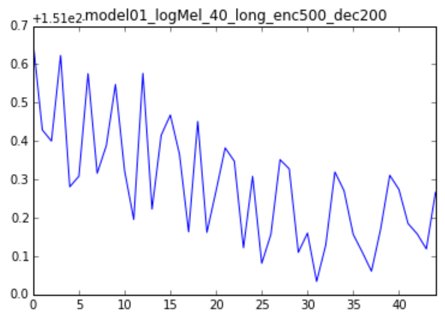

Figure 5: Typical Convergence of Model Without Attention. These models fail to converge. Negative log-likelihood oscillates around the starting value and training eventually ends due to early stopping.

The experiments that used attention achieved lower PER across all specifications. Further, the models showed much better convergence. A typical plot of NLL is show in Figure 6. Despite this, our models achieved much higher PER than other deep learning model such as (Hinton et al., 2012) and especially (Chorowski et al., 2015) and (Chorowski et al., 2014). These model typically report a test PER around 20% - 17% while our lowest PER was 77%.

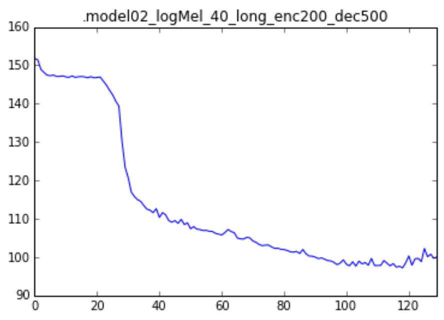

Figure 6: Typical Convergence of Model With Attention. Models trained with attention exhibit much better convergence behaviour.

Comparing our model architectures with these others, we find that the consistent difference is the depth of our encoder. Even in their baseline model, (Chorowski et al., 2015) uses a three layer bi-directional RNN as the encoder.

In designing our experiments, we mainly wanted to investigate the effects of attention, hidden state sizes, and feature representation. For ease of training, we used a single layer encoder and a single layer decoder. It seems, however, that a single layer encoder might not have enough capacity to learn a useful representation of the spectrogram. As a result, this prevented our other changes from making a difference as the effect of the weak encoder dominated.

6 Future Work

As a follow up to this analysis, we would like to rerun these experiments with more expressive encoders. The obvious choice would be to use at least two or three layers in the encoder. Also, given the spatial nature of the input spectrograms, we would also like to try using convolutional neural networks as the encoder.

References

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *Multimedia, IEEE Transactions on*, 17(11):1875–1886.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*.

Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. 2011. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, pages 669–674.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2008. Phoneme recognition in timit with blstm-ctc. *arXiv preprint arXiv:0804.3269*.

John S Garofolo, Linguistic Data Consortium, et al. 1993. *TIMIT: acoustic-phonetic continuous speech corpus*. Linguistic Data Consortium.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. 2014. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.