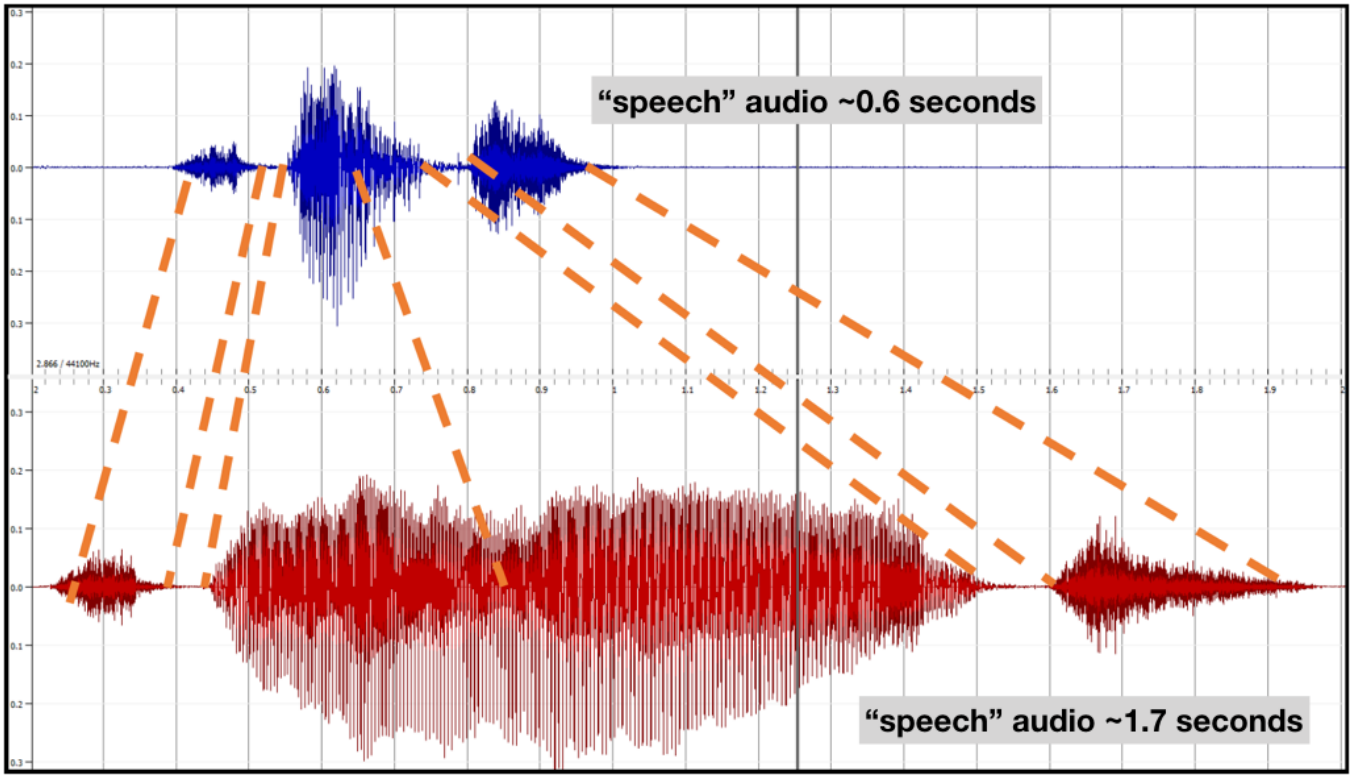
Connectionist Temporal Classification (CTC)

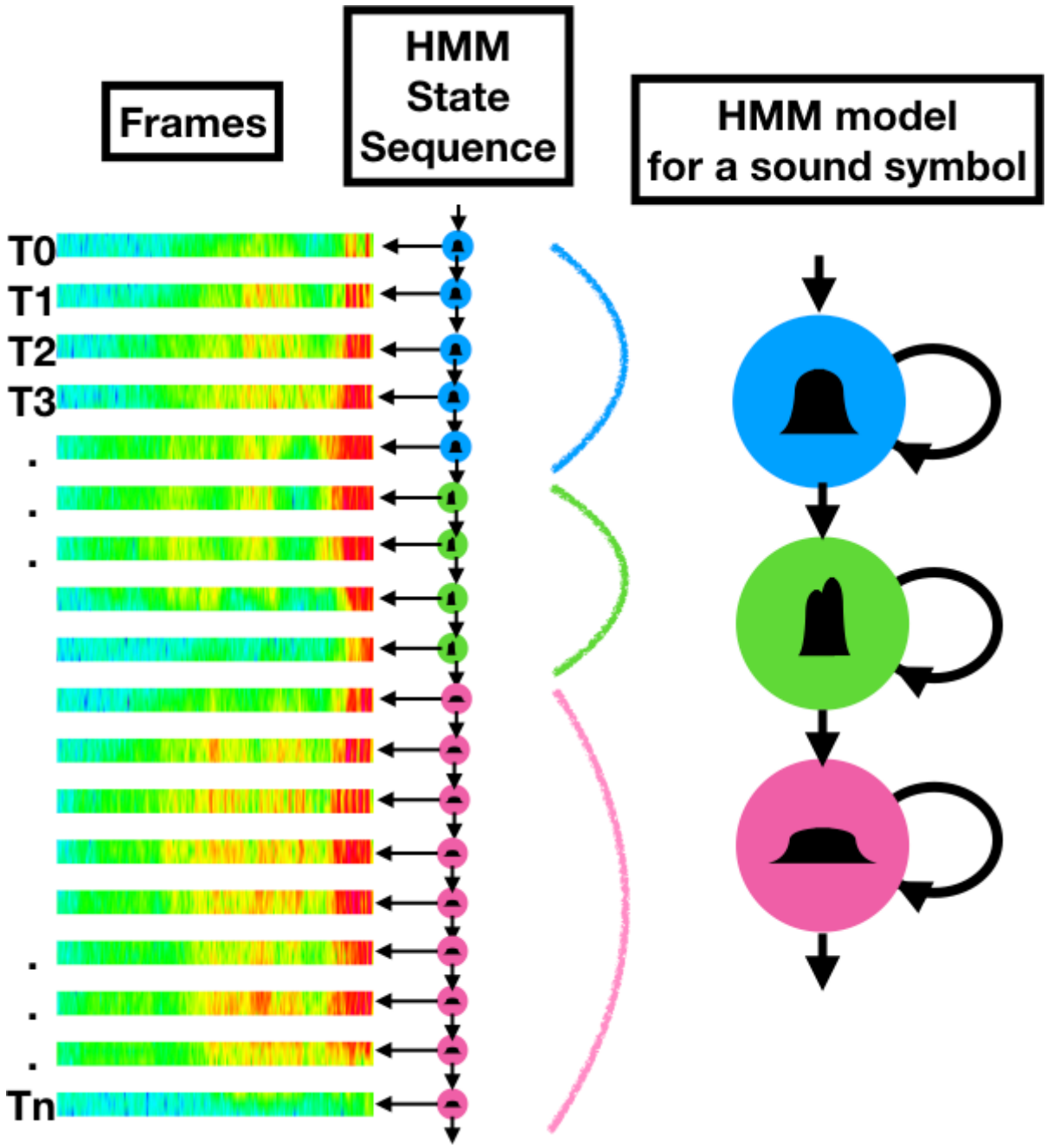
The Sequencing Problem

The input data in speech recognition is a sequence of observations in the form of frame vectors from regular time intervals. The desired output is a series of symbols: phonemes, graphemes, or words. The basic problem is that the number of frames does not have a predictible correspondence to the number of the output symbols. For example, if we assume 20ms per frame, the following audio signals of the word "speech" spoken at two different speeds have about 300 frames in the first example and something like 850 frames in the second example, yet they should both be decoded as the six-letter word, "speech".



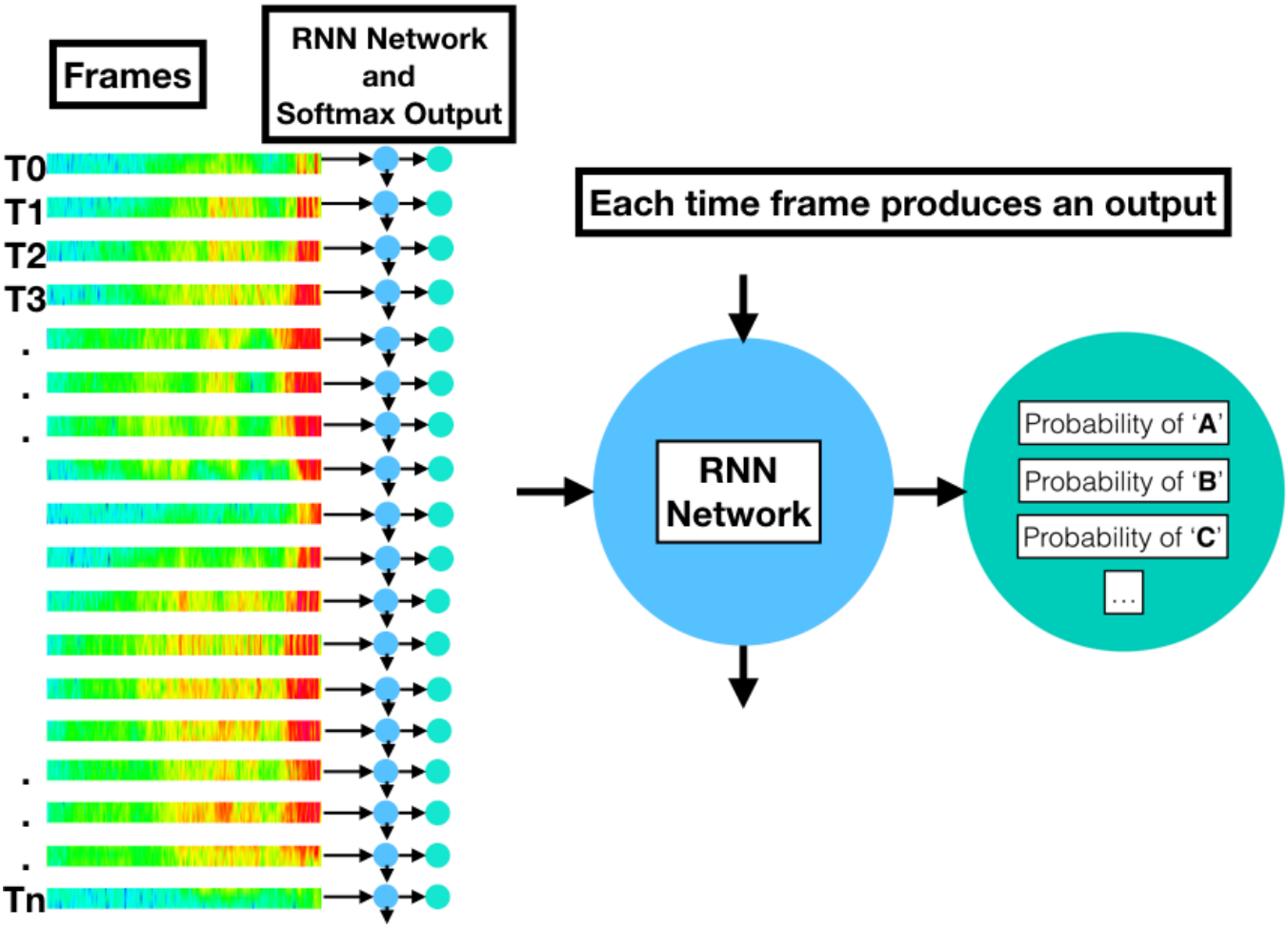
Sequencing with HMMs

Hidden Markov models (HMMs) are well suited to sequencing because each hidden state can be mapped to any number of frames. During training, distributions are clustered in each hidden state to define the model. During decoding, the overall likelihood that a particular HMM model could have emitted a given series of frame vectors is calculated and compared to the likelihood that they were emitted by other HMMs. The following diagram illustrates frame mappings for a single HMM.



Why RNNs can't sequence

By contrast, a Recurrent Neural Network (RNN) produces a probability distribution of output symbols for each frame. Suppose we want to train an RNN network to recognize 26 different letters plus an apostrophe, space, and blank (29 graphemes). There will be a softmax layer output that produces a probability distribution for these possibilities. Each frame will produce one of these probability distribution vectors. If the utterance has 300 observations, there will be 300 RNN softmax vectors. If the utterance consists of 850 observations, there will be 850 vectors.



The RNN could learn what those graphemes should be if there was a label associated with each frame. This would require some sort of manual pre-segmentation.

A more ideal solution would be to provide the network with a loss function across the entire label sequence that it could minimize when training. We would like the probability distribution of the softmax output to "spike" for each grapheme and provide blanks or some other consistently ignored result between the graphemes so that the transcription could be easily decoded. This would solve the sequencing problem as audio signals of arbitrary length are converted to text.

CTC to the rescue

A Connectionist Temporal Classification (CTC) loss function can be calculated to train the network. When combined with the CTC decoding algorithm over the softmax output layer, the sequence-to-sequence transformation is achieved. The details of how this works are in the seminal paper on CTC by Alex Graves:

Graves, Alex, et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

Here are a few take-aways from the paper:

Training with the CTC Loss Function

Graves derives an objective function to calculate the error between the labels expected and the softmax result. The function calculates all the possible paths and likelihoods for the softmax output layer and calculates the maximum log likelihood for the correct labeling. For example, a correct labeling of a "hello" utterance might have several valid paths with varying probabilities:

```
_ _ H _ _ E L _ _ L _ O _  probability = p1
_ H _ E L _ L _ _ _ O _ _  probability = p2
_ _ _ _ H E L _ L _ O _ _  probability = p3
...
_ H E _ _ L _ L _ O _ _ _  probability = pn
```

The maximum log likelihood is the sum of likelihoods for all correct paths. From Graves: "The aim of maximum likelihood training is to simultaneously maximise the log probabilities of all the correct classifications in the training set. In our case, this means minimising the following objective function:"

$$O^{ML}(S, \mathcal{N}_w) = -\sum_{(\mathbf{x},\mathbf{z})\in S} \ln(p(\mathbf{z}|\mathbf{x}))$$

S = the set of all possible tuples ($\mathbf{x}$,$\mathbf{z}$)
$\mathbf{x}$ = the input sequence
$\mathbf{z}$ = the target sequence
N_w = the network weight variables

The loss is then differentiated and used in standard gradient descent training of the network to correct the softmax output such that it matches the labels when decoded.

The output includes "blanks"

The softmax output for the CTC network includes "blank" symbols that can be ignored during decoding.

CTC Decoding

Extracting the most likely symbol from each softmax distribution will result in a string of symbols the length of the original input sequence (the frames). However, with the CTC training,

the probable symbols have become consolidated. The CTC decoding algorithm can then compress the transcription to its correct length by ignoring adjacent duplicates and blanks.

A more complex CTC decoding can provide not only the most likely transcription, but also some number of top choices using a beam search of arbitrary size. This is useful if the result will then be processed with a language model for additional accuracy.

CTC Implementation options

- TensorFlow
- Keras
- CTC-Warp