Demonstrating Connectionist Temporal Classification (CTC)

Ryan Brigden

When to use CTC?

- Many-to-many sequence prediction.
- Labelling order matters, but not a one-to-one correspondence between outputs and labels.
- Need to impose structural constraints on the output sequence (e.g. finite state machine)

CTC Loss intuition

1. Add a blank symbol to the set of possible labels.
2. Design the network to output a classification at each input value.
3. “Collapse” repeated outputs and blanks
    - AAAAA → A
    - AAAABB → AB
    - AAAA___BB → AB
    - AAA__AA → AA (blanks separate outputs of the same symbol)
4. Compare collapsed output with the ground truth label

CTC Loss intuition

Input:

Network outputs before training:

_CA_AB_C_BB_AC_A_B_AA_CC_

Collapsed:

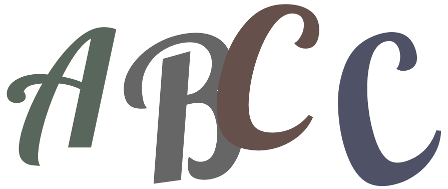
CAABCBACABAC

Ground truth:

ABCC

CTC Loss intuition

Input:



Network outputs after training:

_________AAAA_BBBCC__CCCC

Collapsed:

ABCC

Ground truth:

ABCC

Probabilistic values

At each time step, the network actually outputs a probability distribution over all the possible labels + the blank symbol.

If we randomly select a label independently from each of these distributions, what is the probability that we will get an output sequence that collapses into the ground truth sequence?

The CTC loss is the negative logarithm of this probability.

More formal description of CTC Loss

Let L be the set of labels and L’ be the set of labels with the “blank” label.

For a sequence of length T, we denote the set of possible paths $L'^T = \boldsymbol{\pi}$.

Given a sequence of inputs **x** and labelling **z**, where |**z**| <= |**x**|, try to maximize the probability of the labelling given the sequence (maximum likelihood estimate).

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} p(\mathsf{z}^{(i)} \mid \mathsf{x}^{(i)}; \theta)$$

More formal description of CTC Loss

Define the many-to-one map **B** : L'^T -> $L^{\leq T}$ that maps length T label sequence of alphabet L' to their labelling equivalent in L, while removing all blanks and repeated labels. In effect, **B** performs the "collapsing" operation discussed previously. For example:

B(_A__AAAA_BBBCCCC) = **B**(A_A_BBBB_CC) = AABC

Define $\mathbf{B}^{-1}$ to map a label sequence **z** to the set of all possible label sequences (paths in π) that collapse to **z**. So {**B**(x) | x ϵ $\mathbf{B}^{-1}$(y)} = y.

Therefore, we can consider the likelihood of a given labelling **z** as the sum of the probabilities of all the paths that can collapse to **z**.

$$p(\mathrm{z} \mid \mathrm{x}; \theta) = \sum_{\pi \in B^{-1}(\mathrm{z})} p(\pi \mid \mathrm{x}; \theta)$$

More formal description of CTC Loss

Our objective is now more clear and we can plug in our formulation of the likelihood of a labelling. The objective of the argmin is the CTC loss.

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} \sum_{\pi \in \mathsf{B}^{-1}(\mathsf{z}^{(i)})} p(\pi \,|\, \mathsf{x}^{(i)}; \theta)$$

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log \left[\sum_{\pi \in \mathsf{B}^{-1}(\mathsf{z}^{(i)})} p(\pi \,|\, \mathsf{x}^{(i)}; \theta) \right]$$

More formal description of CTC Loss

Change the problem from maximization to minimization to

$$\hat{\theta} = \arg\min_{\theta} - \sum_{i=1}^{N} \log \left[\sum_{\pi \in \mathsf{B}^{-1}(\mathsf{z}^{(i)})} p(\pi \,|\, \mathsf{x}^{(i)}; \theta) \right]$$

Computing the probability

Naive method:

Generate all possible sequences with their probabilities, collapse each sequence, sum the probabilities of those that collapsed to the correct value.

Let N be the number of inputs, M be the number of labels, C be the number of classes that can be output at each input (including the blank symbol)

Computational complexity: $O(C^N M)$

Better way to compute the probabilities

We can derive a recursive formula for the probability and exploit the existence of common sub-paths (think dynamic programming).

See Lecture 13 for the in-depth explanation of this method.

Inference

What sequence of labels do we output at test time?

When we used the “many-to-one” architecture, this was easy. For classification tasks, we picked the argmax of the network outputs. For regression, we just returned the network output unchanged.

For CTC, it’s less obvious what to do. We want to output the most likely sequence of “collapsed” labels. But a large number of paths can collapse into that sequence of labels.

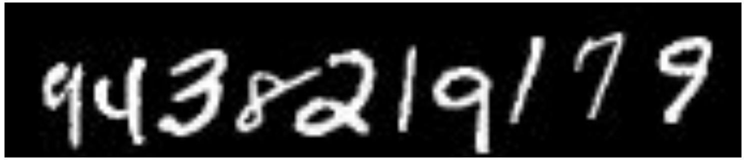
- We can generate the most likely path, and collapse that. This is a greedy approach though, and doesn’t guarantee that we get the most likely sequence of labels.
- We can run a beam search, a method where we keep track of multiple possible best paths as we iterate over the network’s output probabilities to reduce the greediness of our search.
- We can use a similar dynamic programming algorithm to compute the true answer.

Variants of CTC

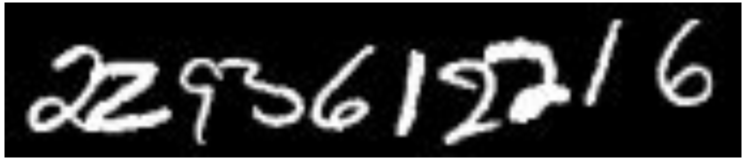
CTC actually describes a family of loss functions. The formulation with blanks is just the original example. Other examples:

- If you can guarantee that the same label cannot occur twice in succession, you can create a version of CTC without blanks.
- There are times when there are multiple types of blanks, perhaps one for each possible non-blank symbol.
- Can even follow certain finite state machines. For instance, the formulation in HW1 where each phoneme contains three sections that must occur in some order. You can create a CTC that only considers paths that outputs symbols in the right order.

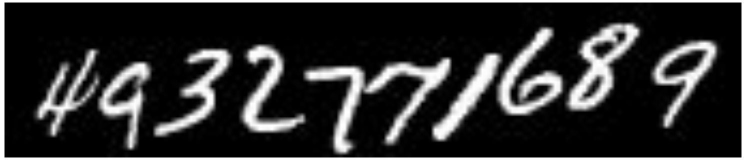
Handwritten Digit Recognition



“9438219179”



“2293612216”



“4932771689”

- Binary images
- HxW = 36x172
- 10 digits per sample