Sequence Learning with Connectionist Temporal Classification

Alex Graves, 2006

Rakesh Rao
Stanford University

June 14, 2015

References

- Graves, Alex. Supervised sequence labelling with recurrent neural networks. Vol. 385. Heidelberg: Springer, 2012.
- Graves, Alex, et al. "A novel connectionist system for unconstrained handwriting recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 31.5 (2009): 855-868.
- 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.

Motivation

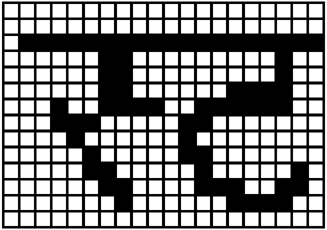
Learn a sequence of labels from an input stream

- Input and output of variable lengths
- Location unknown/undefined

Figure : Statistics

The Model

- Input is a sequence in $\mathbb{R}^n$ (here $n = 20$, the image height)
- Output is a sequence in $\mathbb{R}^k$ (here $k = 26$, the number of classes)
- k-vector sums to unity



`---sssssss-----tttttt--`

Recurrent Neural Network

- Say, in previous slide, $20 = 2$ and $26 = 2$, i.e. image height is two pixels, and there are only two classes. Then we can ...

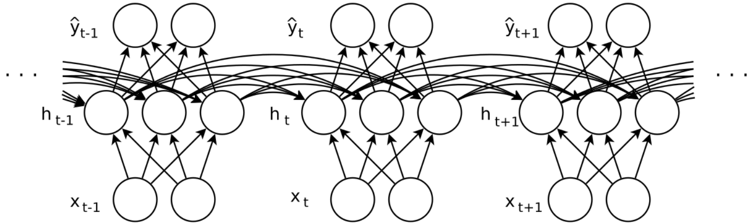


Figure : Parallel lines share weights

Recurrent Neural Network

- Now each input **node** is n vector and output **node** k vector
- Each arrow is now a matrix multiplication
- Parallel arrows have same weight matrix

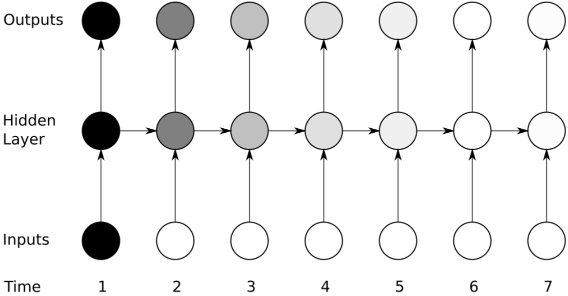


Figure : RNN for a sequence with seven timesteps

A Good Output for `CAB`

- Alphabet $= \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$, $k = 4 + 1$
- Add a `blank` or `null` class the network can fall back to. It reduces memory burden on the network and allows repeated labels

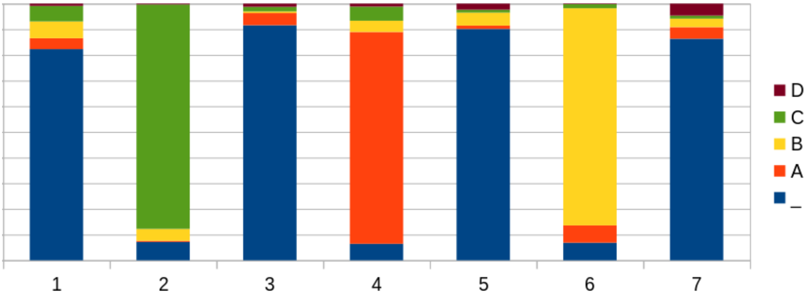


Figure : RNN outputs for a sequence with seven timesteps. `_C_A_B_` is the most likely labeling given the output.

Looking for the intractable `CAT`

- Out of the t time steps, the letters we want can 'stand out' anytime.
- Each such sequence is called a *path*
  e.g:- For $t = 7$ and output $=$ cat, we can have $\_ca\_\_t\_$, $\_c\_a\_t\_$, $\_\_c\_at\_$, $ca\_\_\_\_t$, etc. are all *good* paths
- Intractable number of paths for large t

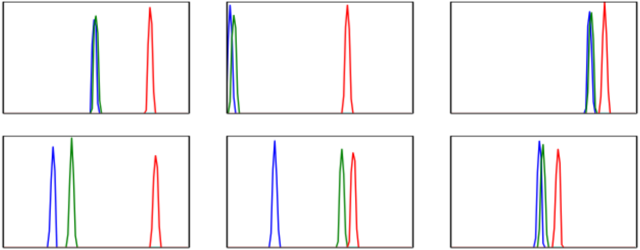


Figure : All six excitations give us a strong `cat` $(t = 100)$

Given all the above

- How do we train the network?
- How do we bell the cat?

Notation

- Alphabet $A' = A \cup \{\texttt{blank}\}$
- y_k^t - activation of k^{th} class at time t (interpreted as probability)
- A'^T - set of length T sequences over A'
- π one such *path* in A'^T
  e.g. $\pi = \texttt{\_ca\_tt\_}$ for $T = 7$.
- According to the model,

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t$$

- $\mathcal{F} : A'^T \to A^{\leq T}$ mapping from path to labeling.
  e.g:- $\mathcal{F}(\_c\_a\_t\_) = \mathcal{F}(ccaa\_t\_) = \cdots = cat$
- Probability of a/correct labelling:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{F}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$$

All the `cat`'s paths

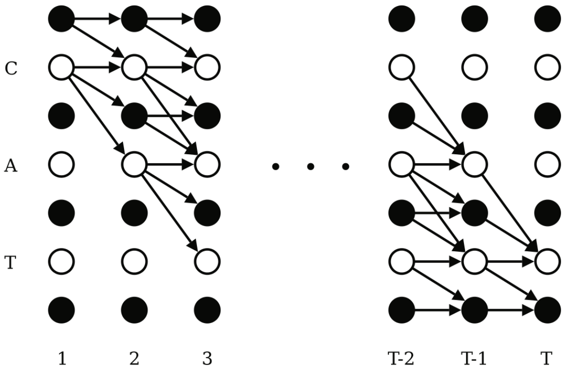


Figure : Black circles are `blanks`, white are labels. Arrows are allowed transitions. One traverasl from left to right is a *path* corresponding to the labelling `cat` or equivalently `_c_a_t_` (wlog)

Forward probabilities

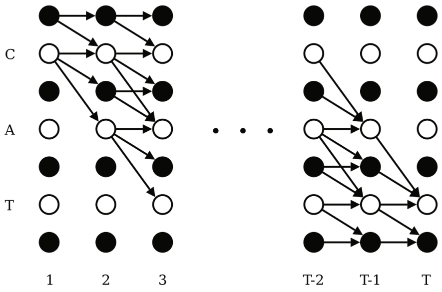
U is length of $\mathbf{l}$ (i.e. height of the picture in previous slide)

$$\alpha(t,u) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y^i_{\pi_i}$$

where $V(t,u)$ is the set of all paths going through label u at time t.
(t,u) a circle in picture.

$$
\begin{aligned}
p(\mathbf{l}|\mathbf{x}) &= \alpha(T,U) + \alpha(T,U-1) \\
\alpha(1,1) &= y^1_{\texttt{blank}} \\
\alpha(1,2) &= y^1_{l_1} \\
\alpha(1,3) &= \alpha(1,4) = \cdots = \alpha(1,U) = 0
\end{aligned}
$$

CTC will bell the `CAT`



Forward recursion: Just add the paths entering a circle and multiply the sum by the activation of that circle

$$\alpha(t+1,u) = \{\alpha(t,u) + \alpha(t,u-1) + \mathbb{1}(l_u \neq \texttt{blank})\ \alpha(t,u-2)\}\ y^{t+1}_{l_u}$$

Summary

- Apply RNN on input

$$\mathbf{y} = RNN(\mathbf{x}; \Theta)$$

- Find Forward probabilities

$$\alpha(t+1, u) = y_{l_u}^{t+1} \left[\alpha(t, u) + \alpha(t, u-1) + \mathbb{1}(l_u \neq \texttt{blank})\ \alpha(t, u-2)\right]$$

- Find probability of the desired labelling

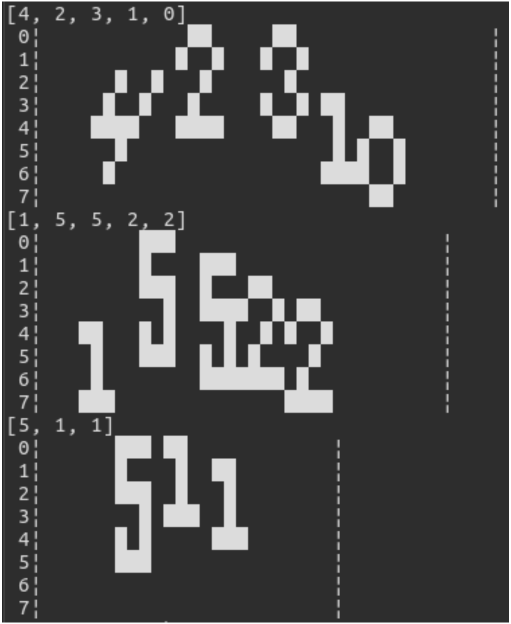
$$p(\mathbf{l}|\mathbf{x}) = \alpha(T, U) + \alpha(T, U-1)$$

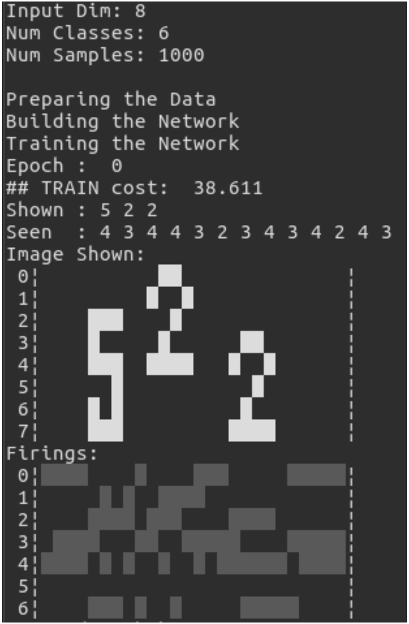
- Calculate Negative log-likelihood loss over the entire dataset S

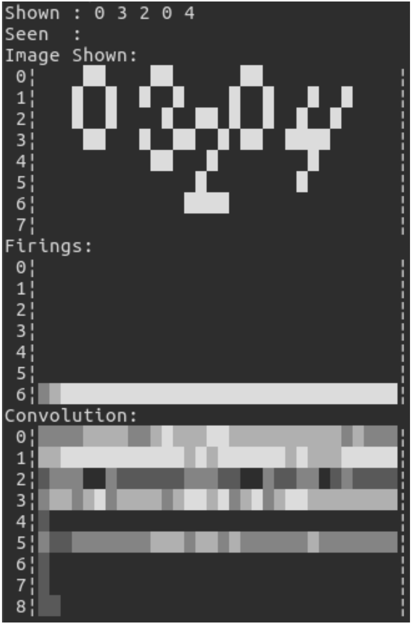
$$\mathcal{L}(S) = -\sum_{(\mathbf{x},\mathbf{l})\in S} \ln p(\mathbf{l}|\mathbf{x})$$

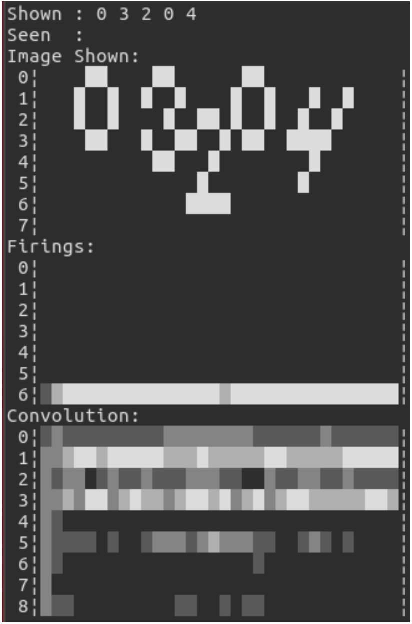
- ~~Back-propagate~~ Symbolic-differentiate $\mathcal{L}$ and gradient descend in the weight space for the $\mathrm{argmin}\ \Theta \equiv \{\mathbf{W}_{ih}, \mathbf{W}_{hh}, \mathbf{W}_{ho}\}$

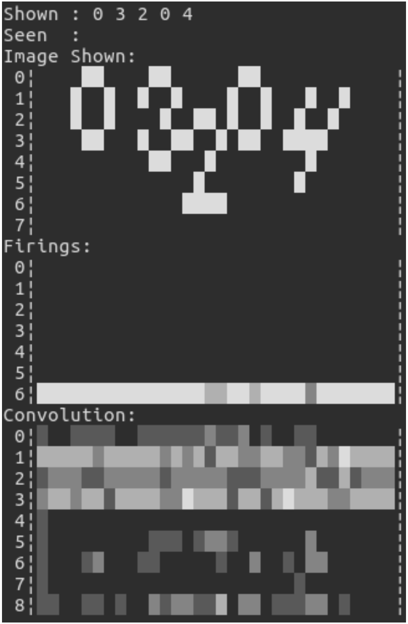
Really? Like, did it ever even, like, actually work, like, at all?

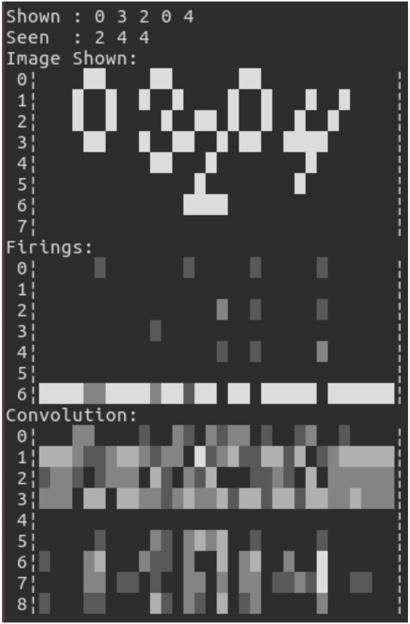

[4, 2, 3, 1, 0]
0
1
2
3
4
5
6
7
[1, 5, 5, 2, 2]
0
1
2
3
4
5
6
7
[5, 1, 1]
0
1
2
3
4
5
6
7

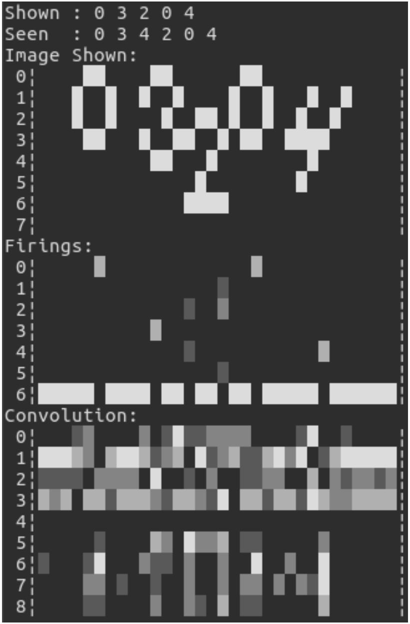

Input Dim: 8
Num Classes: 6
Num Samples: 1000
Preparing the Data
Building the Network
Training the Network
Epoch : 0
TRAIN cost: 38.611
Shown : 5 2 2
Seen : 4 3 4 4 3 2 3 4 3 4 2 4 3
Image Shown:
Firings:

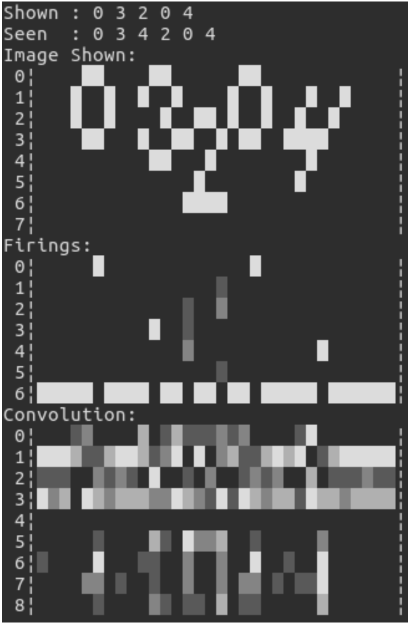

Shown : 0 3 2 0 4
Seen :
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8

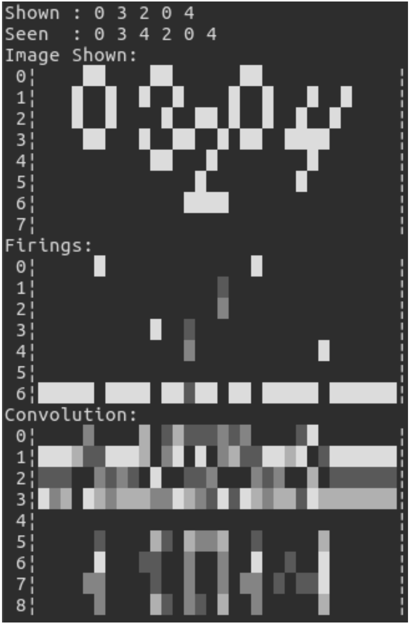

Shown : 0 3 2 0 4
Seen :
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8

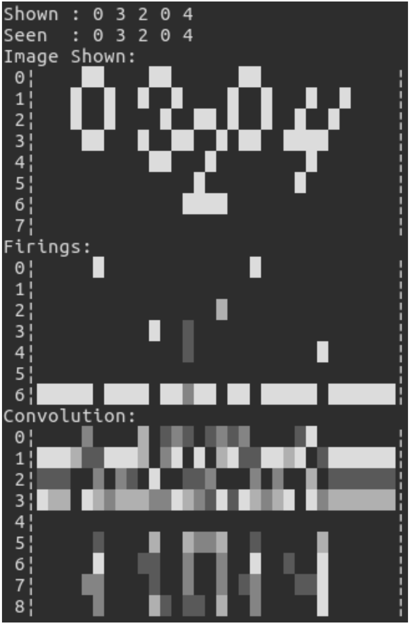

Shown : 0 3 2 0 4
Seen :
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8

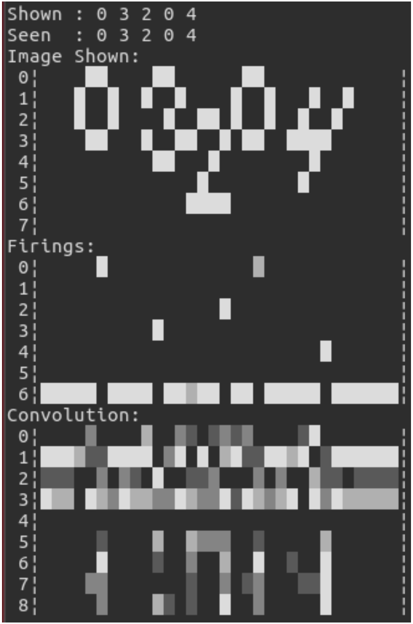

Shown : 0 3 2 0 4
Seen : 2 4 4
Image Shown:
Firings:
Convolution:

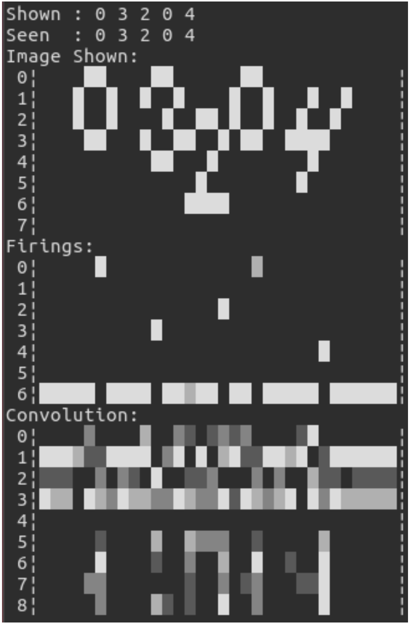

Shown : 0 3 2 0 4
Seen : 0 3 4 2 0 4
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8

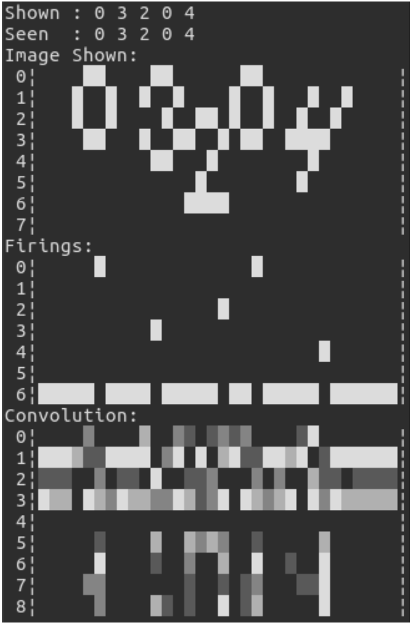

Shown : 0 3 2 0 4
Seen : 0 3 4 2 0 4
Image Shown:
Firings:
Convolution:

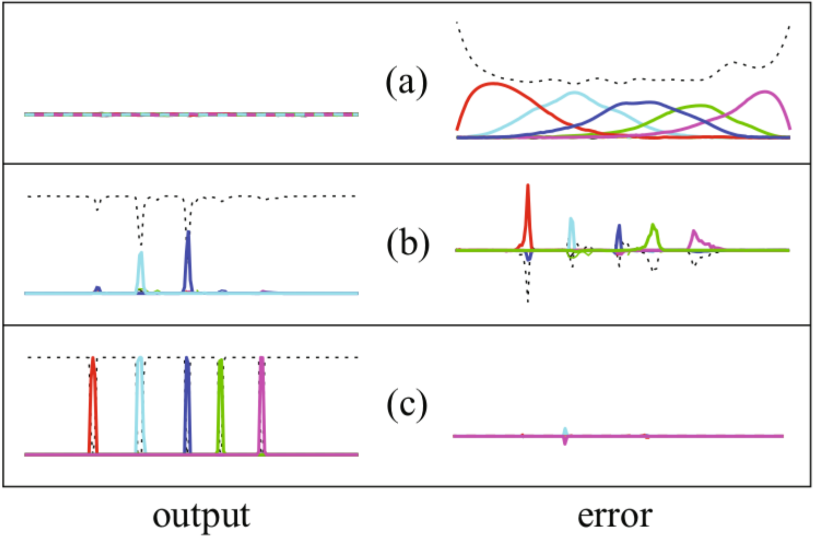

Shown : 0 3 2 0 4
Seen : 0 3 4 2 0 4
Image Shown:
0 1 2 3 4 5 6 7
Firings:
0 1 2 3 4 5 6
Convolution:
0 1 2 3 4 5 6 7 8


Shown : 0 3 2 0 4
Seen : 0 3 2 0 4
Image Shown:
Firings:
Convolution:


Shown : 0 3 2 0 4
Seen : 0 3 2 0 4
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8


Shown : 0 3 2 0 4
Seen : 0 3 2 0 4
Image Shown:
0
1
2
3
4
5
6
7
Firings:
0
1
2
3
4
5
6
Convolution:
0
1
2
3
4
5
6
7
8


Shown : 0 3 2 0 4
Seen : 0 3 2 0 4
Image Shown:
Firings:
Convolution:


(a)
(b)
(c)
output
error

_ t _ h _ a _ n _ k _ ‘ ’ _ y _ o _ u _