Connectionist Temporal Classification

Waseem Gharbieh

twentybn

Outline

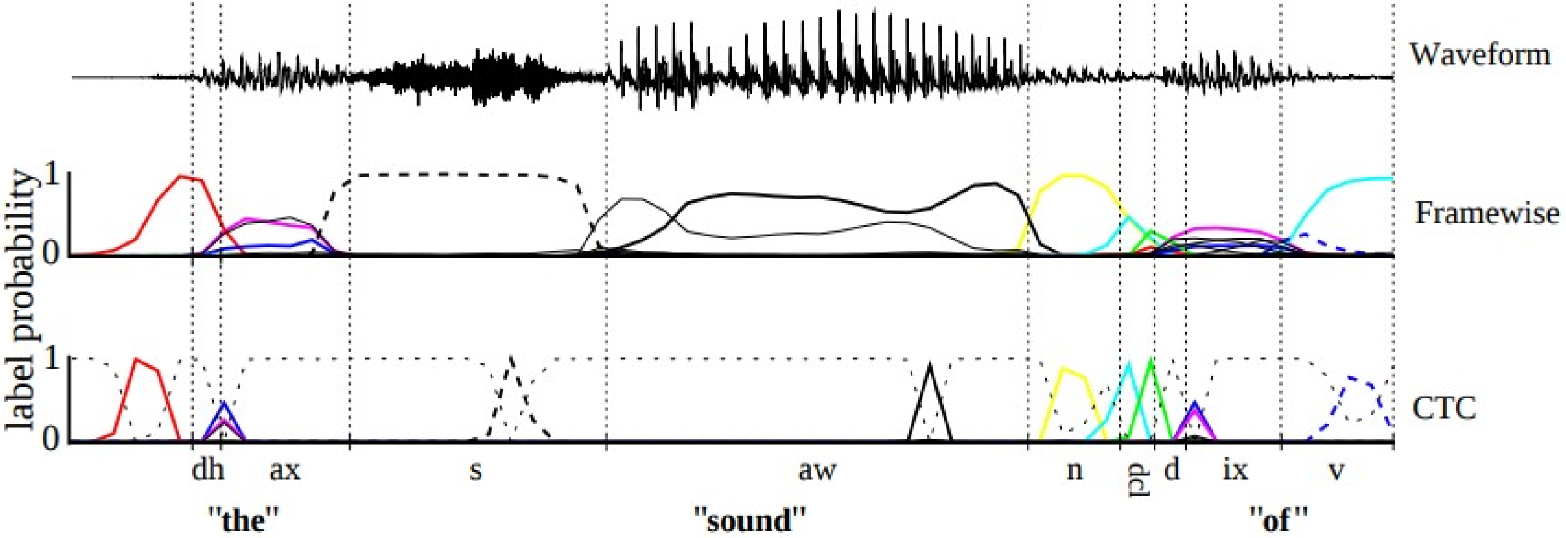
- Motivation
- Aligning Outputs And Labels
- CTC Loss
- Forward-Backward Algorithm
- Backprop
- Conclusions

Motivation

- Popular loss functions (such as MSE and XE) assume a one-to-one correspondence between the network's output and the target labels
- But what if there is no one-to-one correspondence? (as in speech recognition, or on-line handwriting recognition)
- Need a way to find the alignment between network outputs and target labels

Connectionist Temporal Classification (CTC)

- Given a sequence of N inputs and M labels, compute the loss between the N outputs and M labels

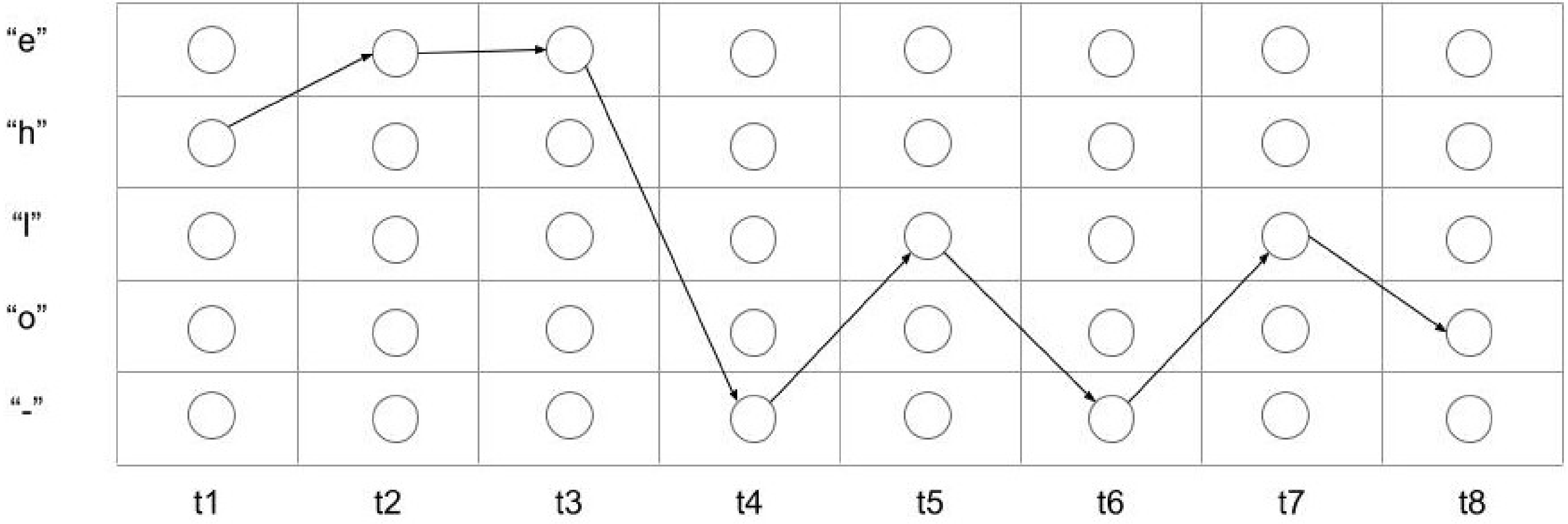


How To Align Outputs And Labels?

- Collapse repeated letters together
- Define a new token called “Blank” (will be represented as “-”)
- Assuming an input of length 8, we can define a function such that:
    - f(“hell-loo”) = f(“hel-lo”) = “hello”
    - f(“hellllloo”) = “helo”
    - f(“cc-a--tt”) = f(“c-a-t”) = “cat”
    - f(“-c-a-t--”) = f(“-c-a-t-”) = “cat”
    - f(“c-aaa-at”) = f(“c-a-at”) = “caat”

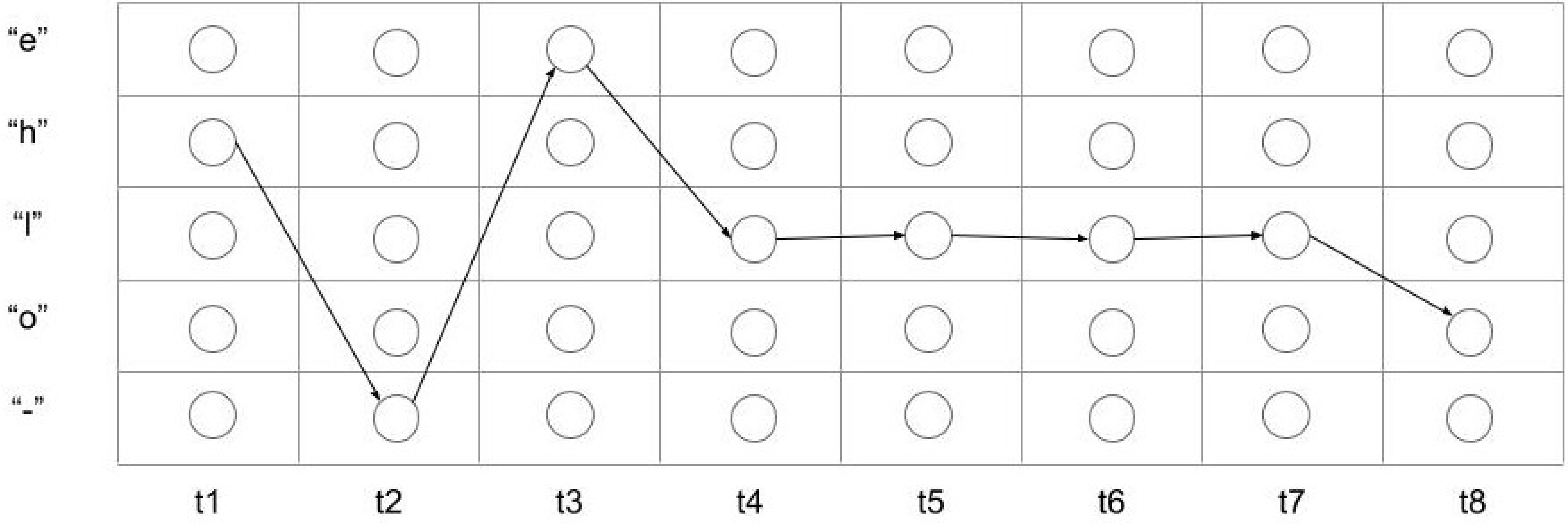
Some Examples

- f(“hee-l-lo”) = “hello”



Some Examples

- f(“h-ellllo”) = “helo”

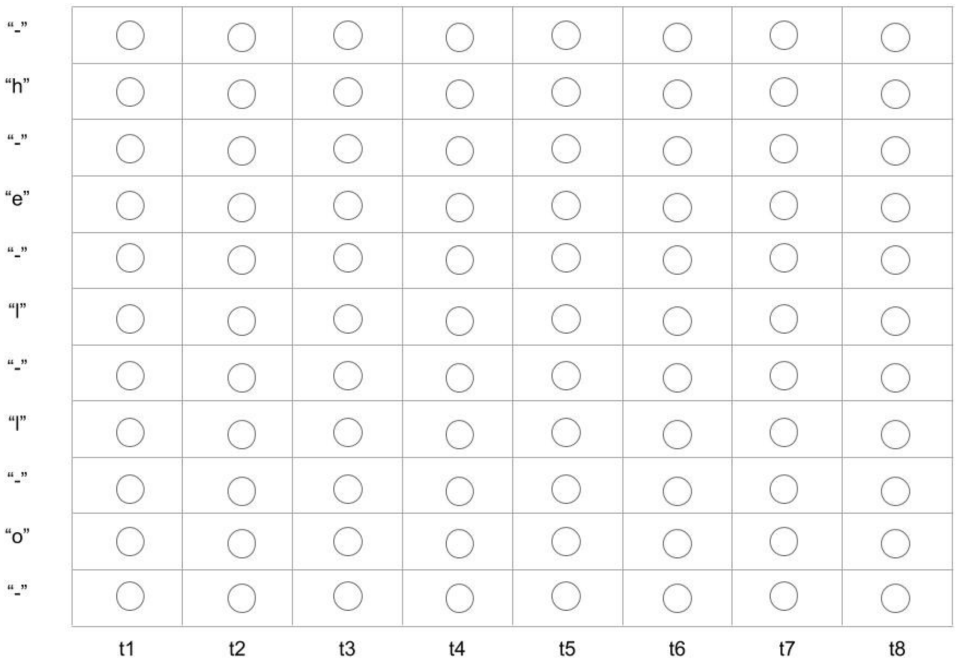


There are 5^8 ways to go from t1 to t8, that’s **390,625** possible paths!

CTC Loss

- CTC loss = -**ln p(W)**
- So for the word “hello”, we would like to minimize **-ln p(“hello”)**
- Cannot be solved trivially, otherwise number of paths explode
- What can be done then? **Dynamic programming**

How Is This Done In Practice?

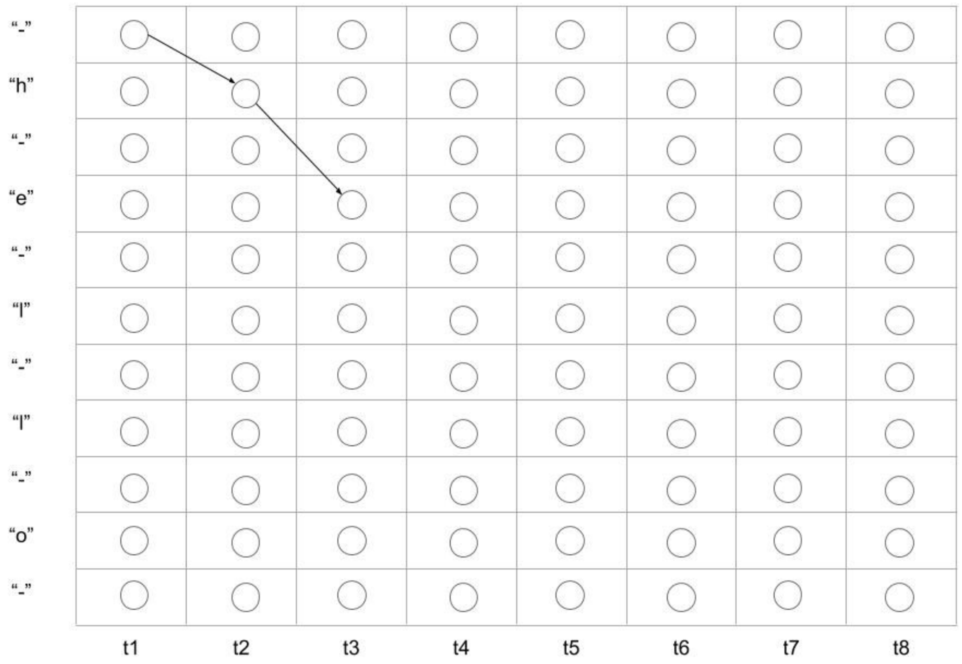


How Is This Done In Practice?

- f(“-hel-lo-”) = “hello”

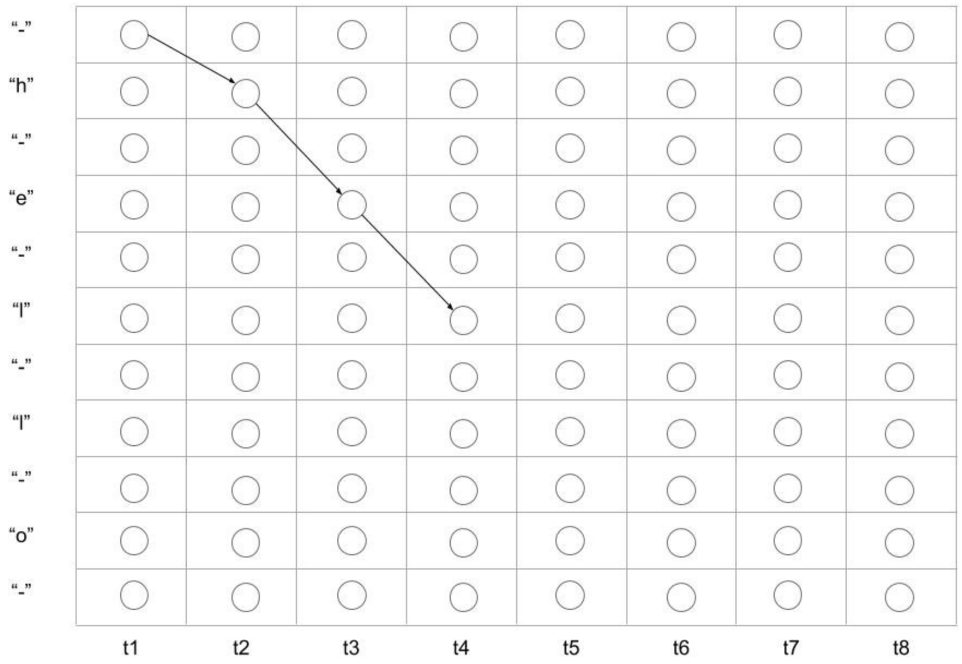
How Is This Done In Practice?

- f(“-hel-lo-”) = “hello”



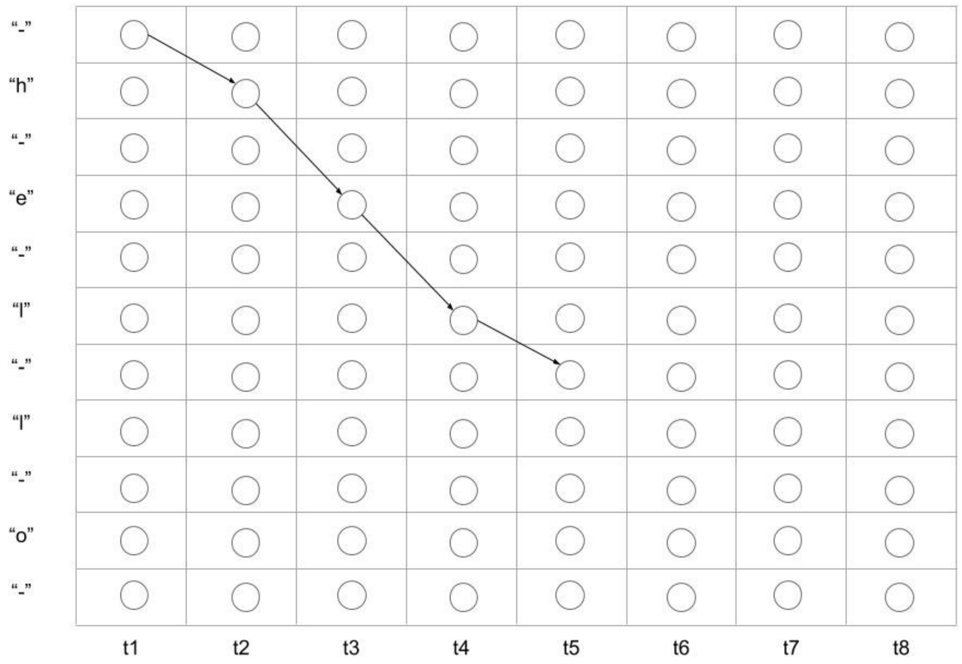
How Is This Done In Practice?

- f(“-hel-lo-”) = “hello”



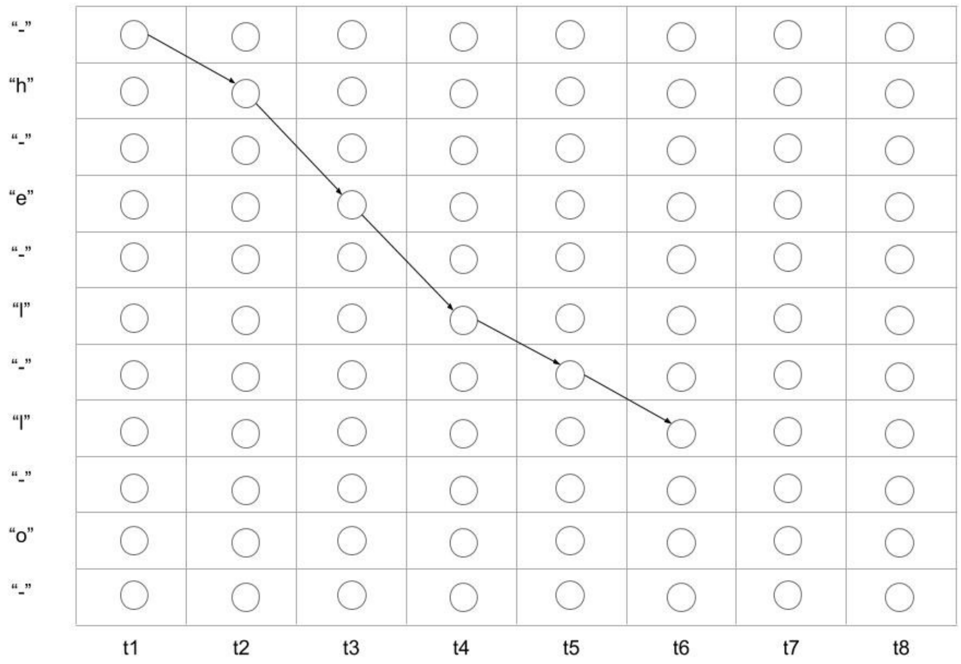
How Is This Done In Practice?

- f("-hel-lo-") = "hello"



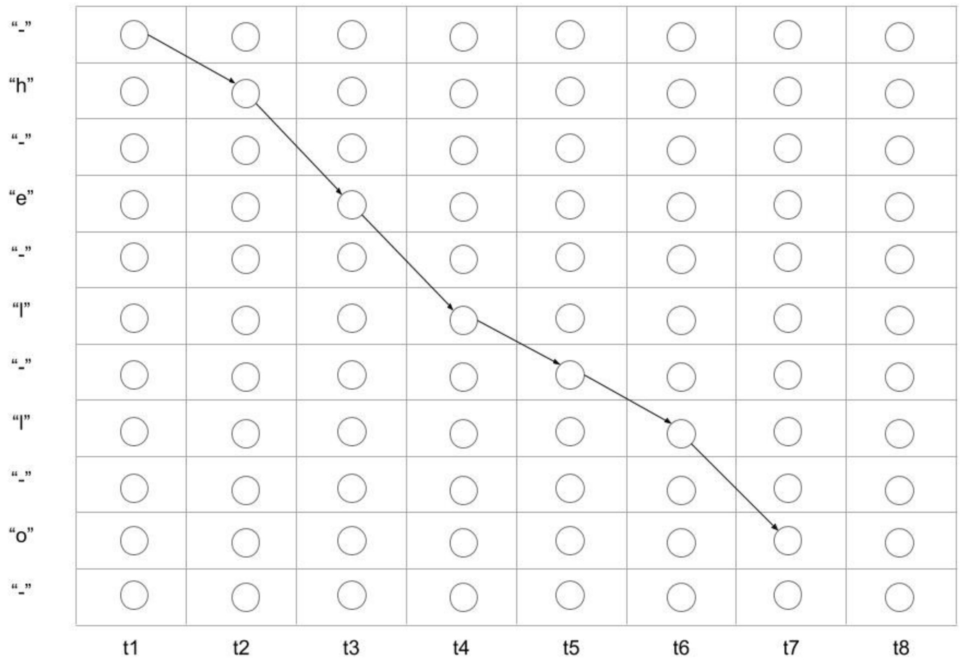
How Is This Done In Practice?

- f(“-hel-lo-”) = “hello”



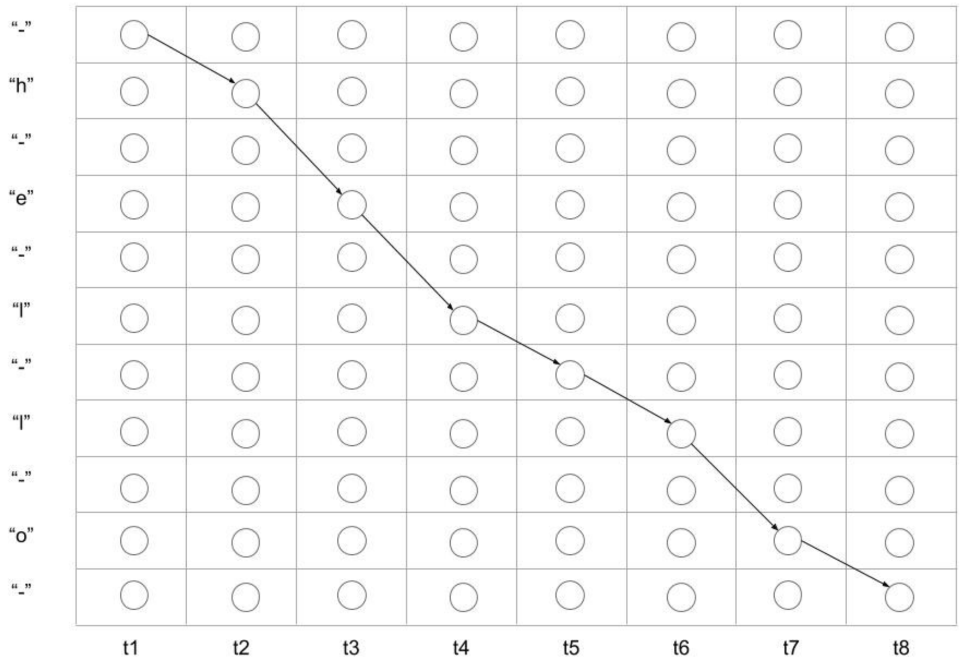
How Is This Done In Practice?

- f(“-hel-lo-”) = “hello”



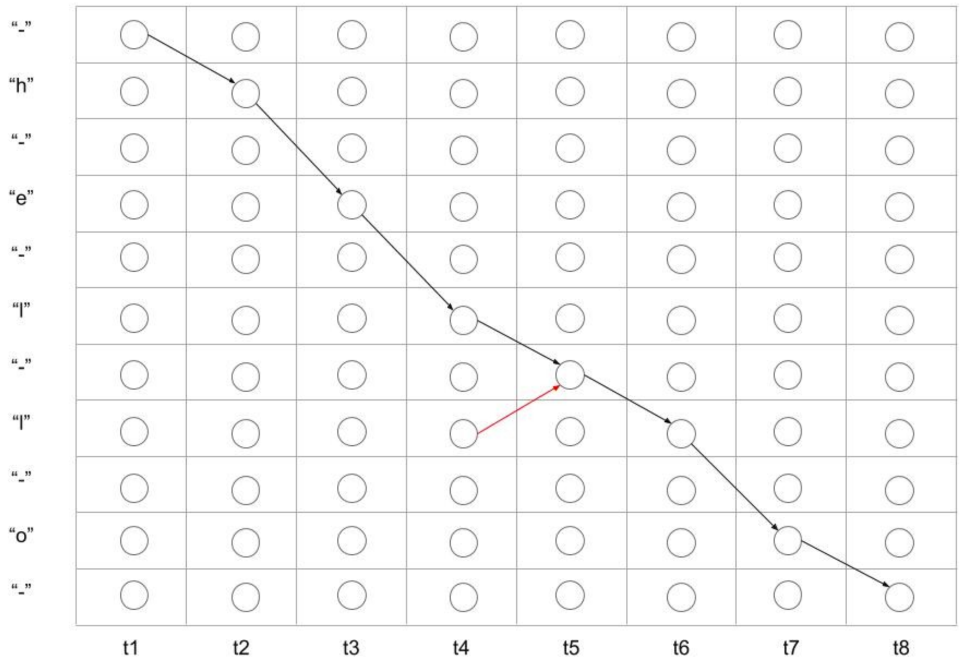
How Is This Done In Practice?

- f("-hel-lo-") = "hello"



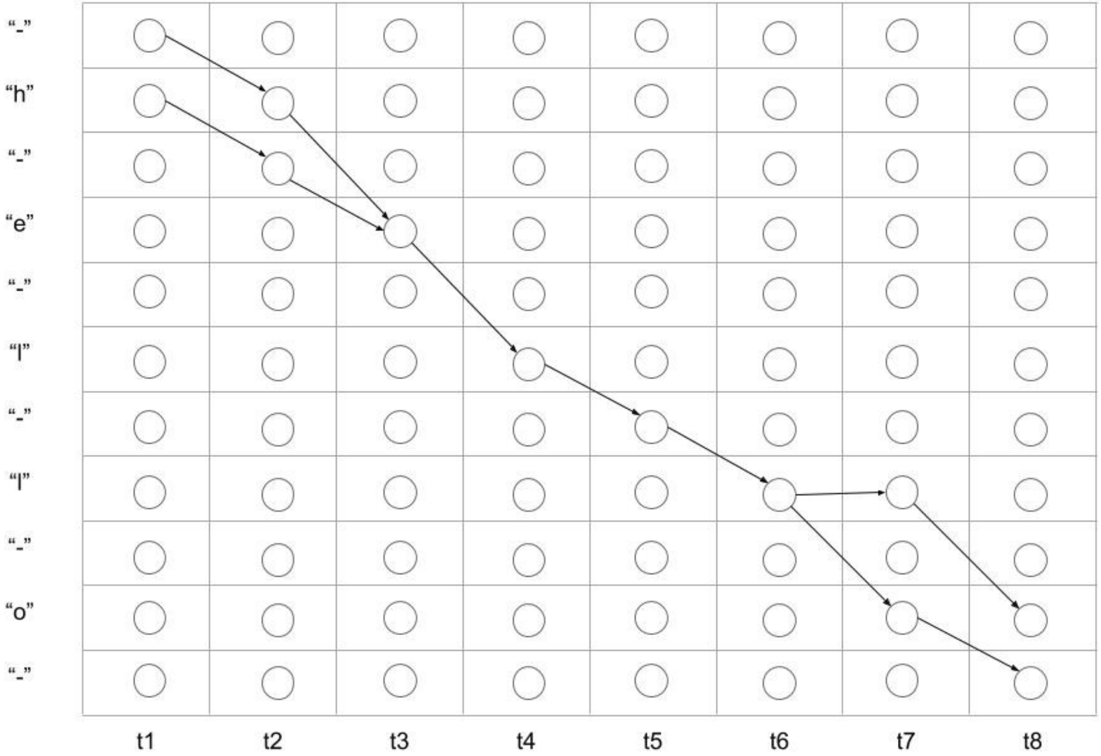
How Is This Done In Practice?

- f("-hel-lo-") = "hello"



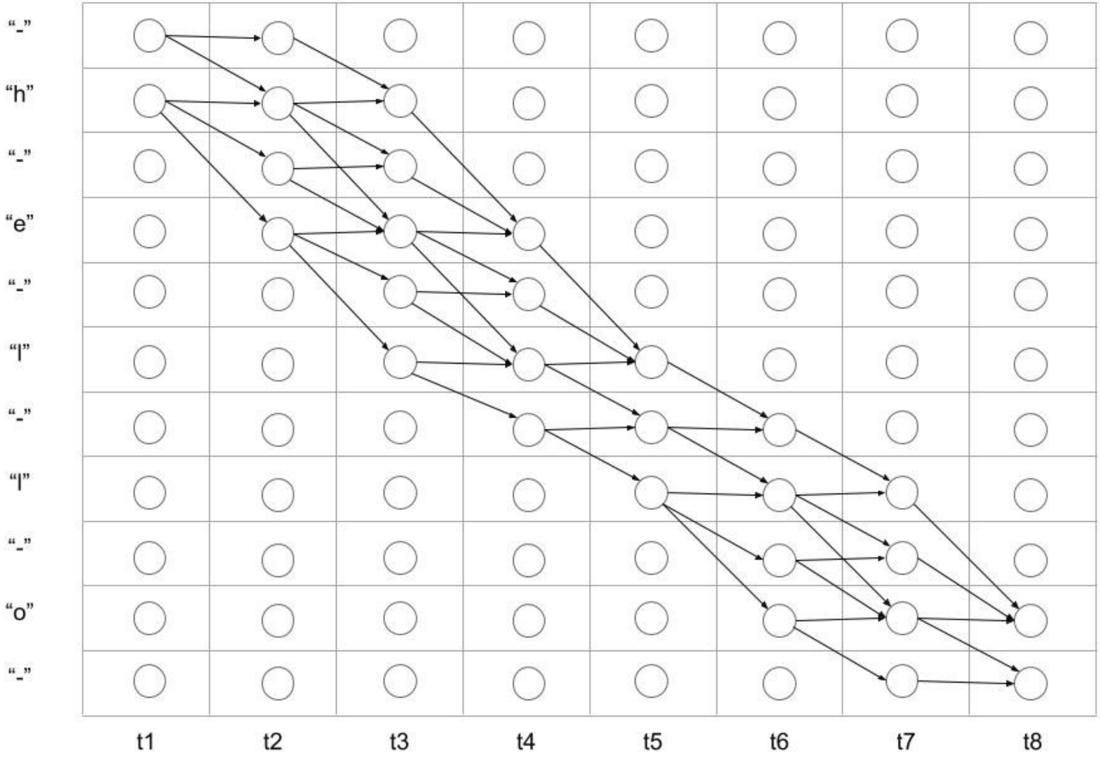
Alternative Paths

- f(“-hel-lo-”) = “hello”
- f(“-hel-llo”) = “hello”
- f(“h-el-llo”) = “hello”
- f(“h-el-lo-”) = “hello”



All Possible Paths

- Captures variations in pronunciation
- From f(“--hel-lo”)
- To f(“hel-lo--”)
- Note f(“hel---lo”)



CTC Loss Components

$$\alpha_t(s)$$

- Forward Variable: Calculates the total probability from the first timestep till timestep **t** and token **s**

$$\beta_t(s)$$

- Backward Variable: Calculates the total probability from timestep **t** and token **s** till last timestep

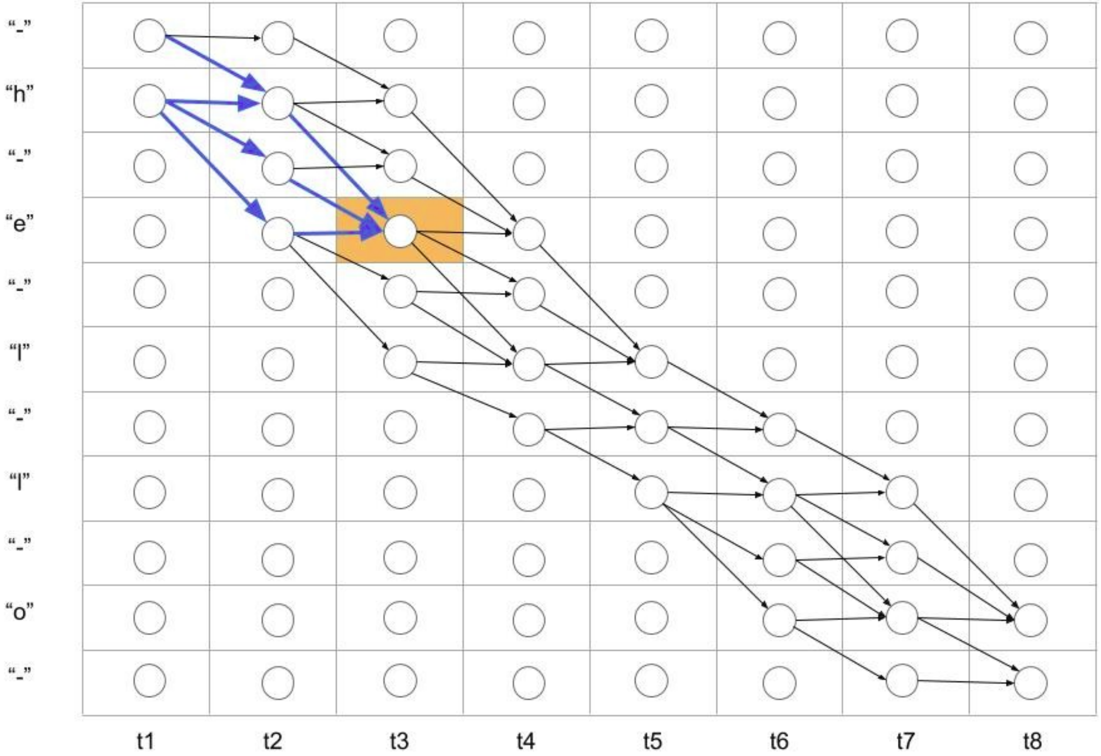
Forward Calculation Example 1

- Let's calculate $\alpha_3(4)$
- There are 4 paths

$$p("-he") = y^1_{-} . y^2_h . y^3_e$$
$$p("hhe") = y^1_h . y^2_h . y^3_e$$
$$p("h-e") = y^1_h . y^2_{-} . y^3_e$$
$$p("hee") = y^1_h . y^2_e . y^3_e$$

- Final probability is

$$p("-he") + p("hhe") + p("h-e") + p("hee")$$



Forward Calculation Example 1

$$p("-he") + p("hhe") + p("h-e") + p("hee")$$

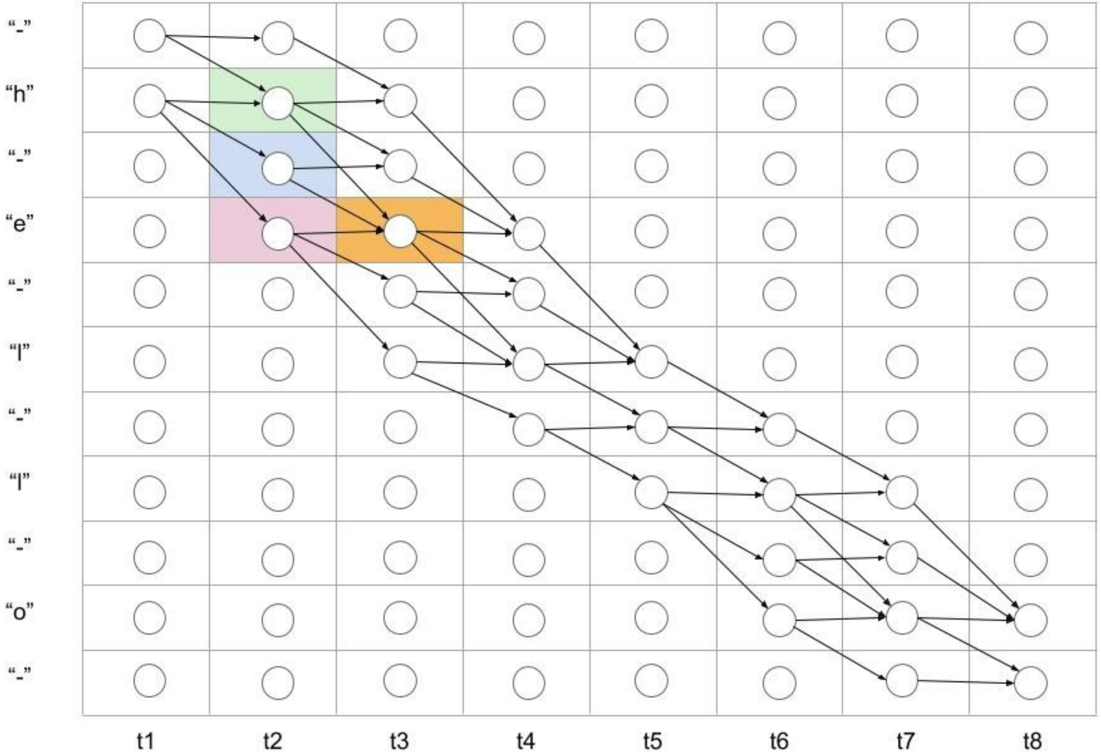
- This expression can also be calculated using dynamic

  programming

  $$\boxed{\alpha_3(4) = (\alpha_2(4) + \alpha_2(3) + \alpha_2(2))\ .\ y_e^3}$$

- Or in general

  $$\boxed{\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2))\ .\ y^t_{seq(s)}}$$



Forward Calculation Example 2

- The top component can be calculated using dynamic programming

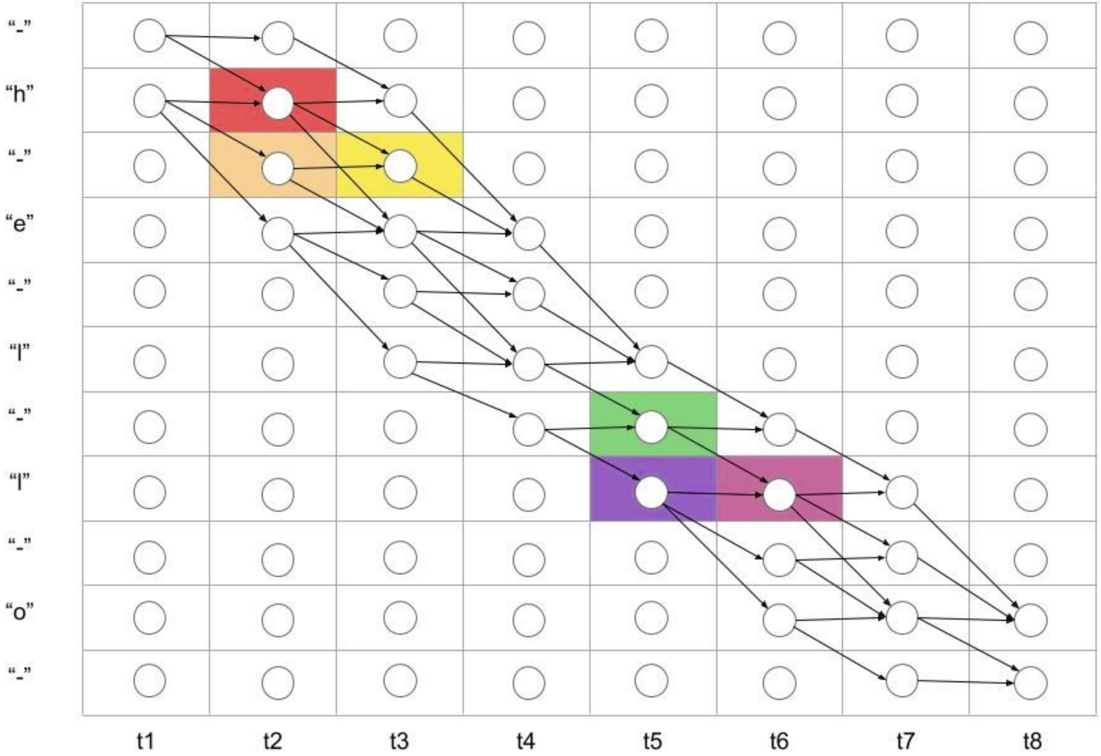
$$\alpha_3(3) = (\alpha_2(3) + \alpha_2(2))\ .\ y^3_{-}$$

- Or in general (for both)

$$\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1))\ .\ y^t_{seq(s)}$$

- Note that

$$p(\text{"hello"}) = \alpha_8(10) + \alpha_8(11)$$

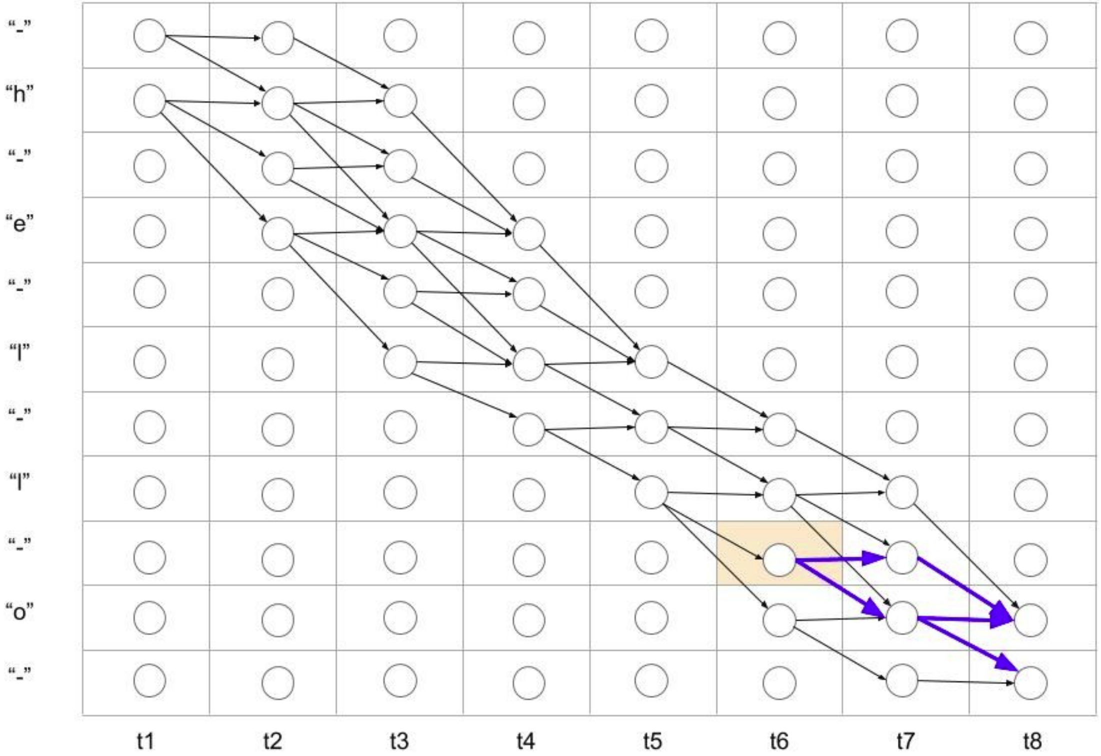


Backward Calculation Example

- This phase is just the opposite of the forward phase

$$\boxed{\beta_6(9) = p("--o") + p("-oo") + p("-o-")}$$

- The same logic here applies as the forward calculation



Complete Path Calculation Example

- The forward pass gives

$$\alpha_3(2) = p("--h") + p("-hh") + p("hhh")$$

$$\alpha_3(2) = y^1_- y^2_- y^3_h + y^1_- y^2_h y^3_h + y^1_h y^2_h y^3_h$$

- The backward pass results in

$$\beta_3(2) = p("hel-lo") = y^3_h y^4_e y^5_l y^6_- y^7_l y^8_o$$

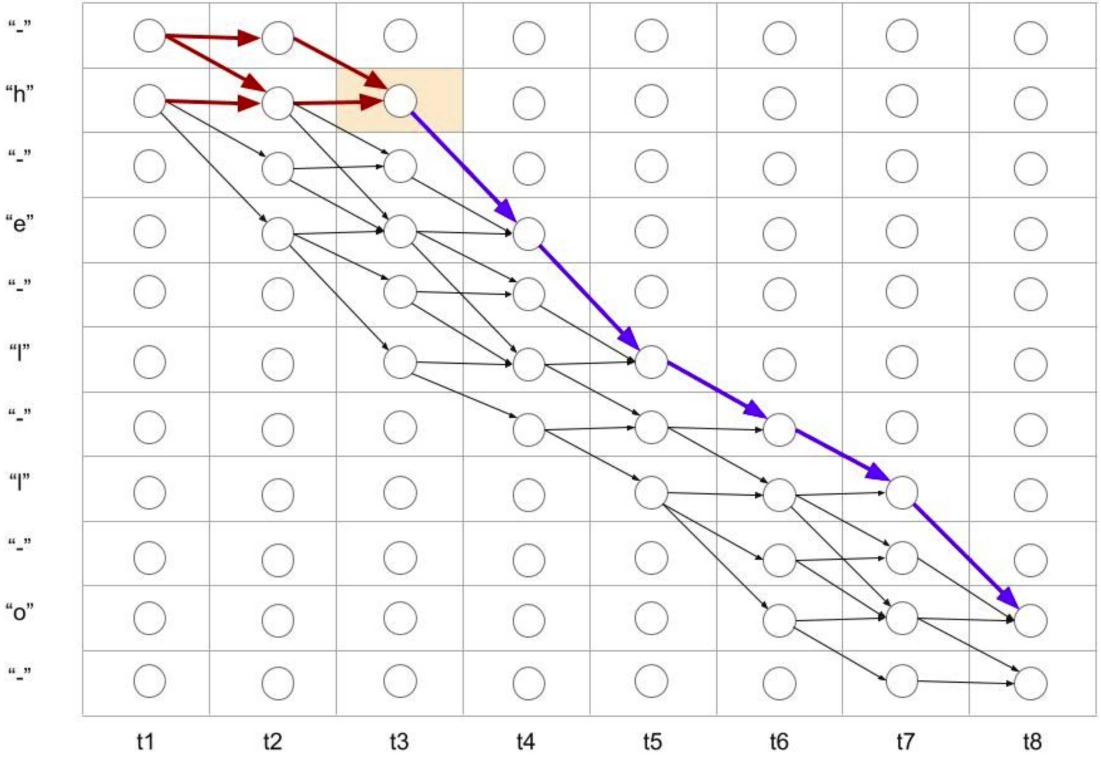
- The final result

$$\begin{aligned}\alpha_3(2).\beta_3(2) = & y^1_-.y^2_-.y^3_h.y^3_h.y^4_e.y^5_l.y^6_-.y^7_l.y^8_o \\ & + y^1_-.y^2_h.y^3_h.y^3_h.y^4_e.y^5_l.y^6_-.y^7_l.y^8_o \\ & + y^1_h.y^2_h.y^3_h.y^3_h.y^4_e.y^5_l.y^6_-.y^7_l.y^8_o\end{aligned}$$

$$\boxed{= [p("--hel-lo") + p("-hhel-lo") + p("hhhel-lo")].y^3_h}$$

- Total probability of all paths going through **h** at t=3

$$\frac{\alpha_3(2).\beta_3(2)}{y^3_h}$$



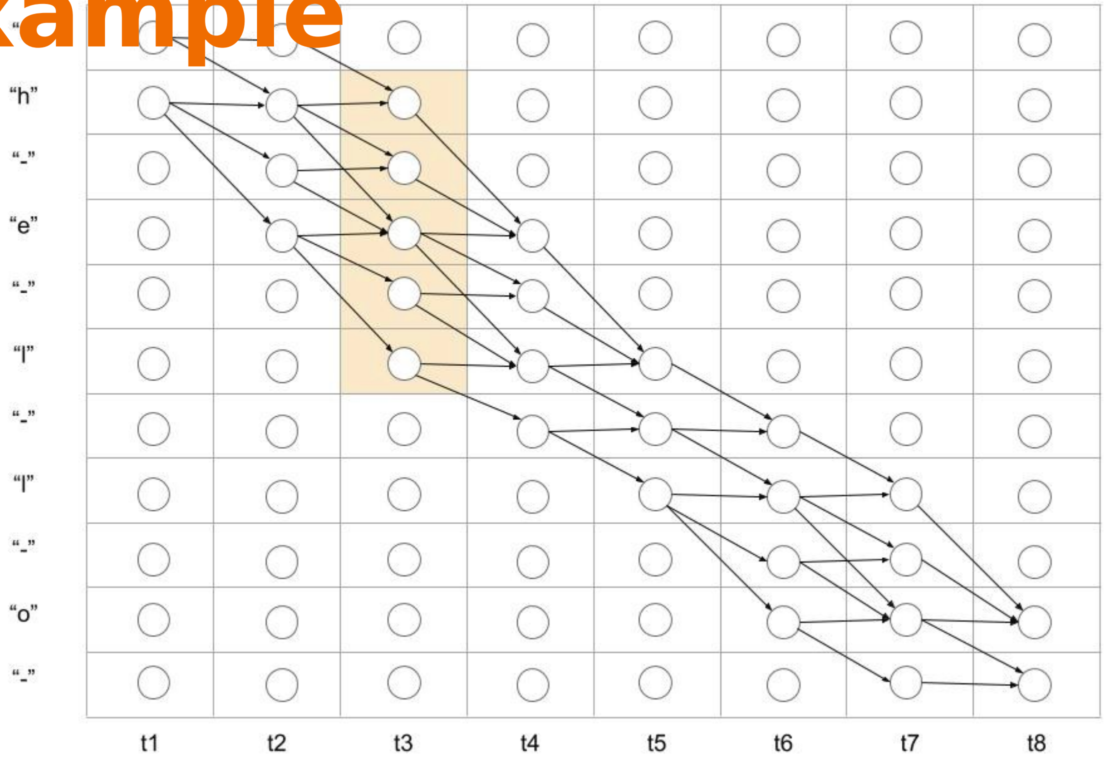
Calculating Loss At A Particular Timestep Example

- p("hello") at t=3 is the sum of the probabilities of all paths through all the symbols

$$p(\text{"}hello\text{"}) = \sum_{s=2}^{6} \frac{\alpha_3(s).\beta_3(s)}{y^3_{seq(s)}}$$

- In general, the probability of the ground truth label is

$$p(\text{"}hello\text{"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s).\beta_t(s)}{y^t_{seq(s)}}$$



How To Do Backprop?

$$\frac{\partial(-\ln p(\text{"hello"}))}{\partial y_k^t} = \frac{-1}{p(\text{"hello"})} \frac{\partial p(\text{"hello"})}{\partial y_k^t}$$

- Since $p(\text{"hello"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s).\beta_t(s)}{y_{seq(s)}^t}$

$$\frac{\partial p(\text{"hello"})}{\partial y_k^t} = \frac{-1}{{y_k^t}^2} \sum_{s:seq(s)=k} \alpha_t(s).\beta_t(s)$$

Backprop Example 1

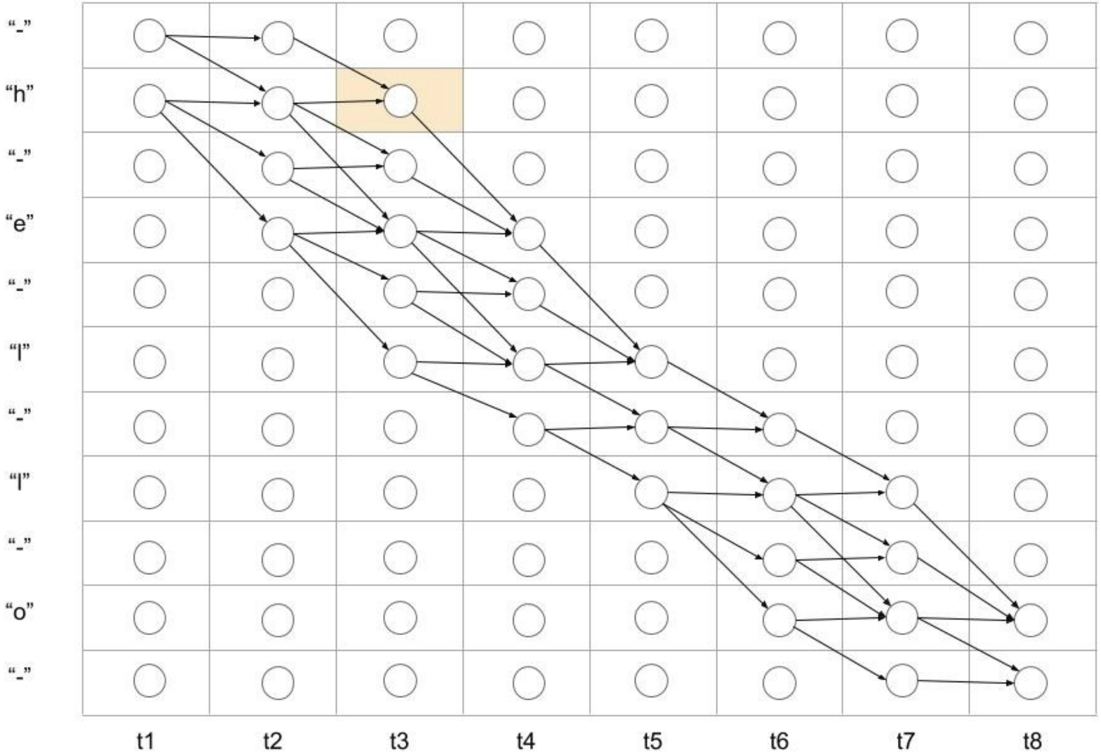
- Given

$$\frac{\partial p(\text{"hello"})}{\partial y_k^t} = \frac{-1}{{y_k^t}^2} \sum_{s:seq(s)=k} \alpha_t(s).\beta_t(s)$$

- For t=3, and k=**h**

$$\frac{\partial p(\text{"hello"})}{\partial y_h^3} = \frac{-1}{{y_h^3}^2} .\alpha_3(2).\beta_3(2)$$

Since **h** occurs only at s=2



Backprop Example 2

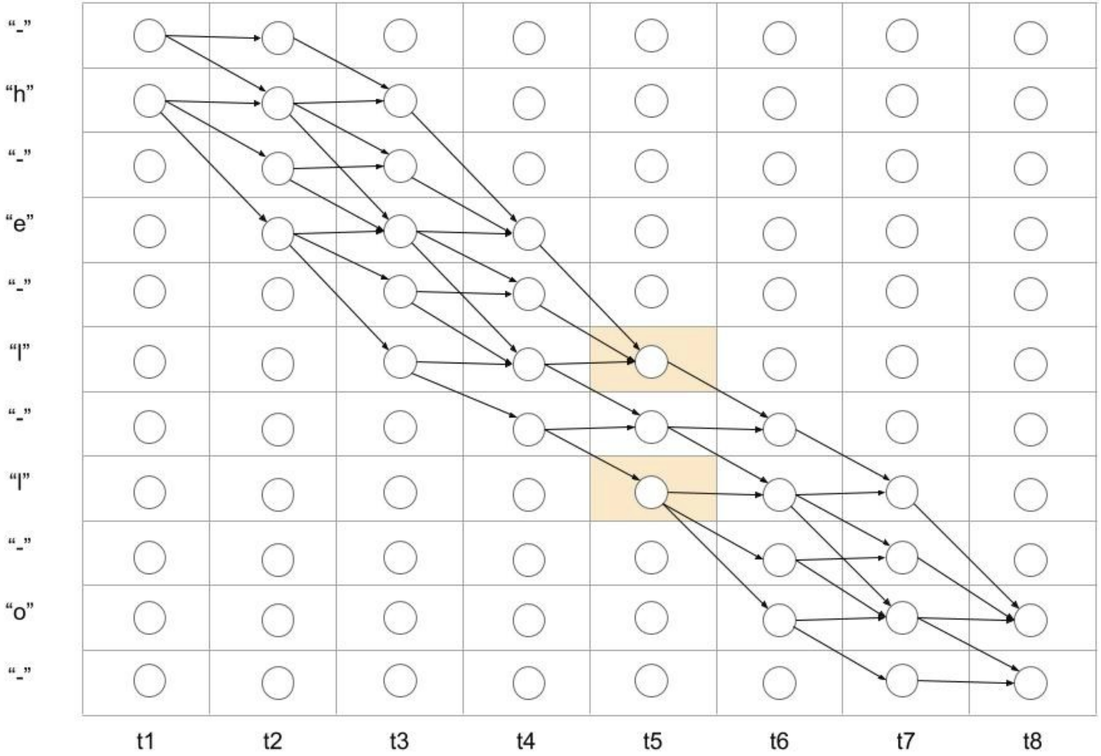
- Given

$$\frac{\partial p(\text{"hello"})}{\partial y_k^t} = \frac{-1}{{y_k^t}^2} \sum_{s:seq(s)=k} \alpha_t(s).\beta_t(s)$$

- For t=5, and k=**l**

$$\frac{\partial p(\text{"hello"})}{\partial y_l^5} = \frac{-1}{{y_l^5}^2}.(\alpha_5(6).\beta_5(6) + \alpha_5(8).\beta_5(8))$$

Since **l** occurs at s=6 and s=8



Conclusions

- CTC Loss allows training models on sequences whose number of inputs is different than the number of labels
- It makes use of dynamic programming to calculate path probabilities efficiently
- CTC treats every timestep independently

Th-aa-nk -Yo-uu--!

References

1. 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.
2. https://en.wikipedia.org/wiki/Connectionist_temporal_classification
3. https://www.youtube.com/watch?v=c86gfVGcvh4
4. https://www.youtube.com/watch?v=eYIL4TMAeRI