Wav2Letter - Implementing an open source E2E ASR in PyTorch

Assaf Mushkin

Everything you need to know

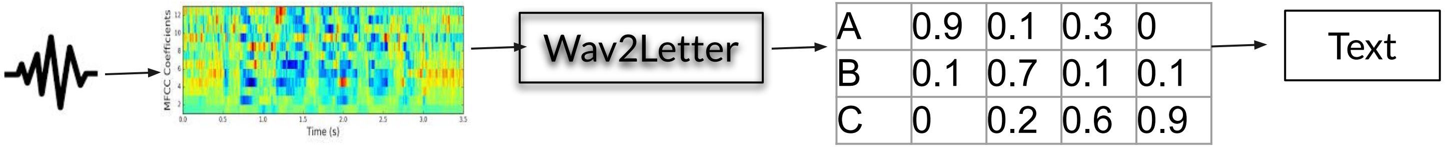
- Wav2Letter is an easy to understand, accurate model for ASR
- There are many improvements on it
- I’ve made a simple, minimalist PyTorch implementation available online

Agenda

- CTC and Wav2Letter
- Spinoffs and improvements
- Lessons from working with open source projects
- Live Demonstration!

Wav2Letter (2016 - FAIR)

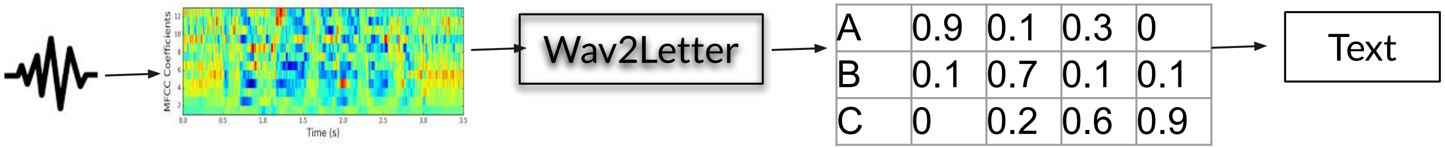
- Input spectrograms, Output is CTC (probability of each letter per frame)



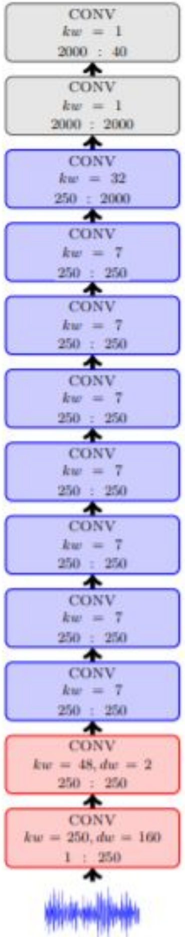
A	0.9	0.1	0.3	0
B	0.1	0.7	0.1	0.1
C	0	0.2	0.6	0.9

Wav2Letter (2016 - FAIR)

- Input spectrograms, Output is CTC (probability of each letter per frame)
- Fully convolutional, just a long stack of convolutions

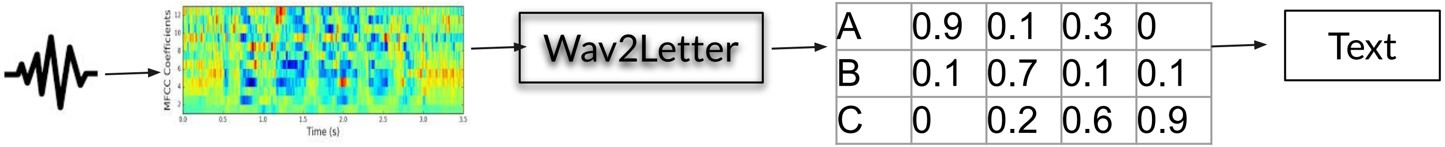


A	0.9	0.1	0.3	0
B	0.1	0.7	0.1	0.1
C	0	0.2	0.6	0.9

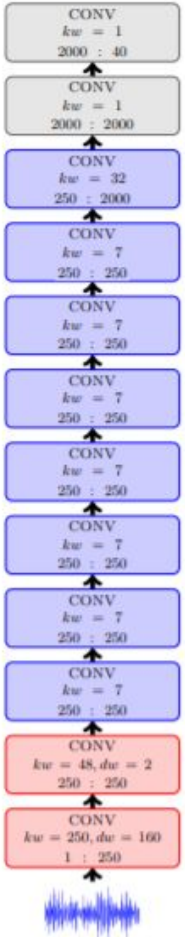


Wav2Letter (2016 - FAIR)

- Input spectrograms, Output is CTC (probability of each letter per frame)
- Fully convolutional, just a long stack of convolutions
- MFCC features (spectrogram - only slightly worse)



A	0.9	0.1	0.3	0
B	0.1	0.7	0.1	0.1
C	0	0.2	0.6	0.9



CTC - “Connectionist Temporal Classification”

- For each frame / window, predict probability (or score) of each letter.
  - Add unique “blank” character, _
- When interpreting, drop consecutive identical letters, then drop blanks.

Frame	1	2	3	4	5
A	0.6	0.3	...	...	...
B	0.3	0.2	...	...	...
blank	0.1	0.5	...	...	...

__ABA -> ABA
AA_BA -> ABA
AB_BA -> ABBA
AAA_BB_A__ -> ABA

CTC example

CTC : **HH_E_LLLLL_LL_OO   HH_O_WWW ARRRR____EE YOUUU**

OUTPUT: **HELLO HOW ARE YOU**

CTC - cont.

- CTC can be interpreted “greedily” - pick letter for each frame independently, then run reduction

A	**0.9**	0.1	0.3	0
B	0.1	**0.7**	0.1	0.1
_	0	0.2	**0.6**	**0.9**

CTC - cont.

- CTC can be interpreted “greedily” - pick letter for each frame independently, then run reduction
- Can also be decoded exhaustive - find output string with highest total score, considering all possible reductions

CTC - cont.

- CTC can be interpreted “greedily” - pick letter for each frame independently, then run reduction
- Can also be decoded exhaustive - find output string with highest total score, considering all possible reductions
- Greedy results aren’t the best, but are much faster to compute

CTC - cont.

- CTC can be interpreted “greedily” - pick letter for each frame independently, then run reduction
- Can also be decoded exhaustive - find output string with highest total score, considering all possible reductions
- Greedy results aren’t the best, but are much faster to compute
- Practically, we use beam search, weighed with a LM
  - This adds a bunch more hyperparameters to the process, can be found independently of the acoustic model

ASG - CTC but with a twist

- Used in Wav2letter (2016)

ASG - CTC but with a twist

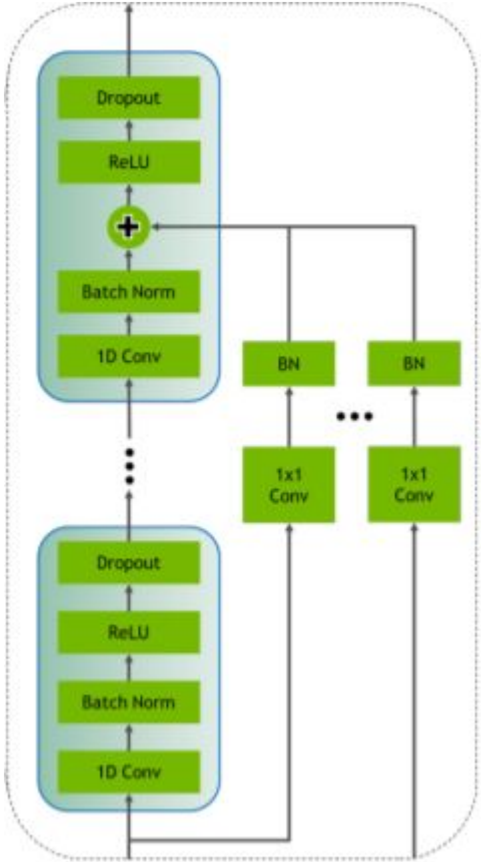
- Used in Wav2letter (2016)
- **TL;DR: Rarely used in future works, use CTC instead.**

ASG - CTC but with a twist

- Used in Wav2letter (2016)
- **TL;DR: Rarely used in future works, use CTC instead.**
- No usage of blank char, instead repeat last character and use 2 and 3 for repetition (“**bo2k**” instead of “**book**”)
    - This defines a different “reduce” function, and a different simpler decoding algorithm
- Global normalization instead of per-frame normalization
    - This isn’t connected to ASG necessarily, but was in the same paper.

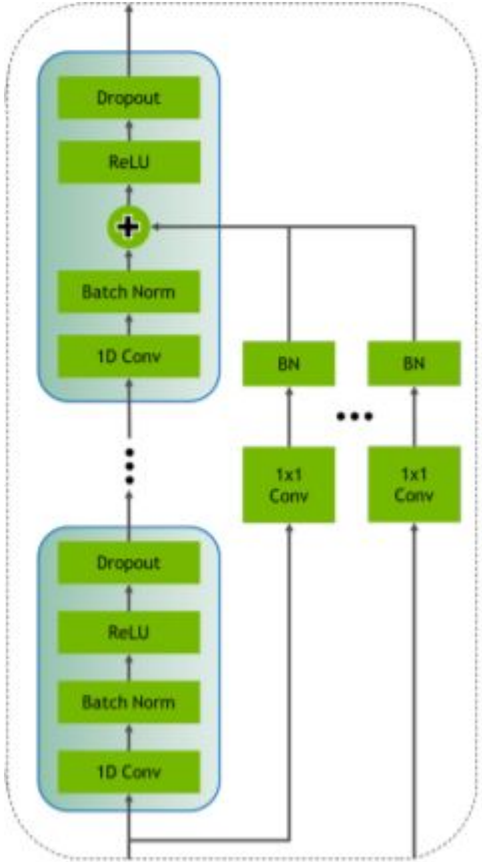
Jasper (2019) - Bigger! Better!

- Pretty much Wav2Letter, but with residual connections
- “JasperBlock”:
  - **R** times: [Conv1d,BN, Relu,Dropout]
  - + residual connection
- Jasper architecture is **B** repetitions of JasperBlock



Jasper (2019) - Bigger! Better!

- Pretty much Wav2Letter, but with residual connections
- “JasperBlock”:
  - **R** times: [Conv1d,BN, Relu,Dropout]
  - + residual connection
- Jasper architecture is **B** repetitions of JasperBlock
- From the paper, **B**=10 and **R**=3 (LibriSpeech) or **R**=5 (WSJ)
- 3X10 = 30 convolutions, compared to 12 in Wav2letter



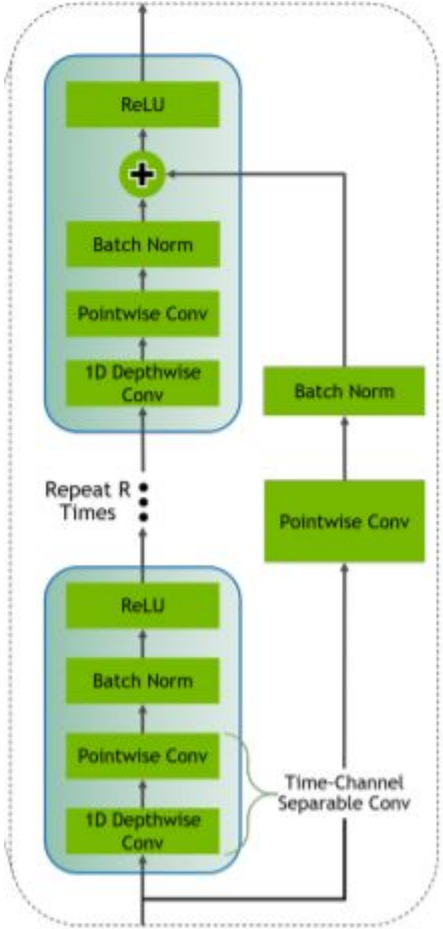
Novograd - Jasper’s Optimizer

- It’s Adam, but momentum is calculated per layer instead of per weight
- Sometimes better end results, but mostly just speedup for ASR task (anecdotally, 30% less epochs to reach same loss)
- Current rumour: No speedup in other tasks, compared to Adam

- Not implemented in PyTorch yet, but is open-source and available online

QuartzNet (2019 - 1 month later)

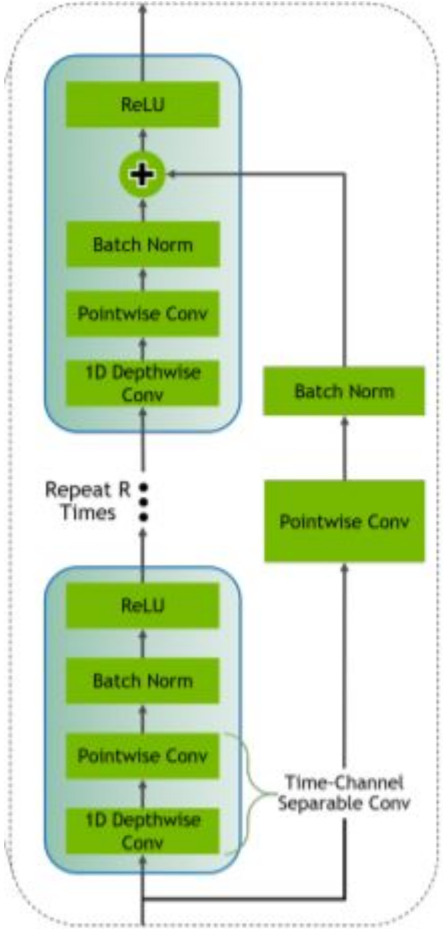
- Jasper + 1D Time-Depth-Separable convolutions *



** First use of TDS was in FAIR in a different article*

QuartzNet (2019 - 1 month later)

- Jasper + 1D Time-Depth-Separable convolutions *
- Instead of passing each convolution a matrix of size [window X channels], break into two convolutions.
- First convolve over time (treating each channel the same), then convolve over channels with a window of 1. Called depthwise and pointwise convolutions

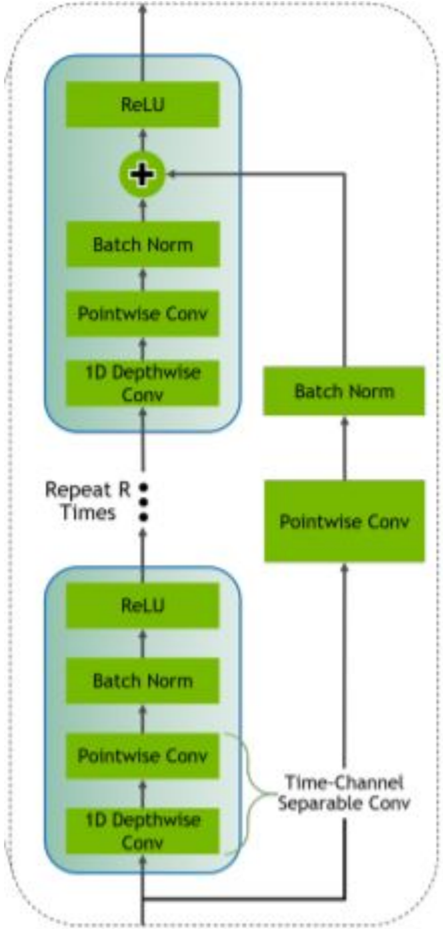


* *First use of TDS was in FAIR in a different article*

QuartzNet (2019 - 1 month later)

- Jasper + 1D Time-Depth-Separable convolutions *
- Instead of passing each convolution a matrix of size [window X channels], break into two convolutions.
- First convolve over time (treating each channel the same), then convolve over channels with a window of 1. Called depthwise and pointwise convolutions
- Drastically decrease the number of parameters (20 mil. Instead of 100)

** First use of TDS was in FAIR in a different article*



Wav2letter improvements

- Shorter future context, speedups - better for online decoding (link)
- Semi-supervised training (link) - also included ResNet style acoustic models
- Lexicon free decoding (link) - better performance on OOV

Sidenote - integrating open source projects

- Actually not that hard

Sidenote - integrating open source projects

- Actually not that hard
- Projects are not libraries, different mindset required

Sidenote - integrating open source projects

- Actually not that hard
- Projects are not libraries, different mindset required
- It will not work out-of-the-box, you will have to read and edit the code

Sidenote - integrating open source projects

- Actually not that hard
- Projects are not libraries, different mindset required
- It will not work out-of-the-box, you will have to read and edit the code
- They made mistakes, bad assumptions, or are out-of-date

Sidenote - integrating open source projects

- Actually not that hard
- Projects are not libraries, different mindset required
- It will not work out-of-the-box, you will have to read and edit the code
- They made mistakes, bad assumptions, or are out-of-date
- The price we pay for cutting-edge developments

Show, don’t tell

- Demonstration training for a very small dataset
- Tensorboard is useful for visualizing training process

- Repository available at https://github.com/assafmu/wav2letter_pytorch
- PyTorch implementation, intentionally minimalist