OPENSEQ2SEQ: A DEEP LEARNING TOOLKIT FOR SPEECH RECOGNITION, SPEECH SYNTHESIS, AND NLP

Oleksii Kuchaiev, Boris Ginsburg

3/19/2019

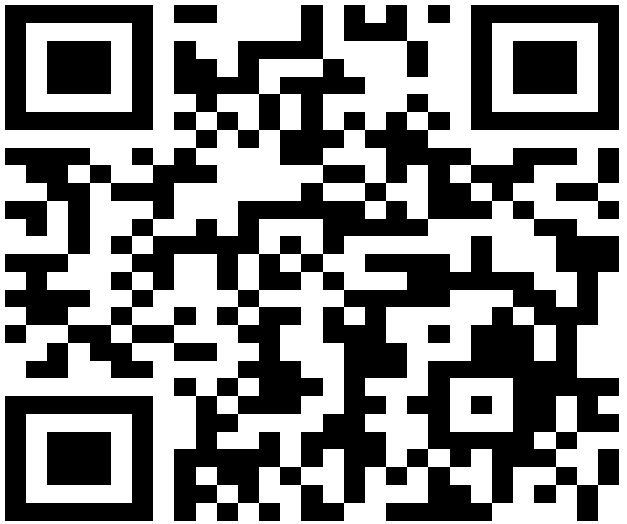
Contributors

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Chip Nguyen, Jonathan Cohen, Edward Lu, Ravi Gadde, Igor Gitman, Vahid Noroozi, Siddharth Bhatnagar, Trevor Morris, Kathrin Bujna, Carl Case, Nathan Luehr, Dima Rekesh

Contents

- **Toolkit overview**
  - Capabilities
  - Architecture
  - Mixed precision training
  - Distributed training
- **Speech technology in OpenSeq2Seq**
  - Intro to Speech Recognition with DNN
  - Jasper model
  - Speech commands

Code, Docs and Pre-trained models:
https://github.com/NVIDIA/OpenSeq2Seq



Capabilities

1. TensorFlow-based toolkit for sequence-to-sequence models
2. Mixed Precision training
3. Distributed training: multi-GPU and multi-node
4. Extendable

Supported Modalities

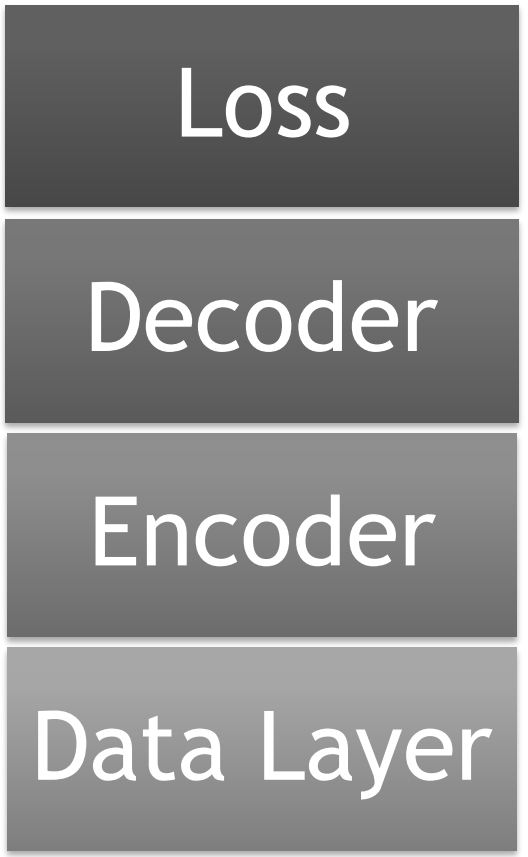
- Automated Speech Recognition
  - DeepSpeech2, Wav2Letter+, Jasper
- Speech Synthesis
  - Tacotron2, WaveNet
- Speech Commands
  - Jasper, ResNet-50
- Neural Machine Translation
  - GNMT, ConvSeq2Seq, Transformer
- Language Modelling and Sentiment Analysis
- Image Classification

Core Concepts

Flexible Python-based config file

```
42    "encoder": BidirectionalRNNEncoderWithEmbedding,
43    "encoder_params": {
44      "initializer": tf.glorot_uniform_initializer,
45      "core_cell": tf.nn.rnn_cell.LSTMCell,
46      "core_cell_params": {
47          "num_units": 512,
48          "forget_bias": 1.0,
49      },
50      "encoder_layers": 2,
51      "encoder_dp_input_keep_prob": 0.8,
52      "encoder_dp_output_keep_prob": 1.0,
53      "encoder_use_skip_connections": False,
54      "src_emb_size": 512,
55      "use_swap_memory": True,
56    },
57
58    "decoder": RNNDecoderWithAttention,
59    "decoder_params": {
60      "initializer": tf.glorot_uniform_initializer,
61      "core_cell": tf.nn.rnn_cell.LSTMCell,
62      "core_cell_params": {
63          "num_units": 512,
64          "forget_bias": 1.0,
65      },
66      "decoder_layers": 2,
```

Seq2Seq model



How to Add a New Model

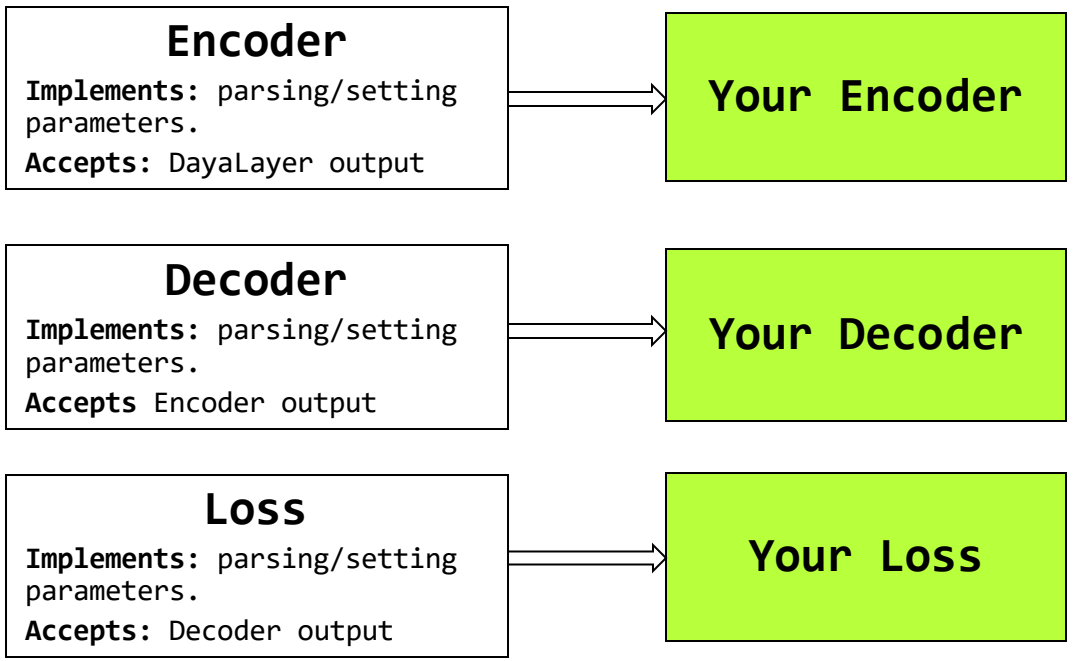
Contributions Welcome! https://nvidia.github.io/OpenSeq2Seq

Supported Modalities:

- Speech to Text
- Text to Speech
- Translation
- Language modeling
- Image classification

For supported modalities:

1. Subclass from *Encoder, Decoder and/or Loss*
2. Implement your idea



You get: logging, mixed precision and distributed training from toolkit.
No need to write any code for it. You can mix various encoders and decoders.

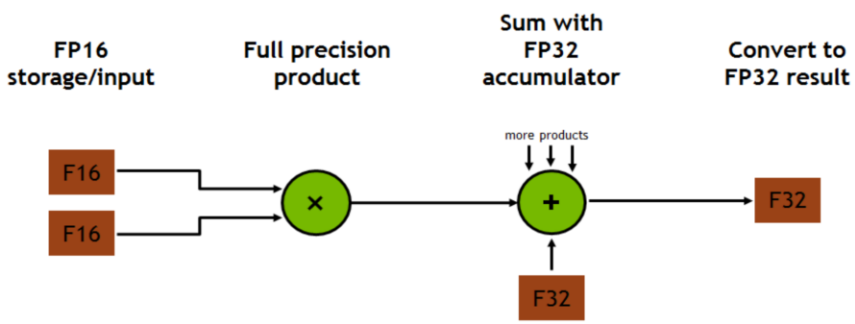
Mixed Precision Training in OpenSeq2Seq

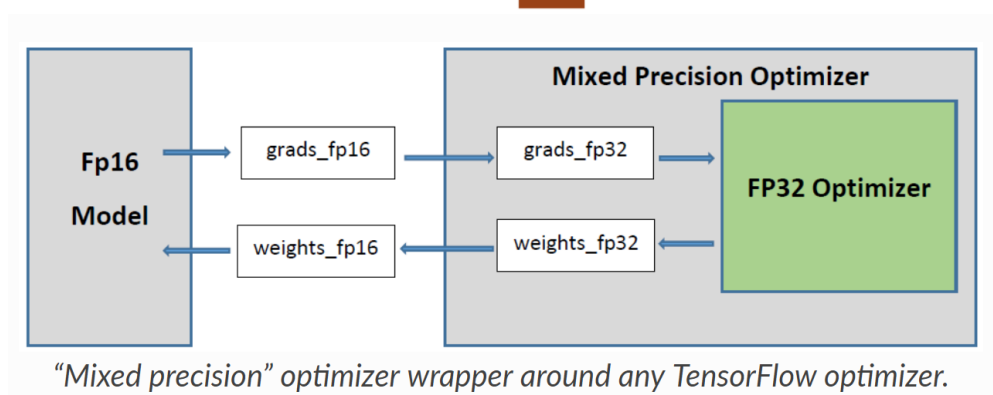
- **Train SOTA models faster and using less memory**
- **Keep hyperparameters and network unchanged**

Mixed Precision training*:

1. Different from “native” tf.*float16*
2. Maintain tf.*float32* “master copy” for weights update
3. Use the tf.*float16* weights for forward/backward pass
4. Apply loss scaling while computing gradients to prevent underflow during backpropagation
5. NVIDIA’s Volta or Turing GPU



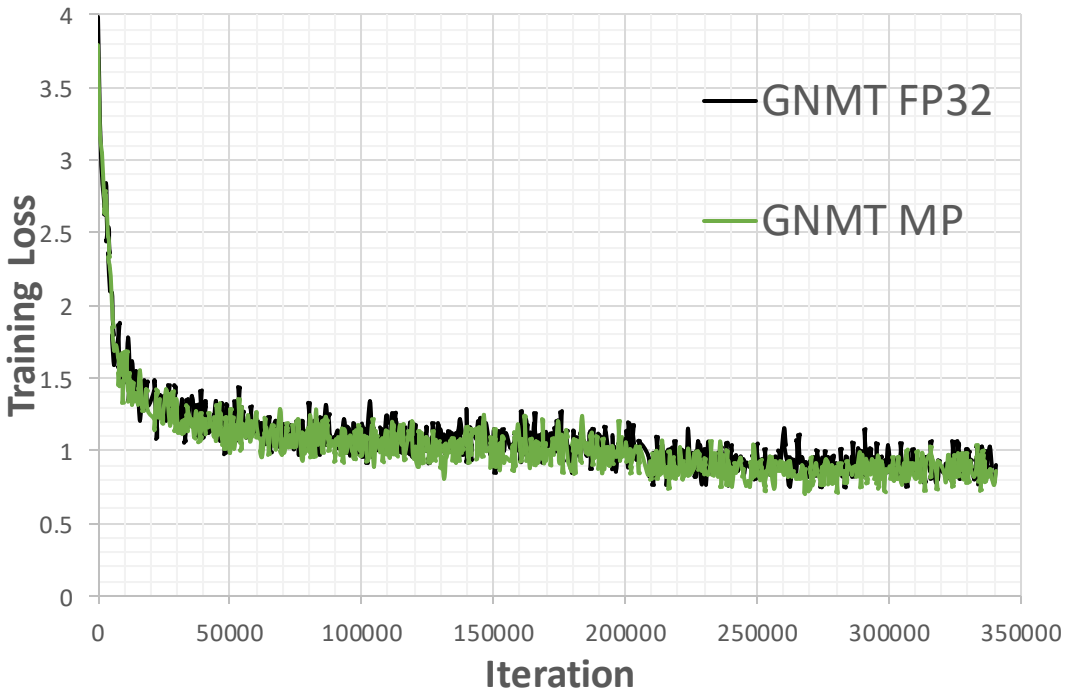


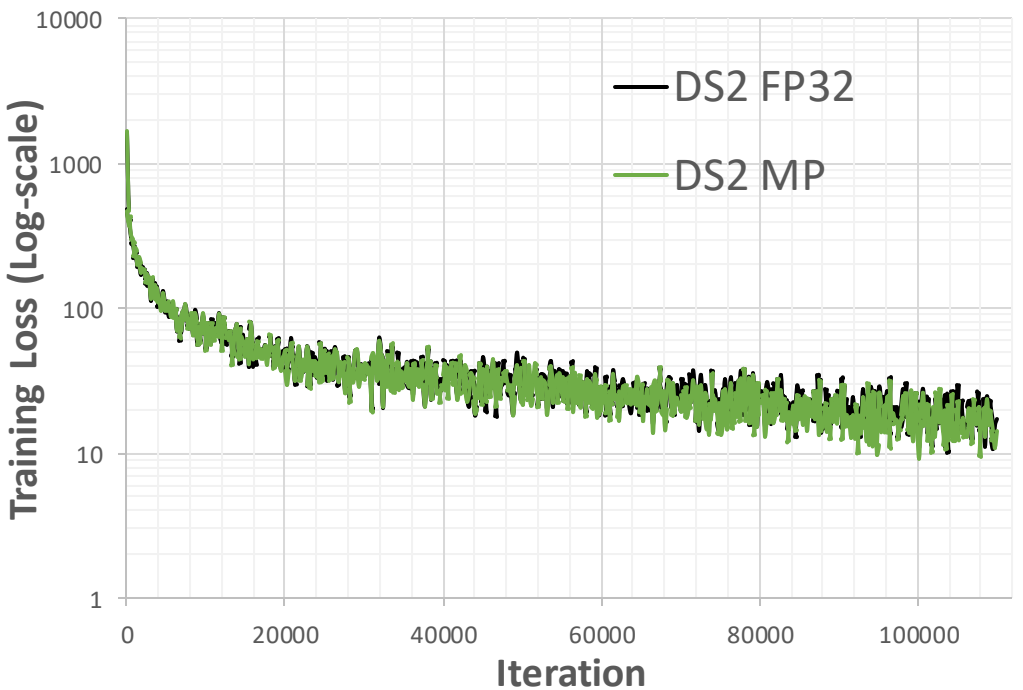
“Mixed precision” optimizer wrapper around any TensorFlow optimizer.

* Micikevicius et al. “Mixed Precision Training” *ICLR 2018*

Mixed Precision Training

Convergence is the same for *float32* and *mixed precision* training





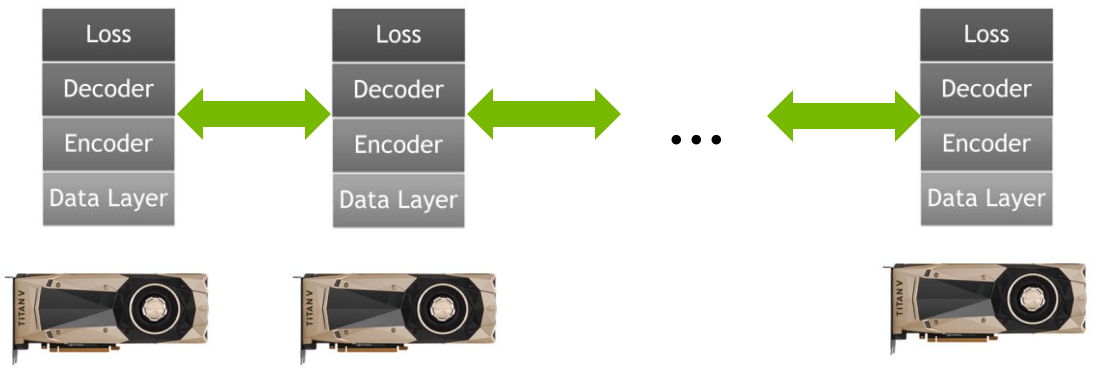
Mixed Precision Training

Faster and uses less GPU memory

- Speedups of 1.5x - 3x for same hyperparameters as *float32*
- You can use larger batch per GPU to get even bigger speedups

Distributed Training

Data Parallel Training
Synchronous updates



Two modes:

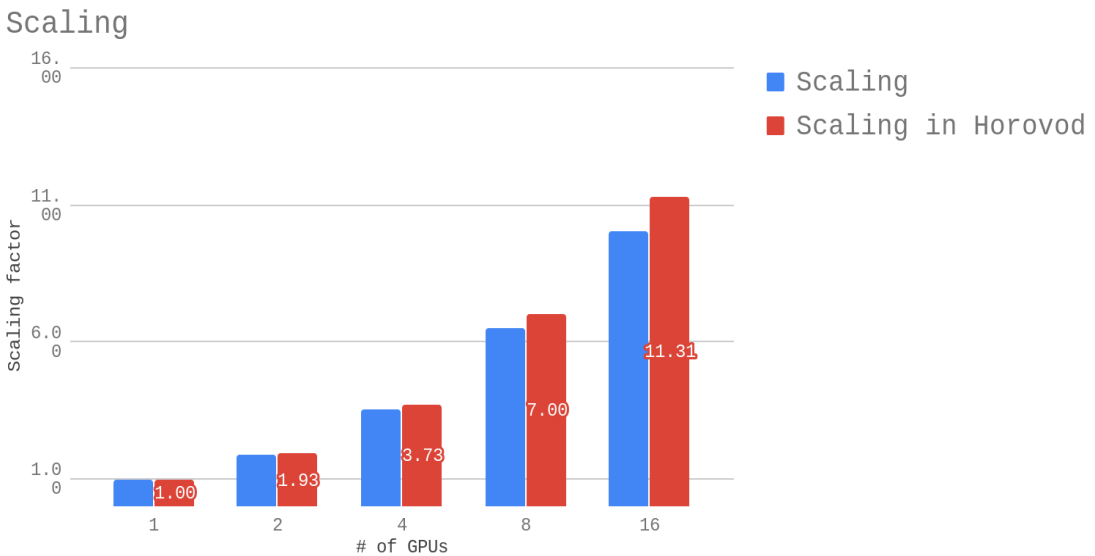
1. Tower-based approach
   - Pros: simple, less dependencies
   - Cons: single-node only, no NCCL

2. Horovod-based approach
   - Pros: multi-node support, NCCL support
   - Cons: more dependencies

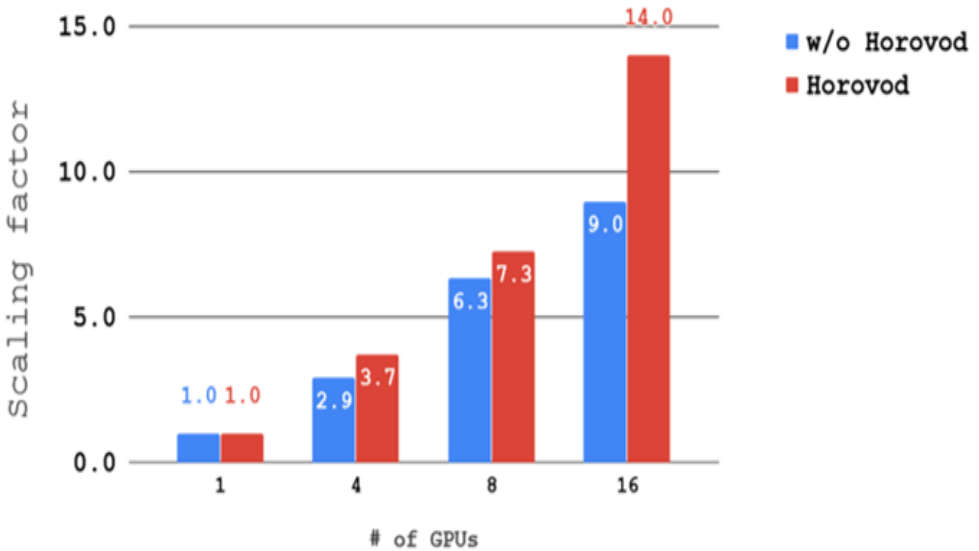
Tip: Use NVIDIA's TensorFlow container.
https://ngc.nvidia.com/catalog/containers/nvidia:tensorflow

Distributed Training

Transformer-big Scaling



ConvSeq2Seq Scaling



NVIDIA

OPENSEQ2SEQ: SPEECH TECHNOLOGY

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Chip Nguyen, Jonathan Cohen, Edward Lu, Ravi Gadde

OpenSeq2Seq: Speech Technologies

OpenSeq2Seq has the following speech technologies:

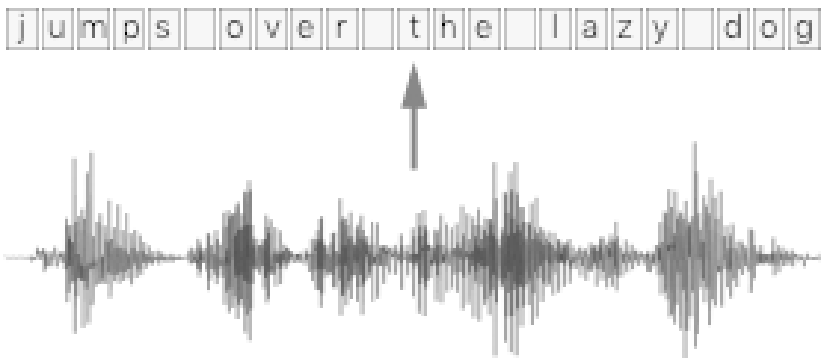
1. Large Vocabulary Continuous Speech Recognition:
    - DeepSpeech2, Wav2Letter+, Jasper
2. Speech Commands
3. Speech Generation (Tacotron2 + WaveNet/Griffin-Lim)
4. Language Models

Agenda

- Intro to end-to-end NN based ASR
  - CTC-based
  - Encoder-Decoder with Attention
- Jasper architecture
- Results

Traditional ASR pipeline

- All parts are trained separately
- Need pronunciation dictionary
- How to deal with out-of-vocabulary words
- Need explicit input-output time alignment for training: severe limitation since the alignment is very difficult

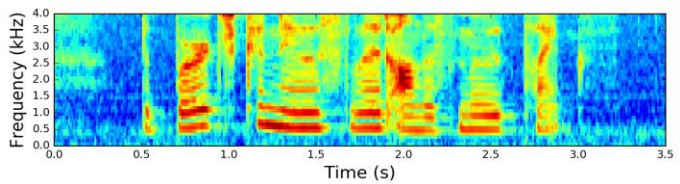


Language Model:
N-gram

Pronunciation model (Words):
n-state HMM

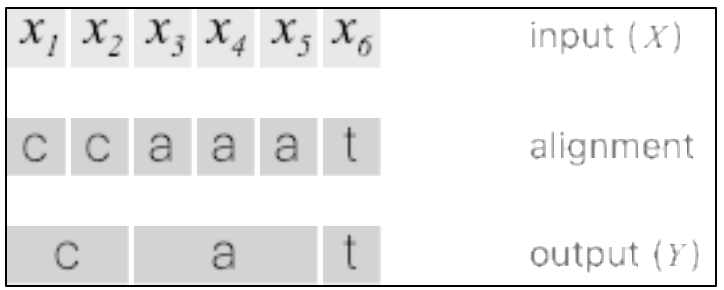
Acoustic model (Tri-phones):
3-state HMM

Acoustic model (Senones):
Gaussian Mixtures (GMM)



Hybrid NN-HMM Acoustic model

- DNN as GMM replacement to predict senones
- Different types of NN:
  - Time Delay NN, RNN, Conv NN
- Still need an alignment between the input and output sequences for training

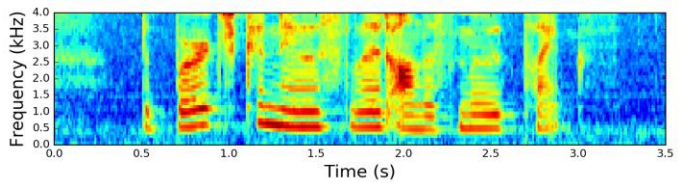


Language Model:
N-gram

Pronunciation model (Words):
n-state HMM

Acoustic model (Tri-phones):
3-state HMM

Acoustic DNN



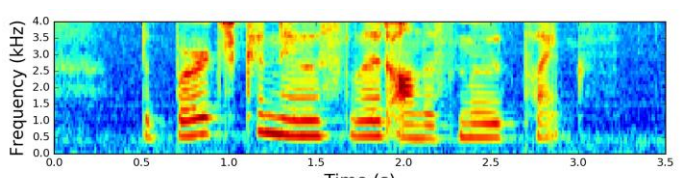
NN End-to-End: Encoder-Decoder

- No explicit input-output alignment
- RNN-based Encoder-Decoder
- RNN transducer (Graves 2012)
  - Encoder: Transcription net B-LSTM
  - Decoder: Prediction net (LSTM)

Language Model: N-gram

RNN-Decoder

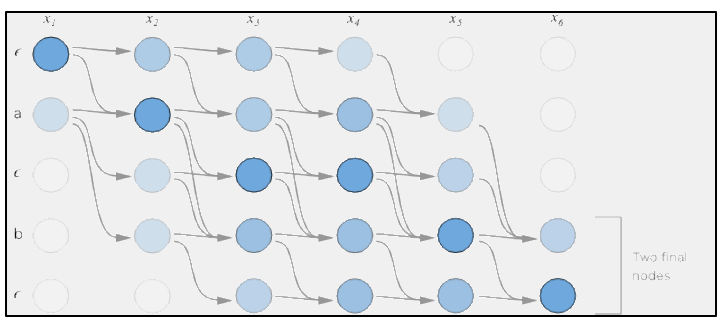
Encoder



Courtesy of Awni Hannun , 2017 https://distill.pub/2017/ctc/

NN End-to-End: Connectionist Temporal Classification

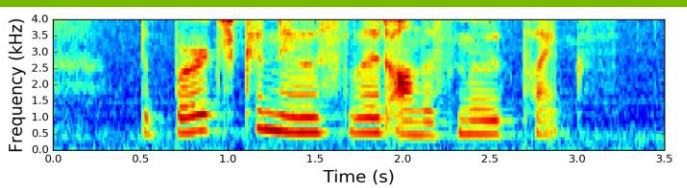
The CTC algorithm (Graves et al., 2006) doesn't require an alignment between the input and the output.
To get the probability of an output given an input, CTC takes sum over the probability of all possible alignments between the two. This 'integrating out' over possible alignments is what allows the network to be trained with unsegmented data



Language Model: N-gram

Connectionist Temporal Classification

Acoustic DNN:
Predicts output characters



Courtesy of Awni Hannun https://distill.pub/2017/ctc/

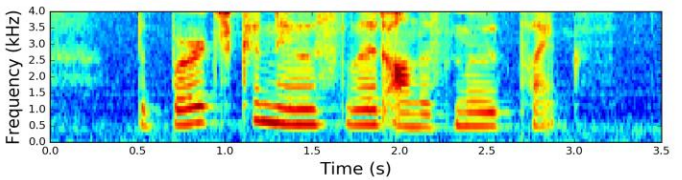
NN End-to-End models: NN Language Model

Replace N-gram with NN-based LM

NN Language Model

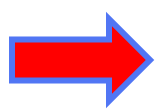
Connectionist Temporal Classification

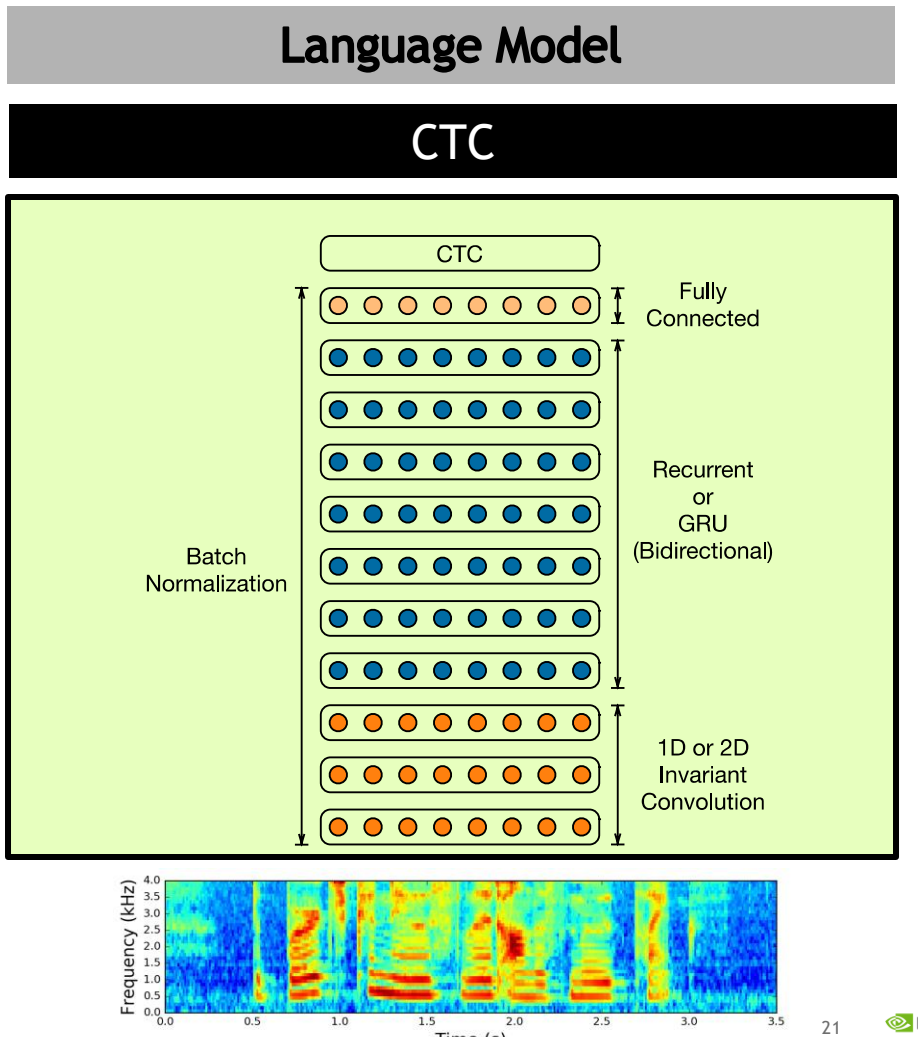
Acoustic DNN:
Predicts output characters



DeepSpeech2 = Conv + RNN + CTC

Deep Conv+RNN network
- 3 conv (TDNN)
- 6 bidirectional RNN
- 1 FC layer
- CTC loss

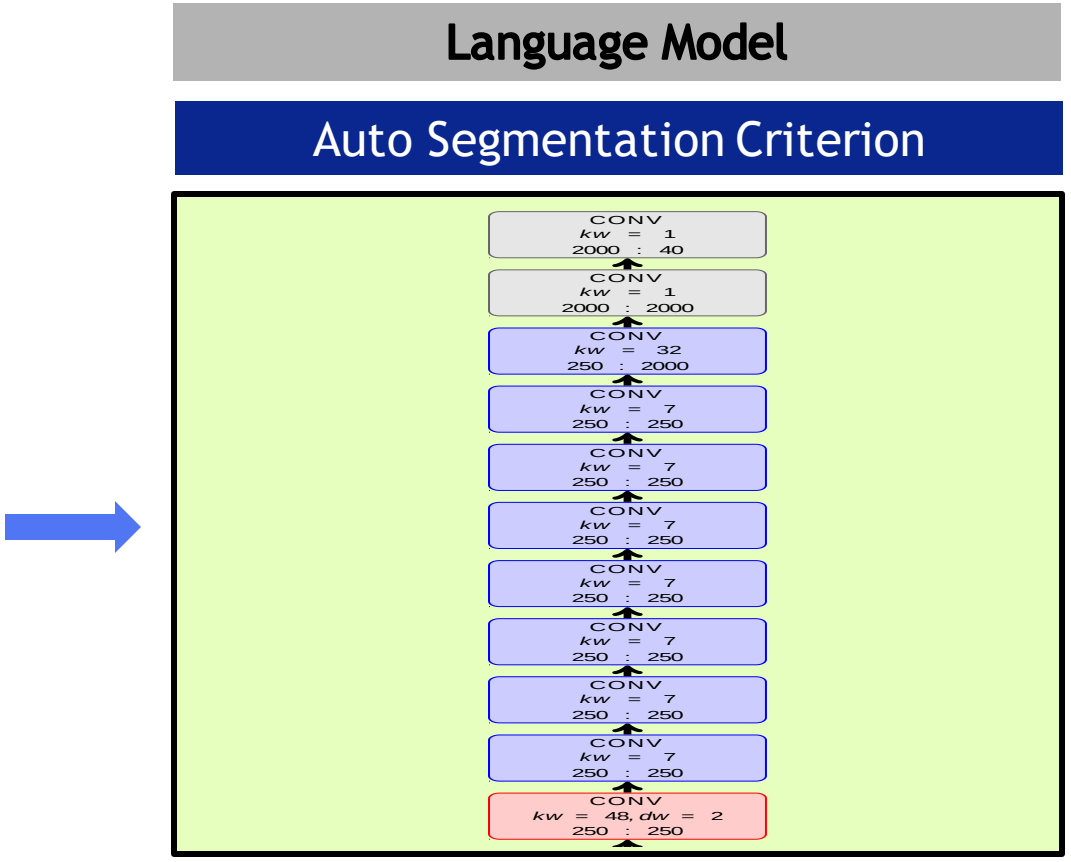


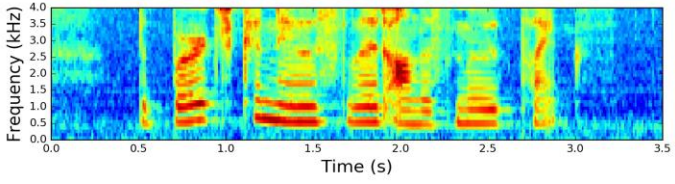


Amodei, et al “Deep speech 2 : End-to-end speech recognition in english and mandarin,” in ICML , 2016

Wav2Letter = Conv Model + CTC

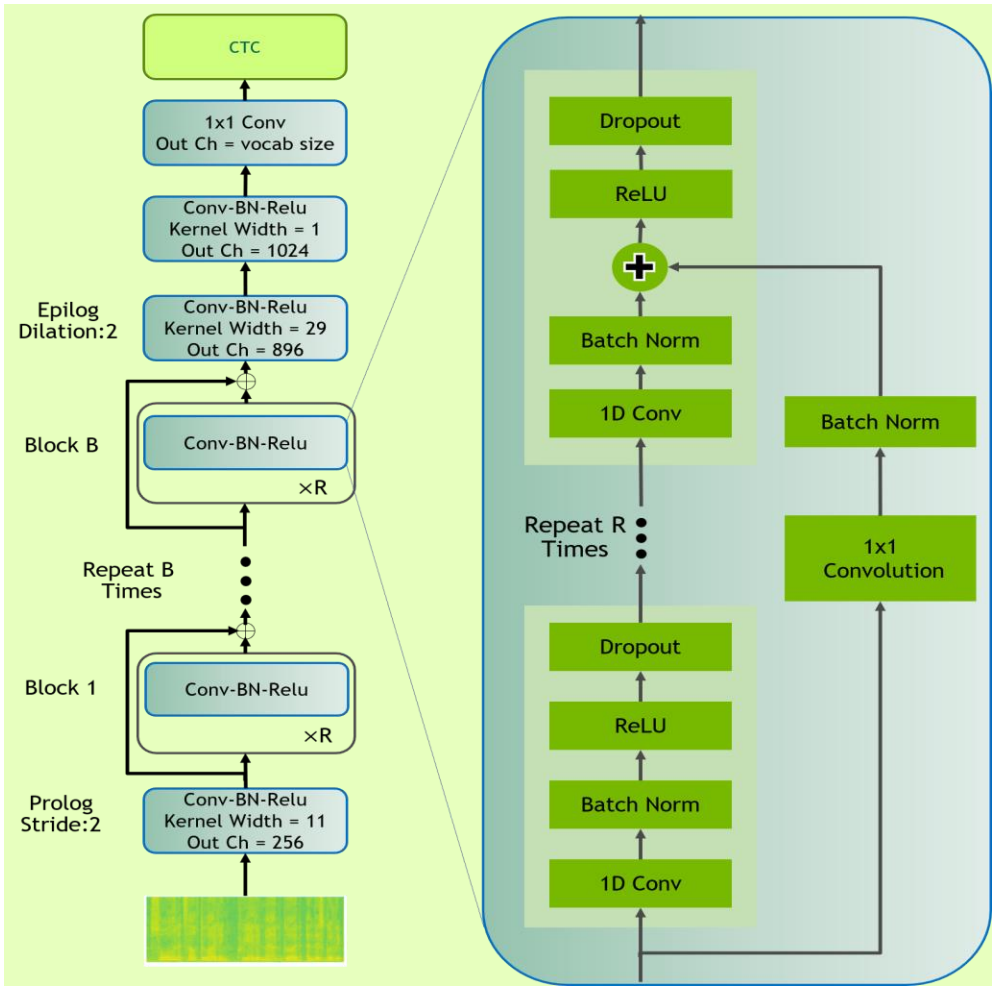
Deep ConvNet network
- 11 1D-conv layers
- Gated Linear Units (GLU)
- Weight Normalization
- Gradient clipping
- Auto Segmentation Criterion (ASG) = fast CTC

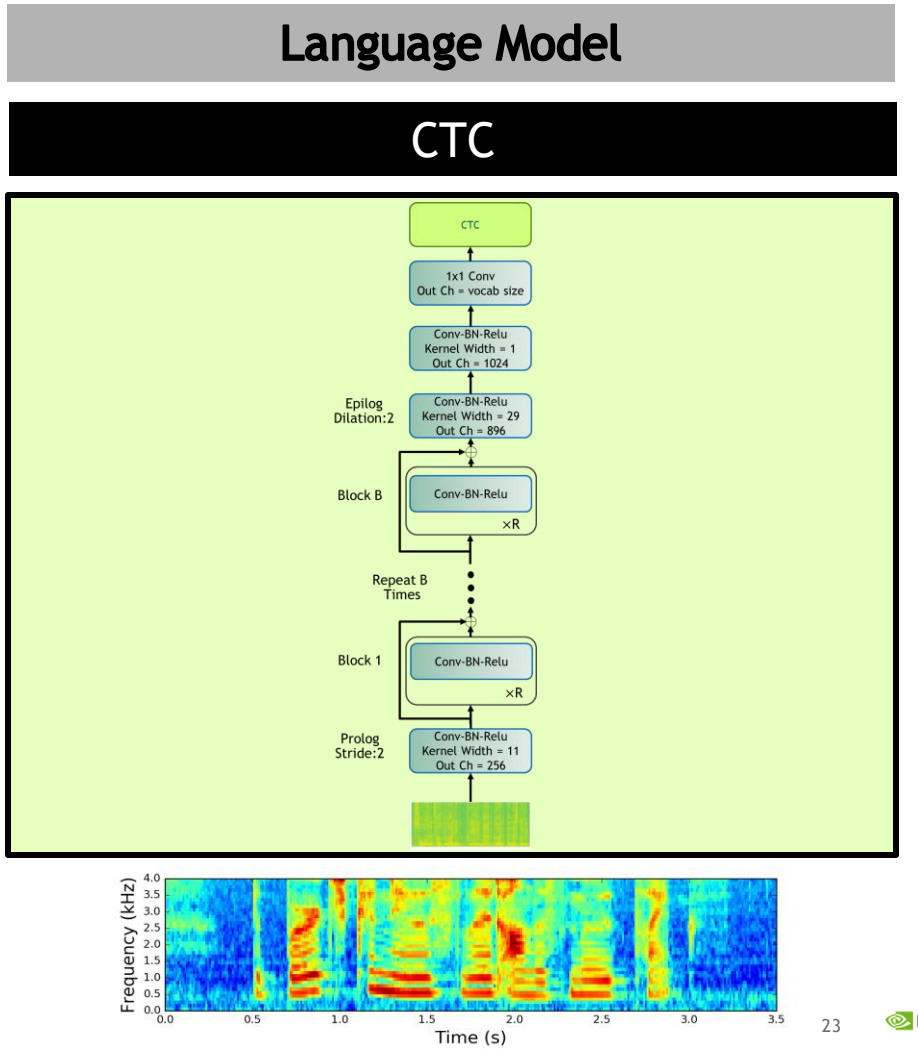




Collobert, et al . "Wav2letter: an end-to-end convnet-based speech recognition system." *arXiv preprint arXiv:1609.03193* (2016).

Jasper = Very Deep Conv NN + CTC





OpenSeq2Seq: Speech Technologies

Very Deep Conv-net:

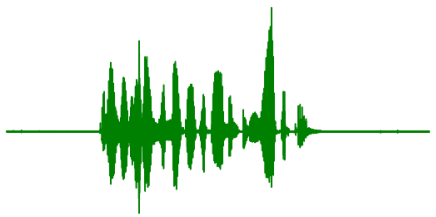
- 1D Conv-BatchNorm-ReLU-Dropout
- Residual Connection (per block)
- Jasper10x5
- 54 layers
- 330M weights

Trained with SGD with momentum:

- Mixed precision
- ~8 days on DGX1

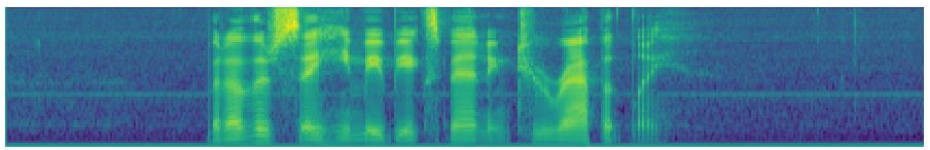
Block	Kernel	Channels	Dropout keep	Layers/ Block
Conv1	11 str 2	256	0.8	1
B1	11	256	0.8	5
B2	11	256	0.8	5
B3	13	384	0.8	5
B4	13	384	0.8	5
B5	17	512	0.8	5
B6	17	512	0.8	5
B7	21	640	0.7	5
B8	21	640	0.7	5
B9	25	768	0.7	5
B10	25	768	0.7	5
Conv2	29 dil 2	896	0.6	1
Conv3	1	1024	0.6	1
Conv4	1	vocabulary		1

Jasper: Speech preprocessing



Speech waveform

Signal Preprocessing

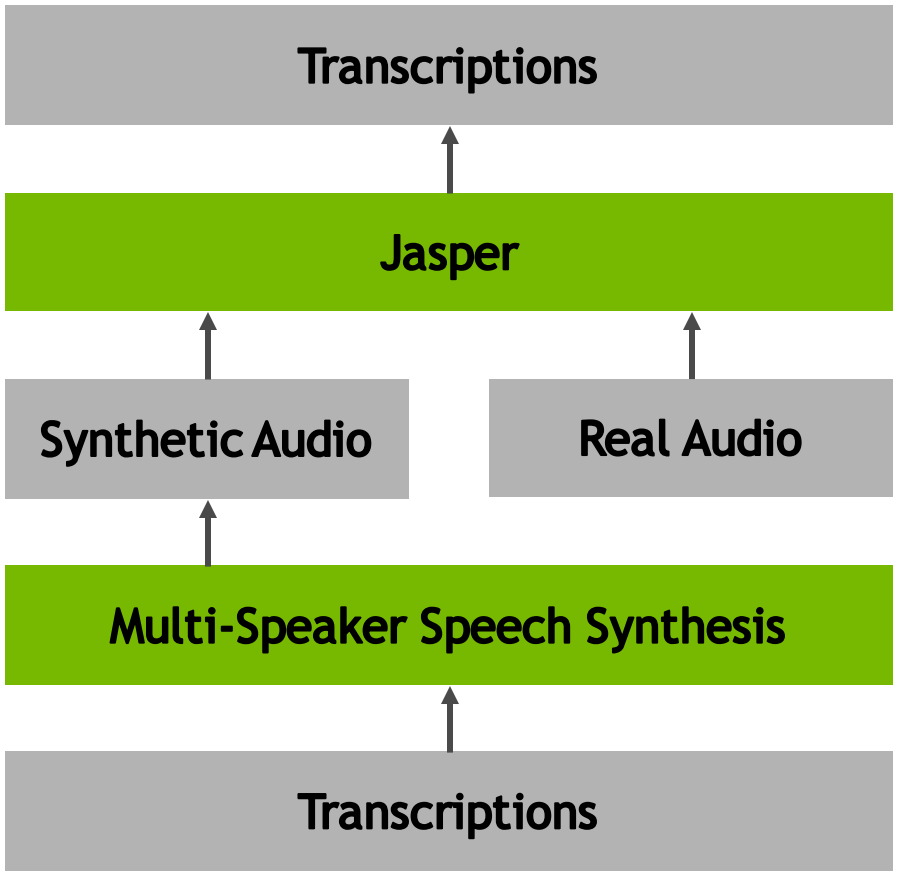


Log mel spectrogram

Speed perturbation	Noise Augmentation	Power Spectrogram	Mel Scale Aggregation	Log Normalization
faster / slower speech (resampling)	additive background noise	windowing, FFT	log scaling in frequency domain	log scaling for amplitude, feature normalization

Jasper: Data augmentation

- Augment with synthetic data using speech synthesis
- Train speech synthesis on multi-speaker data
- Generate audio using LibriSpeech transcriptions
- Train Jasper by mixing real audio and synthetic audio at a 50/50 ratio



Jasper: Correct ratio for Synthetic Data

- Tested difference mixtures of synthetic and natural data on Jasper 10x3 model
- 50/50 ratio achieves best results for LibriSpeech

Model, Natural/Synthetic Ratio (%)	WER (%), Test-Clean	WER (%), Test-Other
Jasper 10x3 (100/0)	5.10	16.21
Jasper 10x3 (66/33)	4.79	**15.37**
Jasper 10x3 (50/50)	**4.66**	15.47
Jasper 10x3 (33/66)	4.81	15.81
Jasper 10x3 (0/100)	49.80	81.78

27

Jasper: Language models

WER evaluations* on LibriSpeech

Language Model	WER (%), Test-Clean	WER (%), Test-Other
4-gram	3.67	11.21
5-gram	3.44	11.11
6-gram	3.45	11.13
Transformer-XL	3.11	10.62

- Jasper 10x5 dense res model, beam width = 128, alpha=2.2, beta=0.0

Jasper: Results

LibriSpeech, WER %, Beam Search with LM		
	test-clean	test-other
Jasper-10x5 DR Syn	3.11	10.62
Published results		
DeepSpeech2	5.33	13.25
Wav2Letter	4.80	14.50
Wav2Letter++	3.44	11.24
CAPIO**	3.19	7.64

**CAPIO Augmented with additional training data

29

OPENSEQ2SEQ: SPEECH COMMANDS

Edward Lu

OpenSeq2Seq: Speech Commands

Dataset: Google Speech Commands (2018)

- V1: ~65,000 samples over 30 classes
- V2: ~110,000 samples over 35 classes
- Each sample is ~1 second long, 16kHz recording in a different voice includes commands (on/off, stop/go, directions), non-commands, background noise

Previous SoA:

- Kaggle Contest: 91% accuracy
- Mixup paper: 96% accuracy (VGG-11)

Speech Commands: Accuracy

Dataset	Validation	Test	Training Time
ResNet-50			
V1-12	96.6%	96.6%	1h56m
V1	**97.5%**	**97.3%**	3h16m
V2	95.7%	95.9%	3h49m
Jasper-10x3			
V1-12	97.1%	96.2%	3h13m
V1	**97.5%**	**97.3%**	8h10m
V2	95.5%	95.1%	9h32m
VGG-11 with Mixup			
V1	96.1%	96.6%	

Code, Docs and Pre-trained models:
https://github.com/NVIDIA/OpenSeq2Seq