TEXT-TO-SPEECH SYNTHESIS USING TACOTRON 2 AND WAVEGLOW WITH TENSOR CORES

Rafael Valle, Ryan Prenger and Yang Zhang
1. Text to Speech Synthesis
2. Tacotron 2
3. WaveGlow
4. TTS and TensorCores
TEXT TO SPEECH SYNTHESIS (TTS)

Human to ? Interaction

Apple Siri
Microsoft Cortana
Amazon Alexa / Polly
Nuance Vocalizer
Google TTS

Global TTS Market Value

USD Billions

2016

2022

1 https://www.marketsandmarkets.com/PressReleases/text-to-speech.asp
APPLICATIONS OF TTS

- Health Care
- Smart Home Devices
- Audio Books
- Vocaloids
- Self-Driving Cars
- Video Games
TEXT TO SPEECH SYNTHESIS

Text Input

Forty percent of a

Speech Output

for - ty per - cent of a
SPEECH SYNTHESIS: THE VODER 1939
PARAMETRIC SPEECH SYNTHESIS

Pneumatic speech synthesizer developed by von Kempelen in 1791.

Voder speech synthesizer developed by Homer Dudley in 1939.
CONCATENATIVE TTS SYNTHESIS

First practical application in 1936: British Phone company’s Talking Clock

Database

for -

ty

per -
c -

ent

of

a
CONCATENATIVE TTS SYNTHESIS

- Requires collecting speech units
- Requires designing cost heuristics
- Requires acoustic processing

https://wezs.com/~danguy/monguy/TTS.html
PARAMETRIC (DEEP LEARNING) TTS SYNTHESIS

Text Input
Forty percent of a

Deep Learning

Audio Output

for - ty per - cent of a
DEEP LEARNING TTS SYNTHESIS

Text Input
Forty percent of a

Linguistic or Acoustic features

Audio Output

TACOTRON

1°

X

WAVENET

2°
OUTLINE

1. Text to Speech Synthesis
2. Tacotron 2
3. WaveGlow
4. TTS and TensorCores
TEXT TO (MEL) SPECTROGRAM WITH TACOTRON

Tacotron
CBHG:
Convolution Bank (k=[1, 2, 4, 8…])
Convolution stack (ngram like)
Highway
bi-directional GRU

Tacotron 2
Location sensitive attention, i.e. attend to:
Memory (encoder output)
Query (decoder output)
Location (attention weights)
Cumulative attention weights (+= )
Tacotron 2 (without wavenet)

PyTorch implementation of Natural TTS Synthesis By Conditioning Wavenet On Mel Spectrogram Predictions.

This implementation includes distributed and fp16 support and uses the LJSpeech dataset.

Distributed and FP16 support relies on work by Christian Sarofeen and NVIDIA’s Apex Library.

Visit our website for audio samples using our published Tacotron 2 and WaveGlow models.

Pre-requisites

1. NVIDIA GPU + CUDA cuDNN

Setup

1. Download and extract the LJ Speech dataset
2. Clone this repo: git clone https://github.com/NVIDIA/tacotron2.git
3. CD into this repo: cd tacotron2
4. Initialize submodule: git submodule init; git submodule update
5. Update .wav paths: sed -i 's/DUMMY,ljs_dataset_folder/wavs,g' filelists/*.txt
   ○ Alternatively, set load_mel_from_disk=True in hparams.py and update mel-spectrogram paths
6. Install PyTorch 1.0
7. Install python requirements or build docker image
   ○ Install python requirements: pip install -r requirements.txt

Training

1. python train.py --output_directory=outdir --log_directory=logdir
2. (OPTIONAL) tensorboard --logdir=stdout/logdir

Training using a pre-trained model

Implementations
https://github.com/NVIDIA/tacotron2/
https://github.com/NVIDIA/OpenSeq2Seq/

Deep Learning Framework and Libraries
– PyTorch
– TensorFlow
– NVIDIA’s Automatic Mixed Precision

Training Setup
– NVIDIA’s Tesla V100
– Good results in less than a day starting fresh
– Good results in a few hours warm-starting
LJS (Linda Johnson: single native speakers, ~24 hours)

- 7 non-fiction books
- “All of my recordings were done from the sofa in my family room!”
- “All of my recordings were done on a MacBook Pro.”
- https://keithito.com/LJ-Speech-Dataset/
- https://librivox.org/reader/11049

Sometimes raw text, other times ARPAbet

<table>
<thead>
<tr>
<th>ARPABET</th>
<th>IPA</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>balm, bot</td>
</tr>
<tr>
<td>@</td>
<td>æ</td>
<td>bat</td>
</tr>
<tr>
<td>A</td>
<td>æH</td>
<td>butt</td>
</tr>
<tr>
<td>c</td>
<td>æO</td>
<td>bought</td>
</tr>
</tbody>
</table>
MEL TO AUDIO WITH WAVENET

\[ p(x | h) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}, h) \]

Sampling Rates
- 44100 Hz
- 22050 Hz
- 16000 Hz

https://deepmind.com/blog/wavenet-generative-model-raw-audio/
WAVENET IMPLEMENTATION DETAILS

Naïve PyTorch -> 20 samples per second

Inference PyTorch on Volta -> 200 samples per second

nv-wavenet -> 20000 samples per second
# MEAN OPINION SCORES: TACOTRON AND WAVENET

<table>
<thead>
<tr>
<th>System</th>
<th>MOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametric</td>
<td>3.492 ± 0.096</td>
</tr>
<tr>
<td>Tacotron (Griffin-Lim)</td>
<td>4.001 ± 0.087</td>
</tr>
<tr>
<td>Concatenative</td>
<td>4.166 ± 0.091</td>
</tr>
<tr>
<td>WaveNet (Linguistic)</td>
<td>4.341 ± 0.051</td>
</tr>
<tr>
<td>Ground truth</td>
<td>4.582 ± 0.053</td>
</tr>
<tr>
<td>Tacotron 2 (this paper)</td>
<td><strong>4.526 ± 0.066</strong></td>
</tr>
</tbody>
</table>

https://arxiv.org/abs/1712.05884
OUTLINE

1. Text to Speech Synthesis
2. Tacotron 2
3. WaveGlow
4. TTS and TensorCores
WAVENET IS THE BOTTLENECK

TacoTron2

1. Block diagram of the Tacotron 2 system architecture.

DeepVoice 3


WAVENET IS THE BOTTLENECK


AUTO-REGRESSION IS INHERENTLY SERIAL

\[ P(x_0, x_1, x_2, \ldots) = P(x_0)P(x_1|x_0)P(x_2|x_1, x_0)\ldots \]

AUTO-REGRESSION IS INHERENTLY SERIAL

\[ P(x_0, x_1, x_2, \ldots) = P(x_0)P(x_1|x_0)P(x_2|x_1, x_0)\ldots \]


https://github.com/NVIDIA/nv-wavenet
TRANSFORMING WHITENOISE TO AUDIO IS PARALLEL

Gaussian Noise

Mel-Spectrogram

...
AUTO-ENCODER (APPROXIMATING LIKELIHOOD)

Gaussian Noise → Loss 1 → Mel-Spectrogram

Loss 2
INVERTIBLE NETWORK (EXACT LIKELIHOOD)

Gaussian Noise → Loss 1

Mel-Spectrogram →

Loss 1 →

Loss 1 →

Loss 1 →
HOW TO MAKE A NETWORK INVERTIBLE
HOW TO MAKE A NETWORK INVERTIBLE
HOW TO MAKE A NETWORK INVERTIBLE
HOW TO MAKE A NETWORK INVERTIBLE
HOW TO MAKE A NETWORK INVERTIBLE
HOW TO MAKE A NETWORK INVERTIBLE

Coupling network

(s, b) / s
(s, b) / s
(s, b) / s
(s, b) / s
(s, b) / s
(s, b) / s

\[
\log p_\theta(x) = -\frac{x^T z(x)}{2\sigma^2} + \text{#coupling} \sum_{j=0}^{\text{#conv}} \log s_j(x, \text{mel-spectrogram}) + \sum_{k=0}^{\text{#conv}} \log \det |W_k|
\]

https://github.com/NVIDIA/waveglow

Fig. 1: WaveGlow network
DECREASING TEMPERATURE CAN HELP

Gaussian Noise

Mel-Spectrogram

\[
\sigma \sim 0.8
\]
## PARALLEL SOLUTION WORKS

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Opinion Score (MOS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griffin-Lim</td>
<td>3.823 ± 0.1349</td>
</tr>
<tr>
<td>WaveNet</td>
<td>3.885 ± 0.1238</td>
</tr>
<tr>
<td>WaveGlow</td>
<td>3.961 ± 0.1343</td>
</tr>
<tr>
<td>Ground Truth</td>
<td>4.274 ± 0.1340</td>
</tr>
</tbody>
</table>

NV-WaveNet: 24-48khz (1.2x - 2.4x realtime)

WaveGlow (published): 520 khz (24.5x realtime)
## PARALLEL SOLUTION WORKS

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Opinion Score (MOS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griffin-Lim</td>
<td>3.823 ± 0.1349</td>
</tr>
<tr>
<td>WaveNet</td>
<td>3.885 ± 0.1238</td>
</tr>
<tr>
<td>WaveGlow (published)</td>
<td>3.961 ± 0.1343</td>
</tr>
<tr>
<td>Ground Truth</td>
<td>4.274 ± 0.1340</td>
</tr>
</tbody>
</table>

NV-WaveNet: 24-48khz (1.2x - 2.4x realtime)

WaveGlow (published): 520 khz (24.5x realtime)
WaveGlow (internal smaller): 1,500 khz (70x realtime)
RELATED WORK

Parallel WaveNet/ClariNet
- Very similar network/inference
- Very different training procedure

WaveRNN
- More like optimized auto-regressive
- Can get some parallelism with subscale trick
1. Text to Speech Synthesis
2. Tacotron 2
3. WaveGlow
4. TTS and Tensor Cores
INFERECE SPEED UP

with Tensor Cores - Automatic Mixed Precision

On DGX-1
1 Tesla V100 GPU
Batch size: 1

samples/s in MHz

w/o Tensor Cores  w/ Tensor Cores

1.8x
INFERENCe SPEED UP
with Tensor Cores - Automatic Mixed Precision

On DGX-1
1 Tesla V100 GPU
Batch size: 1
TENSOR CORES SPEED UP MATRIX MULTIPLICATIONS

\[
D = \begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\
A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3}
\end{pmatrix}
\times
\begin{pmatrix}
B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\
B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\
B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\
B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3}
\end{pmatrix}
+ \begin{pmatrix}
C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\
C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\
C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\
C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3}
\end{pmatrix}
\]
w/o Tensor Cores  w/ Tensor Cores

Inference time  29ms  15ms

2X FASTER INference WITH TENSOR CORES
TRAINING SPEED UP
with Tensor Cores - Automatic Mixed Precision

On DGX-1
1 Tesla V100 GPU
over 1000 Epochs
Tensor Cores achieve similar training loss.
USING TENSOR CORES WITH AMP

- Automatic Mixed Precision library that enables Tensor Cores transparently
  - manages type conversions and master weights
  - automatic loss scaling to prevent gradient underflow
- Different levels of optimization
  - white/black list allow user to enforce precision
- Easy code adjustment
INFERENC WITH AMP IS EASY

Code Example

FP32

```python
from glow import Wave Glow
model = Wave Glow(**json.loads(config_data)['waveglow_conf g']).cuda()

input_data = torch.rand((batch_size, 80, n_frames)).cuda()
with torch.no_grad():
    result = model.infer(input_data)
```
INFEERENCE WITH AMP IS EASY

Code Example

FP32

```
... from glow import WaveGlow
model = WaveGlow(**json.loads(config_data)[‘waveglow_config’]).cuda()
input_data = torch.rand((batch_size, 80, n_frames)).cuda()
with torch.no_grad():
    result = model.infer(input_data)
...`

1x

Tensor Cores with AMP

```
... from glow import WaveGlow
model = WaveGlow(**json.loads(config_data)[‘waveglow_config’]).cuda()
# use AMP to adjust model and select optimization level
from apex import amp
model, __ = amp.initialize(model, [], opt_level=’O1’)
input_data = torch.rand((batch_size, 80, n_frames)).cuda()
with torch.no_grad():
    result = model.infer(input_data)
...`

1.8x
TRAINING WITH AMP IS EASY

Code Example

```python
from torch.utils.data import DataLoader
from glow import WaveGlow, WaveGlowLoss
from mel2samp import Mel2Samp

def train(num_gpus, rank, group_name, output_directory, epochs, learning_rate,
          sigma, iters_per_checkpoint, batch_size, seed, checkpoint_path):
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    
    ....

    for epoch in range(epoch_offset, epochs):
        print("Epoch: {}".format(epoch))
        for i, batch in enumerate(train_loader):
            model.zero_grad()
            mel, audio = batch
            mel = torch.autograd.Variable(mel.cuda())
            audio = torch.autograd.Variable(audio.cuda())
            outputs = model((mel, audio))

            loss = criterion(outputs)
            if num_gpus > 1:
                reduced_loss = reduce_tensor(loss.data, num_gpus).item()
            else:
                reduced_loss = loss.item()
            loss.backward()
            optimizer.step()
            
            ....
```
TRAINING WITH AMP IS EASY

Code Example

```python
from torch.utils.data import DataLoader
from glow import WaveGlow, WaveGlowLoss
from mel2samp import Mel2Samp

from apex import amp

def train(num_gpus, rank, group_name, output_directory, epochs, learning_rate,
          sigma, iter_per_checkpoint, batch_size, seed, checkpoint_path):
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)

    amp_handle = amp.init()

    for epoch in range(epoch_offset, epochs):
        print("Epoch: ", epoch)
        for i, batch in enumerate(train_loader):
            model.zero_grad()

            mel, audio = batch
            mel = torch.autograd.Variable(mel.cuda())
            audio = torch.autograd.Variable(audio.cuda())
            outputs = model((mel, audio))

            loss = criterion(outputs)
            if num_gpus > 1:
                reduced_loss = reduce_tensor(loss.data, num_gpus).item()
            else:
                reduced_loss = loss.item()

            with amp_handle.scale_loss(loss, optimizer) as scaled_loss:
                scaled_loss.backward()
            optimizer.step()
```

Tensor Cores with AMP

1.9x speed up
CONCLUSION

- Tensor Cores achieve close to 2x faster inference and training on Waveglow
- AMP enables Tensor Cores transparently for training and inference
- Code available on NGC and github
  - https://ngc.nvidia.com/catalog/model-scripts/
  - https://github.com/NVIDIA/tacotron2
  - https://github.com/NVIDIA/waveglow
  - https://github.com/NVIDIA/apex/tree/master/apex/amp