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

Rafael Valle, Ryan Prenger and Yang Zhang

OUTLINE

1. Text to Speech Synthesis

2. Tacotron 2

3. WaveGlow

4. TTS and TensorCores

TEXT TO SPEECH SYNTHESIS (TTS)

Human to ? Interaction

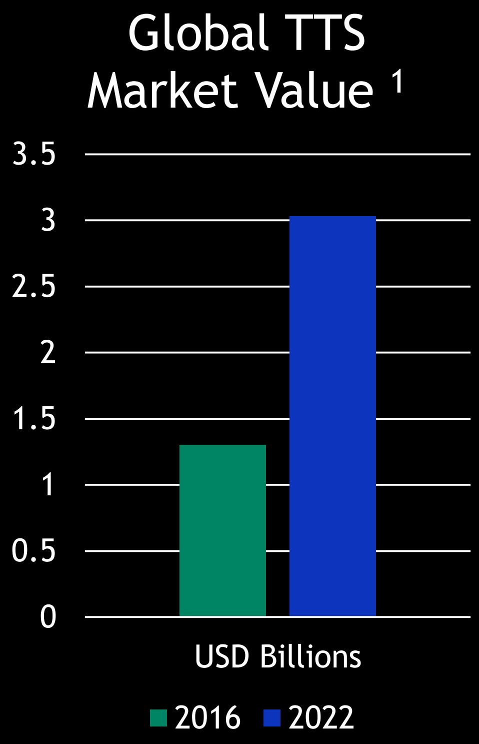
Apple Siri

Microsoft Cortana

Amazon Alexa / Polly

Nuance Vocalizer

Google TTS



[1] https://www.marketsandmarkets.com/PressReleases/text-to-speech.asp

APPLICATIONS OF TTS

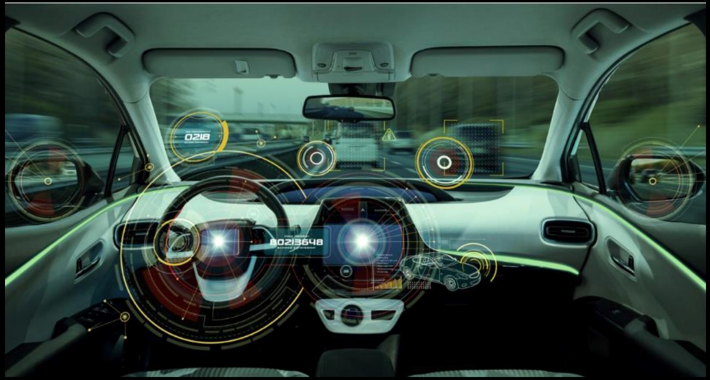
Health Care

Smart Home Devices

audible

Audio Books

Vocaloids

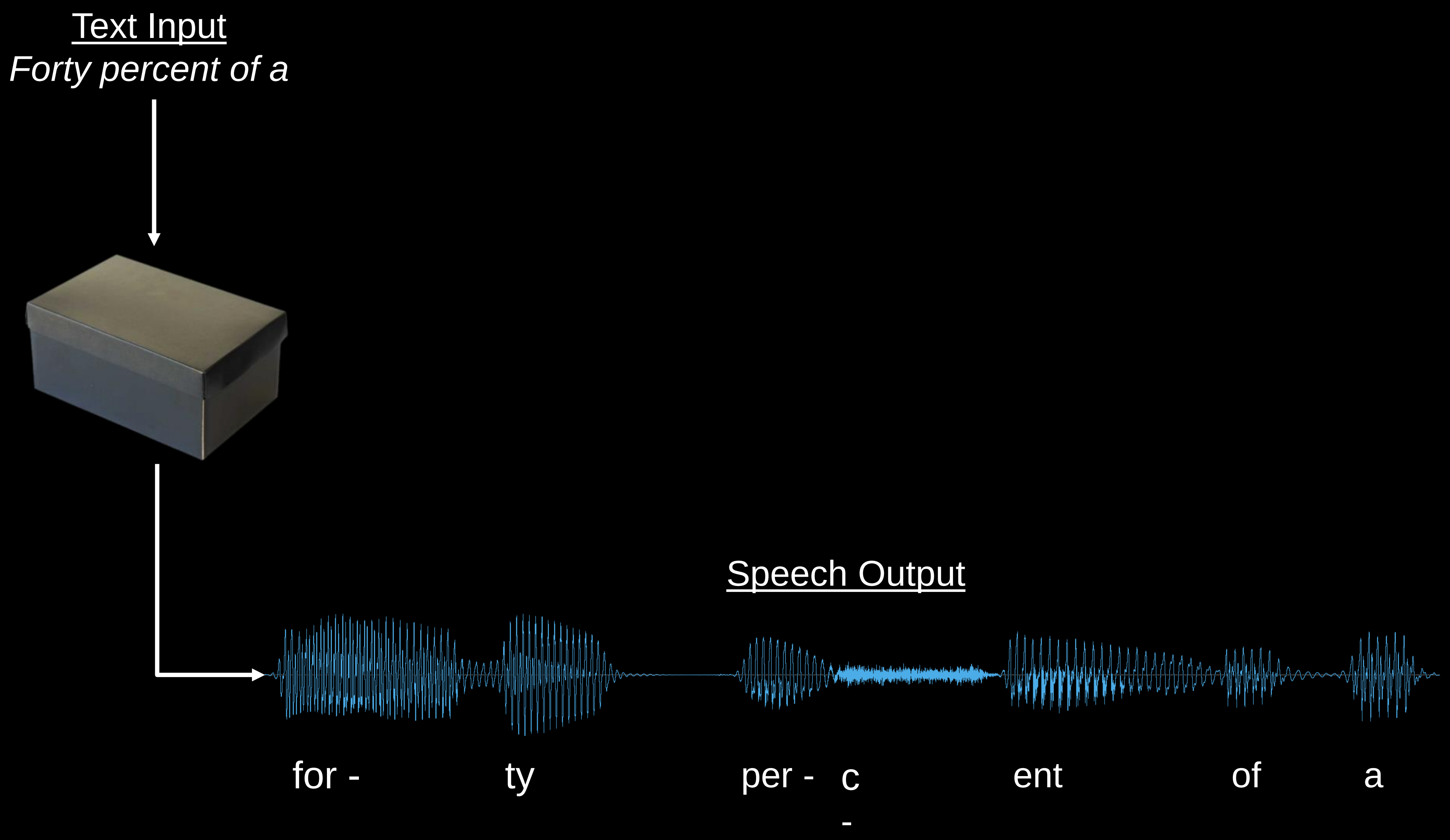


Self-Driving Cars

Video Games

NVIDIA

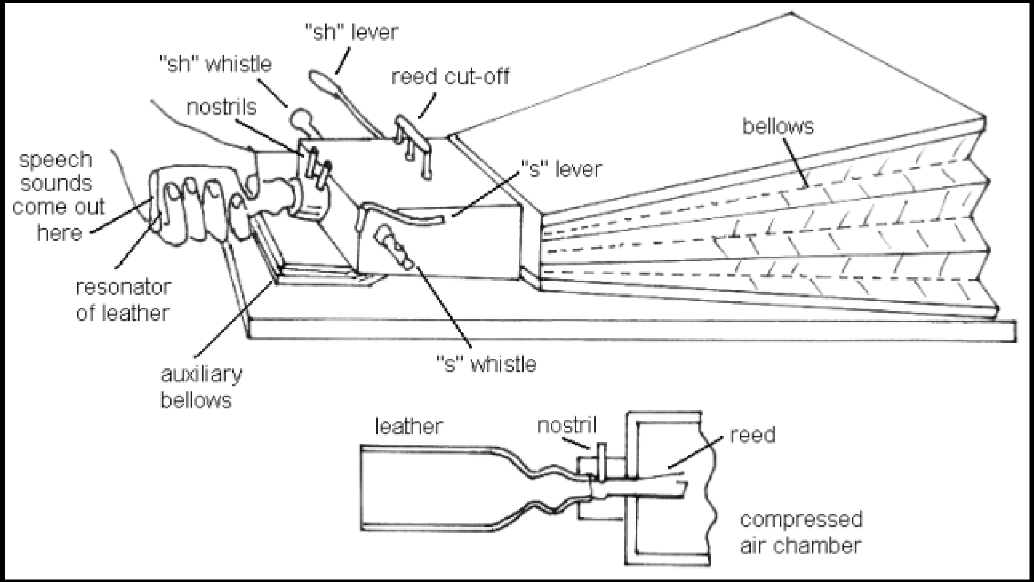
TEXT TO SPEECH SYNTHESIS



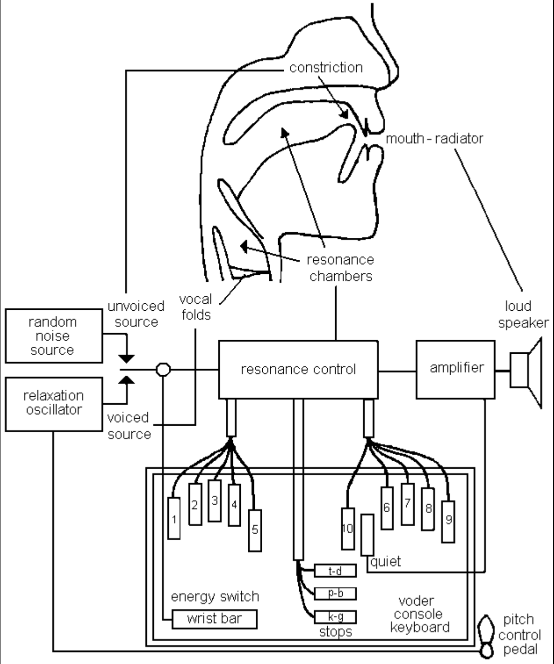
SPEECH SYNTHESIS: THE VODER 1939

NVIDIA

PARAMETRIC SPEECH SYNTHESIS

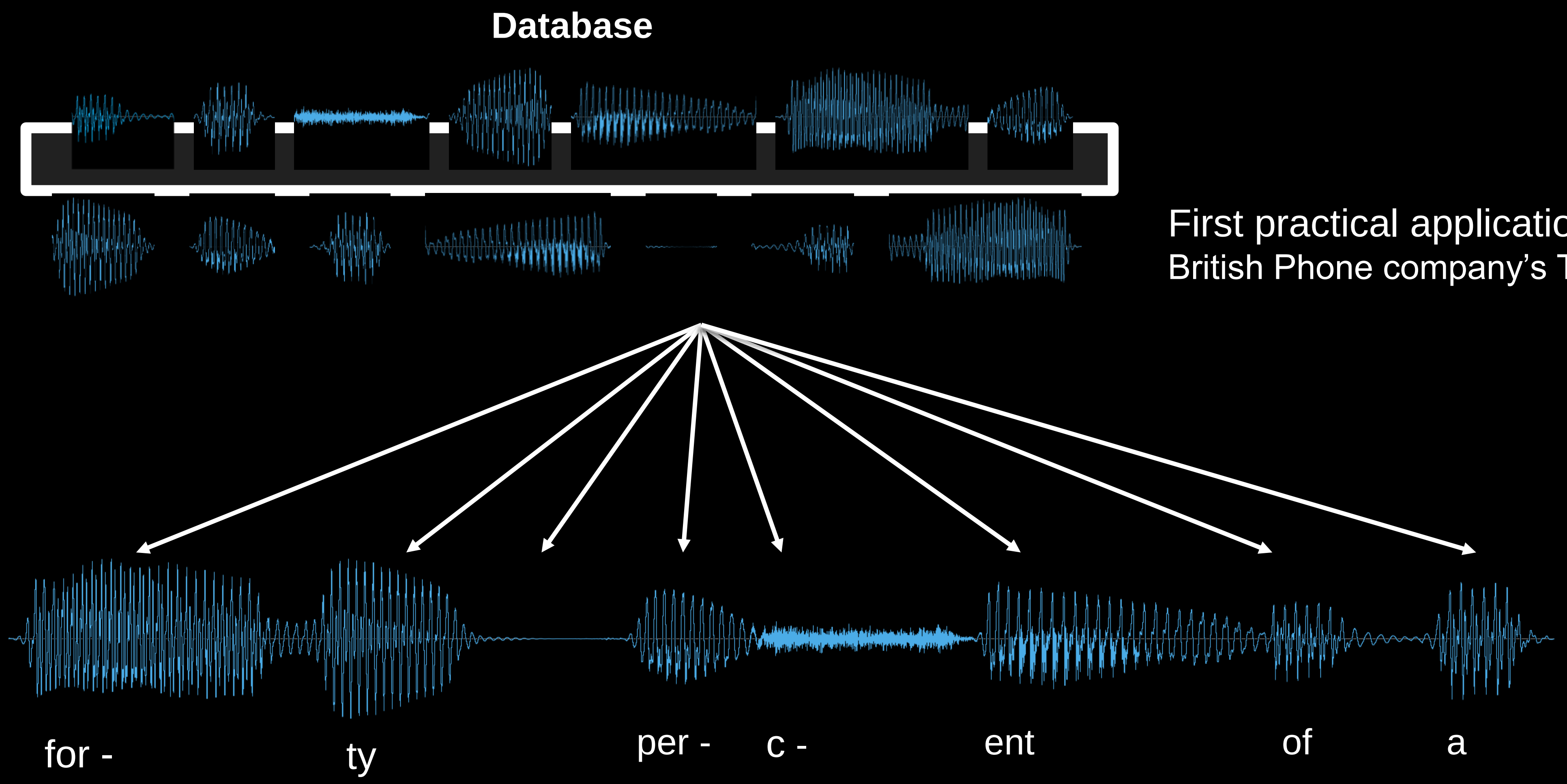


Pneumatic speech synthesizer developed by von Kempelen in 1791.



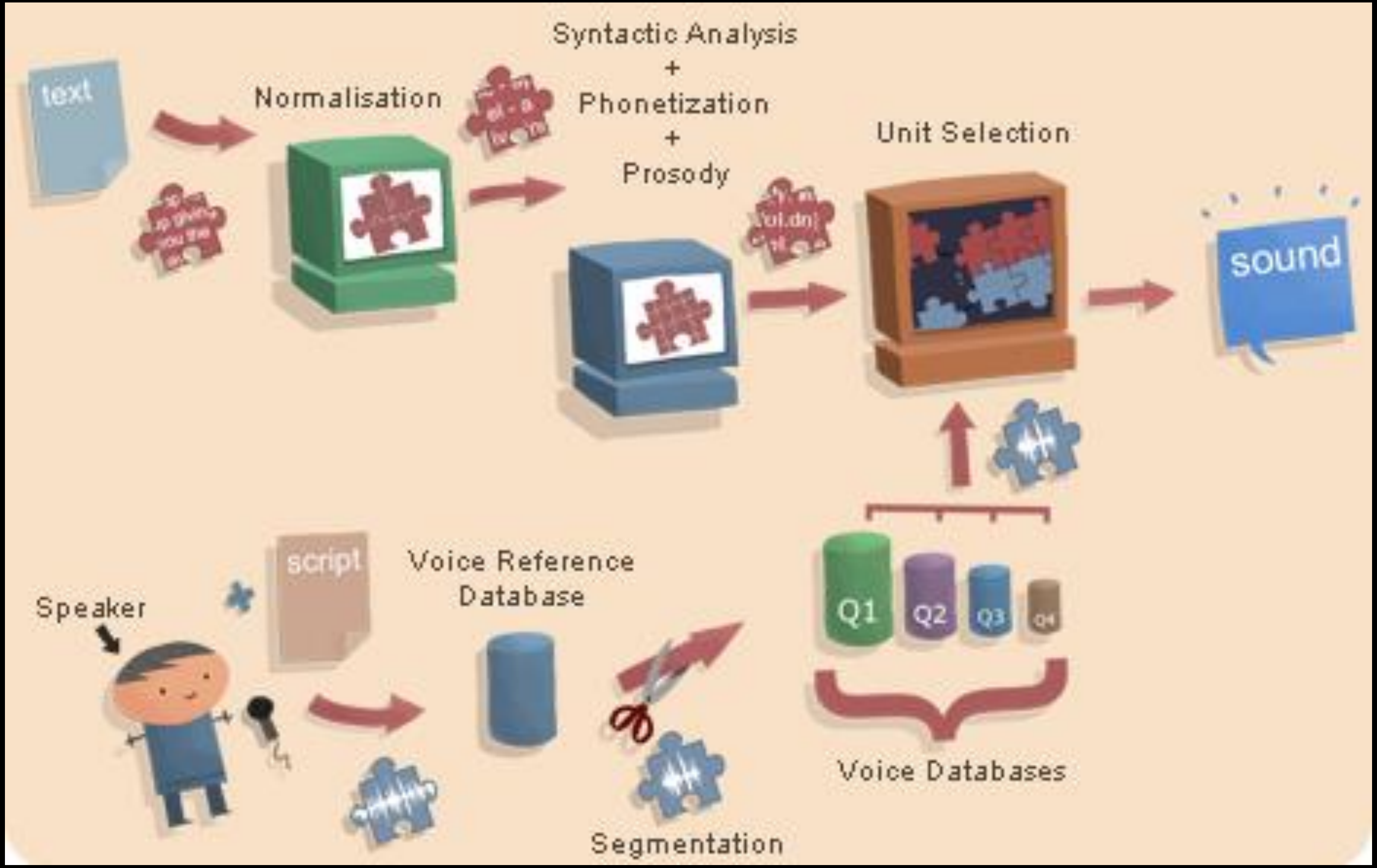
Voder speech synthesizer developed by Homer Dudley in 1939.

CONCATENATIVE TTS SYNTHESIS



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

CONCATENATIVE TTS SYNTHESIS



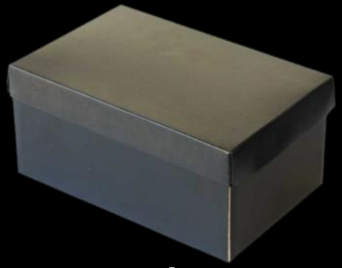
- Requires collecting speech units
- Requires designing cost heuristics
- Requires acoustic processing

https://wezs.com/~danguy/monguy/TTS.html

PARAMETRIC (DEEP LEARNING) TTS SYNTHESIS

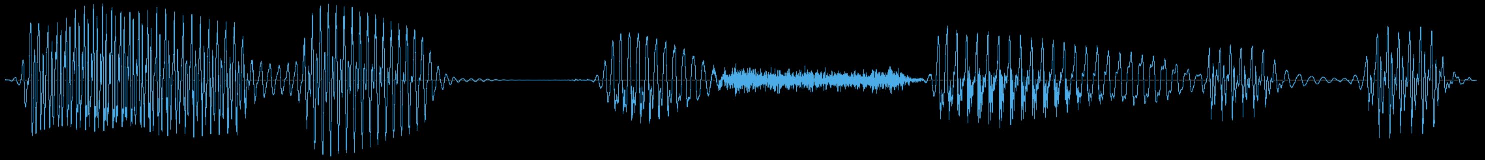
Text Input

Forty percent of a



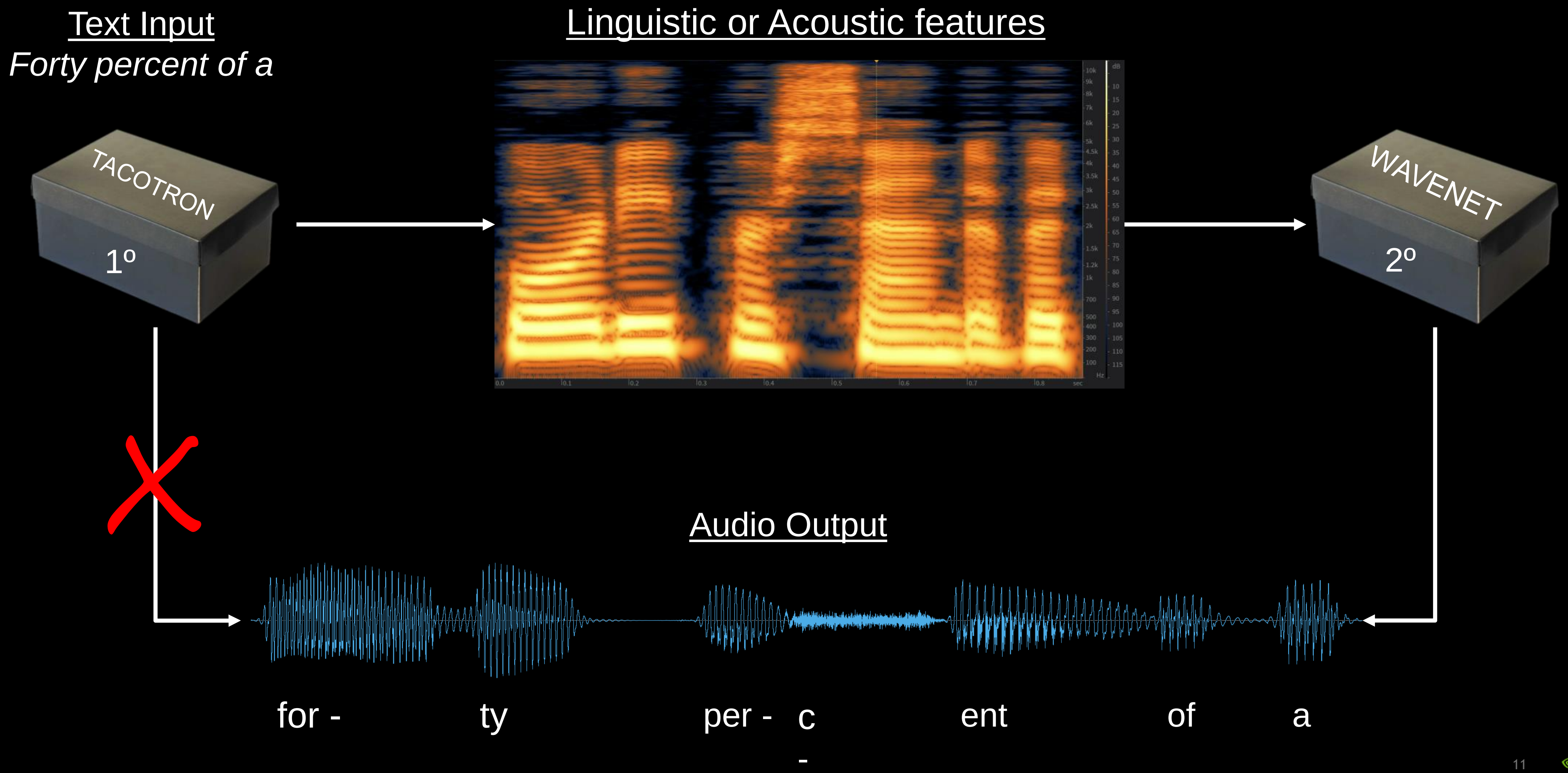
Deep Learning

Audio Output



for - ty per - c - ent of a

DEEP LEARNING TTS SYNTHESIS



OUTLINE

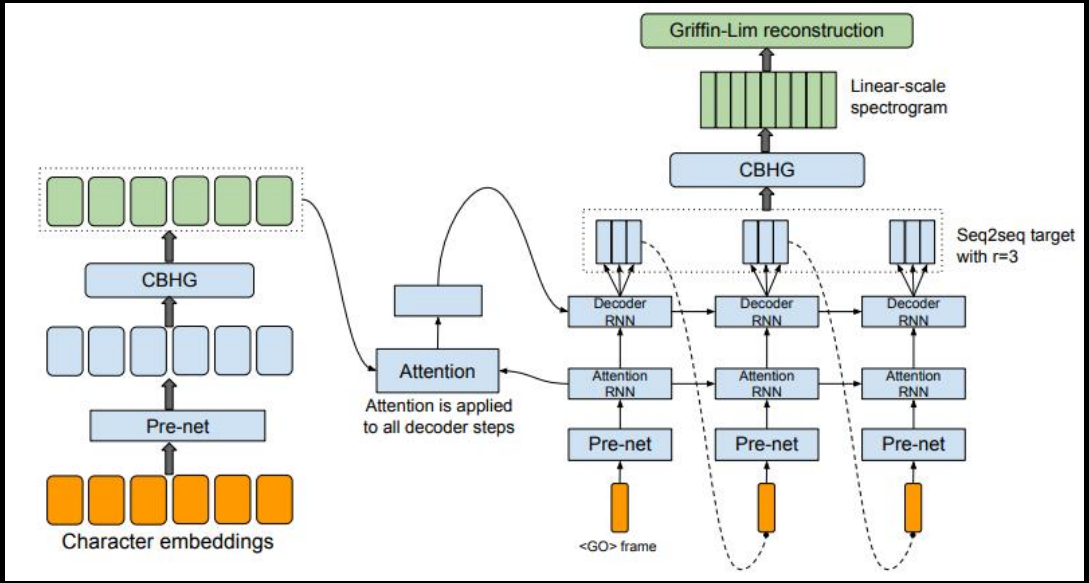
1. Text to Speech Synthesis

2. Tacotron 2

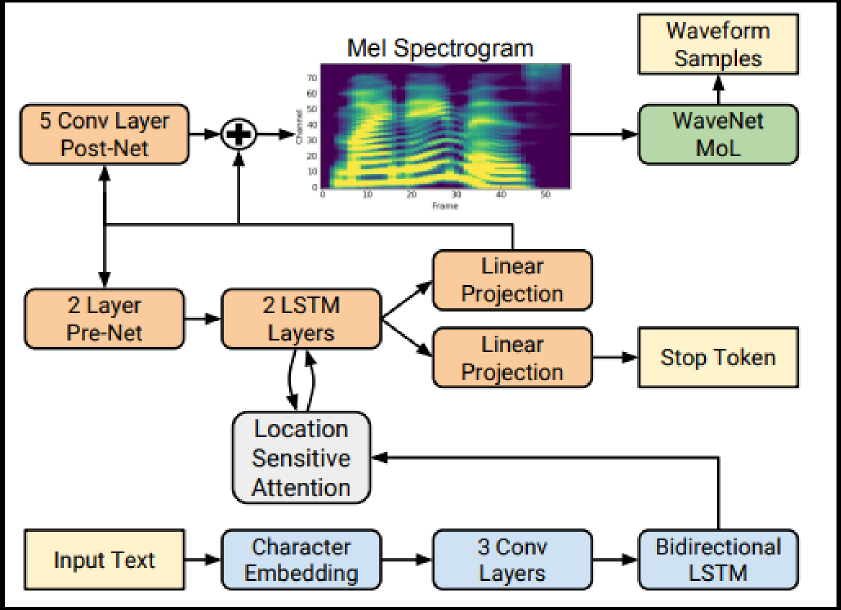
3. WaveGlow

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 (+=)

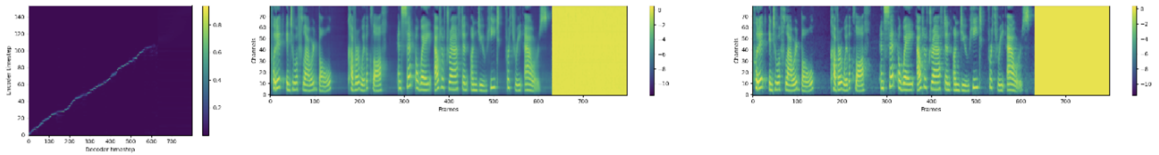
Tacotron 2 (without wavenet)

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=outdir/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

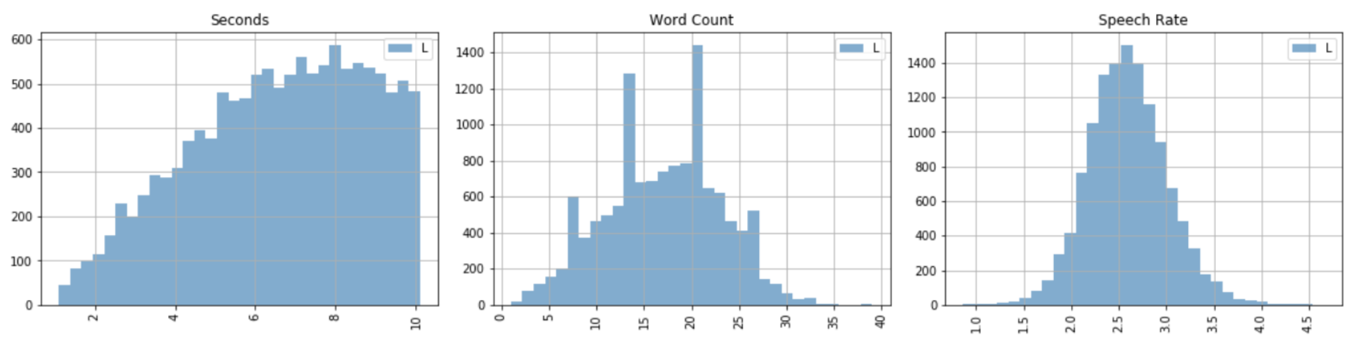
TTS DATASET

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

ARPABET		IPA	Example(s)
1-letter	2-letter		
a	AA	ɑ	balm, bot
@	AE	æ	bat
A	AH	ʌ	butt
c	AO	ɔ	bought

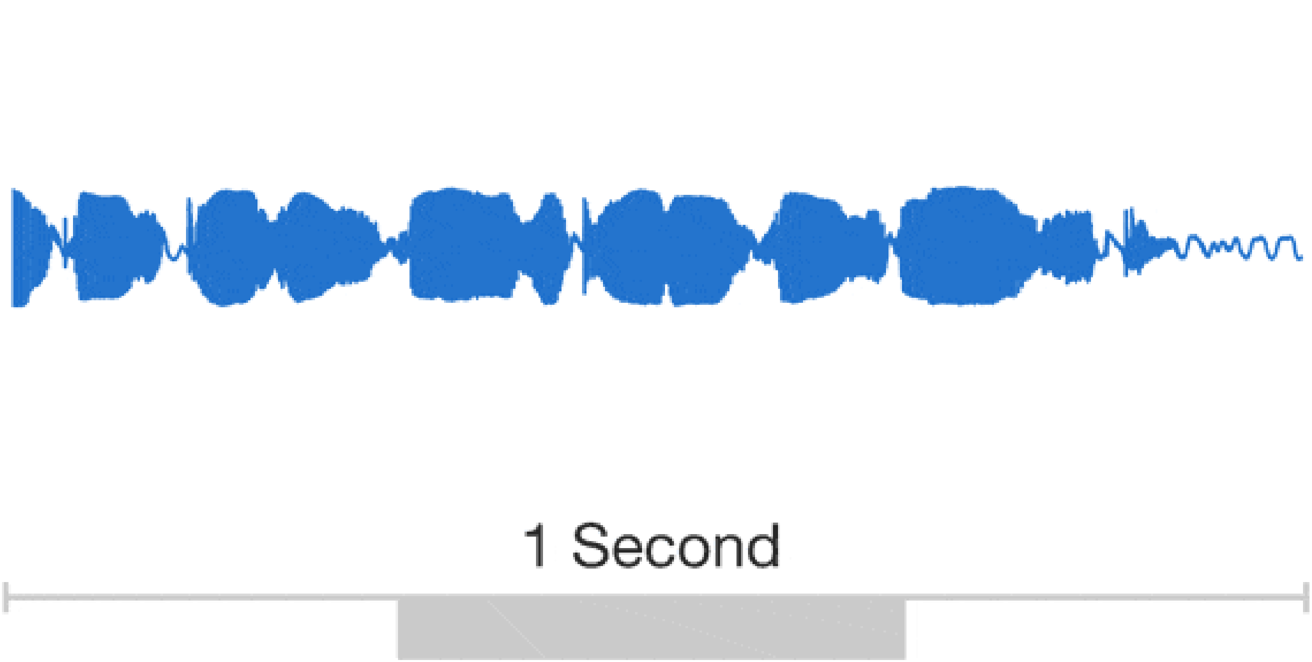


NVIDIA

MEL TO AUDIO WITH WAVENET

$$p(\mathbf{x} \mid \mathbf{h}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}, \mathbf{h})$$

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

System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2 (this paper)	$\mathbf{4.526 \pm 0.066}$

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

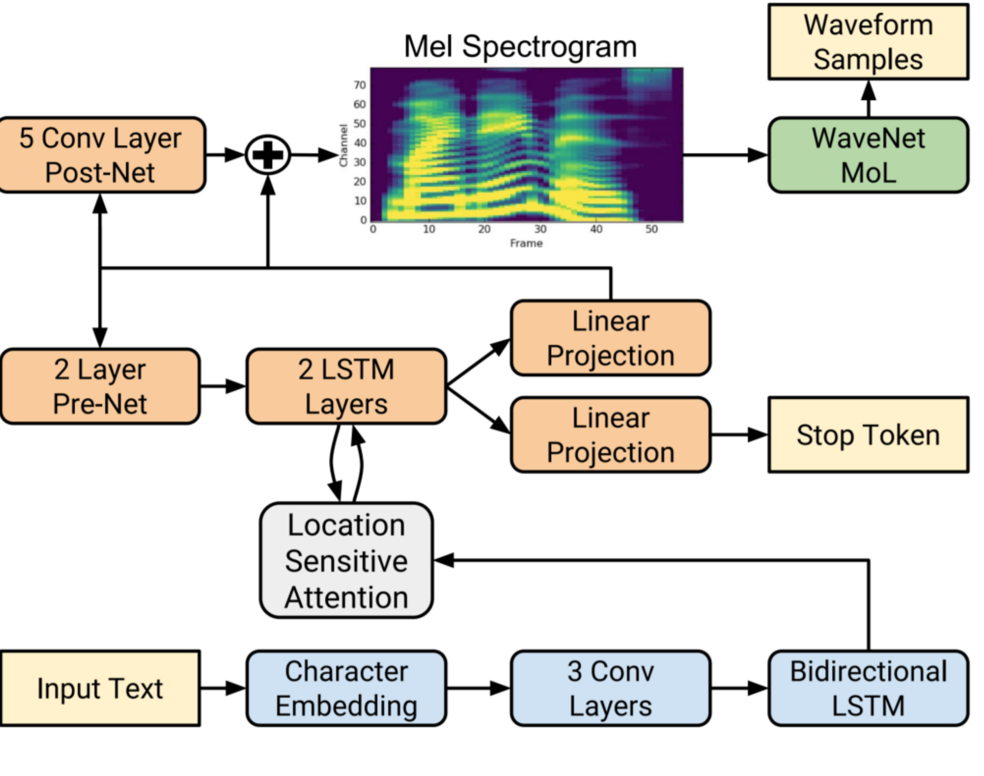
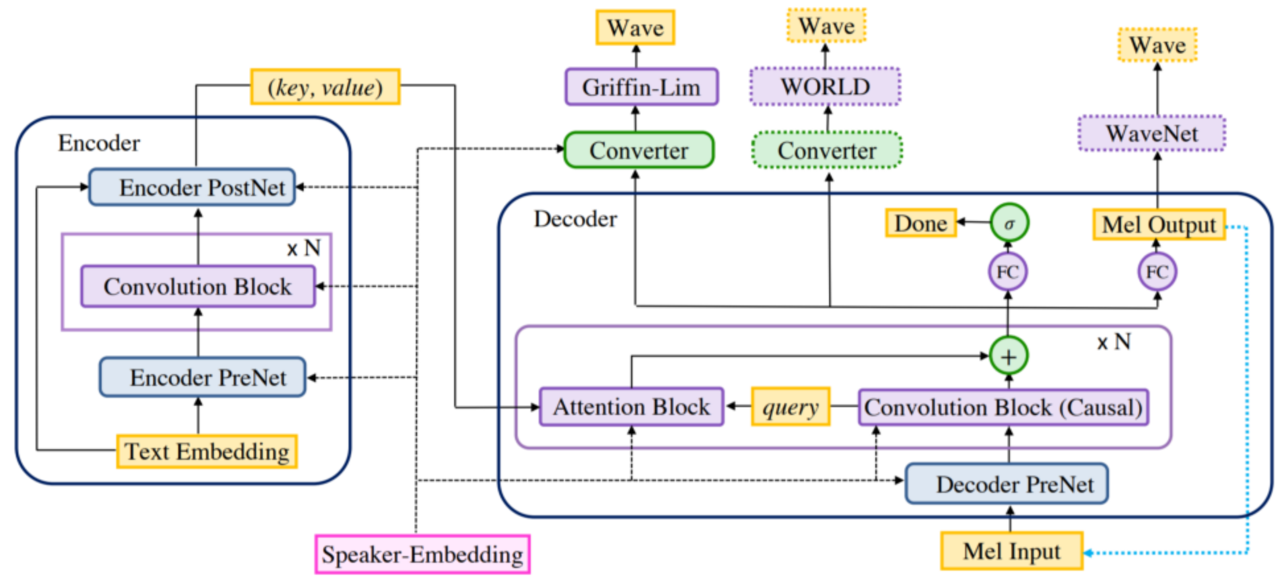


Fig. 1. Block diagram of the Tacotron 2 system architecture.

Shen, J. Et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. https://arxiv.org/abs/1712.05884

DeepVoice 3



Ping, W. *Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning*. https://arxiv.org/abs/1710.07654

WAVENET IS THE BOTTLENECK

TacoTron2

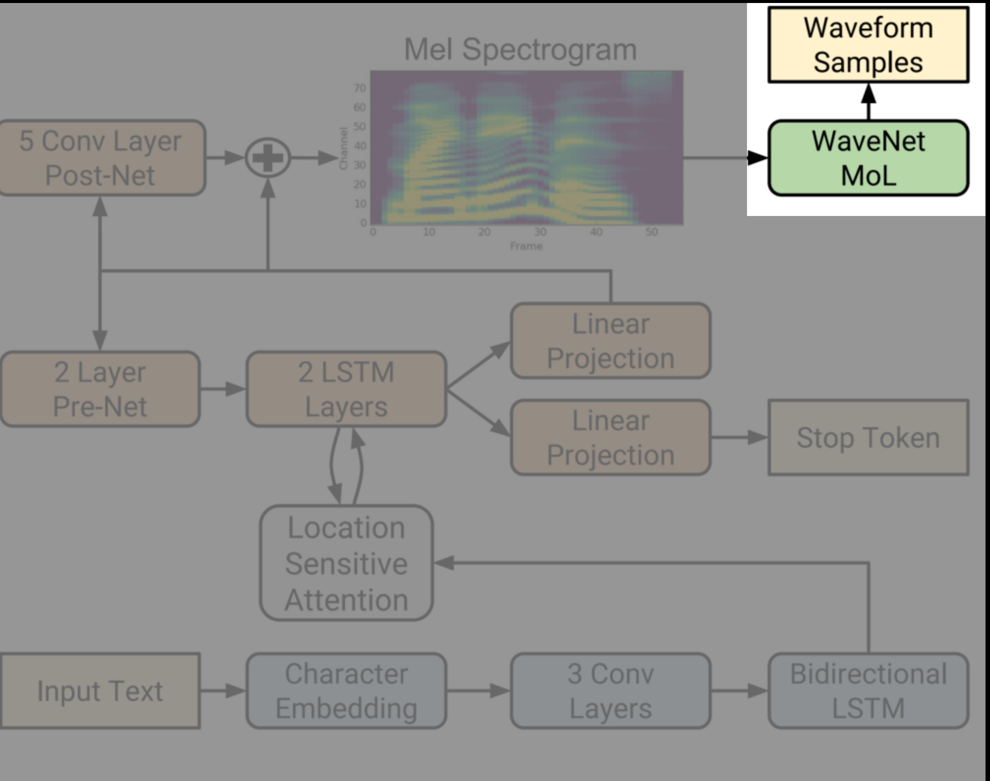
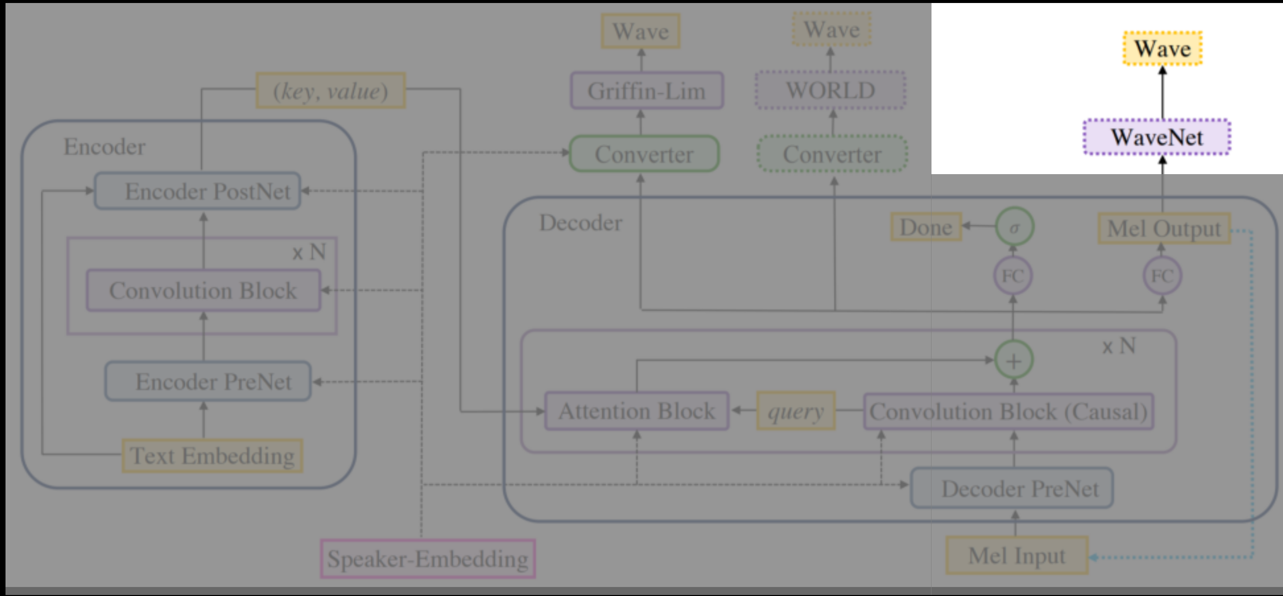


Fig. 1. Block diagram of the Tacotron 2 system architecture.

Shen, J. Et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. https://arxiv.org/abs/1712.05884

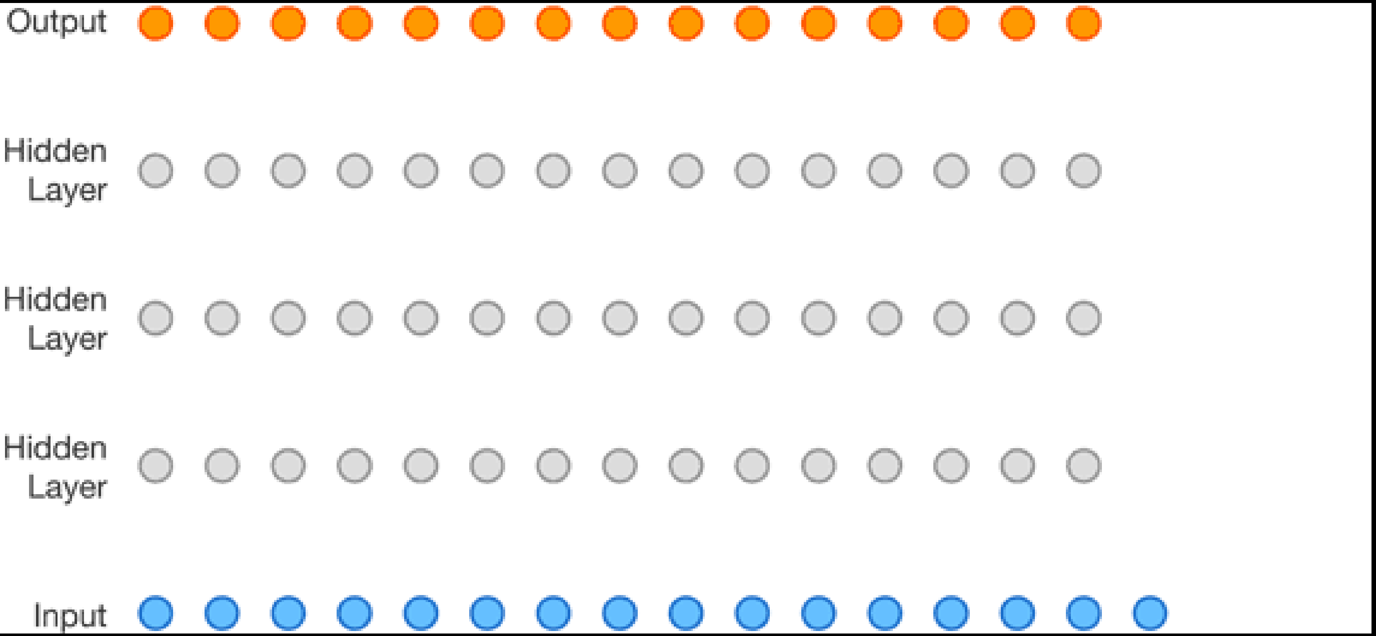
DeepVoice 3



Ping, W. *Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning*. https://arxiv.org/abs/1710.07654

AUTO-REGRESSION IS INHERENTLY SERIAL

$$P(x_0, x_1, x_2, \dots) = P(x_0)P\,(x_1|x_0)P(x_2|x_1, x_0)\dots$$

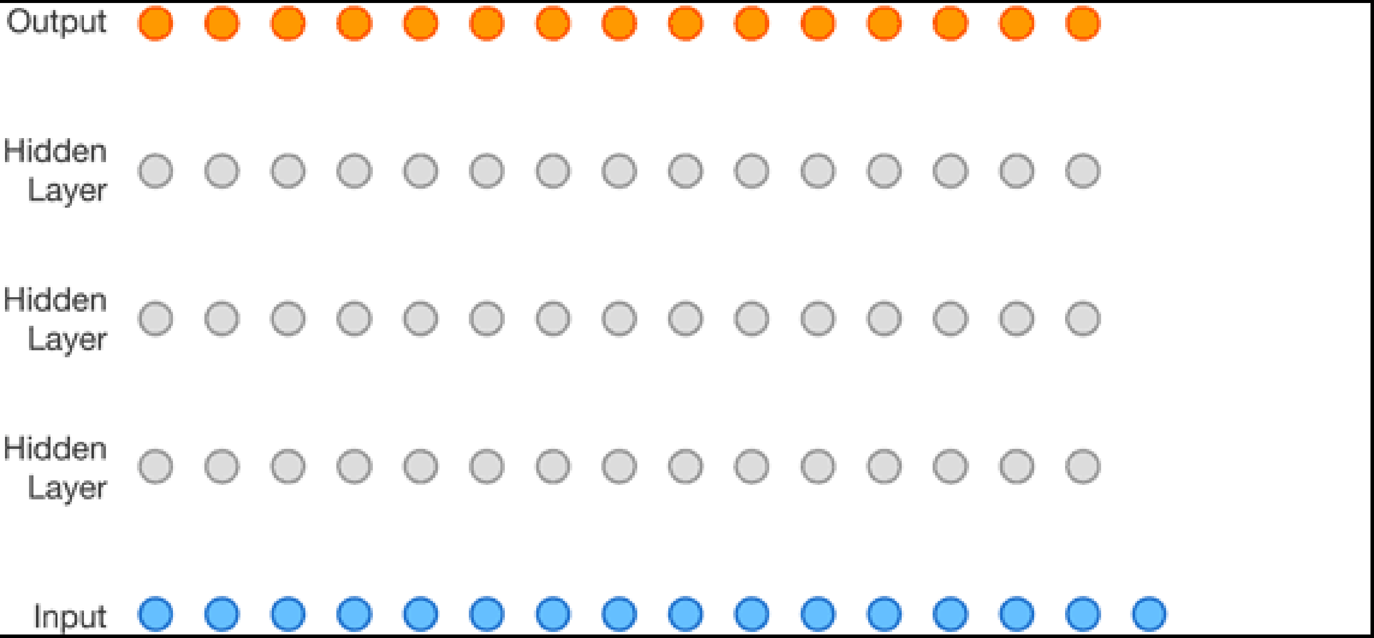


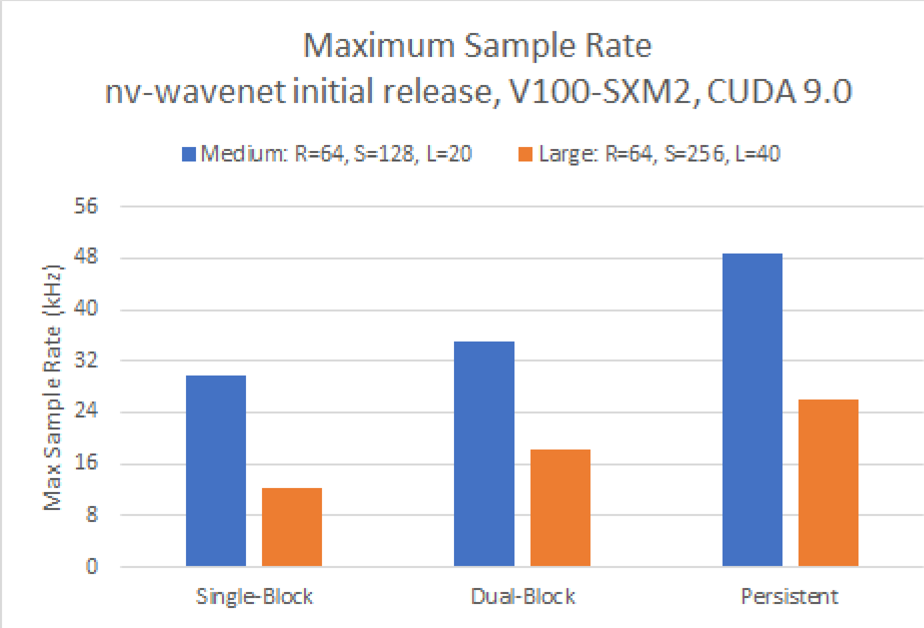
van den Oord, A. *WaveNet: A Generative Model for Raw Audio.*
https://arxiv.org/pdf/1609.03499.pdf

AUTO-REGRESSION IS INHERENTLY SERIAL

$$P(x_0, x_1, x_2, \dots) = P(x_0)P\,(x_1|x_0)P(x_2|x_1, x_0)\dots$$

NV-WaveNet

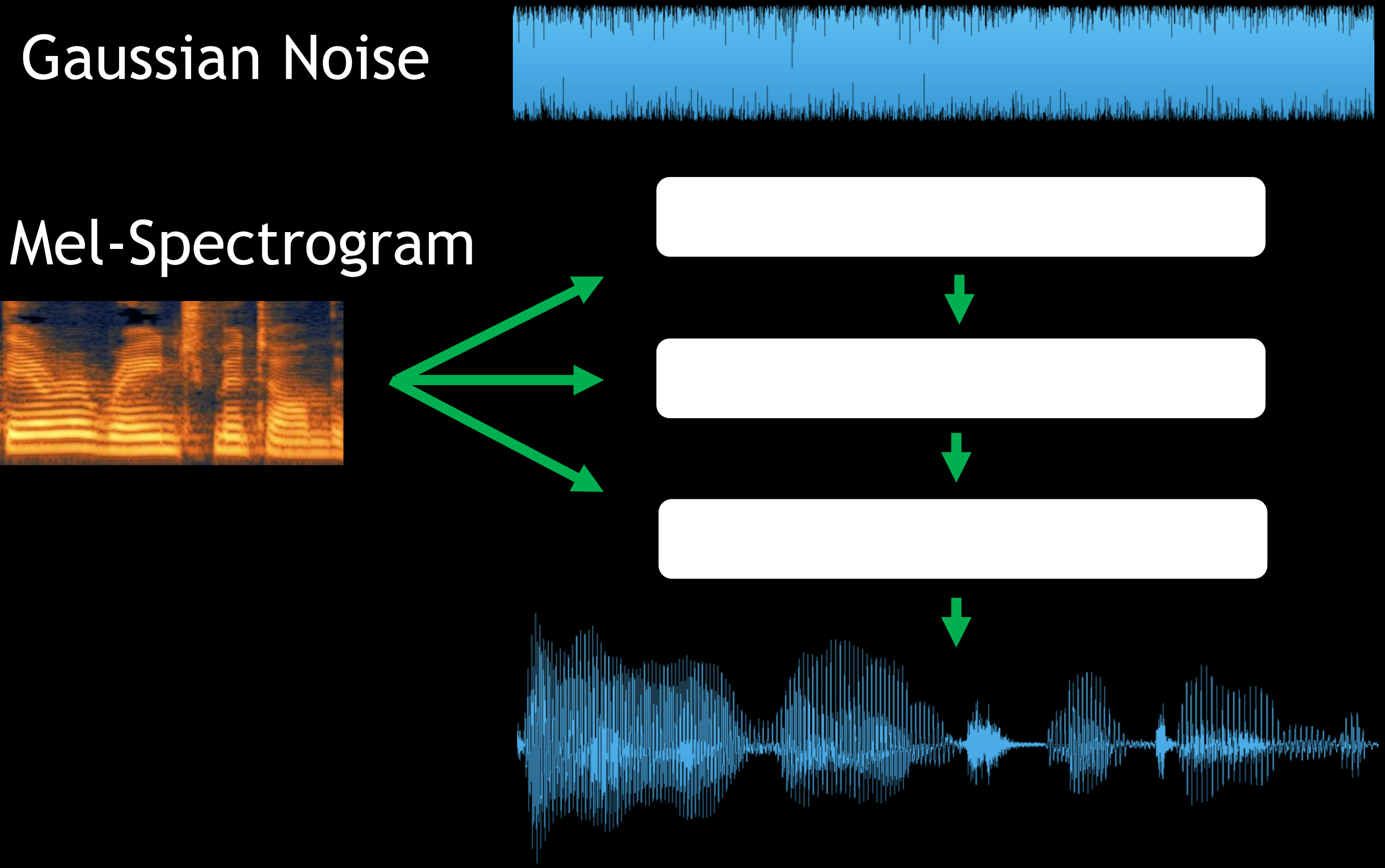




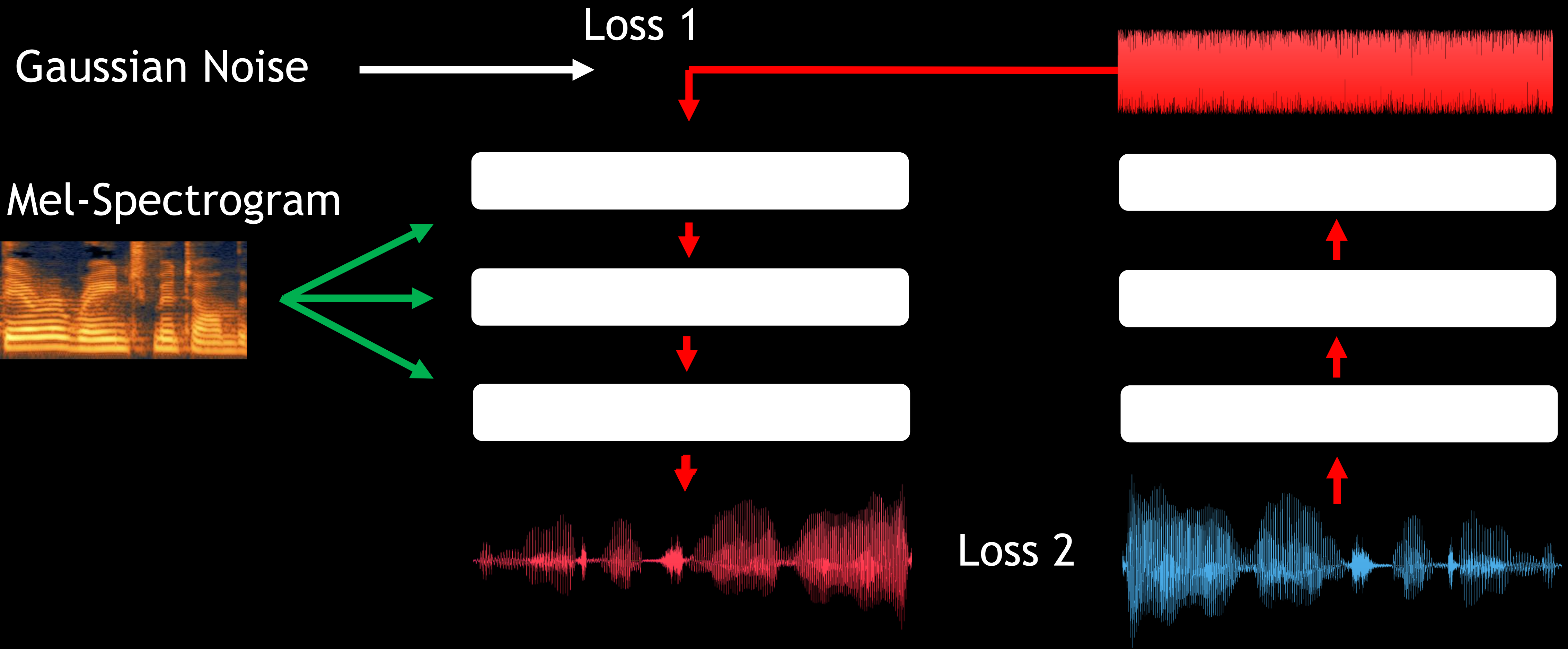
van den Oord, A. *WaveNet: A Generative Model for Raw Audio.* https://arxiv.org/pdf/1609.03499.pdf

https://github.com/NVIDIA/nv-wavenet

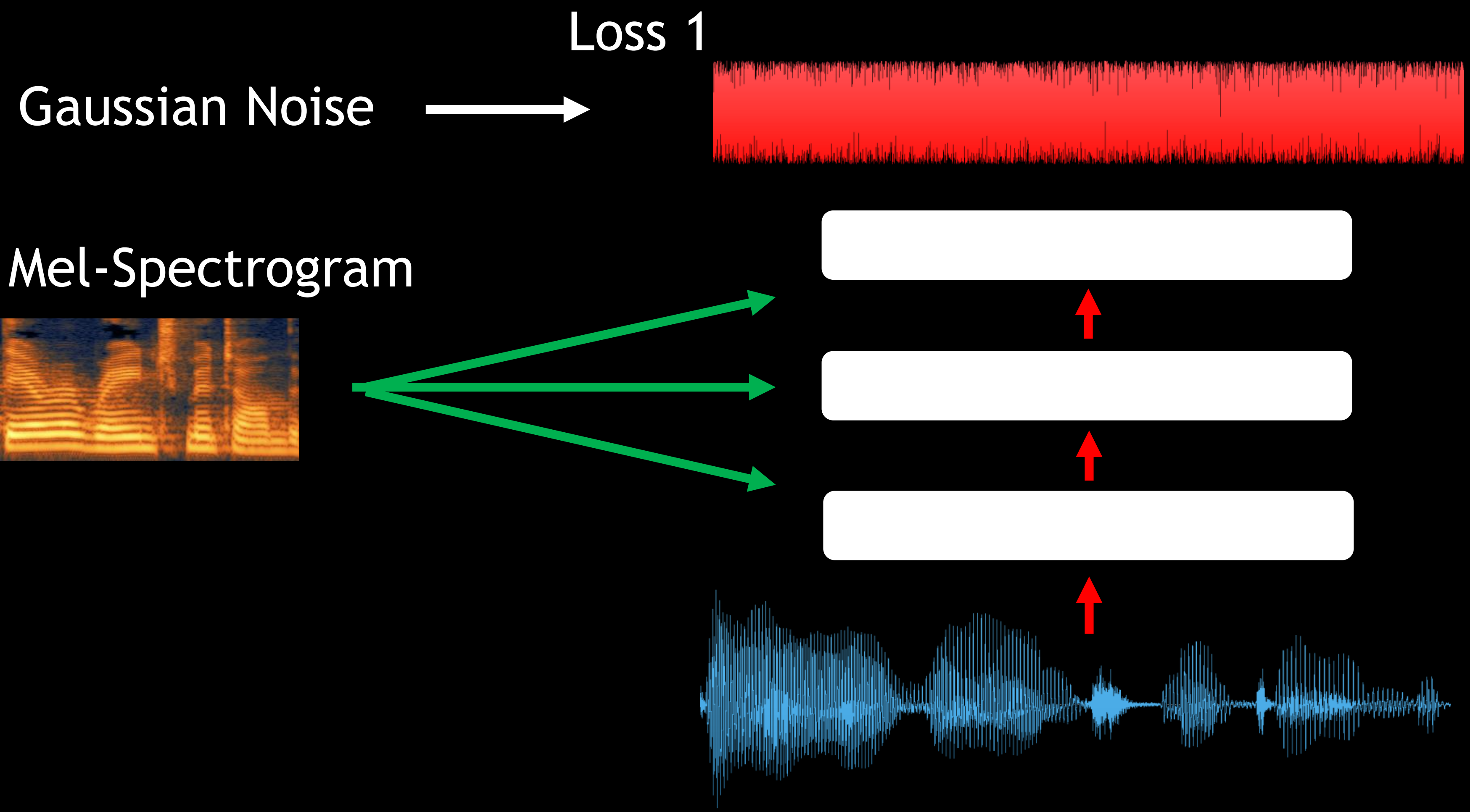
TRANSFORMING WHITENOISE TO AUDIO IS PARALLEL



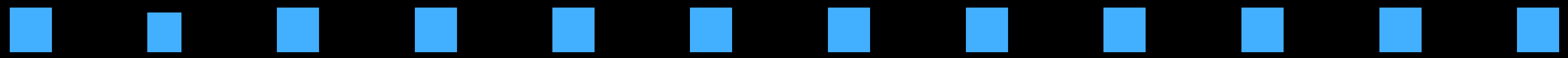
AUTO-ENCODER
(APPROXIMATING LIKELIHOOD)



INVERTIBLE NETWORK (EXACT LIKELIHOOD)

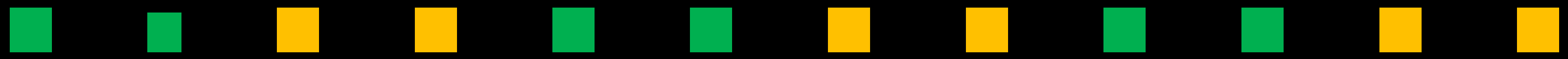


HOW TO MAKE A NETWORK INVERTIBLE



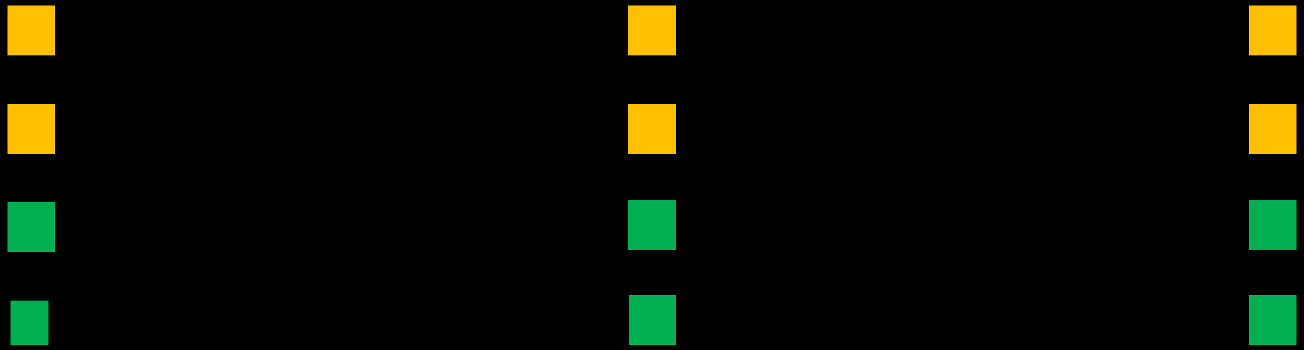
audio samples

HOW TO MAKE A NETWORK INVERTIBLE



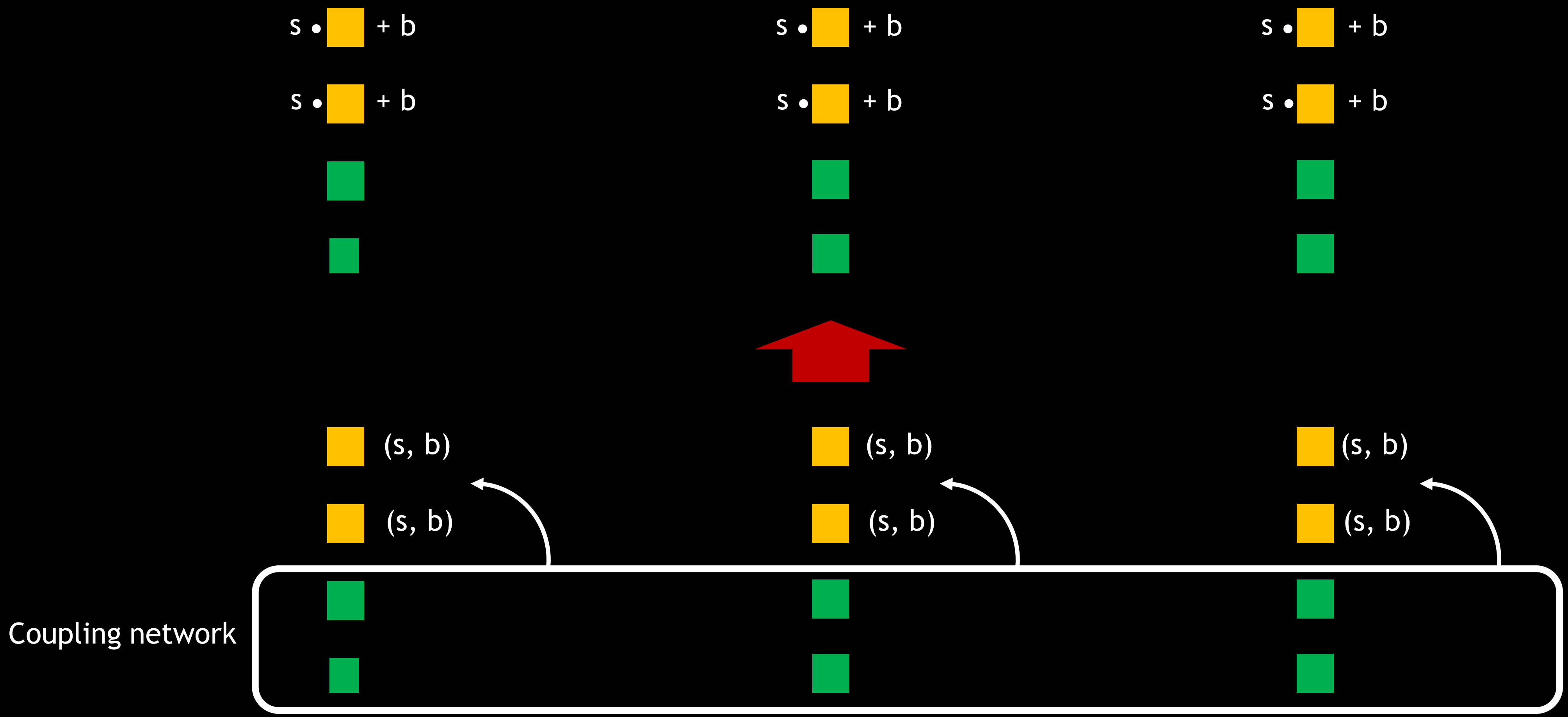
audio samples

HOW TO MAKE A NETWORK INVERTIBLE

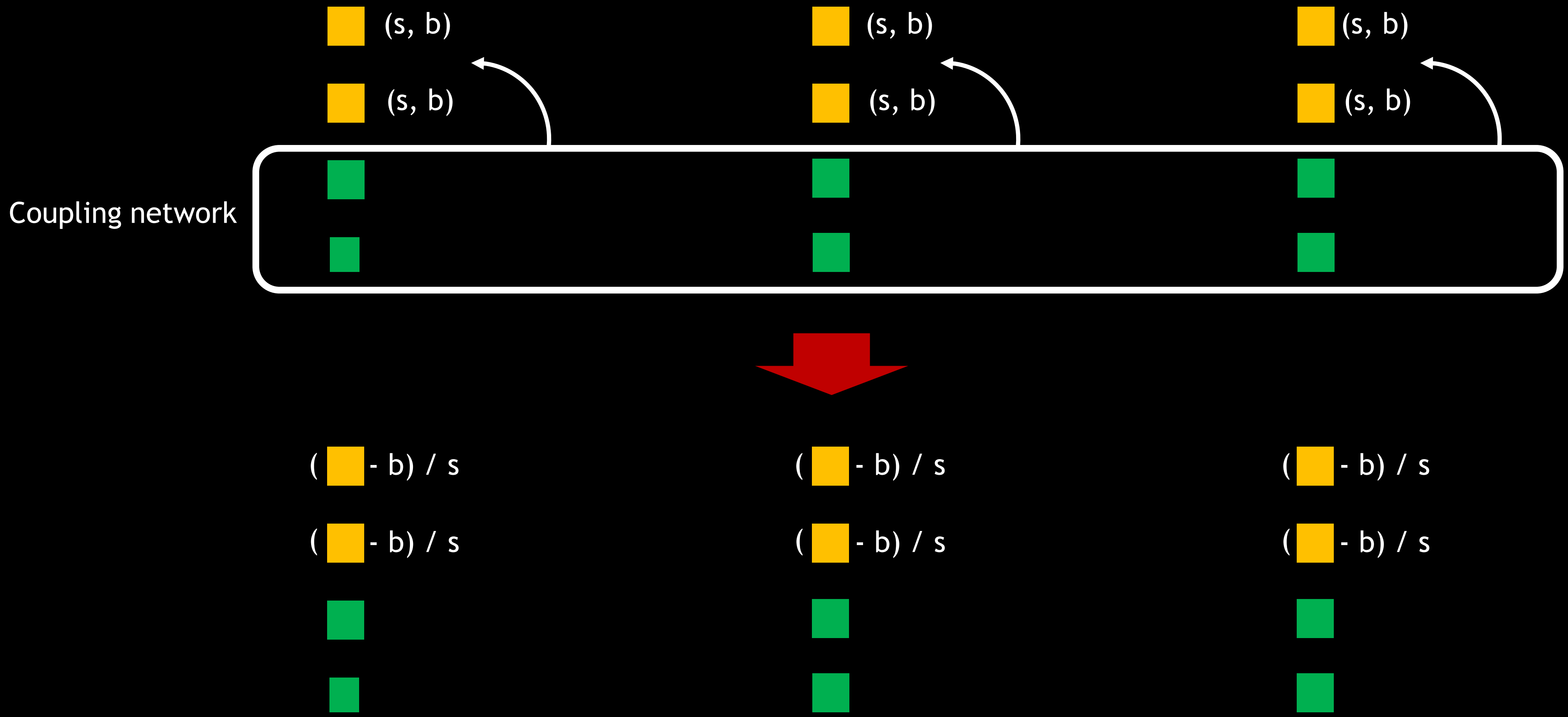


HOW TO MAKE A NETWORK INVERTIBLE

HOW TO MAKE A NETWORK INVERTIBLE



HOW TO MAKE A NETWORK INVERTIBLE



Waveglow

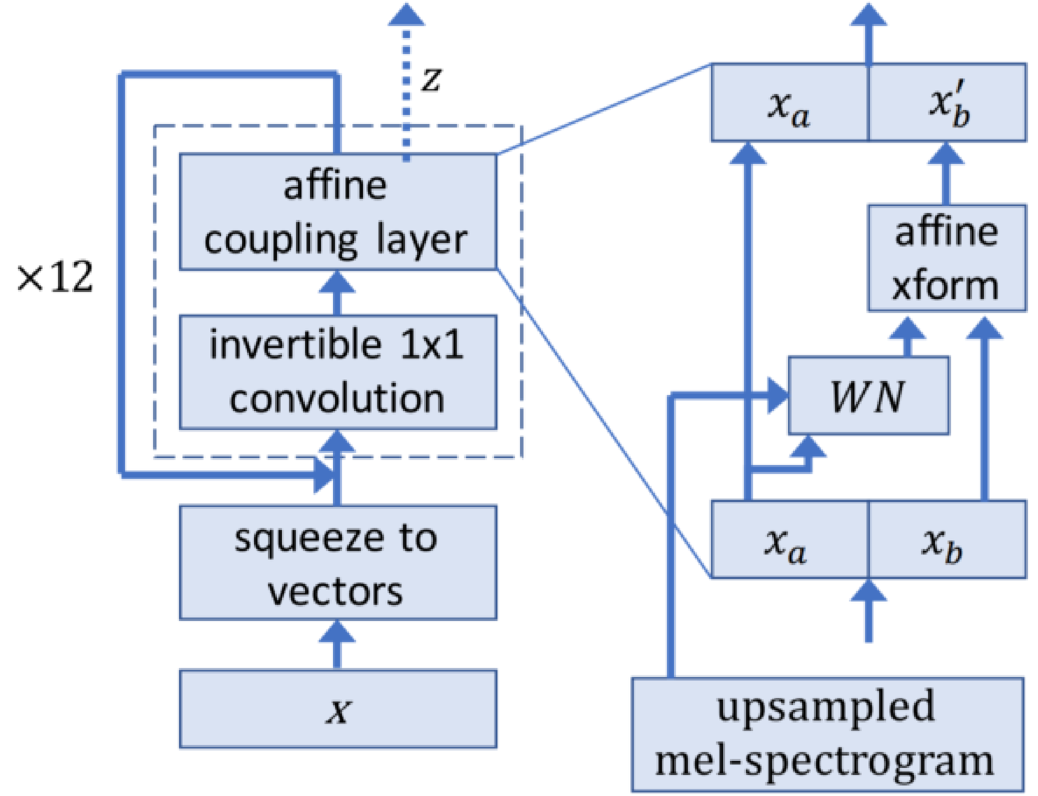
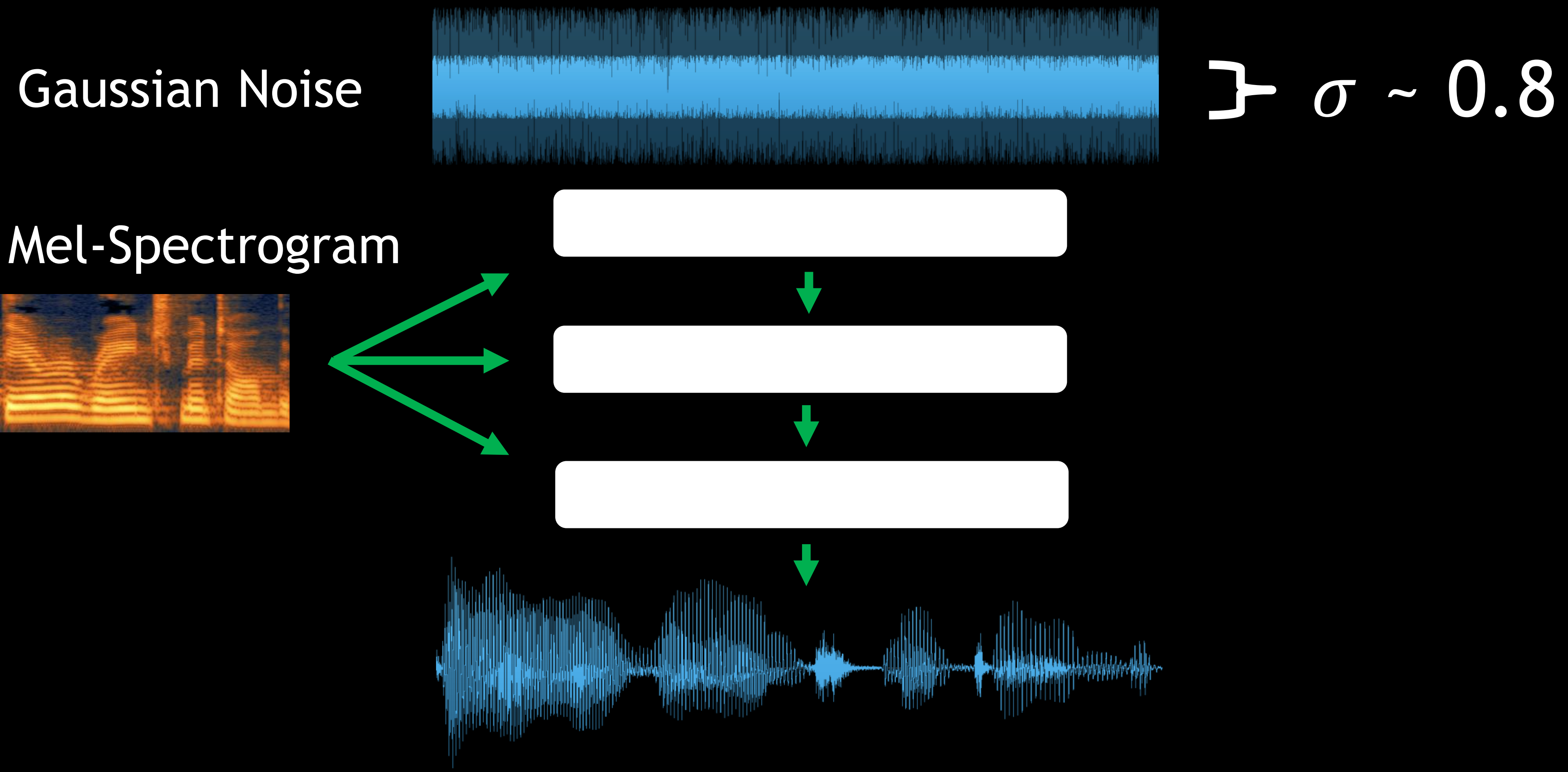


Fig. 1: WaveGlow network

$$\log p_\theta(\boldsymbol{x}) = -\frac{\boldsymbol{z}(\boldsymbol{x})^T \boldsymbol{z}(\boldsymbol{x})}{2\sigma^2} + \sum_{j=0}^{\#coupling} \log \boldsymbol{s}_j(\boldsymbol{x}, \textit{mel-spectrogram}) + \sum_{k=0}^{\#conv} \log \det |\boldsymbol{W}_k|$$

https://github.com/NVIDIA/waveglow

DECREASING TEMPERATURE CAN HELP



PARALLEL SOLUTION WORKS

Model	Mean Opinion Score (MOS)
Griffin-Lim	3.823 ± 0.1349
WaveNet	3.885 ± 0.1238
WaveGlow	3.961 ± 0.1343
Ground Truth	4.274 ± 0.1340

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

WaveGlow (published): 520 khz (24.5x realtime)

PARALLEL SOLUTION WORKS

Model	Mean Opinion Score (MOS)
Griffin-Lim	3.823 ± 0.1349
WaveNet	3.885 ± 0.1238
WaveGlow	3.961 ± 0.1343
Ground Truth	4.274 ± 0.1340

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

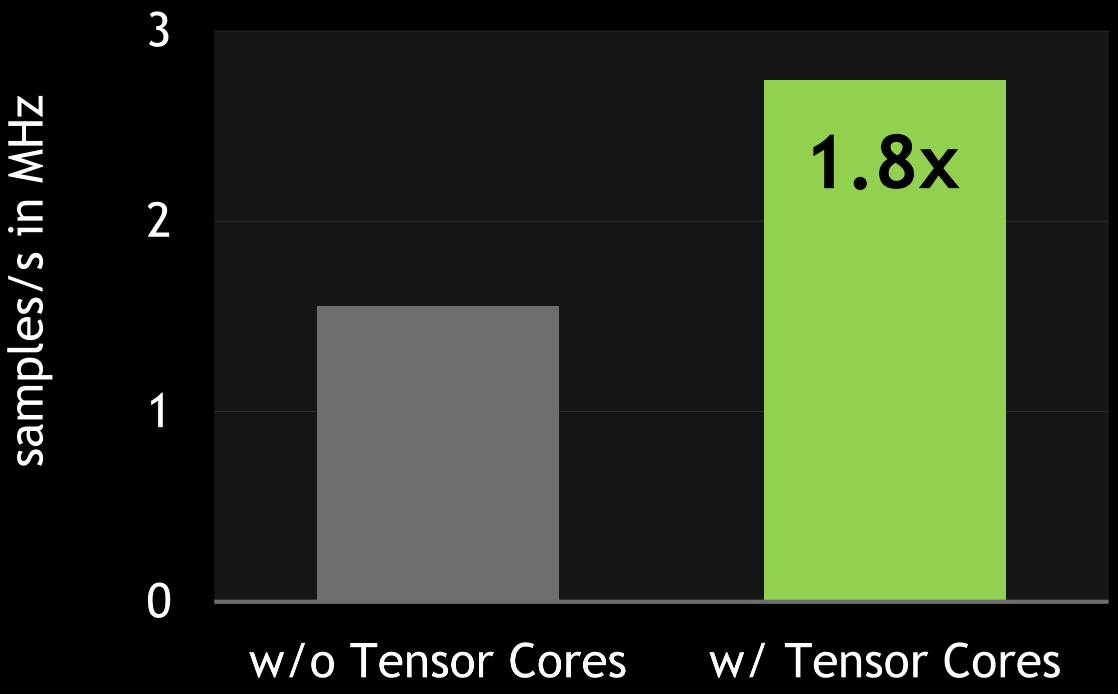
- More like optimized auto-regressive
- Can get some parallelism with subscale trick

OUTLINE

1. Text to Speech Synthesis
2. Tacotron 2
3. WaveGlow
4. TTS and Tensor Cores

INFERENCE SPEED UP

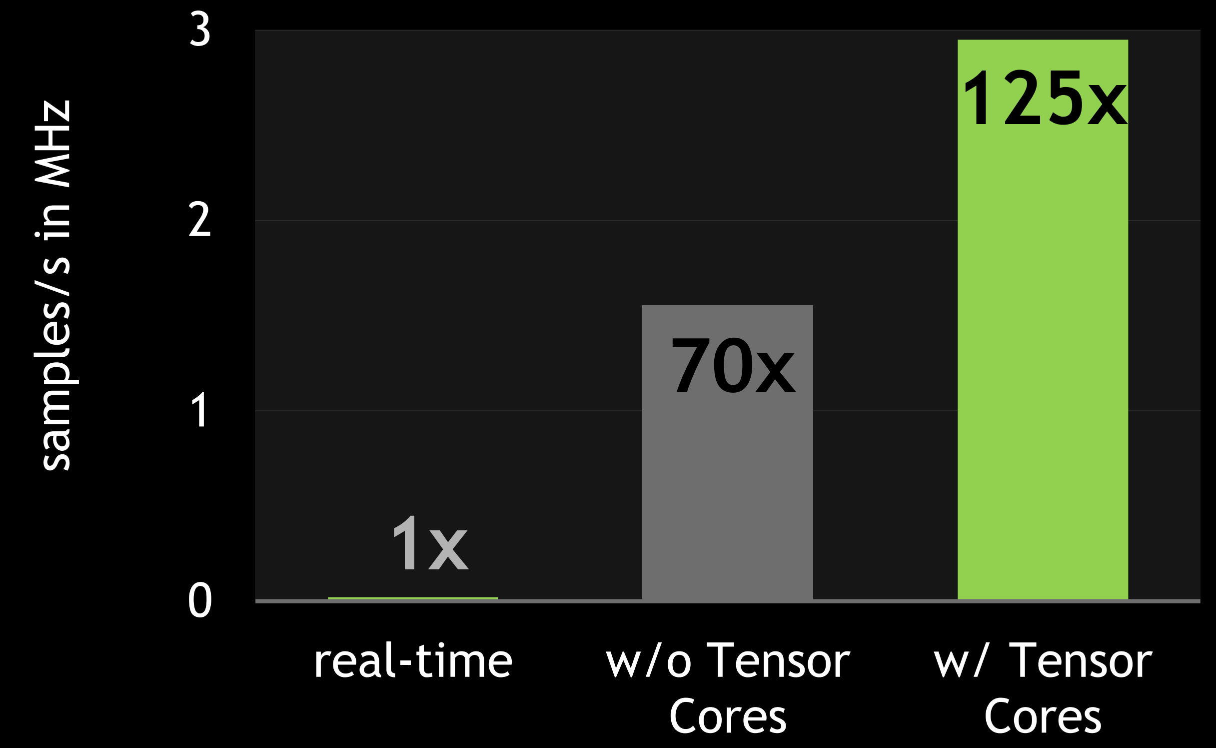
with Tensor Cores – Automatic Mixed Precision



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

INFERENCE SPEED UP

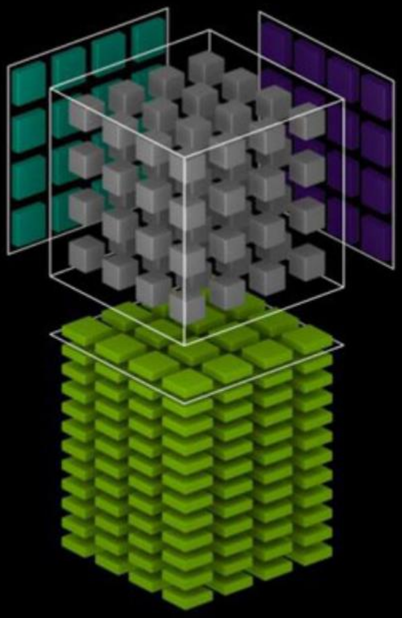
with Tensor Cores – Automatic Mixed Precision



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

NVIDIA

TENSOR CORES SPEED UP MATRIX MULTIPLICATIONS



$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \times \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

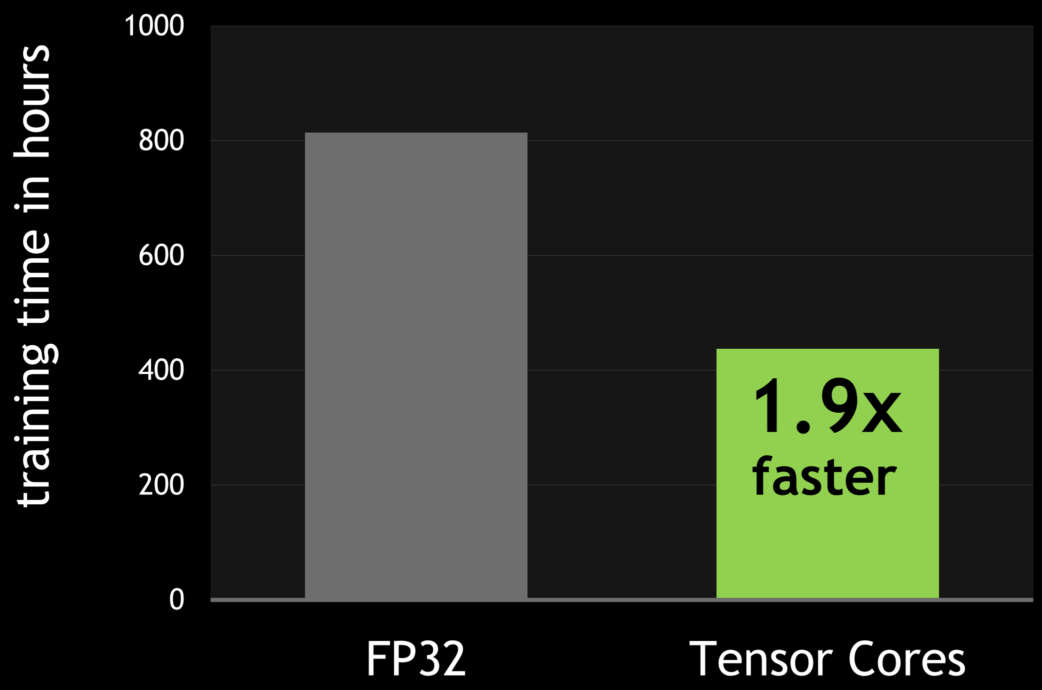
FP16 x FP16 + FP32

	w/o Tensor Cores	w/ Tensor Cores
Inference time	29ms	15ms

2X FASTER INFERENCE WITH TENSOR CORES

TRAINING SPEED UP

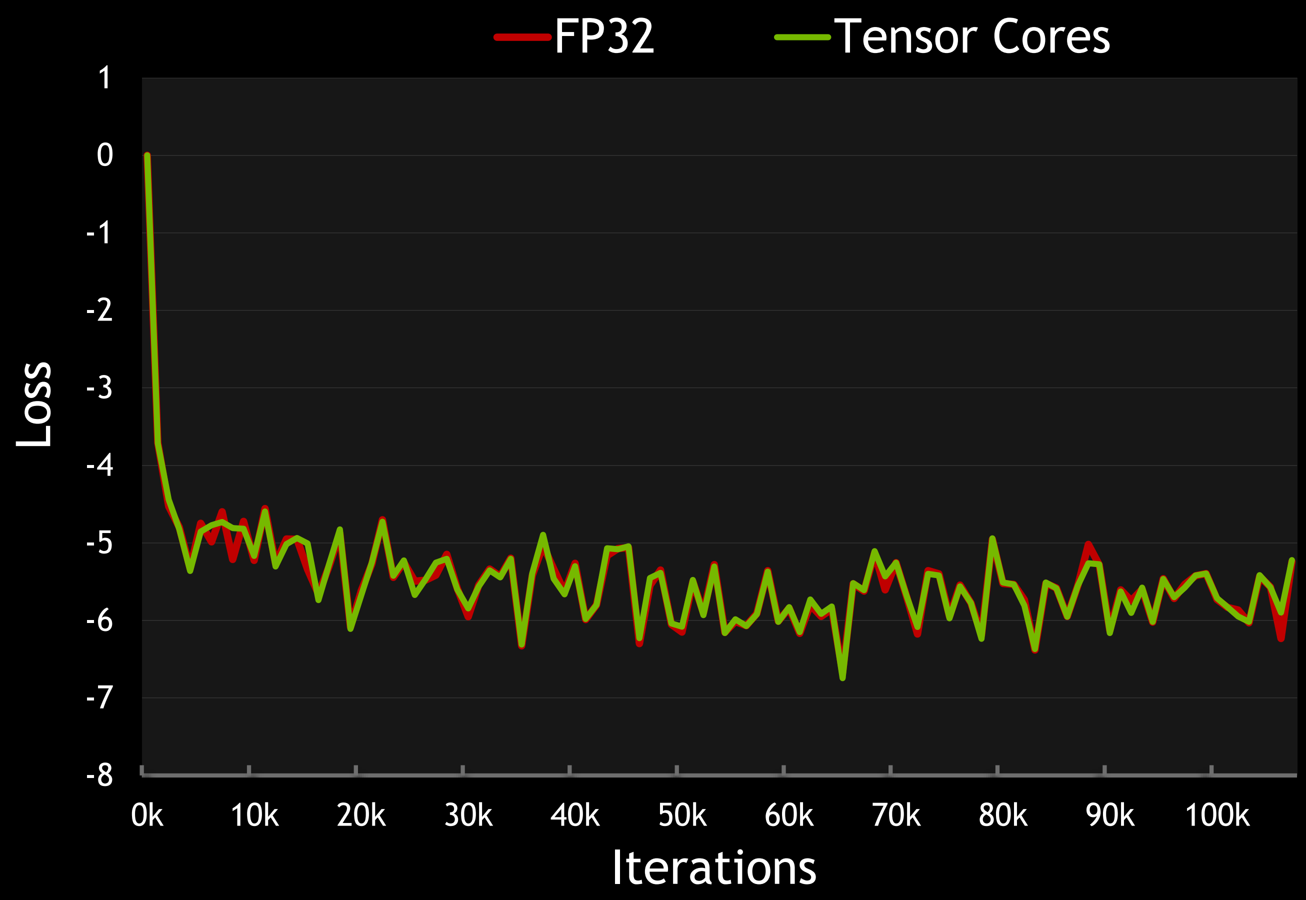
with Tensor Cores – Automatic Mixed Precision



On DGX-1
1 Tesla V100 GPU
over 1000 Epochs

TRAINING WITH TENSOR CORES

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 prevents gradient underflow
- Different levels of optimization
  - white/black list allow user to enforce precision
- Easy code adjustment

INFERENCE 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)

...
```

INFERENCE 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

FP32

```
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

Tensor Cores with AMP

1.9x speed up

```
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, iters_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: {}".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()
            with amp_handle.scale_loss(loss, optimizer) as scaled_loss:
                scaled_loss.backward()
            optimizer.step()
....
```

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

NVIDIA®