Non-autoregressive TTS

Normalizing Flows

Anastasia Kunzetsova, ECE 477, 10/27/2026

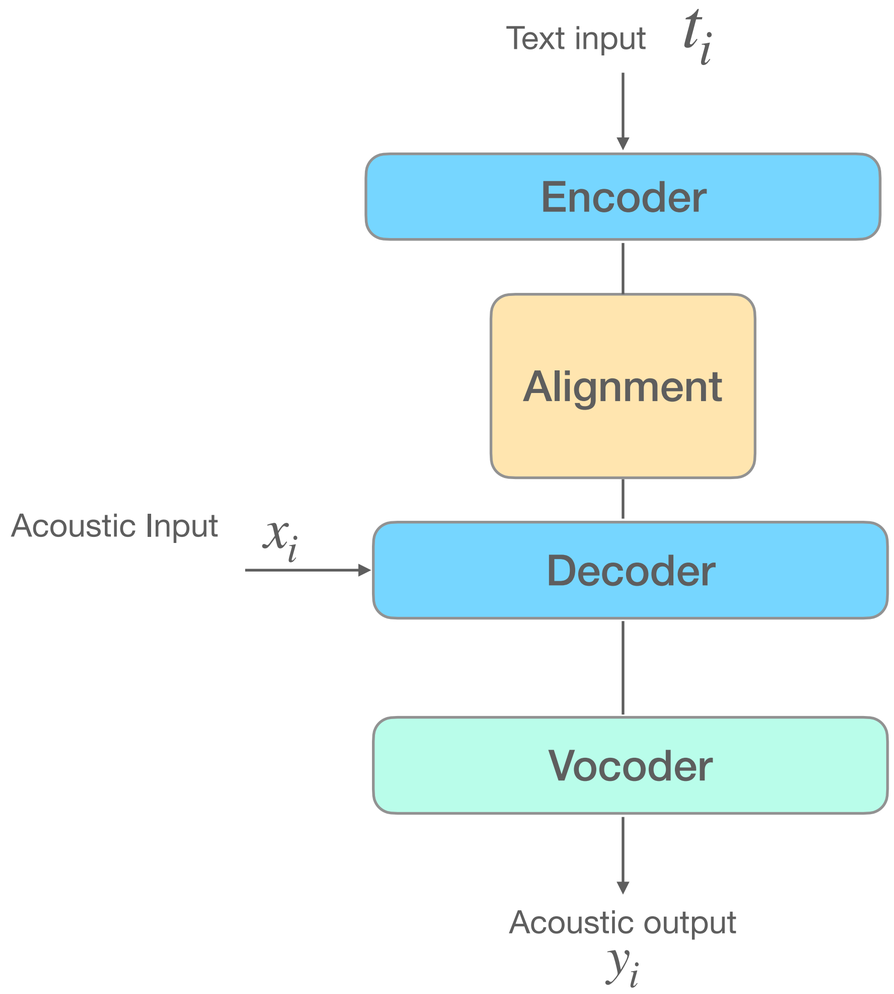
Introduction

Text-to-Speech (TTS) and auto-regressiveness

- TTS maps a text input to the acoustic input through latent representations to produce speech output.
- Both speech and text are time-variable thus autoregressive models (e.g. LSTMs) are intuitive architecture choice for TTS.

Drawbacks:

- **Slow inference:** dependency of t_i on t_{i-1}, we cannot generate next frame before generating the previous;
- Scales poorly to large inputs; linear dependence on the input size and inference time. Thus unsuitable for on-device and online computations.
- Slow to train.



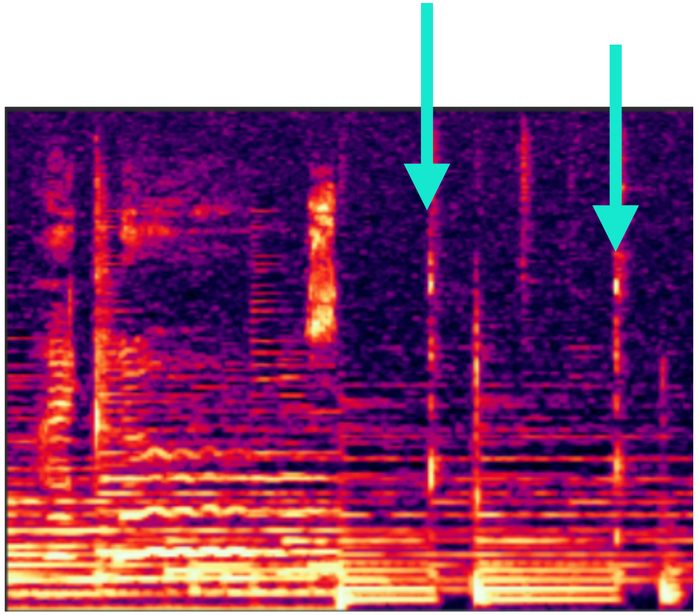
Introduction

Text-to-Speech (TTS) and auto-regressiveness

What can we do to speed up autoregressive training/inference?

- Teacher forcing (improves convergence);
- Knowledge distillation (reduce the size of the model) ;
- **Use a different, more parameter efficient architecture altogether.**

Result in poor quality (artifacts, unnaturalness)

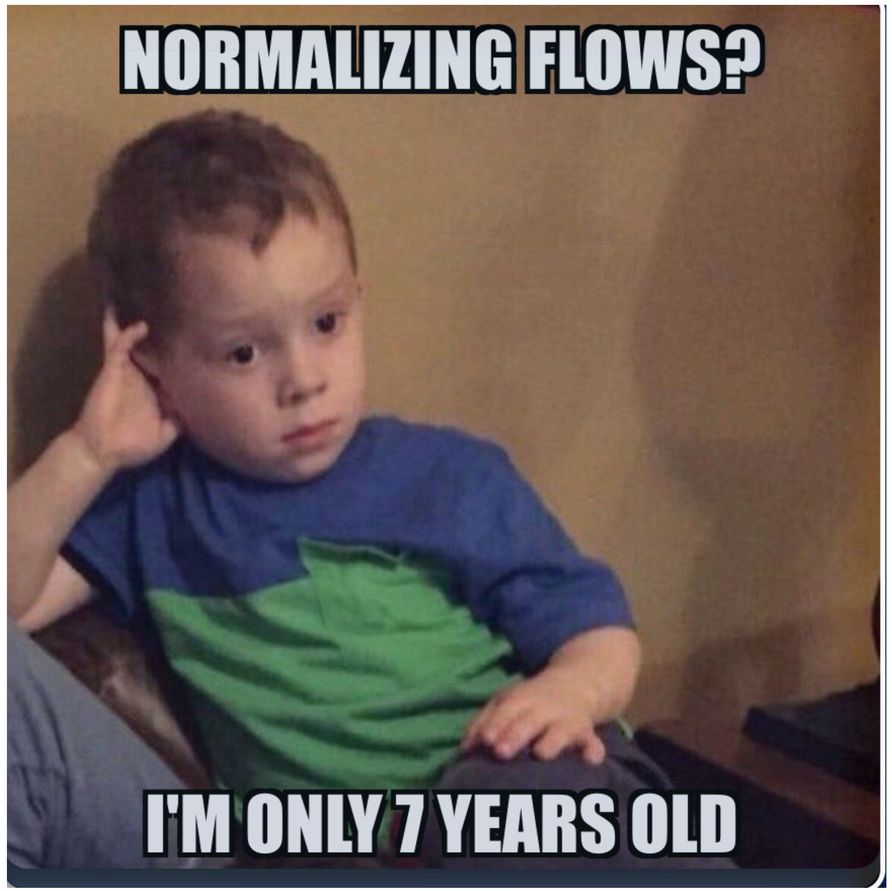


Outline

- **Introduction.** What is TTS and limitations of auto-regressive models;
- **Normalizing flows.** Basic formal definition and roots of normalizing flows;
    - Change of variables, Compositionality principle and invertibility;
    - Sampling from normalizing flows;
    - Coupling flows.
- **Glow-TTS.** A flow-based TTS model.

What are the normalizing flows?

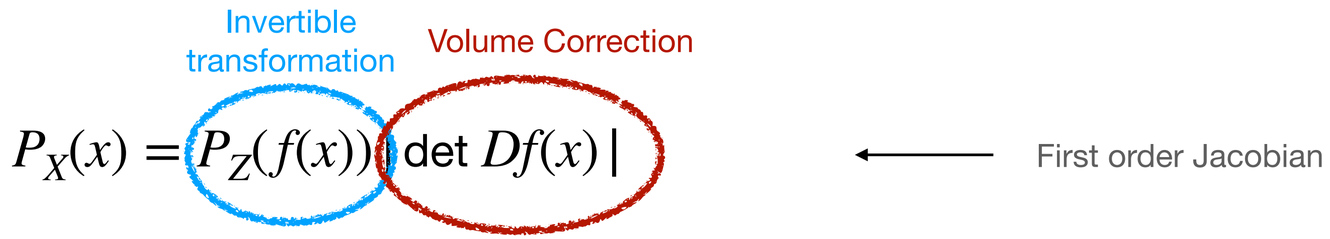
- Efficient to sample;
- Highly expressive;
- Allow for **exact** mapping of distributions through *invertible transformations*;
- Simple MaxLikelihood objective can be trained with conventional SGD easily;
- Easy *interpolation* in the *latent space*.



Normalizing flows
(Formally)

Imagine a PDF of data $P_X(x)$ (could be anything)

Change of variable formula:



Tells you how the data changes under the certain transformation $f(x)$, could be any function.

$Z = f(x)$ - **latent** (unknown) variable, invertible, differentiable.

Invertible: because we need to know how to go both directions $X \rightarrow Z$ and $X \leftarrow Z$;

Differentiable: we need to use gradient descent based algorithm, to apply chain rule, the function has to be differentiable.

$Df(x)$ is the **volume correction** term. Defines how the density function stretches/compresses in space.

Cred. M. Brubaker. Introduction to Normalizing Flows (ECCV 2020)

Normalizing flows

(Formally)

So what are we trying to do?

Goal: transform some **very** complex distribution (e.g. ImageNet) into something very simple, for example, normal Gaussian:

$\mathcal{N}(z \mid 0, I)$ this is our latent distribution $P_Z(z)$ we are trying to achieve. **Why?**

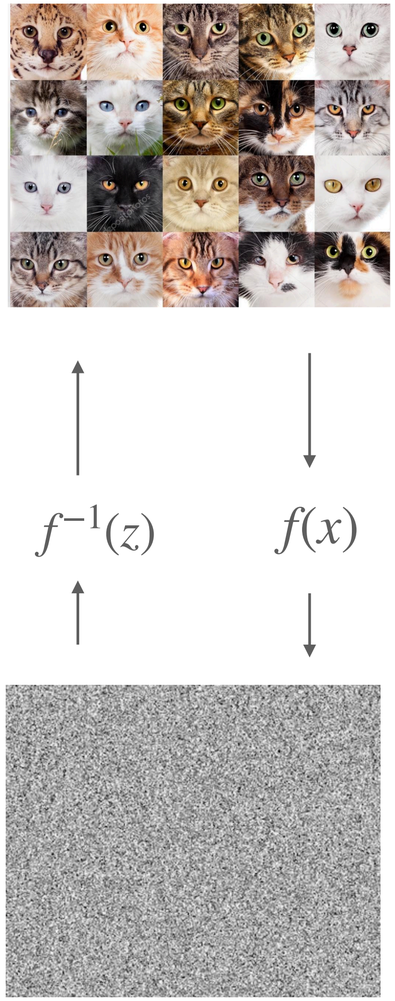
Normal Distribution is easy to sample from. Convenient in many other ways.

Sampling:

1. Sample $z \sim P_Z(\cdot)$
2. Compute the inverse $x = f^{-1}(z)$
3. Maximize LogLikelihood with SGD.

Flow with parameters θ

$$\max_{\theta} \sum_{i=1}^{N} \log P_Z(f(x_i \mid \theta)) + \log \left| \det Df(x_i \mid \theta) \right|$$



Cred. M. Brubaker. Introduction to Normalizing Flows (ECCV 2020)

So what are flows?

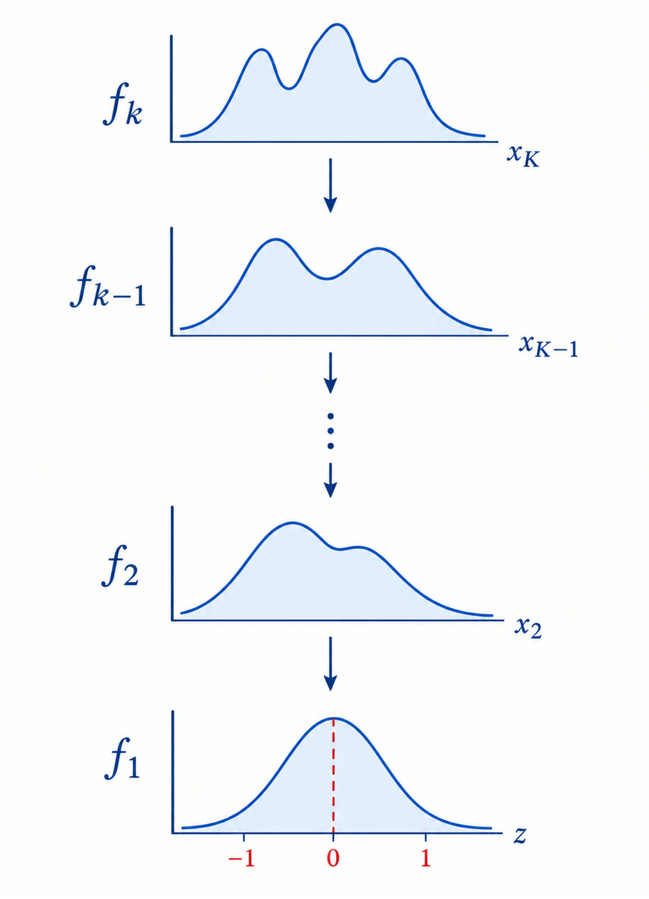
$$f = f_K \circ f_{K-1} \circ \cdots \circ f_2 \circ f_1$$

Normalizing flow is a parametric function $f(x)$ with following properties:

- Invertible
- Differentiable
- Inverse is computationally efficient, and Jacobian is computationally tractable

BUT: one function cannot model complex non-linearities. What do we do then? Flows are closed under composition. Which means

We can have any number of transformations to form complex distributions



Complex to normal distribution transform

Cred. M. Brubaker. Introduction to Normalizing Flows (ECCV 2020)

Normalizing flows applications

Computer Vision

Interpolation between faces:

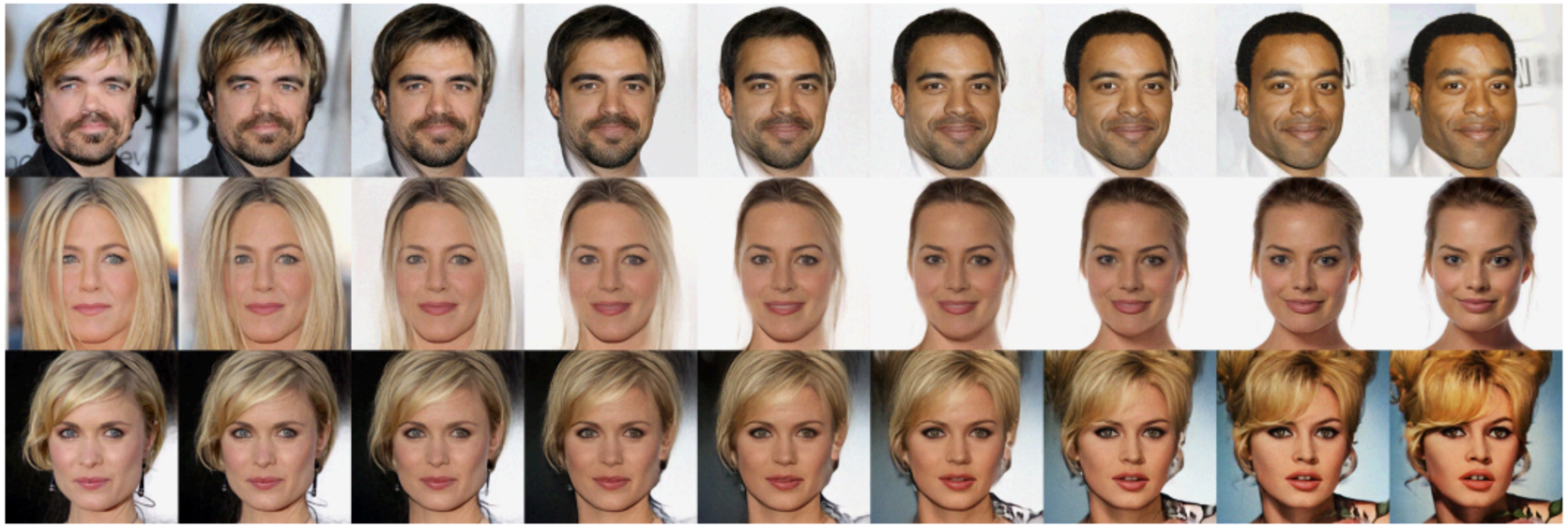


Figure 5: Linear interpolation in latent space between real images

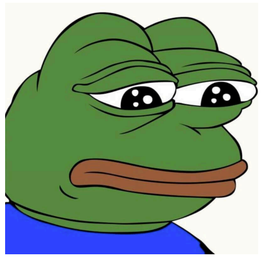
Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." *Advances in neural information processing systems* 31 (2018).

What is the simplest flow

Linear flow:

Forward computation : $f(x) = Ax + b$

Inverse: $f^{-1} = A^{-1}(z - b)$



Why not? Inexpressive, cannot model much in multi-dimensional space.

Coupling flow (non-linear invertible transformation):

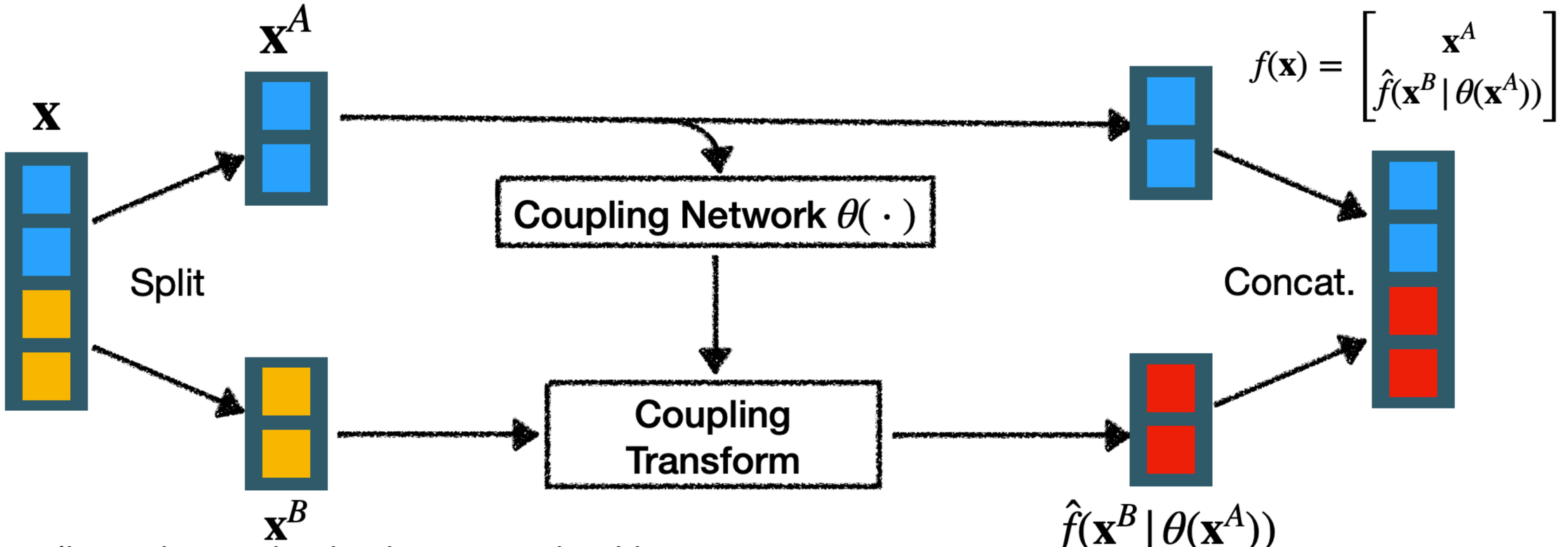
Partition the input into two disjoint subsets $X = (X^A, X^B)$

Apply flow transformation to half of the input:

$$f(x) = (X^A, \hat{f}(X^B \mid \theta(X^A))$$

Where $\hat{f}(\cdot)$ is another flow and parameters depend on X^A and $\theta(\cdot)$ is a coupling network. **Wait what?**

I. Kobyzev, et al. ”Normalizing Flows: An Introduction and Review of Current Methods," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*



The split can be randomized, permuted or Linear transformation can apply before coupling transform.

What is coupling transform?

Most commonly used is *element-wise affine* transformation:

$$\hat{f}(x \mid s, t) = s \odot x + t$$

$$s(X^A) \odot X^B + t(X^A)$$

Cred. M. Brubaker, J, Yu

Where’s TTS?



Glow-TTS

Kim, Jaehyeon, et al. "Glow-tts: A generative flow for text-to-speech via monotonic alignment search." *Advances in Neural Information Processing Systems* 33 (2020): 8067-8077.

Limitations of auto-regressive models:
- Slow inference, ineffective in real time applications;
- Non-robust attention, introducing errors: mispronouncing, skipping, or repeating words;

Contributions:
- Inference speed up, 15.7 times as opposed to Tacotron 2;
- Inference on long utterances due to internal aligner;
- Ability to control intonation and pitch patterns thanks to latent space representations learned by normalizing flows

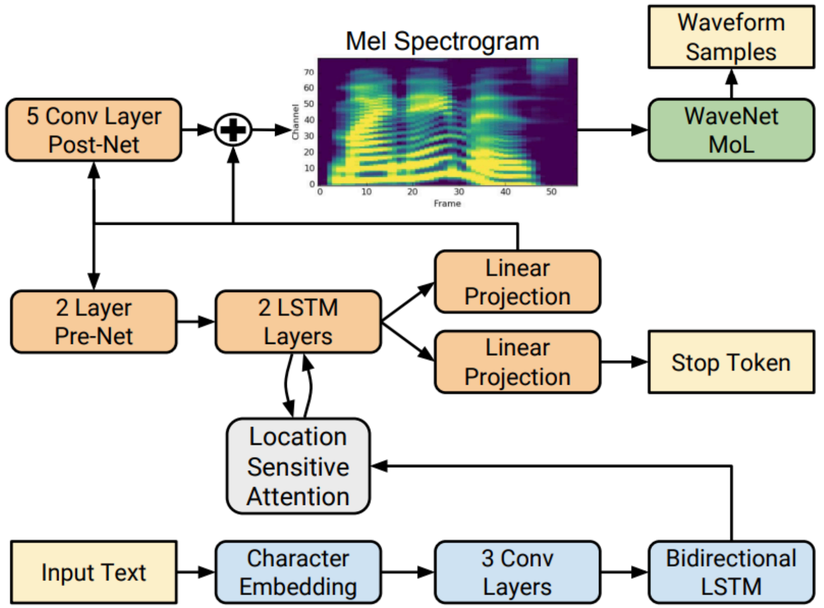


Fig. 1. Block diagram of the Tacotron 2 system architecture.

Shen et al., 2018

Glow-TTS

High-Level

What are we trying to find essentially? Remember Bayes formula

$$P_X(x \mid c) = \frac{P(c \mid x)P(x)}{P(c)} \propto P(c \mid x)P(x)$$

Likelihood (audio) — Likelihood (text) — Prior

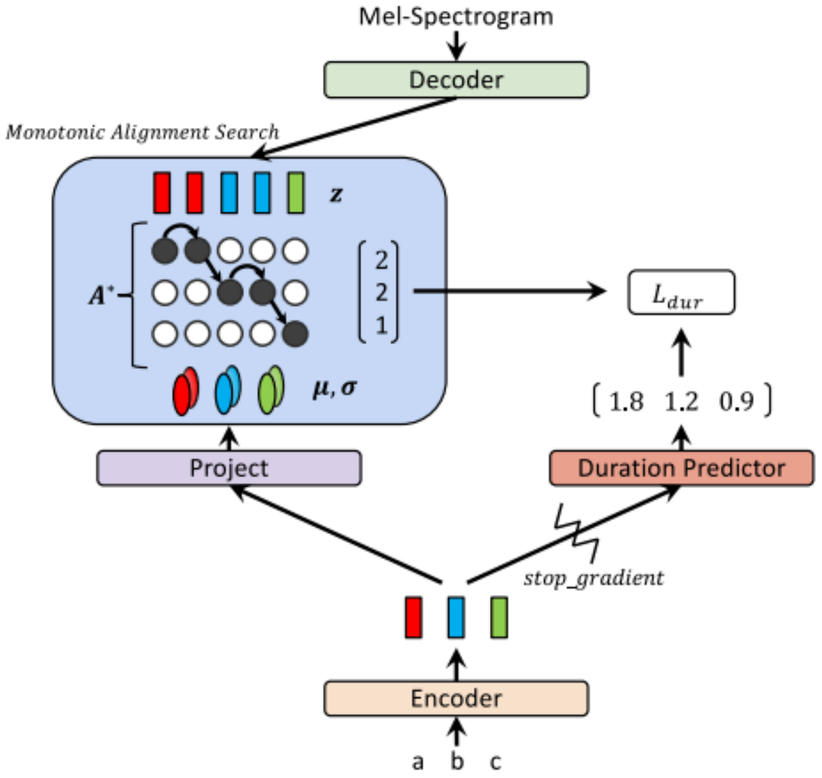
Where x is audio and c is text.

Text Encoder: models conditional prior $P_Z(z \mid c)$

μ_i and σ_i obtained from the f_{enc} modeling prior. P_Z is a multi-variate Gaussian i.e. each feature dim has it's own

Flow decoder: $f_{dec} : z \rightarrow x$

Models overall likelihood $P_X(x \mid c)$



(a) An abstract diagram of the training procedure.

Glow-TTS

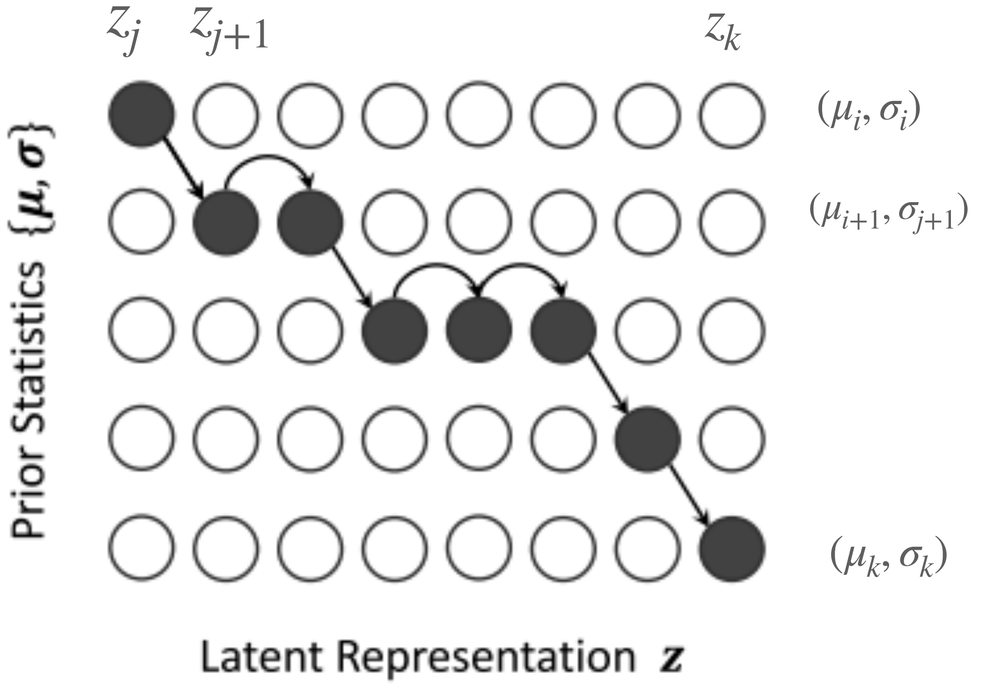
Monotonic Alignment Search

Is a dynamic programming algorithm (or alignment function A), which calculates the mapping between latent representation and statistics of Gaussian until (T_{text}, T_{mel})

$$Q_{i,j} = \max(Q_{i-1,j-1}, Q_{i,j-1}) + \log \mathcal{N}(z_j; \mu + i, \sigma_i)$$



Thus our system is parametrized by θ
(NN parameters)
And alignment function A so log likelihood can be
Defined as …



(a) An example of monotonic alignments

$$\log P_Z(z|c;\theta,A) = \sum_{j=1}^{T_{mel}} \log \mathcal{N}(z_j; \mu_{A(j)}, \sigma_{A(j)}),$$

Glow-TTS

Duration predictor



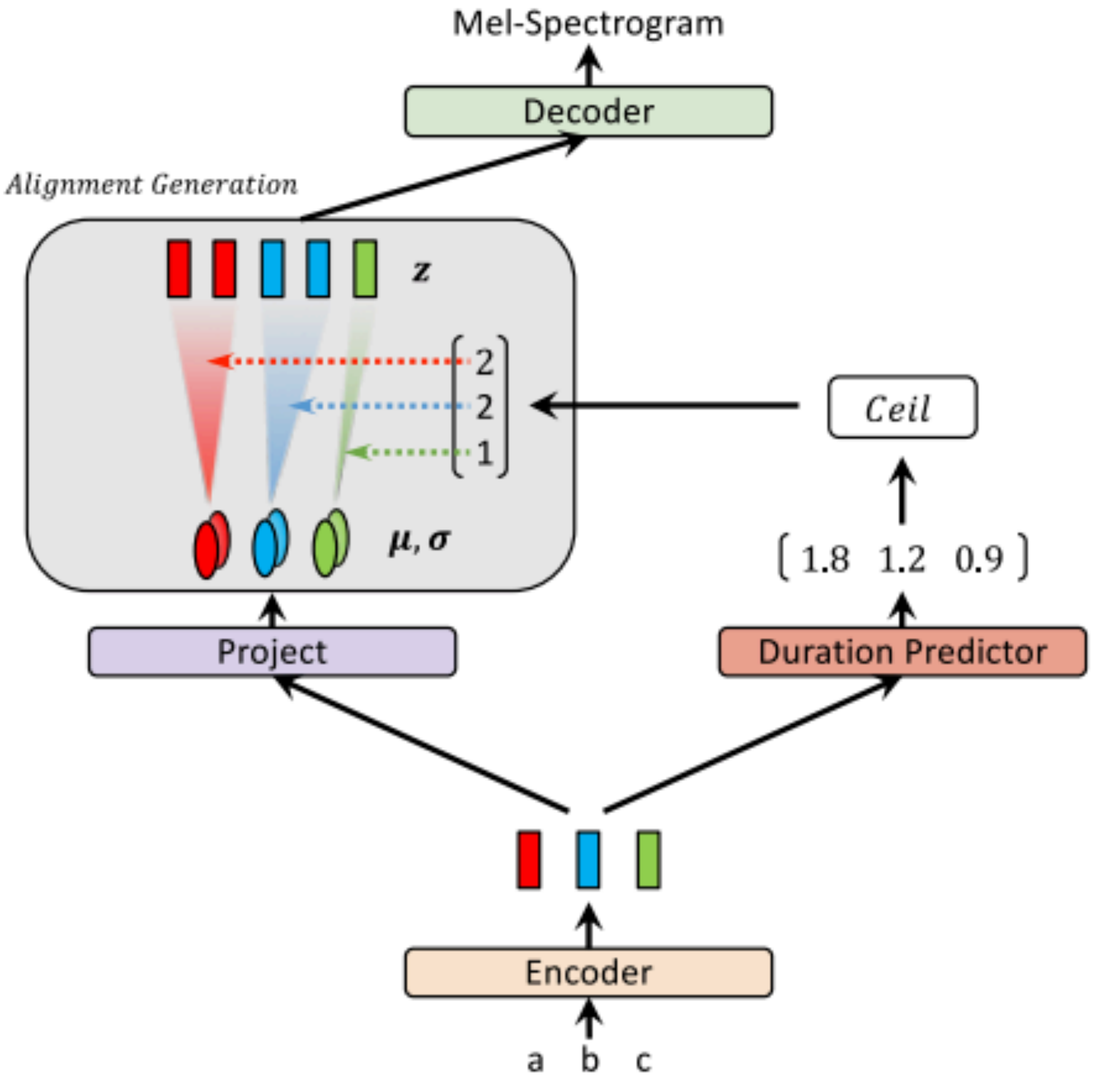
Why do we need duration predictor if we already have an internal aligner?

Alignment is supported by duration predictor which tries to match the aligner's output.

We compute the MSE between the aligner predicted length and the output of the f_{enc}.

$$d_i = \sum_{j=1}^{T_{mel}} 1_{A^*(j)=i}, i = 1, ..., T_{text}$$

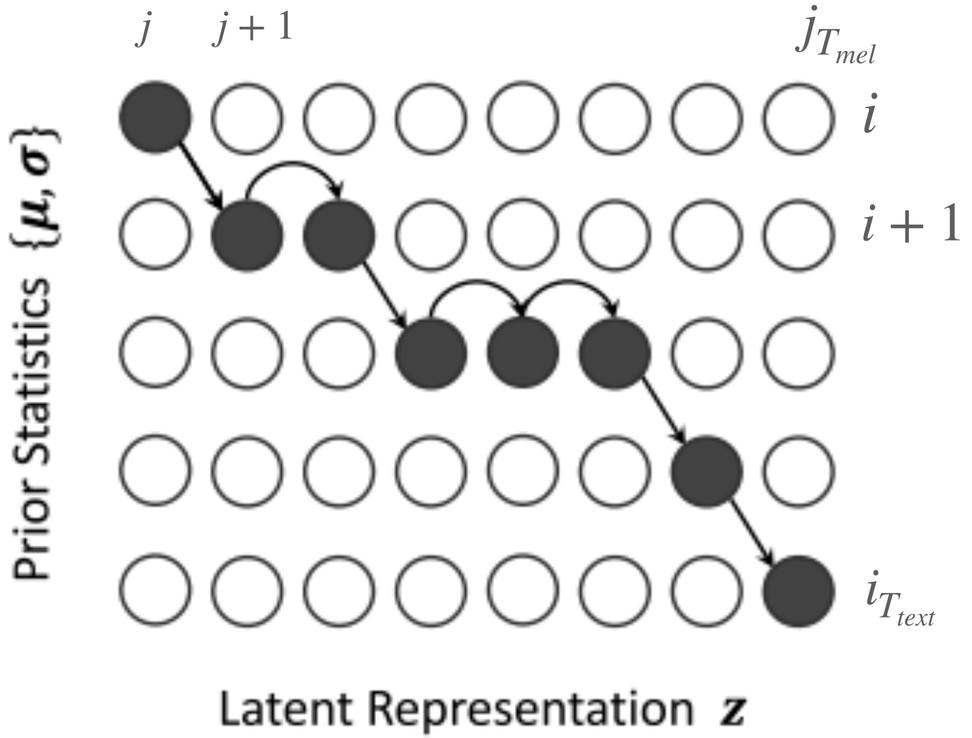
$$L_{dur} = MSE(f_{dur}(sg[f_{enc}(c)]), d)$$



Glow-TTS
Duration predictor

$$d_i = \sum_{j=1}^{T_{mel}} 1_{A^*(j)=i}, i = 1, ..., T_{text}$$

$$L_{dur} = MSE(f_{dur}(sg[f_{enc}(c)]), d)$$

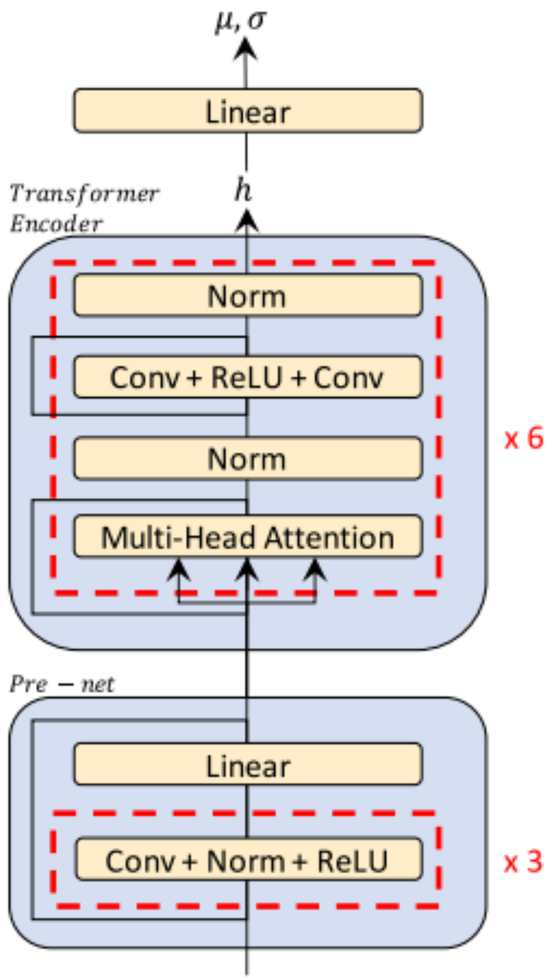


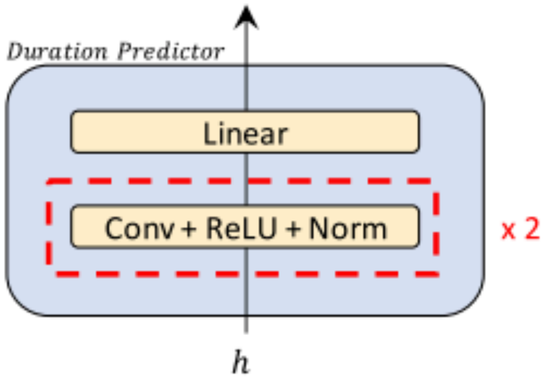
(a) An example of monotonic align-ments

Glow-TTS
Architecture (briefly)

Encoder: the architecture of encoder is fully convolutional with MHSA layers after pre-net. MHSA is used to capture global textual dependencies, each layer is followed by layer norm for stabilizing.

The output of the encoder is projected by linear layer.





Glow-TTS

Architecture (briefly)

Decoder:

The decoder is constructed from 1x1 convolutional layers (since it has to be invertible) followed by affine coupling layer.

Squeeze and unsqueeze operations are used to increase the number of channels in convolutions since the flow permutations happen across channels.

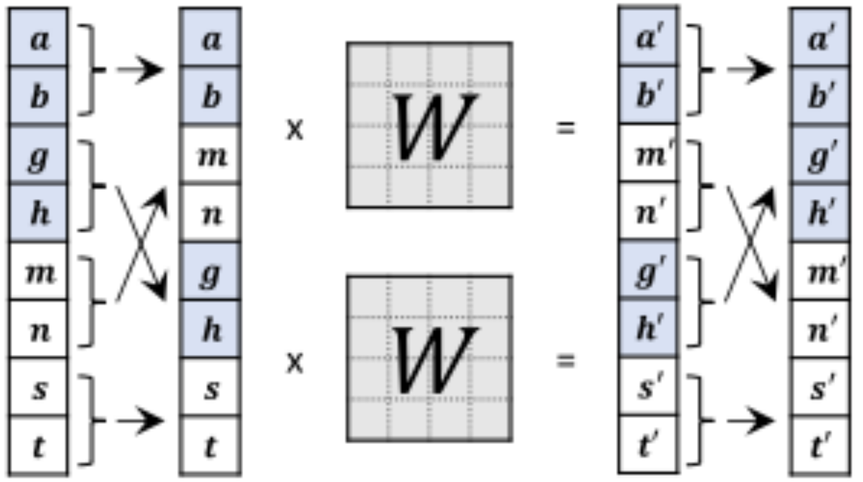
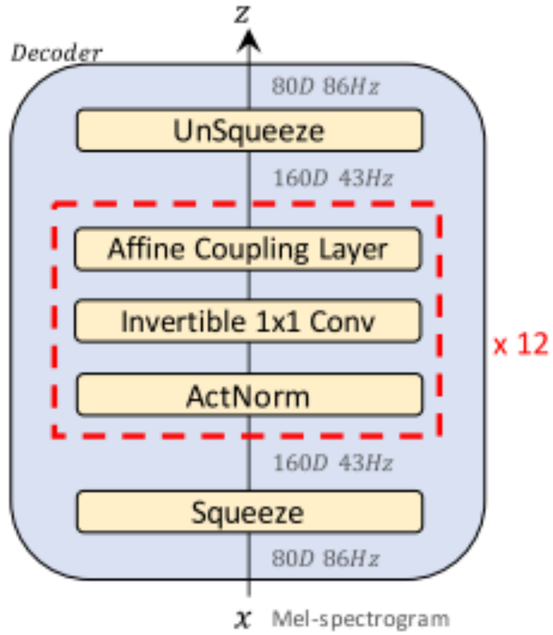
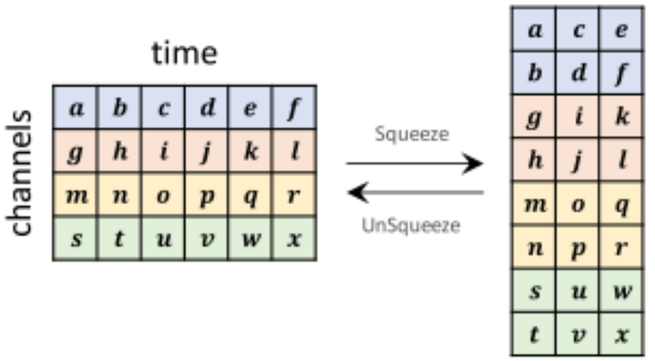


Illustration of 1x1 convolution. Invertible kernel is applied to each group separately within the flow partition.





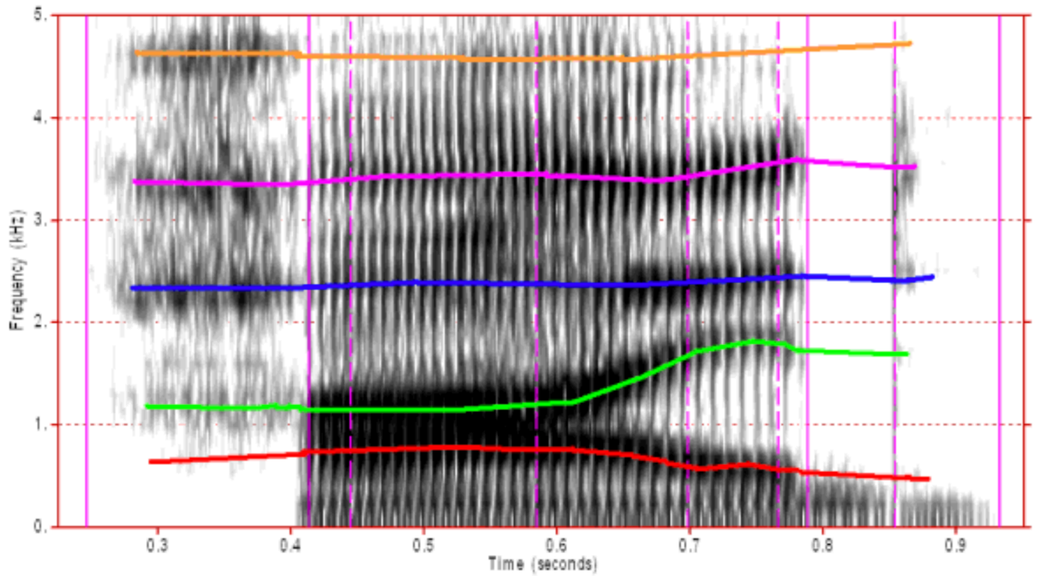
Glow-TTS

Experimental Results

Data:

- *LJSpeech* (for mono-speaker mode) 24h
- Subset of *LibriTTS* (for multi-speaker mode) 54h, 247 speakers.

See parameters and configs in the Appendix of the paper.



Img: Macquaire University Linguistics department

Evaluation

What are the characteristics we looking at while evaluating TTS system?

- Inference speed;
- Controllability of the speaker attributes (intonation, pitch for generating more diverse samples);
- Quality and naturalness of the speech;

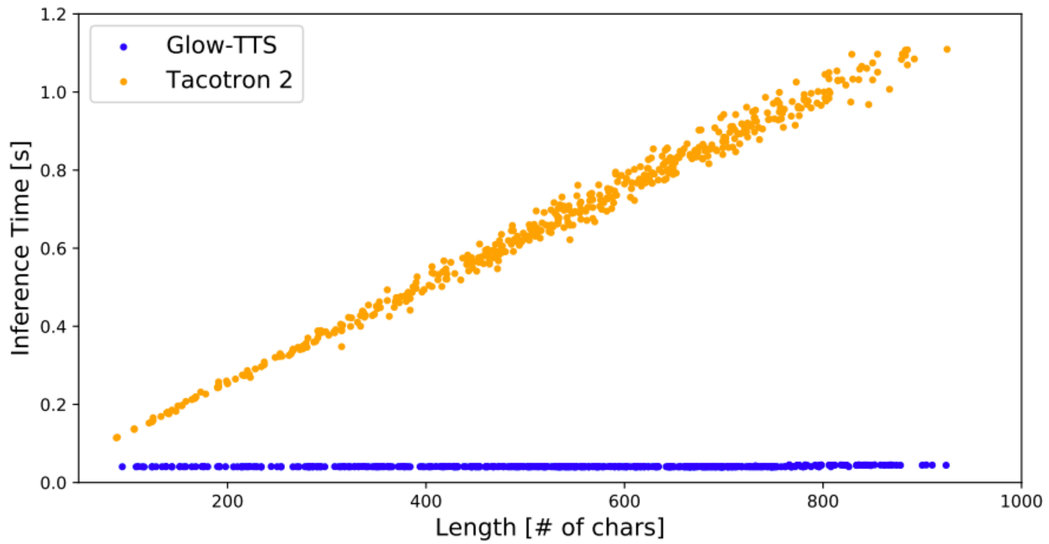
Mean Opinion Score (MOS) - a subjective measure to define the naturalness and overall speech quality: e.g. absence of reverberation, background noise, distortions.

Pitch Tracks (F0) help to see the changes in pitch

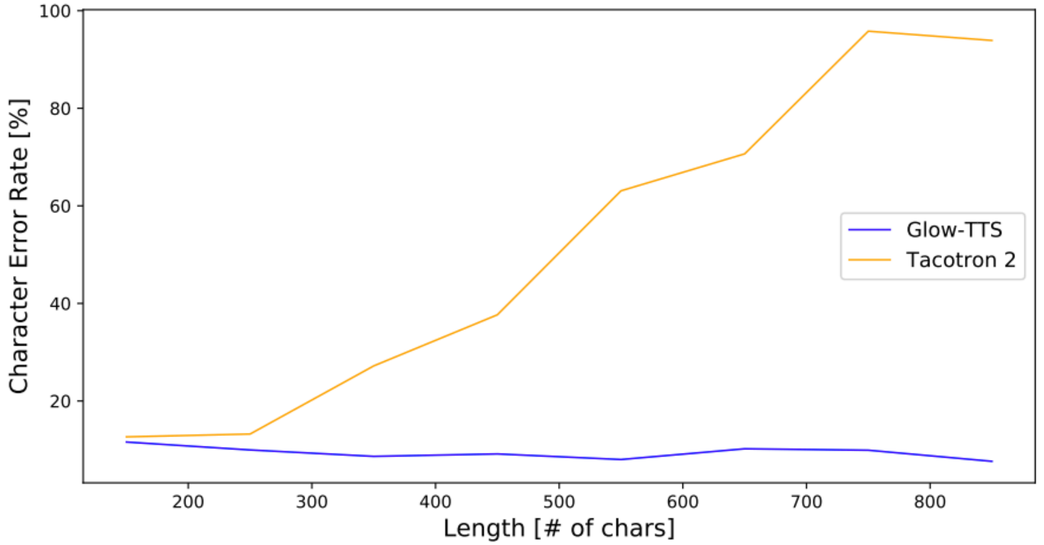
Glow-TTS

Experimental Results

Sampling Speed and Robustness



(a) The inference time comparison for Tacotron 2 and Glow-TTS (yellow: Tacotron2, blue: Glow-TTS).



(b) Robustness to the length of input utterances (yellow: Tacotron2, blue: Glow-TTS).

CER is calculated with Google Cloud STT interface

Glow-TTS

Experimental results

Diversity and Controllability

How to sample?

Reparametrization trick

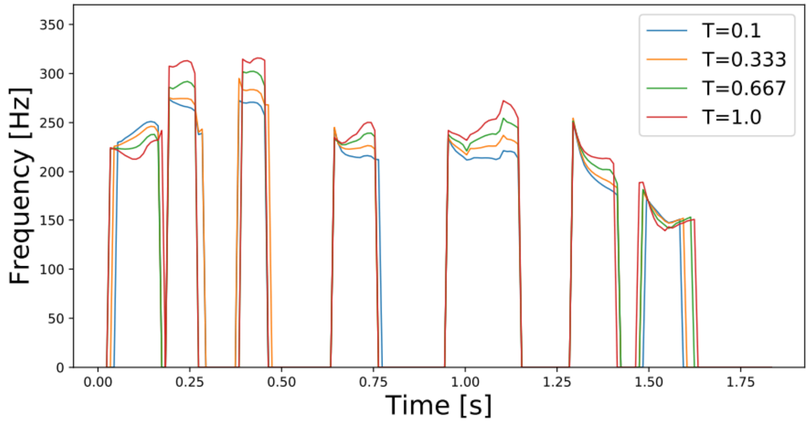
$$z = \mu + \epsilon * T$$

Where ϵ is a sample from Gaussian Normal, T is standard deviation (temperature). Since SD controls the width of the distribution, you can control pitch by varying T.

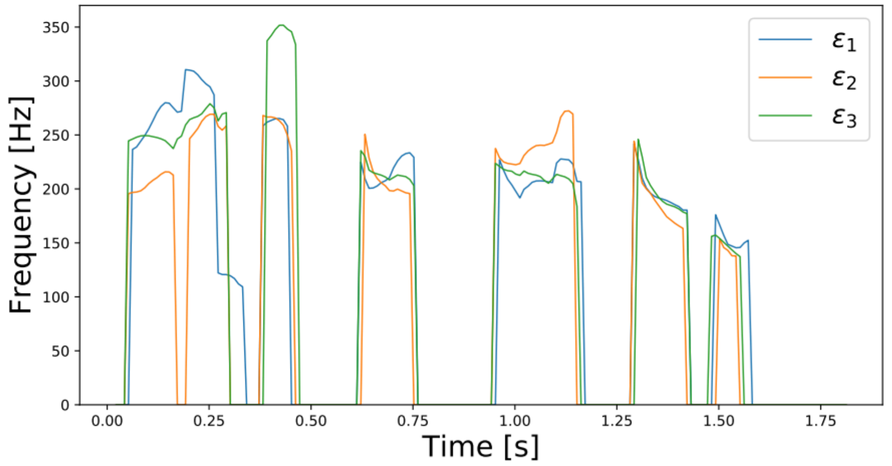
The higher T value the higher the pitch.

ϵ controls the diversity of intonation and stress patterns.

We can also control the speaking rate by scaling durations.



(b) Pitch tracks of the generated speech samples from the same sentence with different temperatures T and the same gaussian noise ϵ.



Glow-TTS

Experimental Results

A couple of boring MOS tables

Table 1: The Mean Opinion Score (MOS) of single speaker TTS models with 95% confidence intervals.

Method	9-scale MOS
GT	4.54 ± 0.06
GT (Mel + WaveGlow)	4.19 ± 0.07
Tacotron2 (Mel + WaveGlow)	3.88 ± 0.08
Glow-TTS ($T = 0.333$, Mel + WaveGlow)	4.01 ± 0.08
Glow-TTS ($T = 0.500$, Mel + WaveGlow)	3.96 ± 0.08
Glow-TTS ($T = 0.667$, Mel + WaveGlow)	3.97 ± 0.08

Single Speaker

Table 2: The Mean Opinion Score (MOS) of a multi-speaker TTS with 95% confidence intervals.

Method	9-scale MOS
GT	4.54 ± 0.07
GT (Mel + WaveGlow)	4.22 ± 0.07
Tacotron2 (Mel + WaveGlow)	3.35 ± 0.12
Glow-TTS ($T = 0.333$, Mel + WaveGlow)	3.20 ± 0.12
Glow-TTS ($T = 0.500$, Mel + WaveGlow)	3.31 ± 0.12
Glow-TTS ($T = 0.667$, Mel + WaveGlow)	3.45 ± 0.11

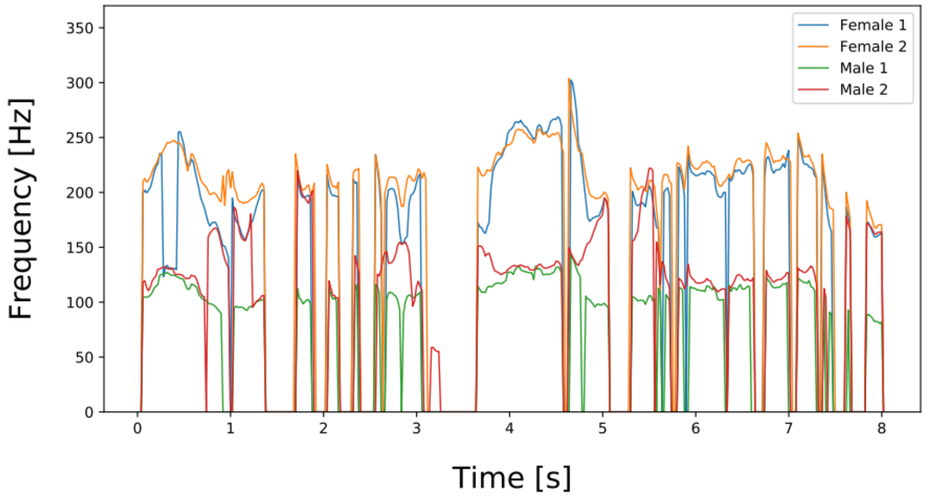
Multi-Speaker

Model Conditioning

How can we make TTS system talk different voices?

We can condition the decoder on speaker ID. There is a variety of models for learning speaker-dependent representations, containing characteristics of the particular speaker (e.g. x-vectors).

We concatenate each time-step of the decoder's input with the speaker vector.



(b) Pitch tracks of the voice conversion samples with different speaker identities.

Glow-TTS samples: https://jaywalnut310.github.io/glow-tts-demo/index.html

What did we learn?

- Normalizing flows and how to learn the exact transition between the distributions using normalizing flows;
- How to sample from NFs
- Why NFs are used in TTS and why they are useful?
- A little review of probability theory.



References

Normalizing flows

1. *M. Brubaker* full tutorial at ECCV 2020: https://www.youtube.com/watch?v=u3vVyFVU_lI&t=2302s

2. *Kobyzev, Ivan, Simon JD Prince, and Marcus A. Brubaker.* "Normalizing flows: An introduction and review of current methods." *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020): 3964-3979.

3. *Kingma, Durk P., and Prafulla Dhariwal.* "Glow: Generative flow with invertible 1x1 convolutions." *Advances in neural information processing systems* 31 (2018).

TTS

1. *Kim, Jaehyeon, et al.* "Glow-TTS: A generative flow for text-to-speech via monotonic alignment search." *Advances in Neural Information Processing Systems* 33 (2020): 8067-8077.

2. *Shen, Jonathan, et al.* "Natural TTS synthesis by conditioning WaveNet on Mel Spectrogram predictions." *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018.

Thank you!