VoiceBox: Text-Guided Multilingual Universal Speech Generation at Scale

Application of Flow Matching for Speech

Presenter: Desh Raj

September 20, 2023

FLOW MATCHING FOR GENERATIVE MODELING

Yaron Lipman[1,2] Ricky T. Q. Chen[1] Heli Ben-Hamu[2] Maximilian Nickel[1] Matt Le[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science

Feb ’23: FAIR Labs folks develop new generative modeling technique

June ’23: FAIR Accel + some Meta AI folks apply it for speech generation

Voicebox: Text-Guided Multilingual Universal Speech Generation at Scale

Matthew Le* Apoorv Vyas* Bowen Shi* Brian Karrer* Leda Sari Rashel Moritz
Mary Williamson Vimal Manohar Yossi Adi† Jay Mahadeokar Wei-Ning Hsu*

Meta AI

https://ai.meta.com/blog/voicebox-generative-ai-model-speech/

Outline

1. Generative Models and Normalizing Flows
2. Continuous Normalizing Flows
3. Flow Matching
4. **VoiceBox**

Generative Models and Normalizing Flows

Resources:

1. CVPR 2021 tutorial: https://mbrubake.github.io/cvpr2021-nf_in_cv-tutorial/
2. Lilian Weng. *Flow-based deep generative models*. https://lilianweng.github.io/posts/2018-10-13-flow-models

Generative models

Introduction

- A generative model is a probability distribution over random variable $\mathbf{X}$ which we attempt to learn from a set of observed data $\{\mathbf{x}_i\}_{i=1}^{N}$ with some probability density $p_{\mathbf{X}}(x)$ parameterized by θ.
- What do we want from generative models?
  - Evaluating $p_{\mathbf{X}}(x)$ for some x
  - Sampling from $p_{\mathbf{X}}(x)$
  - $p_{\mathbf{X}}(x)$ can be complex data distribution

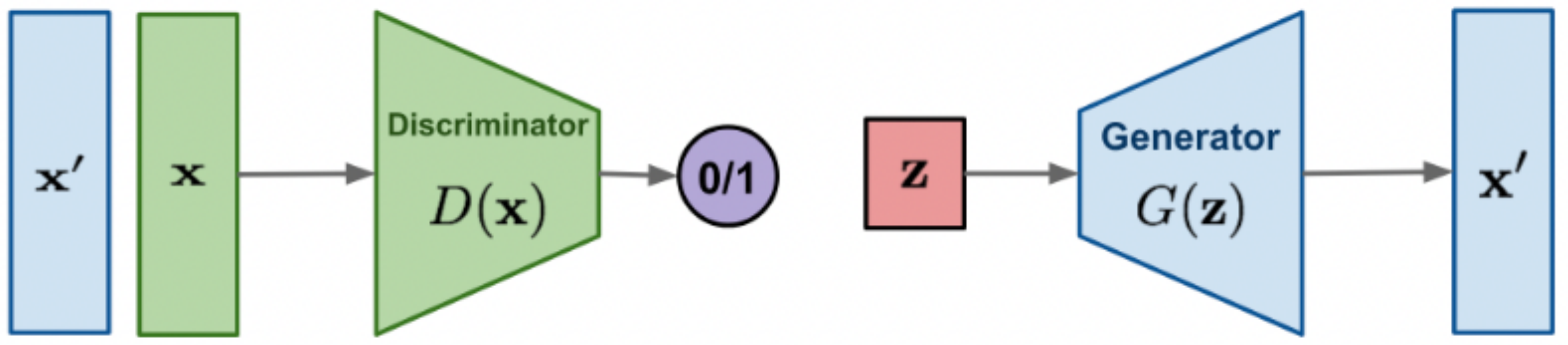
Generative models

Gaussian mixture models (GMMs)

- Trained either via maximum likelihood (ML) or a variational bound on likelihood
- What do we want from generative models?
  - Evaluating $p_{\mathbf{X}}(x)$ for some x ✓
  - Sampling from $p_{\mathbf{X}}(x)$ ✓
  - $p_{\mathbf{X}}(x)$ can be complex data distribution ✗

Generative models

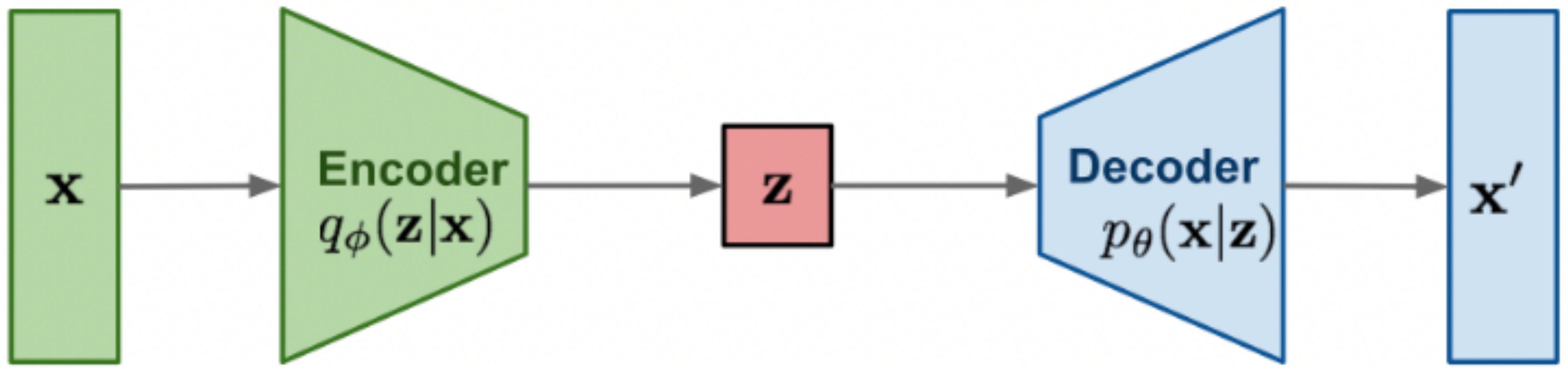
Generative adversarial networks (GANs)



- Trained through an adversarial process (playing a minimax game) –> turns an unsupervised problem into a supervised one.
- What do we want from generative models?
  - Evaluating $p_{\mathbf{X}}(x)$ for some x ✗
  - Sampling from $p_{\mathbf{X}}(x)$ ✓
  - $p_{\mathbf{X}}(x)$ can be complex data distribution ✓

Generative models

Variational auto-encoders (VAEs)



- Trained with a bound on maximum likelihood (ELBO)
- What do we want from generative models?
  - Evaluating $p_{\mathbf{X}}(x)$ for some x
  - Sampling from $p_{\mathbf{X}}(x)$ ✓
  - $p_{\mathbf{X}}(x)$ can be complex data distribution

Preliminary

Jacobian matrix

- $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a function. Then the Jacobian of f is the matrix of partial derivatives.

$$\mathbf{J} = Df(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

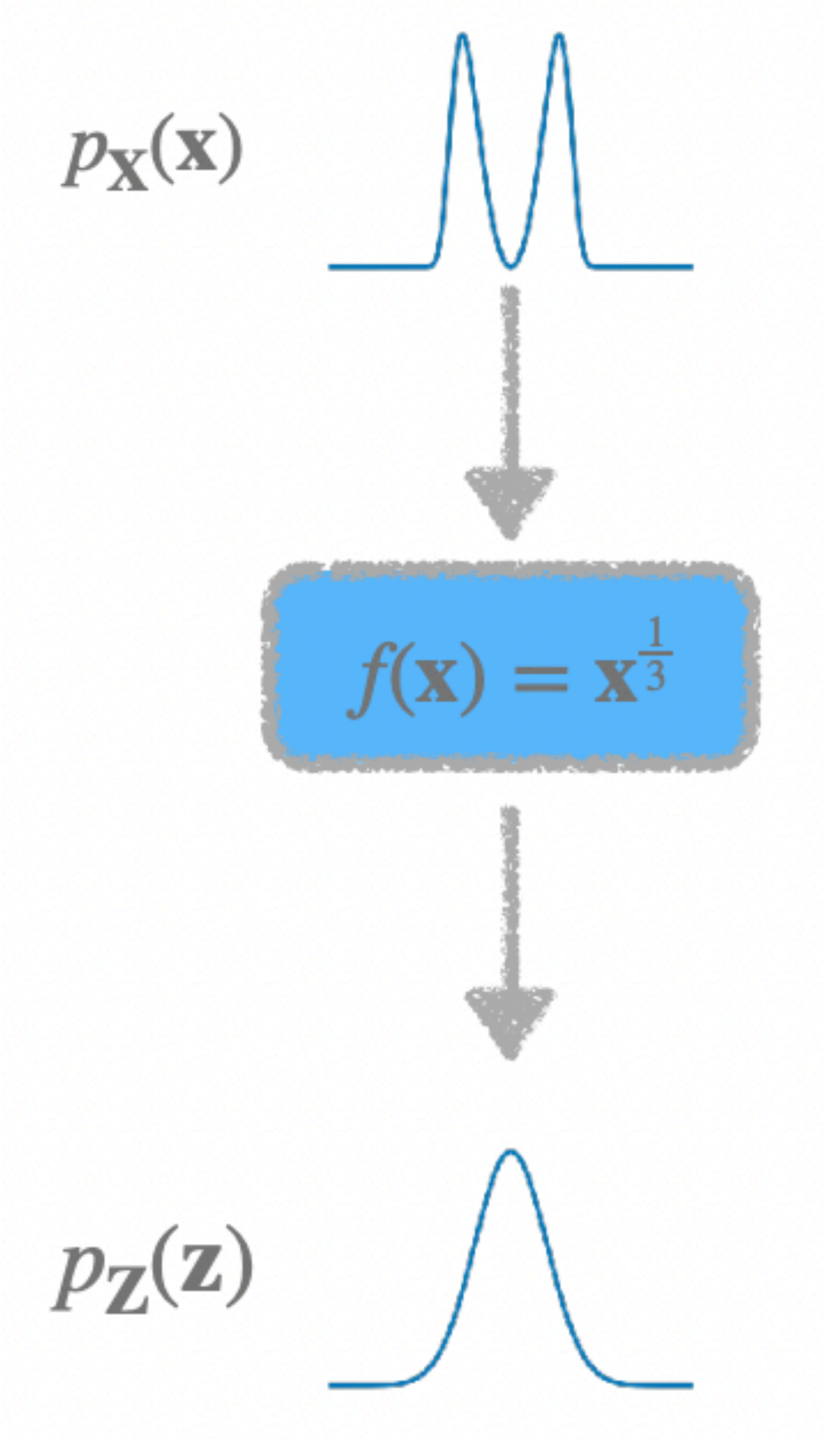
Preliminary

Change of variables

- Let $p_{\mathbf{X}}(x)$ and $p_{\mathbf{Z}}(z)$ be two distributions over random variables $\mathbf{X}$ and $\mathbf{Z}$.

- $\mathbf{Z} = f(\mathbf{X})$ is an invertible, differentiable function.

$$p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(f(x))\,|\det\, Df(x)\,|$$



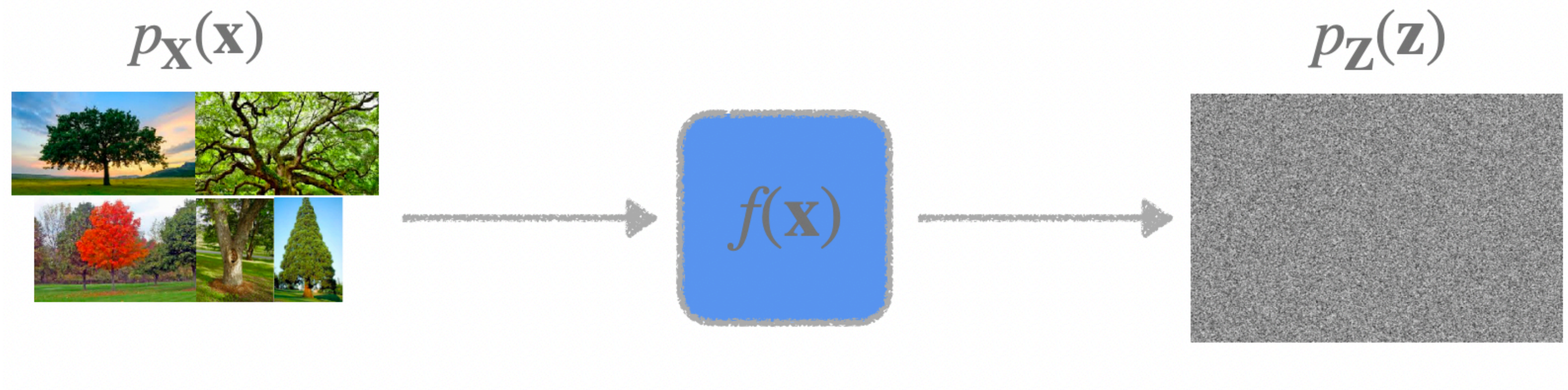
Cool visualization: https://www.youtube.com/watch?v=hhFzJvaY__U

Preliminary

Change of variables

- Let $p_{\mathbf{X}}(x)$ and $p_{\mathbf{Z}}(z)$ be two distributions over random variables $\mathbf{X}$ and $\mathbf{Z}$.
- $\mathbf{Z} = f(\mathbf{X})$ is an invertible, differentiable function.

$$p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(f(x))\,|\det Df(x)|$$



Normalizing flows

Flow function

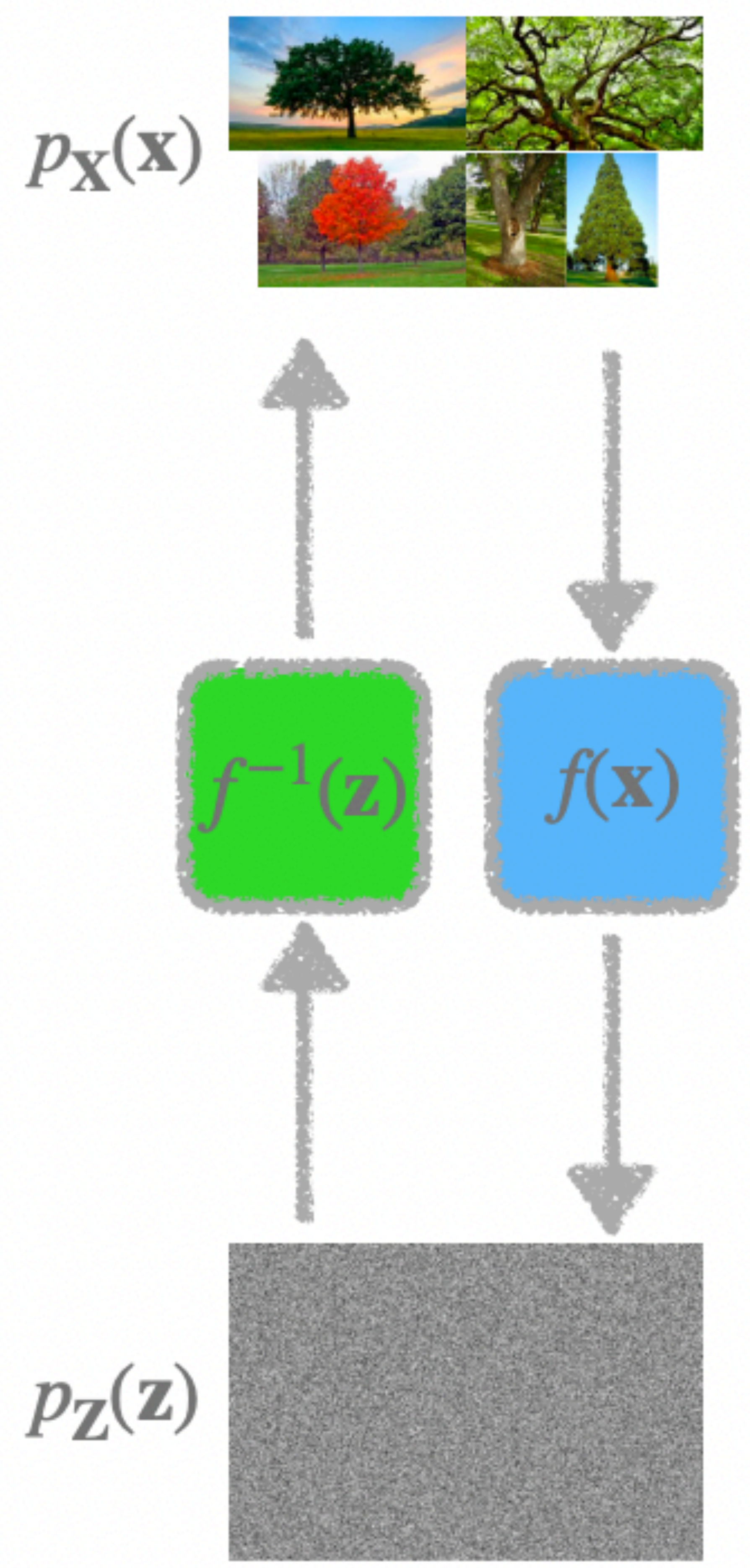
- Learn $f(\mathbf{X})$ to transform $p_{\mathbf{X}}(x)$ into $p_{\mathbf{Z}}(z)$.
- Two components:
  - Base measure: $p_{\mathbf{Z}}(z)$, usually selected as $\boxed{\mathcal{N}(0,\mathbf{I})}$
  - Flow: $f(\mathbf{X})$, must be invertible and differentiable

"Normalizing" flow

Normalizing flows

Does it satisfy what we want?

- What do we want from generative models?
  - Evaluating $p_{\mathbf{X}}(x)$ for some x ✔ $p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(f(x))\,|\det\, Df(x)\,|$
  - Sampling from $p_{\mathbf{X}}(x)$ ✔ Sample $\mathbf{z} \sim p_{\mathbf{Z}}(\,\cdot\,)$, then compute $\mathbf{x} = f^{-1}(\mathbf{z})$
  - $p_{\mathbf{X}}(x)$ can be complex data distribution ?



Normalizing flows

Training

- Maximum likelihood (θ are parameters of the flow function)

$$\max_{\theta} \sum_{i=1}^{N} \log p_{\mathbf{X}}(x) = \max_{\theta} \sum_{i=1}^{N} \log p_{\mathbf{Z}}(f(\mathbf{x}_i \,|\, \theta)) + \log |\det\, Df(\mathbf{x}_i \,|\, \theta)|$$

Normalizing flows

What are flows?

- A flow is a parametric function $f(\mathbf{x})$ which:
  - Is invertible
  - Is differentiable
  - Has an efficiently computable inverse and Jacobian determinant $|\det Df(\mathbf{x})|$

Normalizing flows

Composition of flows

- Invertible, differentiable functions are closed under composition.
- Build up a complex flow from a composition of simple flows.

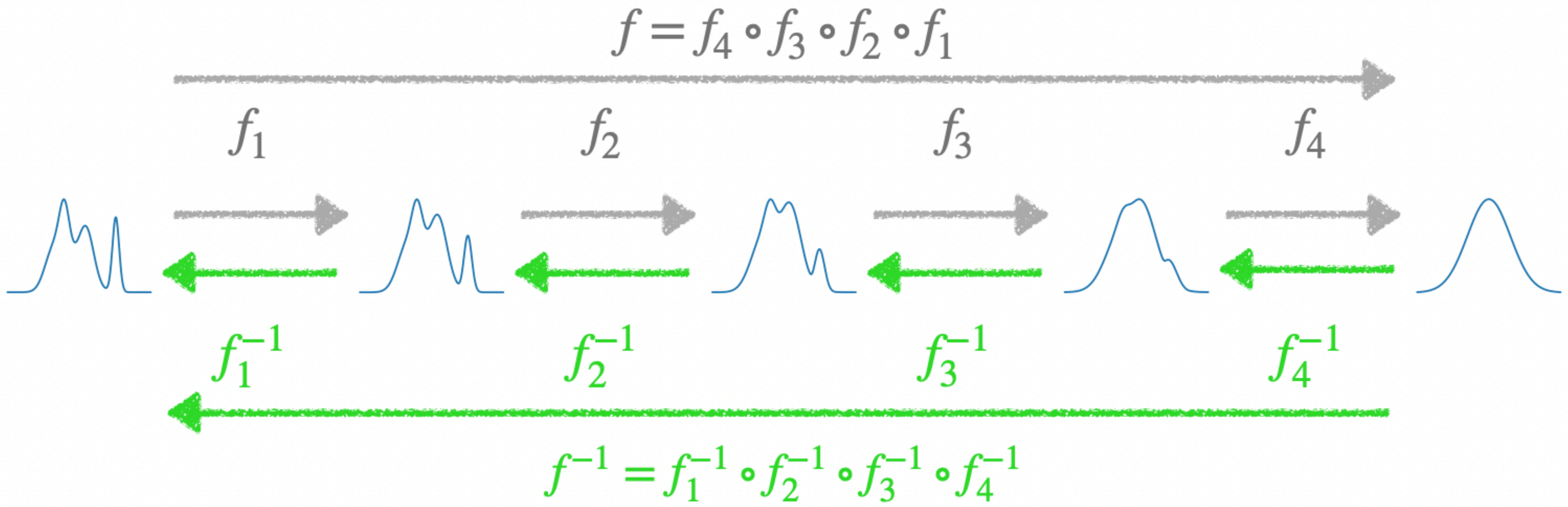
$$f = f_K \circ f_{K-1} \circ \ldots \circ f_2 \circ f_1$$

- Determinant computation is simple, because

$$\det\, Df = \det\, \prod_{k=1}^{K} Df_k = \prod_{k=1}^{K} \det\, Df_k$$

Normalizing flows

Composition of flows



Normalizing flows

Reverse flows

- Can also think about flowing from normal distribution to data distribution.

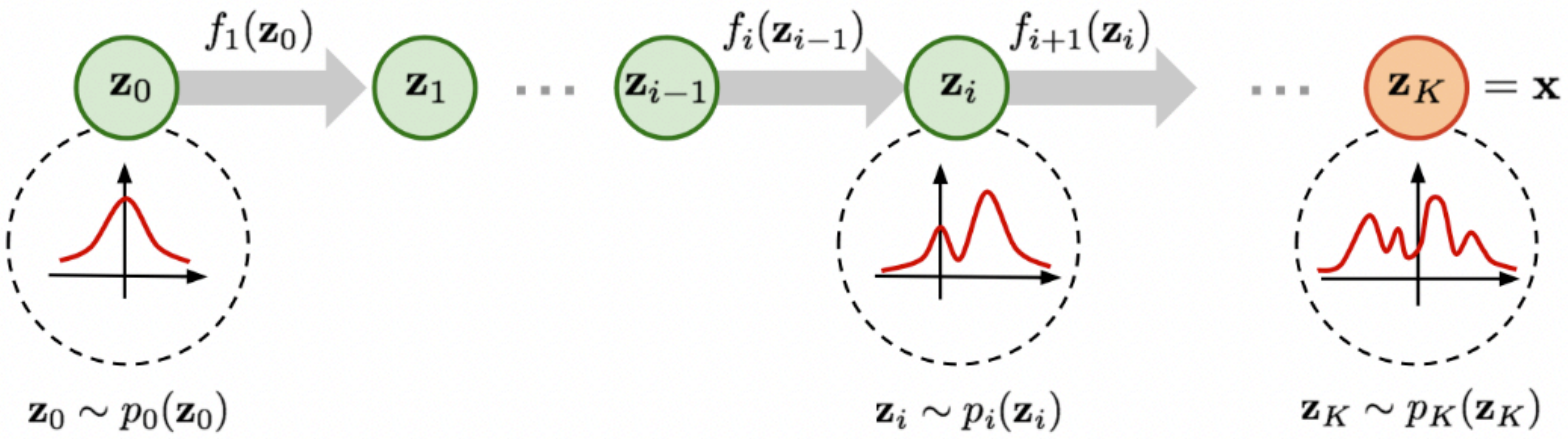


Fig. 2. Illustration of a normalizing flow model, transforming a simple distribution $p\_0(\mathbf{z}\_0)$ to a complex one $p\_K(\mathbf{z}\_K)$ step by step.

https://lilianweng.github.io/posts/2018-10-13-flow-models/normalizing-flow.png

Normalizing flows

Reverse flows

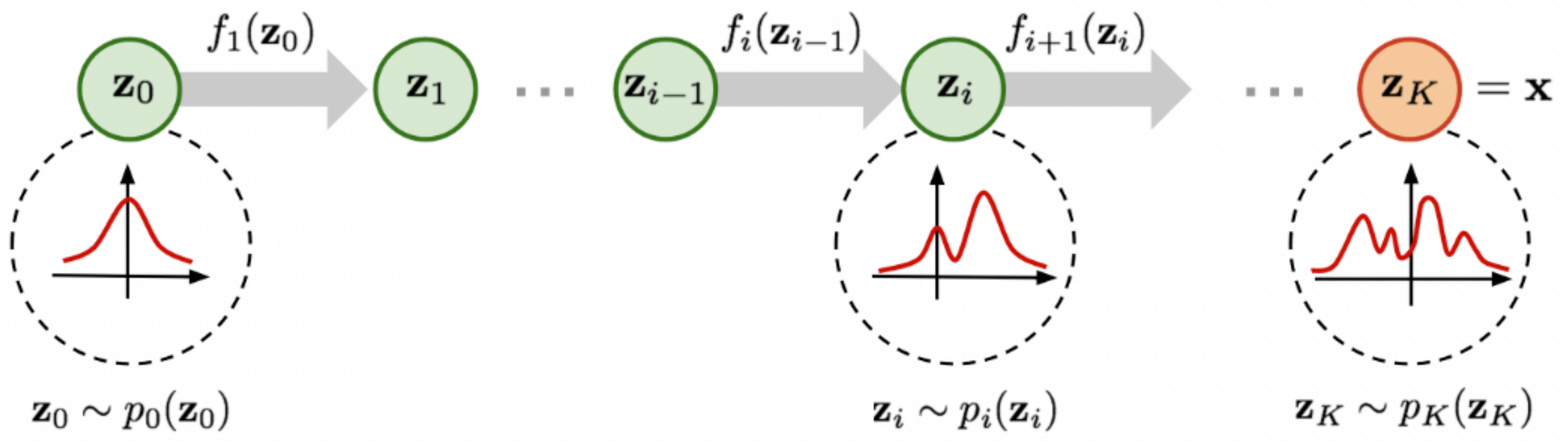


Fig. 2. Illustration of a normalizing flow model, transforming a simple distribution $p\_0(\mathbf{z}\_0)$ to a complex one $p\_K(\mathbf{z}\_K)$ step by step.

$$
\begin{aligned}
\log p_{\mathbf{X}}(\mathbf{x}) = \log p_K(\mathbf{z}_K) &= \log p_{K-1}(\mathbf{z}_{K-1}) - \log|\det Df_K| \\
&= \log p_{K-2}(\mathbf{z}_{K-2}) - \log|\det Df_{K-1}| - \log|\det Df_K| \\
&= \ldots \\
&= \log p_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log|\det Df_k|
\end{aligned}
$$

Normalizing flows

Limitations

- A flow is a parametric function $f(\mathbf{x})$ which:
  - Is invertible
  - Is differentiable
  - Has an efficiently computable inverse and Jacobian determinant $|\det Df(\mathbf{x})|$
- **It is hard to design flow functions with these constraints!**

Continuous Normalizing Flows

Resources:

1. Chen, T.Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D.K. (2018). **Neural Ordinary Differential Equations**. *Neural Information Processing Systems*.
2. https://slideslive.com/38917901/neural-ordinary-differential-equations-for-continuous-normalizing-flows

Continuous Normalizing flows

From discrete to continuous time steps

Discrete flow	Continuous flow
$z_0 \sim p(z)$	$z_0 \sim p(z)$
$z_t = F_t(z_{t-1};\theta)$	$\dfrac{dz}{dt} = f(z_t, t, \theta)$
$x = z_T = F_T \circ \ldots \circ F_1(z_0)$	$x = z_T = z_0 + \int_0^T f(z_t, t, \theta)dt$

Parameterize "instantaneous" change in state

Numerical solvers to solve this equation

Continuous Normalizing flows

Invertibility

Discrete flow	Continuous flow
$z_0 \sim p(z)$	$z_0 \sim p(z)$
$z_t = F_t(z_{t-1}; \theta)$	$\dfrac{dz}{dt} = f(z_t, t, \theta)$
$x = z_T = F_T \circ \ldots \circ F_1(z_0)$	$x = z_T = z_0 + \int_0^T f(z_t, t, \theta)dt$
The functional form of F_t needs to be invertible.	No constraints on the functional form of f.

Continuous Normalizing flows

Change of variables

Discrete flow	Continuous flow
$z_0 \sim p(z)$	$z_0 \sim p(z)$
$z_t = F_t(z_{t-1}; \theta)$	$\dfrac{dz}{dt} = f(z_t, t, \theta)$
$x = z_T = F_T \circ \ldots \circ F_1(z_0)$	$x = z_T = z_0 + \int_0^T f(z_t, t, \theta) dt$

$$\log p(x) = \log p(z_T) = \log p(z_0) - \sum_{t=1}^{T} \log \left| \frac{\partial F_t}{\partial z_t} \right|$$

$$\log p(x) = \log p(z_T) = \log p(z_0) - \int_0^T \mathrm{Tr} \left[\frac{\partial f}{\partial z_t} \right] dt$$

Jacobian **determinant**: $\mathcal{O}(N^3)$

Trace: $\mathcal{O}(N)$

Continuous Normalizing flows

Training

- Trained using maximum likelihood, i.e., maximize $\sum_{i=1}^{N} \log p(\mathbf{x}_1 \,|\, \theta)$

$$\log p(x) = \log p(z_T) = \log p(z_0) - \int_0^T \mathrm{Tr}\left[\frac{\partial f}{\partial z_t}\right] dt$$

Continuous Normalizing flows

Limitations

- Training is expensive, since we need to use **ODE solvers** at every step!
- Computing the trace is linear if Jacobian is known, but **quadratic** otherwise:
  - Usually replaced with a “stochastic estimator” of the trace

$$\log p(x) = \log p(z_T) = \log p(z_0) - \int_0^T \mathrm{Tr}\left[\frac{\partial f}{\partial z_t}\right] dt$$

Flow Matching for Generative Modeling

Resources:

1. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., & Le, M. (2022). **Flow Matching for Generative Modeling**. *ICLR 2023 (spotlight paper)*.
2. Alex Tong, et al. Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport. https://www.youtube.com/watch?v=UhDtH7Ia9Ag&t=514s

What is Flow Matching?

Training objective for continuous normalizing flows

- Suppose $p_t(x)$ is the **target probability path** (indexed by time-step t) –> think of the "path" of the probability distribution going from normal to data distribution

- We assume that there is some **vector field** $u_t(x)$ which gives rise to this probability path.

- **Flow matching** tries to directly learn this vector field, i.e.,

$$\mathscr{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\substack{t \sim U(0,1) \\ x \sim p_t(x)}} \| v_\theta(t, x) - u_t(x) \|^2$$

What is Flow Matching?

Training objective for continuous normalizing flows

- **Flow matching** tries to directly learn this vector field, i.e.,

$$\mathscr{L}_{\text{FM}}(\theta) = \mathbb{E}_{\substack{t \sim U(0,1) \\ x \sim p_t(x)}} \| v_\theta(t, x) - u_t(x) \|^2$$

- Problem: $p_t(x)$ and $u_t(x)$ are unknown!

Conditional Flow Matching

Solution to the problem

- Replace the “marginal” target probability and vector field with conditional, where we condition on the given data:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\substack{t \sim U(0,1) \\ x_1 \sim q(x_1) \\ x \sim p_t(x \mid x_1)}} \|v_\theta(t, x) - u_t(x \mid x_1)\|^2$$

- The gradients of this loss w.r.t. θ are same as the gradient of original loss.

- Now we only need to define $q(x_1)$, $p_t(x \mid x_1)$, and $u_t(x \mid x_1)$.

Conditional Flow Matching

Defining the conditional probabilities

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\substack{t \sim U(0,1) \\ x_1 \sim q(x_1) \\ x \sim p_t(x|x_1)}} \| v_\theta(t, x) - u_t(x|x_1) \|^2$$

- $q(x_1)$ is set as the uniform distribution over the training data.
- We want a conditional vector field that flows from standard Normal distribution to a Gaussian distribution centered at x_1 with std σ

Conditional Flow Matching

Defining the conditional probabilities

- We want a conditional vector field that flows from standard Normal distribution to a Gaussian distribution centered at x_1 with std σ

$$p_t(x \mid x_1) = \mathcal{N}(x \mid t x_1, (t\sigma - t + 1)^2)$$

$$u_t(x \mid x_1) = \frac{x_1 - (1-\sigma)x}{1 - (1-\sigma)t}$$

Conditional Flow Matching

Training loop

Algorithm 1 Conditional Flow Matching

Input: Efficiently samplable $q(z)$, $p_t(x|z)$, and computable $u_t(x|z)$ and initial network v_θ.

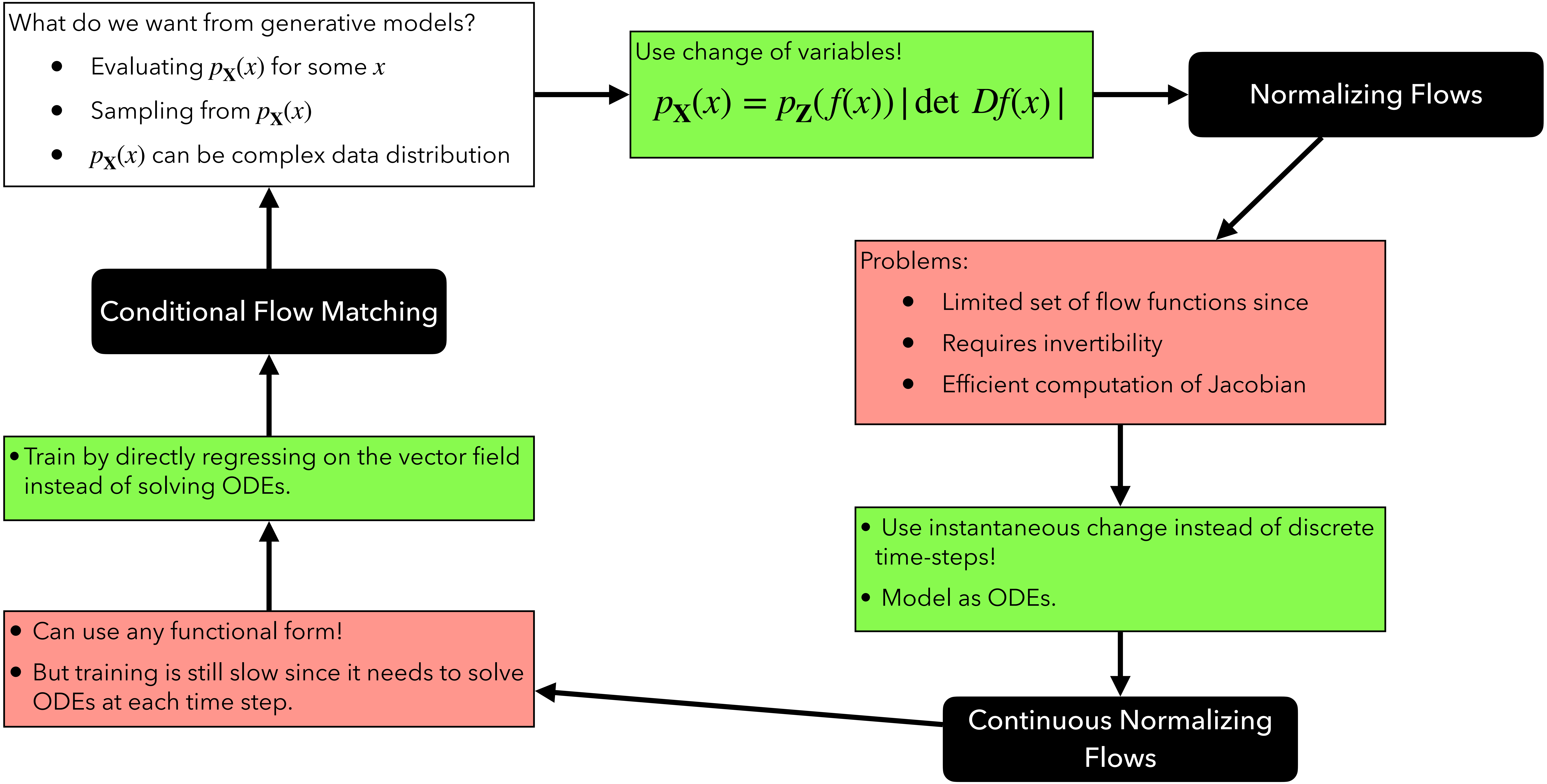
while Training **do**

$z \sim q(z); \quad t \sim \mathcal{U}(0,1); \quad x \sim p_t(x|z)$

$\mathcal{L}_{\text{CFM}}(\theta) \leftarrow \|v_\theta(t,x) - u_t(x|z)\|^2$

$\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta))$

return v_θ


What do we want from generative models?
• Evaluating $p_{\mathbf{X}}(x)$ for some x
• Sampling from $p_{\mathbf{X}}(x)$
• $p_{\mathbf{X}}(x)$ can be complex data distribution
Use change of variables!
$p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(f(x))\,|\det\, Df(x)\,|$
Normalizing Flows
Problems:
• Limited set of flow functions since
• Requires invertibility
• Efficient computation of Jacobian
• Use instantaneous change instead of discrete time-steps!
• Model as ODEs.
Continuous Normalizing Flows
• Can use any functional form!
• But training is still slow since it needs to solve ODEs at each time step.
• Train by directly regressing on the vector field instead of solving ODEs.
Conditional Flow Matching

VoiceBox

Resources:

1. Le, M., Vyas, A., Shi, B., Karrer, B., Sari, L., Moritz, R., Williamson, M., Manohar, V., Adi, Y., Mahadeokar, J., & Hsu, W. (2023). **Voicebox: Text-Guided Multilingual Universal Speech Generation at Scale**. *ArXiv, abs/2306.15687*.

Methodology

Task

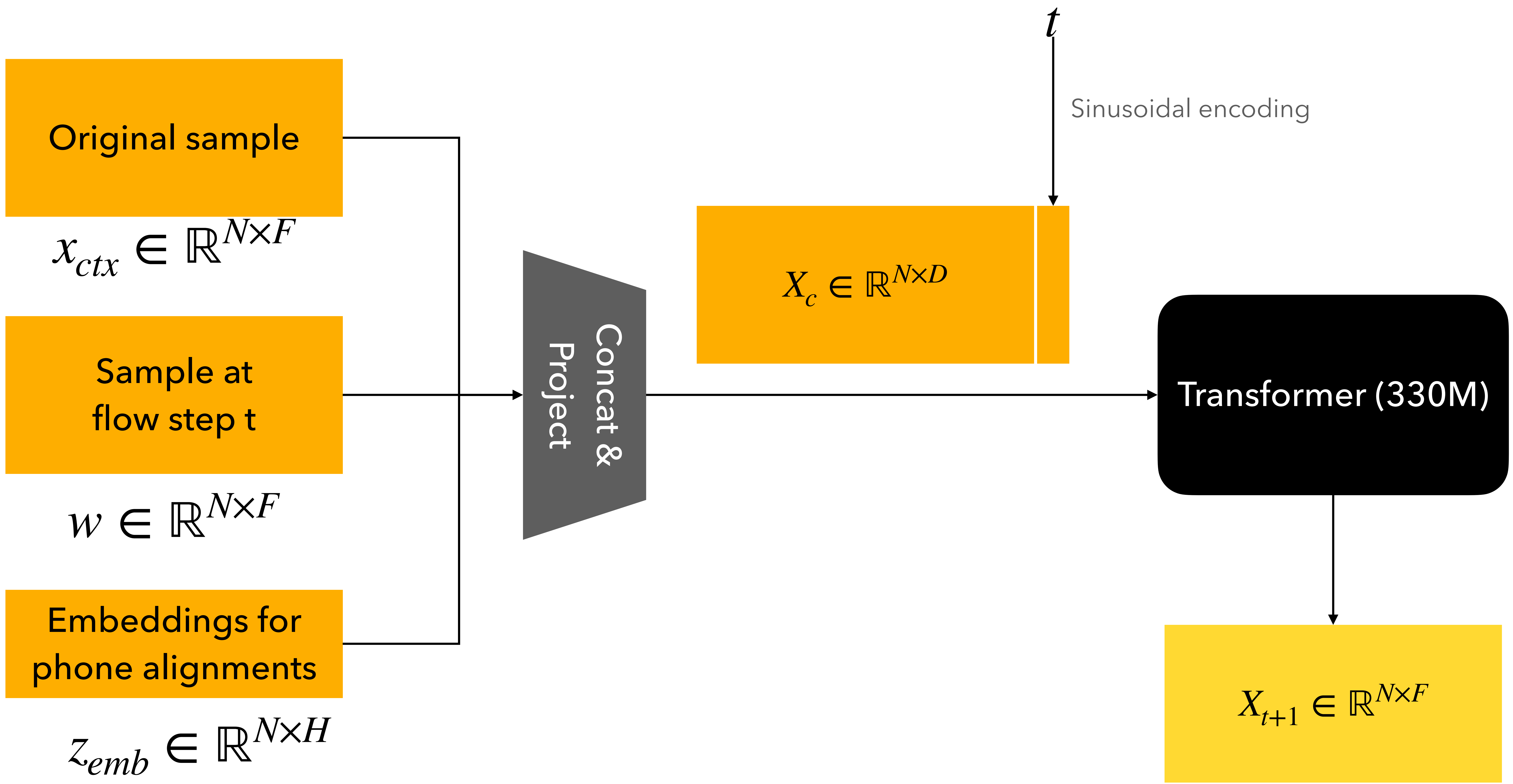
- Task: text-guided speech infilling
- Predict masked segment of speech based on surrounding audio context and complete text transcript
- Text transcript is provided as frame-level phone alignments

Methodology
Model

- Model is divided into 2 components:
  - Audio model
  - Duration model
- **Audio model** is a CNF trained on 80-dim log Mel spectrogram extracted at 100Hz frame rate.
- **Duration model** is a simple regression trained with L1 loss.

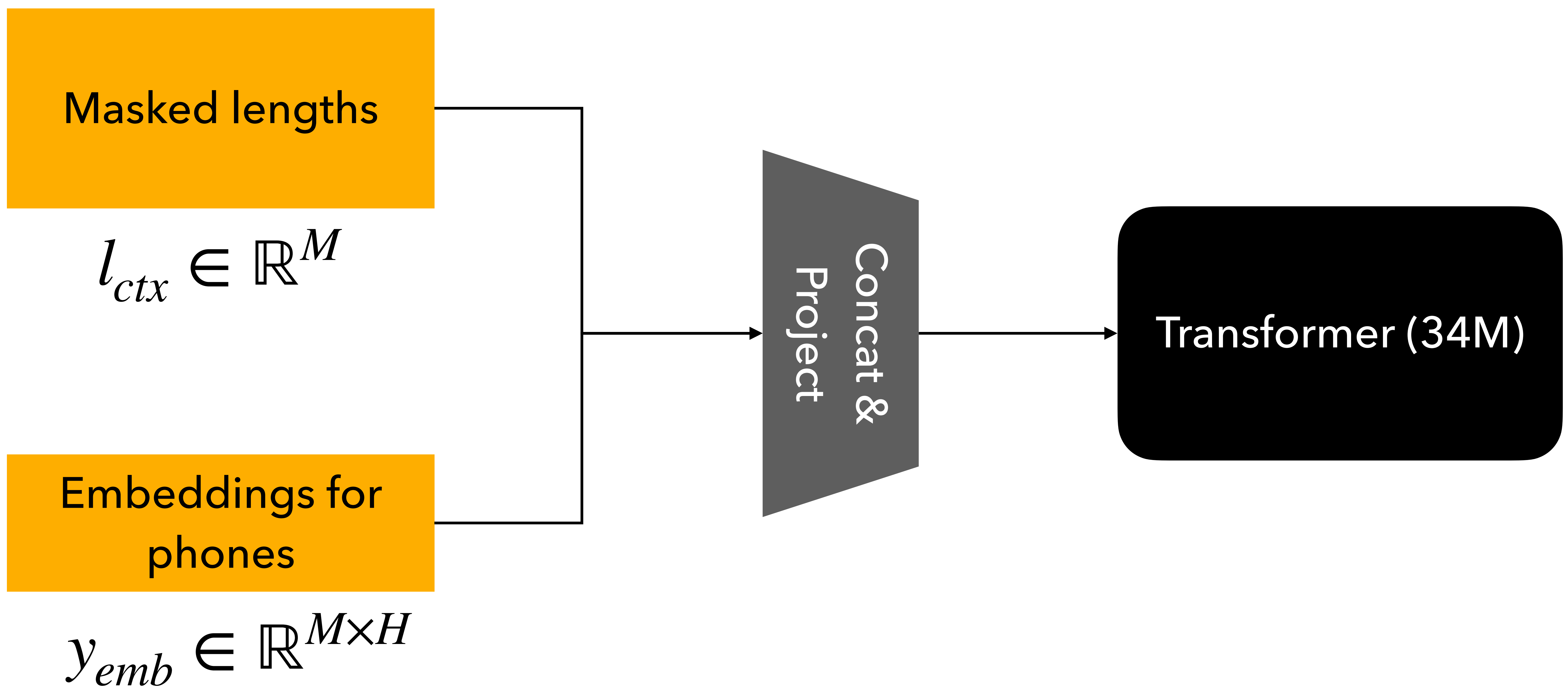
Methodology

Audio Model



Methodology

Duration Model (same as FastSpeech)



Methodology

Inference

$$x = x_T = x_0 + \int_0^T f(x_t, t, \theta)dt$$

1. Sample x_0.
2. Use numerical ODE solver to solve the above equation for x.

Experiments

Setup

- 330M parameter transformer is used for audio model.
- VB-En model trained on 60k hours of audiobooks in English
- VB-Multi trained on 50k hours of audiobooks in 6 languages

Results

Monolingual zero-shot TTS

Table 2: English zero-shot TTS results on filtered LS test-clean. "-" results are not available. We obtain VALL-E continuation SIM result through communication with the authors.

Model	WER	SIM-o	SIM-r	QMOS	SMOS
Ground truth	2.2	0.754	n/a	$3.98_{\pm 0.14}$	$4.01_{\pm 0.09}$
cross-sentence					
A3T	63.3	0.046	0.146	-	-
YourTTS	7.7	0.337	n/a	$3.27_{\pm 0.13}$	$3.19_{\pm 0.14}$
VALL-E	5.9	-	0.580	-	-
VB-En	1.9	0.662	0.681	$3.78_{\pm 0.10}$	$3.71_{\pm 0.11}$
continuation					
A3T	18.7	0.058	0.144	-	-
VALL-E	3.8	0.452*	0.508	-	-
VB-En ($\alpha = 0.7$)	2.0	0.593	0.616	-	-

Results

Cross-lingual zero-shot TTS

Table 4: Multilingual zero-shot TTS SMOS/QMOS results on filtered MLS English test set with prompts in different languages. YT/VB-Multi refers to YourTTS/multilingual Voicebox. "Ref" shows the audio context language.

	Ref=De	Ref=En	Ref=Es	Ref=Fr	Ref=Pl	Ref=Pt
	SMOS (target text = En)					
YT	$3.26_{\pm 0.11}$	$3.24_{\pm 0.11}$	$3.22_{\pm 0.12}$	$3.48_{\pm 0.10}$	$3.26_{\pm 0.09}$	$3.38_{\pm 0.11}$
VB-Multi ($\alpha = 1.0$)	$3.89_{\pm 0.10}$	$3.93_{\pm 0.08}$	$3.84_{\pm 0.10}$	$3.92_{\pm 0.09}$	$3.81_{\pm 0.08}$	$3.96_{\pm 0.09}$
	QMOS (target text = En)					
YT	$3.29_{\pm 0.12}$	$3.17_{\pm 0.13}$	$3.29_{\pm 0.12}$	$3.08_{\pm 0.12}$	$3.35_{\pm 0.12}$	$3.21_{\pm 0.12}$
VB-Multi ($\alpha = 1.0$)	$3.67_{\pm 0.09}$	$3.48_{\pm 0.09}$	$3.45_{\pm 0.11}$	$3.31_{\pm 0.12}$	$3.75_{\pm 0.11}$	$3.35_{\pm 0.13}$

Results

Transient noise removal

- A3T and VB-En use transcript and location of noisy segments.
- VB-En is basically doing infilling rather than denoising.

Table 5: Transient noise removal where noise overlaps with 50% of the speech at a -10dB SNR.

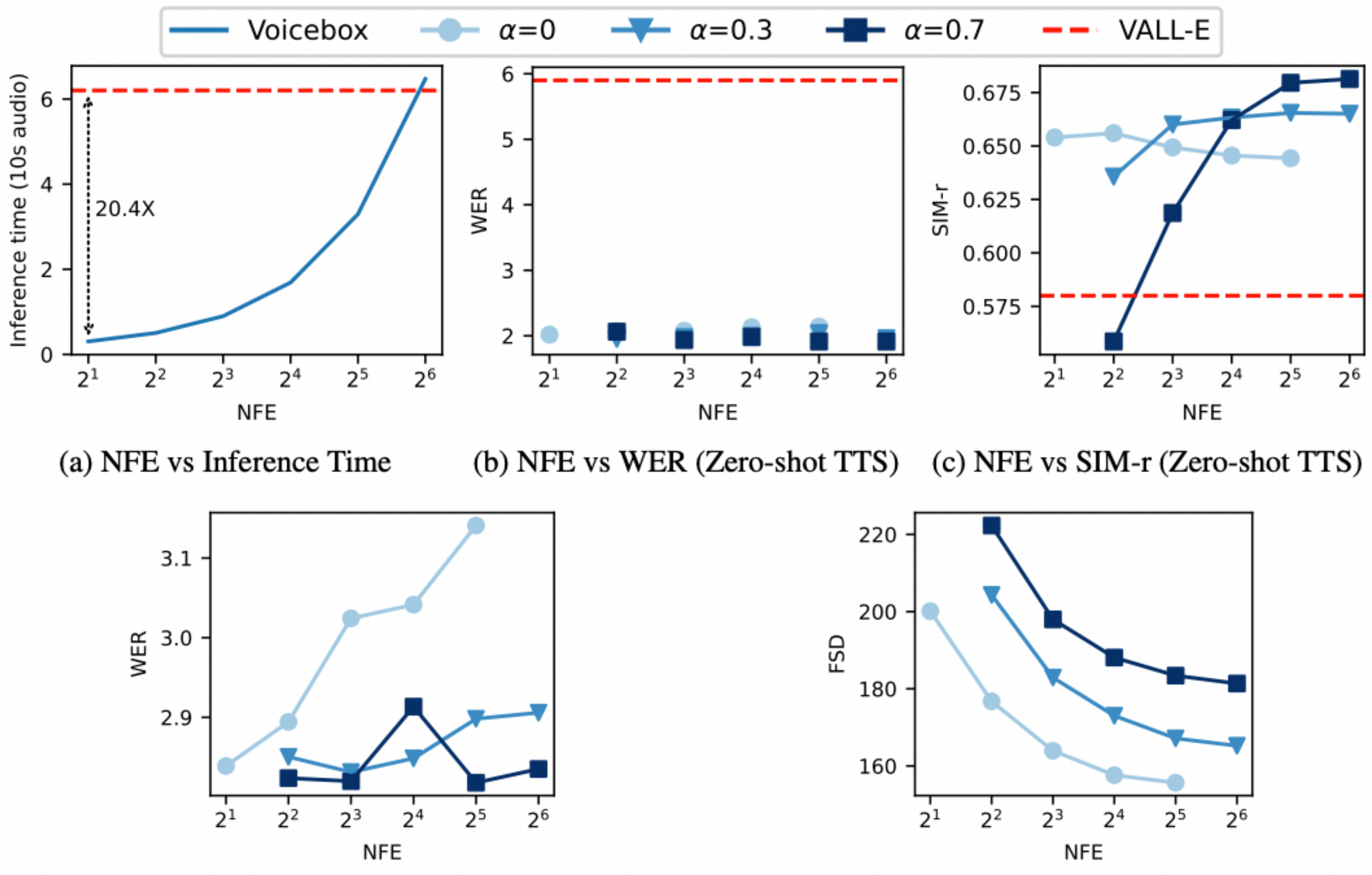
Model	WER	SIM-o	QMOS
Clean speech	2.2	0.687	$4.07_{\pm 0.15}$
Noisy speech	41.2	0.287	$2.50_{\pm 0.15}$
Demucs	32.5	0.368	$2.86_{\pm 0.17}$
A3T	11.5	0.148	$3.10_{\pm 0.15}$
VB-En ($\alpha = 0.7$)	2.0	0.612	$3.87_{\pm 0.17}$

Results

Inference time

- Proportional to number of function evaluations (NFEs)

$$x = x_T = x_0 + \int_0^T f(x_t, t, \theta)dt$$



(a) NFE vs Inference Time (b) NFE vs WER (Zero-shot TTS) (c) NFE vs SIM-r (Zero-shot TTS)

(d) NFE vs WER (Diverse speech sampling) (e) NFE vs FSD (Diverse speech sampling)

Figure 2: Trade-off between NFE and different metrics of interest.

Results

Effect of context duration (on similarity)

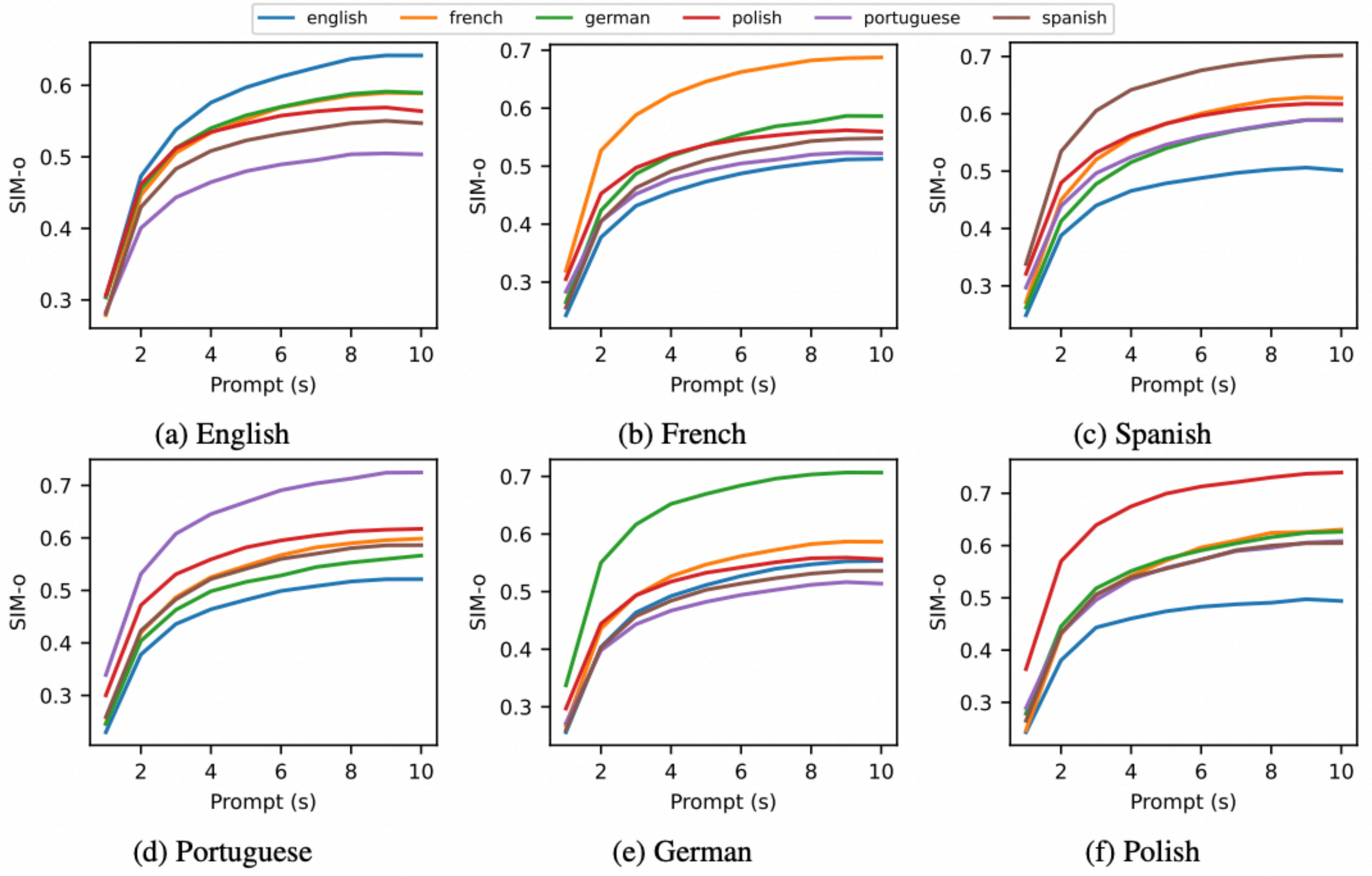


Figure 4: Each subplot considers one of the six target language and shows SIM-o (speaker similarity) as a function of prompt audio duration in seconds for cross-lingual style transfer from different source language. We set the classifier-free guidance strength (α) to 1.0 and use midpoint ODE solver with a NFE of 32.

Results

Effect of context duration (on WER)

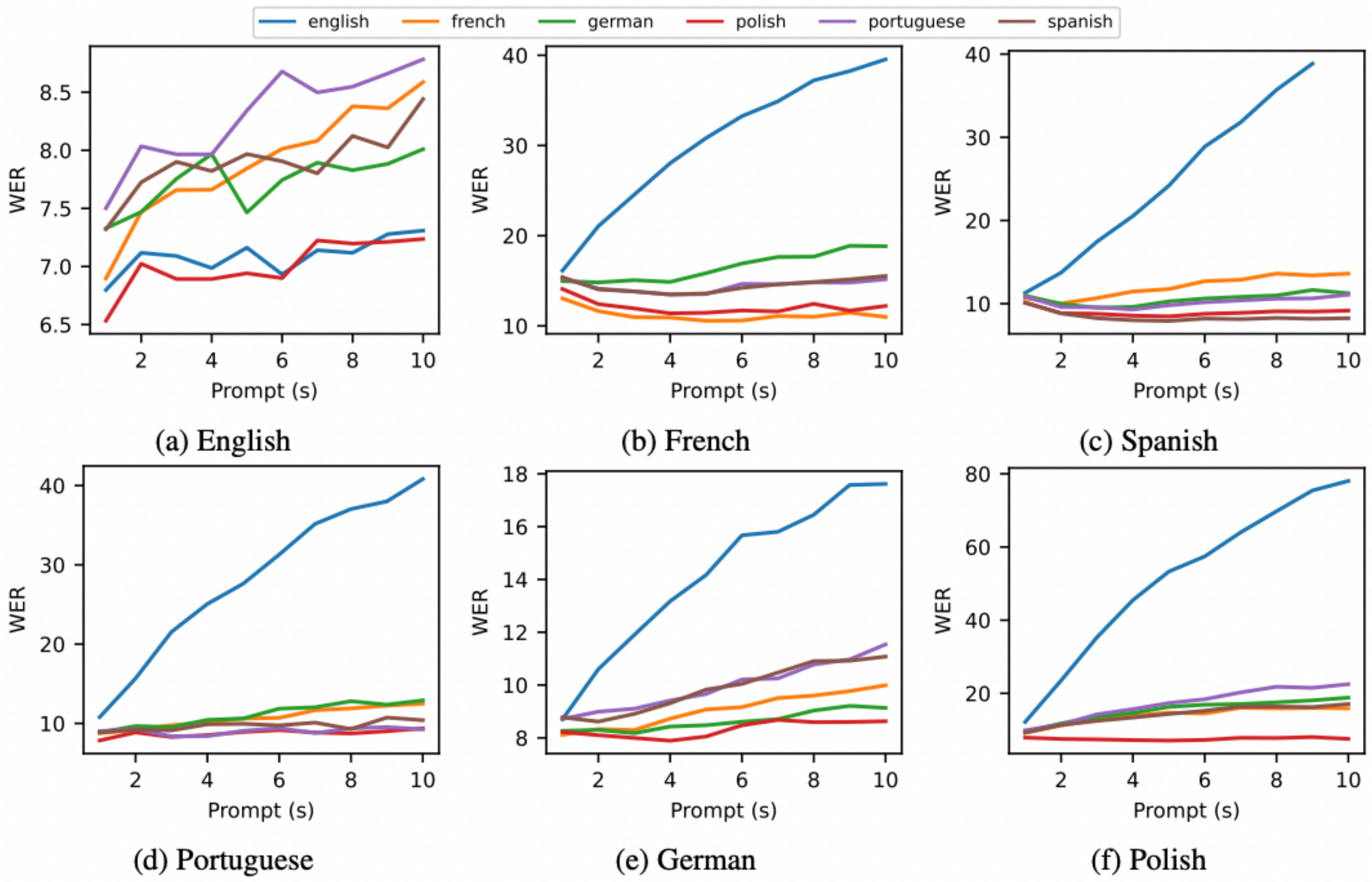


Figure 5: Each subplot considers one of the six target language and shows WER as a function of prompt audio duration in seconds for cross-lingual style transfer from different source language. We find WER remain reasonably low for all cases except for "English" to "X" style transfer.We set the classifier-free guidance strength (α) to 1.0 and use midpoint ODE solver with a NFE of 32.