Energy-Based Models for Speech Synthesis

Wanli Sun

Supervisor: Dr Anton Ragni

School of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

Speech and Hearing Research Group July 2026

This thesis is dedicated to my parents for their unconditional love, encouragement, and support. In particular, I dedicate it to my father, whose sacrifices paved the way for my academic career. Although you are no longer with me, I hope this achievement would have made you proud.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Wanli Sun
July 2026

Acknowledgements

First of all, I feel extremely fortunate to have been supervised by Dr Anton Ragni. His patience, insight, and guidance provided a strong foundation for my research career. His support and encouragement throughout my PGT and PhD studies were invaluable, particularly during moments of self-doubt. The research skills and his emphasis on logical and critical thinking will serve me well throughout my future career. I am deeply grateful to him for his mentorship, his feedback on my writing, and his guidance into the field of computer science research.

I would like to thank all past and present members in Regent Court for building an excellent academic environment in which I learnt a lot from everyone. Special thanks to Dr Jisi Zhang, Dr Mingjie Chen, and Dr Zehai Tu in the SpandH group for their help and advice throughout my PhD journey. I would also like to thank Dr. Yida Mu and Dr. Xianyuan Liu for our many late-night conversations.

Thanks to all my hiking, football, badminton, and basketball friends for the happy moments we spent together.

Special thanks to Jiaxuan Shi for the encouragement, support, and happy moments we spent together. To hold on to the memories we created is my good fortune. Thanks to the family of my eldest cousin for their encouragement and support.

Finally, I am grateful to my parents for their endless love and support.

Abstract

Speech synthesis is an essential technology in a range of widely used applications. Recent progress in speech synthesis has largely been driven by autoregressive (AR) and non-autoregressive (NAR) models that generate highly natural speech but can still produce inconsistencies. This thesis is motivated by the need to address these inconsistencies using generative models, particularly energy-based models (EBMs), and investigates how EBMs can be made practical by connecting them to related frameworks such as diffusion and flow-matching methods. This thesis presents a study on applying energy-based models for speech synthesis. The primary objective of this research is to train energy-based TTS models with commonly used noise contrastive estimation (NCE). During this development the close connection between EBMs and diffusion models developed at the same time became apparent. Building on this connection, score matching, which does not rely on noisy samples, was investigated as an alternative training approach. However, score matching approaches, such as sliced score matching (SSM), is complicated to implement and requires careful tuning. A new loss function is proposed in this thesis to simplify SSM and achieve similar performance. This loss function has close connection to flow matching models which further strengthens the link between EBMs and diffusion like models. The results presented in this thesis show that the proposed methods can effectively improve the quality of synthesized speech compared to pre-trained TTS models.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AR Autoregressive Model

ASR Automatic Speech Recognition

CV Computer Vision

DDPM Denoising Diffusion Probabilistic Model

DM Diffusion Model

DNN Deep Neural Network

DSM Denoising Score Matching

DTW Dynamic Time Warping

EBM-TTS Energy-Based Text-to-Speech

EBM Energy-Based Model

FFE F0 Frame Error

FM Flow Matching

FM Frequency Masking

G2P Grapheme-to-Phoneme

GD Gradient Descent

HMM Hidden Markov Model

LR Learning Rate

MAS Monotonic Alignment Search

MCD Mel-Cepstral Distortion

MCMC Markov Chain Monte Carlo

MLE Maximum Likelihood Estimation

MOS Mean Opinion Score

NAR Non-Autoregressive Model

NCE Noise Contrastive Estimation

NF Normalizing Flow

NLP Natural Language Processing

ODE Ordinary Differential Equation

PCM Pulse Code Modulation

PDF Probability Density Function

RM Random Masking

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SM Score Matching

SSM Sliced Score Matching

TM Time Masking

TTS Text-to-Speech

TW Time Warping

WD Weight Decay

Chapter 1

Introduction

Speech synthesis, also known as text-to-speech (TTS), is a popular and important area of research. It is widely used in various applications, such as voice assistants and dialogue systems. The goal of TTS is to convert written text into spoken language to enable machines to communicate with humans in a natural and intelligible manner. Traditional TTS techniques include concatenative speech synthesis [84, 11, 50] and statistical parametric speech synthesis (SPSS) [125, 145, 31, 144]. Concatenative TTS involves gluing pre-recorded speech segments to form complete utterances, while SPSS uses statistical models (e.g., hidden Markov models (HMMs), deep neural networks (DNNs), and recurrent neural networks (RNNs)) to generate acoustic parameters, such as log-Mel spectrogram frames that are then synthesized into audio.

In recent years, deep learning techniques have significantly advanced the field of TTS, leading to the development of more natural and expressive synthetic voices. Autoregressive (AR) models, such as Tacotron 2 [112] and Transformer-TTS [69], model the distribution of acoustic parameters of the entire utterance as a product of conditional distributions along time $t = 1, \ldots, T$, where each conditional distribution at time t depends on the previously generated output frames at times $\tau = t-1, \ldots, 1$. However, these AR models are almost exclusively trained by replacing the model's generated output frames with reference values (i.e., teacher forcing [139]). This mismatch between training and inference causes an inconsistency known as *exposure bias* [103], which can lead to poor audio quality (e.g., repetitions, skips, and long pauses) [113]. Few attempts have been made to address this inconsistency. Perhaps the most popular method is scheduled sampling [8], which uses a combination of generated outputs and reference values during training. However, this is still an inconsistent strategy for training AR models [51]. Other proposed solutions include variants of attention mechanisms, such as introducing monotonic attention to prevent skipping and repeating [42, 146], applying adversarial methods in the training algorithm [38], and

using regularization to relieve propagated errors [148]. Another issue is that training and inference with AR models are slow because the model must generate audio one frame at a time, which is time-consuming and inefficient.

Another approach is to utilize non-autoregressive (NAR) models, which generate all audio frames in parallel. NAR models, such as FastSpeech 2 [106], Glow-TTS [59], and Grad-TTS [98], are trained to predict all audio frames simultaneously, which can significantly speed up inference. Although many NAR architectures can model long-range dependencies, for example, FastSpeech 2 uses Transformer blocks, they still involve practical trade-offs between quality, robustness, inference speed, and training complexity. Therefore, AR and NAR approaches are best viewed as complementary paradigms with different strengths and limitations rather than strictly conflicting assumptions.

Energy-based models (EBMs) [44, 66] are a class of models that address the inconsistencies of both AR and NAR models by leveraging the concept of energy. These EBMs can be trained using noise contrastive estimation (NCE) [39, 79], which compares real data samples (reference audio) with noisy samples (low-quality audio). An ideal EBM should assign low energy to real audio and high energy to other samples (often called fake audio). EBMs have been applied to tasks with discrete outputs using the NCE method, such as in Natural Language Processing (NLP) [7, 71, 21, 10, 41] and speech recognition [70]. Another training method for EBMs involves score-matching-related techniques [117], such as sliced score matching (SSM) [116] and denoising score matching (DSM) [130]. SM methods estimate the gradient of the log-probability density function (PDF) of the data distribution, which can then be used to train EBMs. DSM method is closely related to diffusion modeling, and diffusion models have already been extensively studied for speech synthesis [98]. In this thesis, what remains comparatively underexplored is the use of these score-based objectives as direct training methods for residual EBM-based TTS, compared with diffusion-based TTS modelling. These EBMs also support a residual formulation, where a base model such as an AR Tacotron 2 can be augmented with an energy term and renormalised to yield a valid generative model. Here, a residual formulation means augmenting a base TTS model with an energy-based correction term that iteratively refines generated acoustic features. Its motivation is not limited to exposure bias: even when strong NAR models perform well, an energy model can still serve as a plug-in refinement objective for improving robustness, controlling the quality–speed trade-off at inference time, and providing a unified view with EBMs, diffusion-based and flow-based optimization.

The overall aim of this thesis is to establish a practical and theoretically grounded framework for using energy-based models (EBMs) in TTS. To achieve this aim, the thesis has three concrete objectives: (i) to determine whether residual EBM formulations can

consistently improve synthesized speech quality over strong AR and NAR baselines; (ii) to compare training strategies, especially noise-contrastive and score-based methods, in terms of stability, data efficiency, and output quality; and (iii) to evaluate inference procedures as optimization algorithms, including their quality–speed trade-offs in realistic deployment settings. The research questions are therefore important because each one maps directly to an objective and to a corresponding experimental design, including controlled model comparisons, objective acoustic metrics, and subjective listening tests.

1.1 Research questions

1.1.1 Can energy-based models solve exposure bias in TTS?

Audio synthesized by AR models can be seriously affected because of the shortcomings (e.g. repetitions, skippings, long pauses) caused by exposure bias. *Can EBMs, in particular residual EBMs, fix the exposure bias of AR models?* A review of the literature suggests that energy-based models (EBMs) are a potential solution for addressing the exposure bias of AR models. Differently to popular AR and non-AR speech synthesis models, energy-based models (EBMs) [44, 66] map a pair of audio and text to an energy function, which outputs a scalar value referred to as energy. It is assumed that an ideal EBM would relate energy function with audio quality by assigning positive (high) energy to low quality audio and negative (low) energy to low quality audios. A well-trained EBM can iteratively improve audio quality through inference. When noisy samples are generated from some existing speech synthesis, base, model (e.g. Tacotron 2, FastSpeech 2) such EBMs are called residual EBMs (R-EBM). These R-EBMs need not inherit assumptions, approximations or inconsistencies of base models and thus are suitable for addressing issues such as exposure bias.

EBMs have been successfully applied to discrete output tasks (e.g., automatic speech recognition) and NLP tasks using noise contrastive estimation. However, two specific issues remain to be investigated in the TTS domain:

- Can EBMs be trained successfully with continuous outputs as in TTS tasks?
- How to obtain effective negative samples for training continuous output EBMs?

More precisely, a well-trained EBM for TTS should satisfy two related criteria. First, it should provide a meaningful ranking of speech candidates for a given text. Second, its energy surface should be useful for inference: the gradient of the energy function should guide an initial acoustic feature sequence towards a more plausible speech realisation rather than merely

separating obviously corrupted examples. This makes the construction of negative samples central to NCE training. A "good" negative sample should be difficult but informative: it should be close enough to realistic speech to expose perceptually relevant errors, but different enough from the reference to provide a reliable contrastive learning information. Since the model operates on speech representations, negative sampling should therefore be investigated using speech perturbations and augmentations.

1.1.2 Can we train EBMs with other approaches?

The training methods discussed in the previous section rely on noise contrastive estimation (NCE). While NCE works well for models with discrete outputs (e.g., language models), its effectiveness for continuous output models, such as in TTS, is less established. Compared with diffusion models, one practical advantage of EBMs is that training does not require a long sequential noising and denoising process. Positive and negative speech examples can be scored independently, so the training computation can be parallelised over examples in a minibatch. However, the main limitation appears during inference. EBMs usually refine acoustic features through gradient-like iterative updates. Unlike diffusion models, where each denoising step is trained with respect to a prescribed noise schedule, each EBM update simply follows the learned energy gradient. Therefore, an individual refinement step is not guaranteed to improve the sample, especially if the step size is poorly chosen or the learned energy surface is not smooth around the current estimate. The latter may suffer from training difficulties and the challenge of constructing negative samples, which is a crucial step for NCE. *Can we train EBMs with other approaches?* It is worth investigating whether alternative training methods, such as score matching (SM) techniques, can be applied to train EBMs for TTS tasks without requiring negative sampling.

1.1.3 Which inference approaches are better suited for EBMs?

Benefiting from the continuous form of output (e.g., spectral features), sampling from a trained EBM can be done using various inference approaches developed for EBMs and other generative models, such as Langevin dynamics [114], and other sampling methods. All these scheme can be cast as a form of iterative optimization, where the model is used to iteratively refine its output. Given the wide variety of optimisation and sampling methods available, *which inference approaches are better suited for EBMs?* It is important to investigate various inference methods to determine their effectiveness and fit specifically with EBMs.

1.1.4 Is there a strong relationship between energy-based and diffusion-like TTS models?

As briefly mentioned above and more comprehensively explored later in this thesis, inference with EBMs can be viewed as a form of iterative optimization, similar to diffusion and related flow matching models. Diffusion models have been shown to be effective in generating high-quality speech samples. *Is there a strong relationship between energy-based TTS and diffusion-like models?* From a theoretical perspective, it is worth investigating the relationship between EBMs and diffusion-like models. From a performance perspective, it is also worth investigating whether conceptually simpler EBMs can produce speech of competitive quality.

1.2 Contributions

The contributions of this thesis are summarized as follows:

- This thesis introduces the first energy-based paradigm for text-to-speech (EBM-TTS). It further identifies negative-sample quality as a core challenge when applying noise contrastive estimation (NCE) to continuous TTS outputs, and proposes several negative sampling strategies to address this challenge. The effectiveness of the model in generating high-quality speech is demonstrated through objective and subjective evaluations. This work has led to a publication at ICASSP 2024 [120].

- This thesis establishes score-based training as an alternative to NCE for EBM-TTS by applying sliced score matching (SSM) to speech synthesis. This contribution clarifies the relationship between energy-based and score-based parameterisations, and shows how EBMs can be trained without explicitly constructing negative samples. The proposed approach provides a common framework for comparing NCE and score-based EBM training, and is also evaluated. This work has led to a publication at INTERSPEECH 2025 [119].

- The third contribution is an exploration of the theoretical connections between EBMs and other generative models. A theoretical connection between EBMs and diffusion- or flow-based generative models has been established. It shows that their inference procedures can be interpreted as related forms of iterative optimisation. It also proposes delta loss, a simple score-based objective that achieves competitive performance and provides a new perspective on the relationship between EBMs and flow matching.

1.3 Thesis outline

The structure of this thesis is as follows:

- **Chapter 1: Introduction.** This chapter reviews TTS, highlights the inconsistencies in autoregressive (AR) and non-autoregressive (NAR) models, and presents the research questions and contributions of this thesis.
- **Chapter 2: Fundamentals of Text to Speech.** This chapter provides a comprehensive review of fundamental components in TTS, including text and audio preprocessing, the structure of acoustic models, and common vocoders. Datasets and evaluation metrics for TTS are also discussed.
- **Chapter 3: Probabilistic Non-autoregressive Acoustic Models.** This chapter reviews state-of-the-art non-autoregressive (NAR) generative models, including energy-based models, diffusion models, and flow matching. It also discusses their applications in TTS, along with their training and inference methods.
- **Chapter 4: Score-based Training Methods.** This chapter introduces a family of score-based methods for training energy-based models (EBMs) in TTS. It discusses the principles of score matching (SM), denoising score matching (DSM), and sliced score matching (SSM), and how these methods can potentially be applied to train EBMs for TTS tasks.
- **Chapter 5: Noise Contrastive Estimation for Energy-based TTS Models.** This chapter investigates a novel approach to training energy-based models (EBMs) for TTS using noise contrastive estimation (NCE). It discusses the design of the EBM architecture, the negative sampling methods, the relation to diffusion models, and the evaluation of the model's performance in generating high-quality speech.
- **Chapter 6: Score-based Training for Energy-based TTS Models.** This chapter proposes an alternative approach to NCE for training EBM-TTS, which addresses the difficulty of constructing negative samples. The relationship to flow matching, along with the training and inference methods, are discussed. The proposed method is evaluated on the LJSpeech dataset, demonstrating its effectiveness in generating high-quality speech.
- **Chapter 7: Inference Approaches for Energy-based TTS Models.** This chapter studies the overall performance of different inference optimizers for the proposed training methods. Instead of using the output of pre-trained TTS models as the initial

samples, starting from a simple distribution (e.g., Gaussian) is presented as a potential method.

- **Chapter 8: Conclusion.** This chapter concludes the thesis by summarizing the key findings and contributions, and discussing the future work in TTS.

Chapter 2

Fundamentals of Text to Speech

2.1 Pipeline

The task of text-to-speech (TTS) is to synthesise speech from text. This work focuses on neural speech synthesis. For an overview of earlier approaches, see concatenative TTS [124, 50], hidden Markov model (HMM)-based TTS [145, 142], and hybrid TTS [144, 100, 143, 31]. Figure 2.1 illustrates a modern neural TTS pipeline, commonly referred to as end-to-end TTS. The term is not used here in a strict sense, since many such systems still employ intermediate representations, such as linguistic features and Mel spectrograms. This class of systems was popularised by Tacotron [133]. In general, the TTS process comprises three main components: text processing, acoustic modelling, and vocoding. Direct approaches that generate waveforms from text have also received attention [89]. Such approaches are closely related to the end-to-end perspective outlined above, as they seek to reduce or eliminate hand-engineered intermediate stages and learn a larger portion of the text-to-waveform mapping within a single trainable framework. However, they do not render the modular pipeline shown in Figure 2.1 obsolete. In practice, many effective systems continue to decompose TTS into text processing, acoustic modelling, and vocoding, because this factorisation can improve training stability, data efficiency, controllability, and interpretability. Direct waveform models may therefore be understood as occupying a different position on a design spectrum: at one end are modular pipelines with explicit intermediate representations, such as Mel spectrograms, whereas at the other end are more tightly integrated models that attempt to learn the mapping more directly.

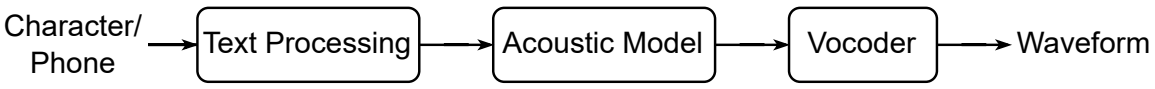


Fig. 2.1 The process of end-to-end text to speech (TTS) production.

Text processing converts text into a sequence of linguistic features, such as high-dimensional embeddings [83] of characters or phones. Acoustic models then map these linguistic features to acoustic features. Finally, vocoders transform acoustic features into waveforms. The following sections provide a more comprehensive description of these steps.

2.2 Preprocessing

Data pre-processing in TTS, and in this thesis in particular, consists of two parts: text preprocessing and audio preprocessing.

2.2.1 Text preprocessing

Characters

Since computers can process only numerical values, every character in the text transcripts must be transformed into a suitable numerical representation. All characters and symbols occurring in the transcripts constitute a vocabulary $\boldsymbol{V}$ (of size 28 in this thesis). The most general method for encoding symbols as numerical values is the one-hot representation. Each dimension in a one-hot vector represents a character or symbol. A value of 1 indicates that the corresponding symbol is present, and 0 otherwise. An example of one-hot encoding for the symbols `a, z, w, o, r` and `d` is shown below.

$$
\begin{aligned}
a &= \{1,0,0,0,0,...,0\} \\
z &= \{0,0,0,0,0,...,1\} \\
w &= \{0,0,0,0,1,...,0\} \\
o &= \{0,1,0,0,0,...,0\} \\
r &= \{0,0,1,0,0,...,0\} \\
d &= \{0,0,0,1,0,...,0\}
\end{aligned}
$$

As can be seen above, different characters have different representations.

The most critical limitation of one-hot encoding is that any two characters are represented independently. This form of encoding cannot convey much information, because the similarity between any two characters, for example as computed using the dot product, would be zero, even though some characters are clearly more closely related than others, such as vowels *versus* consonants. To address this issue, character embeddings are commonly used instead of one-hot encoding. Embedding-based character representations can have an arbitrary number of dimensions, unlike the one-hot representation ($|\boldsymbol{V}|$). The values of embedding vectors are not limited to zero or one. In this thesis, the dimension of the character embeddings is set to 256. An example of character embeddings is shown below:

$$
\begin{aligned}
a &= \{1.3, 0.5, 0.9, ..., 0.5\} \\
z &= \{0.4, -4.1, 0.1, ..., 1.5\} \\
w &= \{3.1, 1.9, -2.1, ..., 2.6\} \\
o &= \{-2.4, 3.5, -5.5, ..., 4.3\} \\
r &= \{-0.5, 3.2, 0.3, ..., 2.7\} \\
d &= \{2.2, 3.2, -2.5, ..., 0.9\}
\end{aligned}
$$

Phones

Conventionally, text processing maps character sequences to phone sequences through grapheme-to-phoneme (G2P) conversion in both automatic speech recognition (ASR) [33] and text-to-speech [107, 106].

A phone refers to any distinct speech sound unit produced by the human vocal tract. Phones can therefore be characterised in terms of their acoustic properties, such as frequency, amplitude, and duration, and can also be observed through articulatory movements, such as tongue position, lip rounding, and vocal fold vibration. Since phones are language-independent representations, although some phones are found only in particular languages, texts in any language can be transcribed using the International Phonetic Alphabet (IPA).

Phones, however, have a wide range of realisations because of the surrounding sounds, their position within a word, and various external (e.g. a cold) and internal (e.g. anxiety) factors. Allophones are variant phonetic realisations of a single phone. In TTS, allophones have previously been preferred over phones as more precise phonetic transcriptions for representing different features, such as aspirated and unaspirated stops, for example the /k/ sounds in "kit" and "skirt", respectively. However, in more recent work, simpler phone representations have become more common.

A grapheme-to-phoneme (G2P) model predicts phone sequences from grapheme sequences. Rule-based G2P systems are commonly used when grapheme-to-phone mapping is straightforward. In more complex cases, probabilistic methods [127, 5] can be employed. More recently, neural G2P systems, for example recurrent neural network-based [105] and transformer-based models [105], have been developed to achieve better performance.

2.2.2 Audio preprocessing

The audio preprocessing mainly includes threes steps: normalization, transformation to Mel spectrogram and logarithm.

Waveforms are commonly represented by sequences of positive and negative integer. The typical range is from -32768 (-2^{15}) to 32768 (2^{16}), which corresponds to commonly used 16-bit PCM (Pulse-code modulation) format. In the first preprocessing step the waveform is normalised to [-1, 1] range by dividing 16-bit integers by 32768.

In the second preprocessing step normalised waveforms are transformed to Mel spectrogram. The normalised waveforms are divided into a fixed number of analysis windows or frames. Consecutive windows overlap to address spectral distortions arising from applying analysis over short-term audio segments. A sampling rate of 22050 Hz (e.g. LJ-Speech) means that the waveform is sampled 22050 times per second. In this thesis, the analysis window duration is approximately 46.4 ms and the frame shift is approximately 11.6 ms. Expressing these quantities in time, rather than in samples, makes the description independent of the particular sampling rate used. Sequences of spectral vectors are produced by passing analysis windows through the Fast Fourier Transformation (FFT). These spectral vectors are then enhanced by a set (or bank) of filters spaced in frequency according to a Mel-scale [23]. The Mel-scale bank of filters has more filters placed in the lower frequency and less filters in the higher frequency. The overall number of filters is typically set to 80, as was done in this thesis. These 80 filters yield an 80-dimensional Mel spectrogram.

An example of Mel spectrogram is shown at the top of Figure 2.2, where brighter regions correspond to higher amplitudes and darker regions correspond to lower amplitudes. However, as the range of amplitude is compressed into a very small dynamic range (e.g. 10^{-3} to 10^{-5} interpreting Mel spectrogram visually is rather challenging. The highly compressed dynamic range leads also to challenges in applying predictive models. To overcome the issue, the original Mel spectrogram is additionally transformed into a logarithmic space (e.g. base-10). To avoid negative infinite values, the original Mel spectrogram values are clipped to a small value, such as 10^{-5}, before taking the logarithm. The Mel spectrogram after taking logarithm is shown at the bottom of Figure 2.2. In the following Mel-spectrogram will refer to the above Mel-spectrogram after application of logarithm for brevity.

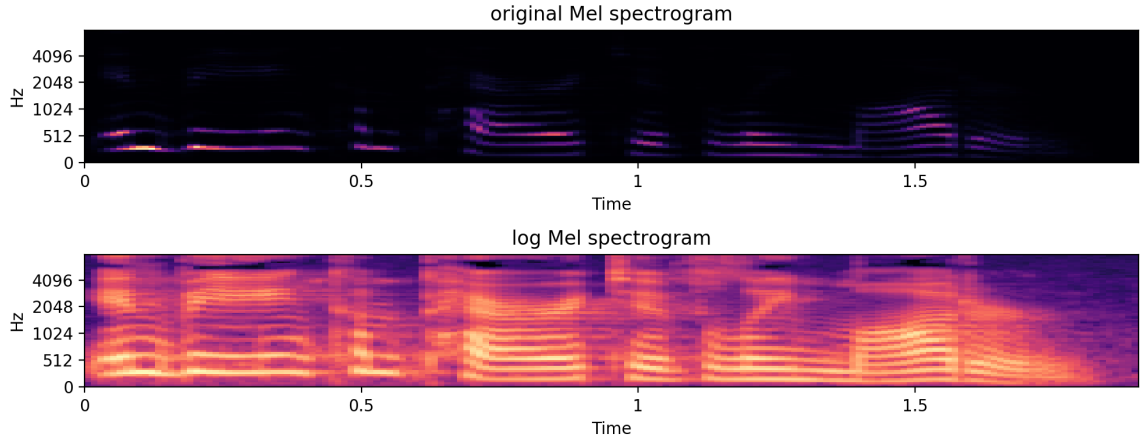


Fig. 2.2 Mel spectrogram and log-Mel spectrogram.

2.3 Acoustic models

This section discusses neural network-based acoustic models, the core components that define a neural network-based TTS (Neural TTS) system. The acoustic model is the core of Neural TTS, mapping linguistic features (e.g., characters or phones) to Mel-spectrograms. Recurrent Neural Networks (RNNs) are introduced first as a primary choice for sequence-to-sequence (seq2seq) models. Then, seq2seq TTS models using RNNs with an attention mechanism are discussed, which provide better alignments for sequential data. However, these models have two main drawbacks: 1) they are difficult to train in parallel, and 2) they struggle to handle long sequences, even with advanced RNN architectures like LSTM [46] and GRU [16]. To address these issues, Transformer-based TTS models are introduced, which have been shown to outperform RNN-based seq2seq models in TTS domain.

2.3.1 Recurrent neural networks (RNN)

Although simpler forms of deep neural networks (DNN) such as feed-forward and convolutional find their use in other areas, applying them for synthesis is complicated [144, 89]. One limitation of the feed forward DNN-based acoustic modeling is that the sequential nature of speech is ignored. Although certainly there are correlations between frames in speech, the DNN-based approach assumes that each frame is independent. Recurrent neural networks (RNN) [109] provide an elegant way to model speech-like sequential data. RNN iteratively computes hidden state vector sequence $\boldsymbol{h}_{1:T}$ and outputs vector sequence $\boldsymbol{Y}_{1:T}$ (e.g., spectral features) given input vector sequence $\boldsymbol{x}_{1:T}$ (e.g., characters spoken at each time t) as follows,

$$\begin{aligned} \boldsymbol{h}_t &= \sigma\left(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\right) \\ \boldsymbol{y}_t &= \sigma\left(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y\right) \end{aligned} \tag{2.1}$$

where $\boldsymbol{W}_h$, $\boldsymbol{W}_y$ and $\boldsymbol{U}_h$ are weight matrices; $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are weight (or bias) vectors; σ

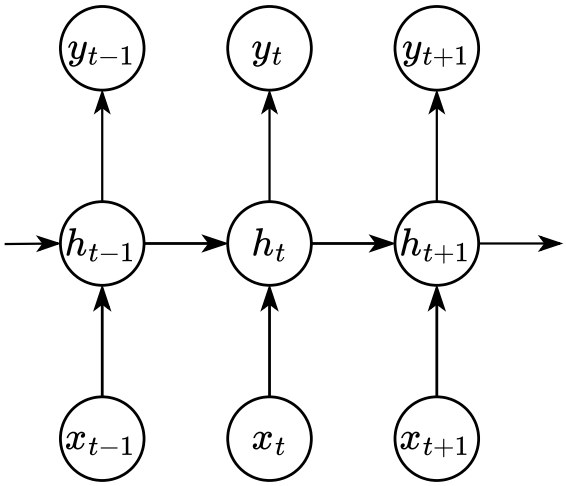


Fig. 2.3 Illustration of a RNN.

is the nonlinear activation function. Although their main purpose is to learn long-term dependencies, training them to practically take advantage of this property has proven to be problematic because the gradients either grow or shrink at each time step, so over many time steps they often either explode or vanish. Theoretical and empirical evidence shows that it is difficult for RNNs to learn to store information for very long [94]. To solve this problem, many variations have been proposed. Two popular proposals are the Gated Recurrent Unit (GRU) [20] and Long Short-Term Memory (LSTM) [46]. Bidirectional LSTM was applied to acoustic modelling as a part of neural network-based TTS (Neural TTS) in [31]. A unidirectional LSTM was then applied to achieve low-latency speech synthesis in [143].

2.3.2 Sequence to sequence and Tacotron 2

Tacotron 2 [112] is a deep neural network (DNN) model which produces spectral features $\boldsymbol{Y}_{1:T}$ from label sequences $\boldsymbol{x}_{1:L}$ (e.g. character sequences). As an example of a two-stage approach it utilizes a vocoder module (e.g. Griffin-Lim [37] algorithm, WaveGlow [99] and HiFi-GAN [63]) to generate speech waveform given the predicted spectral feature sequence $\boldsymbol{Y}_{1:T}$.

The Tacotron 2 model consists of three modules: the encoder, the attention module and the decoder. The encoder converts the input sequence $\boldsymbol{x}_{1:L}$ to hidden features $\boldsymbol{h}^{enc}_{1:L}$ containing

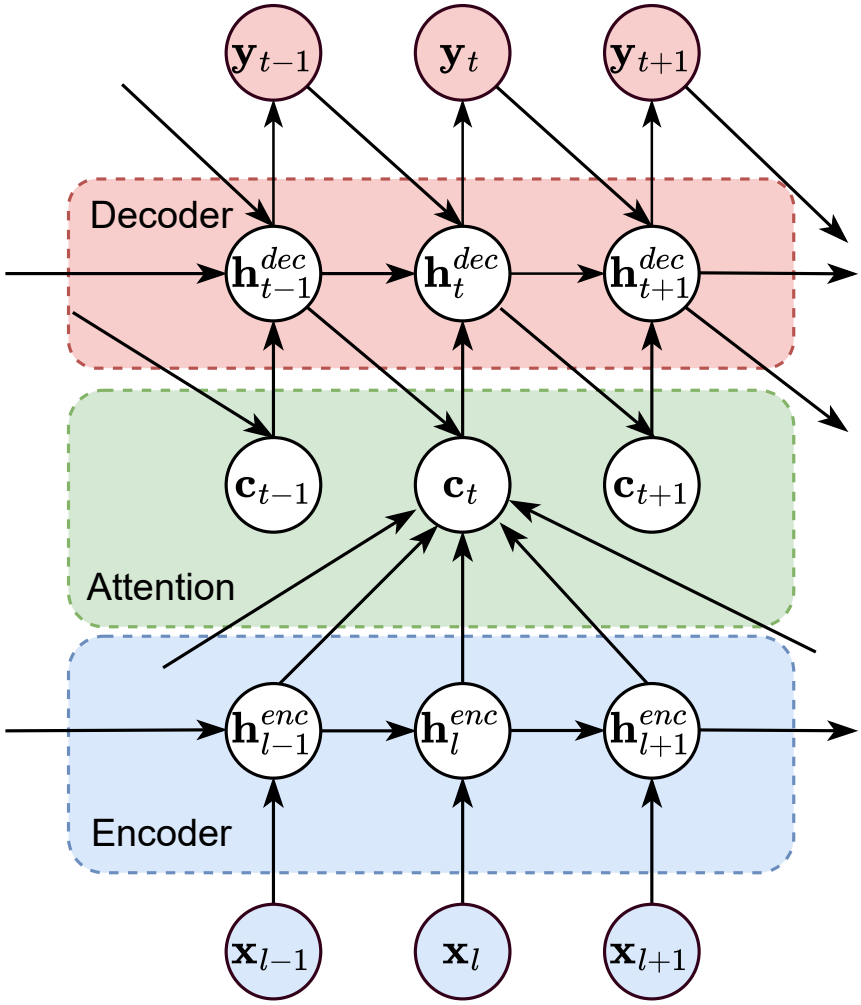


Fig. 2.4 Sequence-to-sequence model with attention. For clarity, only the relationship between $\boldsymbol{c}_t$ and the hidden states $\boldsymbol{h}^{enc}_{1:L}$ is drawn. In practice, every context vector in the sequence $\boldsymbol{c}_{1:T}$ utilizes all the hidden states $\boldsymbol{h}^{enc}_{1:L}$.

an optimal lexical representation for each character. For time step l, the hidden feature $\boldsymbol{h}^{enc}_l$ can be obtained using the following equation,

$$\boldsymbol{h}^{enc}_l = \boldsymbol{f}\left(\boldsymbol{x}_l, \boldsymbol{h}^{enc}_{l-1}\right), \tag{2.2}$$

where $\boldsymbol{f}$ is a nonlinear function (activation function), e.g. ReLU [86]. A number of recurrent neural network units, such as LSTM [46] and GRU [16], can be adopted to model the recursive relationship above.

The common use of hidden features $\boldsymbol{h}^{enc}_{1:L}$ is converting them to a fixed-length *context vector*. The initial version [121] of encoder-decoder models applied the last hidden feature $\boldsymbol{h}^{enc}_L$ as context vector. However, this may lead to bad performance[12] when the length of

input sequence $\boldsymbol{x}_{1:L}$ is very long and some of it is lost in the final state $\boldsymbol{h}_L^{enc}$. To solve these problems, an attention mechanism [17] was proposed to extend from one context vector per an output sequence to a sequence of context vector $\boldsymbol{c}_{1:T}$ between encoder and decoder [17]. Such a position dependent context vector is defined as follows,

$$\boldsymbol{c}_t = \sum_{l=1}^{L} w_{t,l} \boldsymbol{h}_l^{enc}. \tag{2.3}$$

where $w_{t,l}$ is an *attention weight* that is calculated as,

$$w_{t,l} = \frac{exp\left(s_{t,l}\right)}{\sum_{l=1}^{L} exp\left(s_{t,l}\right)}, \tag{2.4}$$

where $s_{t,l}$ is a *attention score* [6, 35, 78] that considers both hidden feature $\boldsymbol{h}_l^{enc}$ in the encoder and hidden feature $\boldsymbol{h}_{t-1}^{dec}$ in the decoder. One option for computing attention scores is given by,

$$s_{t,l} = \boldsymbol{\phi}\left(\boldsymbol{W}\boldsymbol{h}_l^{enc} + \boldsymbol{U}\boldsymbol{h}_{t-1}^{dec} + \boldsymbol{b}\right), \tag{2.5}$$

where $\boldsymbol{\phi}$ is a nonlinear function (e.g. ReLU), $\boldsymbol{W}$ and $\boldsymbol{U}$ are parameter matrices, $\boldsymbol{b}$ is a bias parameter vector. These context vectors $\boldsymbol{c}_{1:T}$ then take part in predicting spectral features $\boldsymbol{Y}_{1:T}$ generated by the decoder in place of global context vectors $\boldsymbol{h}_L^{enc}$ in encoder-decoder architectures.

Mel-spectrogram frames $\boldsymbol{Y}_{1:T}$ are produced frame by frame from hidden features $\boldsymbol{h}_{1:T}^{dec}$ of decoder, where $\boldsymbol{h}_t^{dec}$ integrates previous frame $\boldsymbol{y}_{t-1}$, context vector $\boldsymbol{c}_t$ and previous hidden feature $\boldsymbol{h}_{t-1}^{dec}$

$$\boldsymbol{h}_t^{dec} = \boldsymbol{f}\left(\boldsymbol{h}_{t-1}^{dec}, \boldsymbol{c}_t, \boldsymbol{y}_{t-1}\right) \tag{2.6}$$

The initial hidden state of decoder, $\boldsymbol{h}_0^{dec}$, is typically initialised using some form of random initialization [9].

The full architecture of Tacotron 2 is illustrated in Figure 2.5. To control the length T of spectral sequences $\boldsymbol{Y}_{1:T}$, a stop linear layer maps the output of decoder to a binary stop token. Predicting of each step $\boldsymbol{y}_t$ ceases once the stop token is predicted as 1. In particular, the spectrograms at the bottom of Figure 2.5 are a reflection of teacher forcing [139], which is used during training but not inference. This causes a mismatch for $\boldsymbol{y}_{1:t-1}$ between training and inference; the ground truth $\boldsymbol{y}_{1:t-1}$ is used during training, while the predicted $\boldsymbol{y}_{1:t-1}$ is used during inference. This mismatch is called exposure bias [103] and can lead to a discrepancy between the model's output distribution and the true data distribution.

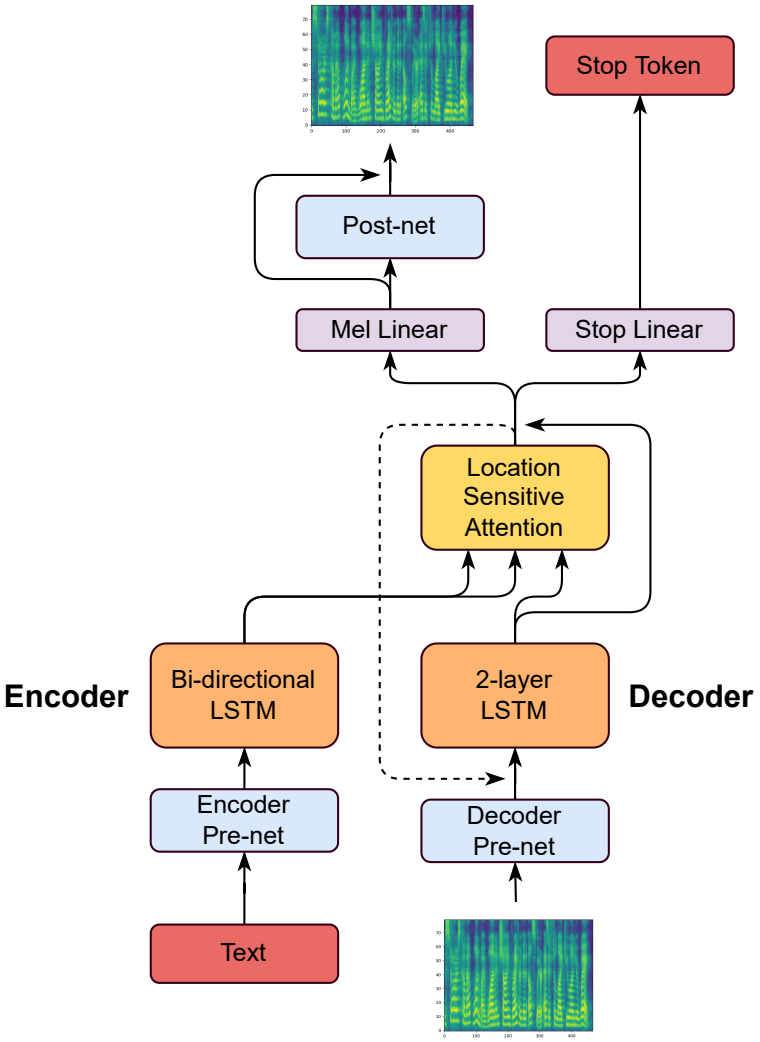


Fig. 2.5 Illustration of Tacotron 2. The figure is taken from the original paper [112].

However, Tacotron 2 may produce unnatural sound when dealing with long sentences [12]. Another issue is that it is costly to train and large quantities of data are typically required for training [122]. The use of recurrent units also makes parallelisation complicated. Although parallel training is possible [134], this has not yet been investigated for architectures like Tacotron 2.

2.3.3 Transformer-based TTS

Since the inputs and outputs of TTS models are sequential, errors can accumulate over many steps, leading to a significant degradation in performance. Another issue is that the current hidden states require the previous hidden states, which makes it difficult to parallelize training

and inference. To address these issues, the Transformer architecture was proposed [128] to replace RNNs in seq2seq models. Transformer-based TTS models leverage this architecture to achieve better performance and faster training.

Self-attention

So far the attention mechanism has been applied for alignment between two sequences with different length. In contrast, self-attention mechanism, which was inspired by the previously discussed attention mechanisms, was proposed to provide a representation of sequences. The context sequence in self-attention case, $\boldsymbol{c}_{1:L}$, has the same length as the sequence of input embeddings $\boldsymbol{h}_{1:L}$ and can be viewed as an improved in some sense representation of the input. The process involves three sequence: a query sequence $\boldsymbol{q}_{1:T}$ ($\boldsymbol{Q}$), a key sequence $\boldsymbol{k}_{1:L}$ ($\boldsymbol{K}$) and a value sequence $\boldsymbol{v}_{1:L}$ ($\boldsymbol{V}$). The self-attention $\boldsymbol{c}_{1:T}$ ($\boldsymbol{C}$) can be defined as follows,

$$\boldsymbol{c}_{1:T} = f(\boldsymbol{q}_{1:T}, \boldsymbol{k}_{1:L}, \boldsymbol{v}_{1:L}) \tag{2.7}$$

in a sequence form, where f applies the same attention mechanism for each query $\boldsymbol{q}_t$ in turn, or

$$\boldsymbol{C} = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{2.8}$$

in a block/matrix form, which can take advantage of parallelism of the modern hardware, where d_k denotes the dimension of $\boldsymbol{K}$. The above forms describe the general attention mechanism in Eq. 2.4. In self-attention the last two sequences are typically set to $\boldsymbol{h}_{1:L}$

$$\begin{aligned} \boldsymbol{Q} &= \underbrace{\boldsymbol{g}_{1:T}}_{\boldsymbol{q}_{1:T}} \\ \boldsymbol{K} &= \underbrace{\boldsymbol{h}_{1:L}}_{\boldsymbol{k}_{1:L}} \\ \boldsymbol{V} &= \underbrace{\boldsymbol{h}_{1:L}}_{\boldsymbol{v}_{1:L}} \end{aligned} \tag{2.9}$$

Multi-head attention

Multi-head attention can be defined as a stack of several (self-)attention mechanisms. Each head typically follows the same architecture, but does not share parameters with other heads. Let h be an index of head, $\boldsymbol{W}_q^{(h)}$, $\boldsymbol{W}_k^{(h)}$ and $\boldsymbol{W}_v^{(h)}$ are the corresponding linear projection matrices. The sequence of context vectors handled by head h can be formulated as

$$\boldsymbol{c}_{1:T}^{(h)} = f\left(\boldsymbol{W}_q^{(h)}\boldsymbol{q}_{1:T}, \boldsymbol{W}_k^{(h)}\boldsymbol{k}_{1:L}, \boldsymbol{W}_v^{(h)}\boldsymbol{v}_{1:L}\right) \tag{2.10}$$

where f is shown by Eq. 2.8. At time step t, the multi-head attention is used to formulate context vectors as follows,

$$\boldsymbol{c}_t = \boldsymbol{W}_c \left[\boldsymbol{c}_t^{(1)}; \boldsymbol{c}_t^{(2)}; \ldots \boldsymbol{c}_t^{(H)} \right] \tag{2.11}$$

where $[;]$ denotes concatenation of sequences. The self-attention and multi-head attention are two key elements of Transformer [128]. The architecture of Transformer is shown in Fig. 2.6. The Transformer is also a seq2seq model, which consists of an encoder and a decoder. The encoder is composed of a stack of identical layers, each containing two sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network. The decoder is also composed of a stack of identical layers, each containing three sub-layers: a multi-head self-attention mechanism, a multi-head attention mechanism over the encoder's output, and a fully connected feed-forward network. Although the Transformer is more parallelized than RNN-based seq2seq models in training, it is still autoregressive in nature, meaning that it generates one token at a time during inference.

Transformer TTS

Transformer-TTS [69] leverages Transformer [128] based encoder-attention-decoder architecture to generate Mel-spectrograms from phone sequences. The authors of Transformer-TTS argued that sequence-to-sequence models like Tacotron 2 suffer from two key issues: 1) due to the recurrent nature, both encoder and decoder can not be parallelised in both training and inference; 2) since text and speech sequences are usually very long, RNN is not good at modelling long-term dependencies. Transformer-TTS adopts the basic model structure of Transformer and achieves similar voice quality to that of Tacotron 2. Transformer-TTS benefits from faster training time, since the encoder can be parallelised in training. However, it still suffers from the slow inference speed because of the autoregressive decoder as does Tacotron 2 which Transformer-TTS was set to improve over.

FastSpeech 2

FastSpeech [107] was proposed as a solution to the slow inference speed and robustness exhibited by Tacotron 2 and Transformer-TTS. FastSpeech borrows many of design decisions made by the latter. It makes use of a feed-forward Transformer network to generate Mel-spectrograms in parallel, which greatly speeds up inference. It removes the attention mechanism between text and speech to avoid word skipping and repeating frame(s) issues. Fastspeech also makes use of a different mechanism to model duration called length regulator, which provides not only more refined duration modelling (at the level of individual phones) but also explicit control of duration.

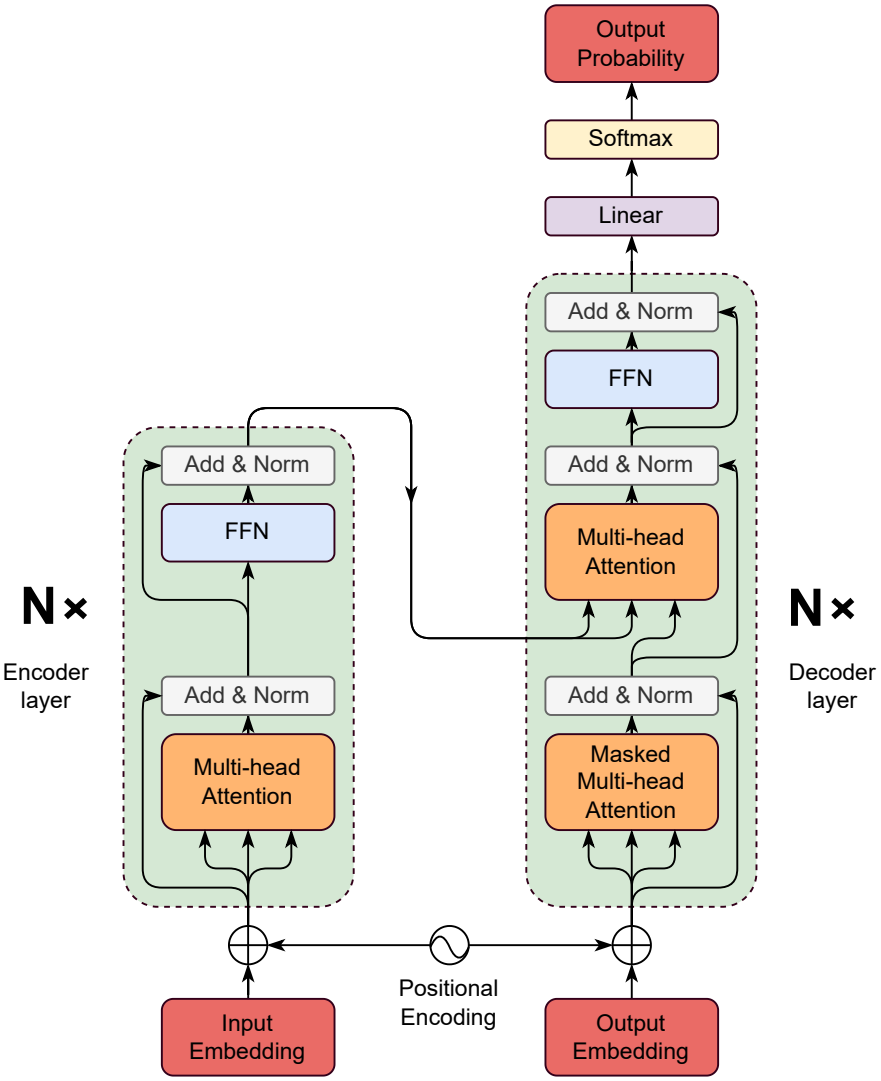
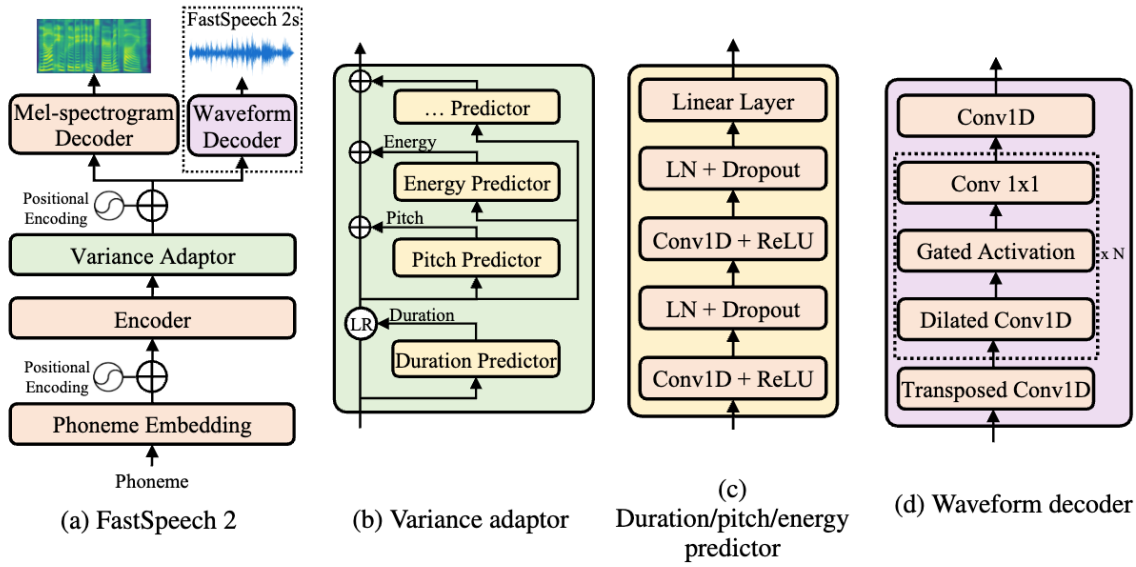


Fig. 2.6 Illustration of Transformer.

FastSpeech 2 [106] shown in Figure 2.7 was proposed shortly after to further enhance FastSpeech capabilities, mainly along two primary directions: 1) using ground-truth Mel-spectrograms as training targets, instead of distilled Mel-spectrograms from an autoregressive teacher model; 2) providing more variance-related information such as pitch, duration, and energy as an input to decoder, which eases the notorious one-to-many mapping problem [57, 32, 133, 150] in text to speech synthesis. The one-to-many mapping in TTS refers to the fact that there are multiple possible speech sequences corresponding to a single text sequence due to variations in speech, such as pitch, duration, sound volume, prosody, etc.



(a) FastSpeech 2 (b) Variance adaptor (c) Duration/pitch/energy predictor (d) Waveform decoder

Fig. 2.7 The architecture of FastSpeech 2. Subfigure (a) shows the overall architecture. Subfigure (b) illustrates the variance adaptor, which consists of a duration predictor, a pitch predictor, and an energy predictor. Subfigure (c) presents the similar architecture used for the duration, pitch, and energy predictors. Subfigure The architecture of the waveform decoder is shown in subfigure (d), which provides text-to-waveform synthesis without a vocoder. The figure is taken from the original paper [106].

2.4 Neural vocoders

Neural vocoders are a key component of two-stage TTS models. Those models produce speech waveforms by means of neural vocoders from spectral features (e.g., Mel-spectrograms) generated by acoustic models (e.g. Tacotron 2, Transformer-TTS, FastSpeech(2)).

2.4.1 Griffin-Lim

Griffin-Lim [37] algorithm was commonly used before neural vocoders became popular due to its ease of use. Given a target magnitude spectrogram, Griffin-Lim reconstructs a time-domain waveform by iteratively estimating the missing phase. Each iteration alternates between converting the current waveform estimate to the time–frequency domain, replacing its magnitude with the target magnitude, and transforming it back to the waveform domain. This attribute of iteration is related to the iterative refinement used later in energy-based and score-based inference, although Griffin-Lim operates on phase reconstruction rather than on learned energy or score functions. However, there are two severe drawbacks of

Griffin-Lim that led to its replacement by modern neural vocoders [89, 99, 63]. The first is that high-quality speech waveforms have to be slowly generated due to its iterative nature. The second is that the final quality of speech was still not good enough.

In recent years, the developments in the area of DNNs have significantly benefitted the area of vocoders for reaching both faster speed of inference and better quality of the final generated waveform. It has become possible to synthesize speech comparable to human quality in faster than real-time time. Three such vocoders have been employed in this thesis for generating high-quality speech: Parallel WaveNet [88], WaveGlow [99] and HiFi-GAN [63].

2.4.2 Parallel WaveNet

Parallel WaveNet [88] was developed as a faster neural vocoder based on the original WaveNet architecture [89]. While WaveNet can generate high-quality speech, its autoregressive formulation produces waveform samples one at a time, making inference slow. Parallel WaveNet addresses this limitation by training a parallel feed-forward student network to approximate an autoregressive WaveNet teacher through probability density distillation. Its main advantage is therefore fast, parallel waveform generation while retaining much of the perceptual quality associated with WaveNet. However, this improvement comes at the cost of a more complex training procedure: the student model must be carefully distilled from a strong teacher model, and training can be difficult to successfully convergence.

2.4.3 WaveGlow

WaveGlow [99] is a flow-based neural vocoder developed from the Glow generative framework. It combines invertible transformations with WaveNet-style coupling networks, allowing waveform likelihoods to be computed exactly during training and enabling audio samples to be generated in parallel during inference. Compared with Parallel WaveNet, an important advantage of WaveGlow is that it eliminates the need for a separately trained autoregressive teacher model and the distillation procedure. This makes the training objective simpler, since the model can be trained directly by maximum likelihood on waveform data conditioned on Mel-spectrograms. However, WaveGlow also has limitations: the model is relatively large, training can be computationally demanding, and high-quality generation may still require careful hyperparameter tuning and post-processing steps such as denoising.

2.4.4 HiFi-GAN

HiFi-GAN [63] is a generative adversarial network (GAN)-based neural vocoder. Instead of modelling waveform likelihood explicitly, as in flow-based vocoders, it trains a generator to transform Mel-spectrograms into waveforms and one or more discriminators to distinguish generated speech from real speech. The generator is fully convolutional and upsamples the input Mel-spectrogram to the waveform sampling rate using transposed convolutions and residual blocks with multiple receptive-field sizes. This design allows the model to capture both short-term waveform structure and longer-range periodic patterns that are important for natural speech. A key advantage of HiFi-GAN is its inference efficiency: waveform samples are generated in a fully parallel feed-forward network, and the generator contains far fewer layers than flow-based models. As a result, HiFi-GAN can be faster than flow-based vocoders while maintaining high perceptual quality. Its main drawback is that adversarial training can be sensitive to the balance between generator and discriminator objectives, and may require careful tuning to avoid training instability or artefacts in the generated waveform.

Although an important direction and a key component of any two-stage TTS model, (neural) vocoding is outside the scope of this thesis. For more information please see references made in this section.

2.5 Data

Several characteristics are critical for a TTS dataset: speech quality, transcripts, and speaker diversity. High-quality or 'clean' speech refers to recordings with minimal background noise. While normalized transcripts are often provided, utilizing large amounts of speech without corresponding text is an popular area of research. The number, diversity, and accent of speakers are also essential factors for building a robust TTS system. This thesis introduces the following datasets: LJ-Speech, VCTK, and LibriTTS.

2.5.1 LJ-Speech

In this thesis, the widely popular LJ-Speech Dataset [54] will be used as the training and testing dataset. It is an open-source speech synthesis dataset with several features. Firstly, the duration of the dataset is approximately 24 hours which is large enough to train a reasonably high-quality speech synthesis system [122][99]. Secondly, the audio has been segmented based on extended periods of silence so that input spectral feature sequences would not be very long to cause recurrent and attention-based models issues. Finally, audio transcripts have been carefully labelled with some text normalization of abbreviations. Table 2.1 lists a

Abbreviation	Expansion
Mr.	Mister
Mrs.	Misess
Dr.	Doctor
No.	Number
St.	Saint
Co.	Company
Jr.	Junior
Maj.	Major
Gen.	General
Drs.	Doctors
Rev.	Reverend
Lt.	Lieutenant
Hon.	Honorable
Sgt.	Sergeant
Capt.	Captain
Esq.	Esquire
Ltd.	Limited
Col.	Colonel
Ft.	Fort

Table 2.1 Abbreviation to expansion.

rather comprehensive list of expansions applied. Further details of the LJ-Speech dataset are provided below:

- 13,100 short audio files
- audio file length varies from 1 to 10 seconds
- recordings are of single speaker
- single-channel 16-bit PCM WAV file format
- sampling rate is 22050 Hz

2.5.2 VCTK

The VCTK (Voice Cloning Toolkit) [129] corpus contains speech samples from 110 English speakers with different accents, totalling approximately 44 hours of speech. Around 400 sentences from newspapers are recorded by each speaker. The sampling rate of the speech data is 48kHz. Although not used in this thesis, the VCTK dataset is popular in both voice cloning research as well as multi-speaker speech synthesis explorations.

2.5.3 LibriTTS

LibriTTS [141] is a dataset created for (multi-speaker) TTS that is derived from the LibriSpeech [90] corpus. Because LibriSpeech samples that contained noise were discarded, LibriTTS only contains approximately 585 hours of speech data recorded by 2,456 speakers. LibriTTS splits this data further to provide high-quality speech files for training validation and testing following the strategy adopted for Librispeech. The recordings are accompanied by normalized text and additional metadata. The details of subsets of LibriTTS are shown in Table 2.2.

Subset	Hours	Female speakers	Male speakers	Total speakers
dev-clean	8.97	20	20	40
test-clean	8.56	19	20	39
dev-other	6.43	16	17	33
test-other	6.69	17	16	33
train-clean-100	53.78	123	124	247
train-clean-360	191.29	430	474	904
train-other-500	310.08	560	600	1,160
Total	585.80	1,185	1,271	2,456

Table 2.2 The numbers of sentences in the original partial text, filtered sentences, and final output.

Table 2.2 features 'clean' subsets containing relatively high quality data as well as more noisier subsets labelled with 'other'. There are three possible training sets with 100, 360 and 500 hours of data providing possibility for building text-to-speech systems with different computational budgets.

2.6 Evaluation

For TTS, subjective metrics (Sec. 2.6.1) are the golden-standard for speech quality assessment. Speech quality should be assessed from a range of quality dimensions such as likeability, intelligibility, perceived intelligence [131]. To obtain any subjective metric, a group of participants needs to be asked to rate quality of speech samples from one or multiple TTS systems as well as real speech to determine ranking.

Meanwhile, objective metrics (Sec. 2.6.2), although less reliable and only correlated to subjective metrics, can still indicate quality of speech along various aspects. Moreover, ob-

jective metrics are time efficient and allow to save conducting expensive and time consuming measurements till the latter stages of development.

Recently, there has been interest in predictive metrics (Sec. 2.6.3) that combine features of subjective and objective metrics. Similar to subjective metrics they aim to provide an estimate of subjective characteristics (e.g. likeability or naturalness). Similar to objective metrics, they offer an algorithmic approach to compute them (e.g. a neural network).

2.6.1 Subjective evaluation

To balance efficiency and accuracy, it is common to employ 10 to 30 evaluators whose first language matches one being evaluated in the subjective test. Subjective tests can be efficiently and effectively conducted on various online platforms, such as Amazon Mechanical Turk. Audio samples, both real and generated, should be randomly selected from the test set to avoid bias to any specific order (e.g. book and other long-form media format).

Mean opinion score (MOS) [18]

In a MOS test, evaluators typically rate naturalness of audio samples on a scale of 1-5. The precision of scale could be 1 (*i.e.* 1, 2, 3, 4, and 5) or 0.5 (*i.e.* 0.5, 1, 1.5, ...). The final score attributed to an audio sample is the average over all scores received. The overall score for a system (or real data) is the average of these averages. MOS scores is the most significant criterion in evaluating the overall quality of speech.

Although MOS remains widely used, it should not be interpreted as a comprehensive description of speech quality. Recent studies have shown that MOS can be sensitive to the listener population, the set of systems included in the test, the presence or absence of low and high quality examples, the choice of utterances, and the statistical treatment of ordinal ratings [65, 97]. A single averaged naturalness score can also hide important perceptual dimensions, such as intelligibility, speaker similarity, prosody, expressiveness, pronunciation accuracy, and rare but severe artefacts. These limitations are particularly important for modern neural TTS systems, where several systems may be close to natural speech and small differences may not be reliably captured by a simple average MOS. For this reason, MOS is best treated as one useful but incomplete indicator of perceived quality, and should be complemented by confidence intervals, significance testing, listener- and utterance-level analyses, and more targeted listening tests when appropriate.

A/B preference

In A/B preference testing, two audio samples generated from different models, which may include real speech, are presented to an evaluator who is then expresses preference for either one (A) or the other (B). Results are reported in terms of percentage that system A is preferred over system B. The main drawback of this simple method is that it is not easy to compare the performance of system A with the performance of other systems obtained in other labs unless direct access to their files is available, unlike with MOS scores which are comparable assuming comparable evaluation settings.

Comparison mean opinion score (CMOS)[77]

Comparison Mean Opinion Score is similar to A/B preference test in that they are both comparative evaluation methods. While the results of A/B preference tests show the percentage of preference for system A and system B, they do not tell us by how much in a granular fashion. The CMOS test was designed to quantify the magnitude of preference. In the CMOS test, evaluators are asked to listen to two audio samples and evaluate how the former one (A) compares to the latter one (B) using a score in $[-3, 3]$ range. The average of average scores in the CMOS test provides a preference score calibrated to reflect the magnitude of preference.

Another commonly used subjective protocol is the MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) test [55]. In a MUSHRA-style evaluation, listeners are presented with several samples corresponding to the same transcript, including a reference, a hidden reference, and often one or more degraded examples. They then rate all samples on a continuous scale, typically from 0 to 100. This design encourages direct comparison between systems and can be more sensitive than MOS when systems are similar in quality. In TTS, MUSHRA-style tests are useful for comparing naturalness, speaker similarity, or prosodic quality across multiple systems, although care is still required because the original MUSHRA standard was developed for intermediate audio-quality assessment rather than specifically for modern speech synthesis.

2.6.2 Objective evaluation

Subjective listening tests provide reliable methods for assessment of speech quality. However, they can be very time consuming and expensive to rely on in the development of text-to-speech models. This section provides a short overview of the most popular objectives metrics.

Dynamic timing wraping Mel cepstral distance (DTWMCD)

Mel Cepstral Distance (MCD) measure [64] is used to measure how different are sequences of Mel cepstral features are. Note that Mel cepstral features and Mel-spectrogram features are not the same. The distance between sequences of Mel-cepstrum coefficients $\boldsymbol{c}_r$ and sequence of Mel-cepstrum coefficients $\boldsymbol{c}_s$ can be calculated as follows,

$$\text{MCD}(\boldsymbol{c}_r, \boldsymbol{c}_s) = \frac{10}{\log_e 10} \sqrt{2 \sum_{m=1}^{M} (c_r(m) - c_s(m))^2}, \tag{2.12}$$

where M represents the order of the Mel-cepstrum. Subscripts r and s represent real and synthesized speech. In practice, the length of two Mel-cepstra is usually not equal. So the Dynamic Timing Warping [85] algorithm is applied to align two sequences. Once aligned, Eq. 2.12 can then be used to calculate MCD between two cepstra with different length.

F0 frame error (FFE)

Accurate modelling of fundamental frequency (F0) is traditionally linked (correlated) with perceptual quality of generated speech. Automatic computation or estimation of F0 is however challenging in quiet as well as in noisy conditions. Gross Pitch Error (GPE) [101] and Voicing Detection Error (VDE) [101] are two types of errors that have to be balanced by any F0 tracking method. F0 Frame Error (FFE) [19] provides a trade-off measurement of both GPE and VDE. It could be represented as follows,

$$\begin{aligned} FFE &= \frac{\text{\# of error frames}}{\text{\# of total frames}} \times 100\% \\ &= \frac{N_{U\to V} + N_{V\to U} + N_{F0E}}{N} \times 100\% \\ &= \frac{N_{F0E}}{N} \times 100\% + \frac{N_{U\to V} + N_{V\to U}}{N} \times 100\% \\ &= \frac{N_{VV}}{N} \times GPE + VDE, \end{aligned} \tag{2.13}$$

where N is the number of the frames, $N_{V\to U}$ is the number of voiced frames which are classified as unvoiced frames, $N_{U\to V}$ is the number of unvoiced frames which are classified as voiced frames, N_{VV} is the number of frames which both the F0 tracker and the ground truth consider to be voiced, N_{F0E} is the number of frames for which

$$\left| \frac{F_{0_i}^{\text{estimated}}}{F_{0_i}^{\text{reference}}} - 1 \right| > \delta\% \tag{2.14}$$

where i is the frame number , and δ is a threshold which is typically 20. Thus, N_{F0E} is the number of frames where estimate of F0 has error 20% higher or lower than the ground truth F0 estimate. Using these quantities, GPE could be calculated as follows,

$$GPE = \frac{N_{F0E}}{N_{VV}} \times 100\% \tag{2.15}$$

Logarithm root mean square error of F0 (Log $F0$ RMSE)

Since the fundamental frequency is a key aspect of speech that determines prosody, intonation, and speaker identity, the Logarithm Root Mean Square Error of F0 (Log $F0$ RMSE) is another objective metric used to evaluate the accuracy of the predicted pitch contour. It calculates the root mean square error between the logarithm of the predicted F0 and the ground-truth F0. This calculation is typically performed only on voiced frames, where F0 is defined. The formula is given by:

$$\text{Log } F0 \text{ RMSE} = \sqrt{\frac{1}{N_v}\sum_{i=1}^{N_v}(\log F_{0,i}^{\text{pred}} - \log F_{0,i}^{\text{ref}})^2} \tag{2.16}$$

where N_v is the number of voiced frames, and $F_{0,i}^{\text{pred}}$ and $F_{0,i}^{\text{ref}}$ are the predicted and reference F0 values for the i-th voiced frame, respectively. Using the logarithm makes the metric more aligned with human perception, as pitch perception is logarithmic.

2.6.3 Predictive evaluation

Although subjective evaluation is reliable it is costly for assessment of speech quality. Thus, predictive evaluation is an attractive and effective approach to automatically predict the subjective rating for synthetic speech. Early works tried to condition simple statistical models like linear regression with carefully designed hand-crafted features, while recent works use deep neural networks (DNNs) to extract rich feature representations from inputs like magnitude spectrum, resulting in high correlations in both utterance-level and system-level scores [76].

Given an input speech sample, predictive models are trained to predict the mean of ratings produced by evaluators (listeners). Although DNN-based models require a large amount of data, due to budget constraint, typical listening test only engage a limited number of samples from each system to be rated. As a result, the number of per-utterance scores can be too small for DNN models to yield robust and reliable predictions. A more effective approach is to leverage all available ratings (predicting ratings rather than means of ratings) in the dataset.

In the literature this is known as listener-dependent (LD) modelling, and has been studied in the context of speech emotion recognition [2]. In addition to enlarging the amount of available data, another advantage of LD modelling is in accounting for individual preferences of listeners which are averaged out otherwise.

Another recent development in this field is so-called Mean-Bias network (MBNet) [67, 126], which consists of a mean subnet that predicts the utterance-level score of each utterance (mean) and a bias subnet that predicts the bias (defined as the difference between the predicted mean score and averaged listeners score). During inference, given an input speech, the bias net is discarded and only the mean net is used to make the prediction. On the other hand, approaches like LDNet [49] directly learn to predict the LD score given the input speech and the listener ID.

The recent popularity of speech self-supervised learning (SSL) models [4, 48, 13] have provided new feasible metrics [111, 3] for speech evaluations. One group of such approaches are inspired by automatic evaluation metrics (e.g., BLEU [91] and BERTScore [147]) found in other areas like natural language processing (NLP). For example, an SSL model could be used to generate discrete semantic tokens (rather than commonly used continuous acoustic features) given input speech. Then, any standard NLP metric, such as BLEU or BERTScore, can be computed. Such metrics [111, 3] leverage the power of pre-trained SSL models to better evaluate contextual aspects of speech generation, which might not be measurable by other commonly used objective metrics. Another advantage of these metrics is that they could provide a preliminary evaluation of speech before it is adopted for downstream applications.

2.7 Summary

This chapter provided a comprehensive overview of the fundamental topics in modern neural text-to-speech (TTS) synthesis. The chapter began by outlining the standard TTS pipeline, which consists of three main stages: text preprocessing, acoustic modelling, and vocoding.

The text and audio preprocessing stages were briefly introduced by covering topics such as transformation of text into numerical representations like indices or phone embeddings, and the conversion of audio waveforms into log-Mel spectrograms suitable for model training.

A review of foundational acoustic models traced their evolution from Recurrent Neural Networks (RNNs) to more advanced sequence-to-sequence architectures like Tacotron 2. The chapter highlighted the limitations of these autoregressive models, such as slow inference speed and potential robustness issues. Then transformer-based models were introduced, culminating in non-autoregressive systems like FastSpeech 2, which enable fast, parallel,

and robust spectrogram generation by explicitly modelling speech variance information like duration and pitch.

The role of neural vocoders, such as WaveGlow and HiFi-GAN, was highlighted, emphasizing their ability to convert acoustic features into high-fidelity waveforms, surpassing older methods like the Griffin-Lim algorithm.

Furthermore, the chapter described key datasets commonly used in TTS research, including LJ-Speech (also employed in this thesis), VCTK, and LibriTTS, outlining their respective characteristics.

Finally, a range of evaluation metrics were presented. These included traditional subjective tests (MOS, A/B, CMOS), objective metrics (DTWMCD, FFE), and the emerging field of predictive evaluation, which uses deep learning and self-supervised models to automatically estimate speech quality and wider contextual characteristics, offering a scalable alternative to manual listening tests.

Chapter 3

Probabilistic Non-autoregressive Acoustic Models

Given a text sequence $\boldsymbol{x}_{1:L}$ of length L, the simplest form of a probabilistic non-autoregressive acoustic model could model sequences of spectral features $\boldsymbol{Y}_{1:T}$ of length T as the product of conditional probabilities for each frame:

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T \mid \boldsymbol{x}_{1:L}\right) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_t \mid \boldsymbol{x}_{1:L}\right), \tag{3.1}$$

where $\boldsymbol{\theta}$ are model parameters. The term 'non-autoregressive' means that the model does not condition on previously generated outputs, such as on $\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}$ and so on when predicting the current frame $\boldsymbol{y}_t$. The loss function for this model is simply the sum of the negative log-likelihoods:

$$\mathscr{L}(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}\left(\boldsymbol{Y}_{1:T} \mid \boldsymbol{x}_{1:L}\right) = \sum_{t=1}^{T} -\log p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_t \mid \boldsymbol{x}_{1:L}\right), \tag{3.2}$$

Since there is no global normalization over the entire sequence $\boldsymbol{Y}_{1:T}$, it is a locally-normalized model.

More general probabilistic models do not assume such a strong independence among output variables. Instead they typically assume whole sequence dependencies. Such models can somewhat easier represent discrete domains (e.g., text), but are often intractable for continuous and high-dimensional domains (e.g., spectral features). This chapter is focused on the most popular models for continuous high-dimensional domains.

3.1 Energy-based models

An energy-based Model (EBM) [66] is a particular kind of probabilistic model defined through an energy function $E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$. The energy function is a scalar function that assigns a real-valued energy to each pair $(\boldsymbol{x},\boldsymbol{Y})$ (e.g. text $\boldsymbol{x}$ and $\boldsymbol{Y}$ speech). The energy function is parameterized by parameters $\boldsymbol{\theta}$ and is usually implemented as a neural network. An energy-based model (EBM) is defined as follows:

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})}}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})} \tag{3.3}$$

where $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is an intractable partition function that involves an integral over all possible high-dimensional $\boldsymbol{Y}$

$$Z_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int e^{-E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})} d\boldsymbol{Y} \tag{3.4}$$

As shown in Figure 3.1, a trained EBM assigns low energy to data with a higher likelihood and high energy to data with a lower likelihood.

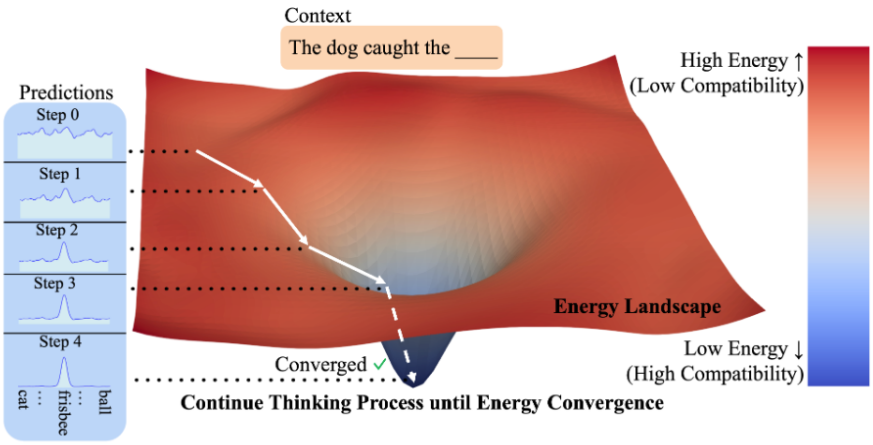


Fig. 3.1 EBMs assign low energy to ground truth data and higher energy to low likelihood data. The figure is taken from the original paper [34].

3.1.1 Training energy-based models

The objective function of training an EBM (Eq. 3.3) using maximum likelihood estimation (MLE) is defined as follows:

$$\mathscr{J}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{Y})\sim p_{\text{data}}}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right] = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{Y})\sim p_{\text{data}}}\left[E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) + \log Z_{\boldsymbol{\theta}}(\boldsymbol{x})\right] \tag{3.5}$$

Hence, the loss function can be written as:

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{Y})\sim p_{\text{data}}}\left[E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) + \log Z_{\boldsymbol{\theta}}(\boldsymbol{x})\right] \tag{3.6}$$

To optimize EBMs using gradient descent algorithm, the gradient of the logarithm of $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be approximated by sampling from the model [117]. The gradient of the loss function can be written as follows:

$$\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) \approx \mathbb{E}_{(\boldsymbol{x},\boldsymbol{Y})\sim p_{\text{data}}}\left[\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right] - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{Y})\sim p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})}\left[\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right] \tag{3.7}$$

The first term is easy to evaluate using automatic differentiation, while the second term is approximated by drawing random samples from the model. This allows the use of Markov chain Monte Carlo (MCMC), Langevin dynamics, or other sampling methods to optimize the model with a gradient descent algorithm.

Optimising maximum likelihood estimation (MLE) using MCMC

In addition to using MCMC methods for computing the gradient of the MLE function with respect to $\boldsymbol{\theta}$, it is possible to use these same methods (e.g., Langevin MCMC [92, 36] and Hamiltonian Monte Carlo [29, 87]) to compute gradients with respect to $\boldsymbol{Y}$:

$$\nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = -\nabla_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) - \underbrace{\nabla_{\boldsymbol{Y}}\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})}_{=0} = -\nabla_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}). \tag{3.8}$$

and use them to infer the optimal output sequence $\boldsymbol{Y}$. For example, the Langevin MCMC starts the sampling process from a simple prior distribution, and then iteratively updating it using the following update rule:

$$\boldsymbol{Y}^{(n+1)} \leftarrow \boldsymbol{Y}^{(n)} - \frac{\varepsilon^2}{2}\nabla_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})|_{\boldsymbol{Y}=\boldsymbol{Y}^{(n)}} + \varepsilon\boldsymbol{Z}^{(n)}, \quad n = 0,1,\cdots,N-1 \tag{3.9}$$

where $\boldsymbol{Z}^{(n)} \sim \mathscr{N}(\boldsymbol{0},\boldsymbol{I})$ is a Gaussian distribution, and ε is the step size. The Langevin MCMC method guarantees that $\boldsymbol{Y}^{(N)}$ approximates the true posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})$ as N approaches infinity. The Langevin MCMC method is a special case of Hamiltonian Monte Carlo (HMC) [29]. The latter (HMC) is more effective, but less efficient than Langevin MCMC. One drawback of Langevin MCMC is that it requires sufficient computational resource until convergence is reached.

Noise contrastive estimation (NCE)

In addition to MLE discussed above, another training method for EBMs is noise contrastive estimation (NCE) [39]. The idea of NCE is to bypass the partition function $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ by introducing a noise distribution with a known density. Samples drawn from the data distribution $(\boldsymbol{x}, \boldsymbol{Y}^{+})$ and the noise distribution $(\boldsymbol{x}, \boldsymbol{Y}^{-})$ are called 'positive samples' and 'negative samples', respectively. Typically, positive samples are ground truth samples. The loss function for NCE can be written as follows:

$$\begin{aligned} \mathscr{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{Y}^{-}) = &- \log \left(\frac{1}{1+\exp\left(E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\right)} \right) \\ &- \log \left(\frac{1}{1+\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{-})\right)} \right). \end{aligned} \tag{3.10}$$

As shown in Eq. 3.10, the NCE loss function is a binary classification problem where the model is trained to increase the energy for negative samples and decrease the energy for positive samples. Producing high-quality negative samples is critical. For instance, random noise can be used as negative samples, but the model can easily learn to assign high energy to them, making the training of EBMs ineffective. Instead, negative samples should be drawn from a distribution that is similar to that of the positive samples [7].

The method of negative sampling varies depending on the application. Specifically, it depends on whether the output is discrete (e.g., text) or continuous (e.g., spectral features). For example, in an automatic speech recognition (ASR) [70], negative samples can be the n-best hypotheses generated by a pre-trained ASR model via beam search. In machine translation [10], negative samples can be outputs from a pre-trained machine translation model that has a lower BLEU score [91]. In text generation [24, 7], negative samples can be produced by a pre-trained auto-regressive language model with a given ground-truth prefix.

Scores and score matching

The gradient of the log-probability density function (PDF) of a model $\boldsymbol{\theta}$ with respect to $\boldsymbol{Y}$ is defined as the *score* function:

$$s_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = -\nabla_{\boldsymbol{Y}} E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}), \tag{3.11}$$

where the score function is equal to the negative gradient of the energy function. Let $p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})$ be the true, unknown data distribution. The score function of the data distribution

is defined as follows:

$$s_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}) = \nabla_{\boldsymbol{Y}} \log p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}). \tag{3.12}$$

Using the score function provides a method for training EBMs without computing the intractable partition function $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$. Score matching [52] minimizes the Fisher divergence between the score function of the model and the score function of the data distribution:

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{Y}} \log p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}) - \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right\|^2\right]. \tag{3.13}$$

However, this objective is still intractable because the true data distribution $p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})$ is unknown. Further approximations allowing to simplify the score matching problem and yield practical training algorithms will be discuss in the Chapter 4.

The score function, defined as the gradient of the log-probability density function (PDF) with respect to $\boldsymbol{Y}$, is not the only score-related optimization method/technique in the literature. Another example is *Fisher score* [56] aimed to capture the feature space of a data distribution. This score is defined as the gradient of the log-likelihood function with respect to the model parameters $\boldsymbol{\theta}$:

$$s^F_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) \tag{3.14}$$

The Fisher score indicates the direction in which to adjust the parameters to increase the likelihood of an observed data point. Fisher kernels [56], derived from the Fisher score, can estimate generative models for variable-length sequential data by mapping sequence data into fixed-length feature vectors. Afterwards, these feature vectors can be used to train any standard discriminative classifier (e.g. maximum entropy), resulting in a powerful combined classifier with features of both generative and discriminative frameworks.

3.1.2 Inference methods

Inference methods for EBMs differ for continuous and discrete outputs. For continuous outputs (e.g., spectral features), inference is performed by iteratively updating hypotheses using the gradient of the energy function, such as with Langevin dynamics MCMC [92, 36] first mentioned in Section 3.1.1:

$$\boldsymbol{Y}^{(n+1)} \leftarrow \boldsymbol{Y}^{(n)} - \lambda\, \nabla_{\boldsymbol{Y}} E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})|_{\boldsymbol{Y}=\boldsymbol{Y}^{(n)}} + \sqrt{2\lambda}\boldsymbol{Z}^{(n)}, \tag{3.15}$$

where $\boldsymbol{Z}^{(n)} \sim \mathscr{N}(\boldsymbol{0}, \mu\boldsymbol{I})$, λ is an update rate, and μ is commonly set to 1. The iterative process starts from an initial hypothesis $\boldsymbol{Y}^{(0)}$, which can be sampled from a prior distribution, such as a Gaussian distribution, or generated by a pre-trained neural network. The choice

of the initial hypothesis can significantly affect the convergence speed and the quality of the final output. There is also a trade-off between the number of iteration steps N and the quality of the final output. A larger N leads to better convergence in theory but requires more computational resources. However, since the inference process can be viewed as a gradient descent optimization, the final output is also affected by the local optima of the energy function.

For discrete outputs (e.g., text), inference is performed by reranking hypotheses using algorithms such as beam search and top-k algorithm. These methods generate sequences of tokens by selecting the most probable tokens at each step, while also considering the context provided by the previous tokens. Hence, energy score as outputs of EBMs can be treated as alternative or additional criteria. For instance, in automatic speech recognition (ASR) [70], combinations of standard model scores with energy scores could be used to yield higher quality hypotheses given the n-best hypotheses generated by a pre-trained ASR model. In text generation [24, 7], energy scores are used for resampling top n samples with probabilities computed by $\frac{\exp(-s^i)}{\sum_{j=1}^{n}\exp(-s^j)}$, where s^i is the energy score of the i-th sample. This resampling process can be viewed as a form of importance sampling, where samples with lower energy scores are more likely to be selected.

3.2 Diffusion models

3.2.1 Grad-TTS

Grad-TTS [98] is an acoustic model that uses a diffusion model to generate high-quality spectral features. Unlike FastSpeech 2 [106] (Sec. 2.3.3), Grad-TTS does not rely on an external alignment model to determine alignment between sequences of lexical representations (phones) and sequences of frames. Similar to other TTS models [59, 60], monotonic alignment search (MAS) [59] (a realization of the Viterbi algorithm [102]) is used to predict the alignment. The architecture of Grad-TTS is shown in Figure 3.2. The model consists of a text encoder, a duration predictor, and a diffusion decoder. The text encoder encodes the input into a sequence of hidden states, which are then used to predict the duration of each state. The predicted durations are used to expand the sequence of lexical embeddings, which are then fed into the diffusion model to generate spectral features. During the inference stage, a rough sequence of spectral features is predicted by the encoder and the duration predictor, and the diffusion model updates these spectral features over T iterations. The hyperparameter T represents a trade-off balancing the quality of the generated spectral features and the inference speed. During the training stage, the loss function involves calculating the encoder

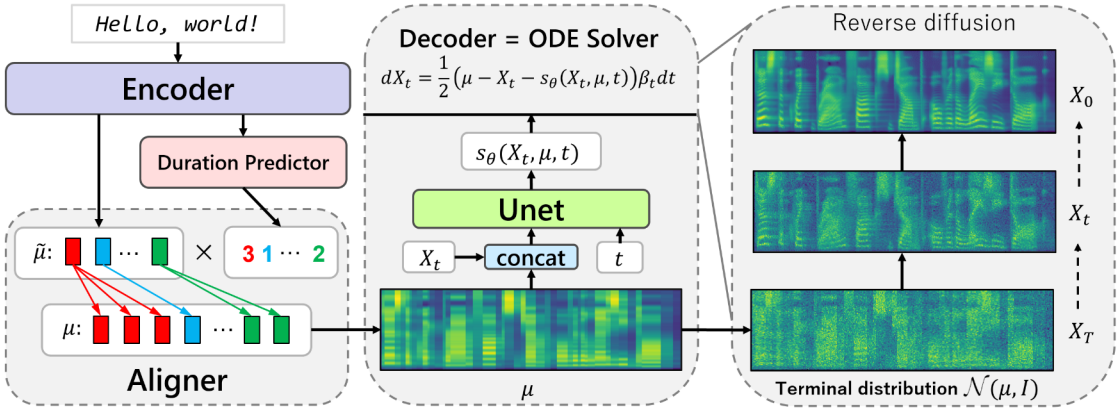


Fig. 3.2 Grad-TTS inference pipeline. The figure is taken from the original paper [98].

loss, the duration loss, and the diffusion loss. The diffusion loss similar to DDPM is to predict noise added at time t to the clean data sample $\boldsymbol{Y}_0$.

3.2.2 Denoising diffusion probabilistic model

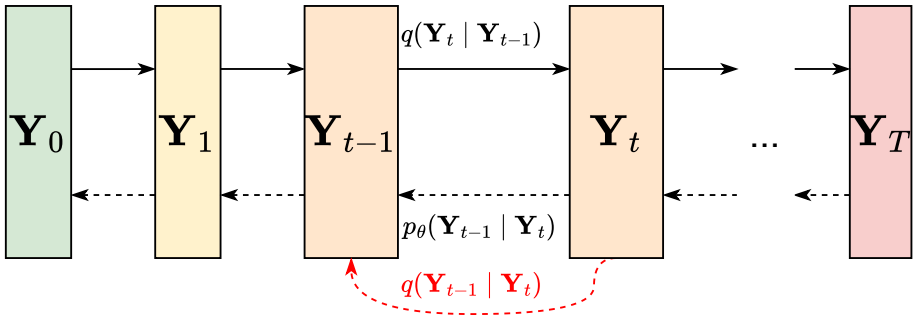


Fig. 3.3 Illustration of diffusion model: top process is *diffusion* process and bottom process is *reverse* process.

Another example of probabilistic non-autoregressive models is a denoising diffusion probabilistic model (DDPM) [45]. The DDPM consists of two parameterized Markov chains: *diffusion* process and *reverse* process. These two processes are shown in Figure 3.3. In diffusion process, a data distribution $\boldsymbol{Y}_0$ is gradually corrupted to a prior distribution $\boldsymbol{Z}$ (e.g., standard Gaussian distribution) by adding noise in T steps. A pre-defined Gaussian noise $\beta_t \in (0,1)$ is added to transitions at each time step t

$$\boldsymbol{Y}_t = \sqrt{1-\beta_t}\boldsymbol{Y}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{3.16}$$

where $\boldsymbol{\varepsilon}_t$ is a standard Gaussian distribution. The distribution of $\boldsymbol{Y}_t$ given $\boldsymbol{Y}_{t-1}$ is given by

$$q(\boldsymbol{Y}_t \mid \boldsymbol{Y}_{t-1}) = \mathcal{N}\left(\boldsymbol{Y}_t; \sqrt{1-\beta_t}\boldsymbol{Y}_{t-1}, \beta_t \boldsymbol{I}\right) \tag{3.17}$$

It is possible to show that both conditional expressions (Eqs. 3.16 and 3.17) can be rewritten as a function of data $\boldsymbol{Y}_0$ as follows:

$$\begin{aligned} \boldsymbol{Y}_t &= \sqrt{\bar{\alpha}_t}\boldsymbol{Y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_0 & (3.18)\\ q(\boldsymbol{Y}_t \mid \boldsymbol{Y}_0) &= q(\boldsymbol{Y}_t \mid \boldsymbol{Y}_{t-1}) \cdots q(\boldsymbol{Y}_1 \mid \boldsymbol{Y}_0)\, q(\boldsymbol{Y}_0) = \mathcal{N}\left(\boldsymbol{Y}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{Y}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\right) & (3.19) \end{aligned}$$

where $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. When $\bar{\alpha}_T$ approaches 0, $\boldsymbol{Y}_T$ is practically indistinguishable from pure Gaussian noise: $p(\boldsymbol{Y}_T) \approx \mathcal{N}(\boldsymbol{Y}_T; \boldsymbol{0}, \boldsymbol{I})$. The above simplification enables noisy versions of data $\boldsymbol{Y}_t$ to be computed in closed form for any time t. Thus, expensive simulations common with some of the previous approaches are not required with DDPMs.

If we can reverse the above process and sample from $q(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t)$, it will then be possible to recreate the true sample $\boldsymbol{Y}_0$ from a Gaussian noise input $\boldsymbol{Y}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. It has been proven that $q(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t)$ is approximately Gaussian if β_t is sufficiently small and $q(\boldsymbol{Y}_t \mid \boldsymbol{Y}_{t-1})$ is Gaussian [137]. Therefore, for all t the reverse process is parameterized by $p_{\boldsymbol{\theta}}$ as follows:

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_T) = p_{\boldsymbol{\theta}}(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t) \cdots p_{\boldsymbol{\theta}}(\boldsymbol{Y}_{T-1} \mid \boldsymbol{Y}_T)\, p_{\boldsymbol{\theta}}(\boldsymbol{Y}_T) \tag{3.20}$$

where

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t) = \mathcal{N}(\boldsymbol{Y}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t)) \tag{3.21}$$

Although $q(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t)$ can not be easily estimated, it is tractable when conditioned on $\boldsymbol{Y}_0$:

$$q(\boldsymbol{Y}_{t-1} \mid \boldsymbol{Y}_t, \boldsymbol{Y}_0) = \mathcal{N}\left(\boldsymbol{Y}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{Y}_t, \boldsymbol{Y}_0), \tilde{\beta}_t \boldsymbol{I}\right). \tag{3.22}$$

After simplifying the above equation, the mean and the variance of the reverse process can be written as follows:

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{Y}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{Y}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_t\right). \tag{3.23}$$

The above equation relies on an unknown estimate of noise $\boldsymbol{\varepsilon}_t$, which needs to be predicted by a neural network $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t)$. Thus, the loss function of DDPMs is simple and easy to implement [45] and can be written as follows:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \boldsymbol{Y}_0, \boldsymbol{\varepsilon}}\left[\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{Y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t\right)\right\|^2\right] \tag{3.24}$$

Given estimates of noise and data at the previous time t, the sample of $\boldsymbol{Y}_{t-1}$ at the following time $t-1$ can be drawn from the following distribution

$$\boldsymbol{Y}_{t-1} \sim \mathscr{N}\left(\boldsymbol{Y}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{Y}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{Y}_t, t\right)\right), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\boldsymbol{Y}_t, t\right)\right) \tag{3.25}$$

The pipeline of reverse (sampling) process is illustrated by Figure 3.4 for time t. The reverse process starts from a sample $\boldsymbol{Y}_T \sim \mathscr{N}(\mathbf{0}, \boldsymbol{I})$ and iteratively samples $\boldsymbol{Y}_{T-1}$ from the conditional distribution Eq.3.25 until reaching $\boldsymbol{Y}_0$ at time 0. In practice, the sampling process

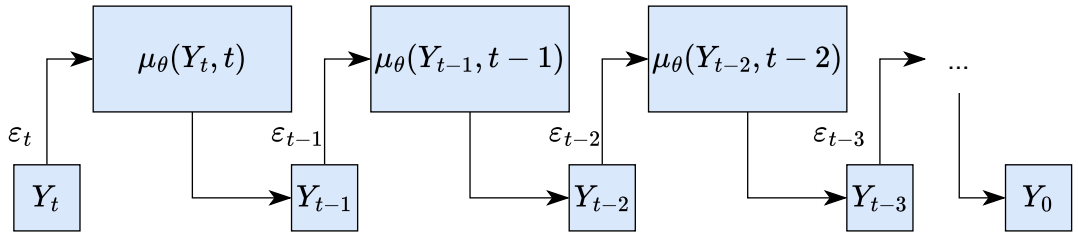


Fig. 3.4 Illustration of *reverse process*.

is often simplified by using a fixed variance factor σ_t instead of the variance predicted by the neural network. This simplification leads to the following training and sampling algorithm:

Algorithm 1 Training

1: **repeat**
2: $\quad \boldsymbol{Y}_0 \sim q$
3: $\quad t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\varepsilon} \sim \mathscr{N}(\mathbf{0}, \boldsymbol{I})$
5: $\quad$ Take gradient descent step on: $\nabla_\theta \left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{Y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t\right)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\boldsymbol{Y}_T \sim \mathscr{N}(\mathbf{0}, \boldsymbol{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \boldsymbol{Z} \sim \mathscr{N}(\mathbf{0}, \boldsymbol{I})$ if t > 1, else $\boldsymbol{Z} = \mathbf{0}$
4: $\quad \boldsymbol{Y}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{Y}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta\left(\boldsymbol{Y}_t, t\right)\right)$
5: $\quad + \sigma_t \boldsymbol{Z}$
6: **end for**
7: **return** $\boldsymbol{Y}_0$

where $\sigma_t = \eta\sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t}$. The temperature term η is a scaling factor of the variance.

3.3 Flow matching

3.3.1 Applications in TTS

Originally, discrete-time normalizing flows were utilized for acoustic models [82, 59], which are trained without the need for external alignment models. VITS [60] is a fully end-to-end (text-to-waveform) parallel TTS model that adopts variational inference augmented

with normalizing flows and an adversarial training process. Recently, optimal-transport conditional flow matching (OT-CFM) [72] has been utilized for training non-autoregressive acoustic models, for example Matcha-TTS [81]. The OT-CFM decoder learns a velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t)$ over spectral features, where the target distribution $\boldsymbol{Y}_1$ is the ground-truth spectral features and the prior distribution $\boldsymbol{Y}_0$ is a matrix of Gaussian noise with the same shape. A closely related recent example is F5-TTS [15]. As in Matcha-TTS, flow matching serves as the decoder and is used to learn a velocity field for speech generation. The key difference, however, is not the use of flow matching itself, but the surrounding TTS formulation. Matcha-TTS follows a more conventional acoustic-model pipeline, in which flow matching is used to generate spectral features from text-conditioned representations. By contrast, F5-TTS adopts a more streamlined formulation that avoids explicit duration modelling, a separate text encoder, and phoneme alignment, instead using padded text conditioning within the same flow-matching-based generation framework. This comparison is useful because it shows that, within TTS, the same flow-matching decoder principle can be embedded in substantially different architectural formulations. The general flow-matching framework underlying such models is introduced below.

3.3.2 Normalizing flow (NF) and flow matching (FM)

Normalizing flows (NF) [108, 25] are a class of generative models that learn to transform a simple prior distribution into a complex target distribution by applying a series of invertible transformations. They are trained using maximum likelihood estimation (MLE), which requires computing the determinant of the Jacobian matrix—a computationally expensive operation for high-dimensional data. Due to limitations such as the high computational cost of the Jacobian and the architectural constraints required for invertibility, Flow Matching (FM) [72, 1, 74] has been proposed as a more flexible alternative to normalizing flows.

Let $\boldsymbol{Y}_1$ be a sequence of high-dimensional vectors, e.g., spectral features, sampled from a target distribution $\boldsymbol{Y}_1 \sim q$.[1] Flow matching (FM) is a method for training generative models [72, 1, 74] by constructing a probability path p_t over time $t \in [0,1]$. This path transforms a simple prior distribution p_0 (e.g., Gaussian distribution, $p_0 \sim \mathcal{N}(0,\boldsymbol{I})$) into the target data distribution $p_1 := q$. The objective is to learn a velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t)$, which describes the direction and speed of the probability path. This velocity field is defined by the ordinary

[1] Note that flow matching literature assumes that $\boldsymbol{Y}_0$ are samples from the prior and $\boldsymbol{Y}_1$ are samples from the true data distribution. The diffusion model literature makes a different assumption where $\boldsymbol{Y}_T$ are samples from the prior and $\boldsymbol{Y}_0$ are samples from the true data distribution. This chapter follows the conventions of flow matching literature.

differential equation (ODE):

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t(\boldsymbol{Y}_t) = \boldsymbol{u}_t\left(\boldsymbol{\psi}_t(\boldsymbol{Y}_t)\right), \tag{3.26}$$

where $\boldsymbol{\psi}_t(\boldsymbol{Y}_t)$ is the time-dependent flow. Formally, the goal is to learn a velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t)$ such that its flow $\boldsymbol{\psi}_t(\boldsymbol{Y}_t)$ generates a probability path p_t satisfying $p_0 = p$ and $p_1 = q$.

To construct the probability path p_t, one intuitive approach is the *conditional optimal-transport* (COT) or *linear path* [72], which takes a linear combination of samples from the two distributions $\boldsymbol{Y}_0 \sim p_0$ and $\boldsymbol{Y}_1 \sim p_1$:

$$\boldsymbol{Y}_{t|1} = t\boldsymbol{Y}_1 + (1-t)\boldsymbol{Y}_0 \sim p_{t|1}(\cdot|\boldsymbol{Y}_1), \tag{3.27}$$

where $\boldsymbol{Y}_{t|1}$ is a realization of the flow $\boldsymbol{\psi}_t(\boldsymbol{Y}_t)$. By taking the derivative with respect to time t, the conditional velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t|\boldsymbol{Y}_1)$ can be defined as follows:

$$\boldsymbol{u}_t(\boldsymbol{Y}_t|\boldsymbol{Y}_1) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t(\boldsymbol{Y}_t) = \frac{\boldsymbol{Y}_1 - \boldsymbol{Y}_t}{1-t}. \tag{3.28}$$

The next step is to learn parameters $\boldsymbol{\theta}$ of a velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t;\boldsymbol{\theta})$ implemented by a neural network. The conditional flow matching loss is defined as follows:

$$\mathscr{L}_{\mathrm{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{Y}_t,\boldsymbol{Y}_1}\left\|\boldsymbol{u}_t\left(\boldsymbol{Y}_t;\boldsymbol{\theta}\right) - \boldsymbol{u}_t\left(\boldsymbol{Y}_t \mid \boldsymbol{Y}_1\right)\right\|^2, \text{ where } t \sim U[0,1], \boldsymbol{Y}_0 \sim p, \boldsymbol{Y}_1 \sim q. \tag{3.29}$$

By solving $\boldsymbol{Y}_t := \boldsymbol{Y}_0$ in Eq. 3.27, the loss function simplifies to:

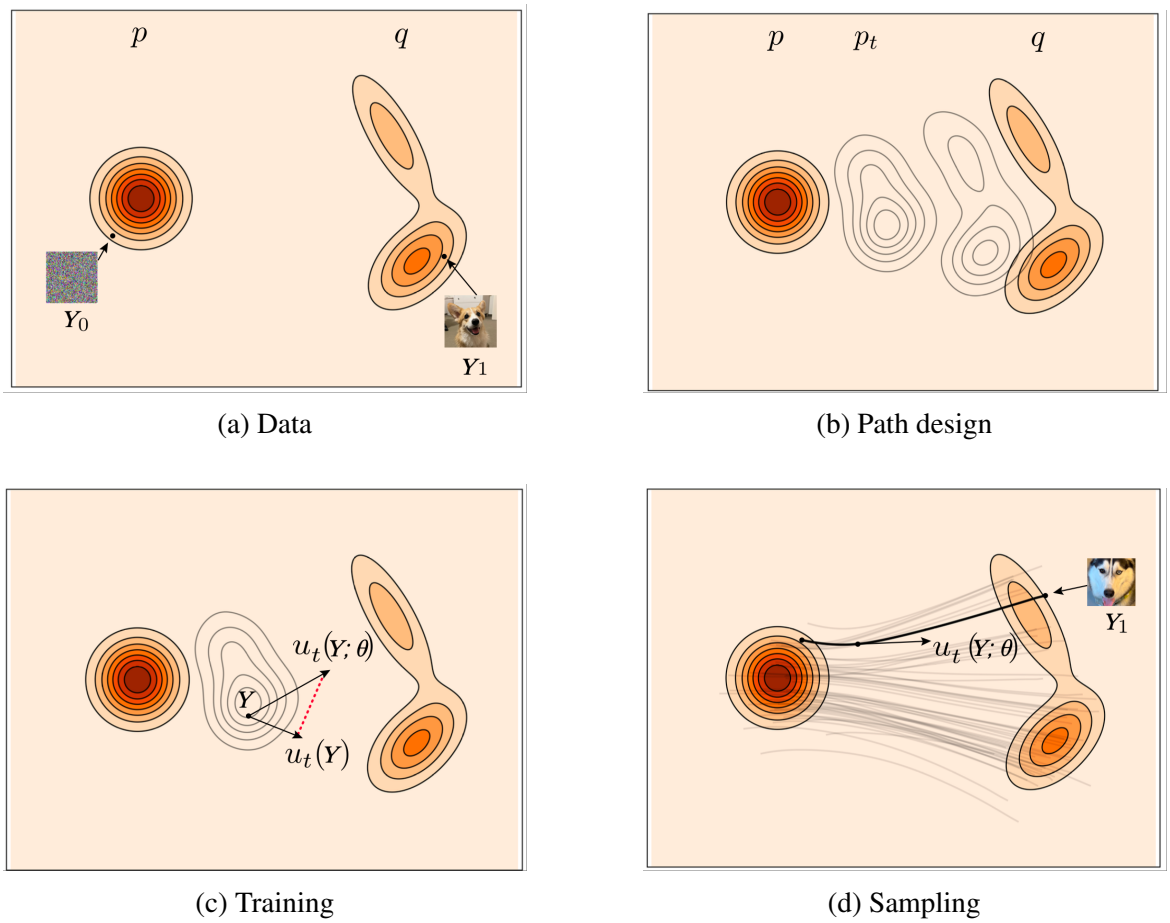
$$\mathscr{L}_{\mathrm{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{Y}_0,\boldsymbol{Y}_1}\left[\left\|\boldsymbol{u}_t(\boldsymbol{Y}_t;\boldsymbol{\theta}) - (\boldsymbol{Y}_1 - \boldsymbol{Y}_0)\right\|^2\right], \text{ where } t \sim U[0,1], \boldsymbol{Y}_0 \sim \mathscr{N}(0,\boldsymbol{I}), \boldsymbol{Y}_1 \sim q \tag{3.30}$$

This loss function can be optimized using gradient descent to learn the parameters $\boldsymbol{\theta}$ of the velocity field $\boldsymbol{u}_t(\boldsymbol{Y}_t;\boldsymbol{\theta})$. The learned velocity field can then be used to generate samples approximating to target distribution.

3.3.3 Sampling

Drawing a sample $\boldsymbol{Y}_t$ from an FM model is equivalent to solving the ODE by starting from an initial sample $\boldsymbol{Y}_0$. Numerical methods such as the Euler method or the mid-point method exist to solve the ODE. The Euler method is a simple and efficient way to compute the sample $\boldsymbol{Y}_t$:

$$\boldsymbol{Y}_{t+h} = \boldsymbol{Y}_t + h\boldsymbol{u}_t\left(\boldsymbol{Y}_t\right), h = \frac{1}{K}, K \in \mathbb{N}, \tag{3.31}$$



(a) Data

(b) Path design

(c) Training

(d) Sampling

Fig. 3.5 The Flow Matching (FM) framework. (a) shows the source distribution p and the target distribution q. (b) illustrates the design of the time-continous probability path p_t interpolating between $p := p_0$ and $q := p_1$. (c) shows the training process, where the model learns to approximate the velocity field $\boldsymbol{u}_t(\boldsymbol{Y})$ through a neural network parameterized by $\boldsymbol{\theta}$. (d) illustrates drawing a novel target sample from a novel source sample. The figures are taken from the original paper [73].

where h is the step size and K is the total number of steps. The mid-point method is a more accurate method to compute the sample $\boldsymbol{Y}_t$:

$$\boldsymbol{Y}_{t+h} = \boldsymbol{Y}_t + h\boldsymbol{u}_{t+\frac{h}{2}}\left(\boldsymbol{Y}_{mid}\right), h = \frac{1}{K}, K \in \mathbb{N}. \tag{3.32}$$

where

$$\boldsymbol{Y}_{mid} = \boldsymbol{Y}_t + \frac{h}{2}\boldsymbol{u}_t\left(\boldsymbol{Y}_t\right) \tag{3.33}$$

The Euler method is a first-order numerical solver, while the mid-point method, a type of second-order Runge-Kutta (RK) method, offers greater accuracy for the same step size h.

This improved accuracy comes at the cost of an additional function evaluation per step. While higher-order RK methods can further increase accuracy, they also increase the computational load. The choice of solver thus represents a trade-off between sample quality and inference speed.

3.4 Summary

This chapter has continued the exploration of machine learning approaches for speech synthesis started in Chapter 2 and provided an overview of the three outstanding families of probabilistic non-autoregressive models for generating high-dimensional continuous data, with a focus on their application in acoustic modeling for speech synthesis.

First, **Energy-Based Models (EBMs)** were introduced, which define a probability distribution through a learnable energy function. This chapter discussed how EBMs are trained to assign low energy to data samples from the true distribution and high energy in all other cases. Key training paradigms, including Maximum Likelihood Estimation (MLE) via MCMC, Noise-Contrastive Estimation (NCE), and Score Matching, were reviewed, along with corresponding inference techniques.

Next, **Denoising Diffusion Probabilistic Models (DDPMs)** were introduced. These models consist of a forward process that gradually adds noise to data and a learned reverse process that denoises a sample from a simple prior distribution to generate a data sample. The underlying theory, the simplified training objective, and the iterative sampling procedure were discussed. The highly popular Grad-TTS model was presented as a practical application of diffusion models and their principles in text-to-speech.

Finally, **Flow Matching (FM)**, a framework for training continuous normalizing flows is discussed. By learning a velocity field that transforms a prior distribution into the target data distribution, FM avoids the architectural constraints and expensive computations of traditional normalizing flows. The conditional flow matching objective and the use of numerical ODE solvers for sampling was briefly outlined.

In summary, EBMs, diffusion models, and flow matching represent powerful and increasingly popular approaches for generative modelling, offering distinct methodologies to tackle the challenges of generating complex, high-dimensional data such as speech.

Chapter 4

Score-based Training Methods

Score matching (SM) was briefly introduced as a training method for EBMs in Chapter 3. However, due to its practical training challenges, it was not discussed in detail. This chapter provides a more thorough examination of three score-based training methods: Score Matching (SM), Sliced Score Matching (SSM), and Denoising Score Matching (DSM). These methods provide an alternative to training techniques like Noise Contrastive Estimation (NCE).

4.1 Score matching (SM)

As discussed in Section 3.1.1, score matching is a popular method for estimating score functions of probability distributions in situations where such distributions cannot be (easily) normalized. The score function is the gradient of the log-probability density function with respect to output $\boldsymbol{Y}$ given input $\boldsymbol{x}$ and is equal to the negative gradient of the energy function $E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$. Score matching provides an alternative method for training EBMs that avoids the negative sampling required by NCE.

The score matching objective is to minimize the Fisher divergence between the model distribution $p_{\boldsymbol{\theta}}$ and the data distribution p_{data} , which can be denoted as follows:

$$\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\|\nabla_{\boldsymbol{Y}} \log p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}) - \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\|_2^2\right]. \tag{4.1}$$

However, this objective function is not computable in practice, as it involves the unknown data distribution p_{data}. By expanding the integration in the Fisher divergence [52], the data

distribution p_{data} can be bypassed as follows:

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}) =& \frac{1}{2}\int p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})\left(\nabla_{\boldsymbol{Y}}\log p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}) - \nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{Y} \\ =& \underbrace{\frac{1}{2}\int p_{\text{data}}(\bar{\boldsymbol{Y}}|\boldsymbol{x})\left(\nabla_{\boldsymbol{Y}}\log p_{\text{data}}(\bar{\boldsymbol{Y}}|\boldsymbol{x})\right)^2 \mathrm{d}\bar{\boldsymbol{Y}}}_{\text{Term 1}} + \underbrace{\frac{1}{2}\int p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})\left(\nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{Y}}_{\text{Term 2}} \\ & \underbrace{-\int p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})\cdot\frac{1}{p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})}\cdot\nabla_{\boldsymbol{Y}}p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})\cdot\nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\mathrm{d}\boldsymbol{Y}}_{\text{Term 3}}.\end{aligned} \tag{4.2}$$

Term 1 is a constant term that does not depend on the model parameters $\boldsymbol{\theta}$, and Term 3 can be simplified as follows:

$$\int p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})\nabla^2_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\mathrm{d}\boldsymbol{Y} = \mathbb{E}_{p_{\text{data}}}\left[\nabla^2_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right]. \tag{4.3}$$

Hence, the score matching objective can be rewritten as

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}) &= \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}(\nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}))^2 + \nabla^2_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})\right] + \text{constant} && (4.4)\\ &= \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}(\nabla_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}))^2 - \nabla^2_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right] + \text{constant} && (4.5)\\ &= \mathscr{J}(\boldsymbol{\theta}) + \mathrm{C} && (4.6)\end{aligned}$$

where the above made use of the following relationship

$$\underbrace{\nabla_{\boldsymbol{Y}}\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = -\nabla_{\boldsymbol{Y}}E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})}_{\text{score } \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})} \tag{4.7}$$

Eq. 4.5 indicates that the score matching objective can be expressed in terms of the energy function. In particular, its first and second order derivative with respect to the output (spectral features) $\boldsymbol{Y}$. It is possible to show that $\mathscr{J}(\boldsymbol{\theta})$ can be re-written as

$$\mathscr{J}(\boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}}\left[\text{tr}\left(\nabla_{\boldsymbol{Y}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right) + \frac{1}{2}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right\|_2^2\right] \tag{4.8}$$

where $\text{tr}(\cdot)$ denotes the trace of the Hessian. Although the trace only requires the diagonal elements of the Hessian, computing these second-order derivatives of the log-probability density is computationally expensive for high-dimensional data, even with modern automatic differentiation packages [80].

4.2 Sliced score matching (SSM)

Although proven effective, the standard form of SM has clear efficiency issues as outlined in the previous section. To address those efficiency concerns, a new variant called *Sliced Score Matching* (SSM) [114] has been proposed. The idea of SSM is to project both the score function of the underlying data distribution and the score function of the model distribution onto M random directions, and then compare the average distance over the M pairs of projections. The SSM objective using the sliced Fisher divergence can be defined as follows:

$$\mathscr{J}(\boldsymbol{\theta}; p_{\boldsymbol{v}}) = \frac{1}{2}\mathbb{E}_{p_{\boldsymbol{v}}}\mathbb{E}_{p_{\text{data}}}\left[\left(\boldsymbol{v}^\top \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) - \boldsymbol{v}^\top \boldsymbol{s}_{\text{data}}(\boldsymbol{x},\boldsymbol{Y})\right)^2\right] \tag{4.9}$$

where $p_{\boldsymbol{v}}$ is a distribution of directions $\boldsymbol{v}$ that satisfies $0 < \mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}\boldsymbol{v}^\top]$ and $\mathbb{E}_{p_{\boldsymbol{v}}}[\boldsymbol{v}^\top \boldsymbol{v}] < +\infty$. There are several choices for $p_{\boldsymbol{v}}$, such as the multivariate Rademacher distribution (a uniform distribution sampled from $\{-1,+1\}$). Using an approach similar to that for Eq. 4.8, the SSM objective can be written as:

$$\mathscr{J}(\boldsymbol{\theta}; p_{\boldsymbol{v}}) = \mathbb{E}_{p_{\boldsymbol{v}}}\mathbb{E}_{p_d}\left[\boldsymbol{v}^\top \nabla_{\boldsymbol{Y}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\boldsymbol{v} + \frac{1}{2}\left(\boldsymbol{v}^\top \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right)^2\right]. \tag{4.10}$$

Given a dataset $\{(\boldsymbol{x}_1,\boldsymbol{Y}_1)\ldots,(\boldsymbol{x}_N,\boldsymbol{Y}_N)\}$ and random direction vectors $\{\boldsymbol{v}_{ij}\}_{1\le i\le N, 1\le j\le M}$, the SSM objective can be approximated as follows:

$$\mathscr{J}\left(\boldsymbol{\theta}; \boldsymbol{v}_{11}^{NM}\right) = \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{v}_{ij}^\top \nabla_{\boldsymbol{Y}} \boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i,\boldsymbol{Y}_i\right)\boldsymbol{v}_{ij} + \frac{1}{2}\left(\boldsymbol{v}_{ij}^\top \boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i,\boldsymbol{Y}_i\right)\right)^2 \tag{4.11}$$

Note that if $p_{\boldsymbol{v}}$ is a multivariate Rademacher or a multivariate standard Gaussian distribution, the following holds:

$$\mathbb{E}_{p_{\boldsymbol{v}}}\left[\left(\boldsymbol{v}^\top \boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x},\boldsymbol{Y}\right)\right)^2\right] = \left\|\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x},\boldsymbol{Y}\right)\right\|_2^2 \tag{4.12}$$

Hence

$$\mathscr{J}\left(\boldsymbol{\theta}; \boldsymbol{v}_{11}^{NM}\right) = \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{v}_{ij}^\top \nabla_{\boldsymbol{Y}} \boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i,\boldsymbol{Y}_i\right)\boldsymbol{v}_{ij} + \frac{1}{2}\left\|\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i,\boldsymbol{Y}_i\right)\right\|_2^2 \tag{4.13}$$

The algorithm for computing the approximated SSM loss is presented in Alg. 3.

As shown in Alg. 3, the first-order derivative of the log-probability density function (*score*) is computed first. Then, the higher-order derivative is obtained. Both gradient computations are made possible thanks to the automatic differentiation support in modern deep learning packages, such as `torch` used in this thesis.

Algorithm 3 Sliced Score Matching

Require: $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})$, data $(\boldsymbol{x},\boldsymbol{Y})$, random projection $\boldsymbol{v}$

1: Compute score:

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) \leftarrow \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})$$

2: Project score and compute directional gradient:

$$\boldsymbol{v}^\top \nabla_{\boldsymbol{Y}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) \leftarrow \nabla_{\boldsymbol{Y}} \left(\boldsymbol{v}^\top \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) \right)$$

3: Initialize loss:

$$J \leftarrow \frac{1}{2} \|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\|_2^2$$

4: Add trace (Hessian-vector product) term:

$$J \leftarrow J + \boldsymbol{v}^\top \nabla_{\boldsymbol{Y}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}) \boldsymbol{v}$$

return J

4.3 Denoising score matching (DSM)

Fig. 4.1 Adding perturbation noise to an image in DSM.

One important condition for $\log p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})$ in Eq. 4.4 is that it should be differentiable and finite [117]. This strict condition makes it difficult to apply score matching to real-world data, as the data distribution is often not differentiable or finite. For example, the value range of features $\boldsymbol{Y}$ is often discontinuous. In case of images, each element $Y_{i,j}$ could belong to one of $0,\ldots,255$ discrete classes and be undefined elsewhere, whereas spectral features $Y_{i,j}$ belong to $(-\infty,0]$ range. To address this issue, *Denoising Score Matching* (DSM) [130] was proposed to estimate the score function of the data distribution by adding noise $\boldsymbol{\varepsilon}$ to the data:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{Y} + \boldsymbol{\varepsilon} \tag{4.14}$$

To minimize the Fisher divergence between the model distribution $p_{\boldsymbol{\theta}}$ and the data distribution p_{data}, DSM uses a noise-conditional score matching objective:

$$\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{q(\tilde{\boldsymbol{Y}}|\boldsymbol{x})} \left[\left\| \nabla_{\boldsymbol{Y}} \log q(\tilde{\boldsymbol{Y}}|\boldsymbol{x}) - \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\tilde{\boldsymbol{Y}}|\boldsymbol{x}) \right\|_2^2 \right]. \tag{4.15}$$

where

$$q(\tilde{\boldsymbol{Y}}|\boldsymbol{x}) = \int q(\tilde{\boldsymbol{Y}} \mid \boldsymbol{Y},\boldsymbol{x}) p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x}) d\boldsymbol{Y} \tag{4.16}$$

is the marginal distribution of the noisy data $\tilde{\boldsymbol{Y}}$ given $\boldsymbol{x}$.

This new objective appears to create a mismatch, as the model learns the score of the noisy data distribution $q(\tilde{\boldsymbol{Y}}|\boldsymbol{x})$ instead of the original data distribution $p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})$. To avoid this inconsistency, a small noise perturbation $\boldsymbol{\varepsilon}$ is used to ensure that the noisy data $\tilde{\boldsymbol{Y}}$ remains close to the original data $\boldsymbol{Y}$. One solution is to add Gaussian noise $\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ to the data $\boldsymbol{Y}$:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{Y} + \sigma_i \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.17}$$

where $\sigma_{1:L}$ is a sequence with length L that satisfies

$$\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1 \tag{4.18}$$

with σ_L being a small value. Under these conditions the noise perturbation $\boldsymbol{\varepsilon}$ becomes small enough to ensure that the noisy data $\tilde{\boldsymbol{Y}}$ is close to the original data $\boldsymbol{Y}$. A neural network parameterized by $\boldsymbol{\theta}$ is trained to ensure the score function $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}, i) \approx \nabla_{\boldsymbol{Y}} \log q_{\sigma_i}(\boldsymbol{Y}|\boldsymbol{x})$ for all $i \in [1, L]$. The DSM objective can be written as:

$$\mathscr{L}(\boldsymbol{\theta}; \boldsymbol{\sigma}) = \frac{1}{2} \sum_{i=1}^{L} \lambda_i \mathbb{E}_{q_{\sigma_i}(\tilde{\boldsymbol{Y}}|\boldsymbol{x})} \left[\left\| \nabla_{\boldsymbol{Y}} \log q_{\sigma_i}(\tilde{\boldsymbol{Y}}|\boldsymbol{x}) - \boldsymbol{s}_{\theta}(\boldsymbol{x}, \tilde{\boldsymbol{Y}}, i) \right\|_2^2 \right] \tag{4.19}$$

where λ_i is a positive weighting function that controls contribution of each noise level to the overall objective, and is usually set to $\lambda_i = \sigma_i^2$. Since the noise perturbation $\boldsymbol{\varepsilon}$ is sufficiently small, the log-probability noise density function is:

$$\nabla_{\tilde{\boldsymbol{Y}}} \log q_{\sigma_i}(\tilde{\boldsymbol{Y}} \mid \boldsymbol{Y}, \boldsymbol{x}) = -(\tilde{\boldsymbol{Y}} - \boldsymbol{Y}) / \sigma_i^2 \tag{4.20}$$

Using the above equation, the DSM objective in Eq. 4.19 can be simplified as follows:

$$\mathscr{L}(\boldsymbol{\theta}; \boldsymbol{\sigma}) = \frac{1}{2L} \sum_{i=1}^{L} \left\| \frac{(\tilde{\boldsymbol{Y}} - \boldsymbol{Y})}{\sigma_i} - \sigma_i \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \tilde{\boldsymbol{Y}}, i) \right\|_2^2 \tag{4.21}$$

which provides a simple and elegant expression for optimisation.

4.3.1 Inference

After a denoising score matching model is trained, Langevin dynamics (Eq. 3.15) can be used to sample from the noise distribution $q_{\sigma_1}(\tilde{\boldsymbol{Y}}|\boldsymbol{x})$.

$$\tilde{\boldsymbol{Y}}_t \leftarrow \tilde{\boldsymbol{Y}}_{t-1} + \frac{\alpha_i}{2}\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}, \tilde{\boldsymbol{Y}}_{t-1}, i\right) + \sqrt{\alpha_i}\boldsymbol{Z}_t \tag{4.22}$$

where the step size $\alpha_i = \beta \cdot \sigma_i^2/\sigma_L^2$, β is a hyperparameter with small positive value, and $\boldsymbol{Z}_t \sim \mathcal{N}(0, I)$. Finally, the sample drawn from the noise distribution $q_{\sigma_L}(\tilde{\boldsymbol{Y}}|\boldsymbol{x})$ is expected to be close to $p_{\text{data}}(\boldsymbol{Y}|\boldsymbol{x})$ after L iterations.

4.3.2 Connection to DDPM

DSM establishes a strong connection between score-based models and diffusion models [118, 45]. Both DSM and DDPM are denoising and time-dependent generative models. The variational bound for training a DDPM is equivalent to a weighted form of the DSM objective [45]. Furthermore, both DSM and DDPM can be unified within a framework of stochastic differential equations (SDEs) [118].

However, they differ in their approaches to noise perturbation and inference. From a theoretical perspective, DSM is developed to estimate the score function of the data distribution. This contrasts with the DDPM approach, which reverses the noising process step-by-step to recover the original data. Consequently, DSM samples from a perturbed data distribution with a decreasing standard deviation, whereas DDPM calculates the noise at each step using approximation methods. During inference, DSM uses Langevin dynamics to sample from the perturbed data distribution, which is analogous to the reverse diffusion process in DDPM.

4.4 Summary

This chapter introduced three score-based training methods for generative models: Score Matching (SM), Sliced Score Matching (SSM), and Denoising Score Matching (DSM). These methods provide an alternative to training techniques like Noise Contrastive Estimation (NCE) by directly estimating the score function of the data distribution, thereby avoiding the computationally expensive partition function of Energy-Based Models.

Score Matching (SM) was presented as the foundational method. Since the *score* is related to *energy*, the SM training method can be seen as an extension of EBM training that minimizes the Fisher divergence between the model and data distributions. However, its

practical application is limited by the high computational cost of calculating the trace of the Hessian matrix for high-dimensional data.

To address this limitation, Sliced Score Matching (SSM) was introduced. SSM offers a more scalable solution by projecting the score functions onto random vectors and computing Hessian-vector products instead of the full Hessian trace, making the training process significantly more efficient.

Finally, Denoising Score Matching (DSM) was discussed as a method to handle data distributions that are not differentiable or are defined on a low-dimensional manifold. By adding small noise to the data, DSM trains a model to predict the score of the perturbed data distribution, which simplifies the objective to a denoising task. Inference is then performed using Langevin dynamics to generate samples. This method establishes a strong connection between score-based models and diffusion models.

Chapter 5

Noise Contrastive Estimation Training for Energy-based TTS Models

5.1 Introduction

Previous chapters introduced energy-based models (EBMs) as probabilistic non-autoregressive acoustic models and identified noise contrastive estimation (NCE) as one possible training criterion. They also motivated residual EBMs as a potential solution for addressing *exposure bias* [104] in autoregressive TTS: rather than modifying the autoregressive TTS (e.g., scheduled sampling [8] and attention mechanisms [40]), an EBM can be trained to assign lower energy to reference-like speech and higher energy to imperfect hypotheses, and then used to refine hypotheses generated by a base model. This differs from many earlier applications of EBMs to discrete-output tasks. For text generation [7], machine translation [10], or speech recognition [70], an EBM is often used mainly as a re-ranking model: a separate system first produces a finite list of candidate sequences, and the EBM assigns an energy score to each candidate so that better hypotheses can be selected. This difference makes negative-sample design more important, since the negatives should provide useful directions for refinement rather than only support a coarse ranking decision. This chapter therefore focuses on whether NCE can be used in practice to train such a residual EBM for continuous acoustic features, and on what kinds of negative speech samples make this training effective.

Since Chapters 3.1 and 3.2 have already introduced EBMs and diffusion models, this chapter only recalls the connection, which is a similar iterative algorithm. Although it is possible to formulate the conditional probability distribution of speech given text for EBMs, the intractable normalisation term would make training and inference approaches relying on the probability distribution infeasible. Both methods can be used to improve an

initial acoustic representation through an iterative procedure. In diffusion TTS systems such as Grad-TTS [98], the iterations correspond to a learned denoising process. In the EBM framework considered here, the iterations instead follow the gradient of an energy function, for example using Langevin MCMC [92, 36]. This connection motivates comparing EBM inference with diffusion-style refinement, while keeping the focus of this chapter on the NCE training problem. In particular, because NCE depends on contrasting reference speech with imperfect speech, the quality of negative samples becomes the central practical issue for training EBMs on continuous TTS outputs. This chapter describes a number of effective strategies to generate negative examples, including by means of existing TTS models.

This chapter makes the following specific contributions:

1. first energy-based text-to-speech model;
2. a range of methods for generating effective negative samples to use in noise contrastive estimation and elsewhere;
3. link between diffusion models and energy-based models.

The rest of this chapter is organized as follows. Section 5.2 describes energy-based models (EBM) for TTS, which includes inference, training and negative sampling methods. Section 5.3 relates EBMs to filtering approaches and diffusion models. Experimental results and discussion are presented in Section 5.5. Conclusions drawn from this work and future research directions are presented in Section 5.6.

5.2 Energy-based models

Given a text sequence $\boldsymbol{x}$, an energy-based model (EBM) of speech feature sequences $\boldsymbol{Y}$ (e.g. log-Mel spectrograms) can be defined by[1]

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})} \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})\right), \tag{5.1}$$

where $\boldsymbol{\theta}$ are model parameters, $E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$ is an energy between text and speech, $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a normalisation term. Unlike speech signal energies commonly used in models like FastSpeech 2, EBM energies $E_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$ reflect the correspondence between text $\boldsymbol{x}$ and speech $\boldsymbol{Y}$ pairs rather than quantify the amount of spectral energy contained in a frame of speech signal. Better matching pairs are expected to yield lower energies and *vice versa*. The normalising term

[1] An alternative formulation would involve parameterising the gradient of energy instead. Such an approach is possible due to existence of inference and training approaches that rely only on the gradient of energy.

$Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable to compute exactly as it requires integration over all possible acoustic realisations $\boldsymbol{Y}$. Thus, only certain inference and parameter estimation approaches can be used for EBMs.

5.2.1 Inference

As was first discussed in Section 3.1, when $\boldsymbol{Y}$s are continuous variables, inference with EBMs can be viewed as a hypothesis refinement process using Langevin Markov Chain Monte-Carlo (MCMC). The Langevin MCMC is an iterative process where, given an initial hypothesis $\boldsymbol{Y}^{(0)}$, the final hypothesis, $\boldsymbol{Y}^{(N)}$ is obtained by executing the following update rule

$$\boldsymbol{Y}^{(n+1)} \leftarrow \boldsymbol{Y}^{(n)} - \lambda \, \nabla_{\boldsymbol{Y}} E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})|_{\boldsymbol{Y}=\boldsymbol{Y}^{(n)}} + \sqrt{2\lambda}\boldsymbol{Z}^{(n)}, \tag{5.2}$$

where $\boldsymbol{Z}^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \mu\boldsymbol{I})$, λ is an update rate and μ is commonly set to 1.

The need to specify initial hypotheses $\boldsymbol{Y}^{(0)}$ offers a number of interesting options. In the standard Langevin MCMC initial hypotheses are drawn from a simple prior distribution, such as Gaussian [51]. However, more informative priors, such as high-performing TTS models (e.g. Tacotron 2 and FastSpeech 2), can also be explored.

5.2.2 Training

Since the normalising factor $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is intractable, approaches relying on $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})$ can not be used with EBMs. Furthermore, popular gradient-based MCMC approaches [28] can not be applied with discrete input, as in this work (text), and Gibbs sampling [136] would be too computationally expensive. Fortunately, noise contrastive estimation (NCE) [79] provides a feasible solution for optimizing general energy functions, including those considered in this work. The NCE loss function for EBMs in eq. (5.1) is given by

$$\begin{aligned} \mathscr{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{Y}^{-}) = &- \log\left(\frac{1}{1+\exp\left(E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\right)}\right) \\ &- \log\left(\frac{1}{1+\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{-})\right)}\right) \end{aligned} \tag{5.3}$$

where $\boldsymbol{Y}^{+}$ are called positive samples and $\boldsymbol{Y}^{-}$ are called negative samples. According to eq. (5.3), energy functions are optimal when high energy is assigned to negatives and low energy is assigned to positives. Once trained, energy functions can be used for ranking

hypotheses generated by other TTS models or inferring hypotheses using the Langevin MCMC in eq. (5.2).

5.2.3 Positive and negative sampling methods

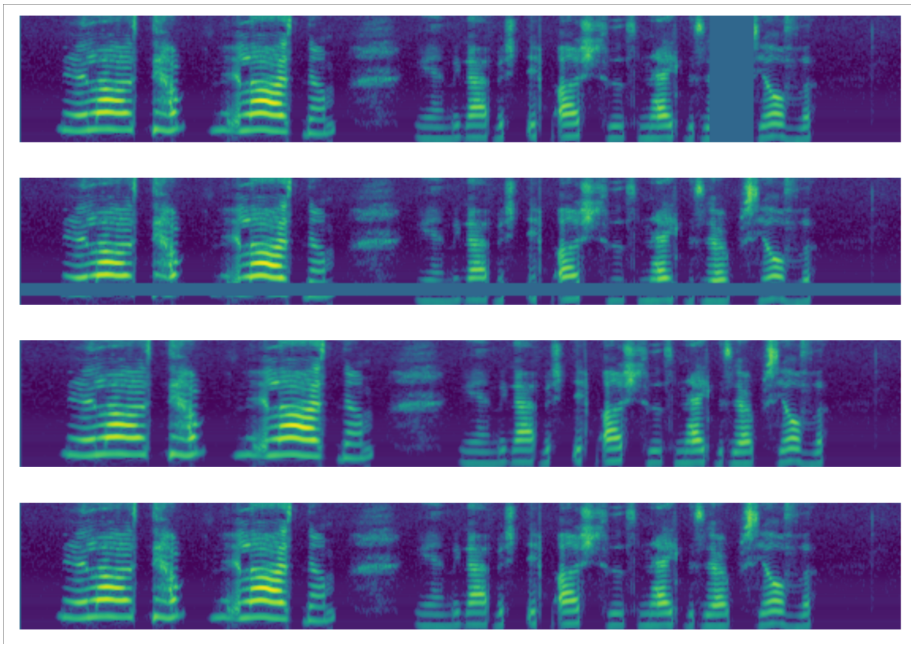


Fig. 5.1 SpecAugment methods. From the bottom to the top shows log-Mel spectrograms without augmentation, with time warpping, with frequency masking, and with time masking. The figure is taken from the original paper [93].

Positive samples in NCE are usually represented by reference sequences. In TTS these will be log-Mel spectrograms of ground truth waveforms from a given dataset. On the other hand, negative samples are not prescribed and need to be designed. In text generation [7] it is argued that negative examples with poor quality make the task of learning the energy function easier which leads to poor quality energy functions. This work proposes using pre-trained TTS models as a key ingredient of generating high-quality negative examples. Note that when a pre-trained model is used as a part of training process then it is also possible to adopt it for initialising the Langevin MCMC in eq. 5.2, which is expected to speed up inference and lead to higher quality hypotheses. The key drawback is the need of a pre-trained model and possibility that EBMs might, although do not have to, inherit some of the wrong modelling assumptions.

The simplest method to generate negative samples from pre-trained TTS models would use hypotheses generated by those models directly. Such approach may fail to work due to

high level of similarity between high-quality hypotheses and reference sequences. Other possible options include applying random masking (RM) and SpecAugment [93]:

- time masking (TM)
- frequency masking (FM) and
- time warping (TW))

to those hypotheses. The TM and FM methods can be seen as specific cases of the RM method and may prove less effective. For example, the FM may drastically affect pitch information by masking a whole frequency range. The TW method is also expected to face challenges as utterance-wide shortening/elongation of all sounds may be hard to separate from reference sequences.

5.2.4 Architecture

The architecture of EBM explored in this work is inspired by Transformer TTS [68] and is shown in Figure 5.2. The EBM in Fig. 5.2 consists of two blocks: energy estimator (top) and feature enhancement (bottom). The goal of the energy estimator is to derive utterance-level EBM energies $E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})$ from the output of the feature enhancement block. This work assumes that these energies can be derived from frame-level EBM energies. As shown in Fig. 5.2, the energy estimator consists of two key elements: frame-level EBM energy estimation and frame-level EBM energy weighting. The latter element is motivated by an intuition that frame-level EBM energies are unlikely to make equally important contributions. For example, speech and non-speech frames will likely make different contributions to the utterance-level EBM energy

$$E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}) = \sum_{t=1}^{T} \alpha_t e_t \tag{5.4}$$

where $\alpha_{1:T}$ is the sequence of attention weights generated by the EBM energy weighting module, $e_{1:T}$ is the sequence of frame-level EBM energies and T is the number of frames. The attention weights are derived from the frame-level EBM energies. The frame-level EBM energies $e_{1:T}$ are computed by

$$e_t = \boldsymbol{a}^\top \boldsymbol{g}_t + b \tag{5.5}$$

where $\boldsymbol{g}_t$ is the output of the feature enhancement block, and $\boldsymbol{a}$ and b are parameters of the frame energy module. The goal of the feature enhancement block is to enhance typically short-term spectral information available in standard speech features, such as log-Mel spectrograms,

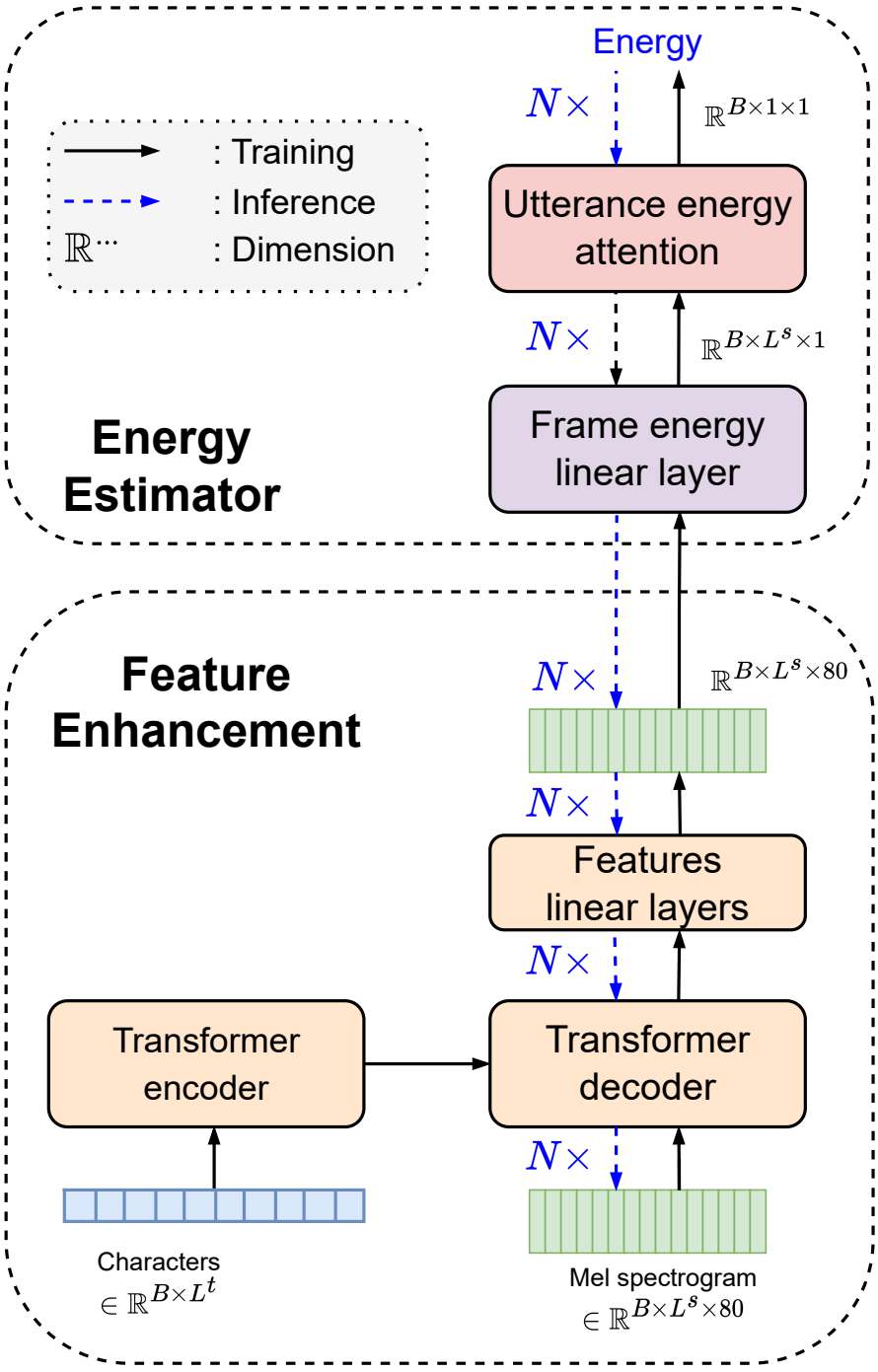


Fig. 5.2 Architecture of EBMs examined in this work (inspired by Transformer TTS).

with more advanced acoustic and linguistic information. The estimator is a transformer-based [68] model using text as the input to encoder, and spectral features (due to auto-regression and teacher forcing) and the output of encoder as the input to the decoder. The output of decoder after linear transformation, $\boldsymbol{g}_t$, is passed to the energy estimator block. Note that the decoder does not use masking to constrain the underlying attention mechanism from attending over previous spectral features, which makes $\boldsymbol{g}_t$ a function of entire text and spectral feature sequences.

5.3 Related work

EBMs have been applied in a wider range of domains, e.g. natural language processing [7, 21] and automatice speech recognition [70]. The EBM proposed in this work can be related to a number of previously proposed approaches in TTS. The use of hypotheses generated by pre-trained TTS models as a part of training and inference allows to connect this EBM to post-filtering methods. Statistical post-filtering approaches, such as [123], aim to address over-smoothing in hypotheses generated by statistical speech synthesis models. However, these approaches suffer from difficulties in accurately modelling probability density functions of the underlying speech parameterisations. Recently, there has also been interest in deep learning based post-filtering approaches. In [58], frequency band specific generative adversarial networks (GAN) were trained to improve the quality of hypotheses generated by deep learning based speech synthesis models. However, this approach assumes independence among frequency bands which may lead to suboptimal results.

5.3.1 Connection to diffusion models

More recently, there has been a lot of interest in diffusion-based TTS models [98] discussed in Section 3.2, which have been extended to audio synthesis [95] and singing voice synthesis [140]. In these models training (forward) and inference (reverse) processes iteratively build a connection between data and noise. Although seemingly different, such diffusion models and the EBM proposed in this work have clear connections.

Consider, for example, the iterative inference process used by one of those (continuous space) denoising score matching models [118]

$$\boldsymbol{Y}^{(n+1)} \leftarrow \frac{1}{\sqrt{1-\lambda_n}}(\boldsymbol{Y}^{(n)} + \lambda_n S_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{(n)}, n)) + \sqrt{\lambda_n}\boldsymbol{Z}^{(n)}, \tag{5.6}$$

where $0 < \lambda_n < 1$. Compared to the Langevin MCMC in eq. (5.2) the key difference stems from modelling iteration, n, specific $S_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{(n)}, n)$ score (gradient of log-likelihood) rather than the iteration independent $\nabla_{\boldsymbol{Y}^{(n)}} E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{(n)})$ score of the EBM given by eq. (5.1). In addition, score matching approaches to training diffusion models can also be adopted with EBMs [118], which further strengthens the connection between these models.

Consider now the inference process of (discrete space) denoising diffusion probabilistic models (DDPM) in Eq. 3.25, which is reproduced here for convenience:

$$\boldsymbol{Y}_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}}_{\text{Term 1}} \boldsymbol{Y}_t - \underbrace{\frac{\beta_t}{\sqrt{\alpha_t}\sqrt{(1-\bar{\alpha}_t)}}}_{\text{Term 2}} \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t) + \underbrace{\sigma_t}_{\text{Term 3}} \boldsymbol{Z} \tag{5.7}$$

where $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t)$ can be shown to be proportionate to the score [117]. Let γ, η, and ξ represent Term 1, Term 2, and Term 3 respectively. The above equation can then be rewritten as:

$$\boldsymbol{Y}_{t-1} = \gamma \boldsymbol{Y}_t - \eta \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t) + \xi \boldsymbol{Z}, \tag{5.8}$$

Because $\gamma > 1$ ($\alpha < 1$), this equation can be rewritten as:

$$\boldsymbol{Y}_{t-1} = \underbrace{\boldsymbol{Y}_t - \eta \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{Y}_t, t)}_{\text{Term } a} + \underbrace{(\gamma - 1)\boldsymbol{Y}_t}_{\text{Term } b} + \underbrace{\xi \boldsymbol{Z}}_{\text{Term } c} \tag{5.9}$$

Equation 5.9 can be seen as an instance of a gradient descent algorithm, where Term c is an additional denoising term and Term b is a momentum term with hyperparameter $\gamma - 1$ to accelerate convergence. Comparing Eq. 5.9 with Eq. 5.2, aside from the iteration-independent function in the latter, both can be viewed as instances of gradient descent.

5.4 Experimental setup

5.4.1 Dataset

The dataset used in this work is LJSpeech [54], which includes 13,100 audio clips totalling approximately 24 hours from one female speaker. The dataset is split randomly into training (10,000 clips), validation (1800 clips) and test (1300 clips) sets. Since LJSpeech contains only 24 hours of single-speaker speech recordings, the models' generalisation ability should be examined on larger multi-speaker datasets. Thus, the additional datasets used are train-clean-100 and train-clean-360, which are two subsets of LibriTTS introduced in Section 2.5.3. There are approximately 245 hours of speech, 149,736 samples, and 1,151 speakers in total.

The datasets are randomly split into 124,736 samples for training, 10,000 for validation, and 15,000 for testing. The same experiments are conducted on these two datasets, but only for objective evaluation. Objective evaluation is performed over the entire test set whilst subjective evaluation is performed over 100 randomly chosen test set clips. Front-end pre-processing of audio follows the open-source implementation available as a part of NVIDIA's Tacotron 2. [2]

5.4.2 Models

The pre-trained TTS model providing hypotheses for training and inference is the open-source implementation [3] of Tacotron 2 [112] by NVIDIA. The default provided configuration was adopted in this work.

The structure of EBM discussed in Section 5.2.4 follows the corresponding elements of Transformer-TTS [68] available through an open-source implementation [4] except that:

1. positional and character embeddings are 256-dimensional;
2. two EBMs with different number of hidden features, 128 and 256 respectively, are explored in the study.

Utterance-level energy is predicted by frame-level EBM energy prediction module, which consists of two 512-dimensional fully-connected layers. Although it is possible to backpropagate gradients through the pre-trained TTS model, for simplicity this was not explored in this work. Both EBMs are trained for 125K iterations using Adam optimizer with a batch size of 16 and a constant learning rate of 1×10^{-4} on a single NVIDIA 3090 GPU. The number of parameters of these 2 EBMs and Tacotron 2 are shown in Table 5.1. We use an open-source implementation[5] of the WaveGlow [99] vocoder and adopt its default settings.

5.4.3 Evaluation

Mel cepstral distortion (MCD), F0 frame error (FFE) and log-scale F0 root mean square error (log F0 RMSE) are adopted as objective metrics in this work. The MCD metric calculates distance between cepstral coefficient sequences of different lengths on the Mel frequency scale. The FFE metric measures discrepancy of fundamental frequency (F0) between synthesized and reference waveforms. In all these cases, a dynamic time warping

[2]https://github.com/NVIDIA/tacotron2
[3]https://github.com/NVIDIA/tacotron2
[4]https://github.com/soobinseo/Transformer-TTS
[5]https://github.com/NVIDIA/waveglow

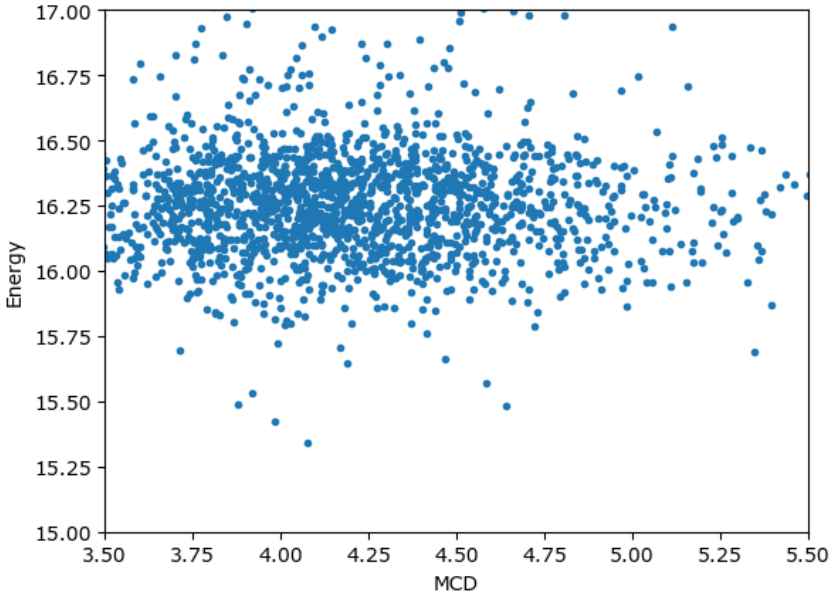


Fig. 5.3 Energy density of negative samples from Tacotron 2 as a function of their MCD.

(DTW) is used to align the predicted Mel-spectrogram and the reference. FAIRSEQ S^2 toolkit [135] is used to compute MCD and FFE scores. Mean opinion score (MOS) evaluation is conducted to evaluate speech naturalness by scoring each speech sample on a scale between 1 to 5 with 1 point intervals. Waveforms synthesized by 3 models compared in this work are mixed with test set waveforms. To obtain five independent ratings for each audio sample, the evaluation contains 2,000 ratings in total (4 models $\times$ 100 test clips $\times$ 5 ratings). With each listener evaluating one set of 100 audio clips, this design requires 20 listeners, which is consistent with the participant numbers discussed in Section 2.6.1.

5.5 Experiments

5.5.1 Initial study

Table 5.1 compares Tacotron 2 and two initial EBMs. These EBMs were trained using Tacotron 2 generated hypotheses as negative samples and a single step ($N = 1$) Langevin MCMC, where μ was set to 0 for simplicity and Adam rather than gradient descent update rule was adopted. Both large (256 hidden features) and small (128 hidden features) EBMs perform slightly better than the baseline. On LibriTTS, the large EBM achieves the best objective scores among the three models, but all LibriTTS results are worse than the corresponding

Dataset	Model	MCD ↓	FFE ↓	$\log f_0$ ↓	Parameters
LJSpeech	Tacotron 2	4.218	47.31%	0.292	28.19M
	EBM$^{(1)}$ (small)	4.163	47.06%	0.289	2.30M
	EBM$^{(1)}$ (large)	4.178	47.05%	0.291	7.64M
LibriTTS	Tacotron 2	5.591	48.72%	0.318	28.19M
	EBM$^{(1)}$ (small)	5.503	48.49%	0.315	2.30M
	EBM$^{(1)}$ (large)	**5.476**	**48.37%**	**0.313**	7.64M

Table 5.1 Comparison between Tacotron 2 and two EBMs utilizing Tacotron 2 hypotheses as negative samples.

LJSpeech results, suggesting that the more variable LibriTTS setting is more challenging for this initial EBM configuration.

Figure 5.3 shows the energy density of negative samples from the base model (Tacotron 2) as a function of their MCD. The energy is the output from the large EBM. The figure shows that there is no strong correlation between energy and MCD. This might mean that the EBM might not be able to provide a nuanced discrimination among high-quality samples. It could also mean that the MCD might not accurately capture the quality of generated speech.

The energy density before and after training the large EBM is shown in Figure 5.4 and Figure 5.5, respectively. These figures show that the energy distributions are well-separated after training. As expected, the EBM learns to assign low energies to positive samples and high energies to negative samples.

5.5.2 Negative sampling methods

Table 5.2 summarises performance of the alternative negative sampling methods (see Sec. 5.2.3) with the large EBM, where the simplified Langevin MCMC was run for $N = 100$ steps. Many of these methods show significantly better performance than the baseline. Comparing between compressed and stretched spectral features suggest no strong preference for any particular method of time warping (TW). The method of 5% time masking (TM) achieves lower MCD and FFE compared to other time masking methods, while the trend is opposite for frequency masking (FM), where higher percentage points (15%) appear to be yielding better MCD results and worse FFE results. The likely reason is the negative interaction between FFE and frequency masking.

Table 5.3 investigates the impact of the Langevin MCMC steps on the performance of the best system in Table 5.2. As the number of steps increases, the EBM applying 25% random masking to negative samples generally performs better on both datasets. The

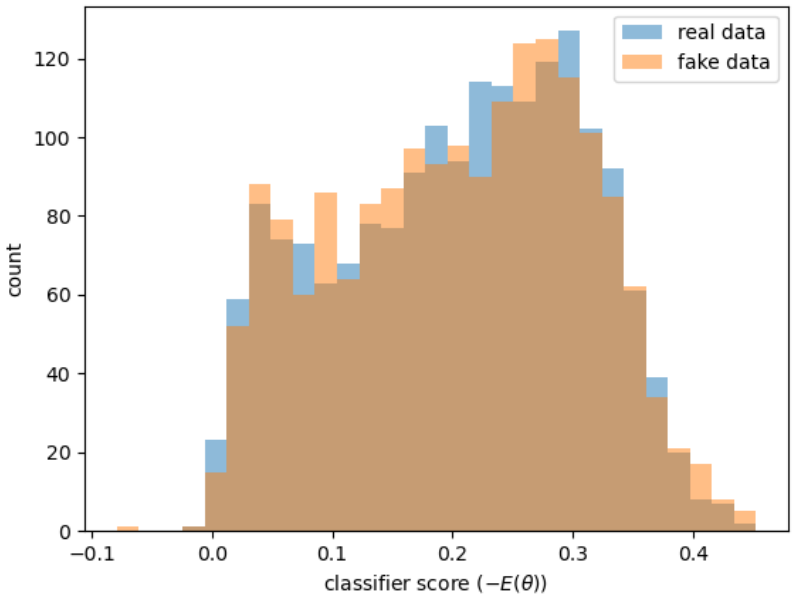


Fig. 5.4 Energy distribution before training.

improvement is modest after one step but becomes clearer after 100 and 300 steps, suggesting that NCE-trained EBMs require relatively long iterative refinement to produce consistent gains.

Although random masking appears to be the most effective negative sampling method, the other masking methods may bring additional complementary information. Table 5.4 summarises performance of different combination approaches involving the RM 30% EBM in Table 5.2. Table 5.4 shows that adding TM, FM, or TW to RM improves performance on both datasets, with FM giving the strongest FFE gains and TW giving the strongest MCD and log f_0 gains. Combining all four masking methods gives the best overall results, although LibriTTS remains more challenging than LJSpeech.

5.5.3 Subjective evaluation

To solicit subjective assessment, a range of listening tests were conducted (see Sec. 5.4.3). Table 5.5 shows that the proposed EBM shows generally better MOS scores than the baseline Tacotron 2. Furthermore, the detailed breakdown of MOS score counts in Table 5.6 shows that the EBM significantly reduced the number of MOS scores 2 (-2) and 3 (-27) and increase the number of MOS scores 4 (+26) and 5 (+3).

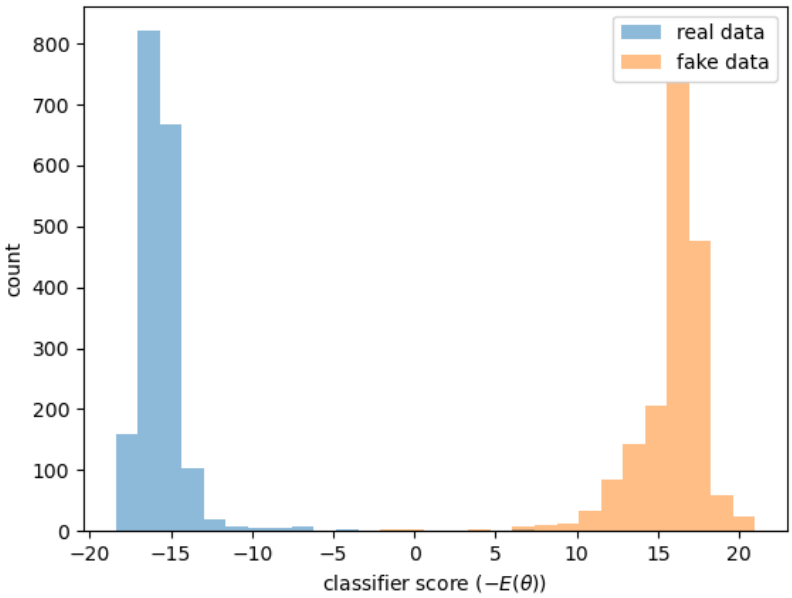


Fig. 5.5 Energy distribution after training.

5.6 Summary

5.6.1 Overview

This chapter proposed a new class of non-autoregressive (non-AR) text-to-speech (TTS) models called energy-based models (EBM). It showed how powerful forms of EBMs can be designed by adopting architectures of state-of-the-art AR models like Transformer TTS. Although training models like EBMs is more complicated due to the intractability of normalisation terms, a range of training approaches is available. This chapter described how one such approach called noise contrastive estimation (NCE) can be adopted for training. As the NCE critically relies on the quality of negative samples used to contrast reference speech feature sequences, this chapter proposed and evaluated a wide range of negative sampling methods. It found that random masking is the single best method but the combination of all proposed methods yielded the best performance.

This chapter also showed how sampling from EBMs can be performed by means of Langevin Markov Chain Monte-Carlo (MCMC). Since Langevin MCMC is closely linked with an iterative method used by popular diffusion models, this chapter briefly discussed similarities between EBMs and diffusion models. The work in this chapter was concluded by subjective evaluation and found that the proposed EBM model provides improvements over Tacotron 2.

Dataset	Condition		MCD ↓	FFE ↓	$\log f_0$ ↓
LJSpeech	**TM:**	5%	4.149	46.89%	0.290
		10%	4.166	46.98%	0.284
		15%	4.161	47.30%	0.285
	FM:	5%	4.166	46.88%	0.292
		10%	4.138	47.27%	0.286
		15%	4.097	47.35%	0.284
	TW:	1.2 (compress)	4.134	47.05%	0.291
		1.1 (compress)	4.170	47.03%	0.291
		0.9 (stretch)	4.168	47.28%	0.284
		0.8 (stretch)	4.159	47.29%	0.286
	RM:	25%	**3.943**	**46.16%**	0.282
		30%	4.013	46.57%	**0.280**
		Baseline	4.218	47.31%	0.292

Table 5.2 Negative sampling methods on LJSpeech (95% confidence intervals).

5.6.2 Limitations and future work

The work presented in this chapter has a number of limitations. The first is that inference was explored using only one of many possible choices. The use of Adam optimizer, albeit a popular choice among gradient-based optimizer, enforces the view on inference with EBMs as an optimization problem rather than sampling as advocated by Langevin MCMC and alike. Furthermore, other optimizers (e.g., gradient descent (GD) and AdamW) were not explored and may lead to suboptimal results in terms of speech quality.

Dataset	Step	MCD↓	FFE↓	$\log f_0$ ↓
LJSpeech	0	4.218	47.31%	0.292
	1	4.217	47.31%	0.292
	100	3.943	46.16%	0.282
	300	**3.937**	**45.85%**	**0.276**
LibriTTS	0	5.591	48.72%	0.318
	1	5.588	48.69%	0.318
	100	5.281	47.91%	0.311
	300	**5.236**	**47.54%**	**0.307**

Table 5.3 Simplified Langevin MCMC (95% confidence intervals).

Dataset	RM 30%	TM 5%	FM 5%	TW 1.2	MCD ↓	FFE ↓	$\log f_o$ ↓
LJSpeech	✓				4.013	46.57%	0.280
	✓	✓			3.958	46.31%	0.276
	✓		✓		3.997	45.63%	0.278
	✓			✓	3.927	45.98%	0.267
	✓	✓	✓	✓	**3.882**	**45.36%**	**0.258**
LibriTTS	✓				5.312	48.01%	0.309
	✓	✓			5.246	47.82%	0.306
	✓		✓		5.295	47.13%	0.307
	✓			✓	5.213	47.41%	0.299
	✓	✓	✓	✓	**5.164**	**46.92%**	**0.292**

Table 5.4 Combination of negative sampling methods (95% confidence intervals).

Method	MOS
Ground Truth	4.53±0.05
Ground Truth (log-Mel + WaveGlow)	4.39±0.07
Tacotron 2 (log-Mel + WaveGlow)	3.77±0.11
EBM (log-Mel + WaveGlow)	3.84±0.13

Table 5.5 Subjective evaluation of NCE training method.

The second limitation is that only one pre-trained TTS model was used as the base model for crafting negative samples and providing starting points for inference.

1. Although this chapter explored a range of negative sampling methods, it has not verified whether these methods would work with other pre-trained TTS models (e.g., Transformer-TTS and FastSpeech 2). Since most applications of energy-based models (EBMs) rely on NCE for training but there seems not to be a universal, high-quality, negative sample generation method, a clear case could be put forward to develop alternative criteria related to score matching or diffusion models that do not come with those assumptions and requirements.

MOS	1	2	3	4	5
Tacotron 2	0	3	121	344	15
EBM	0	1	94	370	18

Table 5.6 MOS score counts.

2. Inference schemes used with EBMs offer flexibility regarding initialization. One option, and the one explored in this chapter, is to sample from a pre-trained TTS model. Another option is to sample from a simpler prior distribution (e.g., a standard Gaussian distribution) as common with Langevin MCMC schemes. The latter is practically more challenging due to the low quality of the initial speech sample. However, it may still be possible to produce high-quality samples from a simple prior distribution with EBMs, similar to diffusion models [45] and denoising score matching [118, 114, 115], if sufficient number of iterations is used.

Although NCE developed in this chapter provides a feasible and efficient solution for training EBMs in TTS, inference is an iterative process which may prove to be computationally demanding. Thus, exploring inference with fewer iterations in EBMs remains an open research direction.

Most of research on EBMs, as is the investigation presented in this chapter, are limited to external post-filtering, rescoring or additional pre-processing as a part of some application. The recently proposed diffusion transformer [96] combines diffusion methods with the transformer architecture in a single model and serves as an example for developing fully integrated EBMs. Although some work has been done in this direction [34, 47], none has so far been applied to TTS. Despite this, those studies prove that it is possible to design energy-based architectures that do not rely on residual connections/pre-trained models. This may lead to more efficient and effective TTS models.

Chapter 6

Score-based Training for Energy-based TTS Models

6.1 Introduction

Energy-based models (EBMs) [66, 138] may be expected to remain as a popular choice for probabilistic modelling beyond TTS due to the lack of strong modelling assumptions and the simplicity they offer for formulating the log-likelihood functions. The simple difference between the negative (unnormalised) energy function and the logarithm of the corresponding normalisation term may appear quite appealing to practitioners. Unfortunately, those normalisation terms are difficult to compute in most of cases of interest. Noise contrastive estimation (NCE) has been previously utilised to train EBMs in applications such as speech recognition [70], machine translation [10], text generation [7], and TTS [120] as was discussed in Chapter 5. The NCE compares unnormalised log-likelihoods of 'positive' (reference) examples to unnormalised log-likelihoods of 'negative' (noisy) samples to decide the direction for updating model parameters. Unlike language modelling, where NCE appears to work well with a wide choice of negative samples [14], TTS appears to be much more sensitive to the quality of negative samples and necessiates the development of elaborate sampling schemes [120].

Several alternative training approaches for EBMs have been proposed in the literature: score matching (SM) [52], denoising score matching (DSM) [130], and sliced score matching (SSM) [116]. As was discussed in Chapter 4, all these alternatives attempt to predict the gradient of log-likelihood with respect to data (speech in TTS), or *score*. However, the use of SM would require computing the trace of the Hessian, which is expensive for automatic differentiation packages [117]. The introduction of noise to data in DSM training leads

to inconsistency between inference and training [117]. Instead of matching scores, SSM matches their projections on randomly chosen directions, thus avoiding much expensive computation. SSM has been used with EBMs in many domains, e.g., imitation learning [61] and out-of-distribution detection [30, 75], but has not yet been investigated in TTS.

Score-based training of EBMs and currently popular diffusion style models [114, 74, 59] is typically followed by a variant of Markov Chain Monte-Carlo (MCMC) for performing inference. As a type of first-order optimisation[1], MCMC-style approaches are sensitive to the nature of objective function they optimise. For example, a highly non-convex shape may lead to issues in converging to an acceptable solution within acceptable time. Despite this, score-based approaches appear to focus on learning gradients as accurately as possible, regardless of the shape of the underlying objective.

This chapter introduces a new score-based learning objective, called *delta loss*, that penalises those scores that do not lie on linear paths between current speech estimates and reference speech samples. As such, delta loss is believed to be more appropriate for score-based models that utilise MCMC-style first-order optimisation approaches for inference. This chapter also shows that although differently motivated the new loss function is closely linked with training objectives of currently popular flow matching approaches [74].

Thus, this chapter makes the following key contributions:

1. first application of SSM to training EBMs in TTS;

2. a new score-based loss function for EBMs;

and the rest of it is organised as follows. Section 6.2 provides a brief overview of EBMs and related works, including training methods and connections between EBMs and diffusion models. Although some of this material has been previously presented in this thesis, it is repeated here for ease of reference. Section 6.3 describes existing score-based training approaches in the context of TTS and introduces delta loss as a possible alternative. Experimental results and discussion are presented in Section 6.5. Conclusions drawn from this chapter are presented in Section 6.6.

[1]The connection between inference and optimisation and the performance of different optimizers will be explored in Chapter 7.

6.2 Energy-based models

Given a sequence of text tokens $\boldsymbol{x}$, an energy-based model (EBM) of spectral feature sequences $\boldsymbol{Y}$ could be defined as [120]

$$p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})} \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})), \tag{6.1}$$

where $E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})$ is an energy function with parameters $\boldsymbol{\theta}$ that connects text and speech such that smaller values correspond to better matches between text and speech and *vice versa*, and $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ denotes the normalising term, which is intractable to calculate. Although intractable, normalising terms cancel out in log-likelihood ratios, which is the fact exploited in Section 6.2.1, or disappear in the following gradients of log-likelihood

$$\nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x}) = -\nabla_{\boldsymbol{Y}} E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}) - \underbrace{\nabla_{\boldsymbol{Y}} \log Z_{\boldsymbol{\theta}}(\boldsymbol{x})}_{=0}, \tag{6.2}$$

which is the fact exploited in Section 6.3 and by gradient-based inference schemes. In its simplest form, inference with EBMs could be performed using a gradient descent algorithm (GD)

$$\boldsymbol{Y}^{(N+1)} \leftarrow \boldsymbol{Y}^{(N)} + \rho \nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}^{(N)}|\boldsymbol{x}), \tag{6.3}$$

where ρ is an learning rate. Typically, a large number of iterations is required to yield accurate speech samples [120], which is consistent with other related models [114, 118, 45].

6.2.1 Noise contrastive estimation

Because of the intractable normalising term $\boldsymbol{Z}_{\boldsymbol{\theta}}(\boldsymbol{x})$, training approaches that do not involve normalising terms, such as noise contrastive estimation (NCE) [39], could be adopted. As its name suggests, learning in NCE is performed by contrasting positive samples $\boldsymbol{Y}^{+}$ with negative, noisy, samples $\boldsymbol{Y}^{-}$. Specifically, the loss function in NCE is given by

$$\begin{aligned} \mathscr{L}^{\texttt{nce}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{Y}^{-}) = &- \log \left(\frac{1}{1 + \exp\left(E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\right)} \right) \\ &- \log \left(\frac{1}{1 + \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{-})\right)} \right), \end{aligned} \tag{6.4}$$

where $\boldsymbol{Y}^{+}$ is a reference speech and $\boldsymbol{Y}^{-}$ is a noisy speech. According to equation (6.4), optimal energy functions assign high energies to noisy samples and low energies to positive

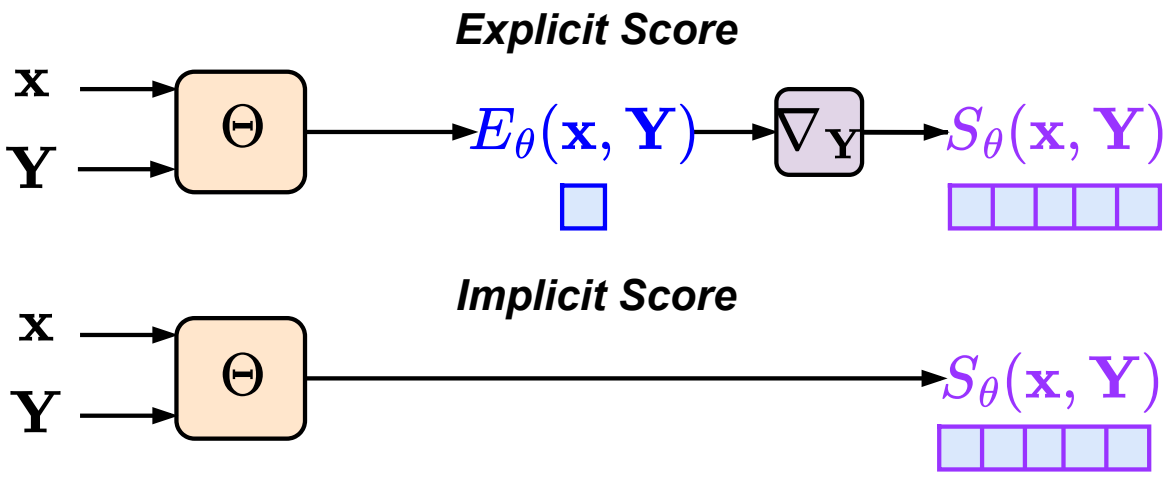


Fig. 6.1 Two ways of computing *scores* for EBMs: analytic (top) and predictive (bottom).

samples. Although NCE does not prescribe which noisy samples to use, the previous work in TTS found it sensitive to the quality of noisy speech [120]. This appears to be different to language modelling where simple negative sampling approaches were found to lead to better results [14]. Although high-quality noisy speech could be obtained by combining outputs from well-trained TTS models [112, 68, 106, 81] and data augmentation methods [120], such an approach would lead to an increased training cost.

6.2.2 Scores

The gradient of log-likelihood with respect to speech $\nabla_{\boldsymbol{Y}} \log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{x})$ is known in the literature as a *score* $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$. Such scores appear not only in EBMs but also in closely related diffusion models (DMs) where the following inference rule may be used to denoise speech

$$\boldsymbol{Y}^{(N+1)} \leftarrow \frac{1}{\sqrt{\alpha_N}}\{\boldsymbol{Y}^{(N)} + \beta_N \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}^{(N)},N)\} + \sigma_N \boldsymbol{Z}^{(N)}, \tag{6.5}$$

where $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y}^{(N)},N)$ denotes a time (iteration) dependent score, $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, and α_N, β_N, and σ_N are hyperparameters. The close connection between EBMs in equation (6.3) and DMs becomes apparent by noticing that $\frac{1}{\sqrt{\alpha_N}}$ is a scale that combines iteration dependent weight decay parameters, $1 - \frac{1}{\sqrt{\alpha_N}}$ with 1 from the current speech estimate. Thus, setting α_N to 1 and σ_N to 0 (removes stochastic term) would lead to the update rule in equation (6.3).

In addition to scores not depending on iteration index in EBMs ($\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y})$ vs $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{Y},N)$), another interesting aspect of EBM scores is that they can either be modelled directly as in DMs or obtained by computing the gradient of log-likelihood. Figure 6.1 illustrates these two different approaches. The 'Explicit Score' approach in Figure 6.1 illustrates an approach

that models the energy function $E_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})$ first, instead of only modelling the score function $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})$. The energy and score functions are connected via a gradient operation. The 'Implicit Score' approach in Figure 6.1 illustrates score matching (SM), which is discussed in more detail in Section 6.3. Although more expensive, the former approach ensures that scores are true gradients of some true log-likelihood function. While the goal during training using either score is to optimise the score matching objective, only the score function is used for iterative inference.

6.3 Score-based training

This section will first discuss score matching approaches, which were initially introduced in Chapter 4 and then will introduce a new criterion, which is shown to be linked with currently popular flow matching [74, 72, 43], that addresses some of score matching limitations.

6.3.1 Sliced score matching (SSM)

Rather than using NCE it is also possible to optimise EBMs by matching their scores to those of the true data distribution. If the true data distribution, $p_{\texttt{data}}(\boldsymbol{x}, \boldsymbol{Y})$ was known then EBM training could be performed by minimising the distance between model score $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y})$ and data score $\nabla_{\boldsymbol{Y}} \log p_{\texttt{data}}(\boldsymbol{x}, \boldsymbol{Y})$. As the latter is seldom known, an equivalent, practically feasible, formulation that does not depend on data scores has been found by [53]. It's unbiased estimator is given by

$$\mathscr{L}^{\texttt{sm}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}) = \operatorname{tr}\left(\nabla_{\boldsymbol{Y}^{+}} \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\right) + \frac{1}{2}\|\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\|_2^2, \tag{6.6}$$

where the first term is the trace of the Hessian matrix, which is computationally expensive for high-dimensional data [116] even with modern hardware [80]. Hence, direct score matching (SM) is only suitable for simple EBMs. Note that $\boldsymbol{Y}^{+}$ above are the ground-truth or reference spectral features.

Fortunately, a computationally efficient variant of SM called sliced score matching (SSM) [116] has been proposed and could be adopted with complex EBMs. Given a random projection vector $\boldsymbol{v}$, the loss function of SSM is given by:

$$\mathscr{L}^{\texttt{ssm}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}) = \boldsymbol{v}^{\top} \nabla_{\boldsymbol{Y}^{+}} \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}) \boldsymbol{v} + \frac{1}{2}\|\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+})\|_2^2. \tag{6.7}$$

Computing the first term in equation (6.7) requires only 2 gradient operations rather than $\dim(\boldsymbol{Y}^{+})$ gradient operations required in equation (6.6). Although it is possible to use K

random projection rather than one, the single random projection version above is used in this chapter. Unlike NCE, which requires special care to crafting negative samples, SSM appears to be a straightforward, albeit an expensive, training approach.

6.3.2 Delta loss (Delta/Δ)

Apart from requiring $1+K$ gradient computations, another drawback of SM-style approaches is that they focus on matching scores without explicitly considering whether the resulting score field is convenient for the first-order inference procedure used by EBMs. In the discussion below, the model parameters and conditioning text are fixed, as is the case during inference. The log-likelihood can therefore be viewed as a function of the candidate spectral features $\boldsymbol{Y}$, or equivalently as an energy surface over possible speech samples. Figure 6.2 illustrates schematic one-dimensional cross-sections of such surfaces. The figure is not intended to assume that the speech distribution is unimodal, concave, or globally simple; in a real TTS problem, the surface may have many modes and highly irregular local geometry. Rather, the figure illustrates that different local score geometries can be more or less suitable for first-order updates from a given initial hypothesis. Even if (S)SM perfectly matched data

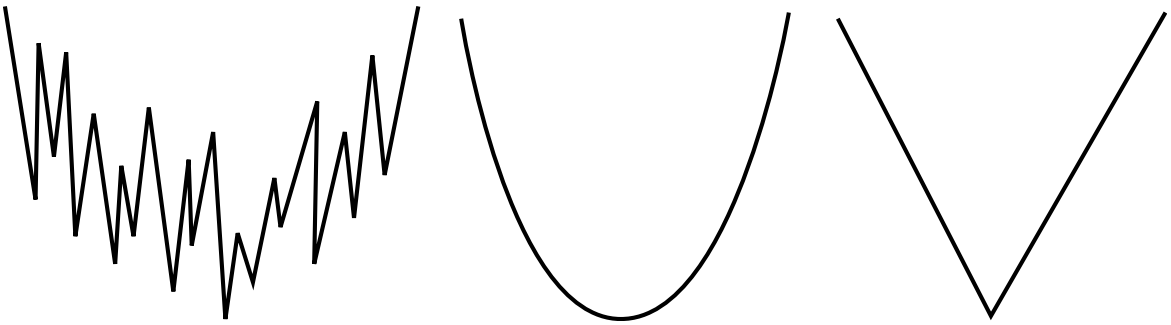


Fig. 6.2 Different score functions: from left to right, the hypotheses list from the worst to the best.

scores with model scores, the resulting score field is not guaranteed to be easy to follow with a finite number of first-order inference steps such as equation (6.5). A jagged local surface, as sketched on the left of Figure 6.2, may require many small steps and can be sensitive to the chosen learning rate. Smoother local surfaces, such as the middle example, are more compatible with first-order inference, but may still require careful tuning and multiple iterations. The right-hand example should therefore be interpreted as locally optimal only in the sense of first-order inference from the current hypothesis: along the path from the noisy hypothesis to its paired reference, the score direction is aligned with the desired update and can reach the reference in one Euler-style step if the step size is chosen consistently. This is

a local training-inference consistency argument, not an assumption that the full probability function of speech is single-mode or concave.

This suggests that, for a noisy spectral-feature estimate $\boldsymbol{Y}^-$ generated by an acoustic model and its paired reference $\boldsymbol{Y}^+$, the model score should point in the direction of the displacement $\Delta(\boldsymbol{Y}^+, \boldsymbol{Y}^-) = \boldsymbol{Y}^+ - \boldsymbol{Y}^-$. With a first-order update using step size η_Δ, this desired one-step relation can be written as

$$\boldsymbol{Y}^+ = \boldsymbol{Y}^- + \eta_\Delta \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^-) \tag{6.8}$$

for the local pair $(\boldsymbol{Y}^-, \boldsymbol{Y}^+)$. Equivalently, the target score is $(\boldsymbol{Y}^+ - \boldsymbol{Y}^-)/\eta_\Delta$. In the experiments below, η_Δ is absorbed into the scale of the learned score, which is equivalent to taking $\eta_\Delta = 1$ during training and tuning the inference learning rate separately. The corresponding objective function, called *delta loss* in this section, is given by

$$\mathscr{L}^{\Delta}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^+, \boldsymbol{Y}^-) = \frac{1}{2} \left\| \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^-) - \frac{\boldsymbol{Y}^+ - \boldsymbol{Y}^-}{\eta_\Delta} \right\|_2^2. \tag{6.9}$$

Compared to NCE and (S)SM, delta loss is a very efficient training approach because it avoids both negative-sample classification and Hessian-vector computations. Its purpose is not to learn the exact data score everywhere, but to learn a local update direction that is useful for the inference scheme actually used by the model.

6.3.3 Connection to flow matching

Interestingly, the delta loss in equation (6.9) can be linked to the Flow Matching (FM) loss [74, 72, 43]

$$\mathscr{L}^{\mathtt{fm}}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^+, \boldsymbol{Y}^0, t) = \frac{1}{2} \| \boldsymbol{V}(\boldsymbol{x}, \boldsymbol{Y}_t, t) - (\boldsymbol{Y}^+ - \boldsymbol{Y}^0) \|^2, \tag{6.10}$$

FM constructs a probability path between a source sample $\boldsymbol{Y}^0$ and a target sample $\boldsymbol{Y}^+$ and trains a velocity field that follows this path. For a straight path, the target velocity is $\boldsymbol{Y}^+ - \boldsymbol{Y}^0$, and applying Euler's method, as introduced in equation (3.31), to the learned velocity field gives a first-order update of the same form as equation (6.8). This explains why the algebraic form of delta loss is close to the FM loss: both encourage a vector field that follows a linear path between an initial point and a target point.

The motivation, however, is different. In FM, the linear path is introduced as a probability path for generative modelling, and the term 'conditional' refers to conditioning this path on a source sample rather than on text alone. In the delta-loss formulation, no claim is made

that the full speech distribution is globally linear, unimodal, or concave. Instead, the paired acoustic-model output $\boldsymbol{Y}^-$ and reference $\boldsymbol{Y}^+$ define a local path used to train a score field that is well matched to first-order EBM inference. Thus, delta loss can be understood as a one-step, locally path-based objective inspired by the same Euler-style reasoning as FM, but motivated by training–inference consistency for EBMs rather than by constructing a complete probability path.

6.4 Experimental setup

6.4.1 Dataset

The dataset utilized in this chapter is LJSpeech [54], which was previously used in Chapter 5. LJSpeech consists of 13,100 audio clips totalling approximately 24 hours of speech from a single female speaker reading passages from public domain texts. The dataset is randomly divided into training (10,000 clips), validation (1,800 clips), and test (1,300 clips) sets. A randomly chosen subset of 600 validation set clips was used in some objective evaluations to streamline the process. Another randomly chosen subset of 100 test set clips was used in subjective evaluations. Considering both the volume of speech recordings and speech quality, this work conducts experiments on two additional datasets: train-clean-100 and train-clean-360. The datasets are randomly split into 124,736 samples for training, 10,000 for validation, and 15,000 for testing, which is the same split as in Section 5.4.1. The same experiments are conducted on these two datasets, but only for objective evaluation.

6.4.2 Models

The pretrained acoustic model generating hypotheses of spectral features is an open-source Tacotron 2[2]. These spectral features are used as-is as an initialisation of inference for all EBMs in this section. They were extracted using a hop length of 256, a window length of 1024, 80 Mel bins, and a frequency range of 0–8000 Hz. For training these hypotheses are either not used (SSM), or adopted as-is (Δ) or additionally transformed to yield higher quality noisy samples (SSM). To ensure that references and hypotheses in training have the same duration (length), the pretrained acoustic model is forced to output spectral feature sequences with reference durations. No such constraint is enforced in inference. The non-score-based EBM trained with NCE is adopted from Chapter 5 [120]. The score-based EBMs are based on the widely popular U-net architecture [110]. The specific architecture is identical to Matcha-

[2]https://github.com/NVIDIA/tacotron2

TTS [81][3] other than time/iteration was removed to yield time-independent scores. The U-Net decoder used two downsampling blocks, followed by two midblocks and two upsampling blocks. Each block comprised a single Transformer layer with a hidden dimensionality of 256, 2 attention heads, an attention dimensionality of 128, and 'snakebeta' activations. Each score-based EBM was trained for 50K steps using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 10 on a single NVIDIA 3090 GPU. During inference, hypotheses from the score-based EBMs are updated using a gradient descent algorithm with a learning rate of 3×10^{-5}. An open-source implementation of HiFi-GAN [63][4] is used as the vocoder to generate waveforms.

6.4.3 Evaluation

The Mel cepstral distortion (MCD), log-scale F0 root mean square error (log f0), Speech-BERTScore [111], and UTMOSv2 [3] are used as objective evaluation metrics. While the former two metrics are well known in the literature, the latter two, SpeechBERTScore and UTMOSv2, are model-based metrics that were previously found to be highly correlated with human perception. An implementation of these objective evaluation metrics is available in the `DiscreteSpeechMetrics` toolkit[5]. Mean Opinion Score (MOS) is used as a subjective evaluation metric to assess the perceived naturalness of synthesized and reference speech. Five predictive models took part in listening tests (1xTacotron 2, 2xSSM, 2xDelta). The MOS evaluation used the same platform and rating scale as Chapter 5. In addition to the ground truth and Tacotron 2 baseline, five score-based models were included. To obtain three independent ratings for each audio sample, the evaluation contained 2,100 ratings (7 models $\times$ 100 test clips $\times$ 3 ratings). Since each native English listener rated one set of 100 audio clips, 21 listeners were recruited through Amazon Mechanical Turk. The average score for each model was then used as its MOS.

6.5 Experiments

6.5.1 Objective evaluation

Table 6.1 compares the performances of EBMs trained using SSM loss, delta loss or SSM and delta loss to the pretrained acoustic model, Tacotron 2. Table 6.2 follows a similar trend to Table 6.1. Both SSM and delta loss improve over the Tacotron 2 baseline in MCD, log

[3]https://github.com/shivammehta25/Matcha-TTS
[4]https://github.com/jik876/hifi-gan
[5]https://github.com/Takaaki-Saeki/DiscreteSpeechMetrics

Dataset	SSM	Delta	MCD ↓	log $f0$ ↓	S-BERT ↑	UTMOSv2 ↑
LJSpeech	✓	✗	**5.312**	**0.296**	**0.905**	3.710
	✗	✓	5.399	0.299	0.900	2.959
	✓	✓	**5.312**	**0.296**	**0.905**	**3.718**
	Tacotron 2		5.770	0.300	0.882	3.409

Table 6.1 Objective evaluation of score-based training on LJSpeech, 100 inference steps, full validation (1800) set.

Dataset	SSM	Delta	MCD ↓	log $f0$ ↓	S-BERT ↑	UTMOSv2 ↑
LibriTTS	✓	✗	5.124	0.309	0.892	3.586
	✗	✓	5.207	0.312	0.887	3.021
	✓	✓	**5.118**	**0.294**	**0.898**	**3.604**
	Tacotron 2		5.591	0.318	0.861	3.284

Table 6.2 Objective evaluation of score-based training on LibriTTS, 100 inference steps.

f_0, and S-BERT, showing that score-based EBMs can refine spectral features beyond the initial acoustic-model output. The delta loss remains competitive in MCD and S-BERT, but again gives a lower UTMOSv2 score, suggesting that matching the direct displacement alone may not preserve perceptual naturalness as reliably as SSM. Combining SSM and delta loss gives the best overall performance on LibriTTS, with slightly lower MCD and log f_0 and slightly higher S-BERT and UTMOSv2 than using SSM alone. This indicates that the two objectives can be complementary: SSM encourages consistency with the data score, while delta loss encourages update directions that are useful for first-order inference. The number of inference steps, N, for all EBMs was set to 100. Despite its simplicity, the delta loss trained EBM is competitive to the SSM trained EBM in all metrics other than UTMOSv2. The delta loss also appears to underperform compared to Tacotron 2 in the same metric while being superior based on all others. Combining SSM and delta losses yields only modest gains on LJSpeech, but Table 6.2 suggests a more consistent improvement on LibriTTS, where the combined objective slightly outperforms SSM alone across all objective metrics.

Table 6.3 shows how the number of inference steps affects objective evaluation metrics for SSM and delta loss trained EBMs. These results suggest that the number of useful inference steps depends on the dataset. On LJSpeech, SSM and delta loss achieve most of their improvement after only one step, which contrasts with the hundreds of steps that may be required with NCE loss EBMs and related diffusion models. On LibriTTS, however, the refinement is more gradual: SSM continues to improve up to 100 steps, while delta

Dataset	Step	MCD↓	$\log f_0 \downarrow$	S-BERT ↑	UTMOSv2 ↑
LJSpeech	0	5.765	**0.297**	0.882	3.412
	1	**5.294**	0.299	**0.904**	3.717
	10	5.294	0.299	0.904	3.708
	50	5.294	0.300	0.904	**3.719**
	100	5.294	0.300	0.904	3.704

(a) SSM

Dataset	Step	MCD↓	$\log f_0 \downarrow$	S-BERT ↑	UTMOSv2 ↑
LJSpeech	0	5.765	0.297	0.882	3.412
	1	5.292	0.299	**0.904**	**3.703**
	10	5.277	0.299	0.904	3.674
	50	**5.273**	**0.297**	0.903	3.400
	100	5.389	0.299	0.900	2.937

(b) Delta

Table 6.3 Objective evaluation of inference for score-based EBMs on LJSpeech, partial validation (600) set.

Dataset	Step	MCD↓	$\log f_0 \downarrow$	S-BERT ↑	UTMOSv2 ↑
LibriTTS	0	5.591	0.318	0.861	3.284
	1	5.268	0.313	0.885	3.514
	10	5.171	0.311	0.890	3.562
	50	5.133	0.310	0.891	3.579
	100	**5.124**	**0.309**	**0.892**	**3.586**

(a) SSM

Dataset	Step	MCD↓	$\log f_0 \downarrow$	S-BERT ↑	UTMOSv2 ↑
LibriTTS	0	5.591	0.318	0.861	3.184
	1	5.389	0.316	0.875	**3.197**
	10	5.276	0.314	0.883	3.118
	50	5.224	**0.311**	0.886	3.062
	100	**5.207**	0.312	**0.887**	3.021

(b) Delta

Table 6.4 Objective evaluation of inference for score-based EBMs on LibriTTS.

loss also benefits from additional steps in MCD and S-BERT. This suggests that the more variable LibriTTS data require more iterative correction than LJSpeech. Nevertheless, the delta-loss UTMOSv2 values decrease as the number of steps increases, indicating that aggressive iterative displacement can improve spectral-distance metrics while reducing perceived naturalness.

Table 6.5 compares NCE [120] and score-based trained EBMs. These results suggest that

Dataset	Loss	Step	MCD↓	$\log f_0 \downarrow$	S-BERT ↑	UTMOSv2 ↑
LJSpeech	NCE*	100	5.762	**0.295**	0.875	2.572
	SSM	1	5.294	0.299	0.904	**3.717**
	Delta	1	**5.292**	0.299	0.904	3.703

Table 6.5 Objective evaluation of NCE (full validation set) and score-based (partial validation set) trained EBMs on LJSpeech.

score-based training is a powerful alternative that yields significantly better MCD, S-BERT and UTMOSv2 values but may be insignificantly worse in $\log f_0$.

6.5.2 Subjective evaluation

Finally, a range of listening tests was conducted to get an insight into score-based EBMs following the protocol outlined in Section 6.4.3. Table 6.6 summarises the conducted MOS study. The first block in Table 6.6 shows MOS scores for original reference, ground truth,

#	Method	MOS
1	Ground Truth	4.29±0.04
2	Ground Truth (log-Mel + HiFi-GAN)	**4.15±0.06**
3	Tacotron 2 (log-Mel + HiFi-GAN)	3.74±0.08
4	SSM (1 steps) (log-Mel + HiFi-GAN)	3.90±0.06
5	Delta (1 steps) (log-Mel + HiFi-GAN)	**3.95±0.05**
6	SSM (10 steps) (log-Mel + HiFi-GAN)	3.82±0.09
7	Delta (10 steps) (log-Mel + HiFi-GAN)	3.76±0.07

Table 6.6 Subjective evaluation of score-based training methods.

waveforms (line 1), ground truth waveforms resynthesised from log-Mel spectrograms using HiFi-GAN vocoder (line 2) and the pretrained acoustic model, Tacotron 2, used in this section (line 3). As expected, these results suggest that some level of degradation could be attributed to the imperfect vocoding process and that the pretrained acoustic model lags 0.41 in MOS

behind that construted result. The second block confirms subjectively that the delta loss (line 5) proposed in this section offers a computationally efficient powerful alternative to a more expensive and elaborate SSM loss (line 4). The delta loss trained EBM reduces the lag in MOS from resynthesised ground truth in line 2 to 0.20. The final, third, block shows that although for SSM objective metrics appear to show little discrimination, the additional inference steps lead to a degradation in MOS. For delta loss, MOS and UTMOSv2 appear to be correlated.

The breakdown of MOS score counts in Figure 6.3 shows that although SSM and delta loss trained EBMs appear to be comparable in terms of the overall MOS scores, there are in fact important differences between these EBM systems. Figure 6.3 suggests that the delta

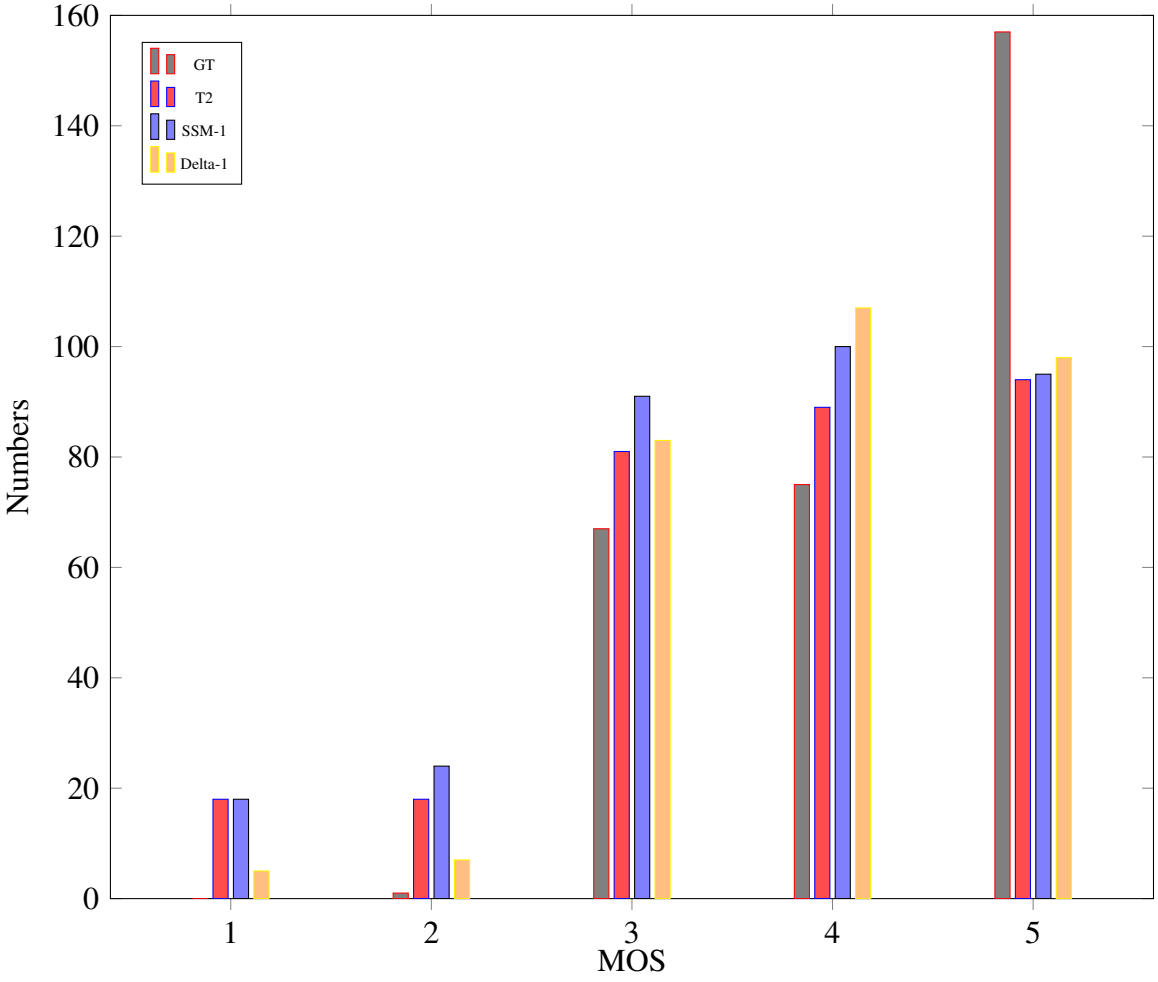


Fig. 6.3 Detailed breakdown of MOS score counts.

loss trained EBM has much fewer 1, 2 and 3 scores and more 4 and 5 scores than SSM trained EBM.

6.6 Summary

This chapter presented two new score-based approaches to training energy-based models (EBMs) for text-to-speech (TTS) as an alternative to more challenging Noise Contrastive Estimation (NCE). The first approach is sliced score matching (SSM) that has been previously used with EBMs in other areas but not in TTS. The second approach is a new objective function called *delta loss* which aims to promote log-likelihoods more suitable for first-order inference schemes used by EBMs and is simpler than NCE and SSM. This new loss was shown to be connected to losses used in currently popular flow matching. Experiments on the LJSpeech dataset show superior performance of score-based approaches compared to NCE. Despite its simplicity, the new delta loss function appears to show comparable performance in objective evaluations and better performance in subjective evaluations (less low scores and more high scores).

6.6.1 Limitations and future work

In flow matching inference methods, the trade-off between inference speed and effectiveness using different solvers is a well-known problem. Similar to the limitation discussed in Chapter 5, the inference for the proposed SSM and delta loss approaches was only performed using a simple gradient descent algorithm. Exploring the use of other, more advanced, optimizers (e.g., Adam), which can be more effective at finding a global optimum, is a promising direction for future work. Related work in this area will be discussed in the next chapter.

The experiments in this chapter were limited to a single-speaker English dataset (LJSpeech) and used a specific acoustic model (Tacotron 2) for generating initial hypotheses. This setup does not fully demonstrate the generalisation capabilities of the proposed score-based training approaches. It remains an open question how these methods would perform with different TTS models, such as non-autoregressive models like FastSpeech 2, or on more challenging datasets, including multi-speaker or multilingual corpora. Future research should therefore focus on evaluating the proposed methods across a diverse range of models and datasets to confirm their broader applicability and robustness. For example, the proposed SSM and delta loss approaches could be trained on the spectral features generated from a pre-trained Tacotron 2. Then the trained models could be applied to a different TTS model, such as FastSpeech 2, to refine its spectral features. This would allow for a more comprehensive evaluation of the generalisation capabilities of the proposed methods.

Another interesting direction for future work is to explore the generalisation capability by training the model based on the difference between a pre-trained model's output $\boldsymbol{Y}^{-}$ and

Applied	Training	Inference
✓	$\mathscr{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{Y}^{-})$	$\boldsymbol{Y}^{-}$
✗	$\mathscr{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{Y}^{-})$	$\boldsymbol{\varepsilon}$
✗	$\mathscr{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{Y}^{+}, \boldsymbol{\varepsilon})$	$\boldsymbol{Y}^{-}$

Table 6.7 Future work for score-based traing methods.

the reference $\boldsymbol{Y}^{+}$, but starting inference from a simple distribution $\boldsymbol{\varepsilon}$, such as a Gaussian distribution. Conversely, it is also worth exploring training the model based on the difference between a simple distribution and the reference, and then applying it to a pre-trained model's output as a speech augmentation task. Potential directions for future work are summarized in Table 6.7. The first line describes the work presented in this chapter, which involves training a model on the difference between the reference and a pre-trained model's output, and then using the pre-trained model's output as the starting point for inference. The second line describes training on the difference between the reference and the pre-trained model's output, but starting inference from a simple distribution. The third line describes training on the difference between a simple distribution and the reference, and then applying the resulting model to the output of a pre-trained model.

As described in the literature, diffusion-like TTS models such as Grad-TTS [98] are jointly trained using a combination of mean square error (MSE) and diffusion loss. This approach provides 'good' quality coarse spectral features, which are ensured by the MSE loss. The proposed SSM and delta loss methods could be integrated with the MSE loss to further improve the performance of the TTS model.

Chapter 7

Inference Approaches for Energy-based TTS Models

As discussed in Chapter 3 sampling from energy-based models (EBMs) is traditionally performed using Langevin Markov chain Monte-Carlo (MCMC) method. In Langevin MCMC candidate outputs are updated based on the gradient of log-likelihood (energy) function with respect to candidate output $\boldsymbol{Y}$ and noise determined by the chosen step size. This method cannot guarantee that candidate outputs would improve from one iteration to another as it is intrinsically linked with the shape of the energy function. Furthermore, EBMs with hard or impossible to compute normalising terms can only provide estimates of energies for each candidate output rather than log-likelihoods, which makes termination (when to stop iterative inference) and analysis of candidates tricky.

In contrast, closely related diffusion models are based on denoising functions that have a more clear purpose of noise removal. Thus, candidate outputs at any given step are expected to contain less noise than candidates at the previous step. Such a more intuitive formulation of generative modelling should make termination and analysis of candidates more straightforward. Although log-likelihood computation is possible with diffusion models, a care needs to be taken interpreting them [22].

Despite their differences, inference with energy-based, diffusion, flow matching and other related models could be viewed as a general-purpose optimization problem, where initial candidate outputs $\boldsymbol{Y}$ are optimized by minimizing given loss function. As such, inference with these models should not be constrained only to the variants of gradient-descent that have been explored so far and more advanced optimization approaches, such as Adam [62], should be explored. This chapter thus provides a case study on the use of different optimization techniques for optimizing energy functions of EBM TTS models.

7.1 Gradient descent

Gradient descent is a first-order iterative optimization algorithm used to find a local minimum of a differentiable function. The core idea is to take repeated steps in the direction opposite to the gradient of the function at the current point, as this is the direction of steepest descent. The update rule for standard gradient descent is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$$

where $\boldsymbol{\theta}_t$ represents the parameters at iteration t, η is the learning rate that determines the step size, and $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$ is the gradient of the loss function J with respect to the parameters.

In the context of energy-based TTS models, gradient descent could be employed to optimize candidate Mel-spectrograms $\boldsymbol{Y}$ by minimizing the energy function. The optimization starts from an initial candidate and iteratively updates it by moving against the gradient of the energy function.

To enhance the performance of standard gradient descent, techniques such as momentum and weight decay are often incorporated. Momentum helps to accelerate convergence by accumulating a velocity in directions of persistent descent. Weight decay (L2 regularization) is used to prevent overfitting by adding a penalty term to the loss function that discourages large parameter values. The algorithm below details an implementation of gradient descent that combines both momentum and weight decay.

Algorithm 4 Gradient Descent with Separate Momentum and Weight Decay Steps

Require: Learning rate η, momentum coefficient γ, weight decay λ, initial parameters $\boldsymbol{\theta}_0$, initial velocity $\boldsymbol{v}_0 = \mathbf{0}$, loss function $J(\boldsymbol{\theta})$

1: **for** $t = 0$ to $T-1$ **do**

2: Compute gradient: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$

3: Apply weight decay:

$$\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) + \lambda \boldsymbol{\theta}_t$$

4: Update velocity with momentum:

$$\boldsymbol{v}_{t+1} \leftarrow \gamma \boldsymbol{v}_t + \eta \boldsymbol{g}_t$$

5: Update parameters:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \boldsymbol{v}_{t+1}$$

6: **end for**

7: **return** $\boldsymbol{\theta}_T$

The following experiments were conducted to evaluate the performance of the gradient descent algorithm with different learning rates and weight decay values. The results are presented in terms of Mel Cepstral Distance (MCD), the logarithm of the root mean square error of F0 (log $f0$ for short), SpeechBERTScore (S-BERT) and UTMOSv2. The model under investigation was trained using noise contrastive estimation (NCE), which performed best in Table 5.4. The negative sampling approach combines 30% random masking (RM), 5% time masking (TM), 5% frequency masking (FM), and 1.2 time (compressing) warping (TW). All experiments were conducted with $N = 100$ steps of gradient descent.

Table 7.1 shows similar performance across different learning rates, with only slight variations in MCD and log $f0$, but noticeable differences in S-BERT and UTMOSv2. A learning rate of 0.01 was chosen for the experiments shown in Table 7.2.

LR	MCD↓	$\log f_0 \downarrow$	S-BERT↑	UTMOSv2↑
0.01	**4.217**	0.301	**0.861**	**2.632**
0.001	4.220	**0.298**	0.836	2.564
0.0003	4.219	0.302	0.817	2.553

Table 7.1 GD with different learning rates.

Table 7.2 presents the results of experiments with different weight decay values. Note that diffusion models also include a weight decay like term. All metrics show that weight

decay could have a significant impact on model performance. The best overall performance is achieved with a weight decay of 0.1, which yields the lowest MCD, the highest S-BERT, and the highest UTMOSv2. However, weight decay appears to have a more minor effect on $\log f0$.

Weight decay	MCD↓	$\log f_0 \downarrow$	S-BERT↑	UTMOSv2↑
0.2	4.435	0.307	0.874	2.841
0.1	**3.872**	0.303	**0.886**	**3.046**
0.01	4.137	0.301	0.830	2.634
0.001	4.208	**0.299**	0.859	2.872
0	4.217	0.301	0.861	2.632

Table 7.2 GD with different weight decay values.

7.2 Adam

Adam, which stands for Adaptive Moment Estimation [62], is an optimization algorithm that computes adaptive learning rates for each parameter. It is designed to combine the advantages of two other extensions of stochastic gradient descent: AdaGrad, which works well with sparse gradients, and RMSProp, which works well in online and non-stationary settings. Adam is particularly effective for training large-scale deep learning models.

The core idea of Adam is to maintain two moving averages for each parameter: the first moment (the mean) and the second moment (the uncentered variance) of the gradients. These are denoted by $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ respectively.

Algorithm 5 Adam Optimizer

Require: Learning rate α, decay rates $\beta_1, \beta_2 \in [0,1)$, smoothing term $\boldsymbol{\varepsilon}$, initial parameters $\boldsymbol{\theta}_0$, initial moments $\boldsymbol{m}_0 = \mathbf{0}$, $\boldsymbol{v}_0 = \mathbf{0}$, loss function $J(\boldsymbol{\theta})$

Ensure: Optimized parameters $\boldsymbol{\theta}$

1: **for** $t = 1$ to T **do**

2: Compute gradient: $\boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})$

3: Update first moment estimate:

$$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$$

4: Update second moment estimate:

$$\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$$

5: Compute bias-corrected first moment:

$$\hat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{1-(\beta_1)^t}$$

6: Compute bias-corrected second moment:

$$\hat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{1-(\beta_2)^t}$$

7: Update parameters:

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}$$

8: **end for**

9: **return** $\boldsymbol{\theta}_T$

The algorithm first computes the moving averages of the gradient and the squared gradient. Since these moving averages are initialized as vectors of zeros, they are biased towards zero, especially during the initial time steps. To counteract this, Adam computes bias-corrected estimates $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{v}}_t$. Finally, it updates the parameters using these corrected estimates. The update rule scales the learning rate for each parameter based on the estimates of the first and second moments of the gradients. Weight decay can be incorporated into Adam to prevent overfitting, typically by adding an L2 regularization term to the loss function. This modifies the gradient by adding a term proportional to the parameter values. The algorithm is detailed in Algorithm 5.

Table 7.3 presents the results of experiments with different learning rates for the Adam optimizer. The experiments in this section were all conducted using the same settings as in Section 7.1. The results show that Adam generally performs better than GD, achieving lower MCD and log $f0$ values, as well as higher S-BERT and UTMOSv2 scores across all learning rates. The best performance is observed with a learning rate of 0.01. This indicates that the choice of learning rate during inference is crucial for the performance of energy-based TTS models when Adam rather than GD optimizer is used.

LR	MCD↓	$\log f_o \downarrow$	S-BERT↑	UTMOSv2↑
0.02	3.906	0.294	0.782	2.879
0.01	**3.882**	**0.258**	**0.891**	3.102
0.001	4.167	0.300	0.864	**3.126**
0.0003	4.213	0.300	0.862	3.115

Table 7.3 Adam with different learning rates.

Table 7.4 shows the results of experiments with different weight decay values for the Adam optimizer. The results indicate that the best performance is achieved without weight decay, yielding the lowest MCD, log $f0$ values, the highest S-BERT and the highest UTMOSv2. Performance improves as the weight decay value decreases, with a weight decay of 0 leading to the best results. This suggests that weight decay may not be beneficial for inference in energy-based TTS models when Adam rather than GD optimizer is used, as it appears to lead to underfitting.

Weight decay	MCD↓	$\log f_o \downarrow$	S-BERT↑	UTMOSv2↑
0.2	4.482	0.307	0.812	2.618
0.1	4.482	0.303	0.826	2.744
0.01	4.466	0.302	0.847	2.836
0.001	4.359	0.304	0.864	2.957
0	**3.882**	**0.258**	**0.891**	**3.102**

Table 7.4 Adam with different weight decay values.

From a theoretical perspective, Adam can be expected to outperform standard GD in this inference setting because it acts as a diagonal preconditioner for the energy landscape. GD uses a single global learning rate for all Mel-spectrogram elements, so the same step size must be applied to regions with very different gradient scales. If the energy landscape is jagged, a learning rate that is safe for high-curvature or large-gradient regions may be too

small for flatter regions, leading to slow progress. Conversely, a larger learning rate may produce unstable updates in sensitive scales.

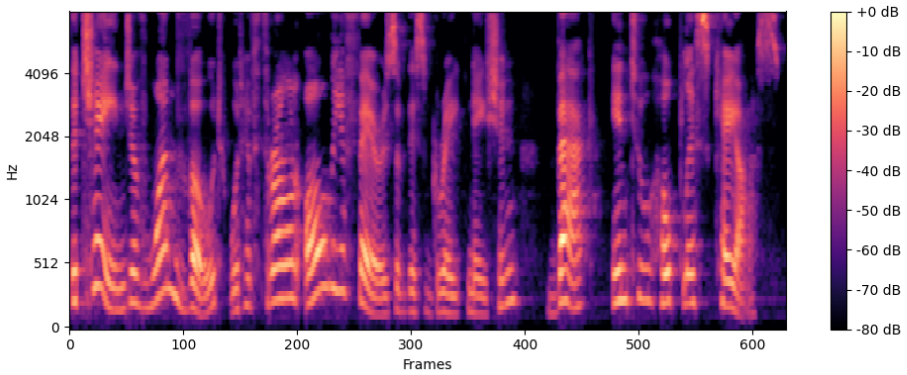
Adam mitigates this issue by normalising the update using an exponential moving average of squared gradients. Coordinates that repeatedly receive large gradients are assigned smaller effective step sizes, whereas coordinates with smaller gradients can receive relatively larger updates. The first-moment estimate also introduces a momentum-like smoothing effect, which can reduce oscillations caused by local irregularities in the learned energy function. Therefore, although Adam does not change the underlying EBM objective or guarantee a globally optimal solution, it provides a better-conditioned local optimisation procedure than GD for refining continuous Mel-spectrograms. This theoretical interpretation is consistent with the empirical results in Tables 7.3 and 7.4, where Adam achieves better objective scores than GD under comparable inference settings.

7.3 Sampling from a noise

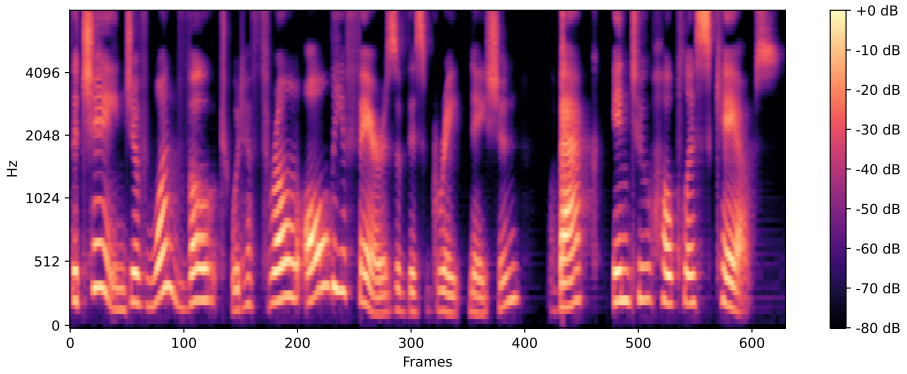
Inference methods for EBMs provide flexibility in how they are initialized. One approach involves using samples from a pre-trained TTS model as the initial sample. An alternative strategy is to initialize from a simpler prior distribution, such as the standard Gaussian with zero mean and identity covariance matrix. This latter option is more challenging in practice because initial speech samples are likely to be outside of the domain of spectral features, unless Gaussian mean and covariance are carefully set. Nevertheless, it may still be possible to generate high-quality results from a simple prior with EBMs by using a sufficient number of iterations, similar to the approach taken by diffusion models and denoising score matching techniques. In this section, the latter approach is explored. The candidates are initialized from a standard Gaussian distribution, and the inference process iteratively refines these candidates by minimizing the energy function.

The model explored in this section was trained with sliced score matching (SSM), as detailed in Table 6.3a, and represents one of the highest performing EBMs trained as a part of this work. The inference process iteratively refines candidate outputs $\boldsymbol{Y}$, which are initialized from a Gaussian distribution. The optimizer is GD with a learning rate of 3×10^{-5}, consistent with the settings in Table 6.3a. A case study is presented to illustrate how the candidates $\boldsymbol{Y}$ evolve during the inference process. Log-Mel spectrograms are generated using three methods from a randomly selected sample.

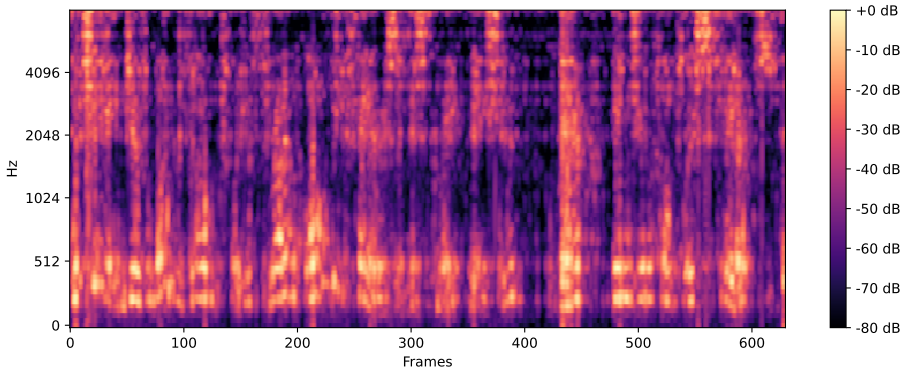
The results are shown in Figure 7.1. The first row shows the log-Mel spectrogram of the ground truth, which serves as a reference. The second row presents the log-Mel spectrogram generated by Tacotron2. The third row displays the log-Mel spectrogram produced by



(a) Log-Mel spectrogram of ground truth



(b) Log-Mel spectrogram produced from the Tacotron2 base model



(c) Log-Mel spectrogram initialized from a Gaussian noise with 100 iterations of gradient descent

Fig. 7.1 Noise initialization experiment. Transcript: *the change of one vote would have thrown all the affairs of this great Nation back into hopeless chaos.*

initializing from a Gaussian noise and applying 100 iterations of gradient descent. As shown in Figure 7.1c, the resulting sample is of poor quality, and is rather unintelligible. These results suggest that EBMs struggled to generalise to the large domain mismatch and would require smarter priors, such as those employed with Grad-TTS [98], or those used in this work for successful inference.

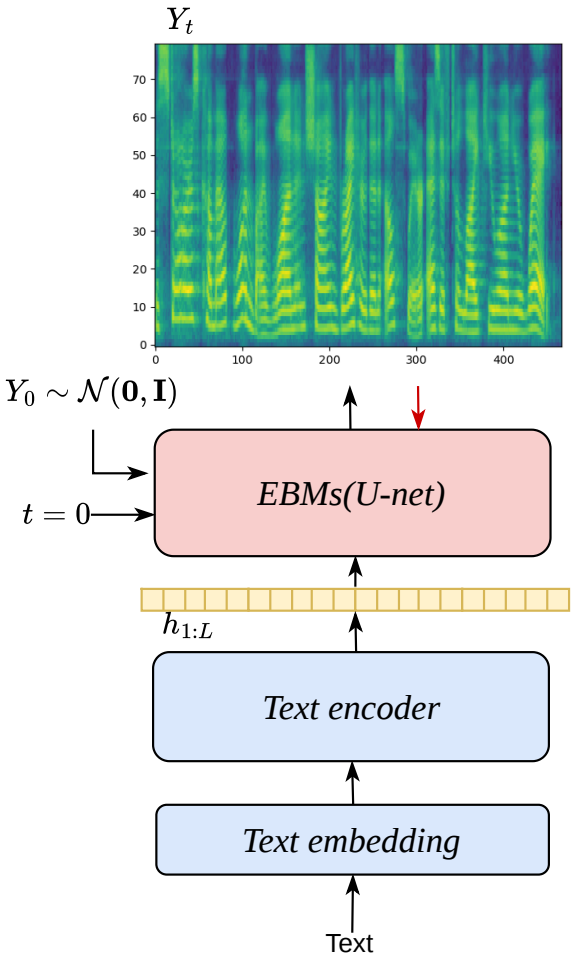


Fig. 7.2 Architecture of end-to-end energy-based TTS(EBTTS) (inspired by Grad-TTS [98] and Matcha-TTS [81]).

One reason for this limitation is that the current experiments are conducted using high-quality speech priors, namely Mel-spectrograms produced by a pre-trained acoustic model. Starting directly from Gaussian noise is therefore difficult for EBMs, because the model must simultaneously discover the global speech structure and refine local acoustic details. A potential solution, inspired by Grad-TTS [98] and Matcha-TTS [81], is to avoid generating directly from Mel-spectrograms produced by pre-trained acoustic models and instead refine Mel-spectrogram-like hidden features. These hidden features could be more flexible and self-adjusting than fixed acoustic-model outputs, while still providing the EBM with a speech-like prior. In this setting, the EBM can act as a decoder that transforms the hidden representation into high-quality speech features. The proposed model is shown in Figure 7.2.

This also highlights a broader trade-off between inference quality and computational time across Chapters 5, 6, and 7. Chapter 5 showed that EBMs can improve quality through iterative Langevin-style refinement, but each additional update increases inference cost. Chapter 6 then showed that score-based objectives can make the learned energy landscape more suitable for first-order updates, potentially reducing the number of iterations required for useful refinement. The experiments in this chapter further show that optimizer choice affects this trade-off: Adam can produce better samples than standard GD under the same iteration budget, but it also introduces additional per-step computation and memory for moment estimates. Therefore, practical EBM-based TTS systems should not only maximize objective quality metrics, but also select the initialization, optimizer, and number of refinement steps according to the target latency. This is another motivation for using stronger speech-like priors or hidden features, because they can reduce the amount of iterative correction required at inference time.

7.4 Conclusions

This chapter has provided a comprehensive analysis of inference approaches for energy-based TTS models, focusing on how different optimization strategies and initialization methods impact performance.

The experiments demonstrated that the choice of optimizer is a critical factor. The Adam optimizer consistently outperformed standard gradient descent, yielding significantly lower MCD and log f_0 errors. This highlights the benefits of adaptive learning rates for navigating the complex energy landscape of TTS models during inference. Furthermore, hyperparameter tuning, particularly the learning rate, was shown to be essential for achieving optimal results. In contrast, weight decay did not prove to be a beneficial regularizer for inference, with the best performance for Adam achieved when no weight decay was applied.

Another key finding of this chapter relates to the limited generalization of EBMs trained using approaches presented in the previous chapters to initialization of the inference process. While EBMs theoretically allow for generation from a simple prior like Gaussian noise, the experiments revealed that this approach fails to produce intelligible speech. High-quality results were only achieved when initializing the inference process with outputs from a pre-trained TTS model like Tacotron2 consistently with the training process. This is similar to models like Grad-TTS where samples from prior are intelligible and diffusion process is employed for quality refinement rather than generation from scratch.

7.4.1 Limitations and future work

This chapter focused primarily on two inference approaches: gradient descent and Adam. While these are classic and popular optimization techniques in machine learning, they are not the most advanced. Future work could investigate more current optimization methods, such as L-BFGS or AdamW. Furthermore, due to budget and time constraints, all experiments were conducted using only objective metrics. A more thorough evaluation would include subjective metrics, such as Mean Opinion Score (MOS) or ABX tests, which are considered the gold standard for assessing speech quality in TTS tasks.

Another limitation is the poor generation quality when initializing from a Gaussian noise. Given their ability to improve the speech quality of pre-trained TTS models, energy-based TTS models show potential as standalone generators, rather than just refinement modules, if denoising-like optimizers are applied. Although the samples generated from noise were unintelligible, the log-Mel spectrogram exhibits similar acoustic features (e.g., formants). Furthermore, while the U-Net architecture is a common choice for denoising tasks, models with larger capacity (e.g., more parameters or different architectures) could be explored. Since EBMs do not depend on iteration during training, it may be possible to enhance them with fewer training resources compared to diffusion models.

Chapter 8

Conclusion

This thesis focused on the challenges of applying EBMs to the speech synthesis task. The challenges mainly include applying the commonly used noise contrastive estimation (NCE) to EBMs with continuous outputs, finding an alternative training method for EBMs, and investigating the performance of different inference methods. In addressing these challenges, connections between EBMs and other generative models (e.g., diffusion models and flow matching models) were also established.

The challenges can be summarized as following four research questions:

- Can energy-based models be applied to TTS?
- Can we train EBMs with approaches other than NCE?
- Which inference approaches are better suited for EBMs?
- Is there a strong relationship between energy-based and diffusion-like TTS models?

This chapter summarizes the contributions that answer the research questions. Limitations and potential future work are discussed in Section 8.3.

8.1 Contributions

Energy-based TTS models using NCE training

In Chapter 5, an energy-based TTS model that uses NCE training was proposed. Motivated by successful applications of EBMs in various generative tasks, the EBM in this work was designed to learn an energy function that can be used to improve the speech quality of pre-trained TTS models. Such an approach is known in the literature as a residual EBM.

Several novel negative sampling methods were proposed to enhance the quality of negative hypotheses, which are crucial for the NCE training of EBMs. The combination of all negative sampling methods was observed to achieve the best performance in terms of speech quality. Both subjective and objective evaluations validate the effectiveness of the proposed methods.

Energy-based TTS models using sliced score matching (SSM) training

To avoid the challenges of designing negative samples, an alternative training method for EBMs was proposed in Chapter 6. The method is based on sliced score matching (SSM), a variant of score matching applicable to high-dimensional data such as TTS. Score matching and its variants, however, are focused on learning scores as accurately as possible and disregard the shape of the underlying loss function. Such an approach is problematic given that inference with those models is based on a variant of first-order optimisation, which is sensitive to the shape of the loss function. A novel training loss, the delta loss, was proposed to train the EBM that promotes loss functions consistent with inference algorithms. Unlike SSM, the delta loss is simple to implement and does not require extensive tuning. Both SSM and delta loss were shown to be effective for training EBMs for TTS, according to objective and subjective evaluations.

Analysis of inference approaches for energy-based TTS

Inference with EBMs is traditionally performed using Langevin MCMC. This approach, as well as inference approaches used with diffusion-like models could be viewed as instances of a general-purpose optimization problem. Different optimization strategies for inference with EBMs were analyzed in Chapter 7, including higher-order optimization algorithm (Adam) that has not yet been explored with any of those models. The analysis showed that the choice of inference method is crucial for the performance of energy-based TTS. The connection between EBMs and diffusion models was explored in practice by sampling iteratively from an uninformed noise distribution. The experiment suggested that transforming an uninformed noise distribution into a speech distribution using EBMs is still a challenging and open problem.

Establishment of connections between EBMs and diffusion-like TTS models

Although EBMs rely on training an energy function, which seemingly differs from a denoising function of diffusion models, both models share the goal of iteratively transforming an initial distribution into a target distribution. Furthermore, by promoting energy functions that are

easier for inference with first-order optimisation methods (delta loss) enables to link EBMs to flow matching models.

8.2 Limitations

The work presented in this thesis has several limitations that should be considered when interpreting the results.

1. The explored investigation was limited to a relatively small set of base acoustic models and datasets. Although the residual EBM framework was shown to improve hypotheses generated by an autoregressive model [112], it remains unclear whether the same conclusions hold for other architectures, such as Transformer-TTS [68] and non-autoregressive models such as FastSpeech 2 [106]. This is important because residual EBMs learn from the gap between error distribution of a base model and the reference, and different base models may produce different types of duration, attention, pronunciation, and over-smoothing errors. The generalisation of the proposed EBM training and inference methods to multilingual and multi-speaker TTS models has also not yet been fully established.

2. The theoretical relationship between EBMs, diffusion models, and flow-matching models was explored partially. EBMs and diffusion models can both be interpreted as iterative processing that transform an initial sample into a finer data sample [114, 45, 118]. However, this thesis did not provide formal convergence guarantees for inference of EBMs in continuous spectral feature spaces. Specifically, because EBMs and diffusion models may introduce randomness at each inference step, each update is not guaranteed to improve speech quality; instead, it may preserve or introduce unpredictable degradation on the learned energy or score landscape, the step size, and the initial sample. Flow matching provides a useful comparison because it learns a vector field connecting Gaussian and data distributions [74], and therefore each inference step is designed to move samples towards the data distribution and, in principle, guarantee a progressive improvement in speech quality.

3. The acceleration of EBM inference was not explored comprehensively. The inference algorithms considered in this thesis are based on gradient-style iterative refinement, e.g., gradient descent (GD). Unlike diffusion models, for which sampling schedules and fast samplers have been studied extensively, the number of EBM refinement steps, the choice of step-size schedule, and possible early-stopping or distillation strategies remain underexplored for TTS. This limitation is practically important because the

computational cost of iterative inference may offset the quality improvements obtained from the EBM.

4. Most experiments relied on hypotheses from a pre-trained TTS model as either negative samples for training or initial samples for inference. This residual formulation is useful for improving an existing system, but it does not yet demonstrate a fully end-to-end EBM-based TTS model. One possible direction would be to train EBMs to update samples directly from a Gaussian distribution towards the data distribution, in a method conceptually similar to diffusion-based TTS models such as Grad-TTS [98] and flow-matching-based TTS models such as Matcha-TTS [81]. If successful, such a model could forward EBM-based TTS closer to an end-to-end generative framework, rather than using EBMs only as post-filtering or refinement models.

5. A further limitation concerns negative-sample design for NCE training. This thesis investigated several masking and warping strategies, but did not systematically construct negative samples that imitate common TTS failure patterns such as repetition, skipping, long pauses, incorrect duration modelling, or pronunciation errors. Since NCE training EBM is highly dependent on the informativeness of negative samples [120], future work should investigate more perceptually motivated negative-sample generation methods and assess whether they lead to more reliable improvements in human listening tests.

8.3 Future work

Applying EBMs to dicrete tokens of spectral representations

With the development of neural codec language models (e.g., VALL-E [132]), speech can be represented as a sequence of discrete tokens. Inspired by the success of EBMs in tasks with discrete outputs, future work could explore applying EBMs to rank the discrete tokens of spectral representations. The key challenge is to design a suitable metric for the discrete tokens, which is crucial for training EBMs using NCE.

EBMs for joint training of TTS

This includes two directions for the joint training of TTS. The first is to train a model similar to Grad-TTS, which replaces the diffusion model with an EBM. The EBM can be used to iteratively improve the quality of coarse spectral features generated by the encoder. The second is to synthesize speech from a noise distribution using an EBM. This research

direction has not been explored in either the computer vision (CV) domain or the TTS domain with diffusion models.

Combining energy methods with specific architectures for TTS

Diffusion methods traditionally rely on UNet architecture but have been recently successfully combined with the Transformer architecture in a single model. Although some work [34, 47] has been done on combining energy methods with the Transformer, it is worth exploring whether EBMs can be combined with the Transformer architecture to enhance the performance of speech synthesis tasks.

Applying EBMs to speech language models as decoders of continuous spectral features

Speech language models (SpeechLMs) have become an important class for modern TTS. Popular SpeechLMs, including CosyVoice 2 [27], CosyVoice 3 [26], and IndexTTS 2 [149], combine LLM-style sequence modelling with discrete speech representations. In a common SpeechLM pipeline, prompt text, prompt speech, and target text are first converted and concatenated into discrete tokens. A decoder-only transformer then predicts a sequence of target discrete speech tokens conditioned on sequences. Finally, an acoustic decoder converts these discrete tokens into continuous acoustic features, such as Mel-spectrograms, which are subsequently transformed into a waveform by a neural vocoder.

This decoder is therefore a critical bridge between discrete token modelling and continuous acoustic features. Recent systems increasingly use diffusion-based [98] or flow-matching-based [81] decoders for this stage, because these models are well suited to iterative refinement in continuous acoustic spaces and can generate high-quality spectral features from token-level conditions. This direction is closely related to the findings of this thesis: EBMs, diffusion models, and flow-matching methods can all be interpreted as defining iterative processing from an initial distribution or coarse representation towards a target speech distribution. The main difference lies in how the update direction is parameterised: diffusion and flow-matching models usually learn a denoising score or vector field directly, whereas EBMs define an energy landscape whose gradient can guide refinement.

Using EBMs as acoustic decoders in SpeechLMs remains largely unexplored. Such a decoder could receive discrete speech tokens from the language model and iteratively refine a Mel-spectrogram by descreasing the learned energy. This formulation would extend the residual and score-based EBM methods investigated in this thesis from post-filtering or refinement of existing outputs of acoustic model to a more central role inside SpeechLM TTS. It would also provide a natural paradigm for studying whether the connections between

EBMs, diffusion models, and flow matching can be exploited in large-scale token level generation. Future work could investigate inference efficiency and speech quality generated by EBM decoders against diffusion and flow-matching decoders, and explore whether energy functions can provide better controllability, robustness for SpeechLM outputs.

8.4 Conclusion

This thesis has addressed the challenges of applying energy-based models to the task of speech synthesis. The main contributions include the development of EBMs using both noise contrastive estimation and sliced score matching for training, a comprehensive analysis of various inference methods, and the establishment of theoretical and practical connections between EBMs and other generative models like diffusion and flow matching models. The findings presented in this work not only demonstrate the potential of EBMs to enhance speech synthesis quality but also provide a deeper understanding of their underlying mechanisms. The exploration of future research directions, such as applying EBMs to discrete representations and joint training frameworks, opens up new avenues for advancing the field of generative speech modeling.

References

[1] Albergo, M. S. and Vanden-Eijnden, E. (2022). Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*.

[2] Ando, A., Mori, T., Kobashikawa, S., and Toda, T. (2021). Speech emotion recognition based on listener-dependent emotion perception models. *APSIPA Transactions on Signal and Information Processing*, 10.

[3] Baba, K., Nakata, W., Saito, Y., and Saruwatari, H. (2024). The t05 system for the voicemos challenge 2024: Transfer learning from deep image classifier to naturalness mos prediction of high-quality synthetic speech. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 818–824. IEEE.

[4] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

[5] Bagshaw, P. C. (1998). Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech & Language*, 12(2):119–142.

[6] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[7] Bakhtin, A., Deng, Y., Gross, S., Ott, M., Ranzato, M., and Szlam, A. (2021). Residual energy-based models for text. *Journal of Machine Learning Research*, 22(40):1–41.

[8] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

[9] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

[10] Bhattacharyya, S., Rooshenas, A., Naskar, S., Sun, S., Iyyer, M., and McCallum, A. (2020). Energy-based reranking: Improving neural machine translation using energy-based models. *arXiv preprint arXiv:2009.13267*.

[11] Black, A., Taylor, P., Caley, R., and Clark, R. (1998). The festival speech synthesis system.

[12] Chamchong, R. and Wong, K. W. (2019). *Multi-disciplinary Trends in Artificial Intelligence: 13th International Conference, MIWAI 2019, Kuala Lumpur, Malaysia, November 17–19, 2019, Proceedings*, volume 11909. Springer Nature.

[13] Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., et al. (2022). Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518.

[14] Chen, X., Liu, X., Gales, M. J., and Woodland, P. C. (2015). Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5411–5415. IEEE.

[15] Chen, Y., Niu, Z., Ma, Z., Deng, K., Wang, C., Zhao, J., Yu, K., and Chen, X. (2024). F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching. *arXiv preprint arXiv:2410.06885*.

[16] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[17] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

[18] Chu, M. and Peng, H. (2006). Objective measure for estimating mean opinion score of synthesized speech. US Patent 7,024,362.

[19] Chu, W. and Alwan, A. (2009). Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3969–3972. IEEE.

[20] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[21] Clark et al., K. (2020). Pre-training transformers as energy-based cloze models. *arXiv preprint arXiv:2012.08561*.

[22] Cross, M. and Ragni, A. (2024). What happens to diffusion model likelihood when your model is conditional? In Coelho, C., Zimmering, B., Costa, M. F. P., Ferrás, L. L., and Niggemann, O., editors, *Proceedings of the 1st ECAI Workshop on "Machine Learning Meets Differential Equations: From Theory to Applications"*, volume 255 of *Proceedings of Machine Learning Research*, pages 1–14. PMLR.

[23] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366.

[24] Deng, Y., Bakhtin, A., Ott, M., Szlam, A., and Ranzato, M. (2020). Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*.

[25] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

[26] Du, Z., Gao, C., Wang, Y., Yu, F., Zhao, T., Wang, H., Lv, X., Wang, H., Ni, C., Shi, X., An, K., Yang, G., Li, Y., Chen, Y., Gao, Z., Chen, Q., Gu, Y., Chen, M., Chen, Y., Zhang, S., Wang, W., and Ye, J. (2025). CosyVoice 3: Towards in-the-wild speech generation via scaling-up and post-training. *arXiv preprint arXiv:2505.17589*.

[27] Du, Z., Wang, Y., Chen, Q., Shi, X., Lv, X., Zhao, T., Gao, Z., Yang, Y., Gao, C., Wang, H., et al. (2024). Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*.

[28] Du et al., Y. (2019). Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*.

[29] Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222.

[30] Elflein, S., Charpentier, B., Zügner, D., and Günnemann, S. (2021). On out-of-distribution detection with energy-based models. *arXiv preprint arXiv:2107.08785*.

[31] Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. (2014). Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth annual conference of the international speech communication association*.

[32] Gadermayr, M., Tschuchnig, M., Gupta, L., Krämer, N., Truhn, D., Merhof, D., and Gess, B. (2021). An asymmetric cycle-consistency loss for dealing with many-to-one mappings in image translation: a study on thigh mr scans. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 1182–1186. IEEE.

[33] Gales, M., Young, S., et al. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304.

[34] Gladstone, A., Nanduru, G., Islam, M. M., Han, P., Ha, H., Chadha, A., Du, Y., Ji, H., Li, J., and Iqbal, T. (2025). Energy-based transformers are scalable learners and thinkers. *arXiv preprint arXiv:2507.02092*.

[35] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[36] Grenander et al., U. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581.

[37] Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243.

[38] Guo, H., Soong, F. K., He, L., and Xie, L. (2019). A new gan-based end-to-end tts training algorithm. *arXiv preprint arXiv:1904.04775*.

[39] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

[40] He, M., Deng, Y., and He, L. (2019). Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural tts. In *Interspeech*.

[41] He, T., McCann, B., Xiong, C., and Hosseini-Asl, E. (2021). Joint energy-based model training for better calibrated natural language understanding models. *arXiv preprint arXiv:2101.06829*.

[42] He et al., M. (2019). Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural tts. *arXiv preprint arXiv:1906.00672*.

[43] Heitz, E., Belcour, L., and Chambon, T. (2023). Iterative α-(de) blending: A minimalist deterministic diffusion model. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–8.

[44] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[45] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

[46] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[47] Hoover, B., Liang, Y., Pham, B., Panda, R., Strobelt, H., Chau, D. H., Zaki, M., and Krotov, D. (2023). Energy transformer. *Advances in neural information processing systems*, 36:27532–27559.

[48] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460.

[49] Huang, W.-C., Cooper, E., Yamagishi, J., and Toda, T. (2022). Ldnet: Unified listener dependent modeling in mos prediction for synthetic speech. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 896–900. IEEE.

[50] Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings*, volume 1, pages 373–376. IEEE.

[51] Huszár, F. (2015). How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

[52] Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

[53] Hyvärinen, A., Hurri, J., Hoyer, P. O., Hyvärinen, A., Hurri, J., and Hoyer, P. O. (2009). Estimation of non-normalized statistical models. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pages 419–426.

[54] Ito, K. and Johnson, L. (2017). The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/.

[55] ITU-R (2015). Method for the subjective assessment of intermediate quality level of audio systems. Technical Report BS.1534-3, International Telecommunication Union.

[56] Jaakkola, T. and Haussler, D. (1998). Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, 11.

[57] Jayne, C., Lanitis, A., and Christodoulou, C. (2012). One-to-many neural network mapping techniques for face image synthesis. *Expert systems with applications*, 39(10):9778–9787.

[58] Kaneko et al., T. (2017). Generative adversarial network-based postfilter for stft spectrograms. In *Interspeech*, pages 3389–3393.

[59] Kim, J., Kim, S., Kong, J., and Yoon, S. (2020). Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077.

[60] Kim, J., Kong, J., and Son, J. (2021a). Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR.

[61] Kim, K., Jindal, A., Song, Y., Song, J., Sui, Y., and Ermon, S. (2021b). Imitation with neural density models. *Advances in Neural Information Processing Systems*, 34:5360–5372.

[62] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[63] Kong, J., Kim, J., and Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033.

[64] Kubichek, R. (1993). Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE pacific rim conference on communications computers and signal processing*, volume 1, pages 125–128. IEEE.

[65] Le Maguer, S., King, S., and Harte, N. (2024). The limits of the mean opinion score for speech synthesis evaluation. *Computer Speech & Language*, 84:101577.

[66] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F., et al. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0).

[67] Leng, Y., Tan, X., Zhao, S., Soong, F., Li, X.-Y., and Qin, T. (2021). Mbnet: Mos prediction for synthesized speech with mean-bias network. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 391–395. IEEE.

[68] Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6706–6713.

[69] Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., and Zhou, M. (2018). Close to human quality tts with transformer. *ArXiv*, abs/1809.08895.

[70] Li, Q., Zhang, Y., Li, B., Cao, L., and Woodland, P. C. (2021). Residual energy-based models for end-to-end speech recognition. *arXiv preprint arXiv:2103.14152*.

[71] Li, S., Du, Y., van de Ven, G. M., and Mordatch, I. (2020). Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*.

[72] Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

[73] Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T. Q., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. (2024). Flow matching guide and code. *ArXiv*, abs/2412.06264.

[74] Liu, X., Gong, C., and Liu, Q. (2022a). Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

[75] Liu, X., Staudt, D., Lin, C.-T., and Zach, C. (2022b). Effortless training of joint energy-based models with sliced score matching. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2643–2649. IEEE.

[76] Lo, C.-C., Fu, S.-W., Huang, W.-C., Wang, X., Yamagishi, J., Tsao, Y., and Wang, H.-M. (2019). Mosnet: Deep learning based objective assessment for voice conversion. *arXiv preprint arXiv:1904.08352*.

[77] Loizou, P. C. (2011). Speech quality assessment. In *Multimedia analysis, processing and communications*, pages 623–654. Springer.

[78] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[79] Ma, Z. and Collins, M. (2018). Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*.

[80] Martens, J., Sutskever, I., and Swersky, K. (2012). Estimating the hessian by back-propagating curvature. *arXiv preprint arXiv:1206.6464*.

[81] Mehta, S., Tu, R., Beskow, J., Székely, É., and Henter, G. E. (2024). Matcha-tts: A fast tts architecture with conditional flow matching. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11341–11345. IEEE.

[82] Miao, C., Liang, S., Chen, M., Ma, J., Wang, S., and Xiao, J. (2020). Flow-tts: A non-autoregressive network for text to speech based on flow. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7209–7213. IEEE.

[83] Mikolov, T. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781.

[84] Moulines, E. and Charpentier, F. (1990). Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467.

[85] Müller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.

[86] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[87] Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

[88] Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. (2018). Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR.

[89] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[90] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

[91] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

[92] Parisi, G. (1981). Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384.

[93] Park et al., D. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

[94] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

[95] Pascual et al., S. (2023). Full-band general audio synthesis with score-based diffusion. In *ICASSP*, pages 1–5. IEEE.

[96] Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.

[97] Perrotin, O., Stephenson, B., Gerber, S., Bailly, G., and King, S. (2025). Refining the evaluation of speech synthesis: A summary of the blizzard challenge 2023. *Computer Speech & Language*, 90:101747.

[98] Popov et al., V. (2021). Grad-tts: A diffusion probabilistic model for text-to-speech. In *ICML*, pages 8599–8608. PMLR.

[99] Prenger, R., Valle, R., and Catanzaro, B. (2019). Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE.

[100] Qian, Y., Fan, Y., Hu, W., and Soong, F. K. (2014). On the training aspects of deep neural network (dnn) for parametric tts synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3829–3833. IEEE.

[101] Rabiner, L., Cheng, M., Rosenberg, A., and McGonegal, C. (1976). A comparative performance study of several pitch detection algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(5):399–418.

[102] Rabiner, L. R. (2002). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[103] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015a). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

[104] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015b). Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

[105] Rao, K., Peng, F., Sak, H., and Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229. IEEE.

[106] Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2020). Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*.

[107] Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2019). Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32.

[108] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

[109] Robinson, A. and Fallside, F. (1987). Static and dynamic error propagation networks with application to speech coding. In *Neural information processing systems*.

[110] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer.

[111] Saeki, T., Maiti, S., Takamichi, S., Watanabe, S., and Saruwatari, H. (2024). Speech-BERTScore: Reference-aware automatic evaluation of speech generation leveraging nlp evaluation metrics. In *Interspeech 2024*, pages 4943–4947.

[112] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE.

[113] Shen et al., J. (2020). Non-attentive tacotron: Robust and controllable neural tts synthesis including unsupervised duration modeling. *arXiv preprint arXiv:2010.04301*.

[114] Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

[115] Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448.

[116] Song, Y., Garg, S., Shi, J., and Ermon, S. (2020a). Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR.

[117] Song, Y. and Kingma, D. P. (2021). How to train your energy-based models. *arXiv preprint arXiv:2101.03288*.

[118] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

[119] Sun, W. and Ragni, A. (2025). Score-based training for energy-based tts models. *arXiv preprint arXiv:2505.13771*.

[120] Sun, W., Tu, Z., and Ragni, A. (2024). Energy-based models for speech synthesis. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12667–12671. IEEE.

[121] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[122] Tachibana, H., Uenoyama, K., and Aihara, S. (2018). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788. IEEE.

[123] Takamichi, S., Toda, T., Neubig, G., Sakti, S., and Nakamura, S. (2014). A postfilter to modify the modulation spectrum in hmm-based speech synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 290–294. IEEE.

[124] Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge university press.

[125] Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., and Kitamura, T. (2000). Speech parameter generation algorithms for hmm-based speech synthesis. In *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*, volume 3, pages 1315–1318. IEEE.

[126] Tseng, W.-C., Huang, C.-y., Kao, W.-T., Lin, Y. Y., and Lee, H.-y. (2021). Utilizing self-supervised representations for mos prediction. *arXiv preprint arXiv:2104.03017*.

[127] Van Den Bosch, A. and Daelemans, W. (1993). Data-oriented methods for grapheme-to-phoneme conversion. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*.

[128] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[129] Veaux, C., Yamagishi, J., and MacDonald, K. (2016). Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit.

[130] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

[131] Wagner, P., Beskow, J., Betz, S., Edlund, J., Gustafson, J., Eje Henter, G., Le Maguer, S., Malisz, Z., Szekely, E., Tånnander, C., et al. (2019). Speech synthesis evaluation—state-of-the-art assessment and suggestion for a novel research program. In *Proceedings of the 10th Speech Synthesis Workshop (SSW10)*.

[132] Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., et al. (2023). Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

[133] Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017). Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*.

[134] Wang, Z. and Ragni, A. (2021). Approximate fixed-points in recurrent neural networks. *arXiv preprint arXiv:2106.02417*.

[135] Wang et al., C. (2021). fairseq sˆ 2: A scalable and integrable speech synthesis toolkit. *arXiv preprint arXiv:2109.06912*.

[136] Welling et al., M. (2004). Exponential family harmoniums with an application to information retrieval. *Advances in neural information processing systems*, 17.

[137] Weng, L. (2021). What are diffusion models? *lilianweng.github.io*.

[138] Wenliang, L., Sutherland, D. J., Strathmann, H., and Gretton, A. (2019). Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746. PMLR.

[139] Williams et al., R. J. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

[140] Xue et al., H. (2022). Learn2sing 2.0: Diffusion and mutual information-based target speaker svs by learning from singing teacher. *arXiv preprint arXiv:2203.16408*.

[141] Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., Chen, Z., and Wu, Y. (2019). Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*.

[142] Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A. W., and Tokuda, K. (2007). The hmm-based speech synthesis system (hts) version 2.0. *SSW*, 6:294–299.

[143] Zen, H. and Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4470–4474. IEEE.

[144] Zen, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *2013 ieee international conference on acoustics, speech and signal processing*, pages 7962–7966. IEEE.

[145] Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064.

[146] Zhang, J.-X., Ling, Z.-H., and Dai, L.-R. (2018). Forward attention in sequence-to-sequence acoustic modeling for speech synthesis. In *2018 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 4789–4793. IEEE.

[147] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

[148] Zheng, Y., Wang, X., He, L., Pan, S., Soong, F. K., Wen, Z., and Tao, J. (2019). Forward-backward decoding for regularizing end-to-end tts. *arXiv preprint arXiv:1907.09006*.

[149] Zhou, S., Zhou, Y., He, Y., Zhou, X., Wang, J., Deng, W., and Shu, J. (2026). Indextts2: A breakthrough in emotionally expressive and duration-controlled auto-regressive zero-shot text-to-speech. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35139–35148.

[150] Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. (2017). Toward multimodal image-to-image translation. *Advances in neural information processing systems*, 30.