1 A Deep Dive into DiffWave and Diffusion-Based Audio Synthesis

1.1 Abstract

DiffWave is a diffusion probabilistic model that enables high-fidelity audio synthesis using a non-autoregressive denoising process. While the original work demonstrates strong empirical performance, several aspects of its probabilistic formulation and training objective are often presented concisely, making it difficult for new researchers to follow the full derivation. In this paper, we provide a pedagogical and self-contained exposition of DiffWave, focusing on the diffusion process, the evidence lower bound (ELBO) derivation, and the resulting noise-prediction objective. We further explain the architectural design choices and conditioning mechanisms from a probabilistic perspective. This work does not propose a new model or training method; rather, it aims to improve clarity and accessibility for students and researchers seeking a rigorous understanding of diffusion-based audio synthesis.

1.2 Introduction

DiffWave is a versatile **diffusion probabilistic model** for both **conditional** and **unconditional waveform generation**.

DiffWave has the following characteristics:

- It is **non-autoregressive**.
- It converts a **white noise signal** into a structured waveform through a **Markov chain** with a **constant number of steps during synthesis**.
- It is efficiently trained by **optimizing a variant of the variational bound**.

1.3 Scope and Contribution

1.3.1 Scope.

This paper is intended as a tutorial and expository reference for diffusion-based audio models, with DiffWave as the primary case study. The focus is on mathematical clarity, consistent notation, and intuitive explanation of design choices.

1.3.2 Contribution.

- A step-by-step derivation of the DiffWave training objective starting from the intractable data likelihood and leading to the noise-prediction loss.
- A unified presentation of the forward diffusion process, reverse denoising model, and their connection through the ELBO.
- An intuitive explanation of how architectural choices (non-autoregressive WaveNet-style backbone, diffusion-step embeddings, and conditioning injection) align with the probabilistic formulation.
- A consolidated reference that bridges diffusion theory and practical neural vocoder design.

1.4 Related Work

1.4.1 DiffWave

DiffWave extends diffusion probabilistic modeling to raw audio waveform synthesis, demonstrating strong performance in both conditional (neural vocoding) and unconditional generation settings. By predicting the additive noise at each diffusion step rather than the waveform directly, DiffWave enables stable training with a single objective and parallel waveform generation during inference. Its non-autoregressive formulation allows synthesis that is orders of magnitude faster than traditional autoregressive models while maintaining comparable perceptual quality.

1.4.2 Denoising Diffusion Probabilistic Models (DDPM)

Denoising Diffusion Probabilistic Models formalize generative modeling as a learned reverse process of a fixed Gaussian noising chain. DDPMs introduce the ELBO-based training framework and show that, under specific parameterizations, the objective reduces to denoising score matching or noise prediction. DiffWave directly builds on this formulation, adapting it to high-dimensional audio signals and integrating it with convolutional architectures suitable for waveform modeling.

1.5 The Direct Likelihood Approach and Its Failure

Consider a dataset consisting of waveforms

$$x_0 \sim q_{data}(x_0)$$

A natural approach would be to define the model as $p_\theta(x_0)$ and optimize the objective:

$$\max_\theta \quad E_{x_0 \sim q_{data}}[\log p_\theta(x_0)]$$

However, this approach **does not work** because $p_\theta(x_0)$ is **intractable**.

1.5.1 Why the Likelihood is Intractable

In diffusion models, the data sample x_0 is generated through a sequence of latent variables:

$$x_1, x_2, \ldots, x_T$$

Therefore, the likelihood becomes:

$$p_\theta(x_0) = \int p_\theta(x_0, x_1, \ldots, x_T) dx_{1:T}$$

This integral marginalizes **all intermediate noise variables** $x_{1:T}$, which is computationally infeasible. Hence, the likelihood $p_\theta(x_0)$ is **intractable**.

Diffusion models can use a diffusion (noise-adding) process **without learnable parameters** to obtain the "whitened" latents from training data. Therefore, **no additional neural networks are required for training**, in contrast to other models (e.g., the encoder in VAE or the discriminator in GAN). This avoids challenging issues such as **posterior collapse** and **mode collapse**, which stem from joint training of multiple networks.

1.6 Advantages of DiffWave

DiffWave has several advantages over previous work:

1. **Non-autoregressive**
    - Synthesize high-dimensional waveforms **in parallel**.
2. **Architectural flexibility**
    - Does not impose architectural constraints, unlike **flow-based models**, which must maintain a bijection between latents and data.
    - Enables **small-footprint neural vocoders** with high-fidelity speech generation.

3. **Single training objective**
    - Uses a **single ELBO-based objective**.
    - Does not rely on auxiliary losses (e.g., spectrogram-based losses).
4. **Versatility**
    - Produces high-quality audio for both **conditional** and **unconditional** waveform generation.

1.7 Contributions

Specifically, the following contributions are made [1]:

1. **Architecture**
    - Uses a **feed-forward, bidirectional dilated convolution** architecture inspired by **WaveNet**.
    - Matches WaveNet vocoder quality (MOS: **4.44 vs. 4.43**).
    - Synthesizes **orders of magnitude faster**, requiring only a few sequential steps (e.g., **6**) even for very long waveforms.
2. **Efficiency and footprint**
    - A small DiffWave model has **2.64M parameters**.
    - Synthesizes **22.05 kHz high-fidelity speech** with MOS **4.37**.
    - Runs **>5× faster than real-time** on a V100 GPU without engineered kernels.
    - Although slower than state-of-the-art flow-based models, it has a **much smaller footprint**.
    - Further speed-ups are expected through inference optimization.
3. **Unconditional generation performance**
    - Significantly outperforms **WaveGAN** and **WaveNet**.
    - Excels in **unconditional** and **class-conditional** waveform generation.
    - Achieves better audio quality and sample diversity based on both **automatic** and **human evaluations**.

1.8 Diffusion Probabilistic Models

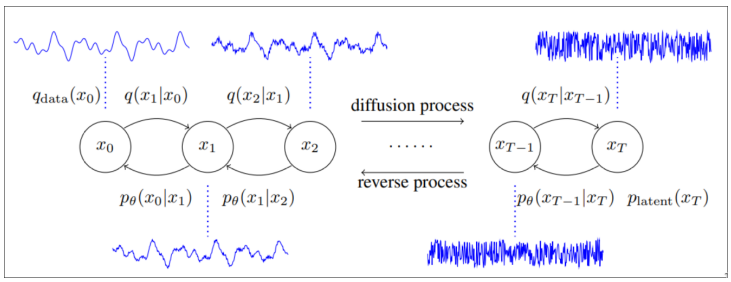

Figure 1: The diffusion and reverse process in diffusion probabilistic models [1]

We define the following notation:

- $q_{data}(x_0)$: data distribution on R^L
- L: data dimension
- Let $x_t \in R^L$ for $t = 0, 1, \ldots, T$ be a sequence of variables with the same dimension
- t: index for diffusion steps

1.8.1 Forward Diffusion Process

The forward diffusion process is defined by a fixed Markov chain from data x_0 to the latent variable x_T [2]:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

1.8.2 Motivation for the Markov Assumption

The Markov assumption provides several benefits:

- Forward noising becomes simple: $q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$. This is a simple Gaussian noise step that is stable.
- It gives a closed form for sampling at any time: $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$. If it wasn't Markov, transitions like $q(x_t \mid x_{t-1}, x_{t-2}, ..., x_0)$ would be required, which is difficult to design and sample from.
- Reverse process becomes learnable step-by-step: $p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$. The model only needs to learn $p_\theta(x_{t-1} \mid x_t)$, not a complex global conditional involving the entire past.

1.8.3 Diffusion Process Specification

The forward transition is defined as:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$$

where $\beta_t = linspace(\beta_{\min}, \beta_{\max}, T)$. For training with $T = 200$ steps: $\beta_t \in [10^{-4}, 0.02]$ (linearly spaced).

The function of $q(x_t \mid x_{t-1})$ is to add small Gaussian noise to the distribution of x_{t-1}. The whole process gradually converts data x_0 to whitened latents x_T according to a variance schedule $\beta_1, \ldots, \beta_T$.

We define: - $p_{latent}(x_T) = \mathcal{N}(0, I)$ is isotropic Gaussian

1.8.4 Reverse Process

The reverse process is parameterized as:

$$p_\theta(x_0, \ldots, x_{T-1} \mid x_T) = \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

The transition probability is:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)^2 I)$$

where $\mu_\theta \in R^L$ and $\sigma_\theta \in R$.

The goal of $p_\theta(x_{t-1} \mid x_t)$ is to eliminate the Gaussian noise (i.e., denoise) added in the diffusion process.

1.8.5 Sampling Procedure

Given the reverse process, the generative procedure is to first sample $x_T \sim \mathcal{N}(0, I)$, and then sample $x_{t-1} \sim p_\theta(x_{t-1} \mid x_t)$ for $t = T, T-1, \ldots, 1$. The output x_0 is the sampled data.

1.8.6 Training Objective

The likelihood $p_\theta(x_0) = \int p_\theta(x_0, \ldots, x_{T-1}|x_T) \cdot p_{latent}(x_T)dx_{1:T}$ is intractable to calculate in general. The model is thus trained by maximizing its variational lower bound (ELBO).

1.9 Derivation of the Evidence Lower Bound

We have:

$$p_\theta(x_0, \ldots, x_T) = p_{latent}(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

The likelihood of a datapoint x_0 is:

$$p_\theta(x_0) = \int p_\theta(x_0, \ldots, x_T) dx_{1:T}$$

This integral is intractable. The forward process (noising chain) is:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

Now multiply and divide inside the integral by this same quantity (this is just multiplying by 1):

$$p_\theta(x_0) = \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_0, \ldots, x_{T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)} dx_{1:T}$$

For any function f:

$$E_{q(z)}[f(z)] = \int q(z) f(z) dz$$

Therefore:

$$p_\theta(x_0) = E_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_0, \ldots, x_{T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)} \right]$$

Now take log and average over real data $x_0 \sim q_{data}$:

$$E_{q_{data}(x_0)} \log p_\theta(x_0) = E_{q_{data}(x_0)} \log E_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_0, \ldots, x_{T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)} \right]$$

Define the random variable (for fixed x_0):

$$Y := \frac{p_\theta(x_0, \cdots, x_{T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)}$$

Then:

$$E_{q_{data}(x_0)} \log p_\theta(x_0) = E_{q_{data}(x_0)} \log E_{q(x_{1:T}|x_0)} [Y]$$

1.10 Application of Jensen's Inequality

Key fact: - $\log(\cdot)$ is **concave** - For concave ϕ, Jensen's inequality states:

$$\phi(E[Y]) \geq E[\phi(Y)]$$

With $\phi = \log$:

$$\log E[Y] \geq E[\log Y]$$

Therefore:

$$\log E_{q(x_{1:T}|x_0)}[Y] \geq E_{q(x_{1:T}|x_0)}[\log Y]$$

Taking expectation over $x_0 \sim q_{data}$ on both sides:

$$E_{q_{data}(x_0)} \log E_{q(x_{1:T}|x_0)}[Y] \geq E_{q_{data}(x_0)} E_{q(x_{1:T}|x_0)}[\log Y]$$

Note that:

$$q(x_{0:T}) = q_{data}(x_0) q(x_{1:T} \mid x_0)$$

Therefore:

$$E_{q_{data}(x_0)} E_{q(x_{1:T}|x_0)}[\cdot] = E_{q(x_{0:T})}[\cdot]$$

The right-hand side becomes:

$$E_{q(x_{0:T})} [\log Y]$$

We obtain:

$$E_{q_{data}(x_0)} \log p_\theta(x_0) \geq E_{q(x_{0:T})} \log \frac{p_\theta(x_0, \cdots, x_{T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)} := ELBO$$

1.11 Parameterization of the Reverse Process

Ho et al. (2020) showed that under a certain parameterization, the ELBO of the diffusion model can be calculated in closed-form. This accelerates the computation and avoids Monte Carlo estimates, which have high variance. This parameterization is motivated by its connection to denoising score matching with Langevin dynamics. To introduce this parameterization, we first define some constants based on the variance schedule $\{\beta_t\}_{t=1}^T$ in the diffusion process:

$$\alpha_t = 1 - \beta_t, \qquad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t for t > 1 and \tilde{\beta}_1 = \beta_1.$$

The parameterizations of μ_θ and σ_θ are defined by:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right), \qquad and \qquad \sigma_\theta(x_t, t) = \tilde{\beta}_t^{\frac{1}{2}}.$$

where $\epsilon_\theta : R^L \times N \to R^L$ is a neural network also taking x_t and the diffusion-step t as inputs. Note that $\sigma_\theta(x_t, t)$ is fixed to a constant $\tilde{\beta}_t^{\frac{1}{2}}$ for every step t under this parameterization.

1.11.1 Summary of Forward and Reverse Processes

Forward diffusion (known, fixed):

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t I\right), \quad \alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Reverse model (learned):

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)^2 I\right)$$

DiffWave uses the Ho et al. parameterization:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\Big), \qquad \sigma_\theta(x_t, t) = \sqrt{\tilde{\beta}_t}$$

where

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.$$

1.12 ELBO Expansion into KL Divergence Terms

From the ELBO definition, we have:

$$\text{ELBO} = E_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T-1} \mid x_T) p_{latent}(x_T)}{q(x_{1:T} \mid x_0)}$$

where:

- **forward/diffusion**:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

- **reverse model**:

$$p_\theta(x_{0:T-1} \mid x_T) = \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

Substituting the products inside the log:

$$\text{ELBO} = E_q\Big[\log p_{latent}(x_T) + \sum_{t=1}^{T} \log p_\theta(x_{t-1} \mid x_t) - \sum_{t=1}^{T} \log q(x_t \mid x_{t-1})\Big]$$

To convert the forward term $\log q(x_t \mid x_{t-1})$ into a KL divergence, we use the key identity. Because the forward chain is Markov:

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1})q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}$$

Taking log:

$$\log q(x_t \mid x_{t-1}) = \log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0)$$

Rearranging:

$$\log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})} = \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \log \frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}$$

Applying this rewrite to all terms $t = 2, \ldots, T$ while keeping $t = 1$ separate:

$$\text{ELBO} = E_q\Big[\log p_{latent}(x_T) - \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)} - \sum_{t=2}^{T} \Big(\log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} + \log \frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)}\Big)\Big]$$

The last ratio terms form a telescoping sum:

$$\sum_{t=2}^{T} \log \frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)} = \log q(x_T \mid x_0) - \log q(x_1 \mid x_0)$$

After cancellation, we obtain:

$$\text{ELBO} = E_q\Big[\log \frac{p_{latent}(x_T)}{q(x_T \mid x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \log p_\theta(x_0 \mid x_1)\Big]$$

1.13 Conversion to KL Divergence Form

Recall:

$$\text{KL}(q|p) = E_q\left[\log \frac{q}{p}\right] \quad \Rightarrow \quad E_q\left[\log \frac{p}{q}\right] = -\text{KL}(q|p)$$

1.13.1 Latent Term

$$E_q\Big[\log \frac{p_{latent}(x_T)}{q(x_T \mid x_0)}\Big] = -\text{KL}(q(x_T \mid x_0)|p_{latent}(x_T))$$

1.13.2 Intermediate Diffusion Steps

$$E_q\Big[\log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)}\Big] = -\text{KL}(q(x_{t-1} \mid x_t, x_0)|p_\theta(x_{t-1} \mid x_t))$$

Therefore:

$$\text{ELBO} = E_q\Big[-\text{KL}(q(x_T \mid x_0)|p_{latent}(x_T)) - \sum_{t=2}^{T} \text{KL}(q(x_{t-1} \mid x_t, x_0)|p_\theta(x_{t-1} \mid x_t)) + \log p_\theta(x_0 \mid x_1)\Big]$$

Negating both sides:

$$-\text{ELBO} = E_q\Big[\text{KL}(q(x_T \mid x_0)|p_{latent}) + \sum_{t=2}^{T} \text{KL}(q(x_{t-1} \mid x_t, x_0)|p_\theta(x_{t-1} \mid x_t)) - \log p_\theta(x_0 \mid x_1)\Big].$$

The first KL becomes a constant c with respect to θ.

1.14 Closed-Form KL Divergence for Gaussian Distributions

We have:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right).$$

It can be shown that $q(x_{t-1} \mid x_t, x_0)$ is Gaussian:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

with

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t.$$

For $t \geq 2$: - $q(x_{t-1} \mid x_t, x_0)$ is Gaussian with covariance $\tilde{\beta}_t I$ - $p_\theta(x_{t-1} \mid x_t)$ is Gaussian with covariance $\tilde{\beta}_t I$ (by design)

The KL divergence between two Gaussians with the same covariance simplifies to:

$$\mathrm{KL}\big(\mathcal{N}(m_1, \Sigma)|\mathcal{N}(m_2, \Sigma)\big) = \frac{1}{2}(m_1 - m_2)^\top \Sigma^{-1}(m_1 - m_2)$$

Here $\Sigma = \tilde{\beta}_t I$, so:

$$\mathrm{KL}(q|p_\theta) = \frac{1}{2\tilde{\beta}_t}|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)|_2^2.$$

Substituting:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\Big).$$

Also using the forward reparameterization:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon.$$

Through algebraic manipulation, the mean difference collapses to a constant times:

$$\epsilon - \epsilon_\theta(x_t, t)$$

Therefore, the KL becomes:

$$E_q\mathrm{KL}(q(x_{t-1} \mid x_t, x_0)|p_\theta(x_{t-1} \mid x_t)) = \frac{\beta_t}{2\alpha_t(1-\bar{\alpha}_{t-1})}E_{x_0,\epsilon}|\epsilon - \epsilon_\theta(x_t, t)|_2^2.$$

This gives the weight:

$$\kappa_t = \frac{\beta_t}{2\alpha_t(1-\bar{\alpha}_{t-1})} \quad (t > 1)$$

The remaining term is $-\log p_\theta(x_0 \mid x_1)$. Since:

$$x_1 = \sqrt{\alpha_1}x_0 + \sqrt{1-\alpha_1}\epsilon,$$

it can be shown that:

$$-E_q \log p_\theta(x_0 \mid x_1) = \frac{1}{2\alpha_1}E_{x_0,\epsilon}|\epsilon - \epsilon_\theta(x_1, 1)|_2^2 + const.$$

Therefore:

$$\kappa_1 = \frac{1}{2\alpha_1}$$

1.15 Final ELBO Expression

Collecting all θ-independent constants into c and summing the weighted MSEs from $t = 1$ to T:

$$-\mathrm{ELBO} = c + \sum_{t=1}^{T} \kappa_t E_{x_0,\epsilon}\left|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right|_2^2$$

1.16 Unweighted Loss Variant

Ho et al. (2020) reported that minimizing the following unweighted variant of the ELBO leads to higher generation quality:

$$\min_{\theta} \; L_{\text{unweighted}}(\theta) = E_{x_0,\epsilon,t} \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon,\; t\right) \right\|_2^2$$

1.17 Diffusion-Step Embedding

It is important to include the diffusion-step t as part of the input, as the model needs to output different $\epsilon_\theta(\cdot, t)$ for different t.

A 128-dimensional encoding vector is used for each t:

$$t_{\text{embedding}} = \left[\sin\left(10^{\frac{0\times4}{63}}\, t\right), \ldots, \sin\left(10^{\frac{63\times4}{63}}\, t\right), \cos\left(10^{\frac{0\times4}{63}}\, t\right), \ldots, \cos\left(10^{\frac{63\times4}{63}}\, t\right) \right]$$

Three fully connected (FC) layers are applied to the encoding, where the first two FCs share parameters among all residual layers. The last residual-layer-specific FC maps the output of the second FC into a C-dimensional embedding vector. This embedding vector is broadcast over length and added to the input of every residual layer.

1.18 DiffWave Architecture

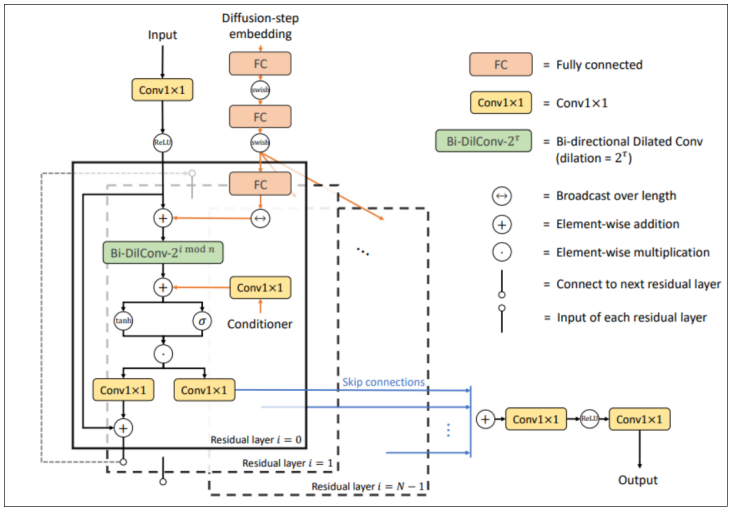

Figure 2: The network architecture of DiffWave [1]

1.18.1 Core Idea

DiffWave is a diffusion model where the neural network does not predict the waveform directly but predicts the **noise component** $\epsilon_\theta(x_t, t)$. This prediction is used to compute the reverse denoising step parameters $\mu_\theta(x_t, t)$ (mean), while the variance is fixed by the schedule.

1.18.2 Key Architectural Choice: Non-Autoregressive WaveNet-Style Backbone

Unlike WaveNet, DiffWave has **no causality constraint** because generation is not left-to-right autoregressive. It uses:

- **Bidirectional dilated convolutions** (not causal)
- **Residual stack with skip connections** like WaveNet
- **Kernel size = 3**

- Dilation cycles doubling inside each block: $[1, 2, 4, \ldots, 2^{n-1}]$. This gives large receptive fields per denoising step while remaining efficient.

Each residual layer takes: - current noisy audio (x_t) - diffusion step embedding (t) - optional conditioner (mel or label)

The processing steps are: 1. Dilated convolution 2. Add conditioning bias 3. Gated activation (tanh/sigmoid-like gating) 4. Produces: - residual output to next layer - skip output to final head

This is a WaveNet residual block but bidirectional and diffusion-conditioned.

1.18.3 Diffusion-Step Embedding Integration

DiffWave needs to denoise at different noise levels ,so the network must behave differently for each step t. The diffusion step is encoded using a sinusoidal positional embedding (Transformer-style):

$$t_{embedding} = [\sin(10^{i \cdot 4/63} t), \cos(10^{i \cdot 4/63} t)]$$

(128-D total).

The encoding is then: - passed through **3 FC layers** - first two FCs are shared across layers - last FC is layer-specific - broadcast over time axis and added into each residual layer input

Therefore, diffusion-step conditioning is injected deeply, not just at the input.

1.19 Conditional Generation

DiffWave supports conditioning in two modes:

1.19.1 Local Conditioner (Mel Spectrogram Vocoding)

This is the classic TTS-vocoder setup.

Pipeline: 1. Take mel spectrogram (80-band) 2. **Upsample** it to waveform length using transposed 2D convolution layers 3. In each residual layer: - apply a layer-specific 1×1 convolution mapping mel $\rightarrow 2C$ channels - add it as **bias term** into the residual block (right before gated nonlinearity)

Every residual layer receives local time-aligned guidance, not only the first layer.

1.19.2 Global Conditioner (Discrete Labels)

For class-conditional generation (digit labels, speaker IDs, etc.): - use shared label embedding ($d_{label} = 128$) - in each residual layer: - map label embedding $\rightarrow 2C$ channels via layer-specific 1×1 convolution - add as a bias term after dilated convolution

This makes conditioning global but injected deeply.

1.20 Unconditional Generation

Unconditional waveform generation is difficult because audio is long (example: 16k samples for 1 second at 16kHz), and coherence requires a **very large receptive field**.

1.20.1 Receptive Field Limitation in Dilated Convolution Stacks

For a dilated convolution stack, the receptive field is:

$$r = (k-1) \sum_i d_i + 1$$

Even a 30-layer model with dilation cycle up to 512 only gives a receptive field of approximately 6k samples (approximately 0.38 seconds at 16kHz), which is too small to maintain consistency over a full utterance.

1.20.2 DiffWave's Advantage: Receptive Field Multiplies Across Denoising Steps

DiffWave runs **T denoising iterations**. The effective receptive field can grow roughly like:

$$T \times r$$

This is the key DiffWave advantage for unconditional audio: long-range structure emerges progressively across denoising steps. This is why it performs well in unconditional settings where autoregressive WaveNet tends to collapse into unstable speech-like noises.

1.21 Experiments

DiffWave is evaluated on three major tasks:

1.21.1 Neural Vocoding (Mel → Waveform)

- Dataset: **LJ Speech** (approximately 24 hours, 22.05kHz, single speaker)
- Conditioning: 80-band mel spectrogram
- DiffWave variants:
  - varying diffusion steps ($T \in \{20, 40, 50, 200\}$)
  - varying residual channels ($C \in \{64, 128\}$)
- Baselines compared:
  - WaveNet
  - ClariNet
  - WaveGlow
  - WaveFlow

Fast sampling ($T_{infer} \ll T$) is also tested using a modified inference schedule, but using the same trained checkpoint.

1.21.2 Unconditional Waveform Generation

- Dataset: **SC09** (Speech Commands digits subset)
  - 0–9 digit utterances
  - 1 second audio at 16kHz ($L = 16000$)
- Models compared:
  - WaveNet (different channel sizes)
  - WaveGAN
  - DiffWave with larger dilation cycle (up to 2048) and more layers
- Key evaluation:
  - human MOS
  - classifier-based metrics (using a ResNeXT classifier trained on SC09 features/logits)

1.21.3 Class-Conditional Generation (Digit Labels as Conditioner)

- Same dataset as unconditional (SC09)
- Conditioning: injected digit label using the global conditioner method
- Compared to WaveNet under the same label conditioning setup

1.22 Conclusion

This paper presented a detailed and pedagogical exposition of DiffWave, emphasizing the probabilistic foundations underlying diffusion-based audio synthesis. By explicitly deriving the ELBO, explaining the noise-prediction objective, and connecting these elements to architectural design choices, we aim to make DiffWave more accessible to readers with a theoretical background in machine learning. We hope this exposition serves as a useful reference for students and researchers entering the area of diffusion models for audio and other continuous signal domains.

1.23 Limitations

This work is purely expository in nature. It does not introduce new models, training objectives, or experimental results, nor does it provide a comprehensive empirical comparison with alternative diffusion-based audio models. The analysis is limited to the formulation presented in prior work, and performance claims rely entirely on previously published results. Future work could complement this exposition with empirical reproductions, ablation studies, or extensions to related diffusion architectures.

References

[1] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *International Conference on Learning Representations*, 2021.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.