FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Super-Resolution Enhancement of Weather Data Using Diffusion Models
Student:	Jan-Matyáš Martinů
Supervisor:	Mgr. Petr Šimánek
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2025/2026

Instructions

Topic Overview:
The accurate forecasting and analysis of weather patterns heavily depend on the resolution of available meteorological data. Super-resolution (SR) techniques, particularly those based on deep learning, have emerged as powerful tools to enhance the spatial resolution of weather datasets, thereby improving the precision of weather predictions, climate modeling, and environmental monitoring. Among the deep learning methods, diffusion-based models, such as Denoising Diffusion Probabilistic Models (DDPM), have shown promising results in generating high-resolution imagery from low-resolution inputs. This project compares two diffusion-based super-resolution models, SRDiff and ResDiff, using the WeatherBench dataset—a comprehensive dataset for benchmarking weather forecasts. By enhancing the resolution of weather data, these models have the potential to significantly contribute to more accurate and detailed meteorological analyses and predictions, facilitating better-informed decisions in fields ranging from agriculture to disaster preparedness and climate change adaptation.

Tasks for the Student:

1/ Literature Review: Survey the current landscape of super-resolution techniques with a focus on diffusion models in the context of weather data enhancement. Highlight the strengths and weaknesses of these approaches, particularly their ability to capture and reconstruct fine-scale atmospheric features.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 28 February 2024 in Prague.

2/ In-depth Method Study: Delve into the specifics of the SRDiff and ResDiff models. Understand the underlying mechanisms that enable these diffusion-based models to enhance the resolution of weather data, paying special attention to their model architectures, training strategies, and the theoretical principles that guide their performance.
3/ Dataset Acquisition and Preprocessing: Obtain the WeatherBench dataset and perform the necessary preprocessing steps. This includes data cleaning, normalization, and any other transformations required to make the data compatible with the input requirements of both diffusion models.
4/ Model Implementation and Training: Implement the SRDiff and ResDiff models using the deep-learning framework PyTorch. Train the models on the preprocessed WeatherBench dataset, fine-tuning the hyperparameters and employing robust validation strategies to ensure the models are well-optimized and generalizable.
5/ Performance Evaluation: Assess the performance of both models using appropriate evaluation metrics tailored to super-resolution, such as the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) or MSE. Conduct a comparative analysis to identify which model demonstrates superior capability in enhancing the resolution of weather data.
6/ Error Analysis and Model Limitations: Analyze instances where the models reconstruct high-resolution weather data accurately. Identify and discuss the limitations of each model, considering aspects like computational efficiency, scalability, and sensitivity to input data quality.
7/ Documentation and Implications: Thoroughly document the research process, including model architectures, dataset preparation, training protocols, and evaluation results. Discuss the potential implications of the findings of meteorology and climate science, especially in improving the accuracy and utility of weather forecasts and climate models.

Bachelor's thesis

SUPER-RESOLUTION ENHANCEMENT OF WEATHER DATA USING DIFFUSION MODELS

Jan-Matyáš Martinů

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Mgr. Petr Šimánek
May 16, 2024

Citation of this thesis: Martinů Jan-Matyáš. *Super-Resolution Enhancement of Weather Data Using Diffusion Models.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

List of code listings

I would like to thank my supervisor, Mgr. Petr Šimánek, for his time, guidance, and support throughout this project. Additionally, I would like to thank the FIT Data Science Laboratory for providing the computational resources, without which this work could not have been realized.

Declaration

In Prague on May 16, 2024

Abstract

This thesis investigates the application of advanced deep-learning diffusion models, specifically SR3, SRDiff, and ResDiff architectures, for super-resolution of weather data. The primary focus is to evaluate these model's capability to enhance the resolution of meteorological variables from low-resolution inputs, a crucial aspect for an accurate weather forecasting and climate analysis.

Through experiments, conducted using the WeatherBench dataset, this work compares the performance of these models using a variety of validation metrics and explores enhancements through physics-based modifications and architectural improvements. The findings indicate that SRDiff, and ResDiff, further improved by incorporating physics-based filters, significantly outperform the traditional SR3 method, offering substantial improvements in capturing high-frequency details essential for accurate meteorological representations.

This thesis underscores the potential of integrating artificial intelligence with meteorological science to advance weather prediction capabilities, setting a foundation for future improvements in deep-learning diffusion models for weather data super-resolution.

Keywords weather modelling, denoising diffusion probabilistic models, super-resolution, PyTorch, deep learning

Abstrakt

Tato práce se zabývá použitím pokročilých difuzních modelů založených na hlubokém učení a to konkrétně architektur SR3, SRDiff a ResDiff, pro superrozlišení meteorologických dat. Hlavním cílem je posoudit schopnost těchto modelů zvyšovat rozlišení meteorologických proměnných ze vstupů s nízkým rozlišením. Tato schopnost je klíčová pro analýzu klimatu a přesnou předpověď počasí.

Prostřednictvím experimentů prováděných na WeatherBench datasetu, tato práce porovnává tyto modely pomocí různých validačních metrik a vylepšuje jejich architekturu. Výsledky experimentů ukazují, že SRDiff, ResDiff a jeho varianta vylepšená pomocí fyzikálních konvolučních filtrů, výrazně překonávají původní SR3 model a lépe zachycují vysokofrekvenční detaily důležité pro přesnou reprezentaci počasí.

Tato práce ukazuje potenciál využití umělé inteligence v meteorologii a vytváří základ pro budoucí pokrok difuzních modelů v tomto oboru.

Klíčová slova modelování počasí, difúzní modely, superrozlišení, PyTorch, hluboké učení

Seznam zkratek

DDPM	Denoising Diffusion Probabilistic Models
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
ResDiff	Residual-structure-based diffusion model
HR	High-Resolution
LR	Low-Resolution
SR	Super-Resolution
ResNet	Residual Network
FFT	Fast Fourier transform
ResSE	Residual Squeeze-and-Excitation
GCM	Global climate model
VLB	Variational lower bound
DWT	Discrete Wavelet Transform
RRDB	Residual in residual dense block
ReLU	Rectified linear unit
MSE	Mean squared error
SSIM	Structural similarity index
PSNR	Peak signal-to-noise ratio
MAE	Mean absolute error

Introduction

Global climate models are used to study global weather. These models operate on a global and continental scale and thus are unable to capture local weather information. Local weather information is crucial to society as it affects many aspects of our daily lives, such as choice of clothing, planning outdoor activities, energy production, transportation or agriculture. While global climate models provide valuable climate information, they can't provide detailed forecasts for a specific place at a specific time.

To address this limitation, one promising approach is the super-resolution of weather data. This technique enhances the resolution of weather variables measured at lower resolutions.

In the field of super-resolution currently excel deep learning models. There are many classes of deep learning models used for the super-resolution of images. One of these classes is known as diffusion models, which have recently shown excellent results in generating high-quality images and are therefore integrated into models like ChatGPT-4.

Diffusion models work by a process that gradually adds noise to an image, transforming it into a Gaussian noise distribution, and then learning to reverse this process to reconstruct the original image from the noise. [1]

Because diffusion models offer a promising approach to super-resolution, this thesis aims to explore the application of advanced deep-learning diffusion models, specifically, SR3 [2], SRDiff [3], and ResDiff [4], for the super-resolution of weather data, to compare them, and to improve their performance through the integration of physics-based modifications and architectural enhancements.

This work is organized into four chapters. Chapter 1 introduces the concept of super-resolution and its application in meteorology, detailing various techniques. Chapter 2 provides explanation of diffusion models, detailing the underlying mathematical principles, various architectures, and advanced variants specifically designed for super-resolution applications that are utilized in this work. Chapter 3 provides an overview of the WeatherBench dataset [5] and describes the implementation details. In Chapter 4 are presented conducted experiments and analysis of their outcomes.

Chapter 1

Weather Super-Resolution

1.1 Super-Resolution

Single Image Super-Resolution (SISR) is a technique in image processing and computer vision that reconstructs a high-resolution (HR) image from a single low-resolution (LR) image. The primary challenge of SISR lies in efficiently extracting useful features from the LR input and utilizing them to produce an HR image with enhanced details. Furthermore, this process is complicated by the possibility of generating multiple HR images from the same LR image, resulting in a one-to-many mapping issue that adds complexity to the reconstruction process. The approaches to address this challenge fall into two broad categories: traditional methods and deep learning-based methods. [6, 7]

Traditional SR methods primarily include interpolation and reconstruction-based techniques. Interpolation methods, such as Bicubic and Bilinear Interpolation [8], are straightforward and fast, offering basic image enhancement by filling in new pixel values based on nearby pixels. However, these methods often lack accuracy and can lead to blurred images, especially at higher magnifications. Reconstruction-based SR techniques [9] improve upon interpolation by using prior knowledge to impose constraints that guide the high-resolution image creation. These methods, however, tend to falter as the desired resolution scale increases. [6]

The domain of super-resolution has been revolutionized with the advent of deep learning, particularly through the use of Convolutional Neural Networks (CNNs) [10]. The first significant model in this area was the Super-Resolution Convolutional Neural Network (SRCNN) [11], which demonstrated that CNNs could effectively learn mapping from low-resolution to high-resolution images. Following SRCNN, there have been significant advancements with the introduction of models such as Very Deep Super Resolution (VDSR) [12], which utilizes deeper networks for better performance, and Enhanced Deep Super-Resolution network (EDSR) [13] that employs deeply Recursive Convolutional Networks for refined learning processes.

Generative Adversarial Networks (GANs) [14] are another class of machine learning models that achieve impressive results in super-resolution. The architecture of GANs includes two neural networks: a generator and a discriminator. These networks are trained together through adversarial processes. The generator's goal is to create data that closely mimics real data, while the discriminator works to distinguish between the generator's output and genuine data. This competition improves the generator's ability to produce high-quality images. In the domain of Generative Adversarial Networks (GANs), advancements include models like SRGAN [15] and ESRGAN [16], or SRGAN-DAM-SISR [17].

In addition to CNNs and GANs, Transformer-based models [18, 19, 20] have emerged as a powerful new approach in the field of super-resolution. Transformers, originally designed for natural language processing tasks, have been adapted to handle image data through the introduction of Vision Transformers [21]. These model's utilize self-attention mechanisms [22] that allow them to focus on relevant parts of an image, facilitating a more global understanding compared to the local receptive fields of CNNs. This characteristic is particularly advantageous for super-resolution tasks, where contextual information from distant image regions can be crucial for reconstructing high-resolution details.

Diffusion models [1] represent an innovative approach to super-resolution, offering new techniques for enhancing image quality. These models have shown substantial capabilities in producing high-quality images. Particularly in the context of super-resolution, diffusion models like SR3 [2] and SRDiff [3] have been employed with promising results.

1.2 Climate Downscaling

Global Climate Models (GCMs) are fundamental tools in climate science, used to simulate and predict the evolution of the climate system. These models numerically solve the physical equations governing the different components of the earth's climate system—the atmosphere, hydrosphere, cryosphere, and lithosphere—and their interactions. [23] By integrating data from experiments, observations, and theoretical physics, GCMs provide insights into the long-term patterns of weather variables such as temperature, precipitation, humidity, and wind across the globe. [24] Despite their comprehensive nature, GCMs have a significant limitation: their coarse spatial resolution. This limitation arises from the computational demands of running these complex models, which necessitate dividing the earth into a grid with cells that can be quite large, often spanning hundreds of kilometers on each side. This scale is adequate for global or continental analyses but fails to capture the finer details necessary for understanding regional and local climate. [25, 26]

Therefore, an alternative procedure called downscaling is employed to address the spatial resolution limitations of Global Climate Models. Downscaling is a procedure that enables generating forecasts on a smaller, local level using climate data originally available at a larger scale. [26] In climate science, climate fields are often represented using structures similar to images, with different dimensions. This analogy allows for conceptualizing grid points on a climate map as pixels in an image, suggesting that downscaling can be approached as a single image super-resolution task. Although this process is termed "downscaling," it involves what is essentially an image upscaling task, where the goal is to create finer-resolution maps from coarser ones by adding more grid points, resulting in maps with smaller horizontal resolutions. [27]

There are two primary methods for the downscaling of climate variables: dynamical and statistical downscaling. Dynamical downscaling utilizes Regional Climate Models (RCMs), which simulate local physical processes such as convective and vegetation schemes. This method involves a physics-based model that employs equations to represent various components of the climate system and their interactions. However, RCMs are computationally demanding and their applications are not easily transferable across different regions. [28, 29]

Statistical downscaling provides a more adaptable approach by estimating the relationship between coarse-scale and fine-scale variables. Among the leading-edge techniques in this field are deep learning models that excel in capturing complex nonlinear relationships inherent in climatic data. [28] Notable examples primarily include super-resolution (SR) models adapted for downscaling, such as those based on Convolutional Neural Networks (CNNs) [30, 31], Generative Adversarial Networks (GANs) [27, 32], and other models [33, 34]. Furthermore, diffusion models are emerging as a promising approach for generating high-resolution climatic predictions. Despite earlier assessments, such as those by [35] which reported suboptimal results using the SR3 [2] model, this work demonstrates the superior capabilities of diffusion models in this field.

Chapter 2
Diffusion Models

This chapter describes diffusion models and their advanced variants designed for the super-resolution of weather data. Section 2.1 investigates the mathematical foundations, optimization, and training of diffusion models. Sections 2.2, 2.3, and 2.4 discuss advanced variants such as SR3, SRDiff, and ResDiff. In Section 2.5, we introduce a novel variant of the ResDiff model that incorporates physics-based improvements. Given the focus on advanced deep learning architectures, it's assumed that the reader has a fundamental understanding of neural networks, including the principles of convolutional neural networks and training methods.

2.1 DDPM

Denoising Diffusion Probabilistic Models (DDPMs) [1], inspired by nonequilibrium thermodynamics [36] represent a significant breakthrough in the domain of generative machine learning [37]. Their application spans a wide array of fields, notably including image generation and super-resolution, where they are prized for their ability to generate high-quality samples. This capability has elevated DDPMs to the forefront of image synthesis architectures. In contrast to Generative Adversarial Networks (GANs) [14], DDPMs offer a simpler training process and are relatively straightforward to define. [38]

Unlike traditional generative models that directly learn the data distribution or map from a latent space to the data space, DDPMs adopt a novel approach by gradually transforming data into a known distribution, typically Gaussian noise through a process known as the forward diffusion process. This transformation is achieved by iteratively adding Gaussian noise to the data over a series of steps, effectively diffusing the data into noise. The beauty of this process lies in its reversibility. The model learns to reverse this noise addition, effectively denoising the data to reconstruct the original input from noise, in what is known as the backward or reverse diffusion process. [1]

2.1.1 Forward Diffusion Process

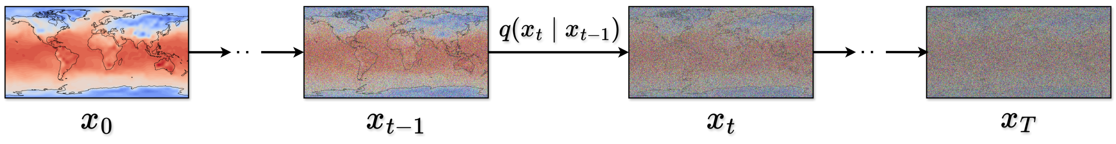

Figure 2.1 Illustration of the Forward Diffusion Process where noise is incrementally introduced to an image across T steps.

The forward diffusion process is essentially a Markov chain, starting with the data point $\mathbf{x}_0$ obtained from data distribution $q(\mathbf{x})$. It systematically introduces a small amount of Gaussian noise to $\mathbf{x}_0$ in each iteration, progressively increasing the noise level across the samples $\mathbf{x}_1, ..., \mathbf{x}_T$. It's important to note that as T increases, the data point $\mathbf{x}_T$ progressively loses more detail. Ultimately, as $T \to \infty$, $\mathbf{x}_T$ converges towards an isotropic Gaussian distribution. This iterative process over T steps can be represented as follows [38]:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \tag{2.1}$$

To incrementally introduce noise, we establish a variance schedule characterized by the sequence $(\beta_1, \beta_2, ..., \beta_t)$ where $\beta_t \in (0, 1)$. The goal is to initially add a substantial amount of noise to the image and then gradually reduce the noise intensity as we approach the final iteration T. A single step in this process can be defined as follows [39]:

$$\begin{aligned} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) &= \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \boldsymbol{I}\right) \\ &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathbf{e} \quad \text{for } \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \end{aligned} \tag{2.2}$$

A highly beneficial characteristic of this process is the capability to directly sample at an arbitrary timestep t, conditioned on $\mathbf{x}_0$, in closed form by employing the reparameterization trick. By introducing $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ with $\alpha_t = 1 - \beta_t$ and given that $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, ... \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ we can rewrite diffusion steps as follows [38, 39]:

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t \\ &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-1}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\overline{\boldsymbol{\epsilon}}_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\overline{\boldsymbol{\epsilon}}_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\overline{\boldsymbol{\epsilon}}_{t-2} \\ &\vdots \\ &= \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\overline{\boldsymbol{\epsilon}}_0 \\ q\left(x_t \mid x_0\right) &= \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, \left(1-\bar{\alpha}_t\right)I\right) \end{aligned} \tag{2.3}$$

It demonstrates that the entire forward process can be computed in a single step.

The original authors [1] implemented DDPM using a linear variance schedule ($10^{-4} \leq \beta_i \leq 0.02$ for $i = [0, T]$), achieving notable outcomes. However, Nichol and Dhariwal [40] proposed a cosine-beta schedule that is more effective, enhancing sample quality and training efficiency [38]:

$$\bar{\alpha}_t = \frac{1}{2}(\cos(t/T \cdot \pi) + 1) \tag{2.4}$$

By structuring the diffusion process in this way it provides a controlled way of transforming the original data into noise, setting the stage for the reverse process where the model learns to reconstruct the original data from the noise.

2.1.2 Reverse Diffusion Process

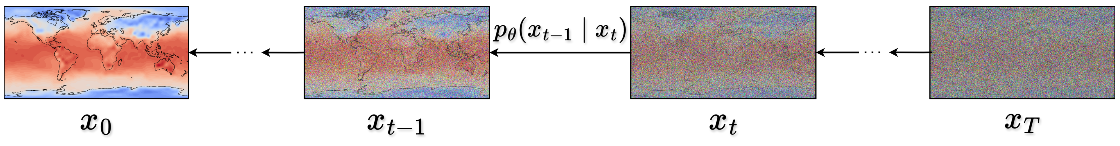

Figure 2.2 Illustration of the Backward Diffusion Process where an image is incrementally denoised using the model p_θ across T steps.

The reverse diffusion is a Markov process that is characterized by gradually refining a sample from a simple noise by applying learned reverse diffusion steps. The objective of reverse diffusion processes is to reconstruct an original data point x_0 from a noisy distribution, specifically Gaussian noise $\mathcal{N} \sim (\mathbf{0}, \boldsymbol{I})$. Directly reversing the diffusion sequence, characterized by $q(x_t|x_{t-1})$, to sample from $q(x_{t-1}|x_t)$ is not computationally feasible due to the complexity and requirements of utilizing the entire dataset for accurate estimation. [38, 1] As a solution, we leverage a learned model p_θ, parameterized by θ, which aims to approximate the conditional probabilities essential for reverse diffusion. For some fixed sequence of $\sigma_1 ... \sigma_T$, model p_θ is represented as follows [39]:

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right), \sigma_t^2 \boldsymbol{I}\right) \tag{2.5}$$

The whole reverse process can be represented as so:

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{0:T}\right) = p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) \tag{2.6}$$

It's important to note that reverse of $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$ is tractable when conditioning on x_0. This posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)$ can be derived using Bayes theorem with $\tilde{\beta}_t$ and $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right)$ which are represented as follows [39]:

$$\begin{aligned}
\tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \\
\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{e}_t\right) \\
q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) &= \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \boldsymbol{I}\right)
\end{aligned} \tag{2.7}$$

This posterior distribution plays a crucial role in parameterizing the reverse chain and in establishing a variational lower bound [41] for the log-likelihood of the reverse process. [1] The detailed derivations of these expressions can be found in [38].

2.1.3 Model Optimization

Initially, we start with the image x_0, to which we apply the forward and backward processes. The goal is to carry out these processes to maximize the probability of the result being our original image. This leads to the minimization of the negative log-likelihood.

$$-\mathbb{E}_{q(\mathbf{x}_0)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}_0\right)\right] \tag{2.8}$$

The probability of $p_{\boldsymbol{\theta}}\left(\mathbf{x}_0\right)$ cannot be directly computed in a straightforward manner, as it depends on the entire sequence of time steps $\{x_0, x_1, ..., x_T\}$. This would require tracking T random variables, which is impractical in real-world application. To address this issue, we can compute the variational lower bound (VLB) [41] for this objective, leading to a formula that is more computationally feasible [38, 42]:

$$\begin{aligned}
-\log p_\theta(x_0) &\leq -\log p_\theta(x_0) + D_{KL}\left(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)\right)\\
&= -\log p_\theta(x_0) + \mathbb{E}_{x_{1T}\sim q(x_{1:T}|x_0)}\left[\log \frac{(q(x_{1:T}|x_0))}{p_\theta(x_{0:T})/p_\theta(x_0)}\right]\\
&= -\log p_\theta(x_0) + \mathbb{E}_q\left[\log \frac{(q(x_{1:T}|x_0))}{p_\theta(x_{0:T})} + \log p_\theta(x_0)\right]\\
&= \mathbb{E}_q\left[\log \frac{(q(x_{1:T}|x_0))}{p_\theta(x_{0:T})}\right]\\
\text{Let } L_{VLB} &= \mathbb{E}_{q(x_{0:T})}\left[\log \frac{(q(x_{1:T}|x_0))}{p_\theta(x_{0:T})}\right] \geq -\mathbb{E}_{q(x_0)}\log p_\theta(x_0)
\end{aligned} \tag{2.9}$$

In order to make each term in this equation analytically computable, we further decompose the objective into a series of Kullback-Leibler (KL) divergence and entropy terms [1]:

$$\begin{aligned}
L_{VLB} &= \mathbb{E}_{q(x_{0:T})}\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]\\
&= \mathbb{E}_{q(x_{0:T})}\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\left(\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]\\
&= \mathbb{E}_{q(x_{0:T})}\left[-\log p_\theta(x_T) + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} + \log\frac{q(x_T|x_0)}{q(x_1|x_0)} + \log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right]\\
&= \mathbb{E}_{q(x_{0:T})}\left[\log\frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right]\\
&= \mathbb{E}_{q(x_{0:T})}\left[\underbrace{D_{KL}\left(q(x_T|x_0)\|p_\theta(x_T)\right)}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{KL}\left(q(x_{t-1}|x_t,x_0)\|p_\theta(x_{t-1}|x_t)\right)}_{L_t} - \underbrace{\log p_\theta(x_0|x_1)}_{L_0}\right]
\end{aligned}$$

The total loss consists of three parts: L_T, L_t and L_0. Since L_T represents a forward process that does not involve learning and is constant relative to θ, it can be disregarded. Term L_0 facilitates the generation of a discrete image representation from a continuous form. The authors [1] chose to eliminate this term, altering only the sampling method at time step $t = 1$ [42, 39].

Particularly significant is term L_t, representing the KL divergence between the inverse of the forward process and the model p_θ. This KL divergence is calculated between two Gaussian distributions $\mathcal{N}\left(x_{t-1}; \tilde{\mu}_t, (x_t, x_0), \tilde{\beta}_t I\right)$, $\mathcal{N}\left(x_{t-1}; \mu_\theta\left(x_t, t\right), \sigma_t^2 I\right)$ and can be integrated, allowing for explicit computation as follows [42, 38]:

$$L_t = \mathbb{E}\left[\frac{1}{2\left\|\sigma_t \mathbf{I}\right\|_2^2} + \left\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\right\|_2^2\right] \tag{2.10}$$

To enable our model to accurately approximate the conditional probability distributions inherent in the reverse diffusion process, we aim to train model, denoted as $\boldsymbol{\mu}_\theta\left(x_t, t\right)$, to predict the outcome of $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right)$. As previously demonstrated, reverse process can be represented by:

$$\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{e}_t\right) \tag{2.11}$$

This suggests that our model can be adapted to focus on predicting the Gaussian noise added to an image in its forward process, instead of reconstructing the denoised image. The model's formulation can be redefined as $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, illustrated by the following equation:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{2.12}$$

where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ represents the neural network designed for noise prediction [39]. The architecture of this network is described later in this work for each specific model.

Since the model only predicts noise added to the image, the loss can then be reparametrized as follows [39]:

$$\begin{aligned}
L_t &= \mathbb{E}\left[\frac{1}{2\left\|\sigma_t \mathbf{I}\right\|_2^2} + \left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{e}_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)\right\|_2^2\right] \\
&= \mathbb{E}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\left\|\sigma_t \mathbf{I}\right\|_2^2} + \left\|\mathbf{e}_t - \epsilon_\theta(\mathbf{x}_t, t)\right\|_2^2\right] \\
&= \mathbb{E}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\left\|\sigma_t \mathbf{I}\right\|_2^2} + \left\|\mathbf{e}_t - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{e}_t, t\right)\right\|_2^2\right].
\end{aligned} \tag{2.13}$$

Authors [1] empirically found that simplifying the loss function by removing the constant term $\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\sigma_t \mathbf{I}\|_2^2}$, leads to improved model performance. Therefore, the final loss function is expressed as [39]:

$$\mathbb{E}_{t \in \{1..T\}, \mathbf{x}_0, \mathbf{e}_t}\left[\left\|\mathbf{e}_t - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{e}_t, t\right)\right\|_2^2\right] \tag{2.14}$$

The objective of the final loss function is to minimize the error between the noise predicted by the model and the actual sampled noise.

2.1.4 Model Training and Sampling

When training the DDPM model, as depicted in 2, the focus is on teaching the model to accurately predict the noise that has been added to the original image. The process starts with sampling data from the distribution q. Next, a timestep T and Gaussian noise are sampled. A noised image is then generated, and the model is updated to minimize the squared error between the predicted noise and the actual noise. This procedure is iteratively repeated during the training phase.

The DDPM sampling process starts with an initial sample of Gaussian noise. For each timestep from T down to 1, the model predicts the noise that was incrementally added to the image, and then subtracts it from the current image state. Through this iterative process, the addition of noise is effectively reversed, ultimately resulting in the production of a denoised image. A notable drawback of this algorithm is its computational intensity, as the model must utilize the denoising network T times, one for each timestep, making sampling even a single image notably time-consuming.[1] Detailed descriptions of these algorithms are provided in figure 2.3.

Algorithm 1 DDPM Training

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

Algorithm 2 DDPM Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:     $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:     $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 2.3 DDPM Training and Inference Algorithms [1]

2.2 SR3

Building on the foundational principles of Denoising Diffusion Probabilistic Models (DDPMs) [1], the SR3 model, first proposed in [2], introduces several significant enhancements developed for the task of super-resolution. Unlike the basic DDPM framework, which focuses on generating images from noise without explicitly specifying the target image's appearance, SR3 adapts and extends this concept for super-resolution by utilizing a conditional generation strategy. This approach enables SR3 to convert low-resolution images into high-resolution counterparts through iterative detail refinement.

In the pursuit of enhancing image resolution, slight modifications are necessitated in the diffusion model. The model must be conditioned on a low-resolution image which can be reconstructed into the original high-resolution image. This process is initiated with a dataset comprising pairs of HR and LR images, denoted as $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i signifies the i-th low-resolution image within the dataset, and y_i indicates the i-th high-resolution counterpart. In this work, these pairs represent values of weather temperature measurements captured in low and high resolution. [2, 35]

The objective is to learn a parametric approximation of the conditional probability $p(y|x)$, mapping a source LR image x to a target HR image y, through a stochastic iterative refinement process. To accomplish this, the HR image undergoes a controlled degradation process by the sequential addition of Gaussian noise throughout the forward diffusion phase. A conditional DDPM is then trained to invert this process, aiming to reconstruct the original HR image from its LR counterpart. Given that we start with y_0. Thus, the forward process, which involves the addition of noise to the HR image, can be reformulated as follows [35, 2]:

$$q(y_{1:T} \mid y_0) = \prod_{t=1}^{T} q(y_t \mid y_{t-1}) \tag{2.15}$$

$$q(y_t \mid y_{t-1}) = \mathcal{N}(y_t; \sqrt{\alpha_t} y_{t-1}, (1-\alpha_t)\boldsymbol{I}) \tag{2.16}$$

$$q(y_t \mid y_0) = \mathcal{N}(y_t; \sqrt{\gamma_t} y_0, (1-\gamma_t)\boldsymbol{I}) \tag{2.17}$$

The above formulas closely follow the fundamental concepts of DDPM [1], utilizing an identical approach for derivation. A significant enhancement to the model includes the authors [2] deciding to further refine the forward process by adapting the noise schedule. They introduced a piecewise distribution [43, 44] for γ, denoted as $p(\gamma) = \sum_{t=1}^{T} \frac{1}{T} U(\gamma_{t-1}, \gamma_t)$, which has proven to enhance the generation of three-channel super-resolution images. However, in this work, an SR3 model with a cosine noise schedule 2.4 has been employed, as empirical validation has revealed that it is more effective with climate variable data.

In the backward process, the model is conditioned on both a low-resolution image and a high-resolution noised image, described as:

$$p_\theta(y_{t-1} \mid y_t, x) = \mathcal{N}\big(y_{t-1}; \mu_\theta(x, y_t, \gamma_t), \sigma_t^2 I\big) \tag{2.18}$$

The objective function aims to maximize the likelihood of accurately reconstructing the noise $\mathbf{e}_t$ added to the HR image, presented as:

$$\mathbb{E}_{t, y_0, x, \mathbf{e}_t} \left[\left\| \mathbf{e}_t - f_\theta \left(x, \sqrt{\gamma_t} y_0 + \sqrt{1-\gamma_t} \mathbf{e}_t, t \right) \right\|^2 \right] \tag{2.19}$$

where, f_θ represents the model's estimation of the noise $\mathbf{e}_t$ at step t, based on the LR image x and the noised HR image y_t. [2]

2.2.1 SR3 Inferencre and Training

The SR3 training process depicted in Algorithm 3 innovates on traditional DDPM by conditioning on variance γ rather than the timestep t, enabling a more adaptable approach to training. This conditioning allows for flexibility in the number of diffusion steps and the customization of the noise schedule during inference.

The SR3 inference process depicted in Algorithm 4, begins with y_T, which is approximately Gaussian noise. Throughout the T iterations, normally distributed noise Z is sampled, and at each step, the mean is computed by a denoising model. This mean is then adjusted by adding noise according to a predefined variance schedule, acting as noise perturbation [45]. This process progresses iteratively, with each step gradually reducing the noise in y_t and ultimately producing a reconstructed SR image. The iteration count, T, serves as a hyperparameter. The authors [2] define $T = 2000$ for the training phase and $T = 100$ for the inference phase. [2]

Algorithm 3 SR3 Training

```
1: repeat
2:     $(\mathbf{x}, \mathbf{y}_0) \sim p(\mathbf{x}, \mathbf{y})$
3:     $\gamma \sim p(\gamma)$
4:     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:     Take a gradient descent step on
           $\nabla_\theta \left\| f_\theta(\mathbf{x}, \sqrt{\gamma}\mathbf{y}_0 + \sqrt{1-\gamma}\boldsymbol{\epsilon}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p$
6: until converged
```

Algorithm 4 SR3 Sampling

```
1: $\mathbf{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:     $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:     $\mathbf{y}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta(\mathbf{x}, \mathbf{y}_t, \gamma_t) \right) \sqrt{1-\alpha_t}\mathbf{z}$
5: end for
6: return $\mathbf{y}_0$
```

Figure 2.4 SR3 Training and Inference Algorithms. [2]

2.2.2 SR3 Architecture

The architecture of a neural network designed for predicting noise in SR3, depicted in Figure 2.5, involves several key components, each serving a unique purpose to enhance the model's ability make accurate predictions. Here, We investigate the architecture's building blocks.

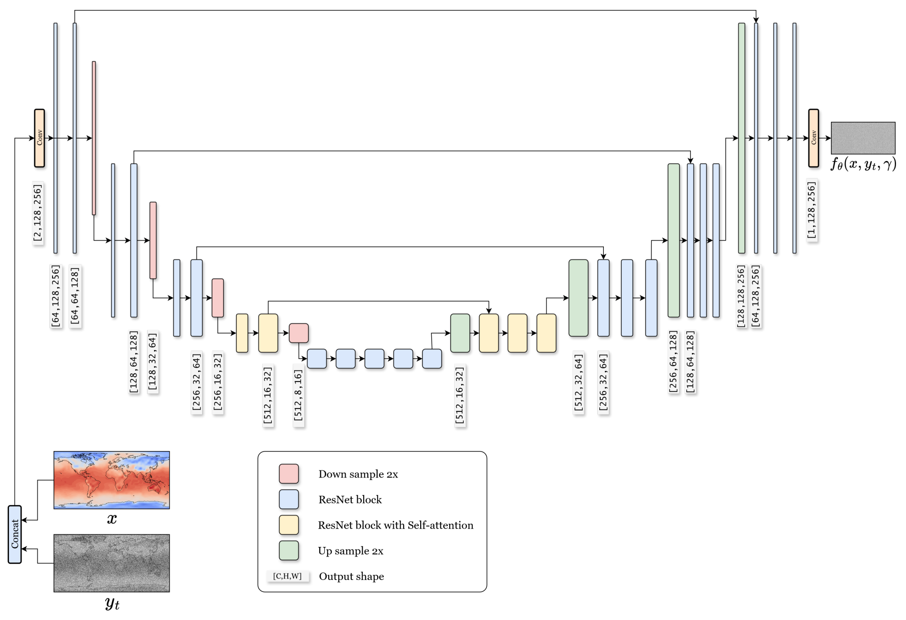

Figure 2.5 Architecture of SR3 model used for Climate Variable downscaling.

The input for the denoising neural network includes a noisy image y_t and a bilinearly [46] interpolated low-resolution image x. These images are concatenated and inputted into the network. [47]

U-Net Structure

The core of the SR3 neural network f_θ is based on a U-Net [48] architecture, which is characterized by its U-shaped structure. This structure includes an encoder that compresses the input image into a dense feature representation and a decoder that reconstructs the image details from this compressed form. The segment of the network serving as a transition point between the encoder and decoder is known as the bottleneck. The bottleneck handles the most compact version of the data, helping to move essential details from the encoder to the decoder for image reconstruction. [48, 49]

In SR3, the encoder employs convolutional layers [10] with stride [50] for downsampling of the image at each level by a factor of 2, as illustrated in Fig. 2.5. Conversely, the decoder utilizes nearest neighbor interpolation [51] for upsampling, aiming to expand the compressed feature representation back to the original image dimensions. [47] A distinctive feature of U-Net is its use of skip connections [48], which bridge the encoder and decoder sections. These connections forward feature maps from each level of the encoder directly to the corresponding level in the decoder, ensuring that detailed spatial information lost during compression is effectively reintroduced. Additionally, the use of skip connections contributes to stabilizing the training process and enhancing the convergence of the model. [49, 52]

ResNet Block

A ResNet block, also known as a residual block, is a foundational component of the ResNet architecture [53] designed specifically to combat the challenges associated with training deeper neural networks. As networks increase in depth, they often face the vanishing gradient problem [54], where gradients diminish as they are propagated back through the layers during training. This issue can worsen the training and overall network performance. The ResNet block architecture helps mitigate this issue. [55, 53]

A ResNet block is typically composed of a few stacked layers of convolutional neural networks, each followed by batch normalization [56] and an activation function. Unlike traditional neural networks where layers attempt to directly approximate the desired function $H(x)$ for an input x, ResNet blocks take a different approach by focusing on learning a residual function. Specifically, the layers in a ResNet block are designed to approximate the residual $F(x)$ defined as $F(x) := H(x) - x$. The output of the ResNet block is then obtained by adding the input x back to the output of the residual mapping, yielding $H(x) = F(x) + x$. [57, 53]

In a ResNet block, the input x not only passes through the stacked convolutional layers to compute $F(x)$, but it also travels via shortcut connections that bypass these layers, as shown in Figure 2.6. This configuration allows gradients to propagate directly through the network. [57, 53]

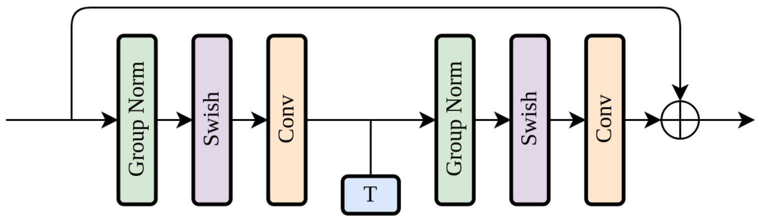

Figure 2.6 Architecture of ResNet block utilized in SR3 model.

In the SR3 ResNet block, depicted in Figure 2.6, there are two convolutional layers preceded by Group Normalization [58] and the Swish activation function [59]. Between these two convolution layers, the SR3 ResNet block is additionally conditioned on a specific diffusion timestep t [2, 39]. This conditioning is achieved by integrating the timestep into the network using sinusoidal position embeddings inspired by the Transformer [22] architecture, where position embeddings are used to provide sequence order information.

Self-attention

Self-attention is a mechanism that enhances neural network architectures, enabling them to dynamically focus on relevant parts of the input data. By computing attention scores, self-attention allows models to determine the importance of various input elements and adjust their influence accordingly, ensuring a context-aware understanding. This capability is particularly crucial in language processing tasks, as it enables the model to adapt the interpretation of a word depending on its contextual placement within a sentence or a larger document. [60]

Self-attention can be described through a sequence of mathematical steps involving three primary vectors for each element in the input: queries q, keys k, and values v. These vectors are projections from the original input vectors, created by multiplying the input with three separate weight matrices W^Q, W^K, and W^V, which are learned during training. [61, 22]

The attention mechanism computes the attention scores by taking the dot product of the query vector with all key vectors. The attention scores are then scaled by the dimensionality of the keys to help stabilize the gradients during training. Mathematically, the score between a query q and a key k is calculated as $\frac{q \cdot k}{\sqrt{d_k}}$, where d_k is the dimensionality of the key vectors. [22]

After computing these scores, a softmax function is applied to convert them into a probability distribution, ensuring that they are positive and sum to one. This softmax operation is performed for each query against all keys and is defined as $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$, where softmax is applied column-wise across the resulting matrix. Here, Q and K are matrices containing all query and key vectors, allowing for efficient simultaneous computation. [61, 22]

The output is then computed as a weighted sum of the value vectors, with weights given by the softmax output. For each query, the final output vector is the sum of all values V, weighted by the attention scores, effectively allowing each output element to consider information from the entire input sequence described as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

This sequence of operations allows the self-attention mechanism to dynamically focus on different parts of the input data. [61]

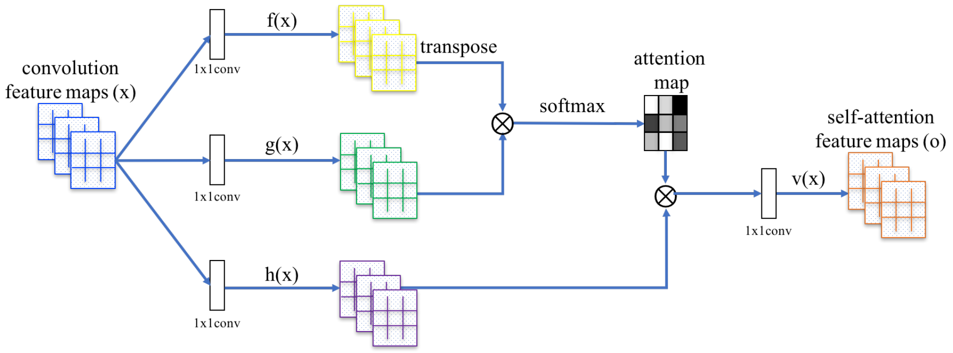

Figure 2.7 Architecture of image self-attention module first used in SAGAN [62]

In the SR3 architecture, specific ResNet blocks incorporate self-attention mechanisms to address global dependencies in addition to local ones captured by convolutional layers. The inputs to this self-attention are the 2D convolutional feature maps from the output of the ResNet block. These feature maps are treated as sequences of feature vectors, which are processed using 1x1 convolutions, as illustrated in Figure 2.7. Subsequently, these vectors are utilized to generate the Q, K, V matrices essential for the self-attention process. [39, 47]

As depicted in Figure 2.5, ResNet blocks with self-attention are employed only at lower resolution levels in the U-net architecture due to the high computational cost associated with processing large images.

2.3 SRDiff

SRDiff [3] is a conditional diffusion model for super-resolution that shares a similar architecture with SR3 but introduces modifications in its conditioning approach by using a pre-trained LR encoder and incorporating residual prediction.

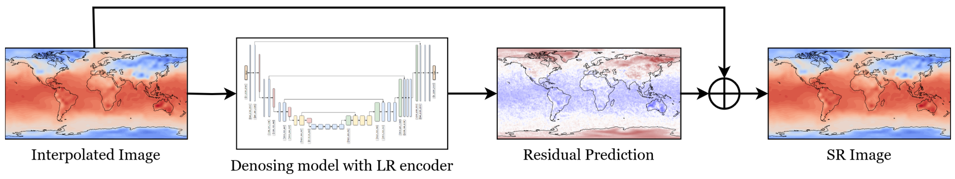

Figure 2.8 Illustration of residual-based prediction.

Unlike SR3 method which aims to predict a high-resolution image directly, SRDiff adopts a residual prediction strategy. This approach involves estimating the difference between the HR image and an upsampled low-resolution image, referred to as the input residual image. The model transforms the residual image into a latent variable with a Gaussian distribution using a diffusion process. Subsequently, it employs a denoising network conditioned on the encoded low-resolution image using an RRDB-based [63] LR encoder, refining the latent variable into a detailed residual image. This residual image is then combined with the upsampled LR image to reconstruct the final SR image. [3]

2.3.1 SRDiff LR Encoder

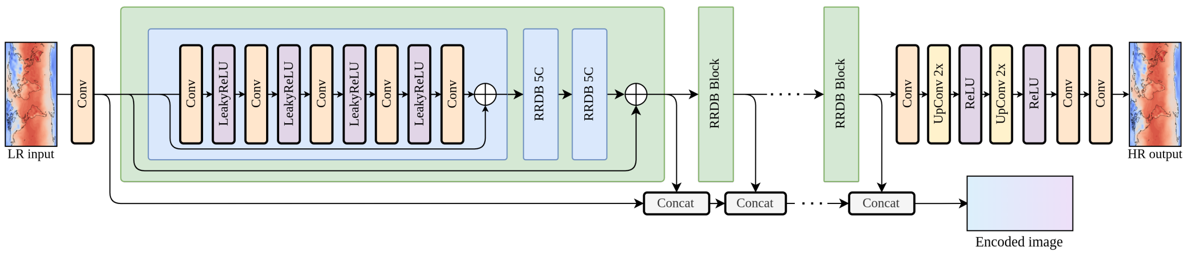

Figure 2.9 Architecture of RRDB encoder utilized in SRDiff model.

The SRDiff LR Encoder is a neural network designed to encode information from low-resolution images, which is essential for conditioning of denoising model. The authors adopted an encoder architecture from [63], which incorporates Residual in Residual Dense Blocks (RRDB) originally used in ESRGAN [16]. As shown in Fig. 2.9, these blocks feature densely connected convolutional layers with Leaky ReLU [64] activations and multiple dense skip connections, deliberately omitting batch normalization [56] layers.

The use of this neural network for image encoding is not standard. The network is trained to reconstruct high-resolution images from LR ones using L1 loss. However, to obtain the encoded image, instead of using the network's output, which is the HR image reconstruction, the concatenated outputs from all RRDB blocks are used. This concatenation results in a multi-level encoding of the LR image, which is ideal for conditioning. Additionally, the number of RRDB blocks within the encoder is a hyperparameter that offers flexibility to adjust the complexity of the encoded image and optimize the encoder's performance. [3, 63]

2.3.2 SRDiff Inferencre and Training

As illustrated in Algorithm 5, the training process begins with sampling low-resolution image x and high-resolution image y. The residual image is then created by subtracting the bicubic-interpolated LR image x from the HR image y. This LR image is encoded using a pre-trained RRDB encoder. Subsequently, the encoded image x_e and the residual image y_r are used to compute the gradients, following the methodology of the basic DDPM training Algorithm 1.

The inference phase, illustrated in Algorithm 6, starts with sampling y_t from the standard Gaussian distribution and encoding the LR image x using the RRDB encoder to produce x_e. Over the course of T iterations, the residual image undergoes denoising through a network that is conditioned on the encoded image, similar to the methodology used in the SR3 model. After completing T iterations, the denoised residual image is combined with an interpolated image to produce the final SR image. [3]

Algorithm 5 SRDiff Training

```
1: **Input**: LR image and HR image pairs $P = \{(x^k,\ y^k)\}_{k=1}^{K}$, total diffusion step $T$
2: **Initialize**: randomly initialized conditional noise predictor $\epsilon_\theta$ and pretrained LR encoder $\mathcal{D}$
3: **repeat**
4:     Sample $(x,\ y) \sim P$
5:     Upsample $up(x)$, compute $y_r = y - up(x)$
6:     Encode LR image $x$ as $x_e = \mathcal{D}(x)$
7:     Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
8:     Sample $t \sim \text{Uniform}(\{1, \cdots, T\})$
9:     Take gradient step on:
       $\nabla_\theta \|\epsilon - f_\theta(y_t, x_e, t)\|,\ y_t = \sqrt{\bar{\alpha}_t} y_r + \sqrt{1 - \bar{\alpha}_t}\epsilon$
10: **until** converged
```

Algorithm 6 SRDiff Sampling

```
1: **Input**: Low-resolution image $x$, total diffusion step $T$
2: **Load**: Conditional noise predictor $\epsilon_\theta$ and low-resolution encoder $\mathcal{D}$
3: Sample $y_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: Upsample $x$ to $up(x)$
5: Encode $x$ as $x_e = \mathcal{D}(x)$
6: **for** $t = T$ **to** 1 **do**
7:     Sample $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $z = \mathbf{0}$
8:     Compute $y_{t-1}$:
       $y_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(y_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(y_t, x_e, t)\right) + \sigma_t z$
9: **end for**
10: **return** $y_0 + up(x)$ as the super-resolved (SR) prediction
```

Figure 2.10 SRDiff Training and Inference Algorithms. [3]

2.4 ResDiff

The Residual-structure-based diffusion model [4] is an architecture based on SR3 [2] and SRDiff [3] that combines Convolutional Neural Networks (CNNs) [10] and conditional diffusion models. ResDiff adapts the residual prediction strategy from SRDiff and also replaces the initial bicubic interpolation [46] used for conditioning the SR3 denoising network with a pre-trained CNN, specialized in capturing major low-frequency components. Furthermore, ResDiff introduces High-Frequency Guided Diffusion that modifies the denoising network to focus on enhancing high-frequency image components. This method ensures that the details lost in the SR3 diffusion process, are meticulously reconstructed.

The ResDiff training process is practically the same as in the SRDiff training algorithm 5 with a modification that instead of using a pre-trained LR encoder for model conditioning, a pre-trained CNN is utilized. The CNN generates a super-resolution version of the low-resolution input image, which is then concatenated with the noised residual image. This approach resembles the process in SR3, where the interpolated image is combined with the noised input to condition the denoising network. [4]

The inference process is also similar to the SRDiff sampling algorithm 6, with the only difference being that the model is initially conditioned on the SR output from the pre-trained CNN, instead of being conditioned on the interpolated image as in SR3.

2.4.1 Pre-trained CNN

The primary function of the CNN in the ResDiff model is to generate an initial HR prediction that effectively recovers the major low-frequency components and partial high-frequency components of a low-resolution image better than bicubic interpolation. It's designed to be lightweight, containing fewer parameters to reduce computational demands while still effectively capturing essential image features. [4]

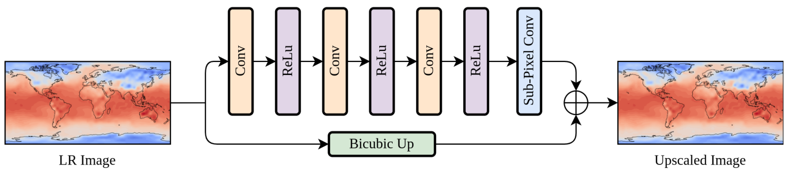

Figure 2.11 Architecture of pre-trained CNN utilized in ResDiff.

The architecture of the pre-trained CNN, depicted in Figure 2.11, consists of three convolutional layers, each followed by a ReLU activation function. The output from the final ReLU activation serves as the input for the Pixel Shuffle operation, also known as sub-pixel convolution. This technique upscales images from a low to high spatial resolution by rearranging elements from multiple feature map channels into a larger spatial grid. The process increases the number of channels in the feature maps by a factor of r^2 (where r is the upscaling factor) through a convolutional layer. This is followed by reorganizing these channels back into the spatial dimensions to create the upscaled image [65]. The output from the Pixel Shuffle is then merged through a residual connection with bicubic upsampled low-resolution input, forming the final initial prediction. [4]

The most important component of the simple CNN is its loss function, which comprises three distinct losses: FFT (Fast Fourier Transform) Loss, DWT (Discrete Wavelet Transform) Loss, and GT (spatial domain) Loss.

Spatial Domain Loss is a component of the loss function, calculated as the mean square error between the predicted and ground-truth high-resolution images. The primary purpose of the

L_{GT} is to ensure that the network accurately captures the overall pixel intensity and placement. The L_{GT} is mathematically represented as:

$$L_{GT} = \mathbb{E}\left[\|Y - \hat{Y}\|^2\right] \tag{2.20}$$

where Y is the ground-truth image and $\hat{Y}$ is the image predicted by the CNN. [4]

Frequency Domain Loss is complement the spatial domain loss, it handles the frequency aspects of the image reconstruction. It involves applying the Fast Fourier Transform (FFT) [66] to both the predicted and ground-truth images, converting them from the spatial to the frequency domain. The loss is then calculated as the MSE between the magnitudes of these FFT coefficients. This approach focuses the network's learning on preserving the structural integrity of the image in the frequency domain. The L_{FFT} is defined as:

$$L_{FFT} = \mathbb{E}\left[\|M - \hat{M}\|^2\right] \tag{2.21}$$

where M and $\hat{M}$ are the magnitudes of the FFT coefficients of the ground-truth and predicted images, respectively. [4]

Discrete Wavelet Transform Loss enhances the models focus on specific high-frequency components. It works by decomposing both the predicted and actual images into four sub-bands using the Discrete Wavelet Transform (DWT) [67]. These sub-bands include low-low (LL), low-high (LH), high-low (HL), and high-high (HH) components of the image. From high frequency components HL, LH and HH are then extracted wavelet coefficients H, V, and D which represent horizontal, vertical, and diagonal directions in the image. The low-frequency component, LL, which contains the structural information of the image, is further decomposed into another set of four sub-bands: LL, LH, HL, and HH. These sub-bands are processed in the same manner as the initial set. This decomposition process is recursively repeated four times, with each application of the DWT reducing the size of the LL component by half. Consequently, the LL component at the lowest level is reduced to 1/16th of the original image size in both dimensions. The H, V, and D coefficients extracted at multiple levels are then used to calculate the loss, which is computed as the sum of the mean squared errors across these coefficients:

$$L_{DWT} = \sum_{i=1}^{L} \mathbb{E}\left[\|H_i - \hat{H}_i\|^2 + \|V_i - \hat{V}_i\|^2 + \|D_i - \hat{D}_i\|^2\right] \tag{2.22}$$

where H_i, V_i, D_i are the high-frequency sub-bands of the ground-truth image and $\hat{H}_i, \hat{V}_i, \hat{D}_i$ are high-frequency sub-band of the predicted image, across different levels i of the wavelet decomposition. [4, 68]

Finall CNN loss is then computed as sum of there three losses:

$$L_{SimpleCNN} = \alpha \cdot L_{GT} + \beta \cdot L_{FFT} + L_{DWT} \tag{2.23}$$

where α, β are parameters that determine the relative importance of the spatial and frequency domain losses. [4]

2.4.2 ResDiff Architecture

As shown in Figure 2.12, ResDiff adds the FD Info Splitter and HF-guided Cross-Attention (CA) into denoising network architecture.

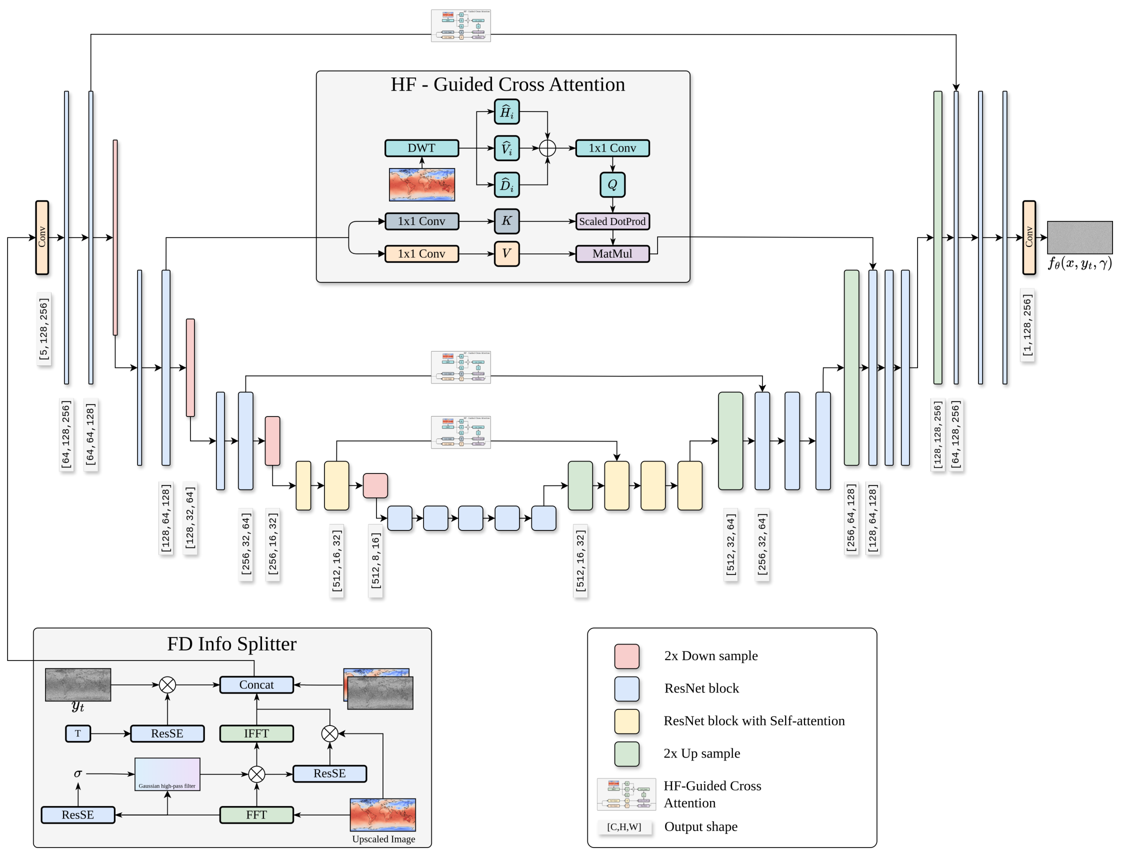

Figure 2.12 Architecture of ResDiff model used for Climate Variable downscaling.

FD Info Splitter

The Frequency-Domain Information Splitter (FD Info Splitter) is a component of ResDiff that segregates image data into high and low-frequency bands. This segregation is fundamental for focusing the model's attention on preserving or enhancing the details critical for high-quality image super-resolution. The process begins with the application of 2D FFT to both the interpolated and the noised residual images. This transformation allows the model to analyze and manipulate the frequency components of the image data directly.

Innovation within the FD Info Splitter is the use of a Residual Squeeze-and-Excitation (ResSE) block, proposed and described in [69]. The ResSE block processes the FFT-transformed feature maps, dynamically adjusting the importance of different channels in the output feature map. Following this, the computation of the standard deviation, σ, is carried out with formula:

$$\sigma = \min\left(|\,\mathrm{ResSE}(M)| + \frac{l}{2}, l\right) \tag{2.24}$$

The high-pass filtering, represented by $H(u, v)$, is then applied to isolate and enhance these high-frequency components:

$$H(u, v) = 1 - e^{-\frac{D^2(u,v)}{2\sigma^2}} \tag{2.25}$$

where $D(u, v)$ measures the distance of a frequency component from the origin in the frequency domain. This filtering process results in a feature map L, enriched with high-frequency details.

Following the high-pass filtering, an inverse FFT is applied to L, facilitating the generation of a low-frequency focused image representation, x_{LF}. Concurrently, the ResSE block refines L

to extract attention weights that are specific to the high-frequency domain. These weights are then applied to the upsampled image x to isolate its high-frequency components, resulting in x_{HF}.

The outcome of this process is a set of five distinct feature maps: $[x, y_t, x_{HF}, x_{LF}, y'_t]$, where x and y_t represent the source and the noise-added residual images, respectively, x_{HF} and x_{LF} denote the high and low-frequency components extracted, and y'_t is the target high-resolution output at iteration t. These feature maps are utilized to condition the model, enhancing its ability to reconstruct images, particularly in the high-frequency bands. [4]

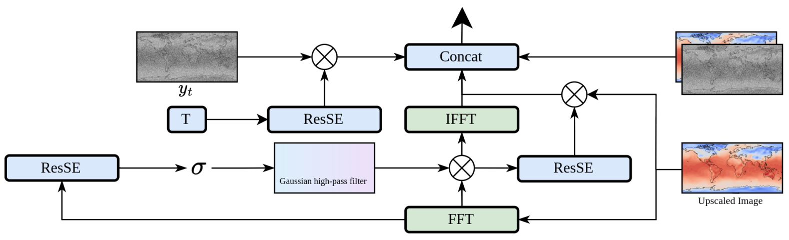

Figure 2.13 Architecture of FD Info Splitter.

HF-guided Cross-Attention (CA)

The HF-guided Cross-Attention mechanism enhances the model's capability to focus on and refine high-frequency details. Unlike typical self-attention, where all elements—queries, keys, and values—are derived from the same input to correlate different parts of that input, cross-attention employs queries from one input and matches them with keys and values from another. Within the ResDiff architecture, the cross-attention mechanism utilizes outputs from the Discrete Wavelet Transform (DWT) and feature maps sourced from the denoising network's skip connections. As depicted in Figure 2.14, linear projections of these feature maps serve as keys and values in the attention mechanism, while the combination of the DWT components H, V, and D forms the query. The result of this cross-attention mechanism is a feature map that is dynamically adjusted to emphasize high-frequency details, which are critical for reconstructing the image's finer structures and textures. [4]

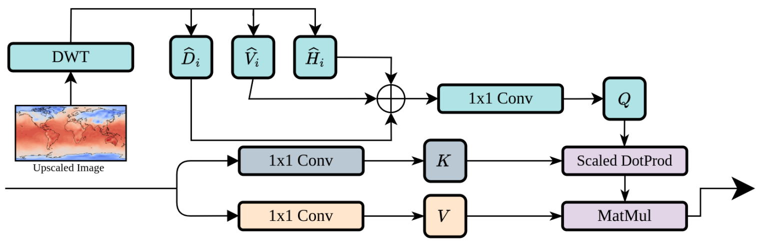

Figure 2.14 Architecture of HF-guided Cross-Attention.

2.5 ResDiff Enhanced with Physics-Inspired Convolutional Filters

In an effort to further refine the ResDiff architecture for applications requiring a nuanced understanding of dynamic systems, such as weather prediction, we introduce a novel modification termed "ResDiff + Physics". This adaptation integrates physics-inspired convolutional filters that mimic finite difference schemes used for computing derivatives. This approach allows the model to explicitly focus on derivative features that are fundamental to the Navier-Stokes equations [70], which govern fluid dynamics and are critical in meteorological modeling.

Motivation

The integration of physics-based principles into deep learning models offers a promising direction for improving the accuracy of predictions in fields heavily reliant on physical laws. By embedding convolutional filters that approximate spatial derivatives, "ResDiff + Physics" aims to capture the underlying physical processes that drive weather patterns.

Convolutional Derivative Filters

To achieve this, the Frequency-Domain Information Splitter was replaced with a set of three specialized convolutional filters designed to approximate first and second-order spatial derivatives, key components in differential equations like the Navier-Stokes. These filters are applied to the interpolated image, capturing the gradient and curvature information relevant to fluid motion and atmospheric dynamics. The filters are defined as follows:

$$\partial_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \partial_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \nabla^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Each filter is applied to the interpolated image using a reflect padding of 1 to ensure boundary consistency. The resulting derivative feature maps are then concatenated, forming a comprehensive tensor of shape [B [1] , 3 , H [2] , W [3]] that encapsulates spatial variation information.

Integration into ResDiff

Following the convolutional operation, the derivative feature maps are concatenated with the noised residual image and the original upsampled image. This enriched tensor serves as the input to the ResDiff U-net architecture, conditioning the model on physically meaningful derivatives rather than solely on frequency-domain information. This modification aims to ground the model's learning process in the physical reality of atmospheric dynamics, providing a more informed basis for generating high-resolution weather predictions.

Furthermore, We refined the HF-guided cross-attention 2.14 mechanism by incorporating a 1x1 convolution directly applied to the high-frequency components derived from the Discrete Wavelet Transform. This adjustment further enhances the model's ability to focus on and reconstruct high-frequency details, now informed by derivative-based features that reflect underlying physical processes. Additionally, we replaced the pretrained CNN with bicubic interpolation to further enhance the model's overall performance on weather data.

[1] Size of mini Batch
[2] Image Height
[3] Image Width

Chapter 3

Dataset and implementation

This chapter describes the dataset used for the experiments, along with the technologies employed and implementation details.

3.1 WeatherBench dataset

For model training and evaluation, we utilized the WeatherBench dataset [5], a benchmark dataset designed for assessing and comparing machine learning models in weather forecasting tasks. This dataset encompasses a variety of meteorological variables in different resolutions derived from the ERA5 [71] reanalysis dataset. ERA5 integrates modeled data with global observations to provide a consistent, gridded representation of historical weather conditions. In this work, we focused on the downscaling of the T2M (two-meter temperature) variable, which represents temperature readings in Kelvin on a latitude-longitude grid, situated two meters above the ground level. Data are recorded hourly, resulting in 24 distinct temperature measurements per day, covering the entire surface of the Earth and forming a rectangular image. In the context of our super-resolution task, we utilized data pairs with grid spacings of 5.625° and 1.40525°. Consequently, the low-resolution images consist of 32×64 pixels, and the high-resolution images consist of 128×256 pixels, effectively enhancing the resolution by four times in each dimension. These images are single-channel, as only one climate variable is used.

The dataset has a total volume of 45 GiB and includes 341,880 climate image pairs, collected between January 1, 1979, and December 31, 2018. For the training of the models, we utilized data pairs from January 1, 1979, to February 1, 2015. The validation set covers the period from January 1, 2016, to December 31, 2016.

Due to limited computational resources and the slow validation of models when utilizing 1000 timesteps for sampling, we downsized the dataset by using only the values measured in January for both the validation and training sets. These data are then standardized separately for each resolution. In some experiments, models were trained using data from other months, but always in such a way that no more than one distinct month was used at a time for training any single model.

The acquired data are stored in the NETCDF format, a standard for multi-dimensional scientific data such as atmospheric measurements, oceanographic metrics, and other environmental variables. For easier manipulation and enhanced training efficiency, these data are then converted into the standard NumPy binary format.

3.2 Implementation and tools

All experiments were conducted on a single NVIDIA A100 graphics card with 40 GB of memory and AMD Epyc 7742, 64 cores 3.6GHz processor. The whole project is implemented in Python 3.11, with the machine learning library PyTorch 2.0. For experiment tracking and logging was used Weights and Biases. The implementation of this work is based on the weather SR3 implementation [72] and the original ResDiff implementation [73]. The SR3 implementation includes code for handling weather data, which we utilized in our work. However, we found parts of this code to be extremely slow and inefficient, requiring us to rewrite them completely. We revised and optimized the code, greatly improving its speed and efficiency for handling large datasets. Additionally, we integrated new features related to regularization, handling of monthly data, training, and visualization. We sourced the ResDiff and SR3 models from the original ResDiff implementation and further refined them for weather data, as the original implementations were designed for working with fixed-size RGB images. The architecture of the RRDB encoder was obtained from [74], while the rest of the code, including the mentioned modifications, is our own.

Chapter 4
Experiments

4.0.1 Training details

This section describes the common hyperparameters and training details for the individual models. All changes to these hyperparameters and training procedures are always specified in the experiments.

Pretrained RRDB encoder and CNN

The RRDB encoder and the Simple CNN were both trained for up to 200 epochs, with early stopping mechanisms implemented to prevent overfitting. After each epoch, validation, and model checkpoints were conducted. Both models were trained on the same dataset as the other diffusion models utilized in the experiments. The Simple CNN employed an Adam optimizer with a learning rate of 1e-4 and due to its simplicity, a batch size of 128 was used. In CNN loss function we set parameters to $\alpha = 0.2$ and $\beta = 0.1$. Conversely, the RRDB encoder utilized an Adam optimizer with a slightly higher learning rate of 2e-4 and a smaller batch size of 32. The number of input channels for the RRDB blocks was set at 64.

Diffusion models

We trained models for 200,000 iterations, using various batch sizes. Specifically, SR3 and SRDiff models utilized a batch size of 16, while ResDiff-based models required a batch size of 4 due to their higher memory demands during experimentation. Validation was performed every 10,000 iterations across the entire validation set. Despite the computational intensity of diffusion models during inference, as they must utilize the entire denoising network for each timestep T, we kept $T = 1000$ for both training and validation to have accurate validation error estimates. As a result, the total training time for a single run was approximately 50 hours. For both training and validation, we utilized a linear noise schedule ranging from 1e-6 to 1e-2 and applied a dropout rate of 0.2 within each ResNet block. The dropout layer was placed after each Swish activation function, meaning each block contained two dropout layers. Additionally, similar to the authors in the original paper [1], we applied the Exponential Moving Average to model parameters with a decay rate of 0.9999. For optimization, we utilized Adam with a learning rate of 1e-4.

Validation metrics

As validation metrics, we employed basic measures such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), along with more sophisticated metrics designed for super-resolution evaluation. One such metric is the Structural Similarity Index Measure (SSIM) [75]. SSIM is a method for measuring perceived image quality, which surpasses traditional metrics like MSE by considering image structure that aligns more closely with human visual perception. SSIM is based on the computation of three key comparison measurements Luminance, Contrast, and Structure. These measurements are being computed between test image window x and reference image window y and can be defined as:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.1}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.2}$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.3}$$

The luminance comparison function $l(x, y)$ evaluates the average brightness of the pixels in the images, where μ_x and μ_y represent the mean luminance values of image windows x and y, respectively. The contrast comparison function $c(x, y)$ assesses the similarity in contrast between images, which is determined by the standard deviations of x and y, denoted as σ_x and σ_y. The structure comparison function $s(x, y)$ measures the correlation coefficient, where σ_{xy} represents the covariance between x and y. The terms C_1, C_2, C_3 are small constants added to avoid division by zero when the denominators are very small. The overall SSIM index is then a weighted combination of these three components:

$$\text{SSIM}(x, y) = [l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma] \tag{4.4}$$

To compute the final SSIM index between two images, it is calculated using a sliding window of size $n \times n$ that moves pixel-by-pixel across the entire image. At each position, the SSIM index is calculated, resulting in a matrix of SSIM values. The mean of all these values is the final result. The result is a value between -1 and 1, where 1 indicates perfect similarity (images are identical) and -1 indicates maximal dissimilarity. In this work, we utilized a sliding window of size 11×11 for the computation of all SSIM scores.[76, 75]

Along with SSIM we also utilized Peak signal noise ration (PSNR) metric. PSNR measures the quality of a reconstructed image compared to the original image. It is expressed in decibels (dB) and quantifies the ratio between the maximum possible signal power and the power of corrupting noise and is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right) \tag{4.5}$$

Where MAX represents the maximum possible pixel value of the image and MSE represents the mean squared error between the original and compressed image. A higher PSNR value typically indicates better image quality, as it suggests a lower level of error or noise. [77]

4.0.2 RRDB encoder

In this experiment, we explored various configurations of the RRDB encoder by altering the number of RRDB blocks within the network. Motivated by the fact that each block contains 15 convolutional layers, which are both memory and computationally intensive, our goal was to identify the setup that delivers optimal performance in terms of the SSIM while minimizing the number of blocks used. It's important to note that a higher SSIM score does not necessarily reflect better performance of the encoder within the SRDiff model. This is because the SRDiff model is not conditioned on the output of the RRDB network, but on the internal representations of the image derived from all activations within each RRDB block. Thus, the output of the encoder depends on the number of RRDB blocks. Having too many blocks in the encoder would also require compressing the encoder output into a smaller dimension to make it usable for conditioning the SRDiff model.

Table 4.1 Validation results of RRDB Encoder with different numbers of RRDB blocks.

RRDB Blocks	MSE ↓	SSIM ↑	PSNR ↑	MAE ↓
17 Blocks	29.05	0.767	25.66	2.987
23 Blocks	29.05	0.769	25.66	2.988
27 Blocks	29.04	0.770	25.66	2.988
30 Blocks	29.03	0.770	25.66	2.981
32 Blocks	29.04	0.769	25.66	2.986
Bicubic	6.866	0.854	31.52	1.542
Ground Truth	0	1	∞	0

The data presented in Table 4.1 indicate that the number of RRDB blocks in the encoder has minimal impact on the overall validation scores. Among the configurations tested, the architecture with 23 RRDB blocks emerges as the optimal choice, balancing efficiency and performance with only a marginal 0.001 difference in SSIM compared to the versions with more blocks. As the number of blocks increases, the models do not show improved performance; therefore, it does not make sense to use more than 27 blocks in the encoder.

4.0.3 SRDiff

We experimented with two methods of training the SRDiff model. In the first approach, we locked the weights of the pre-trained RRDB neural network, preventing the encoder from being trained alongside the SRDiff. In the second approach, we unlocked these weights, allowing the RRDB encoder to undergo further training during the SRDiff training process. To enable this, both the L1 loss of the RRDB encoder and the ResDiff L1 loss are summed together before gradient computation.

Table 4.2 Validation results of SRDiff with locked and unlocked RRDB encoder.

Model	MSE ↓	SSIM ↑	PSNR ↑	MAE ↓
SRDiff + (locked RRDB)	**1.748**	**0.951**	37.81	0.820
SRDiff + (unlocked RRDB)	1.761	0.950	**37.84**	**0.819**
Bicubic	6.866	0.854	31.52	1.542
Ground Truth	0	1	∞	0

One might assume that unlocking the encoder's weights and allowing further training would enable the SRDiff model to utilize the encoder even more effectively. Table 4.2 reveals that although the locked version is more easier to train, it achieves slightly better SSIM and MSE scores. On the other hand, the unlocked RRDB encoder performs better in terms of PSNR and MAE scores. This indicates that overall, both models perform very similarly. As depicted in Figure 4.1, both versions of SRDiff achieve exceptional results in weather super-resolution. Compared to bicubic interpolation over oceans, both achieve similar outcomes; however, on land, primarily in the Southern Hemisphere, there is a very notable improvement in favor of SRDiff. Comparing the locked and unlocked versions of SRDiff, they predict very similarly, but in regions such as Africa and Australia, the SRDiff with the locked RRDB is slightly more accurate from a visual perspective.

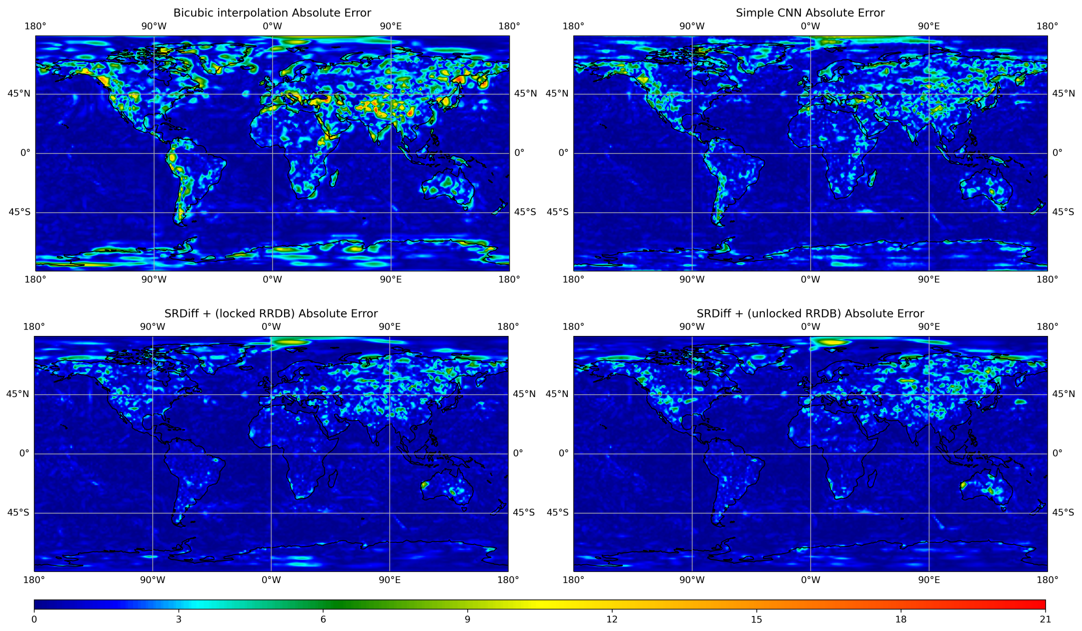
Bicubic interpolation Absolute Error
Simple CNN Absolute Error
SRDiff + (locked RRDB) Absolute Error
SRDiff + (unlocked RRDB) Absolute Error
180°
90°W
0°W
90°E
45°N
0°
45°S
0
3
6
9
12
15
18
21

Figure 4.1 Comparative visualization of absolute error in Kelvins across models: Bicubic interpolation, Simple CNN, SRDiff with locked RRDB encoder, SRDiff with unlocked RRDB encoder. The color scale at the bottom indicates the magnitude of errors, with areas of large errors highlighted in red and areas of minimal errors in blue. This visual representation helps in evaluating model accuracy over different global regions, demonstrating ; however each model performs.

4.0.4 ResDiff

As shown in Table 4.3, the Simple CNN outperforms bicubic interpolation in all validation metrics. This comparison is significant because, unlike the RRDB encoder, the Simple CNN serves as a direct replacement for interpolation within ResDiff. These results imply that because the Simple CNN surpasses bicubic interpolation, it should also enhance ResDiff performance. However, despite these favorable results achieved by the Simple CNN, we experimented with removing it from the ResDiff architecture and replacing it with bicubic interpolation, conditioning the model in the same manner as in the SR3 base model.

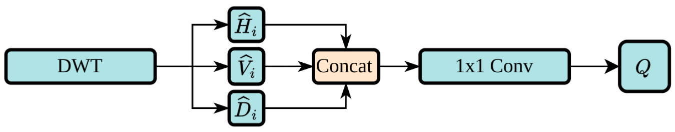

Figure 4.2 Part of the HF-guided Cross Attention module, illustrating the modification where the summation is replaced by concatenation.

Furthermore, as depicted in Figure 4.2, we modified the HF-guided Cross Attention in ResDiff. We changed the processing of the wavelet coefficients for the high-frequency bands H, V, and D—corresponding to the horizontal, vertical, and diagonal directions, respectively in the image—from summing to concatenating these outputs. These concatenated outputs are then processed by a 1x1 convolution. Our motivation for this adjustment was to preserve more crucial high-frequency information in the image for cross-attention, which might otherwise be lost through summation.

Table 4.3 Validation results of various ResDiff versions.

Model	MSE ↓	SSIM ↑	PSNR ↑	MAE ↓
Simple CNN	3.072	0.907	35.01	1.096
ResDiff + Simple CNN	1.801	0.952	37.59	0.824
ResDiff	**1.768**	**0.953**	**37.66**	**0.813**
ResDiff with concat	1.842	0.951	37.52	0.828
Bicubic	6.866	0.854	31.52	1.542
Ground Truth	0	1	∞	0

Table 4.2 demonstrates that incorporating high-frequency components through concatenation in cross-attention did not enhance model performance. All validation metrics were worse than those of the unmodified ResDiff. Figure 4.3 indeed shows that although ResDiff with concatenation committed fewer errors in regions like the South Pole and South America, its overall performance still lags behind other versions of ResDiff as indicated by the validation metrics.

On the other hand, Table 4.2 also shows that eliminating the pre-trained CNN from the model led to unexpectedly positive outcomes. The ResDiff variant without the CNN surpassed the version with the SimpleCNN across all key metrics MSE, SSIM, PSNR, and MAE. This suggests that the pre-trained CNN might be omitting critical information that is preserved by bicubic interpolation. A significant advantage of removing the CNN is that it makes the ResDiff model faster, more efficient, and simpler to train.

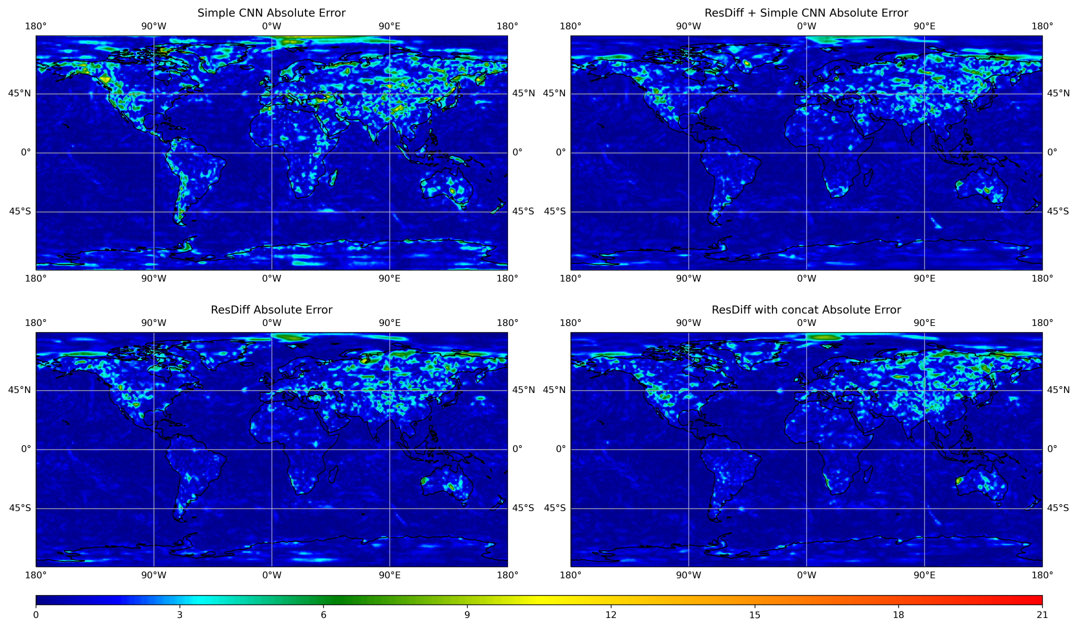
Simple CNN Absolute Error
ResDiff + Simple CNN Absolute Error
ResDiff Absolute Error
ResDiff with concat Absolute Error
180°
90°W
0°W
90°E
45°N
0°
45°S
0
3
6
9
12
15
18
21

Figure 4.3 Comparative visualization of absolute error in Kelvins across various models: Simple CNN, ResDiff + Simple CNN, ResDiff, and ResDiff with concatenation. The color scale at the bottom indicates the magnitude of errors.

4.0.5 Model comparison

Moreover, we analyzed the top-performing variants of each diffusion model discussed in this work. Utilizing SR3 as a reference point, data from Figure 4.4 shows that SRDiff, ResDiff, and ResDiff+Physics all significantly outperform the SR3 method across the entire 200,000 iterations. Unlike SR3, which shows considerable fluctuations in accuracy during training, the SRDiff and ResDiff-based methods consistently achieve an SSIM above 0.9. Notably, SR3 reaches its highest validation SSIM at 120,000 iterations, earlier than the other models. ResDiff+Physics and ResDiff achieve their optimal validation SSIM at 190,000 and 170,000 iterations, respectively, with SRDiff peaking at 150,000 iterations. Interestingly, ResDiff+Physics demonstrates considerable instability in MSE, PSNR, and MAE metrics up to 80,000 iterations. This instability is interesting since the integration of convolutional physical-based filters into the ResDiff framework does not add any new learnable parameters. Beyond 80,000 iterations, ResDiff+Physics's performance aligns closely with that of ResDiff and SRDiff.

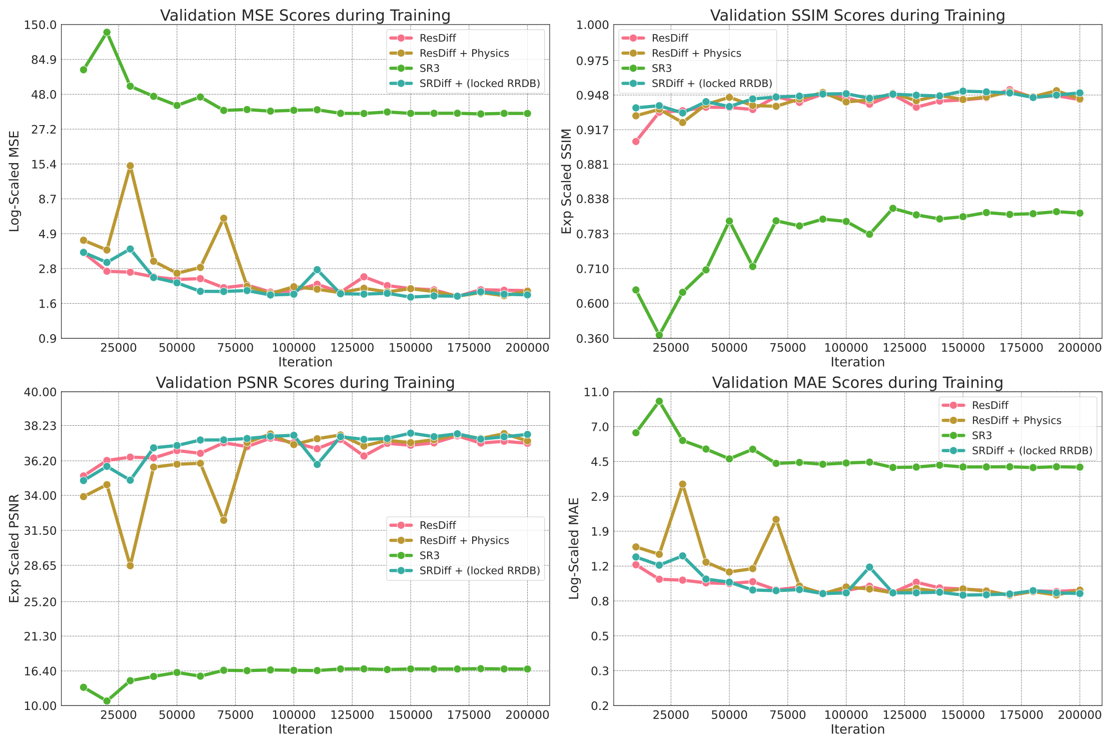

Figure 4.4 Model validation scores during training across models: SR3, ResDiff, ResDiff + Physics, and SRDiff. The Y-axis is scaled exponentially for SSIM and PSNR, and logarithmically for other metrics.

Overall, as shown in Table 4.4, the ResDiff model achieves the highest SSIM and MAE scores across all models tested, while the SRDiff architecture achieves the best MSE and PSNR scores. ResDiff+Physics represents a middle ground between these models, achieving better SSIM than SRDiff but not as high as ResDiff, and better PSNR than ResDiff but not equaling SRDiff. In contrast, SR3 performs significantly worse in MSE and other metrics, even underperforming bicubic interpolated outputs. This underscores the substantial improvement in performance when residual-based prediction is incorporated into diffusion models for weather super-resolution. As demonstrated, models such as SRDiff, ResDiff, and ResDiff+Physics effectively capture low-frequency information in weather data, producing results that are difficult to distinguish from high-resolution reference images, as illustrated in Figure 4.5.

Table 4.4 Validation results of top-performing variants of each diffusion model discussed in this work.

Model	MSE ↓	SSIM ↑	PSNR ↑	MAE ↓
SR3	35.15	0.824	16.71	4.477
SRDiff + (locked RRDB)	**1.748**	0.951	**37.81**	0.820
ResDiff	1.768	**0.953**	37.65	**0.813**
ResDiff + Physics	1.789	0.952	37.78	0.821
Bicubic	6.866	0.854	31.52	1.542
Ground Truth	0	1	∞	0

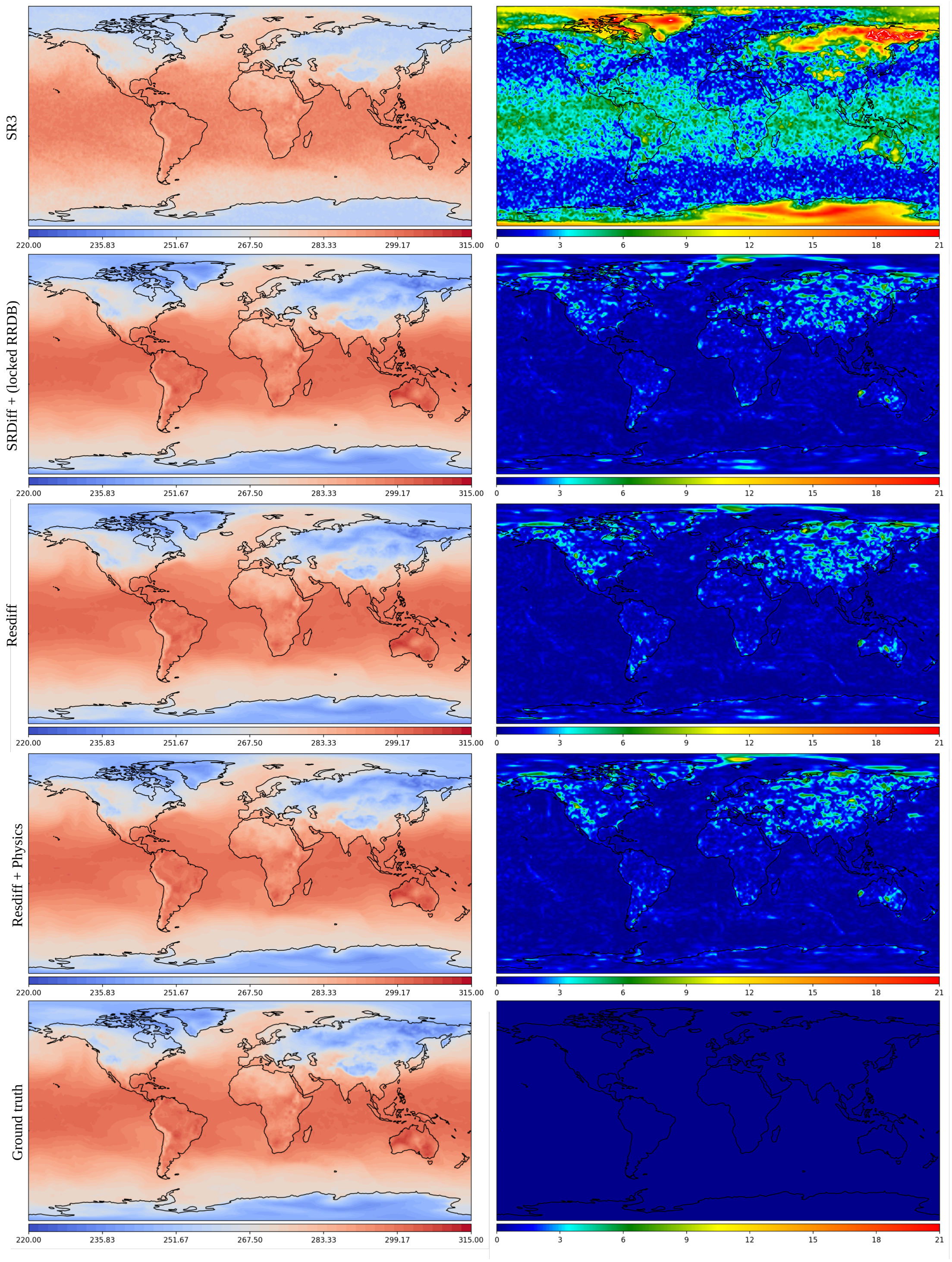

Figure 4.5 Comparison of images generated by the SR3, SRDiff, ResDiff, and ResDiff+Physics models. The left column displays super-resolution images generated by each model, annotated with their corresponding temperatures in Kelvins. The right column illustrates the absolute error of each model compared to the high-resolution ground truth image, with error magnitudes indicated on the color bar under each image.

4.0.6 More experiments

In all prior experiments, we exclusively used data from January as our training set. To evaluate performance across various months and seasons, we trained 12 distinct ResDiff+Physics models, one for each month. All these models were trained using the same hyperparameters for 190,000 iterations. Each model was evaluated on data from the same month it was trained on, using data collected from January 1, 2016, to December 31, 2016.

Table 4.5 Validation results of various models trained across different months.

Metric	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
MSE ↓	1.789	1.654	1.753	1.621	1.617	1.609	1.487	1.435	1.351	1.523	1.732	1.949
RMSE ↓	1.338	1.286	1.324	1.273	1.294	1.268	1.219	1.198	1.162	1.234	1.316	1.396
SSIM ↑	0.951	0.956	0.958	0.962	0.960	0.966	0.967	0.964	0.965	0.956	0.949	0.950
PSNR ↑	37.78	38.60	38.97	39.39	39.49	40.20	40.77	40.55	41.06	39.32	37.72	37.33
MAE ↓	0.821	0.813	0.852	0.815	0.822	0.790	0.772	0.768	0.760	0.803	0.825	0.864

Table 4.5 reveals that model performance varies significantly with the changing months. Nearly all models outperform the baseline January model used in previous experiments. From January to September, there is a general upward trend in validation scores for all metrics, with some exceptions. Particularly notable are the results from July, where ResDiff+Physics reaches an impressive SSIM of 96.7, and September, where it achieves an MSE of 1.351, representing a 25% improvement compared to the January MSE of 1.789. November and December, while being the lowest-performing months, still produce very good results. This demonstrates that ResDiff+Physics consistently performs well throughout the year. Notably, as shown in Figure 4.6, the models have a tendency to overestimate temperatures in certain areas while underestimating them in others, depending on the month or season.

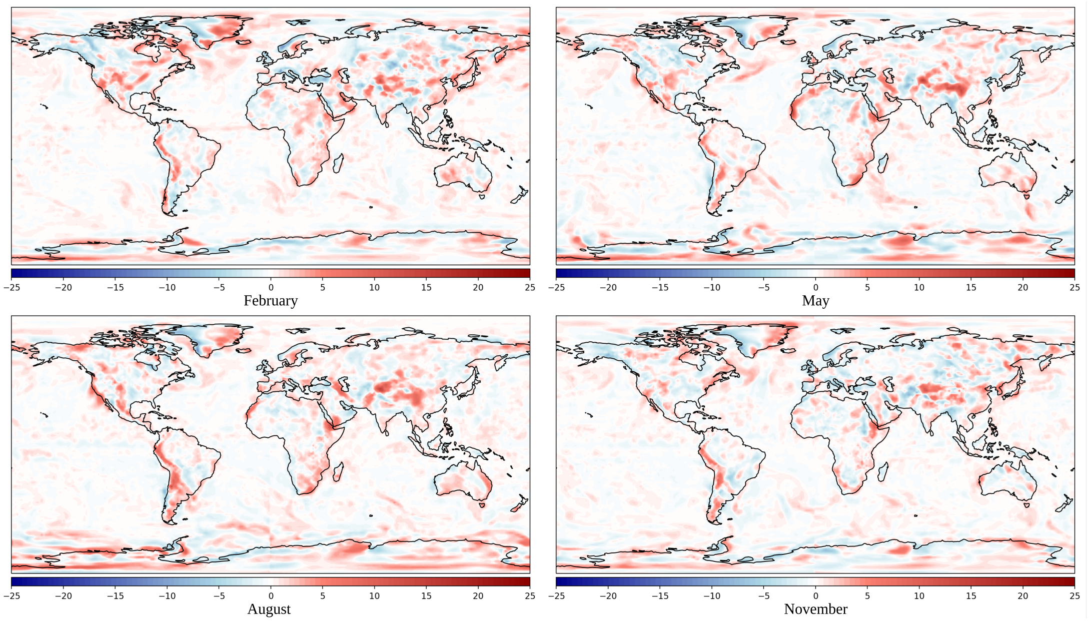

Figure 4.6 Average error of models in February, May, August, and November measured in Kelvins. Red areas indicate regions where the model's predictions were overestimated, blue areas where they were underestimated, and white areas represent locations with an error of less than one Kelvin.

Diffusion models are computationally intensive during inference, as they must utilize a T-step denoising process for image sampling. To find an optimal balance between inference duration (number of steps T) and output quality, we utilized Phydiff + Physics with varying numbers of diffusion steps during both inference and training phases. We trained four different models, each with a distinct number of training time steps: ($T_t = 100$, $T_t = 500$, $T_t = 1000$, $T_t = 2000$). The training timestep parameter does not affect inference time. It determines the range of numbers used in forward noising process and for conditioning the denoising network. Each of these four models were then evaluated using sampling timesteps of ($T_s = 100$, $T_s = 500$, $T_s = 1000$). These numbers specify how many times the denoising network is utilized to gradually denoise the image to its final high-resolution version. In the original DDPM paper [1], the authors suggest that the ideal number of diffusion steps for both training and validation is ($T = 1000$).

Table 4.6 Validation results of models trained with varying numbers of training and validation steps

Training	Validation	MSE ↓	SSIM ↑	PSNR ↑	MAE ↓
T_t=100	T_s=100	5.674	0.772	32.90	1.858
	T_s=500	**1.773**	**0.950**	37.72	**0.823**
	T_s=1000	1.791	**0.950**	**37.74**	0.829
T_t=500	T_s=100	5.822	0.769	32.79	1.881
	T_s=500	**1.855**	**0.949**	37.45	**0.842**
	T_s=1000	1.869	**0.949**	**37.49**	0.846
T_t=1000	T_s=100	5.01	0.797	33.48	1.734
	T_s=500	**1.778**	**0.952**	37.73	**0.819**
	T_s=1000	1.789	**0.952**	**37.78**	0.821
T_t=2000	T_s=100	4.687	0.809	33.65	1.671
	T_s=500	**1.955**	**0.944**	**37.53**	**0.881**
	T_s=1000	1.974	**0.944**	37.45	0.887

Table 4.6 demonstrates that the number of training steps (T_t) has less of an impact on the final validation scores compared to the number of sampling steps (T_s). Training with ($T_t = 1000$) achieves the best results in terms of SSIM, PSNR, and MAE. Surprisingly, training with a smaller number of timesteps ($T_t = 100$) results in the best MSE, which is unexpected, as one would typically expect that a model trained on fewer timesteps and validated on more timesteps would underperform compared to a model trained and validated on the same number of steps. Using a higher number of training timesteps ($T_t = 2000$) results in worse performance with lower validation scores compared to other models. When a smaller number of sample timesteps ($T_s = 100$) is used, models trained on a larger (T_t) demonstrate improved performance. However, these results are suboptimal, with an MSE of around 5 and SSIM of around 0.78. A key observation from Table 4.6 is that using ($T_s = 500$) for sampling often performs similarly or better than ($T_s = 1000$). This suggests that using ($T_s = 500$) for model inference not only doubles the model inference speed but also enhances the outcomes compared to ($T_s = 1000$) used in all previous experiments.

Conclusion

In this thesis, we researched the application of advanced deep-learning diffusion models, specifically SR3, SRDiff, and ResDiff for the super-resolution of weather data. Initially, we introduced the reader to the problem of weather super-resolution, and then we focused on understanding the fundamental operational principles of each model, followed by a series of experiments to evaluate their performance on weather data using a variety of validation metrics. To enhance the performance of these models, we experimented with architectural changes and the incorporation of physical-based convolutional filters.

Experiment results demonstrated that advanced architectures such as ResDiff, SRDiff, and ResDiff+Physics achieved significantly better results than the SR3 model or interpolation methods. One of the main reasons for the superior performance of ResDiff, SRDiff, and ResDiff+Physics is their use of a residual-based prediction strategy. A smaller but still significant influence on this performance also comes from how these diffusion models are conditioned. Experiments showed that adding more feature maps for conditioning does not always lead to better results, as demonstrated by ResDiff+Physics. However, this does not imply that integrating domain-specific knowledge, such as physics filters, can't further improve super-resolution outcomes. Although ResDiff+Physics did not achieve the highest score in any single validation metric, it outperformed both ResDiff and SRDiff in two metrics. Additionally, ResDiff+Physics has a simpler architecture because the FD info splitter was replaced with convolutional filters, making this model faster. Further experiments revealed that for weather data using half as many iterations during sampling as originally suggested by the authors results in better validation scores and a significant speedup in inference.

This work has demonstrated the effectiveness of deep-learning diffusion models, specifically SRDiff, ResDiff, and ResDiff+Physics, in the super-resolution of weather data and highlights the potential for further exploration and improvement of diffusion models in meteorology.

Outline of future work

Future work should focus on further refining diffusion model architectures to even more enhance their performance on weather data. In this thesis, we attempted to improve the ResDiff model by incorporating convolutional filters designed to approximate spatial derivatives. However, these filters did not significantly change the validation results of this model. One idea is to allow the model to autonomously learn these spatial derivative filters, which could advance their ability to adaptively understand atmospheric dynamics.

Another potential challenge is to further refine the loss function to account for the physical properties of weather, which could improve the results of these models.

More effort could also be put into evaluating these models on different datasets and comparing them to other weather super-resolution models.

Bibliography

1. HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. *Denoising Diffusion Probabilistic Models*. 2020. Available from arXiv: `2006.11239 [cs.LG]`.
2. SAHARIA, Chitwan; HO, Jonathan; CHAN, William; SALIMANS, Tim; FLEET, David J.; NOROUZI, Mohammad. *Image Super-Resolution via Iterative Refinement*. 2021. Available from arXiv: `2104.07636 [eess.IV]`.
3. LI, Haoying; YANG, Yifan; CHANG, Meng; FENG, Huajun; XU, Zhihai; LI, Qi; CHEN, Yueting. *SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models*. 2021. Available from arXiv: `2104.14951 [cs.CV]`.
4. SHANG, Shuyao; SHAN, Zhengyang; LIU, Guangxing; ZHANG, Jinglin. *ResDiff: Combining CNN and Diffusion Model for Image Super-Resolution*. 2023. Available from arXiv: `2303.08714 [cs.CV]`.
5. RASP, Stephan; DUEBEN, Peter D.; SCHER, Sebastian; WEYN, Jonathan A.; MOUATADID, Soukayna; THUEREY, Nils. WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting. *Journal of Advances in Modeling Earth Systems*. 2020, vol. 12, no. 11. Available from DOI: `10.1029/2020ms002203`.
6. YANG, Wenming; ZHANG, Xuechen; TIAN, Yapeng; WANG, Wei; XUE, Jing-Hao; LIAO, Qingmin. Deep Learning for Single Image Super-Resolution: A Brief Review. *IEEE Transactions on Multimedia*. 2019, vol. 21, no. 12, pp. 3106–3121. ISSN 1941-0077. Available from DOI: `10.1109/tmm.2019.2919431`.
7. SHI, Zhanpeng; GENG, Huantong; WU, Fangli; GENG, Liangchao; ZHUANG, Xiaoran. A Weather Radar Image Super-Resolution Generation Model Based on SR3. *Atmosphere*. 2024, vol. 15, no. 1. ISSN 2073-4433. Available from DOI: `10.3390/atmos15010040`.
8. FADNAVIS, Shreyas. Image interpolation techniques in digital image processing: an overview. *International Journal of Engineering Research and Applications*. 2014, vol. 4, no. 10, pp. 70–73.
9. SUN, Jian; XU, Zongben; SHUM, Heung-Yeung. Image super-resolution using gradient profile prior. In: 2008. Available from DOI: `10.1109/CVPR.2008.4587659`.
10. O'SHEA, Keiron; NASH, Ryan. *An Introduction to Convolutional Neural Networks*. 2015. Available from arXiv: `1511.08458 [cs.NE]`.
11. DONG, Chao; LOY, Chen Change; HE, Kaiming; TANG, Xiaoou. *Image Super-Resolution Using Deep Convolutional Networks*. 2015. Available from arXiv: `1501.00092 [cs.CV]`.
12. KIM, Jiwon; LEE, Jung Kwon; LEE, Kyoung Mu. *Accurate Image Super-Resolution Using Very Deep Convolutional Networks*. 2016. Available from arXiv: `1511.04587 [cs.CV]`.

13. LIM, Bee; SON, Sanghyun; KIM, Heewon; NAH, Seungjun; LEE, Kyoung Mu. *Enhanced Deep Residual Networks for Single Image Super-Resolution.* 2017. Available from arXiv: `1707.02921 [cs.CV]`.

14. GOODFELLOW, Ian J.; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. *Generative Adversarial Networks.* 2014. Available from arXiv: `1406.2661 [stat.ML]`.

15. LEDIG, Christian; THEIS, Lucas; HUSZAR, Ferenc; CABALLERO, Jose; CUNNINGHAM, Andrew; ACOSTA, Alejandro; AITKEN, Andrew; TEJANI, Alykhan; TOTZ, Johannes; WANG, Zehan; SHI, Wenzhe. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.* 2017. Available from arXiv: `1609.04802 [cs.CV]`.

16. WANG, Xintao; YU, Ke; WU, Shixiang; GU, Jinjin; LIU, Yihao; DONG, Chao; LOY, Chen Change; QIAO, Yu; TANG, Xiaoou. *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks.* 2018. Available from arXiv: `1809.00219 [cs.CV]`.

17. XU, Min; DING, Youdong. Single-image Super-resolution Based on Generative Adversarial Network with Dual Attention Mechanism. In: *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC).* 2022, vol. 5, pp. 239–250. Available from DOI: `10.1109/IMCEC55388.2022.10020076`.

18. LIANG, Jingyun; CAO, Jiezhang; SUN, Guolei; ZHANG, Kai; GOOL, Luc Van; TIMOFTE, Radu. *SwinIR: Image Restoration Using Swin Transformer.* 2021. Available from arXiv: `2108.10257 [eess.IV]`.

19. CAO, Jiezhang; LI, Yawei; ZHANG, Kai; GOOL, Luc Van. *Video Super-Resolution Transformer.* 2023. Available from arXiv: `2106.06847 [cs.CV]`.

20. CHEN, Hanting; WANG, Yunhe; GUO, Tianyu; XU, Chang; DENG, Yiping; LIU, Zhenhua; MA, Siwei; XU, Chunjing; XU, Chao; GAO, Wen. *Pre-Trained Image Processing Transformer.* 2021. Available from arXiv: `2012.00364 [cs.CV]`.

21. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain; USZKOREIT, Jakob; HOULSBY, Neil. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* 2021. Available from arXiv: `2010.11929 [cs.CV]`.

22. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need.* 2023. Available from arXiv: `1706.03762 [cs.CL]`.

23. BAÑO-MEDINA, Jorge; GUTIÉRREZ, José Manuel; HERRERA, Sixto. Deep Neural Networks for Statistical Downscaling of Climate Change Projections 1. In: [online]. 2018 [visited on 2024-01-02]. Available from: `https://api.semanticscholar.org/CorpusID:203644846`.

24. U.S. DEPARTMENT OF ENERGY. *DOE Explains: Earth System and Climate Models* [online]. U.S. Department of Energy. [visited on 2024-04-15]. Available from: `https://www.energy.gov/science/doe-explainsearth-system-and-climate-models`.

25. ARC CENTRE OF EXCELLENCE FOR CLIMATE EXTREMES. *What Does Climate Model Resolution Mean?* [online]. ARC Centre of Excellence for Climate Extremes, 2024-05. [visited on 2024-04-02]. Available from: `https://climateextremes.org.au/what-does-climate-model-resolution-mean/`.

26. HEWITSON, Bruce; CRANE, Robert. Climate Downscaling: Techniques and Application. *Climate Research.* 1996, vol. 07, pp. 85–. Available from DOI: `10.3354/cr007085`.

27. ACCARINO, Gabriele; CHIARELLI, Marco; IMMORLANO, Francesco; ALOISI, Valeria; GATTO, Andrea; ALOISIO, Giovanni. MSG-GAN-SD: A Multi-Scale Gradients GAN for Statistical Downscaling of 2-Meter Temperature over the EURO-CORDEX Domain. *AI*. 2021, vol. 2, no. 4, pp. 600–620. ISSN 2673-2688. Available from DOI: `10.3390/ai2040036`.

28. VANDAL, Thomas; KODRA, Evan; GANGULY, Sangram; MICHAELIS, Andrew; NEMANI, Ramakrishna; GANGULY, Auroop R. DeepSD: Generating High Resolution Climate Change Projections through Single Image Super-Resolution. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 1663–1672. KDD '17. ISBN 9781450348874. Available from DOI: `10.1145/3097983.3098004`.

29. LAPRISE, René. Regional climate modelling. *Journal of Computational Physics*. 2008, vol. 227, no. 7, pp. 3641–3666. ISSN 0021-9991. Available from DOI: `https://doi.org/10.1016/j.jcp.2006.10.024`. Predicting weather, climate and extreme events.

30. SERIFI, Agon; GÜNTHER, Tobias; BAN, Nikolina. Spatio-Temporal Downscaling of Climate Data Using Convolutional and Error-Predicting Neural Networks. *Frontiers in Climate*. 2021, vol. 3. ISSN 2624-9553. Available from DOI: `10.3389/fclim.2021.656479`.

31. SEKIYAMA, Tsuyoshi Thomas. *Statistical Downscaling of Temperature Distributions from the Synoptic Scale to the Mesoscale Using Deep Convolutional Neural Networks*. 2020. Available from arXiv: `2007.10839 [physics.ao-ph]`.

32. LEINONEN, Jussi; NERINI, Daniele; BERNE, Alexis. Stochastic Super-Resolution for Downscaling Time-Evolving Atmospheric Fields With a Generative Adversarial Network. *IEEE Transactions on Geoscience and Remote Sensing*. 2021, vol. 59, no. 9, pp. 7211–7223. Available from DOI: `10.1109/TGRS.2020.3032790`.

33. HARDER, Paula; HERNANDEZ-GARCIA, Alex; RAMESH, Venkatesh; YANG, Qidong; SATTIGERI, Prasanna; SZWARCMAN, Daniela; WATSON, Campbell; ROLNICK, David. *Hard-Constrained Deep Learning for Climate Downscaling*. 2024. Available from arXiv: `2208.05424 [physics.ao-ph]`.

34. GERGES, Firas; BOUFADEL, Michel; BOU-ZEID, Elie; NASSIF, Hani; WANG, Jason. A Novel Deep Learning Approach to the Statistical Downscaling of Temperatures for Monitoring Climate Change. In: 2022, pp. 1–7. Available from DOI: `10.1145/3523150.3523151`.

35. PAPIKYAN, Davit; HÖHLEIN, Kevin; WESTERMANN, Rüdiger. *Probabilistic Downscaling of Climate Variables Using Denoising Diffusion Probabilistic Models* [online]. 2022. [visited on 2023-10-10]. Available from: `https://github.com/davitpapikyan/Probabilistic-Downscaling-of-Climate-Variables/blob/main/report.pdf`.

36. SOHL-DICKSTEIN, Jascha; WEISS, Eric A.; MAHESWARANATHAN, Niru; GANGULI, Surya. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. Available from arXiv: `1503.03585 [cs.LG]`.

37. RUTHOTTO, Lars; HABER, Eldad. *An Introduction to Deep Generative Modeling*. 2021. Available from arXiv: `2103.05180 [cs.LG]`.

38. WENG, Lilian. What are diffusion models? *lilianweng.github.io* [online]. 2021 [visited on 2024-02-02]. Available from: `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`.

39. STRAKA, Milan. *Deep Learning – Summer 2022/23* [online]. 2023. [visited on 2023-10-25]. Available from: `https://ufal.mff.cuni.cz/~straka/courses/npfl114/2223/slides.pdf/npfl114-2223-13.pdf`. This work is licensed under the CC BY-SA 4.0 License. To view a copy of this license, visit `https://creativecommons.org/licenses/by-sa/4.0/`.

40. NICHOL, Alex; DHARIWAL, Prafulla. *Improved Denoising Diffusion Probabilistic Models*. 2021. Available from arXiv: `2102.09672 [cs.LG]`.

41. KINGMA, Diederik P; WELLING, Max. *Auto-Encoding Variational Bayes.* 2022. Available from arXiv: `1312.6114 [stat.ML]`.

42. NAIN, Aakash. *A Deep Dive into Ddpms* [online]. 2022. [visited on 2024-01-02]. Available from: `https://magic-with-latents.github.io/latent/posts/ddpms/part3/`.

43. CHEN, Nanxin; ZHANG, Yu; ZEN, Heiga; WEISS, Ron J.; NOROUZI, Mohammad; CHAN, William. *WaveGrad: Estimating Gradients for Waveform Generation.* 2020. Available from arXiv: `2009.00713 [eess.AS]`.

44. GELIMSON, Lev. *Piecewise Probability Distribution Theory.* 2022. Available from arXiv: `2201.12730 [math.PR]`.

45. DALM, Sander; GERVEN, Marcel van; AHMAD, Nasir. *Effective Learning with Node Perturbation in Deep Neural Networks.* 2024. Available from arXiv: `2310.00965 [cs.LG]`.

46. KEYS, R. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 1981, vol. 29, no. 6, pp. 1153–1160. Available from DOI: `10.1109/TASSP.1981.1163711`.

47. LIANGWEI, Jiang; AYUSH, Thakur; ANDREW, Howard; MAZEOFENCRYPTION; DEVJAKE. *Image Super-Resolution via Iterative Refinement Implementation* [`https://github.com/Janspiry/Image-Super-Resolution-via-Iterative-Refinement`]. GitHub, 2021 [visited on 2024-04-25].

48. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015. Available from arXiv: `1505.04597 [cs.CV]`.

49. TOMAR, Nikhil. What is U-Net? *Medium* [`https://medium.com/analytics-vidhya/what-is-unet-157314c87634`]. 2021 [visited on 2024-03-11].

50. RIAD, Rachid; TEBOUL, Olivier; GRANGIER, David; ZEGHIDOUR, Neil. *Learning strides in convolutional neural networks.* 2022. Available from arXiv: `2202.01653 [cs.LG]`.

51. OLIVIER, Rukundo; HANQIANG, Cao. Nearest Neighbor Value Interpolation. *International Journal of Advanced Computer Science and Applications.* 2012, vol. 3, no. 4. ISSN 2156-5570. Available from DOI: `10.14569/ijacsa.2012.030405`.

52. ADALOGLOU, Nikolas. Intuitive Explanation of Skip Connections in Deep Learning. *AI Summer* [online]. 2020 [visited on 2024-02-01]. Available from: `https://theaisummer.com/skip-connections/`.

53. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition.* 2015. Available from arXiv: `1512.03385 [cs.CV]`.

54. PASCANU, Razvan; MIKOLOV, Tomas; BENGIO, Yoshua. *On the difficulty of training Recurrent Neural Networks.* 2013. Available from arXiv: `1211.5063 [cs.LG]`.

55. SAHOO, Sabyasachi. Residual Blocks: Building Blocks of ResNet. *Towards Data Science* [`https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec`]. 2018 [visited on 2024-01-05].

56. IOFFE, Sergey; SZEGEDY, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. Available from arXiv: `1502.03167 [cs.LG]`.

57. ZHANG, Aston; LIPTON, Zachary C.; LI, Mu; SMOLA, Alexander J. *Residual Networks (ResNet)* [`https://d2l.ai/chapter_convolutional-modern/resnet.html`]. d2l.ai, 2021 [visited on 2024-02-20].

58. WU, Yuxin; HE, Kaiming. *Group Normalization.* 2018. Available from arXiv: `1803.08494 [cs.CV]`.

59. RAMACHANDRAN, Prajit; ZOPH, Barret; LE, Quoc V. *Searching for Activation Functions.* 2017. Available from arXiv: `1710.05941 [cs.NE]`.

60. RASCHKA, Sebastian. *Self-Attention From Scratch* [https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html]. Sebastian Raschka's Blog, 2023 [visited on 2024-02-27].

61. 3BLUE1BROWN. *Attention in Transformers, Visually Explained — Chapter 6, Deep Learning* [https://www.youtube.com/watch?v=eMlx5fFNoYc]. 2024. [visited on 2024-04-25].

62. ZHANG, Han; GOODFELLOW, Ian; METAXAS, Dimitris; ODENA, Augustus. *Self-Attention Generative Adversarial Networks*. 2019. Available from arXiv: 1805.08318 [stat.ML].

63. LUGMAYR, Andreas; DANELLJAN, Martin; GOOL, Luc Van; TIMOFTE, Radu. *SRFlow: Learning the Super-Resolution Space with Normalizing Flow*. 2020. Available from arXiv: 2006.14200 [cs.CV].

64. XU, Bing; WANG, Naiyan; CHEN, Tianqi; LI, Mu. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. Available from arXiv: 1505.00853 [cs.LG].

65. SHI, Wenzhe; CABALLERO, Jose; HUSZÁR, Ferenc; TOTZ, Johannes; AITKEN, Andrew P.; BISHOP, Rob; RUECKERT, Daniel; WANG, Zehan. *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. 2016. Available from arXiv: 1609.05158 [cs.CV].

66. COOLEY, James W.; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation* [online]. 1965, vol. 19, pp. 297–301 [visited on 2024-03-03]. Available from: https://api.semanticscholar.org/CorpusID:121744946.

67. MALLAT, Stéphane. Mallat, S.G.: A Theory of Multiresolution Signal Decomposition: The Wavelet Representation. IEEE Trans. Pattern Anal. Machine Intell. 11, 674-693. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 1989, vol. 11, pp. 674–693. Available from DOI: 10.1109/34.192463.

68. SHAHBAHRAMI, Asadollah. Algorithms and architectures for 2D discrete wavelet transform. *The Journal of Supercomputing*. 2012, vol. 62, pp. 1045–1064.

69. HU, Jie; SHEN, Li; ALBANIE, Samuel; SUN, Gang; WU, Enhua. *Squeeze-and-Excitation Networks*. 2019. Available from arXiv: 1709.01507 [cs.CV].

70. SCHNEIDERBAUER, Simon; KRIEGER, Michael. What do the Navier–Stokes equations mean? *European Journal of Physics*. 2013, vol. 35, no. 1, p. 015020. Available from DOI: 10.1088/0143-0807/35/1/015020.

71. HERSBACH, Hans; BELL, Bill; BERRISFORD, Paul; HIRAHARA, Shoji; HORÁNYI, András; MUÑOZ-SABATER, Joaquín; NICOLAS, Julien; PEUBEY, Carole; RADU, Raluca; SCHEPERS, Dinand; SIMMONS, Adrian; SOCI, Cornel; ABDALLA, Saleh; ABELLAN, Xavier; BALSAMO, Gianpaolo; BECHTOLD, Peter; BIAVATI, Gionata; BIDLOT, Jean; BONAVITA, Massimo; DE CHIARA, Giovanna; DAHLGREN, Per; DEE, Dick; DIAMANTAKIS, Michail; DRAGANI, Rossana; FLEMMING, Johannes; FORBES, Richard; FUENTES, Manuel; GEER, Alan; HAIMBERGER, Leo; HEALY, Sean; HOGAN, Robin J.; HÓLM, Elías; JANISKOVÁ, Marta; KEELEY, Sarah; LALOYAUX, Patrick; LOPEZ, Philippe; LUPU, Cristina; RADNOTI, Gabor; ROSNAY, Patricia de; ROZUM, Iryna; VAMBORG, Freja; VILLAUME, Sebastien; THÉPAUT, Jean-Noël. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*. 2020, vol. 146, no. 730, pp. 1999–2049. Available from DOI: https://doi.org/10.1002/qj.3803.

72. DAVITPAPIKYAN. *Probabilistic Downscaling of Climate Variables* [https://github.com/davitpapikyan/Probabilistic-Downscaling-of-Climate-Variables]. GitHub, 2022 [visited on 2023-04-09].

73. LIN, Yunlong. *ResDiff* [https://github.com/LYL1015/ResDiff]. GitHub, 2023 [visited on 2023-10-08].

74. LEIALI. *SRDiff* [https://github.com/LeiaLi/SRDiff]. GitHub, 2023 [visited on 2023-05-07].

75. WANG, Zhou; BOVIK, A.C.; SHEIKH, H.R.; SIMONCELLI, E.P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing.* 2004, vol. 13, no. 4, pp. 600–612. Available from DOI: 10.1109/TIP.2003.819861.

76. WANG, Zhou; LU, Ligang; BOVIK, Alan C. Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication.* 2004, vol. 19, no. 2, pp. 121–132. ISSN 0923-5965. Available from DOI: https://doi.org/10.1016/S0923-5965(03)00076-6.

77. HORÉ, Alain; ZIOU, Djemel. Image Quality Metrics: PSNR vs. SSIM. In: *2010 20th International Conference on Pattern Recognition.* 2010, pp. 2366–2369. Available from DOI: 10.1109/ICPR.2010.579.

Contents of the attachment

`configs`	directory containing model and data configurations
`data`	directory containing data manipulation modules
`example`	directory with examples
`experiments`	default directory for experiment saves
`logger`	directory with logging modules
`models`	
`diffusion_models`	directory containing implementation of diffusion models
`rrdb_encoder`	RRDB encoder implementation
`simple_cnn`	simple CNN implementation
`base_model.py`	model base abstract class
`thesis`	
`src`	directory containing thesis LaTeX source files
`thesis.pdf`	thesis document in PDF format
`training`	directory containing modules used during training
`pretrain.py`	script for pretraining models
`README.md`	general information and instructions about code
`sample.py`	script for image sampling
`train.py`	main training script
`example.ipynb`	code examples
`requirements.txt`	list of dependencies