3D Fractal Pre-training for Action Recognition

AALBORG UNIVERSITY
STUDENT REPORT

Master's Thesis

Marko Putak

CE-AVS10 Aalborg University

AALBORG UNIVERSITY
STUDENT REPORT

CE-AVS10
Aalborg University
http://www.aau.dk

Title:
3D Fractal Pre-training for Action Recognition

Project Period:
Spring semester 2025

Project Group:
AVS10 (CE10)

Participants:
Marko Putak

Supervisor:
Thomas B. Moeslund
Joakim B. Haurum

Copies: 1

Page Numbers: 76

Date of Completion:
June 4, 2025

Abstract:

Recent advances in computer vision have underscored the importance of large, diverse datasets for deep learning, but collecting real-world action recognition videos remains costly and challenging. This thesis explores synthetic data generated from 3D fractal geometry as a scalable, privacy-preserving resource for pre-training action recognition models. Leveraging Iterated Function Systems (IFS), a pipeline was developed to generate diverse 3D fractal point clouds, transform them into synthetic videos and construct large datasets for neural network pre-training. Experiments with a ResNet-50 backbone and Temporal Shift Module (TSM) show that pre-training on fractal-based datasets significantly outperforms training from scratch. Systematic studies reveal that dataset size, transformation strength, color augmentation and fractal geometry control all impact downstream performance. Data-driven methods for controlling fractal structure, such as condition number constraints and SVM-informed weighting of singular values, further enhance the visual diversity and quality of the data. While 3D fractal pre-training does not yet surpass the strongest 2D fractal baselines, it narrows the gap and demonstrates the practical potential of formula-driven synthetic data for scalable action recognition in domains with limited real data.

Preface

Note: Certain aspects of this thesis, including grammar revision and the preparation of LaTeX figures and tables, were supported by GPT-based language models [72, 37]. Code completion and simple visualization was aided by Github Copilot [35].

I am sincerely grateful to my supervisors, Thomas B. Moeslund and Joakim B. Haurum, for their significant effort and insightful mentorship throughout this thesis. I am also deeply grateful to Bruna Duspara for her unwavering support and encouragement during my studies.

Aalborg University, June 4, 2025

Marko Putak
mputak23@student.aau.dk

Contents

Acronyms

CNN Convolutional Neural Network. 2, 5, 14, 33

CPU Central Processing Unit. 30

CV Computer Vision. 2, 3

DL Deep Learning. 2, 17, 24

FDSL Formula-Driven Supervised Learning. 3, 13–17, 65, 66

FPS Frames Per Second. 28, 31

IFS Iterated Function System. 11, 13, 18–21, 24, 26, 27, 35, 56, 58, 60, 61, 66, 67

SotA State of the Art. 5

SSL Self-supervised Learning. 13

SVD Singular Value Decomposition. 20, 56

SVM Support Vector Machines. 4, 59, 65, 67

TSM Temporal Shift Module. 2, 33, 34, 64

1 Introduction

The rapid progression of Computer Vision (CV) and Deep Learning (DL) technologies in recent years has enabled new directions in machine perception, making it possible to tackle complex visual understanding tasks. Among these, *Action Recognition* stands out as a vital research area. Its primary goal is to accurately identify and classify human actions captured in video sequences [106]. It is a non-trivial task demanding systems capable of discriminating subtle variations in movement and temporal dependencies. The successful deployment of robust action recognition systems is critical for numerous real-world applications, including but not limited to: sport analysis, robotics, healthcare monitoring, advanced surveillance, and enabling more robust interaction in human-computer interfaces [50, 6].

1.1 Computer Vision and Action Recognition

Defined broadly as the science of enabling computers to ”see”, interpret, and understand visual data from images or videos, Computer Vision (CV) has recently aided an abundance of real-world problems. A significant catalyst for this transformation was the integration of neural networks, particularly the rise of the Convolutional Neural Network (CNN) [82]. CNNs gained substantial traction following the groundbreaking performance of AlexNet on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [58], which revolutionized image processing by effectively learning hierarchical feature representations directly from raw pixel data. Initially applied primarily to static image tasks like object classification and detection [34, 58, 44], the success of Convolutional Neural Networks paved the way for applying Deep Learning techniques to the more complex domain of video analysis.

Analyzing video, however, introduces the complexity of a temporal dimension. Unlike static images, understanding actions requires processing not just spatial features within individual frames, but also their temporal relationships across a sequence of frames. This necessity drove the development of spatio-temporal deep learning architectures specifically designed for video, such as 3D Convolutional Neural Networks [91], Temporal Segment Networks (TSN) [95], Temporal Shift Module (TSM)s [63], and more recently, Vision Transformers adapted for video, such as the Video ViT from Arnab et al. [9]. These models extend the feature learning capabilities of 2D Convolutional Neural Networks to capture motion and temporal context, making robust action recognition feasible.

Despite these architectural advancements and the growing capabilities of models and hardware, a persistent challenge in action recognition, as with many data-hungry Deep Learning tasks, is the reliance on large-scale, diverse, and well-annotated video datasets for training. Acquiring and labeling such datasets is often a time-consuming, expensive, and labor-intensive process. Furthermore, real-world data can suffer from inherent biases and inconsistencies. Lastly, a significant downside is the privacy concern and GDPR-related issues associated with creating or using datasets featuring individuals.

These limitations in data availability can impede the training of models capable of generalizing to the multitude of environments, viewpoints, and action variations encountered in practical applications. These inherent data constraints necessitate the exploration of alternative data sources and training strategies. One promising direction lies in leveraging synthetic data: artificially generated data that can potentially overcome the limitations of real-world datasets in terms of scale, diversity, annotation accuracy, and privacy concerns. The following section delves into the motivation for employing synthetic data, specifically data derived from 3D fractals, as a novel approach to address the data scarcity challenge in action recognition through effective pre-training.

1.2 Research Motivation

Building upon the identified challenges regarding the availability and limitations of real-world video datasets for training robust action recognition models, this research is fundamentally motivated by exploring alternative data generation strategies. While one synthetic generation strategy might involve generating videos of simulated human avatars, such methods are costly and labor-intensive. It diminishes the idea that leveraging synthetic data offers a promising approach to overcome issues of scale, diversity, annotation cost and privacy inherent in real-world data collection. While various forms of synthetic data have been applied in Computer Vision, this thesis investigates the novel application of data derived from 3D fractals as a source for pre-training for action recognition. The rationale stems from the unique properties of fractals: their ability to be generated computationally with infinite variability and their inherent visual complexity, including intricate shapes and contours. A large benefit of fractal generation originates from Formula-Driven Supervised Learning (FDSL) [40], as it enables labels to be assigned automatically and with perfect correspondence to the generation parameters. In contrast to real-world datasets, where annotation errors and ambiguities can arise, synthetic fractal data ensures that every sample is labeled with complete accuracy by design, eliminating labeling noise during pre-training. It is hypothesized that training a model on dynamic sequences of such abstract, yet structured, patterns can inject powerful feature learning capabilities transferable to the task of action recognition. Moreover, the approach holds the ability for extension to other domains, such as animal action recognition.

2 Background and Related Work

This chapter is devoted to providing a comprehensive overview of the foundational concepts and relevant prior research underlying this thesis. The field of action recognition is first introduced, with its key challenges and the evolution of approaches used to address them being outlined. The domain of synthetic data generation in computer vision is subsequently examined, with existing methodologies and their respective advantages and disadvantages being reviewed. Fractal geometry is then introduced, with particular emphasis placed on the unique properties of 3D fractals that render them promising for synthetic data generation in the context of deep learning. Pre-training techniques are then discussed and a comparative analysis of supervised, self-supervised and formula-driven paradigms is undertaken. Finally, prior research utilizing synthetic or abstract data patterns for pre-training is surveyed, thereby situating the focus of this work, specifically the use of 3D fractal-based synthetic data, within the broader landscape of computer vision studies.

2.1 Introduction to Action Recognition

Action recognition, a crucial problem in video analysis, involves identifying and classifying actions from video sequences. Unlike image-based tasks that analyze static scenes, action recognition requires understanding dynamic processes unfolding over time. This introduces several key challenges [13]:

- **Spatio-temporal Variability:** Actions can be performed at different speeds, durations and with variations in style and execution. Capturing the interaction between spatial appearance and temporal motion is vital.
- **Viewpoint Changes:** The same action can appear significantly different when viewed from various camera angles. Robust models aim to be invariant to viewpoint variations.
- **Background Clutter and Occlusions:** Distractions in the background and partial occlusions of the actor or action of interest can make it difficult to isolate and identify the action.
- **Illumination and Resolution Changes:** Variations in lighting conditions and video quality can impact the appearance of actions.
- **Inter-class and Intra-class Variability:** Different instances of the same action can vary greatly (e.g., different people performing ”walking”), while different actions can look very similar (e.g., ”waving” and ”clapping” from a distance).
- **Data Scarcity and Annotation Cost:** As discussed in the Introduction Chapter 1, obtaining large, diverse and accurately labeled video datasets for training is a significant obstacle.

Historically, traditional approaches to action recognition primarily relied on hand-crafted features designed to capture motion and shape. These methods often involved detecting interest points or local features in space and time, describing them and then using classical machine learning classifiers like Support Vector Machines (SVM) or Hidden Markov Models (HMMs) for recognition [78]. Specific

examples include techniques based on Space-Time Interest Points (STIPs) and optical flow-based features [61]. Beyond these, other hand-crafted approaches leveraged depth information captured from RGB-D cameras to extract human body poses and corresponding actions [100]. Furthermore, the skeletonization of the human body into a low-dimensional data representation also proved successful for action extraction [25]. While these techniques provided initial progress, they often struggled with the nuances of complex actions and the high variability encountered in real-world scenarios [69].
The involvement of deep learning has revolutionized action recognition, enabling end-to-end learning of spatio-temporal features directly from raw video data. Early approaches adapted 2D CNNs by processing individual frames or using two-stream networks that separately processed spatial information from RGB frames and temporal information from optical flow fields [85].
More advanced architectures explicitly model the spatio-temporal nature of video:

- **3D CNNs:** Extend standard 2D convolutions to three dimensions (width, height and time), allowing kernels to capture features across consecutive frames [91]. Models like I3D [16] are a notable example, which inflated 2D CNN kernels into 3D to learn spatio-temporal features.
- **Recurrent Neural Networks (RNNs):** Particularly Long Short-Term Memory (LSTM) networks, were used to model the temporal dependencies between features extracted from individual frames by CNNs [96].
- **Temporal Modeling Architectures:** Networks like Temporal Segment Networks (TSN) [95] sample sparse frames across the video and combine their predictions, while Temporal Shift Modules (TSM) [63] enable efficient temporal modeling in 2D CNNs by shifting channels across frames.
- **SlowFast Networks:** The SlowFast architecture [32] introduces a dual-pathway approach, where a ”slow” pathway operates at a low frame rate to capture semantic information, and a ”fast” pathway processes video at a higher frame rate to capture motion dynamics.
- **Transformers:** Originally dominant in natural language processing, Transformer architectures have been adapted for video. Models like the Video Vision Transformer (ViViT) [9] process video as sequences of spatio-temporal patches, leveraging self-attention mechanisms to capture long-range dependencies in both space and time [83].

These deep learning approaches have significantly advanced the State of the Art (SotA) in action recognition by learning powerful hierarchical spatio-temporal representations. Deep learning architectures have become the dominant paradigm. However, their success is heavily conditioned on the availability of massive annotated video datasets, leading back to the data challenges that motivate this research.

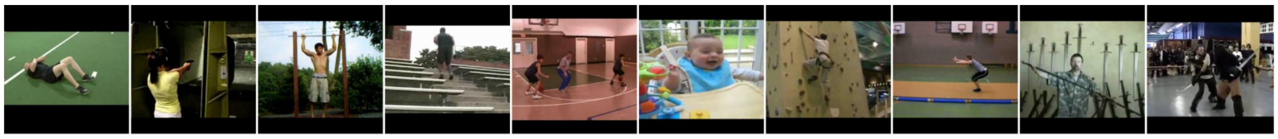
2.1.1 Benchmark Datasets for Action Recognition

To evaluate and compare action recognition models, several benchmark datasets have been established, each offering unique characteristics in terms of scale, modality, and complexity:

- **UCF101** [86]: Comprising 13,320 video clips across 101 action categories, UCF101 includes a diverse set of human actions ranging from sports to daily activities. The videos are sourced from YouTube, exhibiting variations in camera motion, object appearance, and background clutter.

- **HMDB51** [59]: This dataset contains 6,766 video clips spanning 51 action categories. The videos are collected from movies and online sources, presenting challenges such as camera motion, viewpoint changes, and occlusions, making it suitable for evaluating robustness of recognition models.
- **Kinetics** [54]: A large-scale dataset with approximately 500,000 video clips covering 600 human action classes. Each clip lasts around 10 seconds and is sourced from YouTube, providing a wide variety of scenes, subjects, and actions, which aids in training deep learning models with high generalization capabilities. Due to its extensive size and diversity, Kinetics is commonly used for pre-training action recognition models (similar to ImageNet's role in image classification).
- **Something-Something V2** [38]: Focused on fine-grained human-object interactions, this dataset comprises over 220,000 video clips annotated with 174 action categories. The actions are defined by templates requiring models to understand subtle temporal dynamics and object manipulations.
- **NTU RGB+D** [84]: Featuring 56,880 video samples across 60 action classes, this dataset includes RGB videos, depth maps, infrared videos, and 3D skeletal data. Captured with Microsoft Kinect v2 sensors, it provides multi-modal data suitable for 3D human activity analysis and cross-view evaluation.

These datasets serve as standard benchmarks for developing and assessing action recognition models. Figure 2.1 illustrates representative frames from the benchmark datasets that are of particular interest in this thesis. Their varying complexities and modalities highlight the challenges in capturing spatio-temporal patterns and emphasize the need for models that can generalize across diverse scenarios. The reliance on large-scale annotated datasets further underscores the motivation for exploring synthetic data generation and pre-training strategies, as discussed in the following sections.


(a) Sample frames from the HMDB51 dataset [59]

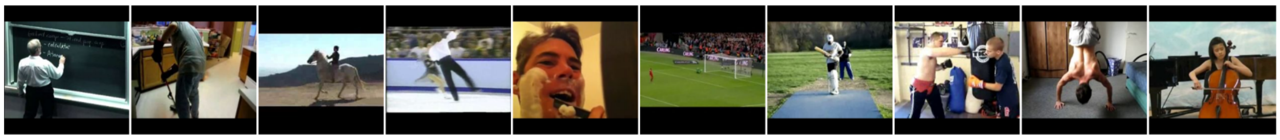

(b) Sample frames from the UCF101 dataset [86]

Figure 2.1: Examples of video frames from the two benchmark action recognition datasets used in this thesis: (a) HMDB51 and (b) UCF101. [88].

2.2 Synthetic Data Generation in Computer Vision

The increasing demand for large, diverse and accurately annotated datasets for training deep learning models in computer vision has led to a growing interest in synthetic data generation. Synthetic data refers to artificially created data that mimics the properties of real-world data but is generated programmatically. This approach offers a compelling solution to the limitations of real data, particularly the high cost and labor associated with manual collection and annotation, as well as privacy concerns. To illustrate the rising public awareness of this topic, Figure 2.2 shows worldwide Google Search interest for *”synthetic data generation”* The y-axis indicates relative search interest (0-100), with 100 being peak popularity for the period.

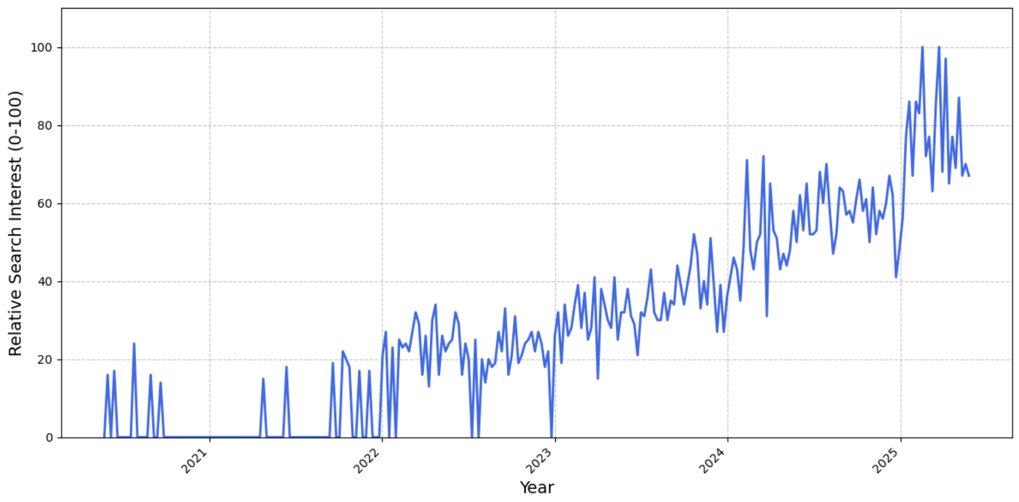


Figure 2.2: Google Search trends for *”synthetic data generation”* (Worldwide, May 2020 - May 2025), showing a general upward trajectory [1].

Figure 2.2 reveals a visible upward trend in search queries for ”synthetic data generation,” suggesting expanding curiosity from a broad audience. While not a direct measure of academic output or industry adoption, this trend serves as a proxy for the increasing mindshare and relevance of synthetic data. This aligns with synthetic data becoming a key solution for data challenges in AI and computer vision, potentially fostering further research and development.

2.2.1 Existing Methods for Synthetic Data Generation

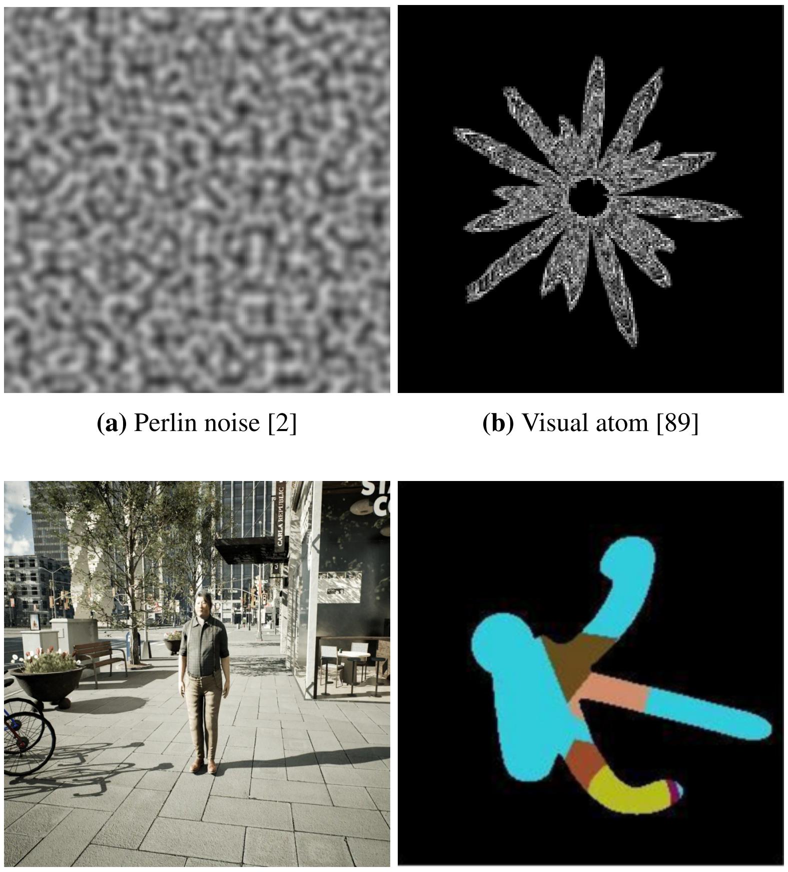
Various methodologies have been developed to generate synthetic data for diverse computer vision tasks:

- **Procedural Generation of Textures and Scenes:** Simple algorithms can generate textures like Perlin noise [77] for realistic-looking surfaces or ”dead leaves” patterns [62] for abstract scene understanding, often used in tasks related to material perception or visual reasoning. These methods allow for infinite variations by simple parameter adjustment.
- **Rule-Based or Geometric Models:** For tasks requiring specific object shapes or arrangements, synthetic data can be generated using explicit geometric models. This includes creating simple sinusoidal waves [89] or basic shapes with defined properties.
- **Simulation Environments:** More complex synthetic datasets are often generated within 3D simulation environments (e.g., Unity [93], Unreal Engine [29], Blender [18]). These environments

allow for rendering more realistic images and videos of objects, scenes and even human avatars performing actions, with ideal ground truth labels (e.g., bounding boxes, depth maps, semantic segmentation and action labels) automatically available. One such high-fidelity simulator used for autonomous driving systems is Carla [27], while some examples of synthetic datasets include houses, aerial scenery, human poses and many more. [71, 94, 55]

- **Generative Models (e.g., GANs, VAEs):** While not offering ”out-of-the-box” synthetic data in the same procedural sense, generative adversarial networks (GANs) [103] and variational autoencoders (VAEs) [24] can learn to generate new data samples that resemble a given real dataset. However, these models typically require real data to learn from and can struggle with generating truly novel or out-of-distribution samples.
- **Synthetic Data for Action Recognition:** For action-related tasks, synthetic data has been explored to teach models motion and temporal understanding. This can involve animating simple abstract shapes (e.g., a ”moving octopus” or other basic geometric transformations) [88] or generating synthetic human motion sequences in virtual environments [94]. These methods aim to provide a controlled environment for learning spatio-temporal patterns before fine-tuning on more complex human actions.

Figure 2.3 illustrates examples of existing synthetic datasets, highlighting the diversity of their generation methods and visual complexity. For instance, the abstract texture in 2.3a, generated by Perlin noise, is computationally much simpler than the photorealistic scenes produced by the CARLA simulator in 2.3c.



(a) Perlin noise [2] **(b)** Visual atom [89]

(c) CARLA Simulator [27] **(d)** Moving octopus [88]

Figure 2.3: Representative examples of synthetic datasets used in computer vision research: (a) Perlin noise [2] provides a simple procedural texture; (b) visual atom [89] represents structured, abstract patterns; (c) CARLA simulator [27] enables generation of realistic urban driving scenes; and (d) the moving octopus dataset [88] exemplifies animated, abstract shapes. The diversity of visual appearances and underlying generative processes is apparent.

2.2.2 Advantages and Disadvantages of Synthetic Data

The use of synthetic data for training deep learning models presents several compelling advantages:

- **Cost-Effectiveness and Scalability:** Generating synthetic data is generally far less expensive and time-consuming than collecting and annotating real-world data. It allows for the creation of arbitrarily large datasets on demand [11].
- **Ideal Annotation and Ground Truth:** Synthetic data inherently comes with precise and exhaustive labels (e.g., object poses, semantic masks, action classes, depth information), eliminating human annotation errors, subjectiveness and ambiguities. This is particularly beneficial for complex tasks like action recognition where spatio-temporal labeling is challenging.
- **Diversity and Control:** Synthetic environments allow for precise control over scene parameters, lighting, viewpoints, object properties and action variations. This enables the generation of highly diverse datasets that can cover rare scenarios or edge cases that are difficult to capture in the real world. It also helps in creating balanced datasets, mitigating biases present in real data.
- **Privacy Compliance:** Since synthetic data does not depict real individuals, it inherently avoids

privacy concerns and compliance issues (e.g., GDPR [30]), making it suitable for sensitive applications [67].

- **Reproducibility:** Most of the time, the generation process is deterministic (given the same parameters and fixed random seeds), ensuring experiments can be perfectly reproduced.

However, synthetic data also comes with inherent disadvantages and challenges:

- **Domain Gap:** The most significant challenge is the ”domain gap” or ”reality gap”: the difference between synthetic and real data. Models trained solely on synthetic data may not generalize well to real-world scenarios due to differences in visual fidelity, texture, lighting, or physical properties [92].
- **Fidelity and Realism:** Achieving high visual fidelity and realism in synthetic data can be computationally intensive and requires sophisticated rendering techniques. Moreover, if synthetic data derived from simulations of real-world environments lacks adequate visual fidelity, this can further widen the domain gap.
- **Complexity of Generation:** While cost-effective in the long run, setting up robust synthetic data generation pipelines, especially for complex scenarios like human actions, can be technically challenging and require upfront labor to create such a pipeline [27].
- **Representativeness:** Ensuring that the synthetic data truly represents the variability and distribution of the target real-world domain is crucial. If the synthetic data is too simplistic or does not capture the underlying complexities, the benefits may be limited [90].

Despite these challenges, ongoing work in domain adaptation and synthetic-to-real transfer for action recognition seeks to align simulated motion distributions with real-world video dynamics, making synthetic sequences an increasingly powerful asset for training robust spatio-temporal deep learning models. This sets the stage for exploring novel sources of synthetic data, such as fractals, which offer unique properties that may help address some of these challenges. [104]

2.3 Fractals

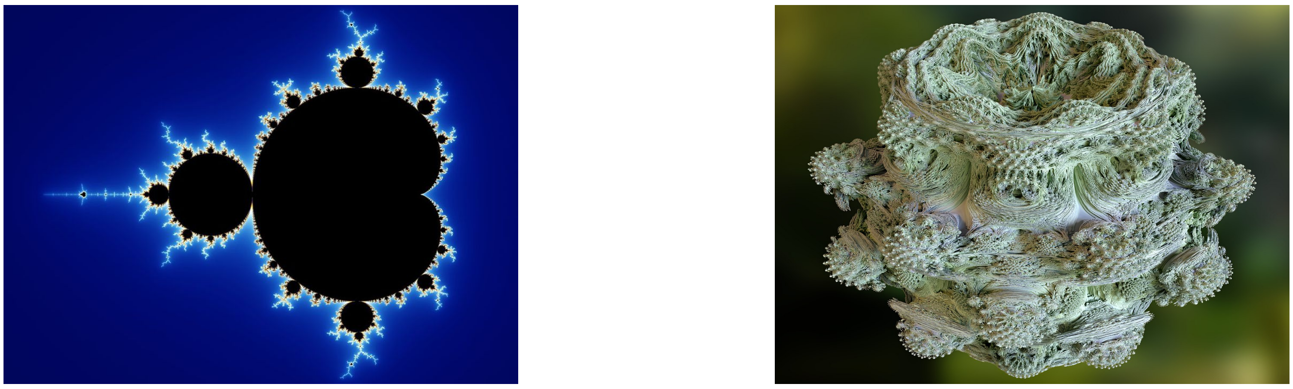
Beyond conventional synthetic data generation techniques, this thesis places particular emphasis on the utility of fractal-based synthetic data. Fractals, in mathematical terms, are sets that exhibit self-similarity across different scales of magnification. This means that a small part of the fractal, when magnified, can resemble the larger structure. Often, they are characterized by a Hausdorff dimension that is greater than their topological dimension, a concept that quantifies their complexity and space-filling properties. [31]
The inherent visual complexity of fractal geometry, manifested in intricate edges, diverse shapes and nuanced contours, is particularly valuable for pre-training computer vision models. These rich visual features can equip models to better discriminate fine-grained details and improve their generalization

capabilities across varied visual contexts. [33]

Fractals are exceptionally well-suited for generating large-scale datasets due to their algorithmic nature. They can be generated through simple iterative rules or recursive functions, such as Iterated Function System (IFS) [10], allowing for the creation of virtually infinite variations with unique details yet consistent underlying structures. The computational efficiency of this generation process, requiring relatively low computation power, makes fractals a highly promising candidate for scalable synthetic data generation. The characteristic property of self-similarity can be highly beneficial, potentially enabling models to learn features that exhibit inherent invariance to scale and resolution, qualities often pursued in robust vision systems [8].

Furthermore, many natural phenomena, from coastlines and mountains to biological structures like ferns or snowflakes, exhibit fractal-like characteristics [23]. Consequently, synthetic data derived from fractals can possess an organic visual complexity that may more closely approximate certain aspects of real-world scenes compared to simpler procedurally generated patterns. This ability to emulate natural visual textures and structures could play a role in mitigating the ”domain gap” between synthetic and real data, thereby enhancing the transferability of learned features to practical applications [21, 90].

While 2D fractals, such as the well-known Mandelbrot or Julia sets [28], offer significant visual complexity on a plane, their utility for representing the three-dimensional world is inherently limited. They primarily provide intricate boundaries and textural details but lack the dimension of depth. This thesis extends its focus to the generation of 3D fractals. This progression is pivotal because 3D fractals allow for the creation of volumetric and spatially rich synthetic environments. Such environments provide a more comprehensive representation of real-world objects and scenes, which are fundamentally three-dimensional. [98]



(a) 2D Mandelbrot Set [12]

(b) 3D Mandelbulb (ray-marched) [51]

Figure 2.4: Comparison between 2D and 3D fractals.

As illustrated in Figure 2.4, the transition from the 2D Mandelbrot set to the 3D Mandelbulb demonstrates the increased complexity and spatial depth achievable with three-dimensional fractals.

The advantages of 3D fractals become particularly apparent for tasks like action recognition. Actions unfold in three-dimensional space, often involve interactions with 3D objects, and are perceived from varying viewpoints. By leveraging fractal geometry in three dimensions, we can generate datasets that feature not only intricate surface details but also complex internal structures and authentic spatial relationships. Pre-training on dynamic sequences derived from 3D fractals can therefore expose a model to more relevant geometric and spatio-temporal cues, such as changes in apparent shape due to rotation in depth or the movement of spatially distinct components within a 3D coordinate frame. This approach provides cues that are vital for learning depth perception and understanding multi-view geometry. Although the computational demands for generating and rendering 3D fractal sequences are typically higher than for their 2D counterparts due to higher number of parameters needed to sample, the anticipated benefit is the cultivation of more robust and transferable foundational representations that help interpret more complex, real-world visual dynamics. For these reasons, 3D fractals present a compelling path for the pre-training strategies explored in this thesis.

2.4 Pre-training Techniques in Deep Learning

The remarkable success of deep learning in computer vision and related fields has been strongly driven by the adoption of pre-training strategies. **Pre-training** refers to the process of initializing a neural network by training it on a large, often generic dataset before adapting (fine-tuning) it to a more specific target task or a commonly smaller dataset. This approach leverages knowledge learned in the pre-training phase, providing a set of robust initial weights and representations that can accelerate convergence and enhance performance in downstream tasks. The broader framework that utilizes this paradigm is commonly known as transfer learning [73, 101].

2.4.1 Transfer Learning: Concept and Motivation

Transfer learning addresses the common problem of insufficient labeled data for a target task by utilizing representations learned from related data-rich domains. For example, models pre-trained on the large-scale ImageNet dataset [22] have shown substantial improvements when fine-tuned for a wide variety of computer vision problems, including image classification, object detection and semantic segmentation [57]. In the context of video understanding and action recognition, similar transfer learning strategies have been employed, with pre-training often conducted on large video datasets such as Kinetics [54]. The fundamental hypothesis underpinning transfer learning is that the features learned from one (often broader or synthetic) domain can generalize to different but related target domains, particularly when both domains share low-level or high-level statistical regularities.

2.4.2 Supervised, Self-supervised and Formula-Driven Pre-training Paradigms

Pre-training can be broadly categorized into supervised and self-supervised paradigms, each with distinct methodologies and trade-offs.

Supervised Pre-training. In supervised pre-training, the model is first trained to perform a specific task (e.g., classification) using large datasets with explicit labels. Notable examples include training on ImageNet or COCO [64] for images or on Kinetics or HMDB51 [59] for videos. Supervised pre-training tends to produce highly discriminative feature representations relevant to the annotated task [44, 16]. When transferred to a downstream task, these representations can accelerate learning and improve generalization, especially when the downstream dataset is limited in size or diversity. However, this approach requires large-scale, manually labeled datasets, which are costly and sometimes impractical to obtain. Additionally, models trained in this manner may encode biases present in the original labeled data, potentially hindering generalization to different domains [49].

Self-supervised Pre-training. Self-supervised Learning (SSL) bypasses the need for manual labels by designing pretext tasks where the supervision signal is derived from the inherent structure of the data itself. In computer vision, popular self-supervised tasks include image colorization [102], context prediction [26] and, more recently, contrastive learning [17, 45], where the model learns to distinguish between different views or augmentations of the same data sample. In video, self-supervised methods often exploit temporal order prediction, future frame generation, or contrastive tasks across video clips [68, 41]. SSL methods can utilize vast amounts of unlabeled data, making them attractive for scenarios where labeled data is scarce. However, the effectiveness of SSL depends strongly on the design of the pretext task; poorly chosen tasks can yield representations that are not useful for downstream applications.

2.4.3 Formula-Driven Supervised Learning (FDSL)

An emerging paradigm in pre-training is **Formula-Driven Supervised Learning (FDSL)**, which leverages mathematically generated synthetic data paired with automatically derived labels. This approach automates the dataset creation process, eliminating the need for manual annotation and addressing issues related to data scarcity, privacy and ethical concerns [40].
FDSL involves generating synthetic data (e.g. images, point clouds) using mathematical formulas, such as fractals and assigning labels based on the parameters used in the generation process. For instance, in FractalDB [52], images are synthesized using Iterated Function System and labels correspond to the specific formula parameters [70]. This method enables the creation of large-scale, diverse datasets without manual labeling, facilitating efficient pre-training of deep learning models.
Recent studies have demonstrated that models pre-trained using FDSL can achieve competitive performance on downstream tasks. For example, pre-training vision transformers on synthetic datasets like FractalDB has yielded results comparable to, or even surpassing those obtained with traditional

supervised pre-training on datasets like ImageNet [70]. Additionally, FDSL has been applied to audio domains, where synthetic patterns are used for pre-training audio encoders, achieving performance comparable to models pre-trained on large-scale real audio datasets [47].

2.4.4 Comparative Discussion of Pre-training Strategies

The choice between supervised, self-supervised, and formula-driven supervised pre-training strategies depends on various factors, including the availability of labeled data, computational resources, and the specific requirements of the downstream task. Supervised pre-training remains the standard when suitable large-scale labeled datasets are available and when the downstream task is closely related to the pre-training labels. Self-supervised approaches are increasingly adopted in domains where unlabeled data is abundant and manual annotation is impractical. FDSL offers a promising alternative by providing a means to generate large-scale, labeled datasets synthetically, thus circumventing the challenges associated with data collection and annotation.

In the specific context of action recognition, recent works have demonstrated that self-supervised pre-training on either real or synthetic video sequences can yield strong performance [48, 79]. Furthermore, the use of synthetic data, such as procedurally generated geometric patterns or, as explored in this thesis, three-dimensional fractals, opens new ways for pre-training without the risks and limitations associated with real video datasets. Such approaches are particularly attractive for addressing data scarcity, privacy, and generalization issues, provided that the synthetic data adequately captures relevant visual and temporal properties for transfer learning.

Overall, pre-training remains a cornerstone technique in deep learning for vision, enabling efficient knowledge transfer, improving robustness, and reducing reliance on scarce annotated data in challenging tasks such as action recognition.

2.5 Related Work

This section reviews key contributions in the domain of synthetic data and abstract pattern-based pre-training, highlighting advancements in fractal-based pre-training, extensions to video and 3D data and alternative synthetic approaches. The exploration of such methods has gained significant traction, offering promising alternatives to the traditional reliance on large-scale natural image datasets.

2.5.1 Fractal-Based Pre-training

The concept of Formula-Driven Supervised Learning was introduced by Kataoka et al. [52]. This approach leverages mathematically generated fractal images to pre-train CNNs, demonstrating that models can achieve competitive performance without exposure to natural images.

Building upon this Anderson and Farrell proposed enhancements in [7] by introducing heuristics to ensure that the sampled fractals were not degenerate (e.g. too constrained or sparse). Their work emphasizes the potential of fractal-based datasets in reducing the dependency on large, labeled image collections.

Further pushing the boundaries, Nakamura et al. [70], explored the efficacy of pre-training with extremely limited synthetic data. Remarkably, they demonstrate that even a single fractal image, when subjected to perturbations, can suffice for effective pre-training, challenging conventional notions of dataset scale requirements.

2.5.2 Extensions to Video and 3D Data

The application of fractal-based pre-training has been extended to the video domain by Svyezhentsev et al. [88]. They generate synthetic video clips by interpolating between two 2D fractals and applying video effects such as camera shake and zoom, as illustrated in Figure 2.5. Their findings indicate that such pre-training can rival traditional methods on benchmarks like HMDB51 and UCF101 [86].

In the realm of 3D data, Yamada et al. [99] applied FDSL to generate synthetic point cloud datasets. This method facilitates the pre-training of models for 3D object recognition tasks, reducing the need for extensive real-world 3D data collection.

Expanding on this, the same group [97] proposed integration of synthetic images and point clouds to train unified transformers capable of handling both modalities. This cross-modal approach underscores the versatility of formula-driven synthetic data in multi-modal learning scenarios.

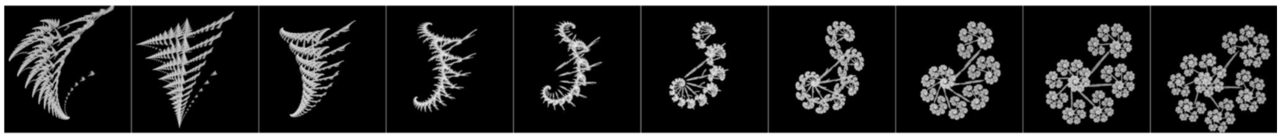


Figure 2.5: Decomposed interpolation between two 2D fractals, as proposed by Svyezhentsev et al. [88]

2.5.3 Alternative Synthetic Approaches

Beyond fractals, researchers have explored other synthetic patterns for pre-training. Takashima et al. [89] utilized sinusoidal wave patterns to train vision transformers. Their work highlights the effectiveness of structured, non-natural patterns in model pre-training.

Similarly, Kataoka et al. [53] demonstrated that models pre-trained on auto-generated contour images can achieve performance comparable to those trained on large-scale natural image datasets. This approach offers a promising method for reducing reliance on labeled real-world data.

In [39], researchers explore pre-training with 3D mesh renders to instill a strong 3D inductive bias in models. This technique aims to enhance generalization capabilities in tasks requiring spatial understanding.

These studies collectively underscore the potential of synthetic and abstract pattern-based data in pre-training deep learning models. By leveraging mathematically generated structures, researchers can mitigate challenges associated with data collection, labeling and privacy, paving the way for more efficient and accessible model training methodologies.

3 Problem Statement and Research Objectives

The need for large, diverse, and high-quality video datasets remains a significant bottleneck in the development of robust action recognition models. While Chapter 2 detailed the evolution of deep learning architectures and the challenges inherent in collecting real-world data, this chapter formulates the core research problem addressed in the thesis, building directly upon the conceptual foundations, prior work, and gaps identified earlier.

3.1 Motivation and Problem Statement

As established in Chapter 2, collecting and curating extensive, annotated video datasets for action recognition is resource-intensive, expensive, and fraught with privacy and bias concerns [50, 11]. Traditional approaches relying on large-scale real-world data are not only costly but also potentially limited by annotation errors and domain-specific biases [49, 90].

Pre-training on large generic datasets, such as ImageNet or Kinetics, has proven effective for transfer learning and model initialization [73, 101]. However, these methods largely adhere to the core limitations of manual annotation and real-world data dependence [22, 54].

Chapters 2 and 2.4.3 have discussed the emerging paradigm of *formula-driven supervision* (FDSL), which leverages programmatically generated data with labels that are intrinsically tied to the generation process. The use of FDSL enables perfectly aligned, error-free supervision at scale, bypassing the annotation bottleneck entirely [52, 40]. Despite its success in domains such as image classification or object detection [98, 7, 52], there remains a clear lack of research on applying FDSL and synthetic data generation to video-based action recognition; an important gap this thesis seeks to address.

Fractal-based datasets represent a particularly attractive source for FDSL due to their infinite variability, algorithmic complexity, and capacity for rich structural diversity [10, 31]. Importantly, each sample is automatically labeled according to its underlying generative parameters. Compared to synthetic data produced by other means (such as 2D fractal interpolations [88]), 3D fractal data inherently incorporates depth and more realistic spatial relationships, potentially yielding richer visual features for model pre-training. Prior works focused on interpolating 2D fractals may be limited by their lack of depth, spatial expressiveness, and applicability to three-dimensional perception tasks.

3.2 Research Questions and Objectives

This thesis seeks to answer the following central research question:

> *Can pre-training on synthetic datasets generated from 3D fractals enhance the downstream performance of action recognition models, compared to conventional pre-training datasets?*

To address this question, the following objectives are pursued:

- Develop a pipeline for algorithmically generating diverse 3D fractal-based synthetic datasets, including point clouds and video sequences, with a range of geometric and temporal transformations.
- Pre-train widely adopted and empirically validated action recognition architectures (such as ResNet50 [44] equipped with a Temporal Shift Module [63]) on these fractal-derived datasets using FDSL principles.
- Benchmark the resulting models against counterparts pre-trained on standard real-world datasets (e.g., ImageNet, Kinetics) and models trained from scratch, utilizing widely-used benchmarks such as HMDB51 [59] and UCF101 [86].
- Analyze the learned representations in terms of generalizability, robustness, and transferability, identifying both the strengths and limitations of fractal-based pre-training.

This structured approach aims to provide a rigorous assessment of the viability and advantages of leveraging 3D fractal-based synthetic data in Deep Learning for action recognition.

4 Technical Foundations

In this chapter, the key principles underlying 3D fractal generation are introduced. The formulation of Iterated Function Systems (IFS) and the associated Chaos Game theory are first presented. Subsequently, the parameters that control fractal shape are detailed, followed by an overview of multiple dataset representations for 3D fractals and the motivation behind the chosen representation. Lastly, the notion of class actions is then discussed.

4.1 Principles of 3D Fractal Generation

Fractals are complex geometric structures exhibiting self-similarity across different scales. The generation of 3D fractals extends the concept of 2D fractals into three-dimensional space, allowing for more intricate and realistic models. This section delves into the mathematical foundations and algorithms used for creating 3D fractals.

4.1.1 Iterated Function Systems (IFS)

An Iterated Function System (IFS) is a foundational mathematical framework used to construct self-similar fractals. At its core, an IFS consists of a finite set of contraction mappings on a complete metric space (X,d). Specifically, let $\{f_i : X \to X \mid i = 1,2,\ldots,N\}$ be a collection of contraction mappings, where each mapping f_i satisfies

$$d(f_i(x), f_i(y)) \leq \lambda_i \cdot d(x,y) \quad \text{for all } x,y \in X,$$

with $\lambda_i \in [0,1)$ denoting the contraction constant for f_i [10]. The requirement that $\lambda_i < 1$ ensures that each function brings points closer together, a key property of contraction mapping. This property guarantees, via Banach's Fixed Point Theorem [56], that every contraction has a unique fixed point.

The collective action of all such functions in the IFS can be described using the *Hutchinson operator*, introduced by Hutchinson [46]. The operator acts on the space of non-empty compact subsets of X, denoted $\mathscr{K}(X)$, and is defined as

$$\mathscr{F}(A) = \bigcup_{i=1}^{N} f_i(A), \quad \text{for any } A \in \mathscr{K}(X).$$

Importantly, the Hutchinson operator is itself a contraction with respect to the Hausdorff metric H [43], which quantifies the distance between sets and is defined as

$$H(A,B) = \max\left\{\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right\},$$

where sup denotes the supremum operator and inf the infimum operator [80].

A central concept in IFS theory is the *attractor*: a non-empty compact set $A^* \subseteq X$ that satisfies $\mathscr{F}(A^*) = A^*$. Hutchinson's theorem ensures that, starting from any non-empty compact set A_0, the sequence defined recursively by $A_{n+1} = \mathscr{F}(A_n)$ converges to this unique attractor in the Hausdorff metric:

$$\lim_{n\to\infty} H(A_n, A^*) = 0.$$

In practice, this iterative process generates increasingly precise approximations of a fractal, the attractor, by repeatedly applying the set of contraction mappings. Each iteration constructs the next level of self-similarity, with every function f_i responsible for producing a scaled-down copy of the current structure. Thus, IFSs naturally produce self-similar fractal objects, which is the mathematical essence behind many natural and synthetic fractal shapes.
The mathematical framework of IFS provides a reliable way to construct detailed, self-similar patterns in both two and three dimensions. Thanks to their versatility, IFS-based methods can be adapted to a wide range of uses, from procedural graphics to generating synthetic datasets, making them especially useful for applications like action recognition explored in this thesis.

4.1.2 Chaos Game Algorithm

The Chaos Game is a stochastic method to generate fractals, particularly useful for visualizing the attractor of an IFS [10]. Starting from an arbitrary point $x_0 \in \mathbb{R}^n$, the algorithm iteratively applies randomly selected functions from the IFS:

$$x_{k+1} = f_{i_k}(x_k) \tag{4.1}$$

where i_k is chosen randomly from $\{1, 2, ..., N\}$ at each iteration. The entire sequence of generated points, denotes as $\{x_k\}_{k\geq 0}$ or $x_0, x_1, x_2, ...$, converges to the fractal attractor.
In practical programming implementations, the Chaos Game is executed over a large number of iterations, with each computed point plotted to reveal the fractal pattern. Additionally, each function f_i can be selected according to a set of probability weights, allowing for the adjustment of point density in different regions of the resulting fractal [10, 7].

4.2 Key Parameters Controlling Fractal Shape

The construction of 3D fractals using IFSs relies on specifying the affine transformation for each function f_i. Each transformation is typically parameterized by a 3×3 real matrix A and a translation vector $b \in \mathbb{R}^3$. In the simplest case, these parameters (matrix entries and vector components) are sampled independently from a uniform distribution over a predetermined range (e.g., $[-1, 1]$). This naive approach allows for fast random exploration of the transformation space and, by extension, a wide variety of fractal shapes.

However, purely uniform sampling can often result in a high proportion of transformation matrices that do not yield contractive or visually interesting fractals. Many sampled matrices may not satisfy the contraction condition required for convergence, or may lead to degenerate and unbalanced attractors.
To address these issues and gain better control over the generated fractal geometries, matrix decomposition techniques, most notably the Singular Value Decomposition (SVD), are employed in the parameterization and filtering of affine transformations.
SVD is a primary concept in analyzing and generating affine transformations for fractal geometry. For any real $m \times n$ matrix A, SVD provides the factorization:

$$A = U\Sigma V^T \tag{4.2}$$

where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and Σ is an $m \times n$ diagonal matrix whose non-negative entries σ_i are the singular values of A [87].
Geometrically, this decomposition interprets any affine transformation as a sequence of operations: an initial rotation (by V^T), followed by axis-aligned scaling (by Σ), and a final rotation (by U). This view is especially useful in the context of Iterated Function Systems, where each function is an *affine contraction mapping* and its effect on the fractal's shape can be directly understood via the scaling and rotation parameters influenced by the SVD.

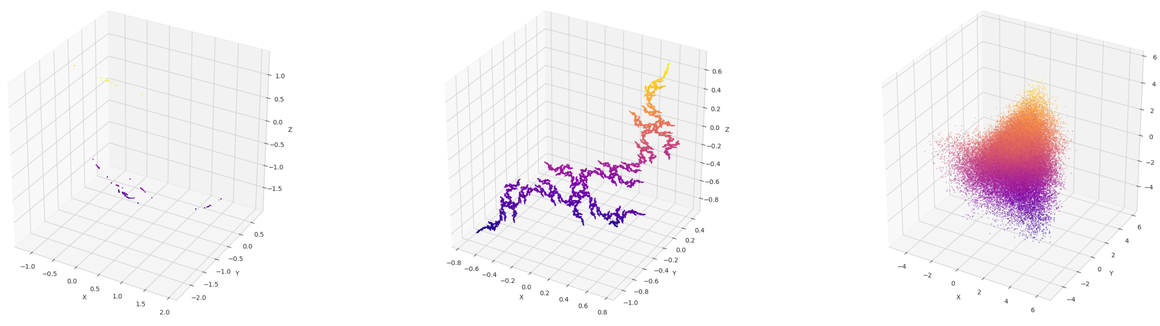
In order for an affine map to be contractive, a requirement for the existence of a unique fractal attractor, the largest singular value $\sigma_{\max}$ of the transformation matrix must satisfy $\sigma_{\max} < 1$ [46]. This ensures that all points are brought closer together under the transformation, guaranteeing convergence of the iterated process.

Traditionally, random sampling of IFS parameters is followed by rejecting those systems that are not contractive. This approach can be inefficient, as a large fraction of randomly sampled transformations may not satisfy the contraction condition. By instead sampling the singular values directly within the interval $(0, 1)$ and then constructing the transformation matrix, one ensures contractivity by design. This method both increases the efficiency of the sampling process and allows for better geometric control, directly influencing properties such as anisotropy and the overall complexity of the generated fractals.

Earlier research on 2D fractals has shown that the sum of the singular values associated with the linear components of each affine transformation, commonly referred to as the σ-factor, exhibits a strong empirical correlation with the appearance of fractal sets that are both visually complex and structurally connected [7]. In particular, Anderson and Farrell demonstrated that constraining the σ-factor within a specific range results in a significantly higher proportion of fractals exhibiting appealing and non-degenerate geometry. Conversely, when the sum of singular values is too low, the attractor tends to be overly contractive resulting in a collapse and lack of detail; when too high, the structure can become excessively expansive and shapeless. Thus, principled selection and control of transformation parameters, including the number of component functions and the σ-factor play a decisive role in ensuring diversity and visual richness within generated fractal datasets.

Directly sampling or filtering affine transformation parameters based on singular values (and derived measures such as the σ-factor) provides a mathematically principled and efficient method for producing fractals with desirable structural characteristics.

Although the majority of this analysis has focused on the geometry of 2D fractals, the underlying principles have an inherent extension to higher-dimensional systems. Figure 4.1 provides a 3D visualization of the qualitative effects of key parameters: it contrasts examples of fractals that are overly contractive, well-balanced, and too dense in three dimensions. This visualization serves to highlight how parameter selection in 3D IFS generation directly influences the spatial organization and visual complexity of the resulting attractors.



(a) Overly contractive **(b)** Well-balanced **(c)** Too dense

Figure 4.1: Qualitative examples of 3D fractals generated with different parameterizations: (a) overly contractive, (b) well-balanced, and (c) too dense.

In addition to transformation matrices, each affine function in an IFS is assigned a sampling probability p_i. While these probabilities do not alter the overall shape of the attractor, they do control the frequency with which points in different regions of the attractor are visited and rendered during the Chaos Game algorithm. Empirically, setting p_i proportional to the magnitude of the determinant of the linear part of the transformation, i.e., $p_i \propto |\det A_i|$, ensures more uniform coverage of the attractor, especially when some transformations are much ”larger” than others [7]. In contrast, uniform probabilities may result in under-representation of regions associated with transformations of larger determinant, producing artifacts or ”gaps” in the rendered fractal. Example of different probability sampling methods is depicted in Figure 4.2.

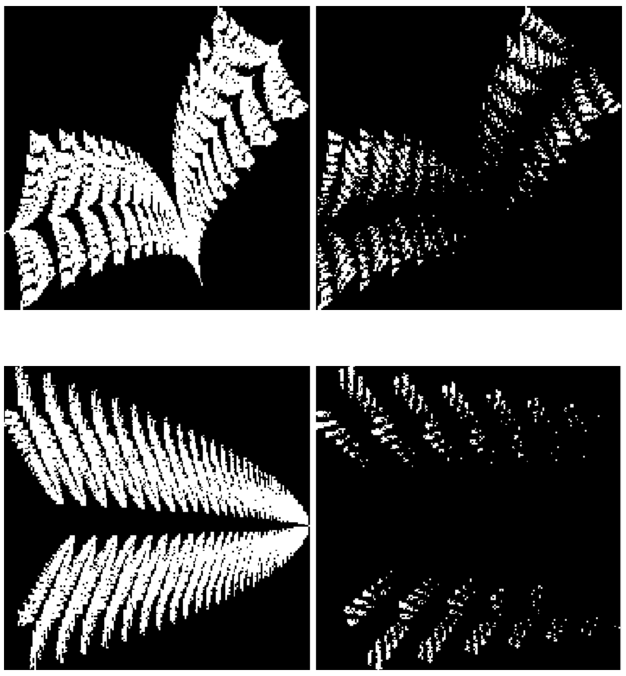


Figure 4.2: Comparison of probability sampling strategies for fractal generation in 2D, as illustrated in [7]. Left: determinant-based sampling, where the probability of selecting each affine transformation is proportional to the absolute value of its determinant; right: uniform probability sampling. The choice of sampling method has a visible effect on the final appearance of the fractals.

4.3 Dataset Representation from 3D Fractals

The process of transforming 3D fractal structures into datasets suitable for action recognition tasks involves careful consideration of data modality, temporal encoding, and compatibility with modern neural architectures. This section discusses common representation strategies in the literature and the reasoning behind the adopted approach in this thesis.

4.3.1 Types of Representations for 3D Fractal Data

Multiple modalities are available for representing 3D fractal structures in a form suitable for machine learning:

- **Point Clouds:** A direct encoding of the fractal as a set of 3D points, commonly used in geometric deep learning and 3D object recognition [99, 97]. While point clouds retain full geometric information, they pose challenges for action recognition, particularly regarding the design of temporally-aware neural architectures and the lack of standardized spatial alignment.
- **Voxel Grids and Mesh Sequences:** Voxelization discretizes space into a grid, providing a volumetric representation that is naturally compatible with 3D convolutional networks, though it is often computationally intensive and may lose fine detail at practical resolutions.
- **Image Sequences (Rendered Frames):** Projecting the evolving 3D fractal onto 2D planes over time allows the use of well-established video recognition architectures. This is the dominant

paradigm in video-based action recognition [9, 95, 91] and has also been successfully applied to synthetic datasets [88]. Image-based representations leverage the efficiency and flexibility of convolutional or transformer models.

- **Parameter Trajectories:** Sequences of parameter vectors (e.g., transformation matrices over time) provide a compact and interpretable modality, but are less commonly used directly as input for end-to-end visual models [15].

In recent formula-supervised and fractal-based pre-training studies, point patch embeddings, image patch embeddings, and their combinations have been explored as input modalities for vision transformers [97]. However, to the best of my knowledge, the explicit construction of evolving 3D fractal sequences for action recognition remains largely unexplored.

4.3.2 Key Design Considerations

Effective dataset representation for action recognition depends on several factors:

- **Temporal Coherence:** Effective action recognition relies on the ability to capture motion and temporal dependencies. Raw data representations that preserve the continuity and ordering of transformations or states over time allow models to learn meaningful dynamics, analogous to those found in real-world video or motion capture data [9, 91, 81].
- **Spatial Fidelity:** The granularity and geometric richness of the underlying fractal structure, whether encoded as dense point clouds, high-resolution images or detailed mesh sequences, influence the capacity of downstream models to distinguish subtle features and patterns.
- **Consistency and Diversity:** A useful dataset should balance class consistency (similarity among instances within a class) with sufficient intra-class diversity. For fractal-based datasets, this can be controlled through the selection of transformation parameters, initial states, or random seeds, ensuring a broad yet coherent sampling of possible motions and structures.
- **Annotation and Labeling:** Synthetic fractal datasets benefit from automatic annotation, as class or action labels can be programmatically defined by the generative process. This enables scalable creation of large datasets without manual labeling effort.
- **Computational Tractability:** The efficiency of generating and rendering fractal data is a practical consideration. Chosen representations should facilitate the creation of large-scale datasets in a computationally reasonable manner, enabling extensive experimentation and model training [7, 88].

These considerations collectively shape the suitability of a representation for learning temporally and spatially complex actions from synthetic fractal data.

4.3.3 Motivation for the Adopted Representation

Selecting an appropriate dataset representation is crucial for enabling effective learning of spatio-temporal patterns in synthetic action recognition tasks. Among the various modalities available for

3D fractal data, such as raw point clouds, mesh sequences, parameter trajectories or volumetric grids, rendered image sequences have distinct advantages.
Firstly, projecting evolving 3D fractals onto 2D frames produces data closely aligned with the input format of most proven action recognition architectures, including convolutional and transformer-based models [9, 91, 95]. This alignment enables direct comparison with established benchmarks and leverages the strengths of well-tested computer vision pipelines.
Secondly, rendered image sequences efficiently capture both spatial detail and temporal dynamics, preserving essential geometric cues and motion patterns while remaining computationally tractable. The use of fixed or systematically varied viewpoints ensures consistency across samples, while transformation-driven changes encode action-like temporal coherence.
Other modalities, such as direct point cloud sequences or mesh-based representations, offer higher geometric fidelity but often require specialized neural architectures (e.g., point-based or graph neural networks) and introduce challenges in standardizing input formats. Meanwhile, parameter trajectories are compact but may not provide the visual richness or interpretability needed for effective action recognition in typical settings.
Thus, representing fractal actions as sequences of 2D rendered frames strikes a practical balance between information richness, visual interpretability and compatibility with widely used Deep Learning models. This approach also facilitates systematic study and fair ”head-to-head” comparison with prior work in synthetic action recognition, particularly studies based on 2D fracals [88, 7].

4.4 Encoding Actions within Synthetic Fractal Data

For synthetic datasets to be useful in action recognition, it is necessary to define and encode temporally coherent ”actions” that neural networks can learn to discriminate. While natural video datasets rely on semantic human or object activities, synthetic fractal data requires designing meaningful analogues to such actions using transformations and parameterized dynamics.
A general strategy involves selecting a base fractal structure (e.g., the attractor of a particular 3D IFS) and applying systematic geometric or parametric transformations over time to create sequences. These temporal sequences can simulate a variety of motion or deformation patterns, including but not limited to:

- **Affine Transformations:** Rotations, translations, scaling, and shearing of the entire fractal point cloud or its localized components.
- **Parameter Animation:** Smoothly varying the parameters of the underlying fractal-generating functions, such as altering contraction ratios or rotation angles in the IFS.
- **Camera Movements:** Changing the viewpoint or projection direction over time, emulating camera motion as in natural video.
- **Compositional Transformations:** Combining or morphing between different fractal attractors, or overlaying noise and perturbations to simulate complex dynamics.

- **Simulation-based Rendering:** Leveraging physics engines or virtual environments to generate video sequences where motion is governed by real-world physical laws.

Each unique transformation sequence or family of parameter evolutions can be associated with a distinct action class, allowing for systematic dataset construction. The temporal coherence of these sequences is crucial: frames must evolve gradually, enabling models to learn temporal dependencies analogous to those found in real-world action videos [88].
This paradigm of defining synthetic ”actions” through controlled transformations is supported by previous work in procedural action recognition datasets, which have used parameterized shape morphing or interpolations between pattern templates to generate temporal data [88, 7, 89]. Notably, these approaches demonstrate that, even in the absence of semantic human motion, carefully designed transformations can produce class-separable and learnable temporal structures.
In this thesis, these principles guide the definition of action classes within the fractal dataset. Specifically, sequences are generated by applying a variety of affine transformations, such as rotation, translation and shearing, to 3D fractal point clouds over time. While prior literature explores a wider range of transformation paradigms, including parameter morphing and compositional augmentations [88, 7, 89], this work focuses on affine geometric transformations due to their interpretability, implementation efficiency, and proven effectiveness in creating temporally coherent, class-separable action sequences. The precise implementation details and labeling protocols are presented in the Methodology Chapter 5.

5 Methodology

This chapter provides a detailed protocol of the procedures employed to construct the 3D fractal-based synthetic video dataset and to train neural networks for action recognition. The methodology encompasses the sampling of fractal parameters, the generation and preprocessing of point clouds, the synthesis of video sequences via temporal transformations, dataset organization and the training of action recognition models. All implementations were conducted in Python utilizing open-source libraries including Open3D [105], NumPy [42] and PyTorch [75], with scripts and configuration files made for full reproducibility.

5.1 3D Fractal Generation

The generation of 3D fractals for this thesis follows a carefully designed process, drawing from established mathematical foundations and recent developments in fractal dataset creation. This section provides an overview of the core methods used to sample parameters, generate fractal instances, and ensure dataset quality. The approach is structured to facilitate large-scale, reproducible, and geometrically diverse datasets suitable for training deep neural networks in action recognition tasks.

5.1.1 Parameter Sampling and IFS Construction

Building on the mathematical foundations of fractals described in Section 4.1, the generation of a single fractal instance follows a systematic process grounded in Iterated Function Systems. To generate each point in a fractal, the following elements are required: (i) an initial coordinate, typically set to the origin $(0,0,0)$; (ii) a set of N affine transformation systems, each comprising a transformation matrix A and a translation vector b. The Chaos Game algorithm is then employed to iteratively generate the sequence $x_{i+1} = A \cdot x_i + b$, where $x_i, x_{i+1} \in \mathbb{R}^3$.

The dataset construction commences by generating a large collection of such 3D fractal point clouds, with the following procedure applied for each class:

```
Randomly select the **system size** $N \in [2,8]$ (i.e., the number of affine functions).
For each function $k$ from 1 to $N$:
    Sample 12 parameters from a Uniform distribution $U(-1,1)$.
    Matrix $A_k$ = first 9 parameters reshaped to $3 \times 3$.
    Vector $b_k$ = last 3 parameters.
For each matrix $A_k$:
    Calculate $d_k = |\det(A_k)|$.
    Assign sampling probability $p_k$ based on $d_k$ (as discussed in Section 4.2).
Accumulate unnormalized probabilities ($d_k$) from all $N$ systems.
Normalize all probabilities such that $\sum_{k=1}^{N} p_k = 1$.
```

Listing 5.1: Algorithm for Iterated Function System (IFS) Parameter Sampling

This parameter sampling process is repeated C times, where C corresponds to the total number of classes required for training. Each class is represented as a separate `.csv` file containing the transformation parameters and their associated probability as 13 entries per system.

5.1.2 Point Cloud Generation and Filtering

The previously saved `.csv` parameter files are subsequently processed to generate the fractal point clouds as follows:

1. For each parameter set (i.e., each IFS defined by a system size N, with N pairs of matrix A and vector b), a point cloud of m points is generated for the fractal. Each point is produced by iteratively applying the Chaos Game algorithm: at each iteration, one of the N system functions is selected randomly according to its assigned probability p_i, and the transformation is applied to the current point to generate the next. To obtain m samples for the point cloud, m number of iterations are applied per IFS for each class.
2. The point cloud is recentered at the origin by subtracting the centroid.
3. The variance along each axis is computed and degenerate point clouds (e.g., collapsed or empty) are filtered out by requiring that the variance along each axis exceeds a threshold (variance > 0.05).
4. Only point clouds passing the variance check are saved as Open3D PointCloud objects for downstream processing.

All datasets in this thesis were generated with $m = 10\,000$ points per fractal. The combination of min-max normalization and centroid alignment is crucial for ensuring numerical stability and consistency across the dataset, particularly when applying subsequent geometric transformations. Centering each point cloud at the origin is a standard preprocessing step that simplifies downstream processing, including transformations and variance-based filtering. To maintain a high standard of geometric diversity, only point clouds whose variance along all axes exceeds a threshold (variance > 0.05) are retained. This filtering step ensures that degenerate fractals, such as collapsed or nearly empty sets, are excluded. All structurally valid fractals that pass these criteria are then stored in the `.ply` format, making them readily accessible for further processing.

During dataset creation, the initial parameter sampling strategy was adopted from Wang et al. [99], who demonstrated the utility of large-scale 3D fractal point cloud datasets for 3D object detection. However, visual inspection revealed that such naive unconstrained parameter sampling often resulted in degenerate or aesthetically suboptimal fractals. Drawing inspiration from the geometric quality criteria proposed by Anderson and Farrell [7], in which subjectively labeled 2D fractals were analyzed to identify empirical parameters of fractal quality, additional geometric constraints were considered for 3D fractal generation. The specific criteria explored, and their impact on dataset properties, are presented in the Improving Fractal Geometry Chapter 8.

5.2 Video Dataset Generation Pipeline

The process of creating a synthetic video dataset from 3D fractal point clouds involves several key stages, ranging from generating temporally coherent video sequences to organizing the resulting data for downstream trasnfer learning tasks. The main components of this pipeline are detailed below.

5.2.1 Action Definition and Temporal Transformation

The creation of each action class is achieved by applying a controlled sequence of geometric transformations to a 3D fractal point cloud, producing a temporally coherent video sequence. The core process is outlined below:

1. For each valid point cloud:
   a) The number of frames per video is randomly sampled within the range of 18 to 20 per [88].
   b) For each video, target transformation parameters are randomly selected. These include rotation angles (up to 360° per axis), shear factors (up to 0.8 per axis), translation vectors (up to 0.8 per axis), and parameters for a non-affine spatial warp. These ranges represent the defaults; alternative values have also been explored in experimental settings to assess their impact on the dataset and downstream model performance.
   c) A fixed coloring scheme is most commonly employed, mapping the z-coordinate of each point to the ”viridis” colormap from the matplotlib library. Additional coloring strategies are available such as coloring by x- or y-coordinate, distance from the center, or using random colors and their effects are compared in the experimental section.
   d) Transformation parameters are interpolated between the identity (no transformation, at frame 0) and the sampled target values (final frame), using either a linear or sinusoidal (ease-in-out) temporal profile, chosen at random for each sequence.
   e) For each frame, the interpolated transformation is applied to the entire point cloud, and the resulting point set is rendered as a 256×256 RGB image using the Open3D library, which is captured directly into an uncompressed in-memory pixel buffer to avoid intermediate saving to lossy image formats like JPEG.
2. The sequence of rendered frames is assembled into a video (`.mp4`, 12 FPS) using the `ffmpeg` toolkit, an industry standard for efficient and widely supported video encoding. The MP4 format is chosen for its broad compatibility and favorable trade-off between file size and visual quality, as a consequence of the use of lossy compression.
3. Each video is saved with a unique file name, which also encodes its corresponding class label.

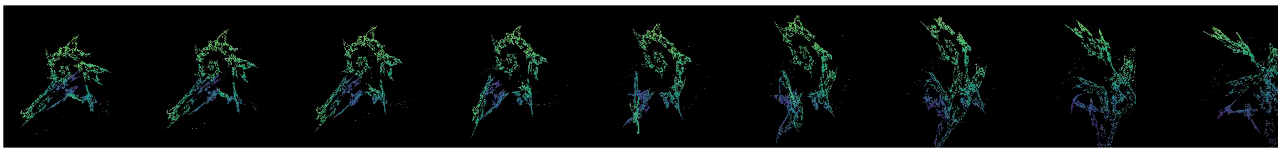


Figure 5.1: Example of a synthetic fractal video, showing every second frame as a 3D fractal undergoes a geometric transformation. Colors reflect the z-coordinate mapped to the ”viridis” colormap. Noticeable non-affine deformation caused by the spatial warp appears toward the end of the sequence.

Figure 5.1 depicts a sample sequence from the synthetic fractal dataset, with every second frame displayed. The fractal point cloud undergoes a temporally coherent geometric transformation, illustrating both affine and non-affine effects. Points are colored by their z-coordinate using the ”viridis” colormap, with more pronounced deformation visible toward the final frames.

This entire process, from fractal parameter sampling to video file export and dataset organization is summarized in Algorithm 1:

Algorithm 1 Pipeline for Generating 3D Fractal-Based Synthetic Video Dataset

```
Input  : $N_{classes}$: Total number of unique fractal classes,
         $N_{instance\_per\_class}$: Number of video sequences per class,
         $M_{points}$: Target number of points per fractal point cloud,
         $V_{thresh}$: Minimum variance threshold for point cloud validity,
         $T_{ranges}$: Parameter ranges for randomized temporal transformations,
         $L_{video\_range}$: Range for video length in frames,
         $S_{config}$: Configuration for train/validation split,
         $OutputDir$: Base directory for storing generated videos,
         $ServerCredentials$: Credentials for server transfer
Output: Synthetic video dataset generated and transferred to the server
// PHASE 1:  FRACTAL STRUCTURE AND VIDEO GENERATION
$i \leftarrow 0$ while $i < N_{classes}$ do
    $IFS_{params} \leftarrow$ SampleIFSParameters()
    $P_{raw} \leftarrow$ GenerateRawPointCloud($IFS_{params}, M_{points}$)
    $P_{processed} \leftarrow$ ProcessPointCloud($P_{raw}$) // Normalize, center
    $is\_valid\_pc \leftarrow$ ValidatePointCloud($P_{processed}, V_{thresh}$) // Variance check
    if $is\_valid\_pc$ then
        for $j \leftarrow 1$ to $N_{instance\_per\_class}$ do
            $video \leftarrow$ CreateTransformedVideo($P_{processed}, L_{video\_range}, T_{ranges}$)
            SaveVideo ($video, OutputDir$)
        end
        $i \leftarrow i + 1$
    end
end
// PHASE 2:  DATASET ORGANIZATION AND TRANSFER
$TrainFiles, ValFiles \leftarrow$ OrganizeAndSplitDataset($OutputDir, S_{config}$)
Dataset.zip $\leftarrow$ ZipDataset($TrainFiles, ValFiles$)
TransferDatasetToServer (Dataset.zip, $ServerCredentials$)
```

It is worth noting that, instead of pre-generating all video sequences, the dataset pipeline could likely be adapted for ”on-the-fly” generation during training, leveraging the just-in-time (JIT) compilation tools such as Numba [60], as demonstrated in [7]. However, implementing such dynamic data generation was not prioritized in this thesis, as the focus remained on controlled dataset analysis and reproducibility.

To expedite the processing of large numbers of point clouds, multiprocessing is leveraged throughout the pipeline. The embarrassingly parallel nature of applying transformations to independent fractal instances allows for substantial speedup, making it feasible to generate large-scale video datasets in a reasonable time-frame. To illustrate, generating one transformed fractal video instance requires approximately 100 ms of computation on a typical laptop-grade CPU. Consequently, producing a dataset of 55 000 such video instances (e.g., 50 000 for training and 5 000 for validation) corresponds to a total computational workload of roughly 1.5 hours. The implementation further facilitates future extensions, such as experimenting with alternative transformation parameters, coloring schemes, or rendering resolutions, which are discussed in later chapters.

5.2.2 Augmentation and Dataset Organization

Given the high degree of geometric and temporal diversity introduced by the randomized transformation targets during video synthesis, no additional explicit spatial or temporal augmentations (such as cropping, flipping, or noise injection) are applied at this stage. This approach ensures that the observed variability in the dataset is directly attributable to the range of transformations applied to the underlying fractal point clouds, simplifying the analysis of how such transformations affect downstream model performance.
The resulting dataset is organized in a hierarchical directory structure, where each class is represented by a separate subdirectory containing its associated video samples. This structure is compatible with common data loading utilities in deep learning frameworks and facilitates straightforward assignment of class labels based on directory names.
For evaluation purposes, the dataset is partitioned into training and validation splits by randomly selecting a fixed number of video samples from each class (e.g., 10 per class) to form the validation set, while the remaining videos are designated for training. This strategy ensures balanced representation of all classes in both splits and enables robust evaluation of model generalization. The splitting process is automated to guarantee reproducibility and to prevent data leakage between training and validation sets.

5.2.3 Dataset Properties

The pipeline detailed in this chapter allows for the generation of synthetic video datasets with a range of configurable characteristics. The specific values for these properties are adapted according to the goals of individual experiments and are therefore detailed in the Experimental Chapter 7. The primary configurable properties that define each dataset instance are outlined below:

- **Total Number of Classes:** Defines the quantity of distinct fractal transformation categories. This parameter is typically varied in experiments to explore model scalability and performance on tasks of differing complexity.
- **Instances per Class:** Specifies the number of unique video sequences generated for each class. Adjusting this property allows for studies on data efficiency, model generalization, and robustness to intra-class variation.
- **Input Frame Resolution:** The final resolution of video frames that are provided as input to the neural network models, typically after preprocessing steps such as downsampling (e.g., 112×112 pixels to match baseline requirements).
- **Video Length (Frames):** The number of individual frames that compose each video sequence (e.g., ranging from 18 to 20 frames). This, along with frame rate, determines the temporal extent of the depicted action.
- **Frame Rate:** The rate at which video frames are sampled or rendered, measured in Frames Per Second (FPS) (e.g., 12 FPS). While frame rate directly affects the perceived speed and smoothness of transformations from a human perspective, it can also have practical implications during model training and evaluation: lower frame rates may limit the temporal resolution available to the neural network, potentially impacting its ability to capture fine-grained motion cues, whereas higher frame rates increase data volume and computational load.
- **Point Cloud Density:** The number of 3D points used to construct and represent the fractal in each video frame (e.g., $10,000$ points). This property affects the visual detail and complexity of the rendered fractals.
- **Train/Validation Split Ratio:** The proportional division of the generated video samples into training and validation sets, typically performed per class to ensure balanced representation (e.g., 90% of videos for training and 10% for validation).

This framework of configurable properties provides the flexibility to systematically investigate the impact of different dataset configurations on the performance of action recognition models.

5.3 Training Data Augmentation Pipeline

To enhance model generalization and robustness, a comprehensive data augmentation pipeline is applied to all training videos. These augmentations are performed on-the-fly during training, ensuring that each batch is exposed to diverse spatial and appearance variations. This strategy mitigates overfitting and simulates real-world variability.
The augmentation pipeline consists of both spatial and appearance-based transformations, implemented using PyTorch and the `pytorchvideo` library. The key components are adapted from [88] and are as follows:

- **Random Resized Crop:** Each video is randomly cropped and resized to the target input size (typically 112×112 pixels), with the crop scale sampled from a specified range (e.g., 0.9-1.0) and aspect ratio variation. This introduces variability in spatial framing.

- **Horizontal Flip:** With configurable probability, the video frames are horizontally flipped to simulate viewpoint changes.
- **RandAugment:** RandAugment [19] is applied, randomly selecting a sequence of parameterized transformations (such as color jitter, rotation, or contrast adjustment) with user-defined magnitude and number of operations per video.
- **Gaussian Blur:** With specified probability, Gaussian blur is applied to each frame to increase robustness to focus and quality variation and simultaneously reduce noise.
- **Random Perspective and Camera Motion:** Additional transformations such as random perspective distortion and camera motion (shift, zoom, shake) are applied with specified probabilities, mimicking real-world camera effects and temporal jitter.
- **Normalization:** All input frames are normalized to match the mean and standard deviation of ImageNet images, ensuring compatibility with pre-trained models and stable optimization.

The order and parameterization of these augmentations are modular, enabling fine-grained ablation studies and adaptation to different experimental setups. Crucially, the application of each augmentation is probabilistic: each transformation (e.g., blur, shift, zoom, shake) is activated independently for each video in a batch according to a user-specified probability (e.g., `prob_blur`, `prob_shift`, etc.). This design allows for flexible control of augmentation strength and diversity, as well as clear tracking of the exact augmentation policy used in each experiment (see Chapter 6 for explicit values).

5.4 Workflow Automation and Practical Implementation

To enable systematic and fast experimentation while ensuring full reproducibility, the entire dataset generation process is modularized and automated. Each key stage, including parameter search, fractal generation, transformation, dataset splitting and packaging, is implemented as a standalone Python script that can be flexibly configured via command-line arguments and configuration files.
The orchestration of these components is handled through a master shell script, which sequences each phase of the pipeline and manages data flow between modules. This approach allows for rapid adjustment of parameters such as the number of classes, point cloud density, transformation ranges, or dataset splits, supporting both parameter sweeps and ablation studies.
This modular and script-driven workflow ensures that the entire process, from parameter search to dataset packaging, can be executed in a single command, minimizing manual intervention and risk of error. Furthermore, logs and configuration snapshots are saved for each run, allowing exact reproduction of any dataset version generated for experimentation.

After local preparation, datasets are compressed and securely transferred via SSH to a high-performance AI server (AI-LAB) [3], where model training and evaluation are performed. This separation of data generation and model training supports decoupled experimentation and efficient use of computational resources.

5.5 Neural Network Architecture: ResNet-50 and Temporal Shift Module

The backbone architecture selected for all experiments is a **ResNet-50** [44] integrated with the **Temporal Shift Module (TSM)** [63], a design that balances computational efficiency with the ability to capture temporal dependencies crucial for action recognition from video data.

5.5.1 ResNet-50 Backbone

ResNet-50 is a deep Convolutional Neural Network originally developed for large-scale image classification. It comprises 50 layers with residual connections that facilitate the training of deep networks by mitigating the vanishing gradient problem [74]. In this work, the standard 2D ResNet-50 serves as the base spatial feature extractor, processing each frame of the video sequence.

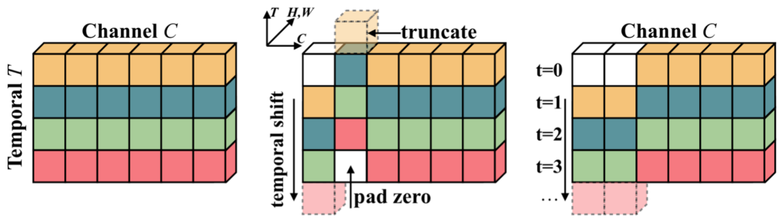
5.5.2 Temporal Shift Module (TSM)

To adapt the spatial backbone for video inputs and enable the modeling of temporal relationships across frames, the Temporal Shift Module (TSM) is inserted into each residual block of the ResNet-50 architecture. The TSM operation is a parameter-free temporal operator that shifts a fraction of feature channels along the temporal dimension, thereby enabling information exchange between adjacent frames without a significant increase in model complexity or memory usage. Specifically, for a feature map tensor of shape $[N,C,T,H,W]$ (where N is batch size, C is channel, T is time, H is height, and W is width), the operation divides the channel dimension and shifts:

$$x'_{n,c,t,h,w} = \begin{cases} x_{n,c,t-1,h,w} & \text{if } c \in C_f \\ x_{n,c,t+1,h,w} & \text{if } c \in C_b \\ x_{n,c,t,h,w} & \text{otherwise} \end{cases} \tag{5.1}$$

Here, C_f and C_b denote the sets of channels shifted forward and backward in time, respectively, while the remaining channels remain unshifted. In practice, it is common to shift one-eighth of the channels forward ($|C_f| = \frac{C}{8}$), one-eighth backward ($|C_b| = \frac{C}{8}$), and to leave the remaining three-fourths of channels unchanged. This simple yet effective mechanism enables temporal information exchange between adjacent frames with minimal computational overhead.

This operation preserves the spatial feature extraction capability of the ResNet-50 backbone while introducing temporal awareness into the network. The overall process is illustrated in Figure 5.2.



(a) The original tensor without shift. **(b)** Offline temporal shift (bi-direction). **(c)** Online temporal shift (uni-direction).

Figure 5.2: Schematic illustration of the Temporal Shift Module (TSM), as adapted from [63]. The figure shows the temporal channel shifting mechanism, where a subset of feature channels is shifted forward and backward in time, while the remainder are unaltered. In this thesis, the ”offline” temporal shift (b) is employed.

5.5.3 Adaptation for Synthetic Fractal Videos

The network is modified to accept a sequence of T frames as input, with each frame of a fixed size (e.g. 112×112 pixels) and three color channels (RGB). All frames in a sequence are jointly processed, with the TSM modules facilitating temporal information flow across the sequence. The final output of the network is fed into a global average pooling layer followed by a fully-connected layer whose size is adapted to the number of classes in the current experiment (corresponding to the number of unique fractal-generated actions or real-world action classes in downstream tasks).

6 Training Setup

This chapter details the experimental protocols established to evaluate the effectiveness of pre-training action recognition models on synthetic 3D fractal-based datasets. The setup covers both the pre-training phase on synthetic data and the subsequent transfer and evaluation on real-world video action recognition benchmarks. The goal is to rigorously assess whether the features learned from 3D fractal geometry generalize to natural video tasks, as well as to ensure full reproducibility of all experiments.

6.1 Pre-training Setup

The core pre-training objective is supervised video classification, where the network is trained to distinguish between action classes corresponding to distinct fractal transformation patterns. Each class is defined by a set of IFS parameters and associated temporal transformation, as described in Chapter 5. The model receives as input a clipped video sequence of rendered frames (18-20 frames, each of size 112×112 pixels, 3 channels), where each video represents a temporally coherent transformation of a unique 3D fractal point cloud.
For the backbone architecture, a **ResNet-50** [44] equipped with a **Temporal Shift Module (TSM)** [63] is used as the primary action recognition model, given its established effectiveness in prior synthetic pre-training work and its efficiency in spatio-temporal modeling. The input to the model consists of downsampled RGB frame sequences (e.g., 112×112 pixels per frame, 8 frames per video, as per dataset generation). The TSM module is integrated into the ResNet-50 backbone, allowing efficient exchange of temporal information across feature channels without substantially increasing model complexity.

The model is trained in a supervised manner using the standard **cross-entropy loss** [36], appropriate for the multi-class video classification scenario. The loss is computed between the predicted logits and the ground-truth class labels over each batch.

The model is implemented in PyTorch, using the official TSM repository as a reference [4]. All training is conducted using mixed-precision (AMP) and multi-GPU support where available. The architecture is compatible with the modular video dataset and training pipeline described in Section 5.2.

6.1.1 Training Environment

All experiments are conducted on a dedicated high-performance computing server (AI-LAB), with the following environment:

- **GPU:** NVIDIA L4
- **CPU:** 16 cores
- **RAM:** 24 GB

- **Software:** Python 3.12, PyTorch 2.6, CUDA 12.8
- **Experiment tracking:** Weights and Biases [14] for full logging and visualization

6.1.2 Training Parameters

Unless otherwise specified, the following hyperparameters, adopted from [88], are used for pre-training:

- **Batch size:** 32
- **Learning rate:** 0.0008
- **Learning rate schedule:** Cosine Annealing [66]
- **Optimizer:** AdamW [65] (betas = 0.9, 0.999; epsilon = 1e-8)
- **Weight decay:** 0.01
- **Number of epochs:** 25
- **Clip length:** 8 frames per clip (randomly sampled for each batch)
- **Mixed-precision training:** Enabled (PyTorch AMP)
- **Seed:** 1 (fixed for reproducibility)

All major parameters, training logs, and model checkpoints are saved for each experiment to support full reproducibility and ablation analysis.

6.1.3 Pre-Training Data Augmentation: Experimental Parameters

As detailed in Section 5.3, all training and fine-tuning employ a modular augmentation pipeline, with the exact parameters and activation probabilities determined by the experiment's configuration file. For full transparency and reproducibility, the specific augmentation settings used in this thesis are inspired from [88] and are reported below.

Pre-training Augmentation. Unless otherwise specified, pre-training on fractal-based datasets uses the following augmentation parameters:

- **Random cropping and resizing:** Minimum crop scale of 0.9 (i.e., `MIN_SCALE` = 0.9)
- **Horizontal flip:** Enabled (`HOR_FLIP` = 1)
- **RandAugment:** Magnitude $M = 7$, $N = 2$ operations per sample
- **Mixup:** Enabled with background blending (`TYPE_MIXUP` = back)
- **Probabilities for advanced augmentations:**
  - **Perspective distortion:** $p = 0.8$
  - **Scaling:** $p = 1.0$
  - **Shift:** $p = 0.3$
  - **Shake:** $p = 0.3$
  - **Zoom:** $p = 0.3$

- **Clone (object copy-paste):** $p = 0.15$
- **Gaussian blur:** 0.5

Each augmentation (except cropping and flipping) is applied independently per video with the listed probability.

6.1.4 Training Framework

The training framework is a modular PyTorch codebase with custom dataset loaders for the fractal video format, distributed training support (multi-GPU) and full configuration via configuration file or command-line arguments. The pipeline supports experiment resumption, evaluation at regular intervals, and on-the-fly dataset augmentation. All runs are tracked using Weights and Biases (wandb) for transparent reporting of results [14].

6.2 Downstream Task Setup: Action Recognition

The pre-trained models are adapted to each downstream dataset by replacing the output head with a randomly initialized fully-connected layer sized for the number of downstream classes. Fine-tuning is performed end-to-end with all weights updated, unless otherwise specified. The backbone may optionally be frozen for several epochs before full training, though in most experiments all layers are trainable. Augmentation during fine-tuning includes random cropping, horizontal flipping, and normalization as per ImageNet statistics.
To evaluate the transferability of fractal-based pre-training, models are fine-tuned and tested on established action recognition benchmarks:

- **UCF101** [86]: 13,320 videos, 101 action categories, diverse scenes and camera motion.
- **HMDB51** [59]: 6,766 videos, 51 action categories, challenging due to realistic occlusion and variability.

All datasets are preprocessed to match the input size and format used in pre-training (e.g., 112×112 pixels, 8-frame clips).

6.2.1 Downstream Training Parameters

Standard training hyperparameters for fine-tuning are adopted from [88] and are as follows:

- **Batch size:** 32
- **Learning rate:** 0.0008
- **Learning rate schedule:** Cosine Annealing
- **Optimizer:** AdamW (betas = 0.9, 0.999; epsilon = 1e-8)
- **Weight decay:** 0.01
- **Number of epochs:** 100

- **Clip length:** 8 frames per clip
- **Evaluation:** Standard train/validation split for each dataset, following official protocols

6.2.2 Downstream Training Data Augmentation: Experimental Parameters

Similar to the pre-training data augmentation, the exact augmentation parameters are reported below.

Downstream Training Augmentation. For fine-tuning and evaluation on real-world datasets (UCF101, HMDB51), the augmentation settings revert to the default, more conservative configuration as per [88]:

- **Random cropping and resizing:** Minimum crop scale of 0.2 (`MIN_SCALE` = 0.2)
- **Horizontal flip:** Enabled (`HOR_FLIP` = 1)
- **RandAugment:** Magnitude M = 7, N = 2 operations per sample
- **Mixup:** Disabled (`TYPE_MIXUP` = none)
- **Probabilities for advanced augmentations:**
  - **Gaussian blur:** $p = 0.5$
  - **Perspective distortion:** $p = 0.0$
  - **Scaling:** $p = 0.0$
  - **Shift:** $p = 0.0$
  - **Shake:** $p = 0.0$
  - **Zoom:** $p = 0.0$
  - **Clone:** $p = 0.0$

6.2.3 Evaluation Metrics

The evaluation of action recognition models in this thesis is guided by metrics that are widely adopted in the field and are tailored to the scope and objectives of this work. Since the primary focus is on assessing classification performance of models pre-trained on synthetic fractal datasets, the following metrics are used throughout all experiments:

- **Top-1 Accuracy:** The proportion of test samples for which the model's highest-confidence prediction exactly matches the ground truth class label. This is the principal metric used for reporting final model performance.
- **Top-5 Accuracy:** The percentage of test samples where the ground truth label appears among the model's five highest-confidence predictions. This metric provides additional insight into the model's ranking ability and its sensitivity to visually similar actions.

These classification metrics are computed on validation splits according to the established protocols for each dataset.
Training Loss Monitoring:

In addition to accuracy, the cross-entropy loss is monitored over the course of training and validation. While not a direct indicator of final task performance, loss curves are used to analyze convergence behavior, detect overfitting or underfitting, and to compare optimization stability across different experimental settings.

Computational Efficiency:

To provide a holistic assessment of model practicality, the total wall-clock time required to train each model to completion on the target dataset is recorded and reported. Monitoring training time is essential for evaluating the feasibility of deploying these methods in real-world or resource-constrained environments.

Qualitative Evaluation of Synthetic Datasets:

As universally accepted quantitative metrics for evaluating the quality of synthetic datasets—such as 3D fractal video sequences—are lacking, this thesis emphasizes qualitative research. Methods for generating representative samples are investigated to help ensure structural complexity, diversity, and the exclusion of degenerate or trivial examples from the training data. The ongoing challenge of developing standardized, objective measures for synthetic data quality is acknowledged as an important area for future work.

6.3 Reproducibility and Experiment Management

To ensure the reproducibility and transparency of all experiments, scripts, configuration files, and random seeds are archived for each run. Model checkpoints and logs are systematically saved for every training and evaluation session. Furthermore, all experimental results, including training and validation metrics, are tracked in real time using the Weights and Biases platform. This approach enables transparent reporting, facilitates in-depth analysis, and supports future benchmarking efforts.

7 Experiments and Results

This chapter presents the experimental evaluation of pre-training action recognition models using 3D fractal-based synthetic datasets. The experiments are designed to systematically investigate how variations in fractal generation parameters, transformation regimes, and color augmentation strategies impact both pre-training effectiveness and downstream performance on real-world video benchmarks. The chapter begins with an overview of the experimental design and a summary table of all experiments. Subsequent sections present quantitative and qualitative results, comparative analysis against standard baselines, and ablation studies.

7.1 Overview and Experimental Design

To ensure full reproducibility and interpretability, each experiment varies only one fundamental factor at a time, such as the magnitude of geometric transformations, the coloring augmentations or the size of the dataset, while holding all other variables fixed. Table 7.1 provides a concise summary of all experimental conditions, highlighting the key differences across runs. This controlled design enables meaningful insight of observed effects to specific dataset or augmentation modifications.

Table 7.1: Summary of main experiments: Each experiment varies a single key property (transformation magnitude, color scheme or class/instance size) to isolate its impact on downstream action recognition. All experiments use the same fractal generation pipeline unless noted otherwise.

ID	Total size	Instances / Class	Transformation	Color Scheme
1	50 000	100/500	None	Low
2	50 000	100/500	Low	Low
3	50 000	100/500	Moderate	Low
4	50 000	100/500	High	Low
5	50 000	100/500	Moderate	Moderate
6	50 000	100/500	Moderate	High
7	100 000	200/500	Moderate	Moderate
8	100 000	400/250	Moderate	Moderate
9	200 000	400/500	Moderate	Moderate

7.2 Description of Experiments

Systematic variation of experimental conditions is essential for understanding which factors most significantly influence pre-training effectiveness and downstream action recognition performance. In this thesis, each experiment is designed to isolate the impact of a single variable, such as the strength of geometric transformations, the coloring strategy or dataset size, while all other parameters are held fixed as described in Chapter 6. The tri-level design (low, moderate, high) for each variable was chosen to efficiently explore the experimental space within practical time constraints, and to show the positive direction required in further optimization.

For clarity, unless otherwise noted, the fractal generation pipeline, training augmentation settings, and training protocol remain unchanged across experiments.

7.2.1 Transformation Level Experiments

This set of experiments investigates the effect of geometric transformation strength during fractal video generation. Specifically, four types of transformations are considered: rotation, shear, translation, and non-affine warp. Notably, the non-affine spatial warp is defined by six parameters: `[scale_x, freq_x, scale_y, freq_y, scale_z, freq_z]`. For each axis, scale controls the maximum displacement amplitude, while freq determines the spatial frequency of the sinusoidal deformation, which is applied based on the point's coordinates on the other two axes. For each fractal video instance, the parameters for these transformations are randomly sampled up to a maximum value corresponding to the designated ”Low”, ”Moderate”, or ”High” transformation level. The precise upper bounds for each transformation type at each level are listed in Table 7.2.
The ”Moderate” setting was established as a baseline, selected empirically based on visual inspection for reasonable but non-degenerate deformation of the 3D fractal point cloud. The “Low” and ”High” settings provide a controlled range around this baseline. Importantly, for each video instance, the actual transformation parameters are drawn randomly from a uniform distribution between zero and the maximum value specified for that experiment level, ensuring transformational diversity within each generated video.

Table 7.2: Transformation Experiment: Magnitude of individual transformations applied to the fractal.

Magnitude	Rotation	Shear	Translation	Warp
No Transformation	[0.0, 0.0, 0.0]	[0.0, 0.0]	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Low	[270, 270, 270]	[0.6, 0.6]	[0.6, 0.6, 0.6]	[0.2, 4.0, 0.2, 4.0, 0.2, 4.0]
Moderate	[360, 360, 360]	[0.8, 0.8]	[0.8, 0.8, 0.8]	[0.4, 6.0, 0.4, 6.0, 0.4, 6.0]
High	[450, 450, 450]	[1.0, 1.0]	[1.0, 1.0, 1.0]	[0.6, 8.0, 0.6, 8.0, 0.6, 8.0]

7.2.2 Color Level Experiments

Following the tri-level approach used for transformation strength, a series of experiments were conducted to assess how different color augmentation strategies affect model pre-training and downstream recognition. Three distinct levels of color augmentation were defined:

- **Low**: Grayscale coloring based on the z-coordinate, reducing depth information to a single intensity channel without color diversity.
- **Moderate**: Application of the ”viridis” colormap [5] to the z-axis, introducing consistent color-depth mapping while adding three-channel input information.
- **High**: Diverse coloring strategies randomly selected for each instance, including a range of colormaps and coloring bases (e.g., by x, y, or z axis, distance from center, random solid color, or random color per point).

Table 7.3 summarizes these three levels and the core strategy used in each.

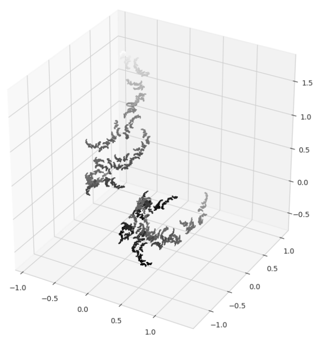
Table 7.3: Color Experiment: Magnitude of color augmentation applied to the fractal.

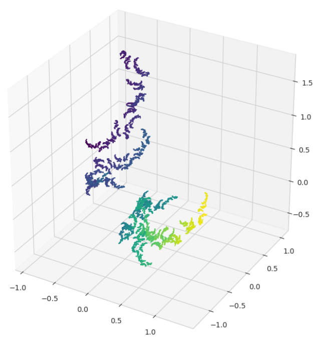
Magnitude	**Strategy**
Low	Grayscale along z-axis
Moderate	Viridis along z-axis
High	Multiple strategies

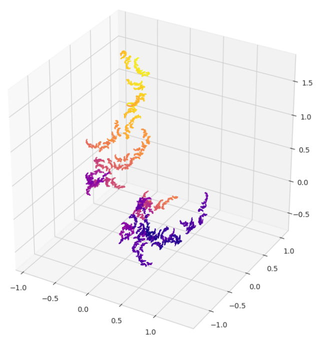
The "High" color augmentation magnitude utilizes a variety of coloring strategies, which are randomly assigned to instances. Table 7.4 details these strategies, while Figure 7.1 presents visual examples.

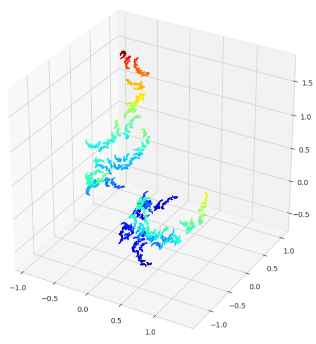
Table 7.4: Color Experiment: Levels of color augmentation applied to the fractals in a dataset.

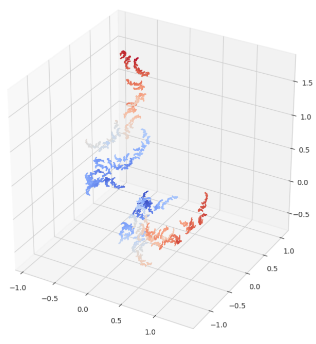
Strategy	**Random Choice**
X-axis coloring	['viridis', 'plasma', 'jet', 'coolwarm']
Y-axis coloring	['viridis', 'plasma', 'jet', 'coolwarm']
Z-axis coloring	['viridis', 'plasma', 'jet', 'coolwarm']
Radial coloring	['viridis', 'plasma', 'jet', 'coolwarm']
Random solid color	All colors
Random color per point	All colors

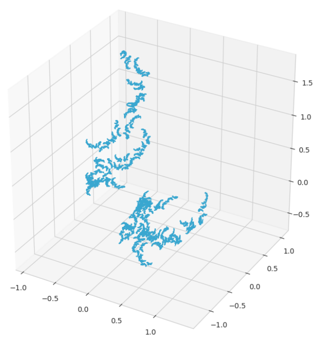

(a) Grayscale along *z*-axis


(b) Viridis along *x*-axis


(c) Plasma along *y*-axis


(d) Jet along *z*-axis


(e) Coolwarm by distance from center


(f) Random solid color

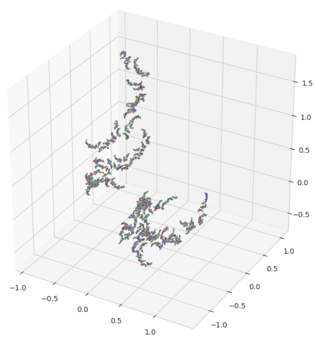

(g) Random color per point

Figure 7.1: Comparison of different coloring strategies applied to the same 3D fractal structure.

This experimental structure allows for controlled investigation of how both the amount and the nature of color information influence feature learning. To avoid an unmanageable number of permutations, the best-performing transformation regime (Moderate) was fixed when exploring color augmentation effects.

Note: The specific coloring strategies applied at the "High" augmentation level are detailed in Table 7.4, while Figure 7.1 shows visual examples of each approach. Together, the table and figure provide both a technical reference and an intuitive sense of how different coloring schemes alter the appearance of the synthetic fractal actions.

7.2.3 Ablation Experiments

After identifying the best-performing settings in the transformation and color augmentation experiments, further ablation studies were conducted to examine the effect of dataset scale. Specifically, these experiments varied the number of instances per class and the total number of classes, with the goal of determining whether and how increased data diversity or quantity translates to improved performance on downstream tasks. The guiding hypothesis was that larger and more varied synthetic datasets may yield better generalization and higher accuracy in real-world action recognition benchmarks.

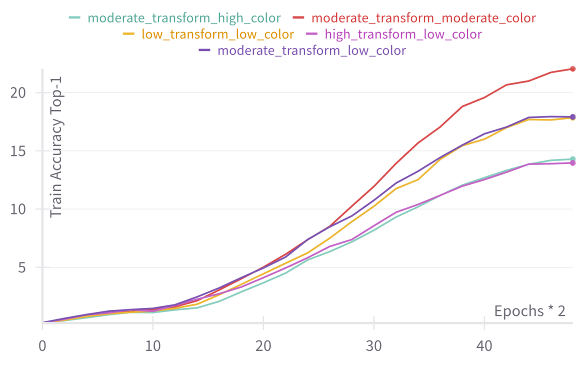
7.3 Pre-training Performance

This section presents the quantitative results of all pre-training experiments, corresponding to the experiment configurations summarized in Table 7.1. For each experiment, the report contains key metrics including Top-1 and Top-5 classification accuracy on both training and validation sets and the lowest achieved train loss.
Table 7.5 provides a side-by-side comparison across all runs, enabling clear and direct interpretation of the impact of transformation magnitude and color augmentation on pre-training effectiveness. Each experiment used the same training configuration and required approximately 3 hours and 15 minutes to complete.
Ablation experiments investigating variations in class and instance count are discussed separately in Section 7.5.

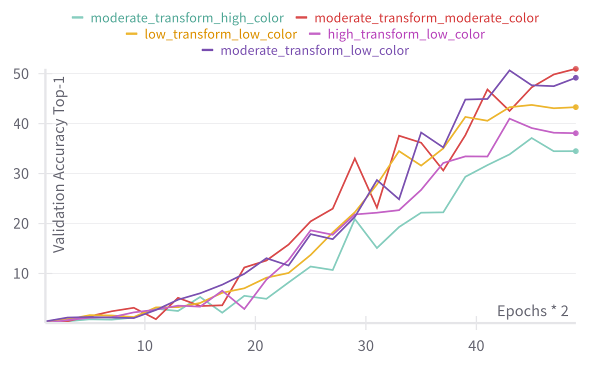
Table 7.5: Pre-training performance summary for all experiment runs. Each row corresponds to an experiment ID as defined in Table 7.1. Metrics are reported for the final checkpoint. <u>**Best**</u> and **second best** results per column are highlighted in <u>**underlined bold**</u> and **bold**, respectively.

ID	Transform	Color	↑Train Top-1	↑Train Top-5	↑Val Top-1	↑Val Top-5	↓Train Loss
1	None	Low	<u>**100.00%**</u>	<u>**100.00%**</u>	<u>**100.00%**</u>	<u>**100.00%**</u>	<u>**≈ 0**</u>
2	Low	Low	17.85%	39.14%	43.30%	76.88%	3.91
3	Moderate	Low	17.92%	40.44%	49.18%	**82.06%**	3.84
4	High	Low	13.96%	34.54%	38.08%	74.02%	4.07
5	Moderate	Moderate	**22.06%**	**45.62%**	**50.98%**	81.96%	**3.58**
6	Moderate	High	14.30%	34.30%	34.50%	69.44%	4.09

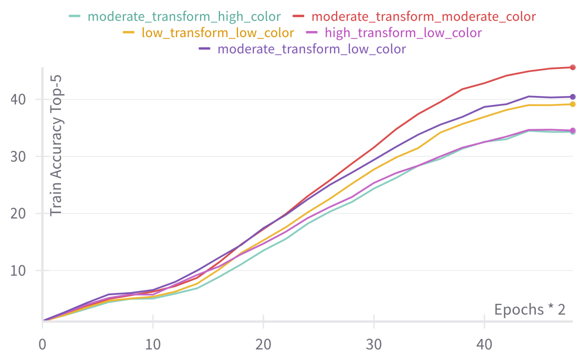
The learning curves in Figure 7.2 complement the tabular results by providing a detailed temporal perspective on training performance. In particular, validation accuracies (Figures 7.2b and 7.2d) demonstrate not only the best-performing runs but also the differences in learning stability and generalization, while the training loss (Figure 7.2e) consistently decreases across all experiments, confirming effective training. The training loss curve doesn't show a plateau, suggesting that increasing the number of epochs could improve the model's accuracy.



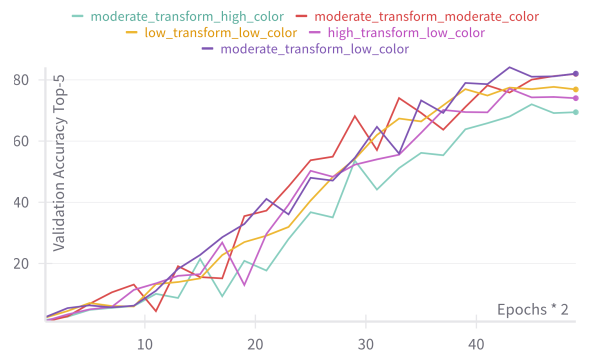
(a) Train set Top-1 accuracy over epochs.



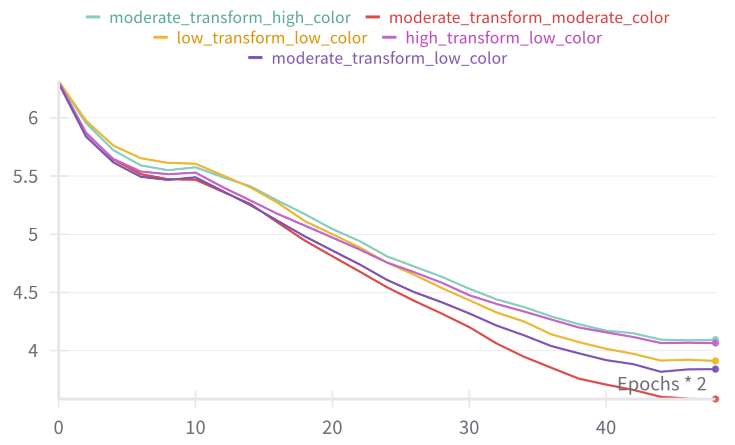
(b) Validation set Top-1 accuracy over epochs.



(c) Train set Top-5 accuracy over epochs.



(d) Validation set Top-5 accuracy over epochs.



(e) Training Loss over epochs

Figure 7.2: Evolution of pre-training performance metrics across 25 epochs for all experiments. (a)–(d): Top-1 and Top-5 accuracy on the train and validation sets, respectively. These curves show increasing accuracy with diminishing slope near the end of training. (e): Training loss (cross-entropy), which steadily decreases, confirming effective learning throughout pre-training.

7.3.1 Analysis and Notes on Individual Experiments

Experiment 1 (No Transformation, Low Color): This run yielded perfect accuracy (100%) and near-zero loss on both training and validation sets. However, these results are not meaningful for representation learning: without any transformations, the generated fractals are highly similar within each class, making classification trivial. The model likely memorized superficial static patterns instead of learning features that could generalize. As further discussed in Section 7.4, this is confirmed by the poor downstream transfer performance. Accordingly, this run is excluded from further comparative analysis.

Experiments 2–4 (Varying Transformation Magnitude, Low Color): Introducing geometric transformations increases dataset complexity and improves the quality of learned features. Among these, the Moderate transformation (ID=3) achieves the highest accuracy (Val Top-1: 49.18%, Top-5: 82.06%), with the Low transformation (ID=2) also performing well (Val Top-1: 43.30%). However, the High transformation (ID=4) results in a substantial drop in performance (Val Top-1: 38.08%), likely due to excessive deformation that makes feature learning more difficult. The learning curves (see Figures 7.2b and 7.2d) show a clear separation between Moderate/Low and High transformation regimes, supporting this conclusion.

Experiments 5–6 (Effect of Color Augmentation): With transformation magnitude fixed at the optimal "Moderate" level, color augmentation is varied:

- **Moderate color** (ID=5), using a three-channel "viridis" colormap along the z-axis, achieves the best overall performance except for Top-5 validation accuracy (Val Top-1: 50.98%, Train Top-1: 22.06%, Loss: 3.58). This highlights the benefit of three-channel information with consistent color-depth encoding.
- **High color** (ID=6), with randomly selected coloring strategies, reduces accuracy across the board (Val Top-1: 34.50%), suggesting that excessive color variation can obscure underlying geometric information and weaken the network performance, as visualized in the slower and poor performing validation accuracy plots (Figures 7.2b, 7.2d).

General Trends: Moderate levels of both transformation and color augmentation yield the highest pre-training accuracy and best balance between diversity and learnability. Overly aggressive augmentation, either in geometric transformations or color variation, diminishes model performance. Run with no transformation should be disregarded as it fails to induce meaningful feature learning.

7.4 Downstream Performance

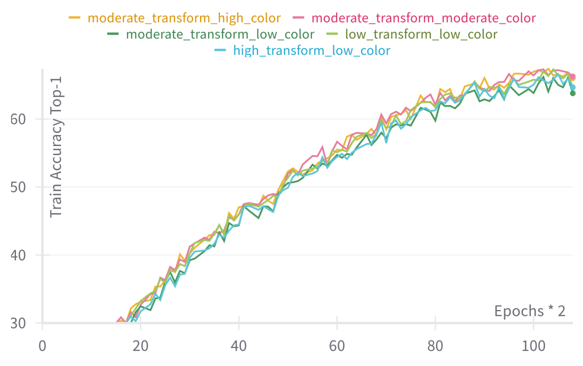
This section presents a detailed analysis of the downstream fine-tuning performance achieved on two widely used action recognition benchmarks: HMDB51 and UCF101. The same experiment configurations as those described for pre-training are adopted from [88], allowing for a direct evaluation of how differences in fractal dataset design and augmentation affect real-world task transfer. Results are reported for the final model checkpoint, and are presented using both summary tables and training/validation curves to enable exact comparison across experimental settings.

Overview of Presented Results: Key metrics include Top-1 and Top-5 classification accuracy for both the training and validation sets, as well as final training loss. The main findings are summarized in Table 7.6 (HMDB51) and Table 7.7 (UCF101). For each dataset, corresponding training and validation curves (Figures 7.3 and 7.4) provide further insight into training dynamics and generalization.
Note: While strong pre-training metrics are often viewed as a sign of effective feature learning, downstream performance does not always correlate perfectly with pre-training results. This section highlights both consistencies and discrepancies, emphasizing that downstream transfer is the ultimate test of representation quality.

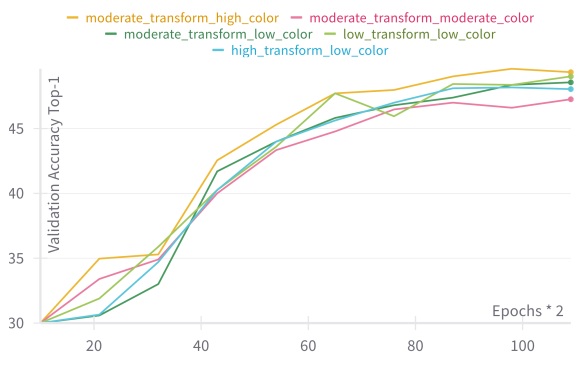
7.4.1 HMDB51 Fine-Tuning Results

Table 7.6: Fine-tuning performance summary on HMDB51. Each row corresponds to an experiment ID. Metrics are reported for the final checkpoint. <u>**Best**</u> and **second best** results per column are highlighted in <u>**underlined bold**</u> and **bold**, respectively.

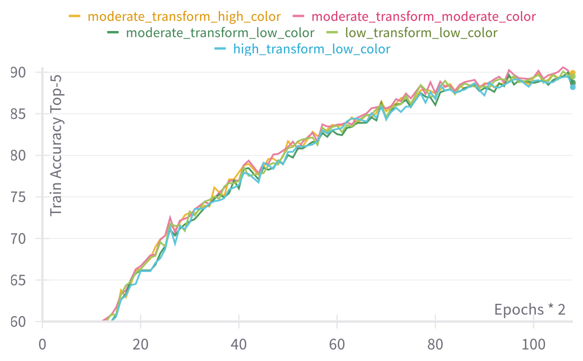
HMDB51 Fine-tuning Results							
ID	**Transform**	**Color**	**↑Train Top-1**	**↑Train Top-5**	**↑Val Top-1**	**↑Val Top-5**	**↓Train Loss**
1	None	Low	62.39%	87.36%	35.95%	69.93%	1.3327
2	Low	Low	64.67%	**89.50%**	**49.02%**	78.82%	1.2029
3	Moderate	Low	63.80%	88.77%	48.56%	**79.15%**	1.2439
4	High	Low	64.67%	88.20%	48.04%	79.02%	1.2371
5	Moderate	Moderate	<u>**66.27%**</u>	88.60%	47.25%	78.10%	<u>**1.1806**</u>
6	Moderate	High	**66.02%**	<u>**89.92%**</u>	<u>**49.35%**</u>	<u>**79.41%**</u>	**1.1930**



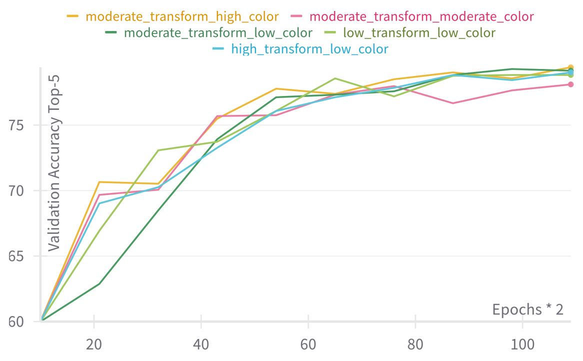
(a) HMDB51: Train set Top-1 accuracy.



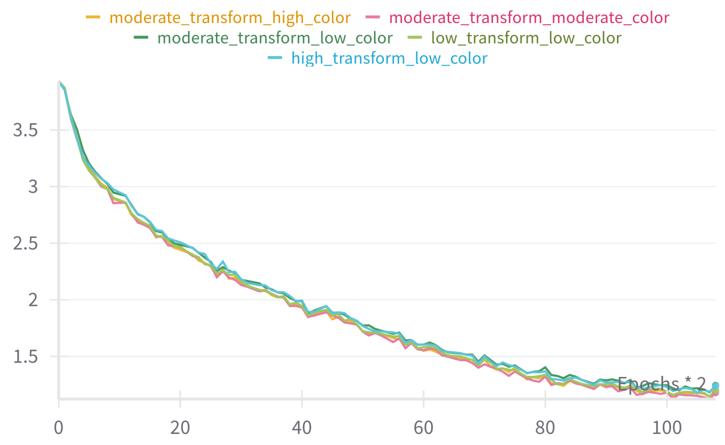
(b) HMDB51: Validation set Top-1 accuracy.



(c) HMDB51: Train set Top-5 accuracy.



(d) HMDB51: Validation set Top-5 accuracy.



(e) HMDB51: Training loss over epochs.

Figure 7.3: Performance metrics during training on the HMDB51 dataset. (a)-(d): Top-1 and Top-5 accuracy curves for the training and validation sets, respectively, showing learning progression. (e): Training loss (cross-entropy) over epochs, indicating model optimization.

Table 7.6 provides a side-by-side comparison of all experimental runs on HMDB51. Figure 7.3 shows the evolution of Top-1 and Top-5 accuracy as well as loss throughout training. All training configurations were kept identical to ensure a fair comparison across experiments. Each training run required approximately 1 hour and 17 minutes to complete.

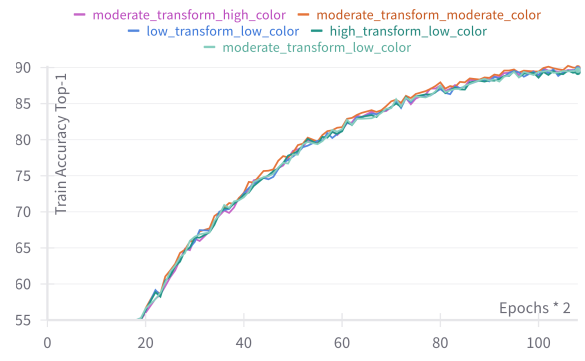
HMDB51 Result Analysis

- **Impact of Color Augmentation:** The HMDB51 results show that the "Moderate Transform, High Color" configuration (ID=6) slightly outperforms other settings, achieving the best validation Top-1 (49.35%) and Top-5 (79.41%) accuracy. This contrasts with the pre-training phase, where moderate color augmentation yielded the highest scores. This suggests that, for HMDB51, a more challenging and diverse dataset generated with higher color diversity during pre-training better prepares the model to generalize to the complexities of real-world actions.
- **Transformation Magnitude:** All runs consistently outperform runs with no transformation (ID=1), again confirming that diversity and geometric complexity in the synthetic pre-training data are essential for transferable feature learning.
- **Comparison to Pre-training Performance:** Notably, experiments with the best pre-training metrics do not always translate directly to superior downstream results. For example, while "Moderate Transform, Moderate Color" was best for pre-training, the "High Color" setting edges it out on HMDB51. This reinforces that tuning pre-training for downstream objectives is nontrivial, and that some degree of mismatch between pre-train and downstream performance is to be expected.

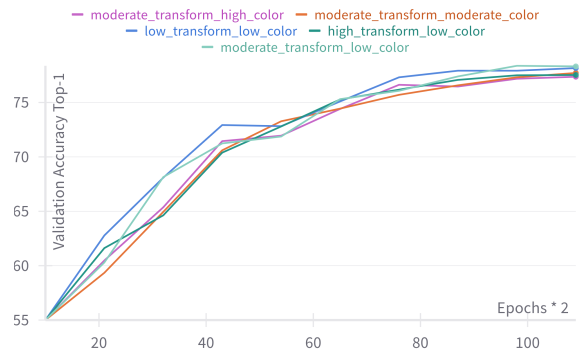
7.4.2 UCF101 Fine-Tuning Results

Table 7.7: Fine-tuning performance summary on UCF101. Each row corresponds to an experiment ID. Metrics are reported for the final checkpoint. **Best** and **second best** results per column are highlighted in **underlined bold** and **bold**, respectively.

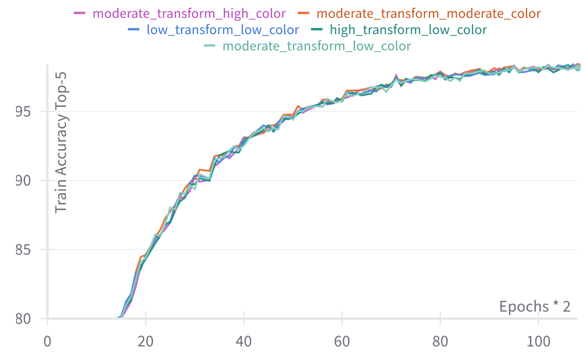
UCF101 Fine-tuning Results							
ID	**Transform**	**Color**	**↑Train Top-1**	**↑Train Top-5**	**↑Val Top-1**	**↑Val Top-5**	**↓Train Loss**
1	None	Low	<u>**90.45%**</u>	98.07%	67.99%	89.19%	**0.3496**
2	Low	Low	**89.81%**	98.12%	**78.17%**	94.61%	0.3528
3	Moderate	Low	89.64%	98.19%	<u>**78.32%**</u>	94.74%	0.3572
4	High	Low	89.37%	98.20%	77.53%	94.90%	0.3628
5	Moderate	Moderate	89.93%	**98.28%**	77.72%	<u>**95.11%**</u>	<u>**0.3415**</u>
6	Moderate	High	89.54%	<u>**98.30%**</u>	77.37%	**94.85%**	0.3505



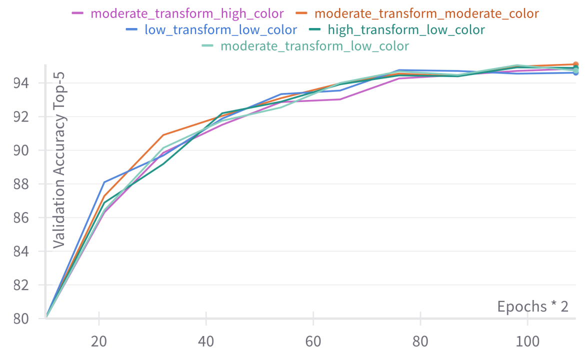
(a) UCF101: Train set Top-1 accuracy.



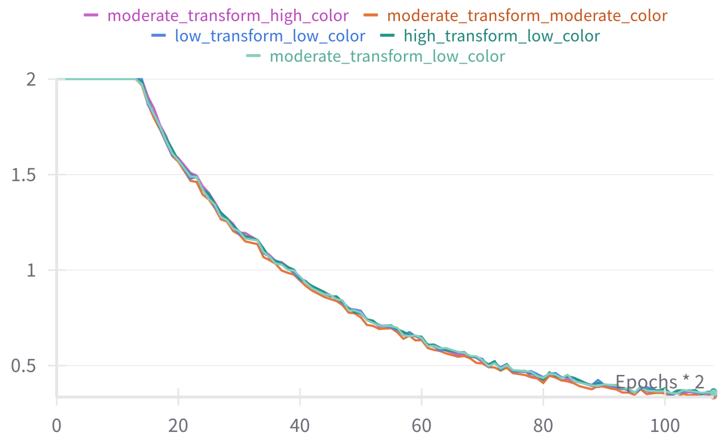
(b) UCF101: Validation set Top-1 accuracy.



(c) UCF101: Train set Top-5 accuracy.



(d) UCF101: Validation set Top-5 accuracy.



(e) UCF101: Training loss over epochs.

Figure 7.4: Performance metrics during training on the UCF101 dataset. (a)-(d): Top-1 and Top-5 accuracy curves for the training and validation sets, respectively, illustrating learning behavior. (e): Training loss (cross-entropy) over epochs, demonstrating model convergence.

Table 7.7 summarizes results on the UCF101 dataset, while Figure 7.4 presents the associated learning curves. All training configurations were kept identical to ensure a fair comparison across experiments. Each training run required approximately 3 hour and 24 minutes to complete.

UCF101 Result Analysis

- **Best Overall Setting:** The ”Moderate Transform, Moderate Color” experiment (ID=5) emerges as the most effective configuration for the UCF101 dataset. It achieves the best validation Top-5 accuracy (95.11%) and the lowest training loss, indicating a strong balance between feature learning and generalization. While the ”Moderate Transform, High Color” setting (ID=6) is competitive, the moderate color augmentation appears to provide slightly more stable and ultimately higher validation performance. This aligns with observations from the pre-training phase, where moderate augmentation settings were generally found to be optimal.
- **Effect of Color Augmentation Levels:** Unlike the HMDB51 dataset, UCF101 does not show a significant benefit from high levels of color diversity in pre-training. The ”High Color” configuration (ID=6), while performing strongly, does not surpass the results obtained with moderate color settings. This suggests that the visual patterns and actions within the UCF101 dataset are effectively captured when the model is pre-trained with moderately varied yet consistent color cues, rather than extreme color augmentation.
- **Learning Curves and Training Dynamics:** The learning curves across all experimental configurations for UCF101 display similar positive trends. Both training and validation accuracies consistently increase throughout the training process, with no indications of premature plateauing. The continued improvement in both Top-1 and Top-5 validation accuracy suggests that extending the number of training epochs could potentially yield further performance gains.
- **Generalization and Overfitting Insights:** As is typical, training accuracy figures are generally higher than their validation counterparts across all UCF101 experiments. However, the observed gap between training and validation performance is narrower for UCF101 compared to HMDB51, indicating more robust generalization on this particular dataset. Consistent with previous findings, the experiment run without any pre-training transformations (ID=1) shows a significant lag in validation accuracy despite achieving high training performance, underscoring the importance of learning meaningful features through diverse transformations during the pre-training stage.

7.4.3 General Observations and Comparison

- **Pre-train vs. Downstream Alignment:** The Moderate-Moderate setting (moderate transformation, moderate color) generally provides the best or nearly-best results on both datasets, confirming it as the most balanced and robust approach overall. However, there is no perfect mapping between pre-train and downstream ranking, highlighting the value of downstream validation as the final judge of model effectiveness.
- **Color Augmentation Insights:** High color diversity is most advantageous for HMDB51. For UCF101, moderate color offers the most stable improvements, indicating that over-augmentation may dilute the underlying spatial cues critical for recognition.
- **Impact of Training Duration:** The learning curves for both datasets rise steadily up to the last epoch, with no convergence plateau reached. This suggests that extending training could yield better results.

- **Consistency with Dataset Design:** The trends observed here are consistent with those found in the synthetic pre-training experiments: diversity in the pre-training data (via geometric transformation and color augmentation) is necessary, but excessive augmentation can harm generalizability. Finding the right balance is key to maximizing downstream performance.

7.5 Ablation Study

To evaluate the sensitivity of downstream performance to the scale of synthetic pre-training data, an ablation study was conducted. As described previously in Section 7.2.3, these experiments systematically vary both the number of instances per class and the total number of classes in the fractal dataset. The primary objective is to assess how expanding dataset size and class diversity during pre-training impacts final performance on both the synthetic validation set and the downstream benchmarks HMDB51 and UCF101.

Table 7.8 summarizes the pre-training results for each ablation configuration, while Tables 7.9 and 7.10 present the corresponding fine-tuning results on HMDB51 and UCF101, respectively. Each row matches an experiment ID from Table 7.1, clearly indicating the number of instances and classes used. In addition to the ablation study results, a best performing non-ablation run is shown for easier comparision.

The duration of the pre-training phase is influenced by the dataset's overall scale and complexity, specifically the total number of instances and the diversity of classes. The recorded training times for the configurations explored in the ablation study (corresponding to IDs 7, 8, and 9 in Table 7.8) were as follows:

- **ID 7** (200 instances/class, 500 classes; 100 000 total instances): 8 hours 41 minutes
- **ID 8** (400 instances/class, 250 classes; 100 000 total instances): 6 hours 22 minutes
- **ID 9** (400 instances/class, 500 classes; 200 000 total instances): 16 hours 2 minutes

Table 7.8: Pre-training performance for dataset size ablation: Varying the number of classes and instances in the fractal dataset. <u>**Best**</u> and **second best** results per column are highlighted.

PRE-TRAIN Dataset Size Experiment Results							
ID	**#Instance**	**#Class**	**↑Train Top-1**	**↑Train Top-5**	**↑Val Top-1**	**↑Val Top-5**	**↓Train Loss**
5	100	500	22.06%	45.62%	50.98%	81.96%	3.58
7	200	500	**66.40%**	**81.64%**	**96.79%**	**99.16%**	**1.4875**
8	400	250	40.39%	65.64%	86.20%	97.88%	2.4903
9	400	500	<u>**83.76%**</u>	<u>**91.36%**</u>	<u>**98.53%**</u>	<u>**99.38%**</u>	<u>**0.7251**</u>

Table 7.9: HMDB51 downstream performance for different synthetic pre-training dataset sizes. Rows show the effect of scaling the number of classes and instances on downstream accuracy. **Best** and **second best** results per column are highlighted.

HMDB51 Fine-tuning Results (Dataset Size Experiments)							
ID	**#Instance**	**#Class**	**↑Train Top-1**	**↑Train Top-5**	**↑Val Top-1**	**↑Val Top-5**	**↓Train Loss**
5	100	500	66.27%	88.60%	47.25%	78.10%	1.1806
7	200	500	**73.73%**	**93.27%**	**49.02%**	**78.04%**	**0.9014**
8	400	250	69.20%	91.24%	47.97%	**79.87%**	1.0564
9	400	500	**80.32%**	**95.02%**	**50.85%**	**79.87%**	**0.6919**

Table 7.10: UCF101 downstream performance for different synthetic pre-training dataset sizes. Rows show the effect of scaling the number of classes and instances on downstream accuracy. **Best** and **second best** results per column are highlighted.

UCF101 Fine-tuning Results (Dataset Size Experiments)							
ID	**#Instance**	**#Class**	**↑Train Top-1**	**↑Train Top-5**	**↑Val Top-1**	**↑Val Top-5**	**↓Train Loss**
5	100	500	89.93%	98.28%	77.72%	95.11%	0.3415
7	200	500	**91.35%**	**98.48%**	**78.38%**	**95.14%**	**0.2916**
8	400	250	90.33%	98.27%	78.09%	94.32%	0.3240
9	400	500	**94.17%**	**99.01%**	**80.41%**	**95.37%**	**0.2090**

7.5.1 Analysis of the Ablation Study

The ablation study demonstrates clear trends regarding the scalability of fractal-based synthetic datasets for pre-training:

- **Effect of Increasing Dataset Size:** Across all three tables (Pre-train, HMDB51, and UCF101 results), the largest dataset configuration (ID=9: 400 instances/class, 500 classes) consistently achieves the best or near-best outcomes in both pre-training and downstream benchmarks. For example, the Top-1 validation accuracy on UCF101 reaches 80.41%, while on HMDB51 it attains 50.85%. This highlights the substantial benefits of increasing both data volume and class diversity for improving transferable representation learning.
- **Impact of Classes versus Instances:** Comparing experiment ID=7 (200 instances/class, 500 classes) to ID=8 (400 instances/class, 250 classes), both of which have the same total dataset sizes but different distributions between class count and instance count, reveals a clear trend: the configuration with more classes and fewer instances per class (ID=7) outperforms the alternative in both pre-training and downstream performance. This suggests that class diversity is a stronger driver of model generalizability than merely increasing the number of examples per class.
- **Pre-training and Downstream Performance Alignment:** The best-performing pre-training setup (ID=9) also yields the strongest downstream results on both HMDB51 and UCF101, reinforcing the conclusion that scaling up the diversity and volume of synthetic pre-training data directly enhances the model's transferability. Nonetheless, it should be noted that improvements diminish

somewhat with increasing dataset size, indicating a saturation effect or the need for additional diversity through more advanced augmentation or fractal generation techniques.

- **Performance of Smaller Dataset Configurations:** The smaller dataset configurations (ID=7 and ID=8) perform competitively, especially on UCF101, but are consistently outperformed by the largest dataset (ID=9). This demonstrates that while moderate-sized synthetic datasets can still provide meaningful pre-training, maximizing both class diversity and the number of instances per class is generally optimal for downstream action recognition.
- **Training Loss and Generalization:** Lower training loss during pre-training is generally associated with higher downstream accuracy on both evaluation benchmarks. However, as established throughout this thesis, it is the validation accuracy on downstream tasks that provides the measure of generalization. Notably, the configuration with the largest pre-training dataset (ID=9) achieves both the lowest training loss and the highest fine-tuning accuracy, supporting the value of large, diverse synthetic pre-training sets.

Overall, these results reinforce the importance of maximizing both class diversity and overall dataset size when constructing synthetic pre-training data for transfer learning. The clear trend that greater class variety is more beneficial than simply increasing the number of instances per class underscores the necessity for representational breadth in synthetic datasets. This insight aligns well with the general objective of promoting generalization across diverse real-world action categories.

7.6 Comparison to Prior Work and Standard Pre-training

Table 7.11 presents a direct comparison of downstream Top-1 validation accuracy (%) between this thesis's best 3D fractal pre-trained models and the most relevant prior 2D fractal pre-training results [88]. Results are grouped by total dataset size and class/instance split to ensure a fair comparison.

While the 3D fractal video pre-training does not surpass the top-performing 2D fractal pre-training method or standardized Kinetics pre-training on either HMDB51 or UCF101, it **consistently outperforms training from scratch by a substantial margin**. For example, on the largest synthetic dataset (400 instances/class, 500 classes; 200k total), the 3D fractal pre-training achieves a Top-1 accuracy of 50.9% on HMDB51 and 80.4% on UCF101, compared to the scratch baseline of 31.5% and 70.3%, respectively (see Table 7.12).

To contextualize the results further, Table 7.12 also includes Kinetics pre-training as a reference. It should be emphasized that this is **not a direct comparison**: the Kinetics-400 results are obtained using a larger dataset (250,000 videos), with a higher input resolution of 224x224 pixels (four times the area per frame) and four times the pre-training epochs (100 epochs versus 25 for fractal pre-training). As such, Kinetics pre-training serves as an upper-bound benchmark.

These findings highlight that, although there remains room for improvement relative to 2D fractal-based approaches, scaling up synthetic 3D fractal video pre-training **provides strong benefits over random initialization**. The clear gain over from-scratch models underlines the practical value of synthetic video pre-training, especially when large-scale real video data (such as Kinetics-400 [54]) is not available. The reasons for the observed performance gap with 2D fractals, and strategies for further closing it, are discussed in detail in Discussion Chapter 9.

Table 7.11: Comparison of downstream Top-1 validation accuracy (%) on HMDB51 and UCF101 between this thesis (3D Fractal) and the best-matching 2D fractal pre-training methods from [88] (2D Fractal). Corresponding configurations based on total dataset size and class/instance distribution are shown. For each row (experimental setup), the <u>**best**</u> and **second best** accuracies across the methods and datasets are highlighted.

#Instances per Class	**#Classes**	**Total Size**	**2D Fractal [88]**		**3D Fractal (This Thesis)**	
			HMDB51	UCF101	HMDB51	UCF101
100	500	50 000	<u>**56.5**</u>	<u>**81.8**</u>	**49.4**	**78.3**
200	500	100 000	<u>**61.5**</u>	<u>**84.9**</u>	**49.0**	**78.4**
400	250	100 000	<u>**60.3**</u>	<u>**85.3**</u>	**47.97**	**78.1**
400	500	200 000	<u>**61.5**</u>	<u>**86.0**</u>	**50.9**	**80.4**

Table 7.12: Comparison of Top-1 Validation Accuracy (%) on HMDB51 and UCF101 between models trained from scratch, models pre-trained with 3D Fractals (this thesis), and models pre-trained with Kinetics dataset. All results correspond to an input resolution of 112x112 except Kinetics, which uses 224x224 and a much larger dataset (250k videos). The <u>**best**</u> result per dataset is highlighted, with the other being **second best**. [88]

Method	**HMDB51**	**UCF101**
Scratch	31.5	70.3
3D Fractal	**50.9**	**80.4**
Kinetics*	<u>**70.1**</u>	<u>**95.3**</u>

While the current approach does not surpass the very best results from 2D fractal-based pre-training and Kinetics pre-training, it robustly outperforms models trained from scratch, validating the value of large-scale synthetic video for pre-training in data-scarce domains. The findings suggest that further advances in fractal data generation or augmentation could help close the remaining gap to state-of-the-art synthetic video pre-training.

These findings comprehensively address the thesis objectives, demonstrating that synthetic 3D fractal video pre-training can yield significant performance gains in real-world action recognition tasks, especially in the absence of large-scale annotated video datasets.

8 Improving Fractal Geometry

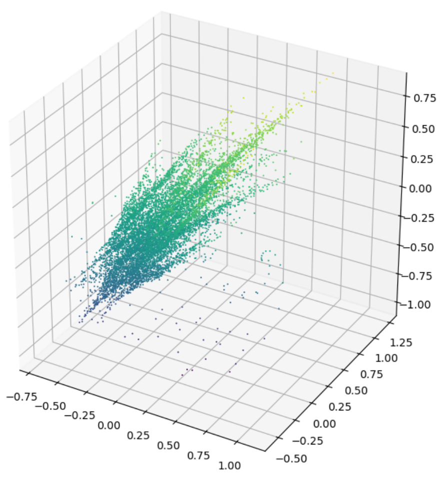
This chapter presents additional research aimed at enhancing the geometric quality of the generated fractals. Visual inspection of the initial datasets revealed that a portion of classes exhibited degenerate or non-representative geometry, motivating a deeper investigation into fractal parameterization. These research was further motivated by plateauing in training results by tuning transformation and color augmentations. The methods and findings discussed in this chapter are closely aligned with the approaches in [7].

8.1 Revisiting Fractal Shape Controlling Parameters

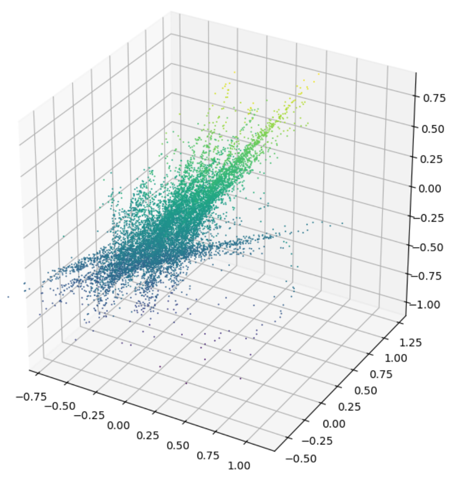
The baseline parameter sampling strategy used thus far is rudimentary, with the primary filtering criterion being sufficient variance across all three axes. While this approach does help eliminate the most trivial degenerate fractals, it remains insufficient in non-representative and poorly structured fractals that still leak into the training dataset.
Given these limitations, it became necessary to revisit and refine the parameters governing fractal shape to further improve dataset quality and, by extension, downstream model performance.
Following the method in [7], each A matrix of the IFS was decomposed into four component matrices using Singular Value Decomposition, as detailed in Section 4.2:

$$A_i = R_{\theta,i} \cdot \Sigma_i \cdot R_{\phi,i} \cdot D_i \tag{8.1}$$

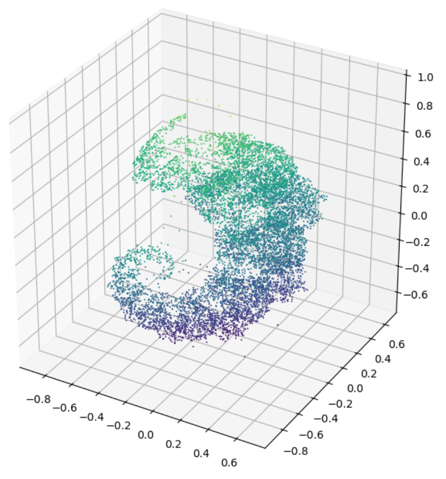
Here, R denotes rotation matrices, Σ is a diagonal matrix of singular values, and D is a diagonal matrix with entries in $\{-1,1\}$ to allow reflections.
To enforce contractivity, the elements of Σ: $[\sigma_1,\sigma_2,\sigma_3]$ were sampled from $[0,1]$ and sorted in descending order according to SVD theorem. Rotation matrices were sampled from $[-\pi,\pi]$. This decomposed sampling approach was visualized and manually inspected to determine if it led to satisfactory fractal geometry. While this method guaranteed contractivity, it did not, by itself, yield universally well-formed fractals, as illustrated in Figure 8.1.



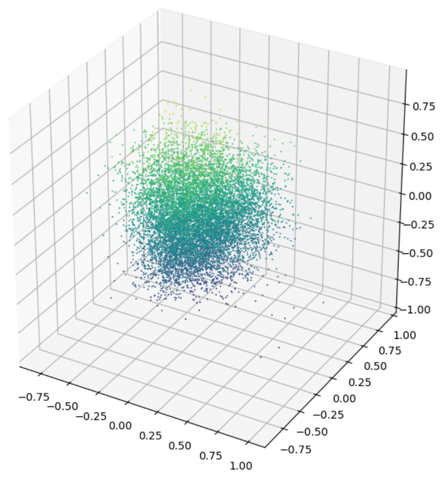
(a) Unconstrained σ_1 to σ_3 ratio.



(b) Unconstrained σ_1 to σ_3 ratio.



(c) Constrained σ_1 to σ_3 ratio.



(d) Constrained σ_1 to σ_3 ratio.

Figure 8.1: Examples of fractals with and without constrained σ_1/σ_3 ratio. Top: Unconstrained systems yield anisotropic or degenerate shapes; Bottom: Constraining σ_1/σ_3 promotes isotropy but does not guarantee non-degenerate geometry.

Notably, the constraint $\frac{\sigma_1}{\sigma_3} \leq 1.5$ was applied to each function in the system, ensuring no function exceeded this anisotropy ratio. Empirical inspection confirmed that reducing this ratio produced fractals with more isotropic (well-balanced) geometry. The relevance of this ratio is further explored through its connection to the matrix condition number, discussed in the next section.

8.2 Condition Number

Formally, the condition number of a matrix A with respect to a given norm is defined as:

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|.$$

When the Euclidean norm (2-norm) is used, the condition number simplifies to the ratio of the largest to the smallest singular value of A:

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest and smallest singular values of A, respectively.
A high condition number indicates that the matrix is ill-conditioned, meaning that small perturbations in the input can lead to large changes in the output. This property is important for fractal geometry, as high condition numbers may result in numerically unstable or highly anisotropic transformations. Conversely, a low condition number suggests numerical stability and more isotropic, well-behaved fractal shapes.

It was observed that many samples exhibited pronounced anisotropy, with fractals tending to align along certain directions in 3D space, resulting in visually poor geometry. Because the condition number is determined by the singular values, controlling its upper bound provides a direct handle on the directionality of the generated fractal.
Through iterative empirical testing, constraining the condition number to $\frac{\sigma_1}{\sigma_3} \leq 1.5$ was found to reliably produce fractals with satisfactory isotropy and structural complexity. However, as shown in Figure 8.1, condition number alone is not sufficient. Other factors also play a role in the emergence of undesirable or degenerate shapes. This observation motivated a more nuanced exploration of fractal geometry using the α (sigma-factor) formula.

8.3 Alpha Formula

As discussed in Section 4.2, the alpha formula (or σ-factor) [7] provides a single quality measure for fractal geometry in 2D. It uses a weighted sum of singular values across all function in the IFS and was empirically found to be:

$$\alpha = \sum_{i=1}^{N} (\sigma_{i,1} + 2 \cdot \sigma_{i,2}) \tag{8.2}$$

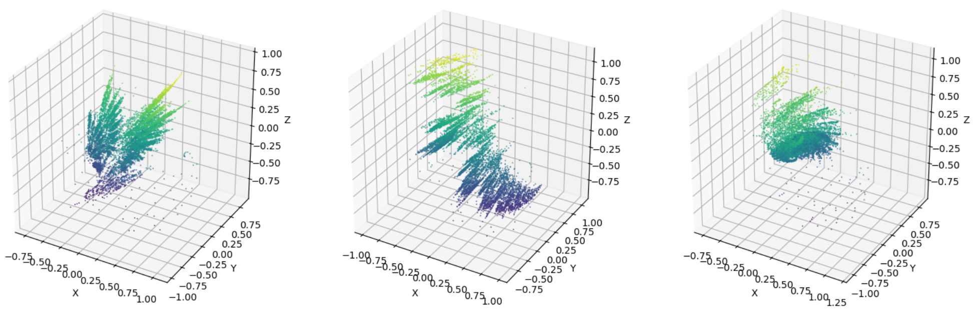
While effective in the 2D case, a direct extension to 3D is nontrivial, as 3D fractals involve three singular values per function.
A simple naive extension to 3D:

$$\alpha = \sum_{i=1}^{N} (\sigma_{i,1} + 2 \cdot \sigma_{i,2} + 3 \cdot \sigma_{i,3}) \tag{8.3}$$

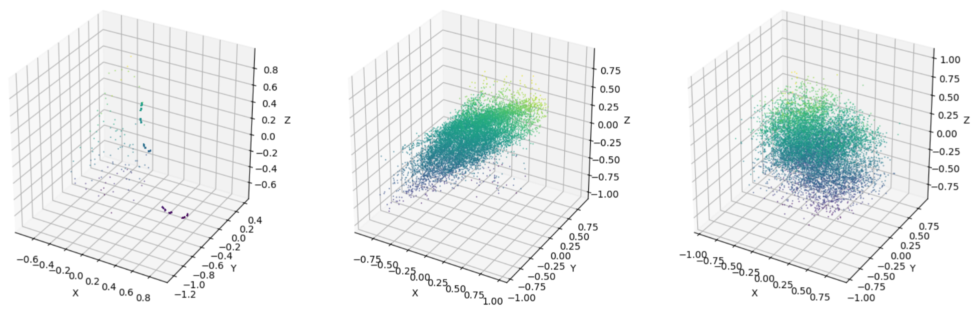
However, the same alpha range bounds as used in 2D ($\alpha_l = \frac{1}{2}(5+N)$, $\alpha_u = \frac{1}{2}(6+N)$) do not guarantee visually satisfying fractals in 3D. Therefore, the naive extension formula was abandoned. Both the sigma weights and the optimal alpha bounds require re-estimation, which led to the use of linear SVMs to empirically determine optimal weights.

8.4 SVM-based Weighting of Singular Values

Inspired by [7], a binary labeling process was undertaken, assigning "Good" or "Bad" geometry labels to fractals. A custom pipeline was developed for rapid annotation, producing a labeled dataset of 800 fractals across four system sizes ($N = 2, 4, 6, 8$). Examples of labeled "Good" and "Bad" fractals are shown in Figure 8.2.



(a) Examples of label "Good".



(b) Examples of label "Bad".

Figure 8.2: Visual examples of fractals labeled as "Good" (top) and "Bad" (bottom) for training the SVM classifier.

To learn the sigma weights, a linear SVM was trained for each system size by providing the concatenated singular values as input, using 10-fold cross-validation within the scikit-learn [76] framework. The SVM coefficients were interpreted as the relative importance of each singular value. Table 8.1 summarizes the mean coefficients across folds and system sizes.

Table 8.1: Mean learned SVM coefficients (c_1, c_2, c_3) for singular values $(\sigma_1, \sigma_2, \sigma_3)$ across IFS system sizes, indicating their importance for classifying fractal geometry.

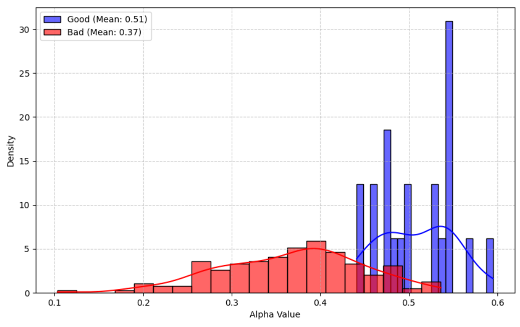
N	c_1	c_2	c_3
2	0.6867	1.2504	1.5885
4	-0.0406	0.1585	1.3995
6	-0.9884	-0.1982	0.2464
8	-0.3594	-0.4300	-0.7421
Average	**-0.1754**	**0.1952**	**0.6231**

The results indicate that σ_3 generally receives the largest absolute weight (most often positive), suggesting it is the most critical singular value for determining fractal quality across most system sizes. Notably, for $N = 8$, this trend does not hold as strongly, reflecting variability in the data or the potential influence of subjective labeling and limited sample size. The weights assigned by the SVM can be either positive or negative, with the absolute value indicating the strength of the contribution of each singular value to the decision boundary. This trend is visualized in Figure 8.3, where the right column (SVM weights) demonstrates improved separation between "Good" and "Bad" fractals compared to the left column (uniform weights).

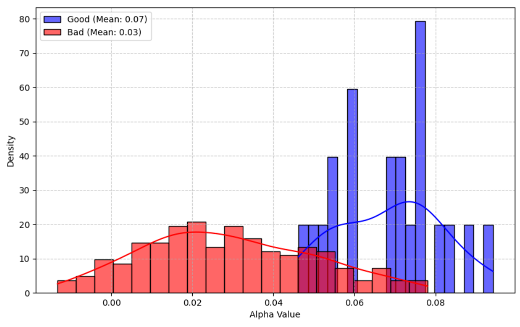
The two alpha formulas compared are:

$$\text{Uniform Weights:} \quad \alpha_{\text{uni}} = \sum_{i=1}^{N} (1 \cdot \sigma_{i,1} + 1 \cdot \sigma_{i,2} + 1 \cdot \sigma_{i,3}) \tag{8.4}$$

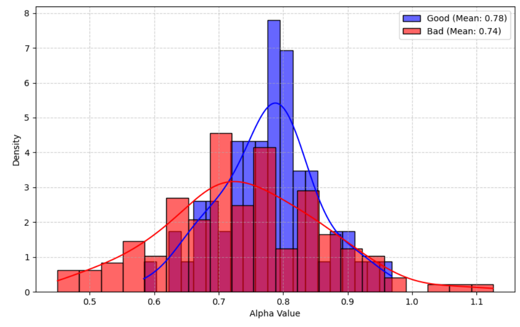
$$\text{SVM Weights:} \quad \alpha_{\text{svm}} = \sum_{i=1}^{N} (-0.1754 \cdot \sigma_{i,1} + 0.1952 \cdot \sigma_{i,2} + 0.6231 \cdot \sigma_{i,3}) \tag{8.5}$$



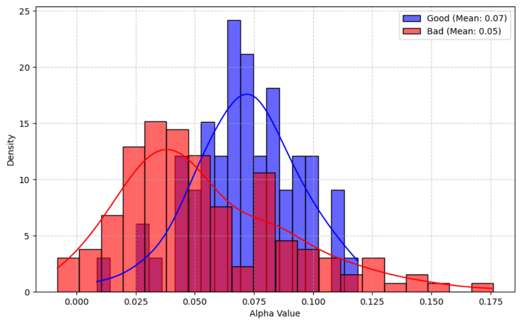
(a) N=2 (Uniform Weights)



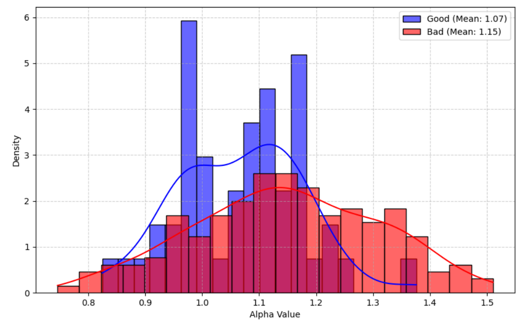
(b) N=2 (SVM Weights)



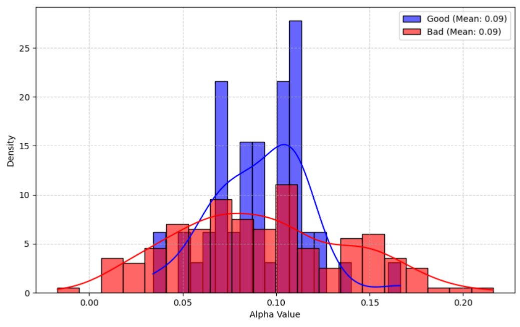
(c) N=4 (Uniform Weights)



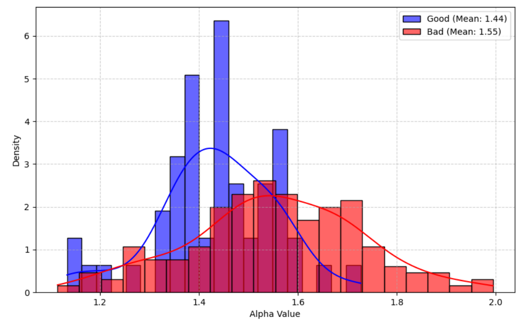
(d) N=4 (SVM Weights)



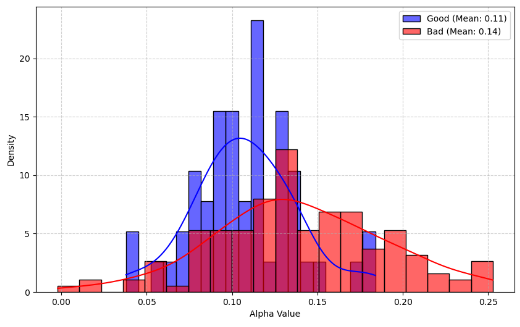
(e) N=6 (Uniform Weights)



(f) N=6 (SVM Weights)



(g) N=8 (Uniform Weights)



(h) N=8 (SVM Weights)

Figure 8.3: Comparison of alpha value distributions for "Good" and "Bad" fractals using uniform (left) versus SVM-learned (right) singular value weights for each IFS system size.

For system sizes $N = 2$ and $N = 4$, the SVM-based weighting yields notably better separation between good and bad examples, supporting the idea that more nuanced, data-driven weighting leads to better geometric control. For larger systems ($N = 6, 8$), the separation is less pronounced, possibly due to subjective labeling, limited sample size, or increased geometric complexity.
Overall, these findings suggest that a higher weight on σ_3 leads to improved geometric separability and that an optimal alpha value, informed by SVM-learned weights and combined with condition number constraints, can guide efficient sampling of well-behaved fractals.

The research presented in this chapter suggests that targeted control of fractal parameters, particularly through bounding the condition number and applying SVM-informed singular value weights, can enhance the quality and utility of generated 3D fractal datasets. While these geometric improvements appear promising, their full impact on synthetic pre-training and downstream transfer tasks remains to be systematically evaluated. These methods offer practical tools and a foundation for future work in 3D fractal-based data generation and action recognition.

9 Discussion

9.1 Interpretation of Experimental Results

The experimental results presented in this thesis demonstrate that pre-training action recognition models on large-scale synthetic datasets generated from 3D fractal geometry offers a significant advantage over training from scratch and narrows the performance gap with standard real-world pre-training (e.g., Kinetics-400). Systematic evaluation across multiple experimental configurations shows that, with careful dataset design and augmentation, 3D fractal pre-training yields robust and transferable representations for video action recognition.
A key finding is that the best downstream performance on both UCF101 and HMDB51 benchmarks was achieved with a "moderate" regime of geometric transformations and color augmentation. The moderate-moderate configuration provides the most consistent improvements, balancing the diversity of fractal video data with sufficient learnability. Notably, while higher levels of augmentation introduce more variability, they can in some cases reduce model performance, particularly when the augmentations distort the geometric structure or obscure informative patterns. This confirms that augmentation must be carefully calibrated to foster generalizable features without degrading the core structure of the data.

It is also evident that strong pre-training accuracy does not always guarantee optimal transfer to downstream tasks. For instance, the "no transformation" configuration achieves perfect accuracy on the pre-training task but fails to generalize to real-world datasets, highlighting the importance of meaningful dataset complexity over mere memorization. Conversely, models pre-trained with well-balanced fractal datasets showed both strong in-distribution and transfer learning performance, confirming the utility of the proposed synthetic data approach.
When compared with previous work on 2D fractal-based pre-training, the 3D fractal pipeline developed in this thesis does not yet surpass the best reported 2D results. Nonetheless, it still consistently and substantially outperforms models trained from scratch, underscoring the practical value of synthetic pre-training, especially where real data is scarce or privacy-restricted. The gap with 2D fractal results is discussed further below, with potential improvement strategies identified in the Future Work Section 9.4.

9.2 Insights from Dataset Generation

The characteristics of the generated fractal dataset were found to have a profound effect on learned representations and transferability:

- **Dataset Complexity and Diversity:** The results from the ablation study demonstrate that both the number of classes and the total number of video instances are strong drivers of model performance.

Increasing class diversity, in particular, yields greater improvements than simply increasing the number of instances per class, confirming the need for a broad and varied action space in synthetic datasets.

- **Effect of Augmentation:** Controlled augmentation, specifically moderate geometric transformations and color diversity, enables models to learn features that are robust to spatial and appearance variations. However, excessive augmentation can harm both in-distribution and transfer performance by introducing artificial variability that is not representative of the target domain.
- **Fractal Generation Parameters:** Attempts to improve the geometric quality of fractals, such as constraining the condition number or using SVM-informed singular value weights, have proven to be promising directions for further reducing the prevalence of degenerate or non-representative classes. While these geometric enhancements have not yet been exhaustively evaluated in the transfer learning pipeline, their impact on visual quality and sample diversity provides a strong foundation for subsequent work.

9.3 Limitations

Despite the encouraging results, several limitations must be acknowledged:

- **Computational Constraints:** The scope and scale of experiments were limited by available computational resources, particularly in generating and processing large 3D fractal datasets and running extensive ablation studies.
- **Dataset Size and Diversity:** While the synthetic datasets generated were large compared to many prior works, further increases in scale, both in the number of classes and instance count, could further enhance performance but were not exhaustively explored.
- **Augmentation Scope:** The augmentation regimes for transformation and color in this thesis were explored using a tri-level (low, moderate, high) approach, which, while systematic, may be overly simplistic. More granular or physics-inspired transformations (such as free fall, elastic collisions, or other motion models) were not implemented and could expand the diversity of synthetic actions in future work.
- **Model Architecture Choices:** The evaluation focused primarily on the ResNet-50 backbone with TSM. While this choice is well-motivated and consistent with prior synthetic pre-training work, it is possible that alternative architectures (e.g., video transformers) may benefit differently from 3D fractal pre-training.
- **Hyperparameter Sweep:** No comprehensive search for optimal hyperparameters was conducted, either for pre-training or fine-tuning. As a result, some of the chosen hyperparameter values may have been sub-optimal, potentially limiting overall model performance.
- **Limited Exploration of Alternative Representations:** The experiments primarily utilized rendered video frames from fractal point clouds. Other data representations, such as raw point cloud sequences or mesh-based videos, were not explored in depth due to time and implementation con-

straints.

- **Synthetic-Real Domain Gap:** Despite improvements, a domain gap remains between synthetic fractal videos and real-world human action videos. Bridging this gap remains an open challenge, particularly in capturing high-level semantics and dynamics present in real actions.
- **Subjectivity in Geometric Quality Assessment:** The identification of "good" versus "bad" fractal geometries relied on subjective visual inspection and manual labeling, which could introduce bias into SVM-guided quality experiments.
- **Fractal Geometry Improvements:** The effectiveness of SVM-learned singular value weights and the condition number constraint for improving fractal quality is currently limited by the amount of labeled data available for SVM training. It is possible that a larger and more diverse labeled dataset, as well as a more refined exploration of condition number thresholds, would yield better geometric sampling and higher downstream performance.
- **Class Similarity in FDSL:** While Formula-Driven Supervised Learning (FDSL) ensures perfect label correspondence to generation parameters, it does not guarantee that all sampled classes are visually distinct. Some classes may appear quite similar despite differing parameterizations, potentially hindering downstream discrimination [20].

9.4 Implications and Future Work

The findings of this thesis support the value of large-scale synthetic fractal datasets for pre-training action recognition models, especially when access to large real video datasets is limited or impractical. The demonstrated transfer gains, even relative to strong baselines, suggest several promising avenues for further research:

- **Improved Fractal Sampling and Filtering:** Future work should further refine the geometric quality of fractal datasets by expanding the use of condition number constraints and data-driven weighting (e.g., SVM-learned weights) to maximize the proportion of visually meaningful and diverse samples. Collecting a larger set of manually labeled fractal geometries would support the development of more robust SVM or other machine learning models for automated quality assessment, and a finer grid search over condition number thresholds may help optimize the balance between isotropy and variety.
- **Scaling Dataset Size:** Increasing both the number of action classes and the diversity of transformations may further boost performance, as indicated by the ablation study. Leveraging distributed or cloud-based resources may help overcome current computational bottlenecks.
- **Richer Augmentation Strategies:** Future research should move beyond the tri-level color and transformation augmentation regime, exploring a broader and more continuous spectrum of augmentation magnitudes. Incorporating physically-inspired transformations, such as free fall, elastic or inelastic collisions, or dynamics derived from simple physics simulations, could enrich the motion complexity in synthetic videos and better mimic real-world variability.

- **Alternative Data Representations:** Exploring alternative representations of fractal data, such as mesh sequences or direct point cloud inputs, could further enhance the utility of synthetic datasets and support evaluation with a wider range of neural architectures.
- **Cross-Domain Applications:** The pipeline and methodologies developed here may be extended to other vision tasks (e.g., object recognition, segmentation) or domains (e.g., medical imaging, animal behavior analysis) that face similar data scarcity challenges.
- **Integration with Domain Adaptation:** Combining fractal-based pre-training with domain adaptation or self-supervised learning techniques could help further bridge the synthetic-real gap and improve transferability.
- **Automated Assessment of Fractal Quality:** Developing objective, automated metrics for evaluating the quality and diversity of fractal geometry would enable more systematic filtering and sampling, reducing the reliance on subjective visual inspection.
- **Class Diversity Enforcement in FDSL:** To address the issue of visually similar classes, future work could explore automatic embedding of fractal point clouds into a learned feature space (e.g., via a shallow neural network), then use this representation to enforce a minimum separation between class prototypes when generating new classes. Such approaches could help ensure that each synthetic action class is not only parametrically unique but also visually distinct, supporting better downstream discrimination [20].
- **Exploring Other Fractal Types and Generative Models:** Expanding beyond the current IFS-based approach to other fractal models, or integrating with deep generative models, may yield new forms of complexity and transfer learning benefit.

10 Conclusion

This thesis set out to investigate whether synthetic data generated from 3D fractal geometry could serve as an effective foundation for pre-training action recognition models. Motivated by the limitations and costs associated with large-scale real video datasets, a comprehensive methodology was developed for the procedural generation, augmentation, and evaluation of fractal-based synthetic video datasets. Leveraging Iterated Function System (IFS) and controlled transformation pipelines, large and diverse datasets were created and used to pre-train neural networks, whose learned representations were then transferred to real-world benchmarks.

The experiments conducted demonstrated that 3D fractal pre-training substantially outperforms training from scratch and achieves competitive results relative to standard real-data pre-training, particularly in the context of privacy-sensitive or resource-limited settings. Key findings include:

- The choice and calibration of fractal transformation and color augmentation strategies significantly influence the learnability and transferability of synthetic datasets, with moderate levels proving most effective.
- Scaling up both the number of classes and instances per class in the fractal dataset leads to robust improvements in both pre-training and downstream action recognition performance.
- Improvements in fractal geometry, such as constraining condition numbers or employing data-driven singular value weighting, provide further avenues for enhancing dataset quality and downstream utility.
- While the 3D fractal approach did not surpass the highest-reported results for 2D fractal pre-training, it consistently outperformed models trained from scratch and provides a strong foundation for further research.

The research questions posed at the outset, concerning the utility of 3D fractal-based synthetic data for pre-training, and the effectiveness of formula-driven supervision in action recognition, have been affirmatively answered. The thesis demonstrates that with principled dataset design and augmentation, synthetic fractal data can meaningfully bridge the gap to large-scale annotated video datasets, enabling scalable and privacy-preserving pre-training.

Contributions of the thesis include:

- The development and validation of a full 3D fractal video generation and augmentation pipeline for action recognition research.
- A systematic evaluation of how dataset parameters and geometric properties influence representation learning and transfer performance.
- Introduction of new techniques for improving 3D fractal geometry using SVM-guided weighting and condition number constraints.
- An empirical comparison to prior 2D synthetic approaches and to conventional baselines.

In conclusion, this work highlights the potential of fractal-based synthetic data as a powerful, flexible, and privacy-friendly alternative for training deep learning models in computer vision. While challenges and opportunities for further improvement remain, the results underscore the promise of formula-driven synthetic data generation for advancing action recognition and related tasks.

Bibliography

[1] URL: https://www.google.com/trends).

[2] Made with perlinNoiseMaker. URL: http://kitfox.com/projects/perlinNoiseMaker/.

[3] URL: https://hpc.aau.dk/.

[4] URL: https://github.com/mit-han-lab/temporal-shift-module.

[5] URL: https://matplotlib.org/stable/users/explain/colors/colormaps.html.

[6] Ali K. AlShami et al. "SMART-vision: survey of modern action recognition techniques in vision". In: *Multimedia Tools and Applications* (2024). ISSN: 1573-7721. DOI: 10.1007/s11042-024-20484-5. URL: http://dx.doi.org/10.1007/s11042-024-20484-5.

[7] Connor Anderson and Ryan Farrell. *Improving Fractal Pre-training*. 2021. arXiv: 2110.03091 [cs.CV]. URL: https://arxiv.org/abs/2110.03091.

[8] Gopalakrishnan Arjunan. "Fractal-Based AI: Exploring Self-Similarity in Neural Networks for Improved Pattern Recognition". In: *International Journal of Innovative Science and Research Technology* 9 (Nov. 2024). DOI: 10.38124/ijisrt/IJISRT24NOV823.

[9] Anurag Arnab et al. "ViViT: A Video Vision Transformer". In: (2021). arXiv: 2103.15691 [cs.CV]. URL: https://arxiv.org/abs/2103.15691.

[10] Michael F. Barnsley. *Fractals Everywhere*. 2nd. Academic Press Professional, 1993.

[11] André Bauer et al. "Comprehensive Exploration of Synthetic Data Generation: A Survey". In: (2024). arXiv: 2401.02524 [cs.LG]. URL: https://arxiv.org/abs/2401.02524.

[12] Wolfgang Beyer. *Mandelbrot Set with Continuously Colored Environment*. https://commons.wikimedia.org/wiki/File:Mandel_zoom_00_mandelbrot_set.jpg. Created with Ultra Fractal 3, licensed under CC BY-SA 3.0. 2005.

[13] Amlaan Bhoi. "Spatio-temporal Action Recognition: A Survey". In: *arXiv preprint arXiv:1901.09403* (2019).

[14] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[15] Fernando Camarena et al. "Action recognition by key trajectories". In: *Pattern Analysis and Applications* 25.2 (2022), pp. 409–423. DOI: 10.1007/s10044-021-01054-z. URL: https://doi.org/10.1007/s10044-021-01054-z.

[16] Joao Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: (2018). arXiv: 1705.07750 [cs.CV]. URL: https://arxiv.org/abs/1705.07750.

[17] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG]. URL: https://arxiv.org/abs/2002.05709.

[18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation, 2018. URL: http://www.blender.org.

[19] Ekin D. Cubuk et al. *RandAugment: Practical automated data augmentation with a reduced search space*. 2019. arXiv: 1909.13719 [cs.CV]. URL: https://arxiv.org/abs/1909.13719.

[20] Yin Cui et al. *Measuring Dataset Granularity*. 2019. arXiv: 1912.10154 [cs.CV]. URL: https://arxiv.org/abs/1912.10154.

[21] Ashley S Dale et al. "Mind the (domain) gap: Metrics for the differences in synthetic and real data distributions". In: *Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*. Vol. 13035. SPIE, 2024, 130350Z. DOI: 10.1117/12.3015657.

[22] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: (2009), pp. 248–255.

[23] Yale Mathematics Department. *Nature and Fractals*. n.d. URL: https://gauss.math.yale.edu/fractals/Panorama/Nature/NatFracGallery/NatFracGallery.html.

[24] Abhyuday Desai et al. "TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation". In: (2021). arXiv: 2111.08095 [cs.LG]. URL: https://arxiv.org/abs/2111.08095.

[25] Maxime Devanne et al. "3-D Human Action Recognition by Shape Analysis of Motion Trajectories on Riemannian Manifold". In: *IEEE Transactions on Cybernetics* 45.7 (2015), pp. 1340–1352. DOI: 10.1109/TCYB.2014.2350774.

[26] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. *Unsupervised Visual Representation Learning by Context Prediction*. 2016. arXiv: 1505.05192 [cs.CV]. URL: https://arxiv.org/abs/1505.05192.

[27] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: (2017), pp. 1–16.

[28] Adrien Douady, John Hamal Hubbard, and P Lavaurs. "Etude dynamique des polynômes complexes". In: (1984).

[29] Epic Games. *Unreal Engine*. Version 4.22.1. Apr. 25, 2019. URL: https://www.unrealengine.com.

[30] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). May 4, 2016. URL: https://data.europa.eu/eli/reg/2016/679/oj (visited on 04/13/2023).

[31] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. 3rd. John Wiley Sons, 2014.

[32] Christoph Feichtenhofer et al. *SlowFast Networks for Video Recognition*. 2019. arXiv: 1812.03982 [cs.CV]. URL: https://arxiv.org/abs/1812.03982.

[33] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: (2022). arXiv: 1811.12231 [cs.CV]. URL: https://arxiv.org/abs/1811.12231.

[34] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: https://arxiv.org/abs/1311.2524.

[35] GitHub and OpenAI. *GitHub Copilot*. https://github.com/features/copilot. Accessed: 2025-06-04. 2021.

[36] Tilmann Gneiting and Adrian E Raftery and. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (2007), pp. 359–378. DOI: 10.1198/016214506000001437. eprint: https://doi.org/10.1198/016214506000001437. URL: https://doi.org/10.1198/016214506000001437.

[37] Google. *Gemini*. 2025. URL: https://gemini.google.com/.

[38] Raghav Goyal et al. *The "something something" video database for learning and evaluating visual common sense*. 2017. arXiv: 1706.04261 [cs.CV]. URL: https://arxiv.org/abs/1706.04261.

[39] Shubhaankar Gupta, Thomas P. O'Connell, and Bernhard Egger. *Beyond Flatland: Pre-training with a Strong 3D Inductive Bias*. 2021. arXiv: 2112.00113 [cs.CV]. URL: https://arxiv.org/abs/2112.00113.

[40] Abdul Mueed Hafiz, Mahmoud Hassaballah, and Adel Binbusayyis. "Formula-Driven Supervised Learning in Computer Vision: A Literature Survey". In: *Applied Sciences* 13.2 (2023). ISSN: 2076-3417. DOI: 10.3390/app13020723. URL: https://www.mdpi.com/2076-3417/13/2/723.

[41] Tengda Han, Weidi Xie, and Andrew Zisserman. "Video Representation Learning by Dense Predictive Coding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019. URL: https://www.robots.ox.ac.uk/~vgg/publications/2019/Han19/han19.pdf.

[42] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[43] Felix Hausdorff. *Grundzüge der Mengenlehre*. Leipzig: Veit & Comp., 1914.

[44] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[45] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV]. URL: https://arxiv.org/abs/1911.05722.

[46] JOHN E. HUTCHINSON. “Fractals and Self Similarity”. In: *Indiana University Mathematics Journal* 30.5 (1981), pp. 713–747. ISSN: 00222518, 19435258. URL: http://www.jstor.org/stable/24893080 (visited on 05/23/2025).

[47] Yuchi Ishikawa, Tatsuya Komatsu, and Yoshimitsu Aoki. “Pre-training with Synthetic Patterns for Audio”. In: *arXiv preprint arXiv:2410.00511* (2024). URL: https://arxiv.org/abs/2410.00511.

[48] Simon Jenni, Givi Meishvili, and Paolo Favaro. “Video Representation Learning by Recognizing Temporal Transformations”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020. URL: https://arxiv.org/abs/2007.10730.

[49] Heinrich Jiang and Ofir Nachum. “Identifying and Correcting Label Bias in Machine Learning”. In: (2019). arXiv: 1901.04966 [cs.LG]. URL: https://arxiv.org/abs/1901.04966.

[50] Misha Karim et al. “Human Action Recognition Systems: A Review of the Trends and State-of-the-Art”. In: *IEEE Access* 12 (2024), pp. 36372–36390. DOI: 10.1109/ACCESS.2024.3373199.

[51] Ondřej Karlík. *Power 8 Mandelbulb Fractal Overview*. https://commons.wikimedia.org/wiki/File:Power_8_mandelbulb_fractal_overview.jpg. Rendered with Corona ray tracer, licensed under CC BY-SA 3.0. 2011.

[52] Hirokatsu Kataoka et al. “Pre-training without Natural Images”. In: (2021). arXiv: 2101.08515 [cs.CV]. URL: https://arxiv.org/abs/2101.08515.

[53] Hirokatsu Kataoka et al. *Replacing Labeled Real-image Datasets with Auto-generated Contours*. 2022. arXiv: 2206.09132 [cs.CV]. URL: https://arxiv.org/abs/2206.09132.

[54] Will Kay et al. “The Kinetics Human Action Video Dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[55] Sahil Khose et al. *SkyScenes: A Synthetic Dataset for Aerial Scene Understanding*. 2024. arXiv: 2312.06719 [cs.CV]. URL: https://arxiv.org/abs/2312.06719.

[56] David Kinderlehrer and Guido Stampacchia. *An Introduction to Variational Inequalities and Their Applications*. Society for Industrial and Applied Mathematics, 2000. DOI: 10.1137/1.9780898719451. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898719451. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719451.

[57] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. *Do Better ImageNet Models Transfer Better?* 2019. arXiv: 1805.08974 [cs.CV]. URL: https://arxiv.org/abs/1805.08974.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: 25 (2012). Ed. by F. Pereira et al. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[59] H. Kuehne et al. “HMDB: a large video database for human motion recognition”. In: (2011).

[60] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6.

[61] Laptev and Lindeberg. "Space-time interest points". In: (2003), 432–439 vol.1. DOI: 10.1109/ICCV.2003.1238378.

[62] A. B. Lee, D. Mumford, and J. Huang. "Occlusion Models for Natural Images: A Statistical Study of a Scale-Invariant Dead Leaves Model". In: *International Journal of Computer Vision* 41.1 (2001), pp. 35–59. DOI: 10.1023/A:1011109015675. URL: https://doi.org/10.1023/A:1011109015675.

[63] Ji Lin, Chuang Gan, and Song Han. "TSM: Temporal Shift Module for Efficient Video Understanding". In: (2019). arXiv: 1811.08383 [cs.CV]. URL: https://arxiv.org/abs/1811.08383.

[64] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: (2015). arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[65] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[66] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG]. URL: https://arxiv.org/abs/1608.03983.

[67] Yingzhou Lu et al. *Machine Learning for Synthetic Data Generation: A Review*. 2025. arXiv: 2302.04062 [cs.LG]. URL: https://arxiv.org/abs/2302.04062.

[68] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. *Shuffle and Learn: Unsupervised Learning using Temporal Order Verification*. 2016. arXiv: 1603.08561 [cs.CV]. URL: https://arxiv.org/abs/1603.08561.

[69] Md Golam Morshed et al. "Human Action Recognition: A Taxonomy-Based Survey, Updates, and Opportunities". In: *Sensors* 23.4 (2023). ISSN: 1424-8220. DOI: 10.3390/s23042182. URL: https://www.mdpi.com/1424-8220/23/4/2182.

[70] Ryosuke Nakamura, Ryosuke Tadokoro, et al. "Scaling Backwards: Minimal Synthetic Pre-training?" In: *European Conference on Computer Vision (ECCV)*. 2024. DOI: 10.1007/978-3-031-25080-4_23. URL: https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/02315.pdf.

[71] Sergey I. Nikolenko. "Synthetic Data for Deep Learning". In: (2019). arXiv: 1909.11512 [cs.LG]. URL: https://arxiv.org/abs/1909.11512.

[72] OpenAI. *ChatGPT 4.1*. [Large language model]. URL: https://chat.openai.com/chat.

[73] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[74] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training Recurrent Neural Networks”. In: (2013). arXiv: 1211.5063 [cs.LG]. URL: https://arxiv.org/abs/1211.5063.

[75] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703.

[76] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[77] Ken Perlin. “An image synthesizer”. In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), 287–296. ISSN: 0097-8930. DOI: 10.1145/325165.325247. URL: https://doi.org/10.1145/325165.325247.

[78] Ronald Poppe. “A survey on vision-based human action recognition”. In: *Image and Vision Computing* 28.6 (2010), pp. 976–990.

[79] Rui Qian et al. “Spatiotemporal Contrastive Video Representation Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 6960–6970. DOI: 10.1109/CVPR46437.2021.00689. URL: https://arxiv.org/abs/2008.03800.

[80] Walter Rudin. *Principles of Mathematical Analysis*. 3rd. New York: McGraw-Hill, 1976. ISBN: 0-07-054235-X.

[81] Adel Saleh et al. “Analysis of Temporal Coherence in Videos for Action Recognition”. In: *International Conference on Image Analysis and Recognition*. Springer. 2016, pp. 325–332.

[82] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[83] Javier Selva et al. “Video Transformers: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.11 (2023), pp. 12922–12943. DOI: 10.1109/TPAMI.2023.3243465.

[84] Amir Shahroudy et al. *NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis*. 2016. arXiv: 1604.02808 [cs.CV]. URL: https://arxiv.org/abs/1604.02808.

[85] Karen Simonyan and Andrew Zisserman. “Two-Stream Convolutional Networks for Action Recognition in Videos”. In: (2014). arXiv: 1406.2199 [cs.CV]. URL: https://arxiv.org/abs/1406.2199.

[86] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV]. URL: https://arxiv.org/abs/1212.0402.

[87] Gilbert Strang. *Introduction to Linear Algebra*. 5th. Wellesley-Cambridge Press, 2016.

[88] Davyd Svyezhentsev, George Retsinas, and Petros Maragos. *Pre-training for Action Recognition with Automatically Generated Fractal Datasets*. 2024. arXiv: 2411.17584 [cs.CV]. URL: https://arxiv.org/abs/2411.17584.

[89] Sora Takashima et al. "Visual Atoms: Pre-training Vision Transformers with Sinusoidal Waves". In: (2023). arXiv: 2303.01112 [cs.CV]. URL: https://arxiv.org/abs/2303.01112.

[90] Josh Tobin et al. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: (2017). arXiv: 1703.06907 [cs.RO]. URL: https://arxiv.org/abs/1703.06907.

[91] Du Tran et al. "Learning Spatiotemporal Features with 3D Convolutional Networks". In: (2015). arXiv: 1412.0767 [cs.CV]. URL: https://arxiv.org/abs/1412.0767.

[92] Jonathan Tremblay et al. "Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.

[93] Unity Technologies. *Unity*. Version 2023.2.3. Game development platform. 2023. URL: https://unity.com/.

[94] Gül Varol et al. "Learning from Synthetic Humans". In: (2017).

[95] Limin Wang et al. "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition". In: (2016). arXiv: 1608.00859 [cs.CV]. URL: https://arxiv.org/abs/1608.00859.

[96] Kun Xia, Jianguang Huang, and Hanyu Wang. "LSTM-CNN Architecture for Human Activity Recognition". In: *IEEE Access* 8 (2020), pp. 56855–56866. DOI: 10.1109/ACCESS.2020.2982225.

[97] Ryosuke Yamada et al. "Formula-Supervised Visual-Geometric Pre-training". In: (2024). arXiv: 2409.13535 [cs.CV]. URL: https://arxiv.org/abs/2409.13535.

[98] Ryosuke Yamada et al. "MV-FractalDB: Formula-driven Supervised Learning for Multi-view Image Recognition". In: (2021), pp. 2076–2083. DOI: 10.1109/IROS51168.2021.9635946.

[99] Ryosuke Yamada et al. "Point Cloud Pre-training with Natural 3D Structures". In: (2022), pp. 21251–21261. DOI: 10.1109/CVPR52688.2022.02060.

[100] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. "Recognizing actions using depth motion maps-based histograms of oriented gradients". In: Association for Computing Machinery, 2012, 1057–1060. ISBN: 9781450310895. DOI: 10.1145/2393347.2396382. URL: https://doi.org/10.1145/2393347.2396382.

[101] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: 27 (2014), pp. 3320–3328.

[102] Richard Zhang, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization". In: (2016). arXiv: 1603.08511 [cs.CV]. URL: https://arxiv.org/abs/1603.08511.

[103] Yuxuan Zhang et al. “DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort”. In: (2021).

[104] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”. In: (2020), pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.

[105] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. *Open3D: A Modern Library for 3D Data Processing*. 2018. arXiv: 1801.09847 [cs.CV]. URL: https://arxiv.org/abs/1801.09847.

[106] Yi Zhu et al. “A Comprehensive Study of Deep Video Action Recognition”. In: (2020). arXiv: 2012.06567 [cs.CV]. URL: https://arxiv.org/abs/2012.06567.