National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Pre-training for Video Action Recognition with Automatically Generated Datasets

Diploma Thesis

by

Davyd Svyezhentsev

Supervisor: Petros Maragos
Professor NTUA

Co-supervisor George Retsinas
Postdoctoral Researcher NTUA

Computer Vision, Speech Communication and Signal Processing Group

Athens, July 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

Προ-εκπαίδευση για Αναγώριση Δράσης σε Βίντεο με Συνθετικά Σύνολα Δεδομένων

Διπλωματική Εργασία

του

Νταβίντ Σβέζεντσεβ

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π

Συνεπιβλέπων Γιώργος Ρετσινάς
Μεταδιδακτορικός Ερευνητής Ε.Μ.Π

Εργαστηριο Ορασης Υπολογιστων, Επικοινωνιας, Λογου και Επεξεργασιας Σηματων

Αθήνα, Ιούλιος 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

Προ-εκπαίδευση για Αναγώριση Δράσης σε Βίντεο με Συνθετικά Σύνολα Δεδομένων

Διπλωματική Εργασία

του

Νταβίντ Σβέζεντσεβ

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π

Συνεπιβλέπων Γιώργος Ρετσινάς
Μεταδιδακτορικός Ερευνητής Ε.Μ.Π

Εγκρίθηκε από την κάτωθι τριμελή επιτροπή την 12η Ιουλίου 2023.

.....	.....	.....
Πέτρος Μαραγκός	Αθανάσιος Ροντογιάννης	Γεράσιμος Ποταμιάνος
Καθηγητής Ε.Μ.Π	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Παν/μιο Θεσσαλίας

Εργαστηριο Ορασης Υπολογιστων, Επικοινωνιας, Λογου και Επεξεργασιας Σηματων

Αθήνα, Ιούλιος 2023

. . . . . .

Νταβίντ Σβέζεντσεβ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Abstract

In recent years, the computer vision community has exhibited growing interest in synthetic data. For the image modality, existing work has proposed learning visual representations by pre-training with synthetic samples produced by various generative processes instead of real data. Such an approach is advantageous as it resolves issues associated with real data: collection and labeling costs, copyright, privacy and human bias. Desirable properties of synthetic images have been carefully investigated and as a result the gap in performance between real and synthetic images has been alleviated significantly.

The present work extends the aforementioned approach to the domain of video and applies it to the task of action recognition. Due to the addition of the temporal dimension, this modality is notably more complex than images. As such, employing fractal geometry and other generative processes, we present methods to automatically produce large-scale datasets of short synthetic video clips. This approach is applicable for both supervised and self-supervised learning. To narrow the domain gap, we manually observe real video samples and identify their key properties such as periodic motion, random background, camera displacement etc. These properties are then carefully emulated during pre-training. Through thorough ablations, we determine the properties that strengthen downstream results and offer general guidelines for pre-training with synthetic videos. The proposed approach is evaluated on small-scale action recognition datasets HMDB51 and UCF101 as well as four other video benchmarks. Compared to standard Kinetics pre-training, our reported results come close and are even superior on a portion of benchmarks.

Keywords - Computer Vision, Deep Learning, Convolutional Neural Networks, Action Recognition, Synthetic Data, Domain Adaptation, Fractal Geometry

Περίληψη

Τα τελευταία χρόνια, η κοινότητα της όρασης υπολογιστών έχει εκδηλώσει αυξανόμενο ενδιαφέρον για τα συνθετικά δεδομένα. Για εικόνες, υπάρχουσες εργασίες πρότειναν την εκμάθηση οπτικών αναπαραστάσεων όχι με πραγματικά δεδομένα αλλά μέσω προ-εκπαίδευσης με συνθετικά δείγματα που παράγονται από διάφορες γενετικές διαδικασίες. Μια τέτοια προσέγγιση είναι επωφελής καθώς επιλύει ζητήματα που σχετίζονται με πραγματικά δεδομένα: κόστος συλλογής και επισήμανσης, πνευματικά δικαιώματα, ιδιωτικότητα και ανθρώπινη προκατάληψη. Οι επιθυμητές ιδιότητες των συνθετικών εικόνων έχουν διερευνηθεί προσεκτικά και ως αποτέλεσμα το χάσμα στην απόδοση μεταξύ πραγματικών και συνθετικών δεδομένων έχει μειωθεί σημαντικά.

Η παρούσα εργασία επεκτείνει την προαναφερθείσα προσέγγιση σε αρχεία βίντεο και την εφαρμόζει στο πρόβλημα της αναγνώρισης δράσης. Λόγω της επιπρόσθετης χρονικής διάστασης, αυτά τα αρχεία είναι σαφώς πιο περίπλοκα σε σχέση με εικόνες. Ως εκ τούτου, χρησιμοποιώντας γεωμετρία φράκταλ και άλλες γενετικές διαδικασίες, παρουσιάζουμε μεθόδους για αυτόματη παραγωγή συνόλων δεδομένων μεγάλης κλίμακας από σύντομα συνθετικά βίντεο κλιπ. Η μέθοδος αυτή είναι εφαρμόσιμη τόσο για επιβλεπόμενη όσο και για αυτο-επιβλεπόμενη μάθηση. Για να γεφυρώσουμε το χάσμα μεταξύ πραγματικών και συνθετικών δεδομένων, προσδιορίζουμε χειροκίνητα τις βασικές ιδιότητες πραγματικών βίντεο όπως περιοδική κίνηση, τυχαίο φόντο, μετατόπιση κάμερας κτλ. Αυτές οι ιδιότητες προσομοιώνονται προσεκτικά κατά τη διάρκεια της προ-εκπαίδευσης μέσω απλών μετασχηματισμών. Μέσα από διεξοδικά πειράματα, προσδιορίζουμε τις ιδιότητες που ενισχύουν αποτελέσματα σε πραγματικά βίντεο και προτείνουμε γενικές οδηγίες για προ-εκπαίδευση με συνθετικά δεδομένα. Η προσέγγισή μας αξιολογείται σε σύνολα δεδομένων αναγνώρισης δράσης μικρής κλίμακας HMDB51 και UCF101 καθώς και σε τέσσερεις άλλες βάσεις βίντεο. Τα αποτελέσματα μας πλησιάζουν αρκετά την καθιερωμένη προ-εκπαίδευση με Kinetics και μάλιστα την υπερτερούν σε ένα μέρος των εξεταστέων βάσεων.

Keywords - Όραση Υπολογιστών, Μηχανική Μάθηση, Συνελικτικά Νευρωνικά Δίκτυα, Αναγνώριση Δράσης, Συνθετικά Δεδομένα, Domain Adaptation, Γεωμετρία Φράκταλ

Acknowledgements

I would like to thank Professor Petros Maragos, Postdoctoral Researcher George Retsinas, my family and my friends.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Πέτρο Μαραγκό, τον μεταδιδακτορικό ερευνητή κ. Γιώργο Ρετσινά, την οικογένεια μου και τους φίλους μου.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

BYOL Bootstrap Your Own Latent

CE Cross Entropy

CNN Convolutional Neural Network

GAN Generative Adversarial Network

GD Gradient Descent

IFS Iterated Function System

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MLP Multilayer Perceptron

MoCov2 Momentum Contrast Version 2

MSE Mean Squared Error

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SSL Semi-Supervised Learning

SVM Support Vector Machine

TSM Temporal Shift Module

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Εικόνες Φράκταλ

1.1.1 Εισαγωγή

Αν και επίκεντρο της παρούσας εργασίας είναι τα αρχεία βίντεο, είναι απαραίτητο πρώτα να μελετηθεί η βασικότερη έννοια της εικόνας. Εξετάζοντας σχετική βιβλιογραφία, μπορεί κανείς να ανακαλύψει αρκετές γενετικές διαδικασίες που παράγουν συνθετικές εικόνες. Ωστόσο, μόνο μία επιλέγεται ως βάση της παρούσας εργασίας: φράκταλ εικόνες που παράγονται μέσω της τεχνικής Iterated Function Systems (IFS). Αυτή η επιλογή δεν είναι αυθαίρετη αλλά στηρίζεται σε κρίσιμους παράγοντες:

- Τα IFS φράκταλ συχνά αναπαράγουν μοτίβα παρόντα σε φυσικές εικόνες. Αυτή η ιδιότητα αναμένεται να περιορίσει το χάσμα μεταξύ συνθετικών και πραγματικών εικόνων.
- Ο αλγόριθμος που παράγει αυτές τις εικόνες είναι εύκολος στην υλοποίηση.
- Με τυχαία δειγματοληψία παραμέτρων, είναι δυνατή η παραγωγή σχεδόν άπλετης ποσότητας εικόνων. Ωστόσο, όπως θα φανεί αργότερα, ορισμένοι περιορισμοί πρέπει να ενσωματωθούν στη διαδικασία λήψης δειγμάτων προκειμένου να αποφευχθούν ελαττωματικά αποτελέσματα.
- Αυτές οι εικόνες έχουν ήδη χρησιμοποιηθεί με επιτυχία στον τομέα της βαθιάς μάθησης [49, 76, 2]. Ως εκ τούτου, αναμένονται θετικά αποτελέσματα και για το επίκεντρο της παρούσας διπλωματικής εργασίας, τα αρχεία βίντεο.

1.1.2 Κλασσικά Iterated Function Systems

Ένα δισδιάστατο Iterated Function System (IFS) μπορεί να οριστεί ως ένα σύνολο n συναρτήσεων $F_i : \mathbb{R}^2 \mapsto \mathbb{R}^2$. Κάθε IFS συνδέεται με έναν ελκυστή, ένα σύνολο $S \in \mathbb{R}^2$ (και

ως εκ τούτου μια εικόνα) που είναι η λύση του συστήματος. S είναι το σταθερό σημείο της αναδρομικής εξίσωσης του Hutchinson [41]:

$$S = \bigcup_{i=1}^{n} F_i(S)$$

Σύμφωνα με τον Barnsley [5], οι προαναφερθείσες συναρτήσεις είναι γραμμικοί μετασχηματισμοί που ορίζονται από πίνακα $A_i \in \mathbb{R}^{2\times 2}$ και διάνυσμα $b_i \in \mathbb{R}^2$:

$$F_i(\boldsymbol{x}; A_i, b_i) = A_i\boldsymbol{x} + b_i = \begin{bmatrix} a_i & b_i \\ d_i & e_i \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} c_i \\ f_i \end{bmatrix}$$

Στη συνέχεια αυτού του εγγράφου, οι παράμετροι IFS θα αναφέρονται ως $W_i \in \mathbb{R}^6$ ή $W \in \mathbb{R}^{N\times 6}$.

Ο ελκυστής ενός δεδομένου IFS μπορεί να απεικονιστεί προσεγγιστικά χρησιμοποιώντας την διαδικασία chaos game, η οποία περιγράφεται στον Αλγόριθμο 1. Ως πρώτο βήμα, αυτός ο αλγόριθμος γεμίζει την εικόνα εξόδου με μηδενικά και επιλέγει τυχαίο αρχικό σημείο στον δισδιάστατο χώρο. Σε κάθε επανάληψη επιλέγεται μια τυχαία συνάρτηση από το IFS η οποία στη συνέχεια εφαρμόζεται στο προαναφερθέν σημείο. Η δειγματοληψία δεν είναι ομοιόμορφη αλλά βασίζεται στις κανονικοποιημένες απόλυτες ορίζουσες των πινάκων με βάρη. Οι συντεταγμένες του σημείου στη συνέχεια κβαντοποιούνται σε εικονοστοιχείο. Η τιμή του εικονοστοιχείου αυξάνεται κατά ένα. Το προηγούμενο βήμα δεν πραγματοποιείται για τις πρώτες K_S επαναλήψεις, καθώς οι προκύπτουσες συντεταγμένες μπορεί να μην ανήκουν στον ελκυστή. Μετά την ολοκλήρωση των K_I επαναλήψεων, η εικόνα, η οποία σε αυτό το σημείο είναι ένα δισδιάστατο ιστόγραμμα, κανονικοποιείται ώστε να παραχθεί αισθητικά ικανοποιητικό αποτέλεσμα. Παραδείγματα εικόνων IFS φράκταλ φαίνονται στο Σχήμα 1.1.

1.1.3 Παράμετροι IFS

Ακολουθώντας το έργο των [2], ένας διαφορετικός φορμαλισμός για τις παραμέτρους των φράκταλ μπορεί να προκύψει με ανάλυση πίνακα σε ιδιάζουσες τιμές:

$$A = U\Sigma V^T$$

- $U, V \in \mathbb{R}^{2\times 2}$ είναι ορθογώνιοι πίνακες και επομένως μπορούν να αναπαρασταθούν ως πίνακες περιστροφής με πιθανή αντανάκλαση. Η ορίζουσα μπορεί να είναι ± 1. Έστω $U = R_{\theta_1} D_1$ και $V^T = R_{\theta_2} D_2$ όπου R_x είναι πίνακας περιστροφής παραμετροποιημένος από γωνία x και D_i διαγώνιος πίνακας με διαγώνια στοιχεία $d_1, d_2 \in \{-1, 1\}$.

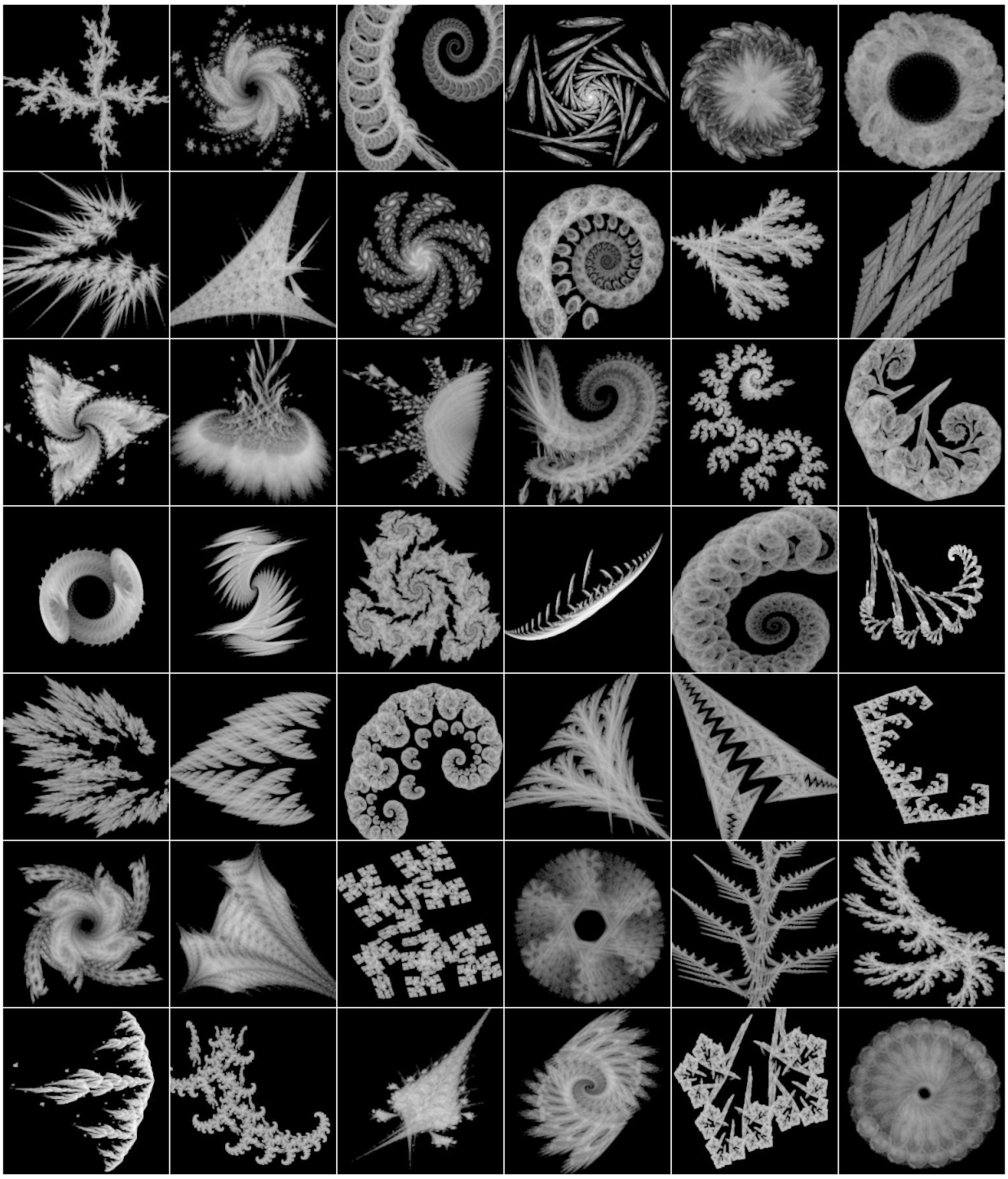


Figure 1.1: Παραδείγματα εικόνων IFS φράκταλ.

Algorithm 1 `chaos-game`(F, K_I, K_S, H, W): Αλγόριθμος απεικόνισης για IFS φράκταλ.

Input 1: N functions $\boldsymbol{F_i} : \mathbb{R}^2 \mapsto \mathbb{R}^2$
Input 2: Number of iterations and skipped iterations $\boldsymbol{K_I}, \boldsymbol{K_S}$
Input 3: Image dimensions $\boldsymbol{H}, \boldsymbol{W}$
Output: Grayscale Image $\boldsymbol{O} \in [0, 1]^{H \times W}$

1: **Initialize:** $O \leftarrow$ `zeros`(H, W)
2: $p_i \leftarrow \frac{|det(W_i)|}{\sum_{j=1}^{N} |det(W_j)|}$ ▷ Compute a probability for each W_i
3: Sample $x \sim U(-1, 1)$ ▷ Initialize random starting point $x \in \mathbb{R}^2$
4: **for** $step = 1$ to K_I **do**
5: `Sample` $F^* \sim P$ ▷ Sample random transformation
6: $x \leftarrow F^*(x)$ ▷ Apply transformation
7: **if** $step \geq K_S$ **then** ▷ Ignore the first K_S steps
8: $h_q, w_q \leftarrow quantize(x, H, W)$
9: $O[h_q, w_q] \leftarrow O[h_q, w_q] + 1$
10: **end if**
11: **end for**
12: $O \leftarrow log(O + 1)$
13: $O \leftarrow O/max(O)$ ▷ Normalize the image inside [0, 1]
14: **return** O

- $\Sigma \in \mathbb{R}^{2\times2}$ είναι διαγώνιος πίνακας που περιέχει τις ιδιάζουσες τιμές $\sigma_1, \sigma_2 \in \mathbb{R}^+$ με $\sigma_1 > \sigma_2$.

Συνεπώς, ο πίνακας A μπορεί να εκφραστεί ως:

$$A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$$

Για τυχαία δειγματοληψία του πίνακα A, είναι απαραίτητη η κατάλληλη δειγματοληψία των παραμέτρων $\{\theta_1, \theta_2, \sigma_1, \sigma_2, d_1^1, d_1^2, d_2^1, d_2^2\}$. Επιπλέον, οι ενδιάμεσοι πίνακες πρέπει να πολλαπλασιαστούν μεταξύ τους όπως αναγράφεται στις προηγούμενες εξισώσεις.

1.1.4 Fractal Flame

Ο αλγόριθμος Fractal Flame [23] είναι μια επέκταση του βασικού IFS, σχεδιασμένη ειδικά για να δημιουργία εικόνων που είναι πιο αισθητικά ευχάριστες. Τέτοιες εικόνες χρησιμοποιούνται συχνά ως ταπετσαρίες και screensavers για προσωπικούς υπολογιστές. Αν και το Fractal Flame εισάγει αρκετές τροποποιήσεις, οι περισσότερες έχουν καθαρά αισθητική αξία (πχ χρώμα) και ως αποτέλεσμα δεν θα αξιοποιηθούν στα πλαίσια αυτής της εργασίας. Η μοναδική τροποποίηση που μας ενδιαφέρει είναι η μη γραμμικότητα. Πρόκειται για μη γραμμικές συναρτήσεις $G : \mathbb{R}^2 \mapsto \mathbb{R}^2$ που μπορούν να εφαρμοστούν σε δισδιάστατες συντεταγμένες μεταξύ των γραμμών 6 και 7 του Αλγορίθμου 1. Στο έργο των [23] αυτές οι

συναρτήσεις αναφέρονται ως “variations”. Μερικά παραδείγματα αυτών των συναρτήσεων είναι:

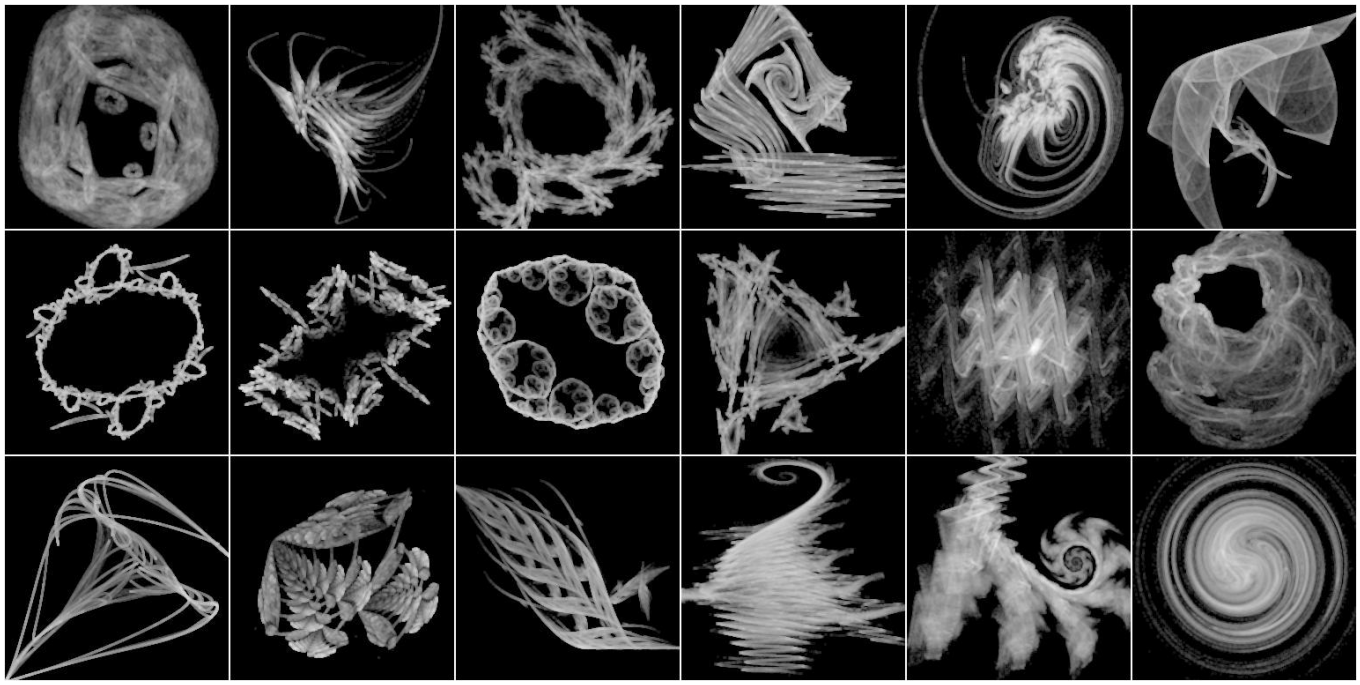


Figure 1.2: Παραδείγματα μη γραμμικών ελκυστών. Τα εμφανιζόμενα σχήματα παρουσιάζουν διαφορετικά μοτίβα σε σύγκριση με τα αρχικά γραμμικά φράκταλ. Αυτό αναμένεται να ενισχύσει σημαντικά τη συνολική ποικιλομορφία.

- $G_1(x, y) = (sinx, siny)$
- $G_6(x, y) = r(sin(\theta + r), cos(\theta - r))$
- $G_{16}(x, y) = \frac{2}{r+1}(y, x)$

Στις παραπάνω εξισώσεις, r και θ είναι πολικές συντεταγμένες: $r = \sqrt{x^2 + y^2}$ και $\theta = arctan(x/y)$.

Παραδείγματα μη γραμμικών ελκυστών φαίνονται στο Σχήμα 1.2. Είναι προφανές ότι οι παραγόμενες εικόνες διαφέρουν σημαντικά από τα βασικά IFS. Τη στιγμή της σύνταξης αυτού του εγγράφου, καμία άλλη εργασία δεν έχει διερευνήσει fractal flames στο πλαίσιο της βαθιάς μάθησης. Όπως θα φανεί στα επόμενα κεφάλαια, η εισαγωγή μη γραμμικών συναρτήσεων ενισχύει την ποικιλομορφία των παραγόμενων δειγμάτων.

1.2 Συνθετικά Βίντεο

1.2.1 Απλή Παρεμβολή

Στόχος μας είναι η επέκταση των φράκταλ στη χρονική διάσταση και η παραγωγή αρχείων βίντεο. Η βάση για την παραγωγή φράκταλ βίντεο παρέχεται από το ακόλουθο θεώρημα:

Theorem 1 *[5] Έστω $\{W_i\}_{i=1}^N$ ένα IFS του οποίου οι συναρτήσεις παραμετροποιούνται από μία μοναδική φραγμένη μεταβλητή $t \in \mathbb{R}$. Τότε, η συνάρτηση $S(t)$ που αντιστοιχίζει την παράμετρο t στον ελκυστή του IFS με παραμέτρους t είναι συνεχής.*

Το παραπάνω θεώρημα μπορεί εναλλακτικά να ερμηνευθεί ως εξής: συνεχείς αλλαγές στις παραμέτρους του IFS έχουν ως αποτέλεσμα συνεχείς αλλαγές στην παραγόμενη εικόνα. Είναι δυνατόν να παραχθούν ομαλές κινούμενες εικόνες μεταβάλλοντας ελαφρώς τις παραμέτρους του IFS σε κάθε διαδοχικό καρέ.

Ως εκ τούτου, μια απλή μέθοδος κατασκευής βίντεο μπορεί να επιτευχθεί με πρώτο βήμα τη δειγματοληψία παραμέτρων για δύο διαφορετικές εικόνες φράκταλ. Αυτές οι εικόνες θα λειτουργήσουν ως το πρώτο και το τελευταίο καρέ του βίντεο αντίστοιχα. Ο μόνος περιορισμός είναι ότι ο αριθμός των συναρτήσεων N πρέπει να είναι κοινός και για τις δύο εικόνες. Στην πράξη το N επιλέγεται τυχαία από το $U(\{3, \ldots, 8\})$. Για την παραγωγή κίνησης, οι παράμετροι των δύο εικόνων παρεμβάλλονται γραμμικά. Περισσότερες λεπτομέρειες εμφανίζονται στον Αλγόριθμο 2. Το αποτέλεσμα είναι μια επιπρόσθετη διάσταση, η διάσταση του χρόνου. Όπως είναι επιθυμητό, αυτή η προσέγγιση παράγει μια συνεχή ακολουθία IFS. Κάθε IFS μπορεί να απεικονιστεί ξεχωριστά και παράλληλα μέσω του Αλγορίθμου 2. Αυτό έχει ως αποτέλεσμα μια ακολουθία καρέ και επομένως ένα βίντεο.

Algorithm 2 `sample-video-naive`(N, T): Παράγει ακολουθία IFS μέσω γραμμικής παρεμβολής παραμέτρων.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Sequence of parameters: $W \in \mathbb{R}^{T \times N \times 6}$

1: Sample $W_{start}, W_{end} \in \mathbb{R}^{N \times 6}$ ▷ Parameters for the first & last frame
2: **Initialize:** $W \leftarrow \texttt{zeros}(T, N, 6)$
3: **for** $t = 0$ to $T - 1$ **do**
4: $\quad W[t] \leftarrow \frac{T-1-t}{T-1} W_{start} + \frac{t}{T-1} W_{end}$ ▷ Linear interpolation
5: **end for**
6: **return** W

Δυστυχώς, χειροκίνητη εξέταση των βίντεο που παράγονται με αυτή τη μέθοδο αποκαλύπτει ένα δυσμενές φαινόμενο. Αν και η αρχή και το τέλος του προκύπτοντος βίντεο είναι ικανοποιητικά, οι ενδιάμεσοι ελκυστές δεν είναι. Τα ενδιάμεσα καρέ τείνουν να είναι εξαιρετικά αραιά, αφήνοντας την πλειοψηφία του χώρου κενή. Τέτοια δείγματα απεικονίζονται στο Σχήμα 1.3.

Υποθέτουμε ότι τέτοια βίντεο δεν είναι κατάλληλα για προ-εκπαίδευση νευρωνικών δικτύων. Το σχήμα του ελκυστή πρέπει να παραμείνει μη τετριμμένο σε όλη τη διάρκεια του βίντεο. Επομένως, η απλή γραμμική παρεμβολή των παραμέτρων IFS δεν είναι αποδεκτή μέθοδος παραγωγής βίντεο

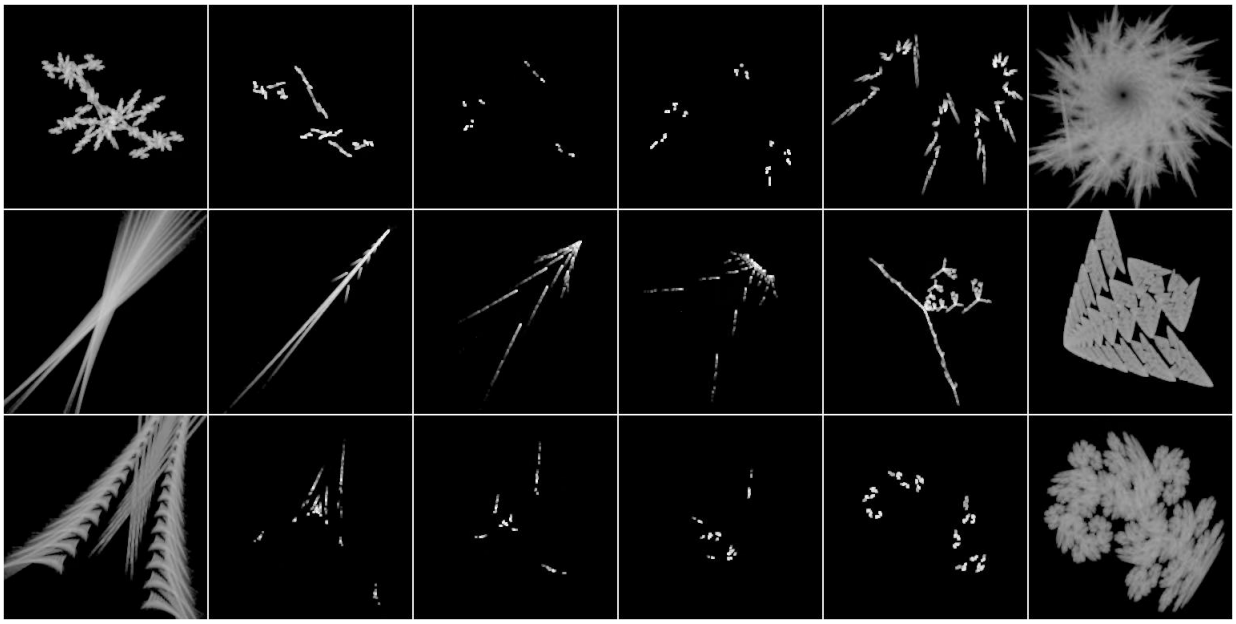


Figure 1.3: Παραδείγματα ανεπιθύμητης συμπεριφοράς. Κάθε γραμμή απεικονίζει μια ακολουθία εικόνων IFS. Η απλή γραμμική παρεμβολή των παραμέτρων IFS προκαλεί ανεπιθύμητη αραιότητα στα ενδιάμεσα καρέ. Μια τέτοια συμπεριφορά δεν είναι ικανοποιητική για μεγάλης κλίμακας προ-εκπαίδευση νευρωνικών δικτύων.

1.2.2 Παραγοντοποιημένη Παρεμβολή

Η ανεπιθύμητη αραιότητα των ενδιάμεσων πλαισίων μπορεί να επιλυθεί με ανάλυση σε ιδιάζουσες τιμές. Όπως εξηγείται στην ενότητα 1.1.3, ο πίνακας IFS μπορεί να εκφραστεί ως $A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$. Κατά συνέπεια, αντί να παρεμβάλλονται άμεσα οι παράμετροί του, αυτό το έγγραφο προτείνει την παρεμβολή κάθε υπο-πίνακα ξεχωριστά. Έτσι, ένα τυχαίο βίντεο μπορεί να δημιουργηθεί με τα ακόλουθα βήματα:

1. Δειγματοληψία των D_1 και D_2. Αυτοί οι πίνακες παραμένουν σταθεροί για όλη τη διάρκεια του βίντεο, δεδομένου ότι περιέχουν ακέραιες τιμές και παρεμβολή δεν μπορεί να εφαρμοστεί. Η κατασκευή αυτών των πινάκων μπορεί να επιτευχθεί με δειγματοληψία τιμών από $\{-1, 1\}$.

2. Δειγματοληψία των γωνιών $\theta_1^{start}, \theta_1^{end}$ από $U(0, 2\pi)$, παρεμβολή τους στη διάσταση του χρόνου και κατασκευή πίνακα περιστροφής R_{θ_1} για κάθε χρονική στιγμή. Η ίδια διαδικασία ισχύει για R_{θ_2}.

3. Δειγματοληψία πινάκων $\Sigma^{start}, \Sigma^{end}$ και παρεμβολή τους στη διάσταση του χρόνου για την κατασκευή του πίνακα Σ. Αυτές οι μήτρες είναι η πιο σημαντική συνιστώσα των παραμέτρων και ακατάλληλες τιμές μπορούν να οδηγήσουν σε μη ικανοποιητικά αποτελέσματα. Ως εκ τούτου, η δειγματοληψία γίνεται υπό συγκεκριμένους περιορισμούς. Περισσότερες λεπτομέρειες εμφανίζονται στο Παράρτημα Α του [2].

4. Σύνθεση πίνακα παραμέτρων μέσω πολλαπλασιασμού των προηγούμενων υπο-πινάκων: $A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$.

5. Δειγματοληψία των διανυσμάτων b^{start}, b^{end} και παρεμβολή τους στη χρονική διάσταση. Αυτό το βήμα είναι πανομοιότυπο με την προηγούμενη απλή προσέγγιση και δεν απαιτεί περαιτέρω τροποποιήσεις. Τέλος, ο προκύπτων πίνακας ενώνεται με τον σύνθετο πίνακα A από το προηγούμενο βήμα με αποτέλεσμα τις τελικές παραμέτρους IFS.

Η προαναφερθείσα διαδικασία είναι προσαρμοσμένη από το έργο των [10]. Τα δείγματα που προκύπτουν δεν υποφέρουν πλέον από αραιότητα αλλά παραμένουν ικανοποιητικά σε όλη τη διάρκεια του βίντεο. Τέτοια δείγματα φαίνονται στο Σχήμα 1.4. Αυτή η προσέγγιση θα αποτελέσει το θεμέλιο για παραγωγή συνθετικών συνόλων δεδομένων μεγάλης κλίμακας για προ-εκπαίδευση νευρωνικών δικτύων.

1.2.3 Χάσμα Μεταξύ Συνθετικών και Πραγματικών Βίντεο

Για συνθετικές εικόνες, προηγούμενες δημοσιεύσεις [4] συμπεραίνουν ότι η απόδοση σε πραγματικά δεδομένα βελτιώνεται όταν οι πρώτες συμπεριλαμβάνουν συγκεκριμένες δομικές ιδιότητες των τελευταίων. Ομοίως, μελέτες μεγάλης κλίμακας για την προ-εκπαίδευση [17, 53, 94] εξάγουν πως η αποτελεσματικότητα μηχανισμών προ-εκπαίδευσης επιδεινώνεται σημαντικά σε περίπτωση χάσματος μεταξύ των αρχικών και τελικών δεδομένων εκπαίδευσης. Ως εκ τούτου, είναι απαραίτητο να γεφυρωθεί το χάσμα μεταξύ συνθετικών βίντεο φράκταλ και δειγμάτων από πραγματικές βάσεις αναγνώρισης δράσης.

Για το σκοπό αυτό, η παρούσα ενότητα παραθέτει χαρακτηριστικά πραγματικών βίντεο από βάσεις αναγνώρισης δράσης [89, 56, 39] που έχουν παρατηρηθεί χειροκίνητα. Για κάθε χαρακτηριστικό προτείνονται μέθοδοί προσομοίωσης στα πλαίσια της προ-εκπαίδευσης. Ο στόχος της προσομοίωσης δεν είναι ο απόλυτος ρεαλισμός, αλλά απλή και διαισθητική προσέγγιση. Η πλειονότητα των προτεινόμενων μεθόδων προσομοίωσης έχει σχεδιαστεί ως απλοί μετασχηματισμοί που μπορούν να υλοποιηθούν αποτελεσματικά ως augmentation. Ένα υποσύνολο των προτεινόμενων τεχνικών απεικονίζεται στο Σχήμα 1.6.

Μη Γραμμική Κίνηση

Η μέθοδος σύνθεσης βίντεο που περιγράφηκε προηγουμένως παράγει απλή ευθεία κίνηση. Ωστόσο, η πραγματική ανθρώπινη κίνηση είναι σημαντικά πιο πολύπλοκη. Για να μειωθεί αυτό το χάσμα, αντί για απλή γραμμική παρεμβολή μεταξύ του πρώτου και του τελευταίου καρέ, μπορούμε να χρησιμοποιήσουμε πιο περίπλοκες συναρτήσεις (Σχήμα 1.5):

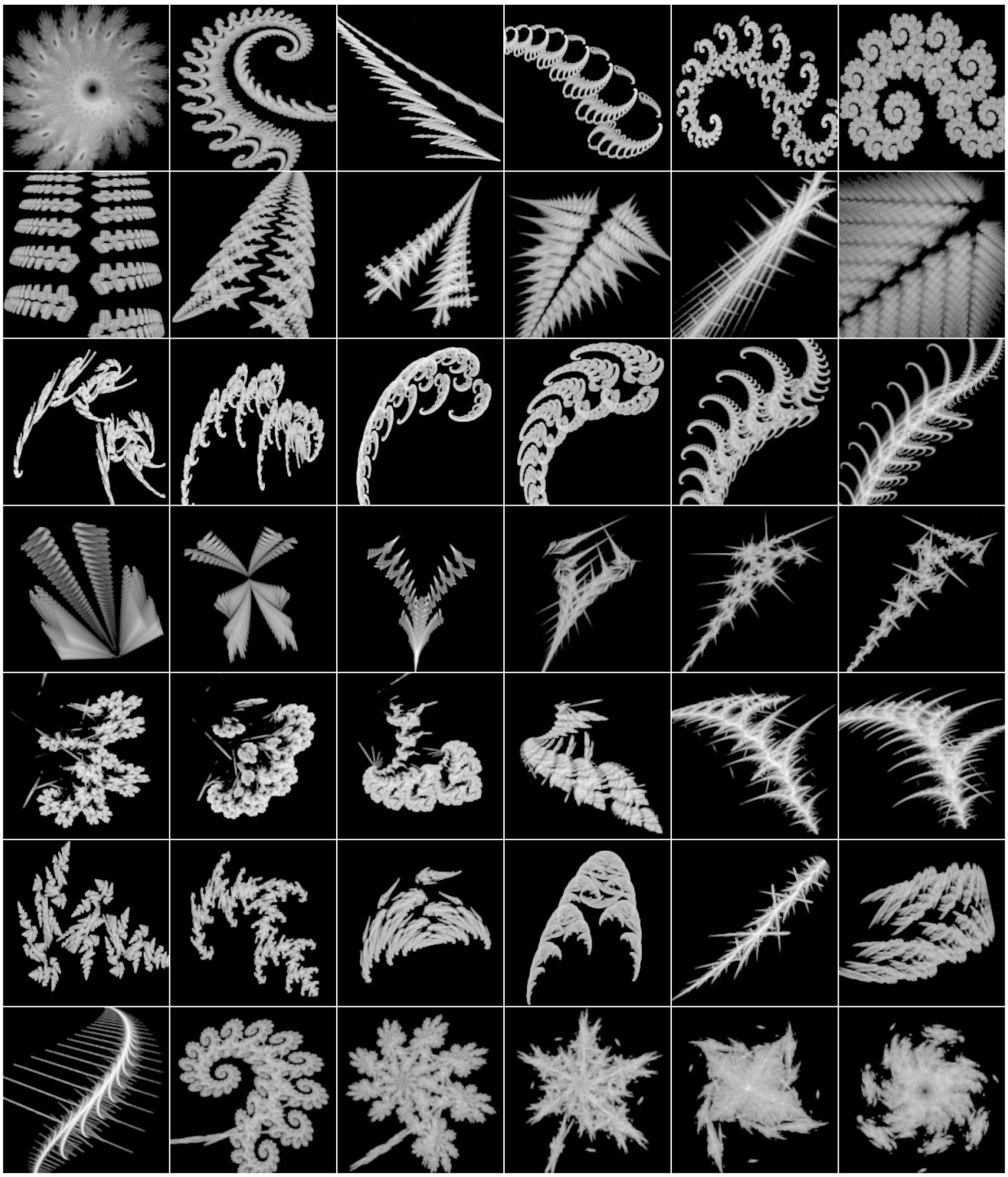


Figure 1.4: Παραδείγματα παραγοντοποιημένης παρεμβολής. Κάθε γραμμή απεικονίζει μια ακολουθία εικόνων IFS. Το ζήτημα της αραιότητας επιλύεται και οι ελκυστές μεταβάλλονται στο χρόνο χωρίς να συρρικνώνονται.

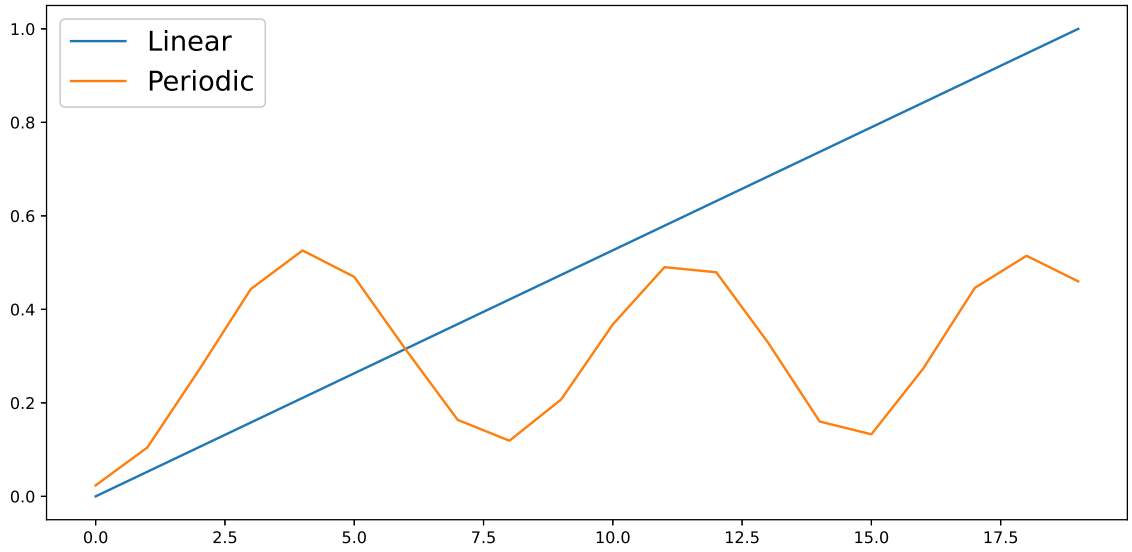


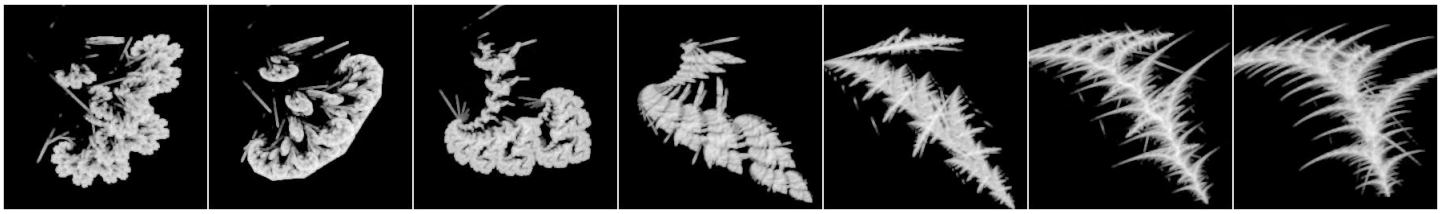
Figure 1.5: Προτεινόμενη περιοδική καμπύλη παρεμβολής.

- **Ημιτονοειδής Παρεμβολή**: Ένα σημαντικό μέρος της ανθρώπινης δραστηριότητας αποτελείται από περιοδική ή σχεδόν περιοδική κίνηση. Για την προσομοίωση τέτοιας κίνησης, μια θορυβώδης ημιτονοειδής συνάρτηση μπορεί να χρησιμοποιηθεί ως παρεμβολή.
- **Απότομη Παρεμβολή**: Πολλά βίντεο χαρακτηρίζονται από γρήγορη και ξαφνική κίνηση. Αυτή η κίνηση προσεγγίζεται από μια γραμμική παρεμβολή με σημαντικά μεγαλύτερο κλίση.
- **Τυχαία Παρεμβολή**: Επιπρόσθετες δραστηριότητες χωρίς σαφή μοτίβα μπορούν να προσομοιωθούν από μια τυχαία κυματομορφή.

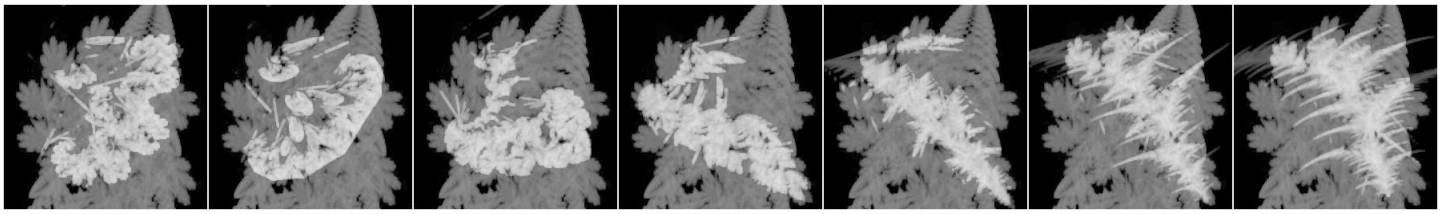
Μια επιπλέον πτυχή της ανθρώπινης κίνησης είναι η συνθετικότητά της: περίπλοκες κινήσεις αποτελούνται από πολλαπλές απλές κινήσεις. Για παράδειγμα, το τρέξιμο αποτελείται από μια περιοδική κίνηση των ποδιών καθώς και από μια διαφορετική περιοδική κίνηση των χεριών. Αυτή η ιδιότητα μπορεί να προσομοιωθεί αποδίδοντας διαφορετική καμπύλη παρεμβολής σε κάθε συνάρτηση του IFS.

Ποικιλομορφία

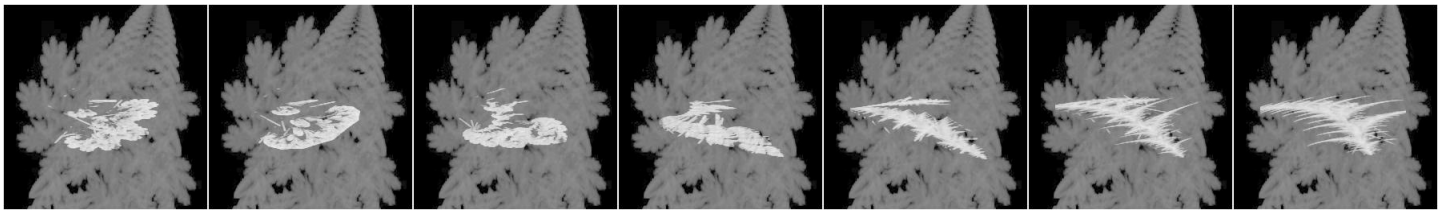
Προηγούμενες δημοσιεύσεις σχετικά με εικόνες [4] συμπεραίνουν ότι η ποικιλομορφία των συνθετικών δεδομένων είναι μια στοιχειώδης ιδιότητα για την εκμάθηση ισχυρών οπτικών αναπαραστάσεων. Ως εκ τούτου, με στόχο την τόνωση της ποικιλομορφίας των παραγόμενων βίντεο, μη γραμμικά φράκταλ (βλ. Ενότητα 1.1.4) περιλαμβάνονται επίσης στο προτεινόμενο σχέδιο προ-εκπαίδευσης.



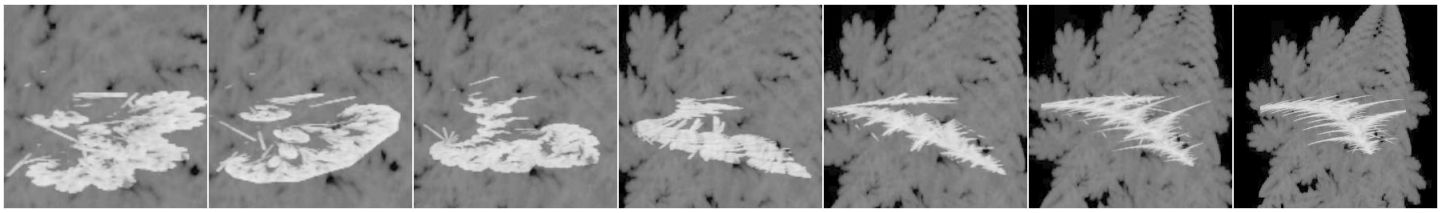
(a) Αρχικό Βίντεο



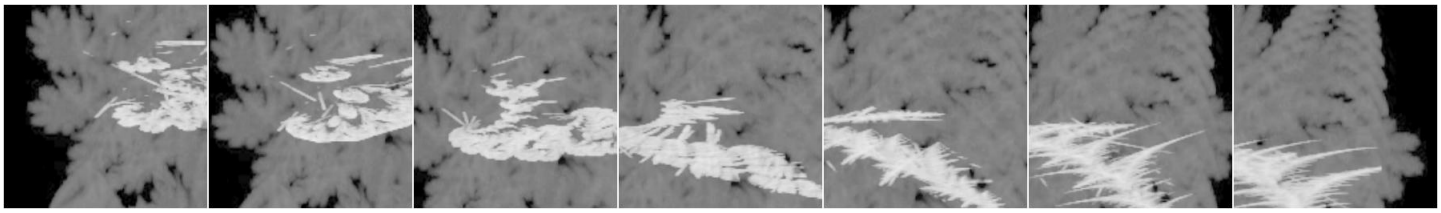
(b) Στατικό Φόντο



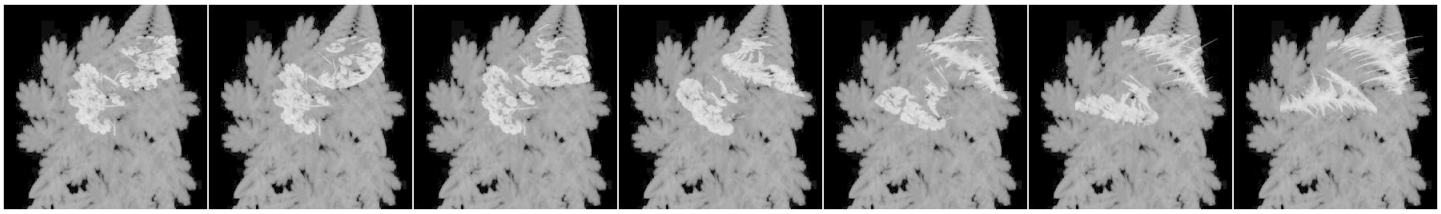
(c) Στατικό Φόντο + Σμίκρυνση Προσκηνίου



(d) Στατικό Φόντο + Σμίκρυνση Προσκηνίου + Ζουμ Κάμερας



(e) Στατικό Φόντο + Σμίκρυνση Προσκηνίου + Μετατόπιση Κάμερας



(f) Στατικό Φόντο + Σμίκρυνση Προσκηνίου + Συλλογική Δραστηριότητα

Figure 1.6: Υποσύνολο από τις προτεινόμενες τεχνικές για τη γεφύρωση χάσματος μεταξύ πραγματικών και συνθετικών βίντεο.

Άνθρωποι που εμφανίζονται σε βίντεο αναγνώρισης δράσης έχουν ξεκάθαρο σχήμα και περίγραμμα. Συνεπώς επιλέγουμε συναρτήσεις που επίσης ικανοποιούν αυτές τις ιδιότητες. Πρόκειται για variations $0, 4, 6, 13, 14, 15, 16, 20, 27$ και 29 από το παράρτημα των [23]. Επιπλέον, αξίζει να αναφερθεί ότι οι προαναφερθείσες μη γραμμικές συναρτήσεις αυξάνουν τις υπολογιστικές απαιτήσεις και επιβραδύνουν την παραγωγή συνθετικών βίντεο. Ως εκ τούτου, τα μη γραμμικά φράκταλς θα αποτελούν μόνο το 50 % των παραγόμενων δεδομένων.

Τυχαίο Στατικό Φόντο

Σκηνές στα πραγματικά βίντεο αναγνώρισης δράσης αποτελούνται από ένα προσκήνιο (άτομο που εκτελεί κάποια ενέργεια) και ένα φόντο (περιβάλλον γύρω από το άτομο). Το τελευταίο θα μπορούσε, για παράδειγμα, να είναι ένα γήπεδο γκολφ ή μια πισίνα. Στην πιο απλή περίπτωση, το φόντο είναι εντελώς στατικό και η μόνη κίνηση στο βίντεο προέρχεται από το προσκήνιο.

Μέχρι στιγμής, αυτή η ιδιότητα απουσιάζει από τα συνθετικά μας βίντεο τα οποία απεικονίζουν μόνο ένα κινούμενο σχήμα χωρίς περιβάλλον. Η ιδιότητα μπορεί να προσομοιωθεί ως ένας άμεσος μετασχηματισμός Κατά τη διάρκεια της εκπαίδευσης, για κάθε δείγμα x_i εντός ενός batch, ένα στατικό καρέ λαμβάνεται από ένα διαφορετικό βίντεο x_j και αναμιγνύεται με κάθε καρέ του x_i μέσω σταθμισμένου αθροίσματος:

$$\tilde{x_i} = (1 - a)x_i + ax_j[f]$$

Εδώ, x_i, x_j είναι δύο διαφορετικά βίντεο από το ίδιο batch, $f \sim U(\{0, \ldots, N_{frames} - 1\}$ και $a \sim U(a_{min}, a_{max})$. Στην πράξη, χρησιμοποιούμε $a_{min} = 0.25$ και $a_{max} = 0.55$.

Σμίκρυνση Προσκηνίου

Στα συνθετικά μας βίντεο, σχήματα των φράκταλ καλύπτουν ένα μεγάλο μέρος της οθόνης και συνήθως τοποθετούνται γύρω από το κέντρο του. Αντιθέτως, σε πραγματικά βίντεο αναγνώρισης δράσης, η θέση και το μέγεθος του προσκηνίου δεν είναι σταθερά αλλά τυχαία. Αυτή η αντίφαση πρέπει να αντιμετωπιστεί στο πλαίσιο προ-εκπαίδευσης. Εφόσον τα φράκταλ καλύπτουν την πλειοψηφία της οθόνης, δεν εφαρμόζεται cropping. Αντ' αυτού, τα βίντεο μπορούν να υποδειγματοληφθούν στις δύο χωρικές διαστάσεις με κλίμακες $s_h, s_w \sim U(s_{min}, s_{max})$ όπου $0 < s_{min} \leq s_{max} \leq 1$ και τοποθετούνται σε μια τυχαία θέση ενός άδειου ορθογωνίου. Στην πράξη, θέτουμε $s_{min} = 0.3$ και $s_{max} = 1.0$. Αυτή η διαδικασία τυχαιοποιεί τόσο το μέγεθος όσο και τη θέση του κινούμενου σχήματος.

Συλλογική Δραστηριότητα

Ένα μεγάλο μέρος των βίντεο δεν απεικονίζει μόνο ένα άτομο αλλά μια ομάδα που εκτελεί πανομοιότυπη ή σχεδόν πανομοιότυπη δραστηριότητα. Παραδείγματα είναι ομαδικά αθλήματα

όπως το βόλεϊ και το ποδόσφαιρο. Αυτό μπορεί να προσεγγιστεί με μια τροποποίηση της προηγουμένως προτεινόμενης σμίκρυνσης προσκηνίου. Συγκεκριμένα, μετά το βήμα παρεμβολής, το συνθετικό βίντεο αντιγράφεται N_{clone} φορές, με κάθε αντίγραφο να λαμβάνει διαφορετικό ελαφρύ augmentation. Σε μεταγενέστερα πειράματα, το N_{clone} ορίζεται σε 2. Τα augmentations περιλαμβάνουν τυχαία περιστροφή, οριζόντια αναστροφή και χρονική μετατόπιση. Η τελευταία καθιστά τα αντίγραφα ασύγχρονα. Όπως και προηγουμένως, κάθε αντίγραφο που προκύπτει τοποθετείται στη συνέχεια σε μια τυχαία θέση ενός άδειου ορθογωνίου. Σε αντίθεση με πριν, θέτουμε $s_{min} = 0.2$ και $s_{max} = 0.5$, καθώς πολλαπλά αντίγραφα μεγάλων σχημάτων είναι πιο πιθανό να επικαλύπτονται.

Μετατόπιση

Η ανθρώπινη δραστηριότητα και το φόντο δεν είναι τα μοναδικά δυναμικά στοιχεία που εμφανίζονται στα πραγματικά βίντεο. Μεγάλο ποσοστό των βίντεο περιέχει επιπρόσθετη κίνηση μετατόπισης. Μπορούν να διακριθούν τρία διαφορετικά είδη μετατόπισης:

- **Μετατόπιση του Προσκηνίου**: Αυτό συμβαίνει όταν άνθρωποι εκτελούν μια ενέργεια ενώ ταυτόχρονα περπατούν ή τρέχουν καθώς η κάμερα παραμένει στατική. Ως εκ τούτου, στο βίντεο που καταγράφεται, η θέση του προσκηνίου μετατοπίζεται ενώ το φόντο παραμένει ανεπηρέαστο.
- **Μετατόπιση Φόντου**: Όπως και προηγουμένως, αυτή η κίνηση είναι επίσης αποτέλεσμα της μετατόπισης του ανθρώπινου στόχου. Ωστόσο, τώρα η κάμερα ακολουθεί το προσκήνιο. Ως αποτέλεσμα, η θέση του προσκηνίου παραμένει ουσιαστικά στατική, αλλά το φόντο αποκτά την ίδια μετατόπιση αλλά με με αντίθετη κατεύθυνση. Ένα αξιοσημείωτο παράδειγμα είναι μια κάμερα που ακολουθεί αθλητές σε έναν αγώνα στίβου.
- **Μετατόπιση Κάμερας**: Στην περίπτωση αυτή, η απόλυτη θέση του ανθρώπινου παραμένει αμετάβλητη, αλλά ο στόχος της κάμερας μετατοπίζεται. Ως αποτέλεσμα, τόσο το προσκήνιο όσο και το φόντο μετατοπίζονται προς κατεύθυνση που είναι αντίθετη με την κάμερα.

Εξοικείωση με τέτοιες κινήσεις μπορεί να ενισχυθεί με απλούς μετασχηματισμούς. Για τη μετατόπιση φόντου, αρχικά μεγεθύνεται ένα στατικό καρέ φόντου και στη συνέχεια δημιουργείται μια ακολουθία crops με διαστάσεις του αρχικού βίντεο. Καθώς τα κέντρα των crops είναι διαδοχικά σημεία σε μια δισδιάστατη ευθεία, το αποτέλεσμα είναι μετατόπιση προς μια σταθερή κατεύθυνση. Για τη μετατόπιση της κάμερας, η διαδικασία είναι η παρόμοια με τη διαφορά ότι κάθε crop λαμβάνεται σε διαφορετικό καρέ. Για τη μετατόπιση του προσκηνίου, η διαφορά είναι ότι το βίντεο αρχικά μειώνεται σε μέγεθος και στη συνέχεια κάθε καρέ τοποθετείται σε διαφορετική θέση ενός κενού ορθογωνίου.

Περαιτέρω Μηχανισμοί

Επιπλέον, προτείνουμε τεχνικές που αφορούν Δυναμικό Τυχαίο Φόντο, Οπτική Γωνία, Ζούμ Κάμερας και Τρέμουλο Κάμερας. Περισσότερες λεπτομέρειες υπάρχουν στην Ενότητα 6.3.

1.3 Πειραματική Αξιολόγηση

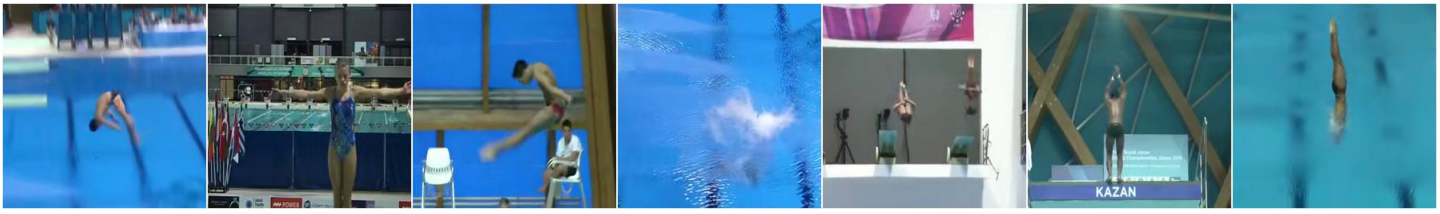
1.3.1 Εξεταστέα Σύνολα Δεδομένων

Το προτεινόμενο σχέδιο προ-εκπαίδευσης αξιολογείται μέσω τελικής εκπαίδευσης του μοντέλου σε 6 πραγματικές βάσεις από σύντομα βίντεο κλιπ. Όλα τα σύνολα δεδομένων έχουν σχεδιαστεί για το πρόβλημα της κατηγοριοποίησης. Η κλίμακα των συνόλων δεδομένων είναι μικρή λόγω υπολογιστικών περιορισμών, ενώ το μήκος περιορίζεται ώστε να ταιριάζει με τα στατιστικά των συνθετικών βίντεο. Τα 6 σύνολα δεδομένων παρουσιάζουν σημαντικές διαφορές μεταξύ τους και επιλέχθηκαν για να μεγιστοποιήσουν τη συνολική ποικιλομορφία. Λεπτομερή στατιστικά φαίνονται στον Πίνακα 1.1. Δείγματα από τυχαία καρέ απεικονίζονται στο Σχήμα 1.7.

- **HMDB51** [56]: Καθιερωμένη βάση δεδομένων για αναγνώριση δράσης. Τα κλιπ συνήθως απεικονίζουν ένα άτομο να εκτελεί μια συγκεκριμένη ενέργεια. Επιπλέον, τα κλιπ συχνά περιέχουν ανεπιθύμητα φαινόμενα όπως κίνηση της κάμερας (μετατόπιση, τρέμουλο) καθώς και απότομες αλλαγές σκηνής.
- **UCF101** [89]: Παρόμοια βάση με το HMDB51, αλλά μεγαλύτερης κλίμακας. Ως αποτέλεσμα ο αναμενόμενος χρόνος για εκπαίδευση είναι σαφώς μεγαλύτερος.
- **DIVING48** [61]: Μια συλλογή από βίντεο από επαγγελματικούς διαγωνισμούς καταδύσεων. Η δυσκολία είναι υψηλή, καθώς όλα τα βίντεο παρουσιάζουν μεγάλη ομοιότητα και πολύ λεπτές διαφορές. Ένα υποσύνολο των βίντεο απεικονίζει συγχρονισμένη κατάδυση με πολλά άτομα.
- **EGTEA GAZE+** [62]: Αποτελείται από βίντεο σε πρώτο πρόσωπο με δραστηριότητες μαγειρικής σε διαφορετικά περιβάλλοντα κουζίνας. Τα βίντεο συχνά περιέχουν μικρά αντικείμενα, όπως εργαλεία και συστατικά μαγειρικής.
- **VOLLEYBALL** [42]: Καθιερωμένη βάση στον τομέα της αναγνώρισης συλλογικής δράσης. Κάθε κλιπ απεικονίζει δύο αντίπαλες ομάδες βόλεϊ όπου η μία παραμένει ανενεργή ενώ η άλλη εκτελεί μια συγκεκριμένη ομαδική δραστηριότητα (σετ, σπάικ, πάσα, νίκη). Αυτό έχει ως αποτέλεσμα 8 τάξεις συνολικά. Το μήκος κάθε βίντεο είναι ακριβώς 41 καρέ. Ο παρεχόμενη επισήμανση έχει αποδειχθεί ότι περιέχει σφάλματα [119].
- **YUP++** [27]: Περιέχει βίντεο δυναμικών σκηνών. Παραδείγματα είναι δασικές πυρκαγιές, καταρρέοντα κτίρια και ορμητικά ποτάμια. Σε κάθε κατηγορία, τα μισά βίντεο προέρχονται από στατική κάμερα ενώ τα υπόλοιπα μισά αποκτώνται με μια κινούμενη κάμερα. Η διάρκεια του κάθε βίντεο είναι 5 δευτερόλεπτα.

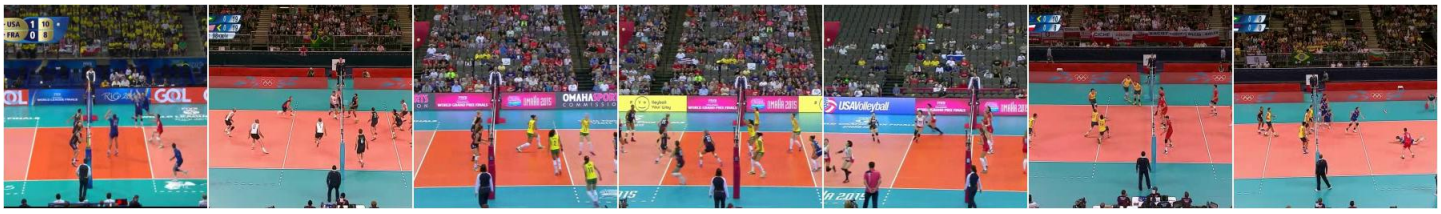
(a) HMDB51

(b) UCF101



(c) DIVING48

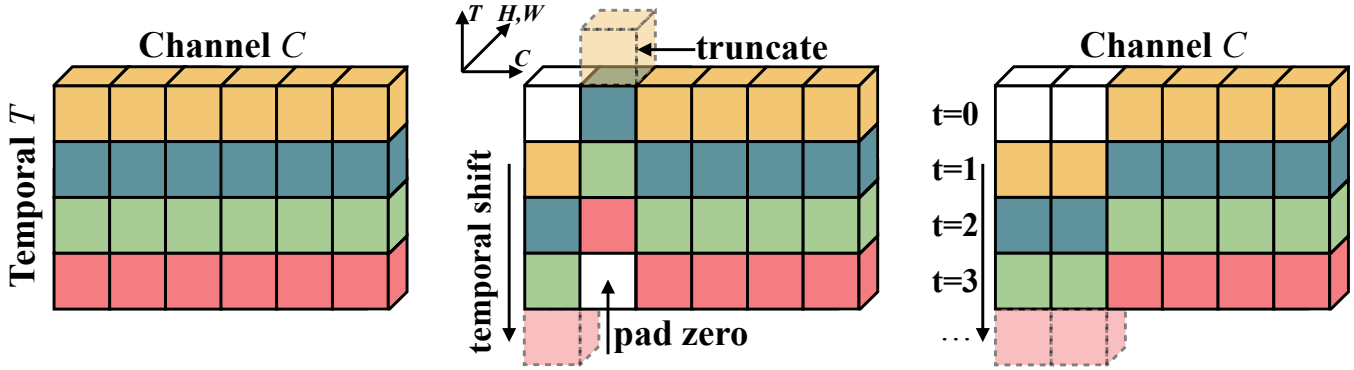
(d) EGTEA GAZE+



(e) VOLLEYBALL

(f) YUP++

Figure 1.7: Τυχαία δείγματα από καρέ τελικών συνόλων δεδομένων. Με εξαίρεση τις δύο πρώτες, όλες οι βάσεις δεδομένων παρουσιάζουν σημαντικές διαφορές μεταξύ τους.



(a) Ο αρχικός πίνακας χωρίς μετατόπιση.

(b) Offline χρονική μετατόπιση (διπλή κατεύθυνση).

(c) Online χρονική μετατόπιση (ενιαία κατεύθυνση).

Figure 1.8: TSM: Σχήματα προσαρμοσμένα από [63].

Σύνολο Δεδομένων	# Βίντεο για Επαλήθευση	# Βίντεο για Αξιολόγηση	# Κατηγορίες
HMDB51	3570	1530	51
UCF101	9537	3783	101
DIVING48	15943	2096	48
EGTEA GAZE+	8299	2022	106
VOLLEYBALL	2152	1341	8
YUP++	120	1080	20

Table 1.1: Στατιστικά στοιχεία τελικών συνόλων δεδομένων. Όλες οι βάσεις είναι μικρής κλίμακας και μήκους.

Για καθένα από τα προαναφερθέντα σύνολα δεδομένων, παρέχεται ένας επίσημος διαχωρισμός εκπαίδευσης-επαλήθευσης. Για την αξιολόγηση των προ-εκπαιδευμένων μοντέλων, τα εκπαιδεύουμε στο σύνολο εκπαίδευσης και στη συνέχεια αναφέρουμε την ακρίβεια στο σύνολο επικύρωσης.

1.3.2 Αρχιτεκτονική του Δικτύου

Η αρχιτεκτονική Temporal Shift Module (TSM) [63] χρησιμοποιείται για όλα τα πειράματα στην παρούσα εργασία. Το ResNet - 50 [38] χρησιμοποιείται ως βάση του TSM. Το TSM είναι μια αποδοτική αρχιτεκτονική Νευρωνικού Δικτύου δύο διαστάσεων που έχει σχεδιαστεί για το πρόβλημα αναγνώρισης δράσης. Επιτυγχάνει ανταλλαγή πληροφοριών μεταξύ γειτονικών καρέ μετατοπίζοντας τμήματα των καναλιών κατά μήκος της χρονικής διάστασης. Περισσότερες τεχνικές λεπτομέρειες παρουσιάζονται στο Σχήμα 1.8.

Το TSM επιλέχθηκε ως βασικό μας μοντέλο λόγω της υπολογιστικής του αποδοτικότητας. Εντυπωσιακά αποτελέσματα μπορούν να επιτευχθούν μόλις με 8 καρέ του βίντεο ως

είσοδο. Τόσο σύντομο μήκος ανακουφίζει σημαντικά τις υπολογιστικές απαιτήσεις τόσο της CPU όσο και της GPU. Η πρώτη αποδεικνύεται από μειωμένη φόρτωση δεδομένων, ενώ η δεύτερη από μειωμένη ποσότητα υπολογισμού εντός του νευρωνικού δικτύου. Εκτός αν ορίζεται διαφορετικά, η προ-εκπαίδευση μας πραγματοποιείται από το μηδέν και δεν χρησιμοποιεί έτοιμα βάρη από άλλες πηγές.

1.3.3 Πείραμα 1 - Χάσμα Μεταξύ Πραγματικών και Συνθετικών Βίντεο

Στόχος

Προηγούμενες δημοσιεύσεις σχετικά με συνθετικές εικόνες [4] συμπεραίνουν ότι, όταν πρόκειται για προ-εκπαίδευση, μίμηση ιδιοτήτων των πραγματικών δεδομένων οδηγεί σε ισχυρότερες οπτικές αναπαραστάσεις και καλύτερα αποτελέσματα. Ως εκ τούτου, ο στόχος αυτού του πειράματος είναι να επαληθεύσουμε αν αυτή η δήλωση ισχύει και για βίντεο. Συγκεκριμένα, χρησιμοποιούμε προσεγγιστικές τεχνικές που προτάθηκαν στην ενότητα 1.2.3. Παρατηρούμε τις επιδράσεις της κάθε τεχνικής στα τελικά αποτελέσματα και προσδιορίζουμε ποιες από αυτές είναι ευεργετικές.

Λεπτομέρειες

- Η προ-εκπαίδευση επιτυγχάνεται αποκλειστικά μέσω του σχεδίου αυτο-επιβλεπόμενης μάθησης MoCoV2 [14]. Το MoCoV2 επιλέχθηκε καθώς προσφέρει ένα συνδυασμό ικανοποιητικών αποτελεσμάτων και υπολογιστικής αποδοτικότητας.
- Τα πειράματα διεξάγονται διαδοχικά. Σε κάθε βήμα, εφαρμόζεται μια συγκεκριμένη τροποποίηση στη διαδικασία προ-εκπαίδευσης. Για τις τροποποιήσεις Κίνηση και Ποικιλομορφία, κατασκευάζεται ένα νέο συνθετικό σύνολο δεδομένων. Οι υπόλοιπες τροποποιήσεις υλοποιούνται ως επιπρόσθετο augmentation και συνεπώς επαναχρησιμοποιούν το προηγούμενο σύνολο δεδομένων.
- Το σύνολο δεδομένων προ-εκπαίδευσης αποτελείται από 100K μη επισημασμένα βίντεο φράκταλ (βλ. ενότητα 1.2.2) και κατασκευάστηκε με τυχαία δειγματοληψία παραμέτρων.
- Τα τελικά σύνολα δεδομένων παρουσιάζουν σημαντικές διαφορές μεταξύ τους. Συνεπώς, είναι πιθανό μια συγκεκριμένη τροποποίηση να οδηγήσει σε αύξηση της ακρίβειας για ένα υποσύνολο των βάσεων και σε μείωση για τα υπόλοιπα. Ως εκ τούτου, η τροποποίηση θα διατηρηθεί για όλα τα υπόλοιπα πειράματα εάν οδηγεί σε βελτίωση για τις βάσεις αναγνώρισης δράσης HMDB51 και UCF101. Αυτές επιλέχθηκαν ως το επίκεντρο της παρούσας εργασίας. Διαφορετικά, η τροποποίηση θα απορριφθεί και δεν θα ενσωματωθεί σε επόμενα πειραματικά βήματα.
- Το αρχικό πλάνο προ-εκπαίδευσης αποτελείται από βίντεο φράκταλ με γραμμική κίνηση και δεν περιλαμβάνουν μεθόδους γεφύρωσης χάσματος.

- Η ακρίβεια επικύρωσης μετά την τελική εκπαίδευση βρίσκεται στον Πίνακα 1.2.

Μέθοδος	Διατήρηση	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Τίποτα	-	31.5	70.3	4.7	50.5	61.1	43.7
Αρχή	-	41.4	72.7	24.3	49.8	80.6	52.3
Φόντο	Ναι	47.4	74.2	23.0	47.1	77.6	56.4
Κίνηση	Ναι	47.3	75.7	21.7	48.6	79.2	59.5
Ποικιλομορφία	Ναι	50.4	77.8	24.9	49.8	80.8	63.1
Σμίκρυνση + Τρέμουλο	Ναι	52.8	78.0	26.0	50.7	80.8	61.9
Μετατόπιση	Ναι	**54.5**	79.6	29.5	50.8	80.9	65.1
Ζουμ	Ναι	54.3	**80.2**	**30.8**	**52.0**	81.4	65.2
Οπτική Γωνία	Όχι	53.3	78.7	25.0	50.0	80.7	62.6
Ομάδα	Όχι	53.3	79.7	30.3	51.9	**81.9**	**66.0**

Table 1.2: Πείραμα 1: Τελική ακρίβεια επικύρωσης. **Bold** γραμματοσειρά υποδεικνύει τα καλύτερα αποτελέσματα στη συγκεκριμένη βάση. Η στήλη Διατήρηση καθορίζει εάν η αντίστοιχη τροποποίηση θα διατηρηθεί για όλα τα υπόλοιπα πειράματα.

Σχόλια

- Η προφανής παρατήρηση είναι ότι η εξομοίωση μιας ιδιότητας έχει θετική επίδραση στα σύνολα δεδομένων που περιλαμβάνουν την εν λόγω ιδιότητα και είτε αρνητική είτε καμία επίδραση στα σύνολα δεδομένων που δεν την περιλαμβάνουν. Για παράδειγμα:
  - Η τυχαιοποίηση του φόντου αυξάνει την ακρίβεια στις βάσεις αναγνώρισης δράσης HMDB51 και UCF101, αλλά τη μειώνει στα DIVING48 και VOLLEYBALL. Αυτό δεν αποτελεί έκπληξη. Για το πρώτο, μπορεί να υποτεθεί ότι το φόντο είναι διαφορετικό για κάθε βίντεο και γενικά άσχετο όσον αφορά την σωστή κατηγορία. Για το δεύτερο, είναι λογικό να θεωρήσουμε ότι το φόντο είναι περίπου κοινό για όλα τα δείγματα και ότι η ακριβής σχέση μεταξύ φόντου και προσκηνίου είναι σημαντική για τον προσδιορισμό της σωστής κατηγορίας.
  - Η τροποποίηση της συλλογικής δράσης είναι επωφελής για το σύνολο δεδομένων VOLLEYBALL, όπου το μοντέλο πρέπει να μάθει να παρατηρεί πολλά άτομα ταυτόχρονα. Ταυτόχρονα, είναι δυσμενής για το HMDB51 όπου η πλειονότητα των δειγμάτων απεικονίζει μόλις ένα άτομο.
- Ορισμένα αποτελέσματα δεν είναι συνεπή. Για παράδειγμα, το DIVING48 περιέχει συγχρονισμένα βίντεο καταδύσεων με πολλά άτομα, αλλά η τροποποίηση της ομάδας επιδεινώνει τα αποτελέσματα. Ωστόσο, η πτώση της ακρίβειας δεν είναι σημαντική.

- Η μόνη τροποποίηση που οδηγεί σε μη τετριμμένη βελτίωση σε όλες τις βάσεις είναι η ενίσχυση της ποικιλομορφίας. Αυτό είναι σύμφωνο με προηγούμενες εργασίες σχετικά με την αυτο-επιβλεπόμενη προ-εκπαίδευση με συνθετικές εικόνες [4], οι οποίες καταλήγουν στο συμπέρασμα ότι η ποικιλομορφία αποτελεί βασική ιδιότητα για την εκμάθηση ισχυρών αναπαραστάσεων.
- Η προ-εκπαίδευση είναι πολύ αναποτελεσματική για το σύνολο δεδομένων EGTEA. Η καλύτερη προσέγγισή μας (Zoum) βελτιώνει την ακρίβεια μόνο κατά 2% σε σύγκριση με την εκπαίδευση από το μηδέν.

1.3.4 Περαιτέρω Πειράματα

Η Ενότητα 7.2 περιέχει επιπλέον πειράματα όσον αφορά:

- Εναλλακτικές μεθόδους παρασκευής συνθετικών βίντεο.
- Εναλλακτικούς μηχανισμούς εκπαίδευσης.
- Σύγκριση μεταξύ σημαντικότητας σχήματος και κίνησης στα συνθετικά βίντεο.
- Διαφορετικά πλήθη δειγμάτων στη συνθετική βάση.
- Διαφορετική διάσταση εικόνων εισόδου.

1.4 Σύνοψη

Στόχος της παρούσας διπλωματικής εργασίας ήταν η επέκταση του θεμελιώδους έργου του [49] και η αυτόματη κατασκευή συνθετικών συνόλων δεδομένων για αναγνώριση δράσης. Αυτά τα σύνολα δεδομένων μπορούν να χρησιμοποιηθούν για προ-εκπαίδευση 3D CNNs αντί για την καθιερωμένη προσέγγιση Kinetics [11, 50]. Αυτή η προσέγγιση μπορεί να μετριάσει ορισμένα ελαττώματά πραγματικών δεδομένων, όπως το κόστος συλλογής και επισήμανσης, τα πνευματικά δικαιώματα, η ιδιωτικότητα καθώς και η ανθρώπινη προκατάληψη.

Ξεκινήσαμε με την επισκόπηση προηγούμενων εργασιών σχετικά με εικόνες φράκταλ IFS. Ιδιαίτερη έμφαση δόθηκε στη σωστή δειγματοληψία παραμέτρων προκειμένου να ελαχιστοποιηθούν ανεπιθύμητα φαινόμενα στις προκύπτουσες εικόνες. Πειραματιστήκαμε με άλλες οικογένειες εικόνων φράκταλ, αλλά συμπεράναμε ότι δεν είναι κατάλληλες για αυτόματη κατασκευή συνόλων δεδομένων.

Στη συνέχεια, δημιουργήσαμε σύντομα βίντεο προσθέτοντας χρονική διάσταση στις προαναφερθείσες εικόνες φράκταλ. Απλή γραμμική παρεμβολή των παραμέτρων φράκταλ οδηγεί σε μη ικανοποιητικά αποτελέσματα. Ωστόσο, η παρεμβολή ενδιάμεσων πινάκων επιλύει αυτήν την ατέλεια. Αυτή η προσέγγιση κατασκευάζει απλά βίντεο με γραμμική κίνηση. Η σημαντικότερη συμβολή της παρούσας διπλωματικής εργασίας είναι οι προτεινόμενες τεχνικές προσομοίωσης πραγματικών βίντεο. Αναγνωρίσαμε χειροκίνητα τις ιδιότητες των πραγματικών

βίντεο και προτείναμε μεθόδους ενσωμάτωσης τους στα συνθετικά βίντεο κατά τη διάρκεια της προ-εκπαίδευσης. Τέτοιες ιδιότητες είναι η μη γραμμική κίνηση, το τυχαίο φόντο, η μετατόπιση της κάμερας και άλλα. Επιδιώκοντας να εντοπίσουμε χαρακτηριστικά συνθετικών δεδομένων που βελτιώνουν τα αποτελέσματα σε πραγματικά βίντεο, κατασκευάσαμε επιπλέον εναλλακτικά σύνολα δεδομένων που διαφέρουν σημαντικά από τα φράκταλ.

Μετά τη διεξαγωγή πολλαπλών πειραμάτων, μπορούμε να εξαγάγουμε αρκετά συμπεράσματα σχετικά με την προ-εκπαίδευση με συνθετικά δεδομένα. Πρώτον, εντοπίσαμε πολλαπλές μεθόδους βελτίωσης της απόδοσης σε πραγματικές βάσεις. Αυτό μπορεί να επιτευχθεί με την ενίσχυση της ποικιλομορφίας των συνθετικών δεδομένων, με την προσομοίωση χαρακτηριστικών πραγματικών δεδομένων όπως τυχαίο φόντο, περιοδική κίνηση και σχετική κίνηση της κάμερας. Επιπλέον, συμπεραίνουμε ότι η εποπτευόμενη εκπαίδευση είναι μια οικονομικότερη λύση σε σύγκριση με εναλλακτικές αυτο-εποπτευόμενες προσεγγίσεις. Επιπλέον, ανακαλύπτουμε ότι το σχήμα των συνθετικών βίντεο είναι πιο σημαντικό από την κίνησή τους. Τέλος, παρατηρούμε ότι τα προ-εκπαιδευμένα μοντέλα συνεχώς υπολειτουργούν σε βίντεο που περιέχουν λεπτομέρειες. Προτείνουμε πιθανές λύσεις για αυτό το ελάττωμα, οι οποίες αφήνονται για μελλοντική εργασία.

1.5 Μελλοντικές Προεκτάσεις

Το προτεινόμενο πλαίσιο παρουσιάζει ορισμένους περιορισμούς και μπορεί συνεπώς να επεκταθεί προς διάφορες κατευθύνσεις. Συγκεκριμένα:

- **Μικρές λεπτομέρειες**: Έχει παρατηρηθεί πως τα προτεινόμενα σχέδια προ-εκπαίδευσης οδηγούν σε μοντέλα που συχνά αποτυγχάνουν να ανιχνεύσουν μικρές λεπτομέρειες που υπάρχουν στα βίντεο. Παραδείγματα αυτού του ελαττώματος είναι εργαλεία, ανθρώπινα άκρα καθώς και εκφράσεις του προσώπου. Η αιτία αυτής της συμπεριφοράς είναι το σύνολο δεδομένων προ-εκπαίδευσης όπου οι ετικέτες εξαρτώνται αποκλειστικά από το συνολικό βίντεο και όχι από μεμονωμένες λεπτομέρειες. Ως εκ τούτου, η απόδοση σε πραγματικά δεδομένα μπορεί να ενισχυθεί εάν κατά τη διάρκεια της προ-εκπαίδευσης το μοντέλο εκτίθεται σε βίντεο των οποίων οι ετικέτες καθορίζονται από ένα μικρό ποσοστό της εμφανιζόμενης κίνησης.

- **Εναλλακτικά συνθετικά δεδομένα**: Η προ-εκπαίδευση με τρισδιάστατα IFS φράκταλ θα μπορούσε να ενισχύσει την χωρική αντίληψη του προκύπτοντος μοντέλου. Επιπλέον, η προσθήκη Julia φράκταλ στο συνθετικό σύνολο δεδομένων θα μπορούσε να ενισχύσει την ποικιλομορφία και συνεπώς την απόδοση. Ωστόσο, η πρώτη διαδικασία υποφέρει από aliasing, ενώ η δεύτερη δεν μπορεί να αυτοματοποιηθεί για τη σύνθεση συνόλων δεδομένων. Ως εκ τούτου, απαιτείται πρόσθετη μελλοντική έρευνα.

- **Συμφραζόμενα**: Όλα τα συνθετικά βίντεο που παράγονται σε αυτό το έργο είναι μικρού μήκους και απεικονίζουν μία μόνο κίνηση. Τα πραγματικά βίντεο είναι πιο σύνθετα και μπορούν να αποτελούνται από πολλά στοιχεία που διαχωρίζονται στη διάσταση του χρόνου. Το κλειδί για την κατανόηση αυτών των βίντεο είναι η ικανότητα

μοντελοποίησης της σχέσης μεταξύ μακρινών γεγονότων. Η προσομοίωση αυτής της πολυπλοκότητας εντός των συνθετικών δεδομένων αναμένεται να ενισχύσει την απόδοση του δικτύου.

- **GAN**: Το χάσμα μεταξύ πραγματικών και συνθετικών βίντεο γεφυρώθηκε με χειροκίνητο εντοπισμό των χαρακτηριστικών των πραγματικών βίντεο και την προσομοίωση τους στα συνθετικά βίντεο. Εναλλακτικά, αυτό το έργο θα μπορούσε να ανατεθεί σε generative adversarial networks (GAN) [33, 44]. Ωστόσο, εκτός από τις αυξανόμενες απαιτήσεις υπολογισμού, τα GANs συχνά οδηγούν σε τεχνικές προκλήσεις στη διαδικασία εκπαίδευσης.

- **Motion Capture**: Σε αυτό το έγγραφο, η κίνηση δημιουργήθηκε με δείγματά τυχαίων καμπυλών. Πιο ρεαλιστική κίνηση θα μπορούσε να επιτευχθεί χρησιμοποιώντας υπάρχοντα σύνολα δεδομένων καταγραφής κίνησης [54, 72]. Η έγχυση γνώσης της ανθρώπινης κινηματικής σε συνθετικά βίντεο θα μπορούσε να συμβάλει στη γεφύρωση του χάσματος. Ωστόσο, κάτι τέτοιο θα έθετε το ζήτημα πνευματική ιδιοκτησίας.

Chapter 2

Introduction

This is the first chapter of the present work. It serves as a more detailed version of the abstract. Section 2.1 discusses the background behind the thesis as well as states its objectives. Next, Section 2.2 lists major contributions of this thesis. Lastly, Section 2.3 presents the thesis structure.

2.1 Motivation

Contemporary computer vision systems employ complex neural networks that require enormous amounts of data for training. Beginning with the ImageNet dataset [85], that consists of 1.4 million labeled images, the scale of vision datasets has been rapidly increasing. Since then, vision models have been trained using datasets whose size varies from tens of millions to a billion of samples [34, 112, 30].

Multiple issues arise regarding such datasets. First, the task of creating (collecting and optionally labeling) and managing (hosting and distributing) such datasets is seriously arduous. Second, it has been noted that vision datasets may inherit human biases [91, 9, 110, 117] and contain inappropriate content [7]. Third, the depiction of humans in these datasets poses questions of privacy [3]. Lastly, ownership concerns limit many datasets to noncommercial usage only.

As a result, there has recently been growing interest in synthetic datasets that mitigate these shortcomings. Amongst them, noteworthy is the seminal work of [49] who proposed to pre-train 2D CNNs on automatically generated images of fractals [5]. Since then, multiple other works have either improved their approach [4, 2, 48] or extended it to other domains [113].

As such datasets can be easily created automatically, the issue of ownership is solved. Additionally, these works do not depict human beings and therefore, there are no questions about biases, inappropriate content, and privacy. However, although this pre-training approach easily surpasses training from scratch, there is still a large gap compared to pre-training on real data.

The objective of this work is extend the aforementioned approach to the task of action recognition. Automatic action recognition with neural networks is of paramount importance as it enables accurate detection and interpretation of human actions from video or sensor data. This technology has broad applications across various domains, including surveillance, healthcare, robotics, sports analysis, and human-computer interaction. The significance of action recognition is additionally outlined by the sheer amount of videos available on the internet. With over 500 hours of video uploaded to YouTube every minute, there is an immediate need for robust algorithms that can help organize, summarize and retrieve this massive amount of data.

2.2 Contributions

As mentioned previously, this thesis seeks to extend the ideas of [49] to the field of video and apply them to the task of action recognition. Specifically, the main contributions of this thesis are the following:

- Using fractal geometry [5] as well as other generative processes, we propose a methodology to automatically construct large-scale datasets of short synthetic video clips. Such datasets could be employed for pre-training 3D CNNs instead of Kinetics [11, 50]. Both supervised and self-supervised learning is applicable.
- Additionally, we narrow the domain gap between synthetic and real videos, significantly improving downstream results. To achieve this, we manually observe real video samples and identify their key properties such a periodic motion, random background, camera displacement etc. These properties are carefully emulated during pre-training via simple transformations. Through rigorous ablations we determine the properties of synthetic data that strengthen downstream results.
- Our proposed pre-training framework is evaluated by fine-tuning on small-scale action recognition datasets HMDB51 [56] and UCF101 [89] as well as four other video benchmarks. Compared to standard Kinetics pre-training, our reported results come close and are even superior on a portion of benchmarks.
- Lastly, we conduct a thorough error analysis of the pre-trained models' predictions. We deduce that on downstream datasets, models systematically misclassify samples which share specific characteristics. To mitigate this inadequacy, we propose tailored modifications that can be applied to synthetic datasets. These modifications are left for future work and are expected to lead to stronger visual representations as well as better downstream results.

2.3 Thesis Structure

The rest of this work is organized as follows:

- Chapter 3 provides a brief overview of theoretical concepts that are necessary for understanding the present work: machine learning, deep learning and fractal geometry.
- Chapter 4 summarizes previously published works on relevant domains: pre-training with synthetic data, spatiotemporal models and action recognition datasets.
- Chapter 5 discusses the concept of fractal images, providing rendering algorithms as well as heuristics that minimize degenerate behavior in rendering.
- Chapter 6 presents a methodology to automatically produce short video clips using fractal geometry. Additionally, it proposes several techniques for domain adaptation.
- Chapter 7 describes the proposed experimental frameworks as well as lists quantifiable results on downstream tasks.
- Chapter 8 summarizes the contributions of this work, presents its conclusions and proposes future directions.

Chapter 3

Background

This chapter provides a brief overview of theoretical concepts that will serve as building blocks for the present work. Specifically, Section 3.1 establishes the field of Machine Learning as well as defines its categories. Next, Section 3.2 discusses the domain of Deep Learning while, Section 3.3 presents relevant models and techniques in more detail. Section 3.4 overviews the notion of Transfer Learning which is of great importance for the present work. Furthermore, Section 3.5 introduces the concept of Self-Supervised Learning as well as several recently introduced algorithms. Last, Section 3.6 lists mathematical concepts that are necessary for the understanding of fractals.

3.1 Machine Learning

Machine learning is a branch of Artificial Intelligence that uses data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Such algorithms are used in a wide variety of applications, including but not limited to computer vision, speech recognition and natural language processing. In the past few decades, this field has enjoyed massive growth and is now amongst the most important sectors of computer science.

3.1.1 Types of Machine Learning

Machine learning techniques are divided into three main categories, based on the input signal:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

The term supervised learning refers to algorithms that work on data that contains both the inputs and the desired outputs. In other words, the dataset is labeled. Mathematically, such algorithms aim to create a mapping $y = f(x)$, where x is the input and y is the output of the given dataset.

Additionally, supervised learning can be divided into two types of problems: classification and regression. Classification algorithms assign data into specific discrete categories. A real world example is the classification of a tumor as benign or malicious. On the other hand, regression techniques, given a data point, predict numerical values. An instance of regression is an algorithm that predicts stock prices.

Some examples of supervised learning techniques are:

- Linear regression
- Naive Bayes
- K-nearest neighbors (KNN)
- Support Vector Machines (SVM)
- Decision trees

Unsupervised Learning

The field of unsupervised learning studies algorithms that work on data that contains only inputs and lacks labels. Instead of responding to feedback, such algorithms detect patterns and structures in data without human intervention.

Unsupervised learning techniques can be further divided into three categories major categories: clustering, dimensionality reduction and association.

- Clustering is the process of dividing data points based on their similarities or differences.
- Dimensionality reduction is the transformation of an initial dataset into a different one where the number of samples remains the same, but the size of each sample is reduced.
- Association is the detection of relationships between different variables in a dataset.

Reinforcement Learning

Reinforcement learning revolves around the concept of a virtual agent that performs a specific task in a given environment. The goal of such an algorithm is for the agent to learn to perform its task correctly. In order to ensure this, the agent is awarded with positive or negative rewards based on its actions and the state of the environment.

3.2 Deep Learning

3.2.1 Introduction

Deep learning is a subset of machine learning, which is centered around models called neural networks. Such models are usually constructed by stacking smaller standardized sub-modules. This results in large and computationally heavy neural architectures. Deep learning eliminates some of data pre-processing and feature extraction that is typically involved with machine learning. It is noteworthy that in many applications, deep learning models surpass human performance.

3.2.2 Concepts

Loss Function

The goal of any Deep Learning algorithm is to return a mapping $f()$ that accurately matches input samples to their corresponding labels. Therefore, it is necessary to measure the error (loss) of such a model, i.e. a distance between the true label y and the model prediction $\hat{y}$. This is done using a loss function $L(\hat{y}, y)$ whose output is a numeric. Such a function must have an infimum, which means that the lower the error value, the better the prediction.

Given a train set $(x_{1\,n}, y_{1\,n})$, a cost function L per sample and a model $f(x;\theta)$, the total loss is defined as the average loss on all training data:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta), y_i)$$

The goal of the training process is to find the optimal parameters θ that minimize the total error:

$$\hat{\theta} = \text{argmin}_\theta \mathcal{L}(\theta)$$

Some widely used loss functions are the following:

Mean Squared Error (MSE):

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

Mean Absolute Error (MAE):

$$L(\hat{y}, y) = |\hat{y} - y|$$

Cross Entropy (CE):

$$L(p, q) = -\sum_{k} p_k log(q_k)$$

In this case, p and q are the true and the predicted probability distributions, respectively.

Gradient Descent

Gradient descent (GD) is the dominant algorithm for training neural networks. A gradient (at a point) is the slope of the tangent to the function at that point and it points to the direction of the most significant increase of the function. Gradient descent computes the gradient of the loss function $\mathcal{L}(\theta)$ with respect to the parameters θ for the entire training set. A hyperparameter η (learning rate) controls the extent to which the model parameters are adjusted concerning the loss gradient. GD is formulated as:

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

Stochastic Gradient Descent (SGD) is a variation of GD that performs a parameter update for each training example x_i and label y_i:

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta; x_i, y_i)$$

SGD is computationally more efficient than GD, as the latter performs redundant computations for large datasets. SGD does away with this redundancy by performing one update at a time. Therefore, SGD is more suitable for online learning.

Underfitting, Overfitting Regularization & Dropout

A key challenge in the field of machine learning is for the model to perform well on new inputs samples that were not seen during training. This property is called generalization. In order to evaluate the generalization of a machine learning model the following steps are executed. First, the model is trained on a training set. After completion, a separate set of samples called test set is fed into the model. Using the model's outputs, numeric values are

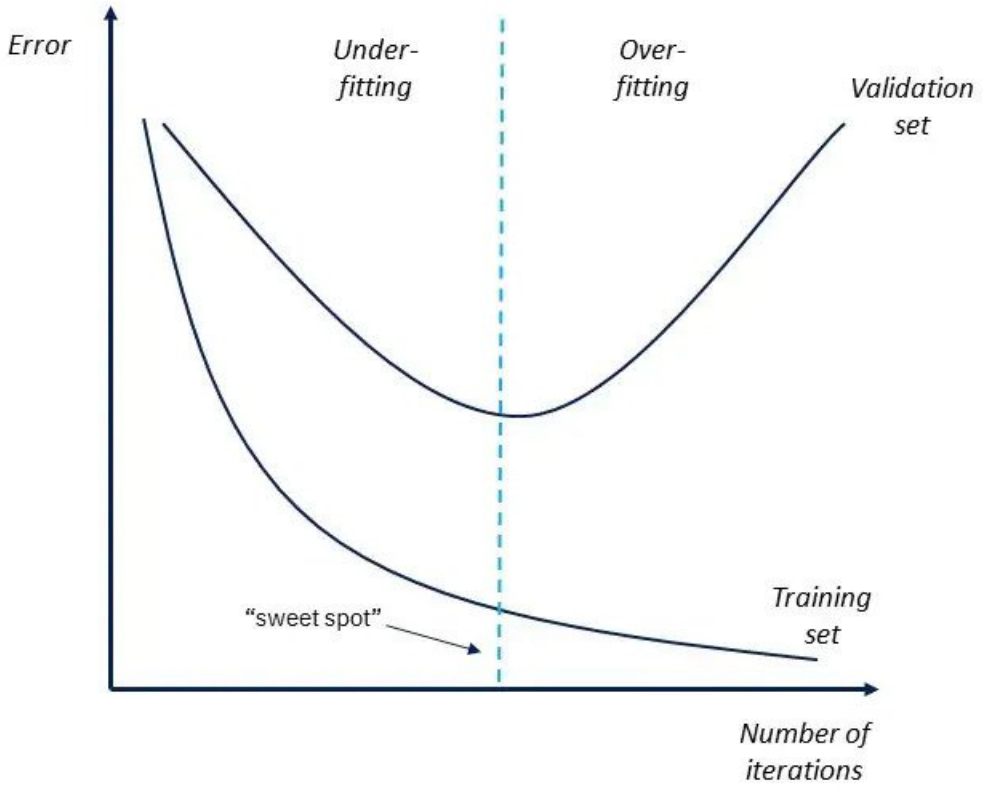


Figure 3.1: Training and test errors behave differently. At the left end of the figure, both errors are high. This is underfitting. Making the model more complex, training error decreases, but so does the gap between training and generalization error. The state are the right end of the figure is overfitting. Figure reproduced from [98].

called metrics are calculated. These metrics measure the model's generalization ability. Examples are accuracy for classification and mean Intersection-Over-Union (mIOU) for semantic segmentation.

Based on the performance of the model on the training set and the test set, the following phenomena can be distinguished:

- **Underfitting** occurs when the model cannot achieve sufficient results on the training set.
- **Overfitting** occurs when the model achieves strong results on the training set but fails to do so on the test set, creating a gap in performance. Both behaviors are demonstrated in Figure 3.1.

Several methods exist that combat overfitting and aid in generalization. The two most common ones are Regularization and Dropout.

Regularization, otherwise known as weight decay, imposes restrictions to the learning objective that force the model's parameters to have numerically smaller absolute values. Mathematically, this can be expressed by adding a penalty $R(\theta)$ to the learning objective, multiplied by a chosen hyperparameter λ as follows:

$$\hat{\theta} = \text{argmin}_\theta \left(\frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) + \lambda R(\theta) \right)$$

The two most common regularization variants are $L1$ and $L2$. $L1$ regularization penalizes the $L1$ norm of the model's weights punishing uniformly low and high values. $L2$ regularization, on the other hand takes the form of the standard $L2$ norm (Euclidean) of the parameters, trying to keep the sum of the squares of the parameter values low.

$$R_{L_1}(W) = ||W||_1^1 = \sum_{i,j} W_{i,j}$$
$$R_{L_2}(W) = ||W||_2^2 = \sum_{i,j} (W_{i,j})^2$$

Dropout [90] is another method that combats overfitting in neural network training. Specifically, dropout is parameterized by a single probability p. During training, matrices that are computed in intermediate (hidden) layers of the neural network have their values randomly replaced with zero with probability p. Additionally, the entire matrix is multiplied by $\frac{1}{1-p}$ so that the distribution of the output remains the same. During testing, dropout is disabled. This method forces the network not to rely on specific weights and thus ameliorates overfitting.

Activation Functions

To model complex non-linear phenomena that occur in the real world, neural networks apply a variety of functions to intermediate representations. Such functions can also help normalize computations and stabilize the training process. Some of the most popular functions are presented next:

Sigmoid Function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This formula constrains the output between 0 and 1 and thus can be employed to model a probability. It is plotted in Figure 3.2.

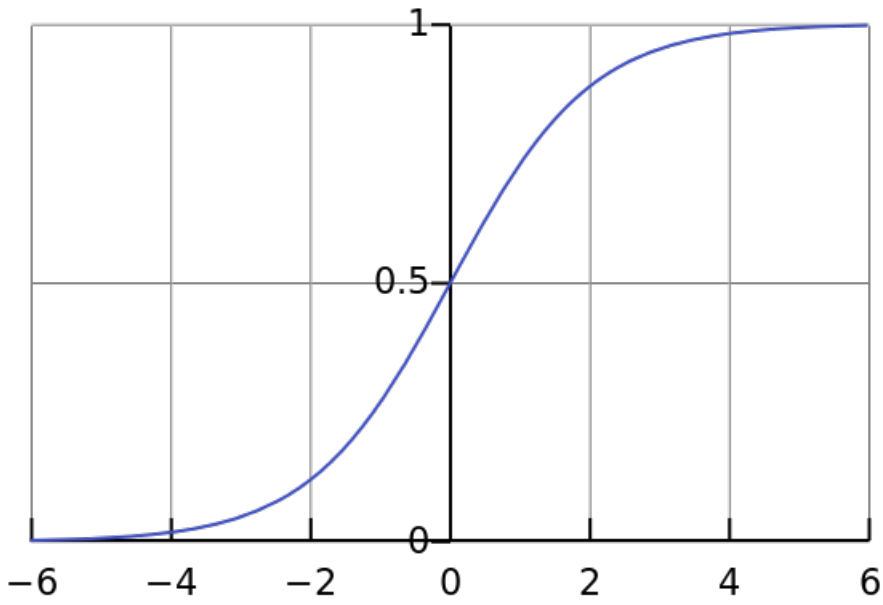


Figure 3.2: The sigmoid activation function. Figure reproduced from [99].

Rectified Linear Unit (ReLU)

$$f(x) = \max(x, 0)$$

ReLU [75] is the most widely used activation function chiefly due to its fast computation. It is plotted in Figure 3.3.

Hyperbolic Tangent (tanh)

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

tanh restricts its outputs in the range between -1 and 1. It is illustrated in Figure 3.4

3.3 Deep Learning Models

3.3.1 Feedforward Neural Networks

Feedforward Neural Networks, otherwise known as Multilayer Perceptrons (MLPs), are the foundation of deep learning models. Although, initially proposed as a model of the human brain, MLPs have a simple structure. An MLP $f\colon \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ consists of n

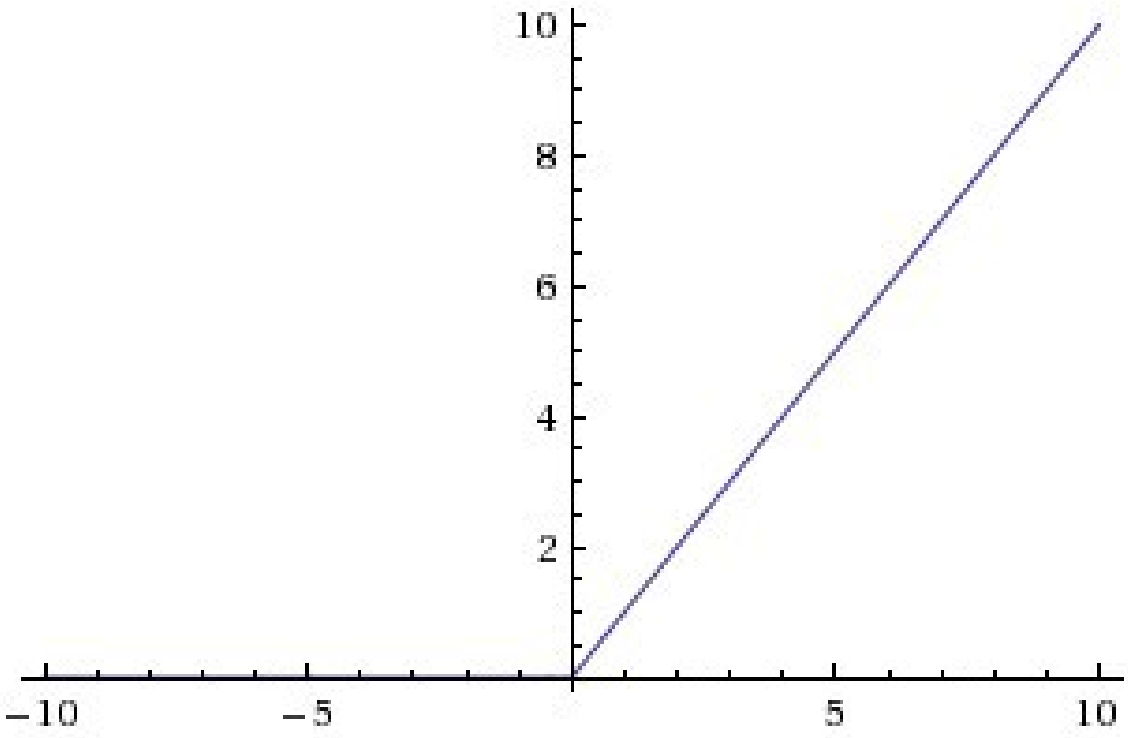


Figure 3.3: The ReLU activation function. Figure reproduced from [100].

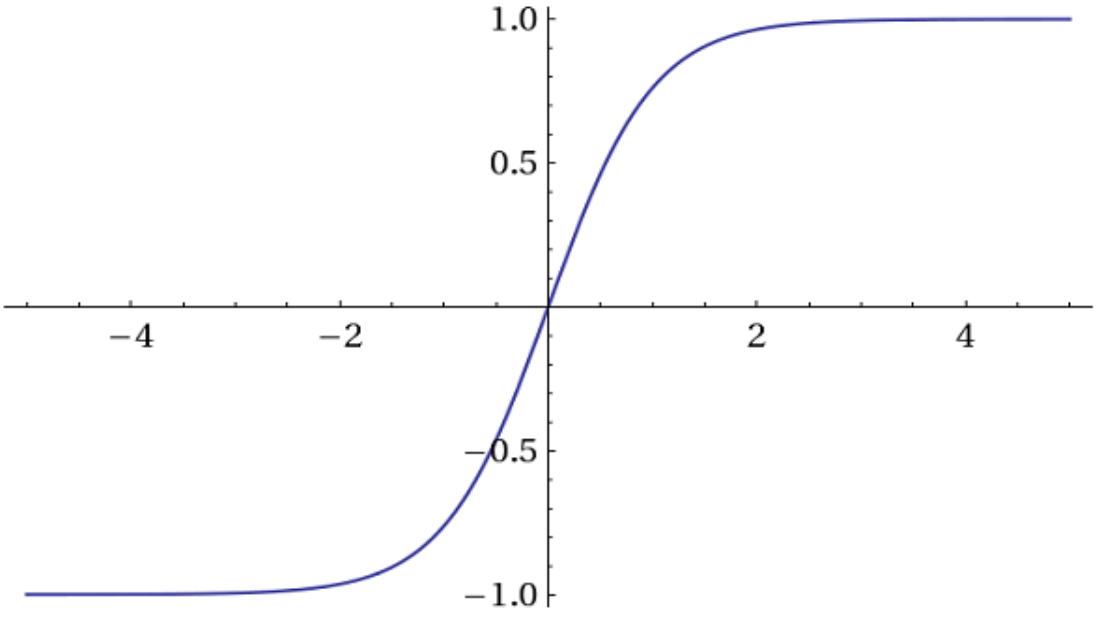


Figure 3.4: The tanh activation function. Figure reproduced from [95].

feedforward layers stacked on top of each other: $f(x) = \left(f^{(n)} \circ f^{(n-1)} \circ \cdots \circ f^{(1)}\right)(x)$. Each layer $f^{(i)} \colon \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ has the form:

$$f^{(i)}(x) = \sigma\left(Wx + b\right)$$

where $x \in \mathbb{R}^{d_{i-1}}$ is the output of the previous layer, σ is an activation function and $W \in \mathbb{R}^{d_{i-1} \times d_i}, b \in \mathbb{R}^{d_i}$ are learnable parameters. An example of an MLP is shown in Figure 3.5.

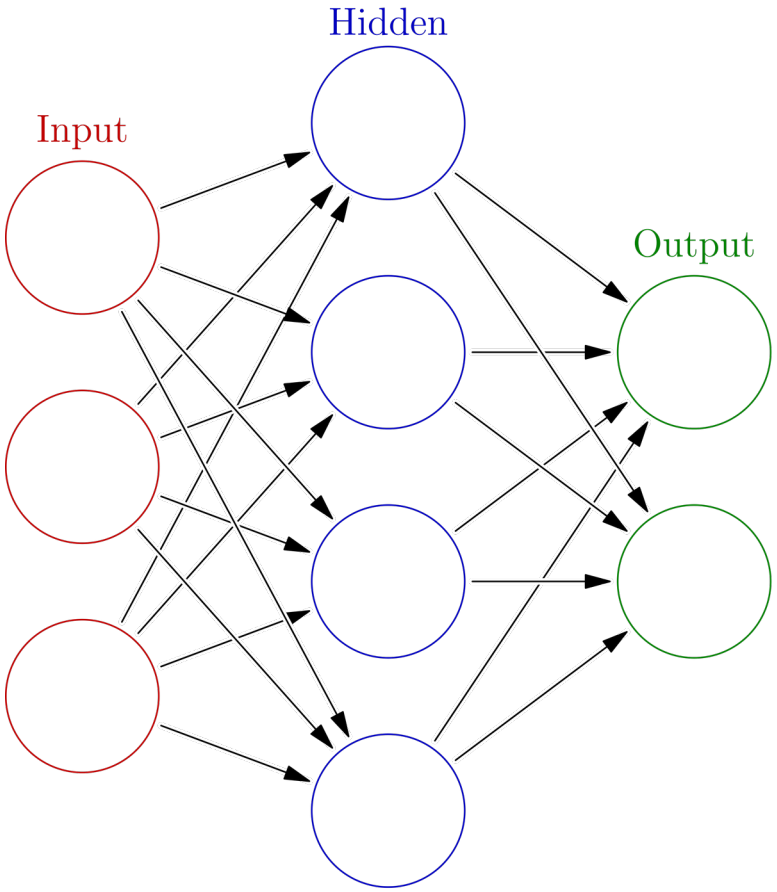


Figure 3.5: An MLP with $n = 2, d_{in} = 3, d_1 = 4, d_{out} = 2$. Figure reproduced from [108].

3.3.2 Recurrent Neural Networks

In the field of computer science, various modalities can be represented as sequences with variable length. Examples are text, audio, music notes, DNA sequences etc. Feedforward neural networks assume that inputs have fixed size and are independent and are therefore not suitable for such data.

Recurrent Neural Networks produce their output taking into consideration accumulated contextual information from previous instances (i.e. memory) [32]. Specifically, information flows from one timestep to the next through a feedback loop as depicted in Figure 3.6

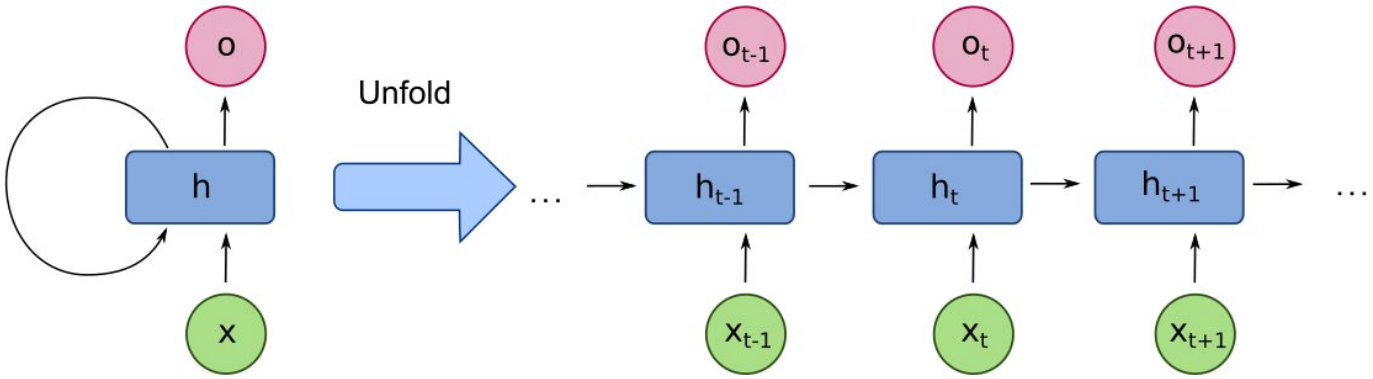


Figure 3.6: Recurrence mechanism used in RNNs. Figure reproduced from [109].

The most simple variant of the RNN is the Elman RNN [25]. Given an input sequence $x_1, x_2, \ldots, x_T$, at time t, this model computes the hidden state h_t and the output o_t as follows:

$$h_t = \sigma_h \left(U x_t + V h_{t-1} + b_h \right)$$
$$o_t = \sigma_y \left(W x_t + b_y \right)$$

where σ_h, σ_y are activation functions, U, V, W are weight matrices and b_y, b_h are bias vectors. The total number of learnable parameters is independent of the sequence length, as the same parameters are used for each time-step.

As the length of the input sequence increases, the Elman RNN becomes more difficult to optimize [6]. Specifically, large input length can result in either the "exploding" or "vanishing" gradients phenomenon. One method that can combat the issue of "exploding" gradients is to constrain the gradients if they exceed a certain threshold. This technique is called gradient clipping [78]. As alternative solution, different RNN architectures have been designed to avoid the aforementioned problem.

Long Short-Term Memory (LSTM) [40] is a subcategory of RNN that was introduced to cope with "vanishing" gradients during the training process, allowing learning long-term dependencies that are not feasible for Elman RNN. The building block of LSTM

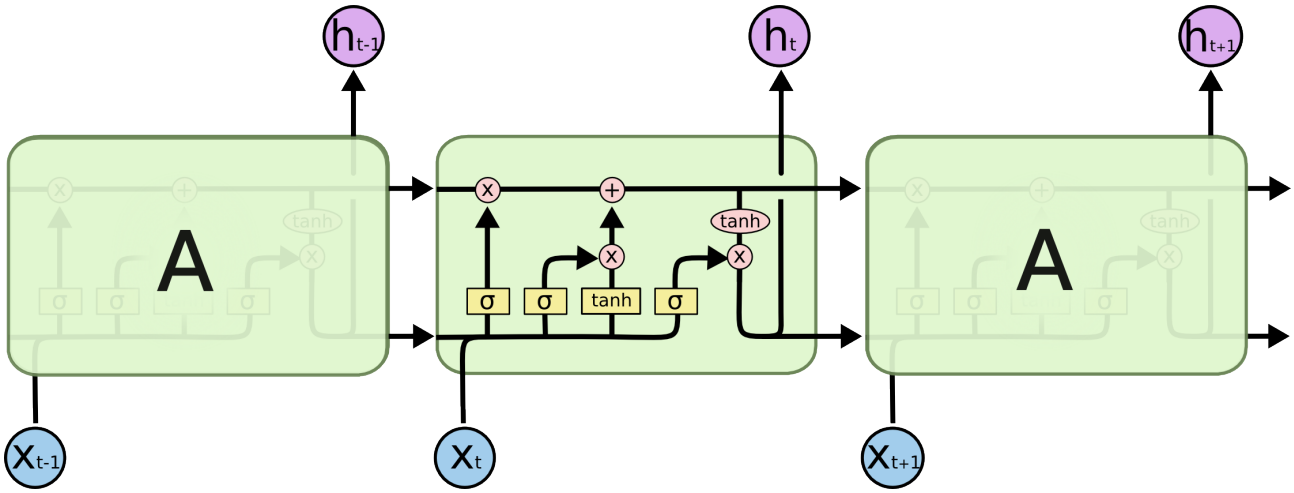


Figure 3.7: Illustration of an LSTM cell. Figure reproduced from [16].

networks, otherwise known as LSTM cell is illustrated in Figure 3.7 and consists of three gating units and two states that control the flow of information.

- The **forget gate** (f_t) decides what information should be thrown away or kept based on previous hidden state h_{t-1} and the current input x_t. The output range is between 0 and 1. When the output is close to 0, information is forgotten, while close to 1 means to it is being retained.
- The **input gate** (i_t) determines what input values will be propagated to the new cell state.
- The **output gate** (o_t) controls the exposure of the cell state to the hidden state based on the current input and the previous hidden state.
- The **cell state** (c_t) contains internal information and is updated based on past information filtered via the forget gate and on current information filtered via the input gate.
- the **hidden state** (h_t) encodes the input sequence until time-step t.

The aforementioned gates and states are computed as follows:

$$
\begin{aligned}
f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\
i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\
o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\
u_t &= tanh\left(W_u x_t + U_u h_{t-1} + b_u\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot u_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

3.3.3 2D Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) [29, 58] is a different variant of neural networks that has gained much popularity in the field of Computer Vision [55]. The cornerstone of CNNs is the operation of convolution, which is defined as follows:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

where I is an input 2D matrix and K is a weighting 2D filter whose dimensions are much smaller than the input I. Intuitively, K can be seen as a detector that finds patterns at different parts of the input and in the context of CNNs, is usually a learnable parameter.

After the convolution operation, the resulting outputs are passed to an activation function such as ReLU [75]. On top of that, CNNs usually incorporate pooling layers that gradually decrease the spatial extent of the output. Max pooling and average pooling return the maximum and the average value respectively from the section of the image covered by the filter. A simple CNN architecture is shown in Figure 3.8. A demonstration of the aforementioned max pooling operation is displayed in Figure 3.9.

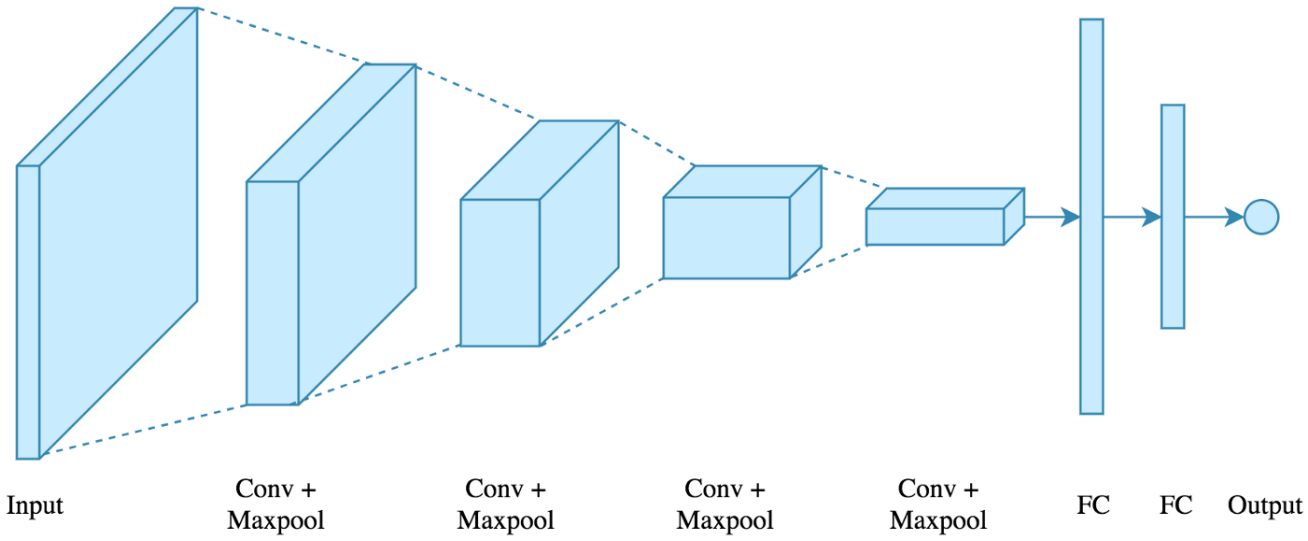


Figure 3.8: A simple CNN architecture. Figure reproduced from [96].

One key feature of CNNs is parameter sharing. This is based on the assumption that a pattern should be detectable irrespective of its position. To enforce this, the filter K is applied in a sliding window and therefore reused for different spatial positions. As a result, the number of parameters is greatly reduced and so is the computational cost. Another important property is equivariance. In other words, translation of the input results in translation of the output feature map. This allows filters to detect patterns across the input.

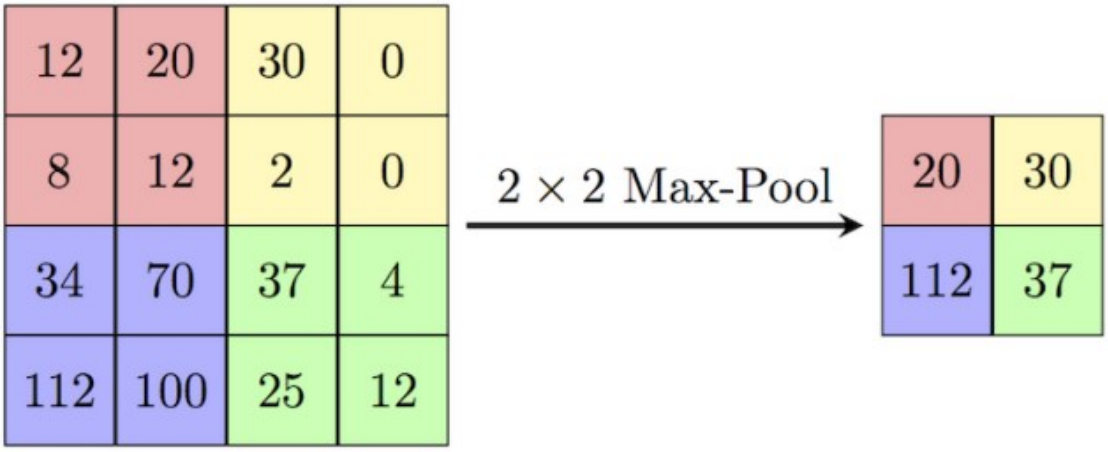


Figure 3.9: Example of 2 × 2 max pooling downsampling on a 4 × 4 matrix. Figure reproduced from [97].

3.3.4 3D Convolutional Neural Networks

The main domain of 2D CNNs is 2D images. 3D CNNs [45], on the other hand are designed for video. Specifically, whereas 2D CNNs extract the spatial features from a single image, 3D CNNs take in to account the temporal dimension and extract spatiotemporal features from a sequence of images.

In order to extract such features, 3D CNNs rely on the operation of 3D convolution. To be precise, this operation uses 3D filters that slide in 3 dimensions.

3.4 Transfer Learning

Large neural networks demand vast amounts of labeled data for training. In many cases such data may be unavailable and as a result, the performance of neural networks drop. This can be mitigated by leveraging knowledge from a source task with ample amounts of data and transferring it to a different target task where data is not sufficient. This procedure is called transfer learning.

3.5 Self-Supervised Learning Frameworks

Supervised learning requires labeled datasets. However, labeled examples are often expensive, difficult and time-consuming to obtain as they demand the efforts of experienced human annotators. Additionally, supervised learning methods suffer from spurious correlations and generalization errors, while also being vulnerable to adversarial attacks. As

such, the machine learning community has recently shown strong interest in alternative pre-training strategies.

Semi-supervised (SSL) learning has been proposed as a solution to the aforementioned limitations of supervised learning and has recently significantly risen in popularity. SSL is regarded as a subset of unsupervised learning. SSL algorithms are capable of producing strong representations from a large number of unlabeled instances without any human annotation. During SSL pre-training, a pretext task is designed for a deep learning algorithm to solve and pseudolabels for the pretext task are automatically constructed based on certain attributes of the input data. After the SSL pre-training, the learned model can be further transferred to downstream tasks.

The following subsections describe a few contemporary SSL frameworks, namely SimCLR [13], MoCoV2 [14] and BYOL [35]. These frameworks will be employed in experiments in later chapters.

3.5.1 SimCLR

SimCLR [13] randomly samples a batch of N samples and defines a contrastive prediction task on pairs of augmented instances from the batch. This results in $2N$ input samples. For each positive pair, SimCLR treats the other $2(N-1)$ augmented instances in the batch as negative instances. Cosine similarity between two instances u and v is calculated as:

$$sim(u, v) = \frac{u^T v}{(||u|| \cdot ||v||)}$$

For a positive pair of instances (i, j), the loss function is calculated as:

$$l_{i,j} = -log \frac{exp(sim(z_i, z_j)/T)}{\sum_{\substack{k=0 \\ k \neq i}}^{2N} exp(sim(z_i, z_k)/T)}$$

where T is the temperature hyperparameter.

The final loss is computed for all positive pairs, including both (i, j) and (j, i). For the domain of images, the employed augmentations techniques usually include cropping, resizing, and color distortion. Compared to the standard supervised classification task, SimCLR is computationally heavier as it requires two forward and two backward passes per training step. Additionally, increasing the batch size N increases the number of negative samples. This renders the task more difficult and therefore has been shown to lead to better downstream results. As such, SimCLR optimally requires a batch size in the order of thousands and is consequentially computationally heavy.

3.5.2 MoCoV2

MoCoV2 [14] similarly employs a contrastive approach and can be considered as an extension of SimCLR. Specifically, a dictionary lookup task is utilized. Consider an encoded query q and several previously encoded examples $\{k_0, k_1, k_2, \cdots\}$, which are the keys of a dictionary. Assume that a single key (denoted as k_+) in the dictionary matches q. A contrastive loss is a function whose value is low if q is similar to its positive key k_+ and dissimilar to all other keys, which are called negative. The loss function in MoCoV2 measures similarity via dot product and is calculated as:

$$L = -log\frac{exp(q \cdot k_+/T)}{\sum_{i=0}^{K} exp(q \cdot k_i/T}$$

where T stands for the temperature hyperparameter. The sum is calculated over one positive example and K negative examples.

MoCoV2 utilizes two neural networks. The first one computes the queries q and is fully differentiable. It requires both a forward and a backward pass. The second one computes the keys k and is not differentiable. It requires only a forward pass. As such, its parameters are updated as an exponential moving average of the first model's, The keys are stored in a circular queue and are updated at each training step. Before training, the queue is randomly initialized and at each step the outputs of the second model are added to it, replacing the oldest entries.

Computationally, MoCoV2 requires two forward steps and one backward step. Hence it is more lightweight that SimCLR but still heavier than the supervised classification objective. In addition, SimCLR's requirement for large batch size is not present in MoCoV2. The number of negative examples and therefore the difficulty depends on the size of the circular queue and not the batch size. As a result, for ImageNet pre-training, MoCoV2 achieves stronger downstream results compared to SimCLR. This is especially evident for smaller batch sizes.

3.5.3 BYOL

Bootstrap your own latent (BYOL) [35] differs from the previously described frameworks as it does not use negative pairs. Similarly to MoCoV2, BYOL employs two neural networks: the online network which is fully differentiable and the target network which is not differentiable. As previously, BYOL updates the target network with a slow-moving average of the online network.

Like in all previous frameworks, the first step constructs two different views q, k of each input sample by applying different augmentations. Afterwards, both views are fed through both encoders resulting in four output representations in total:

$$q_o = encoder\_online(q)$$
$$q_t = encoder\_target(q)$$
$$k_o = encoder\_online(k)$$
$$k_t = encoder\_target(k)$$

BYOL's training objective is to simply maximize the cosine similarity between all matching representations:

$$L = 2 - 2 * (sim(q_o, k_t) + sim(k_o, q_t))$$

Computationally, each training step requires four forward passes and two backward passes. As such, BYOL is significantly more intensive than all other objectives deployed in the present work. This is however justified with improved downstream results. BYOL surpasses both SimCLR and MoCoV2 for ImageNet pre-training and is expected to do so in experiments in later chapters as well. Moreover, negative samples are not employed within this framework and therefore, unlike SimCLR, large batch sizes are not required.

3.6 Mathematical Background for Iterated Function Systems

Aside from machine learning and deep learning, a key element of the present work is the notion of fractals. As such, it is necessary to first establish some relevant mathematical concepts .

Definition 1 *(Metric Space). A metric space is the pairing of a space X, with a function which measures distance $d : X \times X \mapsto \mathbb{R}$. d must satisfy the following requirements:*

1.	$d(x, x) = 0$	$\forall x \in X$
2.	$d(x, y) > 0$	$\forall x, y \in X, x \neq y$
3.	$d(x, y) = d(y, x)$	$\forall x, y \in X$
4.	$d(x, z) \leq d(x, y) + d(y, z)$	$\forall x, y, z \in X$

Definition 2 *(Sequence of Bounded Variation). Let $\{x_n\}_{n=1}^{\infty}$ be a sequence in a metric space (X, d). It is called a sequence of bounded variation if the following property holds:*

$$\sum_{n=1}^{\infty} d(x_{n+1}, x_n) < \infty$$

Definition 3 *(Cauchy Sequence). Let* $\{x_n\}_{n=1}^{\infty}$ *be a sequence in a metric space* (X, d)*. It is called a Cauchy sequence if for any* $\epsilon > 0$*,* $\exists N_\epsilon \in \mathbb{N}$ *such that if* $m, n \in \mathbb{N}$ *and* $m, n \geq N_\epsilon$*, then* $d(x_m, x_n) < \epsilon$*.*

Definition 4 *(Complete Metric Space). If* (X, d) *is a metric space, then the following statements are equivalent:*

- (X, d) *is a complete metric space.*
- *Every Cauchy sequence in* X *converges to a point in* X*.*
- *Every sequence of bounded variation in* X *converges to a point in* X*.*

Definition 5 *(Contractive Function). A function* $f : X \mapsto X$ *is called contractive on a given metric space* (X, d) *if* $\exists c \in [0, 1)$ *such that* $d(f(x), f(y)) \leq c \cdot d(x, y), \forall x, y \in X$*. In this case,* f *is called a contraction mapping, and* c *its contraction factor.*

Definition 6 *(Fixed Point). Let* (X, d) *be a metric space, and* $f : X \mapsto X$*.* $\bar{x} \in X$ *is a fixed point of* f *if* $f(\bar{x}) = \bar{x}$*.*

Definition 7 *(Open Cover and Subcover). Let* S *be a subset of a metric space* (X, d) *and* $\{U_i\}_{i \in I}$ *be a collection of open sets which could be finite, or infinite. Then if* $S \subseteq \cup_{i \in I} U_i$*, then* $\{U_i\}_{i \in I}$ *is called an open cover. Furthermore, if* $\{U_i\}_{i \in J}$*, where* $J \subset I$ *is an open cover for* S*, then* $\{U_i\}_{i \in J}$ *is called a subcover for* S*.*

Definition 8 *(Compact Set). Let* S *be a subset of a metric space* (X, d)*.* S *is called compact if every open cover of* S *has a finite subcover, which a subcover containing a collection of a finite number of open sets.*

Theorem 2 *(Banach's Fixed Point Theorem). Let* (X, d) *be a complete metric space. Then if* $f : X \mapsto X$ *is a contraction mapping, it possesses a unique, globally attractive fixed point.*

Having established the necessary mathematical tools, we can begin examining the concept of iterated function systems (IFS) images. IFS will be introduced in later chapters and will be employed to automatically produce synthetic images and video throughout this work. Aside from the image modality, it is worth mentioning that the aforementioned concepts have been heavily employed in signal analysis [69, 52, 20] as well as speech recognition [70, 82, 120]. However, this aspect will not be investigated in the present work.

Chapter 4

Related work

4.1 Pre-training with Synthetic Data

The seminal work of [49] employed fractal geometry to automatically generate large-scale labeled image datasets. To construct labels, they propose to randomly sample parameters for each category from $U(-1, 1)$ and add variation via random noise. These images were used to train image models instead of the standard ImageNet pre-training [85]. An overview of their approach can be seen in Figure 4.1. Their results can be seen in Table 4.1. Although the reported metrics are inferior to ImageNet pre-training [85], they evidently surpass training from scratch.

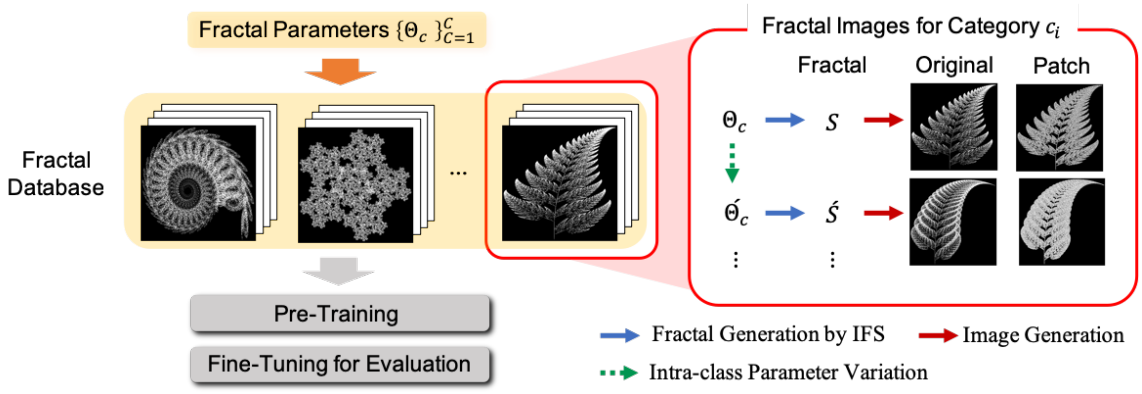


Figure 4.1: Overview of the framework proposed in [49]. Construction of a dataset is achieved without human labeling and image downloading. Such images can be employed to pre-train a convolutional network which will be assigned to conduct transfer learning for other datasets.

Since then, [2] proposed improvements such as a more intuitive augmentation policy, whereas [76] demonstrated that the framework is compatible with different neural architectures. As an orthogonal approach, [4] evaluated a multitude of different generative

Table 4.1: Classification accuracies after fine-tuning as reported in [49]. The last two rows display the results of pre-training with synthetic images. **Bold and underlined** values show the best scores, and **bold** values indicate the second best scores.

Pre-training Dataset	Type	C10	C100	IN1k	P365	VOC12	OG
Scratch	–	87.6	62.7	**76.1**	49.9	58.9	1.1
DC-10k	Self-supervision	89.9	66.9	66.2	**51.5**	67.5	15.2
Places-30	Supervision	90.1	67.8	69.1	–	69.5	6.4
Places-365	Supervision	**94.2**	76.9	71.4	–	**78.6**	10.5
ImageNet-100	Supervision	91.3	70.6	–	49.7	72.0	12.3
ImageNet-1k	Supervision	**96.8**	**84.6**	–	50.3	**85.8**	17.5
FractalDB-1k	Formula-supervision	93.4	75.7	70.3	49.5	58.9	**20.9**
FractalDB-10k	Formula-supervision	94.1	**77.3**	**71.5**	**50.8**	73.6	**29.2**

image processes and determined that performance is boosted if synthetic data is diverse and shares specific properties with real data. More recently, [47] carefully designed a data synthesis method which results in representations that exceed ImageNet pre-training.

Synthetic data has also been utilized in other domains. [113] extend the fractal approach to three dimensions and produce synthetic point clouds. For the field of action recognition, [46] pre-train neural models with three dimensional perlin noise [80, 81]. This work is the most relevant to the present one. Additionally, [118] construct synthetic images of palm prints with the help of Bezier curves [26].

An alternative approach involves computer graphics instead of noise processes. [36] propose to render a large-scele dataset of action clips using the photorealistic video game GTA V. They additionally design a contrastive pre-training framework tailored for the said dataset. A very similar work is that of [51], who combine existing synthetic datasets of 3D rendered video clips and utilize the resultant dataset for pre-training of 3D Convolutional Neural Networks. To bridge the domain gap and improve downstream performance, [115, 114, 111] propose to combine the Fourier Spectrum of synthetic and real images. Synthetic data again originates from the video game GTA V, whereas the downstream task is Semantic Segmentation. [87] employ Generative Adversarial Networks (GANs) to enhance the realism of simulated images of human eyes.

4.2 Spatiotemporal Models

Early work on video recognition focused on keypoint detection. Both sparse [57, 71] and dense [103, 104] method have been employed as spatiotemporal feature representations. With the surge in popularity of convolutional neural networks, two branches of video recognition research appeared: 2D (e.g., [88, 107]) and 3D CNNs (e.g., [12, 37, 101]). The 2D architecture of [88] is visualized in Figure 4.2. More recently, the computer vision

community has shown growing interest in 3D CNNs as pre-training them with the Kinetics datasets [11, 50] enables successful transfer learning for a multitude of video recognition tasks.

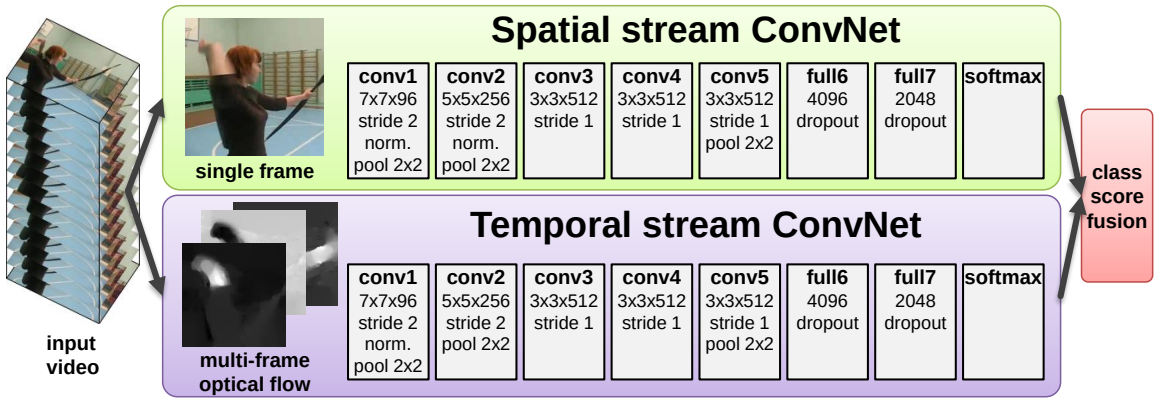


Figure 4.2: Overview of the architecture proposed in [88]. The spatial stream ConvNet operates on individual video frames, effectively performing action recognition from still images. On the other hand, thee input to the Temporal stream ConvNet is formed by stacking optical flow displacement fields between several consecutive frames. Such input explicitly describes the motion between video frames, which makes the recognition easier.

4.3 Action Recognition Datasets

Action Recognition datasets have undergone significant changes over time:

- Initial public benchmarks such as Weizmann [8] and KTH [86] were of small-scale and contained only a few thousands of samples.
- These were followed by medium-scale benchmarks such as UCF101 [89], ActivityNet [39] and HMDB51 [56].
- More recent datasets have significantly grown in size and consist of hundreds of thousands or even millions of annotated videos. Examples are Kinetics [11, 50], the Moments in Time dataset [74], as well as YouTube 8M [1].

Statistics for a subset of the aforementioned datasets can be seen in Table 4.2.

For the last case, a model pre-trained on such large-scale datasets can achieve significantly better results on smaller downstream tasks compared to a model trained from scratch [89, 39, 56].

However, the challenges of curating and defining label taxonomies for massive-scale datasets have shifted the focus towards pre-training on unlabeled videos or datasets with

Dataset	Year	Actions	Clips	Total	Videos
HMDB51 [56]	2011	51	min 102	6,766	3,312
UCF101 [89]	2012	101	min 101	13,320	2,500
ActivityNet200 [39]	2015	200	avg 141	28,108	19,994
Kinetics [50]	2017	400	min 400	306,245	306,245

Table 4.2: Statistics for some human action recognition datasets. 'Actions', specifies the number of action classes; 'Clips', the number of clips per class; 'Total', is the total number of clips; and 'Videos', the total number of videos from which these clips are extracted. Table adapted from [50].

weak supervision. As a result, the present work proposes a method to automatically construct large-scale synthetic datasets for pre-training. Such an approach addresses issues related to privacy, bias, ethics, and copyright.

Chapter 5

Fractal Images

Although, the main interest of the present work is the video modality, it is necessary to first examine key concepts from the simpler domain of images. Thus, this chapter provides an overview of fractals in two dimensions. First, Section 5.1 explains why this specific family of synthetic images was chosen as the backbone of this work. Next, Section 5.2 introduces the concept of iterated function systems and describes an algorithm that can render fractal images. Afterwards, Section 5.3 discusses parameter sampling strategies that yield fractal mages with desirable qualities. Lastly, Section 5.4 presents a modification that can enhance the diversity of the generated samples.

5.1 Introduction

By examining related literature, one can find numerous generative processes that can produce synthetic images. However, only one is chosen as the foundation of the present work: fractal images generated via the iterated function systems (IFS) technique. This choice is not arbitrary but relies on several significant factors:

- IFS fractals often reproduce patterns found in natural images. This is expected to narrow the domain gap between synthetic and real data.
- The algorithm that renders them is easy to implement.
- By randomly sampling parameters, it is possible to produce a near-limitless supply of diverse images. However, as will be seen later, the sampling process needs to constrained in order to avoid degenerate results.
- Such images have already been successfully utilized in the field of deep learning [49, 76, 2]. Therefore, positive results are also expected for the domain of this thesis, which is video.

It is worth mentioning that there exist other families of fractal images. These are briefly described in Section 5.5 along with explanations as to why they are discarded from the present work.

5.2 Classic Iterated Function Systems

A two-dimensional Iterated Function System (IFS) can be defined as a set of n functions $F_i : \mathbb{R}^2 \mapsto \mathbb{R}^2$. Every IFS is associated with an attractor, a set $S \in \mathbb{R}^2$ (and hence an image) which is the solution of the system. S is the fixed point of Hutchinson's recursive set equation [41]:

$$S = \bigcup_{i=1}^{n} F_i(S)$$

Following Barnsley [5], the aforementioned functions are linear transformations defined by a matrix $A_i \in \mathbb{R}^{2\times 2}$ and a vector $b_i \in \mathbb{R}^2$:

$$F_i(\boldsymbol{x}; A_i, b_i) = A_i\boldsymbol{x} + b_i = \begin{bmatrix} a_i & b_i \\ d_i & e_i \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} c_i \\ f_i \end{bmatrix}$$

In the rest of this document, the IFS parameters will be referred to as $W_i \in \mathbb{R}^6$ or $W \in \mathbb{R}^{N\times 6}$.

An approximation of the attractor of a given IFS can be rendered using the chaos game algorithm, which is described in Algorithm 3. At first, this algorithm initializes the output image as zeros and samples an initial point in two-dimensional space. At each iteration a function is sampled from the IFS and applied to the said point. Sampling is not uniform but based on the normalized absolute determinant of the weight matrices. The coordinates of the point are next quantized to pixels and the value of the resulting pixels is incremented by one. The previous step is discarded for the initial K_S steps as the resulting coordinates may not belong to the attractor. After completing K_I iterations, the image, which at this point is a two-dimensional histogram, is normalized to produce aesthetically pleasing visuals. Figure 5.1 displays the output image for various numbers of iterations K_I.

The result of such an approach is an image containing a random shape. Some examples can be seen in Figure 5.4. It is desirable for the shape to cover the entirety of the canvas. Shapes that constitute only a small portion of the canvas or are too large to be contained within it are not acceptable. To achieve this, Algorithm 3 is executed twice. Specifically, this algorithm produces real 2D coordinates. These coordinates are translated into pixel positions as follows:

Algorithm 3 `chaos-game`(F, K_I, K_S, H, W): Algorithm that renders IFS fractal images.

Input 1: N functions $\boldsymbol{F_i} : \mathbb{R}^2 \mapsto \mathbb{R}^2$
Input 2: Number of iterations and skipped iterations $\boldsymbol{K_I}, \boldsymbol{K_S}$
Input 3: Image dimensions $\boldsymbol{H}, \boldsymbol{W}$
Output: Grayscale Image $\boldsymbol{O} \in [0, 1]^{H \times W}$

Initialize: $O \leftarrow$ `zeros`(H, W)
$p_i \leftarrow \frac{|det(W_i)|}{\sum_{j=1}^{N} |det(W_j)|}$ ▷ Compute a probability for each W_i
Sample $x \sim U(-1, 1)$ ▷ Initialize random starting point $x \in \mathbb{R}^2$
for $step = 1$ to K_I **do**
    `Sample` $F^* \sim P$ ▷ Sample random transformation
    $x \leftarrow F^*(x)$ ▷ Apply transformation
    if $step \geq K_S$ **then** ▷ Ignore the first K_S steps
        $h_q, w_q \leftarrow quantize(x, H, W)$
        $O[h_q, w_q] \leftarrow O[h_q, w_q] + 1$
    end if
end for
$O \leftarrow log(O + 1)$
$O \leftarrow O/max(O)$ ▷ Normalize the image inside [0, 1]
return O

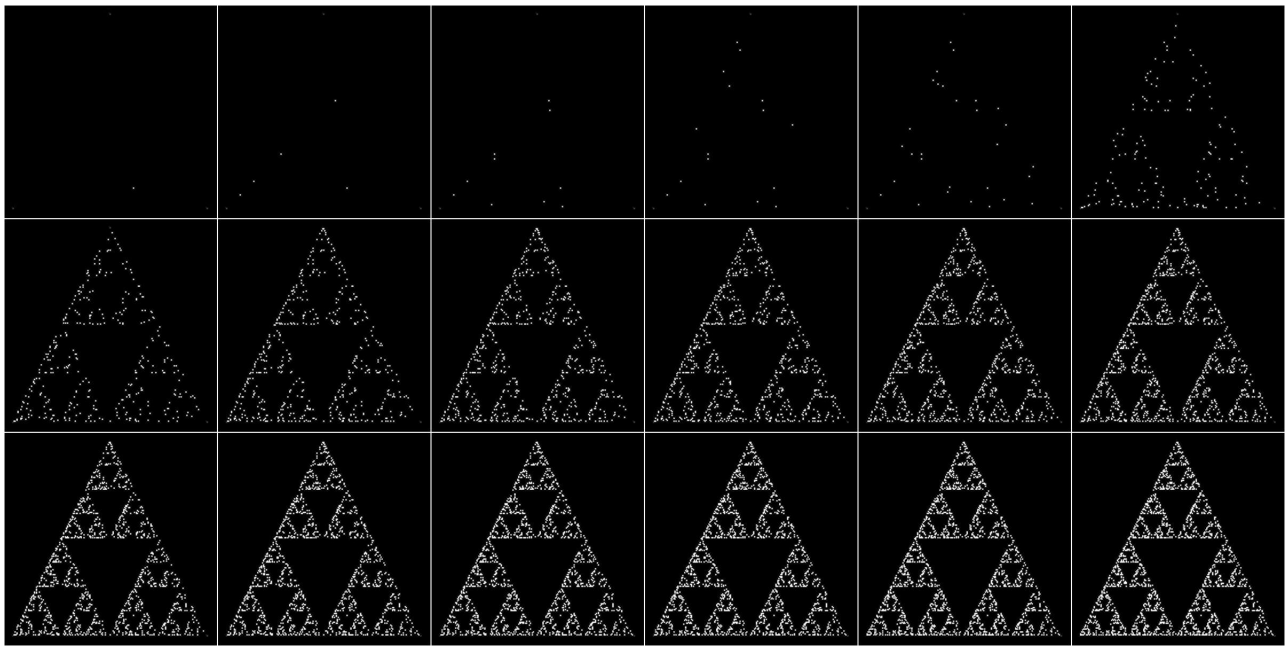


Figure 5.1: Chaos game algorithm in action. The order of frames is from left to right and them from top to bottom. As the number of iterations increases, a distinct shape is formed inside the canvas.

$$h_q = \frac{H-1}{B_h^2 - B_h^1}(h - B_h^1)$$
$$w_q = \frac{W-1}{B_w^2 - B_w^1}(w - B_w^1)$$

Here, (h, w) are the original real coordinates, (h_q, w_q) their quantized pixel indices, (H, W) the resolution of the image which is provided by the user and $(B_h^1, B_h^2, B_w^1, B_w^2)$ the real coordinates of a bounding box that should contain the fractal shape. Points that are outside of the bounding box are not rendered. The bounding box defines the position and shape of the canvas and is different for each fractal image. If the bounding box it too large, the resultant shape will cover only a small fragment of the canvas. If it is too small, only a portion of the shape will be displayed. To calculate a tight bounding box, we first render the fractal with an a default loose bounding box. We employ $(B_h^1, B_h^2, B_w^1, B_w^2) = (-5, 5, -5, 5)$. Afterwards, we detect a tighter bounding box within the canvas and repeat Algorithm 3. To reduce the amount of required computation, the first execution is performed with significantly lower resolution and less iteration steps.

Fractal IFS images are of particular interest for production of synthetic data. This is chiefly due to the fact that they have been shown to capture geometric properties of elements found in nature [68].

5.3 IFS Parameters

An IFS is defined by the number of functions N as well as the $6N$ function parameters. As the goal is to automatically generate large scale datasets of images, it is necessary to carefully investigate the relationship between these values and the quality of the generated samples.

In their seminal work, [49] propose to train visual classifiers not with real data (e.g. ImageNet [85]), but with a large automatically generated dataset of fractal images. As such, they sample N from $U(2, 3, \ldots, 8)$ [1] and the function parameters independently from $U(-1, 1)$. Having manually observed generated images, three types of behavior can be distinguished:

- Category 1 (Figure 5.2): Sparse attractors consisting of mostly blank space.
- Category 2 (Figure 5.3): Divergent attractors without structure.
- Category 3 (Figure 5.4): Neither sparse nor divergent attractors. Images with complex and varied structure. Often aesthetically pleasing.

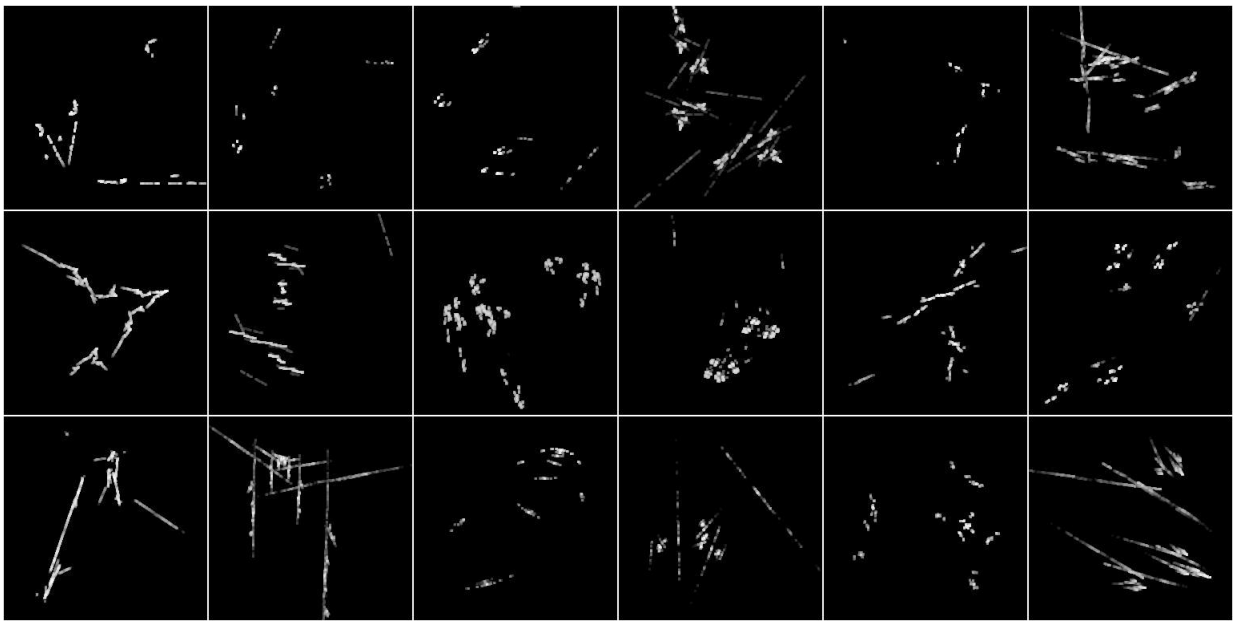


Figure 5.2: Examples of sparse attractors. Only a small percentage of pixels is filled. Such images are of no interest for the present work.

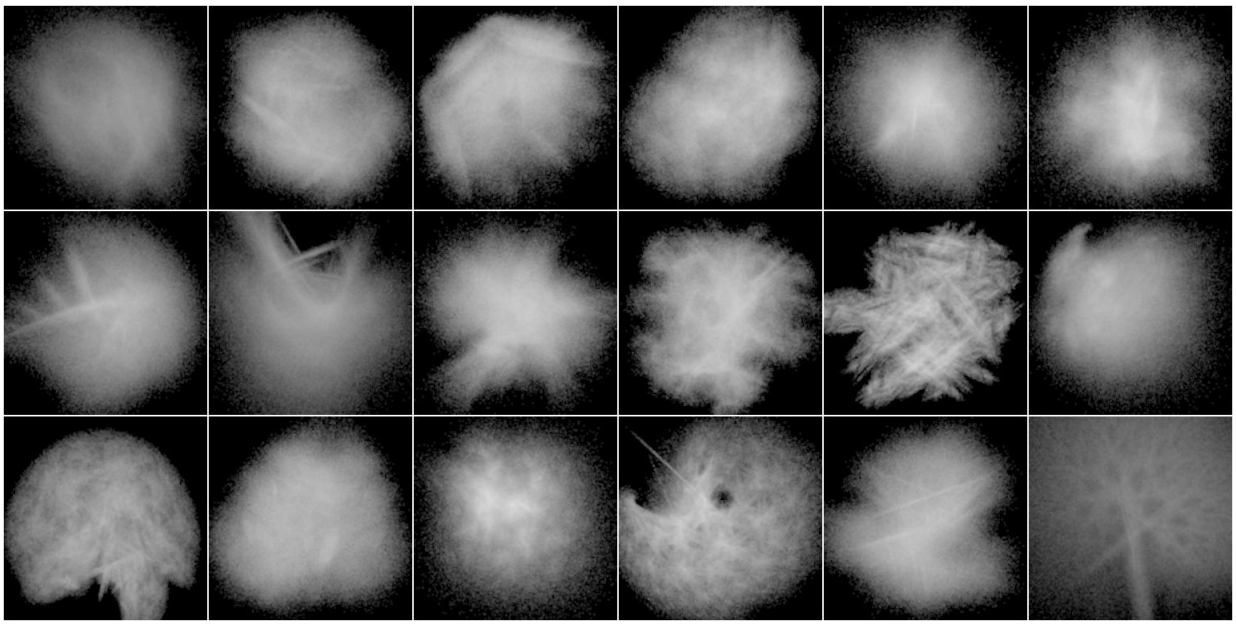


Figure 5.3: Examples of divergent attractors. Almost the entirety of the canvas is filled and no interesting shapes are displayed. Such images are also undesirable.

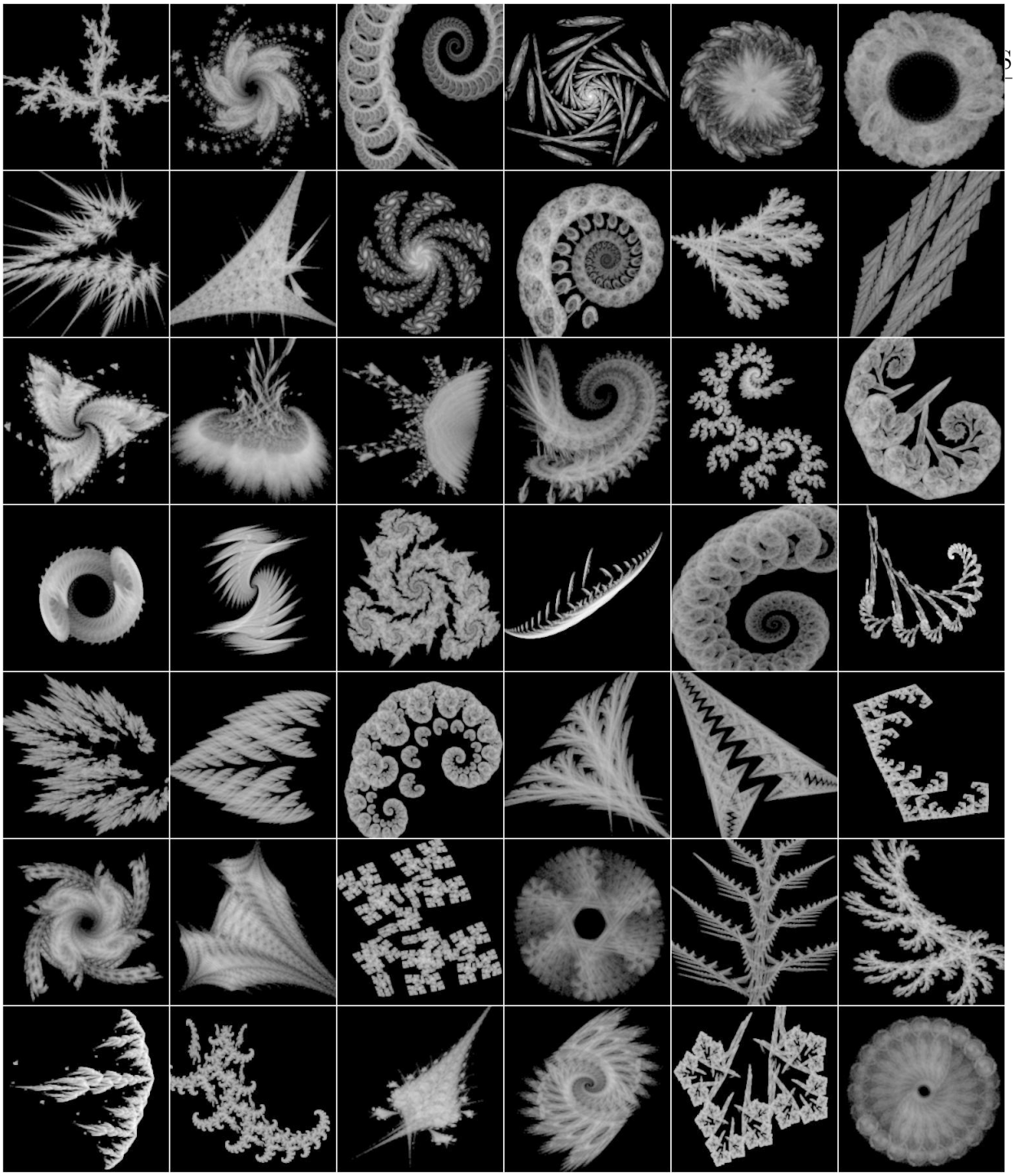


Figure 5.4: Examples of attractors with satisfactory structure.

It is desirable to minimize the number of samples belonging to the degenerate categories 1 & 2 as well as to maximize that of category 3. This can be achieved by applying stricter restrictions to the parameter space. To this end, [2] employ Singular Value Decomposition and rewrite the parameter matrix as:

[1]Throughout this thesis, $U(a, b)$ refers to the continuous uniform distribution whereas $U(\{\cdot\})$ refers to discrete uniform distribution.

$$A = U\Sigma V^T$$

- $U, V \in \mathbb{R}^{2\times 2}$ are orthogonal matrices and therefore can be represented as rotation matrices (with possible reflection, i.e. the determinant can be ± 1). Let $U = R_{\theta_1} D_1$ and $V^T = R_{\theta_2} D_2$ where R_x is a rotation matrix parameterized by angle x and D_i a diagonal matrix with diagonal elements $d_1, d_2 \in \{-1, 1\}$.
- $\Sigma \in \mathbb{R}^{2\times 2}$ is a diagonal matrix that contains the singular values $\sigma_1, \sigma_2 \in \mathbb{R}^+$ with $\sigma_1 > \sigma_2$.

Hence, A can be expressed as:

$$A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$$

To sample A one has to appropriately sample the parameters $\{\theta_1, \theta_2, \sigma_1, \sigma_2, d_1^1, d_1^2, d_2^1, d_2^2\}$, construct the intermediate matrices and multiply them as above.

After manual observation, the authors of [2] determine that the quality of the generated images is strongly correlated with the weighted sum of a system's singular values:

$$a = \sum_{i=1}^{N} (\sigma_{i,1} + 2\sigma_{i,2})$$

They further discover that the sampled images exhibit the desired qualities when the following constraint holds:

$$a_l = \frac{1}{2}(5 + N) \leq a \leq a_u = \frac{1}{2}(6 + N)$$

5.4 Fractal Flame

The Fractal Flame Algorithm [23] is an extension to the ordinary IFS, designed specifically to generate more aesthetically pleasing images. Such images are often employed as wallpapers and screensavers for personal computers [22]. Examples can be seen in Figure 5.5. Accordingly, fractal flames are distinguished by multiple innovations including:

- Non-linear functions
- Structural coloring

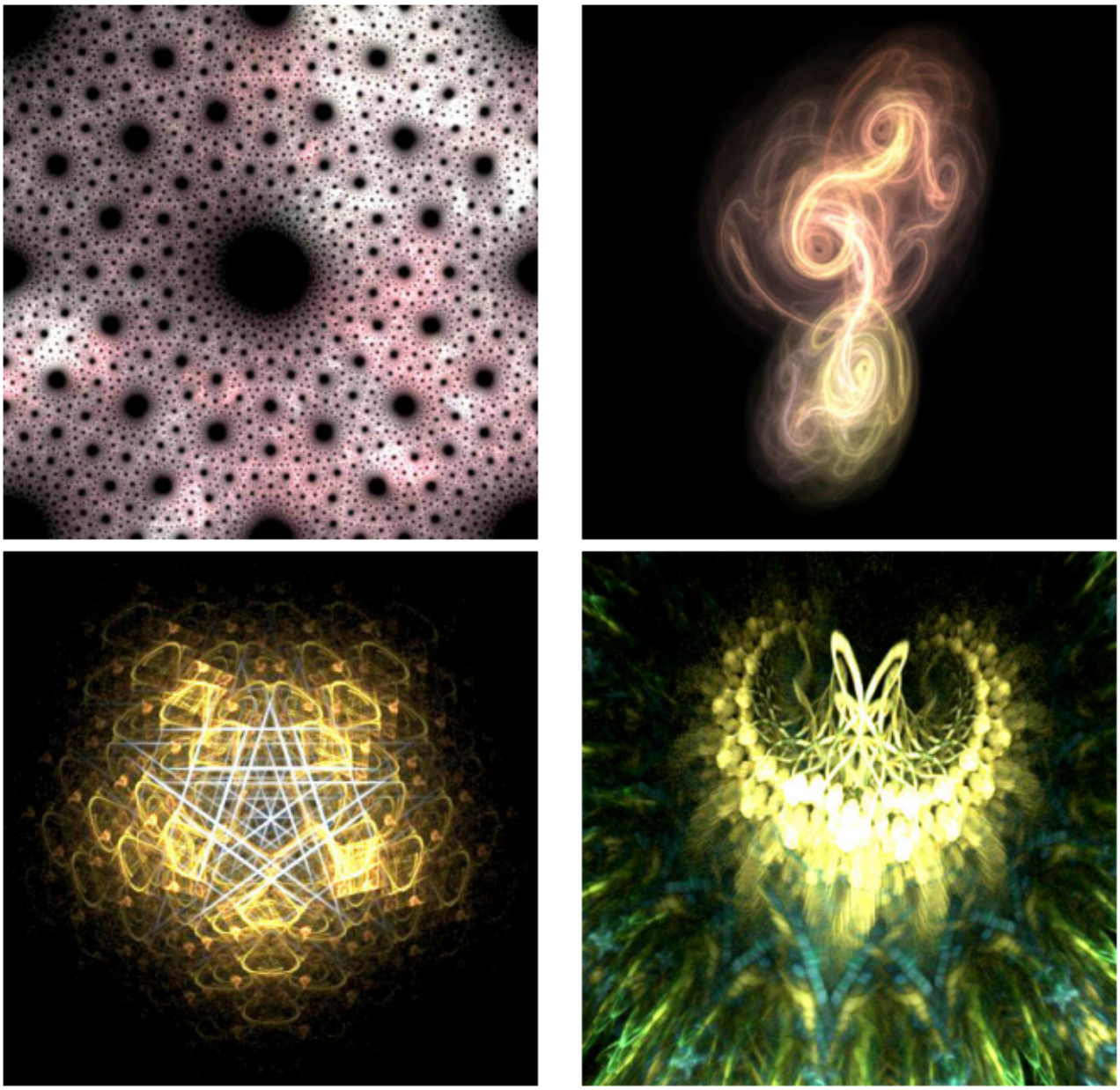


Figure 5.5: Examples of fractal flame images. Picture adapted from [23]. Although such images are much more aesthetically pleasing than simple gray scale fractals, they are also computationally more demanding. As such, the their only property of interest are nonlinearities that boost the diversity of the observed shapes.

- Combination of multiple IFS

Although all of the above features could be helpful, only a subset will be taken into

account within the proposed framework. Color is discarded because, although it greatly contributes to the aesthetics of the resultant image, it has no impact on the generated shape, which is deemed more important for the objective of this thesis. Likewise, combinations of different IFS are discarded due to computational constraints.

As for the modifications that are not discarded, a simple alteration can be applied to Algorithm 3. A non-linear function $G : \mathbb{R}^2 \mapsto \mathbb{R}^2$, which is referred to as variation can be applied to the coordinates between the lines 6 and 7 of the algorithm. The authors of [23] provide 49 such variations. A few examples are:

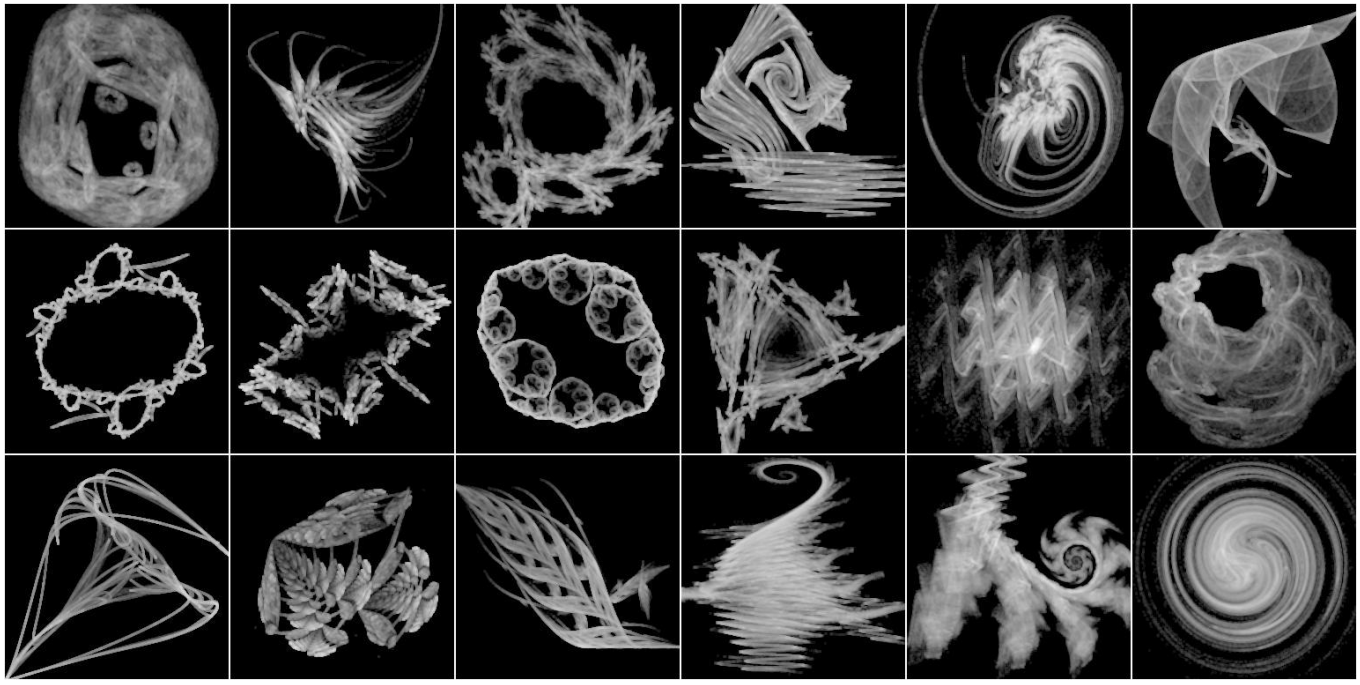


Figure 5.6: Examples of nonlinear attractors. The displayed shapes exhibit different patterns compared to original linear fractals. This is expected to significantly boost the overall diversity.

- $G_1(x, y) = (sinx, siny)$
- $G_6(x, y) = r(sin(\theta + r), cos(\theta - r))$
- $G_{16}(x, y) = \frac{2}{r+1}(y, x)$

In the above equations, r and θ are the polar coordinates: $r = \sqrt{x^2 + y^2}$ and $\theta = arctan(x/y)$.

Examples of non-linear fractal attractors can be seen in Figure 5.6. It is evident that the generated images differ significantly from the ordinary IFS. At the time of writing

this document, no other work has explored fractal flames in the context of deep learning. As shown in later chapters, including non-linear functions boosts the diversity of the generated samples.

5.5 Other Fractal Families

In this section some alternative generative processes of fractal nature are briefly described. However, this is done solely for inclusiveness. None of these methods will be a subject of later experiments. Of course, this dismissal is not arbitrary and proper justification is provided for each process.

5.5.1 3D IFS

By modifying the chaos game algorithm 3, it is possible to extend the IFS methodology from two dimensions to three. As such, the attractor is defined by $12N$ parameters instead of $6N$ and unlike previously is now a three dimensional point cloud. The attractor can be visualized as an image via a 3D rendering algorithm. Examples can be seen in Figure 5.7.

As the resulting images display three dimensional shapes, it is reasonable to assume that pre-training neural networks with such data will strongly enhance the model's spatial perception. Nonetheless, such an approach is still discarded for the following reasons:

- The current objective is to produce synthetic data for the task of video action recognition. It is true that in real videos the target can be captured from any angle in three dimensions. However, in the majority of cases, the observed motion is approximately two dimensional. Therefore, 3D fractals are not expected to significantly improve performance compared to 2D ones.
- It is possible to efficiently render such attractors in a pixel-wise manner [77]. However, this approach results in heavy aliasing which significantly obstructs the generated shapes.
- More advanced 3D rendering algorithms can alleviate the aliasing issue. Unfortunately, they will also significantly increase the time necessary to construct a dataset. As a single dataset can consist of hundreds of thousands videos and therefore millions of frames, such an approach is impractical.

As such, 3D IFS fractals will not be employed in the present document and are instead left for future work.

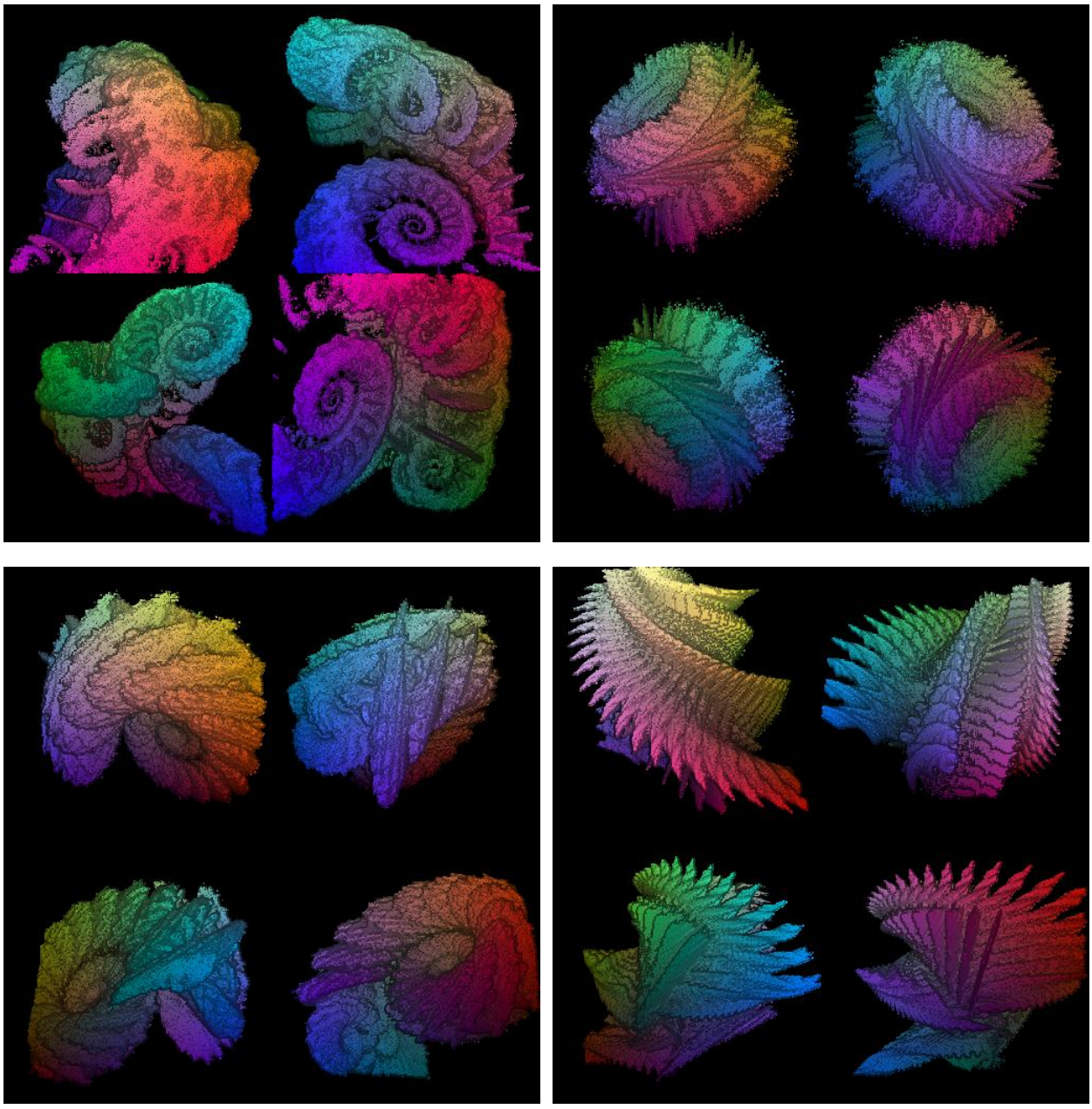


Figure 5.7: 3D IFS point clouds. Each of the 4 subfigures displays 4 different views of the three dimensional attractor. The above images have been hand-picked due to their aesthetic qualities and lack of artifacts. On the contrary, the majority of produced samples exhibit heavy aliasing.

5.5.2 Julia Fractals

In their simplest form, julia fractals are dependant on a complex function $f : \mathbb{C} \mapsto \mathbb{C}$. Common choices for f are simple polynomials $f(z) = z^n + c$, where $n \in \mathbb{R}$ and $c \in \mathbb{C}$. To

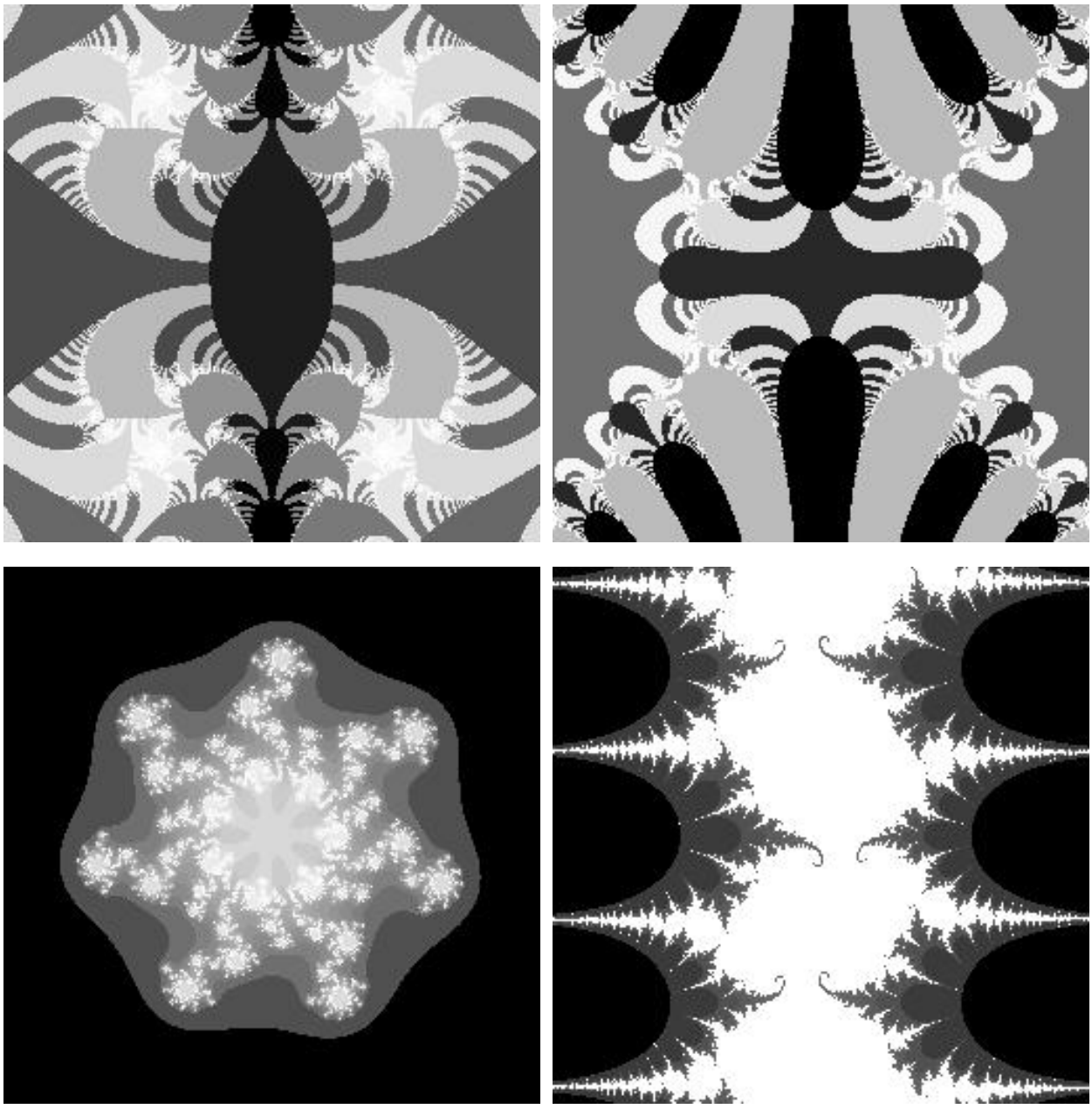


Figure 5.8: Julia fractals produced with different instances of the function f. Note the symmetry in the resultant images. These specific images have been hand-picked and therefore do not contain degeneracies while exhibiting diversity. On the contrary, randomly sampling parameters leads to significantly inferior results.

render a julia fractal, the following procedure is executed:

- For each pixel, the complex number z is initialized as the scaled coordinates of the pixel.

- Then, the function f is sequentially applied to z: $z_{k+1} = f(z_k)$.
- This step is repeated until either the magnitude of z surpasses a given threshold (z escapes) or the maximum number of allowed iterations is achieved. In the former case, the number of iteration necessary to escape is referred to as escape time.
- The result is a grayscale image where the pixels' values are their respective escape time.

As can be seen in Figure 5.8, julia fractals contain interesting spatial patterns and often stand out for their aesthetic qualities.

The aim of the present work is to automatically construct large datasets. Therefore, for a generative process to be acceptable, it must be able to produce a great variety of images by randomly sampling parameters. However, by sampling different functions f, two adverse phenomena were observed in large quantities:

- Duplicate images
- Degenerate empty images

The authors of the present work could not resolve these issues and conclude that the parameters of julia fractals must be chosen manually. This obstructs automatic generation of datasets and therefore this generative process will not be employed in later experiments. As such, Julia Fractals are also left for future work to explore.

Chapter 6

Synthetic Videos

The following chapter is the most important part of the present work and contains the majority of its contributions. Having previously reviewed the concept of fractal images, the objective of this chapter is to add a time dimension and produce elaborate video animations containing a variety of spatio-temporal patterns. As such, Section 6.1 describes a straightforward animation method for fractals which leads to unsatisfactory results. Hence, Section 6.2 proposes a more intricate solution that resolves the previous issue. Afterwards, with the objective of reducing the domain gap, 6.3 lists manually observed properties of real videos and suggests methods of incorporating them in the synthetic ones. Section 6.4 further suggests techniques to automatically split videos into fixed categories, something that is necessary for supervised learning. Lastly, Section 6.5 presents additional generative video processes that will be compared against fractals in later experiments.

6.1 Naive IFS Interpolation

The current objective is to extend fractal images in the time dimension and produce animations. The basis for producing fractal animations is provided by the following theorem:

Theorem 3 *[5] Let $\{W_i\}_{i=1}^N$ be an IFS whose maps are parameterized by a single bounded variable $t \in \mathbb{R}$. Then, the function $S(t)$ that maps the parameter t into the attractor of the IFS parameterized by t is continuous.*

The above theorem can be alternatively interpreted as follows: continuous changes to parameters of the IFS result in continuous changes in the rendered image. It is possible to produce smooth animations by slightly altering the IFS parameters of each consecutive frame.

As such, a very simple animation method can be achieved by first sampling parameters for two different fractal images. These images will serve as the first and last frames of

the video respectively. The only constraint is that number of functions N must be shared by both images. In practise N is sampled from $U(\{3, \ldots, 8\})$. To produce motion, the parameters of the two images are linearly interpolated. More details can be seen in Algorithm 4). The result is an additional dimension, the time dimension. As requested, this approach produces a continuous sequence of iterated function systems. Each IFS can be rendered separately and in parallel via Algorithm 3. This results in a sequence of frames and therefore a video.

Algorithm 4 `sample-video-naive`(N, T): Creates an IFS sequence by simply interpolating parameters.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Sequence of parameters: $W \in \mathbb{R}^{T \times N \times 6}$

1: Sample $W_{start}, W_{end} \in \mathbb{R}^{N \times 6}$ ▷ Parameters for the first & last frame
2: **Initialize:** $W \leftarrow \texttt{zeros}(T, N, 6)$
3: **for** $t = 0$ to $T - 1$ **do**
4: $\quad W[t] \leftarrow \frac{T-1-t}{T-1} W_{start} + \frac{t}{T-1} W_{end}$ ▷ Linear interpolation
5: **end for**
6: **return** W

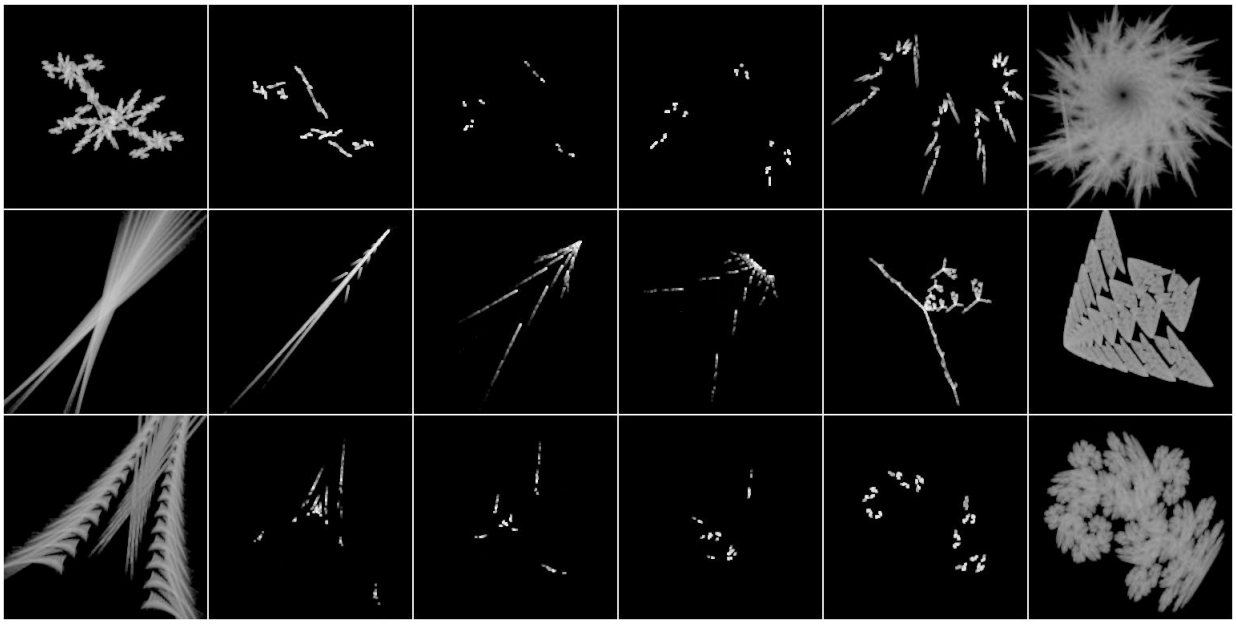


Figure 6.1: Examples of degenerate behavior. Each row illustrates a sequence of IFS images. Naive linear interpolation of IFS parameters causes undesired sparseness in the intermediate frames. Such behavior is unacceptable for large-scale pre-training of neural networks and must be addressed.

It is desirable for the resultant shapes to cover the entirety of the canvas. For the modality of images, a solution was proposed in Section 5.2. Fractal images require a

bounding box $(B_h^1, B_h^2, B_w^1, B_w^2)$ and a tight one can be identified by rendering the image twice. To compute an appropriate bounding box for a video, we first render the first and last frames with a loose bounding box. Their tight bounding boxes are denoted as $(B_{h,s}^1, B_{h,s}^2, B_{w,s}^1, B_{w,s}^2)$ and $(B_{h,e}^1, B_{h,e}^2, B_{w,e}^1, B_{w,e}^2)$ respectively. The bounding box for the full video is computed by identifying the maximal thresholds in all four directions:

$$
\begin{aligned}
B_h^1 &= min\left(B_{h,s}^1, B_{h,e}^1\right) \\
B_h^2 &= max\left(B_{h,s}^2, B_{h,e}^2\right) \\
B_w^1 &= min\left(B_{w,s}^1, B_{w,e}^1\right) \\
B_w^2 &= max\left(B_{w,s}^2, B_{w,e}^2\right)
\end{aligned}
$$

The resultant bounding box is applied to all frames of the video. Only the first and last framed have to be rendered twice. Intermediate frames are rendered only once.

Unfortunately, thorough manual examination of videos produced by this method reveals an adverse phenomenon present in a large portion of the generated samples. Although the beginning and end of the resultant animation are satisfactory, the intermediate attractors are not. Intermediate frames tend to be extremely sparse, leaving the majority of the frames blank. Such samples are visualized in Figure 6.1. We hypothesize that such videos are not appropriate for pre-training neural networks. The shape of the attractor is required to remain nontrivial throughout the entirety of the video. Therefore, present behavior is undesirable. Naive linear interpolation of IFS parameters is not an acceptable animation method. The following section will present an alternative animation technique that will mitigate this shortcoming.

6.2 Decomposed IFS Interpolation

Undesired sparseness of intermediate frames can be alleviated using Singular Value Decomposition. As explained in Section 5.3, the IFS matrix can be expressed as $A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$. Accordingly, instead of directly interpolating IFS parameters, this document proposes to interpolate each sub-matrix separately. To sample a random animation, the following steps should be followed:

1. Sample D_1 and D_2 which will remain constant for each timestep. These matrices are shared by all frames since they contain integer values and interpolation cannot be applied to them. Construction of these matrices can be achieved by sampling values from $\{-1, 1\}$ and is described in detail in Algorithm 6.

2. Sample the angles $\theta_1^{start}, \theta_1^{end}$ from $U(0, 2\pi)$, interpolate them in the time dimension and construct the rotation matrix R_{θ_1} for each timestep. The same process applies to R_{θ_2}. More details can be seen in Algorithm 5.

3. Sample $\Sigma^{start}, \Sigma^{end}$ and interpolate in the time dimension to produce Σ. These matrices are the most important parameter component and incorrect values may lead to unsatisfactory results. As such, sampling is done under specific constraints. More details can be seen in Algorithm 7, Section 5.3 as well as Appendix A of [2].

4. Compose the parameter matrix by multiplying the resultant sub-matirces: $A = R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2$.

5. Sample bias parameters b^{start}, b^{end} and interpolate them in the time dimension. This step is identical to the previous naive approach and no intricate modifications are necessary. More details can be seen in Algorithm 8. Lastly, the resultant bias is concatenated to the composed matrix A to form the final IFS parameters.

This procedure is adapted from [10] and described in detail in Algorithm 9. The resultant samples no longer suffer from sparseness and exhibit adequate shape throughout the entirety of the animation. Such samples can be seen in Figure 6.2. This approach will serve as a backbone for producing large-scale synthetic datasets for pre-training neural networks. However, as will be made clear in later sections, the proposed approach is still too straightforward and not yet complete. Although the sparseness issue has been mitigated, many possible enhancements can still be applied to narrow the domain gap between real and synthetic videos.

Algorithm 5 `sample-rotation`(N, T): Sample interpolated rotation matrix.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Rotation matrices $R_\theta \in \mathbb{R}^{T \times N \times 2 \times 2}$

Initialize: $R_\theta \leftarrow$ `zeros`$(T, N, 2, 2)$
Sample $\theta^{start}, \theta^{end} \in \mathbb{R}^N \sim U(0, 2\pi)$
$\theta \leftarrow$ `interp`$(\theta^{start}, \theta^{end}, T) \in \mathbb{R}^{T \times N}$
for $t = 0$ to $T - 1$ **do**
    for $n = 0$ to $N - 1$ **do**
        $R_\theta[t, n] \leftarrow$ `rot_matrix`$(\theta[t, n]) \in \mathbb{R}^{2 \times 2}$
    end for
end for
return R_θ

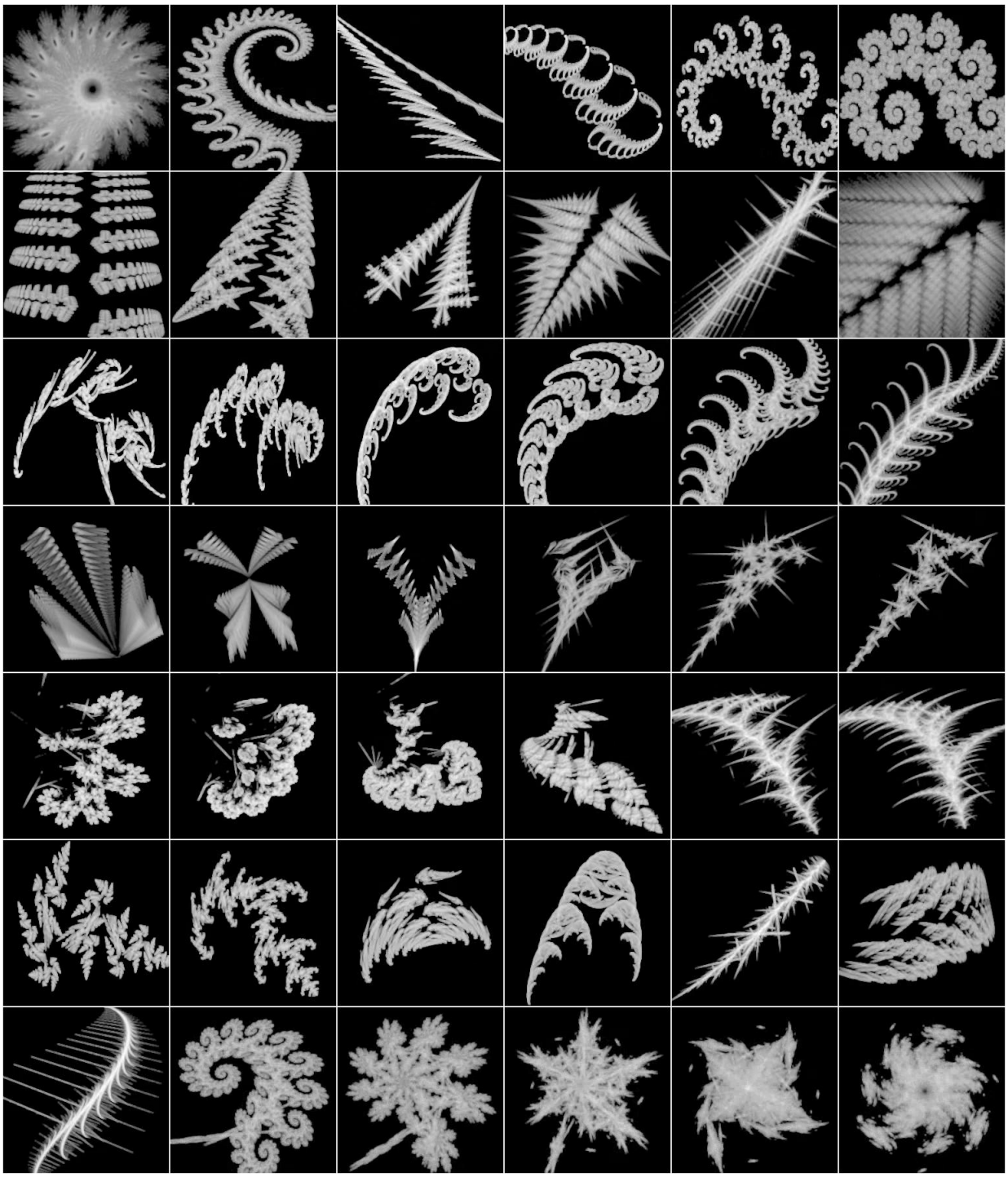


Figure 6.2: Examples of decomposed interpolation. Each row illustrates a sequence of IFS images. The issue of sparseness is resolved and attractors change appearance without shrinking.

Algorithm 6 `sample-delta(`N`)`: Sample delta matrix.

Input 1: Number of transformations N
Output: Delta matrices $D \in \mathbb{R}^{N\times 2\times 2}$

Initialize: $D \leftarrow \texttt{zeros}(N, 2, 2)$
`Sample` $\bar{D} \in \{-1, 1\}^{N\times 2}$
for $n = 0$ to $N - 1$ **do**
    $D[n] \leftarrow \texttt{diag}(\bar{D}[n]) \in \mathbb{R}^{2\times 2}$
end for
return D

Algorithm 7 `sample-sigma(`N, T`)`: Sample interpolated sigma matrix.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Sigma matrices $\Sigma \in \mathbb{R}^{T\times N\times 2\times 2}$

Initialize: $\Sigma \leftarrow \texttt{zeros}(T, N, 2, 2)$
Sample $a^{start}, a^{end} \in \mathbb{R} \sim U(\frac{1}{2}(5 + N), \frac{1}{2}(6 + N))$
$\bar{\Sigma}_{start} \leftarrow \texttt{sample-svs}(N, a^{start}) \in \mathbb{R}^{N\times 2}$ ▷ Appendix A from [2]
$\bar{\Sigma}_{end} \leftarrow \texttt{sample-svs}(N, a^{end}) \in \mathbb{R}^{N\times 2}$
$\bar{\Sigma} \leftarrow \texttt{interp}(\bar{\Sigma}_{start}, \bar{\Sigma}_{end}, T) \in \mathbb{R}^{T\times N\times 2}$
for $t = 0$ to $T - 1$ **do**
    for $n = 0$ to $N - 1$ **do**
        $\Sigma[t, n] \leftarrow \texttt{rot\_matrix}(\bar{\Sigma}[t, n]) \in \mathbb{R}^{2\times 2}$
    end for
end for
return Σ

Algorithm 8 `sample-bias(`N, T`)`: Sample interpolated bias.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Bias vectors $b \in \mathbb{R}^{T\times N\times 2}$

Initialize: $R_\theta \leftarrow \texttt{zeros}(T, N, 2, 2)$
Sample $b^{start}, b^{end} \in \mathbb{R}^{N\times 2} \sim U(-1, 1)$
$b \leftarrow \texttt{interp}(b^{start}, b^{end}, T) \in \mathbb{R}^{T\times N\times 2}$
return b

Algorithm 9 `sample-video-decomposed`(N, T): Sample IFS parameters for an animation by interpolating each sub-matrix separately.

Input 1: Number of transformations N
Input 2: Number of frames T
Output: Sequence of parameters: $W \in \mathbb{R}^{T \times N \times 6}$

$R_{\theta_1} \leftarrow \texttt{sample-rotation}(N, T) \in \mathbb{R}^{T \times N \times 2 \times 2}$
$R_{\theta_2} \leftarrow \texttt{sample-rotation}(N, T) \in \mathbb{R}^{T \times N \times 2 \times 2}$
$D_1 \leftarrow \texttt{sample-delta}(N) \in \mathbb{R}^{N \times 2 \times 2}$
$D_2 \leftarrow \texttt{sample-delta}(N) \in \mathbb{R}^{N \times 2 \times 2}$
$\Sigma \leftarrow \texttt{sample-sigma}(N, T) \in \mathbb{R}^{T \times N \times 2 \times 2}$
$A \leftarrow R_{\theta_1} D_1 \Sigma R_{\theta_2} D_2 \in \mathbb{R}^{T \times N \times 2 \times 2}$ ▷ Compose A
$b \leftarrow \texttt{sample-bias}(N, T) \in \mathbb{R}^{T \times N \times 2}$
$W \leftarrow \texttt{reshape}(\texttt{concat}(A, \texttt{expand}(b))) \in \mathbb{R}^{T \times N \times 6}$
return W

6.3 Domain Gap

The previous section described a method to automatically generate videos containing interesting spatiotemporal patterns. The objective now is to utilize the aforementioned framework to pre-train strong visual representations suitable for the task of video action recognition.

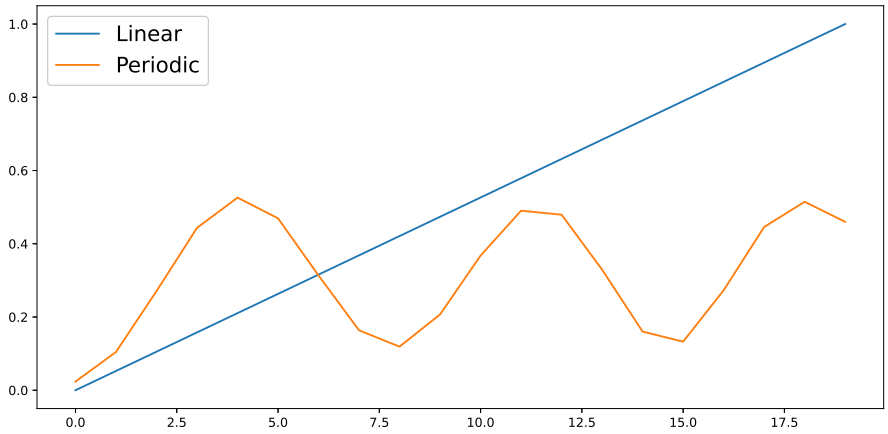
For synthetic images, previous work [4] concludes that performance on real data is improved when the former captures structural properties of the latter. Likewise, large-scale studies on pre-training [17, 53, 94] deduce that the effectiveness of many pre-training frameworks significantly deteriorates when the source and target domains are different. As such, it is necessary to narrow the domain gap between synthetic fractal videos and samples from real action recognition benchmarks.

To do so, this section lists manually observed characteristics of real action recognition data [89, 56, 39] as well as methods to emulate them within the pre-training framework. The objective of the emulation is not absolute realism, but simple and intuitive approximation. The majority of the proposed emulation methods are designed as uncomplicated transformations that can be efficiently implemented as online augmentations under computational constraints.

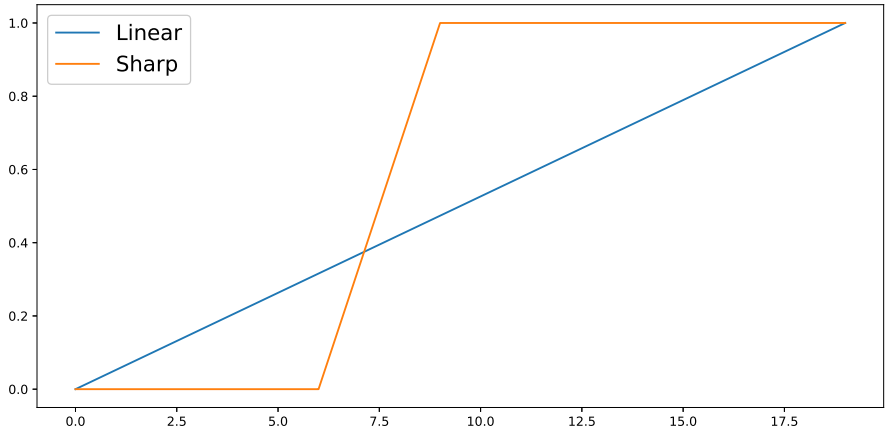
6.3.1 Non-linear Motion

The previously described video synthesis method produces simple "forward" motion from state A to state B. This is not enough to approximate real human motion, which is significantly more complex. To narrow this gap, instead of applying straightforward linear interpolation between the first and last frame, more intricate interpolation functions are employed (Figure 6.3):

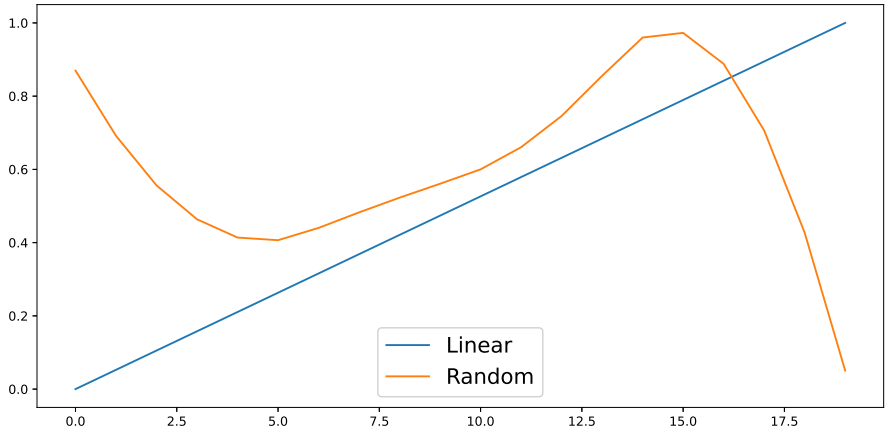
- **Sinusoidal Interpolation**: A significant portion of human activity consists of periodic or quasi-periodic motion. Common examples are walking and any type of repetitive exercise. To simulate such motion, a noisy sine function can be used as interpolant. It is noteworthy that a similar approach [60, 24] has been previously employed for the task of periodicity detection and repetition counting. Such approaches construct large unlabeled video collections by sampling short clips of varying lengths and repeating them with different periods and counts.
- **Sharp Interpolation**: A different category of activity is characterized by quick, sharp and sudden motion. Instances of such behavior are punches in boxing as well as penalty kicks in football. This motion is approximated by a linear interpolant with a significantly larger slope magnitude. The linear interpolant is placed in random timestep while the beginning and end of the produced curve is padded with zeros and ones respectively.



(a) Periodic interpolant



(b) Sharp interpolant



(c) Random interpolant

Figure 6.3: Proposed interpolation curves. The objective is to mimic motion observed in real videos and to narrow the existent domain gap.

- **Random Interpolation**: Other miscellaneous activity without clear patterns can be simulated by a random waveform. To produce such a function, a sequence of real numbers is initially randomly sampled from $U(0, 1)$. Further, the the sequence is made smooth via one dimensional quadratic interpolation [1]. The result is a random curve. Admittedly, such curves bear similarities to the previously proposed sinusoidal approach and sometimes cannot be differentiated from them.

A different aspect of human motion which has not been addressed yet is its compositionality, i.e. intricate motions are made up of multiple simple ones. For instance, running consists of a periodic movement of the legs as well as a different periodic movement of the arms.

An IFS can also be considered composite as it is comprised of multiple linear functions. As each function is responsible for a different facet of the resultant shape, composite motion can be replicated. By assigning each IFS function a different interpolant, multiple motion patterns can be observed in the video. However, doing so without any constraints often results in a incoherent oscillation that lacks the structure and rhythm observed in real human motion.

To mitigate this issue, the following procedure is followed. First, the set of chosen interpolants is initialized as either one or two random samples from the three newly introduced functions. Next, the linear interpolant is added to the set. Lastly, a single interpolant is randomly chosen from the set and applied to every IFS function or each IFS function receives a separate random sample from the set. The result is a video containing between one and three different interpolation curves. As such, this method produces composite and coherent motion. The process in described in detail in Algorithm 10.

It is obvious that the aforementioned approach cannot be implemented as an online augmentation. It is executed offline during the construction of the synthetic dataset.

6.3.2 Diversity

Previous work on images [4] suggests that a key property for learning good representations is diversity of the synthetic data. As such, in order to increase the variance of the produced animations, nonlinear fractals (see Section 5.4) are also included in the proposed framework.

People displayed in action recognition videos possess unambiguous shape and contours. On the other hand, multiple nonlinear functions from the fractal flame algorithm [23] result in attractors that either possess nebulous shape or are too sparse. Assuming that the the former property is beneficial for learning stronger visual representations, only functions leading to dense attractors with distinct shape and contours were selected. These are variations $0, 4, 6, 13, 14, 15, 16, 20, 27$ and 29, with 0 being the identity function that does not affect the output. More details can be seen in the appendix of [23].

[1] The authors employ `scipy.interpolate.interp1d`

Algorithm 10 `sample-interpolant`(T, N): Sample interpolants for IFS parameters.

Input 1: Number of frames in clip T
Input 2: Number of functions in IFS N
Output: Interpolants for each function $O \in \mathbb{R}^{N \times T}$

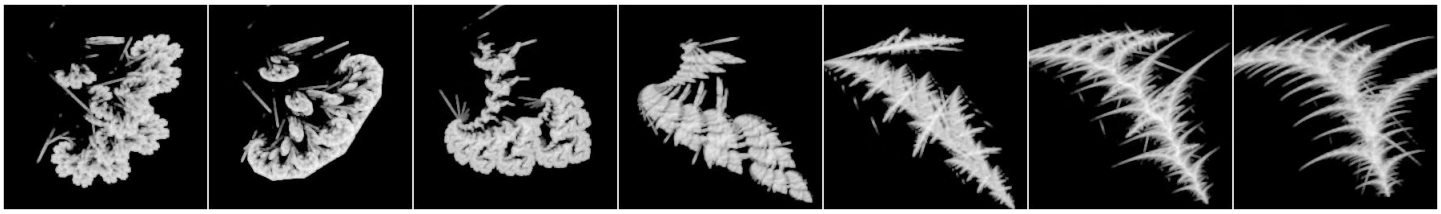
Initialize: $O \leftarrow \texttt{zeros}(N, T)$
$all \leftarrow \{\texttt{interp-sin}, \texttt{interp-sharp}, \texttt{interp-random}\}$
$single \leftarrow \texttt{rand}() < prob_{single}$
Sample $N_{fn} \sim U(\{1, 2\})$
$chosen \leftarrow \texttt{random-sample}(all, N_{fn})$ ▷ List of functions
$chosen \leftarrow chosen \cup \{\texttt{interp-linear}\}$
for $i = 0$ to N_{fn} **do**
    $chosen[i] \leftarrow chosen[i](T)$ ▷ List of arrays
end for
for $i = 0$ to $N - 1$ **do**
    if single **then**
        $O[i] \leftarrow chosen[0]$
    else
        $O[i] \leftarrow chosen[\texttt{randint}(0, N_{fn})]$
    end if
end for
return O

As before, nonlinearities are not an online augmentation, but a fragment of the offline rendering procedure. It is worth mentioning that the aforementioned nonlinear functions increase the amount of required computation and therefore significantly slow down the rendering process for synthetic videos. This is of paramount importance as later experiments will require hundreds of thousands of such samples. Hence, nonlinear fractals will constitute only 50% of the produced datasets, with the reaming 50% being simple linear fractals. The previously proposed motion curves are unaffected by nonlinearities and will be applied to both types of fractals.

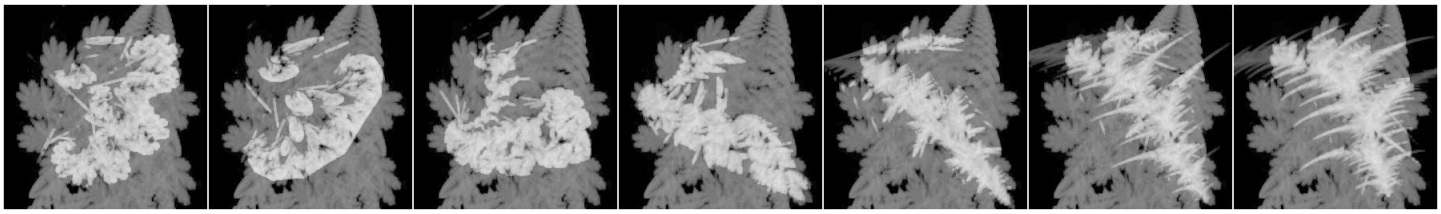
6.3.3 Static Background

Scenes in action recognition benchmark samples consist of a foreground (person performing some action) and a background (animate or inanimate environment around the person). The latter could, for instance, be a golf course or swimming pool. In the most simple case, the background is completely static and the only motion in the video originates from the foreground.

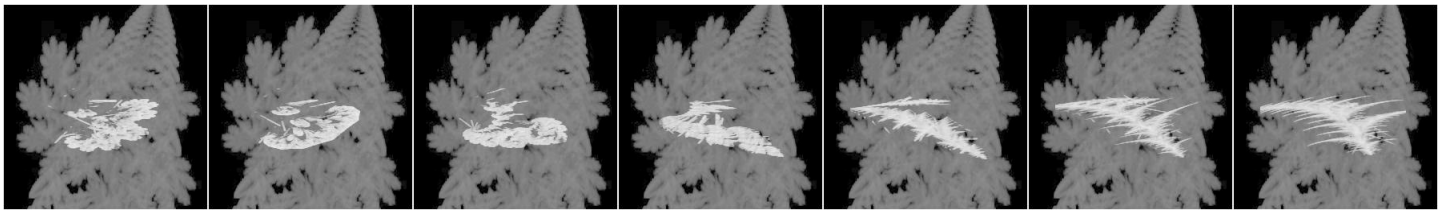
Hitherto, the generated synthetic videos lack this property as they only display a single moving shape without any surroundings. This can be implemented straightforwardly as an online augmentation. During training, for each sample x_i within a batch, a static frame is sampled from a different video x_j and mixed with every frame of x_i via weighted



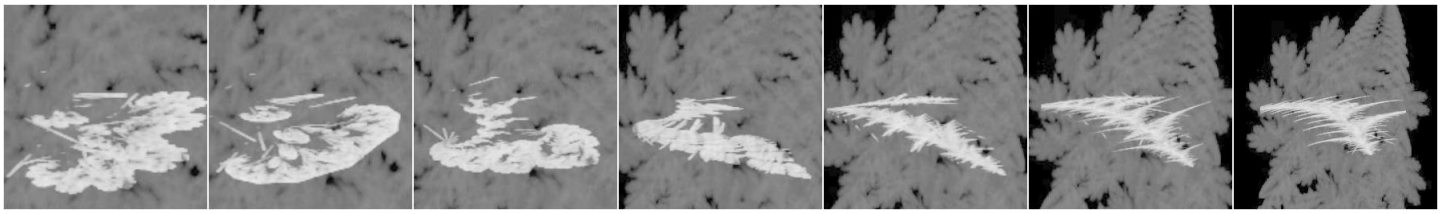
(a) Original video



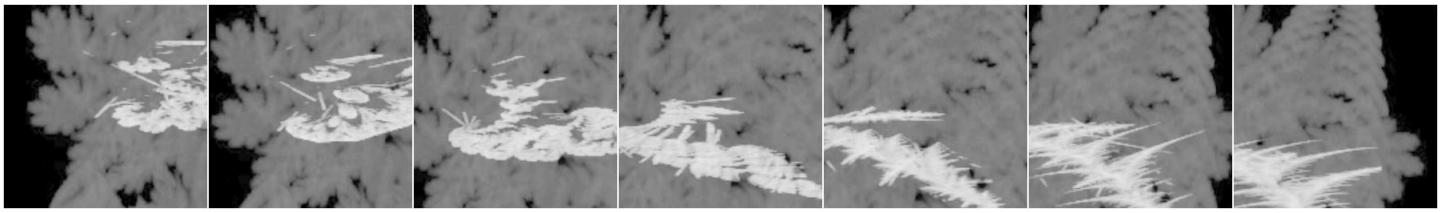
(b) Static background



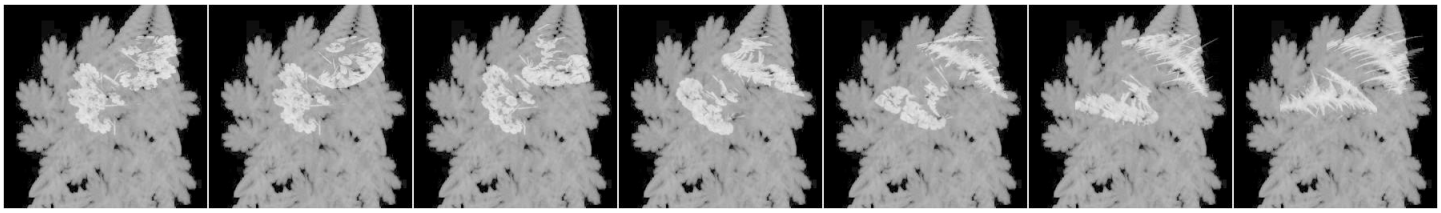
(c) Static background + Foreground scale



(d) Static background + Foreground scale + Camera zoom



(e) Static background + Foreground scale + Camera displacement



(f) Static background + Foreground scale + Group activity

Figure 6.4: Proposed augmentations in action.

sum:

$$\tilde{x_i} = (1 - a)x_i + ax_j[f]$$

Here, x_i, x_j are two samples from the same batch, $f \sim U(\{0, \ldots, N_{frames} - 1\}$ and $a \sim U(a_{min}, a_{max})$. In practise, we use $a_{min} = 0.25$ and $a_{max} = 0.55$.

In real videos background usually covers the entirety of the screen. On the other hand, produced fractals are usually concentrated in the middle with sides and corners of the screen remaining empty. To mitigate this inconsistency, two additional steps are taken before mixing static background frames with videos. First, instead of sampling a single frame for each video, $N_{back} = 2$ are sampled. They are next aggregated into a single frame with the max operation. Second, a random rectangle is cropped from the resultant image and is interpolated to input dimensions. After these modifications, the background shape should be distributed uniformly around the canvas and not in its center.

This procedure is reminiscent of a widely used augmentation technique in the image domain called Mixup [116], However, Mixup also applies a similar operation to the associated labels. This does not occur in the proposed operation, where the labels remain unaltered and only the input is modified.

A similar approach has been adopted by [106, 21] to combat an adverse phenomenon known as background bias. In short, backward bias is the tendency of 3D convolutional neural networks to rely on static background as a shortcut as well as their failure to capture the motion information. Therefore, this augmentation is expected to enhance the model's ability to distinguish between foreground and background when trained with real data. However, for datasets where background is shared amongst all samples, the effect of this modification is unknown and will be examined in later experiments.

6.3.4 Dynamic Background

The previous approach takes for granted that the background remains motionless throughout the entire video. This is often not true as the video may depict dynamic background: bystanders or a moving environment such as water waves or trees swaying in the wind. So far, such behavior has not been incorporated in the proposed pre-training framework and it is expected that the model will fail to differentiate between dynamic background and foreground.

To approximate such phenomena, the previous approach is modified. The given video is mixed not with a static frame but with a sequence of frames sampled from a different video. In such a sequence, at each timestep, the frame index is incremented by one, remains unchanged, or is decremented by one. Additionally, assuming the foreground motion has greater magnitude than the background motion, the difference between the maximum and minimum frame indices is constrained. The process of sampling background frame indices is described in detail in Algorithm 11. As previously, the label is

unaffected. This augmentation is expected to boost the model's resilience against miscellaneous background motion.

The number of the proposed domain adaptation techniques is quite large and our computational resources are limited. To correctly evaluate the value of all techniques, a separate ablation experiment will need to be conducted for each one of them. As such, in later experiments static background and dynamic background will be merged into a single online augmentation and will not be evaluated separately. For each video in a batch, we sample $p \sim U(0, 1)$. If $p < p_{dynamic} = 0.2$, dynamic background is employed. Static background is employed otherwise.

Algorithm 11 `sample-back-frames(`T`)`: Sample frame indices for dynamic background.

Input 1: Total number of frames in clip T
Output: Frame indices $\texttt{idxs} \in \mathbb{Z}^T$

$\texttt{diff\_max} \leftarrow \lfloor 0.25 * T \rfloor$
Sample $\texttt{offset} \in \mathbb{Z} \sim U(\{0, \dots, T - \texttt{diff\_max}\})$
Sample $\texttt{diff} \in \mathbb{Z}^T \sim U(\{-1, 0, 1\})$
$\texttt{diff}[0] \leftarrow 0$
$\texttt{idxs} \leftarrow \texttt{cumsum}(\texttt{diff})$
$\texttt{idxs} \leftarrow \texttt{clamp}(\texttt{idxs}, 0, \texttt{diff\_max} - 1)$
$\texttt{idxs} \leftarrow \texttt{idxs} + \texttt{offset}$
return idxs

6.3.5 Foreground Scaling

In our synthetic videos, fractal shapes cover a large portion of the canvas and are usually positioned around its center. On the contrary, in real action recognition videos, position and size of the foreground are not fixed but random. This contradiction must be addressed within the pre-training framework. The solution is a simple augmentation that is applied online during training and not offline during rendering.

As synthetic videos are rendered with the fractal covering the majority of the screen, no cropping is applied. Instead, videos can be downsampled in the two spatial dimensions with scales $s_h, s_w \sim U(s_{min}, s_{max})$ where $0 < s_{min} \leq s_{max} \leq 1$ and placed in a random position of an empty canvas. In practise, we set $s_{min} = 0.3$ and $s_{max} = 1.0$. This procedure randomizes both the size and the position of the moving shape. Decreasing the size of the moving shape is expected to to strengthen the model's ability to detect details. This is of high importance as the observed benchmarks contain fine-grained classes such as facial expressions and small objects.

6.3.6 Group Activity

A large portion of videos do not display a single person but a group performing identical or almost identical activities. Common examples are team sports such as volleyball

and football. This can be approximated with a modification to the previously proposed foreground scaling. Specifically, after the interpolation step, the synthetic video is copied N_{clone} times with each copy receiving a different mild augmentation. In later experiments, N_{clone} is set to 2. Augmentations include random rotation, horizontal flipping and temporal offset. The last one renders the copies asynchronous. As before, each resultant copy is then placed in a random location of an empty canvas. Unlike before, we set $s_{min} = 0.2$ and $s_{max} = 0.5$, since multiple copies of large shapes are more likely to overlap.

6.3.7 Perspective

In real videos the foreground can be captured by a camera from any angle in three dimensions. On the other hand, synthetic videos are rendered in two dimensions and therefore lack this variance. As a compromise, minor angle variance can be induced using the `RandomPerspective` transformation from the `torchvision` package [79]. This augmentation can be applied online to synthetic videos during training. Although this augmentation does not achieve the realism of action recognition benchmarks, it is still expected to amplify the model's spatial perception and further bridge the domain gap.

This transformation serves as an efficient alternative for three dimensional IFS fractals, that were introduced and described in detail in Section 5.5. The decision not to employ 3D fractals in the upcoming experiments was made due to their computational demand as well as undesired artifacts such as aliasing. 3D fractals are left for future work.

6.3.8 Displacement

Human activity and background is not the only dynamic element present in real videos. Aside from human action, a multitude of videos contain additional displacement motion. Observing such samples, three distinct types of displacement can be distinguished:

- **Foreground displacement**: This occurs when individuals perform an action while simultaneously walking or running and the camera remains static. As such, in the captured video, the position of the foreground is shifted while the background remains unaffected.
- **Background displacement**: As previously, this motion is also the result of the displacement of the human target. However, now the camera follows the foreground. As a result, the position of the foreground remains virtually static, but the background obtains the same displacement motion but with opposite direction. A notable example is a camera following athletes in a sprint race.
- **Camera displacement**: In this case, the absolute position of the human target remains unchanged but the focus of the camera is being shifted. As a result, both the foreground and the background are relatively displaced in the opposite direction of the camera.

Invariance to such movements can be boosted with simple transformations. For background displacement, a static background frame is initially enlarged and then a sequence of crops with dimensions of the original video is created. As the centers of the crops are consecutive points on a two-dimensional line, the result is displacement towards a fixed direction. For camera displacement, the process is the same with the exception that each crop is taken at a different frame. For foreground displacement, the difference is that the video is initially reduced in size and then each frame is placed at a different location of blank canvas. Such operations are parameterized by temporal duration, temporal offset, speed and direction angle.

As in the case of the background domain adaptation, the above three transformations are merged into a single augmentation and not evaluated separately. This augmentation is applied online during pre-training and not offline during rendering.

6.3.9 Camera Zoom

Another extraneous motion involving the camera is zoom. To emulate this motion, an approach resembling displacement is taken. At first, the video is interpolated to larger spatial dimensions. Next, central cropping is applied to each frame with variable scale. Lastly, each cropped frame is interpolated to the original spatial dimensions of the video. Increasing the scale results in zooming out, whereas decreasing it leads to zooming in. Similarly to displacement, this operation is parameterized by temporal duration, temporal offset, and speed. As before, this operation is implemented as an online augmentation.

6.3.10 Camera Shake

A large portion of the observed videos were captured by hand-held cameras and therefore contain unwanted shaking. To synthesize such motion, the method proposed by [83] is adapted. Specifically, the displacement in each of the two spatial dimensions is modelled as:

$$p_t = \sum_{i=1}^{n} w_i \sin(2\pi f_i t + \phi_i) + \eta_t, t = 1, 2, \ldots, T$$

Here, T is the duration of the video in frames, w_i is the weight of each sine function and set to $1/i$, f_i is the different frequency components, which is a random value within a range, ϕ_i denotes the phase and is decided by a same random value for all i, and η is the noise component. In later experiments these parameters are sampled as follows:

$$
\begin{aligned}
n &\sim U(\{2, \dots 5\}) \\
f_i &\sim U(0.1, 1.2) \\
\phi_i &\sim U(0, 2\pi) \\
\eta_t &\sim U(-0.3, 0.3)
\end{aligned}
$$

To apply the shaking effect to an existing video, two displacement sequences are first sampled, one for each spatial dimension. Afterwards, the video is enlarged and each frame is cropped. The position of the crop is determined by the vertical and horizontal displacement values. Camera shaking can be considered as a special case of camera displacement, a phenomenon that was explained in detail in the previous subsection. Camera shaking is implemented as an online augmentation during pre-training.

6.4 Automatic Construction of Categories

In later chapters, pre-training of 3D CNNs will be conducted both with self-supervised and supervised objectives. For self-supervised learning, labels are unnecessary and it is sufficient to construct a dataset by randomly sampling parameters for each video. However, for supervised learning, synthetic videos must be strictly divided into categories. To automatically construct categories, we adapt the approach originally proposed in [49].

Specifically, given the number of desired categories C, the first step is to sample parameters for C different fractal videos following the procedure described in Section 6.2. As such, each category c is represented by a parameter matrix $w_c \in \mathbb{R}^{T_c \times N_c \times 6}$ as well as the index of the variation function var_c. For each category, the interpolation curves and variation are fixed. The former can be either linear or nonlinear and composite following Subsection 6.3.1. The latter is selected from the values presented in Subsection 6.3.2. The importance of these parameters will be thoroughly evaluated in upcoming experiments.

To produce a new sample belonging to category c, we mutate its respective parameter matrix w_c. First, we sample a new matrix $m_{ac} \in \mathbb{R}^{T_c \times 1 \times 6}$, which consists of 6 random curves produced with the procedure proposed in Subsection 6.3.1. The magnitude of these curves is small and is bound between $\epsilon_1^{ac} = -0.35$ and $\epsilon_2^{ac} = 0.35$. We consider this mutation analogous to the AC component of electrical signals, as it changes its magnitude continuously with time. Afterwards, we sample an additional noise matrix $m_{dc} \in \mathbb{R}^{1 \times N_c \times 6}$ from $U(\epsilon_1^{dc}, \epsilon_2^{dc})$, where $\epsilon_1^{dc} = -0.2$ and $\epsilon_2^{dc} = 0.2$. This mutation is analogous to the DC component of electrical signals, as it constant with time with time. The parameter matrix of the new sample is calculated as follows:

$$\tilde{w_c} = m_{ac} \odot w_c + m_{dc}$$

Here, $\odot$ denotes elementwise product. By mutating the parameter matrix, we achieve variance within samples belonging to the same class. This is necessary for increasing the difficulty of the pre-training task and thus achieving stronger visual representations. A result of the proposed mutation scheme can be seen in Figure 6.5, which displays four samples belonging to the same category.

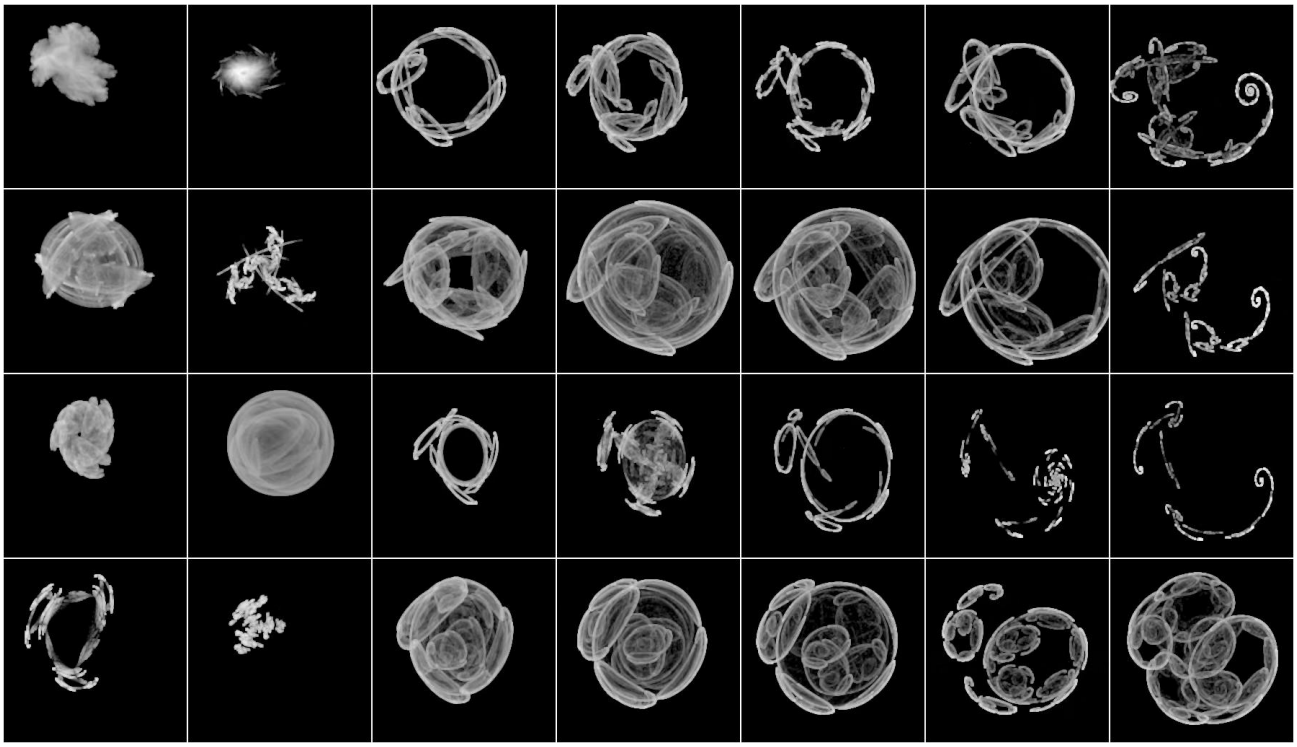


Figure 6.5: Example of the proposed mutation mechanism. Each row displays frames from a different video. Although all videos belong to the same class, differences between them are obvious.

6.5 Alternative Synthetic Data

The previous sections focus on the synthesis of fractal animations. However, it is not clear that this type of video possesses the necessary properties required for training strong visual representations. As such, this section presents alternative methods to automatically generate video clips. These videos will be compared against fractals in experiments in later chapters. As each type of video possesses different characteristics, the objective is to determine which attributes are favorable for downstream results and which are not.

Unfortunately, a subset of the proposed domain adaptation methods cannot be applied to all datasets: Diversity (6.3.2) can only be applied to fractals and Motion (6.3.1) can be applied everywhere except Perlin Noise which does not require an interpolation curve. The

rest of the proposed methods are applicable everywhere. Additionally, each generative process requires a different approach for the construction of categories.

6.5.1 Perlin Noise

As far as this document is concerned, the most relevant work is that of [46]. The authors of that work design a synthetic dataset of generated motion patterns based on perlin noise [80, 81], which is a type of random texture. They propose to initially pre-train 3D CNNs with this dataset, then train on Kinetics [11, 50] and finally train on small-scale downstream datasets UCF101 [89], ActivityNet [39] and HMDB51 [56]. Their method leads to improved results, compared to the standard approach of pre-training on Kinetics followed by downstream fine-tuning. However, due to computational constraints, in the present work, synthetic pre-training will be followed directly by downstream fine-tuning and the Kinetics datasets will not be employed.

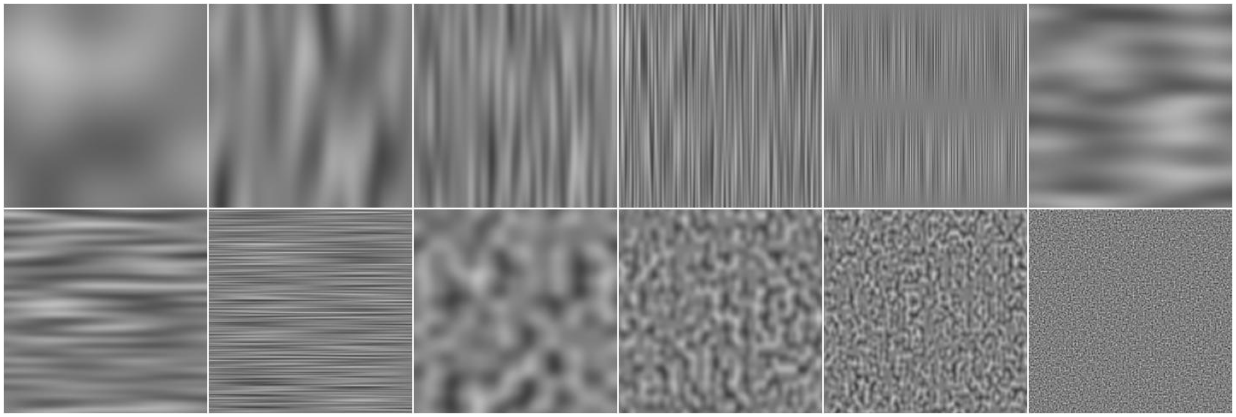


Figure 6.6: Examples of perlin noise images with variable spatial frequencies. A higher rate of change indicates higher frequency in the corresponding direction. The lowest and highest frequencies are displayed in the the top left and the bottom right images respectively.

Each video sample is defined by three frequencies, one for each spatial dimension as well as one for the temporal. These parameters determine the rate of change in their respective dimensions. They additionally define the label of the video. Examples of perlin noise images with various parameters can be seen in Figure 6.6.

Unlike fractals, these videos lack distinct shape and contours. Additionally, the proposed motion interpolation schemes cannot be applied here. Therefore, by comparing perlin noise against fractals in later experiments, it can be determined if solid shape and non-linear motions are necessary properties for synthetic data. In truth, [46] show that pre-training with perlin noise without Kinetcs does not yield significantly better downstream results that training from scratch. As such, perlin noise is expected to underperform compared to fractals.

6.5.2 Octopus

A different type of animation can be produced with random curves. Specifically, as described in the IFS parameter interpolation section (6.3.2), it is possible to produce a waveform by sampling random values and quadratically interpolating the sequence. By repeating this process twice, (one for each spatial dimension), one can construct a random two dimensional curve. To create an animation, two such curves can be sampled and their coordinates interpolated following the same approach used for IFS. To boost complexity, the aforementioned process can be repeated N times. The N resulting curves are conjoined at a fixed point.

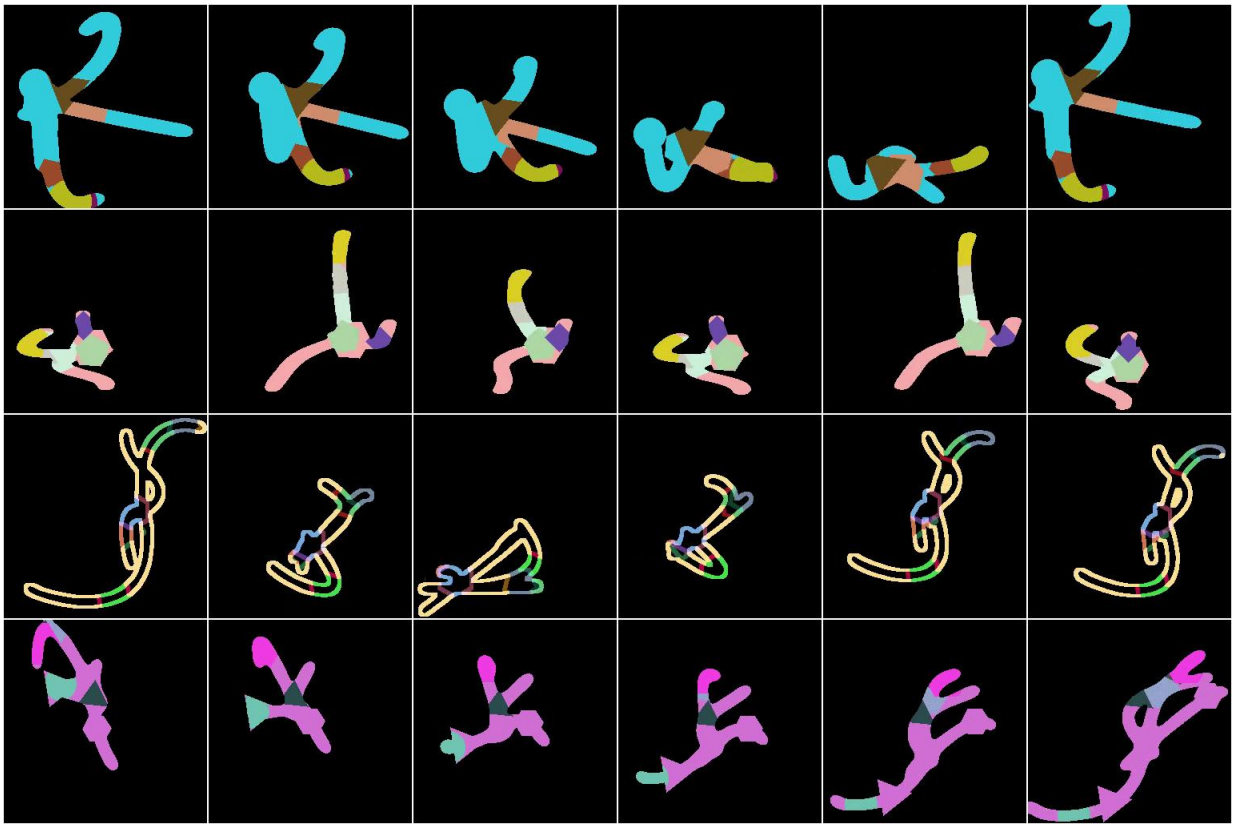


Figure 6.7: Examples of the octopus model. Each row displays a different video. Note the colorization, the removed interior as well as the geometrical shapes embedded within. Such imaged are similar to fractals as both possess distinct contours.

So far, these videos contain thin curves and lack the density and contours present in real videos. As a solution, gaussian blur is applied followed by the morphological operation of closing [102]. The outcome is a shape reminiscent of an octopus with the curves serving the role of tentacles. As such, in the rest of this document, these synthetic videos will be referred to as "octopus".

Each frame of these animations is a binary image. For embellishment, a few additional operations are employed (see Figure 6.7):

- Colorization
- Interior removal via the the morphological operation of gradient [102].
- Addition of random geometrical shapes

Compared to fractals, octopus videos are very similar, since both maintain distinct shape and contours. Furthermore, each limb can execute a different movement and therefore the octopus model is compatible with the composite motion scheme. In addition, all of the proposed foreground, background and camera augmentations are applicable here as well.

The only major difference between fractals and octopus videos is the variety of the generated shapes, where the former significantly surpass the latter. Hence, later experiments will give valuable insight as to the importance of the diversity of synthetic videos. For the domain of images, it has already been shown that diversity is a key property for learning stronger representation [4]. Hence, fractals are expected to lead to superior downstream performance.

6.5.3 Dead Leaves

Dead leaves [84, 59] is a simple image model designed to emulate statistics of natural images, such as having a $1/|f|^a$ power spectrum. Like fractals, it has recently been employed in the domain of deep learning to evaluate representations trained on synthetic data [4, 66]. Such images can be constructed by filling an image canvas with geometric shapes (circles or polygons) which are positioned uniformly at random. [4] demonstrated that better results are obtained when the said shapes differ (polygons with different number of edges). This advice is taken into account within the present work.

Dead leaves images can be easily extended to the domain of video. To do so, for each geometric shape, a two dimensional curve is sampled in the same manner as the limbs in in the octopus model. Then, throughout the video, the said shape traverses this curve. As a result, some shapes are often overlapped by others. The produced animations are reminiscent of Brownian motion, which is the random motion of particles suspended in a medium. Examples of dead leaves videos can be seen in Figure 6.8.

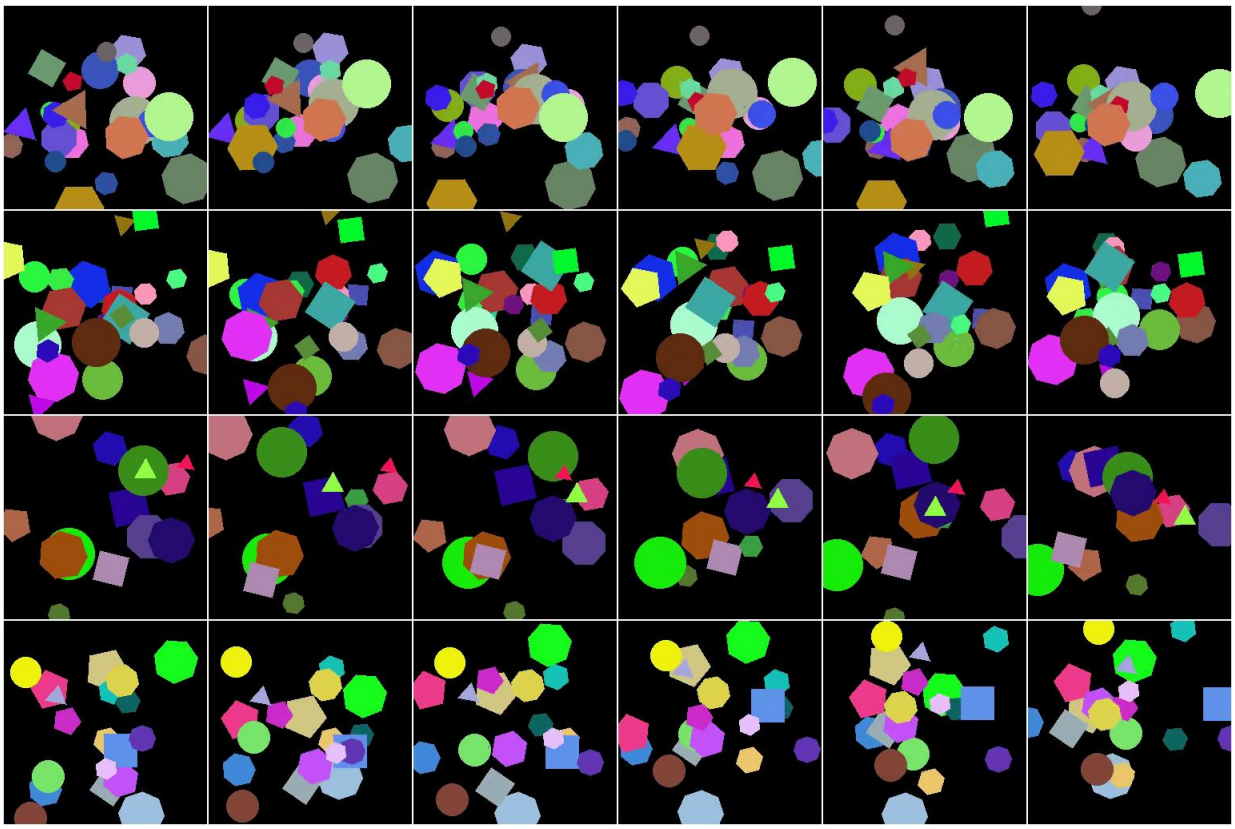


Figure 6.8: Examples of the dead leaves model. Each row displays a different video. Each shape moves independently by traversing a randomly sampled 2D curve. Some shapes are overlapped by others.

Chapter 7

Experiments

The objective of this chapter is to utilize all previously proposed concepts and conduct thorough experiments. Section 7.1 describes the details of the upcoming experiments including downstream tasks, model architectures and selected hyperparameters. Afterwards, Section 7.2 presents numerical results of the conducted experiments as well as commentary on the said results. Section 7.3 analyzes misclassified samples and detects shortcomings of the proposed framework.

7.1 Proposed Framework

7.1.1 Downstream Tasks

The proposed pre-training framework is evaluated by fine-tuning the model on 6 downstream small-scale datasets of short video clips. All datasets are designed for the task of classification. The scale of the datasets is limited due to computational constraints, whereas the length is limited to match the statistics of pre-training datasets. Downstream datasets exhibit significant differences amongst each other and were chosen to maximize overall diversity. Detailed statistics can be seen in Table 7.1. Frames randomly sampled from downstream datasets are displayed in Figure 7.1.

- **HMDB51** [56]: An established action recognition benchmark. Clips usually depict a single person performing a specific action. Additionally, clips often contain unwanted artifacts such as camera motion (shift, shake) as well as scene changes.
- **UCF101** [89]: Similar to HMDB51, but of larger scale and with higher expected accuracy as a result.
- **DIVING48** [61]: A collection of diving competition videos. Considered as a fine-grained dataset, because all videos share a similar background and object features. A subset of clips contains synchronized diving with multiple individuals.

- **EGTEA GAZE+** [62]: Consists of first person videos of cooking activities in different kitchen environments. Videos often contain small objects such as cooking tools and ingredients.
- **VOLLEYBALL** [42]: Established benchmark in the field of group action recognition. Each clip depicts two opposing volleyball teams where one remains inactive while the other performs a specific group activity (set, spike, pass, winpoint). This results in 8 classes in total. The length of each video is exactly 41 frames. The provided annotation has been shown to contain errors [119].
- **YUP++** [27]: Contains videos of dynamic scenes. Examples are forest fires, collapsing buildings and rushing rivers. Half of the videos within each class are acquired with a static camera and half are acquired with a moving camera. Duration for each video is 5 seconds.

Dataset	# Training Videos	# Validation Videos	# Classes
HMDB51	3570	1530	51
UCF101	9537	3783	101
DIVING48	15943	2096	48
EGTEA GAZE+	8299	2022	106
VOLLEYBALL	2152	1341	8
YUP++	120	1080	20

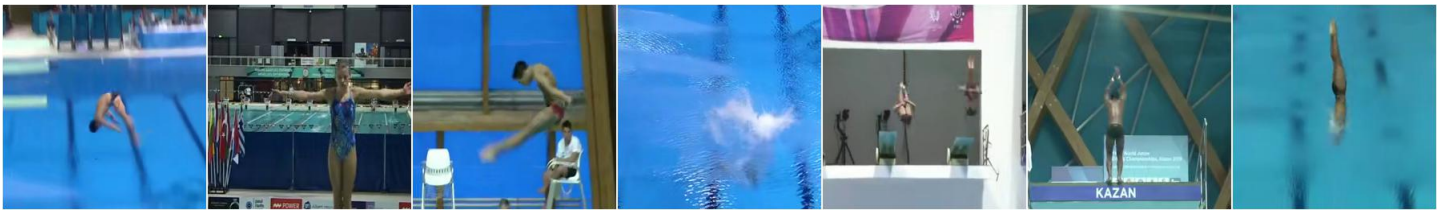
Table 7.1: Statistics of downstream datasets. All benchmarks are of small scale & short length.

For each downstream dataset, an official train-validation split is provided. To evaluate the pre-trained models, we fine-tune them on the training set and then report the top-1 accuracy on the validation set.

Additionally, it is noteworthy that some of the aforementioned datasets have been subjects of thorough research. As a result, specialized neural architectures have been developed for the said datasets (e.g. [119]). Such architectures are focused on innate properties of the datasets and therefore can achieve accuracies significantly higher than those reported in the present work. However, the objective of this document is to develop a general pre-training framework without specialization in any specific benchmarks. As such, we instead will utilize a more regular model which is described in the next section.

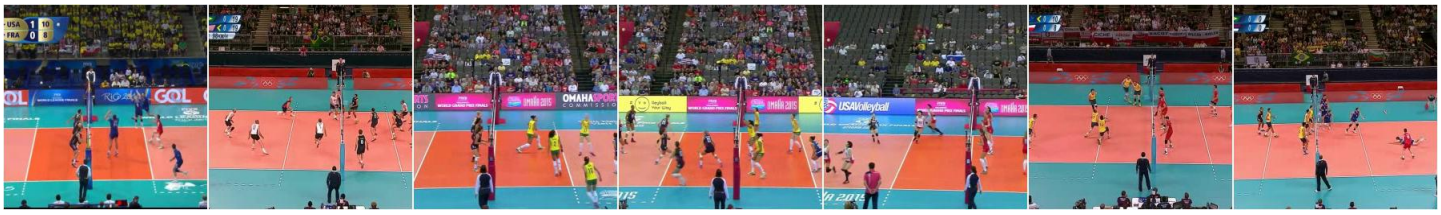
(a) HMDB51

(b) UCF101



(c) DIVING48

(d) EGTEA GAZE+

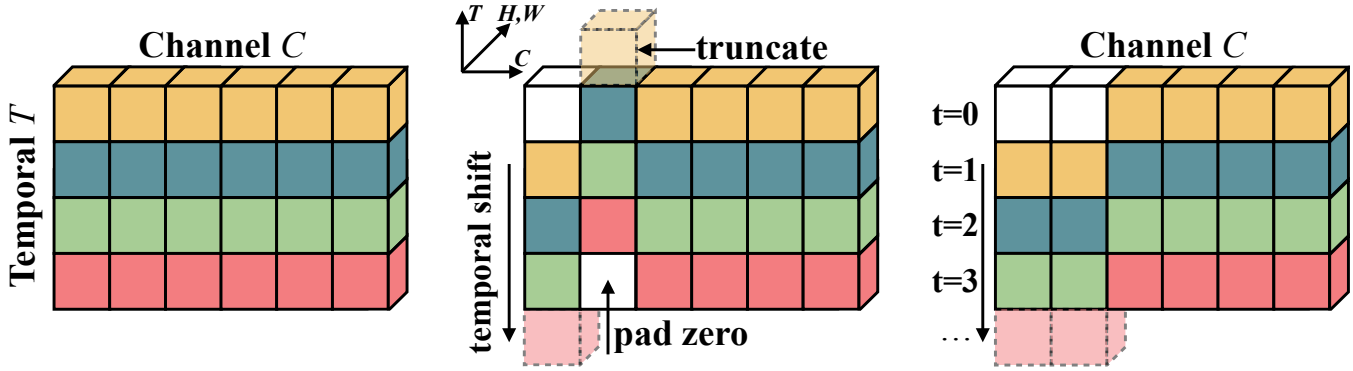


(e) VOLLEYBALL

(f) YUP++

Figure 7.1: Frames randomly sampled from downstream datasets. With the exception of the first two, all datasets are evidently different.

7.1.2 Model Architecture



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

Figure 7.2: Illustrations adapted from [63]. **Temporal Shift Module (TSM)** efficiently executes temporal modeling by moving the feature map along the temporal axis. Despite being computationally free on top of a 2D convolution, it possesses strong temporal modeling ability. TSM is capable of both **offline** and **online** video recognition. Bi-directional TSM mixes both past and future frames with the current frame, which is suitable for high-throughput offline video recognition. Uni-directional TSM mixes only the past frame with the current frame, which is appropriate for low-latency online video recognition.

Temporal Shift Module (TSM) [63] is employed for all experiments in the present chapter. ResNet-50 [38] is utilized as backbone. TSM is an efficient 2-Dimensional Convolution Neural Network architecture designed for action recognition tasks. It achieves information exchange among neighboring frames by shifting parts of the channels along the temporal dimension. More technical details are presented in Figure 7.2.

TSM was chosen as our baseline model due to its computational efficiency: solid results can be delivered with only 8 input frames. Such short length significantly alleviates both the CPU and GPU bottlenecks. The former is evidenced by reduced dataloading whereas the latter by a reduced amount of computation within the neural network. Unless specified otherwise, pre-training does not utilize any off the shelf checkpoints and is done from scratch.

7.1.3 Implementation Details

We now list hyperparameters as well as other technical details chosen for the experiments.

Datasets

All synthetic videos aside from perlin noise are rendered with spatial resolution of 256 × 256 pixels and temporal length which is sampled from U($\{18, \ldots, 20\}$). Perlin noise is

rendered with resolution of 240×240 and length of 30 frames. For fractals, 50% of videos employ nonlinearities. To alleviate the data loading bottleneck, for standard experiments video files from datasets DIVING48, VOLLEYBALL and EGTEA are resized to 256 pixels (short side). For higher resolution experiments, VOLLEYBALL is resized to 512 pixels (short side) while all other datasets are unaltered.

Training

Only RGB frames are employed throughout this work. Unless specified otherwise, the model input is a clip of 8 strided frames with spatial resolution of 112×112 pixels. The stride is 2 for pre-training, 4 for fine-tuning VOLLEYBALL and 6 for fine-tuning all other datasets. During training, one such clip is randomly sampled from a video. During validation, 10 such clips are uniformly selected from a single video and separately fed into the model with the final output being the average of softmax scores. The number of total training epochs is 25 and 100 for pre-training and fine-tuning respectively. The number of warmup epochs for the scheduler is 3 and 10 respectively.

The networks are optimized using Adamw [65] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and cosine scheduling [64] is employed. The true learning rate is scaled according to the batch size: $lr_{true} = \frac{bs}{bs_{base}} lr$, where the batch size and base batch size are set to 32 and 32 respectively. The default learning rate is initialized at $1 \cdot 10^{-6}$, increases to $8 \cdot 10^{-4}$ during warmup and eventually falls to $1 \cdot 10^{-5}$ at the end of training. The default weight decay is set to $1 \cdot 10^{-2}$. Exceptions are the VOLLEYBALL dataset where the learning rates are $2.5 \cdot 10^{-6}$, $2.5 \cdot 10^{-3}$ and $2.5 \cdot 10^{-5}$ and weight decay is set to $1 \cdot 10^{-1}$ as well as DIVING48 where the learning rates are $1.5 \cdot 10^{-6}$, $1.5 \cdot 10^{-3}$ and $1.5 \cdot 10^{-5}$.

Augmentation

For fine-tuning, the augmentation scheme consists of random cropping with bicubic interpolation, horizontal flip, Randaugment [19] and Gaussian blur. For cropping, the scale is sampled from $U(0.2, 1.0)$, and the ratio from $U(0.75, 1.33)$. The augmentations are applied in the same order as they are listed. As an exception, for VOLLEYBALL we use area interpolation and omit horizontal flip and Gaussian blur.

For pre-training, domain adaptation techniques (see Section 6.3) are additionally applied between horizontal flip and Randaugment. Additionally, We employ a curriculum and linearly increase the intensity of domain augmentations for the first 5 epochs. All domain augmentations are applied with a probability of 0.3 with the exception of Background Randomization, Scale, Perspective and Group whose probabilities are 1.0, 1.0, 0.8 and 0.15 respectively. To accelerate training, each augmentation is applied in parallel and identically to all of its selected samples inside a batch. Background Randomization, Scale and Group are exceptions where a different transformation is applied to each sample inside a batch.

Libraries

Synthetic data is produced with with NumPy[1] and Numba[2] and transformed into video files with ffmpeg[3]. Decord[4] is used for reading video files during training. PyTorch[5] is employed for models, training as well as custom augmentations. PyTorchVideo[6] is used for the Randaugment augmentation [19].

The setup sections of upcoming experiments contain additional hyperparameters as well as technical details or overwrite existing ones.

7.2 Experimental Results

Each experiment is comprised of the following segments:

- **Objective** segment, which describes the motivation behind the experiment and states its goals.
- **Setup** segment, which lists the implementation details as well as any other relevant information that is necessary for understanding of the experiment.
- Result **table**, which contains the validation accuracy on downstream datasets for every step of the conducted experiment.
- **Comments** segment, which provides conclusions that can be derived from numerical results as well as in-depth explanations regarding observed behavior.

7.2.1 Experiment 1 - Domain Adaptation

Objective

Previous work on synthetic images [4] suggests that when it comes to pre-training, emulation of various structural properties of real data leads to stronger visual representations and better downstream results. As such, the objective of this section is to verify if this statement holds for the modality of video. Specifically, we utilize domain adaptation techniques that were proposed in Section 6.3. We observe the effects of each technique on downstream results and determine the beneficial ones.

[1] https://numpy.org/
[2] https://numba.pydata.org/
[3] https://ffmpeg.org/
[4] https://github.com/dmlc/decord
[5] https://pytorch.org/
[6] https://pytorchvideo.org/

Setup

- All pre-training is done with the MoCoV2 [14] self-supervised framework, which is described in detail in Section 3.5. MoCoV2 was chosen as it offers a combination of solid downstream results as well as computational efficiency.
- Experiments are carried out in a sequential fashion. At each step, a specific modification is applied to the pre-training process. For Motion & Diversity modifications, a new synthetic dataset is constructed. The rest of modifications are implemented as additional online augmentations and therefore reuse the previous dataset.
- The pre-training dataset consists of 100K unlabeled fractal videos (see Section 6.2) and was constructed by randomly sampling parameters.
- Downstream datasets are different. Therefore, it is possible that a specific modification will result in accuracy increase for a subset of datasets and a decrease for the rest. As such, the modification will be retained in all remaining experiments if it leads to improvement for action recognition benchmarks HMDB51 & UCF101. These have been chosen as the focus of the present work. Otherwise, the modification will be discarded and not incorporated in later experimental steps.
- The initial pre-training framework consists of fractal videos with linear motion and no domain adaptation methods are involved.
- Validation accuracy after fine-tuning can be seen in Table 7.2.

Method	Kept	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Scratch	-	31.5	70.3	4.7	50.5	61.1	43.7
Initial	-	41.4	72.7	24.3	49.8	80.6	52.3
Background	Yes	47.4	74.2	23.0	47.1	77.6	56.4
Motion	Yes	47.3	75.7	21.7	48.6	79.2	59.5
Diversity	Yes	50.4	77.8	24.9	49.8	80.8	63.1
Scale + Shake	Yes	52.8	78.0	26.0	50.7	80.8	61.9
Shift	Yes	**54.5**	79.6	29.5	50.8	80.9	65.1
Zoom	Yes	54.3	**80.2**	**30.8**	**52.0**	81.4	65.2
Perspective	No	53.3	78.7	25.0	50.0	80.7	62.6
Group	No	53.3	79.7	30.3	51.9	**81.9**	**66.0**

Table 7.2: Experiment 1: Downstream validation accuracy after domain adaptation. **Bold** font indicates best results on the specific benchmark. Column Kept specifies if the respective modification will be retained for all remaining experiments.

Comments

- The obvious observation is that emulation of a property has positive effect on datasets that include the said property and either negative or no effect on datasets that do not include it. For instance:
  - Background randomization increases accuracy on action recognition benchmarks HMDB51 and UCF101, but decreases it on DIVING48 and VOLLEYBALL. This is not surprising. For the former, it can be assumed that background is different for each video and generally irrelevant regarding ground truth. For the latter, it is reasonable to believe that background is approximately shared by all samples and that the exact relation between background and foreground is important for determining the correct category.
  - The group modification is beneficial for the VOLLEYBALL dataset where the model must learn to observe multiple individuals at once. Simultaneously, it is adverse for HMDB51 where the majority of samples depict a single person.
- Some results are inconsistent. For example, DIVING48 contains synchronized diving videos with multiple individuals, but the Group modification deteriorates results. However, the accuracy drop is not major.
- The only modification that results in non-trivial improvement across all benchmarks is amplified diversity through the inclusion of non-linear fractals. This is in line with previous work on self-supervised pre-training with synthetic images [4], which concludes that diversity is a key property to learn good representations.
- The only modification that results in non-trivial deterioration across all benchmarks is the random perspective transformation. We do not know why this occurs.
- Pre-training is very ineffective for the EGTEA dataset. Our best approach (Zoom) only improves accuracy by 2% compared to training from scratch.

7.2.2 Experiment 2 - Alternative Synthetic Data

Objective

In the previous experiment, pre-training was conducted exclusively with fractal videos, which were introduced in Sections 6.2 and 6.3. This experiment investigates alternative synthetic data, that can be seen in more detail in Section 6.5. Compared to fractals, each alternative dataset exhibits key differences. For instance, the Octopus dataset is less diverse and Perlin Noise lacks discrete contours. As such, both the current and previous experiments have a shared objective: to determine characteristics of synthetic data that are beneficial for downstream results.

Setup

- For all datasets, we employ the pre-training conditions and domain adaptation methods with the best results from the previous experiment: Zoom modification. This entails that the specific row will be repeated in the next result table.
- A subset of domain adaptation methods cannot be applied to all datasets: Diversity can only be applied to fractals and Motion can be applied everywhere except Perlin Noise which does not require an interpolation curve.
- Validation accuracy after fine-tuning can be seen in Table 7.3.

Dataset	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Fractal	**54.3**	**80.2**	**30.8**	**52.0**	**81.4**	**65.2**
Octopus	50.9	76.5	25.5	49.1	80.7	58.3
Dead Leaves	39.9	70.2	15.8	47.3	75.7	50.2
Perlin Noise	42.4	72.5	21.4	50.1	76.4	58.2

Table 7.3: Experiment 3: Downstream validation accuracy for different pre-training datasets. **Bold** font indicates best results on the specific benchmark. The proposed fractal dataset outperforms all alternatives.

Comments

- Pre-training with fractal videos leads to significantly better results than with alternatives. Therefore, it is reasonable to assume that fractals possess properties that are more favorable for downstream tasks.
- The Octopus dataset is the second-best approach. Such videos have similar appearance to fractals as both are random shapes with distinct contours. However, fractals exhibit greater diversity in the spatial domain with significantly wider range of produced shapes. Furthermore, small changes in the interpolation curve can result in unexpected distortion in the rendered fractals. Consequently, fractals are more diverse in the temporal dimension as well. As such, we deduce that diversity in synthetic datasets leads to better downstream results. This conclusion is in complete agreement with both previous work [4] as well as the previous experiment.
- Domain gap is an additional explanation for the difference in performance between synthetic datasets. Large scale studies have shown that downstream performance of self-supervised learning frameworks deteriorates as the domain shift between the pre-training and downstream datasets becomes larger [17, 53, 94]. It is true that all of the proposed synthetic datasets are significantly different compared to real videos. However, it is reasonable to assume that the domain gap is especially large

for datasets without distinct contours: Dead Leaves and Perlin Noise. The former is a set of overlapped geometric shapes and the latter consists of dynamic nebulous textures. The enlarged domain gap due to a lack of contours justifies why these datasets result in worse downstream performance compared to Fractals and Octopus.

7.2.3 Experiment 3 - Training Objective

Objective

Earlier experiments have exclusively employed the MoCoV2 [14] self-supervised framework for pre-training. It was chosen over others due to its computational efficiency. Thus, this section explores alternative training objectives aiming to maximize downstream performance.

Setup

- Detailed description of each pre-training framework can be found in Section 3.5. The explored self-supervised frameworks require different amounts of computation per training step:
  - **MoCoV2** [14]: 1 forward step with gradients, 1 forward step without gradients.
  - **SimCLR** [13]: 2 forward steps with gradients.
  - **BYOL** [35]: 2 forward steps with gradients, 2 forward steps without gradients.
- All three self-supervised frameworks utilize the unlabeled synthetic dataset from previous experiments.
- For the supervised objective, a new dataset is constructed that consists of 500 classes and 200 samples per class (100K videos in total). Each sample was rendered by mutating fixed class parameters as described. More details can be found in Section 6.4.
- Validation accuracy after fine-tuning can be seen in Table 7.4.

Comments

- The supervised objective confidently leads to best downstream results. This is surprising as the classes are sampled randomly and possess no meaningful semantic information.

Objective	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
MoCoV2	54.3	80.2	30.8	52.0	81.4	65.2
SimCLR	57.5	81.3	34.3	54.7	**83.2**	70.2
BYOL	52.0	78.1	26.1	52.0	80.4	66.0
Supervised	**61.5**	**84.9**	**38.5**	**54.8**	82.7	**73.6**

Table 7.4: Experiment 3 - Downstream validation accuracy for different pre-training objectives. **Bold** font indicates best results on the specific benchmark. The supervised objective is superior. The results for self-supervised frameworks are in complete contrast with ImageNet pre-training where the order is reversed.

- A possible explanation is again related to the domain gap between the pre-training and downstream datasets. As mentioned previously, previous work has determined that self-supervised frameworks are especially vulnerable to this domain shift [17, 53, 94]. On the other hand, it is reasonable to assume that supervised pre-training is more resilient to the domain gap, consequently leading to better downstream results.

- Computational resources are an additional factor. For instance, given an input batch, SimCLR [13] applies augmentations twice and feeds each resulting batch into the model leading to two separate output representations. The objective is to maximize the similarity between the representations of matching samples for the two batches and minimize it for the rest. As such, increasing the batch size also increases the task difficulty and consequently improves downstream results. SimCLR functions well when the batch size is in the order of thousands [13]. On the other hand, the present experiments were conducted with a batch size of 16, which does not lead to optimal results. As a comparison, the difficulty in supervised learning does not depend on the batch size, as the number of classification categories remains fixed.

- Furthermore, self-supervised frameworks benefit from increasing the number of training epochs, requiring up to 800 for ImageNet pre-training [15]. On the contrary, present experiments are conducted with only 25 epochs. Hence, it can be assumed the self-supervised models are probably undertrained and more training time is required.

- As such, it is reasonable to assume that downstream performance of self-supervised pre-training can be improved further, but that would require both additional computational resources and time. Therefore, we deduce that supervised pre-training is a more cost-effective solution when it comes to pre-training with synthetic datasets.

- SimCLR outperforms MoCoV2, which in turn outperforms BYOL. This is in complete contrast with ImageNet pre-training, where the order is reversed [35]. We cannot explain this phenomenon.

7.2.4 Experiment 4 - Importance of Motion

Objective

The previous experiment demonstrated the superiority of supervised pre-training compared to alternative approaches. As proposed in Section 6.4, each automatically generated category can be divided into two components: shape and motion. The objective of the current experiment is to perform ablation studies and measure the importance of each component for downstream results.

Setup

- Each randomly generated class is defined by the fractal parameters of the first and last frames as well as the interpolation curves. The former determines the shape of rendered the rendered animation, whereas the latter is responsible for its motion.
- To measure the importance of motion, a new labeled dataset is constructed. To produce a sample belonging to a given class, the parameters of the first and last frames are kept unaltered. On the contrary, the interpolation curves are sampled randomly. In other words, samples from a specific category will exhibit similar shape but their motion will be arbitrary.
- Validation accuracy after fine-tuning can be seen in Table 7.5.

Motion	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Fixed	**61.5**	84.9	**38.5**	**54.8**	**82.7**	**73.6**
Random	59.1	**85.2**	37.4	54.1	82.6	72.6

Table 7.5: Experiment 4 - Downstream validation accuracy for the motion ablation. **Bold** font indicates best results on the specific benchmark. It is evident that fixed motion leads to better results. However, the drop in accuracy is minor and therefore it can be assumed that motion is not significant.

Comments

- On average, randomizing motion results in a minor accuracy drop across all evaluated benchmarks.
- It is reasonable to assume that the overall contribution of the motion component is not significant. Motion is only a small fragment of the proposed framework. Other components such as shape and various domain adaptation techniques are of more consequence for the model's downstream performance.

7.2.5 Experiment 5 - Scale

Objective

In earlier experiments, statistics of the labeled dataset (# classes & # instances per class) were chosen arbitrarily. As such, this section explores the relationship between the aforementioned statistics and the transferability of the model to downstream tasks. Additionally, we would like to determine if downstream results can be benefited by increasing the dataset size.

Setup

- The previously used dataset consists of 500 classes and 200 instances per class, meaning 100K samples in total.
- The experiment has two stages.
- In the first stage, the number of classes is fixed to 500 as previously, while the number of instances per class is varied: 100, 200, 400.
- In the second stage, the number of instances is fixed to the number with the best results from the first stage. On the other hand, the number of classes is varied: 250, 500, 1000.
- Each stage contains three pre-training datasets of various sizes. To ensure fairness, amongst them, each dataset is a superset for all smaller ones.
- For larger datasets, we additionally evaluate the Perspective transform (see Section 6.3), which led to disappointing results in Experiment 1.
- For the first stage, Validation accuracy after fine-tuning can be seen in Table 7.6, while for the second one in Table 7.7.

#Instance/#Total	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
100/50K	56.5	81.8	35.8	52.6	82.6	69.3
200/100K	61.5	84.9	38.5	54.8	82.7	73.6
400/200K	61.5	86.0	**40.8**	53.8	83.0	75.1
400/200K + Perspective	**63.2**	**86.1**	38.4	**55.2**	**84.0**	**75.7**

Table 7.6: Experiment 5 - Stage 1 - Downstream validation accuracy for different numbers of instances per class. **Bold** font indicates best results on the specific benchmark.

#Classes/#Total	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
250/100K	60.3	85.3	38.1	53.5	84.1	75.6
500/200K	61.5	86.0	40.8	53.8	83.0	75.1
1000/400K	62.4	87.3	**41.2**	**56.1**	**84.0**	**76.0**
1000/400K + Perspective	**65.4**	**87.6**	40.3	56.0	83.1	72.7

Table 7.7: Experiment 5 - Stage 2 - Downstream validation accuracy for different numbers of classes. **Bold** font indicates best results on the specific benchmark.

Comments

- On average, an increase in the number of instances per class improves the model's transferability across downstream benchmarks.
- Likewise, a similar improvement in downstream results is observed after increasing the number of classes.
- As such, it is reasonable to assume that an additional boost in downstream accuracy can be achieved by further increasing the number of synthetic videos. However, due to computational constraints, the pre-training dataset will not be expanded any further and later experiments will not exceed 400K samples.
- The Perspective augmentation strengthens results on action recognition datasets HMDB51 and UCF101. This is in contrast with Experiment 1 where the said augmentation was detrimental. The main difference between Experiment 1 and the current one is the training objective. The former employs self-supervised learning whereas the latter supervised learning. As such, it can be assumed that the exact effect of the proposed augmentations varies for each training objective. Additional experiments are necessary to determine the effects of domain adaptation for supervised objectives. However, these are left for future work.

7.2.6 Experiment 6 - Higher Resolution

Objective

All previous experiments were conducted with a low spatial resolution of 112×112 pixels. In this experiment we increase this parameter to 224×224 and observe the model's performance. Additionally, this is expected to be our strongest model. As such, we compare its results to Kinetics [11, 50], which utilizes real data and is the standard pre-training approach for action recognition.

Setup

- Due to computational constraints, we do not conduct Kinetics pre-training. Instead, we employ a checkpoint generously shared by the authors of [63]. The checkpoint was pre-trained with a resolution of 224×224 and is therefore compatible with the current experiment.
- Pre-training protocols for Kinetics and fractals are different. Kinetics and fractal datasets consist of approximately $250K$ and $400K$ training samples respectively, meaning the latter is larger. However, the authors of [63] employ 100 epochs for Kinetics pre-training, whereas our fractals require only 25 epochs for convergence. Additionally, Kinetics consists of significantly heavier videos, both in terms of resolution and length. potentially causing a data loading bottleneck. As such, pre-training with fractals is sufficiently faster than with Kinetics.
- Again due to computational constraints, all training is performed with a reduced batch size of 11.
- For fine-tuning the Kinetics checkpoint, we employ hyperparameters that are depicted in Table 7.8.
- For fractal pre-training we employ the best setup from the previous experiment: 1000/400K + Perspective.
- Validation accuracy after fine-tuning can be seen in Table 7.9.

Hyperparameter	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Weight Decay	$1 \cdot 10^{-1}$	$1 \cdot 10^{-1}$	$4 \cdot 10^{-1}$	$4 \cdot 10^{-1}$	$4 \cdot 10^{-1}$	$4 \cdot 10^{-1}$
LR-Init	$1 \cdot 10^{-8}$	$1 \cdot 10^{-8}$	$2 \cdot 10^{-7}$	$2.5 \cdot 10^{-8}$	$5 \cdot 10^{-8}$	$5 \cdot 10^{-8}$
LR-Peak	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$2.5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
LR-Final	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$2 \cdot 10^{-6}$	$2.5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$

Table 7.8: Hyperparameters used for fine-tuning the Kinetics checkpoint.

Comments

- For fractal pre-training, increasing resolution leads to nontrivial improvement in downstream results across all benchmarks. This is understandable as a large portion of videos contains tiny details such as small tools that cannot be displayed properly with lower resolutions.
- On most benchmarks, our synthetic approach lags behind Kinetics pre-training. However, as previously explained in the setup, these pre-training approaches are

Pre-training	Res	HMDB51	UCF101	DIVING48	EGTEA	VOLLEY	YUP
Scratch	112	31.5	70.3	4.7	50.5	61.1	43.7
Fractal	112	65.4	87.6	40.3	56.0	83.1	72.7
Scratch	224	35.1	76.9	10.6	53.4	55.7	48.3
Fractal	224	66.5	90.8	**41.2**	59.9	**87.6**	78.2
Kinetics	224	**70.1**	**95.3**	40.9	**64.4**	84.8	**86.9**

Table 7.9: Experiment 6: Downstream validation accuracy for different values of spatial resolution. **Bold** font indicates best results on the specific benchmark.

not comparable, with fractals requiring significantly less time. The required time can be equalized by increasing the number of synthetic videos and according to the previous experiment this would likely result in better downstream performance. As such, the gap between real and synthetic data can be narrowed further.

- On the benchmarks DIVING48 and VOLLEYBALL, synthetic pre-training already surpasses Kinetics despite requiring less resources.

7.3 Manual Error Analysis

To better understand the strengths and weaknesses of the proposed synthetic framework, we manually observed video samples that are misclaissified after fine-tuning. Surprisingly enough, a large portion of such videos share a common characteristic: their label is defined not by a global view of the video but by small details. To be precise, models struggle in the following instances:

- **Interaction with objects**: Downstream datasets contain multiple videos where humans manipulate inanimate objects and tools. In a large portion of such videos, the object of interest is of small scale and covers only a few pixels. The pre-trained models exhibit low accuracy on such samples and it can be assumed that they fail to detect such objects. Some examples of this degenerate behavior are:
  - HMDB51 & UCF101 contain classes such as “Throw”, “Pick Up” “Swing Baseball”, “Brush Teeth” & “Hammering”. Accuracy on these classes is low.
  - The majority of EGTEA videos display cooking ingredients and tools. Failure to detect these objects justifies the extremely poor performance. Large scale pre-training improves accuracy only by 6% compared to training from scratch. This is the smallest improvement amongst all benchmarks.
- **Limb Movement**: Another subset of downstream videos where models underperform involves gestures and other miscellaneous limb motion. As before, in such

samples the label is determined by only a small percentage of the overall motion displayed in the video. Notable examples are:

  - HMDB51 includes classes such as "Clap", "Wave", "Punch" and "Kick". Pre-trained models cannot differentiate between them as well as the object classes which were described earlier.
  - The EGTEA benchmark revolves around cooking. Therefore, most of its videos display hand motion.

- **Facial Movement**: Additionally, pre-trained models struggle with videos that display faces. In such samples the model must learn to differentiate between fine-grained movements of the mouth or recognize subtle facial expressions. Specific instances are the following:

  - HMDB51 incorporates categories that depend on motion of the mouth: "Smoke", "Eat" & "Drink".
  - HMDB51 also contains classes inlolving facial expressions: "Smile" & "Laugh".

As such, it can be deduced that pre-trained models underperform on videos where a very small percentage of the displayed motion determines the correct label. This is not a surprise, as the proposed pre-training framework does not prepare the model for such instances. Indeed, in the proposed synthetic datasets the label depends on the overall fractal formation that is displayed in the video. Cases that depend on local details are nonexistent.

This deficiency should be addressed in future work. We conclude that superior results can be achieved on action recognition benchmarks, if synthetic datasets emulate the described conditions. For a subset synthetic data, the label should be conditioned on a small percentage of pixels, as occurs in real data. This could be achieved either offline with a different generative process that produces videos or online with specialized augmentation modules.

7.4 Conclusions

The present work constructs synthetic datasets used for pre-training neural networks for the task of action recognition. Throughout this chapter, various ablation studies were conducted on multiple downstream datasets. The overall objective was to determine properties of synthetic data as well as general guidelines whose incorporation into pre-training improves downstream performance. Observing experimental results, the following conclusions can be reached:

- Diversity of synthetic pre-training data is a key factor for obtaining stronger visual representations. Diversity can boost results regardless of the characteristics of the downstream dataset.

(a) Throw - Draw Sword (b) Punch - Hug (c) Shoot Gun - Shoot Bow

(d) Smoke - Laugh (e) Talk - Smoke (f) Throw Hammer - Throw Discus

(g) Play Violin - Play Flute (h) Yoyo - Juggle Balls (i) Brush Teeth - Shave Beard

Figure 7.3: Frames from misclassified videos. Green color indicates ground truth, whereas red color indicates the model's incorrect prediction. In such videos the label is often determined by subtle details that cover a small percentage of the overall pixels. As such, the model fails to differentiate between similar categories.

- Downstream results can be additionally strengthened by customizing the pre-training task. This can be achieved by identifying structural properties of downstream datasets and emulating them during pre-training. A few examples are background randomization, periodic motion and camera shaking.
- In case of computational constraints, supervised pre-training is a more cost-effective solution compared to self-supervised counterparts.
- The shape component of synthetic animations is of more consequence for downstream tasks than the component of motion.
- For supervised pre-training, increasing the dataset size consistently improves transferability. The same statement holds for spatial resolution.
- Models pre-trained with synthetic data underperform in the detection of tiny details. This is especially evident for videos whose label is determined by a a small percentage of displayed pixels. Such videos include interactions with tools as well as facial expressions. Future work should mitigate this deficiency by constructing categories that are conditioned on a local cues, as occurs in real data.

Chapter 8

Summary and Future Work

This is the final chapter of the present work. Section 8.1 briefly summarizes its major contributions and conclusions. Next, Section 8.2 presents ideas that could potentially enhance the proposed framework as well as future directions regarding synthetic data for different modalities.

8.1 Summary

The objective of this thesis was to extend the seminal work of [49] and automatically construct synthetic datasets for the task of action recognition, one of the fundamental topics of computer vision. Such datasets can be used for pre-training 3D CNNs instead of Kinetics [11, 50]. This approach can mitigate certain shortcomings of real data such as collection and labeling costs, copyright, privacy as well as human bias.

We commenced by overviewing previous work on IFS fractal images, a generative process that can produce diverse spatial patterns. Particular emphasis was placed on correctly sampling parameters in order to minimize degeneracies in rendered images. We experimented with other families of fractal images, but deduced that they are not suitable for automatic dataset construction.

Afterwards, we produced short video animations by adding a time dimension to the aforementioned fractal images. Naive linear interpolation of fractal parameters leads to unsatisfactory results. However, interpolation of intermediate decomposed matrices mitigates this shortcoming. This approach constructs simple videos containing "forward" motion. The most prominent contribution of this thesis are the proposed domain adaptation techniques. We manually identified properties of real videos and suggested methods of incorporating them in the synthetic animations during pre-training. Such properties are non-linear motion, random background, camera displacement and others. Seeking to identify characteristics of synthetic data that improve downstream results, we additionally constructed alternative datasets that significantly differ from fractals.

After conducting various experimental ablations, we can draw several conclusions regarding pre-training with synthetic data. First, we identify multiple methods of improving downstream performance. This can be achieved by boosting the diversity of synthetic data, by reproducing characteristics of real data such as random background, periodic motion and camera shaking as well as by simply increasing the number of training samples or the spatial resolution. Moreover, we deduce that supervised training objectives are a more cost-effective solution compared to self-supervised alternatives. Furthermore, we discover that the shape of synthetic animations is more significant than their motion. Lastly, we observe that pre-trained models consistently underperform on videos containing subtle details. We propose possible solutions for this shortcoming that are left for future work.

8.2 Future Work

The proposed framework suffers from certain limitations and therefore can be extended in several directions. Specifically:

- **Small Details**: As described in previous chapters, manual observation revealed that the proposed pre-training frameworks results in models that often fail to detect small details present in videos. Examples of this degeneracy are tools, human limbs as well as facial expressions. The cause of this behavior is the pre-training dataset where labels depend exclusively on global context. Therefore, downstream performance can be boosted if during pre-training the model is exposed to videos whose labels are determined by a small percentage of of the displayed motion. This can be implemented via a different noise process that can render videos or specialized augmentation modules.

- **Alternative Generative Processes** : Pre-training with 3D IFS fractals could enhance the resultant model's spatial perception. Additionally, adding julia fractals to the synthetic dataset could boost diversity and therefore performance. However, the former generative process suffers from artifacts such as aliasing, whereas the latter cannot be automated for dataset synthesis. Therefore, additional future work is required.

- **Video Context**: All synthetic videos produced in this work are of short length and depict a single motion. Real videos, however, may display multiple temporally separated elements. The key to understanding such videos is the ability to model contextual relation between distant frames. Emulating this intricacy within synthetic data is expected to boost the network's performance.

- **GAN**: The domain gap between real and synthetic videos was alleviated by manually observing characteristics of the former and replicating them in the latter via handcrafted transformations. Alternatively, this task could be entrusted to generative adversarial networks (GAN) [33, 44]. However, in addition to increasing

computation requirements, GANs often lead to technical challenges involving the training process.

- **Motion Capture**: In this document, motion was generated by sampling random curves. More realistic motion could be achieved by utilizing existing motion capture corpora [54, 72]. Injecting knowledge of human kinematics from such data into synthetic videos could contribute to bridging the domain gap. However, doing so would raise the issue of copyright of the said corpora. Additionally, the construction of synthetic datasets would no longer be completely automatic as it would rely on manually collected data.
- **Other Modalities**: Until now, the only modality employed in training neural networks was raw RGB frames. It is possible to use other modalities such as optical flow [28] to significantly improve results. However, doing so will increase the required computation.

Additionally, as an orthogonal approach, it would be interesting to employ fractals as well as other generative processes to construct synthetic datasets for different domains and tasks:

- **Point Clouds**: By extending the chaos game algorithm to three dimensions, one can create a sequence of moving point clouds. Such synthetic data could be valuable for autonomous driving and robotic manipulation, where similar data formats are employed [105].
- **Biomedicine**: Fractals have been successfully utilized as models of intricate biological structures [18, 43]. With such models, synthetic biomedical datasets could be constructed. This is of great significance as both collection and annotation of real biomedical data is expensive and requires great effort.
- **Other Vision Tasks**: As fractal images produce a great variety of shapes, it is reasonable to employ them for other computer vision tasks such as segmentation [73] and edge detection [92].
- **Music**: It is noteworthy that by using iterated function systems it is possible to generate music scores and produce MIDI files [31]. Neural networks trained on such data would have no issues regarding copyright and ownership of music and could therefore be deployed in commercial applications.

References

[1] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. Youtube-8m: A large-scale video classification benchmark. In *arXiv preprint arXiv:1609.08675* (2016).

[2] Anderson, C., and Farrell, R. Improving fractal pre-training. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022).

[3] Asano, Y., Rupprecht, C., Zisserman, A., and Vedaldi, A. Pass: An imagenet replacement for self-supervised pretraining without humans. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track* (2021).

[4] Baradad Jurjo, M., Wulff, J., Wang, T., Isola, P., and Torralba, A. Learning to see by looking at noise. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2021).

[5] Barnsley, M. F. *Fractals Everywhere*. Morgan Kaufmann, 1993.

[6] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. In *IEEE Transactions on Neural Networks* (1994), vol. 5, no. 2, pp. 157–166.

[7] Birhane, A., and Prabhu, V. U. Large image datasets: A pyrrhic win for computer vision? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2021).

[8] Bregonzio, M., Gong, S., and Xiang, T. Recognising action as clouds of space-time interest points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[9] Buolamwini, J., and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the Conference on Fairness, Accountability and Transparency* (2018).

[10] Burch, B., and Hart, J. C. Linear fractal shape interpolation. In *Proceedings of the Graphics Interface Conference* (1997).

[11] Carreira, J., Noland, E., Hillier, C., and Zisserman, A. A short note on the kinetics-700 human action dataset. In *arXiv preprint arXiv:1907.06987* (2019).

[12] Carreira, J., and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[13] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)* (2020).

[14] Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. In *arXiv preprint arXiv:2003.04297* (2020).

[15] Chen, X., Xie, S., and He, K. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2021).

[16] colah's blog. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 02-09-2022.

[17] Cole, E., Yang, X., Wilber, K., Mac Aodha, O., and Belongie, S. When does contrastive visual representation learning work? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[18] Costabal, F., Hurtado, D., and Kuhl, E. Generating purkinje networks in the human heart. In *Journal of Biomechanics* (2015), vol. 49, pp. 2455–2465.

[19] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2020).

[20] Dimakis, A., and Maragos, P. Phase-modulated resonances modeled as self-similar processes with application to turbulent sounds. In *IEEE Transactions on Signal Processing* (2005), vol. 53, no. 11, pp. 4261–4272.

[21] Ding, S., Li, M., Yang, T., Qian, R., Xu, H., Chen, Q., Wang, J., and Xiong, H. Motion-aware contrastive video representation learning via foreground-background merging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[22] Draves, S. The electric sheep screen-saver: A case study in aesthetic evolution. In *Proceedings of the European conference on Applications of Evolutionary Computing* (2005).

[23] Draves, S., and Reckase, E. The fractal flame algorithm. (2008).

[24] Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. Counting out time: Class agnostic video repetition counting in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[25] Elman, J. L. Finding structure in time. In *Cognitive Science* (1990), vol. 14, no. 2, pp. 179–211.

[26] Farin, G. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press Professional, Inc., 1988.

[27] Feichtenhofer, C., Pinz, A., and Wildes, R. P. Temporal residual networks for dynamic scene recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[28] Fleet, D. J., and Weiss, Y. Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*. Springer, 2006, ch. 15, pp. 237–257.

[29] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In *Biological Cybernetics* (2004), vol. 36, pp. 193–202.

[30] Ghadiyaram, D., Tran, D., and Mahajan, D. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[31] Gogins, M. Iterated functions systems music. In *Computer Music Journal* (1991), vol. 15, no. 1, pp. 40–48.

[32] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT Press Cambridge, 2016.

[33] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2014).

[34] Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., and Bojanowski, P. Self-supervised pretraining of visual features in the wild. In *arXiv preprint arXiv:2103.01988* (2021).

[35] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., and Valko, M. Bootstrap your own latent - a new approach to self-supervised learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2020).

[36] Guo, X., Wu, W., Wang, D., Su, J., Su, H., Gan, W., Huang, J., and Yang, Q. Learning video representations of human motion from synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[37] Hara, K., Kataoka, H., and Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[38] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[39] Heilbron, F. C., Escorcia, V., Ghanem, B., and Niebles, J. C. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[40] Hochreiter, S., and Schmidhuber, J. Long short-term memory. In *Neural Computation* (1997), vol. 9, no. 8, pp. 1735–1780.

[41] HUTCHINSON, J. E. Fractals and self similarity. In *Indiana University Mathematics Journal* (1981), vol. 30, no. 5, pp. 713–747.

[42] Ibrahim, M. S., Muralidharan, S., Deng, Z., Vahdat, A., and Mori, G. A hierarchical deep temporal model for group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[43] Ionescu, C., Oustaloup, A., Levron, F., Melchior, P., Sabatier, J., and De Keyser, R. A model of the lungs based on fractal geometrical and structural properties. In *IFAC Proceedings Volumes* (2009), vol. 42, no. 10, pp. 994–999.

[44] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[45] Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013), vol. 35, no. 1, pp. 221–231.

[46] Kataoka, H., Hara, K., Hayashi, R., Yamagata, E., and Inoue, N. Spatiotemporal initialization for 3d cnns with generated motion patterns. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022).

[47] Kataoka, H., Hayamizu, R., Yamada, R., Nakashima, K., Takashima, S., Zhang, X., Martinez-Noriega, E. J., Inoue, N., and Yokota, R.

Replacing labeled real-image datasets with auto-generated contours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[48] Kataoka, H., Matsumoto, A., Yamada, R., Satoh, Y., Yamagata, E., and Inoue, N. Formula-driven supervised learning with recursive tiling patterns. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops* (2021).

[49] Kataoka, H., Okayasu, K., Matsumoto, A., Yamagata, E., Yamada, R., Inoue, N., Nakamura, A., and Satoh, Y. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision (ACCV)* (2020).

[50] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. The kinetics human action video dataset. In *arXiv preprint arXiv:1705.06950* (2017).

[51] Kim, Y.-w., Mishra, S., Jin, S., Panda, R., Kuehne, H., Karlinsky, L., Saligrama, V., Saenko, K., Oliva, A., and Feris, R. How transferable are video representations based on synthetic data? In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track* (2022).

[52] Kokkinos, I., and Maragos, P. Nonlinear speech analysis using models for chaotic systems. In *IEEE Transactions on Speech and Audio Processing* (2005), vol. 13, no. 6, pp. 1098–1109.

[53] Kotar, K., Ilharco, G., Schmidt, L., Ehsani, K., and Mottaghi, R. Contrasting contrastive self-supervised representation learning pipelines. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2021).

[54] Kovar, L., Gleicher, M., and Pighin, F. Motion graphs. In *ACM Transactions on Graphics* (2002), vol. 21, no. 3, pp. 473–482.

[55] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2012).

[56] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. Hmdb: A large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2011).

[57] Laptev, I., and Lindeberg, T. Space-time interest points. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2003).

[58] Lecun, Y., Haffner, P., and Bengio, Y. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*. Springer, 2000, ch. 19, pp. 319–345.

[59] Lee, A. B., Mumford, D., and Huang, J. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. In *International Journal of Computer Vision (IJCV)* (2004), vol. 41, pp. 35–59.

[60] Levy, O., and Wolf, L. Live repetition counting. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015).

[61] Li, Y., Li, Y., and Vasconcelos, N. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018).

[62] Li, Y., Liu, M., and Rehg, J. M. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018).

[63] Lin, J., Gan, C., and Han, S. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2019).

[64] Loshchilov, I., and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2017).

[65] Loshchilov, I., and Hutter, F. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2019).

[66] Madhusudana, P. C., Lee, S.-J., and Sheikh, H. R. Revisiting dead leaves model: Training with synthetic data. In *IEEE Signal Processing Letters* (2022), vol. 29, pp. 209–213.

[67] Mandelbrot, B. *Les objets fractals : forme, hasard et dimension*. Flammarion, 1975.

[68] Mandelbrot, B. *The fractal geometry of nature*. Freeman, 1982.

[69] Maragos, P. Fractal signal analysis using mathematical morphology. In *Advances in Electronics and Electron Physics* (1994), vol. 88, pp. 199–246.

[70] Maragos, P., and Potamianos, A. Fractal dimensions of speech sounds: Computation and application to automatic speech recognition. In *The Journal of the Acoustical Society of America* (1999), vol. 105, pp. 1925–32.

[71] Marszalek, M., Laptev, I., and Schmid, C. Actions in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[72] Min, J., and Chai, J. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. In *ACM Transactions on Graphics* (2012), vol. 31, no. 6, pp. 1–12.

[73] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., and Terzopoulos, D. Image segmentation using deep learning: A survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), vol. 44, no. 7, pp. 3523–3542.

[74] Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S. A., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., and Oliva, A. Moments in time dataset: One million videos for event understanding. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), vol. 42, pp. 502–508.

[75] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)* (2010).

[76] Nakashima, K., Kataoka, H., Matsumoto, A., Iwata, K., Inoue, N., and Satoh, Y. Can vision transformers learn without natural images? In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022).

[77] Norton, A. Generation and display of geometric fractals in 3-d. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (1982).

[78] Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (2013).

[79] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2019).

[80] Perlin, K. An image synthesizer. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (1985).

[81] Perlin, K. Improving noise. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (2002).

[82] Pitsikalis, V., and Maragos, P. Analysis and classification of speech signals by generalized fractal dimension features. In *Speech Communication* (2009), vol. 51, no. 12, pp. 1206–1223.

[83] Qu, H., Song, L., and Xue, G. Shaking video synthesis for video stabilization performance assessment. In *Proceedings of the Visual Communications and Image Processing (VCIP)* (2013).

[84] Ruderman, D. L. Origins of scaling in natural images. In *Vision Research* (1997), vol. 37, no. 23, pp. 3385–3398.

[85] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision (IJCV)* (2015), vol. 115, no. 3, pp. 1573–1405.

[86] Schuldt, C., Laptev, I., and Caputo, B. Recognizing human actions: a local svm approach. In *Proceedings of the International Conference on Pattern Recognition (ICPR)* (2004).

[87] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[88] Simonyan, K., and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)* (2014).

[89] Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *arXiv preprint arXiv:1212.0402* (2012).

[90] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research* (2014), vol. 15, no. 56, pp. 1929–1958.

[91] Steed, R., and Caliskan, A. Image representations learned with unsupervised pre-training contain human-like biases. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency* (2021).

[92] Sun, R., Lei, T., Chen, Q., Wang, Z., Du, X., Zhao, W., and Nandi, A. Survey of image edge detection. In *Frontiers in Signal Processing* (2022), vol. 2, pp. 420–432.

[93] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[94] Thoker, F. M., Doughty, H., Bagad, P., and Snoek, C. G. M. How severe is benchmark-sensitivity in video self-supervised learning? In *Proceedings of the European Conference on Computer Vision (ECCV)* (2022).

[95] towardsdatascience. Activation functions in artificial neural networks. https://medium.com/swlh/activation-functions-in-artificial-neural-networks-8aa6a5ddf832. Accessed: 02-09-2022.

[96] towardsdatascience. Applied deep learning - part 4: Convolutional neural networks. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed: 02-09-2022.

[97] towardsdatascience. Know your neural network architecture more by understanding these terms. https://medium.com/@shroffmegha6695/know-your-neural-network-architecture-more-by-understanding-these-terms-67faf4ea0 Accessed: 02-09-2022.

[98] towardsdatascience. Overcoming overfitting a model in machine learning. https://medium.com/@jwbtmf/overcoming-overfitting-a-model-in-machine-learning-7dd6324d15bf. Accessed: 02-09-2022.

[99] towardsdatascience. Understanding activation functions in neural networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. Accessed: 02-09-2022.

[100] towardsdatascience. Why relu? tips for using relu. comparison between relu, leaky relu, and relu-6. https://medium.com/@chinesh4/why-relu-tips-for-using-relu-comparison-between-relu-leaky-relu-and-relu-6-969359 Accessed: 02-09-2022.

[101] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015).

[102] Vincent, L. Morphological transformations of binary images with arbitrary structuring elements. In *Signal Processing* (1991), vol. 22, no. 1, pp. 3–23.

[103] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Action recognition by dense trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

[104] Wang, H., and Schmid, C. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2013).

[105] Wang, H., and Tian, Y. Sequential point clouds: A survey. In *arXiv preprint arXiv:2204.09337* (2022).

[106] Wang, J., Gao, Y., Li, K., Lin, Y., Ma, A. J., Cheng, H., Peng, P., Huang, F., Ji, R., and Sun, X. Removing the background by adding the background: Towards background robust self-supervised video representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[107] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2016).

[108] Wikipedia. Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 02-09-2022.

[109] Wikipedia. Recurrent neural network. https://en.wikipedia.org/wiki/Recurrent_neural_network. Accessed: 02-09-2022.

[110] Wilson, B., Hoffman, J., and Morgenstern, J. Predictive inequity in object detection. In *arXiv preprint arXiv:1902.11097* (2019).

[111] Xu, Q., Zhang, R., Zhang, Y., Wang, Y., and Tian, Q. A fourier-based framework for domain generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[112] Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. Billion-scale semi-supervised learning for image classification. In *arXiv preprint arXiv:1905.00546* (2019).

[113] Yamada, R., Kataoka, H., Chiba, N., Domae, Y., and Ogata, T. Point cloud pre-training with natural 3d structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[114] Yang, Y., Lao, D., Sundaramoorthi, G., and Soatto, S. Phase consistent ecological domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[115] Yang, Y., and Soatto, S. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[116] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2018).

[117] Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2017).

[118] Zhao, K., Shen, L., Zhang, Y., Zhou, C., Wang, T., Zhang, R., Ding, S., Jia, W., and Shen, W. Bézierpalm: A free lunch for palmprint recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2022).

[119] Zhou, H., Kadav, A., Shamsian, A., Geng, S., Lai, F., Zhao, L., Liu, T., Kapadia, M., and Graf, H. P. Composer: Compositional reasoning of group activity in videos with keypoint-only modality. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2022).

[120] Zlatintsi, N., and Maragos, P. Multiscale fractal analysis of musical instrument signals with application to recognition. In *Audio, Speech, and Language Processing, IEEE Transactions on* (2013), vol. 21, pp. 737–748.