Video Data Condensation in Machine Understanding

Yi Zhou

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

With the increasing size and complexity of modern datasets and deep neural networks, the computation costs of training deep learning models have grown substantially. This is especially true for video datasets, which tend to be larger in size and more computationally expensive to learn than other types of data. In order to reduce the training computations for the video dataset, we propose a novel video-wise training set condensation method, which synthesizes the original training dataset into a smaller synthetic set by matching the feature distribution between the original training samples and synthetic samples. We evaluate the proposed method by comparing the performance of the synthetic sets with that of single-frame baseline models and models trained on the entire dataset. Our experiments show that firstly, our method outperforms the baseline model under all settings with a maximum normalized accuracy gain of 30.8%. Secondly, our approach significantly reduces training costs in terms of both training time and storage requirement to 2%-5% of the original training set, while maintaining a comparable performance as mentioned above. Furthermore, we also conduct a comprehensive analysis of the impact of different synthetic set generation settings on the synthetic image quality and its performance, which could provide valuable insights for future work on video synthesis.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yi Zhou*)

Acknowledgements

Thanks to all the love and support I received along this journey, and thanks to everyone who has contributed to it.

Many thanks go to my supervisor Hakan Bilen for his invaluable insights throughout the project, to Bo Zhao for his helpful suggestions, to my friends for the constant joy and encouragement they provided, and to my family for their unwavering and irreplaceable love and support.

Table of Contents

1 Introduction **1**
- 1.1 Research Goal and Objectives . . . 3
- 1.2 Contribution . . . 4
- 1.3 Outline . . . 5

2 Background **6**
- 2.1 Maximum Mean Discrepancy (MMD) . . . 6
- 2.2 Related Works . . . 7
  - 2.2.1 Training Set Compression . . . 7
  - 2.2.2 Video Compression and Summerization . . . 8

3 Methodology **10**
- 3.1 Dataset Condensation . . . 10
  - 3.1.1 Problem Definition . . . 10
  - 3.1.2 Condensation and Distribution Matching . . . 11
- 3.2 System Workflow . . . 12
  - 3.2.1 Overall Train-Evaluation Workflow . . . 12
  - 3.2.2 Synthetic Set Generation Workflow . . . 13
- 3.3 Training Model and Evaluation Metrics . . . 14

4 Dataset **16**
- 4.1 Dataset statistics . . . 16
  - 4.1.1 TinyVIRAT . . . 16
  - 4.1.2 UCF101 . . . 17
- 4.2 Preprocessing . . . 18
  - 4.2.1 Frame Extraction . . . 18
  - 4.2.2 Resizing . . . 19
  - 4.2.3 Train-Test Split . . . 19
  - 4.2.4 Dataloader . . . 20
- 4.3 Dataset Performance Evaluation . . . 20
  - 4.3.1 Evaluation Setting . . . 21
  - 4.3.2 Performance Comparison: TinyVIRAT and UCF101 . . . 21
- 4.4 Dataset selection . . . 22

5 Experiments **24**
- 5.1 Experiment settings . . . 24

5.2 Baseline Model 25
5.3 Synthetic Set Generation and Visualisation 27
5.3.1 Challenges in Video-wise Synthetic Set Generation 27
5.3.2 Pre-trained Models for Synthetic Set Generation 28
5.3.3 Synthetic Image Initialisation 29
5.3.4 Difference within Synthetic Set 29
5.4 Test performance 30
5.4.1 Performance Improvement 30
5.4.2 Training Cost 33

6 Conclusion **35**
6.1 Summary 35
6.2 Limitation and future work 35

Bibliography **37**

Chapter 1

Introduction

With the rapid growth of surveillance coverage, the development of filming technologies, and the popularity of social media, there has been a vast increase in the available video resources online [1, 47]. The massive and exponentially growing resources allow for the development of large-scale datasets like ImageNet [11] and Kinetics [25], as well as the large-scale deep models trained on these datasets [40, 24]. However, training such models can be very computationally expensive and time-consuming due to the huge amount of computations involved [50, 20]. Additionally, techniques like hyper-parameter optimization [17, 2, 16] and neural architecture search [43, 68] are often applied during the training process to improve the model performance, and such techniques would require repetitive training on the same dataset multiple times and therefore further increase the training cost [63]. Moreover, storing and transmitting such large-scale datasets is very challenging and costly as well [34].

Various methods have been proposed to reduce the training computations for large datasets by shrinking the training set while maintaining comparable performance. While most of these methods have been primarily focused on image and text datasets, limited research has been conducted on video datasets. However, given the complexity of videos, applying the same method to video datasets may not yield promising results. There are two commonly used methods for training set reduction, namely coreset selection [46] and data distillation [58], which differ in their approach to generating the smaller training set. Coreset selection [27, 26] selects the most representative samples from the original training set to form the output training set. The samples are selected based on certain heuristics such as decision boundary [12, 31], sample diversity [4] and centre distance [46, 26]. Although the selection process can be computationally efficient depending on the selection heuristics, it has two main limitations. Firstly, because the selection heuristics is chosen to target a specific task, its performance would vary a lot across different datasets and tasks, limiting its generalizability. Secondly, the performance of the output training set will be limited to the selected samples since it is only a subset of the original training set. On the other hand, data distillation methods [66, 63] aim to produce a set of newly computed synthetic samples from the original dataset that would achieve comparable performance for a given task. Unlike coreset selection, the performance of the synthetic set is not bounded by a subset of the original

training samples since the features from multiple original samples can be synthesised into a single synthetic sample. However, due to the computational complexity of the synthetic set generation process, data distillation methods often suffer from a lack of computational efficiency [63].

Compared to image and text data, reducing the dataset size for video data is more urgent, as videos are typically much larger in size and require more storage space [47]. While existing video data size reduction methods focus on maintaining the visual quality or interpretability of compressed videos, their performance on machine learning tasks such as classification [29] and recognition [44] is not always considered. Increasing the storage and transmission efficiency is one of the main focuses for video compression, and numerous compression methods have been proposed including standard compression methods like MPEG [30, 48, 56], and relatively new methods like neural compression [65, 53]. They both aim to compress multimedia data including images, videos, and audio, in a reconstructable way while maintaining high visual quality, but with different approaches. Standard compression methods compress data by removing redundant information, while neural compression uses neural networks to learn a representation of the original data, which gives it the potential to achieve a higher compression ratio. Another research focus for video size reduction is to find the highlight for a given video, which is particularly useful in the fields of video browsing and monitoring. A range of video summarization methods [6, 15, 32, 62] addresses this problem by selecting a set of keyframes [19, 51] or key-clips [21, 37] that conclude the main content for a given video. However, these methods are typically evaluated based on the similarity between algorithm-selected and human-selected keyframes, and video data size reduction methods specifically tailored for learning tasks remain highly unexplored.

In this project, we aim to find a way to effectively reduce the training data size for video datasets, with a particular focus on the training performance for learning tasks. The main challenge for video data condensation tasks is that videos are usually high-dimensional, which makes them much harder to learn than other data formats. For the same reason, the training process and condensing process would be usually very computationally expensive and rarely efficient [36, 23].

To address this issue, we propose a novel video data condensation algorithm which uses a distribution matching approach to achieve efficient synthetic set generation. As shown in Figure 1.1, our overall system mainly consists of two stages, synthetic set generation and training/evaluation. The synthetic set is generated by matching the maximum mean discrepancy [18] between the synthetic set and the original set, inspired by Zhao & Bilen (2022) [66]. Compared to other methods, their method has been shown to be efficient in the synthetic set generation process while achieving comparable performance to more computationally expensive approaches. This makes it well-suited for video synthesis tasks that normally require more computations. Unlike Zhao & Bilen (2022), we designed and implemented a video-wise synthetic set generation framework, which has not been used in any other distribution-matching synthesis works. This approach synthesizes images for each video instead of each class, which allows it to (1) maintain the diversity and sample distribution of the original dataset, and (2) have more control and flexibility during the synthesis process. Other than that, space embedding techniques are also used to convert the high-dimensional video data into a family of

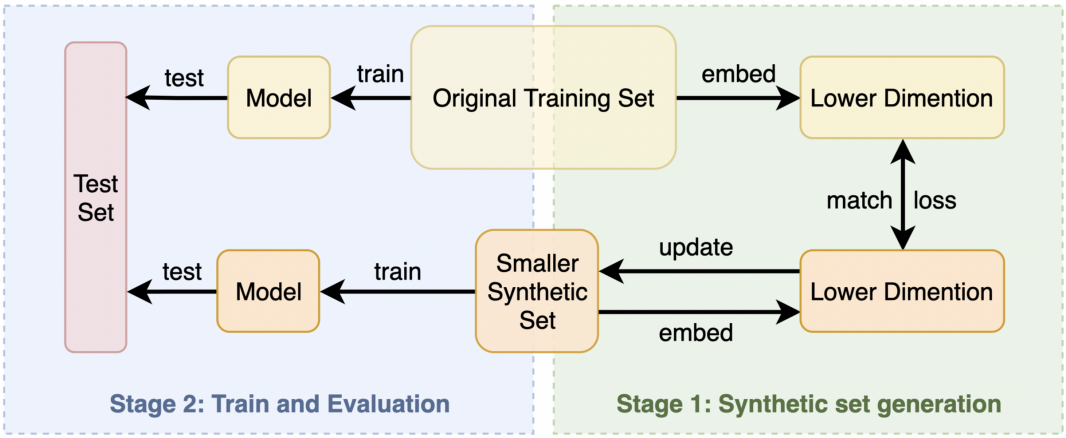


Figure 1.1: Our system workflow mainly consists of two stages: synthetic set generation (right) and training and evaluation (left). Our method aims to synthesise the original dataset at Stage 1 in a way that (1) speedup the training process at Stage 2, and (2) produce comparable performance when testing at Stage 2.

lower-dimensional spaces, which efficiently reduces the training computations without losing key features of a given input. Furthermore, we also include batch division and multiprocessing techniques to address the inefficiency caused by a large number of video samples and therefore further improve the overall system efficiency.

At Stage 2 in Figure 1.1, the generated synthetic set is used to train models, and their performance is evaluated on an unseen test set. We compare the models trained on the synthetic set with the models trained on the single-frame dataset and the original dataset with the same setup. Our method consistently outperforms the baseline model across all synthetic set generation settings, while significantly reducing the training costs in terms of training time and storage required during the training process in Stage 2.

1.1 Research Goal and Objectives

As discussed above, the goal of this project is to develop a video data condensation framework which could efficiently synthesize training data with comparable performance on classification tasks.

Alongside achieving this goal, we also aim to answer the following research questions during the implementation and evaluation process:

- Can the distribution matching data condensation method be effectively applied to video datasets?
- How do the experimental settings, such as different ways of initializing synthetic images and the number of frames extracted from each video, affect the performance of the proposed framework? What specific characteristics have the most significant impact?
- What is the generalizability of the proposed framework across different dataset

sizes and experimental setups?

- What insights can be gained from the experimental results in terms of the challenges that need to be overcome when performing video data condensation?

Given the research questions above, we could further break the overall goal into the following objectives:

- Analysing different video datasets to identify one or more suitable datasets for performance evaluation.
- Preprocessing the selected dataset and building a suitable baseline model that could accurately evaluate the performance of our proposed method.
- Developing and implementing a video-wise synthetic set computation pipeline that utilizes a distribution matching approach to efficiently generate synthetic sets while maintaining diversity and distribution of the original dataset.
- Building an overall train-evaluation system that integrates the synthetic set generation pipeline with model training and performance evaluation to measure the performance gain of our proposed method.
- Evaluating the performance of our method against the baseline model on the selected dataset under different experiment settings and analyzing how different parameters and settings impact the effectiveness of our proposed method.
- Concluding its strengths and limitations, and discussing the potential future directions based on this work.

1.2 Contribution

The project has made several key contributions, including:

- Conducted a comprehensive analysis of TinyVIRAT and UCF101 datasets with basic performance testing and cross-comparison, identifying the primary dataset for experiments.
- Proposed and implemented a novel approach for synthesizing video data by matching data distributions, which can be used to efficiently generate synthetic sets for training models.
- Developed an overall train-evaluation system which includes the synthetic set generation pipeline, model training, and evaluation on both synthetic and real test sets.
- Developed and implemented a synthetic training set generation pipeline with batch division and multiprocessing, achieving efficient per-video synthetic image generation.
- Evaluated the performance of synthetic sets computed under different experimental settings including the dataset sizes and synthetic set generation settings, and investigated their impact on the testing results.

- Examined the limitations of the proposed approach and presented the potential future direction for improvement.

1.3 Outline

The report is structured as below:

- **Chapter 1: Introduction** provides an overview of the project by discussing the motivation and objectives, as well as our main contributions.
- **Chapter 2: Background** discusses the previous work in the fields of training data selection, condensation and video understanding in details by comparing their techniques and limitations.
- **Chapter 3: Methodology** introduces the main train-evaluation framework and especially the synthetic set generation process in details. The overall system workflow, training model used and evaluation metrics are also included.
- **Chapter 4: Dataset** compares and analyses the two potential datasets, TinyVIRAT and UCF101 for the experiment. Based on the results of basic performance testing of each dataset, the primary dataset is selected and then preprocessed.
- **Chapter 5: Experiments** presents the visualisation of synthetic images and the experimental results under various settings. The results are carefully analysed by conducting cross comparisons with each other and with the baseline models.
- **Chapter 6: Conclusions** summarizes the main contributions of this project, discusses the limitations as well as the potential directions of future work.

Chapter 2

Background

In this chapter, we will provide a brief introduction of the basic concept we will use in the following chapters, as well as some related works in the fields of training set compression and video data compression. As this project focuses on the training set compression for video dataset, these related works could provide more insights for the motivation, contribution and the reason for using the specific methodology of our work.

2.1 Maximum Mean Discrepancy (MMD)

Our method aims to achieve video compression by matching the data distribution between the original real dataset and the newly generated small synthetic set. The metric we used to measure the distance of data distributions of two given datasets is Maximum Mean Discrepancy (MMD) [18]. MMD measures the distance between the means of the feature representations of two given datasets in the feature space and has been widely applied to the fields of machine learning [14, 54].

Equation 2.1 shows the general definition of MMD, where H is reproducing kernel Hilbert space (RKHS) [5], P and Q are two data distributions, X and Y are features from the distributions, and $\varphi(X)$ and $\varphi(Y)$ map the features into RKHS.

$$MMD(P,Q) = \| E_{X\sim P}[\varphi(X)] - E_{Y\sim Q}[\varphi(Y)] \|_H \tag{2.1}$$

In our methods, we use the empirical estimate of the data distribution by sampling a number of training samples in both real dataset and synthetic dataset, and compute the estimated MMD based on it. The specific assumptions used in our distribution method, as well as the ways in which we interpret video data, will be further discussed in detail in Section 3.

2.2 Related Works

2.2.1 Training Set Compression

Machine learning models are widely used in numerous fields, to perform tasks including but not limited to object detection [69], face recognition [28], action recognition [59], etc. With the increasing size of datasets [41, 20] and the development of more sophisticated neural network architectures [40, 24], training these models has become increasingly computationally expensive. An effective way of reducing training computations is to compress the training set while maintaining comparable performance. There are two common approaches to reducing the training set size, one is by selecting a subset of the original training samples, and the other is by synthesizing the original training samples into a smaller synthetic set, namely core-set selection [46] and data condensation (distillation) [58, 66], respectively. Our method is inspired by the data condensation approach. In this subsection, we will discuss the two approaches in detail.

2.2.1.1 Core-set Selection

Core-set selection [46] aims to reduce the training dataset size by selecting the most representative samples from the original training set based on some heuristics. The small training set formed by the selected samples should achieve comparable performance to the original training set. The sample selection criteria, i.e. heuristics, differs as the dataset and the targeting task changes. Some commonly used heuristics are sample diversity [27, 46, 60], distance to dataset center[8, 7], forgetfulness [55], uncertainty [22], etc. Depending on the specific task the selected training set is evaluated on, these features would have different levels of repressiveness and importance to the original training set, and therefore different performance.

Toneva et al. [55] focuses on the ”forgetfulness” of the data points. It shows that some data can be forgotten easily during the training process while some others will not be forgotten at all, and therefore some of the unforgettable examples can be dropped without influencing the model generalization. Some other works like Kim & Shin [27] take a diversity-based approach. They focus on selecting diverse examples using local density, and aims to produce a core-set with low similarity.

Depending on the selection heuristics used, the core-set construction process could be computationally efficient. Additionally, methods such as Coleman et al. [9] have been developed to specifically focus on speeding up the selection process. However, this data selection approach have several limitations. Firstly, the performance of the method heavily replies on the chosen selection heuristics for a specific task, and it is not guaranteed to be effective on other tasks. Secondly, since the resulting training set is just a subset of the original set, its performance is limited by the selected samples. In the following section we will discuss the data synthesis methods which do not have such an upper bound.

2.2.1.2 Dataset Condensation

Similar to core-set selection, dataset condensation [58, 66, 67] aims to generate a small training set from a given training set that is much larger in size. However, instead of directly selecting the ”most important” ones from the original set, it chooses to synthesizes new samples which could represent multiple real samples or even the entire dataset. This gives the method potential to exceed the upper limit of the information contained in a given size of dataset.

The problem of data distillation was first introduced by Wang et al. [58]. Their iterative approach aims to minimize the training loss by computing synthetic samples using the network parameter function. Zhao et al. [67] proposes a more efficient approach using gradient matching to match neural network weights trained on the original and synthetic data. More recently, Zhao & Bilen [66] further improves the efficiency of this approach by using Maximum Mean Discrepancy (MMD) [18] to measure and match the distribution difference between the real dataset and synthetic set. This method is more efficient than the previous gradient matching approach. In our project, we choose to use a similar distribution matching method for video dataset synthesis for its efficiency and good performance on very small synthetic sets. The detailed methodology can be found in Section 3.

2.2.2 Video Compression and Summerization

To reduce the storage cost of video datasets, a range of different video data compression techniques have been proposed, such as video summarization[6, 19, 15, 32, 51], and video compression[30, 48, 53, 65]. Most of these methods are evaluated based on the human visual system with different specific focuses, and the performance of compressed video datasets on learning tasks is not taken into consideration. Our method contributes to this highly unexplored domain by focusing on the compression of video datasets for efficient training of machine learning models.

2.2.2.1 Video Summarization

The general concept of video summerization is to detect and present the ”highlight” part of a given video that matches human’s understanding. Depending on different approaches of extract and present the highlight part, these methods can be mainly divided into three categories: video summerization [6, 19, 15, 32, 51], video skimming [64, 13, 21, 37] and video synopsis [61, 39, 35].

Video summarization is a frame-based video condensation technique. It generates a ”summary” of a video by selecting a set of keyframes which could represent the whole video the most. The frames are selected based different heuristics like colour feature [6, 32], motion activity [62, 19, 15] and distance to the cluster center [6, 51]. Unlike video summarization which selects the key-frames, video skimming aims to selects a set of key-clips and form them into a shorter version of video. The key-clips are selected by criteria similar to video summarization methods, like colour feature [21, 57], motion activity [37, 3, 64]. Besides, the audio feature is also considered in some methods [13]. Video synopsis uses an activity as a processing unit instead of frames or clips,

and aims to present actions that originally happened at different times simultaneously [45, 33, 35, 39, 38].

While video summerization methods are widely used in the domain of video monitoring and browsing, it has some limitations. Firstly, similar to core-set selection methods, video summerization relies heavily on the specific heuristics, which highly limits its generalization. Secondly, the videos summary only targets on the human understanding of a video, so its performance on machine learning tasks is unpredictable.

2.2.2.2 Video Compression

Instead of selecting certain parts of videos, video compression methods aim to compress the entire video in a reconstructable way, and maintain a high visual quality of the reconstructed videos at the same time. Standard compression like MPEG [30, 56, 48] achieves this in a traditional way, which removes the redundant information between frames and within each frame based on the computed similarity. A relatively new approach is neural compression [65, 53, 36], which includes neural networks in the compression process, giving them the potential to achieve a higher compression ratio while maintaining the video quality. However, these methods often suffer from computational inefficiency, and similar to video summarization, they focus solely on the human visual system. In contrast, our method also involves neural networks in the compression process but focuses on sample distribution matching rather than simply removing redundancies. This gives it the ability to achieve comparable performance as the original dataset on learning tasks.

Chapter 3

Methodology

In this chapter, we will present the methodology and the system design for the video data condensation method we proposed. We start by introducing the definition of the dataset condensation problem and the existing work on the image datasets that this project is inspired by. We will then discuss how this method will be modified to fit the scenario of video data condensation by presenting our synthetic set generation and evaluation pipeline. Finally, we will introduce the training model and evaluation metrics we used for performance evaluation.

3.1 Dataset Condensation

3.1.1 Problem Definition

Image and Video The data condensation problem for image datasets as defined in Wang et al. [58] is to condense a large scale training set $\boldsymbol{T}$ (See Equation 3.1) into a smaller synthetic set $\boldsymbol{S}$ (See Equation 3.2) while $\boldsymbol{T}$ and $\boldsymbol{S}$ should have comparable performance on unseen testing data.

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{3.1}$$

$$S = \{(s_1, y_1), (s_2, y_2), ..., (s_m, s_m)\} \tag{3.2}$$

where (x_i, y_i) is the ith sample in the training set and its label, (s_j, y_j) is the jth sample and its label, n and m are the size of the original training set and the synthetic set respectively with $m << n$.

Similarly, for video dataset condensation problem, our goal is to condense a training set with video frames $\boldsymbol{V}$ (See Equation 3.3) into a smaller synthetic set with synthetic images $\boldsymbol{S}$ (See Equation 3.4) while $\boldsymbol{V}$ and $\boldsymbol{S}$ should have comparable performance on unseen testing data when performing video classification tasks.

Note that in this specific approach of interpreting video condensation problem we proposed in 3.3 and 3.4, video-wise condensation boundary is used instead of class-wise

boundary like in 3.1 and 3.2. As a result, the number of videos p is the same in V and S, but the number of samples from each video reduces from n to m.

$$V = \{(x_{11}, y_{11}), (x_{12}, y_{12}), ..., (x_{21}, y_{21}), (x_{22}, y_{22}), ..., (x_{pn}, y_{pn})\} \tag{3.3}$$

$$S = \{(s_{11}, y_{11}), (s_{12}, y_{12}), ..., (s_{21}, y_{21}), (s_{22}, y_{22}), ..., (s_{pm}, s_{pm})\} \tag{3.4}$$

where (x_{ij}, y_{ij}) is the jth frame from the ith video in the training set and the label of the ith video, (s_{kl}, y_{kl}) is the lth synthetic image from the kth video and the label of the ith video, p is the number of videos in both the original training set and the synthetic set, and n and m is the number of samples from each video in the original training set and the synthetic set respectively with $m << n$.

Image Condensation v.s. Video Condensation The goal of data condensation for images and videos are similar, as in both problems we want a small synthetic set that is computed from the original training set and is representative of significant features of the training samples in terms of classification performance. While the approach for images could also be applied to the video dataset directly, which is keeping the number of classes fixed and condensing the samples within each class, we took a slightly different approach in this project. Instead of classes, we use videos as our condensation boundary, which means that each synthetic image is generated only from the frames extracted from one single video. (This process is also described with implementation details in Section 3.2 about the system pipeline.)

This approach could ensure that the synthetic set includes the features extracted from each video and therefore no training samples are wasted. This could also help to preserve the diversity of the original set and avoid computing synthetic images with very high similarity, which would affect the system performance. Additionally, it gives us more control of the synthetic sample generation process, and more flexibility in modifying the number of synthetic images generating from each video and also the overall size of the synthetic set.

3.1.2 Condensation and Distribution Matching

To generate synthetic images that are representative of the original training set, we need a method to extract features from the original video frames and compute a set of synthetic images that contain as much useful information as possible. As we aim for the performance of learning tasks like classification, the ”useful information” here can be represented by the key features of a given sample. Therefore, our goal is to match the sample feature distribution between the real and synthetic data, i.e. to minimize the difference or the distance between the data distributions between them.

As the dimension of video and image data is typically very high, computing the real data distribution directly can be computationally expensive. To address this issue, we transform each real image, which has a dimension of d, into a training sample with a lower dimension of d'. This lower-dimension embedding is achieved using a set of parametric functions φ_υ with different parameters υ [66]. Each of these functions

provides a partial interpretation of the input feature, and their combination forms a comprehensive one.

This process is illustrated by Equation 3.5, where S and T are the synthetic set and original set respectively, $Loss(S,T)$ is the training loss that we want to minimize during the distribution matching process, P_υ is the distribution of network parameters with υ being the parameter, and *Distance* is a distance measurement metric.

$$Loss(S,T) = E_{\upsilon\sim P_\upsilon}[Distance(S,T,\upsilon)] \tag{3.5}$$

We adopt the Maximum Mean Discrepancy (MMD) [18] as our distance measurement metric due to its effectiveness in computing the distance between means of the feature representations of given data sets (more details can be found in Section 2.1). As we use the lower-dimensional partial interpretation of an input feature for distribution matching, MMD will be computed using this partial interpretation as well, where the distribution distance between two sample sets is defined as:

$$Distance(S,T,\upsilon) = \| \frac{1}{|T|}\sum_{i=1}^{|T|}\varphi_\upsilon(x_i) - \frac{1}{|S|}\sum_{j=1}^{|S|}\varphi_\upsilon(s_j) \|^2 \tag{3.6}$$

where P_υ is the distribution of parameters with υ being the parameter, $|T|$ and $|S|$ are the sizes of the real and synthetic datasets respectively, $\varphi_\upsilon(x_i)$ is the lower-dimensional ith sample in the real dataset and $\varphi_\upsilon(s_j)$ the lower-dimensional jth sample in the synthetic dataset.

By reducing the MMD loss, the synthetic dataset keeps being updated gradually throughout the training process until the MMD difference between real and synthetic data reaches a minimum.

3.2 System Workflow

3.2.1 Overall Train-Evaluation Workflow

Figure 3.1 shows the train-evaluation pipeline of our system, which consists of three main stages. The first stage is optional and aims at improving the efficiency of synthetic data generation by dividing the original dataset into several batches. This allows for the parallel generation of synthetic images from each video within each batch, as detailed in Section 5.3.1.

The second and third stages, which are the core of our method, are responsible for generating a synthetic dataset and evaluating its performance, respectively. In the synthetic data generation stage, we uniformly extract a fixed number of k frames from each video in the original dataset, and apply pre-processing techniques such as resizing before saving them locally. We also separately save the middle frame of each video to form the baseline dataset for testing the synthetic set performance in the evaluation stage. Further details on the frame extraction criteria, the choice of k, and other implementation

aspects are discussed in Section 4.2.1. The k real frames will then be synthesized into one synthetic image by matching the lower-dimensional features with the original dataset, such that the resulting synthetic images will have a similar data distribution. To achieve an efficient video-wise synthetic image generation process, we designed and implemented a general synthetic image computation pipeline, which will be described in detail in the next section.

At the evaluation stage, the generated synthetic sets, as well as the baseline datasets and the entire original datasets will be used to train randomly initialised models. The models trained from these different datasets will then be evaluated on the same unseen test set. The test accuracy, training time and storage requirement are used as three key metrics to determine the effectiveness of our method.

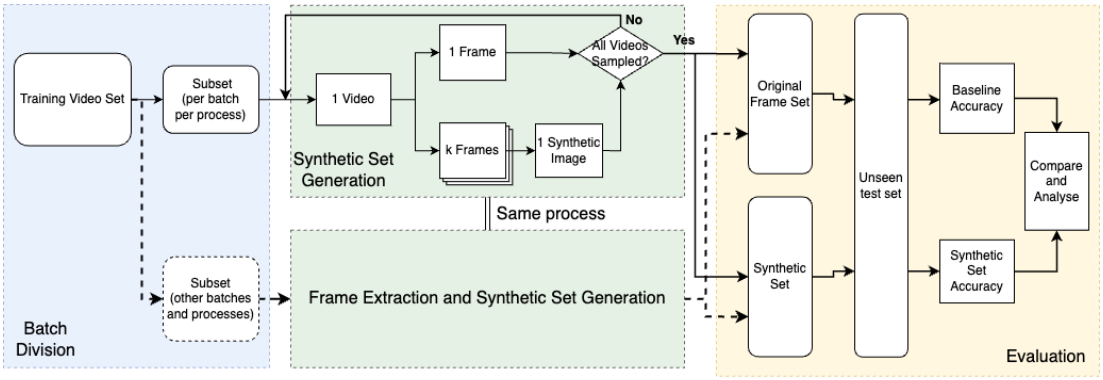


Figure 3.1: The overall train-evaluation system workflow with three main stages: batch division (see blue section), frame extraction and synthetic set generation (see green section), and evaluation (see yellow section).

3.2.2 Synthetic Set Generation Workflow

As the synthetic set generation is the core procedure of our system, we will describe it in detail in this section.

The entire workflow can be divided into four main steps, as shown in Figure 3.2:

1. **Synthetic set initialisation:** There are two available approaches for synthetic set initialisation: Initialise with real images and initialise with random noise. Since the synthetic images will be updated towards the same original dataset, the synthetic images trained with different initialisation settings and their performance is not expected to have much difference. This has also been proved when analysing generated synthetic set in Section 5.3.3.

2. **Lower dimension embedding:** Both the training set and synthetic set are embedded to lower dimension in this step, allowing for a more efficient distribution matching and synthetic set updating in the following steps. The low-dimensional embedding spaces can be sampled from the original dataset using two types of parametric functions: randomly initialised models and pre-trained models. The detailed description of these models and the difference in their performance will be discussed in Section 5.3.2 and 5.4 respectively.

3. **Distribution matching:** The MMD distance in data distribution between the low dimensional training set and synthetic set features is computed in this step. By minimizing the distance loss, the synthetic data distribution will move towards the estimated real data distribution gradually in each iteration.

4. **Synthetic set updating:** Synthetic samples will be updated towards to sample distribution of the real dataset with stochastic gradient descent in this step. This is achieved by minimizing the MMD loss computed from the previous step. After each update, the feature distribution of the synthetic set should be closer to the feature distribution of the real dataset.

Note that steps 2-4 are performed recursively until the MMD loss between the training set and the synthetic set converges to a minimum. The output synthetic images would then be added into the synthetic set and wait to be evaluated once the whole generation process is completed.

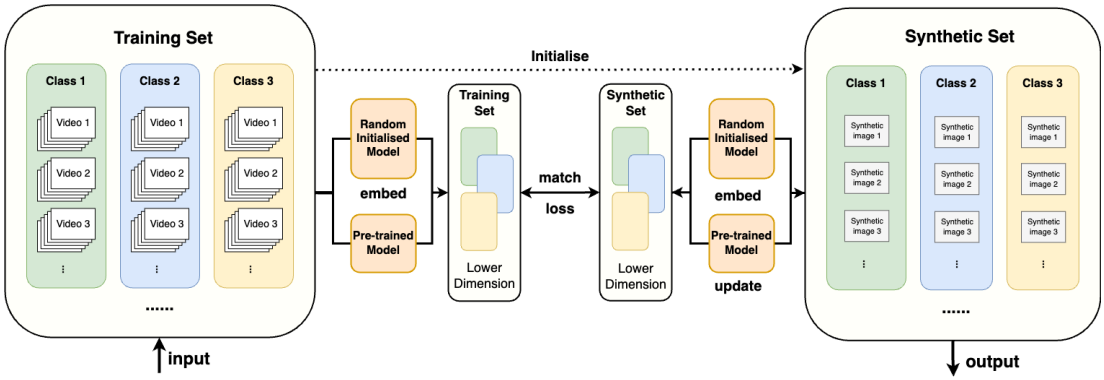


Figure 3.2: The synthetic set generation pipeline, consists of four main procedures: synthetic set initialisation, lower dimension embedding, distribution matching and synthetic set updating.

3.3 Training Model and Evaluation Metrics

Training Model Convolutional Neural Network (ConvNet) is used as both the training model for loss matching and the pre-trained models. Each ConvNet consists of convolutional layers, pooling layers, and fully connected layers, which are applied to extract features, reduce feature size and perform classification respectively. In this project, we use ConvNets that are initialised with random weights and fixed width 128 and depth 3, and activation function ReLU.

Evaluation Metrics In order to evaluate the performance of the synthetic set and the baseline model, a range of evaluation metrics are used for results analysis. Especially, three key metrics are most commonly used throughout the experiments:

- **Class-wise accuracy** The percentage of testing samples that are correctly classified out of all testing samples for each class. This provides us insights on how the model performs for each class, which would be particularly useful when working with an imbalanced dataset.

- **Overall accuracy** The percentage of testing samples that are correctly classified out of all testing samples in the testing set. This provides a general idea of the overall performance of the model.
- **Confusion matrix** It gives us a detailed description of the model performance by providing the number (percentage) of true positive, false positive, true negative, and false negative predicted samples for each class. This information could be particularly helpful when identifying classes that are more likely to be misclassified and the confused classes.
- **Normalized gain/accuracy improvement** Other than the more commonly used standard evaluation metrics as mentioned above, we used an extra parameter, normalized gain, in Section 5 to evaluate the performance of our method more clearly. It is calculated by dividing the accuracy improvement by the difference of the baseline accuracy and accuracy upper bound as in 3.7, where $Accuracy_{syn}$, $Accuracy_{baseline}$ and $Accuracy_{max}$ refer to the test accuracy of our synthetic set, the baseline model and the whole dataset respectively in this project.

$$Gain = \frac{Accuracy_{syn} - Accuracy_{baseline}}{Accuracy_{max} - Accuracy_{baseline}} \tag{3.7}$$

Chapter 4

Dataset

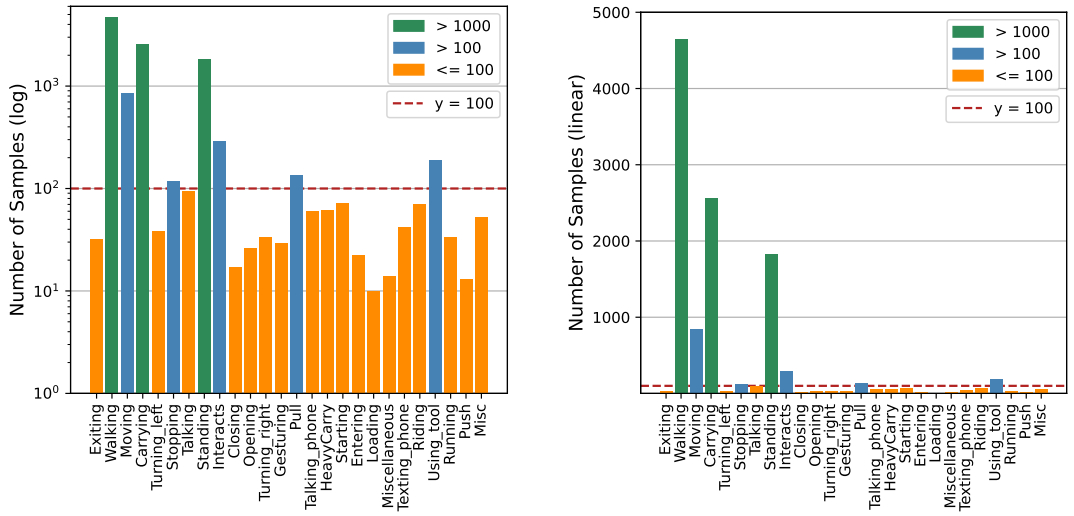
In this chapter, we will discuss our dataset selection criteria and the two datasets we selected and explored as candidates, TinyVIRAT and UCF101. We talk about the data pre-processing techniques used and the design choice we made during the procedure. Finally, a basic performance comparison is presented with a discussion about the dataset selection decision based on this performance.

4.1 Dataset statistics

4.1.1 TinyVIRAT

In this project, we want to focus on action classification datasets with relatively low resolution for computational efficiency and better generalisation. Therefore, we firstly considered TinyVIRAT dataset (Demir et al. (2020) [10]) which is a surveillance video dataset containing 7,663 training and 5,166 testing videos with 26 action labels. This dataset particularly focuses on low-resolution video collected from real-world natural human activities like walking, standing, talking, carrying, etc. The resolution of the videos lies within [10x10 - 128x128], and the average number of frames is around 94. (More details can be found in Table 4.1.)

After carefully analysing the TinyVIRAT dataset, we surprisingly found that it has a very imbalanced sample distribution. Although a class-wise sample distribution plot is provided in their paper (Demir et al. (2020) [10]), this feature is not emphasized. Because in their plot, the y-axis which represents the number of samples for each class, is displayed in a logarithmic scale, just like in Figure 4.1a. As a consequence, even a small visual difference in their plot may indicate a significant difference in the sizes of the two classes. After plotting the class size on a linear scale (see Figure 4.1), it becomes more noticeable that the TinyVIRAT dataset is highly imbalanced. As shown in Figure 4.1, some of the common activities, such as ”walking” and ”carrying”, have more than 4000 and 2500 samples respectively, whereas several less frequent activities, including ”loading”, ”closing”, and ”opening”, have less than 100 samples each. This nature of the dataset would cause several problems as we will further discuss in Section 4.3.2 and 4.4.



(a) Sample distribution in log scale.

(b) Sample distribution in linear scale.

Figure 4.1: The sample distribution of TinyVIRAT dataset in log and linear scales.

Furthermore, the imbalance problem still exists in TinyVIRAT v-2 dataset [52], which is an extended version of the original TinyVIRAT dataset with more indoor activities included. As a result, we turned our attention to the UCF101 dataset [49], a dataset with much more balanced classes, which we will discuss in the following section.

Dataset	# classes	# samples	Resolution	Avg # frames
TinyVIRAT	26	12829	10x10 - 128x128	93.93
UCF101	101	13320	320×240	180.25

Table 4.1: Statistics comparison between TinyVIRAT and UCF101 dataset. (#: Number of, Avg: Average.)

4.1.2 UCF101

UCF101 [49] is a large-scale human action dataset with 101 action classes and 13320 video clips. This dataset contains a wide range of videos in terms of action types and filming environments, which provides us with a diverse training set. Figure 4.2 provides a visualisation of some of the action classes in UCF101.

Compared to TinyVIRAT, UCF101 has a much more balanced sample distribution across all classes, as shown in Figure 4.2. Each action class comprises between 100 to 170 video clips, with clip lengths ranging from 1 second to 71 seconds and a fixed frame rate of 25 fps. This provides a larger training dataset, as the average number of frames in each video in UCF101 (180.25) is nearly twice that of TinyVIRAT (93.93). The detailed statistics comparison can be found in Table 4.1.

Resizing needs to be performed during the pre-processing stage as the video clips in the UCF101 dataset have a relatively high resolution of 320×240 compared to TinyVIRAT. Since we will focus on low-resolution samples in the project, we have to resize the

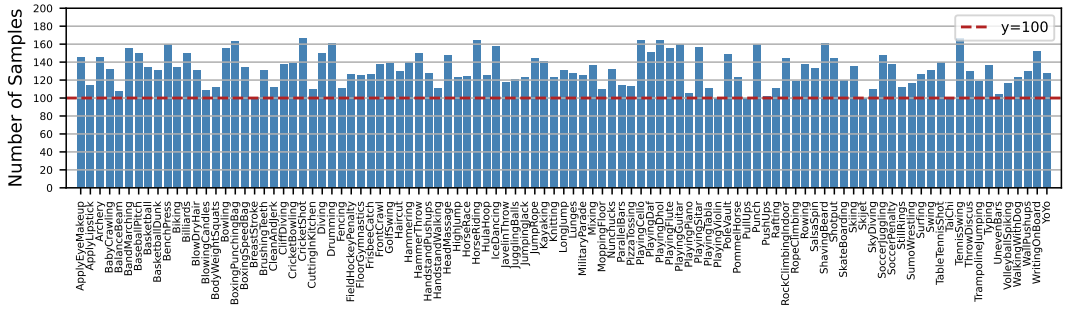


Figure 4.2: UCF101 sample distribution across 101 classes

extracted video frames to lower the frame resolution manually. The details about the data preprocessing procedure can be found in the following section (Section 4.2).

4.2 Preprocessing

A series of data pre-processing steps are performed on the original video dataset. We will discuss them in detail in terms of the design choices we made and the reasons in this section.

4.2.1 Frame Extraction

In our training process and experiments, frames are used as the basic sample unit. As our original training data are videos, we need to extract frames from them in order to get a frame-wise dataset. Since different videos vary in length, the number of frames extracted will also differ. To maintain the original dataset distribution and improve system efficiency, a fixed number of frames N_f, will be extracted from each video.

The value of n_f is chosen from the range $[1, (N_{min} + \theta)]$, where N_{min} is the minimum number of frames in a video sample in the dataset and θ is the window size. The window size θ allows us to extract more frames than N_{min} from the dataset, providing a more flexible frame set size. The reason for adding a window θ is that we do not want to restrict the frame number to N_{min} since the minimum number of frames in some video samples can be very small, whereas the average number of frames is much larger. Therefore, we use a window size θ to control the number of frames to be extracted from each video sample. This approach allows us to maintain the sample diversity in the original dataset without being bounded by the minimum frame number in the dataset or exceeding it without any upper bound. When a video consists of fewer frames than the N_f we choose, we will randomly extract $n_{f'}$ frames from the existing video frames, where $n_{f'}$ is $N_f - n_f$ and n_f is the number of frames in the given video.

Figure 4.3 provides a detailed illustration of the frame extraction process. For instance, suppose we want to extract a fixed number of 3 frames from each video as shown in this figure. For longer video clips containing 100 and 20 frames, we will extract the 1st, 50th, 100th and the 1st, 10th, 20th frames, respectively. However, for shorter clips with fewer than 3 frames, additional frames will be randomly extracted from the existing

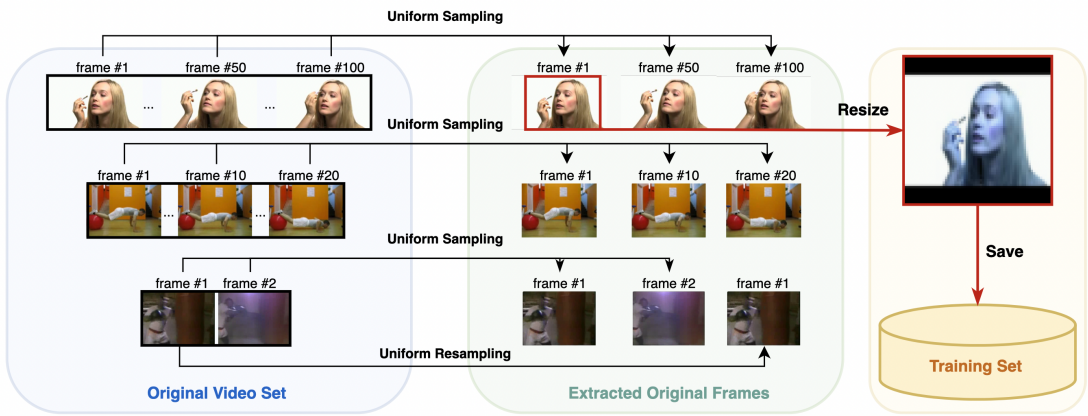


Figure 4.3: The data pre-process workflow mainly consists of three stages: frame extraction, frame resizing and saving frames locally.

frames to form a set of 3 frames. For example, for a clip with 2 frames, we will extract the 1st and 2nd frames and then repeat the 1st frame to form a set of 3 frames.

Additionally, for the baseline model, we extract exactly one frame from the middle of each video to form the training set. The middle frame is chosen as it is the most likely to be the most representative frame and capture the main scene where the action is being performed.

4.2.2 Resizing

As discussed in section 4.1.2, resizing is necessary to have a standardized training set while maintaining a manageable dataset size. There are two options available for resizing images:

- Perform the resizing on-the-fly, i.e. resize each training sample as it is loaded by the dataloader during the training process.
- Resizing images as a preprocessing step and saving the output data to disk.

After trying both approaches, we determined that the second method is better suited for our specific context and is more efficient with the following reasons:

- Performing the resizing on-the-fly significantly slows down the training process, as it reduced the efficiency of GPU parallel processing.
- Since a fixed resolution is used for all training samples, saving resized data locally allowed us to avoid repeatedly performing the resizing operation during training and save more training time.

4.2.3 Train-Test Split

TinyVIRAT The TinyVIRAT dataset comes with a pre-defined split of training and testing sets, consisting of 7,663 and 5,166 video samples, respectively. This split provides a roughly balanced ratio of $Train/Test \approx 60\%/40\%$. In addition, to evaluate

and fine-tune our models, we further split the testing set into two subsets, allocating 50% of it to a validation set. This results in a final split ratio of $Train/Validation/Test \approx 60\%/20\%/20\%$. Notably, the class distribution of samples in the testing set matches that of the training set, ensuring a fair and representative evaluation of our models' performance.

UCF101 The UCF101 dataset consists of 13320 video samples, which we split manually into three exclusive subsets: the training set, the validation set, and the testing set. The split ratio is $75\%/10\%/15\%$, with 9990 samples, 1332 samples and 1998 samples in the training set, validation set, and testing set respectively. The split ratio is chosen to ensure that the model is trained on the majority of the available data while still having enough data reserved for model validation and testing.

Dataset	# classes	# all	# train	# val	# test	Train:Val:Test
TinyVIRAT	26	12829	7663	2583	2583	60:20:20
UCF101	101	13320	9990	1332	1998	75:10:15

Table 4.2: Train/Validation/Test split for TinyVIRAT and UCF101 dataset.
(#: Number of, all: whole dataset, train: training set, val: validation set, test: testing set.)

4.2.4 Dataloader

Pytorch dataloader provides parallelism for data loading, which allows multiple worker threads to load and preprocess data in parallel to speed up the process. Because TinyVIRAT is a relatively new dataset that is not included in the predefined dataset in Pytorch Dataset and we need to load the preprocessed images that are stored locally, we need to customize the PyTorch dataloader to suit our settings. The newly created objects included four main categories:

- Dataset and Dataloader for TinyVIRAT original frames.
- Dataset and Dataloader for UCF101 original frames.
- Dataset and Dataloader for TinyVIRAT synthetic images.
- Dataset and Dataloader for UCF101 synthetic images.

For each of the categories, we can decide to load the corresponding dataset with different settings by passing a parameter indicating the number of frames per video we want. We will discuss the specific settings used for baseline testing and experiments in the following section and Chapter 5.

4.3 Dataset Performance Evaluation

In order to further explore and analyse these two datasets, we performed a few experiments with the basic settings we would use for the algorithm evaluation stage in Chapter 5. We will discuss them in detail in this section.

4.3.1 Evaluation Setting

The dataset performance testing includes two components for each of the datasets:

- Lower bound (the baseline model): Train on a single frame per video (middle frame) for all videos in the training set.
- Upper bound: Train on 20 frames per video for all videos in the training set.

The lower and upper bounds for the number of frames used in training, namely 1 and 20, were chosen based on the experimental design. Our initial experiment plan was to extract 20 frames from each video and use them to create a synthetic image, which was then used to train models for further performance evaluation. Since the synthetic training set was created using the 20 frames from each video, we expected its performance to be bounded between that of the 1-frame-per-video and 20-frames-per-video training sets. Therefore, we want the range between the lower and upper bound to be as wide as possible, which would give us a more clear performance difference for determining any accuracy improvement in the synthetic set, and therefore a more accurate evaluation of the effectiveness of our method.

To ensure a fair comparison between the two datasets, we used a fixed resolution of 64 $\times$ 64 for both of them. The models were trained using a randomly initialised ConvNet with a learning rate of 0.01, a weight decay coefficient of 1e-05, and a batch size of 64 for 8 epochs.

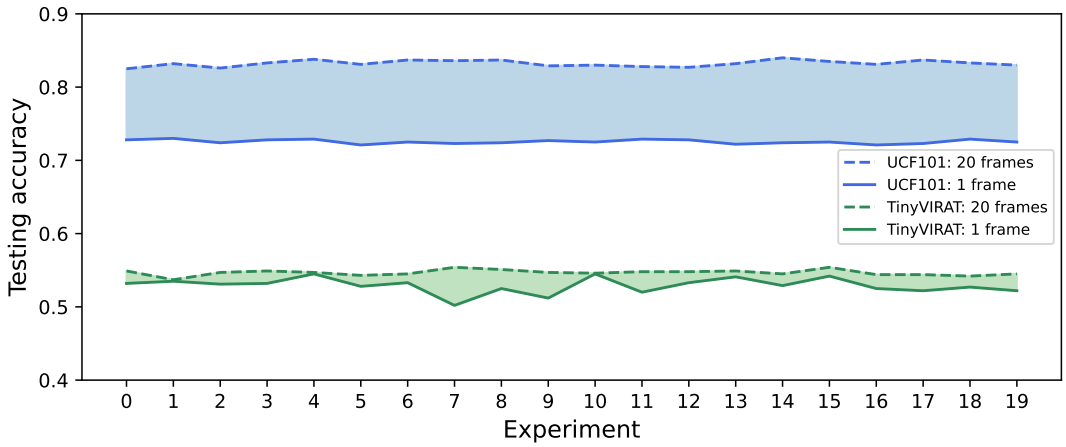


Figure 4.4: The upper and lower accuracy bound for UCF101 and TinyVIRAT datasets. It can be seen that UCF101 has a wider range with more stable upper and lower boundaries.

4.3.2 Performance Comparison: TinyVIRAT and UCF101

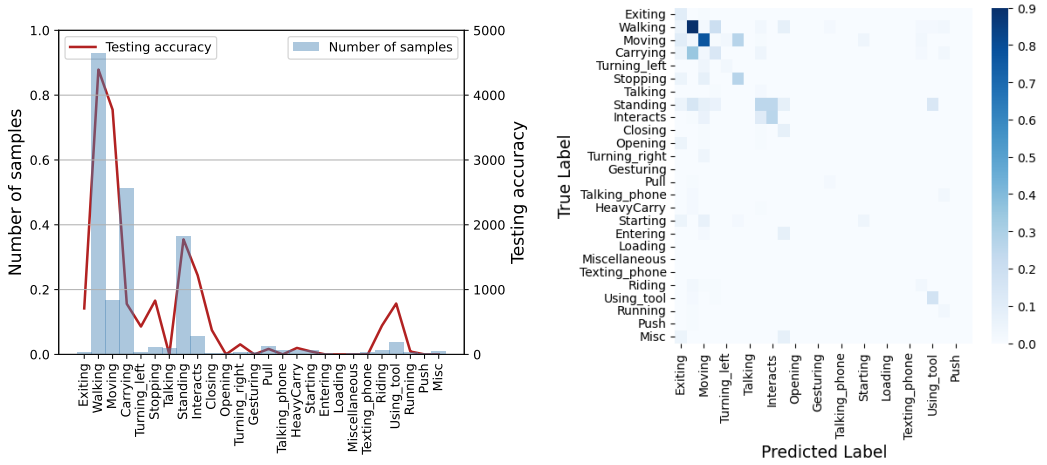
Figure 4.4 shows the model performance for the TinyVIRAT and UCF101 datasets. The y-axis shows the average accuracy among all classes on unseen testing sets for the two datasets, and the x-axis indicates different experiments. The coloured areas between two pairs of lines with colours light blue and light green indicate the expected accuracy range for the synthetic sets for UCF101 and TinyVIRAT datasets respectively.

There are several points that we can learn from Figure 4.4:

- UCF101 has a better overall performance than the TinyVIRAT dataset, although the number of classes of UCF101 is almost more times that of TinyVIRAT.
- The difference in accuracy between the 1-frame and 20-frame training sets of UCF101 is much larger than that of TinyVIRAT.
- The UCF101 performance is more robust than TinyVIRAT.

The imbalanced nature of the TinyVIRAT dataset can be the cause of these issues. Figure 4.5 shows the class-wise accuracy and confusion matrix for the TinyVIRAT testing results. We can see from the figure that larger classes have much higher accuracy than smaller classes with many rarer labels misclassified as the most common labels. As a result, the overall average accuracy is dragged down by the poorer performance of smaller classes. Additionally, the lack of training samples in smaller classes (less than 100 samples per class) leads to the lack of robustness and performance differences between different frame settings.

Furthermore, we also tried different class balancing techniques on the TinyVIRAT dataset including duplicating smaller classes and dropping some of the smallest classes. However, duplication only increases the overall accuracy by around 1% with similar performance for the lower bound baseline model, which gives us a similar narrow accuracy range. On the other hand, dropping classes for TinyVIRAT would lead to insufficient classes to use and test, since there are only 26 classes in TinyVIRAT and only 4 of those have more than 500 samples.



(a) Testing accuracy for each class with class size for TinyVIRAT dataset.

(b) Confusion matrix for TinyVIRAT testing result.

Figure 4.5: Detailed testing performance for TinyVIRAT dataset.

4.4 Dataset selection

After a thorough exploration and analysis of both datasets, we have decided to use UCF101 as our primary dataset for the following experiments. This decision was based

on several factors:

- UCF101 provides a wider range of expected performance between the lower and upper bounds, which is beneficial for assessing the performance of synthetic sets during experiments.
- UCF101 has a balanced class distribution, which makes the model more robust and representative of overall performance improvement. This is in contrast to the imbalanced nature of the TinyVIRAT dataset, which can negatively affect performance.
- UCF101 has a larger average number of frames per video, providing more material for frame condensation during experiments.
- UCF101 has more classes compared to TinyVIRAT, which makes the experiment results more generalized and reflective of real-world scenarios.

Chapter 5

Experiments

In this chapter, we provide a comprehensive analysis of the synthetic image generation process and the performance of the generated synthetic set. We start with introducing the experimental settings and baseline model, and then move on to the visualisation of synthetic images generated from different settings. The experimental results obtained under various settings with the performance evaluation metrics will also be presented in terms of test accuracy, training time and storage requirement, along with the discussion of how different experiment settings influence the synthetic image generation and synthetic set performance.

5.1 Experiment settings

Here is the list of experiment settings used in the following experiments:

- **Dataset:** UCF101
- **Frame resolution:** 64×64
- **Number of frames per video:** 20, 50
- **Training set size:** 20% of the whole dataset (2697 videos), 75% of the whole dataset (9990 videos)
- **Testing set size:** 15% of the whole dataset (1998 videos)
- **Training model:** ConvNet with width 128, depth 3, activation function ReLU
- **Experiment environment:** NVIDIA GTX 1060 GPU with 6GB RAM and 16GB CPU RAM

We used two frame number settings, 20 and 50 frames per video synthesized into a single image in our experiments, for the following reasons. Firstly, while our initial experiment plan using 20 frames showed some improvement over the baseline model, we wanted to investigate whether using 50 frames could extract more information and produce even better results. Secondly, using two different frame settings allowed us to evaluate the effectiveness of the feature extraction and compression process. This helped

us understand how the number of frames affected the quality of synthetic images and whether the computational cost of using more frames was justified by the performance improvement. Additionally, we chose specifically 20 and 50 frames to generate the synthetic images because we wanted to strike a balance between extracting enough information for each video (thus avoiding a frame number that is too small) and ensuring that smaller videos have enough frames to extract (thus avoiding a frame number that is too large). More experiment details can be found in Section 5.4.

Two different training set sizes, 20% and 75% of the entire UCF101 dataset are used in the experiments as well. These two settings were chosen to simulate the model performance on both smaller and larger datasets, providing more information on the generalizability of our method. Furthermore, due to computational limitations and the complexity of the overall synthetic set generation and evaluation system, generating an entire synthetic set and testing its performance is very time-consuming. Using a smaller dataset allows for faster setting testing and identification of the correct direction, providing more insights into the next step of the experiments. More experiment details can be found in Section 5.4.

For all the experiment settings mentioned above, we used a fixed testing set setting, which is 15% of the entire dataset with 50 frames per video. This could make sure that the testing results are consistent and allow us to make direct comparisons between the model performance under different settings, and further analyse how different parameters impact the experimental results. Moreover, the variety of experiments we conducted allowed us to evaluate the generalizability of the proposed method on datasets with different characteristics. This could also deepen our understanding of the method's strengths and weaknesses, and provide us insights into any potential future research.

5.2 Baseline Model

Given the diversity of our experiment settings, we used different baseline models to evaluate the model performance. In this section, we focus on the overall accuracy and expected accuracy range of the baseline model, as it will be used as one of the key metrics when evaluating the performance of the synthetic sets in the latter sections.

	Overall Accuracy (%)	
FPV	Small Dataset	Large Dataset
1	43.5 ($\pm$0.2)	68.6 ($\pm$0.3)
20	55.0 ($\pm$0.6)	84.2 ($\pm$0.3)
50	56.1 ($\pm$0.7)	85.3 ($\pm$0.4)

Table 5.1: Baseline model accuracy for small (20% entire dataset) and large (75% entire dataset) training set with 1, 20 and 50 frames extracted from each video. FPV: frames per video.

Table 5.1 shows the overall accuracy across all classes under different settings. These accuracy values allow us to make quick comparisons between different models, which can be useful in identifying the experiment direction. Specifically, we use the overall

accuracy of fpv (frames per video) = 1 for small and large training sets, which are approximately 44% and 72% respectively, as an essential benchmark to evaluate the synthetic set performance later in Section 5.

Figure 5.1 shows the expected accuracy (performance) range for synthetic sets generated from different sizes of datasets, namely 20% and 75% of the entire UCF101 dataset. For both dataset sizes, there is a significant accuracy difference of approximately 10% in between 1 fpv and 50 fpv settings, while the difference between 20 fpv and 50 fpv is only 1%. This result raises questions about whether the extra 30 frames between the two settings provide much extra information for action recognition, and whether the performance between synthetic sets generated from 20 fpv and 50 fpv settings would have a noticeable difference. In Section 5, we will further investigate this by evaluating the performance of synthetic sets generated from different frame rates and comparing their accuracy to the baseline model. This comparison would provide us insights about the impact of frame number on the performance of the model, and therefore gives us the intuition of the optimization direction.

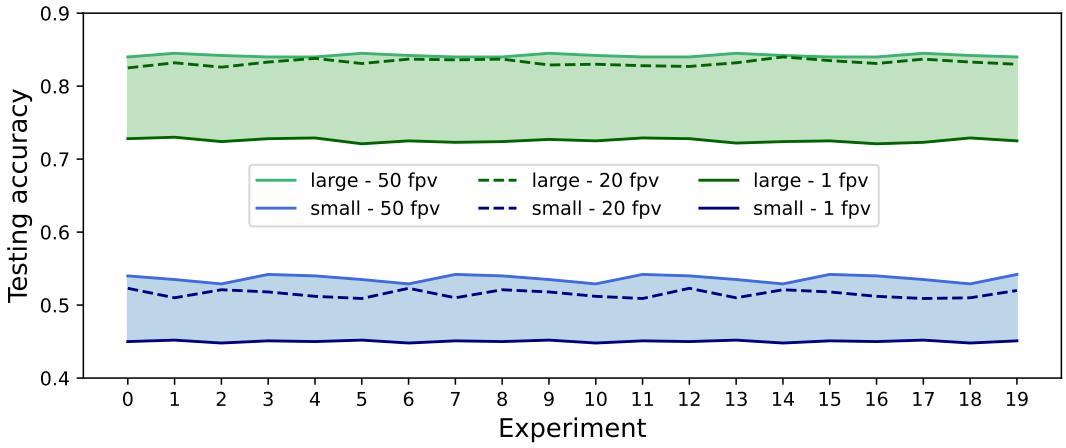


Figure 5.1: The baseline accuracy for different experiment settings.
fpv: number of frames per video. small: training set with 20% entire dataset. large: training set with 75% entire dataset.
The blue and green shadow indicates the expected accuracy range for synthetic sets computed from the small training set (20% of the whole dataset) and the large training set (75% of the whole dataset) respectively. Both of the expected performance ranges are bounded by the performance of the models with 1 frame and 50 frames per video.

In addition to analyzing the overall accuracy, class-wise accuracy is also worth mentioning. As shown in Figure 5.2, for each dataset, we computed the accuracy difference between the 50-fpv and 1-fpv models for each class and plotted their sample distribution for both the small and large datasets. The accuracy difference for most of the classes is larger than 0, indicating that synthetic images have the potential to improve the performance of those classes. We also observed that the accuracy difference between the small and large datasets is quite similar, which is consistent with the previous observation on overall accuracy. The distribution of class-wise accuracy difference is also similar, and this is expected as both datasets are extracted from the same source

and have balanced classes, resulting in comparable sample distributions.

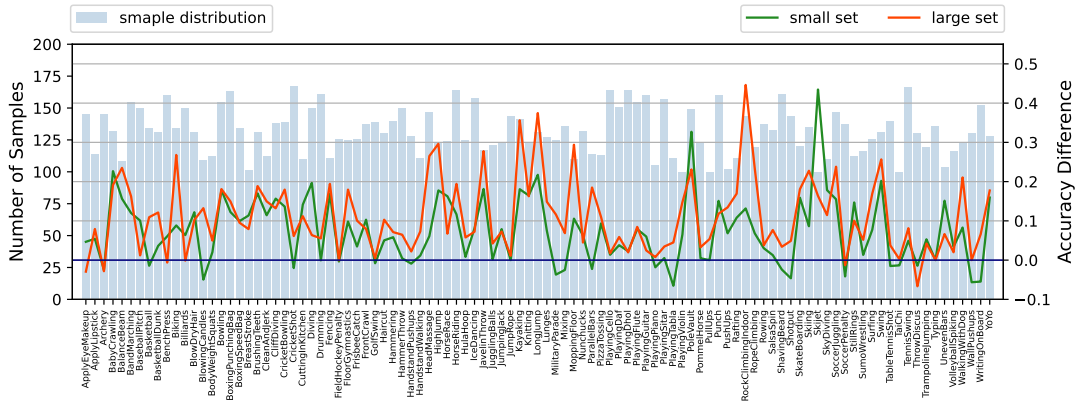


Figure 5.2: The class-wise accuracy difference between 1 frame and 50 frames model for the small and large training set, with 20% and 75% of the entire UCF101 dataset respectively.

5.3 Synthetic Set Generation and Visualisation

5.3.1 Challenges in Video-wise Synthetic Set Generation

In the image data condensation framework, synthetic sets are generated using a class-wise computation approach. This involves updating a set number of synthetic images for each class during each iteration. However, when using video as our condensation boundary instead of class, the simultaneous generation approach can result in computational and memory constraints. For instance, even updating the synthetic images for 20% of the UCF101 dataset would involve generating more than 2600 images at the same time, making the process extremely computationally expensive and time-consuming.

To overcome this challenge, we proposed a batch division and multiprocessing-based sequential generation approach that introduced parallelism into the framework. Specifically, we divided the video dataset into smaller batches and used Python multiprocessing and CPU parallelism to process each batch. This approach reduced the memory requirements for synthetic set generation and accelerated the overall generation process while providing better control of the overall generation flow and data quality.

However, even with the newly proposed synthetic set generation pipeline, the generation process is still very time-consuming due to the computational limitations and the complexity of the overall system. Moreover, due to our experimental design, any changes to the experiment parameters require fully computing a new synthetic set, which involves generating tens of thousands of images from thousands of videos, and training and evaluating models on them. This limitation restricts the number of experiments that can be conducted within a given time, and thus the different settings we could test and analyse. Nonetheless, we managed to compare and analyze the model performance under different settings by performing various experiments, which

gives us valuable insights into the relationship between certain settings and the final performance.

5.3.2 Pre-trained Models for Synthetic Set Generation

As mentioned in Section 3.2 about system workflow, there are two options for the distribution matching procedure when computing synthetic images: using randomly initialized models or using pre-trained models. The pre-trained models are expected to better reflect and extract the feature distribution of the training samples than randomly initialised models, with greater emphasis on active parts of the video such as human motion. Therefore, synthetic sets generated using pre-trained models are expected to outperform those generated using randomly initialized models.

Figure 5.3 shows the differences between synthetic images generated using randomly initialized models and those generated using pre-trained models. As shown in the figure, the images generated using randomly initialized models tend to compress motion paths into a single image with clear object boundaries On the other hand, the images generated using pre-trained models exhibit less predictable changes, such as added blurs and increased contrast.

n Section 5.4, we will compare the performance of synthetic sets computed using randomly initialised models, 50 pre-trained models, and 200 pre-trained models, respectively. This analysis will provide a more quantifiable reflection on how these different network distributions influence the synthetic set's performance. When analysing those results, we will also refer to this section, especially Figure 5.3, to further discuss the potential causes of certain performances.

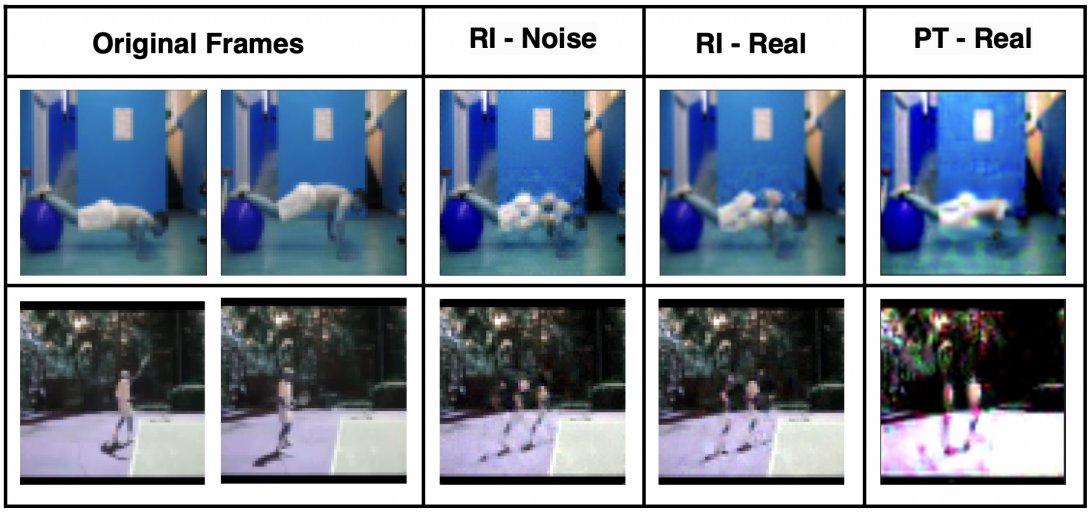


Figure 5.3: Comparison between original frames and synthetic images computed with different settings.
RI: distribution matching using randomly initialised models. PT: distribution matching using pre-trained models. Noise: synthetic images initialised with random noise. Real: synthetic images initialised with the real frames.

5.3.3 Synthetic Image Initialisation

The initialisation of synthetic images is the initial step in the generation process, and there are two options available: initialise with random real frames or initialise with random noise.

Figure 5.3 shows that the synthetic images created using these two options are quite similar in terms of visual appearance, as illustrated in the ”RI-Real” and ”RI-Noise” columns. Further experiments have shown that their performance is also quite comparable with a difference in overall accuracy within $\pm$ 0.2%. Hence, for simplicity and clarity, we will use real images for initialisation in the subsequent experiments.

5.3.4 Difference within Synthetic Set

The synthesized images generated from real images with a randomly initialized model are shown in Figure 5.4. Based on their visual characteristics, the output images can be roughly categorized into three groups:

- **Coherent images:** These synthetic images are human understandable. The main movement and scene changes are effectively compressed into a single synthetic image.
- **Chaotic images:** These synthetic images have a bumpy grid pattern and are difficult to recognize due to their chaotic appearance.
- **Real-like images:** These synthetic images are visually similar to real frames but do not capture the movement very well.

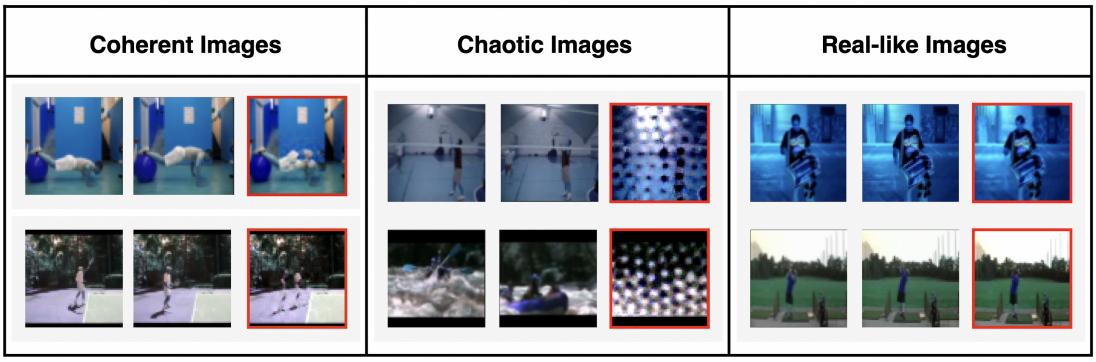


Figure 5.4: Examples of three types of synthetic images: Coherent images, chaotic images and real-like images. The images in red boxes are synthetic images generated and others are real frames that the synthetic images are generated from.

It is also worth mentioning that although we only presented the visualization results for a specific experiment setup, very similar patterns can also be found in the results with other different setups including random noise initialised images and pre-trained model trained images. Therefore when analysing the patterns in the following paragraphs, we refer to the results computed from all the experimental settings generally.

One possible reason for the observed division of output images and the unpredictability of results could be the inherent randomness in the feature embedding process. Both the

use of randomly initialized models and pre-trained models for distribution matching involves a degree of randomness in capturing partial features from the original images using parametric functions. Coherent images are samples in which the key features are successfully captured, while chaotic and real-like images are not. Additionally, since the project deals with low-resolution images and some actions may have limited movement range, the actual pixel-wise changes might be trivial even for the original training images. Therefore, the synthetic images might also be very similar to the real images.

Based on the visualizations in Figure 5.4, it can be inferred that coherent and real-like images are the expected output of the synthetic image generation process, while the chaotic images introduce unexpected behaviour and may cause potential problems during the experiment process. Without the chaotic images, the synthetic images are expected to perform at least as well as, if not better than, the baseline model (i.e., one-real-frame-per-video model). However, with the addition of chaotic images, there is no guarantee that the synthetic images will consistently perform at this level, as they may negatively impact the overall accuracy. This hypothesis is validated by the experimental results presented in the next section.

5.4 Test performance

In this section we present a cross-comparison of the performance of synthetic sets generated under various settings in terms of both test accuracy improvement, and training cost reduction.

5.4.1 Performance Improvement

To evaluate the performance of our proposed method, we conducted experiments by testing the synthetic sets on an unseen test set and comparing their overall accuracy with baseline models under the same testing setting. Both the synthetic sets and the baseline training set consist of one image for each video for all the videos in the training dataset, with the one frame being the middle real frame for the baseline set and being a synthetic image generated from the whole video for the baseline set. To examine the impact of different experimental settings in the synthetic set generation process, we synthesized sets using 20 and 50 frames per video (FPV) and matched the data distribution using different numbers of pre-trained models or no pre-trained models. Specifically, we computed synthetic sets using randomly initialized models, 50 pre-trained models, and 200 pre-trained models.

We present the experimental results in Table 5.2, including the baseline models and synthetic sets' accuracies, the improvement ratio of using our method, and the normalized accuracy gain, which measures the performance improvement of our method (definition see Section 3.3).

Hyper-parameters During the synthetic generation process, we need to tune only one hyper-parameter, which is the learning rate used for optimizing the synthetic images.

Dataset Size	FPV (Ratio)	PM	Accuracy	Baseline	Whole Dataset	Normalized Gain
Small (2697 samples)	20 (0.05)	0	**47.2**	43.5	55.0	**30.8**
		50	45.6			17.5
		200	45.2			14.2
Small (2697 samples)	50 (0.02)	0	46.8	43.5	56.2	26.0
		50	46.3			22.0
		200	**46.9**			**26.8**
Large (9990 samples)	50 (0.02)	0	68.9	68.6	85.3	1.8
		50	70.0			8.4
		200	**70.1**			**9.0**

Table 5.2: The average overall accuracy of synthetic sets, and the normalized gain compared to the baseline model.
(FPV(Ratio): number of frames per video that are used to compute one synthetic image with synthesizing ratio, PM: Number of pre-trained models used to compute the synthetic set, PM=0: Randomly initialised models are used instead of pre-trained models.)

In our experiments, we use a fixed learning rate of 1 for all the experiment settings. Additionally, we use a fixed number of 400 iterations to update the synthetic images for all the settings. This is because, with a compression ratio of 0.02 and 0.05 and low-resolution frames, 400 iterations are sufficient to minimize the MMD loss with no significant loss drop after 200 iterations.

All the synthetic images are initialized with a randomly selected real frame. For training on the synthetic images and baseline models, we use a fixed learning rate of 1, batch size of 16, and train for 20 epochs. These hyperparameters were chosen through experimentation based on the validation set performance and were found to provide satisfactory results across all experimental settings.

Overall accuracy and dataset size Table 5.2 shows that our method outperforms the baseline model under all synthesizing settings, with different levels of performance improvement. The largest gain is achieved on the smaller dataset with 20 and 50 frames per video, where our method achieves approximately 30% improvement over the maximum possible accuracy gain on both fpv settings. However, when it comes to large datasets, the performance improvement is not as substantial as that on smaller ones, with around 10% accuracy gain on the best-performing settings. One possible reason could be that only one synthetic image per video may be insufficient to represent the diversity and complexity of larger datasets, and therefore limits their model performance.

Pre-trained models v.s. randomly initialised models To further analyse and compare the influence of different network parameter distributions on the performance of synthetic images, we conducted experiments using both randomly initialized models and different numbers of pre-trained models for synthetic set generation. The pre-trained models were trained on the entire original training set, and they were expected to better extract frame features and improve the distribution matching efficiency. However, to

our surprise, different trends were observed in the testing performance in Table 5.2. Specifically, for the smaller dataset, the model performs the best when the randomly initialised models instead of the pre-trained models are used. By contrast, when it comes to the larger dataset, the situation is quite the opposite where the using of pre-trained models leads to an increase in the performance.

The inconsistent performance of per-trained models under different settings could be caused by several potential factors. One potential reason could be that the quality of the pre-trained networks could influence the data distribution sampled from the original data, and result in feature leakage. While Zhao & Bilen (2022) [66] mentions that the different network distributions with different levels of validation accuracy produce similar synthetic set performance with small variance under their experiment setting, their synthetic sets are generated from the CIFAR10 dataset with class-wise synthesis boundary. In our project, we use the UCF101 video dataset with low-resolution frames, where the difference in data distribution across frames within a video is small and harder to capture. Any missed features could significantly reduce overall performance. For instance, the pre-trained models may fail to capture certain movements that could be easily captured by randomly initialised models, as shown in the first row of Figure 5.3 where the ”push-up” movement is not captured by the pre-trained models (PT-real column) but is captured with clear movement boundaries by the randomly initialised models (RI-Noise and RI-Real columns).

Another possible reason could be the randomness of the feature sampling process. As shown in Figure 5.4 the generated synthetic images can be classified into three types, with a certain level of unpredictability. While increasing the proportion of coherent images is expected to improve the overall model performance, such an increase is hard to be guaranteed.

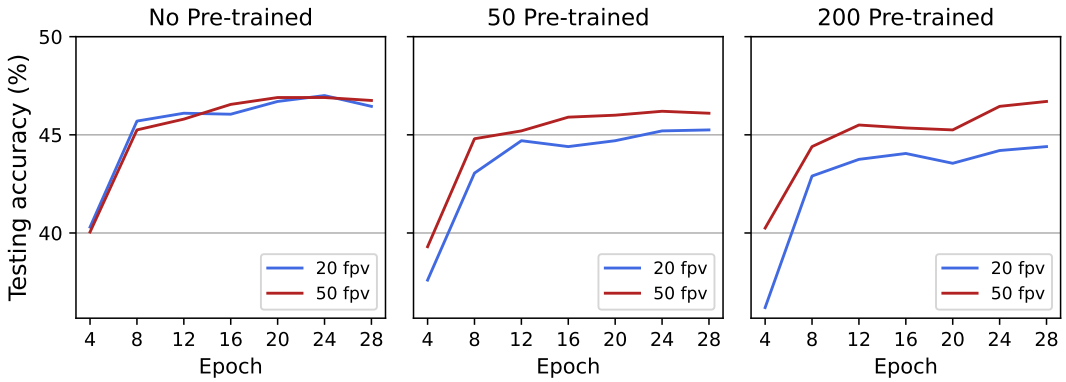


Figure 5.5: Comparison between the performance of synthetic images generated from 20 and 50 original frames under different settings (number of pre-trained models used) on the small UCF101 dataset.

Number of frames per video In Figure 5.5, we compare the performance of synthetic sets computed from different numbers of frames under the same setting on the small dataset, which corresponds to the first two rows in Table 5.2. Intuitively, using more frames to generate synthetic images is expected to be better performance since more

information is synthesised into one image, but we can see that it is not always true under all experiment settings in Figure 5.5.

We can see from this figure that, more frames improve the performance of pre-trained models, while contributing little when randomly initialised models are used. The reason could be that since our frames are uniformly extracted from the original video, the data distribution of the whole video is preserved and can be well extracted from 20 frames when the randomly initialised models are used. However, when pre-trained models are used and the data distribution is not fully extracted with only 20 frames, as in (b) and (c) in Figure 5.5, the extra 30 frames could be a supplement and provide more training resources during the data synthesizing process, and therefore improve the final performance. This can also be seen from the similar performance of 50 FPV models across different pre-trained model settings, which indicates that more frames can help the model extract key information from a given video to the greatest extent. Thus, the optimal number of frames used to synthesize one image should be chosen based on the highest level of performance while still maintaining an efficient synthesizing process, as a larger number of frames does not necessarily lead to better performance.

5.4.2 Training Cost

As described in Section 3.2, our overall train-evaluation system mainly consists of two stages: synthetic set generation, and training and testing on the generated synthetic set. By synthesizing the original training data, we can reduce the size of the training set, and therefore significantly reduce the storage requirement and training time during the second training stage.

Table 5.3 compares the performance and training costs of the synthetic set, the baseline training set (which consists of the middle frame of each video), and the entire dataset with different numbers of frames extracted from each video. We compare them based on three parameters: test performance, training time, and storage requirement.

Here's a detailed description of each parameter:

1. Performance: We measure the average test accuracy of the synthetic set generated under the best-performing experimental setting and compare it to the baseline performance and the performance of the whole dataset with the same testing setting and the same unseen testing set.
2. Training time: We measure the time required to train the model and achieve the above performance on NVIDIA GTX 1060 GPU.
3. Storage requirement: We measure the storage requirement for each training set in terms of the number of images that need to be stored.

It can be seen from the table that while achieving an obvious performance improvement with all the synthetic set settings, the training cost in terms of training time and storage required is significantly compared to the whole dataset. For instance, the synthetic set generated with 20 frames achieves a test accuracy of 47.3%, which is an 86% relative performance compared to the whole dataset, while only requiring 5% of the original training time and storage requirement. Similarly, the synthetic set generated

Training Set		Performance (%)	Time cost (min)	Storage (imgs)
Small dataset (with 20 fpv)	Synthetic Set	**47.3**	**2.1**	**2397**
	Baseline	43.4	**2.3**	**2397**
	Whole dataset	55.0	41.9	47940
Small dataset (with 50 fpv)	Synthetic Set	**47.0**	**2.2**	**2397**
	Baseline	43.4	**2.1**	**2397**
	Whole dataset	56.9	96.6	119850
Large dataset (with 50 fpv)	Synthetic Set	**70.1**	**7.8**	**9990**
	Baseline	68.4	**8.1**	**9990**
	Whole dataset	85.6	337.1	499500

Table 5.3: Test performance with action classification task. The synthetic set performance is compared with baseline dataset and the whole dataset in terms of performance, time cost and storage required.

with 50 frames achieves a test accuracy of 70.1%, which is an 82% relative performance compared to the whole dataset, while only requiring 2% of the original training time and storage requirement. This shows the effectiveness of our synthetic set generation method in reducing the computational and storage costs in the training process while still achieving comparable performance on the testing set.

Chapter 6

Conclusion

6.1 Summary

In this project, we propose a novel video data synthesizing method particularly targeting classification tasks. The training data are synthesized using a distribution matching approach which aims to minimize the maximum mean discrepancy (MMD) between the original training set and the synthetic set. The framework proposed and implemented in this project creatively uses a video-wise synthetic image generation instead of class-wise generation, which could better preserve the original sample distribution and data diversity.

After carefully analysing and performing basic tests on different datasets, UCF101 is chosen as the primary test dataset, and a series of pre-processing techniques are performed. We computed synthetic sets under a range of different experimental settings including different training set sizes, different numbers of frames extracted per video, different numbers of pre-trained models used for distribution matching and different synthetic image initialisation options, and analysed the generated synthetic images in detail. The synthetic sets are then evaluated on the same unseen test set, and their performance is evaluated in terms of the test accuracy and training cost. We showed that our method outperforms the baseline model under all experiment settings and has a maximum of 30.8% normalized accuracy gain than the baseline model, with only 2% training time and storage requirement of the original training set. Although the accuracy gain on larger datasets is not as significant as the smaller ones, our method is still proven to be effective given the performance improvement and significant training cost reduction.

6.2 Limitation and future work

Synthetic set size While our method shows promising results with one synthetic image per video, we recognize that the informative value of the synthesized images can vary due to reasons like the randomness in the feature embedding and distribution sampling process. Therefore, in the future, we would like to include synthesizing

multiple images per video and compare their performance against the baseline models with multiple real frames. This could provide more insights into the optimal synthetic set size for different datasets and further improve the efficiency of our method especially on large datasets.

Dataset UCF101 is used as the primary testing dataset due to its balanced sample distribution and a large number of classes in this project, as discussed in Section 4.4. In the future, additional datasets with diverse characteristics, such as surveillance datasets like [42] or datasets with less balanced sample distribution, could be used to further evaluate the effectiveness and generalizability of our method.

Synthetic set generation setting We observed that the synthetic set generation setting has varying impacts on synthetic set performance, which are dependent on dataset settings such as dataset size. In the future, it would be valuable to further investigate how the pre-trained models and their quality influence synthetic set performance and explore how the optimal synthetic set generation setting varies with changes in dataset size and video quality.

Video resolution In this project, we primarily focus on low-resolution videos for better computational efficiency and generalizability. However, since higher-resolution data is expected to have better performance, it would be informative to evaluate the potential of our method on higher-resolution videos as well. Additionally, higher-resolution data requires more computational resources for training, which makes our methods particularly useful for reducing training costs.

Efficiency comparison In the future, we plan to conduct experiments to compare the efficiency of our synthetic set generation process with other video compression methods or core-set selection-based video set size reduction methods. This would help further demonstrate the efficiency of our method and its potential benefits in reducing training costs.

Bibliography

[1] More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute - Tubefilter.

[2] T Akiba, S Sano, T Yanase, T Ohta Proceedings of the 25th ..., and undefined 2019. Optuna: A next-generation hyperparameter optimization framework. *dl.acm.org*, pages 2623–2631, 7 2019.

[3] Imran Alam, Devesh Jalan, Priti Shaw, and Partha Pratim Mohanta. Motion Based Video Skimming. *2020 IEEE Calcutta Conference, CALCON 2020 - Proceedings*, pages 407–411, 2 2020.

[4] Rahaf Aljundi, Min Lin Mila, Baptiste Goujaud Mila, and Yoshua Bengio Mila. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for Vector-Valued Functions: a Review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 6 2011.

[6] Muhammad Asim, Noor Almaadeed, Somaya Al-Maadeed, Ahmed Bouridane, and Azeddine Beghdadi. A Key Frame Based Video Summarization using Color Features. *2018 Colour and Visual Computing Symposium, CVCS 2018*, 10 2018.

[7] Eden Belouadah and Adrian Popescu. ScaIL: Classifier Weights Scaling for Class Incremental Learning. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 1255–1264, 1 2020.

[8] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-End Incremental Learning, 2018.

[9] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via Proxy: Efficient Data Selection for Deep Learning. 6 2019.

[10] Ugur Demir, Yogesh S. Rawat, and Mubarak Shah. TinyVirat: Low-resolution video action recognition. *Proceedings - International Conference on Pattern Recognition*, pages 7387–7394, 2020.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. pages 248–255, 3 2010.

[12] Melanie Ducoffe and Frederic Precioso. Adversarial Active Learning for Deep Networks: a Margin Based Approach. 2 2018.

[13] Helenca Duxans, Xavier Anguera, and David Conejero. Audio based soccer game summarization. *2009 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB 2009*, 2009.

[14] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via Maximum Mean Discrepancy optimization. *Uncertainty in Artificial Intelligence - Proceedings of the 31st Conference, UAI 2015*, pages 258–267, 5 2015.

[15] Omar Elharrouss, Noor Al-Maadeed, and Somaya Al-Maadeed. Video summarization based on motion detection for surveillance systems. *2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019*, pages 366–371, 6 2019.

[16] S Falkner, A Klein, F Hutter International Conference on, and undefined 2018. BOHB: Robust and efficient hyperparameter optimization at scale. *proceedings.mlr.press*, 2018.

[17] M Feurer, F Hutter Automated machine learning: Methods, and undefined 2019. Hyperparameter optimization. *library.oapen.org*.

[18] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Alexander Smola, Bernhard Schölkopf, and Alexander Smola GRETTON. A Kernel Two-Sample Test Bernhard Schölkopf. *Journal of Machine Learning Research*, 13:723–773, 2012.

[19] Cheng Huang and Hongmei Wang. A Novel Key-Frames Selection Framework for Comprehensive Video Summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):577–589, 2 2020.

[20] Tanveer Hussain, Khan Muhammad, Weiping Ding, Jaime Lloret, Sung Wook Baik, and Victor Hugo C. de Albuquerque. A comprehensive survey of multi-view video summarization. *Pattern Recognition*, 109, 1 2021.

[21] Shruti Jadon and Mahmood Jasim. Unsupervised video summarization framework using keyframe extraction and video skimming. *2020 IEEE 5th International Conference on Computing Communication and Automation, ICCCA 2020*, pages 140–145, 10 2020.

[22] Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. pages 2372–2379, 3 2010.

[23] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, Matei Zaharia, and Stanford Infolab. NoScope: Optimizing Neural Network Queries over Video at Scale. 2150.

[24] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks.

[25] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics Human Action Video Dataset. 5 2017.

[26] Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. RETRIEVE: Coreset Selection for Efficient and Robust Semi-Supervised Learning. *Advances in Neural Information Processing Systems*, 18:14488–14501, 6 2021.

[27] Yeachan Kim and Bonggun Shin. In Defense of Core-set: A Density-aware Core-set Selection for Active Learning. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 804–812, 8 2022.

[28] Yassin Kortli, Maher Jridi, Ayman Al Falou, and Mohamed Atri. Face Recognition Systems: A Survey. *Sensors 2020, Vol. 20, Page 342*, 20(2):342, 1 2020.

[29] SB Kotsiantis, I Zaharakis, P Pintelas intelligence applications in . . . , and undefined 2007. Supervised machine learning: A review of classification techniques. *books.google.com*, 31:249–268, 2007.

[30] Didier J. LeGall. MPEG (moving pictures expert group) video compression algorithm: a review. *https://doi.org/10.1117/12.45402*, 1452:444–457, 6 1991.

[31] Shiye Lei, Fengxiang He, Yancheng Yuan, and Dacheng Tao. Understanding deep learning via decision boundary. 6 2022.

[32] Jiatong Li, Ting Yao, Qiang Ling, and Tao Mei. Detecting shot boundary with sparse coding for video summarization. *Neurocomputing*, 266:66–78, 11 2017.

[33] Xuelong Li, Zhigang Wang, and Xiaoqiang Lu. Surveillance video synopsis via scaling down objects. *IEEE Transactions on Image Processing*, 25(2):740–755, 2 2016.

[34] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning Linear Transformations for Fast Image and Video Style Transfer, 2019.

[35] Weiyao Lin, Yihao Zhang, Jiwen Lu, Bing Zhou, Jinjun Wang, and Yu Zhou. Summarizing surveillance videos with local-patch-learning-based abnormality detection, blob sequence optimization, and type-based synopsis. *Neurocomputing*, 155:84–98, 5 2015.

[36] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and Video Compression with Neural Networks: A Review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 6 2020.

[37] Yu Fei Ma and Hong Jiang Zhang. A model of motion attention for video skimming. *IEEE International Conference on Image Processing*, 1, 2002.

[38] Ansuman Mahapatra, Pankaj K. Sa, and Banshidhar Majhi. A multi-view video synopsis framework. *Proceedings - International Conference on Image Processing, ICIP*, 2015-December:1260–1264, 12 2015.

[39] Ansuman Mahapatra, Pankaj K. Sa, Banshidhar Majhi, and Sudarshan Padhy. MVS: A multi-view video synopsis framework. *Signal Processing: Image Communication*, 42:31–44, 3 2016.

[40] Carmen Morawetz, Michael C. Riedel, Taylor Salo, Stella Berboth, Simon B. Eickhoff, Angela R. Laird, and Nils Kohn. Multiple large-scale neural networks underlying emotion regulation. *Neuroscience & Biobehavioral Reviews*, 116:382–395, 9 2020.

[41] Marcos Vinicius Mussel Cirne and Helio Pedrini. VISCOM: A robust video summarization approach using color co-occurrence matrices. *Multimedia Tools and Applications*, 77(1):857–875, 1 2018.

[42] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J K Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai. A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video.

[43] H Pham, M Guan, B Zoph, Q Le ... conference on machine ..., and undefined 2018. Efficient neural architecture search via parameters sharing. *proceedings.mlr.press*.

[44] Sreenivasan Ramasamy Ramamurthy, Nirmalya Roy, and Correspondence Sreenivasan Ramasamy Ramamurthy. Recent trends in machine learning for human activity recognition—A survey. *Wiley Online Library*, 8(4), 7 2018.

[45] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a Long Video Short: Dynamic Video Synopsis *. 2006.

[46] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 8 2017.

[47] Lifeng Shang, Linjun Yang, Fei Wang, Kwok Ping Chan, and Xian Sheng Hua. Real-time large scale near-duplicate web video retrieval. *MM'10 - Proceedings of the ACM Multimedia 2010 International Conference*, pages 531–540, 2010.

[48] Thomas Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):19–31, 1997.

[49] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. 2012.

[50] Evangelos Stromatias, Francesco Galluppi, Cameron Patterson, and Steve Furber. Power analysis of large-scale, real-time neural networks on SpiNNaker. *Proceedings of the International Joint Conference on Neural Networks*, 2013.

[51] Ningli Tang, Fang Dai, and Wenyan Guo. Video summary generation based on density peaks clustering with temporal characteristics. *Proceedings of the 16th*

IEEE Conference on Industrial Electronics and Applications, ICIEA 2021, pages 1421–1425, 8 2021.

[52] Praveen Tirupattur, Aayush J Rana, Tushar Sangam, Shruti Vyas, Yogesh S Rawat, and Mubarak Shah. TinyAction Challenge: Recognizing Real-world Low-resolution Activities in Videos. 7 2021.

[53] George Toderici, Google Research, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full Resolution Image Compression With Recurrent Neural Networks, 2017.

[54] Ilya O. Tolstikhin, Bharath K. Sriperumbudur, and Bernhard Schölkopf. Minimax Estimation of Maximum Mean Discrepancy with Radial Kernels. *Advances in Neural Information Processing Systems*, 29, 2016.

[55] Mariya Toneva, Adam Trischler, Alessandro Sordoni, Yoshua Bengio, Remi Tachet Des Combes, and Geoffrey J. Gordon. An Empirical Study of Example Forgetting during Deep Neural Network Learning. *7th International Conference on Learning Representations, ICLR 2019*, 12 2018.

[56] G. Tzanetakis and F. Cook. Sound analysis using MPEG compressed audio. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2:761–764, 2000.

[57] Víctor Valdés and José M Martínez. On-line Video Skimming Based on Histogram Similarity. *Proceedings of the international workshop on TRECVID video summarization - TVS '07*, 2007.

[58] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset Distillation. 11 2018.

[59] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-TAD: Sub-Graph Localization for Temporal Action Detection https://www.deepgcns.org/app/g-tad.

[60] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, Alexander G Hauptmann, Y Yang, · X Chang, Z Ma, · A G Hauptmann, and F Nie. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization.

[61] Tong Yao, Maosen Xiao, Caiwen Ma, Chao Shen, and Peng Li. Object based video synopsis. *Proceedings - 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications, WARTIA 2014*, pages 1138–1141, 12 2014.

[62] Ghazaala Yasmin, Sujit Chowdhury, Janmenjoy Nayak, Priyanka Das, and Asit Kumar Das. Key moment extraction for designing an agglomerative clustering algorithm-based video summarization framework. *Neural Computing and Applications*, pages 1–22, 6 2021.

[63] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset Distillation: A Comprehensive Review. 1 2023.

[64] Longfei Zhang, Yuanda Cao, Gangyi Ding, and Yong Wang. A computable visual attention model for video skimming. *Proceedings - 10th IEEE International Symposium on Multimedia, ISM 2008*, pages 667–672, 2008.

[65] Yunfan Zhang, Ties Van Rozendaal, Johann Brehmer, Markus Nagel, Taco S Cohen Qualcomm, and A I Research. Implicit Neural Video Compression. 12 2021.

[66] Bo Zhao and Hakan Bilen. Dataset Condensation With Distribution Matching, 2023.

[67] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. DATASET CONDENSATION WITH GRADIENT MATCHING.

[68] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[69] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, Jieping Ye, and Senior Member. Object Detection in 20 Years: A Survey. 5 2019.