UNIVERSITÀ DEGLI STUDI FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Ingegneria Informatica

Incremental Learning of Stationary Representations

Candidate
Matteo Bruni

Supervisors
Prof. Alberto Del Bimbo

PhD Federico Pernici

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXIII, 2017-2020
Settore Scientifico Disciplinare ING-INF/05

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

Alla mia famiglia

Abstract

Humans and animals, during their life, continuously acquire new knowledge over time while making new experiences. They learn new concepts without forgetting what already learned, they typically use a few training examples (i.e. a child could recognize a giraffe after seeing a single picture) and they are able to discern what is known from what is unknown (i.e. unknown faces). In contrast, current supervised learning systems, work under the assumption that all data is known and available during learning, training is performed offline and a test dataset is typically required. What is missing in current research is a way to bridge the human learning capabilities in an artificial learning system where learning is performed incrementally from a data stream of infinite length (i.e. lifelong learning). This is a challenging task that is not sufficiently studied in the literature.

According to this, in this thesis, we investigated different aspects of Deep Neural Network models (DNNs) to obtain stationary representations. Similar to fixed representations these representations can remain compatible between learning steps and are therefore well suited for incremental learning.

Specifically, in the first part of the thesis, we propose a memory-based approach that collects and preserves all the past visual information observed so far, building a comprehensive and cumulative representation. We exploit a pre-trained fixed representation for the task of learning the appearance of face identities from unconstrained video streams leveraging temporal-coherence as a form of self-supervision. In this task, the representation allows us to learn from a few images and to detect unknown subjects similar to how humans learn.

As the proposed approach makes use of a pre-trained fixed representation, learning is somewhat limited. This is due to the fact that the features stored in the memory bank remain fixed (i.e. they are not undergoing learning) and only the memory bank is learned. To address this issue, in the second part of the thesis, we propose a representation learning approach that can be exploited to learn both the feature and the memory without considering their joint learning (computationally prohibitive). The intuition is that every time the internal feature representation changes the memory bank must be relearned from scratch. The proposed method can mitigate the need of feature relearning by keeping the compatibility of features between learning steps thanks to feature stationarity. We show that the stationarity of the internal representation can be achieved with a fixed classifier by setting the classifier weights according to values taken from the coordinate vertices of the regular polytopes in high dimensional space.

In the last part of the thesis, we apply the previously stationary representation method in the task of class incremental learning. We show that the method is as effective as the standard approaches while exhibiting novel stationarity properties of the internal feature representation that are otherwise non-existent. The approach exploits future unseen classes as negative examples and learns features that do not change their geometric configuration as novel classes are incorporated in the learning model. We show that a large number of classes can be learned with no loss of accuracy allowing the method to meet the underlying assumption of incremental lifelong learning.

Keywords : *machine learning, deep learning, incremental learning, class-incremental learning, continual learning, computer vision, visual object tracking, few-shot learning, metric learning, regular polytopes, feature stationarity.*

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.

International Journals

1. Federico Pernici, **Matteo Bruni** and Alberto Del Bimbo. Self-supervised on-line cumulative learning from video streams In *Computer Vision and Image Understanding (CVIU).* Association for Computing Machinery, Inc, 2020, flore id 2158/1195643.
2. Federico Pernici, **Matteo Bruni**, Claudio Baecchi, and Del Bimbo Alberto. Regular Polytope Networks. In *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2021.3056762.

International Conferences and Workshops

1. Pernici Federico, Bartoli Federico, **Bruni Matteo**, and Alberto Del Bimbo. Memory based online learning of deep representations from video streams. In *CVPR 2018*, pages 1–8. IEEE, 2018, flore id 2158/1118548.
2. Francesco Turchini*, **Matteo Bruni***, Claudio Baecchi, Tiberio Uricchio, and Alberto Del Bimbo. Open set recognition for unique person counting via virtual gates. In *Image Analysis and Processing – ICIAP 2019 Proceedings, Part I*, volume 11751. Springer International Publishing, 2019, flore id 2158/1171646.
3. A. del Bimbo, F. Pernici, **M. Bruni**, and F. Bartoli. Identity recognition by incremental learning. In *Multimedia and Network Information Systems*, volume 833. Springer International Publishing, 2019, flore id 2158/1171650.
4. Federico Pernici, **Matteo Bruni**, Claudio Baecchi, and Del Bimbo Alberto. Maximally compact and separated features with regular polytope networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 46–53. IEEE, 2019, flore id 2158/1171654.

5. Federico Pernici, **Matteo Bruni**, Claudio Baecchi, Francesco Turchini, Alberto Del Bimbo. Class-incremental Learning with Pre-allocated Fixed Classifiers. In *25th International Conference on Pattern Recognition*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural intelligent systems learn incrementally. Animals and humans refine the skills and attention they devote to problems based on the experiences they accumulate throughout their lives. A fundamental ability of these systems is to learn new concepts without forgetting the previous ones, continually updating and refining what they already know.

Most supervised learning systems work under the assumption that all object classes are known in advance and all the training data can be accessed at the same time in arbitrary order. In contrast to natural intelligent systems, when trying to integrate novel data, they suffer from a problem known as *catastrophic forgetting*: the tendency of neural networks to completely and abruptly forget previously learned information [5–7].

In a real-world scenario collecting a sufficiently large number of representative examples before starting the learning process may be difficult due to time or memory constraints. In a non-stationary environment, the statistical properties of the target concept may change over time in unforeseen ways (*concept drift*), requiring the system to self-adapt. As new examples become available, retraining from scratch might address catastrophic forgetting but it might be prohibitively expensive due to computational cost and training data storage requirements.

An artificial intelligent system should be able to adapt continuously in a dynamic environment where situations can significantly change. Continual Learning [8, 9] specifically addresses this problem while also retaining previously learned knowledge, bringing machine learning closer to natural learning. This problem is also related to the plasticity/stability dilemma [10, 11] where too much plasticity will result in previously encoded data being constantly forgotten while too much stability leads to an inability to adapt to novel information.

The settings and applications for incremental learning can vary enormously, depending on training and evaluation protocols. In a Multi-Task scenario, a single model is required to learn incrementally a number of non-overlapping, isolated tasks without forgetting how to solve the previous ones. A typical neural network for this scenario has a multi-headed output layer where each head is reserved for a particular task. At inference time the task head is selected by an oracle in order to setup the right classification layer [12–15]. Thus, the requirement of an oracle, fail to capture real-world incremental learning problems with unknown task labels [16, 17].

Conversely in the class-incremental learning (CIL) scenario, the agent considers a sequence of tasks containing a subset of classes from a single classification problem. Differently from Multi-Task, all class labels are represented in the same classification head whose number of output nodes equal to the number of classes encountered from the start of the training. The single-head final layer of a Neural Network is expanded with an output node when a new class arrives, similarly to natural learning.

The way the human brain learns is incremental, new visual information is continuously incorporated. For example, a child playing with toys learns to recognize new objects discriminating them from the ones he has already seen. This learning setting, quite common in natural learning, is closer to the class-incremental learning scenario.

In this thesis, we mainly focus on the class-incremental learning classification problem where the class label is unknown at inference time, making the predictions of the learned models task-agnostic. In the next sections, we review, compare and discuss relevant literature providing further technical details of this dissertation. Finally, in the last section, we explain the contributions made in this thesis.

1.1 Continual Learning

Continual learning and the problem of alleviating catastrophic forgetting has been has been extensively studied in literature [8, 9, 14]. These studies can be broadly categorized in three main categories: (1) regularization, (2) dynamic architecture methods, and (3) episodic memory-based (also termed

Experience Replay or rehearsal strategies).

Regularization-based approaches to catastrophic forgetting add a regularization term to the loss function restricting updates to the neural network parameters that were important for previous tasks. Learning Without Forgetting (LWF) [18] is a regularization strategy attempting to preserve the model accuracy on old tasks by imposing the predictions of the previous tasks to be stable after adding data of the new task by using a form of knowledge distillation [19]. Elastic Weighted Consolidation (EWC) [12] imposes constraints on network parameters to reproduce biological mechanism of consolidation using a quadratic penalty on the difference between the parameters for the new and the old task. Online-EWC [20] optimizes EWC approach for multiple tasks, overcoming the complexity of original EWC which scales linearly with the number of tasks. Synaptic Intelligence (SI) also replicates biological mechanism of synapses, preventing parameters (synapses) to change based on the relevance of each parameter for the considered task [14]. Memory Aware Synapses (MAS) tackles the problem in a similar fashion, based on the relevance of each parameter to the task.

Dynamic Architectural approaches make changes to the architecture of the network as new tasks are learned. Changing layers, activation functions, or freezing certain weights in a network are some of the practical approaches. [21] simply freeze certain weights in the network so that they stay exactly the same. [22] grows a network searching for similarities between known classes and unseen classes, organizing them into a hierarchy. Predictions are made by visiting the hierarchy, from the super-classes down to the specific class. [23] exploits boosting algorithm to control network architecture growth balancing its complexity with empirical risk minimization. Progressive Neural Network (PNN) [24] proposes a network structure organized in columns that grows every time a new task is added. Each column is a network that learns a new task, sharing features learned by other columns via lateral connections. Since the column for previous tasks do not change over time, this entirely prevents forgetting on earlier tasks but causes the architectural complexity to grow with the number of tasks.

Experience Replay approaches periodically “replay” past information when training a new task. Part of the previous training data is stored in a “episodic memory” to avoid forgetting what the network has already learned. In contrast to dynamical architectural approaches that increase the network complexity, memory-based methods add a relatively small memory overhead for each new task. The concept of experience replay in class-incremental learning has been introduced in [25] where three properties are defined for an algorithm to qualify as class-incremental learning: (1) it can learn from a data stream where examples of different classes occur at different times; (2) it should provide at any time competitive multi-class classification accuracy for the classes observed so far; (3) the computational requirement and memory footprint should remain bounded. By means of rehearsal technique, new and old data are combined when new tasks are learned in order to prevent catastrophic forgetting of old tasks. [26] presents Gradient Episodic Memory technique which does not store and reuse old samples but allows transfer learning between tasks by storing old gradients and updating them to prevent forgetting. An improved, memory-efficient version of GEM is obtained by considering the average of the losses of all tasks rather than each individual loss of single tasks [27]. Full data rehearsal may prevent catastrophic forgetting, but it is unfeasible due to important memory impact, so [28] implements memory-efficient buffer techniques to perform rehearsal without the need for retaining all samples. [29] aims at finding data distribution that can keep optimal performance level overall tasks. This is achieved by choosing an adequate strategy to build data memory, exploiting the biological mechanic of replaying experiences. [30] introduces a technique to avoid learning interference provoked by data coming from different source domains. Dual-memory incremental learning is exploited in [31] to keep track of statistics of past classes, in order to rebalance their prediction scores in later stages of learning.

1.2 Representation Learning

An intelligent agent to perceive and interact with the environment can rely on different sources of data, which can generally be identified as sensors. The raw data is usually transformed into a better representation for complex tasks since the performance of an intelligent agent is heavily dependent on the choice of data representation. In computer vision, for example, the pixel space is not appropriate for tasks like object classification or object detection since each pixel value does not convey much information with respect to the desired task. In order to exploit complex correlations present inside the original data, an intelligent agent needs to transform it into another space where tasks are easier to solve, a process known as feature learning.

One of the reasons that neural networks have been so successful is that they allow machine learning algorithms to learn representations that are task specific directly from training data without human intervention. The handcrafted feature extraction method, in contrast, relies on using a fixed hand-designed function based on human intuition. While handcrafted feature extraction does not require much training and is less dependent on data it is usually too complex to design manually.

In a convolutional neural network, training a task end to end, provide the final layers with a representation where the training tasks are easier to solve [32,33]. Once the training procedure is finished the feature extraction function is fixed since the network parameters are fixed. A feature representation, often called "embedding", is a compressed information of the input data representing low level or mid-level attributes. A good embedding is expected to cluster input data of the same class in the embedding space while also maximizing the distance of examples of different classes. The representation, learned training a network on millions of labeled examples for a classification task, provides a general-purpose feature representation that can be re-used on other tasks through a transfer learning approaches [34].

In a class-incremental learning scenario, this interesting property can be exploited to learn new classes that the network has never seen before. This process is commonly used in modern applications like face recognition where the number of classes (i.e. person identities) cannot be known at training time [35,36]. After training a network on a large dataset of the task domain, the classifier of the network it is no longer used since the system relies on the feature representation. As images of a new class become available, they are collected in a memory module called gallery set. Given a test input image, (*query*) its class is assigned by identifying the closest cluster in the gallery set since images of the same class are expected to cluster in the embedding space. This process is known as visual search.

This approach only performs well when we assume that each sample either in the training data or testing data is an identically and independently distributed (*i.i.d.*) sample drawn from a *fixed* probability distribution. Unfortunately, this is a major simplification of what can happen in a real-world scenario and what happens in human learning. Ignoring this issue can result in clusters of different classes to overlap in the embedding space when new classes are originated from a different distribution from the one sampled in the training dataset. The impossibility of recognizing classes learned in the past leads to catastrophic forgetting.

To achieve an embedding capable of disentangling features from different distributions, a solution is to fine-tune the network on the data collected. Since the network learning is resumed, its parameters change irreversibly from the old ones defining a new embedding space which is no longer compatible with the previous one. Once a new embedding is re-leaned, all the data collected in the gallery set have to be re-processed to generate their new feature representation and recreate the clusters, a process known as "backfilling" [37].

Having feature compatibility between different embedding spaces is a desirable property. New models can be deployed without having to re-process the previously indexed gallery set. In this dissertation, we propose to obtain a feature representation that is compatible by design achieving feature stationary in an incremental learning scenario.

1.3 Contributions

In this thesis, we mainly focus on the class-incremental learning classification problem where the class label is unknown at inference time. Overall, we are interested in finding a way to bridge the human learning paradigm in an artificial learning system, where learning is performed incrementally on all available information that may increase over time as unknown subjects may eventually appear.

First, we propose an approach that collects and preserves a pre-trained fixed representation of the past visual knowledge observed so far in a memory module, building a comprehensive and cumulative representation. The proposed learning procedure relies solely on the memory bank since the feature embedding is fixed and does not undergo learning. This approach can be leveraged in different application scenarios. In particular we are interested in learning the appearance of face/body identities from unconstrained video-streams, with the goal of identity recognition or counting.

Although updating the feature representation is a desirable property, doing so would invalidate all information contained in memory as the feature space would change continuously making it no longer compatible with the previous one. Jointly learning of feature and memory even if it seems a promising solution is computationally prohibitive. Motivated by this problem we propose a representation learning method that is stationary over time and does not change as learning proceeds. This is achieved by fixing (i.e. set as non-trainable) the weights of the last learnable transformation (i.e. the classifier), using values taken from the coordinate vertices of the three regular polytopes available in R^d.

Finally, we propose an approach, that leverages the stationary features learning method previously introduced, to achieve class-incremental learning via a pre-allocated fixed classifier. An arbitrary number of output nodes is pre-allocated at the beginning of the training and as soon as a new class appears, it is assigned to an unused output node. Since the feature learning method is stationary, the embedding space is compatible with the previous ones as new classes appear.
In summary, we show several contributions in this dissertation:

Self-supervised On-line Cumulative Learning
In Chapter 2, we present a novel online self-supervised method for the task of learning the appearance of face identities from unconstrained video streams. The method exploits fixed pre-trained deep face feature descriptors together with a memory based learning mechanism that takes advantage of the temporal coherence of visual data. Specifically, we introduce a memory based cumulative learning strategy that discards redundant features while time progresses. This allows building a comprehensive and cumulative representation of all the past visual information observed so far. The proposed strategy is shown to be asymptotically stable. We argue this is a critical issue for any system that claims to operate lifelong.

Open set recognition for unique person counting
In Chapter 3 we propose the task of unique counting which is a variation of counting task. We present a complete real-time system which is able to perform detection, tracking and unique counting in the wild with user drawn gates. Experiments on the challenging DukeMTMC dataset [38] are reported showing that our method is able to effectively count people in real time and discern between the few persons which do multiple passages through the gates. The system is able to perform incrementally leveraging a memory module where the appearances of people are stored to discern their identities.

Regular Polytope Networks In Chapter 4 we generalize the concept of fixed classifiers and show they can generate stationary and maximally separated features at training time with no loss of performance and in many cases with slightly improved performance. We performed extensive evaluations across a range of datasets and modern CNN architectures reaching state-of-the-art performance. We observed faster speed of convergence and a significant reduction in model parameters. We further provide a formal characterization of the class decision boundaries according to the dual relationship between regular polytope and statistically verify the validity of our method on random permutations of the labels.

Class-incremental Learning with Pre-allocated Fixed Classifiers

In Chapter 5 we address class-incremental learning using a novel classifier in which a number of pre-allocated output nodes are subject to the classification loss right from the beginning. This allows the output nodes of yet *unseen classes* to firstly see negative samples since the beginning of learning together with the positive samples that incrementally arrive. The output nodes can learn from the beginning of the learning phase, *pre-allocating* a special classifier with a large number of output nodes in which the weights are *fixed* (i.e. not undergoing learning) and set to values taken from the coordinate vertices of regular polytopes as introduced in Chapter 4. We achieve similar results with respect to several important baselines on standard benchmarks.

Finally, in Chapter 6, conclusions of the dissertation and future challenges of incremental learning using stationary representations are discussed.

Chapter 2

Self-supervised On-line Cumulative Learning

We present a novel online self-supervised method for face identity learning from video streams. The method exploits deep face feature descriptors together with a memory based learning mechanism that takes advantage of the temporal coherence of visual data. Specifically, we introduce a discriminative descriptor matching solution based on Reverse Nearest Neighbour and a memory based cumulative learning strategy that discards redundant descriptors while time progresses. This allows building a comprehensive and cumulative representation of all the past visual information observed so far. It is shown that the proposed learning procedure is asymptotically stable and can be effectively used in relevant applications like multiple face identification and tracking from unconstrained video streams. Experimental results show that the proposed method achieves comparable results in the task of multiple face tracking and better performance in face identification with offline approaches exploiting future information. [1] [2]

[1]The part of this chapter has been published as “Memory based online learning of deep representations from video streams” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018 Salt Lake City* [39] and as “Self-supervised on-line cumulative learning from video streams” in *Computer Vision and Image Understanding, Volume 197-198 Year 2020* [40]

[2]*Acknowledgments:* This research was supported by the project Ideha 'Innovazioni per L'elaborazione dei Dati nel Settore del Patrimonio Culturale' (MIUR - PON ARS01-00421) and Leonardo SpA, Italy.

2.1 Introduction

Supervised machine learning is a very successful learning paradigm in which a clear distinction is made between the training phase and the testing phase. Once a model is learned, it is no longer subjected to training and inference on novel unseen data tacitly assumes that the data distribution does not change over time. Once the learning phase is concluded no classes other than those used for learning can be predicted. Although, such hard division between training and testing and the availability of large corpus of annotated data have demonstrated exceptional achievements in learning the appearance of objects from images [41], they remain critical as linear improvements in performance require an exponential number of labeled examples [42]. In addition to this, efforts to collect large quantities of annotated images, such as ImageNet [43] and Microsoft COCO [44] don't have the necessary scalability and are hard to be extended, replicated or improved. These issues may also put a performance limit on models learned in this way.

Drawing inspiration from biological systems, a possible attractive alternative would be incrementally to learn the object appearance from never-ending video streams with no supervision, both exploiting the large quantity of unconstrained videos available in the Internet and the fact that adjacent video frames contain semantically similar information. This not only provides a variety of different viewing conditions in which objects can be observed but it also overcomes the restrictive barrier between the training and testing phase being each frame used for both training and testing. Specifically, each frame on a video stream can be used for learning, the following for testing and so on. Accordingly, never-ending tracking multiple subjects in the video could, at least in principle, support a sort of *self-supervised* incremental learning of their appearance. This would avoid or reduce the cost of annotation as time itself would provide a form of self-supervision which does not stop learning, but rather updates the learning model over time by accumulating knowledge without forgetting the past, reaching increasingly better accuracy and better data diversity as time advances.

However, this solution is not without problems. It is practically not possible to store all the data seen so far and re-learn a Deep Neural Network model periodically. Removing past data to adequately incorporate the new information without catastrophic forgetting, (i.e. performing Continual Learning [9]), is still an open challenge [18, 24, 25, 45], especially when new knowledge has to be incorporated in real time while tracking, without the availability of labels and with data coming from a stream which is often non-stationary [46, 47].

Single Object Tracking (SOT) [48] and Multiple Object Tracking (MOT) [49, 50] are closely related to the problem of learning from video streams but they have substantial differences and divergent goals from incremental and cumulative learning. While in SOT the object appearance is learned only for detecting the object in the next frame (the past information is gradually *forgotten* [51, 52]), cumulative learning from a video stream would require that *all* the past visual information of the object observed so far is collected in a *comprehensive* and *cumulative* representation. This not only requires tracking to be robust in the presence of very long term occlusions due intermittent (re)appearance of objects or other severe appearance changes, but that incremental learning is asymptotically stable so that it converges to an univocal cumulative representation. Moreover, modern SOT approaches based on Deep Learning are pre-trained on large video datasets [53–56] and typically do not perform any learning at runtime or perform conservative updates [57]. Their extension to handle cumulative learning remains not trivial and however prone to catastrophic forgetting. *Long-term* SOT methods introduced in [58, 59] implement explicit target re-detection to reacquire the object after long term occlusion, despite of their successful performance on complex extended video sequences, their strategy for learning the appearance model does not substantially differ from those in SOT [60, 61].

Although MOT appears similar to the problem of learning the appearance of objects from video streams, major differences can be identified according to the following four criteria:
(1) Motion Continuity and Data Association. Most methods formulate MOT as a data association problem integrating several cues such as appearance, position, motion, and size into an affinity model to link track fragments (i.e. tracklets) into final trajectories. To be usefully exploited, this formulation implicitly requires that the objects are continuously detected and the camera is stationary, slowly moving or undergoing short-term rapid motions [62–64]. The motion continuity problem can be partially mitigated by the introduction of learned appearance model (i.e. features) trained on large corpus of data to perform short-term re-identification [65–67]. Instead in long-term re-identification after long occlusions, the continuity of motion is *no longer relevant* to the problem of data association [3]. When an object exits the field of view and re-enters after an *unknown* long period of time, re-association

of the correct identity can be related *only* to the appearance of the object observed and the learned appearance model of all the objects observed so far. Video streams with many shot changes further reduce the relevance of motion continuity.
(2) Re-Id and Track deletion. MOT short-term re-identification is typically achieved by storing the appearance models of deactivated tracks for a fixed number frames such that an object can be either re-acquired or deleted (i.e. *forgotten*) [62, 67, 68]. However, simply setting a very large number frames after which to delete tracks, would require the explicit management of an undefined and large number of track identities with their corresponding appearance models undergoing cumulative learning. This issue has not been systematically investigated and therefore extending MOT approaches to include this long-term re-identification learning scenario it is not straightforward.
(3) MOT Datasets. MOT has been extensively studied with a prime focus on human body visual data where it is *not* reasonable to assume that clothes remain unchanged over very long-term periods of time as for face data. Consequently, relevant MOT datasets do not explicitly cover long-term re-acquisition and/or extended appearance variations [69, 70].
(4) Learning Setting. MOT methods have either offline or online processing mode [50] depending on whether observations from future frames are or are not utilized when handling the current frame. However, with the terms "incremental" and "cumulative" the reference here is to a learning setting rather then to a processing mode [46, 71, 72]. The term "online" alone typically used to characterize MOT methods does not reflect the concept of lifelong adaptation and cumulative learning in never-ending data streams that have changing statistics. In order to avoid confusion, we will refer to this learning setting as Multiple Object Cumulative Adaptation Learning (MOCAL).

Differently from SOT and MOT methods, in this Chapter we present a novel online self-supervised method that learns cumulative identity representations adapted to all the visual information observed so far. We evaluated our method on face visual data as it is more intrinsically pertinent to this learning setting. In order to focus on the aspects that distinguish our approach from MOT, we used datasets as in video face clustering [3, 73–75] that include large corpus of face objects with abrupt motions, extended appearance variations and very long-term occlusions.

Specifically, to achieve cumulative learning while handling the non-stationarity of the data stream, we update a representative dataset and use it as a memory of all the past visual information observed so far. To this aim, CNN face features [36, 76] are stored into a memory module and "distilled" based on their redundancy so that a compact and complete appearance representation of the individual identities is cumulative learned over time. The memory module consists of feature-identity pairs as recently introduced in [77–79]. Extracted face features from the current frame are used to both query and learn the memory model according to a Reverse Nearest Neighbor strategy [80]. The features returned by the memory are used to determine the final prediction of the identities. To avoid forgetting, no identity is explicitly deleted after a fixed number of frames has passed. As the memory increases, observed features are selectively removed only if there is subsequent information in their locality in representation space. This makes the representations of each identity more compact and discriminative since they are adapted to incorporate all the past data. When a memory budget is met, new information is written into the least used memory locations. It is further shown that the proposed incremental procedure for learning the memory module approximates asymptotically the case of infinite accumulation of feature data. A preliminary exploration of this work was presented in [39].

In Section 2.4 we expounded the approach in detail, in Section 2.5, experimental results are given and finally in Section 2.6 critical discussion and further experiments are provided.

2.2 Related Works

In this section the general and methodological issues raised in the introduction are examined in different bodies of existing literature. MOCAL setting is closely related to Continual Learning [8, 9] and Open-World learning [81]. We will describe each of them briefly highlighting the relationships/connections with our approach.

Continual Learning deals with the problem of sequentially learning a single model, preserving and reusing the previous knowledge while learning the new one. Instead Open-World learning deals with the problem of detecting new classes at test time (i.e. open-set) to avoid incorrect assignments to known classes. When new classes are incorporated in the model, then

Continual Learning meets the problem of Open-World. In the open set evaluation protocol, learned face features [35,36] with distance thresholding have shown to achieve reliable performance [82]. Our approach follows a similar strategy to detect novel identities also exploiting the fact that a single video frame eventually contains distinct face identities.

In Continual Learning, typically a sequence of tasks is learned one at a time, with label supervision, with all data of current task available and without revisiting past tasks. Task boundaries and class identities are therefore known at all times. This setting, is therefore not appropriate in applications that learn incrementally from unconstrained video streams. A recent and notable exception is provided by [47]. They learn face identities in a self supervised way. First they obtain face tracklets and then use this information to update the face representation. The tracks are then processed in chronological order so as to generate a non-i.i.d. stream of data. In our approach, representation is fixed and face-specific, but it is directly adapted from the data coming from a detector without requiring a multi-pass analysis of the video. According to this, our method allows learning in an online and cumulative fashion from an unconstrained video stream.

2.2.1 Multiple Object Tracking

An alternative approach that partially accomplishes the open-world and class-incremental learning (it does not perform cumulative learning) is Multiple Object Tracking (MOT) [49, 50, 83]. MOT exploits temporal self-supervision to automatically generate labels with data coming directly from the output of a detector. The major issue encountered by MOT when applied to cumulative learning is track management. Track identity creation and deletion are managed by two thresholds: a new identity is created when the object has been constantly detected for a certain number of frames while an identity is deleted if it is not associated for a duration of a predefined number of frames. The value of these thresholds typically depends on both the accuracy of detection models and the frame rate and are set in the order of few seconds [68]. Track deletion basically precludes MOT methods to perform long-term re-identification as required in MOCAL: objects that exit and re-enter the field of view after few seconds are managed as new identities. As a consequence, MOT methods cannot be directly applied to perform cumulative learning nor to handle unconstrained video streams.

In [3], video face clustering is exploited to adapt face appearance. Their method applies MOT in videos consisting of pre-segmented shots taken from different cameras. In order to take advantage of the continuity of the motion, each shot is processed independently to estimate tracklets. Face appearance adaptation learning is achieved with a further pass over the face image crops along the estimated tracklets by fine-tuning the CNN feature representation according to the triplet-loss [84]. This pass can be considered as addressing both adaptive and cumulative learning of the feature representation across the processed video. To overcomes the track deletion limit of MOT the fine-tuned features are then used in a final pass to cluster tracklets across multiple shots. The approach is similar to [47] except that the adaptation is not performed incrementally. Our approach follows the same intents of both [3] and [47] but formulates the problem as cumulative and online learning. Differently from [3] and [47] we can handle an infinite video stream. To this aim we leverage the success of recent tracking-by-detection approaches [85–87].

Tracking-by-detection has become the leading MOT paradigm exploiting both to the improved accuracy of CNN based object detectors [88–91] and CNN feature representation [36, 76, 84, 92]. Performance with respect to earlier methods has been largely improved especially in the online processing modality. In [87], both Faster R-CNN detections and features learned using re-identification datasets [93] are combined obtaining a performance improvement by a margin of 30% with respect to the state of the art, showing that having higher-quality detections and feature representations reduces the need of complex association/tracking algorithms. Other similar tracking-by-detection methods have recently followed: [66, 67, 94, 95]. Specifically, [67] further simplifies the tracking-by-detection MOT paradigm by removing the optimal data-association step and the motion model. Our method exploits this simplified paradigm.

Among MOT methods operating online not based on tracking-by-detection, several interesting attempts has been proposed recently to favor short-term identity preserving (i.e. occlusion between objects) against the most favorable off-line methods exploiting future information. A few methods have exploited Single Object Tracking (SOT) to manage missing detections [4, 68, 96, 97]. Tracks deletion and appearance forgetting still limits the applicability of these methods in the MOCAL setting. In particular [4] addresses tracking multiple faces that exit and re-enter the field of view. The method exploits contextual relations (i.e. upper body appearance and relative camera poses) according to a graphical model to characterize the dependency between multiple objects. It consists of two phases: in the first phase the graphical model is learned off-line from some video sequences, in the second phase the appearance of face objects are learned online according to the SOT model described in [51]. However this method cannot handle an infinite video streams since it relies on pre-segmented shots to exploit motion continuity.

2.2.2 Long-Term Single Object Tracking

Another relevant research subject to our learning setting is long-term Single Object Tracking [58, 60, 61, 98–101]. The aim of long-term SOT is to track a specific object over time and re-detect it when the object leaves and re-enters the scene. Only a few works on tracking have reported drift-free results on on very long video sequences [58, 98, 102–104] among the few, and only few of them have provided convincing evidence on the possibility of incremental appearance learning strategies that are asymptotically stable [58, 98]. However, all of these works perform incremental learning only to detect the object in the next frame and gradually forget the past information. In [3] authors evaluate a MOT baseline in which multiple TLD trackers [58] initialized with the ground-truth bounding box in the first frame are exploited. The baseline so defined can handle unconstrained videos avoiding to segment them into shots to exploit motion continuity.

2.2.3 Learning With a Memory Module

Inclusion of a memory mechanism in learning [105] is a key feature of our approach. On domains that have temporal coherence like Reinforcement Learning (RL), memory is used to store the past experience with some priority and to sample mini-batches to perform incremental/cumulative learning [106] [107]. This makes it possible to break the temporal correlations by mixing more and less recent experiences therefore handling the non-stationarity of data streams. More recently, Neural Turing Machine architectures have been proposed in [108, 109] and [110] that implement an augmented memory to quickly encode and retrieve new information. These architectures have the ability to rapidly bind never-before-seen information after a single presentation via an external memory module. However, in these cases, training data are still provided supervisedly and the methods are not primarily designed for handling video streams.

In [78] a memory module consisting of feature-value pairs to perform predictions based on past knowledge is proposed. Features are activations of the penultimate layer of a deep neural network (i.e. the internal feature representation), and values are the ground-truth targets. The output of the penultimate layer of the neural network is used as query to the memory module and the nearest neighbor returned by the memory is used as the final network prediction. As the memory increases it becomes more useful since it can give predictions that leverage on knowledge from past data with similar features. We use this basic strategy in which the feature-value pair consists in a face specific feature and its associated identity. One main limitation of [78] is in the lack of a mechanism to forget redundant observations to make rooms to novel fresh data. The work [111] suggests a memory based forgetting strategy based on the principle of spatio-temporal locality. We follow a similar principle in which observations are forgotten if there is subsequent information according to a distance ratio criterion between deep features.

2.3 Contributions

Our contributions can be summarized as follows:

1. We present a novel online method for the task of learning the appearance of face identities from unconstrained video streams. As video streams are infinitely long, this requires online accumulation and preservation of all the past visual knowledge observed so far.

2. We propose a memory module that achieves online cumulative learning in two different ways. (a) Avoiding the explicit deletion of object identities after a fixed number of frames has passed. (b) Selectively removing observed features depending on whether subsequent information in their locality is available in representation space.

3. The proposed method firstly addresses very long-term object re-acquisition in online MOT processing mode: when an object leaves the field-of-view and then reappears, it is not treated as an unseen object with a novel Id.

4. The proposed strategy is shown to be asymptotically stable. We argue this is a critical issue for any system that claims to operate lifelong.

5. The proposed method referred as IdOL (Identity Online Learning), performs comparably with offline approaches exploiting future information in the task of multiple face tracking in unconstrained videos while it achieves better performance in the face identification.

2.4 The proposed approach

The block diagram of the solution proposed is shown in Fig.2.1. We used the state of the art *Tiny Face Detector* [112] for detection and the *VGGFace* features [35] to represent faces. A memory module is used to collect the face features. In the ideal case (i.e. perfect invariance of the representation), observations of the same subject originate the same features. In the real case, we must expect that observations of the same subject under changes of pose or illumination or partial occlusions originate different (although correlated) features. The matching module is a discriminative classifier that associates each new observation to the most similar past observations already in the memory. The memory controller has the task of discarding redundant features: highly similar features of the same subject having comparable distance feature already in the memory module. Ideally, a new identity should be created whenever a new individual is observed that has not been observed before.

We loosely follow [78] and the memory module at time t is represented as:

$$\mathcal{M}(t) = \{(\mathbf{x}, \mathrm{Id}, e, a)_i\}_{i=1}^{N(t)} \tag{2.1}$$

where i is the index of a memory element and $\mathbf{x}$ is a deep feature, Id is the face identity associated, e is a value referred to as *eligibility* that accounts for the relevance of the item to be learned (discussed in the following), a is a value that tracks the age from the last match, and *N(t)* is the number of features in the memory at time t. We extend the feature-value pair (i.e. $\mathbf{x}$–Id) and the age in [78] by adding a further scalar quantity.

The mechanisms of identity matching, construction of the identity models, self-supervision using temporal coherence and the asymptotic behavior of the method are separately addressed in detail in the following subsections.

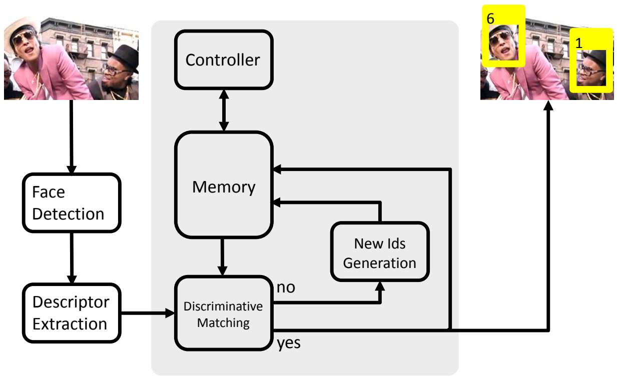


Figure 2.1: Block diagram of the incremental identity learning with basic workflow.

2.4.1 Reverse Nearest Neighbor Matching

The Nearest Neighbor distance ratio criterion [113] allows matching based on a discriminative rule between the most and the second most similar sample. Unfortunately, in our learning scenario, we cannot exploit Nearest Neighbor with distance ratio criterion to assess matching. Since it is likely that detected faces of the same subject in consecutive frames have little differences from one frame to the following, similar features having comparable distances to the nearest and the second nearest will rapidly be stored in the memory module. As a consequence, the distance ratios of observations to the nearest and the second nearest feature in memory will be close to 1 and matchings are undecidable in most cases. To solve this problem, we propose to use Reverse Nearest Neighbor (ReNN) with the distance ratio criterion [80].

With ReNN, each feature in memory is NN-matched with the features of the observations in the incoming frame and distance ratio is used to assess matching. Fig. 2.2 explains this matching mechanism for a sample case. The features $\mathbf{o}_1$ has the same identity as the $\mathbf{x}_i$ in the memory while $\mathbf{o}_2$ has a different identity. Due to the fact that the $\mathbf{x}_i$ are close to each other, the NN-distance ratios of $\mathbf{o}_1$ and $\mathbf{o}_2$ to their nearest and second nearest $\mathbf{x}_i$ are are both close to 1 and both matchings result to be NN-undecidable (Fig. 2.2-*left*). Instead, with ReNN, the NN-distance ratios between the $\mathbf{x}_i$ and $\mathbf{o}_1$ and $\mathbf{o}_2$ clearly assess the matching with $\mathbf{o}_1$ (Fig. 2.2-*right*). The set of $\mathbf{x}_i$

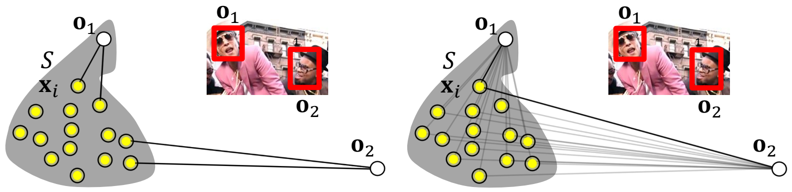


Figure 2.2: Nearest Neighbor (*left*) and Reverse Nearest Neighbor (*right*) matching with distances between the stored features and the observations. The stored features $\mathbf{x}_i$ in the grey area all have the same identity. ReNN can assess matching between $\mathbf{x}_i$ and $\mathbf{o}_1$ according to the distance ratio criterion.

that are ReNN matched to the observations at time t can be written as:

$$\mathcal{M}^+ = \left\{ (\mathbf{x}, \mathrm{Id}, e, a)_i \in \mathcal{M}(t) \mid \frac{d_i^1}{d_i^2} < \bar{\rho} \right\} \tag{2.2}$$

where $\frac{d_i^1}{d_i^2}$ is the distance ratio between $\mathbf{x}_i$ and the nearest and second nearest face features in the frame at time t and $\bar{\rho}$ is the distance ratio threshold.

2.4.2 Learning the Memory Module

Collecting matched features indefinitely will soon accumulate features in the memory module and a large amount of redundant information will be included for each identity model. To avoid such redundancy, we associate to each i-th feature-identity pair a dimensionless quantity e_i referred to as *eligibility-to-be-learned* (shortly *eligibility*) that dynamically indicates the level of redundancy of the feature to be learned as representative of the identity. Eligibility is set to 1 when the feature is loaded into the memory and is decreased at each match with the observations according to:

$$e_i(t+1) = \eta_i \, e_i(t) \eta_i = \left[\frac{1}{\bar{\rho}} \frac{d_i^1}{d_i^2} \right]^{\alpha}, \tag{2.3}$$

where the matching threshold $\bar{\rho}$ of Eq. 2.2 is used for normalization and α to dilate the effect of the distance-ratio. When doing this, we also reset the feature age $a_i = 0$. As the eligibility e_i of a face feature $\mathbf{x}_i$ drops below a given threshold $\bar{e}$ (that happens after a number of matches), the feature is no more eligible to be learned as representative of the identity and is removed from the memory.

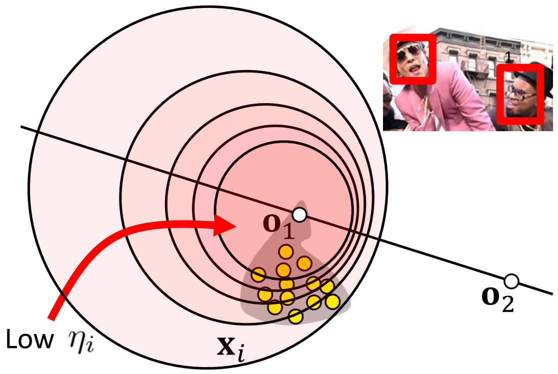


Figure 2.3: Learning the memory module. The 2D shape of the density function (shown by level curves) down-weighting the eligibility associated to each matched feature. Features $\mathbf{x}_i$ in proximity of the observed feature $\mathbf{o}_1$ have their eligibility decreased (low values of η) to reflect their redundancy. The asymmetric shape of the density encourages more diversity in the *open space* far from the identity $\mathbf{o}_2$ rather than close.

Eq. 2.3 down-weights eligibility as a function of the distance ratio at a rate proportional to the success of matching in consecutive frames. According to this, eligibility allows to take into account spatio temporal redundancy in a discriminative way. The equation it is a generalization of the Apollonious circle [3] to multiple dimensions. As shown in Fig. 2.3, features in regions close to $\mathbf{o}_1$ and far from $\mathbf{o}_2$ (dark red) have low η and therefore their eligibility is more down-weighted (they will have higher chance to be replaced in the future). Features in regions far from $\mathbf{o}_1$ and $\mathbf{o}_2$ (light red) have higher η and their eligibility is less down-weighted and their chance of not being discarded is higher. This asymmetry promotes diversity in the open space and defines a learning schema well suited for the *open world* face recognition scenario.

Our method operates on-line and does not require any prior information about how many identity classes will occur and can run for an unlimited amount of time. However, if the number of identities increases indefinitely the eligibility-based exemplar removal may not be sufficient to avoid memory overflow. Similarly to [78, 114], we remove from the memory the least

[3]Apollonius of Perga (c. 262 BC - c. 190 BC) showed that a circle may also be defined as the set of points in a plane having a constant ratio of distances to two fixed foci.

recently matched exemplars (those with the highest value of the paramter a in Eq. 2.1), following the Least Recently Used Access (LRUA) strategy. This also allows to remove false positives by the detector that have not received other matches for a long time.

2.4.3 Self-supervision

The cumulative learning mechanism of the memory module breaks the temporal coherence of the data stream (i.e. *non-iid*) by mixing more and less recent recent observations. Nevertheless temporal coherence is used as form of self-supervision in the assignment of novel identities to limit their fragmentation and proliferation.

Assuming that faces of the same individual have similar features in consecutive frames, in the case in which an observation does not match the feature in memory, its feature is included in the memory with a new identity only if the same identity is assigned also in the following frames (two consecutive identity assignments and at least one matching in the following three frames was experimentally verified to provide good results). With this form of verification potential novel identities in the current frame are included in the memory only if at least one known identity is recognized. Since recognition is obtained according to the RNN with the distance ratio and since observations taken from a single frame derive from distinct identities, the unmatched identities in the current frame are known to be reasonably distant (i.e. different) from the recognized ones and are considered potentially novel.

In the case in which no observations match with features in memory, new identities are assigned to the non-matched observations only if the same situation persists for a time interval.

ReNN matching can determine ambiguous assignments when distinct face observations match with features of the same identity, or an observation matches with features of different identities in memory. In the first case we assign no identity to the observations (i.e. identities in the current frame are unique and therefore duplicated Ids in the same frame are not allowed). In the second case, we assign the most represented identity, i.e. that with the largest number of features in memory.

The complete algorithm of our IdOL (Identity On-line Learning) method for incremental identity learning, is reported in pseudocode in Algorithm 1. We indicate with $\mathcal{O}$ as the set of all the features extracted from the bounding boxes reported by the face detector in the current frame, and with $\mathcal{M}$ as the set of features in the memory module. Correspondences between $\mathcal{M}$ and $\mathcal{O}$ are computed in line 4 according to the Reverse Nearest Neighbour matching. The sets $\mathcal{M}^+$ and $\mathcal{O}^+$ indicate the elements that have established a direct correspondence. The set $\mathcal{I}_{curr}$ will contain (if any) the identity labels of novel subjects (not present in the memory model) detected in the current frame. It is initialized to the empty set for each novel frame (line 8). In line 9 all the matched observations $\mathcal{M}^+$ in the memory module are updated according to Eq. 2.3 and have their age reset to 0. Then, in line 10 potential identities $\mathcal{I}$ are predicted and subsequently intersected with those predicted in the previous frame $\mathcal{I}_{prev}$ to obtain the set $\mathcal{I}_{curr}$ of the identity labels estimated for the current frame (line 11). In line 12 the estimated label identities together with their observed features $\mathcal{O}_{curr}$ will be added in to the memory module as novel Ids with their eligibility and age values are set to 1 and 0 respectively. With an excess of notation we denote $\boldsymbol{1}$ and $\boldsymbol{0}$ as arrays of elements of value 1 and value 0 respectively. Their length is the same as the number of elements in $\mathcal{I}_{curr}$. In line 13 the potential identity labels of the previous frame $\mathcal{I}_{prev}$ are updated with those estimated in the current frame. In line 19 the t_{nc} counter is incremented when a frame has no matched correspondences. It get reset to 0 the first time a match occurs (line 14) or after that a number of frames $\hat{t}_{nc}$ are elapsed (line 16). In the latter case all the detected observations are declared as novel. In line 20 all observations from memory that are never matched after being included are removed if their age is greater than a given threshold $\bar{a}$.

2.4.4 Asymptotic Stability

The cumulative learning procedure described above stabilizes asymptotically around the probability density function of the features of each identity. This is guaranteed by the fact that the memory updating rule of Eq. 2.3 is a *contraction* that converges to its unique *fixed point*[4] according to the Contraction Mapping Theorem [115]:

Banach Contraction Mapping Theorem
Let (X, d) be a complete metric space and $M : X \mapsto X$ be a map

[4]A fixed point of a function is an element of the function's domain that is mapped to itself by the function.

Algorithm 1: IdOL - Identity On-line Learning

```
Input: The video stream.
Output: Assigned identities $\mathcal{I}_{curr}$ in the current frame.
1  repeat
2      Detect faces in the current frame;
3      Extract observations features $\mathcal{O}$;
4      Establish correspondences: $\mathcal{M}^+ \leftrightarrow \mathcal{O}^+ = \mathrm{ReNN}(\mathcal{M}, \mathcal{O})$;
5      Identify non-matching memory elements: $\mathcal{M}^- = \mathcal{M} \setminus \mathcal{M}^+$;
6      Identify non-matched observations: $\mathcal{O}^- = \mathcal{O} \setminus \mathcal{O}^+$;
       // Case with matched observations (eligibility updating and
          temporal coherence verification)
7      if $|\mathcal{O}^+| > 0$ then
           // Initialize the set of identities to be included in the
              memory
8          $\mathcal{I}_{curr} = \varnothing$;
           // Update the eligibility with the matched observations
9          $\mathcal{M} = \left\{(\mathbf{x}, \mathrm{Id}, \eta e, 0)_i \,|\, \forall\, (\mathbf{x}, \mathrm{Id}, e, a)_i \in \mathcal{M}^+\right\} \cup \mathcal{M}^-$;
           // Assign known and novel identities to the observations
10         $\mathcal{I} = \mathit{MajorityId}(\mathcal{M}^+ \leftrightarrow \mathcal{O}^+) \bigcup \mathit{NewId}(\mathcal{O}^-)$;
           // Keep identities assigned in two consecutive frames
11         $\mathcal{I}_{curr} = \mathcal{I} \cap \mathcal{I}_{prev}$;
           // Include them in the memory module with their observations
12         $\mathcal{M} = \mathcal{M} \cup \left\{(\mathcal{O}_{curr}, \mathcal{I}_{curr}, \mathit{1}, \mathit{0})\right\}$ ;
           // Keep the assigned identities for the next frame
13         $\mathcal{I}_{prev} = \mathcal{I}$;
14         $t_{nc} = 0$;
       // Case with no matched observations (novel identity assignment
          after a time interval has elapsed)
15     else if $t_{nc} > \bar{t}_{nc}$ then
16         $\mathcal{M} = \mathcal{M} \cup \left\{(\mathcal{O}, \mathit{NewId}(\mathcal{O}), \mathit{1}, \mathit{0})\right\}$;
17         $t_{nc} = 0$;
18     else
19         $t_{nc} = t_{nc} + 1$
20     $\mathcal{M} = \mathcal{M} \setminus \left\{(\mathbf{x}, \mathrm{Id}, e, a)_i \in \mathcal{M} \,|\, a_i > \bar{a}, e_i = 1\right\}$;
21     $\mathcal{M} = \mathcal{M} \setminus \left\{(\mathbf{x}, \mathrm{Id}, e, a)_i \in \mathcal{M} \,|\, e_i < \bar{e}\right\}$;
22     $\mathcal{M} = \left\{(\mathbf{x}, \mathrm{Id}, e, a+1)_i \,|\forall (\mathbf{x}, \mathrm{Id}, e, a)_i \in \mathcal{M}\right\}$;
23     Apply LRUA;
24 until $\mathit{True}$;
```

(referred to as *contraction*) such that

$$d(M(x), M(x')) \leq c \cdot d(x, x')$$

for some $0 < c \leq 1$ and all x and $x' \in X$. Then M has a unique *fixed point* in X. Moreover, for any $x \in X$ the sequence of iterates

Table 2.1: IDS and MOTA comparative for the methods in [3] (*Music* dataset)

	Apink		BrunoMars		Darling		GirlsAloud	
Method	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑
mTLD*	31	−2.2	35	−8.7	24	−22.0	9	−1.1
mTLD2*	173	77.4	278	52.6	278	59.8	322	46.7
ADMM*	179	72.4	428	50.6	412	53.0	487	46.6
IHTLS*	173	74.9	375	52.7	381	62.7	396	51.8
Siamese*	124	79.0	126	56.7	214	69.5	112	51.6
Triplet*	140	78.9	126	56.6	187	69.2	80	51.7
SymTriplet*	78	80.0	105	56.8	169	70.5	64	51.6
IdOL (VGGFace/VGG16-4096)	191	55.1	420	48.8	449	62.1	339	49.3
IdOL (VGGFace2/ResNet-2048)	178	61.4	375	59.4	432	63.0	315	55.0
IdOL (VGGFace2/SeNet-128)	177	62.6	367	60.1	427	64.2	306	55.8

	HelloBubble		PussycatDolls		Tara		Westlife	
Method	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑
mTLD*	7	−3.5	24	3.1	130	1.4	20	−34.7
mTLD2*	139	52.6	296	68.3	251	56.0	177	58.1
ADMM*	115	47.6	287	63.2	251	29.4	223	62.4
IHTLS*	109	52.0	248	70.3	218	35.3	113	60.9
Siamese*	105	56.3	107	70.3	106	58.4	74	64.1
Triplet*	82	56.2	99	69.9	94	59.0	89	64.5
SymTriplet*	69	56.5	82	70.2	75	59.2	57	68.6
IdOL (VGGFace/VGG16-4096)	88	51.4	83	30.7	270	39.5	76	58.9
IdOL (VGGFace2/ResNet-2048)	92	49.1	80	33.7	257	42.3	70	64.1
IdOL (VGGFace2/SeNet-128)	85	51.5	77	35.2	254	42.5	68	63.9

* Values reported from [3]

x, $M(x)$, $M(M(x)),...,M(...M(M(x)))$ converges to the *fixed point.*

In our case, the memory updating mechanism of Eq. 2.3:

$$e(t+1) = \eta\, e(t) \quad \text{with} \quad \eta = \left[\frac{1}{\bar{\rho}}\frac{d^1}{d^2}\right]^{\alpha}$$

being $\eta \in (0,1]$, satisfies the conditions of the theorem above. It can be observed that the value $e = 0$ is the fixed point of this equation and corresponds to the case of an infinite accumulation of samples. In such a case, the Nearest Neighbor classifier error is bounded by twice the Bayes risk [116]. In our case, to have a finite number of samples, a threshold $\bar{e}$ close to 0 can be set that approximates with continuity the case of the infinite sample set.

2.5 Comparative evaluation

Our method is evaluated over publicly available datasets, namely *Music* and *Big Bang Theory* [117] and *QMUL multi-face dataset* [118]. We use the MOTA (Multiple Object Tracking Accuracy) performance metric defined as [70]:

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDS}_t)}{\sum_t \mathrm{GT}_t} \tag{2.4}$$

where GT_t, FN_t, FP_t and IDS_t are respectively the number of ground truth objects, the number of false negatives, the number of false positives and the number of identity switches at each time t.

We compare our solution with the performance of the offline methods in [3]:

- mTLD running the TLD tracker in each shot [119]
- mTLD2 a modified versions of TLD that generates shot-level trajectories [3]
- ADMM [120];
- IHTLS [121];
- *Siamese*, *Triplet* and *SymTriplet* methods [3],

All these methods operate offline (so they exploit both past and future frames to learn identities). They apply the *Headhunter* version of DPM detector by [122] and detections are then linked into shot-level tracklets. Tracklets across shots are hence merged into trajectories using Hierarchical Agglomerative Clustering [123]. The *Siamese*, *Triplet* and *SymTriplet* methods [3] have a sophisticated refinement of identity assignment. Tracklets are used in pairs (in the *Siamese*) or triplets (in the *Triplet* and *SymTriplet*) to fine-tune an AlexNet-based CNN pretrained on the CASIA-WebFace and the descriptor of the fine-tuned CNN is finally used to link tracklets into shot-level tracklets. Comparison with these methods were made over the Music and Big Bang Theory datasets.

Tab. 2.1 provides a comparative overview of MOTA and IDS scores for the videos of the *Music* dataset. These are YouTube videos of live vocal concert recordings with very frequent shot changes (i.e. unconstrained videos), views from different cameras and special effects. There are a limited number of annotated characters in continuous fast movement. Faces have large variations of appearance due to rapid changes in pose, scale, makeup, illumination, camera motion and occlusions. In total, there are 117,598 face detections and 3,845 face tracks annotations. Our IdOL method has lower MOTA in most videos (although almost the same of ADMM and IHTLS). However, it has comparable IDS for HelloBubble, Apink PussycatsDolls and Westlife videos and lower IDS for Tara.

As feature representation is one of the main component of our method, Tab. 2.1 reports performance evaluated according to three different feature representations:

- 4096-dimensional feature learned from VGGFace with VGG16 architecture [76] (VGGFace/VGG16-4096)
- 2048-dimensional feature learned from MS-Celeb-1M and fine-tuned on VGGFace2 dataset [36] with SeNet [124] (VGGFace/SeNet-128)
- 128-dimensional feature learned from MS-Celeb-1M and fine-tuned on VGGFace2 dataset with ResNet [125] (VGGFace2/ResNet-2048) [5]

As it can be noticed, performance follows the increasing quality of the different feature representation evaluated. This is due to the expressive power of more competitive CNN architectures and the exploitation of richer training datasets.

Fig.2.4 shows the plot of MOTA of our method computed at each frame for the VGGFace/VGG16-4096 features. Due to the incremental learning mechanism, at the beginning there is not sufficient information available so the identity models are largely incomplete and a large number of errors may occur. As more and more observations are received that contain different views and conditions of the faces, MOTA stabilizes. In the *Music* dataset, asymptotic values of MOTA were reached approximately after 1000 frames for all the videos, despite of the different editing and contents.

In order to assess our method on longer video sequences also in conditions similar to surveillance contexts, we compared performance also on the *Big Bang Theory* dataset. This dataset collects six episodes of *Big Bang Theory* TV Sitcom, Season 1. These are much longer videos (approx 20' each) with indoor ordinary scenes under a variety of settings and illumination conditions. They contain a much larger number of identities with crowded scenes. Also in this case, faces have large variations of appearance

[5]Pretrained models are available at https://github.com/ox-vgg/vgg_face2

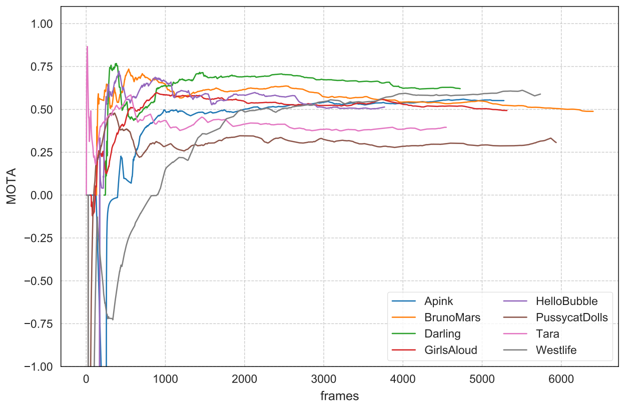


Figure 2.4: MOTA computed at each frame for the videos in the *Music* dataset

due to rapid changes in pose, scale, makeup, illumination, camera motion and occlusions.In total, there are 373,392 face detections and 4,986 face tracks annotations.

Table 2.2 reports MOTA and IDS scores for the videos of the *Big Bang Theory* dataset and Fig. 2.5 shows the plots of MOTA computed at each frame. It can be noticed that considerations similar to those drawn for the *Music* dataset hold also in this case, despite of the differences between the two datasets. The presence of less frequent cuts and less extreme conditions due to editing effects and camera takes than in the *Music* dataset determines sensibly lower Identity Switch values and closer MOTA values in almost all the videos. MOTA plots have earlier convergence to their asymptotic values. They all share similar behavior due to the uniform style of the series.

We further compare our solution with performance of methods reported in [4]:

- Tracking-Clustering [126]

- Min-Cost Flow [127]

- CRF [4]

- M^3 Networks [4]

Table 2.2: IDS and MOTA comparative for the methods in [3] (*Big Bang Theory* dataset)

	BBT_S01E01		BBT_S01E02		BBT_S01E03		BBT_S01E04		BBT_S01E05		BBT_S01E06	
Method	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑	IDS ↓	MOTA ↑
mTLD*	1	−16.3	1	−7.6	5	−2.1	0	−15.9	1	−15.5	0	−3.9
mTLD2*	223	58.4	174	43.6	142	38.0	103	11.6	169	46.4	192	37.7
ADMM*	323	42.5	395	41.3	370	30.8	298	9.7	380	37.4	527	47.5
IHTLS*	312	45.7	394	42.4	376	33.5	295	13.3	360	33.8	515	43.2
Siamese*	144	69.0	116	60.4	109	52.6	85	23.0	128	60.7	156	46.2
Triplet*	164	69.3	143	60.2	121	50.7	103	18.0	118	60.5	185	45.4
SymTriplet*	156	72.2	102	61.6	126	51.9	77	19.5	90	60.9	196	47.6
IdOL	26	60.4	55	45.2	14	46.1	75	53.9	35	44.7	204	43.0

* Values reported from [3]

Table 2.3: IDS and MOTA comparative for the methods in [4] (*QMUL multi-face dataset*)

	Average	
Method	IDS	MOTA
Tracking-Clustering*	23	61.8
Min-Cost Flow*	29	53.7
CRF*	20	65.2
M^3 Networks*	17	68.8
IdOL (VGGFace)	10.0	81.5
IdOL (VGGFace2/SeNet)	**2.3**	**87.5**
IdOL (VGGFace2/ResNet)	2.3	87.2

* Values reported from [4]

Specifically Tracking-Clustering is a modified approach of [126] in which the Haar cascade face detector is replaced with a DPM detector and Min-Cost Flow performs offline optimal data association. Comparison with these methods were made over the *QMUL multi-face dataset*. This dataset consists of three single shot video sequences with four subjects entering and exiting the field of view, namely FRONTAL, FAST and TURNING. Although captured by a static camera, all three video sequences contain intense face motions and occlusions. In addition, subjects change their face poses frequently in the TURNING sequence and perform fast movements in the FAST sequence.

Tab. 2.3 provides a comparative overview of MOTA and IDS scores for the videos of the QMUL Multiple Face Dataset. As can be noticed our approach consistently outperforms the other four compared methods on both MOTA and IDS. Since the dataset is composed by single shot videos, [4] can operate

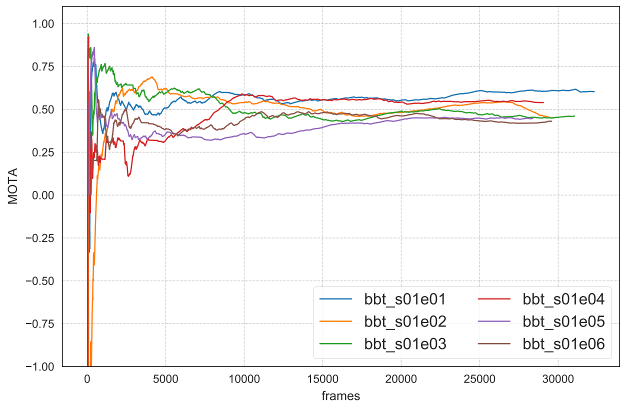


Figure 2.5: MOTA computed at each frame for the videos in the *Big Bang Theory* dataset

online as our method (i.e. the offline shot segmentation pass is not required).

2.6 Critical discussion and additional experiments

The MOTA score is a largely accepted metrics for Multiple Object Tracking. However it has clear limitations to assess the performance of cumulative learning as in the MOCAL learning setting. In the following, we discuss such limitations and perform additional evaluations.

2.6.1 Influence of detection

MOTA produces a cumulative score considering False Positives, False Negatives and Identity Switches. Typically, the number of False Positives and False Negatives are much larger than Identity Switches (see Table 2.5 f.e.). False Positives of MOTA are essentially determined by false positive detections, while False Negatives are in part due to missed detections and in part to the case in which no identity is assigned to a face observation. From the above it descends that MOTA score can be largely influenced by the performance of the detector.

Table 2.4: Influence of detection: FP FN MOTA for different detectors (*TARA* video of *Music* dataset)

	Tara		
Method	FP ↓	FN ↓	MOTA ↑
IdOL *HeadHunter* detector	1939	8592	25.6
IdOL *Tiny-face* detector	259	8259	39.5

Table 2.5: Influence of detection: IDS FN FP MOTA of IdOL using ground-truth bounding boxes versus Tiny Detector bounding boxes (*Music* dataset)

	Ground Truth Bounding Box				Tiny Detector			
Video	IDS ↓	FN ↓	FP ↓	MOTA ↑	IDS ↓	FN ↓	FP ↓	MOTA ↑
Apink	130	2105	0	69.3	191	2627	446	55.1
BrunoMars	391	3644	0	75.8	420	4178	3950	48.8
Darling	361	2620	0	68.7	449	2278	887	62.1
GirlsAloud	469	4837	0	67.6	339	6691	1272	49.3
HelloBubble	160	707	0	83.4	88	2150	301	51.4
PussycatDolls	316	4697	0	64.9	83	7050	2764	30.7
Tara	542	5210	0	60.4	270	8259	259	39.5
Westlife	138	1433	0	86.2	76	3403	1198	58.9

While it can be presumed that a more effective detector has little influence on the performance of the MOT methods reported in the comparison (these methods operate off-line and most tracklets due to erroneous detections can be removed using the future information), the effectiveness of the detector largely influences the performance of online incremental identity learning, since future information cannot be exploited in this case. A key requirement for our task is therefore that the detector has as few False Positives and False Negatives as possible. The *Headhunter* detector was verified being clearly inadequate to this end. Table 2.4 shows the performance gap of the IdOL method with the *Headhunter* and the *Tiny Face Detector*, for the T-ara video of the *Music* dataset (the detections of the *Headhunter* were released only for this video by the authors). However, since the *Tiny Face detector* is capable to detect also very small-sized faces (say less than 40 pixels), given the fact that most of these faces are not ground-truth annotated in the *Music* and *Big Bang Theory* datasets (see Figure 2.6 as an example),

Table 2.6: Influence of detection: IDS FN FP MOTA of IDoL using ground-truth bounding boxes versus Tiny Detector bounding boxes (*Big Bang Theory* dataset)

	Ground Truth Bounding Box				Tiny Detector			
Video	IDS ↓	FN ↓	FP ↓	MOTA ↑	IDS ↓	FN ↓	FP ↓	MOTA ↑
BBT_S01E01	218	1361	0	96.0	26	4631	10 875	60.3
BBT_S01E02	178	3947	0	86.3	55	5525	10 914	45.2
BBT_S01E03	191	6999	0	79.6	14	9668	9285	46.1
BBT_S01E04	341	2345	0	92.1	75	8615	7054	53.9
BBT_S01E05	381	3130	0	89.8	35	9919	9009	44.7
BBT_S01E06	559	5449	0	87.4	204	15 872	11 103	43.0

it happens that False Positives are counted whenever the detector detects a non annotated face. Tables 2.5 and 2.6 show the increase of performance achievable by the IdOL method in the ideal condition of no False Positive detections, considering the ground truth bounding boxes (left side).

False Positives of the detector have the additional drawback of increasing identity switching and the number of wrong new identities. In the IdOL method, the mechanism of Temporal Coherence verification limits the proliferation of such new identities. The effects of the Temporal Coherence verification are shown in Table 2.7 for the *Music* dataset using the ground truth bounding boxes as detections. While it may increase False Negatives, it avoids the increase of Identity Switches.

2.6.2 Cluster Purity

The MOTA score computes a cumulative performance score until the time of evaluation based on instantaneous measures of False Negatives, False Positives and Identity Switches. According to this, if in a sequence an Identity Switch occurs at a frame and the new (incorrect) identity is confirmed in the following frames, MOTA counts one Identity Switch only, at the frame at which it occurred. Instead in the case above, to assess the quality of online learning we should count as many Identity Switches as the times the original identity has been mismatched. According to this, it appears that MOTA is not fully adequate to measure the performance of on-line identity learning.

Table 2.7: Influence of Temporal Coherence: IDS FN MOTA of IDoL using Temporal Coherence versus IdOL without Temporal Coherence (*Music* dataset)

	IdOL with Temporal Coherence			IdOL without Temporal Coherence		
Video	IDS ↓	FN ↓	MOTA ↑	IDS ↓	FN ↓	MOTA ↑
Apink	130	2105	69.3	265	1558	74.9
BrunoMars	391	3644	75.8	741	2301	81.8
Darling	361	2620	68.7	655	2017	72.0
GirlsAloud	469	4837	67.6	897	3927	70.6
HelloBubble	160	707	83.4	295	358	87.5
PussycatDolls	316	4697	64.9	884	3787	67.3
Tara	542	5210	60.4	1025	4436	62.4
Westlife	138	1433	86.2	274	1105	87.9

A better metrics is the Weighted Cluster Purity (WCP), defined as [3]:

$$WCP = \frac{1}{M}\sum_{c} m_c p_c \tag{2.5}$$

where M is the number of identities detected in the video, c the index of the cluster, m_c the number of identity instances in the cluster and p_c the cluster purity, measured as the ratio between the most occurred identity in the cluster and m_c.

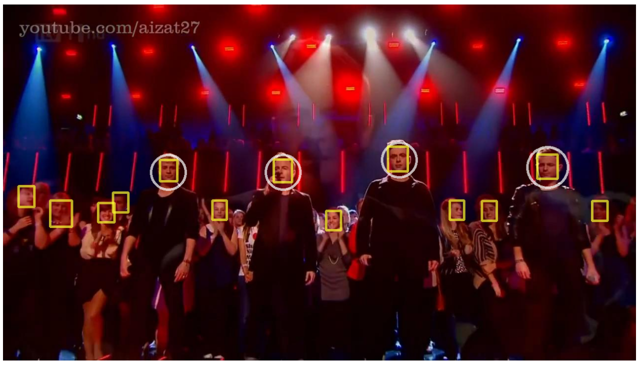


Figure 2.6: Detections of the *Tiny-face* detector for a sample frame (*Music* dataset). Most of the small bounding boxes are not annotated as faces in the ground-truth. Ground-truth annotated faces are circled.

Tables 2.8 and 2.9 present the WCP scores for the *Music* and *Big Bang Theory* datasets, and compare the IdOL method with respect to a few MOT methods of [3]. In almost all the cases the IdOL method method has largely better WCP scores. The lower score in the APINK video with respect to *Siamese*, *Triplet* and *Symtriplet* methods can be ascribed both to the ethnicity bias of the *VGGFace* features and to the effect of fine tuning on these methods.

Table 2.8: Weighted Cluster Purity score comparative. *Music* dataset.

Method	APINK	BRUNO MARS	DARLING	GIRLS ALOUD	HELLO BUBBLE	PUSSYCAT DOLLS	T-ARA	WESTLIFE
VGG-Face*	0.24	0.44	0.20	0.31	0.29	0.46	0.23	0.27
Siamese*	0.48	0.88	0.46	0.67	0.54	0.77	0.69	0.54
Triplet*	0.60	0.83	0.49	0.67	0.60	0.77	0.68	0.52
SymTriplet*	**0.72**	0.90	0.70	0.69	**0.64**	0.78	0.69	0.56
IdOL (VGGFace/VGG16-4096)	0.51	**0.96**	0.73	0.89	0.59	**0.98**	0.72	**0.99**
IdOL (VGGFace2/ResNet-2048)	0.72	0.92	0.78	0.92	0.47	0.97	0.74	0.99
IdOL (VGGFace2/SeNet-128)	**0.73**	0.91	**0.79**	**0.93**	0.48	0.98	**0.78**	0.95

* Values reported from [3]

Table 2.9: Weighted Cluster Purity score comparative. *Big Bang Theory* dataset.

Method	BBT_S01E01	BBT_S01E02	BBT_S01E03	BBT_S01E04	BBT_S01E05	BBT_S01E06
VGG-Face*	0.91	0.85	0.83	0.54	0.65	0.46
Siamese*	0.94	0.95	0.87	0.74	0.70	0.70
Triplet*	0.94	0.95	0.92	0.74	0.68	0.70
SymTriplet*	0.94	0.95	0.92	0.78	0.85	0.75
IdOL	**0.99**	**0.99**	**0.94**	**0.94**	**0.99**	**0.97**

* Values reported from [3]

Fig. 2.7 and 2.8 show the WCP plots for the videos in the *Music* and the *Big Bang Theory* datasets. At each frame WCP is calculated from the beginning up to that frame. It can be noticed that for the *Big Bang Theory* plots rapidly converge to the asymptotic values (each cluster contains a sufficiently complete description of the identity and features of different identities have been discarded). In some videos of the *Music* dataset, the presence of very frequent discontinuities and extreme conditions makes less effective the mechanism for keeping identity switches low.

Fig. 2.10 shows sample frames of the *Big Bang Theory* and *Music* videos with the detected faces and their assigned identities. For each video two frames are shown where the same persons are taken in different conditions, with large appearance variations due to partial occlusions (Figs. 2.10b, 2.10c,

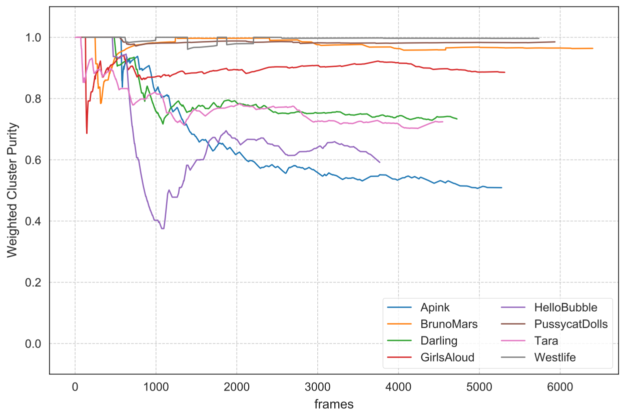


Figure 2.7: Weighted Cluster Purity computed at each frame for the videos in the *Music* dataset

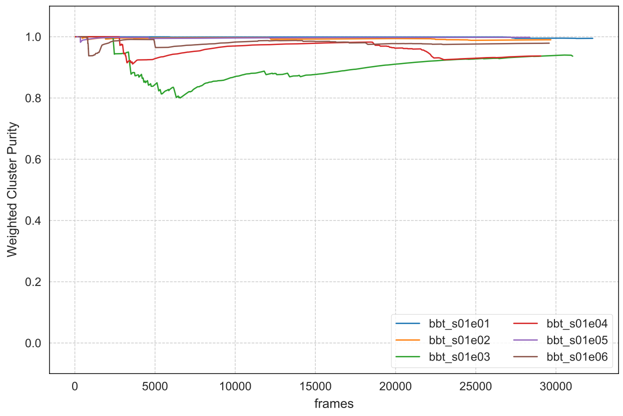


Figure 2.8: Weighted Cluster Purity computed at each frame for the videos in the *Big Bang Theory* dataset

2.10h, 2.10g), pose changes (Figs. 2.10a, 2.10c, 2.10e, 2.10f), aspect change (Figs. 2.10d, 2.10e) and in-plane rotations (Figs. 2.10g). It can be noticed that the learning mechanism is able to distinguish the same identity also in the presence of such large variations. Fig. 2.10i evidences the effect of the ethnicity bias of the *VGGFace* features. In this case, the method is not able to predict unique identities for the faces and does not make any identity assignments.

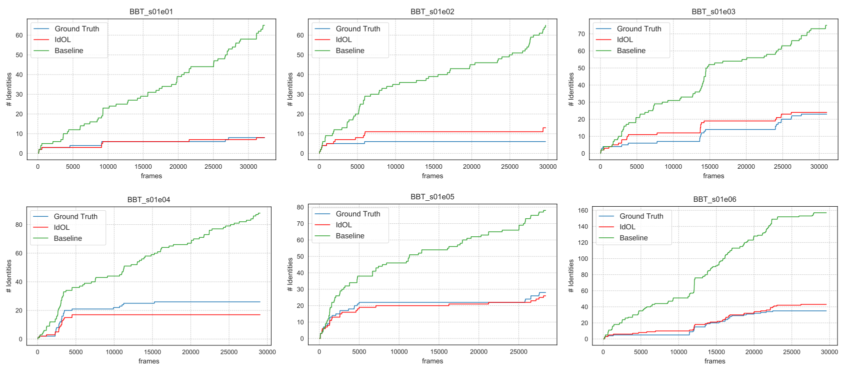
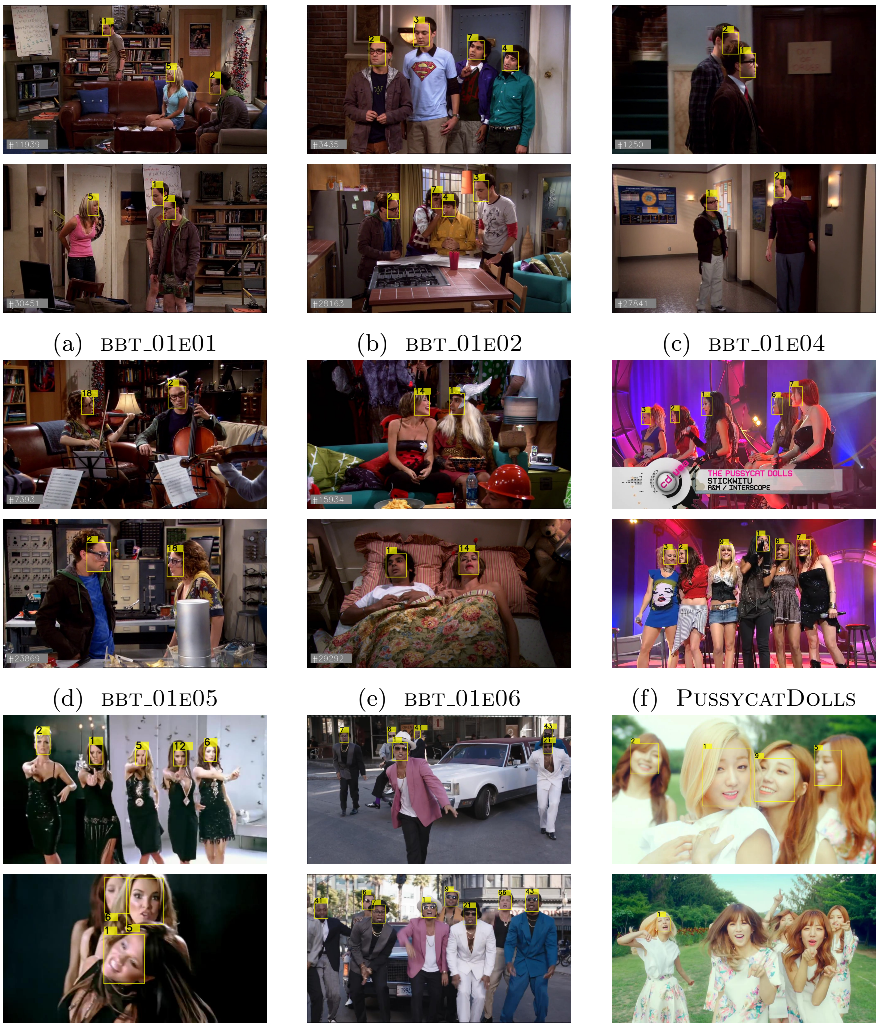


Figure 2.9: Number of identities learned at each frame for the videos in the *Big Bang Theory* dataset: Ground-truth, IdOL and Baseline.

The plots in Fig. 2.9 show the number of identities learned by the IdOL method at each frame in comparison with the ground truth for the videos of the *Big Bang Theory* dataset. For the sake of comparison we also show the plot of a *Baseline* approach that performs 1-Nearest Neighbor matching (i.e. forward) with thresholding of the distance value. For this baseline a maximal number of memory elements for each identity is used and elements are removed randomly when the identity budget is met (the best result of experiments with different distance thresholds and max memory elements is reported). As can be noticed, the number of identities estimated by the *Baseline* method increases with time while our approach closely follows the number of identities of the ground-truth. The results confirm the effectiveness of the IdOL method to learn an unknown number of identities.



(a) BBT_01E01 (b) BBT_01E02 (c) BBT_01E04

(d) BBT_01E05 (e) BBT_01E06 (f) PussycatDolls

(g) GirlsAloud (h) BrunoMars (i) Apink

Figure 2.10: Sample pairs of frames from the *Music* and *Big Bang Theory* datasets showing takes of the same individuals in different conditions and the identity label assigned by the IdOL method

2.6.3 Scaling and Asymptotic Stability

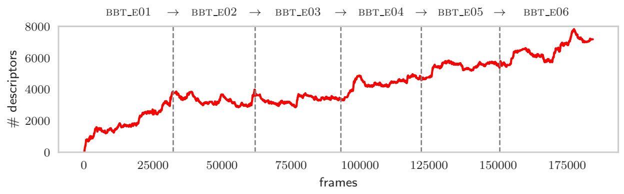
As the number of identities grow, online learning of identities becomes more challenging. In order to verify the behavior of the IdOL method in this case, we concatenated the six sequences of the *Big Bang Theory* dataset to form a single longer sequence of about two hours, and manually annotated all the subjects up to a total number of 99 different identities.

Fig. 2.11a and Fig. 2.11b respectively show the number of features in memory at each time instant and the number of ground-truth identities that showed up until then. It can be noticed that the number of features in memory follows the same trend as the number of identities. Fig. 2.11c shows the MOTA plot on the whole sequence (dark bold line) and the MOTA plots calculated for each video segment (colored lines). As the observations are accumulated, all the identities are progressively learned and MOTA keeps stable despite that the number of identities has been increased of one order of magnitude with respect to the individual sequences. The MOTA fluctuations due to the insufficient information that were observed at the beginning of each sequence are no more present and the MOTA score is higher than the MOTA of the individual video segments in most cases. In Episode 4 (*e04*), a high number of new identities joins and MOTA is temporarily lower due to the increased complexity of learning.

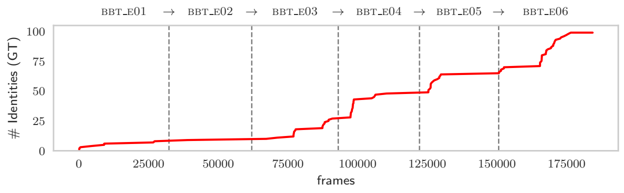
2.6.4 Ablation Study

To demonstrate the effectiveness of the solution we conduct an ablation study in comparison with a baseline in which identities are never explicitly deleted when they exit the field of view. The Baseline uses memory and Reverse Nearest Neighbour as IdOL,and randomly forgets features when a memory budget is met (three different budget values $|\mathcal{M}|$ are evaluated: 100, 500, 1000 and 1500 elements). The Baseline does not use the learning mechanism of Eq.2.3. Since features are randomly deleted in the baseline to maintain the memory budget, performance values are averaged over 100 tests.

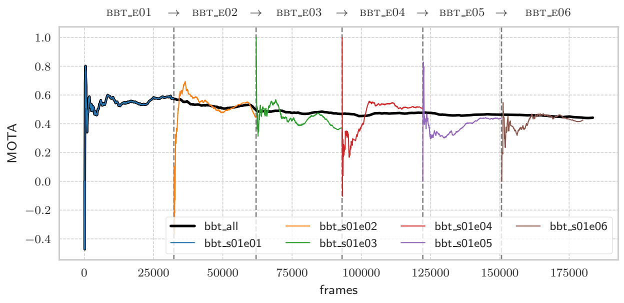
We also consider the effect of different representations, using the features discussed in section 2.5. The *QMUL multi-face dataset* is used for the ablation as it provides specific types of appearance variation of the four subjects. Effects of viewing conditions are considered by evaluating performance for the three videos of the dataset, separately and concatenated in different order. This latter allows to evaluate cumulative learning with curricula of different complexity. The average performance values are also reported in both cases. Results for the three videos of the dataset, separately evaluated are shown in Tab. 2.10a. For the Baseline, the performance increases with the size of the memory and with the quality of the feature representation. The IdOL method scores the best results overall with a sub-



(a)



(b)



(c)

Figure 2.11: *Big Bang Theory* 2 hours sequence (184298 frames) obtained by linking the videos of the 6 Episodes: *(a)* IdOL number of features in memory; *(b)* ground-truth number of identities (cumulative); *(c)* IdOL MOTA computed at each frame for the whole sequence (black bold line) and each Episode (colored lines).

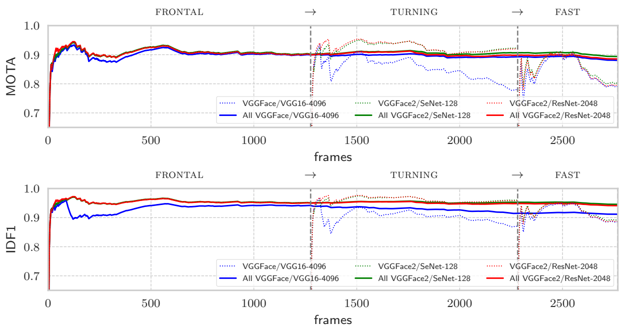
stantially lower number of features stored in the memory. The difference is particularly evident in the case of the VGGFace2/SeNet-128 feature representation. In this case the number of features in memory is one order of magnitude smaller than with VGGFace/VGG16-4096. VGGFace2/SeNet-128 and VGGFace2/ResNet score the best performance. This depends on the CNN architecture used, the VGGFace2 training dataset and on the lower feature dimension. Since the IdOL matching procedure is based on distance ratio, under reasonable assumptions of feature distribution, the distance ratio between the nearest and the farthest neighbors to a given features in high dimensional space is almost 1 [128, 129] so making the criterion less discriminative than it is in a lower dimensional space.

Results for learning curricula of different complexity obtained by concatenating the three videos of the *QMUL multi-face dataset* (FAST → FRONTAL → TURNING, FRONTAL → TURNING → FAST and TURNING → FRONTAL → FAST), are shown in Tab. 2.10b and Fig. 2.12. Results in Tab. 2.10b confirm the conclusions derived for Tab. 2.10a. Fig. 2.12 compares IDF1[6] and MOTA of the three curricula with IDF1 and MOTA of the individual sequences (bold and dotted lines, respectively). It clearly appears the increase of both MOTA and IDF1 by cumulative learning from the previous sequences. Features learned from the VGGFace2 dataset (green and red) show better performance than features trained with the VGGFace one (blue). With gradual increasing of complexity of the curriculum FRONTAL → TURNING → FAST in Fig. 2.12a, IdOL is however able to improve the performance of the TURNING sequence also on the weakest VGGFace/VGG16-4096 representation. On the contrary, as shown in Fig. 2.12b, starting with TURNING does not improve the performance on the FRONTAL. As shown in Fig. 2.12c, similar behaviour of Fig. 2.12a is observed when the FAST video sequence is moved to the the beginning.

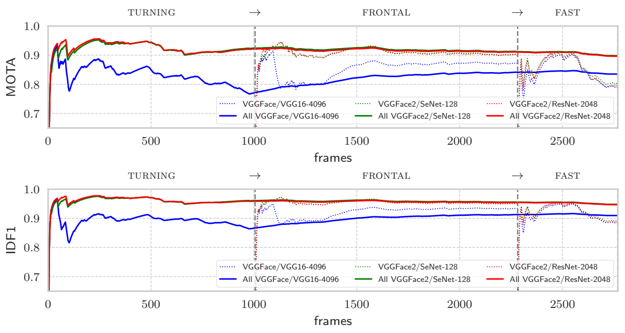
Finally Fig. 2.13 shows a direct comparison between the Baseline with $|\mathcal{M}| = 500$ and IdOL both using the VGGFace2/SeNet-128 feature representation. IdOL and the Baseline concluded the processing with 392 and 500 elements, respectively showing that with a comparable number of features, our method does not change substantially MOTA and IDF1 over time.

The performance evaluations we reported use parameters values that correspond to typical conditions observed in most real sequences. We use the

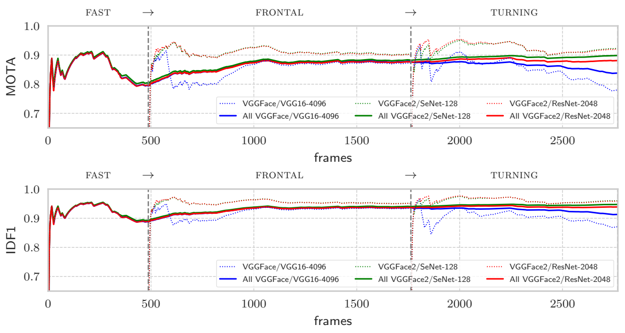
[6]IDF1 is the ratio of correctly identified detections over the average number of ground-truth and computed detections [130].



(a)



(b)



(c)

Figure 2.12: Effects of cumulative learning: performance plots of MOTA and IDF1 over the three curricula (bold) and the individual sequences (dotted) of the *QMUL* multi-face dataset. Plots are shown for different feature representations: VGGface/VGG16 (blue), VGGFace2/SeNet (green) VGGFace2/ResNet (red). Learning performance shows some dependency on initial learning: in the end, learning is clearly improved with curricula starting with the Frontal and Turning sequence; does not improve when curricula starts with the Fast. Effects of the quality of features are clearly evident.

Table 2.10: Ablation study: Baselines with fixed memory budget (100, 500, 1000, 1500 features) versus the IdOL method. MOTA IDF1 and IDS are evaluated for different feature representations (VGGFace/VGG16, VGGFace2/SeNet, VGGFace2/ResNet). (a) performance values over the three sequences of the *QMUL multi-face dataset*; (b) Performance values evaluated over the three curricula. Average performance values of the three cases are also presented.

Feature Representation		FAST				FRONTAL				TURNING				AVERAGE		
Dataset/Architecture	Dim	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS
BASELINES (Tiny Face Detector)																
VGGFace/VGG16	4096	67.526	0.683	10	100	66.261	0.727	16	100	41.502	0.509	11	100	58.430	0.640	12.3
VGGFace2/SeNet	128	76.005	0.750	6	100	63.996	0.683	22	100	43.165	0.513	14	100	61.055	0.649	14.0
VGGFace2/ResNet	2048	76.401	0.745	9	100	64.310	0.697	18	100	50.227	0.618	14	100	63.646	0.687	13.7
VGGFace/VGG16	4096	77.572	0.709	13	500	84.046	0.849	26	500	70.758	0.729	36	500	77.459	0.762	25.0
VGGFace2/SeNet	128	77.258	0.759	11	500	83.073	0.850	38	500	74.979	0.775	32	500	78.436	0.795	27.0
VGGFace2/ResNet	2048	76.518	0.751	17	500	83.987	0.854	38	500	75.839	0.779	43	500	78.781	0.794	32.7
VGGFace/VGG16	4096	78.053	0.706	12	1000	84.670	0.859	25	1000	72.358	0.737	36	1000	78.361	0.767	24.3
VGGFace2/SeNet	128	77.328	0.765	14	1000	86.261	0.901	54	1000	87.659	0.878	68	1000	83.749	0.848	45.3
VGGFace2/ResNet	2048	76.466	0.753	22	1000	86.375	0.887	49	1000	86.892	0.847	66	1000	83.244	0.829	45.7
VGGFace/VGG16	4096	77.989	0.706	12	1123	87.925	0.881	28	1500	73.752	0.743	35	1500	79.889	0.777	25.0
VGGFace2/SeNet	128	77.328	0.765	14	1123	86.292	0.907	55	1500	87.859	0.882	66	1500	83.826	0.851	45.0
VGGFace2/ResNet	2048	76.466	0.753	22	1123	86.389	0.891	54	1500	88.575	0.859	68	1500	83.810	0.834	48.0
IdOL (Tiny Face Detector)																
VGGFace/VGG16	4096	79.138	0.886	4	785	87.152	0.933	6	1501	78.079	0.871	20	1705	81.457	0.897	10.0
VGGFace2/SeNet	128	**80.259**	**0.892**	**0**	203	**90.104**	**0.951**	**2**	321	**92.323**	**0.960**	**5**	323	**87.562**	**0.934**	**2.3**
VGGFace2/ResNet	2048	79.483	0.888	**0**	305	90.007	0.950	**2**	464	92.032	0.959	**5**	576	87.174	0.932	**2.3**

(a)

Feature Representation		FAST→FRONTAL →TURNING				FRONTAL →TURNING →FAST				TURNING →FRONTAL →FAST				AVERAGE		
Dataset/Architecture	Dim	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS	$\lvert\mathcal{M}\rvert$	MOTA	IDF1	IDS
BASELINES (Tiny Face Detector)																
VGGFace/VGG16	4096	29.218	0.402	15	100	42.929	0.504	23	100	24.886	0.322	15	100	32.344	0.409	17.7
VGGFace2/SeNet	128	30.981	0.397	16	100	43.926	0.493	32	100	31.822	0.381	24	100	35.576	0.424	24.0
VGGFace2/ResNet	2048	19.069	0.265	14	100	44.947	0.510	27	100	39.055	0.483	29	100	34.357	0.419	23.3
VGGFace/VGG16	4096	57.070	0.598	50	500	66.803	0.719	53	500	60.892	0.653	64	500	61.588	0.657	55.7
VGGFace2/SeNet	128	45.376	0.485	47	500	67.436	0.729	63	500	63.587	0.662	61	500	58.800	0.625	57.0
VGGFace2/ResNet	2048	37.857	0.418	52	500	69.080	0.739	73	500	62.747	0.680	74	500	56.561	0.612	66.3
VGGFace/VGG16	4096	47.313	0.496	46	1000	69.795	0.741	61	1000	61.828	0.665	64	1000	59.645	0.634	57.0
VGGFace2/SeNet	128	45.610	0.532	83	1000	80.956	0.835	132	1000	83.916	0.842	152	1000	70.161	0.736	122.3
VGGFace2/ResNet	2048	40.397	0.469	75	1000	79.600	0.804	115	1000	76.103	0.775	130	1000	65.367	0.683	106.7
VGGFace/VGG16	4096	49.792	0.522	52	1500	77.317	0.781	86	1500	61.483	0.660	68	1500	62.864	0.654	68.7
VGGFace2/SeNet	128	48.073	0.557	90	1500	85.377	0.876	146	1500	84.866	0.855	160	1500	72.772	0.763	132.0
VGGFace2/ResNet	2048	46.373	0.540	93	1500	85.626	0.843	144	1500	85.096	0.838	167	1500	72.365	0.740	134.7
IdOL (Tiny Face Detector)																
VGGFace/VGG16	4096	83.828	0.913	26	2436	88.049	0.912	78	2376	83.473	0.909	26	2465	85.117	0.911	43.3
VGGFace2/SeNet	128	**89.850**	**0.947**	14	381	**89.380**	**0.946**	**7**	367	**89.792**	**0.948**	**6**	392	**89.674**	**0.947**	9.0
VGGFace2/ResNet	2048	88.118	0.939	**12**	694	88.531	0.941	8	713	89.632	0.947	**6**	660	88.760	0.942	**8.7**

(b)

following values: $\bar{\rho} = 1/1.6 = 0.625$. This setting has the following interpretation: in order to asses the match, the second nearest neighbour must

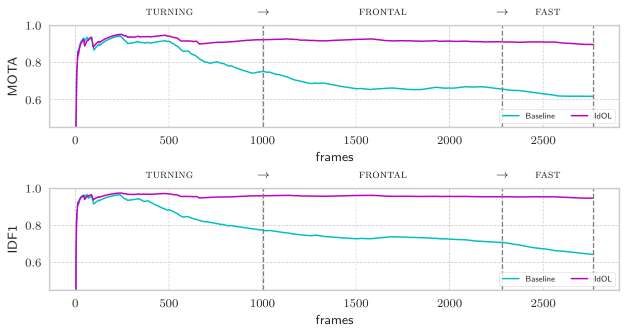


Figure 2.13: Effects of cumulative learning: performance plots of MOTA and IDF1 over the Turning → Frontal → Fast curriculum for the Baseline with 500 features memory budget (light green) versus the IdOL method (392 features). In the IdOL method MOTA and IDF1 soon reach the maximum value and keep stable over time, despite of the lower number of features in memory.

be 1.6 times more distant than the first nearest neighbor. The eligibility threshold $\bar{e}$, used to delete a feature, is set according to the length of the processed video as the length have a direct impact on feature diversity on the memory module. We set: $\bar{e} = 0.5$ for all the videos in the *Music* dataset and *QMUL* multiface-dataset, $\bar{e} = 0.9$ for each single *BBT* video, $\bar{e} = 0.99$ for the concatenated *BBT*. The value α in Eq. 2.3 is set to $\alpha = 0.001$ for all the datasets.

2.6.5 Computational Issues

An evident drawback of ReNN is that in practice a huge number of features (those accumulated in the memory module) is matched against a relatively small set of features (those observed in the current image). Due to that, deciding matching by sorting is prohibitively expensive and tree-based data structures [131] cannot be used effectively since the number of the prototypes in the image is orders of magnitude smaller than the number of prototypes in the memory. However, this drawback can be easily solved by computing the first and second minimum distance through consecutive applications of linear search with GPU implementation. In this way, we can exploit the very efficient CUDA matrix multiplication kernel for the computation of the squared distance matrix and GPU parallelism [132]. Fig. 2.14 shows scaling of performance on Intel i7-2600K 3.40GHz and Nvidia Geforce Titan X as a function of the number of features in the memory module. It is evident that, using GPU, performance keeps almost constant as the number of features in memory increases. With such hardware support, the full system operates on-line at 8 frames per second with 800x600 video frame resolution.

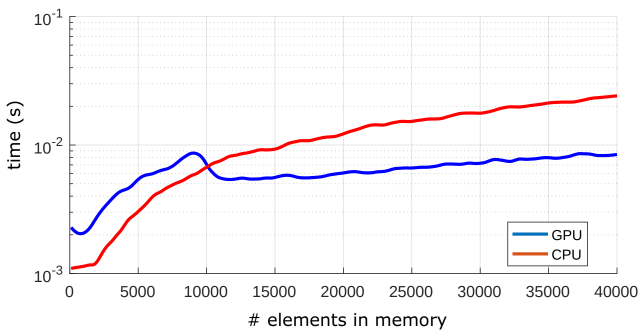


Figure 2.14: Average processing time of Reverse Nearest Neighbor as a function of the number of features in memory: on Intel i7-2600K 3.40GHz and NVIDIA Geforce Titan X GPU.

2.7 Conclusions

In this Chapter, we have presented a novel solution for cumulative learning face identities in unconstrained video streams based on face appearance. We discussed the substantial differences between our learning setting (referred as MOCAL, Multiple Object Cumulative Adaptation Learning), Multiple Object Tracking and Continual Learning when applied to video streams. Our solution updates a representative dataset and use it as a memory of all the past visual information observed so far. This strategy enables the accumulation and preservation of essential knowledge and at the same time allows to handle the non-stationarity of the data stream. We have shown that the proposed method is theoretically sound, asymptotically stable and operates online. Its effectiveness has been demonstrated in comparison with Multiple Object Tracking methods over public datasets. We showed that the method is capable of cumulative learning effectively over long unconstrained video sequences. The method can be applied in principle to any other context for which a detector/feature combination is available (i.e. vehicle, person, boat, traffic sign).

Chapter 3

Open set recognition for unique person counting via virtual gates

Retail shops or restaurants are interested in real-time profiling analysis of customer visit patterns, which could enable efficient management and target marketing. They need to know not only how many people entered but also if they are visiting for the first time and keep track of their exact number. As a result, in this chapter we define the new variant of unique counting *for videos, that is counting new persons who have not already been counted in the past. To this end, we propose a complete real-time system which is able to perform detection, tracking and unique counting in the wild with user drawn gates. A fine-tuned network on persons body is used to extract descriptors which are more privacy-oriented. Experiments of the system on the challenging DukeMTMC dataset show that our method is able to effectively count people in real time and discern between the persons which do multiple passages through the gates.* [1] [2]

[1]The part of this chapter has been published as "Open set recognition for unique person counting via virtual gates" in *ICIAP: International Conference on Image Analysis and Processing, 2019 Trento* [133]

[2]*Acknowledgments:* tThis research was partially supported by NVIDIA Corporation with the donation of Titan X Pascal GPUs and Leonardo Finmeccanica SpA, Italy.

3.1 Introduction

In recent times there is a great interest in computer vision for monitoring all types of environments. Many goals are impacted by new technological advances in video analysis, e.g., security, resource management, urban planning, or advertising. Of these technologies, counting people passing through a place is a fundamental problem. It is an essential building block for crowd analysis that is useful for several different applications, including crowd monitoring [134], scene understanding [135], surveillance [136] and customer analysis [137]. In particular, retail shops or restaurants are interested in real-time profiling analysis of customer visit patterns, which could enable efficient management and target marketing. They need to know not only how many people entered but also if they are visiting for the first time and keep track of their exact number.

As a result, in this chapter we define the new variant of *unique counting* for videos i.e. counting new persons who have not already been counted in the past. At a high level, it requires to detect persons, remember and match previously observed individuals. The main challenge is the open-world setting: for each person detected the system must be able to tell if he is in the set of already known persons or if he is a new person, which is a very hard task and requires a memory based algorithm [138]. The task is different than person re-identification, which is usually performed on images and require only to match query to gallery persons. It is also different than multi-target tracking where it is required to track all people across the scene but does not address counting passages through an area or virtual line.

In the literature, people counting can refer to multiple different settings [134], from counting the number of instances in a single image to counting how many persons crossed a virtual gate or an area. Similarly to [136, 139], in this chapter we address counting by defining virtual gates. These are imaginary lines where the actual counting is made (see Fig. 3.1) and can be drawn freely by the user over the frame. Hence, we perform counting in the realistic setting commonly referred as “in the wild”, i.e. realistic footage, including conditions contaminated by blur, non-uniform lighting, and non frontal pose. Moreover, multiple gates can be defined per single cameras, allowing monitoring of multiple entrances.

The contributions of this work are three-fold: *i.* we propose the task of unique counting which is a variation of counting task; *ii.* we propose a complete real-time system which is able to perform detection, tracking

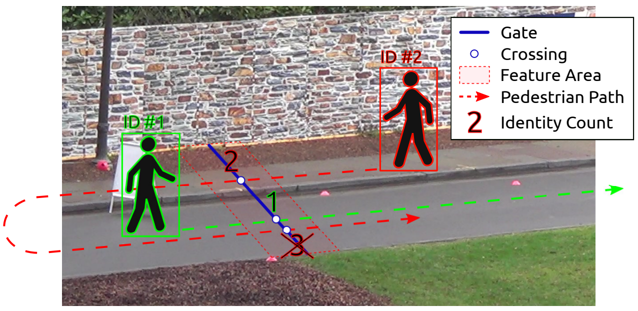


Figure 3.1: **The unique counting task.** The task is to count the number of unique people who cross the virtual gate, drawn in blue. In this example, person #2 crosses the gate twice but he is only counted once.

and unique counting in the wild with user drawn gates, and *iii.* we report experiments on the challenging DukeMTMC dataset [38] showing that our method is able to effectively count people in real time and discern between the few persons which do multiple passages through the gates.

In Section 3.3 we describe our method in detail, including the various parts of our pipeline; Section 3.4 reports experiments on our adaptation of DukeMTMC to the task of unique counting and finally in Section 3.5 our conclusions are reported.

3.2 Related Works

The problem of unique people counting is mainly related to the topics of people counting and open world person re-identification.

3.2.1 People Counting

Vision-based people counting systems have become more popular in recent years, which allow counting in different scenarios [134, 140]. Two different research directions are pursued: spatial people counting and temporal people counting.

Spatial people counting Works in this direction aim at counting the exact number of people who are present in a given image or video frame. The main difficulty of this task is related to detecting persons, taking careful attention to occlusions of people and their different appearance when in a crowd.

Detection based methods typically employ object detection methods specialized to detect people [141]. They train classifiers using features such as Haar wavelets, histogram oriented gradients [141] and more recently with convolutional neural network based methods [134]. To increase robustness to occlusions, Li *et al.* [142] propose to use head and shoulder detectors, which are more distinct than full body. Xu *et al.* [143] add a tracker to reject false pedestrian detections. However, head and shoulder detectors make re-identification more difficult since they usually cover few pixels and the face can be in the opposite direction than the camera.

Temporal people counting These works are applied to videos and aim at counting the number of people who enter the recorded area, pass through a virtual line or a forced passage.

For areas where people flow can be forced, several researchers proposed to use overhead mounted depth cameras. They can accurately count people flowing through an entrance [137]. Nonetheless, they need proper installation and cannot be employed in other environments. For open areas where people can freely move, tracking of people is usually performed exploiting the entire body [144] or the face [145]. Since counting is often employed along waypoints or streets, a popular solution is using a virtual gate where only detection passing thorough are recorded [136,139]. For instance, Liu *et al.* [139] use segmentation to partition groups of people into individuals and individually track their movement crossing the gate. They track people in trajectories by formulating pedestrian hypotheses that are filtered and combined into accurate counting events. In [136], a surveillance system based on recent object detector YOLO [146] exploits an intersection over union tracker to count people who cross a gate.

This work is related to temporal people counting and is based on the use of a virtual gate. However, differently from all these works, we address the task of unique person counting which require re-identification of people who pass through the gate. We use the pedestrian appearance in a free context where a user can freely draw the counting gate. Previous work that perform re-identification uses only overhead mounted camera or face information which is limited to high resolution cameras and may not be privacy compliant.

3.2.2 Open world person re-identification

The proposed method needs to discriminate between already known and unknown persons as in the open-world setting [81]. The majority of person re-identification methods focus on closed world scenarios using discriminative view-invariant features or learning matching distance metrics [147]. Only recent work considers open-set person re-identification [148], first in small scale with basic features and distance learning [149], then with deep learning features in an end-to-end manner [150]. Nonetheless, scaling to large scale is still an open issue where only very recent work tries to address it, for instance, with hashing [151].

Differently from these methods, our approach do not need to explicitly maintain the full appearance of a person. We are only interested in the appearance at the gate proximity, which permits to reduce the uncertainty of the open-world setting and allows scaling with few resources.

3.3 Unique counting system

Given a video stream, the proposed system aims to count the number of unique individuals crossing one or more *gates*. A gate is an imaginary line drawn by a user where the system has to count people, usually used to delimit a part of the scene from another.

Differently from the task of counting [134], unique person counting requires the re-identification of persons in open-world setting [138]. The system, beside detecting when a person crosses a gate, needs also to detect if an instance has already crossed a gate in the past to avoid counting it multiple times. It starts with no knowledge of the persons that will cross the gates, so it has to memorize a discriminative representation of each new person as it sees them.

For person re-identification one approach is to extract face features since they are strongly discriminative [145]. However, their use limit where the system can be applied due to technical requirements and privacy. Face detections should be at least of a minimum size to be discriminative, forcing to employ cameras with high resolutions, mounted to observe people facing the camera. Hence people cannot be recognized in both directions. Moreover, being a sensitive information, the acquisition of faces without explicit consent can raise privacy concerns. For these reasons, we propose to exploit body related features. Body features are not as discriminative as face features but they can be extracted in every pose and hardly poses any privacy concern.

The proposed system is composed of four submodules, shown in Fig. 3.2: *i.* a person detector to identify pedestrians in the frame; *ii.* a tracker to track the trajectory of each pedestrian and detect gate crossings; *iii.* a module to extract body features, and *iv.* a re-identification module that allows to recognize previously observed people. In the following sections we will explain in details each module.

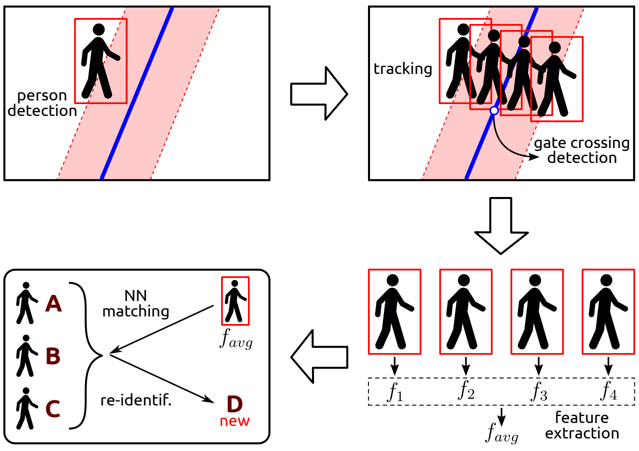


Figure 3.2: **Pipeline of the proposed system.** Given a frame, persons are detected in the scene and tracked when they come near a gate. Upon gate crossing, features of the person in the red area near a gate are extracted and used to perform open-world re-identification.

3.3.1 Person Detector

The person detector module is responsible of detecting pedestrian in the scene. We test two different state of the art methods which allow to process videos in real-time, with different settings. The first method we employ is a YOLO v3 network [146] which is a single-stage object detector that process an image in a single pass and generates a set of boxes with an associated probability. The method exploits a fully convolutional architecture where the last layer uses 1×1 filters to output a fixed amount of windows with different confidence. We used a network trained on the 80 classes in COCO dataset with a TensorRT implementation but detections are only taken for the class of *person*.

For this module, we also test the recent OpenPose detector [152] which is the first real-time multi-person system to jointly detect human body and its parts from single images. OpenPose exploits a sequential architecture composed of convolutional networks that directly operate on belief maps from previous stages. Part locations are increasingly refined without the need for explicit graphical model-style inference. This method emits 25 keypoints that encode the pose of each person detected, from which we derive a bounding box. More specifically, we first split these keypoints to identify head, upper body and lower body separately. Then we ensure that the three blocks that define the human body satisfy human body proportions (i.e. body height should be about 7×head height and body width should be about 2×head height). If we miss at most two of the three boxes due to keypoint absence or to low keypoint score, we can derive them by exploiting body proportions and the measures of the available ones. Finally, we take the box that tightly encompasses the three main human body parts obtained in this way. This makes our detections more robust, especially in borderline situations like cases where there are persons overlapping or partial occlusions.

3.3.2 Tracking and gate crossing detection

Tracking detections allows the system to track movement of pedestrians and understand when they are crossing virtual gates. Each person is represented using a bounding box with its location. The detected boxes are joined together into tracks using a tracking by detection strategy, grouping bounding boxes in consecutive frames by looking at their Intersection over Union (IoU) and the optical flow estimation.

We only evaluate tracks around the gates, so that we can reduce the risk of incurring in tracking errors. We monitor the distance of the middle point of the bottom segment of the bounding box (ideally the point between the feet) from the gate line. Only when distance of the box is less than K pixels (that we empirically set to 100) from the gate, it is tracked.

The tracker continuously monitors all pedestrians in the tracks. At each frame a set of new detections is produced and we update the tracker state by associating each track to every detection, if possible. For unassociated detections we start new tracks. We employ a greedy association approach.

At each frame we get a set of detections D_t; given the set of tracks T_{t-1} detected at the previous frame, we compute an association matrix $\mathbf{A}$ based on IoU, such that $\mathbf{A}_{ij} = \frac{d_i \cap t_j}{d_i \cup t_j}$. Then we apply the function *track* described in Algorithm 2.

Algorithm 2: Data association algorithm. We associate tracks and unassociated detection if `IoU` $> \tau$ and remove a track if it is "dead" for ω frames. Matrix $\mathbf{A}$ keeps track of associations and vector $\mathbf{l}$ counts the amount of frames a track i is not associated with any detection.

```
1  FUNCTION track ($\mathcal{D}_t, \mathcal{T}_{t-1}$)
   Data: $\mathcal{T}_{t-1} : \{t_1 \dots t_n\}, \mathcal{D}_t : \{d_1 \dots d_m\}, \mathbf{A}_{ij} = \frac{d_i \cap t_j}{d_i \cup t_j}$
   Result: $\mathcal{T}_t$
2  while $\max_{ij} \mathbf{A}_{ij} > \tau$ do
3      if not $\mathbf{K}_{ij} \wedge \mathbf{A}_{ij} > \tau$ then
4          $\langle \hat{i}, \hat{j} \rangle \leftarrow \arg\max_{ij} \mathbf{A}_{ij}$;
5          $t_{\hat{i}} \leftarrow d_{\hat{j}}$  $\mathbf{K}_{\hat{i}:} \leftarrow$ TRUE;
6          $\mathbf{K}_{:\hat{j}} \leftarrow$ TRUE;
7      end
8  end
   /* Tracks not used for $\gamma$ frames are removed.                */
9  $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} \setminus \{t_i | \mathbf{l}_i > \gamma\}$;
   /* Unassigned detections initialize new tracks.                    */
10 $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} \cup \{d | \mathbf{K}_{ij} =$ TRUE$\}$
```

The procedure generates the paths, i.e. a sequence of points followed by people on the scene. Gate crossing detection is performed by testing segment intersection between the sequences of points from each track and a given line of the gate. We address it as a segment intersection problem of 2^{nd} degree, also known as the orientation test which robustness has been studied in [153]. Given two line segments $A = (s_1, e_1)$ and $B = (s_2, e_2)$, we can test if they intersect by checking the orientations of the ordered triplets formed by the four points (s_1, e_1, s_2, e_2). There is intersection between A and B if (s_1, e_1, s_2) and (s_1, e_1, e_2) have different orientations and similarly (s_2, e_2, s_1) and (s_2, e_2, e_1) have different orientations. If points are collinear, we handle this case by fitting a line through A and B and checking that the angle is around 0 or 180 degrees. By looking at the orientations of the triplets we can also understand in which direction the intersection occurred.

In case of multiple gates per camera, by tracking a person that is moving across the scene we can re-identify the tracks who crosses multiple gates. In that case, the system can directly ignore the following intersections, without further advancing with the pipeline.

3.3.3 Body feature extractor

The body feature extractor module receives from the tracker those tracks that cross a gate and extract a characteristic representation of each person. We fine-tuned and tested two ResNet50 [125] networks pre-trained on the ImageNet dataset as body feature extractors. To this end, we chose two datasets which are popular for the task of person re-identification and that contain challenging visual conditions of people. The first net, named *ResNet50-Market*, is fine-tuned on the Market1501 dataset [154], while the other, *ResNet50-Duke* on the DukeMTMC-reID dataset [155]. For ResNet50-Market, we use the full training set of the Market1501 dataset which has 6 cameras and contains 32,668 annotated bounding boxes of 1,501 identities. For ResNet50-Duke, we use the full training set of the DukeMTMC-reID dataset, which is a subset of the DukeMTMC dataset, where 1,404 identities are selected and for whom 36,411 bounding boxes are extracted, sampling the videos every 120 frames. The resulting fine-tuned networks are evaluated on the respective validation sets, obtaining a mAP of 79.1% and a Rank-1 of 91.8% and a mAP of 59.4% and a Rank-1 of 77.2% respectively. In both cases the results, specifically the Rank-1 figures, show that the fine-tuned networks exhibit good re-identification capabilities.

For both network, each bounding box of a person is resized to the fixed size of 128x256 pixels as network input. The body feature is obtained by taking as output the penultimate layer feature maps of dimension 2048 normalization.

Given a track K that cross a gate, the module extracts the bounding box of the person and a feature f_i for each one of the $M = 90$ (empirically set) frames before and after the crossing, using one of the fine-tuned networks. As a result we obtain a set of feature per person crossing $F = \{f_{-45}, \dots, f_{45}\}$. We obtain the final feature P_K by applying average pooling to the set following L2 normalization.

3.3.4 Re-Identification

This module is responsible to keep track of the known identities and increment the counter of each gate when a new person is crossing. To this end, the module maintain a set of features $\mathcal{M}$ for each gate where new person features are stored upon crossing. To detect if a person was saw in the past we implement a simple distance strategy. Given a feature P_K, we compute its cosine distance $< P_K, M_i >$ from all features M_i in $\mathcal{M}$. When $M_i > \eta$, with η cross-validated on the training set, we consider the person as new. In that case the module increments the counter of the gate and adds P_K to $\mathcal{M}$. The number of known identities corresponds to the count of unique persons that have crossed each gate and is the final output of the system.

3.4 Experiments

In this section we report our experiments of the whole system and its components. We first describe the dataset used and the experimental settings. Then we report the experiments of our system on the unique counting task.

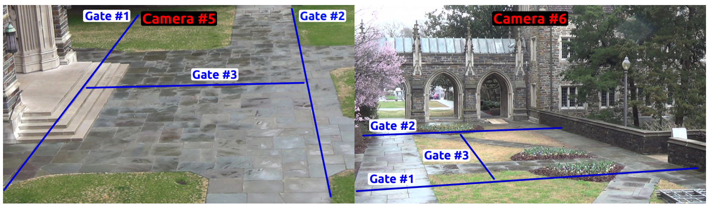


Figure 3.3: **Gates location.** We drawn 3 gates per camera to count the flow of people along the principal directions. Note that the system allow a user to freely drawn them as many as needed.

3.4.1 Dataset and ground truth

We used the challenging DukeMTMC dataset [38]. The dataset is comprised of 8 static camera recordings of the Duke University campus. Each recording consists of roughly 85 minutes of 1080p 60fps video footage with more than 2,000,000 manually annotated frames for multi-target tracking, 7,000 single camera trajectories and more than 2,000 identities. Identities have been manually annotated with the respective bounding box and unique ID. They follow unconstrained paths, moving between different cameras.

3.4.2 Experimental Setup

From the 8 cameras of the DukeMTMC dataset we selected 2 cameras for testing the system, specifically number 5 and 6 as they feature the most number of identities exiting from the scene and later returning. We used the train_val sequence as ground truth annotations are available for this one, while they are not provided for test sequences. We cross-validated the cosine-distance threshold η and the M frame length on camera 4.

For each scene we place 3 gates which cover all the main directions a person can go. Images of the gates for each scene are shown in Fig. 3.3.

Naturally the dataset does not come with unique counting ground truth, but we can generate it starting from its multi-target annotations. To this end, identity annotations are used to identify which gate is crossed and by how many persons. Counting is made when a person crosses one of the gates. Subsequent crossing of the same or any other gate in the scene is ignored. We use the resulting number as ground-truth for our method.

3.4.3 Results

For assessing the performance of the system, we first test the proposed tracker with the two detectors and then we test the performance of the whole system comprised also of feature extraction and re-identification.

Table 3.1: Counting results using the baseline approach.

	Cam. 5	Cam. 6
YL + Tracker	785	1187
OP + Tracker	644	1070
GT + Tracker	459	728
GT Unique	431	725

Detector + Tracker For the first experiment, we test the tracker only in absence of feature extractor and re-identification modules. The detector and tracker can perform unique counting in presence of more than one gate

Table 3.2: Counting results using the full system. R50 represents the ResNet50 network.

	Cam. 5	Cam. 6
YL + Tracker + R50-Market	561	778
OP + Tracker + R50-Market	533	762
GT + Tracker + R50-Market	501	741
YL + Tracker + R50-Duke	438	752
OP + Tracker + R50-Duke	434	748
GT + Tracker + R50-Duke	432	730
GT Unique	431	725

by checking if a track intersects more multiple gates. We use this method as baseline, named YL + Tracker when using the YOLO detector and OP + Tracker when using OpenPose. For reference, we also measure the tracker performance only by using the ground truth boxes as detector. We name this combination GT + Tracker.

We report the resulting persons counted by the baseline in Table 3.1 and compare the methods to the ground truth person counting (GT Unique). We observe that GT + Tracker and GT Unique are very narrow in gap. We note that both cameras exhibit the same observations. Given a perfect detector, our tracking method is able to obtain a very good result with only the 6.4 % of error, confirming that the tracker can effectively discern when the same person cross multiple gates. Looking at YL + Tracker and OP + Tracker we observe that the error is higher, while the latter has a slightly more correct result. This suggest that detector and tracking alone are not sufficient to perform the hard task of unique counting due to missing detections and tracker not able to completely recover from such issues. OpenPose result in more coherent detections as expected. Our tracker may miss some identities due to occlusions or persons abandoning the scene and re-entering later. This leads to double-counting, as re-entering persons would be considered new identities. At the same time, overlapping between tracks may lead to identity swap and thus to erroneous counting.

Full system In this experiment we test the complete system, that is the four modules including feature extraction and re-identification, with the two fine-tuned networks (ResNet50-Market and ResNet50-Duke). The full system run at about ~12.2 FPS. We report in table 3.2 the persons counted by the whole system in the various combinations. We can see that although having achieved a lower Rank-1 for re-identification, fine-tuning on the same domain dataset yields better results. In fact, between the various combinations, we observe that the best combination is OP + Tracker + ResNet50-Duke, resulting in a near perfect result. Comparing Table 3.1 and 3.2, we note that by adding the last two modules, the system is able to recognize more passed people and outputs a counter more near the ground truth. This confirm that our re-identification approach is able to reduce the false counting by re-identify the track of same persons.

3.5 Conclusions

In this chapter we proposed a system to perform the variant of unique counting, that is counting the unique persons which crosses a user drawn gate. The system is able to detect persons, track them when they are near a gate and crosses it. We perform open-world re-identification on the body feature, by exploiting fine-tuned features that we trained on Market and DukeMTMC-reid datasets. Experiments on the challenging DukeMTMC dataset showed that our system is able to effectively count people passing through the gates in real time and recognize already passed people.

Chapter 4

Regular Polytope Networks

Neural networks are widely used as a model for classification in a large variety of tasks. Typically, a learnable transformation (i.e. the classifier) is placed at the end of such models returning a value for each class used for classification. This transformation plays an important role in determining how the generated features change during the learning process. In this work we argue that this transformation not only can be fixed (i.e. set as non trainable) with no loss of accuracy and with a reduction in memory usage, but it can also be used to learn stationary and maximally separated embeddings. We show that the stationarity of the embedding and its maximal separated representation can be theoretically justified by setting the weights of the fixed classifier to values taken from the coordinate vertices of the three regular polytopes available in $\mathbb{R}^d$, namely: the d-Simplex, the d-Cube and the d-Orthoplex. These regular polytopes have the maximal amount of symmetry that can be exploited to generate stationary features angularly centered around their corresponding fixed weights. Our approach improves and broadens the concept of a fixed classifier, recently proposed in [156], to a larger class of fixed classifier models. [1] [2]

[1]Part of this chapter has been published as "Regular Polytope Networks" to *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* [157]

[2]Part of this chapter has been published as "Maximally compact and separated features with regular polytopenetworks" in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* [158]

4.1 Introduction

Deep Convolutional Neural Networks (DCNNs) have achieved state-of-the-art performance on a variety of tasks [159, 160] and have revolutionized Computer Vision in both classification [161, 162] and representation [2, 36]. In DCNNs, both representation and classification are typically jointly learned in a single network. The classification layer placed at the end of such models transforms from the dimension d of the network internal feature representation to the number K of the class categories. Despite the large number of trainable parameters that these layers add to the model (i.e. $d \times K$), their removal only gives a slight increase in error [163]. Additionally, latest architectures achieving improved generalization tend to avoid the use of fully connected layers [164] [165] [125]. Furthermore, it is well known that DCNNs can be trained to perform metric learning without the explicit use of a classification layer [166] [167] [168]. In particular, it has been shown in more detail that excluding the parameters of the classification layer from learning causes little or no decline in performance while allowing a reduction in the number of trainable parameters [156]. Fixed classifiers have also an important role in theoretical convergence analysis of training models with batch-norm [169]. It is shown very recently in [170] that DCNNs with a fixed classifier and batch-norm in each layer establish a principle of equivalence between different learning rate schedules.

All these works seem to suggest that the final fully connected layer used for classification is somewhat redundant and does not have a primary role in learning and generalization. In this Chapter we show *how* a special set of fixed classification layers *has a key role in modeling the internal feature representation* in DCNNs, while still ensuring little or no loss in classification accuracy and a significant reduction in memory usage.

In DCNNs the internal feature representation for an input sample is the feature vector $\mathbf{f}$ generated by the penultimate layer, while the last layer (i.e. the classifier) outputs score values according to the inner product as:

$$z_i = \mathbf{w}_i^\top \cdot \mathbf{f} \tag{4.1}$$

for each class i, where $\mathbf{w}_i$ is the weight vector of the classifier for the class i. To evaluate the loss, the scores are further normalized into probabilities via the Softmax function [171].

Since the values of z_i can be also expressed as $z_i = \mathbf{w}_i^\top \cdot \mathbf{f} = ||\mathbf{w}_i|| \, ||\mathbf{f}|| \cos(\theta)$, where θ is the angle between $\mathbf{w}_i$ and $\mathbf{f}$, the score for the correct label with

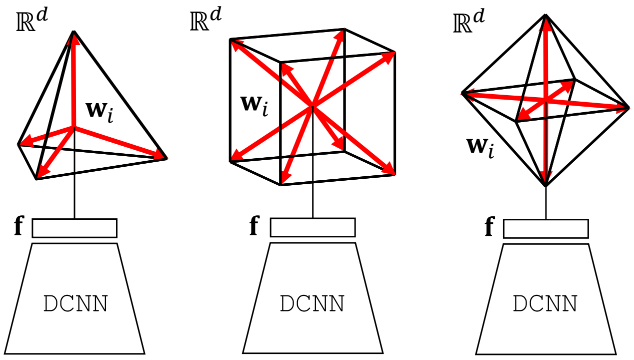
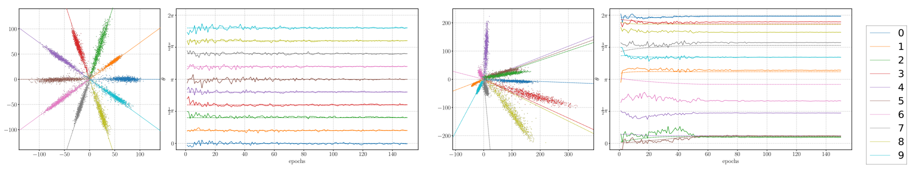


Figure 4.1: Regular Polytope Networks (RePoNet). The fixed classifiers derived from the three regular polytopes available in $\mathbb{R}^d$ with $d \geq 5$ are shown. From left: the d-Simplex, the d-Cube and the d-Orthoplex fixed classifier. The trainable parameters $\mathbf{w}_i$ of the classifier are replaced with fixed values taken from the coordinate vertices of a regular polytope (shown in red).

respect to the other labels is obtained by optimizing $||\mathbf{w}_i||$, $||\mathbf{f}||$ and θ. This simple formulation of the final classifier provides the intuitive explanation of how feature vector directions and weight vector directions align simultaneously with each other at training time so that their average angle is made as small as possible. If the parameters $\mathbf{w}_i$ of the classifier in Eq. 4.1 are fixed (i.e. set as non trainable), *only the feature vector directions* can align toward the classifier weight vector directions and not the opposite. Therefore, weights can be regarded as fixed angular references to which features align.

According to this, we obtain a precise result on the spatio-temporal statistical properties of the generated features during the learning phase. Supported by the empirical evidence in [156] we show that not only the final classifier of a DCNN can be set as non trainable with no loss of accuracy and with a significant reduction in memory usage, but that an appropriate set of values assigned to its weights allows learning a maximally separated and strictly stationary embedding while training. That is, the features generated by the Stochastic Gradient Descent (SGD) optimization have constant mean, angularly centered around their corresponding fixed class weights. Constant known mean implies that features cannot have non-constant trends while learning. Maximally separated features and their stationarity are obtained



(a) (b) (c) (d)

Figure 4.2: Feature learning on the MNIST dataset in a 2D embedding space. Fig. (a) and Fig. (c) show the 2D features learned by RePoNet and by a standard trainable classifier respectively. Fig. (b) and Fig. (d) show the training evolution of the classifier weights (dashed) and their corresponding class feature means (solid) respectively. Both are expressed according to their angles. Although the two methods achieve the same classification accuracy, features in the proposed method are both stationary and maximally separated.

by setting the classifier weights according to values following a highly symmetrical configuration in the embedding space.

DCNN models with trainable classifiers are typically convergent and therefore, after a sufficient learning time has elapsed, some form of stationarity in the learned features can still be achieved. However, until that time, it is not possible to know where the features will be projected by the learned model in the embedding space. An advantage of the approach proposed in this Chapter is that it allows to define (and therefore to know in advance) where the features will be projected before starting the learning process.

Our result can be understood by looking at the basic functionality of the final classifier in a DCNN. The main role of a trainable classifier is to dynamically adjust the decision boundaries to learn class feature representations. When the classifier is set as non trainable this dynamic adjustment capability is no longer available and it is automatically demanded to all the previous layers. Specifically, the work [156] reports empirically evidence that the expressive power of DCNN models is large enough to account for the missing dynamic adjustment capability of the classifier [156]. We provide more systematic empirical evidence confirming and broadening the general validity of DCNNs with fixed classifiers (Sec. 4.5.1).

We show that our approach can be theoretically justified and easily implemented by setting the classifier weights to values taken from the coordinate vertices of a regular polytope in the embedding space. Regular polytopes are the generalization in any number of dimensions of regular polygons and regular polyhedra (i.e. Platonic Solids). Although there are infinite regular polygons in $\mathbb{R}^2$ and 5 regular polyhedra in $\mathbb{R}^3$, there are only three regular polytopes in $\mathbb{R}^d$ with $d \geq 5$, namely the d-Simplex, the d-Cube and the d-Orthoplex. Having different symmetry, geometry and topology, each regular polytope will reflect its properties into the classifier and the embedding space which it defines. Fig. 4.1 illustrates the three basic architectures defined by the proposed approach termed Regular Polytope Networks (RePoNet). Fig. 4.2 provides a first glance at our main result in a 2D embedding space. Specifically, the main evidence from Fig. 4.2a and 4.2b is that the features learned by RePoNet remain aligned with their corresponding fixed weights and maximally exploit the available representation space directly from the beginning of the training phase.

We apply our method to multiple vision datasets showing that it is possible to generate stationary and maximally separated features without reducing the generalization performance of DCNN models and with a significant reduction in GPU memory usage at training time.

4.2 Related Works

4.2.1 Fixed Classifier

Empirical evidence, reported in [172], firstly shows that a convolutional neural network with a fixed classification layer (i.e. not subject to learning) initialized by random numbers does not worsen the performance on the CIFAR-10 dataset. A recent paper [156] explores in more detail the idea of excluding the parameters $\mathbf{w}_i$ in Eq.4.1 from learning. The work shows that a fixed classifier causes little or no reduction in classification performance for common datasets while allowing a noticeable reduction in trainable parameters, especially when the number of classes is large. Setting the last layer as not trainable also reduces the computational complexity for training as well as the communication cost in distributed learning. The described approach sets the classifier with the coordinate vertices of orthogonal vectors taken from the columns of the Hadamard[3] matrix. The paper in question does not investigate on the internal feature representation. A major limitation of this

[3]The Hadamard matrix is a square matrix whose entries are either $+1$ or -1 and whose rows are mutually orthogonal.

method is that, when the number of classes is greater than the dimension of the feature space, it is not possible to have mutually orthogonal columns and therefore some of the classes are constrained to lie in a common subspace causing a reduction in classification performance. We improve and generalize this work by finding a novel set of unique directions that overcomes the limitations of the Hadamard matrix.

The work [173] trains a neural network according to the triplet loss with a set of fixed vertices on a hyper-sphere (i.e. a sphere lattice). The work aims at learning a function that maps real-valued vectors to a uniform distribution over a d-dimensional sphere.

As shown in [170], fixed classifiers are also related to BatchNorm [169] and learning rate schedules. BatchNorm parameterizes the weights of a layer to "normalize" its activations (i.e. the features), however, BatchNorm is not typically applied to normalize the classifier activations (i.e. logits). The work in [170] firstly shows that in the presence of BatchNorm layers and a fixed classifier layer, the $L2$ regularization has the effect of training a Neural Network with an exponentially increasing learning rate.

4.2.2 Softmax Angular Optimization

As originally described in [174], under Softmax loss[4] the label prediction is *largely* determined by the angular similarity to each class since Softmax loss uses cosine distance as classification score. Several papers followed exploiting this intuition to train DCNNs by direct angle optimization [175–178]. The angle encodes the required discriminative information for class recognition. The wider the angles the better the classes are separated from each other and, accordingly, their representation is more discriminative. The common idea of these works is to constrain the features and/or the classifier to be unit normalized.

The works [179], [180] and [178] normalize both features and weights thus obtaining an exact optimization of the angle in Eq. 4.1. Under the only weight normalization label prediction is largely determined by the angular similarity [174] [176]. This is not only because Eq. 4.1 can be e factorized into amplitude and angular component, but also because decision boundaries between adjacent classes are determined by their angular bisectors.

[4]The combination of cross-entropy loss and the Softmax function at the last fully connected layer.

Differently from weight normalization, feature normalization cannot directly perform angle optimization but encourages intra-class compactness of learned features [175]. Specifically, [175] also proposes adding a scale parameter after feature normalization based on the property that increasing the norm of samples can decrease the Softmax loss [178, 181]. Despite not being involved in learning discriminative features, the work [156], in addition to fixing the classifier, normalizes both the weights and the features and exploits the multiplicative scale parameter. In accordance with [156, 175, 182] and [178] we found that feature normalization and the multiplicative scale parameter are hard to optimize for general datasets, having a significant dependence on image quality. According to this, we follow the work [176] that normalizes the classifier weights and sets its biases to zero. A property introduced in [178] and further discussed in [181] shows that setting the classifier bias to zero encourages well-separated features to have bigger magnitudes. This avoids features collapsing into the origin making angles between the fixed weights and features a reliable metric for classification. As further conjectured in [178], if all classes are well-separated, weight normalization will roughly correspond to the means of features in each class. The maximal and fixed separation proposed in this Chapter further strengthens the conjecture producing features more centered around their fixed weights as the training process advances.

Another close related work is [183] in which separability of learned features is improved by injecting a single dynamic virtual negative class into the original softmax. A virtual class is a class that is active in the classifier but has no data available from which to learn. Injecting the virtual class enlarges the inter-class margin and compresses intra-class distribution by strengthening the decision boundary constraint. Due to the fixed classifier we can exploit their result in the case of multiple static virtual classes. This happens when the number of classes does not match the number of vertices of a regular polytope.

While all the above works impose large angular distances between the classes, they provide solutions to enforce such constraint in a local manner without considering global inter-class separability and intra-class compactness. For this purpose, very recently the works [184], [185] and [186] add a regularization loss to specifically force the classifier weights to be far from each other in a global manner. These works draw inspiration from a well-known problem in physics – the Thomson problem [187], where given K

charges confined to the surface of a sphere, one seeks to find an arrangement of the charges which minimizes the total electrostatic energy. Electrostatic force repels charges each other inversely proportional to their mutual distance. In [184], [185] and [186] global equiangular features are obtained by adding to the standard categorical cross-entropy loss a further loss inspired by the Thomson problem. We follow a similar principle for global separability by considering that minimal energies are often concomitant with special geometric configurations of charges that recall the geometry of Platonic Solids in high dimensional spaces [188]. Preliminary and qualitative results of Regular Polytope Networks for compact feature learning have been presented in [158].

4.3 Main Contributions

Our technical contributions can be summarized as follows:

1. We generalize the concept of fixed classifiers and show they can generate stationary and maximally separated features at training time with no loss of performance and in many cases with slightly improved performance.

2. We performed extensive evaluations across a range of datasets and modern CNN architectures reaching state-of-the-art performance. We observed faster speed of convergence and a significant reduction in model parameters.

3. We further provide a formal characterization of the class decision boundaries according to the dual relationship between regular polytope and statistically verify the validity of our method on random permutations of the labels.

4.4 Regular Polytopes and Maximally Separated Stationary Embeddings

We are basically concerned with the following question: *How should the non trainable weights of the classifier be distributed in the embedding space such that they generate stationary and maximally separated features?*

Let $\mathbb{X} = \{(x_i, y_i)\}_{i=1}^{N}$ be the training set containing N samples, where x_i is the raw input to the DCNN and $y_i \in \{1, 2, \cdots, K\}$ is the label of the class that supervises the output of the DCNN. Then, the cross entropy loss can be written as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mathbf{w}_{y_i}^{\top}\mathbf{f}_i + \mathbf{b}_{y_i})}{\sum_{j=1}^{K}\exp(\mathbf{w}_j^{\top}\mathbf{f}_i + \mathbf{b}_j)}\right), \tag{4.2}$$

where $\mathbf{W} = \{\mathbf{w}_j\}_{j=1}^{K}$ are the classifier weight vectors for the K classes. Following the discussion in [176] we normalize the weights and zero the biases ($\hat{\mathbf{w}}_j = \frac{\mathbf{w}_j}{||\mathbf{w}_j||}$, $\mathbf{b}_j = 0$) to directly optimize angles, enabling the network to learn angularly distributed features. Angles therefore encode the required discriminative information for class recognition and the wider they are, the better the classes are represented. As a consequence, the representation in this case is maximally separated when features are distributed at *equal angles* maximizing the available space.

If we further consider the feature vector parametrized by its unit vector as $\mathbf{f}_i = \kappa_i \,\hat{\mathbf{f}}_i$ where $\kappa_i = ||\mathbf{f}_i||$ and $\hat{\mathbf{f}}_i = \frac{\mathbf{f}_i}{||\mathbf{f}_i||}$, then Eq.4.2 can be rewritten as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\kappa_i\hat{\mathbf{w}}_{y_i}^{\top}\hat{\mathbf{f}}_i)}{\sum_{j=1}^{K}\exp(\kappa_i\hat{\mathbf{w}}_j^{\top}\hat{\mathbf{f}}_i)}\right) \tag{4.3}$$

The equation above can be interpreted as if N realizations from a set of K von Mises-Fisher distributions with different concentration parameters κ_i are passed through the Softmax function. The probability density function of the von Mises-Fisher distribution for the random d-dimensional unit vector $\hat{\mathbf{f}}$ is given by: $P(\hat{\mathbf{f}}; \hat{\mathbf{w}}, \kappa) \propto \exp\left(\kappa\hat{\mathbf{w}}^{\top}\hat{\mathbf{f}}\right)$ where $\kappa \geq 0$. Under this parameterization $\hat{\mathbf{w}}$ is the mean direction on the hypersphere and κ is the concentration parameter. The greater the value of κ the higher the concentration of the distribution around the mean direction $\hat{\mathbf{w}}$. The distribution is unimodal for $\kappa > 0$ and is uniform on the sphere for $\kappa = 0$.

As with this formulation each weight vector is the mean direction of its associated features on the hypersphere, equiangular features maximizing the available space can be obtained by arranging accordingly their corresponding weight vectors around the origin. This problem is equivalent to distributing points uniformly on the sphere and is a well-known geometric problem, called Tammes problem [189] which is a generalization of the physic problem firstly addressed by Thomson [187]. In 2D the problem is that of placing K points on a circle so that they are as far as possible from each other. In this

Table 4.1: Number of regular Polytopes as dimension d increases.

Dimension d	1	2	3	4	≥ 5
Number of Regular Polytopes	1	∞	5	6	**3**

case the optimal solution is that of placing the points at the vertices of a regular K-sided polygon. The 3D analogous of regular polygons are Platonic Solids. However, the five Platonic solids are not always the unique solutions of the Thomson problem. In fact, only the tetrahedron, octahedron and the icosahedron are the unique solutions for $K = 4$, 6 and 12 respectively. For $K = 8$: the cube is not optimal in the sense of the Thomson problem. This means that the energy stabilizes at a minimum in configurations that are not symmetric from a geometric point of view. The unique solution in this case is provided by the vertices of an irregular polytope [190].

The non geometric symmetry between the locations causes the global charge to be different from zero. Therefore in general, when the number of charges is arbitrary, their position on the sphere cannot reach a configuration for which the global charge vanishes to zero. A similar argument holds in higher dimensions for the so called generalized Thomson problem [188]. According to this, we argue that, *the geometric limit to obtain a zero global charge in the generalized Thomson problem is equivalent to the impossibility to learn maximally separated features for an arbitrary number of classes.*

However, since the classification task it is not grounded in a specific dimension as for the case of charges, our approach addresses this issue by *selecting the appropriate dimension of the embedding space so as to have access to symmetrical fixed classifiers directly from regular polytopes.* In dimensions 5 and higher, there are only three ways to do that (See Tab. 4.1) and they involve the symmetry properties of the three well known regular polytopes available in high dimensional space [191]. These three special classes exist in every dimensionality and are: the d-Simplex, the d-Cube and the d-Orthoplex. In the next paragraphs the three fixed classifiers derived from them are presented.

The d-Simplex Fixed Classifier. In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. Specifically, a d-Simplex is a d-dimensional polytope which is the convex hull of its $d+1$ vertices. A regular d-Simplex may be constructed from a regular

$(d-1)$-Simplex by connecting a new vertex to all original vertices by the common edge length. According to this, the weights for this classifier can be computed as:

$$\mathbf{W}_{\mathcal{S}} = \left\{e_1, e_2, \ldots, e_{d-1}, \alpha \sum_{i=1}^{d-1} e_i\right\} \tag{4.4}$$

where $\alpha = \frac{1-\sqrt{d+1}}{d}$ and e_i with $i \in \{1, 2, \ldots, d-1\}$ denotes the standard basis in $\mathbb{R}^{d-1}$. The final weights will be shifted about the centroid and normalized. The d-Simplex fixed classifier defined in an embedding space of dimension d can accommodate a number of classes equal to its number of vertices:

$$K = d + 1. \tag{4.5}$$

This classifier has the largest number of classes that can be embedded in $\mathbb{R}^d$ such that their corresponding class features are equidistant from each other. It can be shown (see appendix) that the angle subtended between *any* pair of weights is equal to:

$$\theta_{\mathbf{w}_i,\mathbf{w}_j} = \arccos\left(-\frac{1}{d}\right) \quad \forall i, j \in \{1, 2, \ldots, K\} : i \neq j. \tag{4.6}$$

The d-Orthoplex Fixed Classifier. This classifier is derived from the d-Ortohoplex (or Cross-Polytope) regular polytope that is defined by the convex hull of points, two on each Cartesian axis of an Euclidean space, that are equidistant from the origin. The weights for this classifier can therefore defined as:

$$\mathbf{W}_{\mathcal{O}} = \{\pm e_1, \pm e_2, \ldots, \pm e_d\} \tag{4.7}$$

Since it has $2d$ vertices, the derived fixed classifier can accommodate in its embedding space of dimension d:

$$K = 2d \tag{4.8}$$

different classes. Each vertex is adjacent to other $d-1$ vertices and the angle between adjacent vertices is

$$\theta_{\mathbf{w}_i,\mathbf{w}_j} = \frac{\pi}{2} \quad \forall\, i, j \in \{1, 2, \ldots, K\} : j \in C(i) \tag{4.9}$$

Where each $j \in C(i)$ is an adjacent vertex and C is the set of adjacent vertices defined as $C(i) = \{j : (i, j) \in E\}$. E is the set of edges of the graph $G = (\mathbf{W}_{\mathcal{O}}, E)$. The d-Orthoplex is the dual polytope of the d-Cube and vice versa (i.e. the normals of the d-Orthoplex faces correspond to the the directions of the vertices of the d-Cube).

The d-Cube Fixed Classifier. The d-Cube (or Hypercube) is the regular polytope formed by taking two congruent parallel hypercubes of dimension $(d-1)$ and joining pairs of vertices, so that the distance between them is 1. A d-Cube of dimension 0 is one point. The fixed classifier derived from the d-Cube is constructed by creating a vertex for each binary number in a string of d bits. Each vertex is a d-dimensional boolean vector with binary coordinates −1 or 1. Weights are finally obtained from the normalized vertices:

$$\mathbf{W}_{\mathcal{C}} = \left\{ \mathbf{w} \in \mathbb{R}^d : \left[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}} \right]^d \right\}. \tag{4.10}$$

The d-Cube can accommodate:

$$K = 2^d \tag{4.11}$$

classes. The vertices are connected by an edge whenever the Hamming distance of their binary numbers is one therefore forming a d-connected graph. It can be shown (see appendix) that the angle of a vertex with its adjacent (i.e. connected) vertices is:

$$\theta_{\mathbf{w}_i,\mathbf{w}_j} = \arccos\left(\frac{d-2}{d}\right), \forall\, i,j \in \{1,\ldots,K\} : j \in C(i) \tag{4.12}$$

where $C(i)$ is the set of vertices adjacent to the vertex i.

Fig. 4.3 shows the angle between a weight and its adjacent weights computed from Eqs. 4.6, 4.9 and 4.12 as the dimension of the embedding space increases. Having the largest angle between its weights, the d-Simplex fixed classifier achieves the best inter-class separability. However, as the embedding space dimension increases, its angle tends towards $\pi/2$. Therefore, the largest the dimension of the space the more it becomes similar to the d-Orthoplex classifier. The main difference between the two classifiers is in their neighbor connectivity. The different connectivity of the three regular polytope classifiers has a direct influence on the evaluation of the loss. In the case of the d-Simplex classifier, all the summed terms in the loss of Eq. 4.3 have always comparable magnitudes in a mini batch.

The d-Cube classifier has the most compact feature embedding and the angle between each weight and its d neighbors decreases as the dimension increases. Accordingly, it is the hardest to optimize.

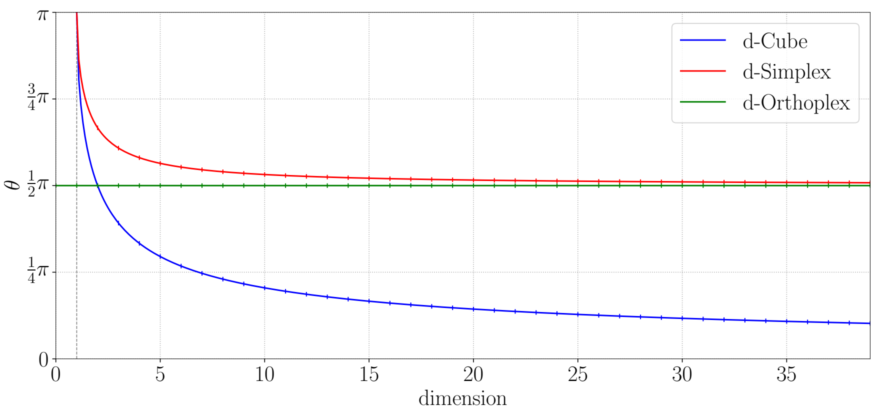


Figure 4.3: The angular space defined by RePoNet classifiers. Curves represent the angle between a weight and its adjacent weights as the dimension of the embedding space increases. The angle between class features follows the same trend.

4.4.1 Implementation

Given a classification problem with K classes, the three RePoNet fixed classifiers can be simply instantiated by defining a non trainable fully connected layer of dimension d, where d is computed from Eqs. 4.5, 4.8 and 4.11 as summarized in Tab 4.2.

Table 4.2: Feature dimension d as a function of the number of classes K.

RePoNet	d-Simplex	d-Cube	d-Orthoplex
Layer dim.	$d = K - 1$	$d = \lceil \log_2(K) \rceil$	$d = \lceil \frac{K}{2} \rceil$

In order to accommodate different CNN architectures having different convolutional activations output size, a middle "junction" linear layer (without ReLu) is required to go for example from the 2048 size of the ResNet50 to the feature size of 10 of the fixed d-Cube classifier with the 1000 classes of ImageNet, or as to go from the 1669 of the DenseNet169 to the 500 of the fixed d-Orthoplex classifier of the ImageNet dataset.

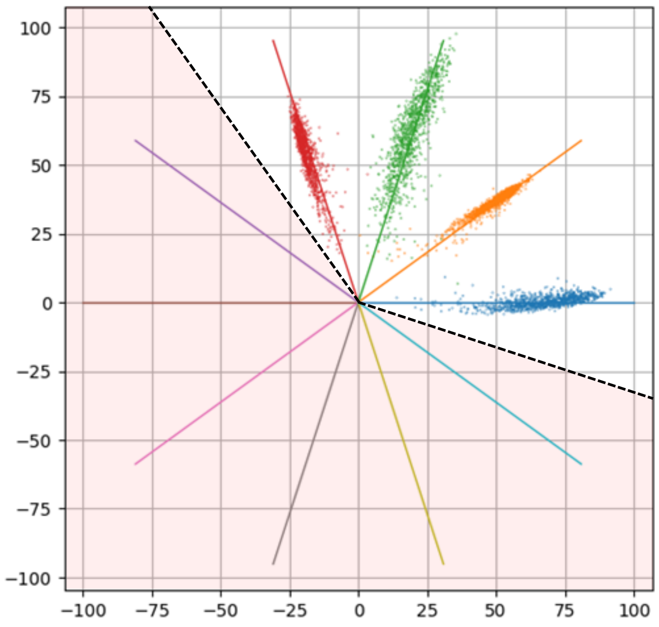


Figure 4.4: Learning with not assigned classes in a 2D embedding space. The figure shows the features learned with virtual negative classes using the MNIST dataset. The distribution of features is learned using a 10-sided regular polygon in which six of the classes are virtual (i.e. only the first four digits are used). Unassigned weights (colored lines inside the shaded region) force a large angular margin region (shaded region) from which features are pushed out.

4.4.2 Exceeding Vertices as Virtual Negative Classes

Except for the d-Simplex that allows to *fully* assign all of its vertices for any given number of classes K, for both the d-Cube and the d-Orthoplex classifiers some of the vertices may be in excess for a given number of classes. As implied by Eq. 4.8, in the case of the d-Orthoplex one vertex remains unassigned when the number of classes K is odd. In the case of the d-Cube classifier, due to the exponential dependency in Eq. 4.11, a large number of vertices may remain not assigned. For example, assuming $K = 100$ the d-Cube fixed classifier has 128 vertices (see Tab. 4.2) and 28 of them are not assigned to any class. As shown in [183], such unassigned classes act as virtual negative classes forcing a margin around the unassigned weights without affecting the correctness of softmax based cross entropy optimization. Indeed, virtual class injection does not change substantially the objective function of Eq. 4.3 that can be rewritten as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\kappa_i \hat{\mathbf{w}}_{y_i}^{\top}\hat{\mathbf{f}}_i)}{\sum_{j=1}^{K}\exp(\kappa_i \hat{\mathbf{w}}_{j}^{\top}\hat{\mathbf{f}}_i) + \sum_{j=K+1}^{K_V}\exp(\kappa_i \hat{\mathbf{w}}_{j}^{\top}\hat{\mathbf{f}}_i)}\right) \tag{4.13}$$

where K_V is the number of virtual classes (i.e. the exceeding polytope vertices). The practical effect is that of packing the features space for each class. Fig. 4.4 illustrates an example similar to Fig. 4.2(a) in which a 10-sided polygon fixed classifier is learned to classify the first four digits of the MNIST dataset $(0, 1, 2$ and $3)$. The remaining six "empty slots" of the classifier are not assigned to any class data and therefore the classifier acts as a virtual negative classifier forcing a large margin (indicated as shaded region) around the virtual class weights (colored lines). This basic result generalizes the proposed method to an arbitrary number of classes.

4.4.3 Fixed Classifier Decision Boundaries

In binary-classification, the posterior probabilities obtained by softmax of Eq.4.3 are:

$$p_1 = \frac{\exp(\kappa \hat{\mathbf{w}}_1^{\top}\hat{\mathbf{f}})}{\exp(\kappa \hat{\mathbf{w}}_1^{\top}\hat{\mathbf{f}}) + \exp(\kappa \hat{\mathbf{w}}_2^{\top}\hat{\mathbf{f}})} \tag{4.14}$$

$$p_2 = \frac{\exp(\kappa \hat{\mathbf{w}}_2^{\top}\hat{\mathbf{f}})}{\exp(\kappa \hat{\mathbf{w}}_1^{\top}\hat{\mathbf{f}}) + \exp(\kappa \hat{\mathbf{w}}_2^{\top}\hat{\mathbf{f}})} \tag{4.15}$$

where $\mathbf{f}$ is the learned feature vector and $\mathbf{w}_1$ $\mathbf{w}_2$ are the fixed classifier weights. The predicted label will be assigned to the class 1 if $p_1 > p_2$ and to the class 2 if $p_1 < p_2$. By comparing the two probabilities p_1 and p_2, $\kappa \hat{\mathbf{w}}_1^{\top}\hat{\mathbf{f}} + \kappa \hat{\mathbf{w}}_2^{\top}\hat{\mathbf{f}}$ determines the classification result. The decision boundary is therefore $\kappa \hat{\mathbf{w}}_1^{\top}\hat{\mathbf{f}} + \kappa \hat{\mathbf{w}}_2^{\top}\hat{\mathbf{f}} = 0$. Due to weight normalization the posterior probabilities result in $p_1 = \kappa||\hat{\mathbf{f}}||cos(\theta_1)$ and $p_2 = \kappa||\hat{\mathbf{f}}||cos(\theta_2)$ and since p_1 and p_2 share the same feature $\hat{\mathbf{f}}$ the equation $cos(\theta_1) - cos(\theta_2) = 0$ is verified at the *angular bisector* between $\mathbf{w}_1$ and $\mathbf{w}_2$. Although the above analysis is built on binary-class case, it can be generalized to the multi-class case [176].

In RePoNet angular bisectors define class decision boundaries that follow a symmetry similar to that of the regular polytope defining the classifier.

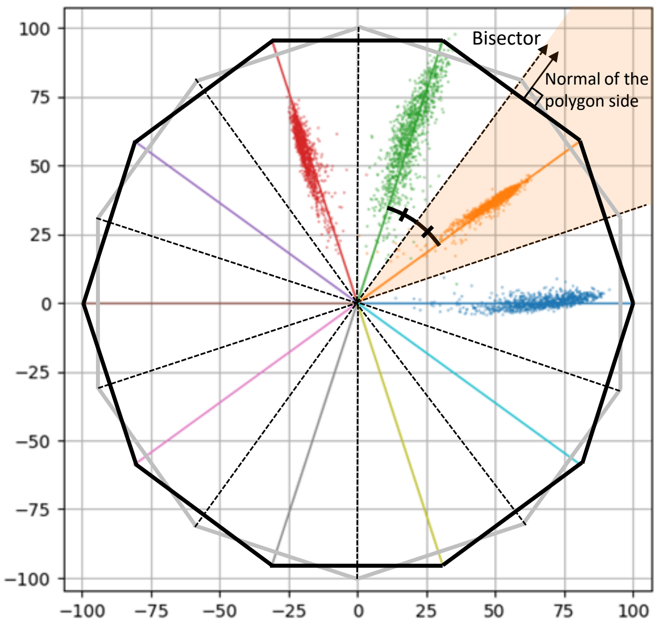
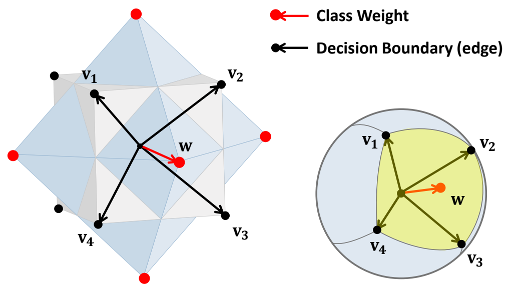


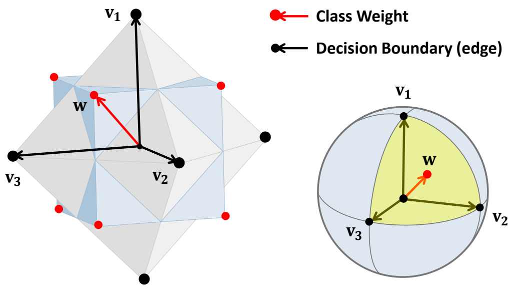
Figure 4.5: The intuition behind the decision boundaries in RePoNet (10-sided polygon). The bisector directions (dotted lines), represent the class decision boundaries. They have the same direction of the normal of the corresponding polygon side (only one shown for clarity). The decision boundaries form a regular polygon that is related with the classifier 10-sided polygon according to duality. For clarity only one class region is highlighted (shaded region).

Specifically, the class decision boundaries and the weights of the classifier are related by the duality relationship that exists between regular polytopes. More practically:

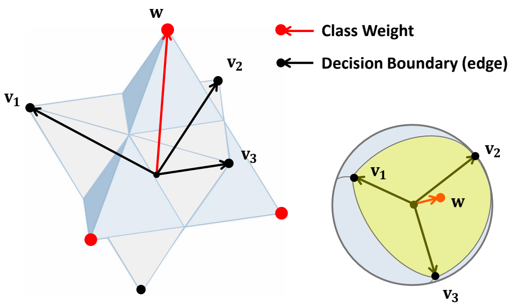
- the set of decision boundaries of the d-Simplex classifier is shaped as a d-Simplex.
- the set of the decision boundaries of the d-Cube classifier is shaped as a d-Orthoplex;
- the set of the decision boundaries of the d-Orthoplex classifier is shaped as a d-Cube.



(a)



(b)



(c)

Figure 4.6: RePoNet fixed classifiers decision boundaries in a 3D embedding space. In each figure, on the *left* it is shown: the regular polytope classifier (light blue), its dual polytope (grey), a classifier weight $\mathbf{w}$ (red) and its edge decision boundaries $\mathbf{v}_1, \mathbf{v}_2, \ldots$ (black). On the *right* the same entities are shown on the unit sphere. Specifically, the yellow region shows the region where class features are located. For clarity, only one class weight and the corresponding edge decision boundaries are shown. Specifically, (a) shows the d-Orthoplex classifer, (b) shows the d-Cube classifer and (c) shows the d-Simplex classifer. While the figures illustrate the situation in $\mathbb{R}^3$, the characterization extends to arbitrary dimensions.

Class decision boundaries are still defined as a regular polytope and the features located within such boundaries are therefore maximally separated. The basic intuition behind this result can be better appreciated in 2D exploiting the well known result stating that all the regular polygons are self-dual [191]. That is, the normal of each side of a regular polygon is parallel to the direction from the origin towards the vertex of its dual polygon. Fig. 4.5 shows the example introduced in Fig. 4.4 in which decision boundaries are highlighted with dotted lines according to the dual regular polygon. Fig. 4.6 illustrates the duality relationship between the weights of the proposed three fixed classifiers and their decision boundaries in the 3D embedding space.

4.5 Experimental Results

We evaluate the correctness (Sec. 4.5.1) and the no loss of performance of our approach with respect to standard baselines using trainable and fixed classifiers across a range of datasets and architectures (Sec. 4.5.2). All the experiments are conducted with the well known MNIST, FashionMNIST [192], EMNIST [193], CIFAR-10, CIFAR-100 [194] and ImageNet (ILSVRC2012) [43] datasets. We chose several common CNN architectures (i.e. LeNet, VGG, ResNet, DenseNet), as well as more recent ones (i.e. SeResNeXt50 [195], SkResNeXt50 [196] and EfficientNet [197]) that improve performance while maintaining or reducing computational complexity and model size.

4.5.1 Hard Permutations Verification

Since fixed classifiers cannot rely on an adjustable set of subspaces for class feature representation, we want to test if some permutations are harder than others for our proposed method. The presence of such hard permutations would preclude the general applicability of our method. The standard trainable classifier does not suffer from this problem, when features cannot be well separated a trainable classifier can rearrange its feature subspace directions so that the previous convolutional layers can better disentangle the non-linear interaction between complex data patterns. A fixed classifier demands this missing capability to all the previous layers.

According to this, we generate random permutations of the ground truth label positions[5] and a new model is learned for each permuted dataset.

[5]This is equivalent to randomly permuting the classifier weight vectors set $\mathbf{W}$ =

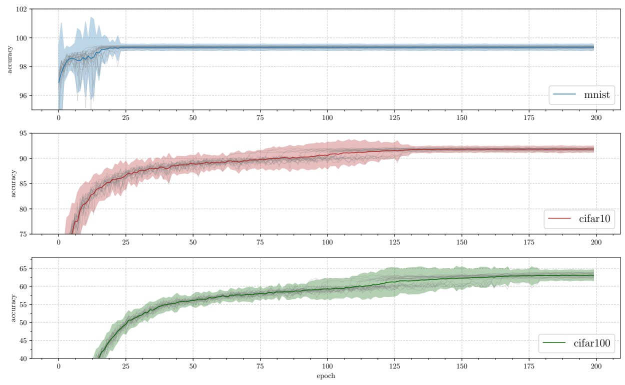


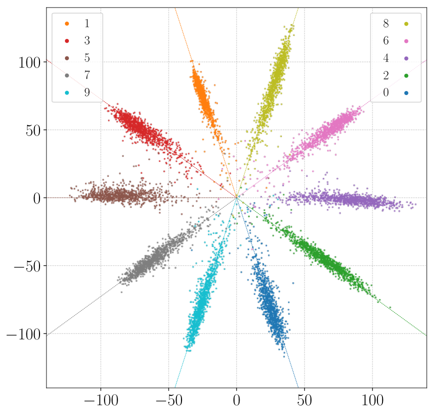
Figure 4.7: Class permutation verification. Average accuracy curves and confidence interval computed from the MNIST, CIFAR-10 and CIFAR-100 datasets (from top to bottom, respectively) under different random permutations of the ground truth labels position.

Fig. 4.7 shows the mean and the 95% confidence interval computed from the accuracy curves of the learned models. To provide further insight into this analysis, 20 out of 500 accuracy curves computed for each dataset are also shown. Specifically, the evaluation is performed on three different datasets with an increasing level of complexity (i.e MNIST, CIFAR-10 and CIFAR-100). All the models are trained for 200 epochs to make sure that the models trained with CIFAR-100 achieve convergence.

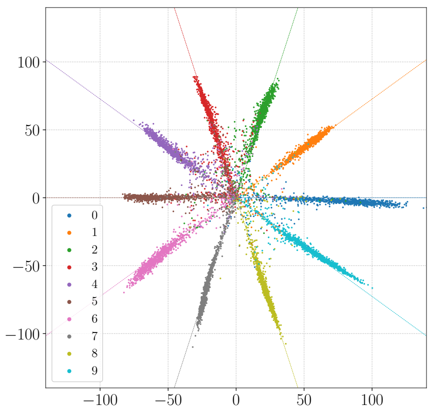
In order to address the most severe possible outcomes that may happen, for this experiment we used the *d*-Cube fixed classifier. Being the hardest to optimize, this experiment can be regarded as a worst case analysis scenario for our method. As shown in the same figure, the performance is substantially insensitive to both permutations and datasets. The average reduction in performance at the end of the training process is negligible and the confidence intervals reflect the complexity of the datasets. Although the space of permutations cannot be exhaustively evaluated even for a small number of classes, we have achieved proper convergence for the whole set of 1500 learned models. The experiment took 5 days on a Nvidia DGX-1.

According to this, it is concluded that fixing the final classifier (therefore not having access to a set of adjustable subspaces for class feature repre-

$\{\mathbf{w}_j\}_{j=1}^{K}$.



(a)



(b)

Figure 4.8: The distribution of features learned using a 10-sided regular polygon. *(a)*: A special permutation of classes is shown in which the MNIST even and odd digits are placed in the positive and negative half-space of the abscissa respectively. *(b)*: The features learned using the CIFAR-10 dataset.

sentation) does not affect the expressive power of neural networks. This experiment also provides a novel and more systematic empirical evidence with respect to [156] (in which basically only one permutation is tested) on the general applicability and correctness of fixed classifiers.

We finally report qualitative results of a learned permuted dataset. Fig. 4.8(a) shows features learned in a k-sided polygon (2d embedding space) on the MNIST dataset. In particular the model is learned with the permutation (manually selected) of the labels that places even and odd digits features respectively on the positive and negative half space of the abscissa. Fig. 4.8(b) shows the features of on CIFAR-10 learned with a similar 10-sided-polygon. It can be noticed that features are distributed following the same polygonal pattern shown in Fig. 4.8(a).

4.5.2 Generalization and Performance Evaluation

Once verified that the order position of the class labels does not adversely affect the proposed method, in this section we evaluate the classification performance of RePoNet on the following datasets: MNIST, EMNIST, FashionMNIST, CIFAR-10, CIFAR-100 and ImageNet. The proposed method is compared with the fixed classifier method reported in [156], here implemented for different architectures and different dimensions of the embedding space. Standard CNN baselines with learned classifiers are also included. Except for the final fixed classifier all the compared methods have exactly the same architecture and training settings as the one that RePoNet uses.

MNIST and CIFAR

We trained the so called LeNet++ architecture [198] on all the MNIST family datasets. The network is a modification of the LeNet [199] to a deeper and wider network including parametric rectifier linear units (pReLU) [200]. We further trained VGG [201] with depth 13 and 19, ResNet50 [125], SeNet [124] and DenseNet169 [202] on the CIFAR-10 and CIFAR-100 datasets. Popular network architectures for ImageNet require modifications to adapt to the CIFAR 32x32 input size. According to this, our experiments follow publicly available implementations[6]. We compared all the variants of our approach for each architecture including trainable classifiers with different dimensions of the feature space.

The mini batch size is 256 for both the MNIST family datasets and the CIFAR-10/100 datasets. Specifically, for the CIFAR datasets, we compared both a ”vanilla” learning setup with no hyperparameters tuning based on the Adam optimizer (learning rate 0.0005) for the VGG architectures and a learning setup based on SGD with a specific learning rate schedule (starting from 0.1 and decreasing by a factor of 10 after 150 and 250 epochs) on the ResNet50, SEnet18 and DenseNet169 architectures. As hyperparameters tuning is an integral part of Deep Learning we provide two opposite learning setup in this regard.

All the results are reported in Tab. 4.3, 4.4 and 4.5 for respectively the MNISTs, CIFAR-10 and CIFAR-100. In addition to the well-known MNIST and FashionMnist, we included EMNIST dataset having 47 classes including lower/upper case letters and digits. This allows to quantify with a specific dataset and architecture, as in CIFAR-10 and CIFAR-100, the classification accuracy with a higher number of classes. Each entry in the tables report the test-set accuracy. The subscript indicates the specific feature space di-

[6] `https://github.com/bearpaw/pytorch-classification` and `https://github.com/kuangliu/pytorch-cifar`

Table 4.3: Reported accuracy (%) of the RePoNet method on MNIST, EMNIST, FashionMNIST datasets on different combinations of architectures and relative learned classifier baselines.

	MNIST ($K = 10$)	EMNIST ($K = 47$)	FashionMNIST ($K = 10$)
Architecture	LeNet++		
RePoNet K-sided-polygon	$99.24_{d=2}$	$72.81_{d=2}$	$92.48_{d=2}$
Hadamard fixed classifier [156]	$21.14_{d=2}$	$4.12_{d=2}$	$19.89_{d=2}$
Learned Classifier	$99.21_{d=2}$	$73.08_{d=2}$	$92.79_{d=2}$
RePoNet d-Cube	$99.58_{d=4}$	$88.12_{d=6}$	$94.01_{d=4}$
Hadamard fixed classifier [156]	$41.99_{d=4}$	$15.12_{d=6}$	$37.16_{d=4}$
Learned Classifier	$99.41_{d=4}$	$86.96_{d=6}$	$93.94_{d=4}$
RePoNet d-Orthoplex	$99.66_{d=5}$	$88.19_{d=24}$	$94.84_{d=5}$
Hadamard fixed classifier [156]	$79.34_{d=5}$	$60.34_{d=24}$	$74.22_{d=5}$
Learned Classifier	$99.07_{d=5}$	$87.66_{d=24}$	$94.21_{d=5}$
RePoNet d-Simplex	$99.71_{d=9}$	$88.89_{d=46}$	$94.29_{d=9}$
Hadamard fixed classifier [156]	$99.12_{d=9}$	$88.48_{d=46}$	$94.30_{d=9}$
Learned classifier	$99.41_{d=9}$	$88.33_{d=46}$	$94.41_{d=9}$
Hadamard fixed classifier [156]	$99.54_{d=512}$	$88.35_{d=512}$	$94.14_{d=512}$
Learned classifier	$99.29_{d=512}$	$88.87_{d=512}$	$94.28_{d=512}$

mension d used for that experiment. The results reveal and confirm that the proposed method achieves comparable classification accuracy of other trainable classifier models. This evidence is in agreement on all the combinations of datasets, architectures, number of classes and feature space dimensions. All the RePoNet variants exhibit similar behavior even in complex combinations such as in the case of the CIFAR-100 dataset in low dimensional feature space. For example, the RePoNet d-Cube fixed classifier implemented with the VGG19 architecture achieves an accuracy of 65.32% in a $d = 7$ dimensional feature space. A fully trainable classifier in a feature space of dimension $d = 512$ (i.e. two orders of magnitude larger), achieves a moderate improvement of about 3% (68.47%). On the other hand, with a significantly shorter feature dimension of $d = 50$, RePoNet d-Orthoplex improves the accuracy to 69.76%. All the RePoNet variants exhibit similar behavior also

Table 4.4: Reported accuracy (%) of the RePoNet method on the CIFAR-10 dataset on different combinations of architectures and relative learned classifier baselines.

	CIFAR-10 ($K = 10$)				
ARCHITECTURE	VGG13	VGG19	RESNET50	SENET18	DENSENET169
Training hyperparameters	ADAM	ADAM	SGD	SGD	SGD
RePoNet K-sided-polygon	$90.79_{d=2}$	$91.54_{d=2}$	$92.78_{d=2}$	$92.63_{d=2}$	$92.74_{d=2}$
Hadamard fixed classifier [156]	$19.45_{d=2}$	$19.19_{d=2}$	$19.69_{d=2}$	$19.77_{d=2}$	$19.76_{d=2}$
Learned classifier	$90.41_{d=2}$	$91.17_{d=2}$	$93.15_{d=2}$	$93.25_{d=2}$	$92.89_{d=2}$
RePoNet d-Cube	$92.26_{d=4}$	$92.58_{d=4}$	$94.86_{d=4}$	$94.96_{d=4}$	$93.94_{d=4}$
Hadamard fixed classifier [156]	$37.19_{d=4}$	$36.95_{d=4}$	$37.89_{d=4}$	$38.05_{d=4}$	$38.12_{d=4}$
Learned classifier	$92.14_{d=4}$	$92.21_{d=4}$	$95.03_{d=4}$	$94.95_{d=4}$	$94.97_{d=4}$
RePoNet d-Orthoplex	$92.51_{d=5}$	$92.47_{d=5}$	$95.25_{d=5}$	$95.05_{d=5}$	$95.16_{d=5}$
Hadamard fixed classifier [156]	$73.77_{d=5}$	$72.46_{d=5}$	$75.99_{d=5}$	$75.95_{d=5}$	$75.73_{d=5}$
Learned classifier	$92.28_{d=5}$	$92.21_{d=5}$	$95.18_{d=5}$	$95.08_{d=5}$	$95.41_{d=5}$
RePoNet d-Simplex	$92.71_{d=9}$	$92.59_{d=9}$	$95.66_{d=9}$	$95.36_{d=9}$	$95.32_{d=9}$
Hadamard fixed classifier [156]	$92.03_{d=9}$	$92.37_{d=9}$	$95.53_{d=9}$	$95.25_{d=9}$	$94.92_{d=9}$
Learned classifier	$91.89_{d=9}$	$92.60_{d=9}$	$95.08_{d=9}$	$95.20_{d=9}$	$95.32_{d=9}$
Hadamard fixed classifier [156]	$90.11_{d=512}$	$88.32_{d=512}$	$95.36_{d=512}$	$95.49_{d=512}$	$95.68_{d=512}$
Learned classifier	$92.34_{d=512}$	$92.42_{d=512}$	$95.53_{d=512}$	$95.26_{d=512}$	$95.68_{d=512}$

in the case of more sophisticated architectures trained with SGD scheduled learning rates to match state-of-the-art performance. RePoNet classifiers are both agnostic to architectures and training setup being able to improve accuracy as in the case of trainable classifier.

Results further show that in the Hadamard fixed classifier [156], when the number of classes is large relative to number of unique weight directions in the embedding space (i.e. $d < K$), no proper learning can be obtained. As expected, this effect is present for simple datasets as the MNIST digits dataset, however as reported in [156] Section 4.2 (Possible Caveats) as the number of classes K increases the effect is less pronounced.

When $d \approx K$ or $d > K$, classification performance is similar. However, as shown in Fig. 4.9(a) RePoNet achieves higher speed of convergence than [156] and equal to that of the trainable baseline. Our conjecture is that with our symmetrical fixed classifiers, each term in the loss function tends to have the same magnitude centered around the mean of the distribution (i.e. von Mises-Fisher distribution is similar to the Normal distribution) and there-

Table 4.5: Reported accuracy (%) of the RePoNet method on the CIFAR-100 dataset on different combinations of architectures and relative learned classifier baselines.

	CIFAR-100 ($K = 100$)				
Architecture	VGG13	VGG19	ResNet50	SENet18	DenseNet169
Training Techniques	ADAM	ADAM	SGD	SGD	SGD
RePoNet K-sided-polygon	$36.22_{d=2}$	$37.65_{d=2}$	$33.39_{d=2}$	$35.26_{d=2}$	$30.04_{d=2}$
Hadamard fixed classifier [156]	$1.75_{d=2}$	$1.75_{d=2}$	$1.61_{d=2}$	$1.80_{d=2}$	$1.64_{d=2}$
Learned classifier	$37.56_{d=2}$	$35.83_{d=2}$	$33.30_{d=2}$	$40.57_{d=2}$	$32.87_{d=2}$
RePoNet d-Cube	$64.35_{d=7}$	$65.32_{d=7}$	$67.27_{d=7}$	$69.38_{d=7}$	$68.99_{d=7}$
Hadamard fixed classifier [156]	$5.96_{d=7}$	$5.52_{d=7}$	$5.91_{d=7}$	$6.27_{d=7}$	$6.08_{d=7}$
Learned classifier	$64.11_{d=7}$	$65.29_{d=7}$	$74.96_{d=7}$	$75.29_{d=7}$	$75.51_{d=7}$
RePoNet d-Orthoplex	$68.78_{d=50}$	$69.76_{d=50}$	$78.23_{d=50}$	$77.24_{d=50}$	$79.41_{d=50}$
Hadamard fixed classifier [156]	$43.88_{d=50}$	$43.89_{d=50}$	$50.33_{d=50}$	$49.56_{d=50}$	$50.65_{d=50}$
Learned classifier	$68.13_{d=50}$	$68.41_{d=50}$	$78.22_{d=50}$	$77.15_{d=50}$	$78.83_{d=50}$
RePoNet d-Simplex	$68.61_{d=99}$	$68.69_{d=99}$	$79.02_{d=99}$	$78.20_{d=99}$	$80.01_{d=99}$
Hadamard fixed classifier [156]	$67.23_{d=99}$	$67.18_{d=99}$	$78.82_{d=99}$	$77.21_{d=99}$	$79.41_{d=99}$
Learned classifier	$68.15_{d=99}$	$68.87_{d=99}$	$78.58_{d=99}$	$77.42_{d=99}$	$79.05_{d=99}$
Hadamard fixed classifier [156]	$63.16_{d=512}$	$64.46_{d=512}$	$78.78_{d=512}$	$77.94_{d=512}$	$79.44_{d=512}$
Learned classifier	$68.56_{d=512}$	$68.47_{d=512}$	$77.96_{d=512}$	$77.63_{d=512}$	$79.63_{d=512}$

fore the average computed in the loss is a good estimator. Contrarily, in Hadamard classifier the terms may have different magnitudes and "important" errors in the loss may not be taken correctly into account by averaging.

ImageNet

We further evaluated our method on the 1000 object category classification problem defined by the ImageNet dataset. This dataset consists of a 1.2M image training set and a 100k image test set. We compared all the variants of our approach on different combinations of architectures and relative trainable classifiers. The comparison also includes the Hadamard classifier.

Experiments have been conducted in two different configurations of the training hyperparameters. First, we performed experiments using Adam optimizer and simple augmentation based on random cropping and horizontal flipping on well-established networks such as ResNet50 [125] and DenseNet169 [202]. The learning rate is automatically adjusted when a plateau in model performance is detected. We trained for 250 epochs with batch size 64 with an initial learning rate of 0.0005. With this configuration, we aim to evaluate our method without performing any specific hyperparameter optimization or exploiting large computational resources.

Second, we evaluated our method with three more sophisticated CNN architectures (SKresNeXt, SEresNeXt, and EfficientNet) and related training hyperparameters. With this configuration, the aim is to evaluate whether our method can reach state-of-the-art performance. The SKresNeXt and SEresNeXt architectures integrate the SE and SK blocks, [124] and [203] respectively, with the ResNeXt architecture [204]. The benefit of these variants is to maintain computational complexity and model size similar to the SEnet and SKnet architectures while further improving performance. The third architecture, EfficientNet [197], achieves state-of-the-art performance using significantly fewer parameters than other state-of-the-art models. As these architectures typically require large effort to tune training hyperparameters, we trained our method on top of these models following the settings reported in their original papers. Specifically, we train EfficentNet-B2 following the original paper [197]: RMSProp optimizer with decay 0.9 and momentum 0.9; batch norm momentum 0.99; initial learning rate 0.256 that decays by 0.97 every 2.4 epochs; weight decay 1e-5. Analogously, SKresNeXt50 and SEresNeXt50 are trained following the ResNeXt50 [204]: SGD optimizer, weight decay 0.0001; momentum 0.9; initial learning rate of 0.1, divided by 10 for three times using a specific schedule reported in the paper. For all the three models we used automated data augmentation techniques from [205] (RandAugment) with distortion magnitude 7. SeResNext50 and SkResNext50 were trained for 250 epochs with 192 batch size. EfficientNet-B2 was trained for 450 epochs with 120 batch size. Our evaluation is based on the pytorch-image-models[7] repository.

Tab. 4.6 summarizes our results. As can be clearly noticed, except for the d-Cube there is no substantial variation between our proposed fixed classifiers and learned classifiers. This holds also in the case of the learned classifiers as defined in their original architecture implementations as shown in the bottom line. The table also reports comparable accuracy with the Hadamard Fixed Classifier [156]. As in the case of CIFAR-10 and CIFAR-100, the overall accuracy of the d-Cube is lower than its corresponding learned classifiers. We argue this is mainly due to the increased difficulty

[7]https://github.com/rwightman/pytorch-image-models

Table 4.6: Reported accuracy (%) of the RePoNet method on the ImageNet dataset on different combinations of architectures and relative Learned Classifier baselines.

Architecture	ResNet50	DenseNet169	SeResNeXt50	SkResNeXt50	EfficientNet-B2
Training hyperparameters	ADAM	ADAM	SGD+RandAug	SGD+RandAug	RMSPROP+RandAug
RePoNet d-Cube	$63.44_{d=10}$	$63.63_{d=10}$	$73.58_{d=10}$	$74.80_{d=10}$	$75.62_{d=10}$
Learned classifier	$68.82_{d=10}$	$68.03_{d=10}$	$76.66_{d=10}$	$77.49_{d=10}$	$77.42_{d=10}$
RePoNet d-Orthoplex	$73.71_{d=500}$	$74.20_{d=500}$	$79.95_{d=500}$	$79.66_{d=500}$	$80.07_{d=500}$
Learned classifier	$73.67_{d=500}$	$73.70_{d=500}$	$77.60_{d=500}$	$80.18_{d=500}$	$79.27_{d=500}$
RePoNet d-Simplex	$74.13_{d=999}$	$74.03_{d=999}$	$80.25_{d=999}$	$80.17_{d=999}$	$80.61_{d=999}$
Learned classifier	$73.96_{d=999}$	$73.37_{d=999}$	$77.99_{d=999}$	$80.08_{d=999}$	$79.36_{d=999}$
Hadamard Fixed Classifier [156]	$74.07_{d=2048}$	$73.95_{d=1669}$	$80.25_{d=2048}$	$80.19_{d=2048}$	$79.74_{d=1408}$
Learned classifier	$74.11_{d=2048}$	$74.01_{d=1669}$	$79.95_{d=2048}$	$80.09_{d=2048}$	$80.57_{d=1408}$

of optimizing in the $d = 10$ dimensional space as the angle between each class weight vector and its d adjacent weight vectors decreases to zero as the dimension increases (Fig. 4.3). However, the d-Cube classifier shows the largest relative improvement as the representational power of the architectures increase (left to right). For example, the accuracy of the d-Cube-EfficientNet-B2 fixed classifier is 12.18 percentage points larger than the d-Cube-ResNet50 (i.e. $75.62 - 63.44 = 12.18$). This relative performance improvement is substantially higher than that of the corresponding learned classifier (i.e. $77.42 - 68.82 = 8.6$). This result is quantitatively consistent with the underlying assumption of this work and provides further support on the fact that the adjustable capability of the final classifier can be successfully demanded to previous layers. The other two RePoNet variants substantially achieve the same accuracy of learned classifiers, irrespective whether they have similar ($d = \{10, 500, 999\}$) or higher feature space dimension ($d = \{2048, 1669, 1408\}$) as in their original architecture implementations. They do not show specific relative performance improvement with increasing representational power. More importantly, both the d-Simplex and the d-Orthoplex classifiers reach state-of-the-art accuracy (around 80%) when combined with competitive architectures. This confirms the validity and the no loss of generalization capability of our method.

Finally, we report on Tab. 4.7 the decrease in model parameters of our method when implemented on top of the evaluated architectures. The table shows the total number of parameters for each network in comparison with their original learned classifiers (i.e. bottom line in Tab. 4.6). As indicated in

Table 4.7: The number of parameters of each network and the percentage (%) of saved parameters on the ImageNet dataset (d indicates the feature dimension).

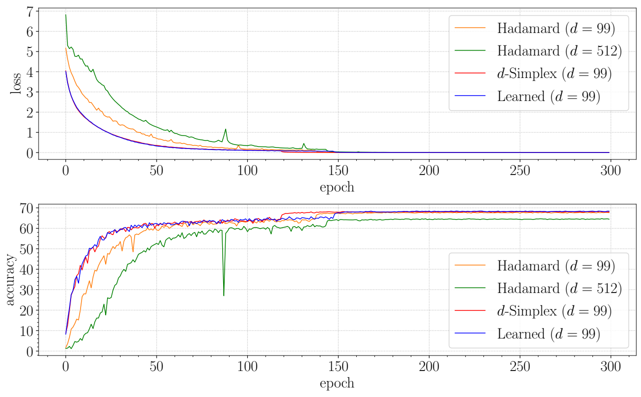
		Saved Params (%)		
Architecture	Param#	d-Cube	d-Orthoplex	d-Simplex
DenseNet169	14.15M	$11.65_{d=10}$	$5.89_{d=500}$	$0.02_{d=999}$
ResNet50	25.56M	$7.94_{d=10}$	$4.01_{d=500}$	$0.01_{d=999}$
SeResNext50	27.56M	$7.36_{d=10}$	$3.72_{d=500}$	$0.01_{d=999}$
SkResNext50	27.50M	$7.38_{d=10}$	$3.73_{d=500}$	$0.01_{d=999}$
EfficientNet-B2	9.11M	$15.31_{d=10}$	$7.74_{d=500}$	$0.03_{d=999}$

Tab. 4.7, the d-Orthoplex-EfficinetNet-B2 fixed classifier saves 7.74% of the network parameters while achieving the same accuracy (around 80%). A further notable example is that the d-Cube-EfficinetNet-B2 with 7.7M of parameters (15.31% savings) achieves similar accuracy of a vanilla ResNet50 baseline (i.e. around 75% accuracy) having 25.5M of parameters.

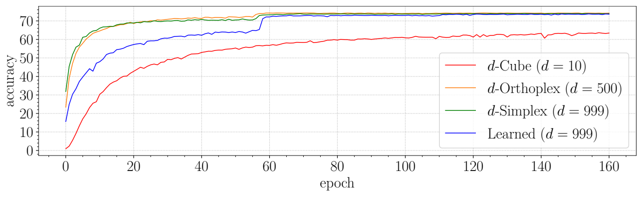
4.5.3 Training Time

The time it takes to train a neural network model to address a classification problem is typically considered as the product of the training time per epoch and the number of epochs which need to be performed to reach the desired level of accuracy [206]. Although training time per epoch is shorter in our case as the weights of the fixed classifier do not require back-propagation, the effect can be considered negligible with respect to the number of epochs required to reach a reasonable desired level of accuracy (i.e. speed of convergence). According to this, we report in Fig. 4.9 and Fig. 4.10 the classification accuracy over the epochs for the two different configurations of the training hyperparameters we evaluated.

Specifically, Fig. 4.9(a)(*top*) and Fig. 4.9(a)(*bottom*) show the training error and the classification accuracy, respectively, over the epochs. The curves are obtained on the CIFAR100 dataset, using the VGG19 architecture and training is performed according to the Adam stochastic optimization. Fig. 4.9(b) shows the accuracy curves of the proposed three fixed classifiers and the best performing relative learned classifier (i.e. $d = 999$). The curves are obtained on the ImageNet dataset with the DenseNet169 architecture and learned according to Adam. As can be noted, the time to reach a desired level of accuracy is shorter or equal in the d-Simplex and d-Orthoplex classifiers in comparison with the learned and Hadamard fixed classifiers. The d-Cube classifier is the slowest and it does not reach a comparable final performance. This is due to the different feature dimension ($d = 10$) and topology. However, when compared with a learned classifier with same feature dimension (as discussed in the next paragraph) the training time is similar.



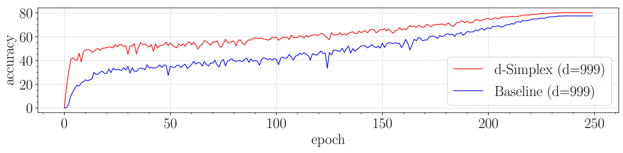
(a)



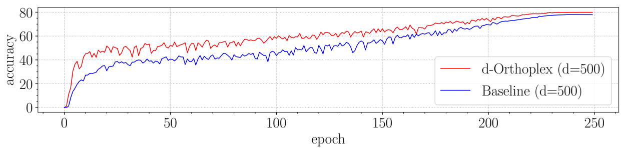
(b)

Figure 4.9: Speed of convergence comparison (Adam). (a): Training error curves (*top*) and test accuracy curves (*bottom*) using the CIFAR-100 dataset with the VGG19 architecture. (b): ImageNet learning speed using DenseNet169. As evidenced from the figures, the proposed method has faster convergence.

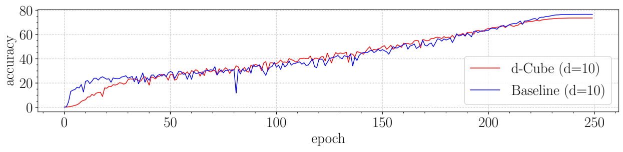
Fig. 4.10(a) and Fig. 4.10(b) show the training configuration using SGD+RandAug on the SeResNeXt50 architecture. The d-Simplex and d-Orthoplex fixed



(a)



(b)



(c)

Figure 4.10: Speed of convergence comparison (SGD+RANDAUG). Test accuracy curves over the epochs on the ImageNet test set for the SeResNeXt50 architecture using the proposed fixed classifiers (red) and the standard trainable baselines (blue). The time to reach the same accuracy is shorter or equal for our method.

classifiers (red) and the learned classifiers (blue) are shown, respectively. As evidenced in the figure, the learned classifiers require to be trained for about 150 epochs to obtain the accuracy our method achieves in 50 and 90 epochs, respectively. Although this advantage reduces as the training progresses toward the end, our method achieves consistently better results. Fig. 4.10(c) shows equal time consumption for the case of the d-Cube classifier. Although the final accuracy is lower, the training time is similar.

The general behavior of the curves shown in Fig. 4.9 and Fig. 4.10 is consistent across combinations of datasets, architectures, classifiers and training strategies. The training time to reach the same accuracy is shorter or equal for our method and the time reduction follows the complexity of the embeddings defined by each regular polytope fixed classifier.

Overall, we observed that Regular Polytope Networks provide a novel, effective and easy approach to fixed classifiers that is no worse than the standard trainable ones and it achieves comparable state-of-the-art performance. Overall we also observed faster speed of convergence and a significant reduction in model parameters.

4.6 RePoNet with Additive Angular Margin Loss

Softmax with Cross Entropy loss is widely adopted by many classification approaches due to its simplicity, good performance and probabilistic interpretation. In applications like face recognition [36] or human body re-identification [207] test samples are not known in advance and recognition at test time is performed according to learned features based on their distance.

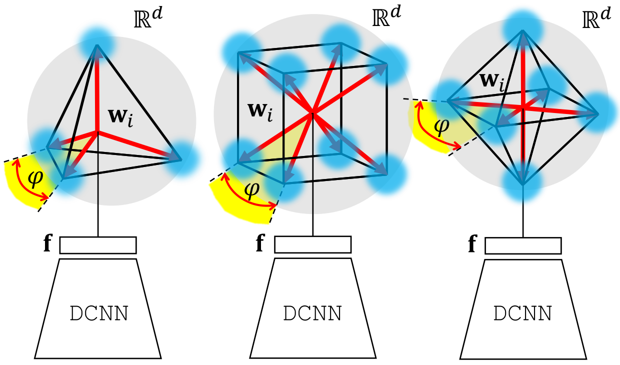


Figure 4.11: Margin Regular Polytope Networks (Margin-RePoNets). Features with *maximal* inter-class separability and intra-class compactness are shown (light blue). These are determined combining fixed classifiers derived from regular polytopes [1] with a recently developed margin loss [2]. Maximal features separation is obtained by setting the classifier weights $\mathbf{w}_i$ according to values following the symmetrical of configuration regular polytopes (red). Maximal compactness is obtained by setting the margin between the features at the maximum allowed (i.e. φ).

The underlying assumption in this learning scenario is that images of the same identity (person) are expected to be closer in the representation space, while different identities are expected to be far apart. Or equivalently, the learned features having low intra-class distance and large inter-class distance are successful at modeling novel unseen identities and for this reason such features are typically defined “discriminative”. Specifically, the Center Loss, firstly proposed in [198], has been proved to be an effective method to compute discriminative features. The method learns a center determined as the average of features belonging to the same class. During training, the centers are updated by minimizing the distances between the deep features and their corresponding class centers. The CNN is trained under the joint supervision of the Softmax loss and the Center Loss by balancing the two supervision signals. Intuitively, the Softmax loss forces the deep features of different classes to be separable while the Center Loss attracts the features of the same class to their centers achieving compactness.

Despite its usefulness, the Center Loss has some limitations: the feature centers are extra parameters stored outside the network that are not jointly optimized with the network parameters. Indeed, they are updated with an autoregressive mean estimator that tracks the underlying representation changes at each step. Moreover, when a large number of classes must be learned, mini-batches do not provide enough samples for a correct estimation of the mean. Center Loss also requires a balancing between the two supervision losses which typically requires a search over the balancing hyper-parameter.

Some works have successfully addressed all the issues described above importing intra-class feature compactness directly into the Softmax loss. This class of methods, including [2, 175, 176, 178, 208], avoids the need of an auxiliary loss (as in the Center Loss) with the possibility of including a margin between the class decision boundaries, all in a single Softmax loss. Other successful works follow a nearly opposite strategy by removing the final classification layer and learn directly a distance evaluated on image pairs or image triplets as shown in [166] and in [167] respectively. Despite the performance results, carefully designed pair and triplet selection is required to avoid slow convergence and instability.

Except for few recent cases [1, 184, 209] inter-class separability and compactness are always enforced in a local manner without considering global inter-class separability and intra-class compactness. For this purpose, the work [184] uses an auxiliary loss for enforcing global separability. The work [209] use an auxiliary loss similar to [184] for enforcing global separability and a further margin loss to enforce compactness. Regular Polytope Networks [157], use a fixed classifier in which the parameters of the final transformation implementing the classifier are *not* subjected to learning and are set with values taken from coordinate vertices of a regular polytope. This avoids optimizing for maximal separation as in [209] and [184] since regular polytopes naturally provide distributed vertices (i.e. the classifier weights) at equal angles maximizing the available space (Sect. 4.4.3).

In this Section we address *all* those limitations including global inter-class separability and compactness in a maximal sense without the need of any auxiliary loss. This is achieved by exploiting the Regular Polytope fixed classifiers and improving their feature compactness according to the additive angular margin described in [2]. As illustrated in Fig. 4.11, the advantage of the proposed combination is the capability of generating global maximally separated and compact features (shown in light blue) angularly centered around the vertices of polytopes (i.e. the classifier fixed weights shown in red). In particular, the angle φ subtended between a class weight and its connected class weights is constant and maximizes inter-class separability in the available space. The angle φ is further exploited to obtain the maximal compactness by setting the angular margin between the features to φ (i.e. the maximum allowed margin). The advantage of our formulation is that the margin is no longer an hyperparameter that have to be searched since it is obtained from a closed form solution.

Although, Eq. 4.3 directly optimizes for small angles, only partial intra-class compactness can be enforced. Fig.4.12 shows (from left to right) the distribution of features learned from the MNIST dataset with the three different classifiers. The features are displayed as a collection of points, each having the activation of one feature coordinate determining the position on the horizontal axis and the value of the other feature coordinate activation determining the position on the vertical axis. All the pairwise scatter plots of the feature activation coordinates are shown and feature classes are color coded. The size of the scatter plot matrices follows the size of the feature dimensionality d of each fixed classifier which can be determined according to the number of classes K as:

$$d = K - 1, \qquad d = \lceil \log_2(K) \rceil, \qquad d = \left\lceil \frac{K}{2} \right\rceil, \tag{4.16}$$

respectively. The scatter plot matrices therefore result in the following di-

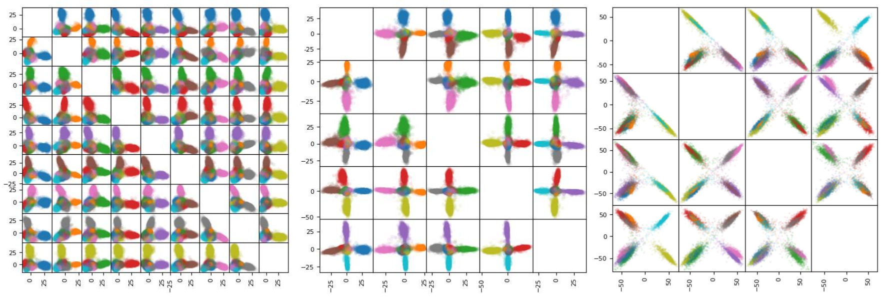


Figure 4.12: The distribution of features learned from the MNIST dataset using the RePoNet classifiers. Features are shown (from left to right) with a scatter plot matrix for the d-Simplex, d-Orthoplex and d-Cube classifier respectively. It can be noticed that features are distributed following the symmetric vertex configuration of polytopes. Although features are maximally separated, their compactness is limited.

mensions: 9×9, 5×5 and 4×4 respectively. As evidenced from the figure, the features follow the symmetric and maximally separated vertex configurations of their corresponding polytopes. This is due to the fact that each single pairwise scatter plot is basically a parallel projection onto the planes defined by pairs of multidimensional axes. According to this, features assume a $\curlywedge$, $+$, and $\times$ shaped configuration for the d-Simplex, d-Orthoplex and d-Cube respectively. Although maximal separation is achieved, the intra-class average distance is large and therefore not well suited for recognition purposes. The plotted features are obtained training the so called LeNet++ architecture [198]. The network is a modification of the LeNet architecture [199] to a deeper and wider network including parametric rectifier linear units (pReLU) [200]. The network is learned using the Adam optimizer [210] with a learning rate of 0.0005, the convolutional parameters are initialized following [211] and the mini-batch size is 512.

To improve compactness keeping the global maximal feature separation we follow [178,208] normalizing the features and multiplying them by a scalar κ: $\hat{\mathbf{f}}_i = \frac{\mathbf{f}_i}{||\mathbf{f}_i||}\kappa$. The loss in Eq.4.3 can be therefore rewritten as:

$$\begin{aligned}\mathcal{L} &= -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\kappa_i \hat{\mathbf{w}}_{y_i}^{\top}\hat{\mathbf{f}}_i)}{\sum_{j=1}^{K}\exp(\kappa_i \hat{\mathbf{w}}_{j}^{\top}\hat{\mathbf{f}}_i)}\right)\\ &= -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\kappa\cos(\theta_{y_i}))}{\sum_{j=1}^{K}\exp(\kappa\cos(\theta_j))}\right)\end{aligned} \tag{4.17}$$

The equation above minimizes the angle θ_{y_i} between the fixed weight corresponding to the label y_i and its associated feature. The equation can be interpreted as if features are realizations from a set of K von Mises-Fisher distributions having a common concentration parameter κ. Under this parameterization $\hat{\mathbf{w}}$ is the mean direction on the hypersphere and κ is the concentration parameter. The greater the value of κ the higher the concentration of the distribution around the mean direction $\hat{\mathbf{w}}$ and the more compact the features. This value has already been discussed sufficiently in several previous works [175,178]. In this work, we directly fixed it to 30 and will not discuss its effect anymore.

To obtain maximal compactness the additive angular margin loss described in [2] is exploited. According to this, Eq.4.17 becomes:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\kappa\cos(\theta_{y_i}+m))}{\exp(\kappa\cos(\theta_{y_i}+m)) + \sum\limits_{\substack{j=1\\ j\neq y_i}}^{n}\exp(\kappa\cos(\theta_j))}\right), \tag{4.18}$$

where the scalar value m is an angle in the normalized hypersphere introducing a margin between class decision boundaries. The loss of Eq. 4.18 together with the fixed classifier weights of Eqs. 4.4, 4.7, 4.10 allows learning discriminative features without using any auxiliary loss other than the Softmax.

The advantage of our formulation is that m is no longer an hyperparameter that have to be searched. Indeed, the loss above when used with RePoNet classifiers is completely interpretable and the margin m can be set according to the angle φ subtended between a class weight and its connected class weights as illustrated in Fig.4.11. For each of the three RePoNet fixed

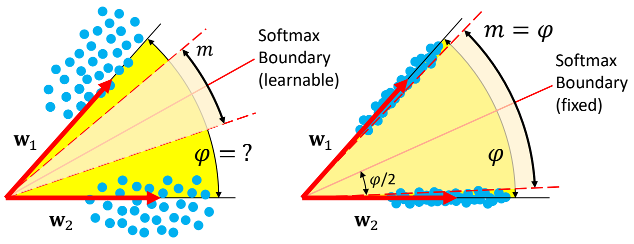


Figure 4.13: Maximally compact feature learning with RePoNet fixed classifiers and the angular margin loss. *Left*: In a standard learnable classifier the decision boundaries (dashed lines) defined by the angular margin m do not push features to their respective weights uniformly (red arrows). *Right*: In RePoNet classifiers the margin can be analytically determined ($m = \varphi$) so that the decision boundaries maximally push the features closer to their respective fixed weight.

classifiers the angle φ can be analytically determined as:

$$\varphi_s = \arccos\left(-\frac{1}{d}\right), \tag{4.19}$$

$$\varphi_o = \frac{\pi}{2}, \tag{4.20}$$

$$\varphi_c = \arccos\left(\frac{d-2}{d}\right), \tag{4.21}$$

respectively, where d is the feature space dimension size.

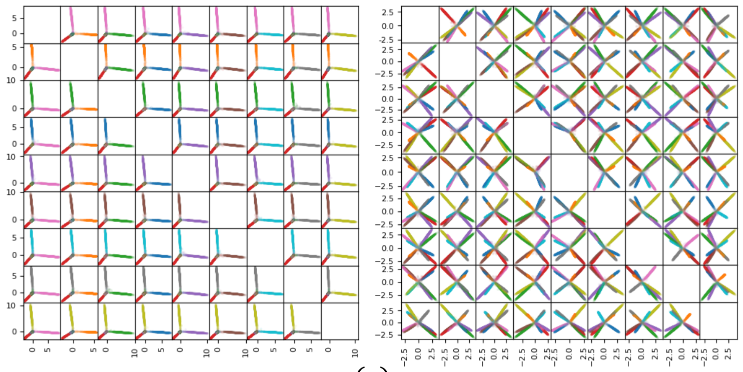
Fig. 4.13 shows the effect of setting $m = \varphi$. We draw a schematic 2D diagram to show the effect of the margin m on pushing the class decision boundary to achieve feature compactness. In the standard case of a learnable classifier, as shown in Fig. 4.13 *(left)*, the value φ is not known in advance, it varies from class to class and features are not guaranteed to distribute angularly centered around their corresponding weights. Therefore, m cannot be set in an interpretable way. Contrarily, in the case proposed in this Chapter and shown in Fig. 4.13 *(right)*, the value φ is constant and known in advance, therefore by setting $m = \varphi$, the class decision boundaries are maximally pushed to compact features around their fixed weights. This because the Softmax boundary (from which the margin is added) is exactly in between the two weights $\mathbf{w}_1$ and $\mathbf{w}_2$. According to this, the features generated by the proposed method are not only *maximally separated* but also *maximally compact* (i.e. maximally discriminative).

4.6.1 Qualitative Results

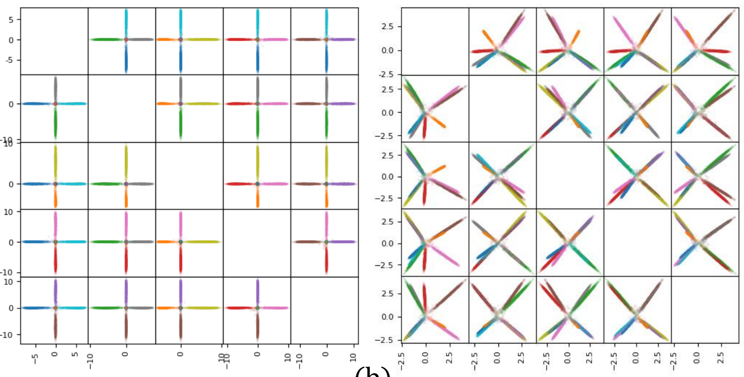
Experiments are conducted with the well-known MNIST and EMNIST [193] datasets. Fig. 4.14 shows a visual comparison between the features generated by the RePoNet fixed classifiers (*left column*) and by a standard CNN baseline with learnable classifiers (*right column*). Both approaches are trained according to the loss of Eq. 4.18 and have exactly the same architecture, training settings and embedding feature dimension used in Fig. 4.12. Results are presented with a scatter plot matrix. Although the two methods achieve substantially the same classification accuracy (i.e. 99.45% and 99.47% respectively), it can be noticed that the learned features are different. Specifically, Margin-RePoNet follows the exact configuration geometry of their related polytopes. Features follow very precisely their relative $\curlywedge$, +, and $\times$ shapes therefore achieving maximal separability. The standard baselines with learnable classifiers (Fig. 4.14 *left column*) achieve good but non maximal separation between features. However, as the embedding dimension decreases, as in Fig. 4.14(c), the separation worsens.

This effect is particularly evident in more difficult datasets. Fig.4.15 shows the same visual comparison using the EMNIST dataset where some of the 47 classes are difficult to be correctly classified due to their inherent ambiguity. Fig. 4.15 shows the scatter plot matrix of the d-Cube classifier (*left*) compared with its learnable classifier baseline (*right*) in dimension $d = 6$. Although also in this case they both achieved the same classification accuracy (i.e. 88.31% and 88.39%), the features learned by the baseline are neither well separated nor compact.

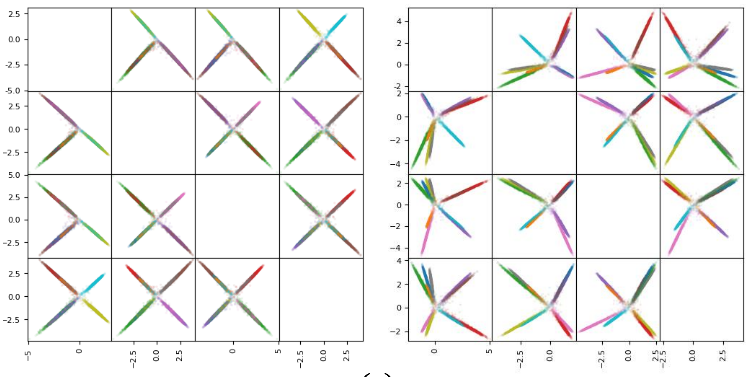
Finally, in Fig. 4.16 we show the L_2 normalized features (typically used in recognition) of both the training (*top*) and test set (*bottom*) for the same experiment shown in Fig. 4.15. Class features in this case correctly follow the vertices of the six-dimensional hypercube since all the parallel projections defined by each pairwise scatter plot result in the same unit square centered at the origin.



(a)



(b)



(c)

Figure 4.14: The distribution of MNIST learned features using the proposed method (*Left*) and learned using a standard trainable classifier (*Right*). The scatter plot highlights the maximal separability and compactness of the extracted features for the (a) d-Simplex, (b) d-Orthoplex and (c) d-Cube classifiers. Class features are color coded. As the feature space dimension decreases standard baselines have difficulty in obtaining inter-class separation.

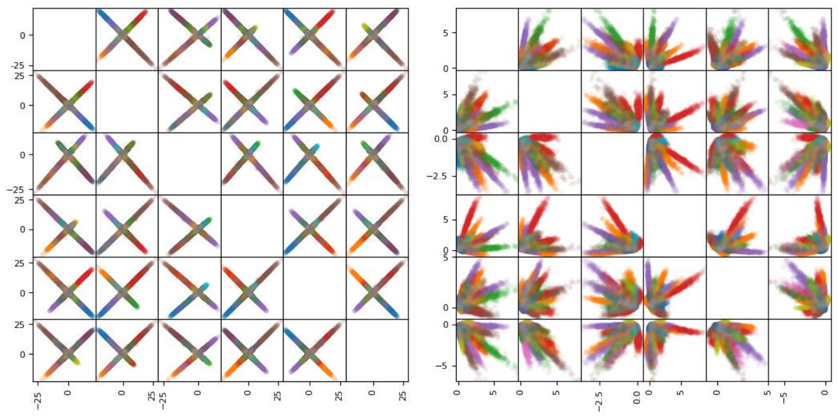
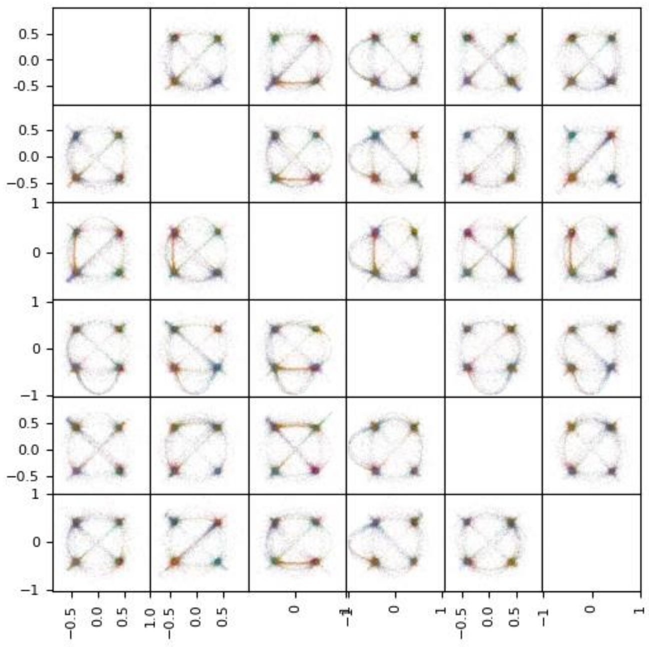


Figure 4.15: The distribution of EMNIST (balanced split) learned features. *Left*: Features learned using the *d*-Cube fixed classifier with the additive angular margin loss. *Right*: Features learned using a standard trainable classifier with the additive angular margin loss. In both cases the feature dimension is 6 and the classification accuracy is comparable. Maximal separability and compactness are evident in our approach.



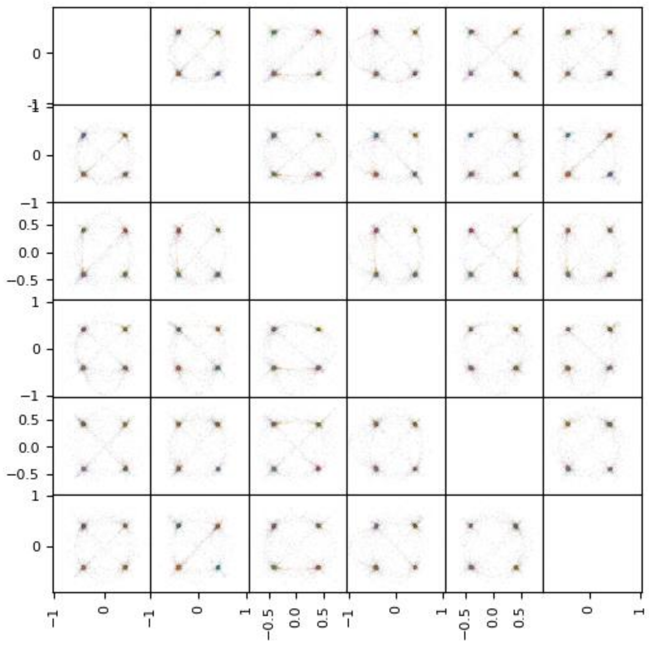


Figure 4.16: The distribution of the EMNIST *normalized* learned features shown in 4.15 *(Left)*. (*Top*) training-set. (*Bottom*) test-set (best viewed in electronic version).

4.7 Discussion: Potential and Challenges

Our finding may have implications in those Deep Neural Network learning contexts in which a classifier must be robust against changes of the feature representation while learning as in incremental learning settings, especially when features are stored in memory banks while learning as introduced in [109, 212, 213]. Despite recent advances, methods inspired by memory-augmented deep neural networks are still limited when it comes to incremental learning. The method [214] simplifies the original fully differentiable end-to-end idea. Except for the nearest-neighbor query to the memory bank, their approach is fully differentiable, can be trained end-to-end and operates in a incremental manner (i.e. without the need to reset it during training). However, the features stored in the memory bank remain fixed (i.e. they are not undergoing learning) and only the memory bank is learned. Our approach may have a promising potential for learning both the feature and the memory without considering their joint learning. The intuition is that every time the internal feature representation changes the memory bank must be relearned from scratch. Our method can mitigate the need of feature relearning by keeping the compatibility of features between learning steps thanks to feature stationarity. Concurrent to this work, [215] addresses a similar problem in terms of feature "back-compatibility" and exploits a pre-trained fixed classifier to avoid re-indexing a memory bank containing the gallery features of a retrieval system that has been updated.

This basic idea can be in principle applied to the many computer vision tasks that have benefited from memory based learning as: [40, 216] for cumulative learning of face appearance models from video stream, [217–221] for object detection, [222] for video object segmentation and [223] for visual object tracking. The works [217–221] accumulate context from pre-computed feature banks (with fixed pre-trained feature extractors i.e. not undergoing learning). These feature banks extend the time horizon of their network up to 60 second in [220] or to one month in [217] achieving strong results on spatio-temporal localization. The works [40, 216] accumulate extracted face features in a memory bank to preserve all the past knowledge without forgetting and at the same time handle the non-stationarity of the data stream. At a high level, all these approaches can be framed as a non-parametric estimation method (like nearest neighbors) sitting on top of a high-powered parametric function (Faster R-CNN in the case of object detection [217] or a face features extractor in [40] and [216] or SiamFC feature extractor [224] for object tracking in [223]). These methods use a fixed representation which is not incrementally learned as it would require re-encoding all the images in the memory bank. Avoiding re-encoding images can be advantageous in applications in which images cannot be stored for privacy reasons (i.e. face recognition, applications in medical imaging, etc.).

4.7.1 Conclusion

We have shown that a special set of fixed classifiers based on regular polytopes generates stationary features by maximally exploiting the available representation space. The proposed method is simple to implement and theoretically correct. Experimental results confirm both the theoretical analysis and the generalization capability of the approach across a range of datasets and architectures. RePoNet improves and generalizes the concept of a fixed classifier, recently proposed in [156], to a larger class of fixed classifier models exploiting the inherent symmetry of regular polytopes in the feature space.

Our finding may have implications in all of those Deep Neural Network learning contexts in which a classifier must be robust against changes of the feature representation while learning as in incremental and continual learning settings.

Chapter 5

Class-incremental Learning with Pre-allocated Fixed Classifiers

In class-incremental learning, a learning agent faces a stream of data with the goal of learning new classes while not forgetting previous ones. Neural networks are known to suffer under this setting, as they forget previously acquired knowledge. To address this problem, effective methods exploit past data stored in an episodic memory while expanding the final classifier nodes to accommodate the new classes. In this work, we substitute the expanding classifier with a novel fixed classifier in which a number of pre-allocated output nodes are subject to the classification loss right from the beginning of the learning phase. Contrarily to the standard expanding classifier, this allows: (a) the output nodes of future unseen classes to firstly see negative samples since the beginning of learning together with the positive samples that incrementally arrive; (b) to learn features that do not change their geometric configuration as novel classes are incorporated in the learning model. [1] [2]

[1] Part of this chapter has been published as "Class-incremental Learning with Pre-allocated Fixed Classifiers" to *25th International Conference on Pattern Recognition*

[2] *Acknowledgments:* This work was partially supported by the Italian MIUR within PRIN 2017, Project Grant 20172BH297: I-MALL.

5.1 Introduction

Natural intelligent systems learn incrementally by continuously receiving information over time. They learn new concepts adapting to changes in the environment by leveraging past experiences. A remarkable capability of these systems is that learning of new concepts is achieved while *not* forgetting previous ones. In contrast, current Deep Learning models, when updated with novel incoming data, suffer from *catastrophic forgetting*: the tendency of Neural Networks to completely and abruptly forget previously learned information [5–7]. This problem is related to the plasticity/stability dilemma in incremental learning [10]. Too much "plasticity" leads to catastrophic forgetting, too much "stability" leads to an inability to adapt to novel information. Continual Learning [8, 9] specifically addresses this problem, bringing machine learning closer to natural learning. In this learning scenario, the agent is presented with a stream of tasks and each new task is learned by reusing and combining the knowledge acquired while learning previous tasks. As the learning agent is processing a stream, it cannot store all examples seen in the past.

Continual learning has recently received increasing attention and several methods have been developed [225–232]. However, despite the intense research efforts, the gap in performance with respect to offline multi-task learning makes continual learning an open problem. Most of the techniques have focused on a sequence of tasks in which both the identity of the task (task label) and boundaries between tasks are provided [12–15]. Thus, many of these methods fail to capture real-world continual learning, with unknown task labels [16] [17]. A typical example that illustrates the difference between using or not the task labels is the Split MNIST experiment, in which the ten digits of the well known handwritten dataset are split into five classification tasks of two-class each. The model has five different final classification layers, one for each task. Those classifiers (i.e. output heads) are indexed by the task identity (1 to 5) that is given at testing time.

This scenario is shown to be easier than class-incremental learning (CIL) since the selection of the output head is given by the task identity [16]. CIL is typically addressed with single-headed variants that do not require task identity, where the model always performs prediction over all classes (i.e. all digits 0 to 9) [9, 16, 17, 25, 233, 234].

In CIL the single head final layer of a Neural Network is *expanded* with an output node when a new class arrives (multiple new classes are expanded

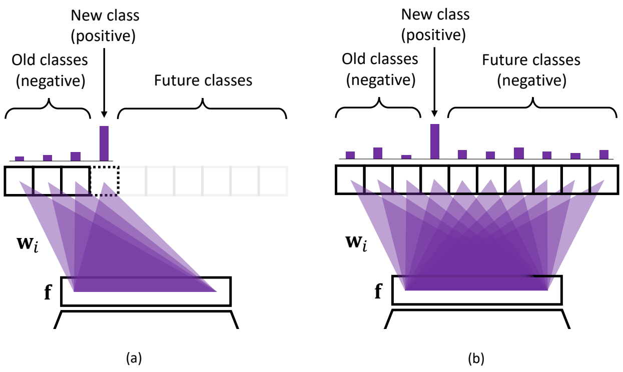


Figure 5.1: Class-incremental classifiers. *(a)*: Expanding classifier. *(b)*: Pre-allocated classifier. The latter can exploit unseen future classes as negative examples.

with multiple nodes); thus, in general, during learning, an output node sees, according to the samples in the current random batch, positive and negative samples in the *newly arrived* class and in the *old seen classes* (i.e. the remaining), respectively (Fig.5.1(a)).

In this Chapter, we address CIL using a novel classifier in which a number of pre-allocated output nodes are subject to the classification loss right from the beginning. This allows the output nodes of yet *unseen classes* to firstly see negative samples since the beginning of learning together with the positive samples that incrementally arrive (Fig.5.1(b)). Contrarily to the *expanding* classifier, in our formulation, the output nodes can learn from the beginning of the learning phase. This is achieved by *pre-allocating* a special classifier with a large number of output nodes in which the weights are *fixed* (i.e. not undergoing learning) and set to values taken from the coordinate vertices of regular polytopes [1]. The classification layer so defined has two intriguing properties. The first is that the features do not change their geometric configuration as novel classes are incorporated in the learning model. The second is that a very large number of classes can be pre-allocated with no loss of accuracy. This allows the method to meet the underlying assumption of lifelong learning as for the case of the expanding classifier.

5.2 Related Works

5.2.1 Continual Learning

Continual learning has been extensively studied in literature [8, 9]. Prior works can be broadly categorized into three main categories: (1) regularization, (2) dynamic architecture methods and (3) episodic memory-based (also termed Experience Replay).

Regularization. Regularization-based approaches reduce forgetting by restricting the updates in parameters that were important for previous tasks. Elastic Weighted Consolidation (EWC) imposes constraints on network parameters to reproduce biological mechanism of consolidation [12]. Online-EWC [20] optimizes EWC approach for multiple tasks, overcoming the complexity of original EWC which scales linearly with number of tasks. Synaptic Intelligence (SI) also replicates biological mechanism of synapses, preventing parameters (synapses) to change based on the relevance of each parameter for the considered task [14]. Memory Aware Synapses (MAS) tackles the problem in a similar fashion, based on the relevance of each parameter to the task. When the number of tasks is large, the regularization of past tasks becomes obsolete, leading to representation drift [235].

Dynamic Architecture. Second, dynamic architecture or modular approaches add new modules to the model architecture as new tasks are learned. [22] grows a network searching for similarities between known classes and unseen classes, organizing them into a hierarchy. Predictions are made by visiting the hierarchy, from the superclasses down to the specific class. [23] exploits boosting algorithm to control network architecture growth balancing its complexity with empirical risk minimization. The work in [24] proposes a network structure organized in columns. Each column is a network which learns a new task, sharing features learned by other columns via lateral connections. Sparse regularization is employed in [236] to decide how many parameters add to each network layer when new tasks are learned. Then, selective retraining is performed.
While modular architectures overcome forgetting by design, these approaches do not scale with the number of tasks as memory requirements increase with the number of tasks.

Experience Replay (ER). Third, Experience Replay methods store a few examples from past tasks in an "episodic memory", to be revisited when training for a new task. In contrast to modular approaches, memory-based methods add a relatively small memory overhead for each new task. The concept of experience replay in class-incremental learning has been introduced in [25]. By means of rehearsal technique, new and old data are combined when new tasks are learned in order to prevent catastrophic forgetting of old tasks. [26] presents Gradient Episodic Memory technique which does not store and reuse old samples but allows transfer learning between tasks by storing old gradients and updating them to prevent forgetting. An improved, memory-efficient version of GEM is obtained by considering the average of the losses of all tasks rather than each individual loss of single tasks [27]. Full data rehearsal may prevent catastrophic forgetting, but it is unfeasible due to important memory impact, so [28] implements memory-efficient buffer techniques to perform rehearsal without the need for retaining all samples. [29] aims at finding data distribution which can keep optimal performance level overall tasks. This is achieved by choosing an adequate strategy to build data memory, exploiting the biological mechanic of replaying experiences. [30] introduces a technique to avoid learning interference provoked by data coming from different source domains. Dual-memory incremental learning is exploited in [31] to keep track of statistics of past classes, in order to rebalance their prediction scores in later stages of learning. Memory-based methods have currently shown state-of-the-art.

5.2.2 Fixed Classifiers

Dynamically freezing[3] weights is a form of implicit dynamic architecture in which some selected weights are not undergoing learning [233, 237–239]. When freezing is applied to the final classification layer, class decision boundaries remain stationary during learning. This was exploited in [240, 241] to reduce catastrophic forgetting in domain adaptation. They show that a frozen classifier, together with a distillation loss on the features preserve the geometric configuration of old classes. These two constraints are recently shown to be resilient to catastrophic forgetting in class incremental learning, and in incremental few-shot learning [227, 230], respectively. Freezing the classifier is also the key strategy used in [215] to learn visual features that are compatible with features computed with different CNN models. Compatibility between features from different models learned at different times means that if such features are used to compare images, then “new” features

[3]The terms frozen and fixed are used interchangeably.

can be compared directly to “old” features, so they can be used interchangeably. This enables visual search systems to avoid re-computing new features for all previously seen images when updating the models. This capability may enable incremental learning of features in more realistic scenarios in which classes can be revisited [138], [47], [242], [243].

Fixing the final classification layer in multi-task supervised classification has been explored in detail in [1,156,244,245] showing that it causes little or no reduction in accuracy for common datasets, while allowing a noticeable reduction in trainable parameters. Fixed classifiers have also an important role in theoretical convergence analysis of training neural network models with batch-norm [169]. It is shown recently in [170] that CNNs with a fixed classifier and batch-norm in each layer establish a principle of equivalence between different learning rate schedules.

5.3 Contributions

- We introduce a novel approach for class-incremental learning that keeps features in a constant specific spatial configuration distributed at equal angles maximizing the available space.

- The approach exploits negative samples from unseen classes since the beginning of learning.

- We achieve similar results with respect to several important baselines on standard benchmarks.

5.4 Proposed Method

5.4.1 Class Incremental Learning Setting

In continual learning, a stream of data triplets (x_i, y_i, t_i) containing an input x_i, a target y_i, and a task identifier $t_i \in \mathcal{T} = \{1, \dots, T\}$ are presented to the learning agent. Each input-target pair $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}_{t_i}$ is an identically and independently distributed (i.i.d.) example drawn from an unknown distribution $P_{t_i}(X, Y)$ that represents the t_i-th learning task. We assume that the tasks are learned in order: $t_i \le t_j$ for all $i \le j$, and that the number of tasks T is not known a priori. Specifically in CIL: a single classifier is learned and the task-membership t_i is ignored. Under this setup, the goal is to estimate a Neural Network based model

$$f_\theta = (\mathbf{w} \circ \Phi) : \mathcal{X} \times \mathcal{T} \to \mathcal{Y}, \tag{5.1}$$

parameterized by $\theta \in \mathbb{R}^p$ where p is the number of parameters of f_θ. The Neural Network model is composed of a feature extractor $\Phi : \mathcal{X} \to \mathcal{H}$ and a classifier $\mathbf{w} : \mathcal{H} \to \mathcal{Y}$, that minimize the multi-task loss

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim P_t} \left[\ell(f(x,t), y) \right], \tag{5.2}$$

where $\mathcal{Y} = \cup_{t\in\mathcal{T}} \mathcal{Y}_t$, and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function.

Following [26], we evaluate the performance of class-incremental learning algorithms according to the *final average accuracy* defined as

$$\text{Accuracy} = \frac{1}{T} \sum_{j=1}^{T} a_{T,j}, \tag{5.3}$$

where $a_{i,j}$ denotes the test accuracy on task j after the model has finished learning the task i. That is, the final average accuracy measures the test performance of the model at every task after the continual learning experience has finished.

5.4.2 Class-Incremental Learning with a Pre-allocated Regular Polytope Classifier

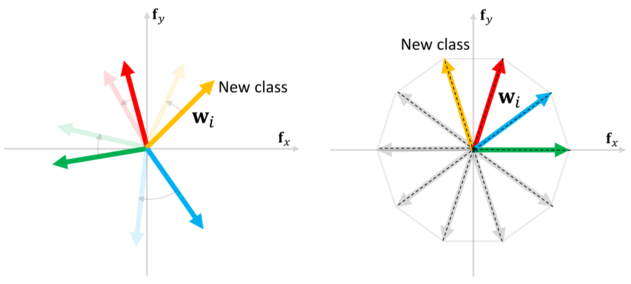
As based on Experience Replay (ER), our method learns the model f_θ by storing few past observed triplets in an episodic memory $\mathcal{M} = \{(x', y', t')\}$. For every new mini-batch of observations $\mathcal{B} = \{(x, y, t)\}$, the learner samples a mini-batch $\mathcal{B}_\mathcal{M}$ from $\mathcal{M}$ at random, and applies the rule

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \, \ell(\mathcal{B} \cup \mathcal{B}_\mathcal{M})$$

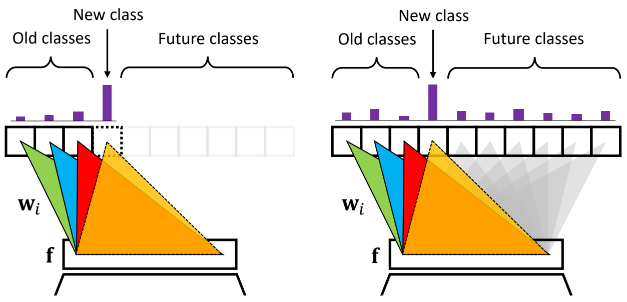
to update the parameters of f_θ, where

$$\ell(\mathcal{A}) = \frac{1}{|\mathcal{A}|} \sum_{(x,y,t)\in\mathcal{A}} \ell(f_\theta(x,t), y)$$

denotes the average loss across a collection of triplets $\mathcal{A}$. Since we address class-incremental learning (i.e. single output head) the task identifiers t and t' are ignored.



(a)



(b)

Figure 5.2: Class-incremental learning classifiers. Comparison between a standard *expanding* classifier (left) and a *pre-allocated* Regular Polytope Classifier (RPC) (right). Both classifiers are shown with three learned classes (green, blue, red) and with a new class under learning (orange). *(a):* The weights of the classifiers are represented in feature space $\mathbf{f}$. The figure illustrates the situation in a 2D scenario, the characterization extends to arbitrary dimensions. As a new class is learned old classes move to include the new one, the motion is shown in transparent colors. This effect is not present in our RPC. *(b):* The final layers of the classifiers in a Neural Network architecture. The pre-allocated RPC weights (grey) can receive negative samples and adapt the network from the beginning of learning as new classes arrive. The logit responses of each class are also shown in purple.

As firstly noted in [25], in this learning condition, it is problematic that the classifier weight vectors $\mathbf{w}$ are decoupled from the feature extraction routine Φ: whenever Φ changes in Eq.5.1, all $\mathbf{w}$ must be updated as well. Otherwise, the network outputs will change uncontrollably, which is observable as catastrophic forgetting. Changes in the extracted features are mainly due to the inclusion of novel classes, that is, when a novel class is incorporated in the neural network model, the classifier weights of the other old classes move to create space to accommodate the novel one. Under the assumption of normalized weights and zero biases for the classifier, as proposed in [178] and [176], the classifier weights are constrained in the unit hypersphere and can be easily visualized. Fig. 5.2(a)(left) shows the geometric configuration of the classifier weights in feature space $\mathbf{f}$. As the new class (orange) is learned, old classes (green, blue, red) move changing their spatial configuration to include the new one. The motion is shown in transparent colors. Basically the set of weights forming an irregular triangle transforms into a set of weights forming an irregular quadrilateral and eventually, if a further class is introduced, the set of weights transforms into an irregular pentagon and so on. As new classes are continually incorporated into the model, the classifier continues to change its configuration (without stopping) with its corresponding features following a similar motion. The same transformative pattern occurs in higher dimensional spaces where sets of weights form convex polytopes. Fig. 5.2(b)(left) shows the corresponding expanding classifier in which the three old classes have been already learned (green, blue, red) and a new class is undergoing learning (orange).

In order to avoid this continuous motion of features, our approach uses a pre-allocated special fixed classifier (i.e. not undergoing the learning procedure) that keeps the features of the learned classes in a constant specific spatial configuration as novel classes are incorporated into the learning model. This allows to partially handle the catastrophic forgetting effect of the final classifier layer. Fig. 5.2(a)(right) shows the proposed fixed classifier consisting of a number of pre-allocated directions (grey) distributed at equal angles maximizing the available feature space with the purpose of defining class decision regions (delimited by decision boundaries) of equal extension for each class. Fig. 5.2(b)(right) shows the corresponding pre-allocated fixed classifier.

The number of pre-allocated classes is typically large because the number of class K is not known a priori, however, this design choice allows our method to receive and learn from negative examples since the beginning of the data stream. More specifically, Fig. 5.2(a)(right) and Fig. 5.2(b)(right) show how this is achieved: at each learning update, the weights of the classifier keep the same constant position [1]. By fixing the weights, the trainable classifier is superseded by previous layers. Fixed classifiers are shown recently to cause little or no reduction in classification performance for common datasets while allowing a noticeable reduction in trainable parameters, especially when the number of classes is large [1, 156].

Since no prior assumption about the semantic similarity between future classes can be made, in order to define the fixed classifier the natural assumption is to use the d-Simplex regular polytope. With the d-simplex, all classes are nearest to all other. In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. Specifically, a d-simplex is a d-dimensional polytope which is the convex hull of its $d+1$ vertices. A regular d-simplex may be constructed from a regular $(d-1)$-simplex connecting a new vertex to all original vertices by the common edge length. According to this, the weights for this classifier can be computed as:

$$\mathbf{W}_{\mathcal{S}} = \left\{ e_1, e_2, \dots, e_{d-1}, \alpha \sum_{i=1}^{d-1} e_i \right\}$$

where $\alpha = \frac{1-\sqrt{d+1}}{d}$ and e_i with $i \in \{1, 2, \dots, d-1\}$ denote the standard basis in $\mathbb{R}^{d-1}$. The final weights will be shifted about the centroid and normalized. The d-Simplex fixed classifier defined in an embedding space of dimension d, can accommodate a number of classes K equal to its number of vertices:

$$K = d + 1. \tag{5.4}$$

This classifier has the largest number of classes that can be embedded in $\mathbb{R}^d$ such that their corresponding class features are equidistant from each other.

5.5 Experimental Results

We perform experiments on four classfication benchmarks for continual learning: SplitMNIST, PermutedMNIST, SplitCIFAR10 and SplitCIFAR100. SplitMNIST splits the MNIST dataset of handwritten digits [248] to create five different tasks with non-overlapping classes. PermutedMNIST is a variant of the MNIST dataset where each task applies a fixed random pixel permutation to the original. A total of ten tasks of ten classes each is created. Although this dataset is unrealistic from the point of view of how images are formed [16], PermutedMNIST allows evaluating and understanding class-incremental learning systems in the extreme case in which tasks are unrelated

Table 5.1: Accuracy (Eq. 5.3) of class incremental learning experiments. Averages and standard deviations are computed over 10 runs using different random seeds.

Method	$\|\mathcal{M}\|$	SplitMNIST	PermutedMNIST	SplitCIFAR10	$\|\mathcal{M}\|$	SplitCIFAR100
EWC [246]	-	19.92 ± 0.06	19.51 ± 0.05	16.89 ± 0.03	-	17.17 ± 0.12
Online-EWC [20]	-	19.93 ± 0.05	31.63 ± 0.11	17.28 ± 0.09	-	17.29 ± 0.06
SI [14]	-	21.01 ± 0.10	18.27 ± 0.07	17.81 ± 0.12	-	14.21 ± 0.29
MAS [247]	-	23.01 ± 0.31	17.53 ± 0.53	17.75 ± 0.87	-	16.97 ± 0.05
GEM [26]	100	74.92 ± 2.97	31.03 ± 3.19	24.48 ± 3.21	1400	22.83 ± 2.17
GEM [26]	1100	95.16 ± 0.15	79.44 ± 0.23	45.48 ± 0.19	5600	N.A.
Expanding Classifer	100	80.10 ± 3.29	64.18 ± 2.85	31.74 ± 2.15	1400	33.79 ± 0.56
Expanding Classifer	1100	96.25 ± 0.28	93.52 ± 0.41	66.93 ± 0.48	5600	51.76 ± 0.55
Pre-allocated RPC (Ours)	100	82.32 ± 2.19	81.43 ± 2.12	31.80 ± 1.61	1400	33.80 ± 0.42
Pre-allocated RPC (Ours)	1100	96.90 ± 0.29	94.63 ± 0.35	67.44 ± 0.50	5600	51.77 ± 0.61

to each other [17, 229]. SplitCIFAR10 and SplitCIFAR100 are variants of the CIFAR10 and CIFAR100 datasets, respectively [14, 249]. In CIFAR100 each task contains the data pertaining to twenty random classes out of the total 100 classes. This benchmark contains five tasks. In CIFAR10 a total of five tasks of two classes each are created.

We implemented our fixed classifier on top of the LeNet architecture [199] for the MNIST and PermutedMNIST datasets. Popular network architectures for ImageNet require modifications to adapt to the CIFAR 32x32 input size. According to this, for the SplitCIFAR10 and SplitCIFAR100 dataset our experiments follow publicly available implementations[4] and for our fixed classifier implemented on top of a ResNet56 architecture.

We compared our proposed model to the following baselines: EWC [12], Online EWC [20], SI [14], MAS [247], GEM [26] and Expanding Classifier with Experience Replay. For both the Expanding Classifier and our method, the mini-batch is constructed by an equal amount (64/64) of new data and the memory data. The buffer size is predefined to match the space overhead used by Online-EWC and SI, which translates to 1100 and 5400 images for the MNIST/CIFAR10 and CIFAR100 datasets, respectively [17].

For a fair comparison, all methods use the same neural network architecture. The classification loss function is the standard cross-entropy in all methods. All models are trained for 5 epochs per task with mini-batch size 128 using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate $= 0.001$) as the default, unless explicitly described. SplitCIFAR10 and SplitCIFAR100

[4] https://github.com/GT-RIPL/Continual-Learning-Benchmark/

Table 5.2: Ablation study on CIFAR-10 for different number of pre-allocated classes.

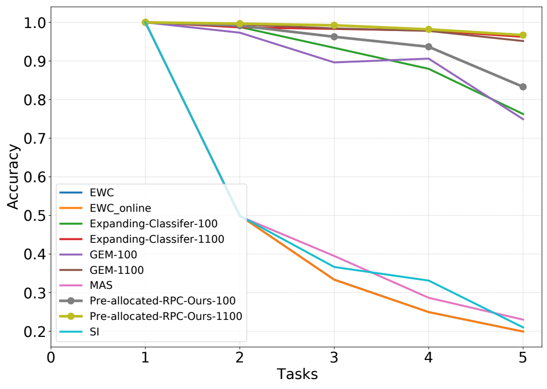
		SplitCIFAR10			
		#Pre-allocated Classes			
Method	$\vert\mathcal{M}\vert$	10	50	100	1000
Pre-allocated Expanding Classifer	100	31.74 ± 2.15	31.59 ± 0.96	31.64 ± 1.37	32.05 ± 1.10
Pre-allocated Expanding Classifer	1100	66.93 ± 0.48	66.57 ± 0.90	66.37 ± 1.20	66.75 ± 0.68
Pre-allocated RPC (Ours)	100	31.80 ± 1.61	32.20 ± 0.70	32.93 ± 1.50	32.27 ± 1.31
Pre-allocated RPC (Ours)	1100	67.44 ± 0.50	67.07 ± 0.12	67.10 ± 0.26	66.77 ± 0.52

Table 5.3: Ablation study on CIFAR-100 for different number of pre-allocated classes.

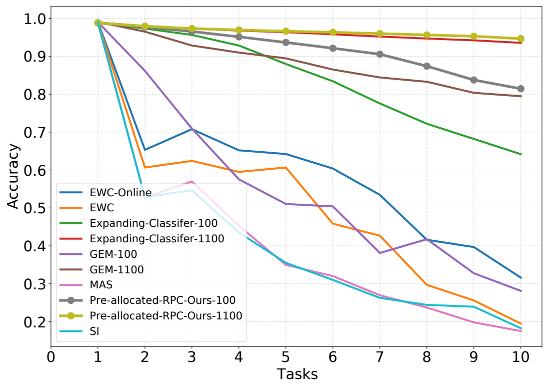
		SplitCIFAR100			
		#Pre-allocated Classes			
Method	$\vert\mathcal{M}\vert$	100	200	500	1000
Pre-allocated Expanding Classifer	1400	33.79 ± 0.56	33.07 ± 0.61	33.37 ± 0.33	33.55 ± 0.48
Pre-allocated Expanding Classifer	5600	51.76 ± 0.55	49.71 ± 0.63	49.18 ± 0.85	48.62 ± 0.96
Pre-allocated RPC (Ours)	1400	33.80 ± 0.42	33.43 ± 0.45	33.98 ± 0.38	33.92 ± 0.47
Pre-allocated RPC (Ours)	5600	51.77 ± 0.61	51.25 ± 0.39	51.28 ± 0.49	51.39 ± 0.52

are trained with 10 epoch per task. For reproducibility, all the results of the baselines are evaluated from scratch (no results are reported from papers) based on the unified code described in [17].

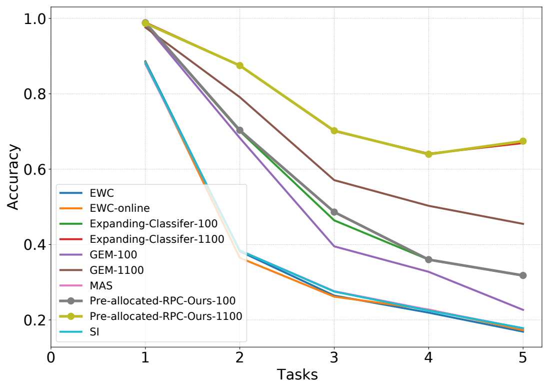
Tab. 5.1 summarizes the main results of our experiments. An ablation study of memory size is also included (100 and 1100 elements for SplitMNIST, PermutedMNIST and SplitCIFAR10; 1400 and 5600 elements for CIFAR100). First, our proposed method achieves similar accuracy (Eq.5.3) to the expanding classifier in all benchmarks. Second, the relative gains from the same methods using one order small memory, namely GEM and Expanding Classifier with 100 memory elements are significant, confirming that the pre-allocated fixed classifier allows Experience Replay methods to work better with less memory. Third, note that approaches making use of memory (GEM, ER based methods) work significantly better in this setup, while regularization methods such as EWC, Online-EWC, SI and MAS are suffering the class-incremental learning setting. Note that for GEM with 5600 memory elements evaluated in the SplitCIFAR100 dataset did not complete due to lack of GPU memory resources.



(a) SplitMNIST



(b) PermutedMNIST



(c) SplitCIFAR10

Figure 5.3: Evolution of Accuracy as new tasks are learned. *(a)*, *(b)* and *(c)* show the results for SplitMNIST, PermutedMNIST and SplitCIFAR10.

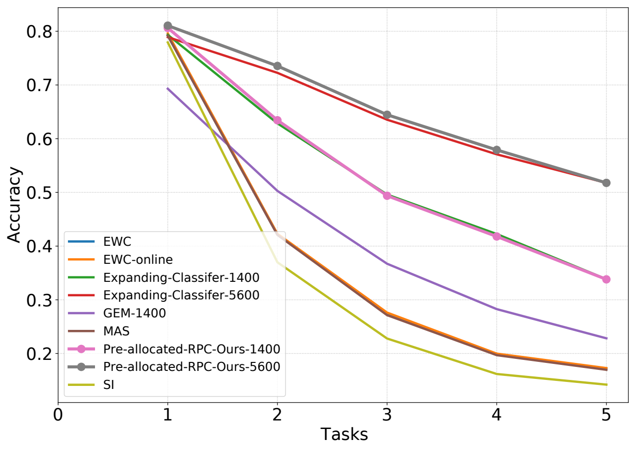
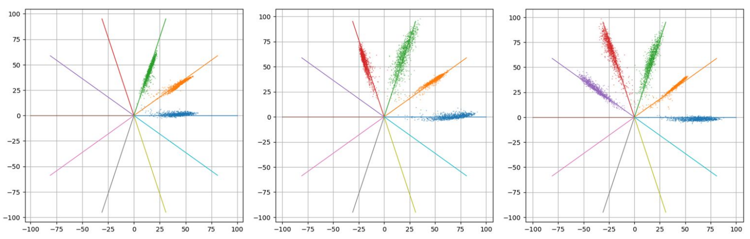


Figure 5.4: Evolution of Accuracy for the SplitCIFAR100 dataset.

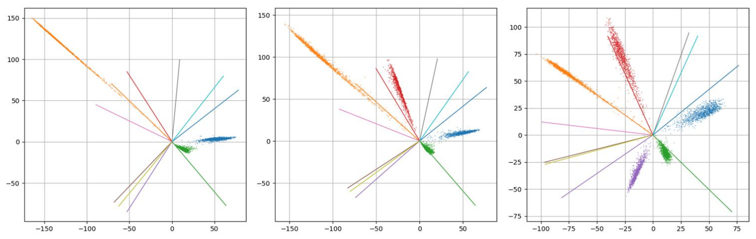
Fig. 5.3 shows a more detailed analysis of the average accuracy as new tasks are incrementally learned. More specifically, Fig. 5.3(a), (b), (c) and Fig. 5.4 show SplitMNIST, PermutedMNIST, SplitCIFAR10, and SplitCIFAR100, respectively. As evident from all the figures the performance of our approach is no worse than other baselines and in some cases, the accuracy is slightly higher. In the case of PermutedMNIST with 100 memory elements is clearly higher.

5.5.1 Ablation Study

We conducted an ablation study on the effect of pre-allocation on both the RPC classifier and the Expanding Classifier. Different number of pre-allocated classes are also evaluated. The quantitative results in Tab. 5.2 and Tab. 5.3 for the CIFAR10 and CIFAR100 datasets, respectively, show that in both cases the pre-allocation does not substantially affect the final performance. The advantage of our method is that the geometric configuration of features in our method does not change as novel classes are incorporated into the CNN model. This property can be appreciated in Fig. 5.5. In particular, Fig. 5.5(a) shows the evolution of the distribution of class features as learned by a 10-sided polygon fixed classifier and Fig. 5.5(b) shows the evolution of a standard pre-allocated Expanding Classifier. Both classifiers are implemented on top of the LeNet architecture and learned using the MNIST



(a)



(b)

Figure 5.5: Class Incremental Feature Learning on the MNIST dataset in a 2D embedding space. Figures (a) and (b) show the 2D features learned by the RPC classifier and by a standard trainable classifier, respectively. The two methods achieve the same classification accuracy. The figures show the training evolution of the classifier weights (colored lines) and their corresponding test-set class feature (2D point cloud) respectively. As it can be noticed, in (b) each novel learned class significantly and unpredictably changes the geometric configuration of already learned features. As shown in (a) this effect is absent in the RPC classifier. The figure is best viewed in color.

dataset. This toy example reduces the output size of the last hidden layer to 2 (i.e. the dimension of the feature is 2) so that we can directly plot the distribution of features on a 2D plane to ease visualization. A k-sided polygon is the equivalent of a Regular Polytope Classifier in a two dimensional space. While the figure illustrates the situation in $\mathbb{R}^2$, the characterization extends to arbitrary dimensions. The evolution of the incrementally learned features is shown starting from three classes and adding two more, one at a time, with ten pre-allocated classes. In the case of the pre-allocated Expanding Classifier, weights are randomly initialized. As it can be noticed in Fig. 5.5(b), each novel learned class, significantly and unpredictably changes the geometric configuration of the already learned features and already learned classifier weights. This effect is particularly evident in the green and blue colored classes when the new purple colored class is learned. Contrarily, as shown in Fig. 5.5(a), our method is not influenced by this effects: both features (colored point cloud) and weights (colored lines) do not change their relative geometric configuration. In addition to this, class feature distributions are perfectly aligned with their corresponding class weights.

5.6 Conclusion

We introduced a novel approach for class-incremental learning that exploits future unseen classes as negative examples and learns features that do not change their geometric configuration as novel classes are incorporated in the learning model. The approach uses a pre-allocated special fixed classifier (i.e. not undergoing the learning procedure) in which weights are set according to the vertices of the d-Simplex regular polytope. As shown in the experiments our method is as effective as the expanding classifier while exhibiting properties of internal feature representation that are otherwise not-existent.

Chapter 6

Conclusion and Future Challenges

In this thesis, we studied the problem of class-incremental learning using feature representations that are kept stationary during its evaluation. The methods we have addressed follows the main approaches used to tackle continual learning: a regularization strategy in the form of keeping the feature embedding fixed and an experience replay approach to replay past information when training a new task.

First in Chapter 2, we have presented a novel solution for cumulative learning face identities in unconstrained video streams based on face appearance. We discussed the substantial differences between our learning setting, Multiple Object Tracking and Continual Learning when applied to video streams. Our solution updates a representative dataset of collected features extracted from a convolutional neural network which is being kept fixed during execution. The dataset collected is used as a memory of all the past visual information observed so far as a rehearsal strategy. This strategy enables the accumulation and preservation of essential knowledge and at the same time allows to handle the non-stationarity of the data stream. We have shown that the proposed method is theoretically sound, asymptotically stable and operates online. Its effectiveness has been demonstrated in comparison with Multiple Object Tracking methods over public datasets. We showed that the method is capable of cumulative learning effectively over long unconstrained video sequences.

In Chapter 3 we proposed a system to perform the variant of unique counting, that is counting the unique persons which crosses a user drawn gate. The system is able to detect persons, track them when they are near a gate and crosses it. We perform open-world re-identification on the body feature, by exploiting fine-tuned features that we trained on Market and DukeMTMC-reid. Similarly to Chapter 2 the feature representation is kept fixed once the evaluation start and a memory module is exploited to handle person re-identification. Experiments on the challenging DukeMTMC dataset showed that our system is able to effectively count people passing through the gates in real time and recognize already passed people.

Finally we presented a novel technique for obtaining feature stationarity and compatibility while training a neural network. In Chapter 4 we have shown that a special set of fixed classifiers based on regular polytopes generates stationary features by maximally exploiting the available representation space. The proposed method is simple to implement and theoretically correct. Experimental results confirm both the theoretical analysis and the generalization capability of the approach across a range of datasets and architectures. RePoNet improves and generalizes the concept of a fixed classifier, recently proposed in [156], to a larger class of fixed classifier models exploiting the inherent symmetry of regular polytopes in the feature space. Finally we have shown that features extracted from a RePoNet classifier can achieve the desirable properties of maximal separation and maximal compactness using an additive angular margin which is constant and known in advance.

In Chapter 5 we introduced a novel approach for class-incremental learning that exploits future unseen classes as negative examples and learns features that do not change their geometric configuration as novel classes are incorporated in the learning model using our findings in Chapter 4. The approach uses a pre-allocated special fixed classifier (i.e. not undergoing the learning procedure) in which weights are set according to the vertices of the d-Simplex regular polytope. As shown in the experiments our method is as effective as the expanding classifier while exhibiting properties of internal feature representation that are otherwise not-existent.

In the future, there are several directions to further investigate on. The scenario we explored in Chapter 5, exploits the properties of RePoNet to pre-allocate a special classifier where the number of classes is set before learning starts. While the number of classes to allocate is arbitrary, we could argue that, eventually, they could all be allocated leaving no extra space for future classes.

Visual retrieval systems, rely directly on the embedding space discarding the classifier, as shown in multiple face recognition systems. These approaches, however, cannot update their feature representation model unless they are willing to reprocess all images stored in memory with the newer embedding model. Recently [215] proposed a method for backward-compatible representation learning, where features of a newer embedding are compatible with the ones of the older model, making it possible to apply visual retrieval between features of different models. Given the effectiveness of RePoNet in obtaining feature stationarity, we plan to use them in the same learning scheme. The feature learned with RePoNet should have a better backward-compatible representation with respect to classical neural networks making them ideal to address the problem of incremental learning and backward-compatible representation.

Appendix A

Appendix

A.1 Computing the Angle Between Adjacent Classifier Weights

The angle between a vertex and its adjacent vertices in a regular polytope can be computed following the same mathematical formulation used to compute its dihedral angle. The dihedral angle of a regular d-Simplex is the acute angle formed by a pair of intersecting faces. In the case $d = 2$ the dihedral angle is the angle at the vertex of an equilateral triangle, while in the case $d = 3$ is the angle formed by the faces of the regular tetrahedron.

Because the dual polytope of a regular d-Simplex is also a regular d-Simplex, the angle θ between pairs of vertices can be expressed as [250]:

$$\theta = \pi - \delta, \tag{A.1}$$

where δ is the dihedral angle. Since the dihedral angle of a regular d-Simplex is known to be [250] [191]:

$$\delta = \arccos\left(\frac{1}{d}\right), \tag{A.2}$$

substituting Eq. A.2 into Eq. A.1 we obtain:

$$\theta = \pi - \arccos\left(\frac{1}{d}\right)$$

which simplifies to:

$$\theta = \arccos\left(-\frac{1}{d}\right).$$

The Eq. above provides the value between any pair of vectors in a d-Simplex.

The calculation for the the d-Cube follows a similar argument. The dihedral angle of a regular d-Orthoplex is known to be $\arccos((2-d)/d)$. Since the d-Cube is the dual of the d-Orthoplex the angle defined by a vertex of a d-Cube and its adjacent vertices is:

$$\theta = \arccos\left(\frac{d-2}{d}\right). \tag{A.3}$$

Bibliography

[1] F. Pernici, M. Bruni, C. Baecchi, and A. Del Bimbo, "Fix your features: Stationary and maximally discriminative embeddings using regular polytope (fixed classifier) networks," *arXiv preprint arXiv:1902.10441*, 2019.

[2] J. Deng, J. Guo, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] S. Zhang, Y. Gong, J.-B. Huang, J. Lim, J. Wang, N. Ahuja, and M.-H. Yang, "Tracking persons-of-interest via adaptive discriminative features," in *European Conference on Computer Vision.* Springer, 2016, pp. 415–433.

[4] M. Du and R. Chellappa, "Face association for videos using conditional random fields and max-margin markov networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1762–1773, 2015.

[5] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation.* Elsevier, 1989, vol. 24, pp. 109–165.

[6] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[8] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[9] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[10] S. Grossberg, "How does a brain build a cognitive code?" in *Studies of Mind and Brain.* Springer, 1982, pp. 1–52.

[11] M. Mermillod, A. Bugaiska, and P. BONIN, "The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in Psychology*, vol. 4, p. 504, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2013.00504

[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, p. 201611835, 2017.

[13] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," *arXiv preprint arXiv:1710.10628*, 2017.

[14] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 3987–3995. [Online]. Available: http://proceedings.mlr.press/v70/zenke17a.html

[15] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[16] S. Farquhar and Y. Gal, "Towards robust evaluations of continual learning," *arXiv preprint arXiv:1805.09733*, 2018.

[17] Y. Hsu, Y. Liu, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *CoRR*, vol. abs/1810.12488, 2018. [Online]. Available: http://arxiv.org/abs/1810.12488

[18] Z. Li and D. Hoiem, "Learning without forgetting," in *European Conference on Computer Vision*. Springer, 2016, pp. 614–629.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS*, 2014.

[20] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4535–4544.

[21] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[22] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 177–186.

[23] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang, "Adanet: Adaptive structural learning of artificial neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 874–883.

[24] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[25] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[26] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 6467–6476. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html

[27] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[28] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efficient experience replay for streaming learning," *arXiv preprint arXiv:1809.05922*, 2018.

[29] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," *arXiv preprint arXiv:1802.10269*, 2018.

[30] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[31] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[32] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[35] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, 2015, pp. 41.1–41.12.

[36] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *International Conference on Automatic Face and Gesture Recognition*, 2018.

[37] Y. Shen, Y. Xiong, W. Xia, and S. Soatto, "Towards backward-compatible representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6367–6376.

[38] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[39] F. Pernici, F. Bartoli, M. Bruni, and A. D. Bimbo, "Memory based online learning of deep representations from video streams," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018 Salt Lake City, Utah, USA, June 18-22*, 2018.

[40] F. Pernici, M. Bruni, and A. Del Bimbo, "Self-supervised on-line cumulative learning from video streams," *Computer Vision and Image Understanding*, vol. 197-198, p. 102983, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314220300588

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in *NIPS*, vol. 1, 2012, p. 4.

[42] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.

[44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision.* Springer, 2014, pp. 740–755.

[45] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," *International Conference on Learning Representations (ICLR)*, 2019.

[46] K. R. Thórisson, J. Bieger, X. Li, and P. Wang, "Cumulative learning," in *International Conference on Artificial General Intelligence.* Springer, 2019, pp. 198–208.

[47] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 254–11 263.

[48] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.

[49] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. Roth, "Tracking the trackers: An analysis of the state of the art in multiple object tracking," *arXiv preprint arXiv:1704.02781*, 2017.

[50] W. Luo, X. Zhao, and T. Kim, "Multiple object tracking: A review," *CoRR*, vol. abs/1409.7618, 2017.

[51] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 10, pp. 1296–1311, 2003.

[52] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6638–6646.

[53] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision.* Springer, 2016, pp. 850–865.

[54] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1420–1429.

[55] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 749–765.

[56] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, “High performance visual tracking with siamese region proposal network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[57] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4293–4302.

[58] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-n learning: Bootstrapping binary classifiers by structural constraints,” in *CVPR*, june 2010.

[59] F. Pernici, “Facehugger: The alien tracker applied to faces,” in *Computer Vision–ECCV 2012. Workshops and Demonstrations*. Springer, 2012, pp. 597–601.

[60] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg *et al.*, “The seventh visual object tracking vot2019 challenge results,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[61] A. Lukežič, L. Č. Zajc, T. Vojíř, J. Matas, and M. Kristan, “Performance evaluation methodology for long-term visual object tracking,” *arXiv preprint arXiv:1906.08675*, 2019.

[62] J. H. Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon, “Structural constraint data association for online multi-object tracking,” *International Journal of Computer Vision*, vol. 127, no. 1, pp. 1–21, 2019.

[63] G. Lisanti, I. Masi, F. Pernici, and A. Del Bimbo, “Continuous localization and mapping of a pan–tilt–zoom camera for wide area tracking,” *Machine Vision and Applications*, vol. 27, no. 7, pp. 1071–1085, 2016.

[64] A. Del Bimbo, G. Lisanti, I. Masi, and F. Pernici, “Device-tagged feature-based localization and mapping of wide areas with a ptz camera,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 2010, pp. 39–44.

[65] B. Yang and R. Nevatia, “An online learned crf model for multi-target tracking,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2034–2041.

[66] E. Ristani and C. Tomasi, “Features for multi-target multi-camera tracking and re-identification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6036–6046.

[67] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[68] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 366–382.

[69] K. W. Lee, N. Sankaran, S. Setlur, N. Napp, and V. Govindaraju, "Wardrobe model for long term re-identification and appearance prediction," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).* IEEE, 2018, pp. 1–6.

[70] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[71] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, 2018.

[72] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016*, 2016. [Online]. Available: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-19.pdf

[73] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji, "Constrained clustering and its application to face clustering in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3507–3514.

[74] M. Tapaswi, O. M. Parkhi, E. Rahtu, E. Sommerlade, R. Stiefelhagen, and A. Zisserman, "Total cluster: A person agnostic clustering method for broadcast videos," in *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing.* ACM, 2014, p. 7.

[75] S. Xiao, M. Tan, and D. Xu, "Weighted block-sparse low rank representation for face clustering in videos," in *European Conference on Computer Vision.* Springer, 2014, pp. 123–138.

[76] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.

[77] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[78] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," *ICLR*, 2017.

[79] F. Pernici and A. Del Bimbo, “Unsupervised incremental learning of deep descriptors from video streams,” in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 477–482.

[80] F. Korn and S. Muthukrishnan, “Influence sets based on reverse nearest neighbor queries,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’00. New York, NY, USA: ACM, 2000, pp. 201–212.

[81] A. Bendale and T. Boult, “Towards open world recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1893–1902.

[82] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain, “Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1931–1939.

[83] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, “Deep learning in video multi-object tracking: A survey,” *Neurocomputing*, 2019.

[84] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” *CoRR*, vol. abs/1503.03832, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[85] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple hypothesis tracking revisited,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.

[86] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[87] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, “Poi: Multiple object tracking with high performance detection and appearance feature,” in *European Conference on Computer Vision*. Springer, 2016, pp. 36–42.

[88] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[89] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[90] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, “Deepercut: A deeper, stronger, and faster multi-person pose estimation

model," in *European Conference on Computer Vision.* Springer, 2016, pp. 34–50.

[91] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.

[92] G. Wang, J. Lai, P. Huang, and X. Xie, "Spatial-temporal person re-identification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8933–8940.

[93] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang, and Q. Tian, "Person re-identification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1367–1376.

[94] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *arXiv preprint arXiv:1701.01909*, 2017.

[95] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3645–3649.

[96] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4836–4845.

[97] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4705–4713.

[98] F. Pernici and A. Del Bimbo, "Object tracking by oversampling local features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2538–2551, 2013.

[99] C. O., C. R., D. B. A., M. J., P. F., and S. S., "Long term detection and tracking workshop." in *Conjunction With The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (LTDT2014)*, June 2014.

[100] A. Moudgil and V. Gandhi, "Long-term visual object tracking benchmark," *arXiv preprint arXiv:1712.01358*, 2017.

[101] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. Smeulders, P. H. Torr, and E. Gavves, "Long-term tracking in the wild: A benchmark," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685.

[102] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *CVPR*, june 2011.

[103] Y. Hua, K. Alahari, and C. Schmid, "Occlusion and motion reasoning for long-term tracking," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 172–187.

[104] Z. Hong, Z. Chen, C. Wang, X. Mei, D. V. Prokhorov, and D. Tao, "Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 749–758. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298675

[105] D. Kumaran, D. Hassabis, and J. L. McClelland, "What learning systems do intelligent agents need? complementary learning systems theory updated," *Trends in cognitive sciences*, vol. 20, no. 7, pp. 512–534, 2016.

[106] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[107] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Learning Representations*, Puerto Rico, 2016.

[108] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[109] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[110] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.

[111] M. Salganicoff, "Density-adaptive learning and forgetting," in *Proceedings of the Tenth International Conference on International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1993, pp. 276–283.

[112] P. Hu and D. Ramanan, "Finding tiny faces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[113] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.

[114] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint arXiv:1605.06065*, 2016.

[115] S. Banach, “Sur les opÃ©rations dans les ensembles abstraits et leur application aux Ã©quations intÃ©grales,” *Fundamenta Mathematicae*, vol. 3, no. 1, pp. 133–181, 1922.

[116] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[117] M. Bäuml, M. Tapaswi, and R. Stiefelhagen, “Semi-supervised Learning with Constraints for Person Identification in Multimedia Data,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 3602–3609.

[118] E. Maggio, E. Piccardo, C. Regazzoni, and A. Cavallaro, “Particle phd filtering for multi-target visual tracking,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, vol. 1. IEEE, 2007, pp. I–1101.

[119] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[120] C. Dicle, O. I. Camps, and M. Sznaier, “The way they move: Tracking multiple targets with similar appearance,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2304–2311.

[121] M. Ayazoglu, M. Sznaier, and O. I. Camps, “Fast algorithms for structured robust principal component analysis,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1704–1711.

[122] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, “Face detection without bells and whistles,” in *European Conference on Computer Vision.* Springer, 2014, pp. 720–735.

[123] J. H. Ward, “Hierarchical grouping to optimize an objective function,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

[124] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[125] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[126] P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, “Finding actors and actions in movies,” in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2280–2287.

[127] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR 2011*. IEEE, 2011, pp. 1201–1208.

[128] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?" in *International conference on database theory*. Springer, 1999, pp. 217–235.

[129] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

[130] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 17–35.

[131] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[132] S. Li and N. Amenta, "Brute-force k-nearest neighbors search on the gpu," in *International Conference on Similarity Search and Applications*. Springer, 2015, pp. 259–270.

[133] F. Turchini, M. Bruni, C. Baecchi, T. Uricchio, and A. D. Bimbo, "Open set recognition for unique person counting via virtual gates," in *Image Analysis and Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9-13, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, E. Ricci, S. R. Bulò, C. Snoek, O. Lanz, S. Messelodi, and N. Sebe, Eds., vol. 11751. Springer, 2019, pp. 94–105. [Online]. Available: https://doi.org/10.1007/978-3-030-30642-7_9

[134] V. A. Sindagi and V. M. Patel, "A survey of recent advances in cnn-based single image crowd counting and density estimation," *Pattern Recognition Letters*, 2018.

[135] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 833–841.

[136] F. Turchini, L. Seidenari, T. Uricchio, and A. Del Bimbo, "Deep learning based surveillance system for open critical areas," *Inventions*, vol. 3, no. 4, 2018.

[137] S. I. Cho and S.-J. Kang, "Real-time people counting system for customer movement analysis," *IEEE Access*, vol. 6, pp. 55 264–55 272, 2018.

[138] F. Pernici, F. Bartoli, M. Bruni, and A. Del Bimbo, "Memory based online learning of deep representations from video streams," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2324–2334.

[139] X. Liu, P. H. Tu, J. Rittscher, A. Perera, and N. Krahnstoever, "Detecting and counting people in surveillance applications," in *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* IEEE, 2005, pp. 306–311.

[140] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent survey on crowd density estimation and counting for visual surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

[141] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[142] M. Li, Z. Zhang, K. Huang, and T. Tan, "Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection," in *2008 19th International Conference on Pattern Recognition.* IEEE, 2008, pp. 1–4.

[143] H. Xu, P. Lv, and L. Meng, "A people counting system based on head-shoulder detection and tracking in surveillance video," in *2010 International Conference On Computer Design and Applications*, vol. 1. IEEE, 2010, pp. V1–394.

[144] E. Zhang and F. Chen, "A fast and robust people counting method in video surveillance," in *2007 International Conference on Computational Intelligence and Security (CIS 2007).* IEEE, 2007, pp. 339–343.

[145] X. Zhao, E. Delleandrea, and L. Chen, "A people counting system based on face detection and tracking in a video," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance.* IEEE, 2009, pp. 67–72.

[146] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[147] K. Wang, H. Wang, M. Liu, X. Xing, and T. Han, "Survey on person re-identification based on deep learning," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 4, pp. 219–227, 2018.

[148] Q. Leng, M. Ye, and Q. Tian, "A survey of open-world person re-identification," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[149] W.-S. Zheng, S. Gong, and T. Xiang, "Person re-identification by probabilistic relative distance comparison," in *CVPR 2011*. IEEE, 2011, pp. 649–656.

[150] Y. Xu, B. Ma, R. Huang, and L. Lin, "Person search in a scene by jointly modeling people commonness and person uniqueness," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 937–940.

[151] X. Zhu, B. Wu, D. Huang, and W.-S. Zheng, "Fast open-world person re-identification," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2286–2300, 2018.

[152] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: real-time multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[153] J. Boissonnat and F. Preparata, "Robust plane sweep for intersecting segments," *SIAM Journal on Computing*, vol. 29, no. 5, pp. 1401–1421, 2000.

[154] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[155] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[156] E. Hoffer, I. Hubara, and D. Soudry, "Fix your classifier: the marginal value of training the last weight layer," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=S1Dh8Tg0-

[157] F. Pernici, M. Bruni, C. Baecchi, and A. D. Bimbo, "Regular polytope networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[158] F. Pernici, M. Bruni, C. Baecchi, and A. Del Bimbo, "Maximally compact and separated features with regular polytope networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[159] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[160] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[161] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[162] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[163] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision.* Springer, 2014, pp. 818–833.

[164] M. Lin, Q. Chen, and S. Yan, "Network in network," *International Conference on Learning Representations (ICLR)*, 2014.

[165] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[166] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[167] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[168] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition.* Springer, 2015, pp. 84–92.

[169] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[170] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," in *International Conference on Learning Representations*, 2019.

[171] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT Press, 2016, vol. 1.

[172] M. Hardt and T. Ma, "Identity matters in deep learning," *International Conference on Learning Representations (ICLR)*, 2017.

[173] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou, "Spreading vectors for similarity search," *International Conference on Learning Representations (ICLR)*, 2019.

[174] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." in *ICML*, vol. 2, no. 3, 2016, p. 7.

[175] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *arXiv preprint arXiv:1703.09507*, 2017.

[176] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *CVPR*, 2017.

[177] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, and L. Song, "Decoupled networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[178] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "Normface: l 2 hypersphere embedding for face verification," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 1041–1049.

[179] Y. Liu, H. Li, and X. Wang, "Learning deep features via congenerous cosine loss for person recognition," *arXiv preprint: 1702.06890*, 2017.

[180] M. Hasnat, J. Bohné, J. Milgram, S. Gentric, L. Chen *et al.*, "von mises-fisher mixture model-based deep learning: Application to face verification," *arXiv preprint arXiv:1706.04264*, 2017.

[181] Y. Yuan, K. Yang, and C. Zhang, "Feature incay for representation regularization," *arXiv preprint arXiv:1705.10284*, 2017.

[182] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[183] B. Chen, W. Deng, and H. Shen, "Virtual class enhanced discriminative embedding learning," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1942–1952. [Online]. Available: http://papers.nips.cc/paper/7464-virtual-class-enhanced-discriminative-embedding-learning.pdf

[184] W. Liu, R. Lin, Z. Liu, L. Liu, Z. Yu, B. Dai, and L. Song, "Learning towards minimum hyperspherical energy," *NIPS*, 2018.

[185] K. Zhao, J. Xu, and M.-M. Cheng, "Regularface: Deep face recognition via exclusive regularization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[186] Y. Duan, J. Lu, and J. Zhou, "Uniformface: Learning deep equidistributed representation for face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[187] J. J. Thomson, "XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 7, no. 39, pp. 237–265, 1904.

[188] J. Batle, A. Bagdasaryan, M. Abdel-Aty, and S. Abdalla, "Generalized thomson problem in arbitrary dimensions and non-euclidean geometries," *Physica A: Statistical Mechanics and its Applications*, vol. 451, pp. 237–250, 2016.

[189] P. M. L. Tammes, "On the origin of number and arrangement of the places of exit on the surface of pollen-grains," *Recueil des travaux botaniques néerlandais*, vol. 27, no. 1, pp. 1–84, 1930.

[190] B. Bagchi, "How to stay away from each other in a spherical universe," *Resonance*, vol. 2, no. 9, pp. 18–26, 1997.

[191] H. Coxeter, *Regular Polytopes*, ser. Macmillan mathematics paperbacks. Macmillan, 1963. [Online]. Available: https://books.google.it/books?id=_22QswEACAAJ

[192] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[193] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, 2017, pp. 2921–2926. [Online]. Available: https://doi.org/10.1109/IJCNN.2017.7966217

[194] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[195] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[196] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[197] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[198] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[199] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[200] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[201] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ICLR*, 2015.

[202] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2261–2269.

[203] X. Li, W. Wang, X. Hu, and J. Yang, “Selective kernel networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 510–519.

[204] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[205] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Randaugment: Practical automated data augmentation with a reduced search space,” in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[206] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, “Predicting the computational cost of deep learning models,” in *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, N. Abe, H. Liu, C. Pu, X. Hu, N. K. Ahmed, M. Qiao, Y. Song, D. Kossmann, B. Liu, K. Lee, J. Tang, J. He, and J. S. Saltz, Eds. IEEE, 2018, pp. 3873–3882. [Online]. Available: https://doi.org/10.1109/BigData.2018.8622396

[207] M. M. Kalayeh, E. Basaran, M. Gökmen, M. E. Kamasak, and M. Shah, “Human semantic parsing for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1062–1071.

[208] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, “Cosface: Large margin cosine loss for deep face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.

[209] M.-M. C. Kai Zhao, Jingyi Xu, "Regularface: Deep face recognition via exclusive regularization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[210] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[211] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision, ICCV*, 2015, pp. 1026–1034. [Online]. Available: https://doi.org/10.1109/ICCV.2015.123

[212] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014. [Online]. Available: http://arxiv.org/abs/1410.5401

[213] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1410.3916

[214] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJTQLdqlg

[215] Y. Shen, Y. Xiong, W. Xia, and S. Soatto, "Towards backward-compatible representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[216] F. Pernici, F. Bartoli, M. Bruni, and A. Del Bimbo, "Memory based online learning of deep representations from video streams," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[217] S. Beery, G. Wu, V. Rathod, R. Votel, and J. Huang, "Context r-cnn: Long term temporal context for per-camera object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[218] H. Deng, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. M. Robertson, and H. Guan, "Object guided external memory network for video object detection," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27*

- *November 2, 2019*. IEEE, 2019, pp. 6677–6686. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00678

[219] M. Shvets, W. Liu, and A. C. Berg, "Leveraging long-range temporal relationships between proposals for video object detection," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 9755–9763. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00985

[220] C. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. B. Girshick, "Long-term feature banks for detailed video understanding," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 284–293. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Wu_Long-Term_Feature_Banks_for_Detailed_Video_Understanding_CVPR_2019_paper.html

[221] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 9216–9224. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00931

[222] S. W. Oh, J. Lee, N. Xu, and S. J. Kim, "Video object segmentation using space-time memory networks," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 9225–9234. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00932

[223] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IX*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11213. Springer, 2018, pp. 153–169. [Online]. Available: https://doi.org/10.1007/978-3-030-01240-3_10

[224] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, G. Hua and H. Jégou, Eds., vol. 9914, 2016, pp. 850–865. [Online]. Available: https://doi.org/10.1007/978-3-319-48881-3_56

[225] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 849–11 860.

[226] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, “Continual unsupervised representation learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 7647–7657.

[227] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, “Learning a unified classifier incrementally via rebalancing,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[228] Y. Liu, Y. Su, A. Liu, B. Schiele, and Q. Sun, “Mnemonics training: Multi-class incremental learning without forgetting,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 245–12 254.

[229] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, “Supermasks in superposition,” *arXiv*, pp. arXiv–2006, 2020.

[230] Q. Liu, O. Majumder, A. Ravichandran, R. Bhotika, and S. Soatto, “Incremental learning for metric-based meta-learners,” *ECCV*, 2020.

[231] R. Kurle, B. Cseke, A. Klushyn, P. van der Smagt, and S. Günnemann, “Continual learning with bayesian neural networks for non-stationary data,” in *International Conference on Learning Representations*, 2019.

[232] J. von Oswald, C. Henning, J. Sacramento, and B. F. Grewe, “Continual learning with hypernetworks,” in *International Conference on Learning Representations*, 2020.

[233] D. Maltoni and V. Lomonaco, “Continuous learning in single-incremental-task scenarios,” *Neural Networks*, vol. 116, pp. 56–73, 2019.

[234] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *arXiv preprint arXiv:1904.07734*, 2019.

[235] M. K. Titsias, J. Schwarz, A. G. d. G. Matthews, R. Pascanu, and Y. W. Teh, “Functional regularisation for continual learning using gaussian processes,” *arXiv preprint arXiv:1901.11356*, 2019.

[236] J. Lee, J. Yun, S. Hwang, and E. Yang, “Lifelong learning with dynamically expandable networks,” *arXiv preprint arXiv:1708.01547*, 2017.

[237] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” *ICML*, vol. 80, pp. 4548–4557, 10–15 Jul 2018.

[238] A. Mallya, D. Davis, and S. Lazebnik, “Piggyback: Adapting a single network to multiple tasks by learning to mask weights,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82.

[239] A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[240] H. Jung, J. Ju, M. Jung, and J. Kim, “Less-forgetting learning in deep neural networks,” *arXiv preprint arXiv:1607.00122*, 2016.

[241] H. Jung, J. Ju, and M. Jung, “Less-forgetful learning for domain expansion in deep neural networks,” *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[242] F. Pernici, M. Bruni, and A. Del Bimbo, “Self-supervised on-line cumulative learning from video streams,” *Computer Vision and Image Understanding*, p. 102983, 2020.

[243] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. Laradji, I. Rish, A. Lacoste, D. Vazquez *et al.*, “Online fast adaptation and knowledge accumulation: a new approach to continual learning,” *arXiv preprint arXiv:2003.05856*, 2020.

[244] F. Pernici, M. Bruni, C. Baecchi, and A. Del Bimbo, “Maximally compact and separated features with regular polytope networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 46–53.

[245] Z. Qian, T. L. Hayes, K. Kafle, and C. Kanan, “Do we need fully connected output layers in convolutional networks?” *arXiv preprint arXiv:2004.13587*, 2020.

[246] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” *PNAS*, 2016.

[247] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11207. Springer, 2018, pp. 144–161.

[248] Y. LeCun, “The mnist database of handwritten digits,” *http://yann.lecun.com/exdb/mnist/*, 1998.

[249] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” *https://www.cs.toronto.edu/ kriz/cifar.html*, 2009.

[250] H. R. Parks and D. C. Wills, “An elementary calculation of the dihedral angle of the regular n-simplex,” *The American mathematical monthly*, vol. 109, no. 8, pp. 756–758, 2002.