UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
**IECS Doctoral School**

# DEEP LEARNING APPROACHES FOR TIME-EVOLVING SCENARIOS

Alessia Bertugli

Advisor:
Prof. Andrea Passerini
Università degli Studi di Trento

Co-Advisor:
Prof. Simone Calderara
Università degli Studi di Modena e Reggio Emilia

April 2023

Review committee composed of:

**Prof. Simone Palazzo**
Assistant Professor
Università degli Studi di Catania
Catania

**Prof. Vincenzo Lomonaco**
Assistant Professor
Università di Pisa
Pisa

A mio nonno Dante, per aver reso Casa del Vento così speciale.

# Abstract

One of the most challenging topics of deep learning (DL) is the analysis of temporal series in complex real-world scenarios. The majority of proposed DL methods tend to simplify such environments without considering several factors. The first part of this thesis focuses on developing video surveillance and sports analytic systems, in which obstacles, social interactions, and flow directions are relevant aspects. A DL model is then proposed to predict future paths, taking into account human interactions sharing a common memory, and favouring the most common paths through belief maps. Another model is proposed, adding the possibility to consider agents' goals. This aspect is particularly relevant in sports games where players can share objectives and tactics. Both the proposed models rely on the common hypothesis that the whole amount of labelled data is available from the beginning of the analysis, without evolving. This can be a strong simplification for most real-world scenarios, where data is available as a stream and changes over time. Thus, a theoretical model for continual learning is then developed to face problems where few data come as a stream, and labelling them is a hard task. Finally, continual learning strategies are applied to one of the most challenging scenarios for DL: financial market predictions. A collection of state-of-the-art continual learning techniques are applied to financial indicators representing temporal data. Results achieved during this PhD show how artificial intelligence algorithms can help to solve real-world problems in complex and time-evolving scenarios.

**Keywords**
[Trajectory forecasting, continual learning, meta-learning, finance]

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 The context

Deep learning is commonly asked to solve difficult problems in real-world scenarios where many factors are involved. Among them, video surveillance and time series forecasting are particularly challenging due to the interaction between agents, the multiplicity of future possible paths, and the presence of static and moving obstacles [2, 94, 117, 56, 121]. State-of-the-art DL methods usually face these problems by reducing the amount of influencing factors, leading to poor performance in the most evolution-prone contexts. Another aspect that influences the performance is how data are presented to the neural network. Indeed, in most real-world applications, data are not available in fixed batches but come as a stream [85, 89, 12, 73]. This way, the models should be able to rapidly adapt to changeling data in order to achieve good performance. One of these targeted problems is financial market prediction, where price time series generate trend regimes evolving and repeating over time [74].
In this thesis, we focus on all these aspects. Firstly, we face trajectory forecasting in video surveillance and sports analytics, proposing two methods able to make predictions with interacting agents and obstacles. Then, continual learning is studied to face a data stream. A novel method is proposed to be applied in complex scenarios where traditional methods fail due to a lack of independent and identically distributed labelled data. Then, continual learning techniques are applied to financial time series predictions, showing successful performance on repeating regimes.

The research on trajectory forecasting rises within PRIN PREVUE - PRediction of activities and Events by "Vision in an Urban Environment" project (CUP E94I19000650001), PRIN National Research Program, MUR.
The research on meta-continual learning is born from a collaboration between the University of Modena and Reggio Emilia and the University of Trento.
Then, the research on continual learning in finance is realised in collaboration with the University of Modena and Reggio Emilia and the company Axyon AI. It is supported by FF4EuroHPC: HPC Innovation for European SMEs, Project Call 1, funded by the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951745.

These three topics find their link to the complexity of the time-evolving scenarios. All these topics are faced with applying novel deep learning techniques to real-world complex environments with the aim of overcoming the limitation of state-of-the-art approaches.

## 1.2 Summary of contributions

This thesis will present several contributions in three research areas: trajectory forecasting, continual learning in complex scenarios, and continual learning applied to finance. We report below a summary of each topic and the proposed solutions.

### 1.2.1 Trajectory forecasting

Anticipating human motion in crowded scenarios is essential for developing intelligent transportation systems, social-aware robots, advanced video surveillance applications and sport tactics analysis. A key component of this task is represented by the inherently multi-modal nature of human paths which makes socially acceptable multiple futures when human interactions are involved. To this end, we propose two solutions. A generative architecture for multi-future trajectory predictions based on Conditional Variational Recurrent Neural Networks (C-VRNNs). Conditioning mainly relies on prior belief maps, representing most likely moving directions and forcing the model to consider past observed dynam-

ics in generating future positions. Human interactions are modeled with a graph-based attention mechanism enabling an online attentive hidden state refinement of the recurrent estimation. This method is particularly useful in crowd scenarios with obstacles, that can be avoided conditioning on belief maps. On the other end, we consider that people activities, especially in sports, are often driven by goals, e.g. reaching particular locations or interacting with the environment. We address this aspect by adding to the aforementioned recurrent generative model the possibility to take into account both single agents' future goals and interactions between different agents. The model exploits a double attention-based graph neural network to collect information about the mutual influences among different agents and integrates it with data about agents' possible future objectives. To corroborate our models, we perform extensive experiments on publicly-available datasets (e.g., ETH/UCY, Stanford Drone Dataset, STATS SportVU NBA, Intersection Drone Dataset and TrajNet++) and demonstrate their effectiveness on both crowded scenes and interactive sports compared to several state-of-the-art methods.

### 1.2.2 Continual learning in complex scenarios

Future deep learning systems call for techniques that can deal with the evolving nature of temporal data and scarcity of annotations when new problems occur. This is indeed the case of video surveillance, autonomous robots, industrial AI applications and all the deployed solutions that should, as humans do, aim for life-long learning capabilities. As a step towards this goal, we present FUSION (Few-shot UnSupervIsed cONtinual learning), a learning strategy that enables a neural network to learn quickly and continually on streams of unlabelled data and unbalanced tasks. The objective is to maximise the knowledge extracted from the unlabelled data stream (unsupervised), favor the forward transfer of previously learnt tasks and features (continual) and exploit as much as possible the supervised information when available (few-shot). The core of FUSION is MEML - Meta-Example Meta-Learning – that consolidates a meta-representation through the use of a self-attention mechanism during a single inner loop in the meta-optimisation stage. To further enhance the capability of MEML to generalise from few data, we extend it by creating various augmented surrogate tasks and by optimising over the hardest. An extensive experimental evaluation on public computer vision benchmarks shows that FUSION outperforms existing state-of-the-art solutions both in the few-shot and continual learning experimental settings. We also empirically demonstrate that FUSION maximises the positive feature

transfer and reuse across different datasets.

### 1.2.3 Continual learning in finance

Nowadays, financial markets produce a large amount of data, in the form of historical time series, which quantitative researchers have recently attempted at predicting with deep learning models. These models are constantly updated with new incoming data in an online fashion. However, artificial neural networks tend to exhibit poor adaptability, fitting the last seen trends, without keeping the information from the previous ones. Continual learning studies this problem, called catastrophic forgetting, to preserve the knowledge acquired in the past and exploiting it for learning new trends. This thesis evaluates and highlights continual learning techniques applied to financial historical time series in a context of binary classification (upward or downward trend). The main state-of-the-art algorithms have been evaluated with data derived from a practical scenario, highlighting how the application of continual learning techniques allows for better performance in the financial field against conventional online approaches.

## 1.3 Structure of the thesis

This thesis is organised as follows. In Chapter 2, a review of the state of the art of the proposed topics is presented. In particular, in Section 2.1 a survey of trajectory forecasting methods is done; in Section 2.2 complex learning environments involving continual learning and meta-learning are described; in Section 2.3 machine learning and deep learning techniques for financial time series prediction are introduced.
Chapter 3 depicts the proposed method for trajectory forecasting. In particular, in Section 3.1, Section 3.2 and Section 3.3 the problem formulation, the variational recurrent neural network architecture and the hidden state refinement method, used in both the proposed models are presented. In Section 3.4 and Section 3.5 AC-VRNN, conditioning the VRNN on belief map, and DAG-Net, conditioning agents' goals are described in details. Finally, Section 3.6 shows extensive experimental analysis of the two methods.
Chapter 4 proposes the description of our learning strategy in a meta-continual learning setting, FUSION and our novel algorithms, MEML and MEMLX in Section 4.1, and the experimental results comparing to other state-of-the-art approaches in Section 4.2.

Chapter 5 investigates continual learning techniques in finance. In detail, Section 5.1 introduces financial time series and indicators, domain regimes, and the most famous state-of-the-art continual learning methods; while Section 5.2 shows results of the latter on market price datasets.
Finally, Chapter 6 presents a discussion on the three topics faced in this thesis, along with a brief introduction to future intentions. Trajectory forecasting, continual learning, and finance conclusions are reported respectively in Section 6.1, Section 6.2, and Section 6.3.

# Chapter 2

# Literature Survey

In this section, a review of the state of the art of the research topics investigated in this thesis is reported. In particular, Sec. 2.1 depicts the principal methods introduced in literature for trajectory forecasting. Sec. 2.2 illustrates continual learning and meta-learning techniques implied in complex scenarios, in which neural networks struggle to learn. Finally, Sec. 2.3 describes financial market prediction, and the continual learning approaches that can be used to face catastrophic forgetting.

## 2.1 Trajectory forecasting

Traditionally, trajectory prediction has been approached with rule-based and social force models ([40, 76, 118]) that have been proven to be effective in simple contexts, but fail to generalize to complex domains. In recent years, generative models ([37, 120, 101, 56, 45]) have been focusing on the multi-modal nature of this task since multiple human paths could be regarded as socially acceptable despite being different from ground-truth annotations. In the following, we group related work into position-based models, which use only spatial information, and graph-based models, which rely on connected structures.

### 2.1.1 Position-based models

Social-LSTM ([2]) models individual trajectory as a long short-term memory (LSTM) encoder-decoder and considers interactions using a social pooling mechanism. Social GAN ([37]) uses a pooling mechanism in combination with a generative model to predict socially acceptable trajectories. SoPhie ([94]) consists of a Generative Adversarial Network (GAN), which leverages the contribution of a social attention module and a physical attention module. SS-LSTM ([113]) uses different inputs to also take into account the influence of the environment and maps of the neighbourhood to narrow the field of mutual influences.

### 2.1.2 Graph-based models

Graph Neural Networks (GNNs) have been used to model interactions between different trajectories. Graph Variational RNNs ([116]) model multi-agent trajectory data mainly focusing on multi-player sports games. Each agent is represented by a VRNN where the prior, the encoder and the decoder are modelled as message-passing GNNs, allowing the agents to weakly share information through nodes. Graph-structured VRNN network ([101]), based on relation networks, infers the current state and forecasts future states of basket and football players' trajectories. SR-LSTM ([121]) uses a state refinement module through a motion gate and pedestrian-wise attention. Social-BiGAT ([56]) presents a graph-based generative adversarial network based on GAT ([108]) that learns reliable future representations that encode the social interactions between humans in the scene and contextual images to incorporate scene information. ST-GAT ([117]) proposes a model based on two levels of LSTMs to incorporate interactions through a hidden state refinement. It uses GAT into the encoding part, while a decoder generates future positions.
Compared to approaches based on Variational Autoencoders (VAEs) ([116, 101, 54]), our methods do not model the prediction as a graph yet use an attentive module to refine the hidden state of a recurrent network. In doing so, information about other agents influences the prediction. Unlike sports games, where all players share the same goal, in urban scenarios neighbourhood information is crucial since future paths may depend on mutual distances among people. Our models resemble SR-LSTM ([121]) and STGAT ([117]) combining LSTMs and GNNs. Nevertheless, SR-LSTM ([121]) exploits cell states of LSTMs limiting the observation horizon. STGAT ([117]) uses GAT ([108]) as hidden state refinement, but it employs a sequence-to-sequence model without an online refinement. Both

methods do not take into account contextual information or collective behaviours (e.g., belief maps) in order to avoid uncommon paths.

## 2.2 Complex learning environments

### 2.2.1 Continual learning

Continual learning is one of the most challenging problems in deep learning research since neural networks are heavily affected by catastrophic forgetting when data are available as a stream of tasks. In more detail, neural networks tend to focus on the most recent tasks, overwriting their past knowledge and consequently causing forgetting. Further, a common continual learning scenario is task-incremental classification, where each task $T_i$ denotes a particular classification dataset. The model's evaluation occurs by averaging the metrics across all tasks (the same tasks, with different samples, already seen at train time). As theoretically exposed in [105], there are three main evaluation protocols for comparing methods' performance: Task-IL, Domain-IL and Class-IL. Task-IL is the easiest scenario since task identity is always provided, even at test-time; Domain-IL only needs to solve the current task, no task identity is necessary; Class-IL instead intends to solve all tasks seen so far with no task identity given. Much of the recent literature [47, 6, 12] is directed towards methods that do not require the detection of task change. Our proposed approach follows this line of research, using a rehearsal technique to avoid forgetting without the need for task identity and targeted solutions to find them. Finally, continual learning methods can be divided into three main categories.

**Architectural strategies**. They are based on specific architectures designed to mitigate catastrophic forgetting [91, 95]. Progressive Neural Networks (PNN) [91] are based on parameter-freezing and network expansion, but suffer from a capacity problem because it implies adding a column to the neural network at each new task, so growing up the number of tasks training the neural network becomes more difficult due to exploding/vanishing gradient problems. Progress & Compress [95] tries to solve this problem by augmenting and reducing the neural network size.

**Regularisation strategies**. They rely on putting regularisation terms into the loss function, promoting selective consolidation of important past weights [55, 119]. Elastic Weights Consolidation (EWC) [55] uses a regularization term to control catastrophic forgetting by selectively constraining the model weights that are im-

portant for previous tasks through the computation of the Fisher information matrix of weights importance. Synaptic Intelligence (SI) [119] can be considered as a variant of EWC, that computes weights importance online during SGD.

**Rehearsal strategies**. Rehearsal strategies focus on retaining part of past information and periodically replaying it to the model to strengthen connections for memories, involving meta-learning [86, 100], a combination of rehearsal and regularisation strategies [73, 19], knowledge distillation [34, 65, 42, 62], generative replay [96, 97, 70] and channel gating [1]. Experience Replay [89] stores a limited amount of information from the past and then adds a further term to the loss that takes into account loss minimization on the buffer data, besides the current data. Meta-Experience Replay (MER) [86] is based on Reptile [78], which induces a meta-learning update in the process and integrates an experience replay buffer, updated with reservoir sampling, facilitating continual learning while maximizing transfer and minimizing interference. Gradient Episodic Memory (GEM) [73] and its more efficient version A-GEM [19] is a mix of regularization and rehearsal strategies. They use a fixed dimensional memory to store the last examples for each task and then add a constraint to the loss that leads to updates that do not increase the loss of the previous tasks.

**Other Approaches**. Only a few recent works have studied how to solve a continual learning task with unlabelled data, which mainly involves representation learning. Previously iCaRL [85] introduces a method involving a representation learning network and an incremental classifier in a supervised setting, resembling the idea proposed by unsupervised methods. CURL [83] propose an unsupervised model built on a representation learning network. This latter learns a mixture of Gaussian encoding task variations, and then integrates a generative memory replay buffer as a strategy to overcome forgetting.

### 2.2.2 Meta-learning

Meta-learning, or learning to learn, aims to improve the neural network's ability to rapidly learn new tasks with few training samples. The tasks can comprise a variety of problems, such as classification, regression and reinforcement learning, but differently from Continual learning, training doesn't occur with incremental tasks and models are evaluated on new unseen tasks. The majority of meta-learning approaches proposed in the literature are based on Model-Agnostic Meta-Learning (MAML) [31, 82, 78, 92].

**MAML**. By learning an effective parameter initialisation, with a double loop procedure, MAML limits the number of stochastic gradient descent steps required to

learn new tasks, speeding up the adaptation process performed at meta-test time. The double loop procedure acts as follow: an inner loop that updates the parameters of the neural network to learn task-specific features and an outer loop generalises to all tasks. In this way, the neural network learns a parameter initialization that needs a small number of SGD steps to learn new tasks. The success of MAML is due to its model-agnostic nature and the limited number of parameters it requires. Nevertheless, it suffers from some limitations related to the amount of computational power it needs during the training phase. To solve this issue, the authors propose a further version, First Order MAML (FOMAML) that focuses on removing the second derivative causing the need for large computational resources. Other works have attempted to solve this issue, such as [82], a MAML algorithm with an implicit gradient that requires only the result of the inner loop optimization and not all the path, considerably reducing the computation cost. ANIL [80] investigates the success of MAML finding that it mostly depends on feature reuse rather than rapid learning. This way, the authors propose a slim version of MAML, removing almost all inner loops except for task-specific heads.

**Unsupervised meta-learning**. Although MAML is suitable for many learning settings, few works investigate the unsupervised meta-learning problem. CACTUs [44] proposes a new unsupervised meta-learning method relying on clustering feature embeddings through the k-means algorithm and then builds tasks upon the predicted classes. The authors employ representation learning strategies to learn compliant embeddings during a pre-training phase. From these learned embeddings, a k-means algorithm clusters the features and assigns pseudo-labels to all samples. Finally, the tasks are built on these pseudo-labels. The authors proposed two versions of the model, one based on MAML and the other on Protonets [98] to classify data. CACTUs exhibits promising results on standard meta-learning benchmarks, reaching performances that are not so far from the supervised version Oracle-MAML. UMTRA [51] is a further method of unsupervised meta-learning based on a random sampling and data augmentation strategy to build meta-learning tasks, achieving comparable results with respect to CACTUs. During meta-training $N$ unlabelled data points are randomly sampled from the training set, and then a single pseudo-label is randomly assigned to each data point, approximating that all data points belong to different classes. In this way, the meta-training is performed in a one-shot learning setting. UFLST [49] proposes an unsupervised few-shot learning method based on self-supervised training, alternating between progressive clustering and update of the representations. It shows promising results not only on standard benchmarks but also on more complex datasets recalling real-world applications.

### 2.2.3 Meta-learning for continual learning

Meta-learning has been extensively merged with continual learning for different purposes. We highlight the existence of two strands of literature [13]: *meta-continual learning*, which aims to incremental task learning, and *continual-meta learning* that instead focuses on fast remembering. To clarify the difference between these two branches, we adopt the standard notation that denotes $S$ as the support set and $Q$ as the query set. The sets are generated from the data distribution of the context (task) $C$ and respectively contain the samples employed in the inner and outer loops (e.g. in a classification scenario, both $S$ and $Q$ contain different samples of the same classes included in the current task). We define the meta-learning algorithm as $ML_\phi$ and the continual learning one with $CL_\phi$.

**Continual-meta learning.** Continual-meta learning mainly focuses on making meta-learning algorithms online, to rapidly remember meta-test tasks [32, 48, 38]. In detail, it considers a sequence of tasks $S_{1:T}$, $Q_{1:T}$, where the inner loop computation is performed through $f_{\theta t} = ML_\phi(S_{t-1})$, while the learning of $\phi$ (outer loop) is obtained using gradient descent over the $l_t = \mathcal{L}(f_{\theta t}, S_t)$. Since local stationarity is assumed, the model fails on its first prediction when the task switches. At the end of the sequence, $ML_\phi$ recomputes the inner loops over the previous supports and evaluates on the query set $Q_{1:T}$. Some recent works propose different strategies to deal with continual-meta learning. Online meta-learning [32] is an online version of MAML, with the limitation of not considering the catastrophic forgetting problem. In [48], the authors propose a Dirichlet process mixture of hierarchical Bayesian models that is able to deal with a potentially infinite mixture in a continual learning fashion. MOCA [38] extends meta-learning to operate with a stream of tasks, finding the change of task through online change-point analysis.

**Meta-continual learning.** More relevant to our work are meta-continual learning algorithms [110, 47, 6, 115, 69, 81, 102], which use meta-learning rules to "learn how not to forget". Resembling the notation proposed in [13], given $K$ sequences sampled i.i.d. from a distribution of contexts $C$, $S_{i,1:T}, Q_{i,1:T} \sim X_{i,1:T}|C_{i,1:T}$, $CL_\phi$ is learned with $\nabla_\phi \sum_t \mathcal{L}(CL_\phi(S_t), Q_t)$ with $i < N < K$ and evaluated on the left out sets $\sum_{i=N}^{K} \mathcal{L}(CL_\phi(S_t), Q_t)$. In particular, OML [47] and its variant ANML [6] favour sparse representations by employing a trajectory-input update in the inner loop and a random-input update in the outer one. The algorithm jointly trains a representation learning network (RLN) and a prediction learning network (PLN) during the meta-training phase. Then, at meta-test time, the RLN layers are frozen and only the PLN is updated. ANML replaces the RLN network

with a neuro-modulatory network that acts as a gating mechanism on the PLN activations following the idea of conditional computation. HSML [115] is a hierarchically structured approach to meta-continual learning involving a hierarchical task clustering strategy to resemble the human brain's way to associate knowledge. Other works, such as [69, 81, 102], focus on incrementally learning new tasks in a few-shot setting, respectively through a metric-based, a task-agnostic learner and a neural gas network.
We follow the setting introduced by these OML [47] and ANML [6], adapting it to an unsupervised stream of data. We further propose a different architecture - employing an attentive module - which not only tackles the memory limitation of MAML-based approaches but also robustly improves the generalization capability of the model.

## 2.3 Financial predictions

Time series research has experienced strong growth in several sectors in the last few years: from the prediction of pedestrian and car trajectories for video surveillance [77, 9] to the prediction of machinery failures in industries. Among them, financial market predictions have been deeply investigated leading to the development of increasingly sophisticated algorithms capable of predicting the trend of the financial market. Machine learning and deep learning techniques have been applied to financial time series, to make these algorithms as automatic as possible to facilitate the traders' decisions. However, due to the unpredictability of the market is still hard to design machine learning algorithms that can properly work on financial time series [25]. In fact, deep learning models, like 1-D convolutional neural networks, multi-layer perceptrons, temporal transformers [107]), that achieve the state-of the-art performance on other tasks, do not always exhibit satisfactory performance on financial problems. For this reason, deep learning for finance is at the cutting edge of research and in the last few years, several methods have been designed or adapted specifically to financial time series [71, 27, 74].

# Chapter 3

# Trajectory Forecasting in Real-World Environment

Trajectory forecasting has recently experienced exponential growth in several research areas such as video surveillance, sports analytics, self-driving cars and physical systems ([90]). Its main applications include pedestrians dynamics prediction ([40, 2, 113, 37, 117, 121]), vehicles behaviour analysis ([50, 75, 15, 63]) as well as intent estimation of people and cars on roads to avoid possible crashes. In sports analytics ([29, 116, 120, 101, 21, 43]), being able to predict players' trajectories can improve the action interpretation of each player while in physical systems it can be fundamental to predict particles dynamics in complex domains ([54, 4, 111]).

We focus on predicting human dynamics in crowded contexts (e.g., shop entrances, university campuses and intersections) where people and autonomous vehicles mainly manifest their complex and multi-modal nature. Typically, two different strategies are employed to model human interactions: *pooling-based* and *graph-based* methods. Pooling-based methods ([2, 113, 37, 46, 67, 68]) employ sequence-to-sequence models to extract features and generate subsequent time steps, interspersed with pooling layers to model interactions between neighbours. By contrast, graph-based methods ([117, 121, 75, 116, 101, 109, 66]) apply graph neural networks to model interactions. Although these approaches have proven to be effective, some problems are still open, such as efficiently exploiting context cues and appropriately capturing human interactions in critical

situations. Another relevant aspect to consider in trajectory prediction is represented by scene constraints like walls and other obstacles which strongly influence human motion. A common approach to overcome this issue is to introduce visual elements into the network such as images or semantic segmentation ([67, 56, 94]) yet this implies the availability of video streams both at train and test time. To this end, we propose two novel methods for multi-future trajectory forecasting that works in a completely generative setting, enabling the prediction of multiple possible futures. During online inference, we integrate human interactions at time step level, allowing other agents to affect the whole trajectory generation process. As a consequence, online interactions computation improves the predicted trajectories as the number of time steps increases limiting the error growth.

**AC-VRNN.** To take into account past human motion, local belief maps steer future positions towards more frequent crossed areas when human interactions are limited or absent. Technically, our model is a Conditional-VRNN, conditioned by prior belief maps on pedestrians' frequent paths, that predicts future positions one time step at a time, by relying on recurrent network hidden states refined with an attention-based mechanism. *

The main contributions of this work are two-fold:

(i) We propose a novel method to integrate human interactions into the model in an online fashion, relying on a hidden state refinement process with a graph attentive mechanism. We employ a similarity-based adjacency matrix to take into account pedestrians' neighbourhoods.

(ii) We introduce local belief maps to encourage the model to follow a prior transition distribution whenever the prediction is uncertain and to discourage unnatural behaviour such as crossing obstacles, avoiding employing additional visual inputs. In this way, future positions may take advantage of prior knowledge while being predicted. Such behaviour is imposed during training by a Kullback–Leibler (KL) divergence loss between ground-truths and samples contributing to the model performance refinement.

**DAG-Net.** Interactions heavily impact future trajectories [37, 2]: since people plan their paths by reading each other's future possible behaviours, each person's motion is influenced by the subjects around them. In team sports, interactions take on an even more important role: an attacker could put in place a certain set of movements just because the rest of his team has a specific disposition, as well

*Code is available at https://github.com/alessiabertugli/AC-VRNN.

as the whole defending team could in turn react and put in place a predefined tactic only because some opponents are arranged in a certain way. The varied nature of interactions and their heavy impact on agents' behaviour leads to develop sophisticated methods to accurately interpret and integrate such information into the prediction method. Even the *future knowledge* about interacting agents' positions can be a relevant feature that affects the development of every single path. Taking into account this aspect during trajectory prediction can improve the accuracy of the model. To address these challenges we propose DAG-Net, a double attentive graph neural network for trajectory forecasting. The network backbone is a recurrent version of the Variational Autoencoder (VAE) [53]: time-step after time-step, the autoencoder is used to generate future positions in terms of displacements from the current locations. The modules of our recurrent autoencoder are conditioned on subjects' current objectives so that the model can accordingly produce likely future positions. The backbone is integrated with a double Graph Neural Network (GNN)-based mechanism: the first GNN defines the future objectives of each agent in a structured way, distilling each goal with proximity knowledge; the second one models agents' interactions, filtering the hidden states of the recurrent network through neighbourhood information. Both the GNNs use a self-attention mechanism to assign different weights to each edge of the graph. [†]

We demonstrate that our models achieve state-of-the-art performance on several standard benchmarks using different evaluation protocols. We also outperform our competitors on the challenging Stanford Drone Dataset (SDD) and the recent Intersection Drone Dataset (InD) showing the robustness of our architecture to more complex urban contexts. Furthermore, we test our models on human dynamics collected from basketball players to analyze their ability to capture complex interactions in confined areas on STATS SportVU NBA dataset. Finally, our architecture positions among the best models on the TrajNet++ benchmark.

## 3.1 Problem formulation

Given a pedestrian at time step $t$, his/her current position is represented by 2-D coordinates. Our models analyze $T_{obs}$ time steps to predict motion dynamics during the next $T_{pred}$ time steps. Similarly to [37], our models use displacements with respect to the previous points. More specifically, given a sequence of displacements $(x_0, .., x_{T_{pred}})$, we observe a part of the sequence $(x_0, ..., x_{T_{obs}})$ and

[†]Code is available at https://github.com/alexmonti19/dagnet.

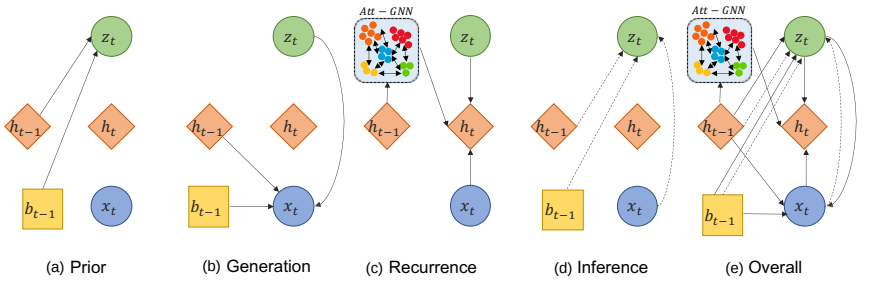



Figure 3.1: Illustration of each phase of our AC-VRNN architecture for a time step $t$. A recurrent variational autoencoder is conditioned on prior belief maps $b_{t-1}$. The hidden state of the RNN $h_{t-1}$ is refined with an attentive module obtaining $h'_t$, that replaces $h_t$ in the next step of recurrence. At inference time, it generates future displacements using the prior network on $\mathbf{h}_t$ and makes an online computation of the adjacency matrix which defines connections between pairs of nodes.

predict the subsequent one $(x_{T_{obs}+1}, ..., x_{T_{pred}})$.

## 3.2 Predictive VRNN

VRNNs ([23]) explicitly model dependencies between latent random variables $\mathbf{z}_t$ across subsequent time steps. They contain a Variational Autoencoder (VAE) ([53]) at each time step conditioned on the hidden state variable $\mathbf{h}_{t-1}$ of an RNN to take into account temporal structures of sequential data. At each time step, prior, encoder and decoder output multivariate normal distributions, with three functions ($f_{\text{pri}}$, $f_{\text{enc}}$ and $f_{\text{dec}}$) modelling their means and variances. Since the true posterior is intractable, it is approximated by a neural network $q_\phi$, which also depends on the hidden state $\mathbf{h}_{t-1}$ under recurrency equations as follows:

$$p_\theta\left(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}\right) = \mathcal{N}\left(\mathbf{z}_t|\boldsymbol{\mu}_{\text{pri,t}}, \left(\boldsymbol{\sigma}_{\text{pri,t}}\right)^2\right), \qquad \text{(prior)} \qquad (3.1)$$

$$q_\phi\left(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}\right) = \mathcal{N}\left(\mathbf{z}_t|\boldsymbol{\mu}_{\text{enc,t}}, \left(\boldsymbol{\sigma}_{\text{enc,t}}\right)^2\right), \qquad \text{(inference)} \qquad (3.2)$$

$$p_\theta\left(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}\right) = \mathcal{N}\left(\mathbf{x}_t|\boldsymbol{\mu}_{\text{dec,t}}, \left(\boldsymbol{\sigma}_{\text{dec,t}}\right)^2\right), \qquad \text{(generation)} \qquad (3.3)$$

$$\mathbf{h}_t = f_{\text{rnn}}\left(\mathbf{x}_t, \mathbf{z}_t, \mathbf{h}_{t-1}\right). \qquad \text{(recurrence)} \qquad (3.4)$$

These functions can be deep neural networks with learnable parameters $\theta$ and $\phi$ that output $(\boldsymbol{\mu}_{\mathrm{pri,t}}, \boldsymbol{\sigma}_{\mathrm{pri,t}})$, $(\boldsymbol{\mu}_{\mathrm{enc,t}}, \boldsymbol{\sigma}_{\mathrm{enc,t}})$ and $(\boldsymbol{\mu}_{\mathrm{dec,t}}, \boldsymbol{\sigma}_{\mathrm{dec,t}})$, respectively. The generative and inference processes are jointly optimized by maximizing the following variational lower bound (ELBO) with respect to their parameters[‡]:

$$ELBO = \mathbb{E}_{q_{\phi,t}(\mathbf{z}_t)}\left[\sum_{t=1}^{T}\left(-\mathrm{KL}\left(q_{\phi,t}(\mathbf{z}_t)\|p_{\theta,t}(\mathbf{z}_t)\right) + \log p_{\theta,t}(\mathbf{x}_t)\right)\right]. \quad (3.5)$$

VRNNs are typically employed to generate sequences from scratch, in a fully generative setting. However, our task is to imitate training data rather than generate completely new data at evaluation time. In a predictive setting, the predicted positions must rely on the observed ones; without any information coming from the past, future positions would only be random. Using a fully generative setting, the model would not have any chance to exploit previous observations. For this reason, we have modified the inference protocol to generate sequences using the hidden state of the last observed time step. VRNN learns at each time step to generate the current displacement, given the input and the RNN's hidden state. At inference time, the model only uses the last hidden state from the observed sequence, then generates the subsequent time step. For the above reasons, AC-VRNN and DAG-Net are generative models used in a predictive setting: they generate one displacement at a time, and embedding human interactions at time step level becomes easy.

## 3.3 Attentive hidden state refinement

Pedestrian dynamics are mainly influenced by surrounding agents. Our models handle human interactions using an attentive hidden state refinement of our recurrent network through a graph neural network, as shown in Figure 3.2(a). Our hidden state refinement resembles the idea proposed by [108] which adopts an attention mechanism to learn relative weights between two connected nodes, through specific transformations called graph attentional layers. At time step $t$, our refinement strategy considers a set of hidden state nodes $\{\mathbf{h}_t^1, \ldots, \mathbf{h}_t^N\}$, where each $\mathbf{h}_t^i \in \mathbb{R}^F$ represents the hidden state of the $i^{th}$ agent in the scene. The attention layer produces a new set of node features $\{\hat{\mathbf{h}}_t^1, \ldots, \hat{\mathbf{h}}_t^N\}$, $\hat{\mathbf{h}}_t^i \in \mathbb{R}^{F'}$ as its output. The transformation is parametrized by a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$

[‡]In order to keep the notation light we omit the conditioning variables.

(shared between graph nodes) and a weight vector $\mathbf{a} \in \mathbb{R}^{2F'}$. Self-attention coefficients $\alpha_{i,j}$ between the nodes $\mathbf{h}_t^i$ and $\mathbf{h}_t^j$ are computed as follows:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}\mathbf{h}_t^i \| \mathbf{W}\mathbf{h}_t^j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}\mathbf{h}_t^i \| \mathbf{W}\mathbf{h}_t^k\right]\right)\right)}, \tag{3.6}$$

where $\|$ represents the concatenation operator. The normalized attention coefficients are used to compute a linear combination of the features which represents the final output feature for every node, followed by an ELU non-linearity ([24]) acting on the neighbourhood $\mathcal{N}_i$ of the $i^{th}$ node:

$$\hat{\mathbf{h}}_t^i = \text{ELU}\left(\sum_{j \in \mathcal{N}_i} \alpha_{i,j} \mathbf{W}\mathbf{h}_t^j\right). \tag{3.7}$$

The neighbourhood $\mathcal{N}_i$ defines the set of nodes with positive adjacency with respect to the $i^{th}$ agent. The adjacency matrix follows a similarity-based principle, and it is computed, inspired by proxemics interaction theory ([87]), considering the heat kernel of the distance $d(i,j)$ between each pedestrian, $\exp\left(-\frac{d(i,j)}{2\sigma^2}\right)$, where $\sigma$ is a smoothing hyperparameter. During training, our VRNN takes as input a set of sequences for a time step $t$. Then, it samples the next position $\mathbf{x}_t^i$ for each pedestrian $i$. Finally, the graph attention mechanism acts on the hidden state $\mathbf{h}_t^i$ (provided by Eq. (3.4)) to compute the corresponding interaction-refined state $\hat{\mathbf{h}}_t^i$. The refined hidden state $\hat{\mathbf{h}}_t^i$ is concatenated to the original one and a final linear projection is applied as follows:

$$\mathbf{h}_t^{'i} = \text{Linear}\left(\mathbf{h}_t^i \parallel \hat{\mathbf{h}}_t^i\right). \tag{3.8}$$

At the next time step, our VRNN uses the refined hidden state $\mathbf{h}_t^{'i}$ which carries information about interactions of previous time steps.

Although AC-VRNN and DAG-Net are different models, designed to address specific problems, they share the predictive VRNN baseline and manage agents' interactions through the attentive hidden state refinement method described above. In the following subsections, the particular features of the two approaches are described in detail.

## 3.4 AC-VRNN: conditional-VRNN on belief maps

Attentive Conditional Variational Recurrent Neural Network (AC-VRNN) is composed of three building blocks: (i) a VRNN to generate a sequence of displacements in a multi-modal way; (ii) a hidden state refinement based on an attentive mechanism to model the interactions within the neighbourhood, performed at a time step level during training and inference phases; (iii) a belief map to encourage the model to follow prior belief maps when it is uncertain, avoiding predicting samples that may fall within never crossed areas. A complete illustration of all phases of AC-VRNN is shown in Figure 3.1.
Since AC-VRNN is a stochastic model, it could potentially exhibit high predictive variance hence generating predictions far from expected ones. To balance the bias/variance trade-off of the predictor, we introduce belief maps on displacements. Belief maps collect data about crossed areas at training time; therefore, they contain information about the collective behaviour of monitored agents. Conditioning the prediction to such maps may lead the model to follow past behaviours and, at the same time, discourage it to predict displacements far from past crossed areas, avoiding the generation of non-realistic paths.
Belief maps are computed by dividing the coordinate space for each scene into a $N \times M$ grid. The boundaries of this grid are defined by minimum and maximum coordinates along $x$ and $y$ directions. Both past and future information on training trajectories are considered. These values could also be obtained manually by defining the allowed area for predicting new coordinates. The values of $N$ and $M$ define the grid coarse and are computed considering the average displacement $\mu$ and its standard deviation $\sigma$ as follows:

$$N = \left\lfloor \frac{(x_{max} - x_{min})}{\frac{\mu+\sigma}{2}} \right\rfloor, \quad M = \left\lfloor \frac{(y_{max} - y_{min})}{\frac{\mu+\sigma}{2}} \right\rfloor. \tag{3.9}$$

For each grid location (bin), a $L \times L$ neighbourhood is then considered (with $L = 5$). For each $(x, y)$ location, we get the corresponding $L \times L$ neighbourhood and compute heat kernels between the next location and the neighbourhood bins centres[§]. This procedure is repeated for all the trajectories and bins values are accumulated by summation. Each belief map $\mathbf{b}$, i.e. a $L \times L$ sub-grid indexed by the $(x, y)$ location in the scene, is subsequently normalized in order to transform the cumulative grid into a probability distribution. Unlike the recurrent process

[§] $5 \times 5$ belief maps along with the proposed global grid's partition guarantee that future displacements fall into the corresponding belief maps.

**Algorithm 1** Belief Maps Generation Algorithm

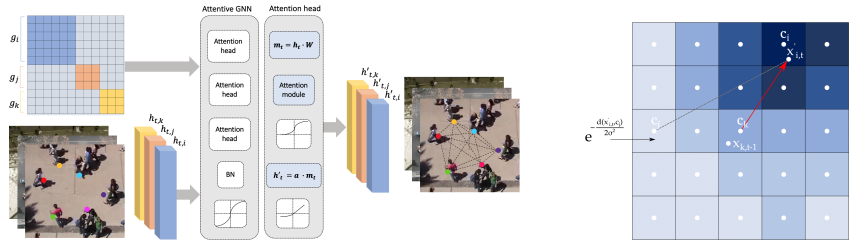
**function** BELIEF_MAPS_GENERATION($trajectories$)
  $N, M, \delta_x, \delta_y \leftarrow get\_grid\_coarse(trajectories)$
  $x_{min}, y_{min}, x_{max}, y_{max} \leftarrow get\_min\_max(trajectories)$
  $global\_grid \leftarrow make\_global\_grid(x_{min}, y_{min}, N, M, \delta_x, \delta_y)$
  **for all** $bin \in global\_grid$ **do**
    $maps \leftarrow [0, .., 0]$
    **for all** $trajectory \in trajectories$ **do**
      $neighbour\_centres \leftarrow get\_neighbour\_centres(bin, \delta_x, \delta_y)$
      **for all** $index, coord \in trajectory$ **do**
        **if** $coord_x \in [bin_x, bin_x + \delta_x]$ **and** $coord_y \in [bin_y, bin_y + \delta_y]$ **then**
          $next\_coord \leftarrow trajectory[index + 1]$
          $map \leftarrow similarity\_matrix(next\_coord, neighbour\_centres, map)$
        **end if**
      **end for**
    **end for**
    $map \leftarrow normalize(map)$
    $maps \leftarrow insert(map)$
  **end for**
  **return** $maps$
**end function**

**function** GET_GRID_COARSE($trajectories$)
  $\mu_x, \mu_y \leftarrow mean\_displacements(trajectories)$
  $\sigma_x, \sigma_y \leftarrow standard\_deviation\_displacements(trajectories)$
  $x_{min}, y_{min}, x_{max}, y_{max} \leftarrow get\_min\_max(trajectories)$
  $N \leftarrow \frac{x_{max} - x_{min}}{\frac{\mu_x + \sigma_x}{2}}; M \leftarrow \frac{y_{max} - y_{min}}{\frac{\mu_y + \sigma_y}{2}}$
  $\delta_x \leftarrow \frac{x_{max} - x_{min}}{N}; \delta_y \leftarrow \frac{y_{max} - y_{min}}{M}$
  **return** $N, M, \delta_x, \delta_y$
**end function**

**function** SIMILARITY_MATRIX($next\_coord, neighbour\_centres, map$)
  **for all** $index, centre \in neighbour\_centres$ **do**
    $map[index] \leftarrow accumulate(e^{-\sqrt{(next\_coord_x - centre_x)^2 + (next\_coord_y - centre_y)^2}})$
  **end for**
  **return** $map$
**end function**

within the VRNN, the creation of belief maps is a Markov process, as their generation only depends on single-step transitions. Details in Algorithm 1 each step for generating our belief maps.

**Conditional-VRNN.** We exploit the belief maps to encourage the model to follow the average behaviour shown by previously observed agents. In our work, we use a recurrent version of CVAE ([52]), conditioning VRNN on belief maps. At each time step, *prior*, *encoder* and *decoder* networks take the belief map at $t - 1$ as input, conditioning the resultant Gaussian distribution. We embed belief maps with a linear projection before feeding them into the VRNN blocks:




(a) (b)

Figure 3.2: Scheme of the proposed attentive hidden state refinement process. (a) The adjacency matrix is an irregular block matrix where each block size is defined by the number of pedestrians in the current scene. (b) Belief map during training for one sample using heat similarity-based strategy. The map is centred at $t-1$ to display the sampled displacements distribution at $t$.

$$\boldsymbol{\mu}_{\mathrm{pri,t}}, \boldsymbol{\sigma}_{\mathrm{pri,t}} = f_{\mathrm{pri}}\left(\mathbf{h}_{t-1}, \mathbf{b}_{t-1}; \theta\right) \tag{3.10}$$

$$\boldsymbol{\mu}_{\mathrm{enc,t}}, \boldsymbol{\sigma}_{\mathrm{enc,t}} = f_{\mathrm{enc}}\left(\mathbf{x}_{t}, \mathbf{h}_{t-1}, \mathbf{b}_{t-1}; \boldsymbol{\phi}\right) \tag{3.11}$$

$$\boldsymbol{\mu}_{\mathrm{dec,t}}, \boldsymbol{\sigma}_{\mathrm{dec,t}} = f_{\mathrm{dec}}\left(\mathbf{z}_{t}, \mathbf{h}_{t-1}, \mathbf{b}_{t-1}; \theta\right) \tag{3.12}$$

In addition to conditioning the model on belief maps, a further loss term is inserted, in order to optimize the affinity between ground-truth maps and those generated by the model. By sampling multiple displacements from the model, we obtain the sampled candidate belief map $\mathbf{b'}_{t-1}$, which identifies a probability distribution over local bin transitions. For each sampled displacement and subsequent location, we firstly index the corresponding grid bin, then the heat kernel value between the sampled next location and the $L \times L$ neighbourhood bin centres is used to fill the grid (see Figure 3.2(b)). To build the non-ground truth belief maps, we only use the information about the position at $x_{t-1}$, and then draw $N$ samples from our model. The aforementioned procedure allows the model to unroll the sub-grids, obtaining for every location a discrete probability density of possible transitions. Thus, it is possible to compare generated belief maps $\mathbf{b'}_{t-1}$ and ground-truth ones $\mathbf{b}_{t-1}$ by means of the KL divergence, exploiting the histogram loss term proposed by [104]. We add this contribution to the ELBO loss in Eq. (3.5) encouraging the model to be compliant with the collective behaviour of all agents. Such a divergence measure is multiplied by a constant $k$ for loss

balancing to ensure that its weight is comparable to the other loss components:

$$\mathcal{L} = \mathbb{E}_{q_{\phi,t}(\mathbf{z}_t)}\Big[\sum_{t=1}^{T}\Big(-\mathrm{KL}\left(q_{\phi,t}(\mathbf{z}_t)\|p_{\theta,t}(\mathbf{z}_t)\right) + \log p_{\theta,t}(\mathbf{x}_t) + k\,\mathrm{KL}(\mathbf{b}_{t-1}\|\mathbf{b'}_{t-1})\Big)\Big]. \tag{3.13}$$

## 3.5 DAG-Net: conditioning VAE to agents' goals

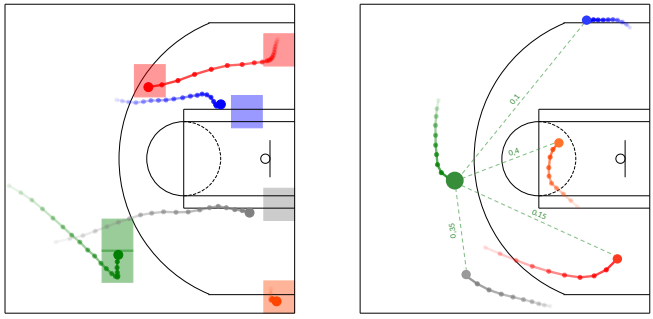
DAG-Net leverages two graph attentive networks to model two different kinds of interactions: the interactions between future goals and the interactions between agents. In structured motion environments, where agents' behaviours are moved not only by single intentions but also by social rules and/or common goals, it is important to condition the trajectory prediction on both individual and neighbour objectives. DAG-Net jointly employs past data and future predictions to improve forecasting in such contexts.
Inspired by [52, 99, 120, 10, 30, 50], we provide additional input to our backbone in order to condition the displacements generation process on agents' future objectives. We choose to describe agents' future goals in terms of spatial information (Fig. 3.3a). To make our model as invariant as possible with respect to the different characteristics of the environment, we divided the top-down view of the scene into a grid of macro-areas: each cell can potentially represent the future objective of a single agent. Agent $i$'s goal at time $t$, $\mathbf{g_t^i}$, is then represented by a one-hot encoding of the grid, where the cell in which the agent will land in the future is filled with a 1.
To obtain ground-truth objectives, a sliding window approach has been used: a window of size $w$ slides through the original absolute trajectory and captures a goal every $w$ time-steps. This information is used to condition the *prior*, the *encoder* and the *decoder* networks, as shown in Eq. (3.14), (3.15) and (3.16) where we drop the superscripts to refer to the behaviour of a general agent.

$$\boldsymbol{\mu}_{\mathbf{0},t}, \boldsymbol{\sigma}_{\mathbf{0},t} = \varphi^{prior}\left(\mathbf{h}_{t-1}, \mathbf{g}_t\right), \tag{3.14}$$

$$\boldsymbol{\mu}_{\mathbf{z},t}, \boldsymbol{\sigma}_{\mathbf{z},t} = \varphi^{enc}\left(\varphi^{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_{t-1}, \mathbf{g}_t\right), \tag{3.15}$$

$$\boldsymbol{\mu}_{\hat{\mathbf{x}},t}\boldsymbol{\sigma}_{\hat{\mathbf{x}},t} = \varphi^{dec}\left(\varphi^{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_{t-1}, \mathbf{g}_t\right). \tag{3.16}$$

To produce likely goals during the inference phase, we employed a further network. This network is again conditioned on the hidden state $\mathbf{h}_{t-1}$ of the recurrent



(a) (b)

Figure 3.3: In (a) we can observe how goals deeply influence past and future trajectories, guiding agents to specific portions of the court. In (b) we can observe the similarities between the green player and his teammates: these values will directly influence the recombination of both goals and hidden states at the green node.

cell and takes as additional inputs the last predicted objective for the agent plus the concatenation $\mathbf{d}_{t-1}$ of the absolute positions of all the other agents in the scene (i.e. their disposition).

$$\mathbf{g}'_t = \varphi^{goal}(\mathbf{g}'_{t-1}, \mathbf{d}_{t-1}, \mathbf{h}_{t-1}) \tag{3.17}$$

$$\begin{aligned} \mathbb{E}_{q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})} \Bigg[ \sum_{t=1}^{T} \sum_{k=1}^{K} \log p_\theta(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t}) - \mathbf{g}_t^k log(\mathbf{g}_t'^k) \\ - D_{KL}\left(q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \,||\, p_\theta(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t})\right) \Bigg] \end{aligned} \tag{3.18}$$

The additional loss term is computed as a Cross-Entropy between the ground-truth goal $\mathbf{g}_t$ and the predicted one $\mathbf{g'}_t$, where $K$ is total the number of cells inside their one-hot encoding.

**Goals Structure.** Agents' future objectives are used to condition the backbone network: nevertheless, without further solutions, a single predicted goal $\mathbf{g}'_t$ focuses only on the corresponding agent objective. To effectively capture the coordination between the different subjects in the scene, DAG-Net shares this kind

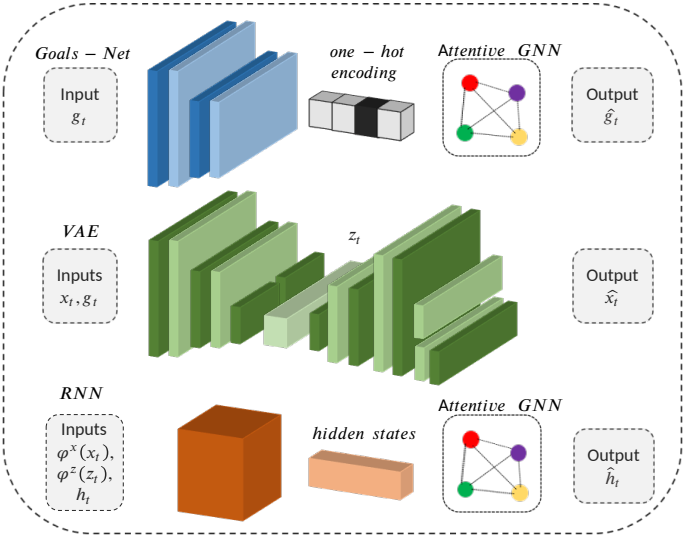



Figure 3.4: Scheme of DAG-Net architecture. It is composed of a Goal-Net that learns to predict agents' future goals; a VAE to generate displacements at every time step; a RNN to consider the temporal nature of the sequence.

of information relying on group interactions. To model the structure of future interactions, an attentive GNN [108] is employed. At every time step, the network takes as an input node the one-hot encoding of each agent's predicted goal $\mathbf{g}'_t$ and produces a new distilled goal $\tilde{\mathbf{g}}_t$ built on proximity notions. After the concatenation of the distilled goal with the original one, the final refined goal is obtained through a linear projection:

$$\hat{\mathbf{g}}_t = W\left(\mathbf{g}'_t \parallel \tilde{\mathbf{g}}_t\right) \tag{3.19}$$

where the parameter matrix $W \in \mathcal{R}^{dxd}$ (with $d$ the number of goals grid cells) is learnt in an end-to-end fashion during training. The new produced goal $\hat{\mathbf{g}}_t$ will then take the place of $\mathbf{g}'_t$ inside the ELBO loss presented in Eq. 3.18. Our whole model is depicted in Fig. 3.4.

## 3.6 Experiments

### 3.6.1 Datasets

We present experiments on different datasets to prove the robustness of our model on various scenarios and protocols. More specifically, we define multiple experiments on ETH ([79]), UCY( [64]), Stanford Drone Dataset ([88]), STATS SportVU NBA [¶], Intersection Drone Dataset (inD) ([11]), and TrajNet++ ([57]).
**ETH-UCY.** ETH ([79]) consists of two scenes, *Eth* and *Hotel*, while UCY ([64]) consists of three scenes, *Zara1*, *Zara2* and *Univ*. The benchmark contains different types of interactions among pedestrians and fixed obstacles such as buildings or parked cars.
**Stanford Drone Dataset (SDD)** ([88]). SDD is a large-scale dataset, containing urban scenes of a university campus, streets and intersections, shot by a drone. More specifically, it is composed of 31 videos of 8 different scenarios. The trajectories are split into segments of 8s: we observe 3.2s of history and predict over a 4.8s future horizon. Operating at 0.4s per time-step results in 8 time steps of observation and a future prediction span of 12 time steps; all the metrics are in meters. This dataset provides more complex scenes compared to the previous ones, involving various types of human interactions. We use the version proposed by TrajNet benchmark ([93, 7]) which contains only pedestrian annotations. We split the training set into three sets for the learning process selecting $70\%$ of data as training, $10\%$ as validation and the remaining part as testing.
**STATS SportVU NBA** [¶]. It consists of tracked trajectories of $10$ basketball players ($5$ attackers, $5$ defenders) during the $2016$ NBA season monitoring $1600$ matches. Each trajectory contains $50$ time steps sampled at $5\ Hz$ with x, y, and z coordinates expressed in feet. $40$ time steps are used as observations and $10$ time steps for predictions. All trajectories are normalized and shifted to obtain zero-centred sequences in the middle of the court.
**Intersection Drone Dataset (inD)** ([11]). It captures four different German intersections from a bird's-eye-view perspective and contains more than $11000$ trajectories of various road users (e.g., pedestrians, cars, cyclists) saved in $33$ recordings. Data is collected at $25\ Hz$ using a drone.
**TrajNet++** ([57]). It is a large-scale interaction-centric trajectory prediction benchmark composed of a real-world dataset and a synthetic dataset. The real-world dataset contains selected trajectories of different datasets (ETH [79], UCY [64],

[¶]SportVU - STATS Perform, https://www.statsperform.com/team-performance/basketball/optical-tracking/

WildTrack [20], L-CAS [114] and CFF [3]). This benchmark defines a *primary* pedestrian per scene and his/her categorization into four different types: static, linear, interacting and non-interacting.

### 3.6.2 Metrics

**TopK Average Displacement Error** (TopK ADE): Average Euclidean distance over all estimated points and ground-truth positions of a trajectory as proposed in [79]:

$$ADE = \sum_{i=1}^{\mathcal{P}} \sum_{t=T_{obs}+1}^{T_{pred}} \frac{\sqrt{(\hat{x}_t^i - x_t^i)^2 + (\hat{y}_t^i - y_t^i)^2}}{T_{pred} \cdot \mathcal{P}}; \tag{3.20}$$

**TopK Final Displacement Error** (TopK FDE): Average Euclidean distance between predicted and ground-truth final destinations:

$$FDE = \sum_{j=1}^{\mathcal{P}} \frac{\sqrt{(\hat{x}_{T_{pred}}^j - x_{T_{pred}}^j)^2 + (\hat{y}_{T_{pred}}^j - y_{T_{pred}}^j)^2}}{\mathcal{P}}. \tag{3.21}$$

$\mathcal{P}$ represents the number of pedestrians and $T_{pred}$ is the predicted time horizon. The above metrics are evaluated using the top-k (or best-of-N) i.e., we sample N trajectories and consider the ADE and FDE of the lowest-error trajectory.
**Average Log-Likelihood** (Avg LL): Average Log-Likelihood of ground truth trajectories over the predicted time horizon considering a distribution fitted with N output predictions. We compute this metric as in ([57]).
**TopK Collisions**: Similarly to [57], we consider two types of collisions, Col-I and Col-II, measuring the collisions of a pedestrian w.r.t his/her neighbours considering a fixed neighbourhood. Col-I (or prediction collision) uses the neighbours' predicted trajectories to check a collision, while Col-II relies on their ground-truth annotations. Nevertheless, since we use these metrics in a multi-modal context, we consider predictions with the lowest ADE (TopK ADE) for both primary and neighbour pedestrians. We report the percentage of collisions averaged over all test scenes.

### 3.6.3 Training protocol

During training, we let the network see the entire $T = T_{obs} + T_{pred}$ time steps from ground-truth sequences. The solution gives the model the possibility to

learn useful features also from the latest time steps of the sequence: this is useful in urban contexts and results particularly effective in sports, where we have long trajectories that usually start as linear but seldom continue in the same way, often bending and turning back upon themselves. On the other hand, during validation and testing, we divide the trajectories into an *observation* and a *prediction* split: specifically, the network burns in for $T_{obs}$ time-steps observing the first pieces of ground-truth trajectories, then it's let alone predicting the remaining $T_{pred}$ time-steps.

### 3.6.4 Quantitative results

**ETH-UCY.** We evaluate our model, AC-VRNN, using different versions of ETH-UCY datasets since multiple data and protocols are available for these scenes. Quantitative results are reported in Table 3.1. We indicate with AC-VRNN our full model including the hidden state refinement process and belief maps and with A-VRNN our model without belief maps. Firstly, we consider a leave-one-out training protocol (A) as in S-GAN ([37]). Our model outperforms all baselines on *Eth* (FDE) and *Zara2* (TopK ADE and TopK FDE with $K = 20$) scenes and exhibits the best values on average metrics. AC-VRNN significantly outperforms A-VRNN suggesting the beneficial effect of belief maps conditioning. For the remaining scenes, the slightly worse performance of AC-VRNN could be ascribed to the leave-one-out protocol since training belief maps may not entirely comply with test scenes increasing uncertainty for future predictions. SR-LSTM ([121]) defines different *Eth* annotations considering 6 frames at $0.4s$ instead of 10 frames due to a frame rate issue of original annotations, affecting each cross-validation fold (B). In this case, our model outperforms SR-LSTM baseline or achieves comparable results on all scenes for both metrics. Finally, S-Ways ([46]) does not use a leave-one-out protocol. Each dataset is split into 5 subsets, using 4 subsets for training and the remaining ones for testing purposes (C). We achieve better performance on TopK ADE and slightly worse performance on TopK FDE. Without the leave-one-out protocol, AC-VRNN significantly outperforms A-VRNN on TopK FDE suggesting the beneficial effect of belief maps conditioning.

**Stanford Drone Dataset.** To consider more complex urban scenarios, we test AC-VRNN and DAG-Net also on Stanford Drone Dataset. We compare our results with S-GAN-P ([37]) and STGAT ([117]). As shown in Table 3.2, AC-VRNN outperforms A-VRNN version and both selected baselines. DAG-Net outperforms all the approaches except for AC-VRNN. With more complex trajectories

| | Method | ETH | HOTEL | UNIV | ZARA1 | ZARA2 | AVG |
|---|---|---|---|---|---|---|---|
| (A) | S-LSTM ([2]) | 1.09/2.35 | 0.79/1.76 | 0.67/1.40 | 0.47/1.00 | 0.56/1.17 | 0.72/1.54 |
| | S-GAN-P ([37]) | 0.87/1.62 | 0.67/1.37 | 0.76/1.52 | 0.35/0.68 | 0.42/0.84 | 0.61/1.21 |
| | S-GAN ([37]) | 0.81/1.52 | 0.72/1.61 | 0.60/1.26 | 0.34/0.69 | 0.42/0.84 | 0.58/1.18 |
| | Trajectron ([45]) | **0.59**/1.14 | 0.35/0.66 | 0.54/1.13 | 0.43/0.83 | 0.43/0.85 | 0.56/1.14 |
| | SoPhie ([94]) | 0.70/1.43 | 0.76/1.67 | 0.54/1.24 | **0.30**/0.63 | 0.38/0.78 | 0.54/1.15 |
| | Social-BiGAT ([56]) | 0.69/1.29 | 0.49/1.01 | 0.55/1.32 | **0.30/0.62** | 0.36/0.75 | 0.48/1.00 |
| | Next ([67]) | 0.73/1.65 | **0.30**/0.59 | 0.60/1.27 | 0.38/0.81 | 0.31/0.68 | 0.46/1.00 |
| | STGAT ([117]) | 0.78/1.60 | **0.30/0.54** | **0.51/1.08** | 0.33/0.72 | 0.29/0.63 | 0.44/0.91 |
| | A-VRNN (Ours) | 0.73/1.45 | 0.34/0.65 | 0.53/1.14 | 0.33/0.69 | **0.26/0.54** | 0.44/0.89 |
| | AC-VRNN (Ours) | 0.61/**1.09** | **0.30**/0.55 | 0.58/1.22 | 0.34/0.68 | 0.28/0.59 | **0.42/0.83** |
| (B) | SR-LSTM ([121]) | 0.63/1.25 | **0.37/0.74** | **0.51/1.10** | 0.41/0.90 | 0.32/0.70 | 0.45/0.94 |
| | A-VRNN (Ours) | **0.60/1.18** | **0.37/0.74** | 0.55/1.20 | **0.39/0.83** | **0.27/0.59** | **0.44/0.91** |
| (C) | S-Ways ([46]) | **0.39/0.64** | 0.39/0.66 | **0.55/1.31** | 0.44/**0.64** | 0.51/0.92 | 0.46/**0.83** |
| | A-VRNN (Ours) | 0.60/1.24 | 0.22/0.45 | 0.61/1.34 | 0.46/1.06 | **0.30/0.67** | **0.44**/0.95 |
| | AC-VRNN (Ours) | 0.55/1.06 | **0.18/0.26** | 0.76/1.59 | **0.37**/0.72 | 0.33/0.70 | **0.44**/0.87 |

Table 3.1: Quantitative results of considered methods for ETH and UCY datasets. We report Average Displacement Error (ADE) and Final Displacement Error for unimodal methods and TopK ADE and TopK FDE (with $K = 20$) for multi-modal ones. The results were obtained for $t_{obs} = 8$ and $t_{pred} = 12$ (in meters). The first block of experiments regards the use of data employed by S-GAN and STGAT models; the second one uses the SR-LSTM version of data while the last experiments are trained with the S-Ways protocol. On average, our model outperforms several methods showing a slightly worse FDE error when the S-Ways protocol is employed. No belief maps appear necessary for SR-LSTM data version.

and scene topologies, our attentive module is able to better capture interactions among pedestrians and belief maps help to avoid incorrect behaviours following the prior distribution of displacements in the monitored scene.

**STATS SportVU NBA.** Additionally, we test our model using basketball players' trajectories whose dynamics are clearly different from the ones exhibited by pedestrians in urban scenes. We evaluated separately offence and defence: since agents are placed in an explicit competitive setting, their nature is intrinsically different, both from the goals and the trajectories points of view. The attackers have to enter into the area and score, while the defenders usually just react to their moves: training the network simultaneously on both teams would distract the final results. As reported in Table 3.3, our A-VRNN reduces TopK ADE and TopK FDE metrics on both offensive and defensive players' trajectories compared to STGAT [117] and Weak-Supervision [120]. Avg LLs are similar for all methods, whereas collision errors given by A-VRNN are mainly smaller than

| Method | SDD | | | | |
|---|---|---|---|---|---|
| | TopK ADE (↓) | TopK FDE (↓) | Avg LL (↑) | Col-I (↓) | Col-II (↓) |
| S-GAN-P ([37]) | 0.65 | 1.26 | −3.79 | 0.00 | 0.33 |
| STGAT ([117]) | 0.57 | 1.09 | −2.70 | 0.00 | 0.40 |
| DAG-Net (Ours) ([77]) | 0.54 | 1.07 | −2.54 | 0.49 | 0.25 |
| A-VRNN (Ours)([9]) | 0.55 | 0.98 | −1.11 | **0.00** | **0.11** |
| AC-VRNN (Ours) ([9]) | **0.51** | **0.90** | **-0.18** | 0.16 | 0.22 |

Table 3.2: Results for $t_{obs} = 8$ and $t_{pred} = 12$ on Stanford Drone Dataset (in meters). AC-VRNN significantly reduces TopK ADE and TopK FDE error metrics. Average NLL is the best one among all approaches while collision errors are below 1% for all methods.

| Team | Method | STATS SportVU NBA | | | | |
|---|---|---|---|---|---|---|
| | | TopK ADE (↓) | TopK FDE (↓) | Avg LL (↑) | Col-I (↓) | Col-II (↓) |
| ATK | STGAT ([117]) | 9.94 | 15.80 | −8.65 | 0.21 | 0.32 |
| | Weak-Supervision ([120]) | 9.47 | 16.98 | −6.29 | 0.57 | 0.20 |
| | A-VRNN (Ours)([9]) | 9.32 | 14.91 | −7.60 | **0.09** | **0.18** |
| | DAG-Net (Ours) ([77]) | **8.98** | **14.08** | **-0.02** | 0.14 | 0.21 |
| DEF | STGAT ([117]) | 7.26 | 11.28 | −7.88 | 0.20 | **0.27** |
| | Weak-Supervision ([120]) | 7.05 | 10.56 | −5.69 | 0.70 | 0.57 |
| | A-VRNN (Ours) ([9]) | 7.01 | 10.16 | −6.70 | **0.13** | 0.43 |
| | DAG-Net ([77]) | **6.87** | **9.76** | **-0.05** | 0.31 | 0.66 |

Table 3.3: Results for $t_{obs} = 10$ and $t_{pred} = 40$ in feet on STATS SportVU NBA dataset.

| Method | inD | | | | |
|---|---|---|---|---|---|
| | TopK ADE (↓) | TopK FDE (↓) | Avg LL (↑) | Col-I (↓) | Col-II (↓) |
| S-GAN ([37]) | 0.48 | 0.99 | −1.84 | **0.51** | 0.55 |
| STGAT ([117]) | 0.48 | 1.00 | −1.55 | 0.60 | 0.58 |
| A-VRNN (Ours) | 0.45 | 0.97 | −1.69 | 0.61 | **0.52** |
| AC-VRNN (Ours) | **0.42** | **0.80** | **-0.29** | 0.78 | 0.61 |

Table 3.4: Results for $t_{obs} = 8$ and $t_{pred} = 12$ in meters on inD dataset.

the errors generated by competitive approaches. In this case, belief maps cannot properly steer future positions since basketball courts do not have obstacles and never-crossed areas. Moreover, basketball players do not typically follow collective behaviour. On the other end, by jointly considering agents' interactions and future goals, DAG-Net is more able to capture the nature of real paths and reach smaller errors with respect to all these competitive methods.

| Method | **TrajNet++** | |
|---|---|---|
| | ADE/TopK ADE (↓) | FDE/TopK FDE (↓) |
| S-LSTM ([2]) | 0.55 | 1.18 |
| S-ATT ([109]) | 0.56 | 1.22 |
| S-GAN ([37]) | **0.51** | **1.09** |
| D-LSTM([57]) | 0.57 | 1.23 |
| AC-VRNN (Ours) | 0.57 | <u>1.17</u> |

Table 3.5: Results for $t_{obs} = 9$ and $t_{pred} = 12$ in meters on TrajNet++. For unimodal methods ADE and FDE metrics are reported while for multimodal ones we reported the TopK ADE and TopK FDE metrics with $K = 3$.

To evaluate whether DAG-Net could show appreciable performance on different prediction horizons, we produced some *long-term evaluations*: since basketball trajectories offered a high number of time steps with which we could produce various splits, we focused on sports. For producing such evaluations, we concentrated on different observation-prediction sequences: given 10 time steps of observation, we evaluated all the methods on increasing prediction splits, from 10 time steps to 40 time steps, with steps of 10. As Fig. 3.5 shows, our method globally outperforms the competitors in all the different evaluations and in both metrics. As for the numbers in Table 3.3, the difference is more pronounced for the attack than for the defence.

We have also run some long-term quantitative evaluations considering a longer observation period (Table 3.6), mainly to observe how the prediction accuracy changes when the model is allowed to adjust to a greater initial period: we let the model burn-in for 20 initial time steps and then predict the remaining ones, again with increasing steps of 10. In this setting, we are able to compare DAG-Net to a further autoencoder architecture, by Felsen et al. [30], that briefly employs a C-VAE [99] conditioned on players' role. Our model, even without additional information about players' identities, shows better metrics in terms of the average distance from ground-truth positions.

**Intersection Drone Dataset.** On InD dataset, we adopt the same evaluation protocol used for Stanford Drone Dataset considering, for each scene, 70% of data as training, 10% as validation and the remaining part for testing. We retain only pedestrians' trajectories and downsample each scene to obtain 20 time steps in 8 s. In Table 3.4 we compare our model to S-GAN ([37]) and STGAT ([117]). AC-VRNN overcomes all the competitive methods on TopK ADE and TopK FDE

| Model | Team | 20-10 Split TopK ADE | 20-20 Split TopK ADE | 20-30 Split TopK ADE |
|---|---|---|---|---|
| C-VAE [30] | ATK | 3.95 | 5.80 | 7.08 |
| DAG-Net (Our) | ATK | **2.09** | **4.58** | **6.66** |
| C-VAE [30] | DEF | 3.01 | 4.10 | **4.98** |
| DAG-Net (Our) | DEF | **2.05** | **4.07** | 5.01 |

Table 3.6: Long-term quantitative evaluations for DAG-Net model on STATS SportVU NBA dataset.

and Avg LL. S-GAN gives a smaller Col-I error with respect to AC-VRNN and S-GAN, while A-VRNN shows a smaller Col-II error.

**TrajNet++.** Finally, we test our model on TrajNet++ ([57]) real-world dataset. The results are reported in Table 3.5 where ADE and FDE metrics are used for unimodal methods and TopK ADE and TopK FDE (with $K = 3$) metrics for multimodal ones. We find that our model reaches competitive performance with respect to other approaches, especially for TopK FDE. Our results are obtained by submitting the results to the evaluation server averaging the results on different types of scenes considering only the real dataset. We compare AC-VRNN against published competitive approaches as competing with a lead board that is updated every day is out of the scope of this quantitative analysis. Other methods' results are reported from ([57, 72]). Since the Avg LL for competitive methods is missing, we do not report this metric in Table 3.5. However, our method attains an Avg LL of $-8.33$.

### 3.6.5 Ablation experiments

We also present an ablation study to show the contribution of different components of our model on the prediction task. In the following, we detail each component and report quantitative results in Table 3.7 and Table 3.8.

**Vanilla Variational Recurrent Network.** We investigate the ability of Vanilla VRNNs to predict accurate trajectories on ETH, UCY and SDD datasets. This model does not consider any human interactions or prior scene knowledge. ETH scenes appear mainly affected by the lack of additional information while UCY

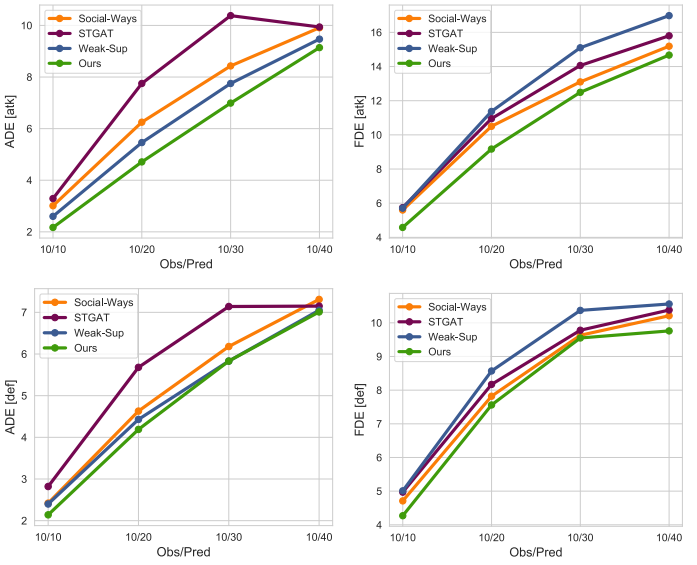



Figure 3.5: Long-term quantitative evaluations: the method is evaluated both in TopK ADE and TopK FDE for increasing prediction lengths, from 10 to 40 time-steps on STATS SportVU NBA dataset. Attack on the top, defence on the bottom. All the metrics are in feet.

scenes attain comparable results to our models, especially for the TopK ADE metric. Such a result highlights the importance of trajectory forecasting task to go beyond a time-series problem and the need of including contextual information about the scene, such as human interactions or experience gained in similar contexts.

**Hidden State Refinement with Graph Convolutional Neural Network.** This experiment models interaction with a hidden state refinement based on a standard Graph Convolutional Networks (GCN). The model has worse performance compared to our models and Vanilla VRNN models on ETH and UCY datasets while obtaining comparable results to AC-VRNN on SDD dataset. The experiments suggest that, for complex contexts, attention mechanisms are able to capture more useful information in order to model interactions among pedestrians compared to simple scenarios where interactions may be reduced.

**Adjacency Matrix.** We also evaluate our model using different kinds of adjacency matrices to corroborate the use of the similarity one. We consider an *all-1* adjacency matrix where edges are equally weighted and all pedestrians in the scene are connected. This model attains good performance but is slightly worse than the ones obtained with a similarity matrix on both ETH/UCY and SDD, proving that assuming the same importance for all involved agents negatively affects the results. $k$-NN matrix only considers nearby pedestrians. The neighbourhood is computed by sorting mutual distances between each pedestrian, retaining only the first $k$ nearest neighbours (with $k = 3$), defined as a set $S_i$. Each element is set to 1 if $a_{i,j} \in S_i$, to 0 otherwise. $k$-NN matrix obtains quite the worst results on ETH and UCY datasets and performs poorly on SDD dataset. This experiment demonstrates that a small neighbourhood is not able to capture interactions in large scenes where pedestrians show mutual influences also at long distances.

**Hidden State Initialization.** The hidden state initialization has a strong impact on the RNN training process. We experiment with three different initialization approaches:

- *Zero initialization*: a simple zero-tensor initialization.
- *Learned initialization*: a linear layer is trained to learn an optimal initialization.
- *Absolute coordinate initialization*: the tensor is initialized with the first absolute coordinates to provide spatial information to the learning process that is based on displacement generation.

We experimentally notice that the *absolute coordinate initialization* has a significant impact on the recurrent process leading to a performance improvement on ETH/UCY dataset and on SDD, while on STATS SportVU NBA InD and TrajNet++ the *zero initialization* is preferable.

**Block Irregular Adjacency Matrix.** AC-VRNN and DAG-Net are based on a single Variational Recurrent Neural Network with shared parameters. To jointly compute a unique adjacency matrix for each time step, we build a block matrix where each block contains the matrix corresponding to a single scene, randomly chosen from the training dataset. Blocks can have different dimensions since a variable number of agents may be present in the scene.

| Method | ETH | HOTEL | UNIV | ZARA1 | ZARA2 | AVG |
|---|---|---|---|---|---|---|
| Vanilla VRNN | 0.79/1.61 | 0.46/0.94 | 0.55/1.20 | 0.34/0.75 | 0.26/0.58 | 0.48/1.02 |
| GCN-VRNN | 0.81/1.58 | 0.41/0.85 | 0.59/1.31 | 0.38/0.84 | 0.41/0.96 | 0.52/1.11 |
| AC-VRNN w/o KLD | 0.73/1.41 | 0.52/1.07 | 0.64/1.36 | 0.43/0.89 | 0.39/0.83 | 0.54/1.11 |
| All-1 ADJ Matrix | 0.77/1.52 | 0.37/0.73 | 0.55/1.19 | 0.34/0.75 | 0.26/0.58 | 0.46/0.95 |
| kNN ADJ Matrix | 0.76/1.54 | 0.47/0.99 | 0.57/1.26 | 0.42/0.95 | 0.26/0.58 | 0.50/1.01 |
| A-VRNN (Ours) | 0.73/1.45 | 0.34/0.65 | **0.53/1.14** | **0.33**/0.69 | **0.26/0.54** | 0.44/0.89 |
| AC-VRNN (Ours) | **0.61/1.09** | **0.30/0.55** | 0.58/1.22 | 0.34/**0.68** | 0.28/0.59 | **0.42/0.83** |

Table 3.7: Ablation experiments showing TopK ADE and TopK FDE for $t_{obs} = 8$ and $t_{pred} = 12$ in meters on ETH, UCY and SDD datasets. The AVG column reports average results for ETH and UCY datasets.

#### 3.6.5.1 AC-VRNN

**AC-VRNN without KLD Loss on Belief Maps.** To demonstrate the importance of KL-divergence loss on belief maps, we train our model without this term yet still conditioning the model on them. We obtain the worst results on all datasets proving that the network is not able to integrate belief maps information conditioning only VAE components. KL-divergence allows the network to generate displacement distributions similar to the ground-truth ones and to follow prior knowledge about local behaviours.

**Belief Maps Dimension.** Since belief maps define the probability that a pedestrian in a cell will move towards another one, it is important to consider a proper cell dimension. If we consider a fine-grained grid ($L = 3$), we could discard information about pedestrians whose displacement is greater than the defined one. Likewise, if we consider a course-grained grid ($L = 9$), the outermost cells may not be properly filled. To select the best value of the parameter $L$, we test our model using different cell dimensions and found that $L = 5$ is the best choice for our datasets.

#### 3.6.5.2 DAG-Net

In Table 3.9 and Table 3.10, we present ablation experiments to prove the improvements performed by each component of our model. Results present two baselines: the Vanilla VRNN and A-VRNN, the version with one single attentive graph on hidden states. Our DAG-Net outperforms both baselines on STATS

| Method | SDD | |
|---|---|---|
| | TopK ADE (↓) | TopK FDE (↓) |
| Vanilla VRNN | 0.56 | 1.15 |
| GCN-VRNN | 0.53 | 1.05 |
| AC-VRNN w/o KLD | 0.60 | 1.11 |
| All-1 ADJ Matrix | 0.57 | 1.11 |
| kNN ADJ Matrix | 0.73 | 1.43 |
| A-VRNN | 0.56 | 1.14 |
| AC-VRNN ($L = 3$) | 0.67 | 1.31 |
| AC-VRNN ($L = 5$) | **0.51** | **0.92** |
| AC-VRNN ($L = 7$) | 0.68 | 1.33 |

Table 3.8: Ablation experiments showing TopK ADE and TopK FDE for $t_{obs} = 8$ and $t_{pred} = 12$ for SDD dataset.

| **Team** | **Model** | **Agents' interact.** | **Future object.** | **ADE** | **FDE** |
|---|---|---|---|---|---|
| ATK | Vanilla VRNN [23] | ✗ | ✗ | 9.58 | 15.83 |
| | A-VRNN | ✓ | ✗ | 9.67 | 15.96 |
| | DAG-Net (Our) | ✓ | ✓ | **9.18** | **13.54** |
| DEF | Vanilla VRNN [23] | ✗ | ✗ | 7.07 | 10.62 |
| | A-VRNN | ✓ | ✗ | 7.01 | 10.42 |
| | DAG-Net (Our) | ✓ | ✓ | **7.01** | **9.76** |

Table 3.9: Ablation experiments for DAG-Net on STATS SportVU NBA dataset.

SportVU NBA and SDD. The Vanilla VRNN experiments show that using a stand-alone network without considering interactions between agents does not allow the model to capture the nature of real paths. For this reason, A-VRNN achieves better performance than Vanilla VRNN; still, the model is not able to capture future structured dependencies between agents. The results obtained with DAG-Net highlight the importance of inserting future information into the prediction and combining humans' objectives in a structured way.


Figure 3.6: Illustration of predicted trajectories using AC-VRNN, baselines and competitive methods on *Eth* (left) and *Zara1* (middle) scenes of ETH and UCY datasets and *gates_0* and *deathCircle_1* of SDD (right).

| Model | Agents' interact. | Future object. | ADE | FDE |
|---|---|---|---|---|
| Vanilla VRNN [23] | ✗ | ✗ | 0.57 | 1.16 |
| A-VRNN | ✓ | ✗ | 0.56 | 1.14 |
| DAG-Net (Our) | ✓ | ✓ | **0.54** | **1.05** |

Table 3.10: Ablation experiments for DAG-Net on Stanford Drone Dataset.

## 3.6.6 Qualitative results

### 3.6.6.1 AC-VRNN

Figure 3.6 presents some qualitative experiments, comparing our model with baselines and competitive methods. On *Eth*, GCN-VRNN, based on a Graph Convolutional Neural Network, generates trajectories that significantly drift from the ground-truth ones. On *Zara1*, all considered models are able to follow correct paths, but AC-VRNN appears more able to predict complex trajectories such as the entrance into a building, following the collective agents' behaviour. For SDD,

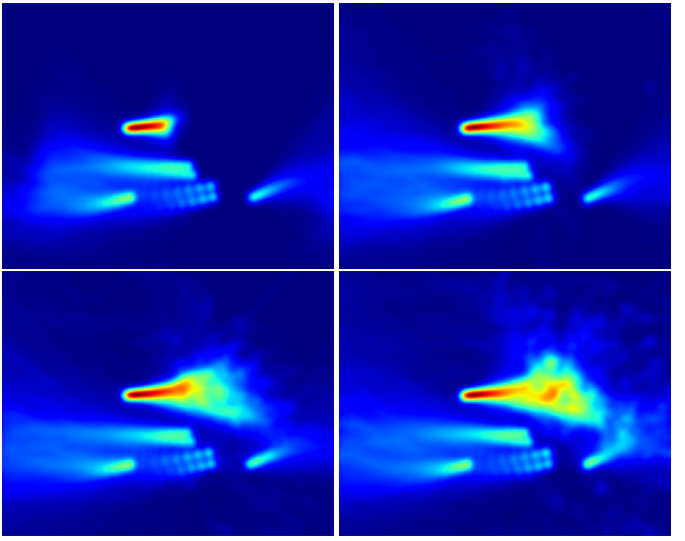


Figure 3.7: Heatmaps of the predictions probability distribution for long-term predictions. The predictions are made for $t_{obs} = 8$ and $t_{pred} = 20, 60, 120$ and 200, respectively (from left to right). We select *Zara1* scene and observe that the trajectories are coherent with the scene topology.

we randomly select two scenes and show our model samples against competitive methods. All methods predict plausible paths, but AC-VRNN generates more realistic trajectories in some cases, following the sidewalk rather than crossing the road diagonally.

**Long-term predictions.** Since AC-VRNN is a completely generative model, it is possible to generate an unlimited number of future positions as well as create trajectories without any observations. This could be especially useful for applications that require sampling a large number of trajectories to simulate realistic motion dynamics as required by synthetic scenarios mimicking real-life situations. Obviously, as the number of time steps increases, the predicted paths tend to drift from realistic ones, but our model qualitatively predicts plausible trajectories even after several time steps. To this end, we show in Figure 3.7 some qualitative experiments considering up to 200 time steps.

**Multimodal predictions.** Figure 3.8 depicts other qualitative examples generated by the AC-VRNN model showing multiple paths to demonstrate the abil-


Figure 3.8: Multiple predictions of AC-VRNN trajectories to highlight the multi-modality nature of our model on ETH and UCY datasets.

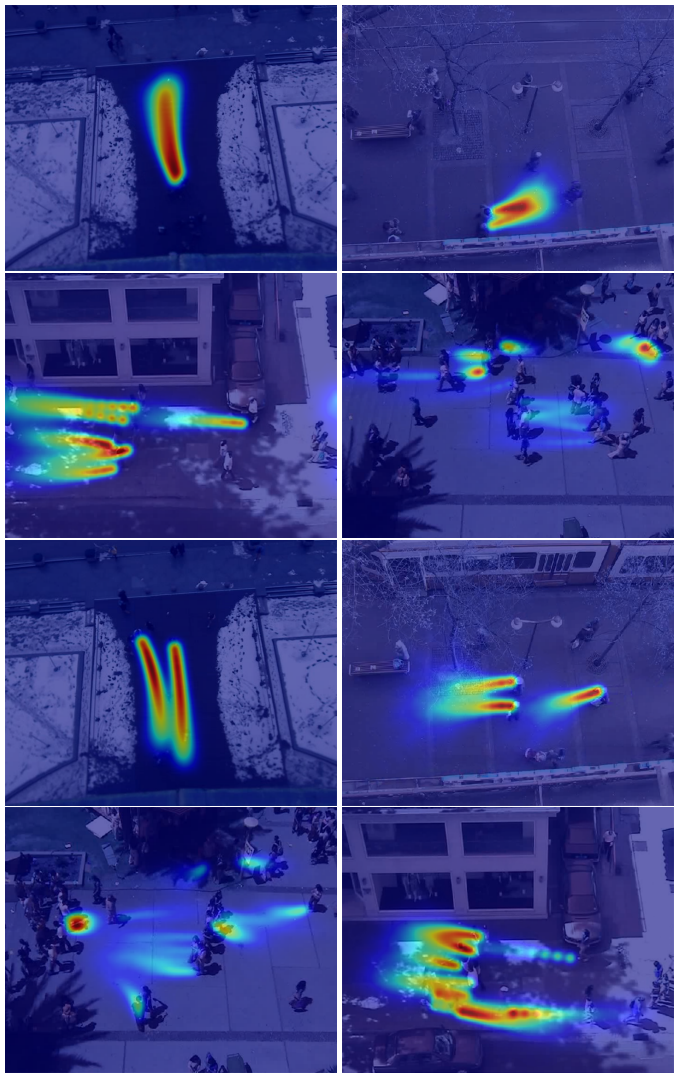


Figure 3.9: Heatmaps representing probability distributions generated by our model for ETH and UCY datasets.

ity of our model to predict multi-modal trajectories. Finally, Figure 3.9 shows probability distributions of future paths. When interactions among pedestrians are limited or absent, our model correctly predicts continuous linear paths. By contrast, the increasing number of human interactions leads the predictions to simulate complex patterns.

#### 3.6.6.2 DAG-Net

In competitive settings such as sports, the opposing teams are deeply different. The attacking team drives the game trying to score, while the defenders often limit to just counter-react to its moves. These behaviours deeply affect the resulting trajectories, as can be clearly seen in the roll-outs presented in Fig. 3.10. Attackers' trajectories tend to be particularly varied and intricate, often bending and intersecting; on the contrary, defenders tend to move linearly and occasionally deflect to follow an opponent or to close a gap. Despite some sudden changes of direction in the real trajectories, especially for the attackers, our model is able to correctly predict the overall future movement of the players. Because of the complexity of such trajectories, the predictions do not always precisely resemble the expected output: nevertheless, even when the predictions fail to follow the real ground-truth trajectories, the model still predicts a likely behaviour coherent with the play development, proving its strength in capturing the multi-modal nature of players' movements.

On the other hand, urban trajectories are more straightforward, because pedestrians obviously tend to move linearly, doing only some occasional deviations to avoid collisions or to turn. Nevertheless, the adoption of agents' goals gives the model the possibility to produce more likely trajectories. Since gents are constrained to pass through specific portions of the scene coherent with their motion behaviour, predictions can closely resemble real future movements: in both the plot reported in Fig. 3.11, DAG-Net is able to keep closer to the ground-truth, while both the competitors tend to predict more linear trajectories and consequently deviate from the expected output. For the very same reasons, final predictions can also be more precise: having important insights about the regions the agent will occupy in the future can help the model appropriately predict the overall portion of the scene where the agent will land at the end of his trajectory. DAG-Net predicted the final location resembles the agent's real destination, while both the competitors fail to approximately forecast such information.

| Hyperparameter | ETH/UCY | SDD | STATS SportVU NBA | TrajNet++ | inD |
|---|---|---|---|---|---|
| Optimizer | Adam | Adam | Adam | SGD | Adam |
| Learning rate | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $3 \times 10^{-4}$ | $10^{-4}/5 \times 10^{-4}$ |
| Batch size | 16 | 16 | 32 | 8 | 16 |
| Latent space size | 16 | 16 | 32 | 16 | 16 |
| Warm-up epochs | 50 | 50 | - | 3 | 50 |

Table 3.11: Main hyperparameters used to train both AC-VRNN and A-VRNN models on tested datasets.

## 3.6.7 Implementation details

### 3.6.7.1 AC-VRNN

We train our model for 500 epochs on ETH-UCY and SDD, for 300 epochs on STATS SportVU NBA, for 300 epochs on inD and for 25 epochs on TrajNet++. Except for ETH/UCY, we re-train all competitive methods for the same number of epochs and report the best results after performing a hyperparameter search retaining the best model on the validation set. For ETH/UCY we report results from the original paper except for STGAT ([117]) which has been re-trained with the best hyperparameters proposed by the authors. We use gradient clipping set to 10. For the SGD optimizer we use a momentum of 0.9. The RNN is a GRU with 1 layer and hidden size equal to 64. The attentive GNN has a hidden size of 8 with 4 attention heads. Each belief map during training is generated by sampling 100 displacements. In Eq. 3.13, $k$ is set to 100 for all the datasets. Other hyperparameters that vary according to the dataset are reported in Table 3.11. In Table 3.12 an overall description of AC-VRNN architecture is reported. [‖]

**Warm-up on VRNN KL-Divergence.** VRNN is trained with the ELBO loss that is composed of two terms: Negative Log-Likelihood and KL-Divergence. To correctly balance these two terms, we use a warm-up method that increases the weight in the range $[0, 1]$ of the KL-Divergence up to $N$ epochs. After this learning period, we fix the KL weight to 1. This technique favours the reconstruction error during the early epochs in order to first teach the network to generate correct samples and then to approach both *encoder*'s and *prior*'s means and log-variances.

[‖]For a more detailed explanation see ([108]) and https://github.com/Diego999/pyGAT

| Module | Architecture |
|---|---|
| Features extraction (trajectory) | Linear (2, 64) $\rightarrow$ LeakyReLU $\rightarrow$ Linear(64, 64) $\rightarrow$ LeakyReLU |
| Features extraction (belief map) | Linear (64, 64) $\rightarrow$ LeakyReLU |
| Prior<br>Mean<br>Log-variance | Linear(128, 64) $\rightarrow$ LeakyReLU<br>Linear(64, 16)<br>Linear(61, 16) |
| Encoder<br>Mean<br>Log-variance | Linear(192, 64) $\rightarrow$ LeakyReLU $\rightarrow$ Linear(64, 64) $\rightarrow$ LeakyReLU<br>Linear(64, 16)<br>Linear(64, 16) |
| Latent space | Linear(16, 64) $\rightarrow$ LeakyReLU |
| Decoder<br>Mean<br>Log-variance | Linear(192, 64) $\rightarrow$ LeakyReLU $\rightarrow$ Linear(64, 64) $\rightarrow$ LeakyReLU<br>Linear(64, 16) $\rightarrow$ HardTanH(-10, 10)<br>Linear(64, 16) |
| Recurrence | GRU(128, 64, 1) |
| Graph | GraphAttentionLayer(64, 64, hidden_units=8, heads=4, $\alpha = 0.2$) $\rightarrow$ BatchNorm1D $\rightarrow$ TanH |

Table 3.12: Detailed description of each module of our AC-VRNN architecture.

#### 3.6.7.2 DAG-Net

The VRNN recurrent cell is a GRU with 1 recurrent layer and a hidden state dimension of 64; the dimension of the latent variable is set to 32. For each graph, we then employ two attentive GNN layers: the first layer reduces the input to lower-dimensional hidden space, and the second layer returns instead to the original input space. Each GNN layer uses 4 attention heads. The entire model has been optimised with Adam optimiser. To cope with the differences between urban and sports settings, we employ different sets of hyper-parameters.
For the urban setting, we use a learning rate of $10^{-4}$ and a batch size of 16; the Cross-Entropy contribution is weighted with a factor of $10^{-2}$. The hidden state dimension between the two graph layers is set to 4. The model has been trained for 500 epochs.
For the sports setting, we use a learning rate of $10^{-3}$ and a batch size of 64; the Cross-Entropy contribution is weighted with a factor of $10^{-2}$. The hidden state dimension between the two graph layers is set to 8. The model has been trained for 300 epochs.

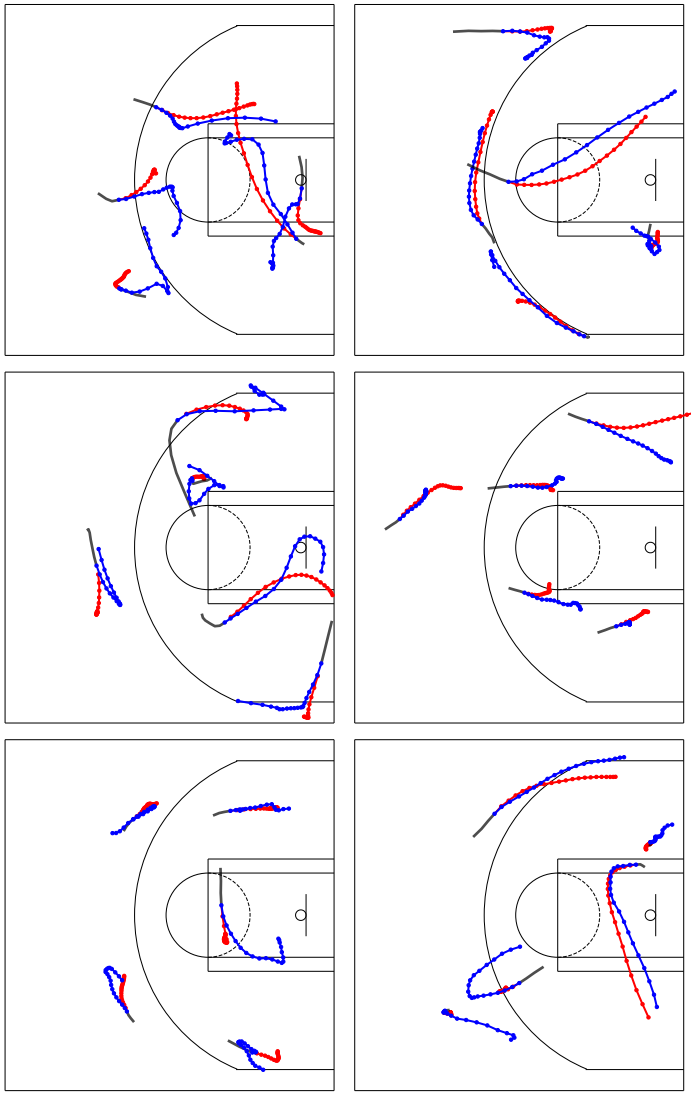


Figure 3.10: Basketball roll-outs. After an initial observation stage (black), model predictions (red) are evaluated against the ground truth (blue), The top roll-outs refer to three different attack plays, while the bottom one represents three different defensive actions.


Figure 3.11: Qualitative samples that compare DAG-Net and state-of-the-art methods on Stanford Drone Dataset.

# Chapter 4

# Meta-Continual Learning in Complex Scenarios

Human-like learning has always been a challenge for deep learning algorithms. Neural networks work differently than the human brain, needing a large number of independent and identically distributed (iid) labelled data to face up the training process. Due to their weakness in directly dealing with few, online, and unlabelled data, the majority of deep learning approaches are bounded to specific applications. Continual learning, meta-learning, and unsupervised learning try to overcome these limitations by proposing targeted solutions.

In particular, continual learning has been largely investigated in the last few years to solve the catastrophic forgetting problem that affects neural networks trained on incremental data. When data are available as a stream of tasks, neural networks tend to focus on the most recent, overwriting their past knowledge and consequently causing forgetting. Several methods [55, 73, 19, 86, 119, 85] have been proposed to solve this issue involving a memory buffer, network expansion, selective regularisation, and distillation. Some works [48, 110, 81, 69, 38, 115] take advantage of the meta-learning abilities of generalisation on different tasks and rapid learning on new ones to deal with continual learning problems, giving life to meta-continual learning [47] and continual-meta learning [13]. Due to the complex nature of the problem, the proposed approaches generally involve supervised or reinforcement learning settings. Furthermore, few works on unsupervised meta-learning [44, 51, 49] and unsupervised continual learning [83] have

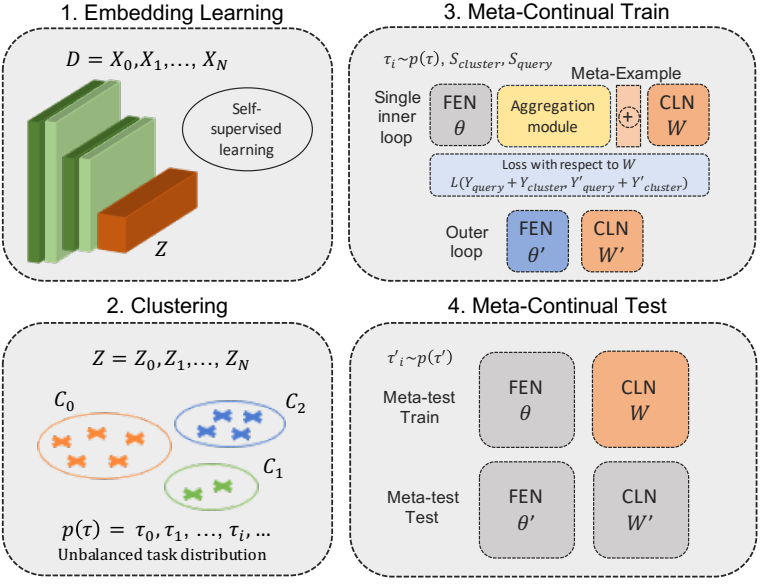



Figure 4.1: Overview of FUSION learning strategy. The model is composed of 4 phases: 1) an embedding learning network that learns a suitable embedding for each sample; 2) an unsupervised task construction phase in which clustering is applied over these embedding 3) a meta-continual training phase consisting of a two-loop procedure performed on the unsupervised tasks built in phase 2. The architecture for meta-continual training consists of a feature extraction network (FEN) that learns features useful across tasks, a self-attention-based aggregation module that collapses examples in the inner loop into a single meta-example, and a classification network (CLN) that performs tasks-level classification. The FEN is frozen in the inner loop (grey box); 4) a meta-continual test phase that fine-tunes only the classification network for new unseen classes.

been recently proposed, but the first ones deal with iid data, while the second one assumes the availability of a huge dataset. Moreover, the majority of continual learning solutions assume that data are perfectly balanced or equally distributed among classes. This problem is non-trivial for continual learning since specific solutions have to be found to preserve a balanced memory in presence of an imbalanced stream of data [22].

In this thesis, we introduce FUSION (standing for Few-shot UnSupervIsed cONtinual learning), a new learning strategy for unsupervised meta-continual learning

| Few-shot | Unsupervised | Continual | Imbalance | OoD | Algorithm |
|---|---|---|---|---|---|
| - | - | ✓ | - | - | iCARL [85] |
| - | ✓ | ✓ | - | - | CURL [83] |
| ✓ | ✓ | - | - | - | CACTUs [44] |
| ✓ | ✓ | - | - | - | UMTRA [51] |
| ✓ | ✓ | - | - | - | UFLST [49] |
| ✓ | - | - | ✓ | ✓ | L2B [61] |
| - | ✓ | ✓ | - | ✓ | GD [62] |
| ✓ | - | ✓ | - | - | OML [47] |
| ✓ | - | ✓ | - | - | ANML [6] |
| ✓ | - | ✓ | - | ✓ | Continual-MAML [13] |
| ✓ | - | ✓ | - | - | iTAML [81] |
| ✓ | ✓ | ✓ | ✓ | ✓ | FUSION (Ours) |

Table 4.1: Features comparison between FUSION and several works recently proposed in the literature involving continual learning and few-shot learning in the wild.

that can learn from small datasets and does not require the underlying tasks to be balanced. *

To the best of our knowledge, an unsupervised meta-continual learning setting has never been studied before in the literature. However, a plethora of various works proposes "learning in the wild" problems involving a mixture of non-trivial settings. In Table 4.1 we provide a comparison between these approaches and FUSION, highlighting the features of each one. FUSION is the only approach that can deal with a complex setting involving few-shot learning, continual learning and unbalanced tasks, while also proving capable of generalizing across datasets.

As reported in Figure 4.1, FUSION is composed of four phases: embedding learning, clustering, meta-continual train and meta-continual test. In the embedding learning phase, a neural network is trained to generate embeddings that facilitate the subsequent separation. Embeddings can be learned in different ways, through generative models [8, 28] or self-supervised learning [14, 5]. Then clustering is applied to these embeddings, and each cluster corresponds to a task (i.e., a class) for the following phase. As clustering is not constrained to produce balanced clusters, the resulting tasks are also unbalanced. For the meta-continual training phase, we introduce a novel meta-learning-based algorithm that can effectively cope with unbalanced tasks. The algorithm, named MEML (for

*The code is available at https://github.com/alessiabertugli/FUSION

Meta-Example Meta-Learning), relies on a single inner loop update performed on an aggregated attentive representation, which we call meta-example. In so doing, MEML learns meta-representations that enrich the general features provided by large clusters with the variability given by small clusters, while existing approaches simply discard small clusters [44] or force the network to generate balanced clusters [5]. Finally, on the meta-continual test, the learned representation is frozen and novel tasks are learned acting only on classifications layers.
We perform extensive experiments on two few-shot datasets, Omniglot [59] and Mini-ImageNet [26] and on two continual learning benchmarks, Sequential MNIST [60] and Sequential CIFAR-10 [58] widely outperforming state-of-the-art methods. We also show the generalisation capability of FUSION across datasets.

**Contributions.** We remark on our contributions as follows:

- We propose FUSION, a novel strategy dealing with unbalanced tasks in an unsupervised meta-continual learning scenario;
- As part of FUSION, we introduce MEML, a new meta-learning-based algorithm that can effectively cope with unbalanced tasks, and MEMLX, a variant of MEML exploiting an original augmentation technique to increase robustness, especially when dealing with undersized datasets;
- We test FUSION on an unsupervised meta-continual learning setting reaching superior performance compared to state-of-the-art approaches. Ablations studies empirically show that the imbalance in the task dimension does not negatively affect the performance, and no balancing technique is required;
- We additionally test MEML, our meta-continual learning method, in standard supervised continual learning, achieving better results with respect to specifically tailored solutions;
- Finally, we report experiments highlighting the generalisation capacity of FUSION across datasets.

## 4.1 Method

Meta-continual learning [47, 6] deals with the problem of allowing neural networks to learn from a stream of few, non-i.i.d. examples and quickly adapt to new tasks. It can be considered as a few-shot learning problem, where tasks are incrementally seen, one class after the others. Formally, we define a distribution of training classification tasks $p(\mathcal{T}) = \mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_i, ....$ During meta-continual training, the neural network sees all samples belonging to $\mathcal{T}_0$ first, then all samples

belonging to $\mathcal{T}_1$, and so on, without shuffling elements across tasks as in traditional deep learning settings. The network should be able to learn a general representation, capturing important features across tasks, without catastrophic forgetting, meaning to overfit on the last seen tasks. During the meta-test phase, a different distribution of unknown tasks $p(\mathcal{T}') = \mathcal{T}'_0, \mathcal{T}'_1, ..., \mathcal{T}'_i, ...$ is presented to the neural network again in an incremental way. The neural network, starting from the learned representation, should quickly learn to solve novel tasks.

In this thesis, differently from standard meta-continual learning, we focus on the case where no training labels are available and tasks have to be constructed in an unsupervised way, using pseudo-labels instead of the real labels in the meta-continual problem. To investigate how neural networks learn when dealing with a real distribution and flow of unbalanced tasks, we propose FUSION, a novel learning strategy composed of four phases reported in Algorithm 2 and described in detail below.

### 4.1.1 Embedding learning

Rather than requiring the task construction phase to directly work on high dimensional raw data, an embedding learning network, which is different from the one employed in the following phases, is used to determine an embedding that facilitates the subsequent task construction. Through an unsupervised training [14, 8], the embedding learning network produces an embedding vector set $Z = Z_0, Z_1, ..., Z_N$, starting from the N data points in the training set $D = X_0, X_1, ..., X_N$ (see Figure 4.1.1). Embeddings can be learned in different ways, through generative models [8, 28] or self-supervised learning [14, 5]. In particular, DeepCluster [14] is a clustering method that jointly learns the parameters of a neural network and the cluster assignments of the resulting features. It groups the features with standard k-means clustering and uses these assignments as labels for a supervised learning procedure. ACAI [8] is an autoencoder-based architecture that aims to encourage high-quality interpolations into the latent space, extracting the most salient features from the dataset (without labels) and performing a sort of dimensionality reduction. The model is based on two encoders, one decoder and one critic. Here, the critic tries to predict the interpolation coefficient corresponding to an interpolated data point, while the autoencoder is trained to fool the critic into outputting a zero coefficient. In Figure 4.1.1 an illustration of an unsupervised embedding learning based on self-supervised learning is shown.

### 4.1.2 Clustering

As done in [44], the task construction phase exploits the k-means algorithm over suitable embeddings obtained with the embedding learning phase previously described. This simple but effective method assigns the same pseudo-label to all data points belonging to the same cluster. This way, a distribution $p(\mathcal{T}) = \mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_i, ...$ of tasks is built from the generated clusters as reported in Figure 4.1.2. Applying k-means over these embeddings leads to unbalanced clusters, which determine unbalanced tasks. This is in contrast with typical meta-learning and continual learning problems, where data are perfectly balanced. To recover a balanced setting, in [44], the authors set a threshold on the cluster dimension, discarding extra samples and smaller clusters. A recent alternative [5] forces the network to balance clusters, imposing a partition of the embedding space that could contrast with the extracted features. We believe that these approaches are sub-optimal as they alter the data distribution. In an unsupervised setting, where data points are grouped based on the similarity of their features, variability is an essential factor. In a task imbalanced setting, the obtained meta-representation is influenced by both small and large clusters. Large clusters allow learning robust features that are broadly useful, while small clusters inject a variability that can help with fine-grained discrimination.

### 4.1.3 Meta-continual train

**Motivation.** The adopted training protocol is related to the way data are provided at meta-test train time. In that phase, the model receives as input a stream of new unseen tasks, each with correlated samples; we do not assume access to other classes (as opposed to the training phase) and only the current one is available. In this respect, since the network's finetuning occurs with this stream of data, during training we reproduce a comparable scenario. In particular, we need to design a training strategy that is sample efficient and directly optimize for a proper initial weights configuration. These suitable weights allow the network to work well on novel tasks after a few gradient steps using only a few samples. In the context of meta-learning, MAML relies on a two-loop training procedure performed on a batch of training tasks. The inner loop completes $N$ step of gradient updates on a portion of samples of the training tasks, while the outer loop exploits the remaining ones to optimize for a quickly adaptable representation (meta-objective). Recent investigations on this algorithm explain that the real reason for MAML's success resides in feature reuse instead of rapid learning [80], proving that learn-

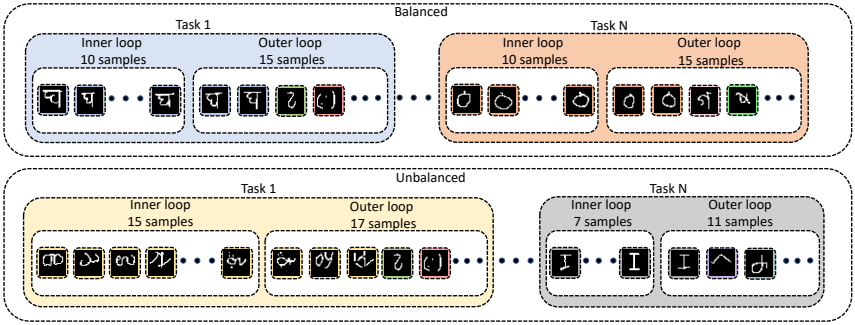



Figure 4.2: Balanced vs unbalanced tasks flow. In the balanced version, tasks contain a fixed number of elements for the inner loop (10 samples) and outer loop (15 samples, 5 from the current cluster and 10 randomly sampled from other clusters). In the unsupervised model, tasks are unbalanced and contain two-thirds of cluster data for the inner loop and one-third for the outer loop in addition to a fixed number of random samples.

ing meaningful representations is a crucial factor. Based on this assumption, we focus on the generalisation ability of the feature extraction layers to develop our meta-continual learning algorithm.

**Procedure.** The created tasks are sampled one at a time $\mathcal{T}_i \sim p(\mathcal{T})$ for the unsupervised meta-continual training phase as shown in Figure 4.1.3. The training process happens in a class-incremental way - where one task corresponds to one cluster - following a two-loop update procedure. The inner loop involves samples belonging to the ongoing task, while the outer loop contains elements sampled from both the current and other random clusters. In fact, during this stage, the network may suffer from the catastrophic forgetting effect on the learned representation if no technique is used to generalise or remember. To this end, the query set, used to update parameters in the outer loop, have to be designed to simulate an iid distribution, containing elements belonging to different tasks. We illustrate the tasks flow in Figure 4.2. The unbalanced case takes two-thirds of the current cluster data for the inner loop and adds one-third to a fixed number of random samples for the outer loop. The balanced case - usually adopted with supervised data - instead takes the same number of samples among tasks for both the inner and the outer loop.

To deal with the meta-continual train in FUSION (Figure 4.1.3), we propose MEML, a meta-learning procedure based on the construction of a meta-example,

a prototype of the task obtained through self-attention. The whole architecture is composed of a Feature Extraction Network (FEN), an aggregation module and a CLassification Network (CLN). The FEN is updated only in the outer loop (highlighted in blue in the figure), while frozen during the inner (grey). Both the aggregation module and the CLN are renewed in the inner and outer loops.

**MEML.** We remove the need for several inner loops, maintaining a single inner loop update through a mechanism for aggregating examples based on self-attention. This way, we considerably reduce the training time and computational resources needed for training the model and increase global performance. The use of a meta-example instead of a trajectory of samples is particularly helpful in class-incremental continual learning to avoid catastrophic forgetting. In fact, instead of sequentially processing multiple examples of the same class and updating the parameters at each one (or at each batch), the network does it only once per class, reducing the forgetting effect. At each time-step, as pointed out in Figure 4.2, a task $\mathcal{T}_i = (\mathcal{S}_{cluster}, \mathcal{S}_{query})$ is randomly sampled from the task distribution $p(\mathcal{T})$. $\mathcal{S}_{cluster}$ contains elements of the same cluster:

$$\mathcal{S}_{cluster} = \{(X_k, Y_k)\}_{k=0}^{K}, \tag{4.1}$$

where $Y_{cluster} = Y_0 = ... = Y_k$ is the cluster pseudo-label and $K$ is the number of data points in the cluster. Instead, $\mathcal{S}_{query}$ contains a variable number of elements belonging to the current cluster and a fixed number of elements randomly sampled from all other clusters:

$$\mathcal{S}_{query} = \{(X_q, Y_q)\}_{q=0}^{Q}, \tag{4.2}$$

where $Q$ is the total number of elements in the query set. $\mathcal{S}_{cluster}$ is used for the inner loop update, while $\mathcal{S}_{query}$ is used to optimise the meta-objective during the outer loop. All the elements belonging to $\mathcal{S}_{cluster}$ are processed by the FEN, parameterised by $\theta$, computing the feature vectors $R_0, R_1, ..., R_K$ in parallel for all task elements:

$$R_{0:K} = f_\theta(X_{0:K}). \tag{4.3}$$

The obtained embeddings are refined with an attention function, parameterised by $\rho$, that computes the attention coefficients $\vec{a}$ from the features vectors:

$$\vec{a} = \textit{Softmax}[f_\rho(R_{0:K})]. \tag{4.4}$$

Then, the final aggregated representation vector $R_{ME}$, for *meta-example* representation, captures the most salient features:

$$R_{ME} = \vec{a}^\intercal R_{0:K}. \tag{4.5}$$

The single inner loop is performed on this meta-example, which adds up the weighted-features contribution of each element of the current cluster. Then, the cross-entropy loss $\mathcal{L}$ between the predicted label and the pseudo-label is computed and both the classification network parameters $W$ and the attention parameters $\rho$ are updated with a gradient descent step:

$$\psi \leftarrow \psi - \alpha \nabla_{\psi} \mathcal{L}(f_{\psi}(R_{ME}), Y_{cluster}), \tag{4.6}$$

where $\psi = \{W, \rho\}$ and $\alpha$ is the inner loop learning rate. Finally, to update the whole network parameters $\phi = \{\theta, W, \rho\}$, and to ensure generalisation across tasks, the outer loop loss is computed from $S_{query}$. The outer loop parameters are thus updated as follows:

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} \mathcal{L}(f_{\phi}(X_{0:Q}), Y_{0:Q}), \tag{4.7}$$

where $\beta$ is the outer loop learning rate.
Note that with the aggregation mechanism introduced by MEML, a single inner loop is made regardless of the number of examples in the cluster, thus eliminating the problem of unbalancing at the inner loop level. However, the representation updated in the outer loop remains unbalanced since the FEN calculate the embeddings for each example.

**MEMLX.** Since the aim is to learn a representation that generalises to unseen classes, we introduce an original augmentation technique inspired by [36]. The idea is to generate multiple sets of augmented input data and retain the set with maximal loss to be used as training data. Minimising the average risk of this worst-case augmented data set enforces robustness and acts as a regularisation against random perturbations, leading to a boost in the generalisation capability. Starting from the previously defined $\mathcal{S}_{cluster}$ and $\mathcal{S}_{query}$ we generate $m$ sets of augmented data:

$$\{\mathcal{S}^i_{cluster}, S^i_{query}\}_{i=1}^{m} \leftarrow A(\mathcal{S}_{cluster}), A(\mathcal{S}_{query}), \tag{4.8}$$

where $A$ is an augmentation strategy that executes a combination of different data transformations for each $i \in m$. Hence, for each of these newly generated sets of data we perform an evaluation forward pass through the network and compute the loss, retaining the $S^{i_c}_{cluster}$ and $\mathcal{S}^{i_q}_{query}$ sets giving the highest loss to be used as input to MEML for the training step:

$$\begin{aligned} i_c &= \text{argmax}_{i \in 1,..m} \mathcal{L}(f(\mathcal{S}^i_{cluster}), Y_{cluster}), \\ i_q &= \text{argmax}_{i \in 1,..m} \mathcal{L}(f(\mathcal{S}^i_{query}), Y_{0:Q}). \end{aligned} \tag{4.9}$$


Figure 4.3: Augmentation technique adopted in MEMLX.

**Algorithm 2** FUSION

**Require:** : $D = X_0, X_1, ..., X_N$: unlabeled training set
**Require:** $\alpha$, $\beta$: inner loop and outer loop learning rates
1: Run embedding learning on $D$ producing $Z_{0:N}$ from $X_{0:N}$
2: Run $k$-means on $Z_{0:N}$ generating a distribution of unbalanced tasks $p(\mathcal{T})$ from clusters
3: Meta-train with MEML in Algorithm 3
4: Meta-test with MEML

In Figure 4.3, we illustrate the overall augmentation process employed in MEMLX. Three different augmented batches are created starting from the input batch, each forwarded through the network producing logits. The Cross-Entropy losses between those latter and the targets are computed, keeping the augmented batch corresponding to the highest value. In detail, we adopt the following augmentation:

- **Augmented batch 1**: vertical flip, horizontal flip;
- **Augmented batch 2**: colour jitter (brightness, contrast, saturation, hue);
- **Augmented batch 3**: random affine, random crop.

We report the whole algorithm in Algorithm 3, where MEMLX steps are highlighted in blue.

### 4.1.4 Meta-continual test

At meta-continual test time, novel and unseen tasks $\mathcal{T}_i' \sim p(\mathcal{T}')$ from the test set are provided to the network, as illustrated in Figure 4.1.4. Here $p(\mathcal{T}')$ represents the distribution of supervised test tasks and $\mathcal{T}_i'$ corresponds to a sampled test class. The representation learned during the meta-train remains frozen, and only the prediction layers are fine-tuned. The test set is composed of novel tasks, that

**Algorithm 3** MEML

**Require:** : $D = X_0, X_1, ..., X_N$: unlabelled training set
**Require:** $\alpha, \beta$: inner loop and outer loop learning rates
Randomly initialise $\theta$ and $W$
**while** not done **do**
  Sample a task $\mathcal{T}_i = (\mathcal{S}_{cluster}, \mathcal{S}_{query}) \sim p(\mathcal{T})$
  Randomly initialise $W_i$
  **if** MEMLX **then**
    $\{\mathcal{S}^i_{cluster}, \mathcal{S}^i_{query}\}^m_{i=1} \leftarrow A(\mathcal{S}_{cluster}), A(\mathcal{S}_{query})$ 4.8
    $i_c = \text{argmax}_{i \in 1,..m} \mathcal{L}(f(\mathcal{S}^i_{cluster}), Y_{cluster})$
    $i_q = \text{argmax}_{i \in 1,..m} \mathcal{L}(f(\mathcal{S}^i_{query}), Y_{0:Q})$ 4.9
    $\mathcal{S}_{cluster} = \mathcal{S}^{ic}_{cluster}, \mathcal{S}_{query} = \mathcal{S}^{iq}_{query}$
  **end if**
  $R_{0:K} = f_\theta(X_{0:K})$ 4.3
  $\vec{a} = Softmax[f_\rho(R_{0:K})]$ 4.4
  $R_{ME} = \vec{a}^\intercal R_{0:K}$ 4.5
  $\psi, \phi = \{W, \rho\}, \{\theta, W, \rho\}$
  $\psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}(f_\psi(R_{ME}), Y_{cluster})$ 4.6
  $\phi \leftarrow \phi - \beta \nabla_\phi \mathcal{L}(f_\phi(X_{0:Q}), Y_{0:Q})$ 4.7
**end while**

can be part of the same distribution (e.g. distinct classes within the same dataset) or even belong to a different distribution (e.g. training and testing performed on different datasets).

## 4.2 Experiments

### 4.2.1 Few-shot Unsupervised Continual Learning

#### 4.2.1.1 Datasets

We employ Omniglot and Mini-ImageNet, two datasets typically used for few-shot learning evaluation. The Omniglot dataset contains 1623 characters from 50 different alphabets with 20 greyscale image samples per class. We use the same splits as [44], using 1100 characters for meta-training, 100 for meta-validation, and 423 for meta-testing. The Mini-ImageNet dataset consists of 100 classes of realistic RGB images with 600 examples per class. As done in [84, 44], we use 64 classes for meta-training, 16 for meta-validation and 20 for meta-test.

#### 4.2.1.2 Architecture

Following [47], we use for the FEN a six-layer CNN interleaved by ReLU activations with 256 filters for Omniglot and 64 for Mini-ImageNet. All convolutional

layers have a $3 \times 3$ kernel (for Omniglot, the last one is a $1 \times 1$ kernel) and are followed by two linear layers constituting the CLN. The attention mechanism is implemented with two additional linear layers interleaved by a Tanh function, followed by a Softmax and a sum to compute attention coefficients and aggregate features. We use the same architecture for competitive methods. We do not apply the Softmax activation and the final aggregation but we keep the added linear layers, obtaining the same number of parameters. The choice in using two simple linear layers as an attention mechanism is made specifically since the aim of the thesis is to highlight how this kind of mechanism can enhance performance and significantly decrease both training time and memory usage. Investigating the behaviour of more sophisticated attention mechanisms is beyond the scope of this work.

#### 4.2.1.3 Training protocol

For Omniglot, we train the model for $60000$ steps while for Mini-ImageNet for $200000$, with meta-batch size equal to $1$. The outer loop learning rate is set to $1e^{-4}$ while the inner loop learning rate is set to $0.1$ for Omniglot and $0.01$ for Mini-ImageNet, with Adam optimiser. As embedding learning networks, we employ Deep Cluster [14] for Mini-ImageNet and ACAI [8] for Omniglot. Since Mini-ImageNet contains 600 examples per class, after clustering, we sample examples between 10 and 30, proportionally to the cluster dimension to keep the imbalance between tasks. We report the test accuracy on a different number of unseen classes, which induces increasingly complex problems as the number increase. Following the protocol employed in [47], all results are obtained through the mean of $50$ runs for Omniglot and $5$ for Mini-ImageNet.

#### 4.2.1.4 Performance analysis

In Table 4.2 and Table 4.3, we report results respectively on Omniglot and Mini-ImageNet, comparing our model with competing methods. To see how the performance of MEML within our FUSION is far from those achievable with the real labels, we also report for all datasets the accuracy reached in a supervised case (*oracles*). We define Oracle OML [47] and Oracle ANML [6] [†] as super-

[†]Our results on Oracle ANML are different from the ones presented in the original paper due to a different use of data. To make a fair comparison we use 10 samples for the support set and 15 for the query set for all models, while in the original ANML paper the authors use 20 samples for the support set and 64 for the query set. We do not test ANML on Mini-ImageNet due to the high computational resources needed.

| Algorithm/Tasks | Omniglot 10 | 50 | 75 | 100 | 150 | 200 |
|---|---|---|---|---|---|---|
| Oracle OML [47] | 88.4 | 74.0 | 69.8 | 57.4 | 51.6 | 47.9 |
| Oracle ANML [6] | 86.9 | 63.0 | 60.3 | 56.5 | 45.4 | 37.1 |
| Oracle MEML (Ours) | **94.2** | **81.3** | **80.0** | **76.5** | **68.8** | **66.6** |
| Oracle MEMLX (Ours) | **94.2** | 75.2 | 75.0 | 67.2 | 58.9 | 55.4 |
| OML | 74.6 | 32.5 | 30.6 | 25.8 | 19.9 | 16.1 |
| ANML | 72.2 | 46.5 | 43.7 | 37.9 | 26.5 | 20.8 |
| MEML (Ours) | **89.0** | 48.9 | 46.6 | 37.0 | 29.3 | 25.9 |
| MEMLX (Ours) | 82.8 | **50.6** | **49.8** | **42.0** | **34.9** | **31.0** |

Table 4.2: Meta-test test accuracy on Omniglot.

vised competitors, and Oracle MEML is the supervised version of our model. MEML outperforms OML on Omniglot and Mini-ImageNet and ANML on Omniglot, suggesting that the meta-examples strategy is beneficial on both FUSION and fully supervised cases. MEMLX, the advanced version exploiting a specific augmentation technique is able to improve the MEML results in almost all experiments. In particular, MEMLX outperforms MEML on both Omniglot and Mini-ImageNet in FUSION and even in the fully supervised case on Mini-ImageNet. The only experiment in which MEML outperforms MEMLX is on Omniglot, in the supervised case. In our opinion, the reason is to be found in the type of dataset. Omniglot is a dataset made up of 1100 classes and therefore characters are sometimes very similar to each other. Precisely for this reason, applying augmentation can lead the network to confuse augmented characters for a class with characters belonging to other classes. In the unsupervised case, the clusters are grouped by features, which should better separate the data from a visual point of view, thus favouring our augmentation technique.

**Time and Computation Analysis.** In Table 4.4, we compare training time and computational resources usage between OML, MEML and MEMLX on Omniglot and Mini-ImageNet. We measure the time to complete all training steps in hours and minutes and the computational resources in gigabytes occupied on the GPU. Both datasets confirm that our methods, adopting a single inner update, train considerably faster and use approximately one-third of the GPU resources with respect to OML. MEMLX undergoes minimal slowdown, despite the use of

| Algorithm/Tasks | Mini-ImageNet 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| Oracle OML [47] | 50.0 | 31.9 | 27.0 | 16.7 | 13.9 |
| Oracle MEML (Ours) | 66.0 | 33.0 | 28.0 | 29.1 | 21.1 |
| Oracle MEMLX (Ours) | **74.0** | **60.0** | **36.7** | **51.3** | **40.1** |
| OML | 49.3 | 41.0 | 19.2 | 18.2 | 12.0 |
| MEML (Ours) | 70.0 | **48.4** | 36.0 | 34.0 | 21.6 |
| MEMLX (Ours) | **72.0** | 45.0 | **50.0** | **45.6** | **29.9** |

Table 4.3: Meta-test test accuracy on Mini-ImageNet.

| Algorithm | Omniglot Time | Omniglot GPU | Mini-ImageNet Time | Mini-ImageNet GPU |
|---|---|---|---|---|
| OML [47] | 1h 32m | 2.239 GB | 7h 44m | 3.111 GB |
| MEML | 47m | 0.743 GB | 3h 58m | 1.147 GB |
| MEMLX | 1h 1m | 0.737 GB | 4h 52m | 1.149 GB |

Table 4.4: Training time and GPU usage of MEML and MEMLX compared to OML on Omniglot and Mini-ImageNet.

our augmentation strategy. To a fair comparison, all tests are performed on the same hardware equipped with an NVIDIA Titan X GPU.

### 4.2.2 Supervised continual learning

To further prove the effectiveness of our meta-example strategy, we put MEML and MEMLX in standard supervised continual learning and show their performance compared to state-of-the-art continual learning approaches.

#### 4.2.2.1 Datasets

We experiment on Sequential MNIST and Sequential CIFAR-10. In detail, the MNIST classification benchmark [60] and the CIFAR-10 dataset [58] are split into 5 subsets of consecutive classes composed of 2 classes each.

#### 4.2.2.2 Architecture

For tests on Sequential MNIST, we employ as architecture a fully-connected network with two hidden layers, interleaving with ReLU activation as proposed in [73], [86]. For tests on CIFAR-10, following [85], we rely on ResNet18 [39].

#### 4.2.2.3 Training protocol

We train all models in a class-incremental way (Class-IL), the hardest scenario among the three described in [105], which does not provide task identities. We train for 1 epoch for Sequential MNIST and 50 epochs for Sequential CIFAR-10. SGD optimiser is used for all methods for a fair comparison. A grid search of hyperparameters is performed on all models taking the best ones for each (for further details see the supplementary material). For rehearsal-based strategies, we report results on buffer sizes 200, 500 and 5120. The standard continual learning test protocol is used for all methods, where the accuracy is measured on test data composed of unseen samples of all training tasks at the end of the whole training process. We adapt our meta-example strategy to a double class per task making a meta-example for each class corresponding to two inner loops. The query set used within FUSION mirror the memory buffer in continual learning. The memory buffer contains elements from previously seen tasks, while the query set samples elements from all training tasks. For MEMLX, we apply our augmentation technique on both current task data and buffer data.

#### 4.2.2.4 Performance analysis

In Table 4.5 we show accuracy results on Sequential MNIST and Sequential CIFAR-10 [‡] respectively. MEML and MEMLX consistently overcome all state-of-the-art methods on both datasets. We denote that MEML is significantly different from MER, which adopts a Reptile [78] update style, processing one sample at a time and making an inner loop on all samples. This greatly increases the training time, making this strategy ineffective for datasets such as CIFAR-10. On the contrary, MEML makes as many inner loops as there are classes per task and finally a single outer loop on both task data and buffer data. This way, MEML training time is comparable to the other rehearsal strategy, but with the generalisation benefit given from meta-learning. To further confirm the beneficial role of the meta-learning procedure, we observe that EXP REPLAY, using only one loop,

[‡]Due to high training time we do not report MER results on Sequential CIFAR-10.

| Algorithm/Buffer | Sequential MNIST | | | | Sequential CIFAR-10 | | | |
|---|---|---|---|---|---|---|---|---|
| | None | 200 | 500 | 5120 | None | 200 | 500 | 5120 |
| LWF [65] | 19.62 | - | - | - | 19.60 | - | - | - |
| EWC [55] | 20.07 | - | - | - | 19.52 | - | - | - |
| SI [119] | 20.28 | - | - | - | 19.49 | - | - | - |
| SAM [35] | 62.63 | - | - | - | - | - | - | - |
| iCARL [85] | - | - | - | - | - | 51.04 | 49.08 | 53.77 |
| HAL [18] | - | 79.80 | 86.80 | 88.68 | - | 32.72 | 46.24 | 66.26 |
| GEM [73] | - | 78.85 | 85.86 | 95.72 | - | 28.91 | 23.81 | 25.26 |
| EXP REPLAY [89] | - | 78.23 | 88.67 | 94.52 | - | 47.88 | 59.01 | 83.65 |
| MER [86] | - | 79.90 | 88.38 | 94.58 | - | - | - | - |
| MEML (Ours) | - | 84.63 | 90.85 | **96.04** | - | **54.33** | **66.41** | 83.91 |
| MEMLX (Ours) | - | **89.94** | **92.11** | 94.88 | - | 51.98 | 63.25 | **83.95** |

Table 4.5: MEML and MEMLX compared to state-of-the-art continual learning methods on Sequential MNIST (left) and Sequential CIFAR-10 (right) in class-incremental learning.

reaches lower performance. In Table 6 we report results on additional continual learning metrics: forward transfer, backward transfer and forgetting. In particular, **forward transfer** (FWT) measure the capability of the model to improve on unseen tasks with respect to a random-initialized network. It is computed by making the difference between the accuracy before the training on each task and the accuracy of a random-initialized network averaged on all tasks. **Backward transfer** [73] (BWT) is computed making the difference between the current accuracy and its best value for each task, making the assumption that the highest value of the accuracy on a task is the accuracy at the end of it. Finally, **forgetting** [16] is similar to BTW, without the letter assumption. We compare the best performer algorithms on both Sequential MNIST and Sequential CIFAR. MEML and MEMLX outperform all the other methods on backward transfer and forgetting, while a little lower performance is reached on forward transfer. Since results are consistent for all buffer dimensions, we report results on buffer $5120$. The results on buffers $200$ and $500$ are reported in the supplementary material.

#### 4.2.2.5 Time analysis

We make a training time analysis between the most relevant state-of-the-art continual learning strategy on Sequential MNIST. We measure the training time in

| Algorithm/Metric | Sequential MNIST | | | Sequential CIFAR-10 | | |
|---|---|---|---|---|---|---|
| | FWT | BWT | Forgetting | FWT | BWT | Forgetting |
| HAL [18] | -10.06 | -6.55 | 6.55 | -10.34 | -27.19 | 27.19 |
| GEM [73] | **-9.51** | -4.14 | 4.30 | -9.18 | -75.27 | 75.27 |
| EXP REPLAY [89] | -10.97 | -6.07 | 6.08 | **-8.45** | -13.99 | 13.99 |
| MER [86] | -10.50 | -3.22 | 3.22 | - | - | - |
| MEML (Ours) | -9.74 | -3.12 | 3.12 | -12.68 | **-10.97** | **10.97** |
| MEMLX (Ours) | -9.74 | **-1.72** | **1.92** | -12.74 | -12.42 | 12.42 |

Table 4.6: Forward transfer, backward transfer and forgetting comparison on Sequential MNIST (left) and Sequential CIFAR-10 (right) in class-incremental learning.

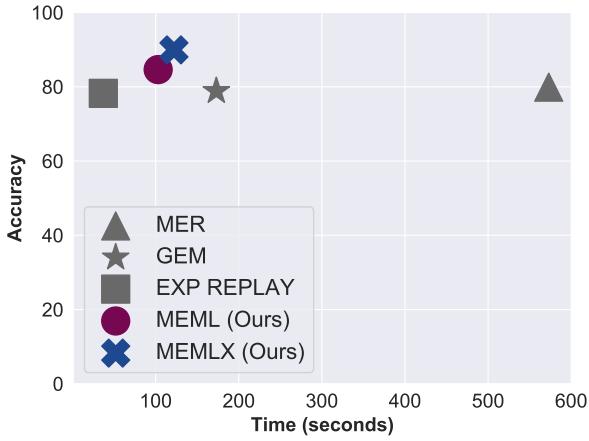



Figure 4.4: Training time comparison with respect to the accuracy between the most important state-of-the-art continual learning methods.

seconds since the last task. We find that MEML and MEMLX are slower only compared to EXP REPLAY due to the meta-learning strategy, but they are faster with respect to both GEM and MER, reaching higher accuracy.

### 4.2.3 Ablation experiments

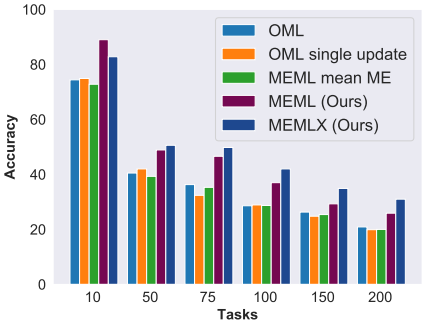



Figure 4.5: Experiments showing the capability of meta-example on Omniglot.

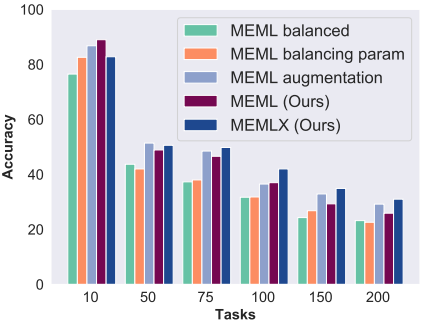



Figure 4.6: Comparison between unbalanced and balanced settings on Omniglot.

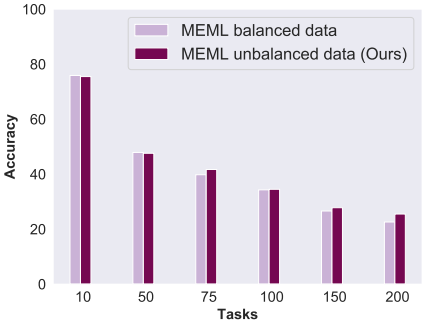



Figure 4.7: Comparison between unbalanced and balanced data on Omniglot.

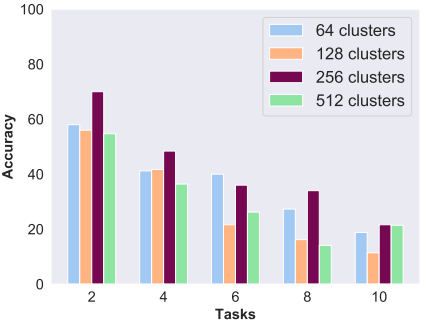



Figure 4.8: Accuracy with different numbers of clusters on Mini-ImageNet.

**Meta-Example Single Update *vs.* Multiple Updates.** To prove the effectiveness of our method - MEML - based on meta-examples, we compare it with:

- OML [47] - performing multiple updates, one for each element of the cluster;
- OML with a single update - adopting a single update over a randomly sampled data point from each task;
- MEML with mean ME - a version exploiting the mean between the feature vector computed by the FEN.

In Figure 4.5, we show that MEML and MEMLX consistently outperform all the other baselines on Omniglot. OML with a single update gives analogous performance to the multiple updates, confirming the idea that the strength of generalisation relies on feature reuse. Also, the MEML with mean ME has performance comparable with the multiple and single update ones, proving the effectiveness of our aggregation mechanism to determine a suitable and general embedding vector for the CLN.

**Balanced *vs.* Unbalanced Tasks.** To justify the use of unbalanced tasks and show that allowing unbalanced clusters is more beneficial than enforcing fewer balanced ones, we present some comparisons in Figure 4.6. First of all, we introduce a baseline in which the number of clusters is set to the true number of classes, removing from the task distribution the ones containing less than $N$ elements and sampling $N$ elements from the bigger ones. We thus obtain a perfectly balanced training set at the cost of less variety within the clusters; however, this leads to poor performance as small clusters are never represented. To verify if maintaining variety and balancing data can lead to better performance, we try two balancing strategies: augmentation, at the data level, and a balancing parameter, at the model level. For the first one, we keep all clusters, sampling $N$ elements from the bigger and using data augmentation for the smaller to reach $N$ elements. At the model level, we multiply the loss term by a balancing parameter to weigh the update for each task based on cluster length. These tests result in lower performance with respect to MEML and MEMLX, suggesting that the only thing that matters is cluster variety and unbalancing does not negatively affect the training. However, this ablation experiment is not complete. In fact, these good results of the unbalanced version could be due to the fact that the original data distribution is balanced. The unbalancing of the tasks is due only to the clustering procedure, but there is not real unbalancing in the feature space. With this aim, we verify the behaviour of our model with the imbalance in the data distribution in Figure 4.7. We sample a balanced and reduced version of Omniglot with $15$ images per class and another unbalanced version sampling randomly $15 \times 1100$ data

| Algorithm/Ways | Shots | 5,1 | 5,5 | 20,1 | 20,5 |
|---|---|---|---|---|---|
| CACTUs-MAML, 20 | Balanced, 5 | 60.50 | 84.00 | 40.50 | 67.62 |
| | Unbalanced, 5 | **62.50** | **85.50** | **42.62** | **71.87** |
| | Balanced, 15 | 67.00 | 86.00 | 32.50 | 64.62 |
| | Unbalanced, 15 | **72.00** | **89.00** | **40.00** | **66.25** |

Table 4.7: Meta-test accuracy on balanced *vs.* unbalanced CACTUs-MAML on Omniglot.

points from the whole distribution to generate two datasets with the same number of data points. For the unbalanced one, since we randomly sample data points from different classes the data resultant data distribution is truly unbalanced. To test different unbalanced distributions, we make 5 runs of the same experiment with random seeds and report the mean. Our results show that MEML with balanced data and MEML with unbalanced data distribution have similar performance. We further want to confirm that our intuition about unbalancing is valid in an unsupervised meta-learning model without incremental training. We perform the balanced/unbalanced experiments also on an alternative method named CACTUs [44]. We compare their standard methodology using balanced clusters as input with another exploiting unbalanced data. In detail, they obtain the balanced version discarding the smaller clusters. The results are shown in Table 4.7 and attest that the model trained on unbalanced data outperforms the balanced one, further proving the importance of task variance to better generalise to new classes at meta-test time. We report the results of training the algorithms on 20 ways and 5 shots and 15 shots, to have enough data points per class to create the imbalance.

**Number of Clusters.** In an unsupervised setting, the number of original classes could be unknown. Consequently, it is important to assess the performance of our model by varying the number of clusters at meta-train time. In Figure 4.8 we report the results on Mini-ImageNet because it is particularly suitable for this type of test, as it has few classes and many examples per class [§]. We test our method with 64, 128, 256, and 512 clusters. We report the mean of 3 runs each with a random seed that determines random number generation for k-means centroid initialization. We obtain the best results with 256 clusters, suggesting

[§]We do not report the same test on Omniglot because it has only 20 examples per class and by increasing the number of classes compared to the original ones we would have classes with very few examples and we would not be able to generate a task containing a sufficient number of examples for training.

that with complex datasets, the number of clusters is crucial to reach higher meta-test performance and to find general features. To corroborate the above findings, we observe that using $512$ clusters degrades performance with respect to the $256$ case, proving that tasks constructed over an embedding space with too specific features fail to generalise. Using a lower number of clusters, such as $64$ or $128$, also achieves worse performance.

### 4.2.4 Generalisation across datasets

We investigate how our model behaves in few-shot continual learning, where the data distribution shift from the meta-continual train to the meta-continual test. Since our model is unsupervised, FEN training is based only on feature embeddings, with no class-dependent bias. This way, our model could generalise across datasets, where the training tasks belong to a different data distribution (i.e., a different dataset) with respect to the test tasks.

#### 4.2.4.1 Datasets

To investigate this conjecture, we test our model on Cub [112] and CIFAR-100 [58] datasets following the split used in [61, 103]. In detail, on Cub, we retain a total of 40 classes, while on Cifar, we keep 10 classes, both with 20 samples each. On both datasets, we exploit 15 samples for the meta-test training phase and the remaining 5 for the meta-test testing, on which the accuracy is calculated.

#### 4.2.4.2 Architecture

We use the same architecture of MEML experiments within FUSION.

#### 4.2.4.3 Training protocol

We perform the meta-continual train of MEML and Oracle MEML on the training set of Mini-ImageNet, while the meta-test is done on the test set of CIFAR-100. Likewise, we use models obtained through the meta-continual train on Omniglot to test on Cub. Oracle OML CIFAR-100 and Oracle OML Cub indicate the supervised OML model trained on the same dataset we test. MEML Mini-ImageNet and Oracle MEML Mini-ImageNet indicate our unsupervised model and the corresponding supervised version trained on Mini-ImageNet (the same notation is used for Omniglot).

| Algorithm/Tasks | CIFAR-100 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| Oracle OML [47] CIFAR-100 | 66.0 | 45.0 | 34.0 | 30.0 | 29.5 |
| Oracle MEML Mini-ImageNet | **70.0** | **56.0** | **46.7** | 45.3 | **35.6** |
| MEML Mini-ImageNet | 64.0 | 42.0 | 43.3 | **49.4** | 31.8 |

Table 4.8: Meta-test test accuracy on CIFAR-100.

| Algorithm/Tasks | Cub 2 | 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|
| Oracle OML [47] Cub | 50.0 | 13.9 | 25.8 | 4.5 | 8.9 |
| Oracle MEML Omniglot | 44.0 | 49.1 | 32.7 | **27.0** | 25.1 |
| MEML Omniglot | **66.0** | **53.3** | **28.3** | 26.2 | **25.6** |

Table 4.9: Meta-test test accuracy on Cub.

#### 4.2.4.4 Performance analysis

Results in Table 4.8 show that by training on Mini-ImageNet and testing on CIFAR-100 Oracle MEML outperforms Oracle OML trained on CIFAR-100 suggesting that learning a good representation can be favourable even with respect to training on the same data distribution. In Table 4.9 MEML generally outperforms the supervised Oracle MEML. In the latter case, it also outperforms the supervised Oracle OML trained on Cub, which is incapable of learning a meaningful representation.

# Chapter 5

# Continual Learning Techniques for Financial Market Predictions

The financial market is a worldwide virtual place meant for the exchange of financial instruments, such as shares, contracts, stocks, or commodities. In the past, investments were made in person by a small circle of domain experts, basing their decisions on experience and a small amount of data. Nowadays, the actors and the dynamics involved have radically changed. The strong availability of real-time data and the great computational capabilities of computers brought that most of the investments are not made only by human traders. Therefore, they are assisted by an ever-increasing number of equipped "intelligent machines" able to understand the best timing for carrying out financial transactions. Consequently, the concept of algorithmic trading has taken hold. The algorithms are characterized by a set of rules defined to perform certain actions based on the state of the market. These rules are written by human traders who study particular techniques for market choices. The natural evolution of these algorithms is the use of cutting-edge machine learning and deep learning techniques to develop predictive models. This way, human intervention is no longer required to define steady rules, and decision support tools can rely only on data and their patterns over time, through the use of tailored neural networks. The problem with using machine and deep learning techniques with time series is the need to periodically

retrain the model to allow for the updating and acquisition of knowledge of the most recent data. This leads neural networks to suffer from catastrophic forgetting, meaning that their weights are overwritten in favour of the last seen data, losing their predictive power over the older data. It has been shown that market trends over time are qualitatively similar, and therefore learning the behaviour of the time series in the past could be useful for predicting its future trend. The study and application of CL techniques could be relevant within the financial market, due to their cyclical nature.

Exploring the potential of continual learning is a novel topic in the financial world and therefore, to the best of our knowledge, this is the first work that analyzes and evaluates the potential of continual learning in-depth. In particular, we analyzed whether catastrophic forgetting could lead to a bad predictive performance in financial time series classification, comparing the classical online learning paradigm with continual learning techniques designed to overcome catastrophic forgetting. *

Specifically, a problem of binary classification of time series has been studied, where each time series consists of a vector of consecutive daily samplings. The goal is to predict whether this will have an increasing or decreasing trend in the future, leading to a "buy" or "sell" decision. We define continual learning tasks as periods, which may have different lengths, but specific behaviour. For example, the first task can be represented by a stationary price trend, the second one by a sudden increase in prices, and the third one by a slow drop. Each task is time bounded by a change in these regimes. Approaching the problem this way, we place it in a Domain-Incremental Learning (Domain-IL) scenario [105], where the distribution of the classes remains unchanged while the change of task is defined by a variation of the distribution of input data. Several state-of-the-art continual learning methods have been developed and deeply analyzed. Eventually, the experimental results achieved suggest that CL techniques, alleviating the forgetting phenomenon, exhibit better performance than online learning.

## 5.1 Method

A great advantage of trading using algorithms is the ability to process a large amount of data in real-time to extract as much information as possible. It is necessary to provide the model with data that best reflects the state of the market

*Code is available at https://github.com/albertozurli/cl_timeseries.

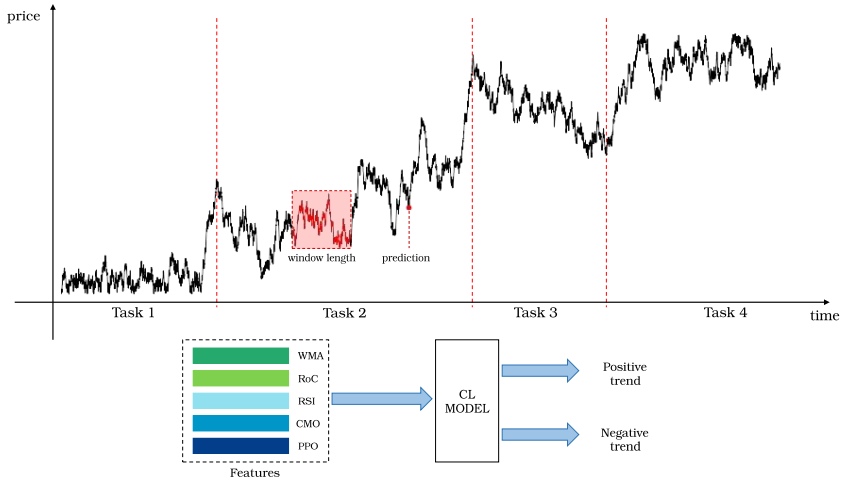



Figure 5.1: Data flow of a financial time series in our setting. The prices time series is subdivided into tasks by a change-point detector (dashed red lines). Raw prices, relative to a certain period (determined by a window length), are turned into financial indicators, that become the inputs of the neural network. The problem consists of a binary classification to predict prices trend (positive or negative) at N time step later.

at each time step to make the most appropriate investment choice. In this section, we present the data, their structure, and the features generated by the models.

### 5.1.1 Financial time series and indicators

The input provided to the model consists of fixed-length sequences representing the trend of financial time series (daily close values) over a period of one month (i.e. 20 business days) of observations. Sequences are constructed by taking all possible consecutive 20-day windows in the available data. Each series is used to obtain the features and time sequences by building a different dataset, giving rise to an analysis of multivariate historical series for each financial series. Even if a neural network can extrapolate the information it deems relevant in a completely autonomous way, using only time series does not carry out any learning despite copying the label of the previous example in the output. This fact demonstrates a dependence between time series even where they have no provable relation-

ships. Consequently, it was necessary to manipulate the series by carrying out various engineering operations to obtain features. The first step is carried out to eliminate copying between outputs, a phenomenon that is not uncommon in time series and which is usually solved by creating a series obtained as the difference between data points of two consecutive instants. The input has doubled its dimensionality from a vector to a matrix. The increase in dimensionality is an aspect to be taken into account in a problem of this type: if, on the one hand, only the raw series is insufficient, on the other hand, using too many features could push the model to focus on unnecessary or worse misleading information. This aspect was taken into account in choosing which financial indicators to use as features that could provide helpful information. Thanks to moving averages, we can catch the real market trends, removing irrelevant fluctuations in the original series. Using *Weighted Moving Average*, we put a specific weight which decades overtime to any timestep, giving a higher impact on more recent timesteps. *Rate of Change* provides a first qualitative analysis regarding possible change-points computing the percentual difference between the current timestep and another $n$ step in the past. A minor variation means that current data do not differ from previous ones, while a bigger one indicates a possible change in the distribution. To measure variation speed and magnitude in time series, we used *Relative Strength Index*. This momentum oscillator reports the strength of the current time series trend, highlighting periods of excessive overconfidence and underconfidence in stocks. *Chande Momentum Oscillator* provides more fine-grain values with respect to RSI, enabling the model to confirm or deny results of the previous indicator while at the same time finding out new change-points. Finally, we can keep track of market reversals with the *Percentage Price Oscillator*, another momentum oscillator able to compare moving averages with different temporal horizons. An exciting feature of the chosen indicators is that they collect information about the past and mediate it with the present. This allows for forming features that represent the current state and a set of past timesteps used to constitute the feature itself. Many indicators are characterized by an observation period over time in the series. Those listed below have been generated for $5$ to $20$ days to extract informative content both in the short and long term. Shorter periods could capture information that is too little specific for the sequence, just as long periods would provide information of a greater window than the observation window of the model itself.

### 5.1.2 Definition of domain regimes

The classic problems of continual learning do not deal with time series, but rather images. Consequently, it is necessary to understand which algorithms could be applied, taking into account that input data of a different nature lead to different assumptions. The first is the temporal aspect of the financial series, where the time series itself defines the order of tasks and examples. Namely, the only way to learn from these series is by respecting the temporal order, hence excluding data shuffling or any kind of offline learning. However, this assumption does not indicate a limitation in the study of the problem since the training of the tasks is sequential. Continual learning is defined for different scenarios (task, domain, and class-IL) and the next step was to identify in which of these the problem treated is found. Since all the tasks come from the same time series, the data distribution of each task will also be the same; therefore, we can discard the class-IL scenario. Considering the problem of predicting the market by classification, for each task, the model receives as input a sequence equal to one month of data updated every day at the close of the markets, for a sequence length equivalent to 20 days. The reason for this choice is dictated above all by the opening and closing of the world market. The second reason for choosing a window of this size is given by the assumption that in a month of daily observations it is possible to collect a sufficient amount of data to make a reliable analysis. There is also a label associated with each sequence. There will be $y = 1$ if the value of the time series 20 days after the end of the sequence will be greater than the last data of the same, zero otherwise. This corresponds to one month of observation and prediction for the following month. The labels predicted by the model indicate how to act: a positive prediction suggests an increase in market value, and therefore it is generally advisable to buy. In contrast, a negative prediction invites you to sell. For each task, the possible outputs will belong to the same domain and consequently, we are faced with a domain-IL scenario. In this scenario, there is no information regarding a task identifier, leading to discard architectural or parameter isolation methods. Figure 5.1 shows the data flow in our setting. Dashed red lines indicate a change of regime (task in the continual learning problem) in the raw time series; the network takes as input a series of financial indicators, derived from the raw prices series, relative to a period, defined by a window on the timeline; the neural networks operate a binary classification defining if the series exhibits a positive or a negative trend.

### 5.1.3 Continual learning techniques

In this section, all the examined algorithms will be explored, describing the tricks we sometimes employed to adapt these methods to the problem. In fact, not all of the state-of-the-art methods are suitable for the problem of classifying financial time series or do not offer the desired performance.

#### 5.1.3.1 Gradient Episodic Memory

The early learning problems of multiple sequential tasks were based on Empirical Risk Minimization (ERM) [106], which defines the theoretical limits of the learning algorithm performance. This limitation comes from the inability to know the data distribution on which the algorithm will work. Gradient Episodic Memory (GEM) [73] was proposed as a learning method disconnected from data distribution and focused on an example-by-example observation. In particular, the classic example-label pair $(x, y)$ is abandoned in favour of a triplet $(x, y, t)$, where $t$ is a task descriptor. Applied to financial time series, the task descriptor can be the task-id, as done in this work, or a complex structure describing the distribution to which the data belong. The main feature of this method is the episodic memory $\mathcal{M}_t$ capable of saving a subset of each task $t$. With $T$ tasks and a total budget of $M$ memory locations, each task will have an exclusive memory equal to $M/T$. If the total number of tasks is not known a priori, it is possible to gradually reduce the number of examples for each task as the number of tasks increases. The goal of this method is to sequentially train a model on $T$ tasks, preventing overwriting in future tasks with the constraint that training a task should not lead to worse performance in previous tasks. Given a triplet $(x, y, t)$, the optimization problem to be solved is the following:

$$\begin{aligned} \text{minimize}_\theta \quad & \ell(f_\theta(x, t), y) \\ \text{subject to} \quad & \ell(f_\theta, \mathcal{M}_k) \leq \ell(f_\theta^{t-1}, \mathcal{M}_k) \text{ for all } k < t, \end{aligned} \tag{5.1}$$

The problem is complex to solve with this formulation, but it is possible to make two observations. The first one concerns the conservation of the parameters of each task: if the constraint on the loss is maintained, it is no longer necessary to save the state of the network at the end of each task. The second and, more important, allow us to represent the variation of the loss between two updates through the angle between the two gradient vectors if the function is locally linear, an assumption valid between two gradient steps. The second observation allows

us to rewrite the optimization constraints:

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t. \tag{5.2}$$

For every training step, there is, therefore, a system of $k$ inequalities to resolve. It is easy to guess that this operation becomes more onerous as the number of tasks increases. In case of at least one constraint is not met, a gradient step in a new direction is required. This makes the optimization problem a QP-complete problem for that specific training step. Only approximations of the optimal solution are valid, and the authors of the method proposed a valid one using the dual problem. Buffer size plays a relevant role in the performance evaluation: if we use more memory, we could expect better performances. But, in time series problems, this becomes tricky. A too big memory should allow saving too much data, going to a pseudo-parallel training of more tasks, and, in a scenario where temporal order is fundamental, this aspect must be avoided.

#### 5.1.3.2 Averaged Gradient Episodic Memory

Averaged Gradient Episodic Memory (A-GEM) [19] has been proposed as an optimization of the forerunner. To alleviate the weight of the computation, the authors opted for a relaxation of the constraints, going from loss reduction on examples of each of past tasks to an average on all the episodic memory. While the objective function to minimize remains the same, the constraints collapse to a single one valid for all past tasks.
As before, we can reformulate the objective function and the constraints regarding the previous observation of the loss variation and gradient vectors:

$$\text{minimize}_{\tilde{g}} \quad \frac{1}{2}||g - \tilde{g}||_2^2 \quad \text{s.t.} \quad \tilde{g}^\top g_{ref} \geq 0 \tag{5.3}$$

where $g_{ref}$ indicates the gradient computed from a random batch obtained by the episodic memory from all previous tasks; in other words, A-GEM replaces $t-1$ inequalities with only one. However, it remains possible that the unique constraint is not met. In this scenario, there is no particular problem or approximation to compute but the solution is given by the projected gradient method:

$$\tilde{g} = g - \frac{g^\top g_{ref}}{g_{ref}^\top g_{ref}} g_{ref} \tag{5.4}$$

As GEM, this method exploits different algorithms to fill the buffer. Since in A-GEM, we got a single batch sampled from the whole buffer, the data distribution of the batch could not reflect the original distribution between tasks. We opted for a different strategy to fill the buffer to maintain the correct distribution, equal as far as possible to the stream one, using the reservoir sampling algorithm.

#### 5.1.3.3 Synaptic Intelligence

A significant limitation in the development of neural networks capable of learning multiple sequential tasks lies in the one-dimensional structure of the neuron, leading a network to catastrophic forgetting. Defining which neurons are most responsible for learning is necessary to consolidate acquired knowledge on a task. The best way to assess how significant a neuron is for a task is to calculate its contribution to the global loss of the current task. In this way, at the end of each task, it will be possible to determine which neurons contribute most to the current task's learning and prevent their update in the future, maintaining knowledge of past tasks, thus avoiding forgetting. Synaptic Intelligence (SI) [119] does not require external memories or architecture variation. Still, it acts only on network parameters, defining an additional loss related to the state of the neurons themselves. When training on a new task, changes to important parameters are penalized to prevent old memories from being overwritten. We can compute weight importance $\omega_k^\mu$ for each neuron $\theta_k$ related to task $\mu$. During the training of a task, a learning trajectory $\theta(t)$ is described in the network parameter space. This trajectory will come as close as possible to a minimum for the loss function for each task. We can now consider an update $\delta(t)$ at time $t$, leading to a variation in the loss of the current task. This variation can be approximated by the gradient $g_k$ and in such case the relation

$$\ell(\theta(t) + \delta(t)) - \ell(\theta(t)) \approx \sum_k g_k(t) \cdot \delta_k(t) \tag{5.5}$$

can be considered valid. The variation $\delta_k(t) = \theta'_k(t) = \frac{\partial \theta_k}{\partial t}$ therefore contributes to the variation of the global loss. If we want to compute the variation over the entire trajectory, we must sum up all small updates. This amounts to computing the path integral of the gradient vector along the parameter trajectory. Since the gradient is a conservative field, the value of the integral is equal to the difference in loss between the endpoint $t_{end}$ and start point $t_{start}$. In addition, the integral can be decomposed as the sum of the impact of the importance $\omega_k^\mu$ on loss variation. In practice, $\omega_k^\mu$ is the online approximation of the running sum

of the product of the gradient $g_k(t) = \frac{\partial L}{\partial \theta_k}$ with the update $\theta'_k$. In a sequential tasks scenario, the model will have only a loss $\mathcal{L}_\mu$ available on the current task $\mu$. Catastrophic forgetting occurs when minimizing $\mathcal{L}_\mu$ there is a significant increase of the loss $\mathcal{L}_\upsilon$ of past tasks $\upsilon < \mu$. In this context, the importance of parameters $\theta_k$ is determined by: 1) how much the parameter contributes to a loss drop and 2) the difference $\theta_k(t^\mu) - \theta_k(t^{\mu-1})$. To avoid a significant variation in these parameters, a modified loss has been proposed:

$$\widetilde{\mathcal{L}_\mu} = \mathcal{L}_\mu + c \sum_k \Omega_k^\mu (\widetilde{\theta_k} - \theta_k)^2 \tag{5.6}$$

with $\widetilde{\theta_k} = \theta_k(t^{\mu-1})$ and parameter $c$ to manage regularization. Coefficient $\Omega_k^\mu$ determines the regularization strength of each parameter:

$$\Omega_k^\mu = \sum_{\upsilon<\mu} \frac{\omega_k^\upsilon}{(\Delta_k^\upsilon)^2 + \epsilon} \tag{5.7}$$

with $\Delta_k^\upsilon = \theta_k(t^\upsilon) - \theta_k(t^{\upsilon-1})$.

#### 5.1.3.4 Elastic Weight Consolidation

Like the previous one, this method presented by Kirkpatrick *et al.* [55] is based on the possibility of determining a coefficient of importance for each neuron to be used in the computing of the global loss. Given task A, multiple valid network weights configurations $\theta_A$ ensure the same performances. In this way, when task B occurs, the model will maintain the performance binding model parameters in a solution space with low error for the previous task while maximizing performance on the new task B. Elastic Weight Consolidation (EWC) does not aim to find the optimal solution for each task but focuses on finding an intersection of low-error solution space. To find which parameters are the most significant for a task, the authors addressed the problem from a probabilistic point of view. Optimizing the parameters of a network given the training set $\mathcal{D}$ is equal to probability $p(\theta|\mathcal{D})$. In presence of two independent tasks A($\mathcal{D}_\mathcal{A}$) and B($\mathcal{D}_\mathcal{B}$), with the Bayes' law we can write:

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B) \tag{5.8}$$

where the right-hand side depends only on the loss on task B $\log p(\mathcal{D}_B|\theta)$. Simultaneously, all the information regarding task A $\log p(\theta|\mathcal{D}_A)$ is soaked up by the

posterior probability. Suppose the true posterior probability cannot be computed. In that case, we can obtain a good approximation from a Gaussian distribution with mean given by parameters $\theta_A$ and angular precision from the diagonal of the Fisher Information Matrix $F$. This matrix has three key properties: 1) it is equivalent to the second derivative of the loss near a minimum, 2) it can be computed from first-order derivates, and 3) it is guaranteed to be positive semi-definite. Given this approximation, the loss function $\mathcal{L}$ to minimize:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{5.9}$$

where $\mathcal{L}_B(\theta)$ is the loss for task B, $\lambda$ sets the importance of the old task compared to the new one. When moving to a third task C, EWC will try to keep the model parameters close to the learned parameters on both tasks A and B, where this can be enforced either with two separate penalties.

#### 5.1.3.5 Experience Replay

The consolidation of acquired knowledge can occur in several ways in the human brain. One consists of periodic observation to consolidate the knowledge acquired but potentially overwritten over time. Experience Replay (ER) [89] uses an external memory buffer to save data from previous tasks, as example-label couple $(x, y)$, without any reference to the task that the data belong to. During each training step, in addition to the batch of examples of the current task, another batch composed of examples from past tasks is sampled from the buffer, and the loss is computed on both of its, driven by hyperparameters $\alpha$ and $\beta$:

$$\mathcal{L} = \alpha \cdot \mathbb{E}_{(x,y)\sim D_t}[l(y, f(x))] + \beta \cdot \mathbb{E}_{(x,y)\sim M}[l(y, f(x))] \tag{5.10}$$

where $\mathcal{D}_t$ denotes training set of the current task and $\mathcal{M}$ the external memory.

#### 5.1.3.6 Dark Experience Replay

Several CL algorithms have been proposed as improvements to ER. Dark Experience Replay (DER) [12] is one of these, and it relies on *dark knwowledge* for distilling past experiences, sampled over the entire training trajectory. Differently from the other rehearsal-based methods, this method retains the network's logits $z \triangleq h_{\theta_t}(x)$, instead of the ground truth labels $y$. This stratagem allows for avoiding the loss of information due to the compression made by the final activation

function. The corresponding loss function results:

$$\mathbb{E}_{(x,y)\sim D_t}[l(y, f(x))] + \alpha \mathbb{E}_{(x,z)\sim \mathcal{M}}\left[ \|z - h_\theta(x)\|_2^2 \right] \tag{5.11}$$

This approach is related to *Knowledge Distillation* [41], a paradigm that allows the transfer of knowledge from a teacher to a student model. In particular, DER exploits a variant of this, known as self-distillation [33], in which transfer occurs between the same architecture. In this scenario, by saving logits of previous task examples, the model transfers knowledge to a version of itself in the future. Moreover, *Dark Experience Replay* ++ has been proposed that equips equation 5.11 with an additional term on buffer datapoints, promoting higher conditional likelihood concerning their ground labels.

## 5.2 Experiments

### 5.2.1 Datasets

For the experimental analysis, two datasets have been employed. We used Brent Oil dataset [†], the historical series of the oil prices on a daily basis. In particular, we used 9282 time steps, collected between $02/01/1986$ and $31/07/2021$ [‡]. We also employed the copper dataset [†], consisting in $8500$ time steps, taken from $02/01/1989$ and $31/07/2021$ [‡]. An example contains twenty consecutive daily observations. The next example is obtained by shifting the time window by one timestep. To provide more refined information to the model, it was decided to proceed towards an engineering of the features using some of the most famous financial and statistical indicators, as explained in section 5.1.1. Finally, for the definition of the various tasks within the time series, Bayesian Online Change-point Detection (BOCD) was used, an online algorithm for the detection of change-points, i.e. moments in which a significant change occurs in the data distribution. To verify the validity of this technique, we asked a financial expert to manually find out the change-points on the time series. The results almost completely coincide with those found by the algorithm, with an occasional variation of maximum 1 or 2 time steps. This allows the whole continual learning process to work without the need for further human intervention to detect regime changes, allowing us to use public datasets without further processing. Within each task,

[†] https://datahub.io/core/oil-prices, https://help.yahoo.com/kb/SLN2311.html

[‡] Dates are reported as DD/MM/YY.

we split the data into two different sets: the train and evaluation sets. Between the two sets, we leave a gap excluding any sample whose evaluation time is posterior to the earliest prediction time in the validation set. This ensures that predictions on the validation set are free of look-ahead bias.

### 5.2.2 Metrics

To properly assess learning quality at training time, it is mandatory to consider both single tasks as the whole training process. In other words, a CL algorithm should be evaluated both on the *past* and the *present* tasks to reflect its behaviour on the *future* unseen tasks. It is crucial to assess the ability to transfer knowledge across tasks to achieve this, along with average accuracy (ACC). More specifically, we would like to measure *Forward Transfer* (FWT) and *Backward Transfer* (BWT) [73] (*Forgetting* (FRG) [17] has been omitted because it is equal to BT except for the sign). The first one assesses the influence that learning a task $t$ has on the performance on a future task $k > t$, whereas the second and third ones measure the performance degradation in subsequent tasks. FWT is computed as the difference between the accuracy before starting training on a given task and the random-initialized network, then averaged across all tasks. FRG and BWT compute the difference between the current accuracy and its best value for each task, presumably at the end of the training of the task itself. Except for FRG, the larger these metrics, the better the model. If two models have similar ACC, the preferable one is the one with a larger BWT and FWT. While BWT measures the influence of a task on the previous ones and FWT the influence on the following ones, it is meaningless to discuss backward for the first task or forward for the last one.

### 5.2.3 Architecture

All the methods under examination were evaluated with the same architecture, using a fully-connected network composed of three hidden layers with respectively 100, 50, and 25 neurons each and with LeakyReLu as an activation function. All methods were tested using Stochastic Gradient Descent (SGD) with momentum as an optimizer for a total of 480000 training steps for each task. For rehearsal methods, we set the buffer size to 500 samples.

| | Online | SI | EWC | ER | GEM | A-GEM | DER | DER++ |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 65.04 | 70.34 | 71.64 | 72.29 | 65.25 | 65.47 | 72.59 | **73.37** |
| Backward Transfer | - | -6.06 | -4.15 | **-3.72** | -11.29 | -5.74 | -4.96 | -4.11 |
| Forward Transfer | - | 25.28 | 26.02 | 24.05 | 12.21 | 25.97 | 23.39 | **27.24** |

Table 5.1: Results of tested methods on the Brent Oil dataset. For accuracy, backward and forward transfer bigger is better.

| | Online | SI | EWC | ER | GEM | A-GEM | DER | DER++ |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 58.28 | 65.46 | 68.01 | 68.00 | 54.23 | 58.86 | 64.09 | **68.77** |
| Backward Transfer | - | -5.08 | -3.28 | **-0.85** | -8.94 | -6.74 | -4.22 | -3.11 |
| Forward Transfer | - | 14.45 | 11.64 | 17.92 | 7.02 | 13.39 | **14.32** | 12.28 |

Table 5.2: Results of tested methods on the copper dataset. For accuracy, backward and forward transfer bigger is better.

### 5.2.4 Quantitative results

This section will discuss the results of each Continual Learning method, comparing them with the sequential training of each task without any continual technique, called *online learning*. Each method is evaluated not only with accuracy across all tasks but also with ad hoc Continual Learning metrics: *forward transfer* and *backward transfer*. In Table 5.1 and Table 5.2 are reported performances measured at the end of the whole training, respectively on the Brent oil dataset and on the copper dataset. To obtain less noisy performance estimates, values are reported as averages of three runs with different initialization. In the regularization methods, SI and EWC, we found an attenuation of the average forgetting across all tasks, even if the task nature heavily influences the accuracy of past tasks. This aspect is emphasized in SI, while EWC demonstrates good stability. Figure 5.2(a) shows the evolution of the accuracy of these methods during the training, compared to online learning on the Brent oil dataset, while Figure 5.2(c) on the copper dataset. SI and EWC experience similar accuracy on both datasets, while in the continual metrics, EWC performs slightly better. In replay-based methods (Figure 5.2(b) on the Bernt oil dataset and Figure 5.2(d) on the copper dataset), GEM and A-GEM are complex and hard-to-evaluate algorithms. GEM experiences the worst performance across all continual methods tested, especially on the copper dataset; probably due to the constraints violations introduced by the technique, even more and more frequently as the number of

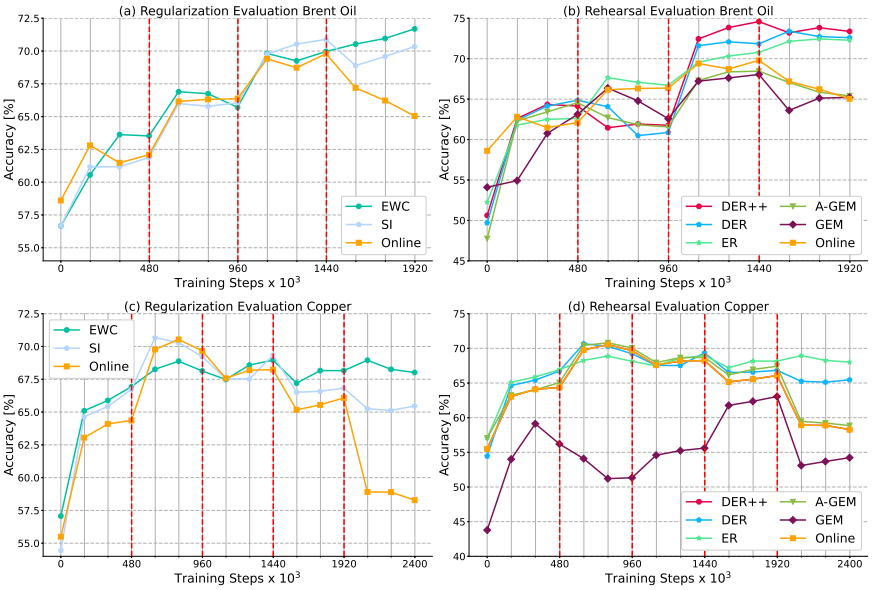



Figure 5.2: Performance results. Dashed lines indicate task change: regularization methods performance on Brent Oil dataset (a), rehearsal method performance on Brent Oil dataset (b), regularization methods performance on the copper dataset (c), rehearsal method performance on the copper dataset (d).

tasks increases. Although A-GEM significantly relaxes the constraints, accuracy remains aligned with baseline online learning while continual metrics are comparable to other methods. Vice versa, ER, DER and, DER++ experience the best performances. DER++ combines the features of the other two methods taking advantage of them: replaying past data from the buffer as ER, it uses logits information context as done in DER to improve further the predictions, bringing it to be the most performing method. In Figure 5.3(a) and (b) are reported the training times of each method, respectively on the Brent oil dataset and on the copper dataset. Again, the behaviour is similar on both datasets, demonstrating the robustness of these algorithms on financial time series. SI demonstrates to be the quickest to be trained thanks to online estimate model weights. Simultaneously, the other regularization method, EWC, is the second most time-consuming algorithm due to the necessity to compute the Fisher Information Matrix at the end

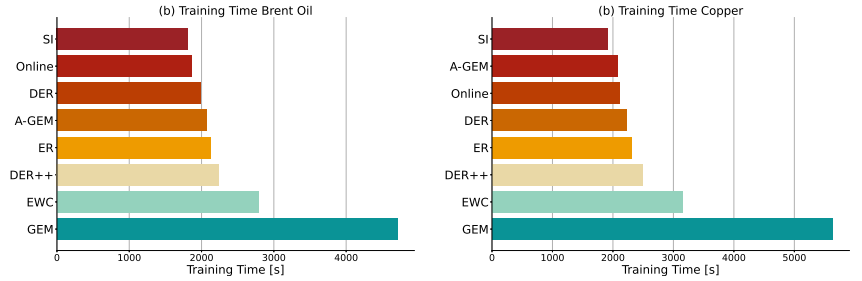



Figure 5.3: Training time comparison on Brent Oil dataset (a) on the left and the copper dataset (b).

of each task. GEM constraints, finally, do not make up a complexity element only from a computational perspective, but they also make this method the most time-consuming for model training.

# Chapter 6

# Conclusions

This thesis involves three important research topics I investigated as a PhD student, resulting in several contributions in trajectory forecasting, continual learning, and financial market prediction areas. In this final chapter, a summary of the contribution of each of them, along with future objectives, is reported.

## 6.1 Trajectory forecasting in real-world environment

We proposed two novel architectures for multi-future trajectory forecasting. The first one uses VRNNs in a predictive setting. An attentive module includes interactions through a hidden state refinement process based on a graph neural network in an online fashion at a time step level. In AC-VRNN, local belief maps encourage the model to follow a future displacement probability grid when the model is not confident about its prediction. We test our algorithm on several trajectory prediction datasets collected in different urban scenarios achieving the best performance compared to state-of-the-art methods. On the other end, DAG-Net is composed of a double graph-based network that deals with both past interactions and future goals through an attentive mechanism. By facing trajectory prediction as a structured problem, our models overcome state-of-the-art performances on several trajectory forecasting benchmarks, proving their strength in team sports and urban contexts. They show impressive results also on long-term predictions. Our future work will be towards a detailed analysis of long-term predictions in order to deal with more complex and uncertain scenarios. Furthermore, an inter-

esting aspect would be to include into the model additional scene context (e.g., depth data or WiFi/BLT signals) in order to design a multi-modal architecture to gain the advantage of multiple modalities. Of course, these two approaches are just a starting point for a research on trajectory forecasting in real-world environments.

Although these approaches are promising in the context of trajectory forecasting, they are not designed to deal with an environment in which interactions and obstacles considerably evolve over time. To face this issue, the aim of our future research is to apply continual learning techniques to trajectory forecasting and to improve video surveillance systems in online and evolving scenarios.

## 6.2 Meta-Continual learning in complex scenarios

We tackle a novel problem concerning few-shot unsupervised continual learning, proposing an effective learning strategy based on the construction of unbalanced tasks and meta-examples. With an unconstrained clustering approach, we find that no balancing technique is necessary for an unsupervised scenario that needs to generalise to new tasks. Our model, exploiting a single inner update through meta-examples, increase performance as the most relevant features are selected. In addition, an original augmentation technique is applied to reinforce its strength. We show that MEML and MEMLX not only outperform the other baselines within FUSION but also exceed state-of-the-art approaches in class-incremental continual learning.

However, our proposed approach has to deal with the limitation of rehearsal strategies, such as the availability of memory, with the risk of incurring catastrophic forgetting. Novel approaches can be studied to overcome this issue. Moreover, our method is applicable to Out-of-Distribution tasks, but it is not specifically designed for them. Interesting future research is to investigate a more effective strategy that further improves performance when facing Out-of-Distribution data and domain shifts.

## 6.3 Continual learning techniques for financial market predictions

An experimental analysis of continual learning algorithms on market predictions has been conducted, highlighting their significant contribution to the field of artificial intelligence applied to finance. A deep investigation of the most promising state-of-the-art continual learning algorithms has been made, discovering that not all of them are suitable for financial time series. Furthermore, we found that other factors such as training time, computational complexity, and memory requirements can be decisive in defining the scenario and choosing the most appropriate algorithm to apply. The formulation adopted represents only an exemplifying model of more complex dynamics, but the development of this tool could be a concrete help for professional traders in the future.

The major limitation of this research is the fact that no specifically designed solution has been introduced to face regime changes in financial time series. In the future, continual learning algorithms could be studied, to take into account the variety and complexity of the markets.

# Acknowledgements

This thesis has been made possible thanks to the help and contributions of a lot of people.

First of all, I would like to acknowledge my advisors, Simone and Andrea, who encouraged and helped me in my research activities. A special thank also goes to Axyon AI, and in particular to Jacopo and Alberto.

I would like to acknowledge my best friends Stefano, Roberta and Chiara, my partner Nicola, and all my family for sustaining me on this long path.

# List of publications

- Alessia Bertugli, Stefano Vincenzi, Simone Calderara, and Andrea Passerini. Generalising via meta-examples for continual learning in the wild. *The 8th International Conference on machine Learning, Optimization and Data science*, 2022.

- Alberto Zurli, Alessia Bertugli, and Jacopo Credi. Does catastrophic forgetting negatively affect financial predictions? *The 8th International Conference on machine Learning, Optimization and Data science*, 2022.

- Alessia Bertugli, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. Ac-vrnn: Attentive conditional-vrnn for multi-future trajectory prediction. *Computer Vision and Image Understanding*, 210, 2021. 25, 43

- Alessia Bertugli, Stefano Vincenzi, Simone Calderara, and Andrea Passerini. Few-shot unsupervised continual learning through meta-examples. *Neural Information Processing Systems Workshops*, 2020.

- Alessio Monti, Alessia Bertugli, Simone Calderara, and Rita Cucchiara. Dagnet: Double attentive graph neural network for trajectory forecasting. In *International Conference on Pattern Recognition*, 2020. 25, 43

# Activities

Beside the research constituting the main corpus of the thesis, I was involved in several supplemental activities, such as teaching and other services. In the remainder of the chapter I will report them.

**Partecipation to national and European research projects**

- European High-Performance Computing Joint Undertaking (JU) project *"FF4EuroHPC: HPC Innovation for European SMEs"*, grant agreement No 951745.
- MIUR PRIN project *"PREVUE: PRediction of activities and Events by Vision in an Urban Environment"*, grant ID E94I19000650001.

**Collaborations**

- Aliza Technologies, Los Angeles, June 2020 - November 2022
- Axyon AI, Modena, January 2020 - May 2020

**Master thesis supervision**

- "Continual learning techniques applied to financial time-series", University of Modena and Reggio Emilia, October 2021 - Dott. Alberto Zurli
- "Meta-learning with unsupervised training in a continual few shot scenario", University of Modena and Reggio Emilia, October 2020 - Dott. Emanuele Frascaroli
- " Forecasting future trajectories by agents' interactions and goals", University of Modena and Reggio Emilia, April 2020 - Dott. Alessio Monti

**Reviewing activities**

- The 8th International Online & Onsite Conference on Machine Learning, Optimization, and Data Science (LOD), 2022
- IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, 2022
- International Conference on Computer Vision (ICCV), 2021
- Neural Information Processing Systems (NeurIPS) Workshop on Meta-Learning, 2020
- International Conference on Pattern Recognition (ICPR), 2020

**Conferences and schools**

- The 8th International Online & Onsite Conference on Machine Learning, Optimization, and Data Science (LOD), 2022
- Mediterranean Machine Learning Summer School, 2021
- Neural Information Processing Systems (NeurIPS) 2020
- Ph.D. School on Advanced Topics in Deep Learning, Verona, 2019

**Seminars and courses**

- Geometric Computer Vision: from Images to 3D Models, Virtual, 2021
- Academic Writing for Sciences and Engineering, Virtual, 2020
- Research Methodology, Trento, 2020
- Teaching Computers to Imagine with Deep Generative Models, Trento, 2019
- Brain-Inspired Computing: from Neuroscience to Artificial Intelligence, Modena, 2019

**Awards**

- Outstanding reviewer at IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)
- Invited spotlight at 25th International Conference on Pattern Recognition (ICPR 2020), International Workshop on Pattern Forecasting

# Bibliography


[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020. 22

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 13, 20, 27, 28, 42, 44

[3] A. Alahi, V. Ramanathan, and L. Fei-Fei. Socially-aware large-scale crowd forecasting. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2014. 40

[4] Ferran Alet, Erica Weng, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Neural relational inference with fast modular meta-learning. In *Neural Information Processing Systems*, 2019. 27

[5] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020. 61, 62, 63, 64

[6] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. In *European Conference on Artificial Intelligence*, 2020. 21, 24, 25, 61, 62, 70, 71

[7] Stefan Becker, Ronny Hug, Wolfgang Hübner, and Michael Arens. Red: A simple but effective baseline predictor for the trajnet benchmark. In *European Conference on Computer Vision Workshops*, 2018. 39

[8] David Berthelot*, Colin Raffel*, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019. 61, 63, 70

[9] Alessia Bertugli, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. Ac-vrnn: Attentive conditional-vrnn for multi-future trajectory prediction. *Computer Vision and Image Understanding*, 210, 2021. 25, 43

[10] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction. In *Neural Information Processing Systems Workshops*, 2019. 36

[11] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The inD Dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *IEEE Intelligent Vehicles Symposium*, 2020. 39

[12] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Neural Information Processing Systems*, 2020. 13, 21, 90

[13] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexande Lacoste, David Vazquez, et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. In *Neural Information Processing Systems*, 2020. 24, 59, 61

[14] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018. 61, 63, 70

[15] Yichuan Charlie, Tang, Ruslan, and Salakhutdinov. Multiple futures prediction. In *Neural Information Processing Systems*, 2019. 27

[16] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision*, 2018. 74

[17] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 92

[18] Arslan Chaudhry, A. Gordo, P. Dokania, P. Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *ArXiv*, abs/2002.08165, 2020. 74, 75

[19] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019. 22, 59, 87

[20] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 40

[21] Chen Chieh-Yu, Lai Wenze, Hsieh Hsin-Ying, Zheng Wen-Hao, Wang Yu-Shuen, and Chuang Jung-Hong. Generating defensive plays in basketball games. In *ACM International Conference on Multimedia*, 2018. 27

[22] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, 2020. 60

[23] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Neural Information Processing Systems*, 2015. 30, 49, 50

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations*, 2016. 32

[25] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition, 2018. 25

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009. 62

[27] Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance*. Springer, Cham, 1st edition, 2020. 25

[28] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017. 61, 63

[29] Panna Felsen, Patrick Lucey, and Sujoy Ganguly. Where will they go? predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders. In *European Conference on Computer Vision*, 2018. 27

[30] Panna Felsen, Patrick Lucey, and Sujoy Ganguly. Where will they go? predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders. In *ECCV*, 2018. 36, 44, 45

[31] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017. 22

[32] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, 2019. 24

[33] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018. 91

[34] Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, 2018. 22

[35] Mykola Pechenizkiy Ghada Sokar, Decebal Constantin Mocanu. Self-attention meta-learner for continual learning. In *International Conference on Autonomous Agents and Multiagent Systems*, 2021. 74

[36] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. Maxup: A simple way to improve generalization of neural network training. *arXiv preprint arXiv:2002.09024*, 2020. 67

[37] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 19, 20, 27, 28, 29, 41, 42, 43, 44

[38] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Neural Information Processing Systems*, 2020. 24, 59

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 73

[40] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 19, 27

[41] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 91

[42] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *European Conference on Computer Vision*, 2018. 22

[43] Hsieh Hsin-Ying, Chen Chieh-Yu, Wang Yu-Shuen, and Jung-Hong Chuang. Basketballgan: Generating basketball play simulation through sketching. In *ACM International Conference on Multimedia*, 2019. 27

[44] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*, 2019. 23, 59, 61, 62, 64, 69, 78

[45] B. Ivanovic and M. Pavone. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *IEEE International Conference on Computer Vision*, 2019. 19, 42

[46] Amirian Javad, Hayet Jean-Bernard, and Pettré Julien. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 27, 41, 42

[47] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Neural Information Processing Systems*, 2019. 21, 24, 25, 59, 61, 62, 69, 70, 71, 72, 77, 80

[48] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Neural Information Processing Systems*, 2019. 24, 59

[49] Zilong Ji, Xiaolong Zou, Tiejun Huang, and Si Wu. Unsupervised few-shot learning via self-supervised training. *ArXiv*, abs/1912.12178, 2019. 23, 59, 61

[50] Li Jiachen, Ma Hengbo, and Tomizuka Masayoshi. Conditional generative neural system for probabilistic trajectory prediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019. 27, 36

[51] Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. Unsupervised meta-learning for few-shot image and video classification. *Neural Information Processing Systems*, 2019. 23, 59, 61

[52] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *Neural Information Processing Systems*, 2014. 34, 36

[53] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014. 29, 30

[54] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, 2018. 20, 27

[55] James N Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *In Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526, 2016. 21, 59, 74, 89

[56] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian D. Reid, Seyed Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Neural Information Processing Systems*, 2019. 13, 19, 20, 28, 42

[57] Parth Kothari, S. Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, in press, 2020. 39, 40, 44, 45

[58] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Canadian Institute for Advanced Research, 2009. 62, 72, 79

[59] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 62

[60] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 62, 72

[61] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *International Conference on Learning Representations*, 2020. 61, 79

[62] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *IEEE International Conference on Computer Vision*, 2019. 22, 61

[63] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Krishna Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017. 27

[64] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):1186–1194, 2007. 39

[65] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2935–2947, 2018. 22, 74

[66] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020. 27

[67] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G. Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2019. 27, 28, 42

[68] Matteo Lisotto, Pasquale Coscia, and Lamberto Ballan. Social and scene-aware trajectory prediction in crowded spaces. In *IEEE/CVF International Conference on Computer Vision Workshops*, 2019. 27

[69] Qing Liu, Orchid Majumder, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Incremental learning for metric-based meta-learners. *ArXiv*, abs/2002.04162, 2020. 24, 25, 59

[70] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D. Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 22

[71] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. In *Neural Information Processing Systems Workshops*, 2020. 25

[72] Yuejiang Liu, Qi Yan, and Alexandre Alahi. Social NCE: Contrastive learning of socially-aware motion representations. *ArXiv*, abs/2012.11717, 2020. 45

[73] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Neural Information Processing Systems*, 2017. 13, 22, 59, 73, 74, 75, 86, 92

[74] Marcos M. López de Prado. *Machine Learning for Asset Managers*. Cambridge University Press, 2020. 13, 25

[75] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *AAAI Conference on Artificial Intelligence*, 2018. 27

[76] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2009. 19

[77] Alessio Monti, Alessia Bertugli, Simone Calderara, and Rita Cucchiara. Dag-net: Double attentive graph neural network for trajectory forecasting. In *International Conference on Pattern Recognition*, 2020. 25, 43

[78] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *ArXiv*, abs/1803.02999, 2018. 22, 73

[79] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *IEEE International Conference on Computer Vision*, 2009. 39, 40

[80] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020. 23, 64

[81] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml : An incremental task-agnostic meta-learning approach. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020. 24, 25, 59, 61

[82] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019. 22, 23

[83] Dushyant Rao, Francesco Visin, Andrei A. Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. In *Neural Information Processing Systems*, 2019. 22, 59, 61

[84] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. 69

[85] Sylvestre-Alvise Rebuffi, Alexander I Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017. 13, 22, 59, 61, 73, 74

[86] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019. 22, 59, 73, 74, 75

[87] J. Rios-Martinez, A. Spalanzani, and C. Laugier. From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, 7(2):137–153, 2015. 32

[88] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, 2016. 39

[89] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Neural Information Processing Systems*, 2019. 13, 22, 74, 75, 90

[90] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. Human motion trajectory prediction: A survey. *International Journal of Robotics Research*, 39(8):895–935, 2020. 27

[91] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016. 21

[92] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. 22

[93] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *ArXiv*, 2018. 39

[94] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 13, 20, 28, 42

[95] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018. 21

[96] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Neural Information Processing Systems*, 2017. 22

[97] Daniel L. Silver and Sazia Mahfuz. Generating accurate pseudo examples for continual learning. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 22

[98] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. 2017. 23

[99] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *Neural Information Processing Systems*, 2015. 36, 44

[100] G. Spigler. Meta-learnt priors slow down catastrophic forgetting in neural networks. *ArXiv*, abs/1909.04170, 2019. 22

[101] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B. Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. In *International Conference on Learning Representations*, 2019. 19, 20, 27

[102] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020. 24, 25

[103] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020. 79

[104] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Neural Information Processing Systems*, 2016. 35

[105] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. In *Neural Information Processing Systems Workshops*, 2018. 21, 73, 82

[106] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, 1991. 86

[107] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 25

[108] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 20, 31, 38, 55

[109] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *IEEE International Conference on Robotics and Automation*, 2018. 27, 44

[110] Risto Vuorio, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim. Meta continual learning. *ArXiv*, abs/1806.06928, 2018. 24, 59

[111] Ezra Webb, Ben Day, Helena Andres-Terre, and Pietro Lió. Factorised neural relational inference for multi-interaction systems. In *International Conference on Machine Learning*, 2019. 27

[112] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 79

[113] H. Xue, D. Q. Huynh, and M. Reynolds. SS-LSTM: A hierarchical lstm model for pedestrian trajectory prediction. In *IEEE Winter Conference on Applications of Computer Vision*, 2018. 20, 27

[114] Zhi Yan, Tom Duckett, and Nicola Bellotto. Online learning for human classification in 3d lidar-based tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 40

[115] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, 2019. 24, 25, 59

[116] R. A. Yeh, A. G. Schwing, J. Huang, and K. Murphy. Diverse generation for multi-agent sports games. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2019. 20, 27

[117] Huang Yingfan, Bi Huikun, Li Zhaoxin, Mao Tianlu, and Wang Zhaoqi. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *IEEE International Conference on Computer Vision*, 2019. 13, 20, 27, 41, 42, 43, 44, 55

[118] F. Zanlungo, T. Ikeda, and T. Kanda. Social force model with explicit collision prediction. *EPL (Europhysics Letters)*, 93(6):68005, mar 2011. 19

[119] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017. 21, 22, 59, 74, 88

[120] Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. Generating multi-agent trajectories using programmatic weak supervision. In *International Conference on Learning Representations*, 2019. 19, 27, 36, 42, 43

[121] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. SR-LSTM: State refinement for lstm towards pedestrian trajectory prediction. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2019. 13, 20, 27, 41, 42