Towards Event Analysis in Time-series Data: Asynchronous Probabilistic Models and Learning from Partial Labels

by

Nazanin Mehrasa

M.Sc., Simon Fraser University, 2017
B.Sc., Amirkabir University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
Department of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Nazanin Mehrasa**
Degree:	**Doctor of Philosophy**
Thesis title:	**Towards Event Analysis in Time-series Data: Asynchronous Probabilistic Models and Learning from Partial Labels**
Committee:	**Chair:** Parmit Chilana Assistant Professor, Computing Science

Abstract

In this thesis, we contribute in two main directions: modeling asynchronous time-series data and learning from partial labelled data. We first propose novel probabilistic frameworks to improve flexibility and expressiveness of current approaches in modeling complex real-world asynchronous event sequence data. Second, we present a scalable approach to end-to-end learn a deep multi-label classifier with partial labels. To evaluate the effectiveness of our proposed frameworks, we focus on visual recognition application, however, our proposed frameworks are generic and can be used in modeling general settings of learning event sequences, and learning multi-label classifiers from partial labels. Visual recognition is a fundamental piece for achieving machine intelligence, and has a wide range of applications such as human activity analysis, autonomous driving, surveillance and security, health-care monitoring, etc. With a wide range of experiments, we show that our proposed approaches help to build more powerful and effective visual recognition frameworks.

Keywords: Point Processes, Temporal Point Processes, Activity Prediction, Visual Recognition, Learning From Partial Labels

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Greg Mori; though, words are powerless to express my appreciation for his inspiring enthusiasm and continuous support throughout my PhD life with his friendship, supervision, and resourcefulness. I am wholeheartedly grateful for all his invaluable contributions of time and insight to make my PhD research productive and exciting. Thank you, Greg!

I would like to thank the members of my PhD dissertation committee: Dr. Angel Chang, Dr. Manolis Savva, Dr. Nicolas Thome, and Dr. Parmit Chilana for their time to read my thesis and attend my defense.

My research achievements benefit significantly from interacting with an amazing group of collaborators and lab-mates. Special thanks to Thibaut Durand, Eric He, and Hossein Hajimirsadeghi for all their help and mentorship. I would also like to thank Micael Carvalho, Ruizhi Deng, Bo Chang, Akash Abdu Jyothi, Mehran Khodabandeh, Fred Tung, Mohamed Osama Ahmed, Srikanth Muralidharan, Zhiwei Deng, Yu Gong, Megha Nawhal, Mengyao Zhai, Lei Chen, Sha Hu, Yifang Fu, Mostafa S. Ibrahim, Mohammad Hadi Salari, and Hamed Shirzad.

I was very fortunate to have many great friends in Vancouver during my grad life. Many thanks to Sima Jamali, Hossein Sharifi, Hossein Asghari, Amir Yaghoubi, Abdollah Safari, Huyen Mori, Ali Arab, Kiarash Zahirnia, Hamid Homapour, Zahra Zohrevand, Mehdi Shirmaleki, Babak Salimi, Kiana Mostaghasi, Ashkan Alinejad, Payam Ahmadvand, Mahsa Gharibi, Mahdi Nemati Mehr, Ramtin Mehdi Zade, Sajjad Gholami, Rana Sadeghi, Ehsan Haghshenas, Sina Salari, Narges Ashtari, Amirali Sharifian, Karoon Rashedi, Saman Taheri, Mina Taheri, Mohammad Tayebi, Sedighe Razmpour, Leo, and the list goes on ...

My heartfelt thanks go to my beloved Akbar for having this journey with me, for never giving up and always encouraging me, for believing in me unconditionally, and for tolerating an always-busy, always-stressed, always-in-a-deadline partner. Thank you Akbar!

I feel deeply indebted to the selfless love and endless support of my lovely mom, dad, and sister for giving me their boundless love, infinite kindness and support to do my best!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we aim to study, explore, and develop a set of robust deep learning approaches in two main directions: modeling asynchronous time-series data and learning from partial label data. On modeling asynchronous time series data, we focus on the anticipatory reasoning of future events given a history of sparse and asynchronous observations of past events in time. We propose novel probabilistic models based on the framework of temporal point process which explicitly models the occurrence rate of future events given the timing and other characteristics of previous events. On learning from partial labels, we tackle the problem of learning deep networks with partial labels for classification of static images, and introduce a novel scalable solution for end-to-end learning of deep convolutional networks with partial labels.

1.1 Motivation

In this thesis, we focus on two challenging problems: (1) modeling event in time-series data with a particular focus on asynchronous event data, and (2) learning from large databases of partially labeled data. We choose visual recognition as a testbed to evaluate the effectiveness of our proposed approaches, however, our proposed frameworks are generic and can be used in modeling general settings of learning event sequences, and learning multi-label classifiers from partial labels. Visual recognition is a fundamental problem in computer vision with a wide range of applications in machine intelligence systems. In the past decade, visual recognition has attracted a significant amount of research. Most recently, with the emergence of deep learning and neural networks, there has been a significant improvement in this field. However due to the challenging nature of this problem, the performance of state-of-the-art models are still far from human recognition performance. This performance gap mainly arises from the fact that designing an effective recognition framework requires the ability to address challenging tasks of visual understanding, working with large databases, learning from noisy and partially labeled data, capturing object correlations, modeling time-series data, learning temporal dependencies and structures, etc. We show that our proposed approaches help to build stronger and more powerful frameworks in modeling time-series and learning from partial-labels domains, helping to build a more effective visual recognition framework.

1.1.1 Event Analysis in Asynchronous Time-series Data.

Event sequences, as a particular form of time-series data, are discrete events in continuous time, meaning that they happen irregularly and asynchronously in continuous time. This type of data is prevalent in a broad spectrum of areas such as human activities, health-care, stock market, seismology, e-commerce, social networks, etc.

Human activities produce sequences of events data whose understanding their complex temporal dynamic plays an important role in many video intelligent system applications such as surveillance and security, health-care monitoring, simulation systems, and etc. In online social media such as Facebook and Twitter, user activities can be seen as another example of event sequences. In this environment, users share news, opinions, and interact with other people. Understanding these social behaviors is of many domains interest, such as economic, advertisement, and marketing. In seismology, scientists work with large databases of earthquake records. Records of earthquakes in time are another example of event sequences since earthquakes occur sparsely and asynchronously in time. One active research field in this area is to predict future earthquakes based on the records of past earthquakes. If successful, it could help to save many lives, prevents major destruction, urban planning, etc.

In all the examples above, each event is discrete, and the temporal dynamics of events are complex and asynchronous, meaning that in a sequence, events happen irregularly in continuous time. There are a variety of complex processes behind these events. Basically, each event is an observation of a complex dynamic process. It is crucial to understand the characteristics and dynamics of this type of data so that plausible future predictions, as well as other downstream applications, such as intervention or recommendation, can be performed.

Although the analysis of sequential data has a very rich literature in time-series analysis, the asynchronous and probabilistic nature of event sequence data makes it challenging to utilize the power of off-the-shelf time-series approaches. In this line, at the first glance, discrete-time Markovian models such as AutoRegressive models [86], Kalman filter models [130], and Hidden Markov Models [35, 148] might seem a good match for modeling event sequences, however, these approaches are designed for discrete time-series which are regularly spaced data points in time. A common approach to use these models is to transform event sequences into regularly spaced data points. For example, for the case of earthquake data, we could represent it as a time-series of zeros and ones, where one indicates occurrences of an earthquake. However, such a setup is sensitive to the choice of aggregation window being used for this transformation, which might cause some information loss due to discretization error. Also, these transformations might make sequences much longer, which increases computational cost. Furthermore, Markov models do not perform well in capturing long-term dependencies due to the state-space explosion issue involved with their model design. Continuous-time variants of Markov models [34, 55] relax the need for regularly spaces data point, but still suffers from the state-space explosion issue for capturing long-term dependencies.

Temporal point processes (TPPs) [22] provide us with an elegant and effective mathematical framework for modeling event sequences data. A temporal point process is defined as a stochastic process whose realizations consist of a list of events with their corresponding occurring times. These occurring times can either be real numbers from an index set (defined from prior knowledge) or sampled from an intensity function. While other time-series models learn temporal patterns synchronously (with each time-step being treated as an input to the model), TPP-based frameworks directly model the time intervals between events as random variables. With such a setup, it allows for modeling long sequences without vanishing gradients or costly memory issues. Moreover, temporal point process is able to mathematically incorporate the whole history in its model design (to capture long-term dependencies) without specifying the order as required by Markovian models.

Formally, a temporal point process is a stochastic process whose realization is a sequence of discrete events in time $t_{1:n} = (t_1, t_2, ..., t_n)$, where $t_i \in \mathbb{R}_{\geq 0}$ is the time when the i^{th} event occurs. These events usually come with other characteristics such as type of event, actor of event, etc. In this literature, this information is known as *mark* and is modeled with the framework of *mark temporal point process* [22]. A mark temporal point process provides a probability distribution over events timing and the corresponding marks. Similarly, a mark temporal point process is an stochastic process whose realization is a list of events described with their corresponding time and mark $x_{1:n} = (x_1, \ldots, x_n)$ where each event $x_i = (t_i, y_i)$ is represented by the time it happens t_i as well as the mark y_i. In this thesis, we formalize the input to the problem, similar to mark temporal point process as sequences of events $x_{1:n} = (x_1, \ldots, x_n)$ described by their occurring times and marks $x_i = (t_i, y_i)$.

A temporal point process is usually characterized with the conditional intensity function $\lambda(t)$ which encodes the expected rate of events happening in a small area around t given the history of past events. More precisely, the intensity function $\lambda(t)$ is defined as the conditional probability of observing an event in an infinitisemal area $[t, t + dt)$:

$$\lambda(t|\mathcal{H}(t))dt = \mathbb{P}\{\text{event in } [t, t + dt)|\mathcal{H}(t)\} \tag{1.1}$$

where $\mathcal{H}(t) = (t_1, t_2, ..., t_{i-1})$ is the ordered sequence of all events that happened before time t with $t_1 < t_2 < \cdots < t_{i-1} < t$. Given the intensity defined as above, the conditional probability density function of the time of the next event in the sequence can be written as follows [105][1]:

$$f(t|\mathcal{H}(t)) = \lambda(t|\mathcal{H}(t)) \exp\left\{ - \int_{t_{i-1}}^{t} \lambda(u|\mathcal{H}(u)) \ \mathrm{d}u \right\} \tag{1.2}$$

For a long while, various works in this literature used to build hand-crafted intensity function in order to define a temporal point process [46, 54, 69]. For example, Poisson process [69] assumes that events happen independent of each other where the intensity is a fixed positive constant. In a more general case, inhomogeneous Poisson process is based on the assumption that intensity could be a function of time, but still independent of other events. The key contribution of all these models is to find a functional form of intensity that fits data distribution well by making various parametric assumptions on the underlying generative process of the data. Although shown effective in modeling simple synthetic datasets, these strong parametric assumptions make such frameworks lack the flexibility to model the generative process for real-life and complex data, hindering wider adoption of TPP-based frameworks.

Deep Neural Network (DNN) based algorithms have been shown effective and promising for various tasks including classification [23, 24], retrieval [10], prediction [47], and more. To improve the flexibility of point processes, multiple works proposed using DNNs especially recurrent neural networks (or its more recent variants such as LSTM [51], GRU [17]) in temporal point process learning [29, 60, 93, 134, 144]. In this line of work, history information is encoded by utilizing recurrent neural networks and exploited in learning the intensity of the point process distribution. Although improving over hand-crafted approaches, in these works, the intensity function is usually limited to simple forms which restricts the model performance. This is because the maximum likelihood training criteria involved with these models requires the intensity function to be simple for the likelihood to stays tractable[2]. More recently, a few works have tried to formulate TPP in an intensity-free manner [79, 132, 133]. WGANTPP [132, 133] introduces modeling the point process distribution using Wasserstein distance with generative adversarial network (GAN). RLPP [79] formulates this problem in a reinforcement learning framework and treats future event predictions as actions taken by an agent. Both of these models are optimized by trying to generate sequences of samples that are indistinguishable from the ground-truth sequences. Although these models are capable of generating realistic sequences, such training criteria fail to model the data distribution, resulting in intractable likelihood.

In this dissertation, we contribute to event analysis in asynchronous time-series data by introducing novel probabilistic models under the prospective of temporal point processes. Our proposed frameworks aim to improve the flexibility and expressiveness of point processes in modeling complex

[1]The proof can be found in Proposition 2.1 of Rasmussen *et al.* [105]

[2]In point processes, when specifying the process by intensity function, an integration over the intensity function will appear in the functional form of the likelihood (Equation 1.2).

real-world event sequences. First, we formulate our model with variational auto encoder (VAE) paradigm, a powerful class of probabilistic models, and present a novel form of VAE modeling the distribution of timing and categories of event sequences. Second, we connect the fields of point processes and neural density estimation and propose a recurrent latent variable framework that directly models point processes distribution by utilizing normalizing flows. This approach is capable of capturing highly complex temporal distribution and does not rely on any restrictive parametric forms. Section 1.2 explains our contribution to this direction in more details.

1.1.2 Learning from Partially Labeled Data.

Recently, Stock and Cisse [116] presented empirical evidence that the performance of state-of-the-art classifiers on ImageNet [109] is largely underestimated – much of the remaining error is due to the fact that ImageNet's single-label annotation ignores the intrinsic multi-label nature of the images. Unlike ImageNet, multi-label datasets (*e.g*. MS COCO [82], Open Images [75]) contain more complex images that represent scenes with several objects. However, collecting multi-label annotations is more difficult to scale-up than single-label annotations [25]. As an alternative strategy, one can make use of partial labels; collecting partial labels is easy and scalable with crowdsourcing platforms like Amazon Mechanical Turk[3], and Google Image Labeler[4] or web services like reCAPTCHA[5] which can scalably collect partial labels for a large number of images.

This direction is actively being pursued by the research community [122, 131, 136, 137]. However, these approaches are not scalable and cannot be used to fine-tune a ConvNet. In this dissertation, we contribute to learning from partial labels by presenting a scalable approach to end-to-end learn a deep network with partial labels. More specifically, we propose a framework for learning from partially labeled image data with a multi-label classifier. First, we empirically compare several labeling strategies to highlight the potential for learning with partial labels. Second, we introduce a new loss function that enables end-to-end learning of a classifier from partially labeled data. Last, we develop a method that uses graph neural networks to capture correlation between different categories to improve label prediction, and we use our model to predict missing labels. Section 1.2 explains our contribution to this direction in more details.

1.2 Contributions

This dissertation contributes to visual recognition in two main directions: modeling asynchronous time-series data and learning from partial labels. Following is a summary of our contributions, followed by sections for more details:

[3]https://www.mturk.com/

[4]https://crowdsource.google.com/imagelabeler/category

[5]https://www.google.com/recaptcha/

- **A Variational Auto-Encoder Model for Stochastic Point Process [92].** We study point processes under the prospective of deep generative models. Recently, deep generative models have achieved tremendous success in modeling complex real-world data distributions [40, 66, 106] in different applications such as images and videos. We propose a probabilistic generative model based on the framework of temporal point process for event sequences. In this work, we focus on modeling human activity sequences as an example of event sequence data. The model is termed the Action Point Process VAE (APP-VAE), a variational auto-encoder [66] that can capture the distribution over the times and categories of action sequences. Modeling the variety of possible action sequences is a challenge, which we show can be addressed via the APP-VAE's use of latent representations and non-linear functions to parameterize distributions over which event is likely to occur next in a sequence and at what time. We empirically validate the efficacy of APP-VAE on challenging human activity datasets.

- **A Flexible Flow-Based Latent Variable Model for Asynchronous Action Sequences [91]** We connect the fields of temporal point process and neural density estimation [43, 106]. We propose an intensity-free recurrent latent variable framework that directly models point process distribution by utilizing normalizing flows. This approach is capable of capturing highly complex temporal distributions and does not rely on restrictive parametric forms. Furthermore, with temporal latent variables, our model is also capable of capturing highly complex temporal dependence structures. In this work, we focus on modeling asynchronous human action sequences characterized by the time and type of actions. Comparisons with state-of-the-art baseline models on challenging real-life datasets show that the proposed framework is effective at modeling the stochasticity of discrete event sequences.

- **Learning Deep Networks with Partially Labeled Data [30].** In this work, we tackle the problem of learning deep networks. Deep networks have shown great performance for single-label image classification (e.g. ImageNet), but it is necessary to move beyond the single-label classification task because pictures of everyday life are inherently multi-label. Modeling multi-label images is a more difficult task than single-label because both the input and output spaces are more complex. Furthermore, collecting clean multi-label annotations is more difficult to scale-up than single-label annotations. To reduce the annotation cost, we propose to train a model with partial labels i.e. only some labels are known per image. We first empirically compare different labeling strategies and show the potential for using partial labels on multi-label datasets. Then to learn with partial labels, we introduce a new classification loss that exploits the proportion of known labels per image. Our approach allows the use of the same training settings as when learning with all the annotations. Experiments are performed on large-scale multi-label datasets.

1.2.1 A Variational Auto-Encoder Model for Stochastic Point Process

Anticipatory reasoning to model the evolution of action sequences over time is a fundamental challenge in human activity understanding. Human activities produce sequences of events data whose complex temporal dynamics need to be studied in order to be able to predict future activities, and is of many domains interests such as surveillance and security, health-care monitoring and etc. The crux of the problem in making predictions about the future is the fact that for interesting domains, the future is uncertain – given a history of actions, the distribution over future actions has substantial entropy.

Much of the work in this domain has focused on taking frame level data of video as input in order to predict the actions or activities that may occur in the immediate future [1, 63, 76, 90, 126]. Such frame-based approaches could be computationally inefficient and limit the model's ability to make long-term predictions. As motivated earlier, point process framework is a better fit for this problem, however, existing related work s [29, 144] make simplified assumption about the action timing distribution which limits the models expressiveness in modeling complex real-word distribution.

In this work, we propose a powerful generative approach that can effectively model the categorical and temporal variability comprising action sequences. The contributions of this work center around the APP-VAE (Action Point Process VAE), a novel generative model for asynchronous time action sequences. We formulated our model with the variational auto-encoder (VAE) paradigm, a powerful class of probabilistic models that facilitate generation and the ability to model complex distributions. We present a novel form of VAE for action sequences under a point process approach. As a generative model, APP-VAE can produce action sequences by sampling from a prior distribution, the parameters of which are updated based on neural networks that control the distributions over the next action type and its temporal occurrence. More specifically, in the generation phase, APP-VAE receives the history of past actions and predicts two distribution over the future action: one categorical distribution over the type of next action and one exponential distribution over its timing.

We empirically validate the efficacy of APP-VAE for modeling human action sequences on the MultiTHUMOS [139] and Breakfast [73] datasets. Experiments shows the efficiency and superior performance of APP-VAE in capturing the uncertainty inherent in tasks such as density estimation of action sequences, future action prediction and anomaly detection.

1.2.2 A Flexible Flow-Based Latent Variable Model for Asynchronous Action Sequences

Predicting action sequences of both *what* and *when* to happen is a fundamental inference task in human activity understanding. We argue that the challenge of this task stems from four aspects: 1) the complex dependence between the past and the future actions; 2) the complex structure of an action, *i.e*. how the timing of an action is related to its label or vice versa; 3) the multi-modal nature of future uncertainty, *i.e.* given a sequence of past actions, multiple different sequences of future events could be of substantial possibility; 4) the diverse and complex distributions of future action times.

Previous models, especially video frame-based ones, deal with the problem on a regularly spaced time grid with short intervals between time-stamps [63, 89, 126]. However, actions are usually sparsely and irregularly spaced in terms of time. Such frame-based setup could be computationally inefficient and limit the expressiveness of model in making long-term predictions across multiple actions. On the other side, in point process literature, existing approaches directly model the marginal distributions of future action time and action category by making oversimplified independence assumptions about the joint distribution. They also neglect the multi-modal possibility of the future [29, 114].

This work improves over APP-VAE proposed in Section 1.2.1. Although APP-VAE provides a more flexible framework compared to previous works, the model assumes that the time of the next action follows an exponential distribution which could restrict the ability in modeling complex time distributions.

We propose a recurrent latent variable model for action sequence generation and anticipation that directly addresses the challenges mentioned before. In our proposed framework, the stochastic latent variable encodes high-level information about possible future actions as well as how the future action times and categories are correlated. When this latent code is combined with the history of actions, it can be decoded into independent action category and time distributions that are consistent with the past actions. Unlike existing approaches that rely on restrictive parametric distributions over action timing, our approach makes use of the normalizing flow to generate flexible distributions of event times. More specifically, we learn a distribution over the action timing by transforming a simple base probability density through continuous normalizing flow, *i.e.* a series of invertible transformations. Furthermore, with temporal latent variables, our model is also capable of capturing highly complex temporal dependence structures. The proposed model is trained in a variational filtering framework; it uses a separate inference network to propose the posterior distribution of the latent variable conditioned on current observations and maximizes a variational lower bound.

The proposed model is evaluated on benchmark action sequence datasets of MultiTHUMOS [139] and Breakfast dataset [73]. The recurrent latent variable model achieves state-of-the-art performance on various tasks including density estimation, short-term prediction, and long-term conditional generation.

1.2.3 Learning Deep Networks with Partially Labeled Data

Recently, Stock and Cisse [116] presented empirical evidence that the performance of state-of-the-art classifiers on ImageNet [109] is largely underestimated – much of the remaining error is due to the fact that ImageNet's single-label annotation ignores the intrinsic multi-label nature of the images. Unlike ImageNet, multi-label datasets (*e.g.* MS COCO [82], Open Images [75]) contain more complex images that represent scenes with several objects. However, collecting multi-label annotations is more difficult to scale-up than single-label annotations [25].

To reduce the annotation cost, as an alternative strategy to fully annotation, one can make use of partial labels; collecting partial labels is easy and scalable with crowd-sourcing platforms. In the first part of this work, we study the problem of learning a multi-label classifier with partial labels. We empirically compare different labeling strategies to show the potential for using partial labels on multi-label datasets. Then to learn with partial labels, we introduce a loss function that generalizes the standard binary cross-entropy loss by exploiting label proportion information. Our approach allows the use of the same training settings as when learning with all the annotations. We further explore several curriculum learning based strategies to predict missing labels. Experiments are performed on three large-scale multi-label datasets: MS COCO, NUS-WIDE and Open Images.

Chapter 2

Background and Related Works

2.1 Event Analysis in Asynchronous Time-Series Data

2.1.1 Temporal Point Process

Temporal point processes (TPPs) [22] provide us with an elegant and effective mathematical framework for modeling event sequences data. A TPP is a stochastic process whose realization is a sequence of discrete events in time $t_{1:n} = (t_1, t_2, ..., t_n)$, where $t_i \in \mathbb{R}_{\geq 0}$ is the time when the i^{th} event occurs. Similarly, the realization could be a sequence of inter-arrival times $\tau_{1:n} = (\tau_1, \tau_2, ..., \tau_n)$, where inter-arrival time τ_i indicates the time difference between the starting time of two consecutive events t_i and t_{i-1}. Figure 2.1 shows these quantities. In this report, we assume that all sequences are simple such that no events coincide *i.e.* the absolute time of events are strictly ordered in time $t_i > t_{i-1}$ and $\tau_i \neq 0$.

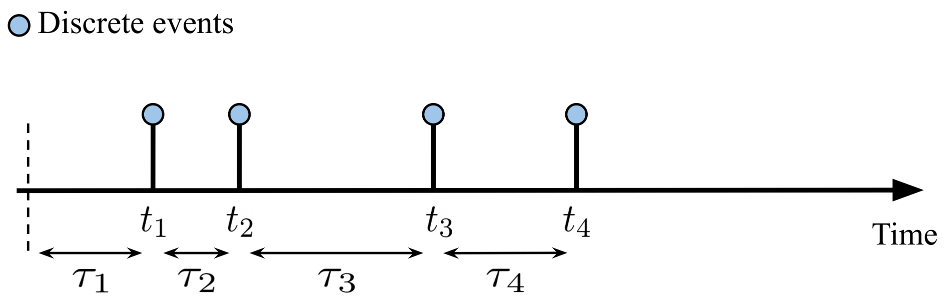


Figure 2.1: Here, we can see a sequence of discrete events $t_{1:4}$ where each t_i shows the absolute time when the i-th event happens and inter-arrival time τ_i shows the time gap between events t_i and t_{i-1}.

Equivalently, a temporal point process can be also represented as a counting process $N(t)$, which defines the number of events that happened in the interval $(0, t)$. Having this representation, we can obtain the absolute time of events by keeping track of the times at which there is an increase in the counting process.

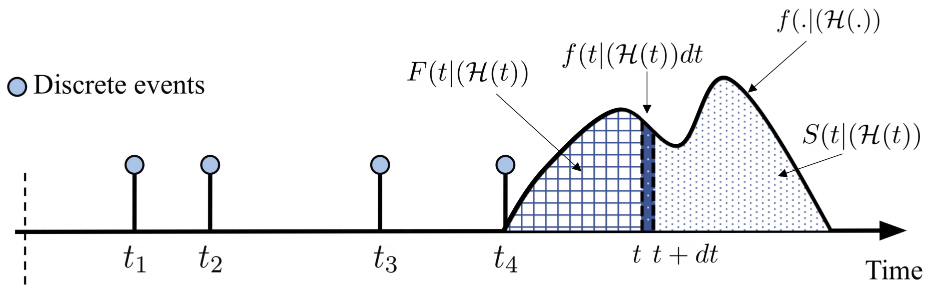


Figure 2.2: Here we can see the three quantities used for defining the intensity function of a temporal point process. $f(t|\mathcal{H}(t))$ is the probability density function for time-step t_5 given the history $\mathcal{H}(t) = (t_1, t_2, t_3, t_4)$. Quantities $F(t|\mathcal{H}(t))$ and $S(t|\mathcal{H}(t))$ are the corresponding cumulative and survival functions respectively.

In general, the choice of any of absolute timing, inter-arrival time, or counting process representation is based on the problem of interest. Each representation has a different probability distribution associated to it's random variable. If one representation has a distribution that is complicated and hard to work with, one can simply transform to the other two representations without any loss of information. In this report, we use these representations interchangeably to define a point process.

2.1.1.1 Conditional Intensity

A temporal point process can be defined by specifying distributions of each inter-arrival time $f(\tau_i)$ in the sequence $\tau_{1:n}$. The simplest case could be Poisson process [69] which assumes that in a sequence, inter-arrival times are independent, and each $f(\tau_i)$ follows an exponential distribution with a constant rate parameter λ:

$$f(\tau) = \lambda \exp(-\lambda\tau) \tag{2.1}$$

In Possion process, the density function f does not depend on history in the sequence. Each event in the sequence might depend on previous events in a very complex way, so its important to take the history information into consideration. In general, it is hard to build intuition on how to design density f to be history-dependent. Alternatively, the conditional intensity function provides a more intuitive and easier way to design a history-dependent process. Conditional intensity $\lambda(t|\mathcal{H}(t))$, also known as hazard function, is a popular way of characterizing a temporal point process. Consider modeling the i-th event in the sequence, given the past history of all events that happened before; for this case, the conditional intensity is defined as [105]:

$$\lambda(t|\mathcal{H}(t)) = \frac{f(t|\mathcal{H}(t))}{1 - F(t|\mathcal{H}(t))} \tag{2.2}$$

where history $\mathcal{H}(t) = (t_1, t_2, ..., t_{i-1})$ is the ordered sequence of all events that happened before time t with $t_1 < t_2 < \cdots < t_{i-1} < t$, and $f(t|\mathcal{H}(t))$ is the probability density function of the i-th event time given history of past event in the sequence and $F(t|\mathcal{H}(t))$ is its corresponding cumulative function. In point process literature, the term $1-F(t|\mathcal{H}(t))$ is known as survival function $S(t|\mathcal{H}(t)) = 1 - F(t|\mathcal{H}(t))$, which shows the probability that the next event will not happen before time t. Figure 2.2 shows these three quantities.

Conditional intensity function can be interpreted as the expected rate of events happening in an infinitisemal area $[t, t + dt)$:

$$
\begin{aligned}
\lambda(t|\mathcal{H}(t))dt &= \frac{f(t|\mathcal{H}(t))dt}{1 - F(t|\mathcal{H}(t))} && (2.3)\\
&= \frac{\mathbb{P}(t_i \in [t, t+dt)|\mathcal{H}(t))}{\mathbb{P}(t_i \notin (t_{i-1}, t)|\mathcal{H}(t))} && (2.4)\\
&= \frac{\mathbb{P}(t_i \in [t, t+dt), t_i \notin (t_{i-1}, t)|\mathcal{H}(t))}{\mathbb{P}(t_i \notin (t_{i-1}, t)|\mathcal{H}(t))} && (2.5)\\
&= \mathbb{P}(t_i \in [t, t+dt)|t_i \notin (t_{i-1}, t), \mathcal{H}(t)) && (2.6)\\
&= \mathbb{P}\{\text{event in } [t, t+dt)|\mathcal{H}(t)\} && (2.7)\\
&= \mathbb{E}[N(t+dt) - N(t)|\mathcal{H}(t)] && (2.8)
\end{aligned}
$$

With this interpretation, it is easier to build intuition on how to design a functional form that depends on history, rather than directly working with the probability density function f. Indeed, given the intensity defined as Equation 2.2, we can write the probability density function f with the following proposition:

Proposition 1 ([105]). *The reverse relation of (2.2) is given by*

$$f(t|\mathcal{H}(t)) = \lambda(t|\mathcal{H}(t)) \exp\left\{-\int_{t_{i-1}}^{t} \lambda(u|\mathcal{H}(u)) \ \mathrm{d}u\right\}.$$

Proof. By 2.2, we get that

$$
\begin{aligned}
\lambda(t|\mathcal{H}(t)) &= \frac{f(t|\mathcal{H}(t))}{1 - F(t|\mathcal{H}(t))} && (2.9)\\
&= \frac{\frac{dt}{d} F(t|\mathcal{H}(t))}{1 - F(t|\mathcal{H}(t))} && (2.10)\\
&= -\frac{dt}{d} \log(1 - F(t|\mathcal{H}(t))). && (2.11)
\end{aligned}
$$

By the fundamental theorem of calculus and integrating both sides, we obtain

$$
\begin{aligned}
\int_{t_{i-1}}^{t} \lambda(s|\mathcal{H}(s))ds &= -(\log(1 - F(t|\mathcal{H}(t)))) - \log(1 - F(t_{i-1}|\mathcal{H}(t))) && (2.12)\\
&= -(\log(1 - F(t|\mathcal{H}(t)))), && (2.13)
\end{aligned}
$$

since $F(t_{i-1}|\mathcal{H}(t)) = 0$ (point $t_i = t_{i-1}$ with probability zero, since the sequences are simple; no events coincide). By isolating $F(t|H(t))$ we get

$$F(t|H(t)) = 1 - \exp(-\int_{t_{i-1}}^{t} \lambda(s|\mathcal{H}(s))ds) \tag{2.14}$$

Then by differentiating $F(t|H(t))$ with respect to t and using the fundamental theorem of calculus, we get the claimed equality. □

2.1.2 Mark Temporal Point Process

In temporal point process, event sequences are characterized by the time they happen. These events usually come with other characteristics such as type of event, actor of event, etc. This information is known as *mark* and is modeled with the framework of *mark temporal point process*.

Formally, a mark temporal point process [22] is a stochastic process whose realization is a sequence of events $x_{1:n} = (x_1, \ldots, x_n)$ where each event $x_i = (t_i, y_i)$ is represented by the time it happens t_i and well as the mark y_i. Mark temporal point process models the underlying distribution of events' times as well as the underlying distribution of events' marks. Similar to TPPs, the distribution over event timing is usually characterized by the intensity function, which could be a function of mark data as well. The choice of modeling mark data is application dependent, *e.g.* if mark represents the type of event with a finite set of possible values *i.e.* $y_i \in 1, 2, ..., K$ (K discrete event categories), it could be modeled as multinomial distribution.

2.1.3 Learning

Point process models are usually optimized by maximizing the likelihood of observed sequences under the point process distribution. The likelihood function L is the joint probability density function of observed sequence $f(t_1, t_2, ..., t_n)$ which can be decomposed into all the conditional probability densities of each event given past history:

$$\begin{aligned} L(\theta) &= f_\theta(t_1, t_2, ..., t_n) && (2.15)\\ &= \prod_i f_\theta(t_i|t_1, t_2, ..., t_{i-1}) && (2.16)\\ &= \prod_i f_\theta(t_i|\mathcal{H}(t_i)) && (2.17)\\ &= \prod_i \lambda_\theta(t_i|\mathcal{H}(t_i)) \exp\left\{-\int_{t_{i-1}}^{t_i} \lambda_\theta(u|\mathcal{H}(u))\ \mathrm{d}u\right\}. && (2.18) \end{aligned}$$

where θ is the set of point process parameters.

For mark temporal point processes, the likelihood function depends on how we factorize the joint probability density of $f(t_i, y_i|\mathcal{H}(t_i))$. One simple factorization could be assuming that time

t_i and mark y_i are conditionally independent given the past history $\mathcal{H}(t_i)$. In this case, given the observed sequence $(t_i, y_i)_1^n = (t_1, y_1), (t_2, y_2), ..., (t_n, y_n)$, the likelihood is defined as:

$$\begin{aligned} L(\theta) &= f_\theta((t_1, y_1), (t_2, y_2), ..., (t_n, y_n)) && (2.19) \\ &= \prod_i f_\theta(t_i, y_i | (t_1, y_1), (t_2, y_2), ..., (t_{i-1}, y_{i-1})) && (2.20) \\ &= \prod_i f_\theta(t_i, y_i | \mathcal{H}(t_i)) && (2.21) \\ &= \prod_i f_\theta(t_i | \mathcal{H}(t_i)) f_\theta(y_i | \mathcal{H}(t_i)) && (2.22) \end{aligned}$$

2.1.4 Basic intensity functions

Although temporal point process has shown to be useful in modeling events sequences, it is usually not trivial to come up with a simple yet flexible intensity function. Various works explored different design choices of intensity function to capture the phenomena of interest. Here we review some popular hand-crafted design choices:

- **Poisson Process.** Poisson process [69] is based on the assumption that events happen independent of each other where the intensity is a fixed positive constant:

  $$\lambda(t) = \lambda > 0, \tag{2.23}$$

  In a more general case, λ could be a function of time $\lambda(t|\mathcal{H}(t)) = \lambda(t)$, but still independent of other events, which is called inhomogeneous Poisson process.

- **Self-exciting Process (Hawkes process).** Self-exciting process [46] assumes that occurrence of an event increases the probability of other events happening in near future. Here, the intensity function has the functional form of:

  $$\lambda(t|\mathcal{H}(t)) = \mu + \alpha \sum_{t_i < t} \exp(-(t - t_i)), \tag{2.24}$$

  where μ and α are positive constants and $t_i < t$ are all the events happening before time t.

- **Self-correcting Process.** In contrast to Hawkes process, self-correcting process [54] is based on the idea that occurrence of an event decreases the probability of an event happening in near future, and as time goes this probability will increase. The intensity function of self-correcting process has the following form:

  $$\lambda(t|H(t)) = \exp(\mu t - \sum_{t_i < t} \alpha), \tag{2.25}$$

  where μ and α are positive constants and $t_i < t$ are all the events happening before time t.

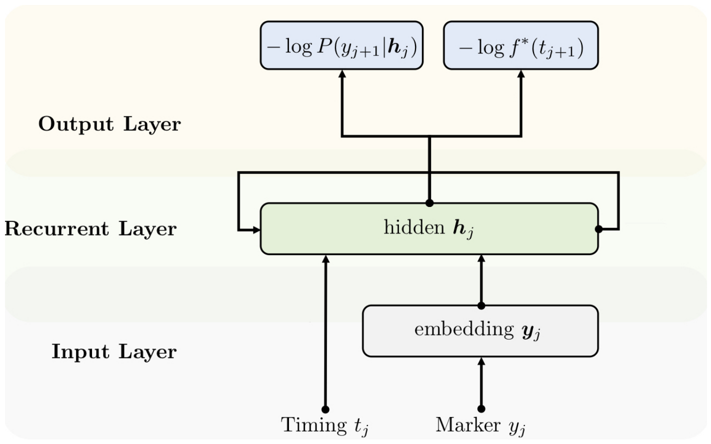

$-\log P(y_{j+1}|\boldsymbol{h}_j)$
$-\log f^*(t_{j+1})$
Output Layer
Recurrent Layer
hidden $\boldsymbol{h}_j$
embedding $\boldsymbol{y}_j$
Input Layer
Timing t_j
Marker y_j

Figure 2.3: From [29]. "Architect of RMTPP. For a given sequence $S = ((tj, yj)_{j=1}^{n})$, at the j-th event, the marker y_j is first embedded into a latent space. Then, the embedded vector and the temporal features are fed into the recurrent layer. The recurrent layer learns a representation that summaries the nonlinear dependency over the previous events. Based on the learned representation h_j, it outputs the prediction for the next marker y_{j+1} and timing t_{j+1} to calculate the respective loss functions"

The key contribution of these models is to find a functional form of intensity that fits data distribution well by making various parametric assumptions on the underlying generative process of the data. Although shown effective in modeling simple synthetic datasets, these strong parametric assumptions make such frameworks lack the flexibility to model the generative process for real-life and complex data, hindering wider adoption of TPP-based frameworks. In general, these types of strong assumptions might not be even true in many real-world scenarios and are only applicable to the cases where there exists a prior knowledge over the underlying generative process of the system.

2.1.5 Toward Deep Learning Approaches

Recently, learning the intensity function using recurrent neural networks (RNNs) to encode history information has received an increasing amount of attention [29, 60, 93, 134, 144]. In this line of work, history information is encoded by utilizing recurrent neural networks and exploited in learning the intensity of the point process distribution. Here, we review [29] known as RMTPP in detail as an example representing this line of work.

Du et al. [29] proposed a recurrent temporal model for learning the next event timing and mark distributions given the history of previous events. In their model, an RNN learns a non-linear map of history to the intensity function parameters of a marked temporal point process. Figure 2.3 shows the general architecture of their proposed model. At time-step j, RNN takes timing t_j and mark y_j and outputs the parameters of the next event time and mark distribution. For the time distribution, the output parameters define the intensity function:

$$\lambda(t|\mathcal{H}(t)) = \exp(\alpha.h_j + \beta(t - t_j) + b) \tag{2.26}$$

where h_j is the hidden state of the RNN at time-step j, and α, β, b are the model parameters which control the effect of past history, the current input and the prior in the intensity respectively.

For the event's mark distribution, the output parameters define a multi-nomial distribution i.e. the probability of occurrence of each action category (mark):

$$p(y_{j+1} = k|\mathcal{H}(t)) = p_k(h_j) \quad \text{and} \quad \sum_{k=1}^{K} p_k(h_j) = 1 \tag{2.27}$$

RNN and intensity function parameters are jointly learned by maximizing the log-likelihood of observed sequences under the predicted distributions.

In this line of literature, the explicit assumption on the forms of dependency over history is relaxed. However, the maximum likelihood training criteria on these models still require the intensity function to be simple for the likelihood to be tractable.

Lately, Omi et al. [96] generalizes previous RNN-based approaches by modeling the evolution of intensity function using recurrent neural network. Despite previous works where RNN was used only for representing the effect of history in the intensity function, here, the whole intensity function is modeled by RNNs. In this approach, first, the integral of the intensity function (cumulative intensity function) is learned using neural networks:

$$\Lambda(\tau|h_i) = \int_0^{\tau} \lambda(s|h_i)ds. \tag{2.28}$$

where h_i is an encoding of the history $\mathcal{H}(i)$ at time-step i given by an RNN. Then, the intensity function is obtained by differentiating the neural network:

$$\lambda(\tau|h_i) = \frac{\partial \Lambda(\tau|h_i)}{\partial \tau}. \tag{2.29}$$

In this approach, the intensity function could be highly complex while the likelihood stays tractable. But this formulation comes with a drawback of a costly sampling from the the probability density function $f(t|\mathcal{H}(t))$, because of not having an analytical form for the intensity.[1]

Recently, Chen et al. [13] has opened up a new prospective in modeling temporal sequences by introducing Neural Ordinary Differential Equations (Neural ODEs) that models the continuous flow of a sequence in time. With Neural ODEs, each sequence $t_0, t_1, t_2, ..., t_n$ is represented by a latent trajectory $z_{t_0}, z_{t_1}, z_{t_2}, ..., z_{t_n}$, where the continuous transformation of latent state z_{t_i} over time is

[1]They need to apply an iterative root finding algorithm to get a sample from the probability density function $f(t|\mathcal{H}(t))$.

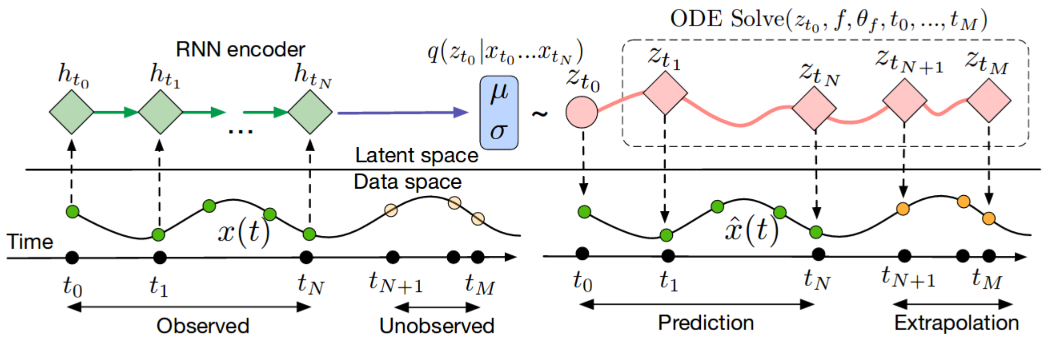


Figure 2.4: From [13]. "Computation graph of the latent ODE model". Here, h_{t_i} represents the hidden state of encoder RNN at time-step i.

modeled as an ODE flow parameterized by a neural network f:

$$z_{t_0} \sim p(z_0) \tag{2.30}$$

$$z_{t_1}, z_{t_2}, ..., z_{t_n} = \text{ODESolve}(z_{t_0}, f, \theta_f, t_0, .., t_n) \tag{2.31}$$

$$\frac{\partial z(t)}{\partial t} = f(z(t), \theta_f) \tag{2.32}$$

where θ_f shows the parameters of f. Figure 2.4 shows the computational graph of their proposed latent ODE model. They embed their proposed latent ODE model in a variational autoencoder framework [66] where the posterior network (parametrized by ϕ) takes a sequence of asynchronous observation $x_1, x_2, ..., x_N$ backward in time and output the latent distribution $q_\phi(z_{t_0}|x_1, x_2, ..., x_N)$. Then, a sample is drawn from the posterior distribution $z_{t_0} \sim q_\phi(z_{t_0}|x_1, .., x_N)$ and latent trajectory $z_{t_1}, z_{t_2}, ..., z_{t_n}$ is obtained by ODESolve$(z_{t_0}, f, \theta_f, t_0, .., t_n, ..., t_M)$. For time-step i in the sequence, the latent state z_{t_i} is passed to the decoder and it outputs $x_i \sim p(x_i|z_{t_i}, \theta_x)$.

They also model intensity of an inhomogeneous Poisson process as a function of latent state, where the likelihood of a sequence of observations in an interval $[t_{start}, t_{end}]$ is as follow:

$$\log p(t_1, ..., t_n|t_{start}, t_{end}) = \sum_{i=1}^{N} \log \lambda(z_{t_i}) - \int_{t_{start}}^{t_{end}} \lambda(z(t))dt \tag{2.33}$$

Most recently, Rubanova et al. [108] extended this approach by replacing the RNNs in the posterior network of Chen et al. [13] model with ODE-RNNs. Unlike standard RNNs where hidden states are constant between observations, they introduced ODE-RNNs where the hidden states follow a complex trajectory learned by Neural ODEs. Comparison of RNNs vs ODE-RNNs is shown in Figure 2.5. In ODE-RNNs, the hidden states evolve according to an ODE between observations instead of being fixed. In other words, it takes the time-gaps between observations into account, which would be beneficial when working with irregularly sampled time-series data.

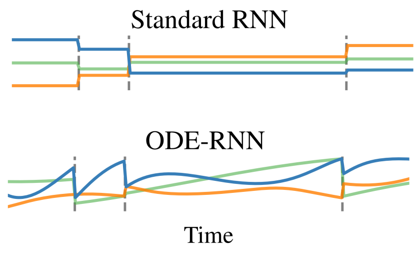


Figure 2.5: From [108]. ODE-RNN compared to standard RNN. "Standard RNNs have constant or undefined hidden states between observations. States of Neural ODE follow a complex trajectory but are determined by the initial state. The ODE-RNN model has states which obey an ODE between observations, and are also updated at observations"

Both models proposed by Chen et al. [13] and Rubanova et al. [108] lack the ability to model the effect of event's history over the dynamic of the system. In other words, the flow trajectory of the latent state does not depend on the history. Recent work by Jia and Benson [57] addresses this shortcoming by introducing Neural Jump Stochastic Differential Equations where the dynamic of the system is governed by a latent state which is history dependent. The system is described by a latent state $z(t)$, which has a continuous flow until an event happens; occurrence of an event makes an abrupt jump in the latent trajectory:

$$dz(t) = f(z(t), \theta) + w(z(t), \theta).dN(t) \tag{2.34}$$

$$\lambda(t) = \lambda(z(t), \theta_f) \tag{2.35}$$

where f controls the continuous flow of the system, w controls the effects of jumps, θ is the set of f and w parameters, and $N(t)$ is the number of events happened before time t. Although this work provides a nice design for incorporating history information in modeling point process sequences, the intensity predicted by the model is a constant, which might not be expressive enough in some cases.

2.1.6 Intensity-free Point Processes

In general, it might not be necessary to explicitly model the intensity when modeling point processes. In this line , a few works have tried to formulate TPP in an intensity-free manner.

WGANTPP [132, 133] introduced an intensity-free framework for modeling point processes using Wasserstein distance built upon a generative adversarial network (GAN). Two models are used to play a min-max game; a generator g_θ which samples from noise $\zeta \sim \mathbb{P}_z$ and tries to change it into a sample in a way to mimic real point process sequences and a discriminator f_w which tries to discriminate between real sequences and fake sequences generated by the generator. Both generator and discriminator are optimized using Wasserestein distance:

$$\min_{\theta} \max_{w \in \mathcal{W}, \|f_w\|_L \leq 1} \mathbb{E}_{\xi \sim \mathbb{P}_r}[f_w(\xi)] - \mathbb{E}_{\zeta \sim \mathbb{P}_z}[f_w(g_\theta(\zeta))] \tag{2.36}$$

where $\mathbb{P}_r$ denotes the real data distribution. It is worth noting that $\mathbb{P}_z$ is a Poisson process and the generator here, is applying a transformation is the space of counting measure.

In this line, RLPP [79] also proposed a new intensity-free framework exploring connections of reinforcement learning and temporal point processes. They formulate future event predictions as actions taken by an agent and the goal is to learn the policy that simulate a point process.

Both WGANTPP and RLPP models are optimized by trying to generate sequences of samples that are indistinguishable from the ground-truth sequences (by a discriminator in WGANTPP and policy learning in RLPP). Although these models are capable of generating realistic sequences, such training criteria results in an intractable likelihood.

2.2 Learning With Partial / Missing Labels.

Multi-label tasks often involve incomplete training data, hence several methods have been proposed to solve the problem of multi-label learning with missing labels (MLML). The first and simple approach is to treat the missing labels as negative labels [5, 88, 94, 118, 119, 128]. The MLML problem then becomes a fully labeled learning problem. This solution is used in most webly supervised approaches [88, 118]. The standard assumption is that only the category of the query is present (*e.g. car* in Figure 4.1) and all the other categories are absent. However, performance drops because a lot of ground-truth positive labels are initialized as negative labels [61]. A second solution is Binary Relevance (BR) [122], which treats each label as an independent binary classification. But this approach is not scalable when the number of categories grows and it ignores correlations between labels and between instances, which can be helpful for recognition. Unlike BR, our proposed approach allows to learn a single model using partial labels.

To overcome the second problem, several works proposed to exploit label correlations from the training data to propagate label information from the provided labels to missing labels. [8, 136] used a matrix completion algorithm to fill in missing labels. These methods exploit label-label correlations and instance-instance correlations with low-rank regularization on the label matrix to complete the instance-label matrix. Similarly, [141] introduced a low rank empirical risk minimization, [131] used a mixed graph to encode a network of label dependencies and [25, 94] learned correlation between the categories to predict some missing labels. Unlike most of the existing models that assume that the correlations are linear and unstructured, [137] proposed to learn structured semantic correlations. Another strategy is to treat missing labels as latent variables in probabilistic models. Missing labels are predicted by posterior inference. [62, 124] used models based on Bayesian networks [56] whereas [18] proposed a deep sequential generative model based on a Variational Auto-Encoder framework [66] that also allows to deal with unlabeled data.

However, most of these works cannot be used to learn a deep ConvNet. They require solving an optimization problem with the training set in memory, so it is not possible to use a mini-batch strategy to fine-tune the model. This is limiting because it is well-known that fine-tuning is important to transfer a pre-trained architecture [72]. Some methods are also not scalable because they require to solve convex quadratic optimization problems [131, 137] that are intractable for large-scale datasets. Unlike these methods, we propose a model that is scalable and end-to-end learnable. To train our model, we introduce a new loss function that adapts itself to the proportion of known labels per example. Similar to some MLML methods, we also explore several strategies to fill-in missing labels by using the learned classifier.

Learning with partial labels is different from semi-supervised learning [12] because in the semi-supervised learning setting, only a subset of the examples is labeled with all the labels and the other examples are unlabeled whereas in the partial labels setting, all the images are labeled but only with a subset of labels. Note that [21] also introduced a partially labeled learning problem (also called ambiguously labeled learning) but this problem is different: in [21], each example is annotated with multiple labels but only one is correct.

Chapter 3

A Variational Auto-Encoder Model for Stochastic Point Processes

We propose a novel probabilistic generative model for action sequences. The model is termed the Action Point Process VAE (APP-VAE), a variational auto-encoder that can capture the distribution over the times and categories of action sequences. Modeling the variety of possible action sequences is a challenge, which we show can be addressed via the APP-VAE's use of latent representations and non-linear functions to parameterize distributions over which event is likely to occur next in a sequence and at what time. We empirically validate the efficacy of APP-VAE for modeling action sequences on the MultiTHUMOS and Breakfast datasets.

This chapter was published as *A variational auto-encoder model for stochastic point processes* in the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019 [92].

3.1 Overview

Anticipatory reasoning to model the evolution of action sequences over time is a fundamental challenge in human activity understanding. The crux of the problem in making predictions about the future is the fact that for interesting domains, the future is uncertain – given a history of actions such as those depicted in Fig. 3.1, the distribution over future actions has substantial entropy.

In this work, we propose a powerful generative approach that can effectively model the categorical and temporal variability comprising action sequences. Much of the work in this domain has focused on taking frame level data of video as input in order to predict the actions or activities that may occur in the immediate future. There has also been recent interest on the task of predicting the sequence of actions that occur farther into the future [1, 29, 144].

Time series data often involves regularly spaced data points with interesting events occurring sparsely across time. This is true in case of videos where we have a regular frame rate but events of interest are present only in some frames that are infrequent. We hypothesize that in order to model future events in such a scenario, it is beneficial to consider the history of sparse events (action

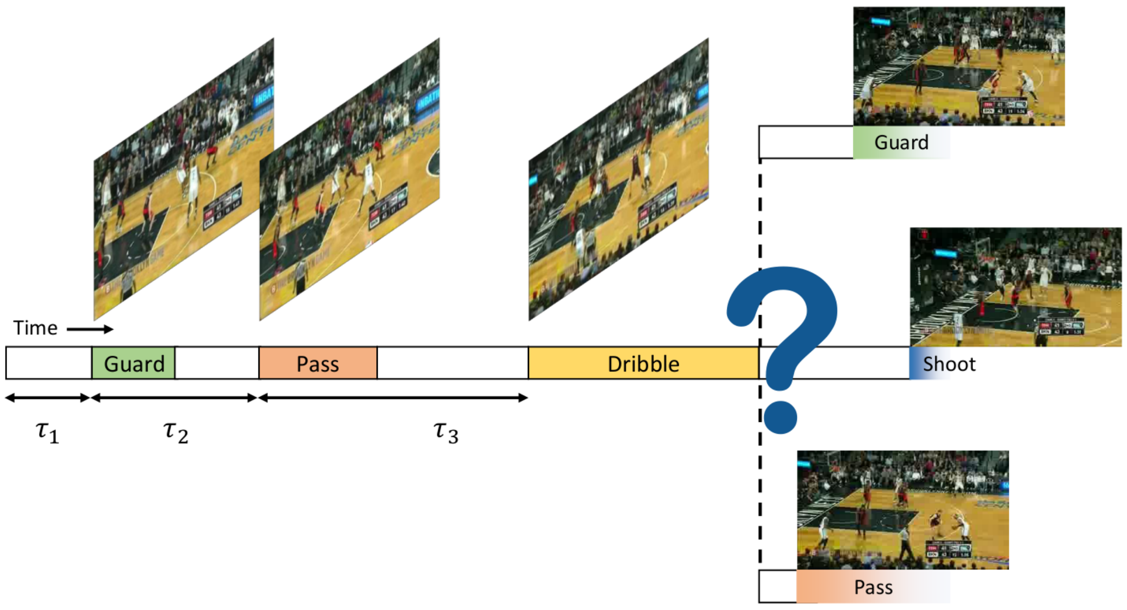


Figure 3.1: It is difficult to make predictions, especially about the future. Given a history of past actions, multiple actions are possible in the future. We focus on the problem of learning a distribution over the future actions – *what* are the possible action categories and *when* will they start.

categories and their temporal occurrence in the above example) alone, instead of regularly spaced frame data. While the history of frames contains rich information over and above the sparse event history, we can possibly create a model for future events occurring farther into the future by choosing to only model the sparse sequence of events. This approach also allows us to model high-level semantic meaning in the time series data that can be difficult to discern from low-level data points that are regular across time.

Figure 3.2 shows the overall structure of our proposed framework. Our model is formulated in the variational auto-encoder (VAE) [66] paradigm, a powerful class of probabilistic models that facilitate generation and the ability to model complex distributions. We present a novel form of VAE for action sequences under a point process approach. This approach has a number of advantages, including a probabilistic treatment of action sequences to allow for likelihood evaluation, generation, and anomaly detection.

Contribution.

The contributions of this work center around the APP-VAE (Action Point Process VAE), a novel generative model for asynchronous time action sequences. The contributions of this work include:

- A novel formulation for modeling point process data within the variational auto-encoder paradigm.
- Conditional prior models for encoding asynchronous time data.

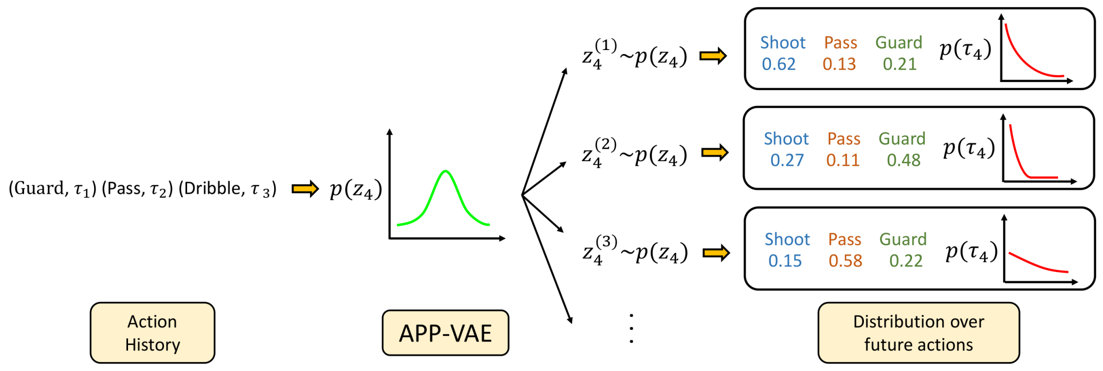

(Guard, τ_1) (Pass, τ_2) (Dribble, τ_3)
$p(z_4)$
$z_4^{(1)} \sim p(z_4)$
$z_4^{(2)} \sim p(z_4)$
$z_4^{(3)} \sim p(z_4)$
Shoot 0.62 Pass 0.13 Guard 0.21 $p(\tau_4)$
Shoot 0.27 Pass 0.11 Guard 0.48 $p(\tau_4)$
Shoot 0.15 Pass 0.58 Guard 0.22 $p(\tau_4)$
Action History
APP-VAE
Distribution over future actions

Figure 3.2: Given the history of actions, APP-VAE generates a distribution over possible actions in the next step. APP-VAE can recurrently perform this operation to model diverse sequences of actions that may follow. The figure shows the distributions for the fourth action in a basketball game given the history of first three actions.

- A probabilistic model for jointly capturing uncertainty in which actions will occur and when they will happen.

3.2 Related Work

Activity Prediction. Most activity prediction tasks are frame-based, *i.e*. the input to the model is a sequence of frames before the action starts and the task is predict what will happen next. Lan *et al.* [77] predict future actions from hierarchical representations of short clips by having different classifiers at each level in a max-margin framework. Mahmud *et al.* [90] jointly predicts future activity as well as its starting time by a multi-streams framework. Each streams tries to catch different features for having a richer feature representation for future prediction: One stream for visual information, one for previous activities and the last one focusing on the last activity.

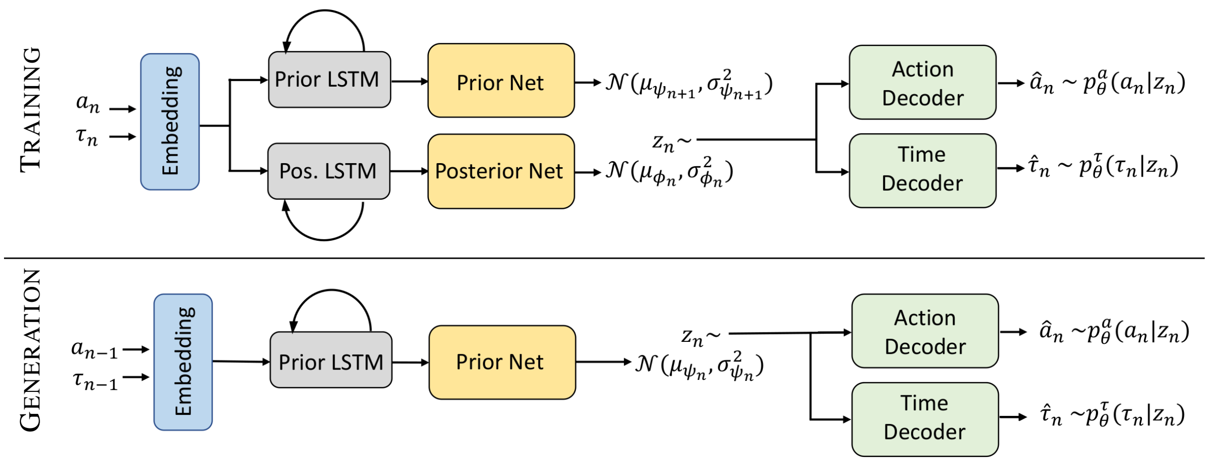
Farha *et al.* [1] proposed a framework for predicting the action categories of a sequence of future activities as well as their starting and ending time. They proposed two deterministic models, one using a combination of RNN and HMM and the other one is a CNN predicting a matrix which future actions are encoded in it.

Asynchronous Action Prediction. We focus on the task of predicting future action given a sequence of previous actions that are asynchronous in time. Du *et al.* [29] proposed a recurrent temporal model for learning the next activity timing and category given the history of previous actions. Their recurrent model learns a non-linear map of history to the intensity function of a temporal point process framework. Zhong *et al.* [144] also introduced a hierarchical recurrent network model for future action prediction for modeling future action timing and category. Their model takes frame-level information as well as sparse high-level events information in the history to learn the intensity function of a temporal point process. Xiao *et al.* [132] introduced an intensity-free generative method for temporal point process. The generative part of their model is an extension of Wasserstein GAN in the context of temporal point process for learning to generate sequences of action.

Early Stage Action Prediction. Our work is related to early stage action prediction. This task refers to predicting the action given the initial frames of the activity [50, 87, 115]. Our task is different from early action prediction, because the model doesn't have any information about the action while predicting it. Recently Yu *et al.* [142] used variational auto-encoder to learn from the frames in the history and transfer them into the future. Sadegh Aliakbarian *et al.* [110] combine context and action information using a multi-stage LSTM model to predict future action. The model is trained with a loss function which encourages the model to predict action with few observations. Gao *et al.* [38] proposed to use a Reinforced Encoder-Decoder network for future activity prediction. Damen *et al.* [7] proposed a semi-supervised variational recurrent neural network to model human activity including classification, prediction, detection and anticipation of human activities.

Video Prediction. Video prediction has recently been studied in several works. Denton and Fergus [26] use a variational auto-encoder framework with a learned prior to generate future video frames. He *et al.* [48] also proposed a generative model for future prediction. They structure the latent space by adding control features which makes the model able to control generation. Vondrick *et al.* [127] uses adversarial learning for generating videos of future with transforming the past pixels. Patraucean et al. [102] describe a spatio-temporal auto-encoder that predicts optical flow as a dense map, using reconstruction in its learning criterion. Villegas et al. [125] propose a hierarchical approach to pixel-level video generation, reasoning over body pose before rendering into a predicted future frame.

3.3 Asynchronous Action Sequence Modeling



Training
a_n
τ_n
Embedding
Prior LSTM
Pos. LSTM
Prior Net
Posterior Net
$\mathcal{N}(\mu_{\psi_{n+1}}, \sigma^2_{\psi_{n+1}})$
$z_n \sim \mathcal{N}(\mu_{\phi_n}, \sigma^2_{\phi_n})$
Action Decoder
Time Decoder
$\hat{a}_n \sim p^a_\theta(a_n|z_n)$
$\hat{\tau}_n \sim p^\tau_\theta(\tau_n|z_n)$
Generation
a_{n-1}
τ_{n-1}
Embedding
Prior LSTM
Prior Net
$z_n \sim \mathcal{N}(\mu_{\psi_n}, \sigma^2_{\psi_n})$
Action Decoder
Time Decoder
$\hat{a}_n \sim p^a_\theta(a_n|z_n)$
$\hat{\tau}_n \sim p^\tau_\theta(\tau_n|z_n)$

Figure 3.3: Our proposed recurrent VAE model for asynchronous action sequence modeling. At each time step, the model uses the history of actions and inter-arrival times to generate a distribution over latent codes, a sample of which is then decoded into two probability distributions for the next action: one over possible action labels and one over the inter arrival time.

We first introduce some notations and the problem definition. Then we review the VAE model and temporal point process that are used in our model. Subsequently, we present our model in detail and how it is trained.

Problem definition. The input is a sequence of actions $x_{1:n} = (x_1, \dots, x_n)$ where x_n is the n-th action. The action $x_n = (a_n, \tau_n)$ is represented by the action category $a_n \in \{1, 2, \dots, K\}$ (K discrete action classes) and the inter-arrival time $\tau_n \in \mathbb{R}^+$. The inter-arrival time is the difference between the starting time of action x_{n-1} and x_n. We formulate the asynchronous action distribution modeling task as follows: given a sequence of actions $x_{1:n-1}$, the goal is to produce a distribution over what action a_n will happen next, and the inter arrival time τ_n. We aim to develop probabilistic models to capture the uncertainty over these what and when questions of action sequence modeling.

3.3.1 Background: Base Models

Variational Auto-Encoders (VAEs). A VAE [66] describes a generative process with simple prior $p_\theta(z)$ (usually chosen to be a multivariate Gaussian) and complex likelihood $p_\theta(x|z)$ (the parameters of which are produced by neural networks). x and z are observed and latent variables, respectively. Approximating the intractable posterior $p_\theta(z|x)$ with a recognition neural network $q_\phi(z|x)$, the parameters of the generative model θ as well as the recognition model ϕ can be jointly optimized by maximizing the evidence lower bound $\mathcal{L}$ on the marginal likelihood $p_\theta(x)$:

$$\begin{aligned}\log p_\theta(x) &= \mathrm{KL}(q_\phi \| p_\theta) + \mathcal{L}(\theta, \phi) \\ &\geq \mathcal{L}(\theta, \phi) = -\mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(z|x)}{p_\theta(z,x)}\right].\end{aligned} \tag{3.1}$$

Recent works expand VAEs to time-series data including video [2, 26, 48], text [20, 52], or audio [140]. A popular design choice of such models is the integration of a per time-step VAE with RNN/LSTM temporal modelling. The ELBO thus becomes a summation of time-step-wise variational lower bound[1]:

$$\begin{aligned}\mathcal{L}(\theta, \phi, \psi) \;&=\; \sum_{n=1}^{N} \Big[\mathbb{E}_{q_\phi(z_{1:n}|x_{1:n})}\left[\log p_\theta(x_n|x_{1:n-1}, z_{1:n})\right] \\ &-\; \mathrm{KL}(q_\phi(z_n|x_{1:n})||p_\psi(z_n|x_{1:n-1}))\Big].\end{aligned} \tag{3.2}$$

with a “prior" $p_\psi(z_n|x_{1:n-1})$ that evolves over the N time steps used.

Temporal point process. A temporal point process is a stochastic model used to capture the inter-arrival times of a series of events. A temporal point process is characterized by the conditional intensity function $\lambda(\tau_n|x_{1:n-1})$, which is conditioned on the past events $x_{1:n-1}$ (*e.g.* action in this work). The conditional intensity encodes instantaneous probabilities at time τ. Given the history of $n-1$ past actions, the probability density function for the time of the next action is:

$$f(\tau_n|x_{1:n-1}) = \lambda(\tau_n|x_{1:n-1}) e^{-\int\limits_0^{\tau_n} \lambda(u|x_{1:n-1})\ \mathrm{d}u} \tag{3.3}$$

The Poisson process [69] is a popular temporal point process, which assumes that events occur independent of one another. The conditional intensity is $\lambda(\tau_n|x_{1:n-1}) = \lambda$ where λ is a positive constant. More complex conditional intensities have been proposed like Hawkes Process [46] and Self-Correcting Process [54]. All these conditional intensity function seek to capture some forms of dependency on the past action. However, in practice the true model of the dependencies is never known [93] and the performance depend on the design of the conditional intensity. In this work, we learn a recurrent model that estimates the conditional intensity based on the history of actions.

3.3.2 Proposed Approach

We propose a generative model for asynchronous action sequence modeling using the VAE framework. Figure 3.3 shows the architecture of our model. Overall, the input sequence of actions and inter arrival times are encoded using a recurrent VAE model. At each step, the model uses the history

[1]Note that variants exist, depending on the exact form of the recurrent structure and its VAE instantiation.

of actions to produce a distribution over latent codes z_n, a sample of which is then decoded into two probability distributions: one over the possible action categories and another over the inter-arrival time for the next action. We now detail our model.

Model. At time step n during training, the model takes as input the action x_n, which is the target of the prediction model, and the history of past actions $x_{1:n-1}$. These inputs are used to compute a conditional distribution $q_\phi(z_n|x_{1:n})$ from which a latent code z_n is sampled. Since the true distribution over latent variables z_n is intractable we rely on a time-dependent inference network $q_\phi(z_n|x_{1:n})$ that approximates it with a conditional Gaussian distribution $\mathcal{N}(\mu_{\phi_n}, \sigma^2_{\phi_n})$. To prevent z_n from just copying x_n, we force $q_\phi(z_n|x_{1:n})$ to be close to the prior distribution $p(z_n)$ using a KL-divergence term. Usually in VAE models, $p(z_n)$ is a fixed Gaussian $\mathcal{N}(0, I)$. But a drawback of using a fixed prior is that samples at each time step are drawn randomly, and thus ignore temporal dependencies present between actions. To overcome this problem, a solution is to learn a prior that varies across time, being a function of all past actions except the current action $p_\psi(z_{n+1}|x_{1:n})$. Both prior and approximate posterior are modelled as multivariate Gaussian distributions with diagonal covariance with parameters as shown below:

$$q_\phi(z_n|x_{1:n}) = \mathcal{N}(\mu_{\phi_n}, \sigma^2_{\phi_n}) \tag{3.4}$$

$$p_\psi(z_{n+1}|x_{1:n}) = \mathcal{N}(\mu_{\psi_{n+1}}, \sigma^2_{\psi_{n+1}}) \tag{3.5}$$

At step n, both posterior and prior networks observe actions $x_{1:n}$ but the posterior network outputs the parameters of a conditional Gaussian distribution for the current action x_n whereas the prior network outputs the parameters of a conditional Gaussian distribution for the next action x_{n+1}.

At each time-step during training, a latent variable z_n is drawn from the posterior distribution $q_\phi(z_n|x_{1:n})$. The output action $\hat{x}_n$ is then sampled from the distribution $p_\theta(x_n|z_n)$ of our conditional generative model which is parameterized by θ. For mathematical convenience, we assume the action category and inter-arrival time are conditionally independent given the latent code z_n:

$$p_\theta(x_n|z_n) = p_\theta(a_n, \tau_n|z_n) = p^a_\theta(a_n|z_n)p^\tau_\theta(\tau_n|z_n) \tag{3.6}$$

where $p^a_\theta(a_n|z_n)$ (resp. $p^\tau_\theta(\tau_n|z_n)$) is the conditional generative model for action category (resp. inter-arrival time). This is a standard assumption in event prediction [29, 144]. The sequence model generates two probability distributions: (i) a categorical distribution over the action categories and (ii) a temporal point process distribution over the inter-arrival times for the next action.

The distribution over action categories is modeled with a multinomial distribution when a_n can only take a finite number of values:

$$p^a_\theta(a_n = k|z_n) = p_k(z_n) \quad \text{and} \quad \sum_{k=1}^{K} p_k(z_n) = 1 \tag{3.7}$$

where $p_k(z_n)$ is the probability of occurrence of action k, and K is the total number of action categories.

The inter-arrival time is assumed to follow an exponential distribution parameterized by $\lambda(z_n)$, similar to a standard temporal point process model:

$$p_\theta^\tau(\tau_n|z_n) = \begin{cases} \lambda(z_n)e^{-\lambda(z_n)\tau_n} & \text{if } \tau_n \geq 0 \\ 0 & \text{if } \tau_n < 0 \end{cases} \tag{3.8}$$

where $p_\theta^\tau(\tau_n|z_n)$ is a probability density function over random variable τ_n and $\lambda(z_n)$ is the intensity of the process, which depends on the latent variable sample z_n.

Learning. We train the model by optimizing the variational lower bound over the entire sequence comprised of N steps:

$$\begin{aligned} \mathcal{L}_{\theta,\phi}(x_{1:N}) = \sum_{n=1}^{N} (&\mathbb{E}_{q_\phi(z_n|x_{1:n})}[\log p_\theta(x_n|z_n)] \\ &- D_{KL}(q_\phi(z_n|x_{1:n})||p_\psi(z_n|x_{1:n-1}))) \end{aligned} \tag{3.9}$$

Because the action category and inter-arrival time are conditionally independent given the latent code z_n, the log-likelihood term can be written as follows:

$$\begin{aligned} &\mathbb{E}_{q_\phi(z_n|x_{1:n})}[\log p_\theta(x_n|z_n)] = \\ &\mathbb{E}_{q_\phi(z_n|x_{1:n})}[\log p_\theta^a(a_n|z_n)] + \mathbb{E}_{q_\phi(z_n|x_{1:n})}[\log p_\theta^\tau(\tau_n|z_n)] \end{aligned} \tag{3.10}$$

Given the form of p_θ^a the log-likelihood term reduces to a cross entropy between the predicted action category distribution $p_\theta^a(a_n|z_n)$ and the ground truth label a_n^*. Given the ground truth inter-arrival time τ_n^*, we compute its log-likelihood over a small time interval Δ_τ under the predicted distribution.

$$\begin{aligned} \log\left[\int_{\tau_n^*}^{\tau_n^*+\Delta_\tau} p_\theta^\tau(\tau_n|z_n)\,\mathrm{d}\tau_n\right] = \log(1 - e^{-\lambda(z_n)\Delta_\tau}) \\ - \lambda(z_n)\tau_n^* \end{aligned} \tag{3.11}$$

We use the re-parameterization trick [66] to sample from the encoder network q_ϕ.

Generation. The goal is to generate the next action $\hat{x}_n = (\hat{a}_n, \hat{\tau}_n)$ given a sequence of past actions $x_{1:n-1}$. The generation process is shown on the bottom of Figure 3.3. At test time, an action at step n is generated by first sampling z_n from the prior. The parameters of the prior distribution are computed based on the past $n-1$ actions $x_{1:n-1}$. Then, an action category $\hat{a}_n$ and inter-arrival time $\hat{\tau}_n$ are generated as follows:

$$\hat{a}_n \sim p_\theta^a(a_n|z_n) \qquad\qquad \hat{\tau}_n \sim p_\theta^\tau(\tau_n|z_n) \tag{3.12}$$

Architecture. We now describe the architecture of our model in detail. At step n, the current action x_n is embedded into a vector representation x_n^{emb} with a two-step embedding strategy. First, we compute a representation for the action category (a_n) and the inter-arrival time (τ_n) separately. Then, we concatenate these two representations and compute a new representation x_n^{emb} of the action.

$$a_n^{emb} = f_a^{emb}(a_n) \qquad\qquad \tau_n^{emb} = f_\tau^{emb}(\tau_n) \tag{3.13}$$

$$x_n^{emb} = f_{a,\tau}^{emb}([a_n^{emb}, \tau_n^{emb}]) \tag{3.14}$$

We use a 1-hot encoding to represent the action category label a_n. Then, we have two branches: one to estimate the parameters of the posterior distribution and another to estimate the parameters of the prior distribution. The network architecture of these two branches is similar but we use separate networks because the prior and the posterior distribution capture different information. Each branch has a Long Short Term Memory (LSTM) [51] to encode the current action and the past actions into a vector representation:

$$h_n^{post} = LSTM_\phi(x_n^{emb}, h_{n-1}^{post}) \tag{3.15}$$

$$h_n^{prior} = LSTM_\psi(x_n^{emb}, h_{n-1}^{prior}) \tag{3.16}$$

Recurrent networks turn variable length sequences into meaningful, fixed-sized representations. The output of the posterior LSTM h_n^{post} (resp. prior LSTM h_n^{prior}) is passed into a posterior (also called inference) network f_ϕ^{post} (resp. prior network f_ψ^{prior}) that outputs the parameters of the Gaussian distribution:

$$\mu_{\phi_n}, \sigma_{\phi_n}^2 = f_\phi^{post}(h_n^{post}) \tag{3.17}$$

$$\mu_{\psi_n}, \sigma_{\psi_n}^2 = f_\psi^{prior}(h_n^{prior}) \tag{3.18}$$

Then, a latent variable z_n is sampled from the posterior (or prior during testing) distribution and is fed to the decoder networks for generating distributions over the action category a_n and inter-arrival time τ_n.

The decoder network for action category $f_\theta^a(z_n)$ is a multi-layer perceptron with a softmax output to generate the probability distribution in Eq. 3.7:

$$p_\theta^a(a_n|z_n) = f_\theta^a(z_n) \tag{3.19}$$

The decoder network for inter-arrival time $f_\theta^\tau(z_n)$ is another multi-layer perceptron, producing the parameter for the point process model for temporal distribution in Eq. 3.8:

$$\lambda(z_n) = f_\theta^\tau(z_n) \tag{3.20}$$

During training, the parameters of all the networks are jointly learned in an end-to-end fashion.

Dataset	Model	Stoch. Var.	LL
Breakfast	APP-LSTM	-	-6.668
	APP-VAE w/o Learned Prior	✓	$\geq$-9.427
	APP-VAE	✓	$\geq$**-5.944**
MultiTUHMOS	APP-LSTM	-	-4.190
	APP-VAE w/o Learned Prior	✓	$\geq$-5.344
	APP-VAE	✓	$\geq$**-3.838**

Table 3.1: Comparison of log-likelihood on Breakfast and MultiTHUMOS datasets.

3.4 Experiments

Datasets. We performed experiments using APP-VAE on two action recognition datasets. We use the standard training and testing sets for each.

MultiTHUMOS Dataset [139] is a challenging dataset for action recognition, containing 400 videos of 65 different actions. On average, there are 10.5 action class labels per video and 1.5 actions per frame.

Breakfast Dataset [73] contains 1712 videos of breakfast preparation for 48 action classes. The actions are performed by 52 people in 18 different kitchens.

Architecture details. The APP-VAE model architecture is shown in Fig. 3.3. Action category and inter-arrival time inputs are each passed through 2 layer MLPs with ReLU activation. They are then concatenated and followed with a linear layer. Hidden state of prior and posterior LSTMs is 128. Both prior and posterior networks are 2 layer MLPs, with ReLU activation after the first layer. Dimension of the latent code is 256. Action decoder is a 3 layer MLP with ReLU at the first two layers and softmax for the last one. The time decoder is also a 3 layer MLP with ReLU at the first two layers, with an exponential non-linearity applied to the output to ensure the parameter of the point process is positive.

Implementation details. The models are implemented with PyTorch [101] and are trained using the Adam [65] optimizer for 1,500 epochs with batch size 32 and learning rate 0.001. We split the standard training set of both datasets into training and validation sets containing 70% and 30% of samples respectively. We select the best model during training based on the model loss (Eq. 3.10) on the validation set.

Baselines. We compare APP-VAE with the following models for action prediction tasks.

- *Time Deterministic LSTM (TD-LSTM).* This is a vanilla LSTM model that is trained to predict the next action category and the inter-arrival time, comparable with the model proposed by Farha *et al.* [1]. This model directly predicts the inter-arrival time and not the distribution over it. TD-LSTM uses the same encoder network as APP-VAE. We use cross-entropy loss for action category output and perform regression over inter-arrival time using mean squared error (MSE) loss similar to [1].

- *Action Point Process LSTM (APP-LSTM)*. This baseline predicts the inter-arrival time distribution similar to APP-VAE. The model uses the same reconstruction loss function as in the VAE model – cross entropy loss for action category and negative log-likelihood (NLL) loss for inter-arrival time. APP-LSTM does not have the stochastic latent code that allows APP-VAE to model diverse distributions over action category and inter-arrival time. Our APP-LSTM baseline encompasses Du *et al.* [29]'s work. The only difference is the way we model the intensity function (IF). Du *et al.* [29] defines IS explicitly as a function of time. This design choice has been investigated in Zhong *et al.* [144]; an implicit intensity function is shown to be superior and thus adapted in our APP-LSTM baseline.

Metrics. We use log-likelihood (LL) to compare our model with the APP-LSTM. We also report accuracy of action category prediction and mean absolute error (MAE) of inter-arrival time prediction. We calculate accuracy by comparing the most probable action category from the model output with the ground truth category. To calculate MAE, we use the expected inter-arrival time under the predicted distribution $p_\theta^\tau(\tau_n|z_n)$:

$$\mathbb{E}_{p_\theta^\tau(\tau_n|z_n)}[\tau_n] = \int_0^\infty \tau_n \cdot p_\theta^\tau(\tau_n|z_n)\mathrm{d}\tau_n = \frac{1}{\lambda(z_n)} \tag{3.21}$$

The expected value $\frac{1}{\lambda(z_n)}$ and the ground truth inter-arrival time are used to compute MAE.

Dataset	Model	Time Loss	stoch. var.	↑ accuracy	↓ MAE
Breakfast	TD-LSTM	MSE	-	53.64	173.76
	APP-LSTM	NLL	-	61.39	152.17
	APP-VAE w/o Learned Prior	NLL	✓	27.09	270.75
	APP-VAE	NLL	✓	**62.20**	**142.65**
MultiTUHMOS	TD-LSTM	MSE	-	29.74	2.33
	APP-LSTM	NLL	-	36.31	1.99
	APP-VAE w/o Learned Prior	NLL	✓	8.79	2.02
	APP-VAE	NLL	✓	**39.30**	**1.89**

Table 3.2: Accuracy of action category prediction and Mean Absolute Error (MAE) of inter-arrival time prediction of all model variants. Arrows show whether lower (↓) or higher (↑) scores are better.

3.4.1 Experiment Results

We discuss quantitative and qualitative results from our experiments. All quantitative experiments are performed by teacher forcing methodology *i.e.* for each step in the sequence of actions, the

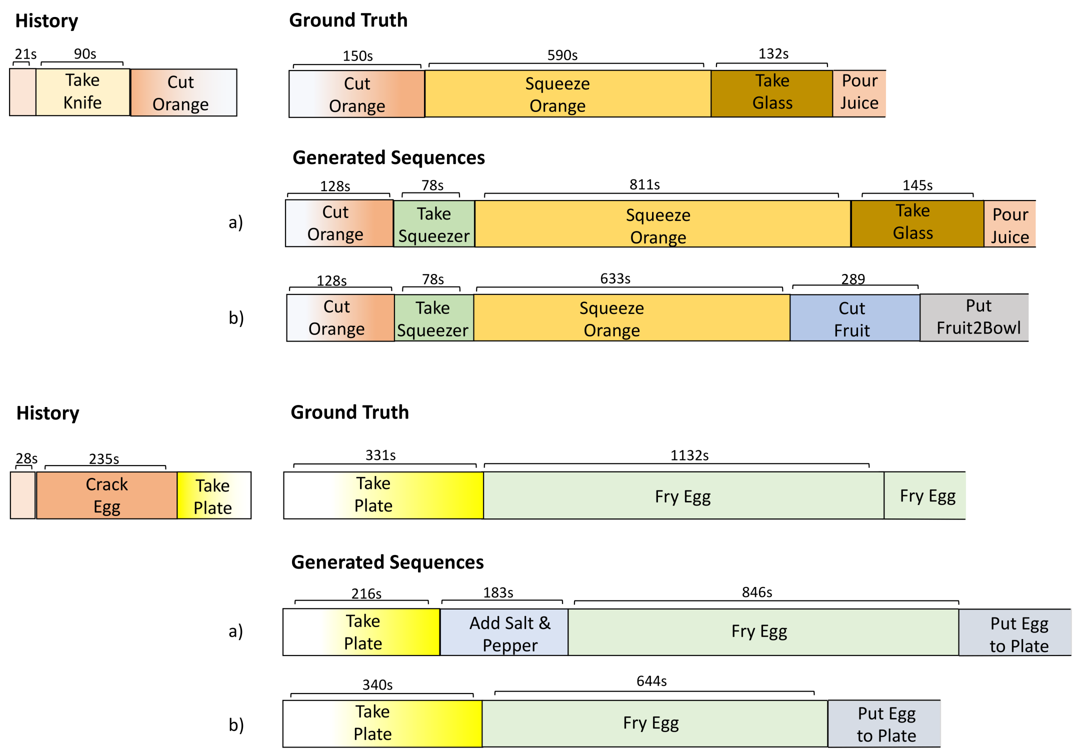


Figure 3.4: Examples of generated sequences. Given the history (shown at left), we generate a distribution over latent code z_n for the subsequent time step. A sample is drawn from this distribution, and decoded into distributions over action category and time, from which a next action/time pair by selecting the action with the highest probability and computing the expectation of the generated distribution over τ (Equation 3.21). This process is repeated to generate a sequence of actions. Two such sampled sequences (a) and (b) are shown for each history, and compared to the respective ground truth sequence (in line with history row). We can see that APP-VAE is capable of generating diverse and plausible action sequences.

models are fed the ground truth history of actions, and likelihood and/or other metrics for the next action are measured.

Quantitative results. Table 3.1 shows experimental results that compare APP-VAE with the APP-LSTM. To estimate the log-likelihood (LL) of our model, we draw 1500 samples from the approximate posterior distribution, following the standard approach of importance sampling. APP-VAE outperforms the APP-LSTM on both MultiTHUMOS and Breakfast datasets. We believe that this is because the APP-VAE model is better in modeling the complex distribution over future actions.

Table 3.2 shows accuracy and MAE in predicting the future action given the history of previous actions. APP-VAE outperforms TD-LSTM and APP-LSTM under both the metrics. For each step in the sequence we draw 1500 samples from the prior distribution that models the next step action. Given the output distributions, we select the action category with the maximum probability as the predicted action, and the expected value of inter-arrival time as the predicted inter-arrival time. Out

	Test sequences with high likelihood
1	NoHuman, CliffDiving, Diving, Jump, BodyRoll, CliffDiving, Diving, Jump, BodyRoll, CliffDiving, Diving, Jump, BodyRoll, BodyContract, Run, CliffDiving, Diving, Jump, ..., BodyRoll, CliffDiving, Diving, BodyContract, CliffDiving, Diving, CliffDiving, Diving, CliffDiving, Diving, Jump, CliffDiving, Diving, Walk, Run, Jump, Jump, Run, Jump
2	CleanAndJerk, PickUp, BodyContract, Squat, StandUp, BodyContract, Squat, CleanAndJerk, PickUp, StandUp, BodyContract, Squat, CleanAndJerk, PickUp, StandUp, Drop, BodyContract, Squat, PickUp, ..., Squat, StandUp, Drop, BodyContract, Squat, BodyContract, Squat, BodyContract, Squat, BodyContract, Squat, BodyContract, Squat, NoHuman
	Test sequences with low likelihood
1	NoHuman, TalkToCamera, GolfSwing, GolfSwing, GolfSwing, GolfSwing, NoHuman
2	NoHuman, HammerThrow, TalkToCamera, CloseUpTalkToCamera, HammerThrow, HammerThrow, HammerThrow, TalkToCamera, ..., HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow, HammerThrow

Table 3.3: Example of test sequences with high and low likelihood according to our learned model

Dataset	Model	Acc	MAE
Breakfast	APP-VAE - avg	59.02	145.95
	APP-VAE - mode	62.20	142.65
MultiTUHMOS	APP-VAE - avg	35.23	1.96
	APP-VAE - mode	39.30	1.89

Table 3.4: Accuracy (Acc) and Mean Absolute Error (MAE) under mode and averaging over samples.

of 1500 predictions, we select the most frequent action as the model prediction for that time step, and compute inter-arrival time by averaging over the corresponding time values.

Table 3.1 and 3.2 also show the comparison of our model with the case where the prior is fixed in all of the time-steps. In this experiment, we fixed the prior to the standard normal distribution $\mathcal{N}(0, I)$. We can see that the learned prior variant outperforms the fixed prior variant consistently across all datasets. The model with the fixed prior does not perform well because it learns to predict the majority action class and average inter-arrival time of the training set, ignoring the history of any input test sequence.

In addition to the above strategy of selecting the mode action at each step, we also report action category accuracy and MAE obtained by averaging over predictions of all 1500 samples. We summarize these results in Table 3.4.

We next explore the architecture of our model by varying the sizes of the latent variable. Table 3.5 shows the log-likelihood of our model for different sizes of the latent variable. We see that as we increase the size of the latent variable, we can model a more complex latent distribution which results in better performance.

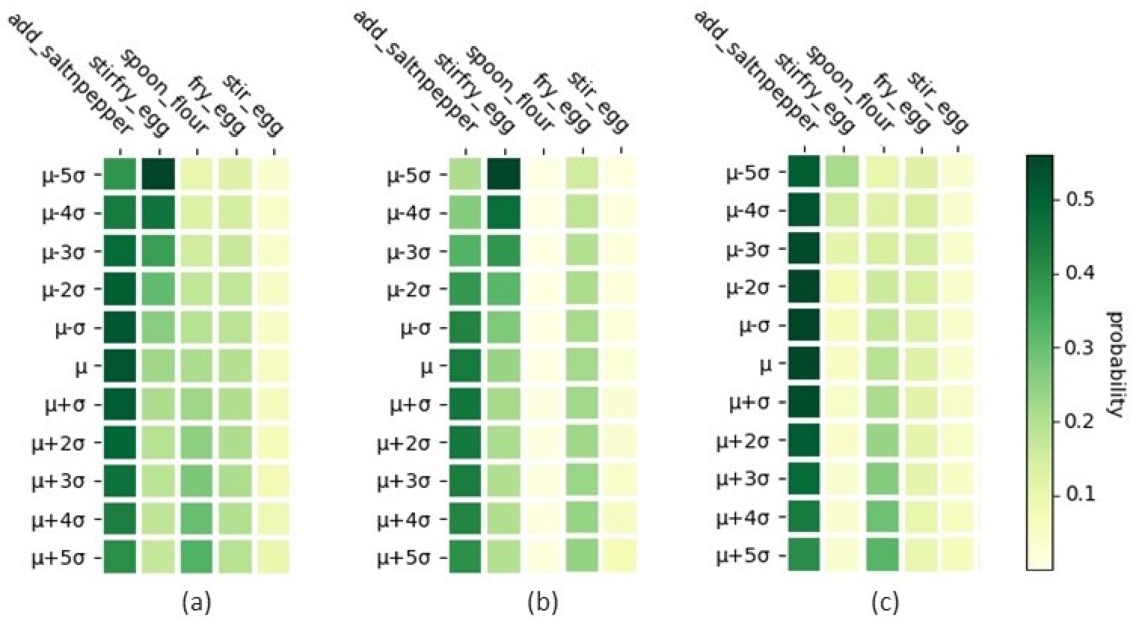


Figure 3.5: **Latent Code Manipulation.** The history + ground-truth label of future action for the sub-figures are: (a) “SIL, crack_egg"→“add_saltnpepper", (b) “SIL, take_plate, crack_egg"→ “add_saltnpepper" and (c) “SIL, pour_oil, crack_egg"→“add_saltnpepper".

Qualitative Results. Fig. 3.4 shows examples of diverse future action sequences that are generated by APP-VAE given the history. For different provided histories, sampled sequences of actions are shown. We note that the overall duration and sequence of actions on the Breakfast Dataset are reasonable. Variations, e.g. taking the juice squeezer before using it, adding salt and pepper before cooking eggs, are plausible alternatives generated by our model.

Fig. 3.5 visualizes a traversal on one of the latent codes for three different sequences by uniformly sampling one z dimension over $[\mu - 5\sigma, \mu + 5\sigma]$ while fixing others to their sampled values. As shown, this dimension correlates closely with the action *add_saltnpepper*, *strifry_egg* and *fry_egg*.

We further qualitatively examine the ability of the model to score the likelihood of individual test samples. We sort the test action sequences according to the average per time-step likelihood estimated by drawing 1500 samples from the approximate posterior distribution following the importance sampling approach. High scoring sequences should be those that our model deems as “normal" while low scoring sequences those that are unusual. Tab. 3.3 shows some example of sequences with low and high likelihood on the MultiTHUMOS dataset. We note that a regular, structured sequence of actions such as jump, body roll, cliff diving for a diving action or body contract, squat, clean and jerk for a weightlifting action receives high likelihood. However, repeated hammer throws or golf swings with no set up actions receives a low likelihood.

Finally we compare asynchronous APP-LSTM with a synchronous variant (with constant frame rate) on Breakfast dataset. The synchronous model predicts actions one step at a time and the se-

Latent size	32	64	128	256	512
LL ($\geq$)	-4.486	-3.947	-3.940	-3.838	-4.098

Table 3.5: Log-likelihood for APP-VAE with different latent variable dimensionality on MultiTHUMOS.

quence is post-processed to infer the duration of each action. The performance is significantly worse for both MAE time (152.17 vs 1459.99) and action prediction accuracy (61.39% vs 28.24%). A plausible explanation is that LSTMs cannot deal with very long-term dependencies.

3.5 Summary

We presented a novel probabilistic model for point process data – a variational auto-encoder that captures uncertainty in action times and category labels. As a generative model, it can produce action sequences by sampling from a prior distribution, the parameters of which are updated based on neural networks that control the distributions over the next action type and its temporal occurrence. The model can also be used to analyze given input sequences of actions to determine the likelihood of observing particular sequences. We demonstrate empirically that the model is effective for capturing the uncertainty inherent in tasks such as action prediction and anomaly detection.

Chapter 4

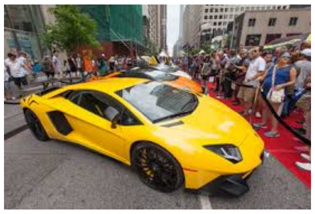
Learning a Deep ConvNet for Multi-label Classification with Partial Labels

Deep ConvNets have shown great performance for single-label image classification (*e.g*. ImageNet), but it is necessary to move beyond the single-label classification task because pictures of everyday life are inherently multi-label. Multi-label classification is a more difficult task than single-label classification because both the input images and output label spaces are more complex. Furthermore, collecting clean multi-label annotations is more difficult to scale-up than single-label annotations. To reduce the annotation cost, we propose to train a model with partial labels *i.e*. only some labels are known per image. We first empirically compare different labeling strategies to show the potential for using partial labels on multi-label datasets. Then to learn with partial labels, we introduce a new classification loss that exploits the proportion of known labels per example. Our approach allows the use of the same training settings as when learning with all the annotations. We further explore several curriculum learning based strategies to predict missing labels. Experiments are performed on three large-scale multi-label datasets: MS COCO, NUS-WIDE and Open Images.

This chapter is published as *Learning a Deep ConvNet for Multi-label Classification with Partial Labels* in proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019 [30].

4.1 Overview

Recently, Stock and Cisse [116] presented empirical evidence that the performance of state-of-the-art classifiers on ImageNet [109] is largely underestimated – much of the remaining error is due to the fact that ImageNet's single-label annotation ignores the intrinsic multi-label nature of the images. Unlike ImageNet, multi-label datasets (*e.g*. MS COCO [82], Open Images [75]) contain more complex images that represent scenes with several objects (Figure 4.1). However, collecting multi-label annotations is more difficult to scale-up than single-label annotations [25]. As an alternative strategy, one can make use of partial labels; collecting partial labels is easy and scalable with



	[a]	[b]	[c]
car	✓	✓	✓
person	✓		✗
boat	✗		✗
bear	✗	✗	✗
apple	✗		✗

Figure 4.1: Example of image with all annotations [a], partial labels [b] and noisy/webly labels [c]. In the partially labeled setting some annotations are missing (person, boat and apple) whereas in the webly labeled setting one annotation is wrong (person).

crowdsourcing platforms. In this work, we study the problem of learning a multi-label classifier with partial labels per image.

The two main (and complementary) strategies to improve image classification performance are: (i) designing / learning better model architectures [31, 32, 33, 49, 83, 99, 103, 117, 120, 135, 146, 147] and (ii) learning with more labeled data [88, 118]. However, collecting a multi-label dataset is more difficult and less scalable than collecting a single label dataset [25], because collecting a consistent and exhaustive list of labels for every image requires significant effort. To overcome this challenge, [80, 88, 118] automatically generated the labels using web supervision. But the drawback of these approaches is that the annotations are noisy and not exhaustive, and [143] showed that learning with corrupted labels can lead to very poor generalization performance. To be more robust to label noise, some methods have been proposed to learn with noisy labels [123].

An orthogonal strategy is to use partial annotations. This direction is actively being pursued by the research community: the largest publicly available multi-label dataset is annotated with partial clean labels [75]. For each image, the labels for some categories are known but the remaining labels are unknown (Figure 4.1). For instance, we know there is a *car* and there is not a *bear* in the image, but we do not know if there is a *person*, a *boat* or an *apple*. Relaxing the learning requirement for exhaustive labels opens better opportunities for creating large-scale datasets. Crowdsourcing platforms like Amazon Mechanical Turk[1] and Google Image Labeler[2] or web services like reCAPTCHA[3] can scalably collect partial labels for a large number of images.

To our knowledge, this is the first work to examine the challenging task of learning a multi-label image classifier with partial labels on large-scale datasets. Learning with partial labels on large-scale datasets presents novel challenges because existing methods [122, 131, 136, 137] are not scalable and cannot be used to fine-tune a ConvNet. We address these key technical challenges by introducing a new loss function and a method to fix missing labels.

[1]https://www.mturk.com/

[2]https://crowdsource.google.com/imagelabeler/category

[3]https://www.google.com/recaptcha/

Our first contribution is to empirically compare several labeling strategies for multi-label datasets to highlight the potential for learning with partial labels. Given a fixed label budget, our experiments show that partially annotating all images is better than fully annotating a small subset.

As a second contribution, we propose a scalable method to learn a ConvNet with partial labels. We introduce a loss function that generalizes the standard binary cross-entropy loss by exploiting label proportion information. This loss automatically adapts to the proportion of known labels per image and allows to use the same training settings as when learning with all the labels.

Our last contribution is a method to predict missing labels. We show that the learned model is accurate and can be used to predict missing labels. Because ConvNets are sensitive to noise [143], we propose a curriculum learning based model [4] that progressively predicts some missing labels and adds them to the training set. To improve label predictions, we develop an approach based on Graph Neural Networks (GNNs) to explicitly model the correlation between categories. In multi-label settings, not all labels are independent, hence reasoning about label correlation between observed and unobserved partial labels is important.

4.2 Related Work

Learning with partial / missing labels. Multi-label tasks often involve incomplete training data, hence several methods have been proposed to solve the problem of multi-label learning with missing labels (MLML). The first and simple approach is to treat the missing labels as negative labels [5, 88, 94, 118, 119, 128]. The MLML problem then becomes a fully labeled learning problem. This solution is used in most webly supervised approaches [88, 118]. The standard assumption is that only the category of the query is present (*e.g. car* in Figure 4.1) and all the other categories are absent. However, performance drops because a lot of ground-truth positive labels are initialized as negative labels [61]. A second solution is Binary Relevance (BR) [122], which treats each label as an independent binary classification. But this approach is not scalable when the number of categories grows and it ignores correlations between labels and between instances, which can be helpful for recognition. Unlike BR, our proposed approach allows to learn a single model using partial labels.

To overcome the second problem, several works proposed to exploit label correlations from the training data to propagate label information from the provided labels to missing labels. [8, 136] used a matrix completion algorithm to fill in missing labels. These methods exploit label-label correlations and instance-instance correlations with low-rank regularization on the label matrix to complete the instance-label matrix. Similarly, [141] introduced a low rank empirical risk minimization, [131] used a mixed graph to encode a network of label dependencies and [25, 94] learned correlation between the categories to predict some missing labels. Unlike most of the existing models that assume that the correlations are linear and unstructured, [137] proposed to learn structured semantic correlations. Another strategy is to treat missing labels as latent variables in probabilistic models. Missing labels are predicted by posterior inference. [62, 124] used models based on Bayesian networks [56] whereas [18] proposed a deep sequential generative model based on a Variational Auto-Encoder framework [66] that also allows to deal with unlabeled data.

However, most of these works cannot be used to learn a deep ConvNet. They require solving an optimization problem with the training set in memory, so it is not possible to use a mini-batch strategy to fine-tune the model. This is limiting because it is well-known that fine-tuning is important to transfer a pre-trained architecture [72]. Some methods are also not scalable because they require to solve convex quadratic optimization problems [131, 137] that are intractable for large-scale datasets. Unlike these methods, we propose a model that is scalable and end-to-end learnable. To train our model, we introduce a new loss function that adapts itself to the proportion of known labels per example. Similar to some MLML methods, we also explore several strategies to fill-in missing labels by using the learned classifier.

Learning with partial labels is different from semi-supervised learning [12] because in the semi-supervised learning setting, only a subset of the examples is labeled with all the labels and the other examples are unlabeled whereas in the partial labels setting, all the images are labeled but only with a subset of labels. Note that [21] also introduced a partially labeled learning problem (also called ambiguously labeled learning) but this problem is different: in [21], each example is annotated with multiple labels but only one is correct.

Curriculum Learning / Never-Ending Learning. To predict missing labels, we propose an iterative strategy based on Curriculum Learning [4]. The idea of Curriculum Learning is inspired by the way humans learn: start to learn with easy samples/subtasks, and then gradually increase the difficulty level of the samples/subtasks. But, the main problem in using curriculum learning is to measure the difficulty of an example. To solve this problem, [74] used the definition that easy samples are ones whose correct output can be predicted easily. They introduced an iterative self-paced learning (SPL) algorithm where each iteration simultaneously selects easy samples and updates the model parameters. [58] generalizes the SPL to different learning schemes by introducing different self-paced functions. Instead of using human-designed heuristics, [59] proposed MentorNet, a method to learn the curriculum from noisy data. Similar to our work, [44] recently introduced the CurriculumNet that is a model to learn from large-scale noisy web images with a curriculum learning approach. However this strategy is designed for multi-class image classification and cannot be used for multi-label image classification because it uses a clustering-based model to measure the difficulty of the examples.

Our approach is also related to the Never-Ending Learning (NEL) paradigm [95]. The key idea of NEL is to use previously learned knowledge to improve the learning of the model. [78] proposed a framework that alternatively learns object class models and collects object class datasets. [9, 95] introduced the Never-Ending Language Learning to extract knowledge from hundreds of millions of web pages. Similarly, [14, 15] proposed the Never-Ending Image Learner to discover structured visual knowledge. Unlike these approaches that use a previously learned model to extract knowledge from web data, we use the learned model to predict missing labels.

4.3 Learning with Partial Labels

Our goal in this work is to train ConvNets given partial labels. We first introduce a loss function to learn with partial labels that generalizes the binary cross-entropy. We then extend the model with a Graph Neural Network to reason about label correlations between observed and unobserved partial labels. Finally, we use these contributions to learn an accurate model that it is used to predict missing labels with a curriculum-based approach.

Notation. We denote by C the number of categories and N the number of training examples. We denote the training data by $\mathcal{D} = \{(\mathcal{I}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathcal{I}^{(N)}, \mathbf{y}^{(N)})\}$, where $\mathcal{I}^{(i)}$ is the i^{th} image and $\mathbf{y}^{(i)} = [y_1^{(i)}, \dots, y_C^{(i)}] \in \mathcal{Y} \subseteq \{-1, 0, 1\}^C$ the label vector. For a given example i and category c, $y_c^{(i)} = 1$ (resp. -1 and 0) means the category is present (resp. absent and unknown). $\mathbf{y} = [\mathbf{y}^{(1)}; \dots; \mathbf{y}^{(N)}] \in \{-1, 0, 1\}^{N \times C}$ is the matrix of training set labels. $f_{\mathbf{w}}$ denotes a deep ConvNet with parameters $\mathbf{w}$. $\mathbf{x}^{(i)} = [x_1^{(i)}, \dots, x_C^{(i)}] = f_{\mathbf{w}}(\mathcal{I}^{(i)}) \in \mathbb{R}^C$ is the output (before sigmoid) of the deep ConvNet $f_{\mathbf{w}}$ on image $\mathcal{I}^{(i)}$.

4.3.1 Binary cross-entropy for partial labels

The most popular loss function to train a model for multi-label classification is binary cross-entropy (BCE). To be independent of the number of categories, the BCE loss is normalized by the number of classes. This becomes a drawback for partially labeled data because the back-propagated gradient becomes small. To overcome this problem, we propose the partial-BCE loss that normalizes the loss by the proportion of known labels:

$$\ell(\mathbf{x}, \mathbf{y}) = \frac{g(p_{\mathbf{y}})}{C} \sum_{c=1}^{C} \Bigg[\mathbb{1}_{[y_c = 1]} \log\left(\frac{1}{1 + \exp(-x_c)}\right) + \mathbb{1}_{[y_c = -1]} \log\left(\frac{\exp(-x_c)}{1 + \exp(-x_c)}\right) \Bigg] \tag{4.1}$$

where $p_{\mathbf{y}} \in [0, 1]$ is the proportion of known labels in $\mathbf{y}$ and g is a normalization function with respect to the label proportion. Note that the partial-BCE loss ignores the categories for unknown labels ($y_c = 0$). In the standard BCE loss, the normalization function is $g(p_{\mathbf{y}}) = 1$. Unlike the standard BCE, the partial-BCE gives the same importance to each example independent of the number of known labels, which is useful when the proportion of labels per image is not fixed. This loss adapts itself to the proportion of known labels. We now explain how we design the normalization function g.

Normalization function g . The function g normalizes the loss function with respect to the label proportion. We want the partial-BCE loss to have the same behavior as the BCE loss when all the

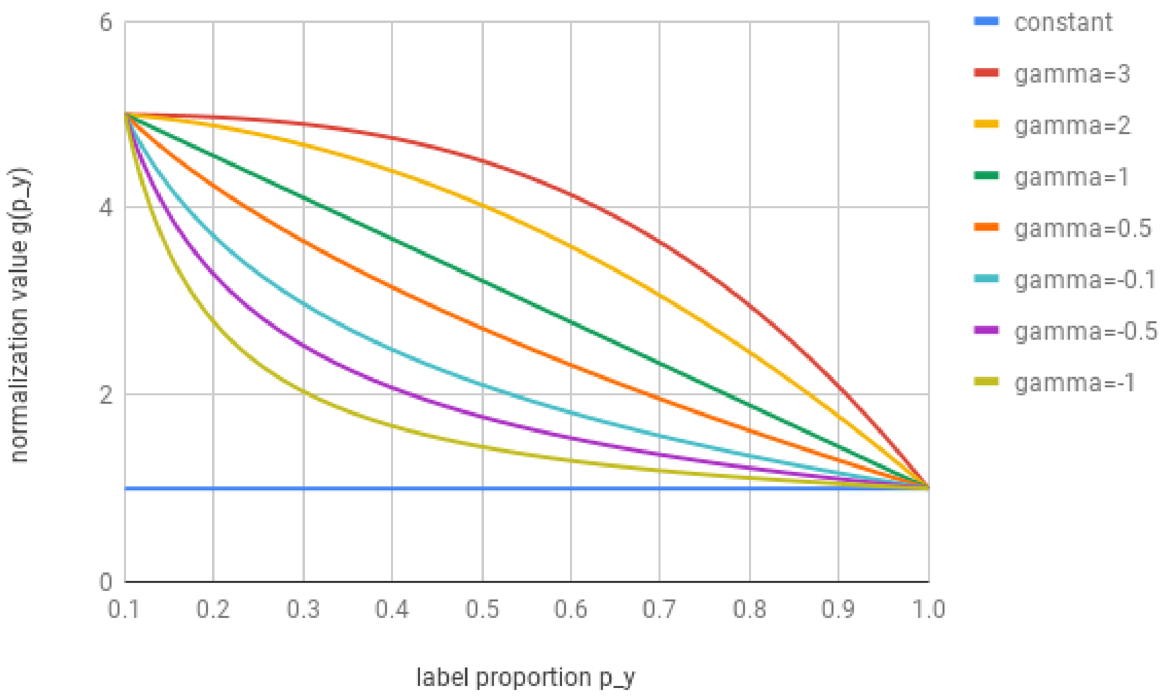


Figure 4.2: Examples of the weight function g (Equation 4.2) for different values of hyperparameter γ with the constraint $g(0.1) = 5$. γ controls the behavior of the normalization with respect to the label proportion $p_{\mathbf{y}}$.

labels are present *i.e.* $g(1) = 1$. We propose to use the following normalization function:

$$g(p_{\mathbf{y}}) = \alpha p_{\mathbf{y}}^{\gamma} + \beta \tag{4.2}$$

where α, β and γ are the hyperparameters that allow to generalize several standard functions. For instance with $\alpha = 1$, $\beta = 0$ and $\gamma = -1$, this function weights each example inversely proportional to the proportion of labels. This is equivalent to normalizing by the number of known classes instead of the number of classes. Given a γ value and the weight for a given proportion (*e.g.* $g(0.1) = 5$), we can find the hyperparameters α and β that satisfy these constraints. The hyperparameter γ controls the behavior of the normalization with respect to the label proportion. In Figure 4.2 we show this function for different values of γ given the constraint $g(0.1) = 5$. For $\gamma = 1$ the normalization is linearly proportional to the label proportion, whereas for $\gamma = -1$ the normalization value is inversely proportional to the label proportion. We analyse the importance of each hyperparameter in Sec.4.4. This normalization has a similar goal to batch normalization [53] which normalizes distributions of layer inputs for each mini-batch.

4.3.2 Multi-label classification with GNN

To model the interactions between the categories, we use a Graph Neural Network (GNN) [41, 112] on top of a ConvNet. We first introduce the GNN and then detail how we use GNN for multi-label classification.

GNN. For GNNs, the input data is a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ (resp. $\mathcal{E}$) is the set of nodes (resp. edges) of the graph. For each node $v \in \mathcal{V}$, we denote the input feature vector $\mathbf{x}_v$ and its hidden representation describing the node's state at time step t by $\mathbf{h}_v^t$. We use Ω_v to denote the set of neighboring nodes of v. A node uses information from its neighbors to update its hidden state. The update is decomposed into two steps: message update and hidden state update. The message update step combines messages sent to node v into a single message vector $\mathbf{m}_v^t$ according to:

$$\mathbf{m}_v^t = \mathcal{M}(\{\mathbf{h}_u^t | u \in \Omega_v\}) \tag{4.3}$$

where $\mathcal{M}$ is the function to update the message. In the hidden state update step, the hidden states $\mathbf{h}_v^t$ at each node in the graph are updated based on messages $\mathbf{m}_v^t$ according to:

$$\mathbf{h}_v^{t+1} = \mathcal{F}(\mathbf{h}_v^t, \mathbf{m}_v^t) \tag{4.4}$$

where $\mathcal{F}$ is the function to update the hidden state. $\mathcal{M}$ and $\mathcal{F}$ are feedforward neural networks that are shared among different time steps. Note that these update functions specify a propagation model of information inside the graph.

GNN for multi-label classification. For multi-label classification, each node represents one category ($\mathcal{V} = \{1, \dots, C\}$) and the edges represent the connections between the categories. We use a fully-connected graph to model correlation between all categories. The node hidden states are initialized with the ConvNet output. We now detail the GNN functions used in our model. The algorithm and additional information are given in the supplementary material.

Message update function $\mathcal{M}$. We use the following message update function:

$$\mathbf{m}_v^t = \frac{1}{|\Omega_v|} \sum_{u \in \Omega_v} f_{\mathcal{M}}(\mathbf{h}_u^t) \tag{4.5}$$

where $f_{\mathcal{M}}$ is a multi-layer perceptron (MLP). The message is computed by first feeding hidden states to the MLP $f_{\mathcal{M}}$ and then taking the average over the neighborhood.

Hidden state update function $\mathcal{F}$. We use the following hidden state update function:

$$\mathbf{h}_v^{t+1} = GRU(\mathbf{h}_v^t, \mathbf{m}_v^t) \tag{4.6}$$

which uses a Gated Recurrent Unit (GRU) [16]. The hidden state is updated based on the incoming messages and the previous hidden state.

4.3.3 Prediction of unknown labels

In this section, we propose a method to predict some missing labels with a curriculum learning strategy [4]. We formulate our problem based on the self-paced model [58, 74] and the goal is to optimize the following objective function:

$$\begin{aligned}\min_{\mathbf{w}\in\mathbb{R}^d, \mathbf{v}\in\{0,1\}^{N\times C}} J(\mathbf{w},\mathbf{v}) &= \beta\|\mathbf{w}\|^2 + G(\mathbf{v};\theta) \\ &+ \frac{1}{N}\sum_{i=1}^{N}\frac{1}{C}\sum_{c=1}^{C} v_{ic}\ell_c(f_{\mathbf{w}}(\mathcal{I}^{(i)}), y_c^{(i)})\end{aligned} \tag{4.7}$$

where ℓ_c is the loss for category c and $v_i \in \{0,1\}^C$ is a vector to represent the selected labels for the i-th sample. $v_{ic} = 1$ (resp. $v_{ic} = 0$) means that the c-th label of the i-th example is selected (resp. unselected). The function G defines a curriculum, parameterized by θ, which defines the learning scheme. Following [74], we use an alternating algorithm where $\mathbf{w}$ and $\mathbf{v}$ are alternatively minimized, one at a time while the other is held fixed. The algorithm is shown in Algorithm 1. Initially, the model is learned with only clean partial labels. Then, the algorithm uses the learned model to add progressively new "easy" weak (*i.e.* noisy) labels in the training set, and then uses the clean and weak labels to continue the training of the model. We analyze different strategies to add new labels:

[a] Score threshold strategy. This strategy uses the classification score (*i.e.* ConvNet) to estimate the difficulty of a pair category-example. An easy example has a high absolute score whereas a hard example has a score close to 0. We use the learned model on partial labels to predict the missing labels only if the classification score is larger than a threshold $\theta > 0$. When $\mathbf{w}$ is fixed, the optimal $\mathbf{v}$ can be derived by:

$$v_{ic} = \mathbb{1}[x_c^{(i)} \geq \theta] + \mathbb{1}[x_c^{(i)} < -\theta] \tag{4.8}$$

The predicted label is $y_c^{(i)} = \text{sign}(x_c^{(i)})$.

[b] Score proportion strategy. This strategy is similar to the strategy [a] but instead of labeling the pair category-example higher than a threshold, we label a fixed proportion θ of pairs per mini-batch. To find the optimal $\mathbf{v}$, we sort the examples by decreasing order of absolute score and label only the top-$\theta\%$ of the missing labels.

[c] Predict only positive labels. Because of the imbalanced annotations, we only predict positive labels with strategy [a]. When $\mathbf{w}$ is fixed, the optimal $\mathbf{v}$ can be derived by:

$$v_{ic} = \mathbb{1}[x_c^{(i)} \geq \theta] \tag{4.9}$$

[d] Ensemble score threshold strategy. This strategy is similar to the strategy [a] but it uses an ensemble of models to estimate the confidence score. We average the classification score of each model to estimate the final confidence score. This strategy allows to be more robust than the strategy

Algorithm 1 Curriculum labeling

```
Input: Training data $\mathcal{D}$
1: Initialize $\mathbf{v}$ with known labels
2: Initialize $\mathbf{w}$: learn the ConvNet with the partial labels
3: repeat
4:     Update $\mathbf{v}$ (fixed $\mathbf{w}$): find easy missing labels
5:     Update $\mathbf{y}$: predict the label of easy missing labels
6:     Update $\mathbf{w}$ (fixed $\mathbf{v}$): improve classification model with the clean and easy weak annotations
7: until stopping criteria
```

[a]. When $\mathbf{w}$ is fixed, the optimal $\mathbf{v}$ can be derived by:

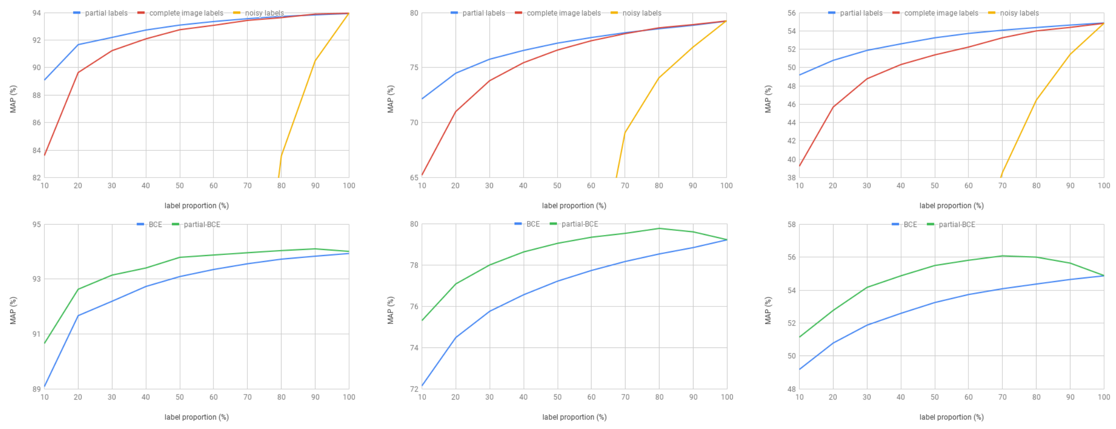
$$v_{ic} = \mathbb{1}[E(\mathcal{I}^{(i)})_c \geq \theta] + \mathbb{1}[E(\mathcal{I}^{(i)})_c < -\theta] \tag{4.10}$$

where $E(\mathcal{I}^{(i)}) \in \mathbb{R}^C$ is the vector score of an ensemble of models. The predicted label is $y_c^{(i)} = \text{sign}(E(\mathcal{I}^{(i)})_c)$.

[e] Bayesian uncertainty strategy. Instead of using the classification score as in [a] or [d], we estimate the bayesian uncertainty [64] of each pair category-example. An easy pair category-example has a small uncertainty. When $\mathbf{w}$ is fixed, the optimal $\mathbf{v}$ can be derived by:

$$v_{ic} = \mathbb{1}[U(\mathcal{I}^{(i)})_c \leq \theta] \tag{4.11}$$

where $U(\mathcal{I}^{(i)})$ is the bayesian uncertainty of category c of the i-th example. This strategy is similar to strategy [d] except that it uses the variance of the classification scores instead of the average to estimate the difficulty.



Pascal VOC 2007 MS COCO NUS-WIDE

Figure 4.3: The first row shows MAP results for the different labeling strategies. On the second row, we shows the comparison of the BCE and the partial-BCE. The x-axis shows the proportion of clean labels.

4.4 Experiments

Static Image Datasets. We perform experiments on several standard multi-label datasets: Pascal VOC 2007 [36], MS COCO [82] and NUS-WIDE [19]. For each dataset, we use the standard train/test sets introduced respectively in [36], [98], and [39] (see Section 4.5.2 for more details). From these datasets that are fully labeled, we create partially labeled datasets by randomly dropping some labels per image. The proportion of known labels is between 10% (90% of labels missing) and 100% (all labels present). We also perform experiments on the large-scale Open Images dataset [75] that is partially annotated: 0.9% of the labels are available during training.

Metrics. To evaluate the performances, we use several metrics: mean Average Precision (MAP) [3], 0-1 exact match, Macro-F1 [138], Micro-F1 [121], per-class precision, per-class recall, overall precision, overall recall. These metrics are standard multi-label classification metrics and are presented in Section 4.5.3. We mainly show the results for the MAP metric but results for other metrics are shown in Section 4.5.

Implementation details. We employ ResNet-WELDON [33] as our classification network. We use a ResNet-101 [49] pretrained on ImageNet as the backbone architecture, but we show results for other architectures in Section 4.5. The models are implemented with PyTorch [101]. The hyperparameters of the partial-BCE loss function are $\alpha = -4.45$, $\beta = 5.45$ (*i.e.* $g(0.1) = 5$) and $\gamma = 1$. To predict missing labels, we use the bayesian uncertainty strategy with $\theta = 0.3$.

4.4.1 What is the best strategy to annotate a dataset?

In the first set of experiments, we study three strategies to annotate a multi-label dataset. The goal is to answer the question: what is the best strategy to annotate a dataset with a fixed budget of clean labels? We explore the three following scenarios:

- **Partial labels**. This is the strategy used in this paper. In this setting, all the images are used but only a subset of the labels per image are known. The known categories are different for each image.
- **Complete image labels or dense labels**. In this scenario, only a subset of the images are labeled, but the labeled images have the annotations for all the categories. This is the standard setting for semi-supervised learning [12] except that we do not use a semi-supervised model.
- **Noisy labels**. All the categories of all images are labeled but some labels are wrong. This scenario is similar to the webly-supervised learning scenario [88] where some labels are wrong.

To have fair comparison between the approaches, we use a BCE loss function for these experiments. The results are shown in Figure 4.3 for different proportion of clean labels. For each experiment, we use the same number of clean labels. 100% means that all the labels are known during training (standard classification setting) and 10% means that only 10% of the labels are known during training. The 90% of other labels are unknown labels for the partial labels and the complete image labels scenarios and are wrong labels for the noisy labels scenario. Similar to [118], we observe that the performance increases logarithmically based on proportion of labels. From this first experiment, we can draw the following conclusions: (1) Given a fixed number of clean labels, we observe that learning with partial labels is better than learning with a subset of dense annotations. The improvement increases when the label proportion decreases. A reason is that the model trained in the partial labels strategy "sees" more images during training and therefore has a better generalization performance. (2) It is better to learn with a small subset of clean labels than a lot of labels with some incorrect labels. Both partial labels and complete image labels scenarios are better than the noisy label scenario. For instance on MS COCO, we observe that learning with only 20% of clean partial labels is better than learning with 80% of clean labels and 20% of wrong labels.

Noisy web labels. Another strategy to generate a noisy dataset from a multi-label dataset is to use only one positive label for each image. This is a standard assumption made when collecting data from the web [80] *i.e.* the only category present in the image is the category of the query. From the clean MS COCO dataset, we generate a noisy dataset (named noisy+) by keeping only one positive label per image. If the image has more than one positive label, we randomly select one positive label among the positive labels and switch the other positive labels to negative labels. The results are reported in Table 4.1 for three scenarios: clean (all the training labels are known and clean), 10% of partial labels and noisy+ scenario. We also show the percentage of clean and noisy labels for each experiment. The noisy+ approach generates a small proportion of noisy labels

model	clean	partial 10%	noisy+
clean / noisy labels	100 / 0	10 / 0	97.6 / 2.4
MAP (%)	79.22	72.15	71.60

Table 4.1: Comparison with a webly-supervised strategy (noisy+) on MS COCO. Clean (resp. noisy) means the percentage of clean (resp. noisy) labels in the training set.

(2.4%) that drops the performance by about 7pt with respect to the clean baseline. We observe that a model trained with only 10% of clean labels is slightly better than the model trained with the noisy labels. This experiment shows that the standard assumption made in most of the webly-supervised datasets is not good for complex scenes / multi-label images because it generates noisy labels that significantly decrease generalization.

Relabeling	MAP	0-1	Macro-F1	Micro-F1	label prop.	TP	TN	GNN
2 steps (no curriculum)	-1.49	6.42	2.32	1.99	100	82.78	96.40	✓
[a] Score threshold $\theta = 2$	0.34	11.15	4.33	4.26	95.29	85.00	98.50	✓
[b] Score proportion $\theta = 80\%$	0.17	8.40	3.70	3.25	96.24	84.40	98.10	✓
[c] Postitive only - score $\theta = 5$	0.31	-4.58	-1.92	-2.23	12.01	79.07	-	✓
[d] Ensemble score $\theta = 2$	0.23	11.31	4.16	4.33	95.33	84.80	98.53	✓
[e] Bayesian uncertainty $\theta = 0.3$	0.34	10.15	4.37	3.72	77.91	61.15	99.24	
[e] Bayesian uncertainty $\theta = 0.1$	0.36	2.71	1.91	1.22	19.45	38.15	99.97	✓
[e] Bayesian uncertainty $\theta = 0.2$	0.30	10.76	4.87	4.66	57.03	62.03	99.65	✓
[e] Bayesian uncertainty $\theta = 0.3$	0.59	12.07	5.11	4.95	79.74	68.96	99.23	✓
[e] Bayesian uncertainty $\theta = 0.4$	0.43	10.99	4.88	4.46	90.51	70.77	98.57	✓
[e] Bayesian uncertainty $\theta = 0.5$	0.45	10.08	3.93	3.78	94.79	74.73	98.00	✓

Table 4.2: Analysis of the labeling strategy of missing labels on Pascal VOC 2007 val set. For each metric, we report the relative scores with respect to a model that does not label missing labels. TP (resp. TN) means true positive (resp. true negative) rate. For the strategy [c], we report the label accuracy instead of the TP rate.

4.4.2 Learning with partial labels

In this section, we compare the standard BCE and the partial-BCE and analyze the importance of the GNN.

BCE vs partial-BCE. The Figure 4.3 shows the MAP results for different proportion of known labels on three datasets. For all the datasets, we observe that using the partial-BCE significantly improves the performance: the lower the label proportion, the better the improvement. We observe the same behavior for the other metrics (Section 4.5.6). In Table 4.3, we show results on the Open Images dataset and we observe that the partial-BCE is 4 pt better than the standard BCE. These experiments show that our loss learns better than the BCE because it exploits the label proportion

	BCE	partial-BCE	GNN + partial-BCE
MAP (%)	79.01	83.05	83.36

Table 4.3: MAP results on Open Images.

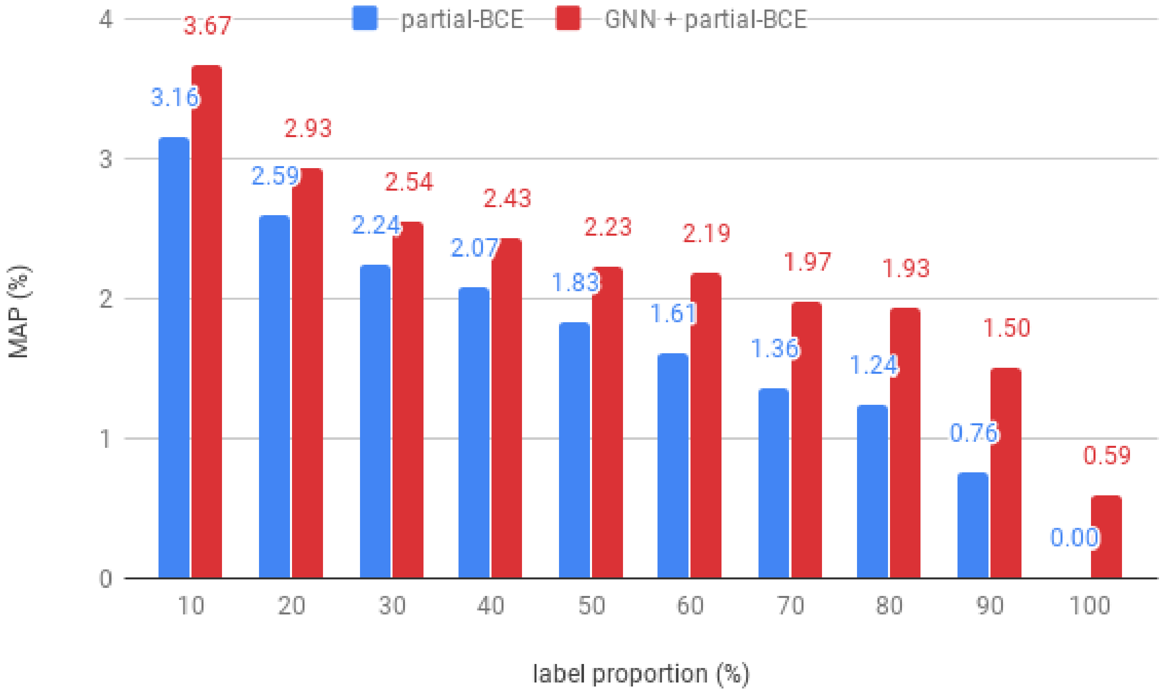


Figure 4.4: MAP (%) improvement with respect to the proportion of known labels on MS COCO for the partial-BCE and the GNN + partial-BCE. 0 means the result for a model trained with the standard BCE.

information during training. It allows to learn efficiently while keeping the same training setting as with all annotations.

GNN. We now analyze the improvements of the GNN to learn relationships between the categories. We show the results on MS COCO in Figure 4.4. We observe that for each label proportion, using the GNN improves the performance. Open Images experiments (Table 4.3) show that GNN improves the performance even when the label proportion is small. This experiment shows that modeling the correlation between categories is important even in case of partial labels. However, we also note that a ConvNet implicitly learns some correlation between the categories because some learned representations are shared by all categories.

4.4.3 What is the best strategy to predict missing labels?

In this section, we analyze the labeling strategies introduced in Section 4.3.3 to predict missing labels. Before training epochs 10 and 15, we use the learned classifier to predict some missing

BCE	fine-tuning	partial-BCE	GNN	relabeling	MAP	0-1 exact match	Macro-F1	Micro-F1
✓					66.21	17.53	62.74	67.33
✓	✓				72.15	22.04	65.82	70.09
	✓	✓			75.31	24.51	67.94	71.18
	✓	✓	✓		75.82	25.14	68.40	71.37
	✓	✓		✓	75.71	30.52	70.13	73.87
	✓	✓	✓	✓	76.40	32.12	70.73	74.37

Table 4.4: Ablation study on MS COCO with 10% of known labels.

labels. We report the results for different metrics on Pascal VOC 2007 validation set with 10% of labels in Table 4.2. We also report the final proportion of labels, the true postive (TP) and true negative (TN) rates for predicted labels. Additional results are shown in Section 4.5.9.

First, we show the results of a 2 steps strategy that predicts all missing labels in one time. Overall, we observe that this strategy is worse than curriculum-based strategies ([a-e]). In particular, the 2 steps strategy decreases the MAP score. These results show that predicting all missing labels at once introduced too much label noise, decreasing generalization performance. Among the curriculum-based strategies, we observe that the threshold strategy [a] is better than the proportion strategy [b]. We also note that using a model ensemble [d] does not significantly improve the performance with respect to a single model [a]. Predicting only positive labels [c] is a poor strategy. The bayesian uncertainty strategy [e] is the best strategy. In particular, we observe that the GNN is important for this strategy because it decreases the label uncertainty and allows the model to be robust to the hyperparameter θ.

4.4.4 Method analysis

In this section, we analyze the hyperparameters of the partial-BCE and perform an ablation study on MS COCO.

Partial-BCE analysis. To analyze the partial-BCE, we use only the training set. The model is trained on about 78k images and evaluated on the remaining 5k images. We first analyse how to choose the value of the normalization function given a label proportion of 10% *i.e.* $g(0.1)$ (it is possible to choose another label proportion). The results are shown in Figure 4.5. Note that for $g(0.1) = 1$, the partial-BCE is equivalent to the BCE and the loss is normalized by the number of categories. We observe that the normalization value $g(0.1) = 1$ gives the worst results. The best score is obtained for a normalization value around 20 but the performance is similar for $g(0.1) \in [3, 50]$. Using a large value drops the performance. This experiment shows that the proposed normalization function is important and robust. These results are independent of the network architectures (Section 4.5.7).

Given the constraints $g(0.1) = 5$ and $g(1) = 1$, we analyze the impact of the hyperparameter γ. This hyperparameter controls the behavior of the normalization with respect to the label proportion.

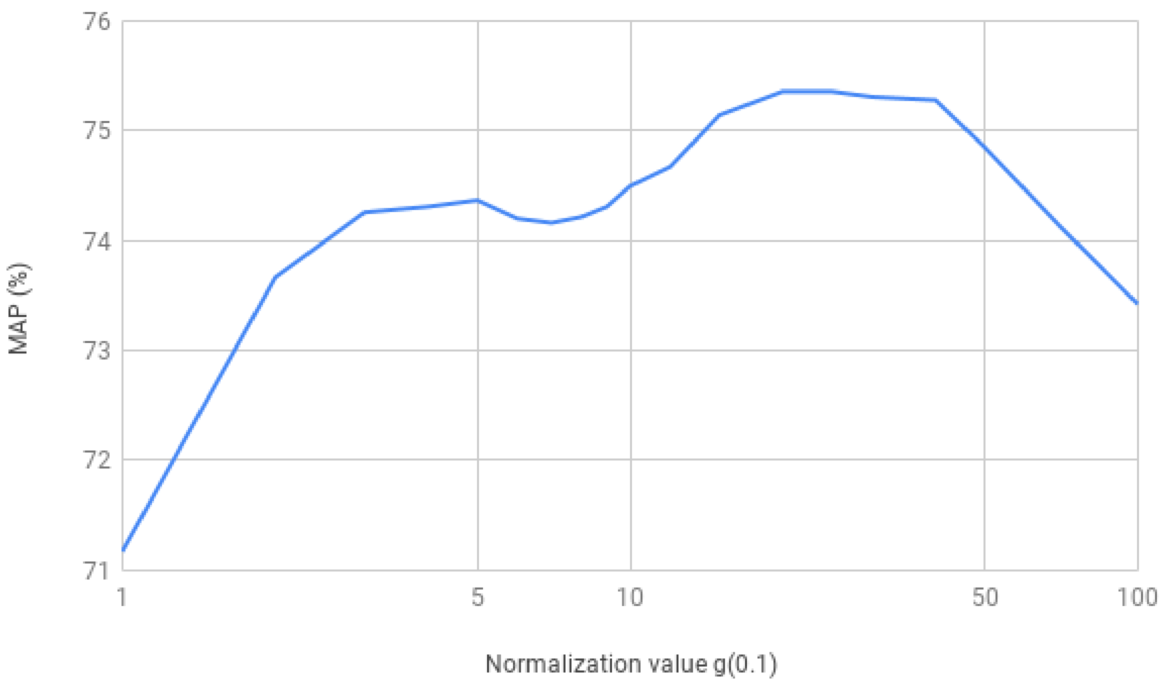


Figure 4.5: Analysis of the normalization value for a label proportion of 10% (*i.e.* $g(0.1)$). (x-axis log-scale)

Using a high value ($\gamma = 3$) is better than a low value ($\gamma = -1$) for large label proportions but is slighty worse for small label proportions. We observe that using a normalization that is proportional to the number of known labels ($\gamma = 1$) works better than using a normalization that is inversely proportional to the number of known labels ($\gamma = -1$).

Ablation study. Finally to analyze the importance of each contribution, we perform an ablation study on MS COCO for a label proportion of 10% in Table 4.4. We first observe that fine-tuning is important. It validates the importance of building end-to-end trainable models to learn with missing labels. The partial-BCE loss function increases the performance against each metric because it exploits the label proportion information during training. We show that using GNN or relabeling improves performance. In particular, the relabeling stage significantly increases the 0-1 exact match score (+5pt) and the Micro-F1 score (+2.5pt). Finally, we observe that our contributions are complementary.

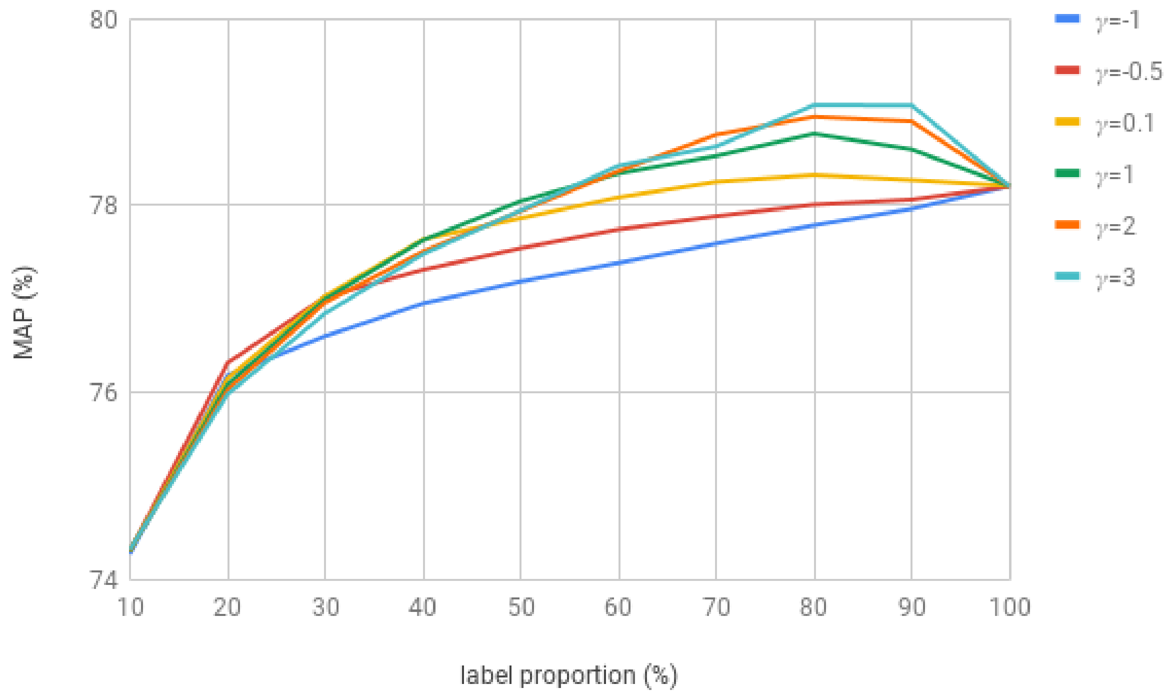


Figure 4.6: Analysis of hyperparameter γ on MS COCO.

4.5 Implementation Details and Analysis

4.5.1 Multi-label classification with GNN

In this subsection, we give additional information about the Graph Neural Networks (GNN) used in our work. We first show the algorithm used to predict the classification scores with a GNN in Algorithm 2. The input $\mathbf{x} \in \mathbb{R}^C$ of the GNN is the ConvNet output, where C is the number of categories.

The $f_{\mathcal{M}}$ function in the message update function $\mathcal{M}$ is a fully connected layer followed by a ReLU. Because the graph is fully-connected, the message update function $\mathcal{M}$ averages on all the nodes of the graph excepts the current node v *i.e.* $\Omega_v = \mathcal{V} \setminus \{v\}$. Similarly to [104], the final prediction uses both first and last hidden states. We observe that using both first and last hidden states is better than using only the last hidden state. According to [104], we use $T = 3$ iterations in our experiments.

Algorithm 2 Graph Neural Network (GNN)

```
Input: ConvNet output $\mathbf{x}$
1: Initialize the hidden state of each node $v \in \mathcal{V}$ with the output of the ConvNet.
       $$\mathbf{h}_v^0 = [0, \dots, 0, x_v, 0, \dots, 0] \qquad \forall v \in \mathcal{V} \qquad (4.12)$$
2: for t = 0 to T-1 do
3:     Update message of each node $v \in \mathcal{V}$ based on the hidden states
       $$\mathbf{m}_v^t = \mathcal{M}(\{\mathbf{h}_u^t | u \in \Omega_v\}) = \frac{1}{|\Omega_v|} \sum_{u \in \Omega_v} f_{\mathcal{M}}(\mathbf{h}_u^t) \qquad (4.13)$$
4:     Update hidden state of each node $v \in \mathcal{V}$ based on the messages
       $$\mathbf{h}_v^{t+1} = \mathcal{F}(\mathbf{h}_v^t, \mathbf{m}_v^t) = GRU(\mathbf{h}_v^t, \mathbf{m}_v^t) \qquad (4.14)$$
5: end for
6: Compute the output based on the first and last hidden states
       $$\bar{\mathbf{y}} = s(\mathbf{h}_v^0, \mathbf{h}_v^T) = \mathbf{h}_v^0 + \mathbf{h}_v^T \qquad (4.15)$$
Output: $\bar{\mathbf{y}}$
```

4.5.2 Experimental details

Datasets. We perform experiments on large publicly available multi-label datasets: Pascal VOC 2007 [36], MS COCO [82] and NUS-WIDE [19]. Pascal VOC 2007 dataset contains 5k/5k trainval/test images of 20 objects categories. MS COCO dataset contains 123k images of 80 objects categories. We use the 2014 data split with 83k train images and 41k val images. NUS-WIDE dataset contains 269,648 images downloaded from Flickr that have been manually annotated with 81 visual concepts. We follow the experimental protocol in [39] and use 150k randomly sampled images for training and the rest for testing. The results on NUS-WIDE cannot be directly comparable with the other works because the number of total images is different (209,347 in [39], 200,261 in [81]). The main reason is that some provided URLs are invalid or some images have been deleted from Flickr. For our experiments, we collected 216,450 images.

We also performs experiments on the largest publicly available multi-label dataset: Open Images [75]. This dataset is partially annotated with human labels and machine generated labels. For our experiments, we use only human labels on the 600 boxable classes. On the training set, only 0.9% of the labels are available.

Implementation details. The hyperparameters of the WELDON pooling function are $k^+ = k^- = 0.1$. The models are implemented with PyTorch [101] and are trained with SGD during 20 epochs with a batch size of 16. The initial learning rate is 0.01 and it is divide by 10 after 10 epochs. During training, we only use random horizontal flip as data augmentation. Each image is resized to 448×448 with 3 color channels. On Open Images dataset, unlike [75] we do not train from scratch the network. We use a similar protocol that on the others datasets: we fine-tune a model pre-train on ImageNet but stop the training when the validation performance does not increase. Because the training set has 1.7M images, the model converge in less than 5 epochs.

4.5.3 Multi-label metrics

In this subsection, we introduce the metrics used to evaluate the performances on multi-label datasets. We note $\mathbf{y}^{(i)} = [y_1^{(i)}, \dots, y_C^{(i)}] \in \mathcal{Y} \subseteq \{-1, 0, 1\}^C$ the ground truth label vector and $\hat{\mathbf{y}}^{(i)} = [\hat{y}_1^{(i)}, \dots, \hat{y}_C^{(i)}] \in \{-1, 1\}^C$ the predicted label vector of the i-th example.

Zero-one exact match accuracy (0-1). This metric considers a prediction correct only if all the labels are correctly predicted:

$$m_{0/1}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[\mathbf{y}^{(i)} = \hat{\mathbf{y}}^{(i)}] \tag{4.16}$$

where $\mathbb{1}[.]$ is an indicator function.

Per-class precision/recall (PC-P/R).

$$m_{PC-P}(\mathcal{D}) = \frac{1}{C} \sum_{c=1}^{C} \frac{N_c^{correct}}{N_c^{predict}} \tag{4.17}$$

$$m_{PC-R}(\mathcal{D}) = \frac{1}{C} \sum_{c=1}^{C} \frac{N_c^{correct}}{N_c^{gt}} \tag{4.18}$$

where $N_c^{correct}$ is the number of correctly predicted images for the c-th label, $N_c^{predict}$ is the number of predicted images, N_c^{gt} is the number of ground-truth images. Note that the per-class measures treat all classes equal regardless of their sample size, so one can obtain a high performance by focusing on getting rare classes right.

Overall precision/recall (OV-P/R). Unlike per-class metrics, the overall metrics treat all samples equal regardless of their classes.

$$m_{OV-P}(\mathcal{D}) = \frac{\sum_{c=1}^{C} N_c^{correct}}{\sum_{c=1}^{C} N_c^{predict}} \tag{4.19}$$

$$m_{OV-R}(\mathcal{D}) = \frac{\sum_{c=1}^{C} N_c^{correct}}{\sum_{c=1}^{C} N_c^{gt}} \tag{4.20}$$

Macro-F1 (M-F1). The macro-F1 score [138] is the F1 score [107] averaged across all categories.

$$m_{MF1}(\mathcal{D}) = \frac{1}{C}\sum_{c=1}^{C} F_1^c \tag{4.21}$$

Given a category c, the F1 measure, defined as the harmonic mean of precision and recall, is computed as follows:

$$F_1^c = \frac{2P^cR^c}{P^c + R^c} \tag{4.22}$$

where the precision (P^c) and the recall (R^c) are calculated as follows:

$$P^c = \frac{\sum_{i=1}^{N} \mathbb{1}[y_c^{(i)} = \hat{y}_c^{(i)}]}{\sum_{i=1}^{N} \hat{y}_c^{(i)}} \tag{4.23}$$

$$R^c = \frac{\sum_{i=1}^{N} \mathbb{1}[y_c^{(i)} = \hat{y}_c^{(i)}]}{\sum_{i=1}^{N} y_c^{(i)}} \tag{4.24}$$

and $y_c^{(i)} \in \{0, 1\}$

Micro-F1 (m-F1). The micro-F1 score [121] is computed using the equation of F_1^c and considering the predictions as a whole

$$m_{mF1}(\mathcal{D}) = \frac{2\sum_{c=1}^{C}\sum_{i=1}^{N} \mathbb{1}[y_c^{(i)} = \hat{y}_c^{(i)}]}{\sum_{c=1}^{C}\sum_{i=1}^{N} y_c^{(i)} + \sum_{c=1}^{C}\sum_{i=1}^{N} \hat{y}_c^{(i)}} \tag{4.25}$$

According to the definition, macro-F1 is more sensitive to the performance of rare categories while micro-F1 is affected more by the major categories.

4.5.4 Analysis of the initial set of labels

In this subsection, we analyse the initial set of labels for the partial label scenario. We report the results for 4 random seeds to generate the initial set of partial labels. The experiments are performed on MS COCO val2014 with a ResNet-101 WELDON. The results are shown in Table 4.5 and Figure 4.7 for different label proportions and metrics. For every label proportion and every metric, we observe that the model is robust to the initial set of labels.

metric	label proportion									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
MAP	72.20±0.04	74.49±0.02	75.77±0.02	76.57±0.03	77.21±0.01	77.73±0.01	78.16±0.02	78.53±0.03	78.85±0.02	79.14±0.05
M-F1	65.84±0.01	69.32±0.04	70.66±0.02	71.37±0.02	71.88±0.03	72.29±0.04	72.61±0.03	72.89±0.03	73.05±0.06	73.24±0.02
m-F1	70.13±0.04	73.97±0.01	75.36±0.01	76.07±0.03	76.54±0.01	76.91±0.02	77.17±0.04	77.42±0.04	77.58±0.05	77.75±0.04
0-1	22.21±0.12	30.44±0.03	34.26±0.11	36.18±0.07	37.44±0.05	38.46±0.04	39.16±0.07	39.83±0.12	40.34±0.04	40.67±0.02
PC-P	59.82±0.05	68.45±0.10	72.56±0.03	74.88±0.11	76.45±0.04	77.70±0.07	78.59±0.05	79.28±0.10	79.80±0.02	80.22±0.05
PC-R	74.74±0.04	71.14±0.07	69.66±0.04	68.96±0.06	68.64±0.04	68.35±0.04	68.26±0.07	68.23±0.08	68.12±0.09	68.16±0.04
OV-P	62.66±0.09	72.36±0.06	76.81±0.04	79.24±0.10	80.75±0.06	82.01±0.08	82.79±0.14	83.44±0.10	83.94±0.06	84.36±0.04
OV-R	79.62±0.04	75.66±0.04	73.97±0.05	73.14±0.04	72.74±0.04	72.40±0.03	72.21±0.04	72.14±0.01	72.07±0.05	72.15±0.06

Table 4.5: Analysis of the initial set of labels for the partial label scenario. The results are averaged for 4 seeds on MS COCO val2014.

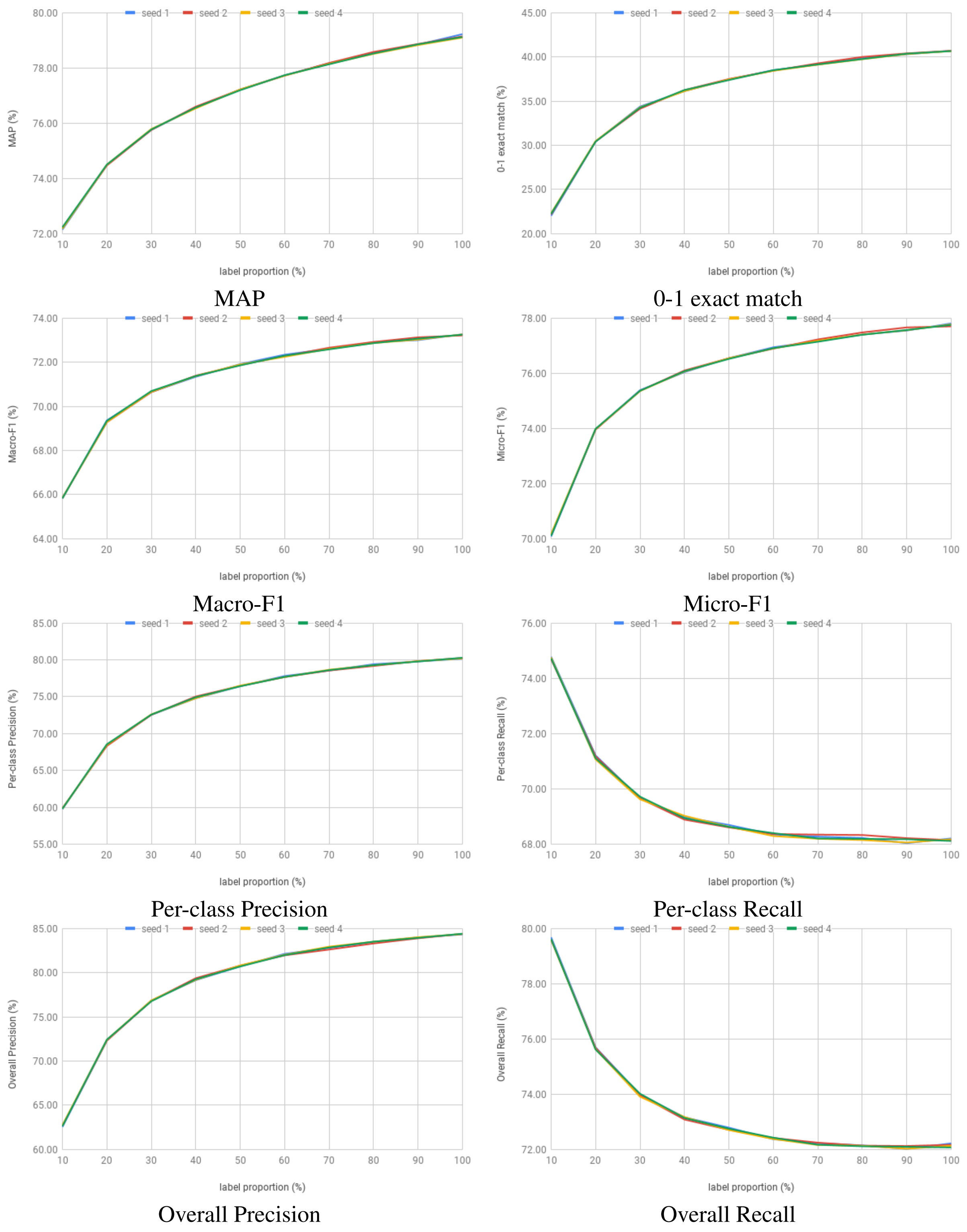


Figure 4.7: Results for different metrics on MS COCO val2014 to analyze the sensibility of the initial label set.

architecture	labels	label proportion									
		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
ResNet-50	partial	61.26	63.78	65.21	66.22	66.97	67.60	68.16	68.58	69.01	69.33
	dense	54.29	59.67	62.50	64.28	65.60	66.68	67.55	68.26	68.80	69.32
	noisy	-	-	-	-	3.75	39.77	56.82	62.93	66.24	69.33
ResNet-50 WELDON	partial	69.91	72.37	73.74	74.53	75.25	75.77	76.25	76.66	77.02	77.28
	dense	62.16	68.04	71.14	73.01	74.17	75.14	75.83	76.42	76.88	77.28
	noisy	-	-	-	-	3.73	52.99	67.08	72.03	74.69	77.29
ResNet-101 WELDON	partial	72.15	74.49	75.76	76.56	77.22	77.73	78.17	78.53	78.84	79.22
	dense	65.22	71.00	73.80	75.44	76.59	77.44	78.08	78.61	78.90	79.24
	noisy	-	-	-	-	3.63	53.10	69.09	74.06	76.85	79.18
ResNeXt-101 WELDON	partial	75.74	77.80	78.95	79.64	80.22	80.61	80.94	81.24	81.48	81.69
	dense	69.03	74.58	77.13	78.50	79.38	80.15	80.65	81.05	81.40	81.71
	noisy	-	-	-	-	3.63	49.26	70.16	75.22	78.28	81.66

Table 4.6: Comparison of the labeling strategies for different label proportions and different architectures on MS COCO val2014.

4.5.5 Analysis of the labeling strategies

In this subsection we analysis the labeling strategies for different network architectures. The results are shown in Table 4.6 and Figure 4.8 on MS COCO dataset. Overall, the results are very similar. For a given proportion of labels, we observe that the partial labels strategy is better that the complete image labels. The improvement increases when the label proportion decreases. The performance of a model learned with noisy labels drops significantly, even for large proportion of clean labels.

In Figure 4.9, we also show the results for different metrics. For MAP, Macro-F1 and Micro-F1, we observe a similar behaviour: the partial labels strategy has better performances than the complete image labels strategy. For the 0-1 exact match metric, we observe that the complete image labels strategy has better performances than the complete image labels strategy. For this metric, the predictions of all the categories must be corrected, so it advantages the complete image labels strategy because some training images have all the labels whereas in the partial labels strategy, none of the training images have all labels. For the precision and recall metrics, the behaviours are different for the complete image labels strategy and the partial labels strategy. We note that the complete image labels strategy has a better per-class/overall precision than the partial labels strategy but is has a lower per-class/overall recall than the partial labels strategy.

Comparison to noisy+ strategy. In Table 4.7, we show results for the noisy+ strategy on Pascal VOC 2007, MS COCO and NUS-WIDE for different metrics. For every dataset, we observe that the noisy+ strategy drops the performances of all the metrics with respect to the model learned with only 10% of clean labels.

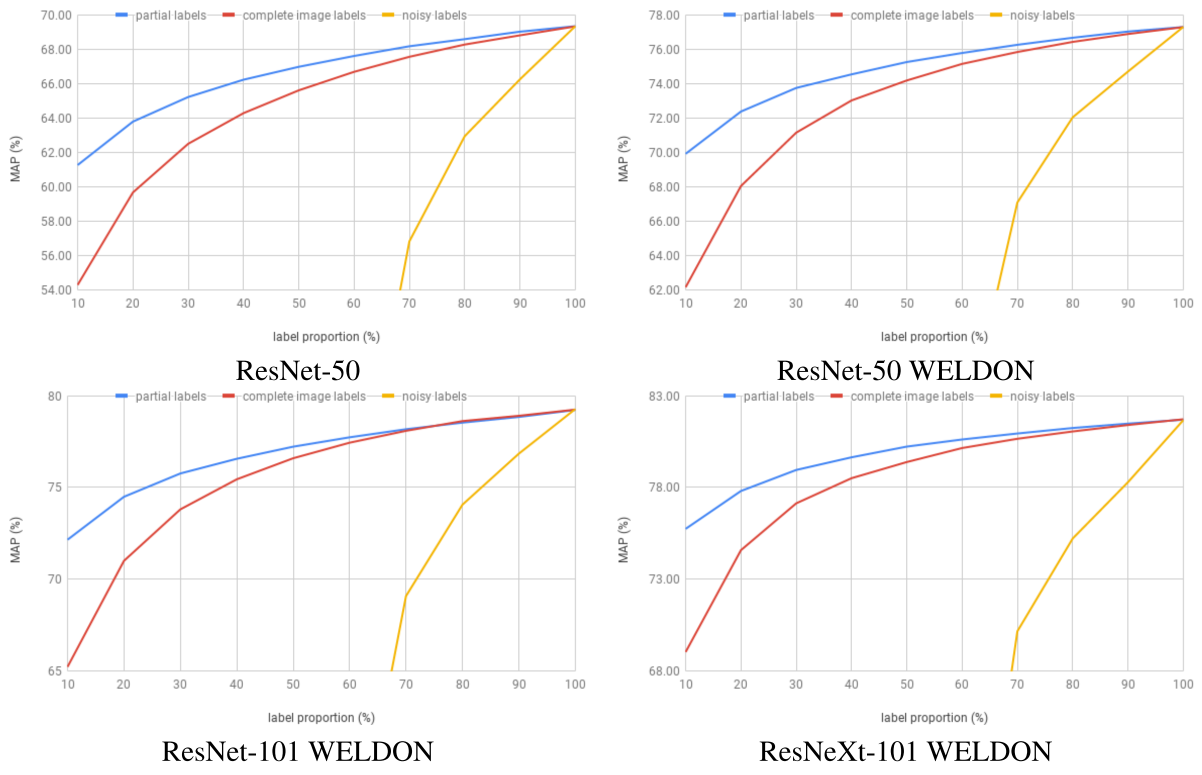


Figure 4.8: Comparison of the labeling strategies for different label proportions and different architectures on MS COCO val2014.

dataset	strategy	clean label	noisy label	MAP	0-1	M-F1	m-F1	PC-P	PC-R	OV-P	OV-R
VOC 2007	clean	100	0	93.93	79.16	88.90	91.12	90.72	87.34	93.40	88.95
	noisy+	97.1	2.9	90.94	62.21	78.11	78.62	95.41	68.64	97.20	66.00
	partial 10%	10	0	89.09	47.46	74.55	77.84	63.35	94.16	66.02	94.81
MS COCO	clean	100	0	79.22	40.69	73.26	77.80	80.16	68.21	84.31	72.23
	noisy+	97.6	2.4	71.60	20.28	38.62	33.72	91.76	28.17	97.34	20.39
	partial 10%	10	0	72.15	22.04	65.82	70.09	59.76	74.78	62.56	79.68
NUS-WIDE	clean	100	0	54.88	42.29	51.88	71.15	58.54	49.33	73.83	68.66
	noisy+	98.6	1.4	47.44	36.07	18.83	28.53	59.71	13.95	83.72	17.19
	partial 10%	10	0	51.14	25.98	51.36	65.52	41.80	69.23	53.62	84.19

Table 4.7: Comparison with a webly-supervised strategy (noisy+) on MS COCO. Clean (resp. noisy) means the percentage of clean (resp. noisy) labels in the training set. Noisy+ is a labeling strategy where there is only one positive label per image.

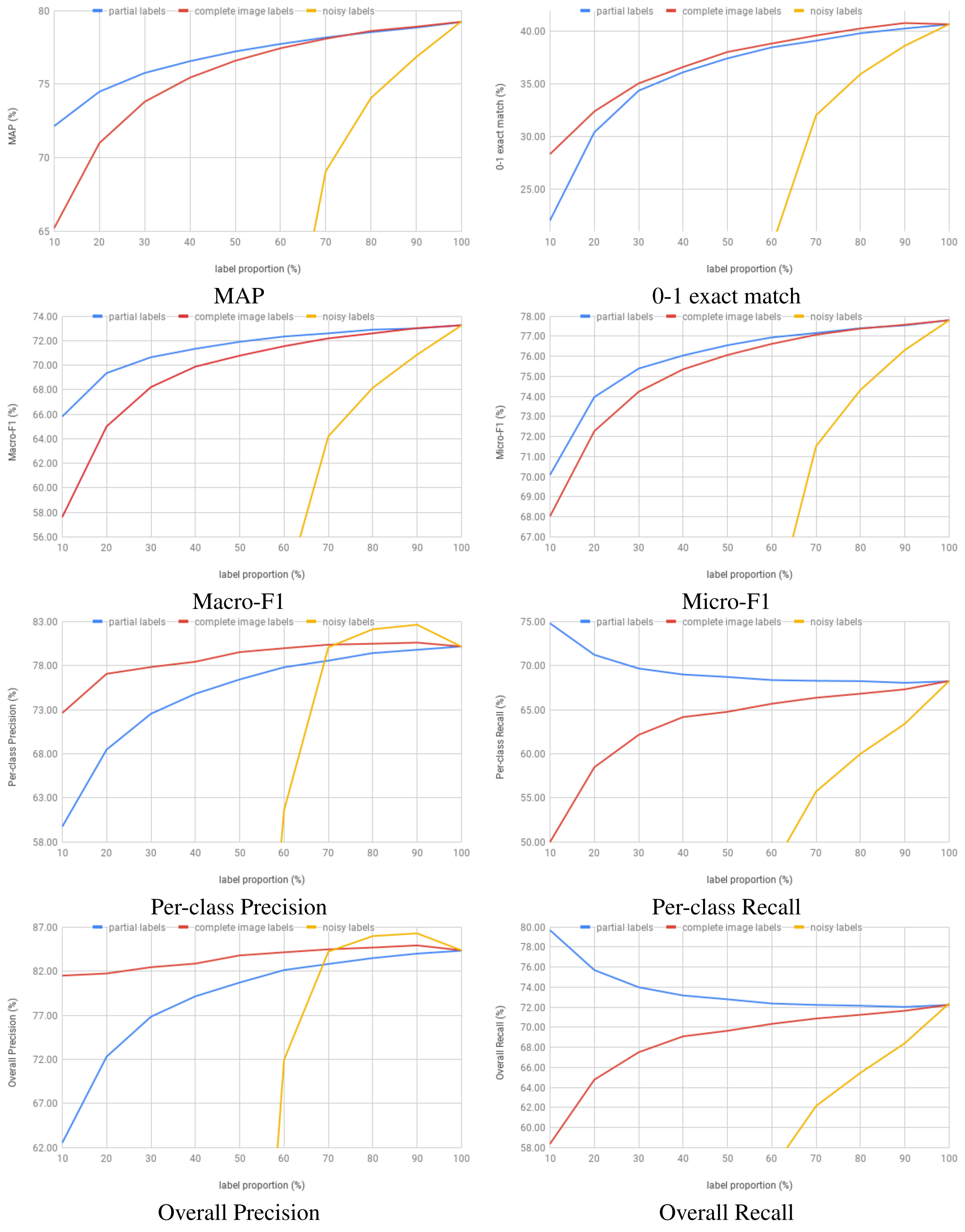


Figure 4.9: Comparison of the labeling strategies for different metrics on MS COCO val2014.

4.5.6 Comparison of the loss functions

In this subsection, we analyse the performances of the BCE and partial-BCE loss functions for different metrics. The results on MS COCO (resp. Pascal VOC 2007) are shown in Figure 4.11 (resp. Figure 4.13) and the improvement of the partial-BCE with respect to the BCE is shown in Figure 4.12 (resp. Figure 4.14). We observe that the partial-BCE significantly improves the performances for MAP, 0-1 exact match, Macro-F1 and Micro-F1 metrics. We note that the improvement is bigger when the label proportion is lower. The proposed loss also improves the (overall and per-class) recall for both datasets. On Pascal VOC 2007, it also improves the overall and per-class precision. However, we observe that the

We observe that decreasing the proportion of known labels can slightly improves the performances with respect to the model trained with all the annotations. This phenomenon is because of the tuning of the learning rate and the hyperparameter γ (Figure 4.6). Note that the BCE and the partial-BCE have the same results for the label proportion 100% because they are equivalent by definition. We used the same training setting (learning rate, weight decay, etc.) as [33] for each model and dataset. In Figure 4.10, we observe that using a learning rate of 0.02 increases the performance and leads to a monotone increase of the performance with respect to the label proportion, but the optimal learning rate depends on the dataset. It is possible to improve the results by tuning carefully these hyperparameters, but we observe that the partial-BCE is still better than the BCE for a large range of LRs and for small label proportions which is the main focus of this work.

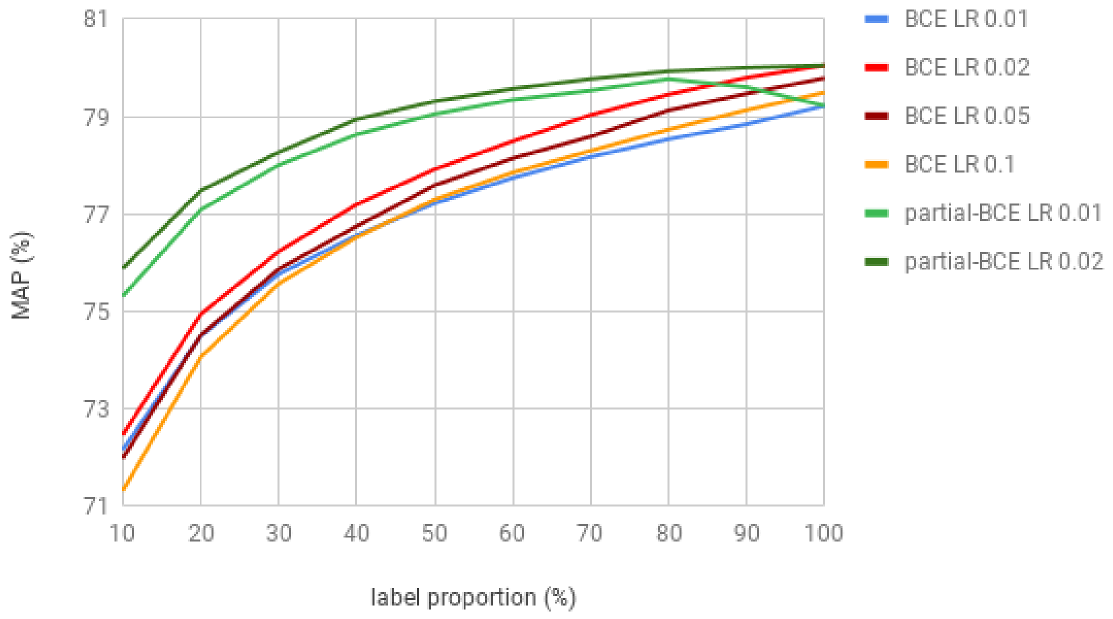
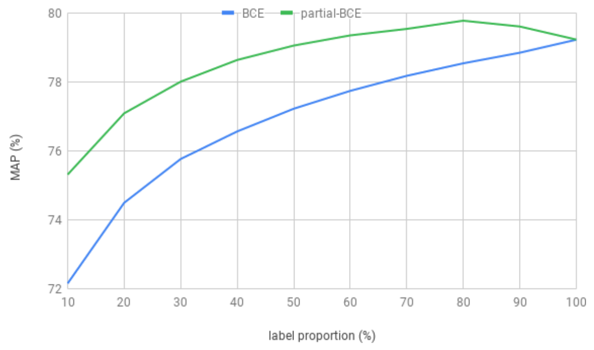
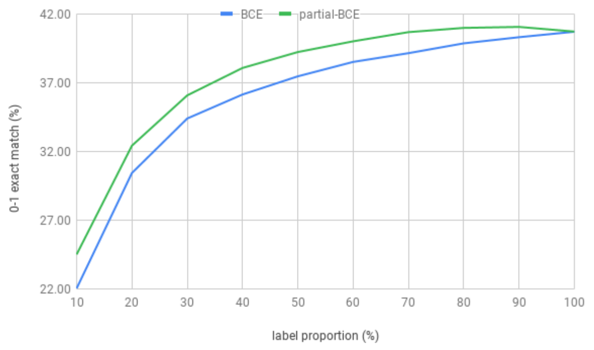


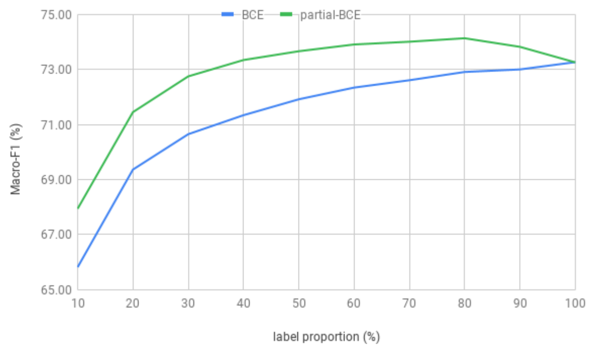
Figure 4.10: Analysis of the learning rate on MS COCO dataset.



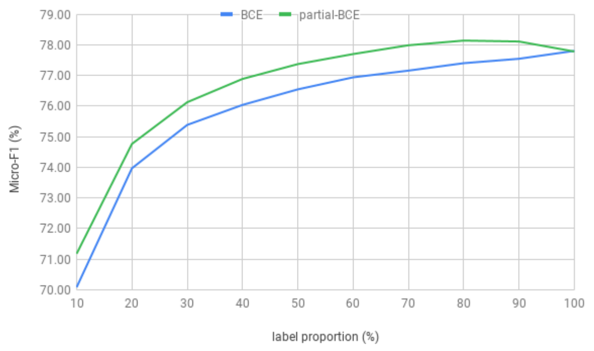
MAP



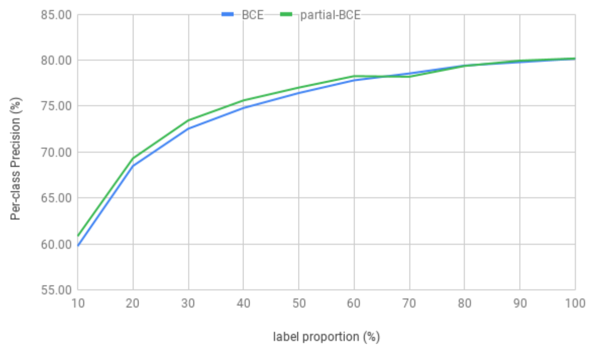
0-1 exact match



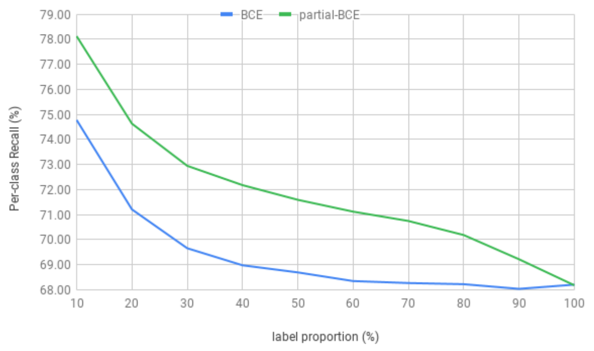
Macro-F1



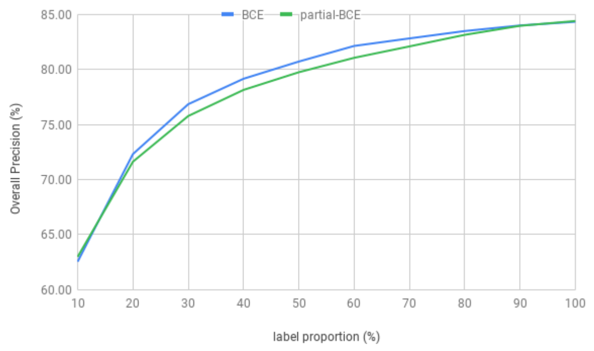
Micro-F1



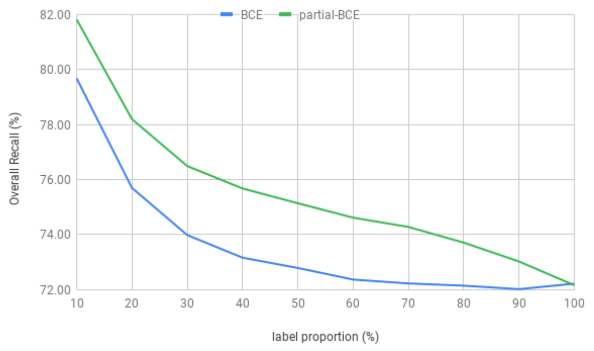
Per-class Precision



Per-class Recall

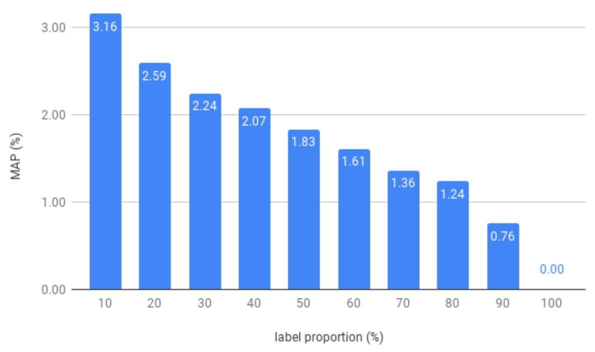


Overall Precision

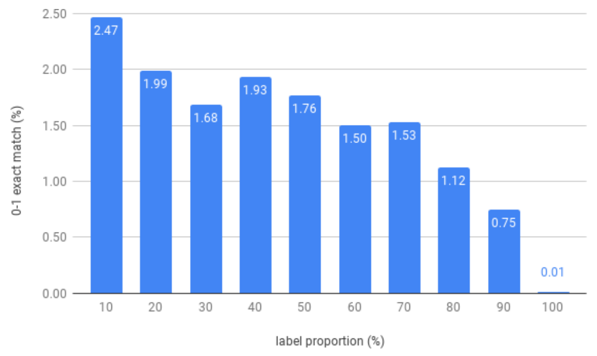


Overall Recall

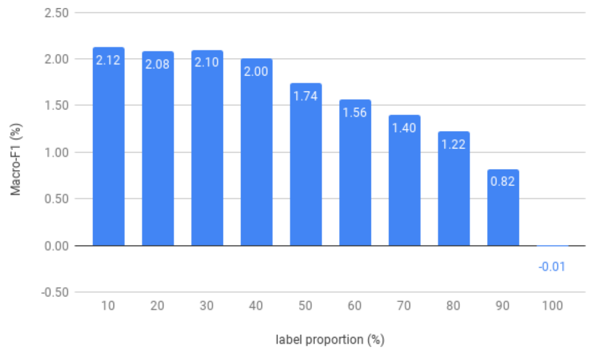
Figure 4.11: Results for different metrics on MS COCO val2014.



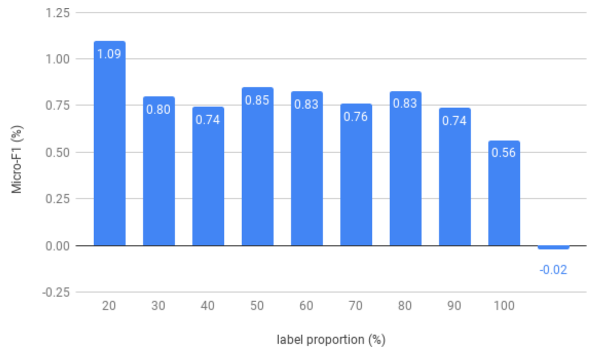
MAP



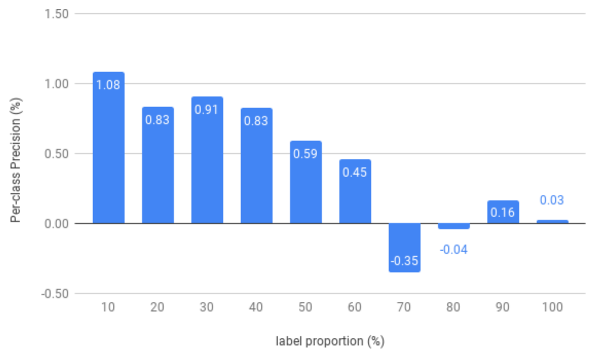
0-1 exact match



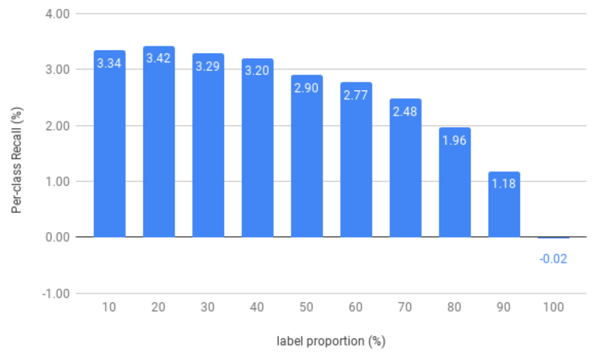
Macro-F1



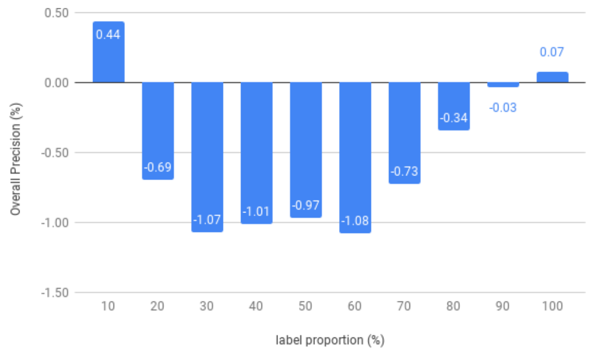
Micro-F1



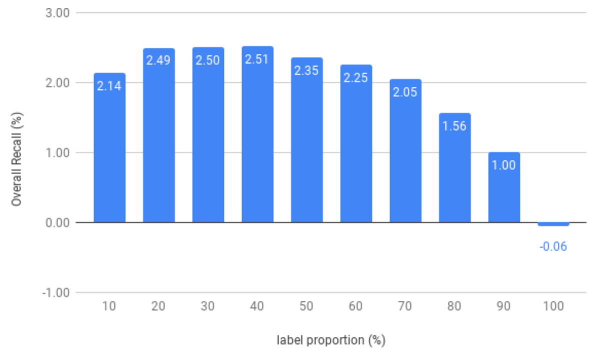
Per-class Precision



Per-class Recall

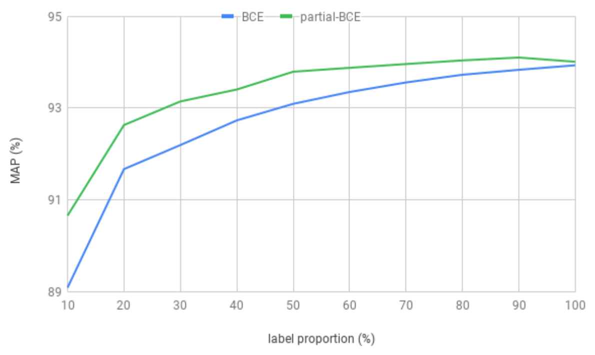


Overall Precision

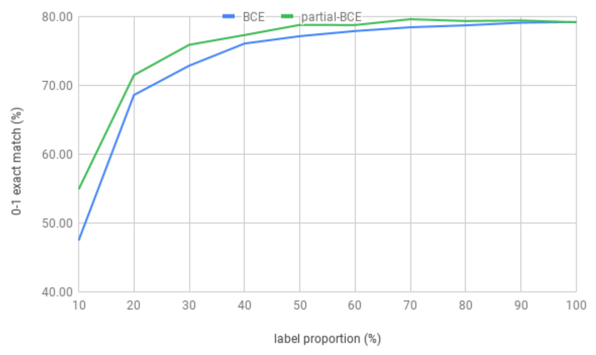


Overall Recall

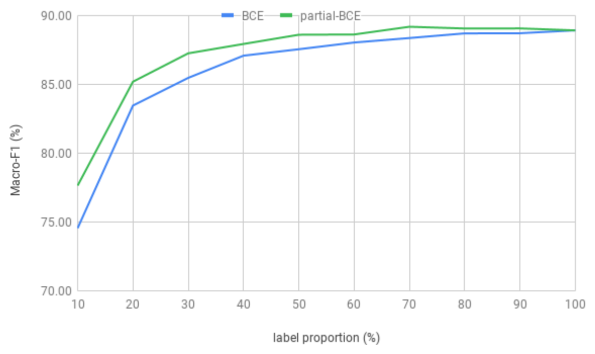
Figure 4.12: Improvement analysis between partial-BCE and BCE for differents metrics on MS COCO val2014.



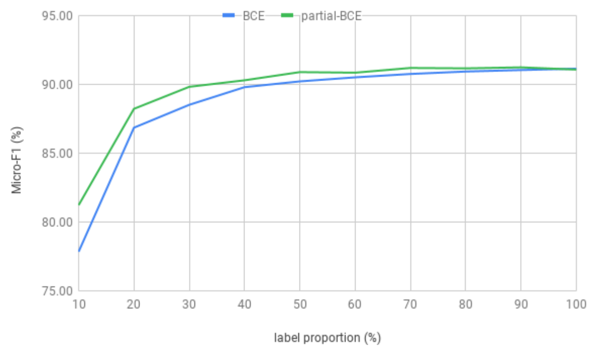
MAP



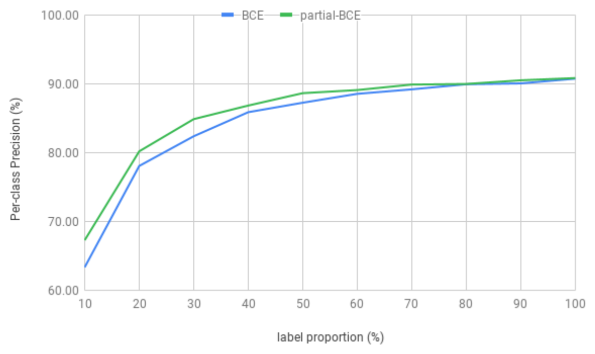
0-1 exact match



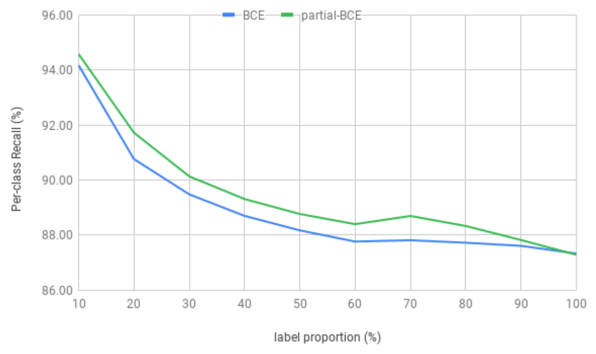
Macro-F1



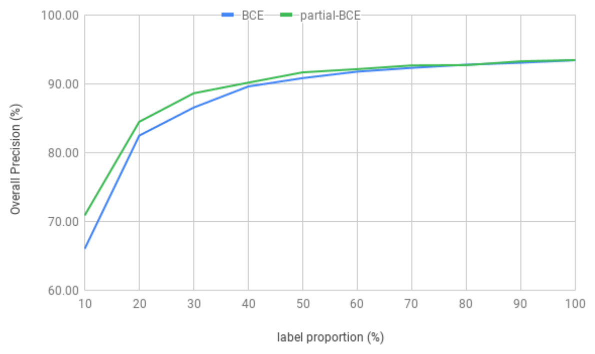
Micro-F1



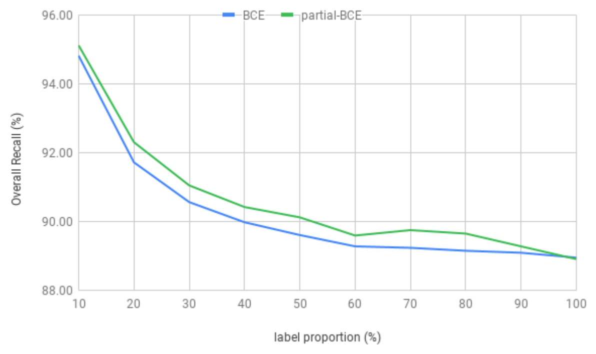
Per-class Precision



Per-class Recall



Overall Precision



Overall Recall

Figure 4.13: Results for different metrics on Pascal VOC 2007.

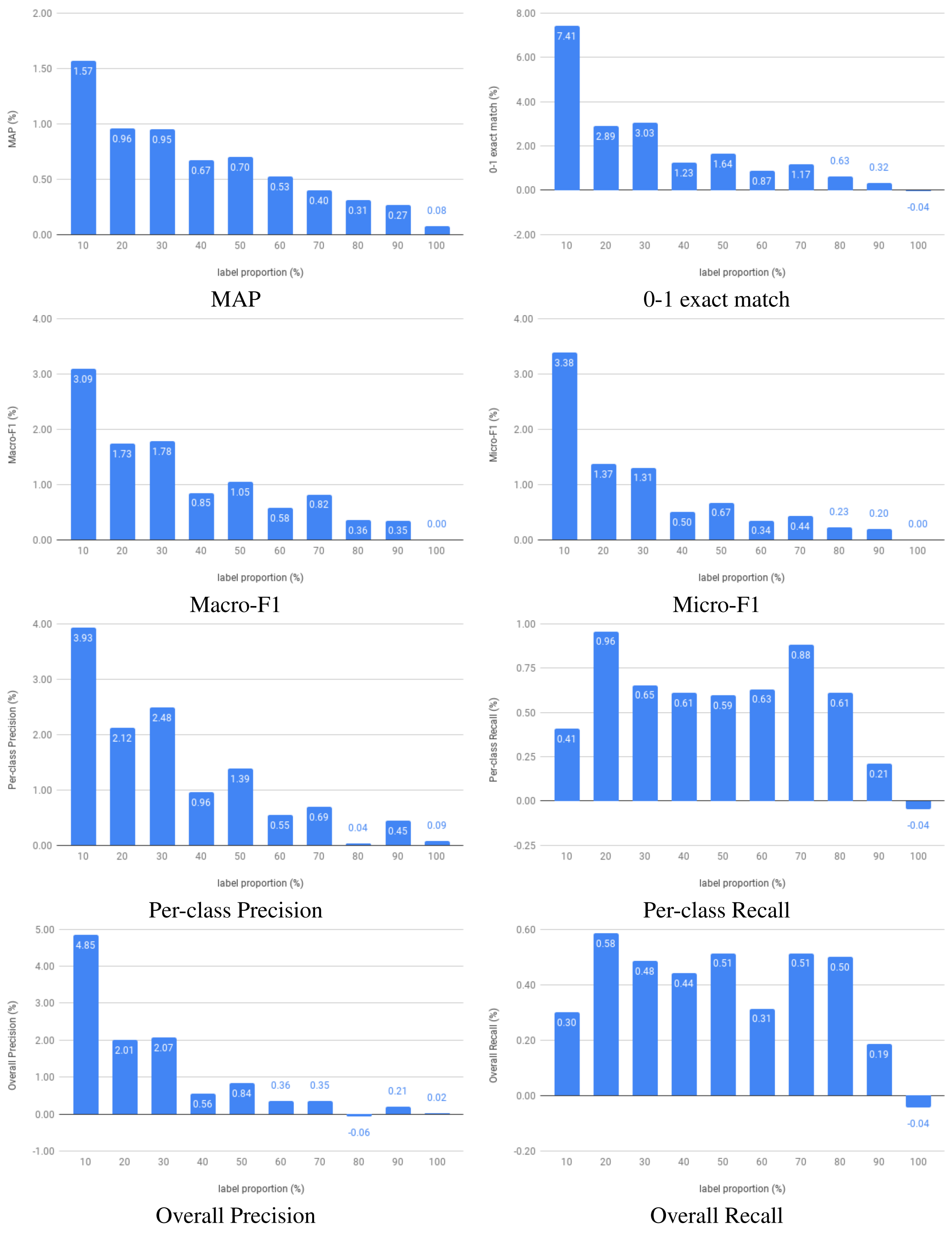


Figure 4.14: Improvement analysis between partial-BCE and BCE for differents metrics on Pascal VOC 2007.

4.5.7 Analysis of the loss function

In this subsection, we analyze the hyperparameter of the loss function for several network architectures. The models are trained on the train2014 set minus 5000 images that are used as validation set to evaluate the performances. The Figure 4.15 shows the results on MS COCO. We observe a similar behavior for all the architectures. Overall, using a normalization value $g(0.1)$ between 3 and 50 significantly improves the performances with respect to the normalization by the number of categories ($g(0.1) = 1$). The loss is robust to the value of this hyperparmeter.

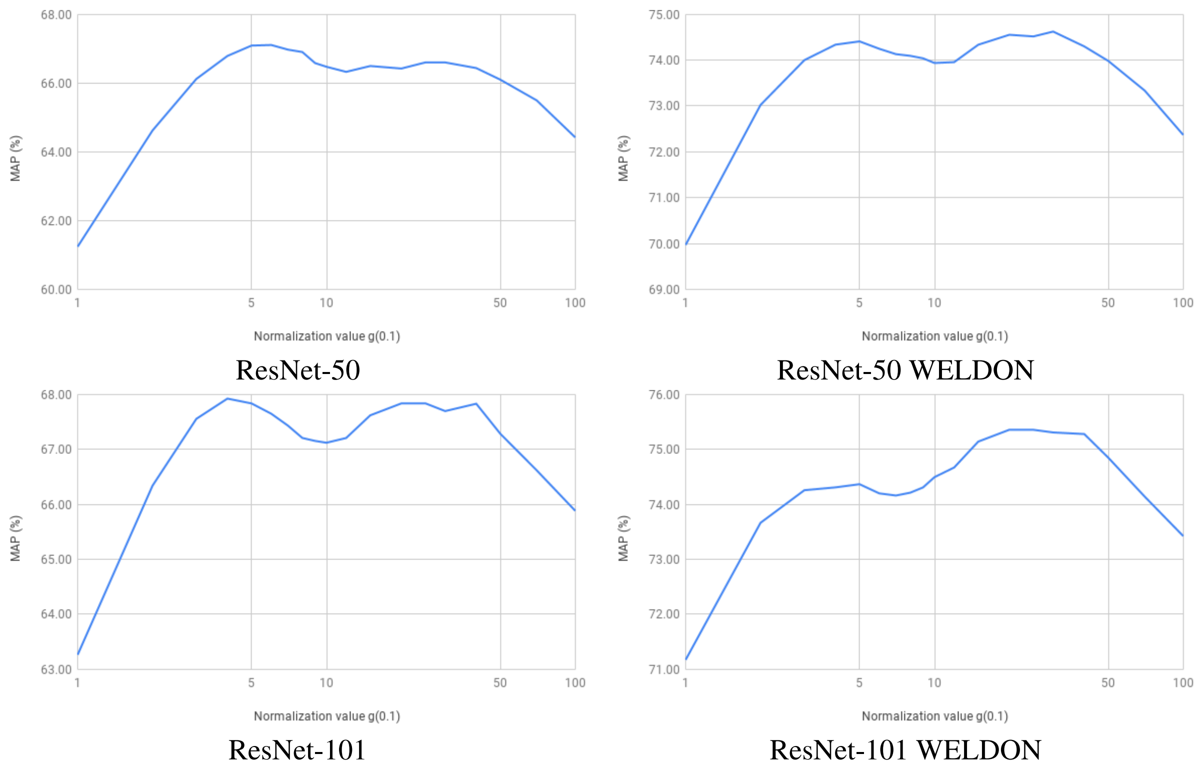


Figure 4.15: Analysis of the normalization value for 10% of known labels (*i.e.* $g(0.1)$) on MS COCO. (x-axis log-scale)

4.5.8 Comparison to existing model for missing labels

As pointed out in the related work subsection, most of the existing models to learn with missing labels are not scalable and do not allow experiments on large-scale dataset like MS COCO and NUS-WIDE. We compare our model with the APG-Graph model [137] that models structured semantic correlations between images on the Pascal VOC 2007 dataset. Unlike our method, the APG-Graph model does not allow to fine-tune the ConvNet.

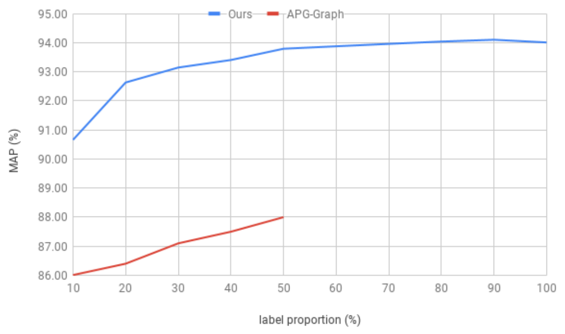


Figure 4.16: Comparison with APG-Graph model on Pascal VOC 2007 for different proportion of known labels.

4.5.9 What is the best strategy to predict missing labels?

This subsection extends the subsection 4.3 of this chapter. First, to compute the Bayesian uncertainty, we use the setting used in the original paper [64]. The results for different strategies and hyperparameters are shown in Table 4.8. G defines how the examples are selected during training. In Section 4.3.3, we only explain how to find the solution with respect to $\mathbf{v}$. G depends on the strategy and is defined as:

$$G(\mathbf{v};\theta) = -\sum_{i=1}^{N}\sum_{c=1}^{C} v_{ic} \log\left(\frac{1}{1+e^{-\theta}}\right)$$

for strategy [a].

For strategy [a] and [d], we observe that using a small threshold is better than a large threshold. On the contrary, for strategy [c] we observe that using a large threshold is better than a small threshold, but the results are worse than strategy [a]. For strategy [b], labeling a large proportion of labels per mini-batch is better than labeling a small proportion of labels. For strategy [e], we note that using a GNN improves the performances of the model and the model is more robust to the threshold hyperparameter θ.

Relabeling	MAP	0-1	Macro-F1	Micro-F1	label prop.	TP	TN	GNN
2 steps (no curriculum)	-1.49	6.42	2.32	1.99	100	82.78	96.40	✓
[a] Score threshold $\theta = 1$	0.00	11.31	3.71	4.25	97.87	82.47	97.84	✓
[a] Score threshold $\theta = 2$	0.34	11.15	4.33	4.26	95.29	85.00	98.50	✓
[a] Score threshold $\theta = 5$	0.31	5.02	2.60	1.83	70.98	96.56	99.44	✓
[b] Score proportion $\theta = 0.1$	0.45	-1.20	-0.28	-0.68	26.70	99.28	99.19	✓
[b] Score proportion $\theta = 0.2$	0.36	0.20	0.70	0.10	42.09	98.35	99.33	✓
[b] Score proportion $\theta = 0.3$	0.28	0.91	1.09	0.37	55.63	97.82	99.38	✓
[b] Score proportion $\theta = 0.4$	0.55	2.95	2.33	1.28	67.41	96.87	99.38	✓
[b] Score proportion $\theta = 0.5$	0.22	4.02	2.76	1.74	77.40	95.52	99.30	✓
[b] Score proportion $\theta = 0.6$	0.41	6.17	3.63	2.52	85.37	93.16	99.15	✓
[b] Score proportion $\theta = 0.7$	0.35	7.49	3.83	3.07	91.69	89.40	98.81	✓
[b] Score proportion $\theta = 0.8$	0.17	8.40	3.70	3.25	96.24	84.40	98.10	✓
[c] Postitive only - score $\theta = 1$	-1.61	-31.75	-18.07	-18.92	16.79	36.42	-	✓
[c] Postitive only - score $\theta = 2$	-0.80	-21.31	-10.93	-12.08	14.71	47.94	-	✓
[c] Postitive only - score $\theta = 5$	0.31	-4.58	-1.92	-2.23	12.01	79.07	-	✓
[d] Ensemble score $\theta = 1$	-0.31	10.16	3.61	3.94	97.84	82.12	97.76	✓
[d] Ensemble score $\theta = 2$	0.23	11.31	4.16	4.33	95.33	84.80	98.53	✓
[d] Ensemble score $\theta = 5$	0.27	3.78	2.38	1.53	70.77	96.56	99.44	✓
[e] Bayesian uncertainty $\theta = 0.1$	0.26	1.84	1.36	0.64	22.63	25.71	99.98	
[e] Bayesian uncertainty $\theta = 0.2$	0.29	8.49	4.05	3.66	60.32	48.39	99.82	
[e] Bayesian uncertainty $\theta = 0.3$	0.34	10.15	4.37	3.72	77.91	61.15	99.24	
[e] Bayesian uncertainty $\theta = 0.4$	0.30	9.05	4.17	3.37	87.80	68.56	98.70	
[e] Bayesian uncertainty $\theta = 0.5$	0.26	8.32	3.83	3.05	92.90	70.96	98.04	
[e] Bayesian uncertainty $\theta = 0.1$	0.36	2.71	1.91	1.22	19.45	38.15	99.97	✓
[e] Bayesian uncertainty $\theta = 0.2$	0.30	10.76	4.87	4.66	57.03	62.03	99.65	✓
[e] Bayesian uncertainty $\theta = 0.3$	0.59	12.07	5.11	4.95	79.74	68.96	99.23	✓
[e] Bayesian uncertainty $\theta = 0.4$	0.43	10.99	4.88	4.46	90.51	70.77	98.57	✓
[e] Bayesian uncertainty $\theta = 0.5$	0.45	10.08	3.93	3.78	94.79	74.73	98.00	✓

Table 4.8: Analysis of the labeling strategy of missing labels on Pascal VOC 2007 val set. For each metric, we report the relative scores with respect to a model that does not label missing labels. TP (resp. TN) means true positive (resp. true negative). Label proportion is the proportion of training labels (clean + weak labels) used at the end of the training. For the strategy labeling only positive labels, we report the label accuracy instead of the TP rate.

4.5.10 Final results

In Figure 4.17, we show the results of our final model that uses the partial-BCE loss, the GNN and the labeling of missing labels. We compare our model to two baselines: (a) a model trained with the standard BCE where the data are labeled with the partial labels strategy (blue) and (b) a model trained with the standard BCE where the data are labeled with the complete image labels strategy (red). We observe that our model has better performances than the two baselines for most of the metrics. In particular, our final model has significantly better 0-1 exact match performance than the baseline (b), whereas the baseline with partial labels (a) has lower performance than the baseline (b). We note that the overall precision of our model is worse than the baseline (b), but the overall recall of our model is largely better than the baseline (b).

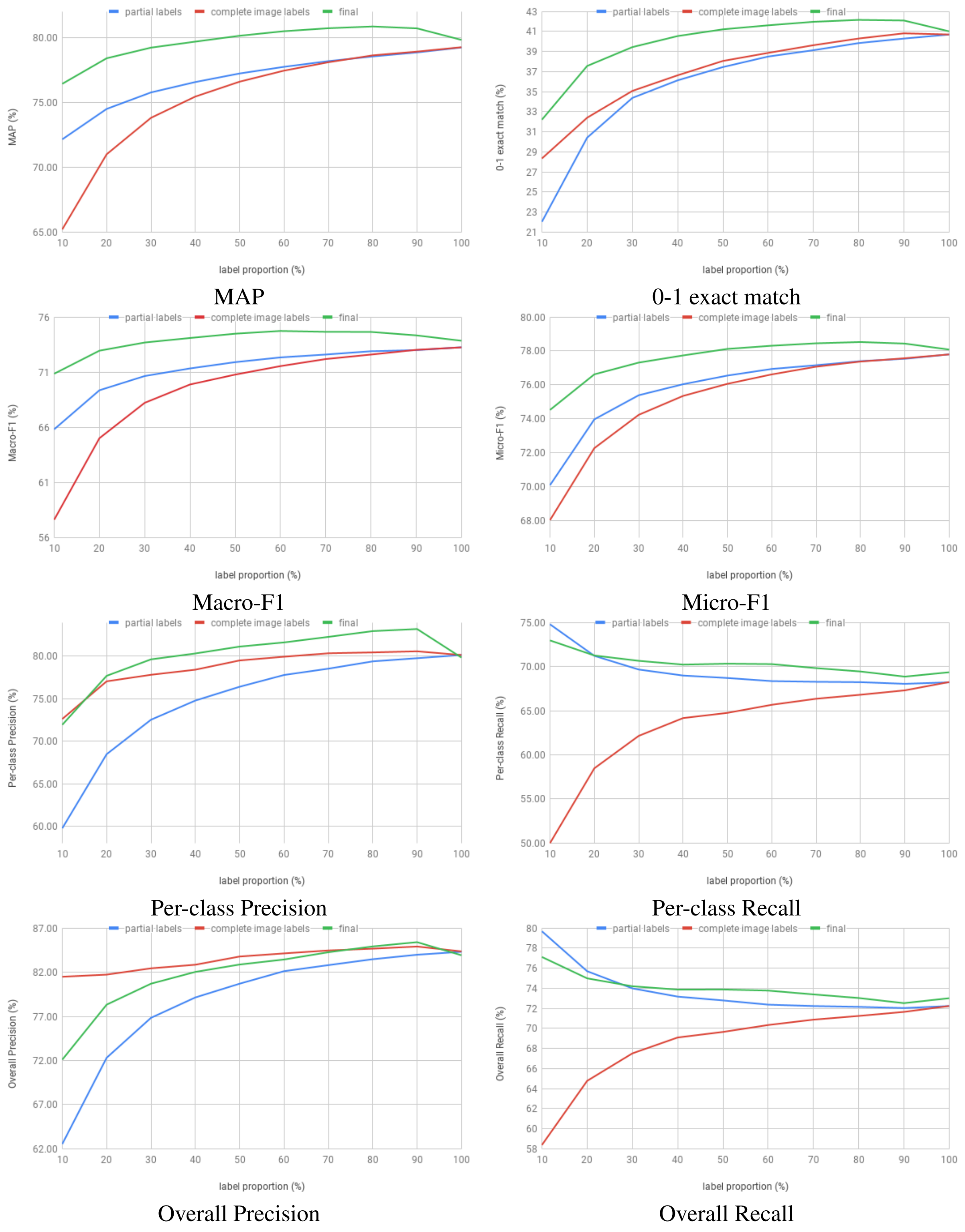


Figure 4.17: The results of our final model with two baselines (complete image labeling and BCE with partial labels) for different metrics on MS COCO val2014.

4.6 Summary

In this work, we present a scalable approach to end-to-end learn a deep network with partial labels. More specifically, we propose a framework for learning from partially labeled image data with a multi-label classifier. We show that our curriculum learning model using bayesian uncertainty is an accurate strategy to label missing labels. In the future work, one could combine several datasets whith shared categories to learn with more training data.

Chapter 5

A Flexible Flow-Based Latent Variable Model for Asynchronous Action Sequences

This chapter focuses on the asynchronous action prediction problem. Given a history of previous actions, the goal is to model the distribution of future actions, including action times as well as action categories. Unlike existing approaches that rely on restrictive parametric distributions, our approach makes use of the normalizing flow model to generate flexible distributions of event inter-arrival times. Furthermore, with temporal latent variables, our model is also capable of capturing highly complex temporal dependence structures. The proposed model is evaluated on benchmark action sequence datasets and shows superior performance over existing methods.

This chapter is published as *Point Process Flows* in Learning with Temporal Point Processes, NeurIPS 2019 workshop [91].

5.1 Overview

Predicting action sequences of both *what* and *when* to happen is a fundamental inference task in human activity understanding. We argue that the challenge of this task stems from four aspects: 1) the complex dependence between the past and the future actions; 2) the complex structure of an action, *i.e*. how the timing of an action is related to its label or vice versa; 3) the multi-modal nature of future uncertainty, *i.e*. given a sequence of past actions, multiple different sequences of future events could be of substantial possibility; 4) the diverse and complex distributions of future action times.

Previous models, especially video frame-based ones, deal with the problem on a regularly spaced time grid with short intervals between time stamps [1, 63, 89, 126]. However, actions are usually sparsely and irregularly spaced in terms of time. Such frame-based approaches could be computationally inefficient and limit the model's ability to make long-term predictions across multiple actions.

Recently, there has been growing interest in Temporal Point Process (TPP) models in the machine learning and vision communities [29, 92, 97, 114]. A TPP models asynchronous action sequences by directly modeling the distribution of the time intervals between actions. We argue that the temporal point process approach, through directly modeling the time intervals between actions, can work around the limits of a predefined time grid due to the sparse and irregular nature of action timing.

An important variant of TPP for action anticipation is the Marked Temporal Point Process (MTPP), which models future action timing in the same way as TPP, and more importantly, provides a framework for modeling action categories. However, existing works directly model the marginal distributions of future action time and action category by making oversimplified independence assumptions about the joint distribution. They also neglect the multi-modal possibility of the future [29, 114]. Recently, Mehrasa et al. [92] proposed a marked temporal point process-based approach to model the sequence of action times and categories for action anticipation. However, the model assumes that the time of the next action follows an exponential distribution, and future actions are conditionally independent of the past given the latent variable. We hypothesize that these assumptions restrict the model's expressiveness in modeling the time distributions of actions as well as anticipatory reasoning of future actions based on history.

In this work, we propose a recurrent latent variable model for action sequence generation and anticipation (see Figure 5.1) that directly addresses the challenges mentioned before. In our proposed framework, the stochastic latent variable encodes high-level information about possible future actions as well as how the future action times and categories are correlated. When this latent code is combined with the history of actions, it can be decoded into independent action category and time distributions that are consistent with the past actions. It also helps to produce a flexible distribution for future action times, which can be further improved by utilizing the normalizing flow to construct complex time distributions. The proposed model can be trained in a variational filtering framework; it uses a separate inference network to propose the posterior distribution of the latent variable conditioned on current observations and maximizes a variational lower bound. The recurrent latent variable model achieves state-of-the-art performance on various tasks including density estimation, short-term prediction, and long-term conditional generation.

In summary, the contributions of this work are:

- We propose a flexible flow-based latent variable model for action sequences capable of capturing (1) the complex dependency between past and future actions; (2) multi-modal distribution of future actions; (3) complex time distribution (3) complex structure of an action timing and label.

- Through a wide range of experiments, we demonstrate how our proposed model can be used for density estimation, short-term prediction, and long-term conditional generation.

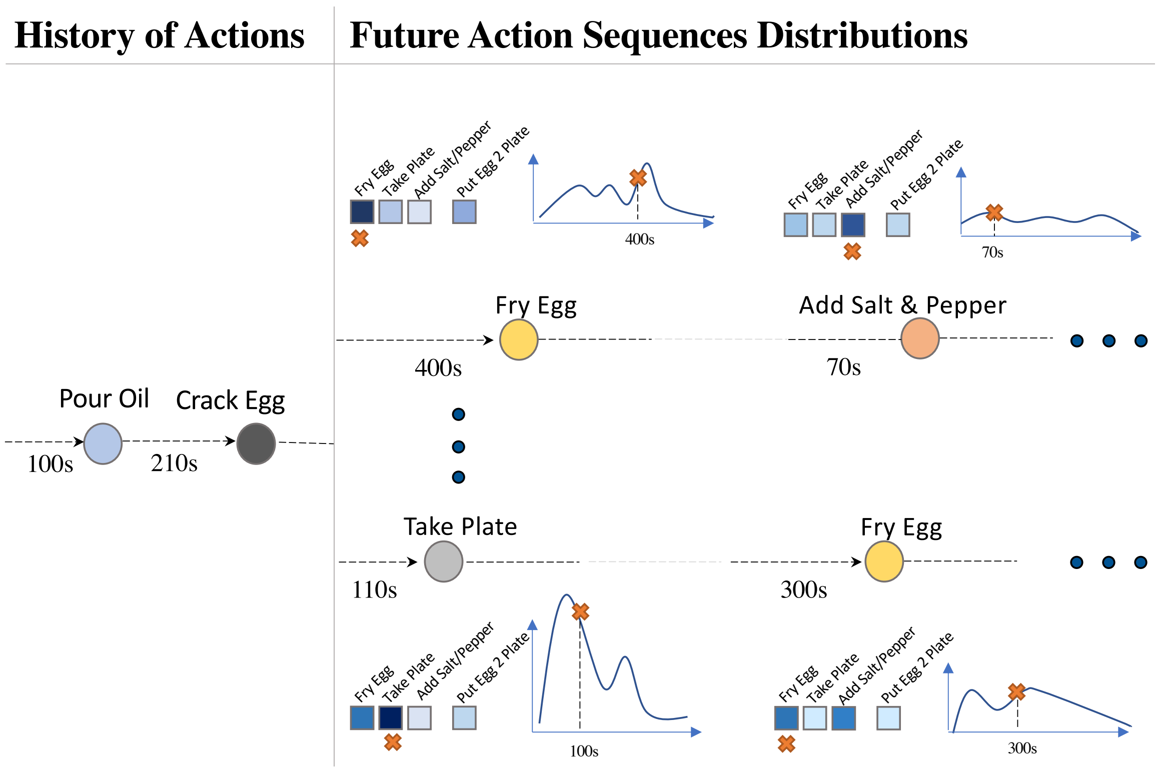


Figure 5.1: We propose a flexible flow-based recurrent latent variable model for action sequences. Our framework takes a sequence of actions as observations and models the distribution of future actions which enables down-stream tasks such as density estimation, short-term prediction and long-term conditional generation.

5.2 Related work

Action Anticipation. In activity anticipation, the goal is to predict the next action before it starts [1, 63, 89, 92, 113, 126]. Vondrick *et al.* [126] propose a framework for learning a representation for future frames in a video, which is used to infer what type of action is going to happen next. Mahmud *et al.* [89] propose a hierarchical model for jointly predicting the next activity type and starting time in a video. Their model consists of three branches that aim to capture scene context, relationships of past activities in the sequence, and inter-activity time context to predict future action. Sener *et al.* [113] introduce a hierarchical zero-shot action prediction framework that generalizes instructional knowledge from text-corpora and transfers the knowledge to the visual domain to anticipate future actions.

Fraha *et al.* [1] propose two methods for long-term activity predictions: (1) An RNN-based framework that takes the history of past activities and predicts the next action label and duration. In order to predict multiple steps in the future, the predicted activities are fed back to the RNN as observations. (2) The second approach is a CNN-based model that predicts a fixed sequence of future activities in one shot, in the form of a matrix encoding future action labels and duration. Recently, Ke *et al.* [63] propose a time-conditioned framework for long-term action anticipation. The model also utilizes a multi-scale attention mechanism to extract features from the observed sequence of actions. Recent work by Mehrasa *et al.* [92] introduce a conditional variational auto-encoder model for modeling future action timing and category. More specifically, in a sequence, their model predicts two distributions over the next action label and timing: a categorical distribution for action label and an exponential distribution for action timing.

Different from previous works which are frame-based [1, 63, 89, 113, 126] or make simplified assumptions on action timing distribution [1, 92], we address the problem of modeling future action timing and label distribution in asynchronous action sequences and employ normalizing flow in constructing a time distribution which makes our model capable of capturing highly complex temporal distributions.

Early Action Prediction. Early action recognition is the task of recognizing an ongoing action as early as possible before its fully executed. Many efforts have been developed for early action prediction [37, 71, 84, 111, 129]. Kong *et al.* [71] propose a deep sequential context network to capture the evolution of video frames and reconstruct missing frames of partial observations for action prediction. Gammulle *et al.* [37] introduce a recurrent generative adversarial network (GAN) framework that jointly learns to predict future video frames as well as action anticipation. Wang *et al.* [129] propose a teacher-student learning framework for early action prediction. A teacher model aims to recognize activity from fully observed videos, and a student model aims to recognize an ongoing action as early as possible given partially observed videos. A teacher-student block is also utilized for distilling knowledge from teacher to student. Our work is different from early action prediction since we anticipate the occurrence of an action before it starts.

Event Prediction in Asynchronous Sequences. Event prediction has a very rich literature in Marked Temporal point processes (MTPPs) [22]. In MTPP, the task of event prediction refers to predicting the conditional distribution of the next event in a sequence given the history of past observations. More specifically, MTPP models the conditional joint distribution of event time and mark, given the history of past events. In this formulation, mark represents any information attached to an event, which in the case of action prediction, we are interested in representing action label as a mark. In this line of work, classic work designs a functional form for a point process based on a prior knowledge over the underlying generative mechanism of the system [46, 54, 69]. Recently, learning point process distributions using recurrent neural networks (RNNs) to encode the history information, has received an increasing amount of attention [29, 60, 93, 144, 145]. In this line of work, history information is encoded and exploited in learning the parameters of the event's timing and category distribution. In very recent work, Shchur *et al.* [114] proposed to estimate the time distribution of point processes with a mixture of Gaussian distribution where the parameters are obtained via an RNN encoding the history. They also use the representation (provided by RNN) to learn the parameters of a categorical distribution for mark (action label).

5.3 Preliminaries

5.3.1 Temporal point process

A temporal point process (TPP) [22] is a mathematical framework for modeling asynchronous sequences of actions. It is a stochastic process whose realization is a sequence of discrete events in time $t_{1:n} = (t_1, t_2, ..., t_n)$ where t_n is the time when the n^{th} event occurred.

A temporal point process distribution is most commonly modeled by specifying the probability density function of the time of the next event:

$$f(t|\mathcal{H}_t) = \lambda(t|\mathcal{H}_t) \exp\left\{-\int_{t_{n-1}}^{t} \lambda(u|\mathcal{H}_u)\ \mathrm{d}u\right\}, \tag{5.1}$$

where $\lambda(t|\mathcal{H}_t)$ is the conditional intensity function which encodes the expected rate of an event happening in a small area around t and $\mathcal{H}_t$ is the history of past events up to time t *i.e.* $\mathcal{H}_t = \{t_1, t_2, ..., t_{n-1}|t_{n-1} < t\}$. Various methods explored different design choices of intensity function to capture the phenomena of interest [46, 54, 69]. Recently, there has been growing interest in the deep learning community to model the distribution of time intervals between events in temporal point processes [29, 60, 93, 134, 145].

A related extension of temporal point processes to action anticipation, is Marked Temporal Point Process (MTPP) which contains a *discrete* event label x_i marked at each time step t_i. Human activity sequences with discrete action labels perfectly fit in the framework of MTPP. Intuitively, the internal dynamics of a system often call for events of specific types to take place at specific times. We argue that learning a good representation of the internal state of a system from observations is critical to modeling and generating asynchronous marked event sequences.

5.3.2 Normalizing flow

Normalizing flows are generative models that allow both density estimation and sampling. They map simple distributions to complex ones using bijective functions. Specifically, if our interest is to estimate the density function $p_{\boldsymbol{X}}$ of a random vector $\boldsymbol{X} \in \mathbb{R}^d$, then normalizing flows assume $\boldsymbol{X} = g_\theta(\boldsymbol{Z})$, where $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is a bijective function, and $\boldsymbol{Z} \in \mathbb{R}^d$ is a random vector with a tractable density function $p_{\boldsymbol{Z}}$. We further denote the inverse of g_θ by f_θ. On one hand, the probability density function can be evaluated using the change of variables formula:

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = p_{\boldsymbol{Z}}(f_\theta(\boldsymbol{x}))\left|\det\left(\frac{\partial f_\theta}{\partial \boldsymbol{x}}\right)\right|, \tag{5.2}$$

where $\partial f_\theta/\partial \boldsymbol{x}$ denotes the Jacobian matrix of f_θ. On the other hand, sampling from $p_{\boldsymbol{X}}$ can be done by first drawing a sample from the simple distribution $\boldsymbol{z} \sim p_{\boldsymbol{Z}}$, and then apply the bijection $\boldsymbol{x} = g_\theta(\boldsymbol{z})$.

Given the expressive power of deep neural networks, it is natural to construct g_θ as a neural network. However, it requires the bijection g_θ to be invertible, and the determinant of the Jacobian matrix should be efficient to compute. Several methods have been proposed along this research direction [27, 28, 67, 68, 100, 106]. An extensive overview of normalizing flow models is given by [70].

5.3.3 Continuous normalizing flow

From a dynamical systems perspective, the residual network can be regarded as the discretization of an ordinary differential equation (ODE) [11, 45, 85]. Inspired by that, Chen *et al.* [13] propose neural ODE, where the continuous dynamics of hidden units is parameterized using an ordinary differential equation specified by a neural network:

$$\frac{d\boldsymbol{z}(t)}{dt} = h(\boldsymbol{z}(t), t, \theta). \tag{5.3}$$

The neural ODE can be used to construct a continuous normalizing flow. The invertibility is naturally guaranteed by the theorem of the existence and uniqueness of the solution of the ODE. Furthermore, using the instantaneous change of variables formula, similar to Equation 5.2, the log-density can be evaluated by solving the following ODE:

$$\frac{\partial \log p(\boldsymbol{z}(t))}{\partial t} = -\text{tr}\left(\frac{\partial h}{\partial \boldsymbol{z}(t)}\right). \tag{5.4}$$

Grathwohl *et al.* [42] propose an improved version of neural ODE, named FFJORD, which has a lower computational cost by using an unbiased stochastic estimation of the trace of a matrix.

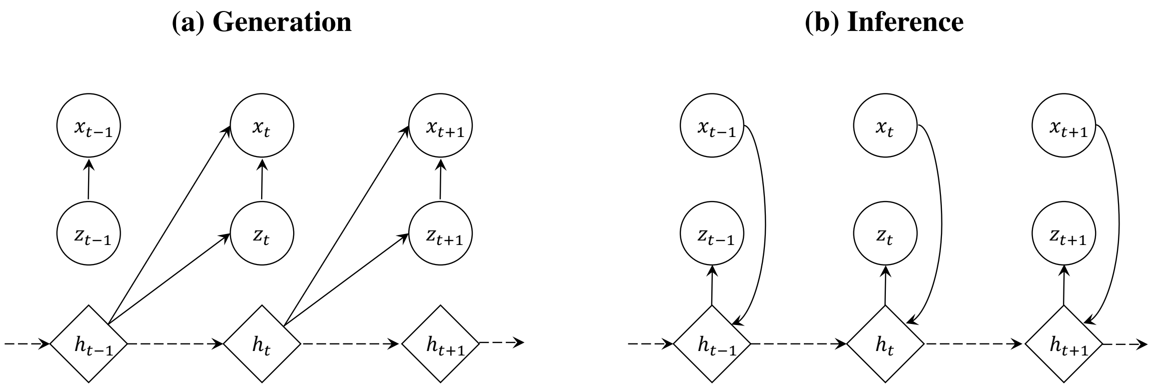


Figure 5.2: Graphical model illustration of our proposed framework during inference and generation.

5.4 Method

5.4.1 Problem definition

The input data is a sequence of actions $y_{1:n} = (y_1, y_2, .., y_n)$, where each action $y_i = (t_i, c_i)$ is a tuple of the time of occurrence $t_i \in \mathbb{R}_{\geq 0}$ and the action category $c_i \in \{1, 2, \ldots, K\}$. An alternative representation of the data is to use the inter-arrival time instead of the time of occurrence. That is, $x_{1:n} = (x_1, x_2, .., x_n)$ and $x_i = (\tau_i, c_i)$, where $\tau_i = t_i - t_{i-1}$ represents the inter-arrival time between actions x_{i-1} and x_i; we define $t_0 = 0$ for consistency. It is obvious that the two representations $x_{1:n}$ and $y_{1:n}$ are equivalent.

The task is to model the joint distribution of $x_{1:n}$ by maximizing the log-likelihood $\log p(x_{1:n})$. Furthermore, the model should also be able to perform action prediction. Specifically, it can predict $p(x_{n+1}|x_{1:n})$, the conditional distribution of the next action x_{n+1} given the history $x_{1:n}$.

5.4.2 Generative model

As a common approach to modeling time-series data, we use an RNN model, in particular an LSTM model, to encode the history. At any time step $1 \leq i \leq n$, the hidden state h_i is a compressed representation of the past actions $x_{1:i}$. The update step is defined as follows: $h_i = \text{LSTM}_\theta(x_i, h_{i-1})$. We also introduce a latent variable z_i at each step i. The prior distribution $p(z_i|x_{1:i-1})$ is a multivariate Gaussian distribution with a diagonal covariance matrix. It is modeled by an MLP with the hidden state h_{i-1} as the input. The generating distribution is $p(x_i|z_i, x_{1:i-1}) = p(\tau_i, c_i|z_i, x_{1:i-1})$. Part(a) of Figure 5.2 shows the graphical model representation during generation. We further assume that the latent variable z_i contains discriminative high-level information such that τ_i and c_i are conditionally independent given z_i and the history $x_{1:i-1}$, *i.e.*

$$p(x_i|z_i, x_{1:i-1}) = p(\tau_i|z_i, x_{1:i-1})\, p(c_i|z_i, x_{1:i-1}). \tag{5.5}$$

Therefore, the log-likelihood can be written as

$$\log p(x_{1:n}) = \sum_{i=1}^{n} \log p(x_i|x_{1:i-1}), \tag{5.6}$$

where

$$p(x_i|x_{1:i-1}) = \int p(x_i, z_i|x_{1:i-1})dz_i = \int p(x_i|z_i, x_{1:i-1})p(z_i|x_{1:i-1})dz_i. \tag{5.7}$$

Now we further elaborate on the two conditional distributions on the right-hand side of Equation 5.5. For the conditional distribution of the action category $p(c_i|z_i, x_{1:i-1})$, it is a discrete distribution and its support is $\{1, 2, \ldots, K\}$. It is modeled by an MLP followed by a softmax layer with z_i and h_{i-1} as input.

We use a normalizing flow $f_\theta : \mathbb{R}_{\geq 0} \to \mathbb{R}$ to model the conditional distribution of the inter-arrival time $p(\tau_i|z_i, x_{1:i-1})$. Specifically, we use a continuous normalizing flow model based on neural ODEs. The normalizing flow is able to model complex distributions. It allows exact computation of log-likelihood and supports efficient sampling from the distribution. The base distribution $p_{\text{base}}(\cdot|z_i, x_{1:i-1})$ is multivariate Gaussian with a diagonal covariance matrix. It is a network that takes the latent variable z_i and the hidden state h_{i-1} as input. Using the change of variables formula, the conditional distribution of τ_i can be written as:

$$p(\tau_i|z_i, x_{1:i-1}) = p_{\text{base}}(f_\theta(\tau_i)|z_i, x_{1:i-1}) \left|\det\left(\frac{\partial f_\theta}{\partial \tau_i}\right)\right|. \tag{5.8}$$

5.4.3 Inference with variational filtering

Directly optimizing the log-likelihood in Equation 5.6 is computationally intractable. Following the variational inference framework, we use a variational distribution $q_\phi(z_i|x_{1:i})$ to approximate the true posterior distribution $p(z_i|x_{1:i})$. Part (b) of Figure 5.2 shows the inference process of our model. Similar to the prior distribution, the variational distribution is a multivariate Gaussian distribution with a diagonal covariance matrix and is parameterized by an MLP with the hidden state h_i as input. A variational lower bound of the log-likelihood can be derived as follows:

$$\log p(x_{1:n}) \geq \sum_{i=1}^{n} \mathbb{E}_{q_\phi(z_i|x_{1:i})}\big[\log p(x_i|z_i, x_{1:i-1}) - \mathrm{KL}(q_\phi(z_i|x_{1:i}) \parallel p(z_i|x_{1:i-1}))\big].$$

The right-hand side is known as the evidence lower bound (ELBO).

5.4.4 Action prediction

The proposed model can easily perform action prediction. That is, given the observed action sequence $x_{1:n}$, we want to predict the distribution of the next action $x_{n+1} = (\tau_{n+1}, c_{n+1})$. It is simply a marginalization over the latent variable z_{n+1}:

$$\begin{aligned} p(x_{n+1}|x_{1:n}) &= \mathbb{E}_{p(z_{n+1}|x_{1:n})} p(x_{n+1}|z_{n+1}, x_{1:n}) \\ &= \mathbb{E}_{p(z_{n+1}|x_{1:n})} \left[p(\tau_{n+1}|z_{n+1}, x_{1:n}) p(c_{n+1}|z_{n+1}, x_{1:n}) \right]. \end{aligned} \tag{5.9}$$

It also provides a way to estimate the conditional distributions of τ_{n+1} and c_{n+1} via Monte Carlo sampling. We first draw N samples $\{\hat{z}_{n+1}^{(j)}\}_{j=1}^{N}$ from the prior distribution $p(z_{n+1}|x_{1:n})$. Then the distribution of action category c_{n+1} can be estimated by

$$\hat{p}(c_{n+1}|z_{n+1}, x_{1:n}) = \frac{1}{N} \sum_{j=1}^{N} p(c_{n+1}|\hat{z}_{n+1}^{(j)}, x_{1:n}). \tag{5.10}$$

For the inter-arrival time, we can further draw M samples $\{\hat{\tau}_{n+1}^{(j,k)}\}_{k=1}^{M}$ from $p(\tau_{n+1}|\hat{z}_{n+1}^{(j)}, x_{1:n})$ for each $j = 1, 2, \dots, N$. Those MN samples represent the conditional distribution of τ_{n+1}, and various statistics can be computed from them; for example, the point estimation is

$$\hat{\tau}_{n+1} = \frac{1}{MN} \sum_{j=1}^{N} \sum_{k=1}^{M} \hat{\tau}_{n+1}^{(j,k)}. \tag{5.11}$$

5.5 Experiments

To show the effectiveness of our approach, we evaluate the performance of our model on challenging tasks of density estimation, short-term action anticipation, and long-term action anticipation, *i.e.* predicting a sequence of future actions. We perform experiments on two challenging activity datasets: **(I) Breakfast** dataset [73] contains 1712 videos with 48 action classes related to breakfast preparation. **(II) Multithumos** dataset [139] contains 400 videos of 65 action classes. In this dataset, the average density of actions per label is 1.5, which makes it very challenging since multiple actions might happen at the same time.

5.5.1 Baselines

We compare our proposed approach with state-of-the-art activity prediction and temporal point process models:

(I) TD-LSTM is an LSTM that takes the history of past activities and predicts the next action inter-arrival time and category. We use the mean squared error (MSE) for regression loss on time and the cross-entropy for category prediction. This model is comparable with the RNN-based framework proposed in [1].

(III) APP-VAE [92] is a latent variable framework for activity prediction. Given a history of past actions, APP-VAE generates two distributions for the next action: one point process distribution over its timing (by predicting the conditional intensity and using it to define point process distributions), and one over its category by predicting the parameters of a categorical distribution over possible action labels. The action time distribution modeled by the APP-VAE model is always an exponential distribution.

(IV) IFL [114] is a recurrent model that takes the history of past actions and models next action timing distribution by density estimation using a mixture of Gaussians. An LSTM encodes history and predicts the parameters of the mixture distribution. It also predicts the parameters of a categorical distribution for the next action label.

(V) Majority Prediction is a simple baseline that predicts the most frequent action label at all time-steps regardless of history.

(VI) Average Time Prediction This baseline predicts the mean of inter-arrival time in the dataset at all time-steps.

We compute the most frequent action label and average inter-arrival time with respect to the training split.

5.5.2 Evaluation metrics

Log-likelihood. We report the negative log-likelihood (NLL) of conditional activity prediction for all models except for TD-LSTM. It measures the ability to model both the distribution of immediate next actions conditioned on history and entire sequences due to the way the joint probability of sequential data is factorized. The negative log-likelihood of an action sequence can be directly evaluated by IFL in closed form. For our proposed model and APP-VAE baseline, we report the importance weighted autoencoder (IWAE) bound [6], which is a lower bound of the true log-likelihood. To compute IWAE, at each time-step, we draw samples from the VAE's posterior distribution and follow the standard procedure for computing IWAE. The value is averaged by the total number of time steps across all the test sequences.

Error in time prediction. We also report the mean absolute error (MAE) between model's mean estimation of future event time and ground truth to evaluate the model's timing performance, *i.e.* $|\mathbb{E}_{\tau_{i+1}\sim p(\tau_{i+1}|x_{1:i})}(\tau_{i+1}) - \tau^*_{i+1}|$. The mean prediction is estimated by sampling from models for IFL, APP-VAE, and our proposed models. It is straightforward to take samples from IFL models. Due to the latent variable, sampling from APP-VAE and our model consists of two stages: (1) First, we draw samples from the prior distribution $z_{i+1} \sim p(z_{i+1}|x_{1:i})$. Then, we pass the samples of z_{i+1} to the decoder along with encoded history and (2) draw samples from each predicted distribution $\tau_{i+1} \sim p(\tau_{i+1}|z_{i+1}, x_{1:i})$. As TD-LSTM models are optimized using MSE loss, we directly take the prediction of the TD-LSTM model as the mean of a predicted Gaussian distribution.

Accuracy in category prediction. We also report the accuracy of the model's prediction of future event labels. In one-step-ahead prediction, we directly take the most probable class of the categorical distribution output by IFL and TD-LSTM as the predicted class of the future step. For APP-VAE and our proposed model, we predict the label of the next action through a similar two-stage sampling. In the first stage, we sample a latent code z from the prior distribution. Then, we take the most probable category predicted by our model conditioned on the latent code as the prediction associated with that sample of z. We take the mode of predictions over all the samples of z as the prediction of our model for the next step.

5.5.3 Experimental results

Table 5.1: **Short-term Action Anticipation Results**

Model	**Breakfast**			**Multithumos**		
	Acc.↑	MAE↓	NLL↓	Acc.↑	MAE↓	NLL↓
TD-LSTM [1]	56.79	172.09	–	36.19	1.97	–
APP-VAE [92]	57.44	159.17	8.82	39.82	1.88	3.57
IFL [114]	**57.96**	284.43	**7.64**	39.88	1.96	3.41
Proposed	55.61	**154.14**	7.89	**40.29**	**1.78**	**3.37**

Short-term action anticipation. Correctly predicting one-step-ahead is the essential building block of generating long-term sequences for sequential models. We report the performance of our model in anticipating the next action label and time with the accuracy and MAE metrics. We also report the negative log-likelihood (NLL) in estimating the density of action sequences. The negative log-likelihood is estimated by the IWAE bound with 1000 samples of the latent variable. The sample size for predicting the next action's time and category is also 1000. Table 5.1 shows the experimental results. On the Multithumos dataset, our proposed model outperforms the baselines on all short-term anticipation metrics. On the Breakfast dataset, it also shows superior performance on sequence density estimation and time prediction with action category prediction performance close to the best-performing model. It is worth noting that despite IFL showing better performance on density estimation, the NLL of our model is estimated by an upper bound of the true NLL while IFL can evaluate the exact NLL values. However, the performance of IFL on the point estimate of next action's time is inferior to all the other models We argue that our model strikes a good balance between modeling the distribution of action sequences and accurately predicting near-future events. This good balance combined with the expressive power of latent variables and flexibility of time distribution contributes to superior performance on long-term prediction tasks.

Long-term action anticipation. We evaluate the performance of our proposed framework in predicting multiple-steps into the future. We show that our model is better at predicting multiple steps ahead into the future than baseline models given contextual history information. For the Breakfast dataset, we report anticipation results within 5 steps in the future given the observation of first 3 actions in a sequence. In the Multithumos dataset, since the sequences are longer with a median length of 49.5 compared to breakfast dataset, we expand the observation window to 50 actions and, prediction window to 6 steps ahead. The method of estimating mean and mode of predicted actions is extended to the multiple-step scenario on a sampling basis by feeding samples from previous steps back to the model to generate samples for future steps. We present more details of the sampling method in subsubsection 5.5.9.3. Except for TD-LSTM, we sample 1000 samples, feed them back to the model and obtain 1 new sample in the following time step for each sample of the previous step. In this method, we can sample 100 sequences from the model. In APP-VAE and our proposed model, we sample 10 latent codes z and 100 category and action pair conditioned on each z at the first step of prediction. We use the sampled sequences to estimate the mean of action time at each step and take the mode of all the sample labels at each step as the prediction. Table 5.5.3 provides the result for Breakfast and Multithumos datasets. The results indicate the better capability of our model in long-term action anticipation. We can see that our model consistently outperforms baseline models across different prediction windows. Compared with latent variable models, including APP-VAE and our proposed model, the IFL model which predicts time and action category independently deteriorates much more quickly. We hypothesize that this independence assumption could make the model generate samples with action time and category inconsistent with each other and this inconsistency could accumulate across time steps causing the model's performance to deteriorate quickly.

5.5.4 Study on structure of action time and label prediction

In this experiment, we empirically show that the latent variable model combined with history permits us to model the distribution of the action category and time independently even though without the abstract latent variable, knowing one of the action categories and time would be informative about the other. In this study, we try two variants of our proposed model: a) we give the action category as the additional input to the time decoder that proposed the base distribution of the normalizing

Metric	Model	**Breakfast** (3 Actions)				**Multithumos** (50 Actions)				
		2	3	4	5	2	3	4	5	6
Acc.↑	MAJE.	17.73	18.07	19.29	19.71	11.51	10.48	10.00	9.73	9.71
	TD-LSTM [1]	**55.25**	51.03	47.45	45.16	31.86	28.45	26.82	25.33	24.35
	APP-VAE [92]	51.94	48.58	47.45	46.97	39.83	33.24	30.46	27.34	25.74
	IFL [114]	41.99	37.55	33.11	32.73	27.43	23.35	19.77	18.53	17.57
	Proposed	54.15	**51.03**	**48.81**	**47.28**	**41.16**	**35.33**	**32.50**	**30.46**	**28.20**
MAE↓	AVG.	260.81	274.61	278.81	285.77	1.97	2.08	2.17	2.14	2.17
	TD-LSTM [1]	248.52	255.9	285.64	317.84	1.84	1.95	**2.06**	**2.04**	**2.08**
	APP-VAE [92]	243.29	258.24	273.93	304.22	1.88	2.01	2.14	2.11	2.14
	IFL [114]	260.95	292.17	302.98	326.22	2.03	2.10	2.43	2.67	6.30
	Proposed	**229.33**	**242.54**	**257.22**	**287.22**	**1.80**	**1.95**	2.11	2.07	2.11

Table 5.2: **Long-term Action Prediction Results Conditioned on History**. The number in the parenthesis in the first row shows the number of actions in the observation window. The number in the second row shows the number of actions in the observation window.

flow; b) we concatenate the time of the next action with the hidden state and latent code as input to the action category decoder. All the other components of the model are exactly the same. Teacher forcing uses teacher forcing during training and testing, which means the ground-truth value of the additional input was fed to the model. The experiment results are presented in Table 5.3 As we can see, the model's performance on density estimation is not significantly impacted by the additional inputs to the time action category decoder. However, both variants' performance on next action category prediction is significantly impacted. The model variant that conditions the action decoder on time also shows substantial performance decay on estimating the next action's time as measured by MAE. The experiment results corroborate our hypothesis that given a latent variable, we can model the distribution of action time and category independently. In contrast, on the Multithumos dataset, we see that conditioning the action category prediction on time in IFL models significantly improves its action category prediction accuracy and conditioning action time prediction on the category also improves its time prediction performance. These results further support our arguments that the independence assumption between time and action label given only history is oversimplified but we can model the distributions of time and action label independently given a latent space.

5.5.5 Ablation study on complexity of time decoder

VAE models have shown the power to model complex and multi-modal distributions, e.g. MNIST, CelebA due to the expressive latent space. In this study we empirically show that even with an expressive latent space, applying a proper normalizing flow model to the distribution proposed by the time decoder could still improve the model's performance, especially the estimation of next action's time. As a baseline, we evaluate the performance of a model without normalizing flow blocks by directly predicting a log-normal distribution. We also evaluated the performance of models with normalizing flows of different complexity by varying the number of neural ODE blocks in the normalizing flow. The results are presented in Table 5.4. With a simple normalizing flow model containing 1 block, we see significant performance gain on short-term estimation of next action's

Dataset	Conditioning Structure			**IFL** [114]			**Proposed**		
	History	Action	Time	Acc.↑	MAE ↓	NLL↓	Acc.↑	MAE ↓	NLL↓
Breakfast	✓			57.96	284.43	7.64	55.61	154.14	7.89
	✓	✓		57.96	174.35	7.76	42.86	157.62	7.96
	✓		✓	56.01	164.51	7.91	42.86	174.54	8.02
Multithumos	✓			39.88	1.99	3.41	40.29	1.78	3.37
	✓	✓		34.45	1.93	3.45	37.72	2.00	3.22
	✓		✓	41.44	2.70	3.33	37.73	1.84	3.28

Table 5.3: **Study on Predefined Structured Prediction.** Experimental results on Breakfast dataset for assuming a pre-defined structured framework for IFL and Proposed Model. We pass the Ground-truth as the condition for time/action to the model during test time. Models with *action* box checked by a ✓means the time decoder is conditioned on action. Models with *time* box checked by ✓means the action decoder is conditioned on time. Models with only *history* box checked by ✓are the original/proposed model.

time, as well as improvements on density estimation and action prediction. The performance on short-term time estimation continues to improve as we increase the complexity of the normalizing flow model. However when the number of blocks in the normalizing flow reaches 10, the density estimation and category prediction's performance decays on the breakfast dataset, showing signs of overfitting while the performance improvements continues on the multithumos dataset. We report the results using a model with 5 blocks in the normalizing flow in previous experiments on both datasets for consistency.

Flow Blocks	**Breakfast**			**Multithumos**		
	Acc.↑	MAE↓	NLL↓	Acc.↑	MAE↓	NLL↓
0	54.31	176.25	8.07	40.64	1.95	3.68
1	**55.78**	157.78	**7.87**	39.24	1.89	3.27
5	55.61	154.14	7.89	**40.29**	1.78	3.37
10	42.96	**152.67**	8.00	40.64	**1.72**	**3.17**

Table 5.4: **Normalizing Flow Complexity Study Results**. The model output parameters of a log-normal distribution for zero blocks in the normalizing flow module.

5.5.6 Qualitative results of future actions anticipation

Figure 5.3 shows action sequences generated by our model on Breakfast dataset. For each example, we illustrate the observation provided for the model (action history), two generated sequences by our model, and the ground-truth of future actions. We can see that our model is able to generate action sequences comparable to ground-truth sequences as well as other plausible action sequences. Results indicate the ability of our model in capturing the multi-modal underlying distribution of future actions.

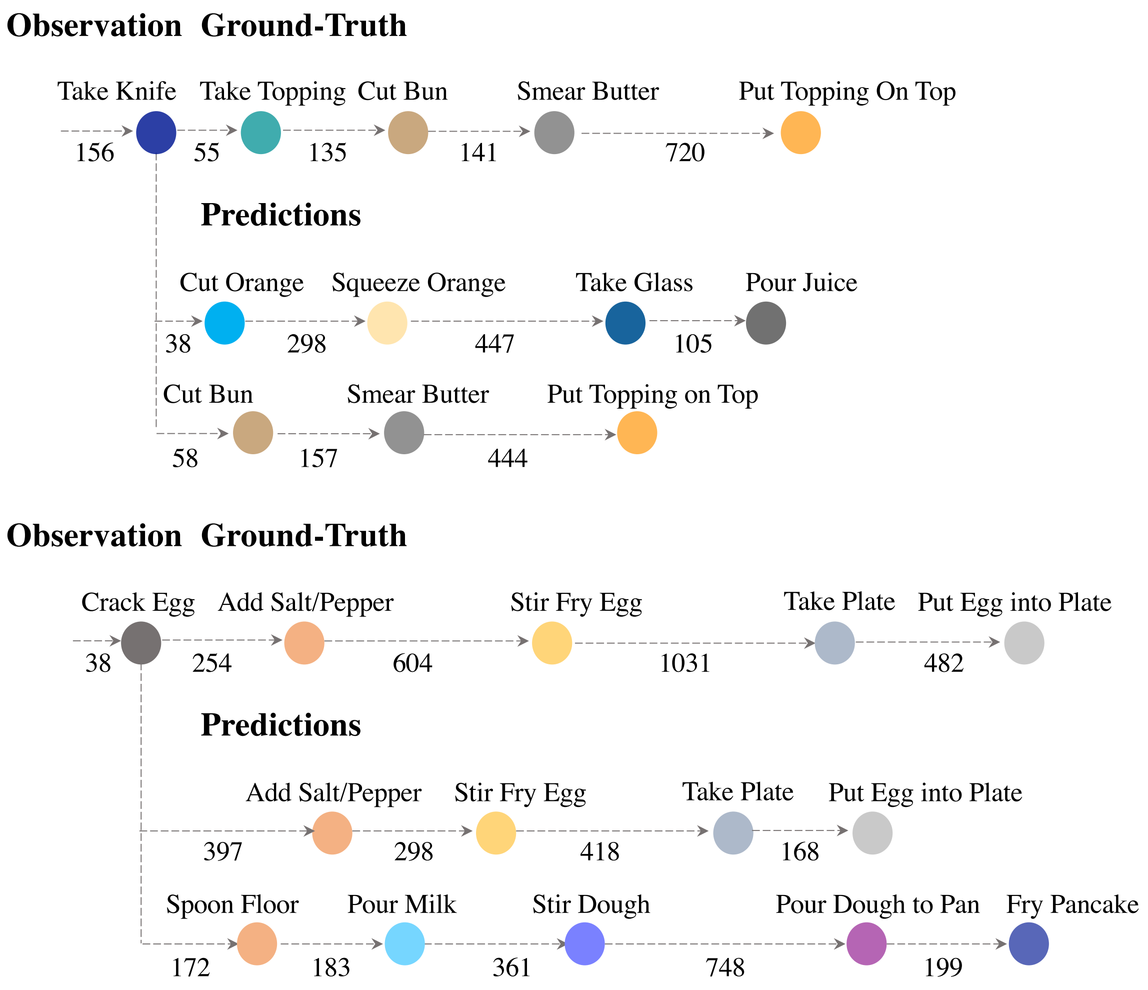


Figure 5.3: Qualitative results of sequence generation on Breakfast dataset.

5.5.7 Multiple Run Results of Long-term Action Anticipation

Due to the relatively large uncertainty in long-term predictions, we further provide the mean and standard deviation of long-term action prediction results on 4 runs with different random seeds. The experiment results are presented in Table 5.5 and Table 5.6 for Breakfast [73] and Multithumos [139] datasets respectively. The two naive baselines, majority prediction and average time prediction, are not affected by random seeds and therefore the standard deviations for them are zero.

Metric	Model	**Breakfast** (3 Actions)			
		2	3	4	5
Acc.↑	MAJE.	17.73±0.00	18.07±0.00	19.29±0.00	19.71±0.00
	TD-LSTM [1]	**54.56±0.53**	50.72±0.40	47.02±0.33	44.78±0.30
	APP-VAE	52.68±1.16	49.26±1.03	47.90±1.05	47.28±0.53
	IFL [114]	41.58±0.28	37.25±0.21	33.62±0.66	32.65±0.52
	Proposed	53.87±0.32	**51.22±0.24**	**48.72±0.17**	**47.20±0.15**
MAE↓	AVG.	260.81±0.00	274.61±0.00	278.81±0.00	285.77±0.00
	TD-LSTM [1]	248.07±0.41	255.75±0.29	274.80±7.24	317.93±0.24
	APP-VAE [92]	243.51±1.37	258.09±0.91	274.00±1.07	303.49±0.96
	IFL [114]	261.88±1.29	292.01±0.74	303.38±0.80	326.43±0.23
	Proposed	**230.29±1.21**	**243.79±1.48**	**258.59±0.99**	**288.89±1.17**

Table 5.5: **Long-term Action Prediction Multiple Run Results Conditioned on History for Breakfast Dataset**. The number in the parenthesis in the first row shows the number of actions in the observation window. The number in the second row shows the number of actions in the observation window.

Metric	Model	**Multithumos** (50 Actions)				
		2	3	4	5	6
Acc.↑	MAJE.	11.51±0.00	10.48±0.00	10.00±0.00	9.73±0.00	9.71±0.00
	TD-LSTM [1]	31.09±0.76	27.85±0.43	26.43±0.39	25.42±0.19	24.47±0.24
	APP-VAE [92]	38.39±1.75	32.49±1.14	29.83±1.11	27.12±0.87	25.62±0.96
	IFL [114]	28.43±0.92	22.38±0.90	20.12±0.40	18.21±0.28	17.38±0.30
	Proposed	**40.49±0.77**	**34.73±0.88**	**32.05±0.81**	**30.41±0.46**	**28.35±0.28**
MAE↓	AVG.	1.97±0.00	2.08±0.00	2.17±0.00	2.14±0.00	2.17±0.00
	TD-LSTM [1]	1.84±0.01	1.96±0.01	**2.06±0.01**	**2.04±0.01**	**2.08±0.01**
	APP-VAE [92]	1.90±0.03	2.02±0.02	2.15±0.02	2.12±0.01	2.14±0.01
	IFL [92]	2.03±0.02	2.10±0.01	3.13±0.69	3.41±0.96	4.29±1.49
	Proposed	**1.82 ±0.02**	**1.96±0.01**	2.08±0.02	2.05±0.02	**2.08±0.01**

Table 5.6: **Long-term Action Prediction Multiple Run Results Conditioned on History for Multithumos Dataset**. The format of this table is similar to Table 5.5.

5.5.8 Unconditional Generation Qualitative Results

In our model, we assume that the prior distribution over the latent code in the first step of the sequence follows a Normal Distribution $z_1 \sim \mathcal{N}(0, 1)$. Figure 5.4 shows examples of action sequences generated by sampling the first action from this prior. In order to do so, we have two stages

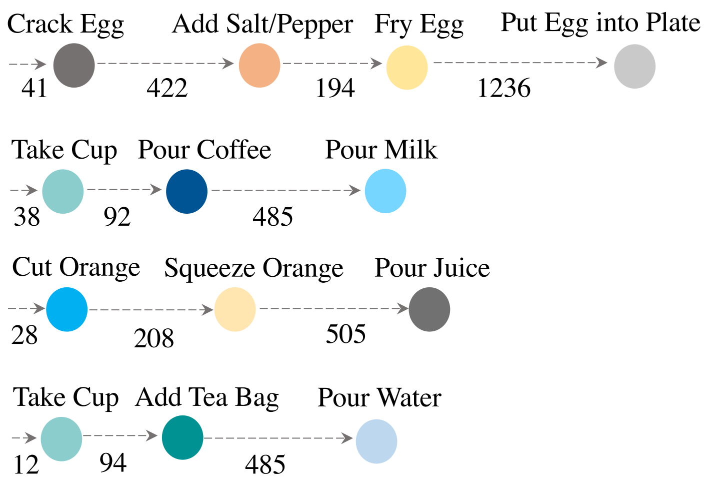


Figure 5.4: Qualitative results of sequence generation on Breakfast dataset. In this setting, no action is observed and sequences are generated by sampling from the prior distribution of the first step in the sequence.

of sampling: (1) First, samples of the prior distribution $z_1 \sim \mathcal{N}(0,1)$ are passed to the decoders which produce inter-arrival time $p(\tau_1|z_1)$ and category $p(a_1|z_1)$ distributions for the first action; (2) Then, samples are taken from each time $\tau_1 \sim p(\tau_1|z_1)$ and category $a_1 \sim p(a_1|z_1)$ distributions and construct the time and category of first action in the sequence. To generate action sequences of multiple steps, the sampled action label and times are fed back to the model as observations to get a sample of the next step. Sequences in Figure 5.4 are obtained from the model trained on the Breakfast dataset. We can see that the generated categories and temporal occurrence are diverse and reasonable.

5.5.9 Experimental Details

5.5.9.1 Model Architecture

Proposed model. We use an LSTM of hidden state size 128 for the prior and posterior networks. The dimension of the latent space is set to 256. Before passing the action label and time to the model, we pass them through two embedding networks at each time step. We pass the 1-hot encoding of the action label to an MLP with two layers of size 64 and 128, respectively. A separate MLP with a similar architecture is used for encoding action inter-arrival time. The final representation for the action is obtained by passing the concatenation of action time and label embeddings through a linear transformation. The dimension of the final representation is 256. To get the parameters of the prior and posterior distributions in the latent space, we pass the hidden states of prior and posterior LSTM cells at the corresponding time steps to two separate MLPs, respectively. Both MLPs have two hidden layers of size 256. We use an MLP containing two hidden layers of size 256 as the action decoder; it receives the concatenation of the latent variable and the hidden state of the prior LSTM, and decodes it into a categorical distribution of action labels. The time decoder is an MLP with a similar architecture. It proposes a base distribution that will be further processed by the normalizing flow module to produce the final distribution of action time.

Continuous normalizing flow. We adopt the continuous normalizing flow based on neural ODE as the normalizing flow model to produce the distribution of action time distribution. The neural ODE implementations provided by Grathwohl *et al.* [42][1] and Chen *et al.* [13][2] are used. We compose a stack of 5 MLPs, each with 3 hidden layers of size 64 for the neural ODE model f_θ^{-1} in Equation 8. To make sure that our model generates positive inter-arrival times $\tau_i \in \mathcal{R}^+$, we apply an additional exponential transformation and accordingly, the Jacobian term is added to training objective and evaluation criterion. An affine transformation is applied right before the exponential transformation to mitigate numerical instability.

Baseline models. To ensure a fair comparison, we share the same architecture for the embedding, LSTM cells, and action and time decoder networks across all the models. We also share a similar pipeline of applying an affine and exponential transformation in the normalizing flow of the IFL model.

5.5.9.2 Training Details

The training set of each dataset is split into subsets for training and validation by the ratio of 8:2. We train all the models for 1500 epochs and use early stopping based on loss on the validation set. We select the model with the minimum validation loss (ELBO) for evaluation. Adam optimizer [65] with a learning rate of 0.001 is used for all the models.

5.5.9.3 Sampling Details in Evaluation

Short-term action anticipation. For all the models except TD-LSTM, we use 1000 samples to evaluate the models for the short-term action anticipation task. For the IFL model, we use 1000 samples from the output distribution to estimate the mean of the time of the immediate next action. We directly take the category with the largest probability as the predicted class of the next action. For latent variable models, including APP-VAE and our proposed model, 1000 samples from the latent space are used to evaluate the log-likelihood and short-term action label prediction. To generate 1000 samples to estimate the mean of the next-step action's time, we sample 10 zs from the latent space and sample 100 samples of τ for each latent code at each time step.

Long-term action anticipation. We use 1000 sampled sequences in long-term prediction evaluation. For IFL models, we sample 1000 pairs of action time and label at the first future step to predict and obtained 1000 sequences of actions that differ by the action of the last step only. For each sequence, we feed it back into the model and sample 1 pair of label/action time at every following step to obtain 1000 action time and label sequences of the desired length. A similar approach is followed

[1]`https://github.com/rtqichen/torchdiffeq`

[2]`https://github.com/rtqichen/ffjord`

for TD-LSTM. The only difference is at the first time-step of prediction. Since TD-LSTM predicts the time in a deterministic way, in the first step of prediction, 1000 samples of action category distribution are taken and each is paired with the predicted time.

For APP-VAE and our proposed model, we sample 10 z's from the latent distribution and 100 pairs of action time and label for each latent code to obtain 1000 samples at the first future step to predict. The 1000 action sequences are fed back to the model as inputs and we sample one latent code and one pair of action time and label for each action sequence at each following step. In this way, we can sample 1000 sequences from the latent variable models.

5.5.9.4 Data Pre-Processing Details

For both Breakfast and Multithumos datasets, we use the annotation of videos in the dataset to create sequences of categories and timing of actions. In the breakfast dataset, each action is annotated by its action category and a frame number indicating when the action starts. We use the frame numbers as our time scale and compute the inter-arrival times based on the difference between the starting frames of actions. In all the videos of this dataset, the first action is 'SIL' and starts at the first frame in the video. In our data preparation, we drop this action from each sequence. APP-VAE [92] follows a different data processing: it works with the original action sequences *i.e.* all sequences start with the action 'SIL' happening at the first frame.

Similarly, in the Multithumos dataset, we use the annotation of videos to create action sequences. Each action is annotated by its category and the time it starts. The scale of action timing is in seconds with 1 decimal point precision. In Multithumos, actions might happen at the same time. In our data pre-processing, for concurrent actions, we use a fixed alphabetical ordering based on action categories and shift the starting time of each concurrent action by an offset $= 0.1$. We find using a too small offset for concurrent actions could cause numerical instability for the ODE solver.

We believe that the difference between the reported results of Mehrasa *et al.* [92] and the results reported in this chapter arises from following a different data pre-processing approach.

5.6 Summary

We propose a recurrent latent variable model for asynchronous action sequence modeling. The model utilizes an expressive latent space to capture complex dependencies between action time and label across steps and a normalizing flow model to generate a flexible distribution of inter-arrival time between actions. We use a standard variational lower bound to optimize the model with a filtering inference network. The model can generate high-quality samples of human action sequences and also shows superior performance on sequence density estimation, short-term prediction and especially long-term action sequence conditional generation.

Chapter 6

Conclusion

In this dissertation, we contribute in two main directions: modeling events in asynchronous time-series data and learning from partial labels:

- **Event Analysis in Asynchronous Time-series Data.** We propose novel probabilistic models based on framework of point processes. Temporal point process provides us a rich framework describing the generative process of event sequence data, which enables various down-stream tasks such as future prediction, sequence generation, density estimation, anomaly detection, and etc. We study point processes under the prospective of deep generative models aiming to improve flexibility and expressiveness of current approaches in modeling complex real-world event sequences. First, we formulate our model with variational auto encoder (VAE) paradigm, a powerful class of probabilistic models, and present a novel form of VAE modeling the distribution of timing and categories of event sequences. Second, we connect the fields of point processes and neural density estimation and propose a recurrent latent variable framework that directly models point processes distribution by utilizing normalizing flows. This approach is capable of capturing highly complex temporal distribution and does not rely on any restrictive parametric forms. We evaluate our proposed frameworks in challenging tasks of future event prediction, sequences density estimation, conditional generation, and anomaly detection. Comparison with the state-of-the-art baseline models on challenging real-life datasets show that our proposed frameworks are effective at modeling discrete event sequences.

- **Learning from Partially Labeled Data.** In this direction, we introduce an end-to-end learning scheme from partially labeled image data with a multi-label classifier. Our first contribution is to empirically compare several labeling strategies to highlight the potential for learning with partial labels. Second, we contribute a new loss function that enables end-to-end learning of a classifier from partially labeled data. Thirds, we develop a method that uses graph neural networks to capture correlation between different categories to improve label prediction, and we use our model to predict missing labels.

Despite the promising progress and success in modelling time-series and learning from partially labeled data, the assumptions of which these problems are defined and formulated on restricts the generality of these methods and limits their performance in real-world scenarios. Removing these constraints and tackling more general problem setups still remain open and is an active research area. For example, in our work, on learning partial-labels, we focused on multi-labels classification task. However, dealing with partially annotated data is a common problem in many different domains. An interesting future direction could be to investigate learning from partial-labels in other settings such as time-series and event sequences. In time-series, annotation cost is much more heavier compared to annotation of image datasets due to the sequential nature, and having a framework defined for it would have many applications. On modelling human actions sequences and event prediction, one future direction in visual recognition application could be using visual data as mark in the framework of mark temporal point process. Visual data carry rich contextual information about ongoing actions and would be beneficial in building more powerful prediction frameworks. Exploring more powerful and expressive generative models in modelling events sequences could be another research direction, for example one limitation of our VAE-based frameworks is that we are not able to perform exact likelihood inference, to address this issue, exploring end-to-end flow-based generative model could be an interesting future direction .

Bibliography

[1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When Will You Do What? - Anticipating Temporal Occurrences of Activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[2] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic Variational Video Prediction. In *International Conference on Learning Representations (ICLR)*, 2018.

[3] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. 1999.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

[5] S. S. Bucak, R. Jin, and A. K. Jain. Multi-label learning with incomplete class assignments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[6] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[7] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Classify, predict, detect, anticipate and synthesize: Hierarchical recurrent latent variable models for human activity modeling. *arXiv preprint arXiv:1809.08875*, 2018.

[8] Ricardo S. Cabral, Fernando Torre, Joao P. Costeira, and Alexandre Bernardino. Matrix Completion for Multi-label Image Classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[9] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Conference on Artificial Intelligence (AAAI)*, 2010.

[10] Micael Carvalho, Remi Cadene, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 35–44, New York, NY, USA, 2018. ACM.

[11] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[12] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. 2010.

[13] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[14] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[15] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[16] Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[17] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[18] Hong-Min Chu, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Deep Generative Models for Weakly-Supervised Multi-Label Classification. In *European Conference on Computer Vision (ECCV)*, 2018.

[19] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: A Real-world Web Image Database from National University of Singapore. In *ACM International Conference on Image and Video Retrieval (CIVR)*, 2009.

[20] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[21] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from Partial Labels. *Journal of Machine Learning Research (JMLR)*, 2011.

[22] Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[23] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[25] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S. Bernstein, Alex Berg, and Li Fei-Fei. Scalable Multi-label Annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014.

[26] Emily Denton and Rob Fergus. Stochastic Video Generation with a Learned Prior. In *International Conference on Machine Learning (ICML)*, 2018.

[27] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[28] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.

[29] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

[30] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 647–657, 2019.

[31] Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. WILDCAT: Weakly Supervised Learning of Deep ConvNets for Image Classification, Pointwise Localization and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[32] Thibaut Durand, Nicolas Thome, and Matthieu Cord. WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] Thibaut Durand, Nicolas Thome, and Matthieu Cord. Exploiting Negative Evidence for Deep Latent Structured Models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[34] Tal El-Hay, Nir Friedman, Daphne Koller, and Raz Kupferman. Continuous time markov networks. *arXiv preprint arXiv:1206.6838*, 2012.

[35] Yariv Ephraim and Neri Merhav. Hidden markov processes. *IEEE Transactions on information theory*, 48(6):1518–1569, 2002.

[36] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 2015.

[37] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Predicting the future: A jointly learnt model for action anticipation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[38] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: reinforced encoder-decoder networks for action anticipation. *CoRR*, abs/1707.04818, 2017.

[39] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. Deep Convolutional Ranking for Multilabel Image Annotation. In *International Conference on Learning Representations (ICLR)*, 2014.

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[41] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2005.

[42] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

[43] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[44] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R. Scott, and Dinglong Huang. CurriculumNet: Weakly Supervised Learning from Large-Scale Web Images. In *European Conference on Computer Vision (ECCV)*, 2018.

[45] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[46] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971.

[47] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–467, 2018.

[48] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] M. Hoai and F. De la Torre. Max-margin early event detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[51] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.

[52] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596, 2017.

[53] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015.

[54] Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and their Applications*, 1979.

[55] Jacques Janssen and Nikolaos Limnios. *Semi-Markov models and applications*. Springer Science & Business Media, 2013.

[56] Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. 2007.

[57] Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems 32*, pages 9843–9854. Curran Associates, Inc., 2019.

[58] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-Paced Curriculum Learning. In *Conference on Artificial Intelligence (AAAI)*, 2015.

[59] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *International Conference on Machine Learning (ICML)*, 2018.

[60] How Jing and Alexander J Smola. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 515–524. ACM, 2017.

[61] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision (ECCV)*, 2016.

[62] Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. Multilabel classification using bayesian compressed sensing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[63] Qiuhong Ke, Mario Fritz, and Bernt Schiele. Time-conditioned action anticipation in one shot. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[64] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[65] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[66] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[67] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[68] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[69] J.F.C. Kingman. *Poisson Processes*. Oxford Studies in Probability. Clarendon Press, 1992.

[70] Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

[71] Yu Kong, Zhiqiang Tao, and Yun Fu. Deep sequential context networks for action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1481, 2017.

[72] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do Better ImageNet Models Transfer Better? 2018.

[73] Hilde Kuehne, Ali Arslan, and Thomas Serre. The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[74] M. P. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[75] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. 2018.

[76] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision (ECCV)*, 2014.

[77] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision (ECCV)*, 2014.

[78] L. J. Li, G. Wang, and Li Fei-Fei. OPTIMOL: automatic Online Picture collecTion via Incremental MOdel Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[79] Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning temporal point processes via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[80] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, Jesse Berent, Abhinav Gupta, Rahul Sukthankar, and Luc Van Gool. WebVision Challenge: Visual Learning and Understanding With Web Data. In *arXiv 1705.05640*, 2017.

[81] Yuncheng Li, Yale Song, and Jiebo Luo. Improving Pairwise Ranking for Multi-label Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[82] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. 2014.

[83] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive Neural Architecture Search. In *European Conference on Computer Vision (ECCV)*, 2018.

[84] Jun Liu, Amir Shahroudy, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ssnet: Scale selection network for online 3d action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8349–8358, 2018.

[85] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3282–3291, 2018.

[86] Helmut Lütkepohl. Comparison of criteria for estimating the order of a vector autoregressive process. *Journal of time series analysis*, 6(1):35–52, 1985.

[87] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *cvpr*, 2016.

[88] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In *European Conference on Computer Vision (ECCV)*, 2018.

[89] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5784–5793, Oct 2017.

[90] Tahmida Mahmud, Mahmudul Hasan, and Amit K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[91] Nazanin Mehrasa, Ruizhi Deng, Mohamed Osama Ahmed, Bo Chang, Jiawei He, Thibaut Durand, Marcus Brubaker, and Greg Mori. Point process flows. *Learning with Temporal Point Processes, NeurIPS Workshop)*, 2019.

[92] Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[93] Hongyuan Mei and Jason Eisner. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[94] Minmin Chen and Alice Zheng and Kilian Weinberger. Fast image tagging. In *International Conference on Machine Learning (ICML)*, 2013.

[95] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-Ending Learning. In *Conference on Artificial Intelligence (AAAI)*, 2015.

[96] Takahiro Omi, naonori ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems 32*, pages 2120–2129. Curran Associates, Inc., 2019.

[97] Takahiro Omi, naonori ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems*, pages 2122–2132, 2019.

[98] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[99] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Is Object Localization for Free? - Weakly-Supervised Learning With Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[100] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[101] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[102] Viorica Pătrăucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *International Conference on Learning Representations (ICLR) Workshop*, 2016.

[103] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Faster Discovery of Neural Architectures by Searching for Paths in a Large Model. In *International Conference on Learning Representations (ICLR)*, 2018.

[104] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3D Graph Neural Networks for RGBD Semantic Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[105] Jakob Gulddahl Rasmussen. Temporal point processes: the conditional intensity function. *Lecture Notes, Jan*, 2011.

[106] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[107] C. J. Van Rijsbergen. *Information Retrieval*. 1979.

[108] Yulia Rubanova, Tian Qi Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems 32*, pages 5321–5331. Curran Associates, Inc., 2019.

[109] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[110] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging LSTMs to Anticipate Actions Very Early. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[111] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging lstms to anticipate actions very early. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[112] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009.

[113] Fadime Sener and Angela Yao. Zero-shot anticipation for instructional activities. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[114] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2020.

[115] Yuge Shi, Basura Fernando, and Richard Hartley. Action anticipation with rbf kernelized feature mapping rnn. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[116] Pierre Stock and Moustapha Cisse. ConvNets and ImageNet Beyond Accuracy: Understanding Mistakes and Uncovering Biases. In *European Conference on Computer Vision (ECCV)*, 2018.

[117] Chen Sun, Manohar Paluri, Ronan Collobert, Ram Nevatia, and Lubomir Bourdev. ProNet: Learning to Propose Object-Specific Boxes for Cascaded Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[118] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[119] Yu-Yin Sun, Yin Zhang, and Zhi-Hua Zhou. Multi-label Learning with Weak Label. In *Conference on Artificial Intelligence (AAAI)*, 2010.

[120] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Conference on Artificial Intelligence (AAAI)*, 2017.

[121] Lei Tang, Suju Rajan, and Vijay K. Narayanan. Large scale multi-label classification via metalabeler. In *WWW*, 2009.

[122] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 2007.

[123] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[124] Deepak Vasisht, Andreas Damianou, Manik Varma, and Ashish Kapoor. Active Learning for Sparse Bayesian Multilabel Classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[125] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to Generate Long-term Future via Hierarchical Prediction. In *International Conference on Machine Learning (ICML)*, 2017.

[126] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–106, 2016.

[127] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[128] Qifan Wang, Bin Shen, Shumiao Wang, Liang Li, and Luo Si. Binary Codes Embedding for Fast Image Tagging with Incomplete Labels. In *European Conference on Computer Vision (ECCV)*, 2014.

[129] Xionghui Wang, Jian-Fang Hu, Jian-Huang Lai, Jianguo Zhang, and Wei-Shi Zheng. Progressive teacher-student learning for early action prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[130] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[131] Baoyuan Wu, Siwei Lyu, and Bernard Ghanem. ML-MG: Multi-Label Learning With Missing Labels Using a Mixed Graph. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[132] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[133] Shuai Xiao, Hongteng Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. Learning conditional generative models for temporal point processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[134] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M Chu. Modeling the intensity function of point process via recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[135] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[136] Miao Xu, Rong Jin, and Zhi-Hua Zhou. Speedup Matrix Completion with Side Information: Application to Multi-Label Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[137] Hao Yang, Joey Tianyi Zhou, and Jianfei Cai. Improving Multi-label Learning with Missing Labels by Structured Semantic Correlations. In *European Conference on Computer Vision (ECCV)*, 2016.

[138] Yiming Yang. An evaluation of statistical approaches to text categorization. 1999.

[139] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos. *International Journal of Computer Vision (IJCV)*, 2017.

[140] Li Yingzhen and Stephan Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, 2018.

[141] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. Large-scale Multi-label Learning with Missing Labels. In *International Conference on Machine Learning (ICML)*, 2014.

[142] Runsheng Yu, Zhenyu Shi, and Laiyun Qing. Unsupervised learning aids prediction: Using future representation learning variantial autoencoder for human action prediction. *CoRR*, abs/1711.09265, 2017.

[143] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[144] Y. Zhong, B. Xu, G.-T. Zhou, L. Bornn, and G. Mori. Time Perception Machine: Temporal Point Processes for the When, Where and What of Activity Prediction. In *arXiv 1808.04063*, 2018.

[145] Yatao Zhong, Bicheng Xu, Guang-Tong Zhou, Luke Bornn, and Greg Mori. Time perception machine: Temporal point processes for the when, where and what of activity prediction. *arXiv preprint arXiv:1808.04063*, 2018.

[146] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[147] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[148] Walter Zucchini, Iain L MacDonald, and Roland Langrock. *Hidden Markov models for time series: an introduction using R*. CRC press, 2017.