UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CLASSIFICATION OF ASTRONOMICAL OBJECTS

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

NICOLÁS JAVIER ASTORGA ROCHA

PROFESORES GUÍA:
NANCY HITSCHFELD KAHLER
PABLO ESTÉVEZ VALENCIA

PROFESOR CO-GUÍA:
PABLO HUIJSE HEISE

MIEMBROS DE LA COMISIÓN:
FELIPE BRAVO-MARQUEZ
GUILLERMO CABRERA-VIVES
FRANCISCO FÖRSTER BURÓN

SANTIAGO DE CHILE
2023

CLASIFICACIÓN DE OBJETOS ASTRONÓMICOS

RESUMEN

La clasificación de objetos astronómicos es un problema desafiante en astronomía y de gran importancia en Chile. Actualmente, en el norte de Chile se construye el Observatorio Vera C. Rubin que espera recibir petabytes de observaciones. Proyectos chilenos han hecho esfuerzos considerables para crear clasificadores automáticos de objetos astronómicos. En esta tesis, investigamos cómo mejorar estos clasificadores considerando técnicas que pueden incluir más información en la clasificación. Seleccionamos dos de los desafíos más importantes de la clasificación de objetos astronómicos: 1) La información necesaria para clasificar los objetos astronómicos proviene de varias fuentes, y 2) existe una cantidad masiva de datos sin etiquetar en comparación con la cantidad limitada de datos etiquetados. En esta tesis se propone una nueva arquitectura llamada *Astronomical Transformer for time series And Tabular data (ATAT)* para hacer frente al primer desafío. Para el segundo desafío, estudiamos varias arquitecturas desde una perspectiva de *Representation Learning*, la perspectiva más estudiada de Machine Learning, para identificar los modelos que mejor utilizan los datos sin etiquetar. Además, se proporciona una metodología sobre cómo usar *ATAT* al considerar datos no etiquetados. Finalmente, concluimos con la importancia del trabajo presentado.

ABSTRACT

Classification of astronomical objects is a challenging problem in astronomy and of great importance in Chile. Currently, in the north of Chile is being built the Vera C. Rubin Observatory that expected to receive petabytes of observations. Chilean projects have made considerable efforts to create automatic classifiers for astronomical objects. In this thesis, we research how to improve classifiers by considering techniques that can include more information in the classification. We select two of the most important challenges of the classification of astronomical objects: 1) The information necessary to classify astronomical objects comes from various sources, and 2) There exists a massive amount of unlabeled data compared to the limited amount of the labeled one. We proposed a new architecture called the Astronomical Transformer for time series And Tabular data (ATAT) to deal with the first challenge. For the second challenge, we studied various architectures from a Representation Learning perspective, the most studied perspective of Machine Learning, to identify the models that best use the unlabeled data. Additionally, we provide guidelines on how to use ATAT when considering unlabeled data. Finally, we conclude with the importance of the presented work.

Esto no habría sido posible, sin las personas que me han apoyado. Para estas personas, muchas gracias.

Agradecimientos

Quiero agradecer profundamente a mi mamá y papá que me han apoyado en todo. Sin toda la ayuda que me han dado a lo largo de mi vida no estaría aquí terminando de escribir esta tesis. Gracias a su apoyo, he podido dedicarle el mayor tiempo posible a lo que me gusta, investigar, así que por esto no tengo palabras que puedan expresar mi gratitud. Quiero agradecer a mi abuela Yeya y abuelo Lolo por acompañarme en mi niñez, y a mi abuela Kako y mi abuelo Nano por acompañarme en mi edad más juvenil. Quiero agradecer a mi hermana Cata, a mis tíos Luli, Socio y Andrea, y a mis primos por estar siempre presentes y por su compañía.

Quiero agradecer a la Yara, mi pareja civil, por hacer mi vida más feliz, ya sea jugando o saliendo. También quiero agradecer a mis amigos del colegio Rocuz, Óscar, Diego, Cristóbal, Fatias, Lolito y Mario por las interminables horas que hemos jugado videojuegos o juegos de mesa. A mis amigos los bellos: Feña, Alonso, Vichotote, Javier, Vicho Vegano, Mauro, Cata y Alfredo por haber hecho esta vida universitaria mucho más alegre y divertida :D. Quiero agradecer a los cuchufletos Maca, Macla, Maos, Mare y Mati por las risas y estupideces que hemos hecho en la universidad. Gracias también a mis amigos que conocí en el extranjero: Belén, Manuel, Mohit e Isa por hacer esta experiencia mucho más divertida y acogedora.

Quiero agradecer al profesor Pablo Estévez que me recibió en su laboratorio cuando apenas sabía que era una red neuronal y por siempre darme consejos desde la sabiduría. También a Francisco Förster y a Guillermo Cabrera por incluirme en sus colaboraciones y dejarme participar en ALerCE. También a la profesora Nancy por dejarme realizar el magíster en Computación, sin ella esto no habría sido posible. Quiero dar gracias a mis amigos del laboratorio Germán, Rodrigo, Esteban, Nico, Rosario, Ignacio, Mau y Óscar por compartir esta única experiencia de aprendizaje, y bueno, por procastinar en el lab cuando era necesario. Quiero agradecer a Pavlos Protopapas por invitarme a realizar una pasantía de investigación y obtener una experiencia inolvidable. Finalmente quiero dar gracias a Pablo Huijse que en términos académicos es la persona que más me ha apoyado y confiado en mí, gracias a él soy el investigador que soy hoy en día.

Aunque no las haya mencionado, muchas gracias a todas las personas con quienes compartí buenos momentos. Saludo especial al Balto, que le gusta acompañarme.

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning is an area of Computer Science that has grown at a fast pace in recent years. This growth has been driven by its great theoretical success as well as its practical applications. Thanks to this success, many laborious human tasks have been replaced by automated machines. Machine Learning has been able to accelerate many areas of science, helping to make scientific discoveries and pushing human technology [62, 74, 91, 106]. Particularly, automating the classification of astronomical objects using machine learning techniques is a big interest in Chile thanks to the facilities installed in the north, to which we have privileged access.

The standard goal in astro-informatic applications is finding sources that allow us to understand better the universe. In practice, supervised classification of light-curves has shown to have considerable success in this task [32, 15, 77, 21, 12]. Despite its great success, light-curves' classification is restricted by two important factors. First, light-curves don't contain all the class information of astronomical objects, and additional relevant information that comes in the form of tabular data also could be helpful. For example, the stellar dust of the sky or the Redshift of stellar objects [42] can be relevant for classification. Second, supervised classification requires a large amount of labeled data to make a successful and high-performing training classification [28, 119, 117, 11]. In reality, in many astronomical applications, it is not possible as labeled data is difficult to obtain in practice.

Classifying astronomical objects with tabular data is not new. For decades, Random Forest models have been used to classify astronomical objects [44, 12, 100]. However, these models don't use light-curves directly, and they rely on hand-crafted features to make classifications. In many cases, this practice can limit their classification power since hand-crafted features cannot contain all the necessary information for classification. Moreover, it is not direct to use Random Forest models in conjunction with unlabelled data, although some approaches exist [71]. On the other hand, many advances in neural networks have been made in Representation Learning and semi-supervised learning [108, 10, 70, 25, 26, 91], techniques that include the use of unlabelled data. Additionally, recent progress in neural network architectures has made considerable advances in processing light-curve time series [82, 88]. However, despite these great advances, most of these models can not use tabular data. In this work, we study the possibility of using new flexible architectures to combine tabular data and light-curve information with improving classification.

To exploit unsupervised information, Representation Learning and semi-supervised learning (SSL) techniques can be used. These techniques are based on the idea of using unlabeled data and not only labeled data. A model that is trained with additional unlabeled data should improve model classification performance more than if only the labeled had been used. Although Representation Learning and SSL have been extensively studied in images, astronomical light-curves are not the case. The light-curves are the modeling of an astronomical event whose length can be variable and with irregular times. The difference between images and astronomical light-curves is huge. Recently, there are many architectures that can deal with the light-curve challenges [104, 88, 105, 82] but currently, there isn't a proper study exploring the Representation Learning and semi-supervised learning capacity of these models. In this work, we want to explore current architectures to use better unlabeled data. We will do this from a Representation Learning perspective, which is the most widespread technique in Machine Learning to study this behavior.

The classification of astronomical objects can mean the development of new astrophysical theories and is of vital importance in research in Chile. Given the advantageous location of Chile with respect to the world, it has attracted the attention of world organizations for the creation of new observatories. Currently, in the north of Chile is being built the Vera C. Rubin Observatory that expected to receive petabytes of observations. Chilean projects, e.g., `http://alerce.science/`, have made considerable efforts to create automatic classifiers for astronomical objects. The development of classifiers is relevant in practice since it is difficult for astronomers to classify the thousands of events that occur every night. The use of tabular data and unlabeled data could further improve the current classifiers since they can use more available information.

1.1 Related Work

In this section, we present a brief introduction to the topics presented in this thesis.

In this thesis, we will be working a lot with **self-supervised models**. Self-supervision refers to processing unlabelled data, such that this training phase obtains useful representations that can help with other downstream learning tasks. How useful a representation space is, is studied by **Representation Learning**. In this case, self-supervision is the task that allows obtaining useful representation, while Representation Learning can include other types of learning.

We will also be working with auto-encoding models, which reconstruct the input data using a bottleneck of information. The compressed space obtained by auto-encoding models should have the most important features of the data. In many Machine Learning applications and in this thesis, auto-encoding models are used as a self-supervised approach. The auxiliary task in this case is reconstructing the input data.

Semi-supervised learning is also a very interesting area to consider. Semi-supervised learning is an area where massive of unlabeled data can be useful by models, and only a few labeled data are present to train the model in a supervised way. Semi-supervised learning in many cases [26, 3] is used as a Representation Learning metric. The Representation Learning perspective is the most widespread approach to testing how capable a model is using unlabeled data, it is the approach we take. One focus that we take is studying how other Representation Learning metrics relate to semi-supervised learning.

In this work, we are interested in classifying astronomical objects. Particularly, we want to use unlabeled data and tabular data as additional sources of information. We used the Representation Learning perspective to study what models take the most advantage of unlabeled data. To do this in the astronomical literature, we need to consider architectures that deal with multivariate variable length and irregular sampling time series (MVLIS).

We limit our scope to using only auto-encoding as the self-supervised approach to obtain good representations. Although there is extensive research on Computer Vision [52, 75, 72, 121] on Representation Learning, there is no related work in astronomical light-curves, being [37] the closest approach. Representation learning of MVLIS time series architecture is a necessary step to understand better what architectures use best the unlabeled data.

We also perform studies on improving astronomical object classification when considering not only light-curves, but also tabular data. We will use one of the most flexible Machine Learning architectures to deal with both modalities, the Transformer [114]. We note that various related works study how to classify MVLIS time series [2, 88, 37] and tabular data [48, 58] separately.

However, there is no related work in the literature that considers both types of classification at the same time. In our work, we want to use one architecture that can perform both tasks. To do this, we adapt most elemental components of related work to propose a general approach. To our knowledge, no related work measures how much the light-curve classification improves when adding tabular data.

1.2 Research problem

In this thesis, we consider two important problems for the classification of astronomical objects: 1) The information necessary to classify astronomical objects comes from various sources, and 2) There exists a massive amount of unlabeled data compared to the limited amount of the labeled one.

The information necessary to classify astronomical objects can come from various sources, light-curves, tabular data, stamps (images), etc. Being the two firsts, the most important. In order to use time series and tabular data is necessary to research new architectures since, in the past, the random forest has beaten all of its competitors using only tabular data. Because of how it works, Random Forests can not use light-curves directly, which means there is a considerable gap for improvements. Particularly, we want to find a neural architecture that can beat Random Forests, which, until this day, haven't been successful. We want to take the neural network approach because there is extensive research on this architecture to use unlabeled data related to our second research problem.

Using unlabeled data for time series is a challenging problem because to study how models use unlabeled data properly, it is necessary to 1) identify architectures that can deal with multivariate variable length and irregularly sampled (MVLIS) time series and 2) study metrics that tell us how well these architectures are using the unlabeled data. To identify the best architecture, considerable related work research is necessary. Moreover, various metrics are required to be computed. These considerations result in extensive experiments to conclude. Additionally, as mentioned in the previous section, this is a highly unexplored territory in astronomical classification.

1.3 Hypothesis

We enunciate two hypotheses:

- H1: The choice of architecture for encoding and decoding MVLIS time series (light-curves) data significantly impacts the performance of representation learning metrics.
- H2: Leveraging both light-curve information and tabular data leads to enhanced classification performance compared to using only light-curve information or tabular data in isolation.

1.3.1 Research question

General research question

- What is the most effective architecture for representation learning and classification of astronomical time series?
- To what extent does the classification performance improve when combining time series and tabular data compared to using light-curve information or tabular data alone?

Specific research questions

- Which encoders/decoders documented in the literature yield the most informative representations of unlabeled data? How do these encoders/decoders compare in terms of representation learning metrics?
- Which Representation Learning metrics are most suitable for semi-supervised classification of MVLIS time series data?
- What approaches can be employed to effectively integrate light-curve information with tabular data in classification tasks?
- How can we combine light-curve information with tabular data information in a self-supervised setting?

1.4 Objectives

1.4.1 General Objective

Design, implement, and test classifiers for light-curves and tabular data. Additionally, design, implement, and test architecture for the classification of light-curves from a Representation Learning techniques perspective.

1.4.2 Specific Objectives

- Design, implement, test and compare different classifiers, encoders, and decoders to identify what architectures extract the most relevant information from the unlabeled data.
- Implement a unified architecture to process both light-curve and tabular data information.
- Provide guidelines of how the architecture that processes both light-curve and tabular data information can be used in self-supervision.
- Leave implementation of the classifiers, encoders, decoders, and Representation Learning techniques tested for future research.

1.5 Methodology

The content of this thesis is separated mainly into three chapters. In Chapter 3, we compare encoders and decoders in terms of Representation Learning using small datasets that contain only real light-curves. In Chapter 4, we extend the most promising technique of the previous chapter and test it in a massive realistic dataset, combining light-curves and tabular data information. In Chapter 5, considering the most promising technique from previous chapters, we provide guidelines on how to use unlabeled data with time series and tabular data.

More specifically, the following steps will be followed for the correct development of this thesis.

1. (Chapter 2) Bibliographic revision of the most relevant work in Representation Learning along with related work of encoders and decoders that can be used for variable length and irregular sampled times series.
2. (Chapter 3) Preprocess Astronomical data two astronomical datasets, 1) the Zwicky Transient Facility (ZTF) light-curve astronomical dataset [30] and 2) the MACHO light-curve astronomical dataset [30].
3. (Chapter 3) Implement an aleatoric variance estimator from a mathematical perspective and see its relevance in step four.
4. (Chapter 3) Implement encoders and decoders that can be used for light-curve astronomical datasets. Compare these models in a Representation Learning scheme to obtain the encoder and decoder that can extract the most useful information of the unlabeled data. Note that all these encoders can also be used as classifiers. For the next steps we will extend the most promising approach to datasets with massive volume.
5. (Chapter 4) Preprocess the massive dataset provided from The Extended LSST Astronomical Time-Series Classification Challenge (ELAsTiCC).
6. (Chapter 4) Extend the best architecture of step four to process the massive ELAsTiCC dataset. Explore how to combine light-curve information and tabular data information with this architecture. Additionally, implement and test a new data-augmentation technique to improve early classification.
7. (Chapter 5) Provide a guideline and a methodology on how to use Transfomers, developed in step six, then using unlabeled data with time series and tabular data.

1.6 Contributions

In summary, this work presents the following main contributions:

- An extensive comparison between encoders and decoders in terms of Representation Learning. We also study what Representation Learning metrics are closer to classification with few labeled data.
- A new aleatoric noise estimator to improve Representation Learning capabilities.
- A new transformer architecture to classify both time series information (light-curves) and tabular data information (metadata, processed features).
- A new data-augmentation method to improve early classification in realistic production scenarios.
- A methodology to use for tabular data and MVLIS time series for self-supervision.

1.7 Thesis Outline

In Chapter 1.1, we provide a background of all models that can encode and decode multivariate variable length and irregular sampling (MVLIS) time series. To process sequences, we study three architectures: Recurrent Neural networks (RNN), Multi-Head Attention (MHA), and Convolution Neural networks (CNN). Additionally, we also explore interpolation techniques that can be used as post or pre-processing techniques of MVLIS time series. Finally, we review some Representation Learning and self-supervised techniques that can use unlabeled data to improve supervised classification.

In Chapter 3, we describe the datasets used for the extensive experiments conducted in this chapter. Afterward, we explain how we will be using different encoders and decoders for our Representation Learning experiments. Additionally, we include the relevance of aleatoric estimation for representation learning. Finally, we conclude based on the large number of experiments conducted.

In Chapter 4, we extend the most promising architecture technique of Chapter 3 to process the massive ELAsTiCC datasets. We propose the Astronomical Transformer for time series And Tabular data (ATAT), a new approach to classify MVLIS time series and tabular data. Additionally, we propose a new data augmentation technique to improve the early classification of the model. Finally, we conclude about the importance of ATAT in the astronomical community.

In Chapter 5, we study and generalize self-supervised models from the Natural Language Processing and Computer Vision community. We propose a methodology and algorithm for using ATAT in a self-supervised way to use unlabeled data. The method proposed is general and can be applied to other domains.

Finally, in Chapter 6 we conclude about the results and experiments conducted in this thesis. We also comment on and justify promising future work to improve the classification of astronomical objects.

Chapter 2

Theoretical Framework

In this chapter, we describe the different architectures and training procedures relevant to the understanding of this thesis. We assume basic knowledge of Artificial Neural Networks, Convolution Neural Networks [69, 90, 103], Recurrent Neural Networks [56, 29] between other neural network architectures. We also assume an understanding of how ANNs are trained in practice; concepts such as learning rate, optimizers, MSE, cross-entropy, etc. A recommended lecture for this topic is the Deep Learning book of Ian Goodfellow [46]. Our work focuses on light-curve classification with ANN. Light-curves are the brightness of a stellar object over a period of time. This brightness is obtained from images captured by observatories [61, 30, 1] at different times. Light-curves can be categorized as multivariate variable length and irregular sampling (MVLIS) time series.

MVLIS time series refers to time series that have a variable number of observations, and the time between each pair of observations is different from other pairs. In many cases, MVLIS time series are also multivariate *i.e.* the observations depend on different filters or bands. A different filter may imply different correlations between observations of different bands or different magnitudes in the observation values. Additional knowledge regarding light-curves and time series will be given in the respective chapters. Some specific architectures that are necessary to deal with MVLIS time series are discussed in Section 2.1.

In this thesis, our main interest is to perform supervised learning with the most available information possible. Particularly, we want to include tabular data in time series classification, which in many cases, provides unique information about stellar objects. The tabular data inclusion is discussed in Chapter 4. Additionally, we want to incorporate unlabeled data when possible. Unlabeled data in astronomy is abundant and obtained every night from observatories. Labeled data, on the contrary, is expensive since astronomers are necessary to label it. To take advantage of unlabeled data we resorted to auto-encoding and self-supervised methods.

We separate this chapter into three sections, one related to each chapter. In Section 2.1, we describe many architectures that can encode or decode MVLIS time series and are used in Chapter 3. In Section 2.2, we describe Transformers that extend from MHA, the most promising architecture from the last step. Transformers are useful for massive datasets and flexible for many input modalities. Finally, in Section 2.3, we describe self-supervision techniques that are used in Chapter 5.

2.1 MVLIS's architectures

We will consider **encoders** as architectures that encode data into latent variables and **decoders** as architectures that can decode latent variables into observed predictions. In Section 3.1, we discuss the **encoders**/**decoders** used. In addition to **encoders**/**decoders**, we also considered interpolation architectures that we call **pre-encoders** and **post-decoders**. The **pre-encoders** are models that transform the MVLIS time series problem into a regularly sampled time series. The **post-decoders** do the contrary, transforming regularly sampled time series into irregularly sampled observations. In both cases, sometimes the interpolation can occur in the embedding space rather than the observed space.

2.1.1 Encoders and Decoders

Recurrent Neural Network

RNNs [98] are specialized neural networks for processing data sequences. There are different types of RNNs with different properties. In our work, we use Long Short-Term Memory Model (LSTM, [56]), which tries to solve typical vanishing and exploding gradients encountered in vanilla RNN. Like RNNs, LSTMs assume the existence of hidden state h_j, but to avoid vanishing gradient, they also consider a memory cell c_j. The sub-index j refers to the sequence step j of sample i, with sub-index i omitted for simplicity. We denote LL as a linear layer (matrix multiplication and bias) and x_j as the input data in step j. Using this notation, we can define the operations of the LSTM network as:

$$
\begin{aligned}
i_j(x_j, h_{j-1}) &= \sigma(\mathrm{LL}^{(1)}(x_j) + \mathrm{LL}^{(2)}(h_{j-1})) && (2.1)\\
f_j(x_j, h_{j-1}) &= \sigma(\mathrm{LL}^{(3)}(x_j) + \mathrm{LL}^{(4)}(h_{j-1})) && (2.2)\\
o_j(x_j, h_{j-1}) &= \sigma(\mathrm{LL}^{(5)}(x_j) + \mathrm{LL}^{(6)}(h_{j-1})) && (2.3)\\
\hat{c}_j(x_j, h_{j-1}) &= \tanh(\mathrm{LL}^{(7)}(x_j) + \mathrm{LL}^{(8)}(h_{j-1})) && (2.4)\\
c_j(\cdot) &= f_j(\cdot) \odot c_{j-1}(\cdot) + i_j(\cdot) \odot \hat{c}_j(\cdot) && (2.5)\\
h_j &= o_j \odot \tanh(c_j) && (2.6)
\end{aligned}
$$

With σ and tanh, the sigmoid activation function and the tangent hyperbolic activation function, respectively. All linear layers $\mathrm{LL}^{(m)}$ with $m \in \{1 \dots 8\}$ have learnable parameters. We refer to $\odot$ as the pointwise multiplication. The variables i_j, f_j, o_j, and c_j are usually called the input, forget, output, and memory gate cell, respectively. A more visual diagram of LSTM is shown in Fig. 2.1.

Multi-Head attention

Multi-Head attention (MHA), in the deep learning context, was introduced by the Transformer [114] and used for Natural Language Translation. Transformers have received a lot

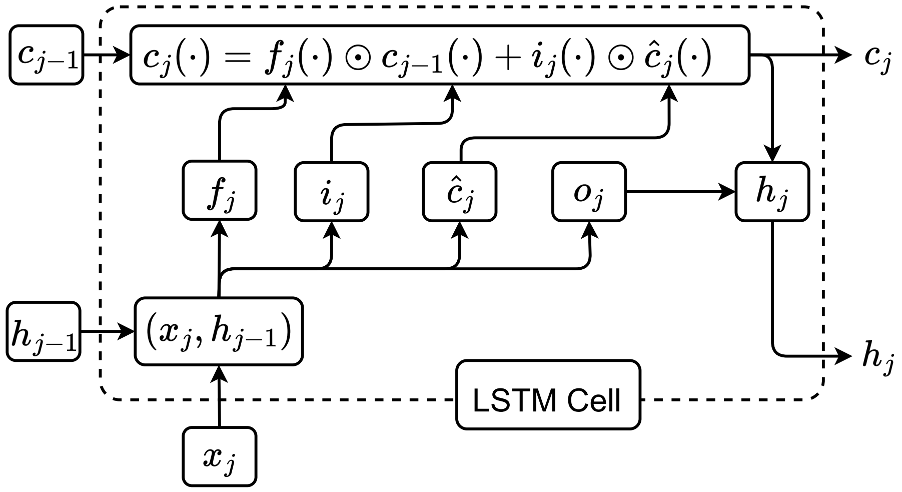


Figure 2.1: LSTM cell. x_j is the input at step j and h_{j-1} the hidden unit of the last step. The variables i_j, f_j, o_j, c_j are the input, forget, output, and memory gate cell, respectively.

of attention in the last few years. Bidirectional Encoder Representations from Transformers (BERT, [31]) extended this work using self-supervision, allowing a wide variety of other tasks by only applying to fine-tuning. Other works have extended the Transformer to use them in Computer Vision [11, 39, 116], being the current state of the art in classification. Recently, DeepMind successfully used only one multi-modal architecture to solve image classification tasks, natural language processing, and reinforcement learning [92]. The flexibility of transformers makes them suitable for multiple tasks, and we think this trend will continue in the future.

In Section 2.2, we will describe Transformers in more detail. In this section, we will focus on MHA, which is the core module of Transformer models. Since MHA is the basic module to deal with sequences, In the next chapter we will compare it to RNNs.

MHA is based on combining input sequence information (values) with the attention that the model should put to each of these values. This attention is learned through backpropagation and is obtained after a Softmax over the dot product between queries and key vectors. How to obtain these queries and keys will depend on the application [114, 105]. In the case of this thesis, we will use Multi-Head self-attention, where the values, queries, and keys are obtained after a linear layer over the input. In general, the linear combination between values and attention is performed in parallel a number of N^{heads} times, and hence, the name of “Multi-Head”. A particular case of MHA is when the queries, values, and keys are obtained from the data input and receives the name Self Multi-Head Attention (Self-MHA). Mathematically, the self-MHA, which generally we will refer to as MHA, can be expressed as:

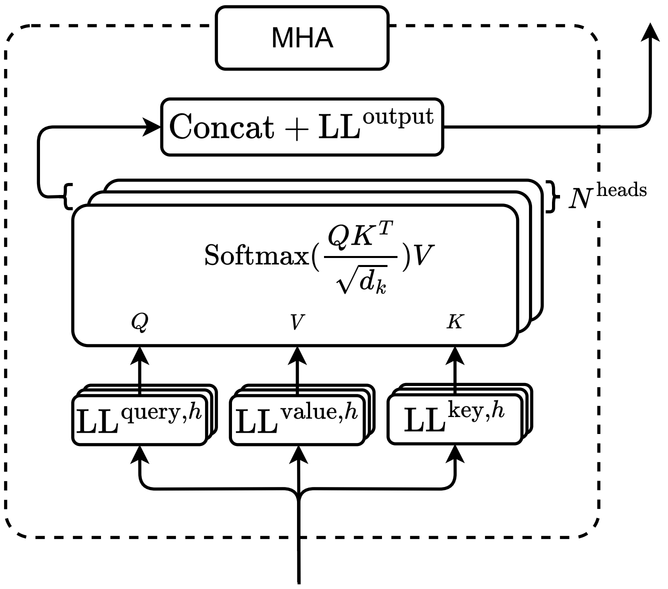

MHA
Concat + $\mathrm{LL}^{\text{output}}$
N^{heads}
$\text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$
Q
V
K
$\mathrm{LL}^{\text{query},h}$
$\mathrm{LL}^{\text{value},h}$
$\mathrm{LL}^{\text{key},h}$

Figure 2.2: Self Multi-Head Attention. Q, V, and K refer to queries, values, and keys, respectively. To obtain these values, a linear layer block, one for each of them, is applied to the input. The next block combines the queries, values, and keys with a dot-product operation. The last and linear layer blocks are repeated N^{heads} times in parallel. The outputs of this parallel operation is combined with a linear layer.

$$v_{j'}^{(h)} = \mathrm{LL}^{\text{value},h}(x_j) \qquad k_{j'}^{(h)} = \mathrm{LL}^{\text{key},h}(x_j) \qquad q_{j'}^{(h)} = \mathrm{LL}^{\text{query},h}(x_j)$$

$$s_{j'}^{(h)} = \frac{1}{\sqrt{d_k}} q_{j'}^{(h)} (k_{j'}^{(h)})^T \tag{2.7}$$

$$a_{j'}^{(h)} = \frac{\exp(s_{j'}^{(h)})}{\sum_{j'=1}^{L} \exp(s_{j'}^{(h)})} \tag{2.8}$$

$$c_{j'}^{(h)} = \sum_{j'=1}^{L} a_{j'}^{(h)} \odot v_{j'}^{(h)} \tag{2.9}$$

$$c_j = \mathrm{LL}^{\text{output}}(\text{Cat}[c_j^{(1)}, \ldots, c_j^{(N^{heads})}]), \tag{2.10}$$

where super-index $h \in [1 \ldots N^{heads}]$ refers to the head index. $v_{j'}^{(h)}$, $k_{j'}^{(h)}$, and $q_{j'}^{(h)}$ refers to values, keys and queries. Usually, $s_{j'}^{(h)}$ is called the alignment score, and $a_{j'}^{(h)}$ is the attention score. d_k is the embedding dimension of the vector of each head. All linear layers are learnable. Note that MHA is completely defined by N^{heads} and d_k. An illustration of the MHA is shown in Fig. 2.2.

Convolution Neural Network

Convolution Neural Network is one of the most revolutionary architectures of Machine Learning. In 2012, AlexNet [69], an architecture based on CNN, won the ImageNet Large Scale

Visual Recognition Challenge (ILSVRC) [1] by a big margin. The main breakthrough of CNNs is the parameter sharing of their weights. They assume local translation invariance that allows for reducing the amount of ANN parameters needed for classification. Consequently, bigger architectures could be developed, increasing the capacity of neural networks.

CNNs have been extensively applied for Computer Vision [34, 26, 63, 112, 5] and for time series [111, 102, 101, 120]. These architectures treat multivariate time series as images with height one and width equivalent to the maximum length of the sequences in the dataset. In this way, CNN can be used straightforwardly. Mathematically, they can be expressed as follows:

$$h^{l+1}_{j,k} = \sum_{n,p} h^{l}_{j-n,p} W^{l}_{n,k} + b^{l}_{k}, \tag{2.11}$$

where variable h^{l+1}_{k} refers to the feature map k in layer $l+1$. The sub-index j refers to the neuron j when considering the horizontal position of the feature map. $W^{l}_{n,p}$ and b^{l}_{k} are learnable parameters. The sub-indexes n and p are auxiliary sub-indexes for neurons in the feature map and the feature map indexes, respectively.

Although CNN has been extensively used in image classification, its original formulation can not handle irregularly sampled data and variable-length data. Pre-processing steps like interpolation models could be useful with CNN to allow these models to be used in the MVLIS time series. In the following section, we describe more about interpolation models.

2.1.2 Interpolation models

Interpolation models in the context of this thesis are models that can transform the MVLIS time series problem into a regularly sampled multivariate time series problem. They can be useful for models that can not directly tackle the MVLIS time series, like CNN. They also may add inductive bias to models that could be useful in the semi-supervised or unsupervised learning case. In the literature, we find two models that stand out, Interpolation-Prediction (InterPred, [104]), described in Section 2.1.2, and Multi-time attention [105], described in Section 2.1.2. In Fig. 2.3 we illustrate the interpolation problem.

Interpolation-Prediction

InterPred is based on the idea of creating "induction points" using high and low bandpass filters. The induction points are regularly sampled, simplifying the MVLIS time series problem. Because induction points are regularly sampled, we can directly apply RNN and CNN after the interpolation prediction step. MHA, in any case, needs to use positional encoding to determine the position of a given observation. For RNN and CNN this step is unnecessary thanks to their inductive bias.

InterPred estimates two sets of induction points positioned in the reference time points $r = [r_1, \dots, r_{N^{ind}}]$. N^{ind} is the number of induction points that are generally equally spaced

[1] `https://www.kaggle.com/getting-started/149448`

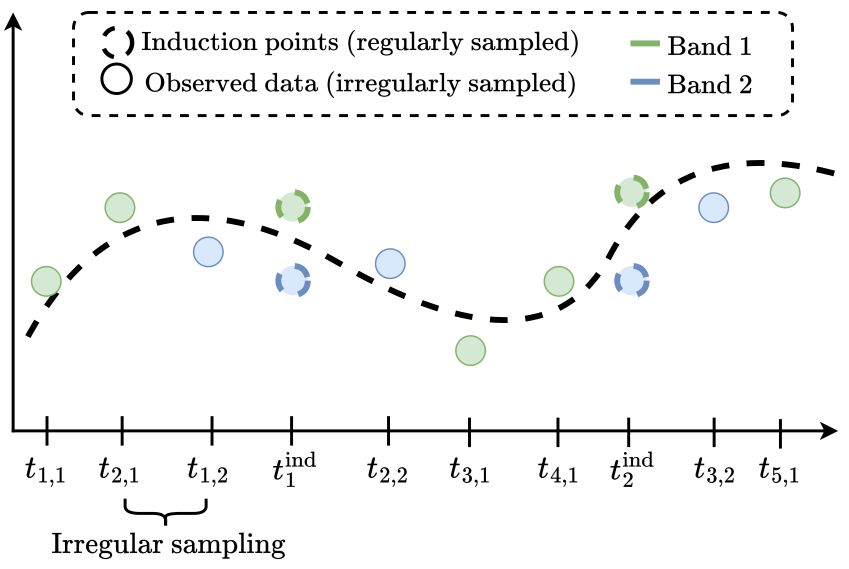

Induction points (regularly sampled)
Observed data (irregularly sampled)
Band 1
Band 2
$t_{1,1}$ $t_{2,1}$ $t_{1,2}$ t_1^{ind} $t_{2,2}$ $t_{3,1}$ $t_{4,1}$ t_2^{ind} $t_{3,2}$ $t_{5,1}$
Irregular sampling

Figure 2.3: Illustration of interpolation points in the observed space for multivariate variable length and irregularly sampled time series. Bands one and two are in green and blue, respectively. Induction points are estimated from the observed data. While, generally, induction points are regularly sampled and have fixed sequence lengths, observed data have variable lengths and are irregularly sampled.

from each other (regularly sampled). One set of reference time points is obtained with a lowpass filter and another set with a highpass filter. Finally, we can estimate induction points using cross-channel or cross-band information. These last induction points subtract the highpass filter induction points obtaining transient information.

Mathematically this can be expressed as:

$$\lambda_{kd}^{\alpha_d} = Z^{\alpha_d}(r_k, \mathbf{t}_d) = \sum_{t \in \mathbf{t}_d} w^{\alpha_d}(r,t), \qquad w^{\alpha_d}(r,t) = \exp(-\alpha_d (r-t)^2), \tag{2.12}$$

$$\sigma_{kd} = \frac{1}{\lambda_{kd}^{\alpha_d}} \sum_{j=1}^{L_d} w^{\alpha_d}(r_k, t_{jd}) x_{jd}, \qquad \gamma_{kd} = \frac{1}{\lambda_{kd}^{\kappa\alpha_d}} \sum_{j=1}^{L_d} w^{\kappa\alpha_d}(r_k, t_{jd}) x_{jd}, \tag{2.13}$$

$$\chi_{kd} = \frac{\sum_{d'} \rho_{dd'} \lambda_{kd}^{\alpha_{d'}} \sigma_{kd}}{\sum_{d'} \lambda_{kd}^{\alpha_{d'}}}, \qquad \tau_{kd} = \gamma_{kd} - \chi_{kd}, \tag{2.14}$$

where sub-index d represents the dimension index and k the $k-th$ induction point. Eq. (2.12). shows the intensity function $\lambda_{kd}^{\alpha_d}$ that measures the amount of real data close to the corresponding induction point. How close an induction point is to a real observation is measured by $w^{\alpha_d}(r,t)$ (exponential quadratic kernel), where α_d is a learnable parameter. In Eq. (2.13), we show σ_{kd} and γ_{kd} induction points obtained by lowpass and highpass filters.

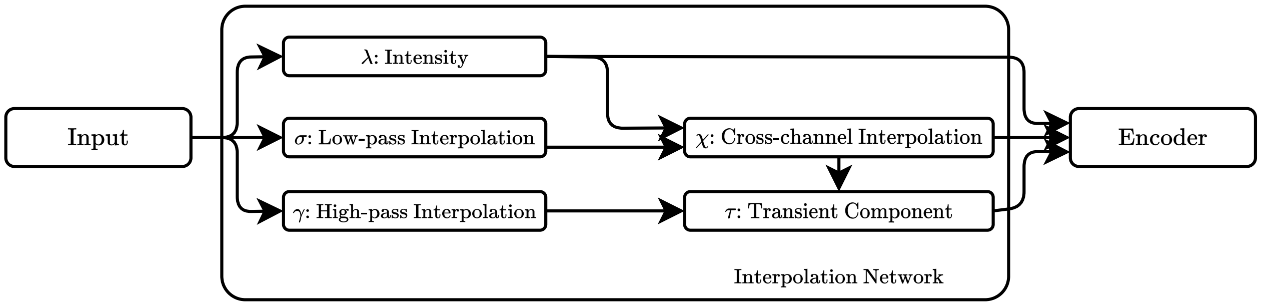

Input
λ: Intensity
σ: Low-pass Interpolation
γ: High-pass Interpolation
χ: Cross-channel Interpolation
τ: Transient Component
Interpolation Network
Encoder

Figure 2.4: Illustration of Interpolation Prediction. σ and γ refer to the low-pass and high-pass filter operation, respectively. With these operations two sets of induction points are. λ refers to the intensity constant of each of the induction points' positions. These induction points in conjunction with their intensity obtain the cross-correlation between channels χ and the transient component τ of the induction points. Finally, λ, χ, and τ are used as input of the encoder.

Here, $\rho_{dd'}$ is a learnable parameter and κ is a hyperparameter that we select equal to 10 as shown in [104]. x_{jd} and t_{jd} the observation and its observed times, respectively. τ_{kd} is the transient information. InterPred use σ_{kd}, γ_{kd} and τ_{kd} as input information of a LSTM. In our work, we also explore CNN and MHA. A flux diagram of InterPred in Fig. 2.4.

Multi-time attention

Multi-time attention (MTAN, [105]) is an interpolation model based on the MHA mechanism. In this model, values and keys are different from queries. The queries are embeddings $\phi_h(t^{\text{ind}})$[2] that codify the positional information of the inductions points, in this case, the times t^{ind}. In simpler words, we ask queries about how an observation (or potentially an embedding) will look at the times of the induction points, see Fig 2.3 for an illustration of the idea of the interpolation problem. MTAN interpolates observed data when encoding and interpolates embeddings when decoding. In what follows, we describe the encoding process of MTAN, the decoding process is similar.

$$v^{(h)}_{j',b} = x_{j,b} \qquad k^{(h)}_{j'} = \phi_h(t_{j,b}) \cdot v \qquad q^{(h)}_{j'} = \phi_h(t^{\text{ind}}_{j,b}) \cdot w$$

$$s^{(h)}_{j',b} = \frac{1}{\sqrt{d_k}} q^{(h)}_{j',b} (k^{(h)}_{j',b})^T \tag{2.15}$$

$$a^{(h)}_{j',b} = \frac{\exp(s^{(h)}_{j',b})}{\sum_{j'=1}^{L} \exp(s^{(h)}_{j',b})} \tag{2.16}$$

$$c^{(h)}_{j',b} = \sum_{j'=1}^{L} a^{(h)}_{j',b} \odot v^{(h)}_{j',b} \tag{2.17}$$

$$c_j = \text{LL}^{\text{output}}(\text{Cat}[c^{(1)}_{j,1}, \ldots, c^{(N^{heads})}_{j,1}, \ldots, c^{(1)}_{j,B}, \ldots, c^{(N^{heads})}_{j,B}]), \tag{2.18}$$

[2]These embeddings based on positional information are constructed in a way to vectorize a scalar value. Usually, they are constructed using sinusoidal waves in each of the embedding dimensions. For more information, please refer to [105]

where v and w are matrices with dimension $d_r \times r_k$, with $d_r \leq d_k$, d_k is the dimension of each head of the attention mechanism and d_r is the dimension of the positional embedding $\phi(t_{j,b})$. The sub-index b refers to the band or filter of the observed data. From the mathematical decomposition, we observe that the only difference with respect to MHA is the addition of the band b as a sub-index and how the queries and keys are obtained. The linear layer $\mathrm{LL}^{\mathrm{output}}$, in this case, does not consider a bias. Intuitively, Eq. (2.15) measures the correlation between the inductions points and the observed data using the time (time embedding). Eq. (2.16) is the attention mechanism to each observed data using this correlation. Eq. (2.17) combines the attention with the input data. Later, Eq. 2.18 combines all attention heads of the model into one output.

2.2 Transformers

Transformers are one of the most flexible architectures being used in the literature. Initially, they were presented in [114] as an encoding/decoding architecture. Nowadays, Transformers are colloquially known by the encoder used in [114] and have been used in Computer Vision [11, 39], Natural language processing [31] and Reinforcement Learning [24]. In this thesis, we will refer to Transformers as the latter architecture, and they are described in the following.

The architecture consists of a Multi-Head Attention step, described in Section 2.1.1, and a forward fully-connected (FF) neural network step with skip connections. The hidden dimension of the FF is $2d_k$, with the architecture shown in Fig. 2.5. Consider the l layer which receives as input the output of layer $l-1$. Then, our transformer architecture can be described as

$$h^l_* = \mathrm{MHA}^{l-1}(h^{l-1}) + h^{l-1}, \tag{2.19}$$

$$h^l = \mathrm{FF}^{l-1}(h^l_*) + h^l_*, \tag{2.20}$$

$$T_{\mathrm{model}} \equiv h^{N^{layers}}, \tag{2.21}$$

where $l \in [1 \dots N^{layers}]$ (N^{layers} being the number of layers of the transformer), h^l is the output of layer l, and h^l_* is the output of the MHA step which includes a skip connection. Notice h^l_* serves as input to a feed-forward network with a skip connection which outputs h^l. Relevant transformer hyperparameters include the number of heads N^{heads} and the number of embedding dimensions d_k. In Fig. 2.5, we give an intuitive diagram of the Transformer model.

2.3 Self-supervision

In this thesis, we consider two self-supervised models widely used in practice to adapt them for MVLIS time series and tabular data in Chapter 4. In Section 2.3.1, we describe the pretraining describe in BERT, one of the most recognized models in NLP. In Section 2.3.2, we describe Masked Autoencoder (MAE) with Vision Transformers used in Computer Vision.

2.3.1 BERT

BERT has been used widely in NLP thanks to its flexible capacity to adapt to many tasks. This capacity comes from pretraining or self-supervision, significantly reducing computational

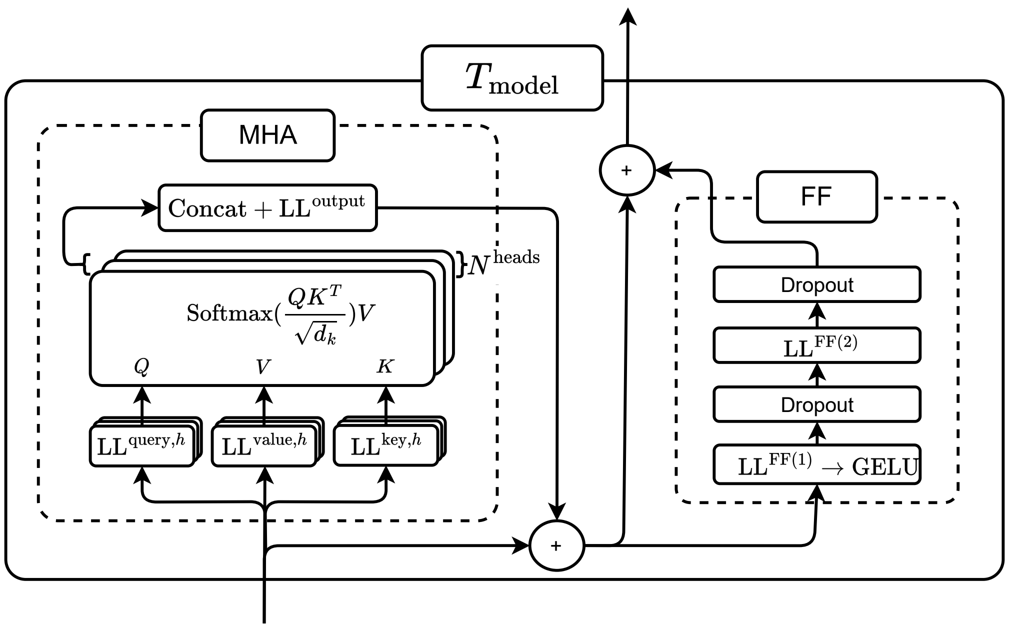
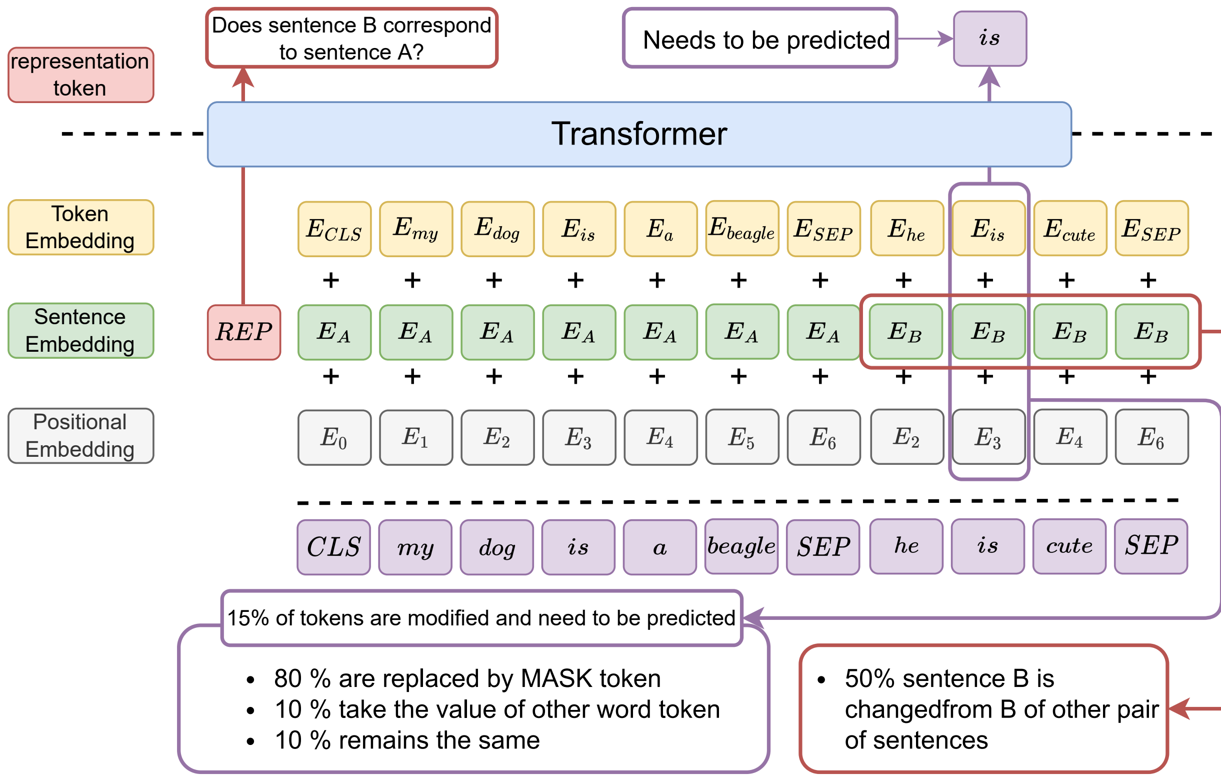

T_{model}
MHA
Concat + $\text{LL}^{\text{output}}$
N^{heads}
$\text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$
Q
V
K
$\text{LL}^{\text{query},h}$
$\text{LL}^{\text{value},h}$
$\text{LL}^{\text{key},h}$
FF
Dropout
$\text{LL}^{\text{FF}(2)}$
Dropout
$\text{LL}^{\text{FF}(1)} \rightarrow \text{GELU}$
+
+

Figure 2.5: Illustration of a Transformer block. A Transformer block is a very general architecture that can be stacked multiple times using skip connections. This is identified in the diagram with the symbol plus. It consists of a Multi-Head attention dot product operation (MHA) on the left. MHA outputs a sequence from the correlation of the input elements. On the right, a non-linear operation is applied after MHA.

resources when tackling new problems. BERT can adapt to text summarization, translation, and text generation, among many others. In what follows, we describe this pretraining phase.

BERT uses the Transformer architecture (see Section 2.2) as its main module to process word sequences. In general, when considering Transformers, it is necessary to transform the input data into tokens such that the model can find all the necessary correlations using the token information. In BERT, three sources of information are necessary to create these tokens. First, the token is created using the positional information to identify if a word occurs after another word. Second, semantic information from each word is included using WordPiece embeddings [118]. Finally, BERT also considers if a word is a part of sentence A (the first part of the word sequence) or sentence B (the second part of the word sequence). This last information source is relevant for pretraining purposes.

The pretraining contemplates two tasks, both of them trying to predict missing information. The first pretraining task consists of replacing, with a 50% probability, sentence B with another sentence B from the dataset. Using a representation token that is added at the beginning of the sequence, the model should predict (after the transformer processing) if the sentence was changed or not. This self-supervised task should force the model to learn the correlation between sentence A and sentence B. The second pretraining task of BERT is trying to predict *masked* words, which is *masked* with a 15% of probability. At the same time, *masked* word can be modified in three ways. First, with an 80 % probability, the token embedding of the word is replaced with a learnable MASK embedding. Note this MASK embedding is equal for all the words in the dataset but with different positional encoding information. In this case, the model should predict the missing or *masked* word considering all contexts of the sentence. Second, with 10 % the embedding is replaced by the embedding of other words, and third, with a 10 % the embedding remains the same. In all cases, the model should predict the correct word that was changed with the *masked.* In Fig. 2.6, we give a more illustrative diagram of this procedure.



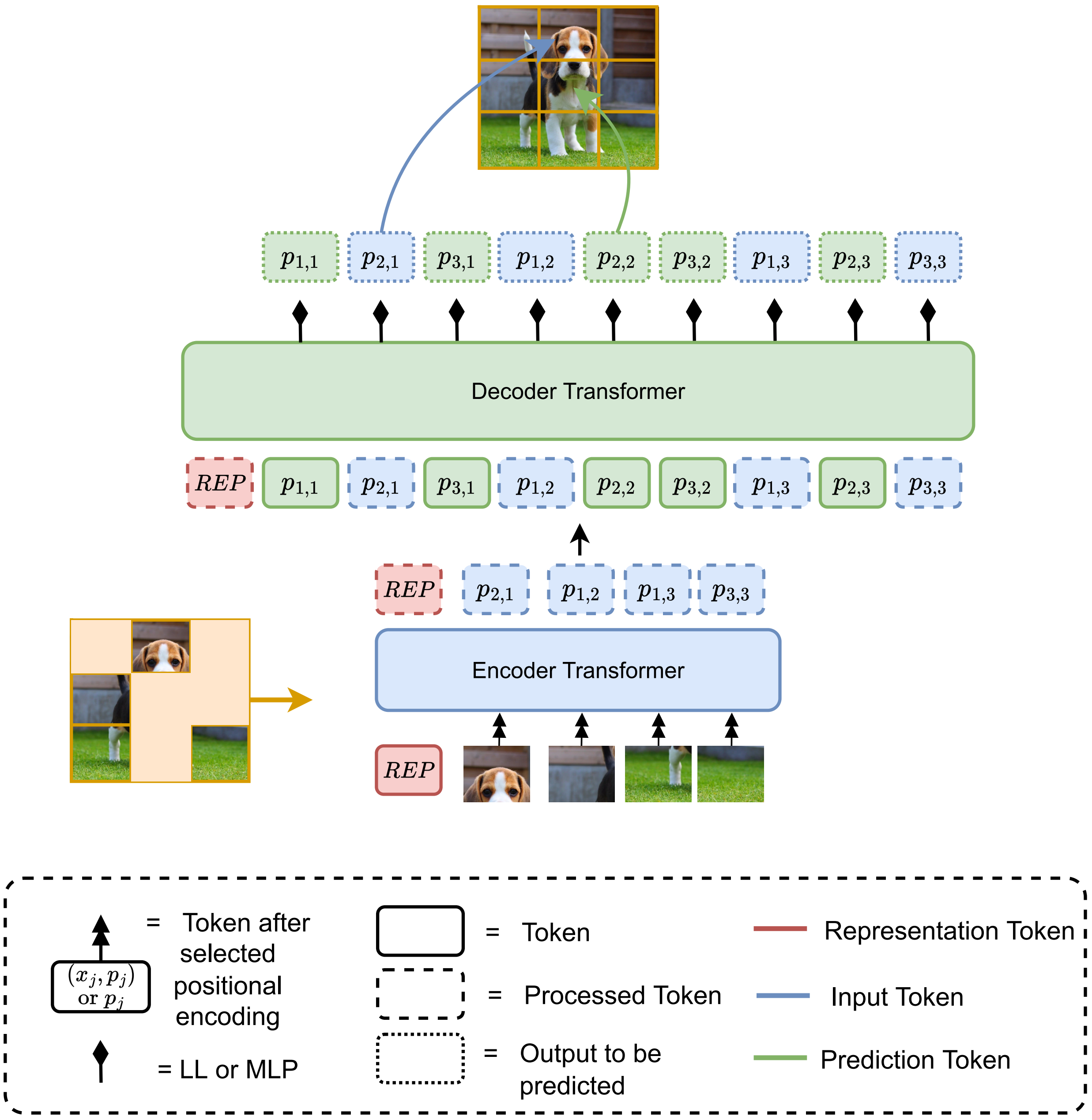
representation token
Does sentence B correspond to sentence A?
Needs to be predicted
is
Transformer
Token Embedding
E_{CLS} E_{my} E_{dog} E_{is} E_{a} E_{beagle} E_{SEP} E_{he} E_{is} E_{cute} E_{SEP}
Sentence Embedding
REP
E_A E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B
Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_2 E_3 E_4 E_6
CLS my dog is a $beagle$ SEP he is $cute$ SEP
15% of tokens are modified and need to be predicted
• 80 % are replaced by MASK token
• 10 % take the value of other word token
• 10 % remains the same
• 50% sentence B is changedfrom B of other pair of sentences

Figure 2.6: Illustration of Bidirectional Encoder Representations from Transformers (BERT). BERT considers two training steps, both trying to predict missing information. First, in red, BERT modifies part of the sequence, and the model should try to predict if part of the sequence was modified or not. Second, in purple, the model modifies word inputs, and it should try to recover such inputs. A representation token in red after that can be used in other applications using fine-tuning.

2.3.2 Masked Vision Autoencoders

Masked Autoencoders [52] is a self-supervised model that is actively used for representation learning in Computer Vision and it is similar to BERT. Using Transformers, MAE tries to predict missing information that is not included in the input. However, MAE uses a procedure that is more similar to autoencoders than only-encoding models like BERT. Additionally, in MAE, the tokens are created using patches of an image, similar to other Visual Transformers [39]. The complete procedure of MAE can be summarized in an encoding and a decoding step that are explained in what follows.

The encoding step consists of a tokenization step, a masking step, and a Transformer layer. The tokenization step separates the input image in patches, applies a linear layer, and adds a positional encoding, obtaining in this way, the input tokens. Note, these tokens can be treated as sequences and we can input them directly into Transformers, similar to BERT. In the masking step, a percentage of these tokens are masked, *i.e.*, they are not considered in the Transformer layer of the encoding step. After encoding these tokens, the decoding step begins, which adds additional tokens to the encoded sequence. These additional tokens are learnable and include their positional information. They have the same position as the masked images patches *i.e.*. these tokens, after the decoder's Transformer will try to recover the masked image patches. In difference with BERT, MAE tries to reconstruct all the input, in this case, the complete image. In Fig. 2.7, we give a more illustrative diagram of the MAE training step.



$p_{1,1}$ $p_{2,1}$ $p_{3,1}$ $p_{1,2}$ $p_{2,2}$ $p_{3,2}$ $p_{1,3}$ $p_{2,3}$ $p_{3,3}$
Decoder Transformer
REP $p_{1,1}$ $p_{2,1}$ $p_{3,1}$ $p_{1,2}$ $p_{2,2}$ $p_{3,2}$ $p_{1,3}$ $p_{2,3}$ $p_{3,3}$
REP $p_{2,1}$ $p_{1,2}$ $p_{1,3}$ $p_{3,3}$
Encoder Transformer
REP
(x_j, p_j) or p_j
= Token after selected positional encoding
= LL or MLP
= Token
= Processed Token
= Output to be predicted
Representation Token
Input Token
Prediction Token

Figure 2.7: Illustration of Masked Autoencoder. First, the image is transformed into patches, and only a subset is selected to be processed by the encoder Transformer. The selected patches are modified into tokens that include the positional information. A representation token is added at the beginning of this sequence of tokens. An encoder Transformer processes the resulting sequence. This encoded sequence is concatenated with learnable tokens that were modified with the positions patches that were not used in the input. The decoder Transformer processes this concatenated sequence, and the output tries to match the original image.

Chapter 3

Representation learning for variable length and irregular sampling time series

Representation learning of data is a fundamental problem in machine learning [9]. The objective of representation learning is to transform the data distribution into a different and more disentangled embedding space (generally, in terms of classification [25, 26, 18]). Ideally, this embedding or "representative space" should be useful for other machine learning tasks since it is a compact representation of data. This general view of representation learning techniques makes them suitable for many machine learning applications like clustering [57, 53, 5, 4], semi-supervised learning [108, 10, 17], anomaly detection [122, 50, 40], and more recently on self-supervised approaches to obtain a state of the art classification performance on images benchmarks.

Representation learning has been studied extensively on image benchmarks but it hasn't been explored with the same extension for other types of data. Multivariate, variable-length, and irregularly sampled (MVLIS) time series is the most general formulation for times series where research on representation learning techniques are necessary; MVLIS time series occur on medical records or astronomical datasets, where classification, semi-supervised learning or anomaly detection techniques could have a great impact. The challenges found in times series are different from the ones found on image benchmarks and depend on the methodology used to obtain a representative space of the data.

We can recognize two general methodologies in representation learning: (1) self-supervision based on auxiliary tasks like contrastive learning, which uses encoders to transform the observable distribution into a low-dimensional space [25, 26, 18, 113] and (2) self-supervision based on encoding/decoding techniques that compress the observable distribution into latent variables through an encoding and decoding process [66, 52] or a reverse, decoding, and encoding process [73, 5]. Both methodologies, when MVLIS time series are considered, require encoders that can deal with variable length and irregularly sampled data. Moreover, for the second methodology (2), we require decoders that can deal with these same difficulties. Our study is focused on representation learning for auto-encoding models where a deep study of encoders and decoders is required.

Different encoders/decoders for fixed-length time series or for regularly-sampled time series have been applied extensively in the literature [111, 102, 120], but not for MVLIS time series. When MVLIS times series are considered, most of the encoder/decoders used are variants of Recurrent Neural Networks [98] including sometimes interpolation models [104, 105] as preprocessing steps (when encoding) or post-processing steps (when decoding). It is not known what encoders/decoders have better representation learning performance for MVLIS time series in contrast to images, which have extensive research [52, 27, 66, 75].

Motivated by this, we test the most relevant encoders/decoders of MVLIS time series literature and we compare them using relevant metrics like accuracy, reconstruction error, and semi-supervised learning fine-tuning. Additionally, we include two decoder baselines, MLP, and attention layers. We note that attention layers have been applied extensively in Natural Language Processing (NLP), vision, and time series literature for pretraining purposes but not when all the input x_i is codified in a single embedding z_i. In NLP usually a sequence of embeddings $z_{i,t}$, dependent on the position t, encodes the information corresponding to the data observation input $x_{i,t}$. In our study, we compress all the input $x_{i,t}$ into a single latent space z_i to follow more closely how representation learning is measured in the image literature.

Additionally, we found that estimating the variance is relevant for the MVLIS time series in terms of representation learning. Time series are noisier than images so the representation learning is benefited from loss functions that measure not only the first moment of the data (mean) like MSE but also from loss functions that measure the second moment (variance).

In summary, the contributions of our work in this Chapter are the following:

- We test various encoders and decoders from the literature that deal with MVLIS time series. We test a combination of pre/post-processing encoders/decoders and encoders/decoders not tested in the literature.
- We provide two decoder baselines not studied in the literature.
- We propose a loss function that forces us to learn the variance of the data.

3.1 Background

Multivariate variable length and irregular sampling time series are sequences of input data $(x_{i,j,b}, t_{i,j,b}) \sim q_\delta(x,t)$, where $x_{i,j,b}$ refers to the magnitude of a given feature sampled at time $t_{i,j,b}$. The sub-index i refers to a given sample $i \in [0, N]$, where N is the number of samples in the dataset. $L_{i,b}$ is the maximum value of j for sample i and band b, with $j \in [1, \ldots, L_i]$. B the numbers of magnitudes filters with $b \in [1, \ldots, B]$. We will consider datasets with correlated bands when $B > 1$ since we expect most of the latent variable z_i will share information from both filters, leaving a small amount to compress specific information from each one.

MVLIS have two main difficulties that not all encoders can fulfill:

1. They have variable length *i.e.* in general $L_{i_1} \neq L_{i_2}$ with $i_1 \neq i_2$.
2. They have irregular sampling *i.e.* in general $t_{i,j_1,b} - t_{i,j_2,b} \neq t_{i,j_2,b} - t_{i,j_3,b}$ with $j_1 \neq j_2 \neq$

j_3.

There are many methods, encoding and decoding techniques that can deal with both difficulties. We will consider **encoders/decoders** as machine learning techniques that can deal with data sequences *i.e.* Multi-Head Attention (MHA), Long-Short Memory (LSTM) and Residual Neural Networks (ResNets). Additionally, we will consider **pre-encoders/post-decoders** as techniques that transform observational (or embedding) data $x_{i,j,b}$ (or $z_{i,j,b}$), which have variable length and irregular time sampling, into induction points $x^{\text{ind}}_{i,j,b}$ (or $z^{\text{ind}}_{i,j,b}$). In the literature, these induction points are regularly sampled and have $L_{i_1} = L_{i_2}$ for $i_1 \neq i_2$. **Pre-encoders** are transformations used in the encoding process $\text{PreEncoder}(x) = x^{\text{ind}}$, where x^{ind} can be encoded afterward with any regular encoder like Transformers, LSTMs or even common Convolutional Neural Networks (CNNs). The **post-decoders** serves the same purpose for the decoding process, we can use any decoder to obtain x^{ind} and afterward a post-decoding process to make predictions $\hat{x}$ of the observational data.

Previous studies [104, 105] have shown that pre-encoders in addition to regular encoders have better classification performance than regular encoders on their own. However, these works have not encoded properly the time information for their baseline methods. Recent studies [88] have shown that modulating the time information, similar to positional encoding [39], could result in an increase in classification performance. In this work, we want to test if pre-encoders have better representation learning performance over regular encoders when the time is properly encoded. We will test the performance of these models using the most common metrics in representation learning literature. Note that measuring representation learning can be more descriptive than purely classification performance delivering information for unsupervised learning tasks like clustering.

Improving representation learning is not only a matter of how the observational data is encoded but also of how the latent space is decoded [27]. For a long time, in VAEs literature [65], the community-made research on decoders since the more complex ones ignore the latent space affecting their representation capabilities. To evaluate how much affect the decoder selection we tested the most common decoder baselines that can deal with MVLIS time series and we also proposed two decoders: A conditioned MLP and a conditional attention model, presented in Section 3.3.5. Finally, we noted that novel variational autoencoding approaches, applied to image benchmarks, estimate not only the pixels but also the covariance between them. This estimation gives place to loss functions that consider high-order statistics that also could affect representation learning. In the following section, we proposed a new loss function that considers higher-order statistics that we included in our representation learning study.

3.2 Decoded distributions in representation learning

Usually in autoencoders, VAEs [65, 94], Autoregressive models [112], Diffusion Probabilistic Models [55, 97] and Normalizing Flows [93, 67] the log-likelihood is estimated as a mean squared error. Using the notation presented in Section 3.1 and considering $B = 1$ we can present the MSE of a single data point as shown in Eq. (3.1).

$$\mathrm{MSE}(x_i, \hat{x}_i) = \frac{1}{L_i} \sum_{j=1}^{L_i} (\mu_{i,j} - \hat{\mu}_{i,j})^2, \tag{3.1}$$

where we implicit wrote μ_i as the first moment of x_i. In general, we will omit j (dimension index) when we refer to a vectorized form of a variable. We can note that MSE is a simplification of the Gaussian log-likelihood given in Eq. (3.2).

$$\begin{aligned}\mathcal{LL}(x_i, \hat{x}_i) =& \frac{1}{2} L_i \log(2\pi) + \frac{1}{2} \log(|\hat{\Sigma}_i|) \\ &+ \frac{1}{2} (\mu_i - \hat{\mu}_i) \hat{\Sigma}_i (\mu_i - \hat{\mu}_i)^T,\end{aligned} \tag{3.2}$$

where $\hat{\Sigma}_i$ is the estimated covariance of the data. If the estimated covariance is diagonal with equal variance for each dimension we would reach MSE up to a normalization constant. These assumptions mean that 1) each dimension j is independent of each other and 2) our estimation of $x_{i,j}$ (reflected by the estimated variance $\hat{\sigma}^2$) is equally good for each dimension j. Depending on the datasets these assumptions can be too restrictive and could affect the representation learning capabilities of the model. Recent approaches [109, 38] have estimated full covariance to improve the generative capabilities of their models. In our work, we found that in MVLIS time series estimating only the variance of the data (first assumption) is sufficiently relevant for representation learning. This case can be written as Eq. (3.3).

$$\mathcal{LL}_{(1)}(x_i, \hat{x}_i) = \frac{1}{2} \sum_{j=1}^{L_i} \left[\log(2\pi\hat{\sigma}^2) + \frac{(\mu_{i,j} - \hat{\mu}_{i,j})^2}{\hat{\sigma}^2} \right], \tag{3.3}$$

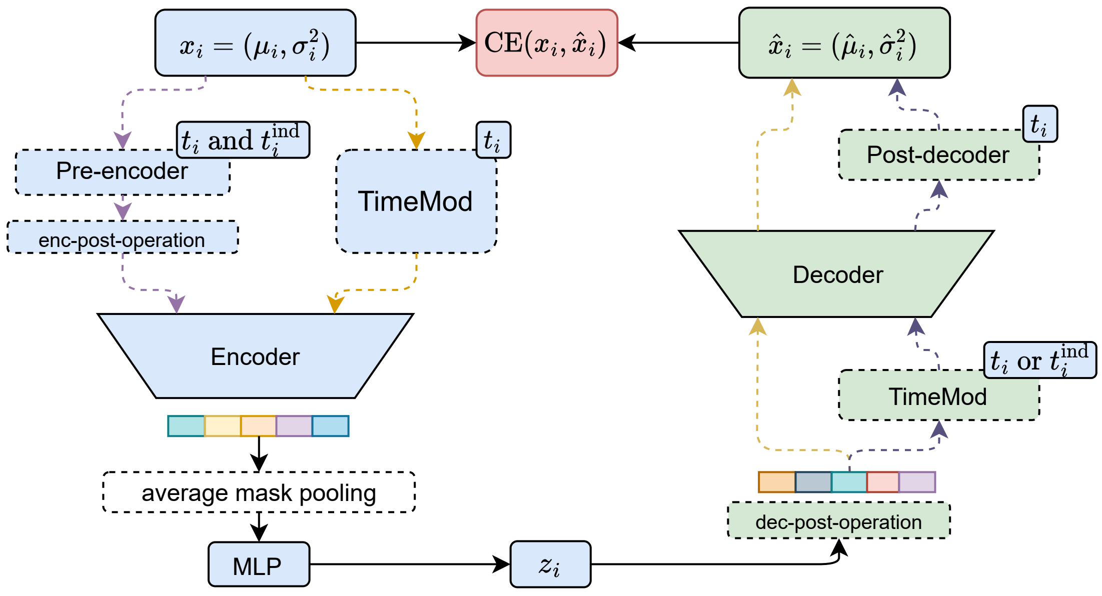
where sub-index (1) on the left-hand side of the equation indicates assuming independence in the dimension of features.

Eq. (3.2) and Eq. (3.3) increases the expressivity over the estimated observational data $\hat{x}$ allowing to integrate $\hat{\Sigma}$ or $\hat{\sigma}^2$ into the loss function. But, how can we add higher-order statistics for observed data x?

For that purpose, we can replace the likelihood of the cross-entropy between two normal distributions. For diagonal covariance, we can write the close form solution of the cross-entropy $\mathrm{CE}_{(1)} = -\mathcal{N}(\mu_i, \sigma^2) \log \mathcal{N}(\hat{\mu}_i, \hat{\sigma}^2)$ as Eq. (3.4).

$$\mathrm{CE}_{(1)}(x_i, \hat{x}_i) = \frac{1}{2} \sum_{j=1}^{L_i} \left[\log(2\pi\hat{\sigma}^2) + \frac{\sigma^2}{\hat{\sigma}^2} + \frac{(\mu_{i,j} - \hat{\mu}_{i,j})^2}{\hat{\sigma}^2} \right]. \tag{3.4}$$

We note that log-likelihood can be interpreted as a simplification of the cross-entropy (indicated in Eq. (3.4)) when $\sigma^2 \to 0$. This case is equivalent to observational data x_i



$x_i = (\mu_i, \sigma_i^2)$
$\mathrm{CE}(x_i, \hat{x}_i)$
$\hat{x}_i = (\hat{\mu}_i, \hat{\sigma}_i^2)$
t_i and t_i^{ind}
Pre-encoder
enc-post-operation
t_i
TimeMod
Encoder
average mask pooling
MLP
z_i
t_i
Post-decoder
Decoder
t_i or t_i^{ind}
TimeMod
dec-post-operation

Figure 3.1: Diagram of the autoencoding process for astronomical light curves using the proposed decoder. Pre-encoders are interpolation models that transform the multivariate variable length and irregular sampling time series problem into regularly sampled and fixed length problems. Post-decoders have the same functionality but inverse. Encoders and Decoders are models to process data sequences. Enc-post-operation adjusts the pre-encoders' output dimensionality with the encoder's input dimensionality. Dec-post-operation transforms the latent embedding into a sequence that is modulated with temporal modulation. Afterward, it is processed by the decoder. The latent embedding results from an average pooling over the sequence obtained by the encoder.

that is completely concentrated in μ_i *i.e.* x_i is a delta function centered in μ_i. To the best of our knowledge, the estimation of the variance $\hat{\sigma}^2$ by using the cross-entropy (Eq. (3.4)) or the likelihood (Eq. (3.2)) has not been used in time series, in the context of representation learning and we found that in some datasets (Section 3.4) these loss functions can be fundamental for it. In practice, we additionally divide the length $L_{i,j}$ to these loss functions to balance the loss function of each sample.

3.3 Methodology

The main purpose of this work is, from an autoencoding perspective, to establish what encoder, decoders, and loss functions obtain the most representative space of the data and if they have an effect on the semi-supervised learning setting. For that purpose, we chose the most common architectures for encoders, decoders, pre-encoders and post-decoders. To compare them fairly we use the most representative architectures, trying to maintain a similar amount of parameters for each of them. For encoders, we selected MHA [114], LSTMs [56] and ResNets [51]. For decoders, we selected LSTM and two new decoder baselines based on MHA and MLPs (Section 3.3.5). For pre-encoders we select Interpolation prediction (InterPred, [104]) and Multi-time attention networks (MTAN, [105]) and for post-decoders MTAN as well.

We note common residual blocks have not been used as baselines for MVLIS time series, and more complex convolutional neural networks have been developed for that purpose. We note that using pre-encoders models or post-decoders models is possible without further sophisticated CNNs, which is why we include common ResNets in our study.

In Fig. 3.1, we showed an illustration of how all the encoders, decoders, pre-encoders, and post-decoders can be simplified in one diagram. In Section 3.3.4, we will describe the implementation details of each architecture used. In what follows, we will discuss our main concerns to compare these models in the most fair way possible.

3.3.1 Fair embedding comparison

This work is centered in low dimensional representations of MVLIS time-series, *i.e.*, all the information is compressed in a low dimensional embedding of the input data. For this purpose, we are using autoencoding models with different encoding architectures, which encode data differently. For example, generally, attention models use tokens to classify the input data, and recurrent neural networks use the last output of the sequence. These differences can affect the embeddings obtained. We aim to test how well these models encode sequences in the fairest way possible. To do this, in each of the encoders we used in our work: MHA, LSTM, and ResNets, we applied an average mask pooling in the sequence output. Additionally, we avoid using skip connections in MHA and LSTM.

3.3.2 Modulating time information in embeddings

Positional encoding is a common practice in MHA to include the positional information of an observation. In this case, the positional information is the time of each observation. We will use a modified version of the temporal modulation proposed in [88] to include the temporal information in the observation. We describe this Temporal Modulation (TM) in Chapter 4, but we also included it here for completeness. Our main interest is observing how different encoder process sequences. In the case of MHA, temporal encoding is always necessary, even when a pre-encoding step is used. This is not the case with LSTM and ResNets, which have an inductive bias to detect if one observation occurs after another. However, in the case of vanilla LSTMs, although they can detect if one observation occurs after another, they can not deal with irregular sampling, which is why in this case, we also applied temporal encoding.

In summary, the temporal modulation consists of applying a linear transformation $\mathrm{LL}_{\mathrm{TM}}$ to the input x. Afterward, we apply an element-wise product $\gamma_1(t_{j,b})$ and a bias $\gamma_2(t_{j,b})$ to the output of the linear layer.

$$\mathrm{TM}(x_{j,b}, t_{j,b}) = \mathrm{LL}_{\mathrm{TM}}(x_{j,b}) \odot \gamma_b^1(t_{j,b}) + \gamma_b^2(t_{j,b}). \tag{3.5}$$

We define the functions $\gamma_b^1(t_{j,b})$ and $\gamma_b^2(t_{j,b})$ as Fourier series

$$\begin{aligned}
\gamma_b^1(t) &= \sum_{h=1}^{H} \alpha_{b,h}^1 \sin\left(\frac{2\pi h}{T_{\max}} t\right) + \beta_{b,h}^1 \cos\left(\frac{2\pi h}{T_{\max}} t\right), \\
\gamma_b^2(t) &= \sum_{h=1}^{H} \alpha_{b,h}^2 \sin\left(\frac{2\pi h}{T_{\max}} t\right) + \beta_{b,h}^2 \cos\left(\frac{2\pi h}{T_{\max}} t\right),
\end{aligned} \tag{3.6}$$

where $T_{\max}$ is an hyperparameter that is set higher than the maximum timespan of the longest light-curve in the dataset (1 in this case), H is the number of harmonics in the Fourier series, and $\alpha^1_{b,h}, \beta^1_{b,h}, \alpha^2_{b,h}$, and $\beta^2_{b,h}$ are learnable Fourier coefficients. Finally, the result of TM can be used as input of MHAs and vanilla LSTMs.

3.3.3 Metrics for representation

Different metrics have measured representation learning in the literature. Being the accuracy in the test set measured with a linear layer trained in the representation space the most common alternative [25, 26, 53]. Other relevant metrics are K-nearest neighbor applied in the embedding space [19] and fine-tuning in a semi-supervised learning way after the pretraining phase [26]. We note that semi-supervised learning has a wide spectrum of models and applications [66, 108, 10, 95]. However, for simplicity, we will refer to semi-supervised learning as the task of pretraining a model in an unsupervised learning manner and fine-tuning this model with a small percentage of labeled data afterward.

In this work, we will consider a linear layer trained in the embedding space to measure representation learning. In the literature, we observed two ways of doing this. The first is obtaining the embedding of the entire training set and training a linear layer until convergence using this dataset. We refer to this linear layer as "offline" (Acc-Linear and Acc-MLP). The second way of doing this is by training the linear layer while the model is in the pretraining or unsupervised phase but without propagating the gradients through the encoder. We refer to this last training as "online" (Acc-On-Linear and Acc-On-MLP). The first way takes longer and more memory, but it is more accurate. However, since a linear layer is a simple model, maybe this step is unnecessary, which is unknown if it makes a big difference in practice.

We note a linear layer tells us how linearly separable is an embedding space, but it does not tell us if the model has compressed all the necessary information for classification. To test this behavior, we follow a similar methodology of a linear layer trained on the embedding space, but instead, we use a multi-layer perceptron. If all the information is compressed, although entangled, it should be separable by a non-linear model.

Finally, we want to identify whether metrics obtained by classifiers trained on the embedding space represent the end-to-end classification task. Accuracy of a fine-tuned model in a semi-supervised manner with a few labeled samples measures exactly this behavior. Currently, in literature, usually linear models are used to test representation. It is unknown if there is a close connection between representation metrics based on linear models or fine-tuned ones.

3.3.4 Implementation details

In what follows, we give details about the hyper-parameters and architectures used in this work. The encoders and decoders used that have the same hyperparameters when encoding and decoding are MHA, LSTM, and ResNet. We used one layer of vanilla MHA, with four heads and a dimension of 48 for each one of them. We used vanilla LSTM with one layer with a hidden dimension of 64. In ResNets, we used the residual block presented in BigGAN [14]. Note in this case, the encoder and decoder are different, but they have the same hyperparameters. We use a scale factor of down-sampling/up-sampling of three, kernel dimension of four, average pooling, and 32 feature maps. The pre-encoders used are MTAN and InterPred. We used 128 induction points and vanilla hyperparameters of their original implementation. In the case of MTAN, this corresponds to an attention model of one head and 48 dimensions, and its temporal embedding has 16 dimensions. The only hyperparameter of InterPred, despite the number of induction points, is the factor κ, which is set as the default value of ten. The post-decoder MTAN has the same hyperparameters as the pre-encoder MTAN. Additionally, a simple MLP decoder is considered based on residual blocks with MLPs. This architecture corresponds to the discriminator used in [33], but we use four residual blocks and 64 hidden units.

In Fig. 3.1, we illustrate how encoders, decoders, pre-encoders, and post-decoders interact. Pre-encoding and post-decoding steps are always necessary when using ResNet as an encoder or decoder. TM is always applied to MHA models; when LSTMs don't use pre-encoder, TM is also applied. The enc-post-operation is an operation needed to adapt the output dimension of the pre-encoder and the input dimension of the encoder. The enc-post-operation is a linear layer for all encoders except for MHA, which always needs TM. The dec-post-operation consists of repeating the latent variable L times, with L the length of the sequence to be reconstructed. Afterward, time modulation with different t_j is applied to each latent variable. The only different dec-post-operation is ResNets. Here, a linear layer is applied to the latent variable. The output is equal to the number of feature maps (in this case, 32) multiplied by three, which is the size of the first feature map. Finally, all models can use a decoder and a post-decoder afterward.

For all experiments, we used a batch size of 256 and a learning rate of $1e-3$. We train until early stopping with a patience of five, and we validate every epoch. For unsupervised learning, the stopping criterion is MSE using the training set. For semi-supervised learning fine-tuning, we use the accuracy in the validation set. All the results provided are the mean and standard deviation of five runs. In Section 3.4.1, we give more details about the sets used.

3.3.5 New decoders

The new decoders correspond to the decoders MHA and MLPs described in Section 3.3.4. Although they are simple, they haven't been compared with the LSTM decoder proposed in [82]. Both, the MHA and the MLP decoders are based on the same idea of this last work. The embedding is repeated a number of times equal to the length of the sequence, as well as the time information included on each embedding. In [82], this is done only using concatenation. In our work, we include TM in MHA and MLP to expand this view.

3.4 Experiments

Considering models, metrics, and datasets, we construct our experiments to determine:

1. What models perform best considering representation learning metrics?
2. If metrics for representation learning metrics based on a linear layer in the embedding space are comparable with semi-supervised fine-tuning representation metrics.

3. If estimating the aleatoric variance has a direct impact on representation learning.

In this section, we did three experiments: a) we varied the decoder used without estimating the variance and with a fixed MHA encoder, b) we performed the same experiment of a) but estimating the variance of the decoder and c) we varied the encoder with a fixed MLP decoder that estimates the variance.

3.4.1 Datasets

Synthetic data

To test if the aleatoric variance estimation $\hat{\sigma}^2$ is relevant for representation learning we construct two simple datasets where the noise is relevant for classification. We named these datasets Synthetic datasets A and B. Additionally, we include a Synthetic dataset C to test the representation learning capabilities of models in which the noise is not relevant for classification. In all Synthetic experiments, the train and test set consists of 5000 and 2000 samples of each class, respectively. All observations are obtained by first sampling 200 samples and maintaining them with a mask sampled from a Bernoulli(0.8) distribution.

To create Synthetic dataset A (Syn-A), we use a simple parameterized sinusoidal wave expressed in Eq. (3.7) with changing parameters for that purpose.

$$f(t, w, \phi, \varepsilon) = \sin(2 \cdot \pi \cdot w \cdot t + \phi) + \varepsilon, \tag{3.7}$$

which classes 0 and 1 are specified as:

1. Class 0: $f(t_A, w_A, \phi_A, \varepsilon_{A,0})$, with $\varepsilon_{A,0} = 0$.
2. Class 1: $f(t_A, w_A, \phi_A, \varepsilon_{A,1})$, with $\varepsilon_{A,1} \sim \mathcal{N}(0, 5)$

Parameters t_A, w_A, ϕ_A are equal for both classes being the only difference the data noise ε. $t_A \sim \mathcal{U}(0, 1)$, $w_A \sim \max(w_A \sim \mathcal{U}(1.4, 3), 0)$.

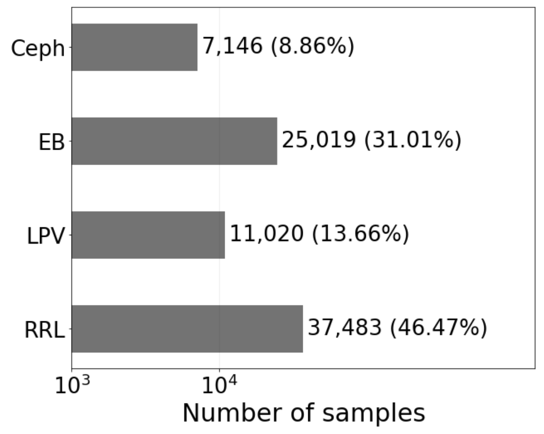
To create Synthetic dataset B (Syn-B) we use a simple parameterized sinusoidal wave modulated inside another sinusoidal wave as expressed in Eq. (3.8)

$$g(t, w, \phi, A, w_2) = \sin(2 \cdot \pi \cdot w \cdot t + \phi) \cdot (1 + A \cdot \sin(2 \cdot \pi \cdot w \cdot t \cdot w_2 + \phi)) \tag{3.8}$$

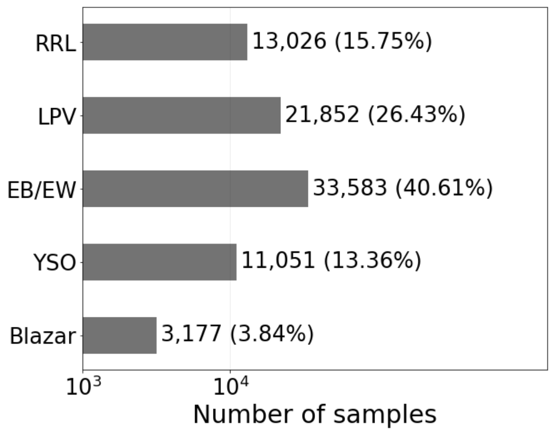
Synthetic dataset B (Syn-B) is constructed with the same parameters t_A, w_A, ϕ_A of Syn-A as follows:

1. Class 0: $f(t_A, w_A, \phi_A, \varepsilon_{A,0})$, with $\varepsilon_{A,0} = 0$.
2. Class 1: $g(t_A, w_A, \phi_A, A_g, w_{A_p})$, with $A_p = 0.25$ and $w_{A_p} = 30$.

Finally, Synthetic dataset C (Syn-C) is obtained by sampling class 0 as $f(t_A, w_A, \phi_A, \varepsilon_{A,1})$. Class 1 is equivalent but replacing the sinusoidal in f by a saw blade function with period one. Although, there is no noise explicitly, the modulated signal has the same purpose.



(a) MACHO dataset



(b) ZTF-DR dataset

Figure 3.2: Histograms of datasets used.

MACHO and ZTF-DR

The Massive Compact Halo Object (MACHO) Project data is a dataset that originally consisted of 21,474 light curves from periodic stars. In [82], this dataset was expanded, producing nearly one hundred thousand light-curves with a length of two hundred and one band. In this work, we used the last dataset and simplified the taxonomy resulting in four classes (see Fig. 3.2). We also used constructed a more complex dataset from The Zwicky Transient Facility (ZTF). ZTF is a public-private partnership aimed at systematically studying the optical night sky. Periodically, ZTF releases data, and thus the name ZTF Data Release or ZTF-DR. However, ZTF-DR releases light-curves but not the classes. To obtain labeled light-curves, we cross-match the ZTF-DR with catalogs that contain labels. This cross-match process consists of locating a source spatially and finding a close candidate in the catalogs. Close enough candidates should be the same object. The catalogs were selected by ALeRCE and are the same ones they used to create their training sets. We selected five classes such as they had enough data for each class. ZTF-DR has two bands, and we constructed it such as it has a maximum number of observations of two hundred in each of its bands.

In both datasets, periodic light-curves were folded, *i.e.*, using the period p, we apply module$(t, p)/p$ to the light-curve. Consequently, the observation times are restricted to intervals 0 and 1. In ZTF-DR, non-periodic light-curves don't have a period, so we applied max-min normalization. Both steps are necessary for interpolation models because they need all the data to be in a fixed interval, in this case, 0 and 1. In Fig. 3.2, we show the class histogram of this dataset. The test set of MACHO and ZTF-DR correspond to 1000 and 500 samples of each class, respectively. The validation set of MACHO and ZTF-DR correspond to 550 and 240 samples of each class, respectively. The validation set is only used in semi-supervised learning experiments. This semi-supervised setting consists of fine-tuning the model end-to-end sampling 10% of the training datasets.

3.4.2 Varying decoders

Synthetic data, Syn-A and Syn-B

When varying decoders, Syn-A and Syn-B synthetic experiments show that estimating the variance improves representation learning. In both cases, Table 3.1 and Table 3.2 show that the model that estimated variance has nearly one hundred percent of Accuracy. On the contrary, the model is confused in terms of the linear model metrics when it doesn't estimate the variance. Note this doesn't mean the model doesn't compress the information of the observed data. For example, in Table 3.1, when the variance is not estimated the model is confused in terms of the linear model metrics, but the Accuracy of the MLP (online and offline) is high. In Table 3.1, when the variance is not estimated, we observed the linear models don't estimate the accuracy well. From these results, we observe that any decoder model in addition to MTAN as a post-decoder performs better than the other models. This suggests that MTAN can capture high-frequency information like the variance of the data for some datasets. Additionally, these experiments don't show an apparent difference between online and offline metrics. In Fig. 3.3 and in Fig. 3.4, we show reconstructed examples from Syn-A and Syn-B, respectively. In Fig. 3.3, we observe the variance of the model is low, which corresponds to the null aleatoric noise of the data. On the contrary, in Fig. 3.4 we observed that the model correctly estimates the variance of the data.

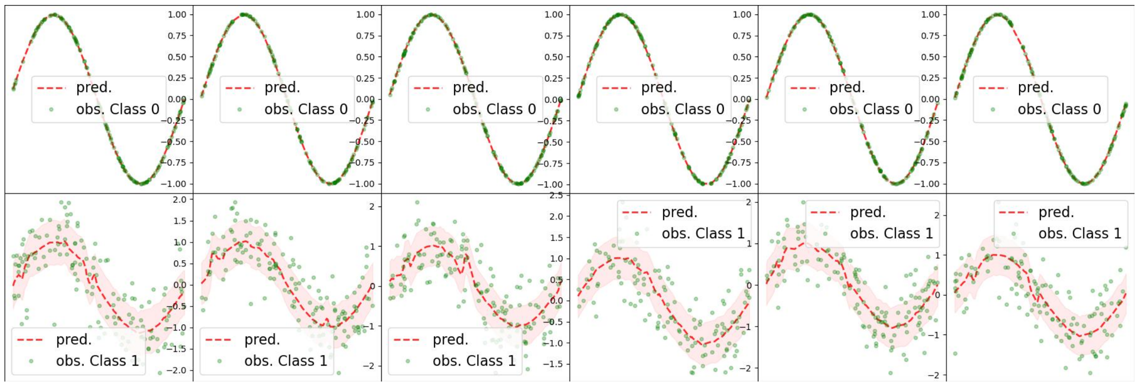


Figure 3.3: Examples of reconstructions in Synthetic dataset A when variance is estimated.

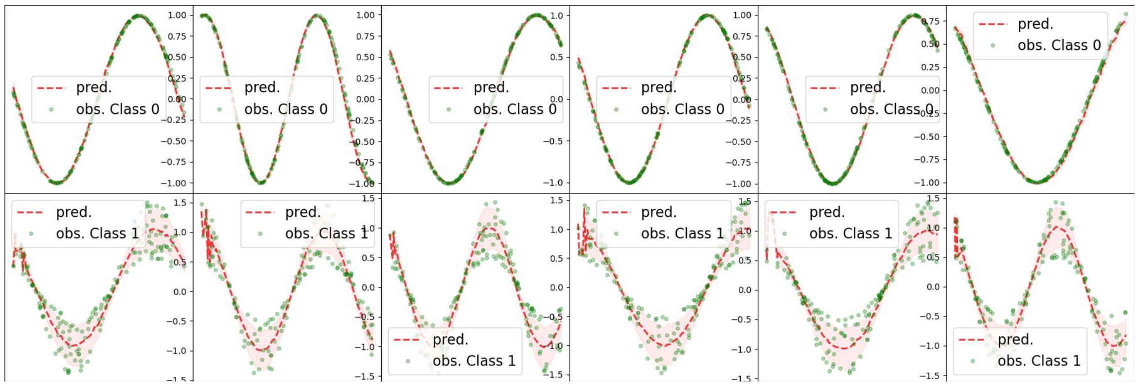


Figure 3.4: Examples of reconstructions in Synthetic dataset B when variance is estimated.

Is var?	Dec	pDec	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	✗	70.15 ± 7.33	68.65 ± 4.28	99.93 ± 0.04	99.74 ± 0.23
		MTAN	87.15 ± 9.45	83.60 ± 8.57	98.99 ± 1.05	99.35 ± 0.86
	MLP	✗	77.62 ± 2.30	74.17 ± 5.89	99.89 ± 0.09	99.77 ± 0.22
	ResNet	MTAN	58.71 ± 8.20	57.60 ± 7.32	55.78 ± 7.66	63.01 ± 17.80
	LSTM	✗	71.14 ± 4.57	63.22 ± 14.11	99.93 ± 0.02	99.91 ± 0.02
		MTAN	79.98 ± 11.27	79.34 ± 12.85	99.94 ± 0.07	99.91 ± 0.10
✓	MHA	✗	99.88 ± 0.10	99.89 ± 0.20	99.95 ± 0.08	99.94 ± 0.07
		MTAN	91.92 ± 17.59	91.88 ± 18.06	99.69 ± 0.62	93.87 ± 13.65
	MLP	✗	99.97 ± 0.03	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	ResNet	MTAN	79.27 ± 26.67	77.31 ± 25.19	79.82 ± 27.08	79.33 ± 26.67
	LSTM	✗	99.81 ± 0.12	99.94 ± 0.04	99.95 ± 0.04	99.95 ± 0.04
		MTAN	99.91 ± 0.09	99.96 ± 0.04	99.98 ± 0.02	99.98 ± 0.03

Table 3.1: Varying decoders, with MHA encoder fixed for Syn-A dataset

Is var?	Dec	pDec	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	✗	62.41 ± 15.46	63.02 ± 14.31	60.98 ± 17.75	73.82 ± 16.38
		MTAN	52.44 ± 3.83	53.43 ± 4.50	55.75 ± 4.24	63.20 ± 8.51
	MLP	✗	90.10 ± 1.42	90.90 ± 1.65	85.98 ± 20.17	95.56 ± 1.83
	ResNet	MTAN	52.87 ± 1.74	55.65 ± 4.19	62.12 ± 3.69	64.46 ± 10.46
	LSTM	✗	90.93 ± 5.99	94.05 ± 3.97	97.56 ± 1.61	97.91 ± 1.15
		MTAN	51.11 ± 1.05	52.28 ± 1.84	56.65 ± 7.86	62.86 ± 7.60
✓	MHA	✗	98.76 ± 0.87	99.13 ± 0.93	99.72 ± 0.23	99.64 ± 0.40
		MTAN	98.76 ± 0.71	99.10 ± 0.46	99.61 ± 0.14	99.88 ± 0.05
	MLP	✗	96.76 ± 2.26	98.52 ± 1.70	99.78 ± 0.11	99.91 ± 0.04
	ResNet	MTAN	73.47 ± 19.57	73.22 ± 19.92	79.29 ± 18.33	78.71 ± 17.97
	LSTM	✗	96.55 ± 2.37	97.61 ± 2.14	99.51 ± 0.22	99.68 ± 0.20
		MTAN	78.54 ± 13.98	79.50 ± 14.80	80.92 ± 17.13	81.51 ± 16.93

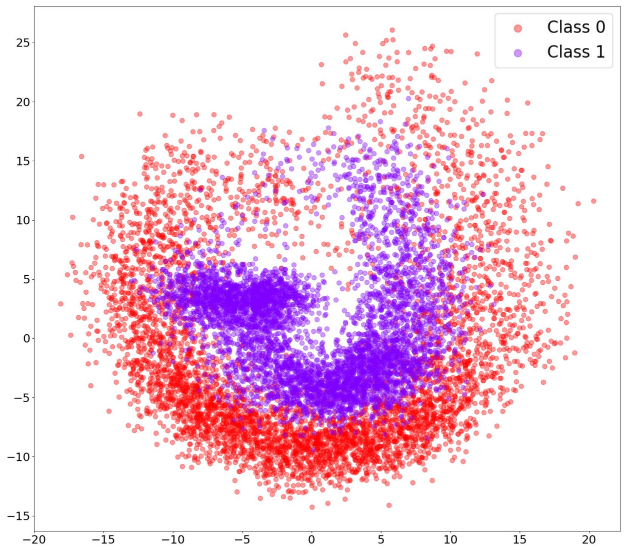
Table 3.2: Varying decoders, with MHA encoder fixed for Syn-B dataset

Synthetic data, Syn-C

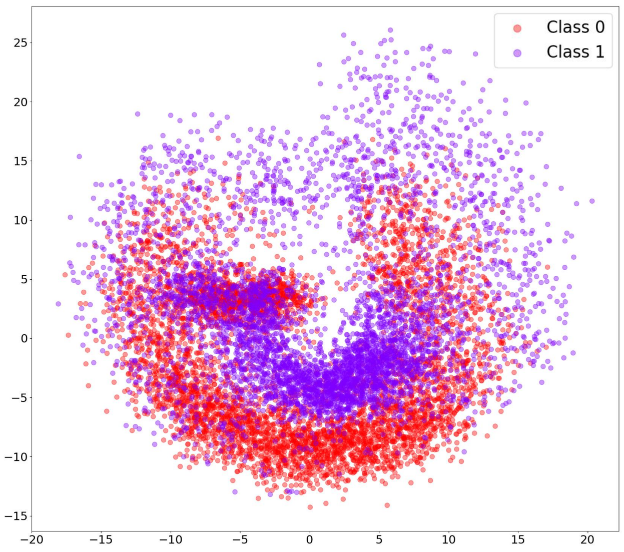
When varying decoders for the Syn-C dataset in Table 3.3, we observe that LSTM or MHA with MTAN is the best model combination when considering the Accuracy of the linear model. In this dataset, we note that estimating the variance doesn't necessarily mean a positive effect on the metrics. This behavior could be contrasted by the fact that this dataset was constructed such as the aleatoric variance of the data wouldn't have an effect in the classes. Note that in all cases the linear models cannot separate the classes but non-linear models can. In Fig. 3.5, we show a UMAP over the embedding of models illustrating this behavior, which doesn't always occur. The best decoder model in terms of Accuracy is MHA and MLP. Here, we also don't observe a big difference between online and offline metrics. Additionally, metrics obtained by non-linear layers obtain nearly a one-hundred percent of Accuracy.

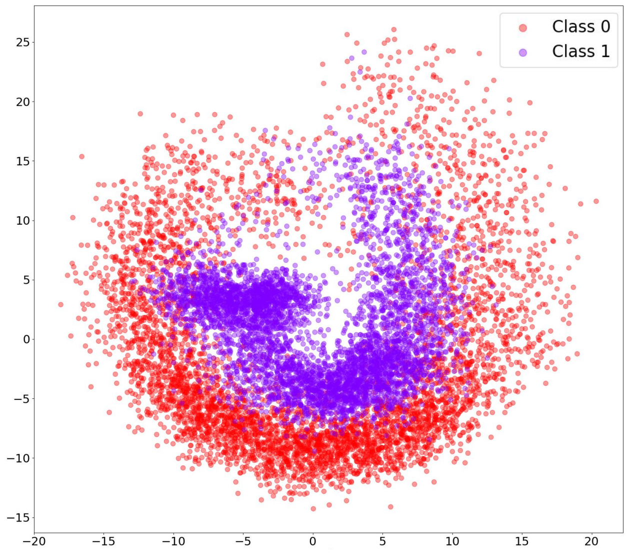
Is var?	Dec	pDec	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	✗	72.17 ± 2.96	71.50 ± 2.93	99.49 ± 0.13	99.84 ± 0.03
		MTAN	85.42 ± 3.21	87.21 ± 3.16	99.33 ± 0.33	99.72 ± 0.14
	MLP	✗	72.57 ± 1.44	73.09 ± 1.02	99.67 ± 0.10	99.89 ± 0.05
	ResNet	MTAN	65.66 ± 20.63	67.47 ± 17.75	78.68 ± 22.25	76.90 ± 22.06
	LSTM	✗	70.01 ± 1.55	70.13 ± 2.17	99.42 ± 0.19	99.82 ± 0.08
		MTAN	60.96 ± 8.55	64.89 ± 6.32	78.13 ± 19.52	80.59 ± 17.76
✓	MHA	✗	70.63 ± 3.26	71.14 ± 3.22	99.48 ± 0.10	99.84 ± 0.05
		MTAN	78.94 ± 4.85	79.32 ± 6.20	99.21 ± 0.20	99.70 ± 0.09
	MLP	✗	71.84 ± 3.17	71.64 ± 2.38	99.69 ± 0.03	99.84 ± 0.04
	ResNet	MTAN	60.43 ± 11.65	58.52 ± 12.12	75.62 ± 19.26	71.02 ± 18.15
	LSTM	✗	66.86 ± 0.58	68.40 ± 0.90	99.33 ± 0.10	99.80 ± 0.03
		MTAN	89.40 ± 8.90	89.49 ± 8.76	96.47 ± 6.26	96.36 ± 6.88

Table 3.3: Varying decoders, with MHA encoder fixed for Syn-C dataset



(a) Real labels





(b) Linear prediction

(c) Non-linear prediction

Figure 3.5: A scatter plot using UMAP over the latent variables. (a) shows real labels, (b) shows a prediction of a linear layer over the embedding, and (c) shows a prediction of an MLP. This figure shows that although the classes are not linearly separable, the embedding contains all the information for classification with a non-linear model.

MACHO and ZTF-DR

MACHO and ZTF-DR are datasets obtained from real data sources. From both datasets in Table 3.5 and Table 3.4, we observe that estimating the variance increases the Accuracy by 2-3 %. The online and offline metrics obtained by the non-linear models don't differ greatly for ZTF-DR. However, in all the other cases there exist a big gap. From these results, we can conclude that we need to use offline metrics to obtain an accurate estimation. In these datasets, we also include the reconstruction error measured in Mean Squared Error. We note that the MLP model is the model with consistently less error, followed by MHA and LSTM. In Fig. 3.6, we show qualitative results of MLP reconstruction in ZTF-DR. Models that use MTAN, generally have a big MSE, which could be due to the higher complexity of the datasets. Finally, we can note that varying the decoders doesn't affect greatly the accuracy, the biggest difference is the reconstruction error.

Is var?	Dec	pDec	MSE	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	✗	9.138 ± 0.775	67.01 ± 0.33	62.15 ± 1.04	82.81 ± 2.82	80.43 ± 3.32
		MTAN	16.250 ± 1.080	64.98 ± 0.68	60.03 ± 0.97	69.04 ± 0.91	64.36 ± 1.27
	MLP	✗	7.823 ± 0.455	66.70 ± 0.91	61.57 ± 0.85	84.40 ± 0.34	81.79 ± 1.53
	ResNet	MTAN	46.448 ± 21.633	66.16 ± 0.40	61.46 ± 0.86	68.68 ± 1.04	65.05 ± 1.19
	LSTM	✗	8.353 ± 0.645	65.93 ± 0.96	61.29 ± 0.93	84.71 ± 0.25	81.31 ± 1.35
		MTAN	12.926 ± 3.758	67.05 ± 0.38	62.23 ± 0.49	73.68 ± 6.63	70.19 ± 6.03
✓	MHA	✗	8.998 ± 0.258	68.97 ± 0.44	63.15 ± 0.86	84.62 ± 1.04	81.53 ± 1.14
		MTAN	16.054 ± 0.438	66.61 ± 0.94	61.40 ± 1.17	71.76 ± 2.58	67.16 ± 2.44
	MLP	✗	8.380 ± 0.290	69.82 ± 0.58	63.55 ± 0.88	85.51 ± 0.39	82.99 ± 0.42
	ResNet	MTAN	61.671 ± 19.524	67.59 ± 0.82	62.47 ± 1.32	75.07 ± 5.42	69.63 ± 7.49
	LSTM	✗	9.044 ± 0.840	69.16 ± 0.87	63.88 ± 1.10	85.78 ± 0.17	82.42 ± 0.85
		MTAN	14.033 ± 2.856	68.31 ± 0.65	63.92 ± 1.08	78.11 ± 4.01	74.71 ± 4.61

Table 3.4: Varying decoders, with MHA encoder fixed for MACHO dataset

Is var?	Dec	pDec	MSE	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	✗	22.75 ± 2.44	85.73 ± 1.34	79.79 ± 1.12	96.07 ± 0.73	95.10 ± 1.21
		MTAN	59.89 ± 1.04	83.92 ± 1.23	77.25 ± 2.75	89.53 ± 1.02	86.36 ± 1.97
	MLP	✗	19.93 ± 1.85	85.24 ± 1.16	80.01 ± 2.29	96.22 ± 0.35	95.27 ± 0.52
	ResNet	MTAN	430.47 ± 265.09	83.86 ± 2.48	76.52 ± 2.98	94.46 ± 3.65	93.14 ± 3.21
	LSTM	✗	21.00 ± 1.47	82.14 ± 1.18	76.18 ± 1.38	96.02 ± 0.30	95.28 ± 0.66
		MTAN	52.46 ± 17.30	84.87 ± 4.38	79.18 ± 4.84	91.42 ± 4.05	89.62 ± 4.73
✓	MHA	✗	22.80 ± 1.15	87.50 ± 1.18	80.74 ± 2.10	96.56 ± 0.56	95.63 ± 0.83
		MTAN	55.36 ± 12.40	84.54 ± 1.94	78.74 ± 3.57	92.42 ± 1.39	89.58 ± 3.57
	MLP	✗	20.92 ± 0.66	88.07 ± 0.63	82.99 ± 2.03	95.70 ± 1.79	95.58 ± 0.60
	ResNet	MTAN	504.94 ± 329.11	87.94 ± 0.85	81.54 ± 3.13	95.79 ± 1.85	93.96 ± 2.54
	LSTM	✗	22.58 ± 2.18	85.18 ± 1.19	78.01 ± 1.90	96.35 ± 0.47	95.10 ± 1.42
		MTAN	44.63 ± 20.09	87.49 ± 1.47	82.24 ± 3.61	94.58 ± 1.32	93.79 ± 1.65

Table 3.5: Varying decoders, with MHA encoder fixed for ZTF-DR dataset

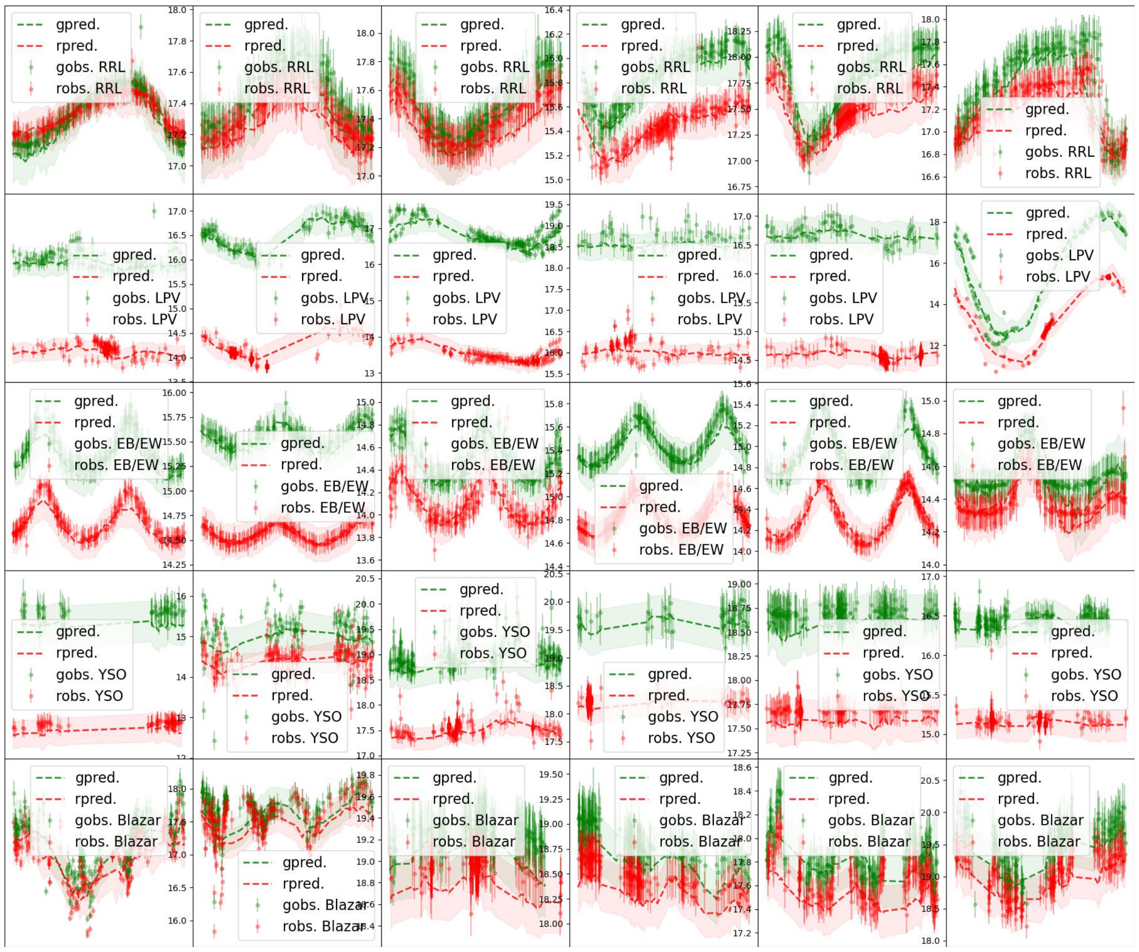


Figure 3.6: Reconstructions in ZTF-DR, with encoder MHA and decoder MLP. Every row a different class. Each color refers to a different band.

3.4.3 Varying encoders

We vary encoders for the datasets of MACHO and ZTF-DR in Table 3.6 and Table 3.7, respectively. When observing the results of both datasets, we note the most robust results correspond to MHA and InterPred-based models. InterPred with LSTM showed the best Acc-Linear, which suggests this model can provide better results in unsupervised learning tasks. Additionally, these models have similar reconstruction errors in MACHO, but when considering more complex dataset like ZTF-DR, the reconstruction error of InterPred-based models increase. Similar behavior is observed with LSTM, this model shows good Accuracy results in MACHO, but its results don't maintain in ZTF-DR. MTAN shows low performance compared with respect to other models. As per previous results, we also observe a big gap between online and offline metrics.

Enc	pEnc	MSE	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	8.719 ± 0.556	69.44 ± 0.92	62.15 ± 1.14	84.71 ± 0.83	83.16 ± 0.58
	LSTM	8.129 ± 0.098	73.17 ± 0.93	67.29 ± 3.57	83.45 ± 0.52	81.35 ± 0.69
MTAN	MHA	12.575 ± 3.213	66.82 ± 0.20	61.83 ± 0.70	72.59 ± 8.09	68.50 ± 7.28
	ResNet	16.209 ± 1.776	66.40 ± 0.81	60.99 ± 1.92	66.69 ± 0.38	62.94 ± 1.90
	LSTM	14.826 ± 0.041	66.79 ± 0.48	61.78 ± 0.62	67.02 ± 0.27	63.35 ± 0.56
Ipred	MHA	8.517 ± 0.074	67.77 ± 0.84	62.22 ± 1.73	82.70 ± 1.36	77.79 ± 0.90
	ResNet	11.037 ± 1.418	72.19 ± 1.17	66.33 ± 1.79	77.76 ± 3.03	72.90 ± 3.05
	LSTM	9.013 ± 0.065	73.39 ± 0.36	69.98 ± 1.31	81.07 ± 0.36	77.94 ± 0.42

Table 3.6: Varying encoders, with MLP decoder fixed and when variance is estimated for MACHO dataset.

Enc	pEnc	MSE	Acc-Linear	Acc-On-Linear	Acc-MLP	Acc-On-MLP
✗	MHA	20.64 ± 0.33	88.22 ± 0.77	83.92 ± 1.38	96.22 ± 1.31	95.90 ± 0.62
	LSTM	234.30 ± 4.06	79.25 ± 0.57	75.92 ± 1.08	86.81 ± 0.09	87.28 ± 0.70
MTAN	MHA	60.36 ± 1.84	77.46 ± 0.31	70.21 ± 2.10	78.40 ± 0.34	69.60 ± 1.83
	ResNet	60.37 ± 2.41	77.02 ± 0.54	68.28 ± 2.98	78.50 ± 0.17	70.45 ± 1.72
	LSTM	58.74 ± 0.41	77.02 ± 0.40	69.74 ± 1.06	78.29 ± 0.34	71.74 ± 0.53
Ipred	MHA	29.64 ± 5.16	83.81 ± 1.94	76.04 ± 8.35	97.27 ± 0.75	83.74 ± 5.10
	ResNet	25.98 ± 3.19	90.07 ± 0.98	83.14 ± 3.06	95.32 ± 1.31	87.49 ± 8.11
	LSTM	22.45 ± 0.96	91.31 ± 0.95	85.73 ± 2.27	96.06 ± 0.70	94.91 ± 0.68

Table 3.7: Varying encoders, with MLP decoder fixed and when variance is estimated for ZTF-DR dataset.

3.4.4 Semi-supervised fine-tuning

Finally, we tested semi-supervised learning fine-tuning with the respective encoders from the last section. We fine-tune these models using 10% of the labeled data from the training set, as specified in Section 3.4.1. In Table 3.8, we observe that the best models in MACHO are MHA, LSTM, and InterPred-based models. However, when considering ZTF-DR, only MHA and InterPred-based models excel.

When we compare the accuracy of fine-tuned models of Table 3.8 with the model's linear Accuracy of Table 3.6, we observed that doesn't exist a clear pattern. For example, in Acc-Linear, LSTM surpass MHA by nearly four points in MACHO, while in fine-tuning MHA surpasses LSTM by two. Another example is InterPred with MHA which has almost the worst accuracy, but when is fine-tuned is the third best. We conclude that the accuracy metric of semi-supervised fine-tuning is very different from a linear layer in the embedding space.

We can note a pattern with accuracy based on MLP (Acc-MLP), in which a low performance in Acc-MLP is connected with a low accuracy in the fine-tuned model. Finally, as in previous experiments, we also noted a big difference between online and offline metrics.

pEnc	Enc	Acc (MACHO)	Acc (ZTF-DR)
✗	MHA	79.97 ± 0.48	94.54 ± 0.54
	LSTM	77.62 ± 0.97	85.30 ± 0.70
MTAN	MHA	68.48 ± 6.96	72.14 ± 0.56
	ResNet	61.72 ± 0.70	72.01 ± 0.26
	LSTM	62.01 ± 0.86	71.96 ± 0.47
Ipred	MHA	76.32 ± 1.48	95.67 ± 0.95
	ResNet	71.83 ± 3.82	92.38 ± 1.75
	LSTM	74.83 ± 1.18	94.20 ± 0.58

Table 3.8: Fine-tuning of different encoders for MACHO and ZTF-DR datasets.

3.5 Conclusion

We noted that time codification is fundamental for comparable baselines of encoders when comparing them against induction-based models. We showed that induction-based models are not better than MHA or LSTM as shown previously in literature [105, 104].

We found that the estimation of $\hat{\sigma}^2$ can be fundamental in tasks when the amount of noise in each class is needed to classify them. In general, across all experiments, we observed that the estimation of $\hat{\sigma}^2$ is helpful for representation learning. To include this estimation we proposed a cross-entropy loss function that can be seen as a more general case of the log-likelihood. To the best of our knowledge, these loss functions have not been used in the context of representation learning of time series.

In terms of metrics, we noted that there exists a big gap between online and offline metrics. Online metrics although faster to compute, produce remarkably worst results and we suggest not using them in practice. We also find a big difference between semi-supervised fine-tuned metrics with respect to linear layers' accuracy on the embedding space. However, we found a strong connection between the accuracy of MLP in the embedding space with respect to semi-supervised based on fine-tuning metrics.

We observed that the best decoding models in terms of representation learning (MHA, LSTMs, and MLPs) showed marginal differences in their results. We found that a simple decoder based on an MLP can be sufficient to decode MVLIS time. MLP also showed the lowest MSE in the reconstruction error. When considering a decoder, we suggest the use of a simple MLP.

When considering encoders, we found that LSTM struggles when considering more complex datasets. MHA and InterPred (with LSTM) showed the most consistent results. InterPred with LSTM showed remarkable results in linear layer accuracy suggesting good capabilities for unsupervised learning tasks. However, MHA showed the best results in non-linear metrics suggesting better capabilities for supervised or semi-supervised learning tasks. Considering these results we will continue working with MHA-based models for the rest of this thesis.

Finally, we plan to share all of our source code to reproduce our experiments. We construct an implementation amenable to the inclusion of new encoders, decoders, pre-encoders, and post-decoders modules. We hope this implementation is helpful for time series literature.

Chapter 4

ATAT: Astronomical Transformer for time series And Tabular data

In this chapter, we present ATAT, the Astronomical Transformer for time series And Tabular data, a classification model that receives as input both light-curves and tabular data from astronomical sources. ATAT consists of a light-curve Transformer with a new time modulation that encodes the time of each observation, and a feature Transformer that uses a Quantile Feature Tokenizer. This model was conceived in the context of the recent Extended LSST Astronomical Time-Series Classification Challenge (ELAsTiCC). ATAT outperforms previous decision tree-based ensemble approaches in terms of classification when trained over the ELAsTiCC dataset. Importantly, some of its variants do not require human-engineered features, with significantly reduced inference computational times (400x faster). The use of Transformer multimodal architectures, combining light-curve and tabular data, opens new possibilities for classifying alerts from a new generation of large etendue telescopes, such as the Vera C. Rubin Observatory, in real-world brokering scenarios.

4.1 Introduction

A new generation of synoptic telescopes are carrying out data-intensive observation campaigns. An emblematic example is the Vera C. Rubin Observatory and its Legacy Survey of Space and Time (LSST) [61]. Starting in 2024, the Rubin Observatory will generate an average of 10 million alerts and 20 TB of data every night. The massive data stream of LSST is to be distributed to Community Brokers[1] that will be in charge of ingesting, processing and serving the annotated alerts to the astronomical community. Collaboration between astronomers, computer scientists, statisticians, and engineers is key to solving the rising astronomical big-data challenges [13, 59].

Automatic data processing based on Feature Engineering (FE, e.g., [85]) and Machine learning (ML), including Deep Learning (DL) have been applied extensively in astronomical data applications, such as light-curve and image-based classification (e.g., [32, 15, 77, 21]), clustering [76, 4], physical parameter estimation [41, 99, 115], and outlier detection [86, 89,

[1]`https://www.lsst.org/scientists/alert-brokers`

100, 60]. The Vera C. Rubin Community Brokers: ALeRCE [43], AMPEL [84], ANTARES [78], BABAMUL, Fink [79], Lasair [107], and Pitt-Google[2] are processing or will process massive amounts of data that is annotated with cross-matches, ML model predictions, and/or other information that is distributed to the community. These scientific products allow astronomers to study transient and variable objects in almost real-time or in an offline fashion for a systematic analysis of large numbers of objects. To enable the former, ML models should be integrated into a complex infrastructure and allow for accurate, rapid and scalable evaluation of tens of thousands of alerts received every minute [81, 96, 16]. In the past, the most common choices have been decision tree-based ensembles (e.g., Random Forest, RF), models with high predictive performance, but high resource usage, due to the FE step. Despite several efforts in applying faster DL-based approaches to the problem of classifying astronomical time series (e.g., Recurrent Neural Networks, RNN, [22, 82, 20, 80, 7, 45, 35]), in practical applications they have not been able to surpass the performance of tree-based ensembles [12, 54, 83, 110].

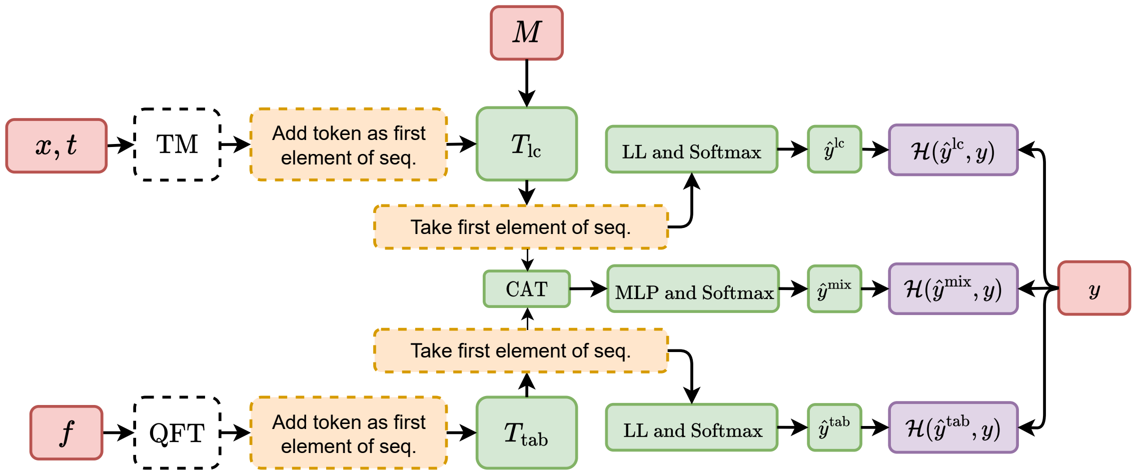
More recently, Multi-Head Attention (MHA, [114]) and Transformers have appeared as promising alternatives to time series encoders in astronomy [2, 88, 37]. These models are faster than RNNs since they have access to all the input simultaneously and not sequentially. However, these works have not explored training with multiple data sources (time series, metadata, and human-engineered features) simultaneously. Moreover, it is unknown if Transformers are useful in more realistic production scenarios.

The astronomical community has made great efforts to create realistic scenarios to test ML models [54], but none of them have contemplated an end-to-end ML pipeline, *i.e.* from the data ingestion to the ML model's outputs. The recent Extended LSST Astronomical Time-Series Classification Challenge (ELAsTiCC[3,4], see Section 4.2) has appeared as a unique opportunity to test the broker's pipelines and ML models in production. ELAsTiCC is a challenge created by the Dark Energy Science Collaboration (DESC) that simulates LSST-like astronomical alerts with the goal of connecting the LSST project, brokers, and DESC by testing end-to-end pipelines in real time. To fulfill this objective, ELAsTiCC started an official data stream on September 28th, 2022. Additionally, ELAsTiCC provided a dataset to train ML models.

In this work, we propose ATAT, an Astronomical Transformer for time series And Tabular data, a model that is based on a Transformer architecture. ATAT is trained with the dataset provided by ELAsTiCC previous to the start of the real-time infrastructure challenge and implemented as an end-to-end pipeline within the ALeRCE [43] broker. ATAT can use time series information and all the available metadata and/or features obtained from other pre-processing steps (see Figure 4.1). In summary, our contributions to this chapter are:

- A new state-of-the-art Transformer model called ATAT, which encodes multivariate, variable length, and irregularly sampled light curves in combination with metadata and/or extracted features.
- A thorough comparison between ATAT and an RF-based baseline (the most competitive

[2]`https://pitt-broker.readthedocs.io/en/latest/`
[3]ELAsTiCC Challenge, link 1
[4] ELAsTiCC Challenge, link 2



M
x, t
TM
Add token as first element of seq.
T_{lc}
LL and Softmax
$\hat{y}^{\text{lc}}$
$\mathcal{H}(\hat{y}^{\text{lc}}, y)$
Take first element of seq.
CAT
MLP and Softmax
$\hat{y}^{\text{mix}}$
$\mathcal{H}(\hat{y}^{\text{mix}}, y)$
y
Take first element of seq.
f
QFT
Add token as first element of seq.
T_{tab}
LL and Softmax
$\hat{y}^{\text{tab}}$
$\mathcal{H}(\hat{y}^{\text{tab}}, y)$

Figure 4.1: Diagram of ATAT, which consists of two branches: 1) on top a Transformer to process light-curves (matrices x, t and M) and 2) at the bottom a Transformer to process tabular data (matrix f). Both information sources are processed by Time Modulation (TM) and Quantile Feature Tokenizer (QFT), respectively, represented as white rectangles. In both cases, the results of this processing are sequences. Subsequently, a learnable token is added as the first element of the sequence. These sequences are processed by the Transformer architectures T_{lc} (light-curves) and T_{tab} (tabular data). Finally, the processed token is transformed linearly and used for label prediction ($\hat{y}^{\text{lc}}$ or $\hat{y}^{\text{tab}}$). In training, we use cross-entropy $\mathcal{H}(\cdot, y)$ to optimize the model (purple rectangle). If both light-curves and tabular information are used at the same time, we additionally minimize the cross-entropy of the prediction $\hat{y}^{\text{mix}}$ resulting from the concatenation of both processed tokens. In the diagram, MLP, LL, and CAT refer to Multi-Layer Perceptron, Linear Layer, and concatenation in the embedding dimension, respectively. For more details see Section 4.3.

model of ALeRCE) in the ELAsTiCC dataset.

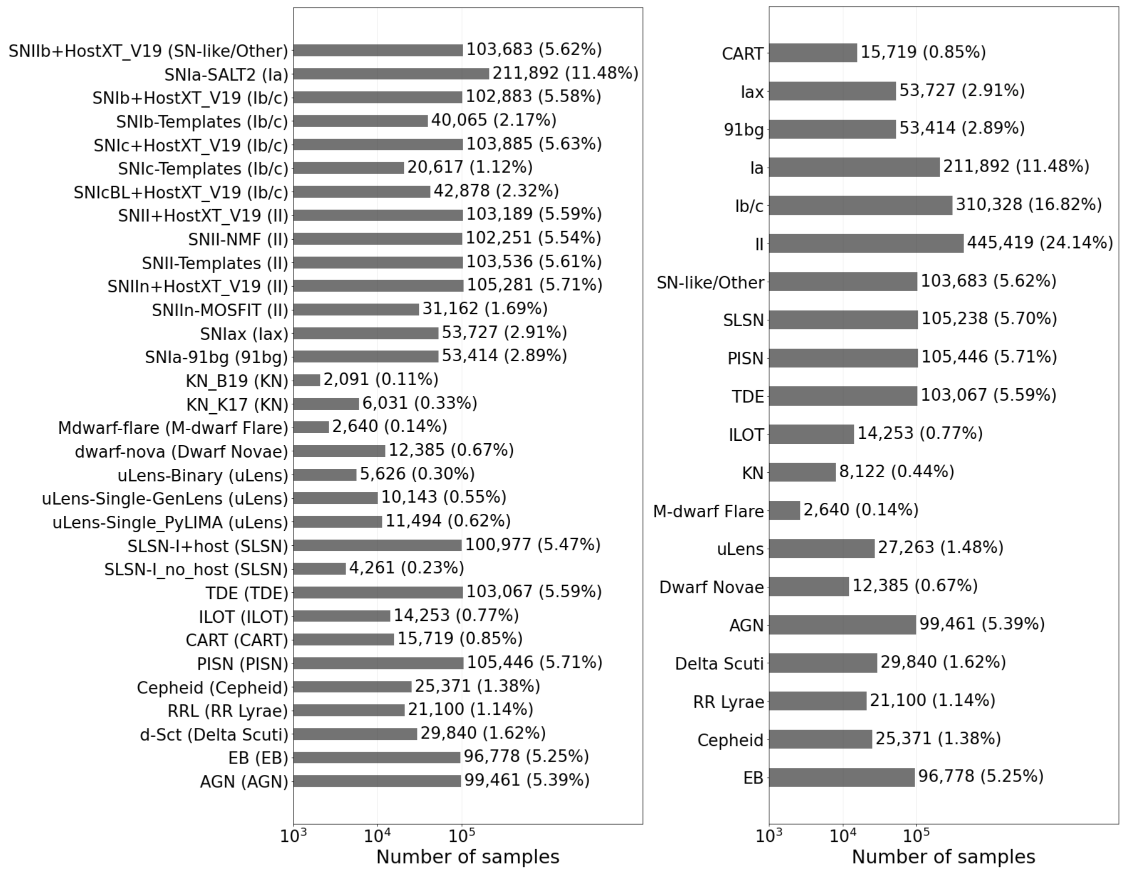
4.2 ELAsTiCC

The ELAsTiCC dataset contains 1,845,146 light-curves in six bands (ugrizY) from simulated astronomical objects distributed in 32 classes as shown in Figure 4.2 (a). We use the same taxonomy than the ELAsTiCC broker's comparison taxonomy, except for the SN-like/other class that includes only SNe IIb (see Figure 4.2 b).

We split the ELAsTiCC dataset into training, validation, and test sets. The test set contains one-thousand samples of each class. The rest of the data is divided into five splits, with which we train five models leaving one fold for validation at each time, used for hyperparameter tuning and early stopping. All the metrics reported in this chapter are the mean of these five models in the test set.

Additionally, the ELAsTiCC dataset was modified by discarding information that is not available in the ELAsTiCC alert stream. For this purpose, we use the PHOTFLAG[5] key to select only non-saturated data, and to only consider forced photometry starting thirty days

[5]Specific format information for the ELAsTiCC dataset.



(a) Original data class distribution (b) Modified data class distribution

Figure 4.2: ELAsTiCC dataset class histogram. In (a) the original taxonomy class distribution is shown. In (b) the taxonomy class distribution selected by Vera Rubin's brokers is shown. Note that we use the SN-like/Other class to include SNe IIb.

before the first alert.

4.3 Methods

4.3.1 ATAT

Here we describe our proposed transformed-based model, ATAT, and the techniques developed to process time series information (light-curves) and tabular data information (metadata and/or processed features). For the rest of the chapter, we will call these models ATAT's variants since different input combinations can be used.

For each astronomical source, we consider two types of data: the light-curve and tabular data composed of static *metadata* (e.g., host galaxy redshifts, if any) and features calculated from the light-curves (e.g., the period of a periodic source). For a particular source, an observation j in band b of its light-curve is described by the observation time $t_{j,b}$ and by the photometric data $x_{j,b} = (\mu_{j,b}, \sigma_{j,b})$, where $\mu_{j,b}$ represents the difference flux[6], and $\sigma_{j,b}$ the flux

[6]Source flux density measured from a difference image.

error. Not all light-curves have the same number of observations. In order to represent this in the model input, we consider fixed size light-curves of the length of the largest light-curve in the dataset and perform zero padding (add zeros for observations after the maximum time in each band, both for $x_{j,b}$ and $t_{j,b}$). At the same time, not all bands are observed at all times. This is represented by adding zeros to $\mu_{j,b}$ and $\sigma_{j,b}$ of the unobserved bands of observation j. In order to mask attention for these unobserved values, we use a binary mask $M_{j,b}$ such that $M_{j,b} = 1$ if observation j is observed at band b, and $M_{j,b} = 0$ if not [114, 31]. For each source, tabular data consists of K features f_k, $k \in \{1, \ldots, K\}$ which, as explained above, may be static or depend on the light-curve.

As a first step, time series and tabular data are processed using Time Modulation and a Quantile Feature Tokenizer, respectively. These steps return sequences that can be used as inputs for common Transformer architectures. Figure 4.1 shows a general scheme of ATAT. Its hyperparameters are further specified in Implementation details section. We noted larger models showed better performance, but we limited their size to reduce the memory requirements in production. For the rest of the chapter we will denote a linear layer as LL.

4.3.2 Time Modulation

Time Modulation (TM) incorporates time information of observation j and band b, $t_{j,b}$, into the difference flux $\mu_{j,b}$ and flux error $\sigma_{j,b}$. Previous works have successfully applied TM in attention models, using processes similar to positional encoding [114]. We construct a variant of the time modulation proposed by [88], which is based on a Fourier decomposition. For each observation j and band b of the light-curve we perform a linear transformation on the input vector, transforming $x_{j,b} = (\mu_{j,b}, \sigma_{j,b})$ to a vector $\mathrm{LL}_{\mathrm{TM}}(x_{j,b})$ of dimension E_{TM}. We modulate this vector by doing an element-wise product with the output of a vector function $\gamma_b^1(t_{j,b})$ and add the output of a second vector function $\gamma_b^2(t_{j,b})$:

$$\mathrm{TM}(x_{j,b}, t_{j,b}) = \mathrm{LL}_{\mathrm{TM}}(x_{j,b}) \odot \gamma_b^1(t_{j,b}) + \gamma_b^2(t_{j,b}). \tag{4.1}$$

We define the functions $\gamma_b^1(t_{j,b})$ and $\gamma_b^2(t_{j,b})$ as Fourier series

$$\begin{aligned} \gamma_b^1(t) &= \sum_{h=1}^{H} \alpha_{b,h}^1 \sin\left(\frac{2\pi h}{T_{\max}} t\right) + \beta_{b,h}^1 \cos\left(\frac{2\pi h}{T_{\max}} t\right), \\ \gamma_b^2(t) &= \sum_{h=1}^{H} \alpha_{b,h}^2 \sin\left(\frac{2\pi h}{T_{\max}} t\right) + \beta_{b,h}^2 \cos\left(\frac{2\pi h}{T_{\max}} t\right), \end{aligned} \tag{4.2}$$

where $T_{\max}$ is an hyperparameter that is set higher than the maximum timespan of the longest light-curve in the dataset, H is the number of harmonics in the Fourier series, and $\alpha_{b,h}^1, \beta_{b,h}^1, \alpha_{b,h}^2$, and $\beta_{b,h}^2$ are learnable Fourier coefficients.

Eq. (4.1) applies a linear transformation to $x_{j,b}$, expanding its dimension. After that, a scale and bias are created as Fourier series (Eq. 4.2) using time $t_{j,b}$. Note that a Fourier series can have enough expressive power for large H. Eq. (4.1) is applied separately for each band, and their output vectors are later concatenated in the sequence dimension. Consequently, the output of TM for a light-curve is a matrix of dimension $L \cdot B \times E_{\mathrm{TM}}$, where B is the number of bands, and L is the maximum number of observations that a band can have for all bands and light-curves in the dataset.

4.3.3 Quantile Feature Tokenizer

Tabular data in ELAsTiCC may include processed light-curve features, static metadata, or a concatenation of both. We process this data before feeding it into a Transformer. We call this process Quantile Feature Tokenizer (QFT) and it comprises two steps. First, a quantile transformation[7] $\mathrm{QT}_k(f_k)$ is applied to each feature f_k of the tabular data of each object, to normalize them as a way to deal with complex distributions. Second, an affine transformation is used to vectorize each scalar value of the attributes recorded in the tabular data

$$\mathrm{QFT}_k(f_k) = W_k \cdot \mathrm{QT}_k(f_k) + b_k, \tag{4.3}$$

where $\cdot$ stands for matrix multiplication, k refers to the index feature, and W_k and b_k are vectors of learnable parameters with dimensions E_{QFT}. In other words, the k^{th} scalar feature f_k is transformed by QT_k and then vectorized by multiplying it by W_k and adding b_k. Notice that a different transformation is applied to each feature of the tabular data. The output dimension E_{QFT} is an hyperparameter to be chosen. This methodology is similar to [48], but we additionally apply the quantile transformation to each feature that is fitted before training the model.

4.3.4 Transformers

The Transformer architecture is based on Bidirectional Encoder Representations from Transformers [BERT, 31] and Vision Transformers [39] which aim at processing sequential information. The architecture consist of a MHA step and a forward fully-connected (FF) neural network step with skip connections.

A Transformer architecture can be summarized as l layer which receives as input the output of layer $l-1$. Then, our Transformer architecture can be described as

$$h^l_* = \mathrm{MHA}^{l-1}(h^{l-1}) + h^{l-1}, \tag{4.4}$$

$$h^l = \mathrm{FF}^{l-1}(h^l_*) + h^l_*, \tag{4.5}$$

$$T_{\mathrm{model}} \equiv h^{N^{layers}}, \tag{4.6}$$

where $l \in [1 \ldots N^{layers}]$ (N^{layers} being the number of layers of the Transformer), h^l is the output of layer l, and h^l_* is the output of the MHA step which includes a skip connection. Notice h^l_* serves as input to a feed-forward network with a skip connection which outputs h^l. Relevant Transformer hyperparameters include the number of heads N^{heads} and the embedding dimensionality E_T, which are specified in Implementation details section. Additionally, we use a learnable classification token of dimension E_T that is concatenated at the beginning of the input sequence. This token representation after the Transformer is fed into an output layer that performs the classification task. The dimension number of E_T is equal to E_{TM} for the light-curve Transformer and E_{QFT} for the tabular data Transformer.

When only a single data source is considered (e.g., only light-curves data), we take the first element of the Transformer's output sequence, and apply a linear layer plus a softmax activation function. When two data sources are considered, two Transformers T_{lc} and T_{tab}

[7]A quantile transformation transforms features into a desired distribution by mapping the cumulative distribution function of the features to the quantile function of the desired distribution.

are used to process light-curve and tabular data information, respectively. The first output elements of both sequences are concatenated, and a multilayer perceptron plus a softmax activation function are applied to produce the label prediction (see Figure 4.1).

4.3.5 Mask temporal augmentation

To improve early classification performance, we train ATAT on light-curves reduced up to a randomly selected time instant. During training, a day $t^* \in \{8, 128, 2048\}$ is randomly selected for each light-curve, and the values of mask M corresponding to times $t > t^*$ are set to zero. Note that $t^* = 2048$ is equivalent to using the complete light-curves. Times are selected from a limited discrete set since it is unfeasible to compute the features at arbitrary times. Hereafter, we refer to this augmentation method as Masked Temporal Augmentation (MTA).

4.3.6 Implementation details

ATAT variants are evaluated every twenty-thousand iterations, and early stopping with a patience of three evaluations is used. Models are trained using the Adam optimizer [64] until early stopping with learning rate of $2 \cdot 10^{-4}$ and a batch size of 256. We use class balanced batches. For both T_{lc} and T_{tab}, $N^{heads} = 4$ and $N^{layers} = 3$. For $T_{\mathrm{lc}}/T_{\mathrm{tab}}$ all input and output dimensions of linear layers are 48/36 with the exception of the hidden layers of FF (Eq. 4.5) which are 96/72. Note that this implies that $E_{\mathrm{TM}} = 48 \cdot 4 = 192$ and $E_{\mathrm{TT}} = 36 \cdot 4 = 144$. We select $T_{max} = 1500$ and $H = 64$. We use a dropout of 0.2 for training. All Nans, inf and $-$ inf in features and metadata are replaced by -9999.

4.4 Random forest baseline

4.4.1 Balanced Hierarchical Random Forest

We compare our Transformer models against the Balanced Hierarchical Random Forest (BHRF) model of [110] adapted for the ELAsTiCC dataset. This section describes the differences between the original BHRF and its ELAsTiCC adaptation.

The original BHRF described in [110] is composed of four Balanced Random Forest models [23] that are used in a hierarchical structure. The top model classifies each light-curve into Transient, Stochastic and Periodic classes. Then each one of these three groups is further classified using its own Balanced Random Forest model. In this work, the Transient group includes the following classes: Calcium Rich Transients (CART), SNe Iax (Iax), SNe 91bg-like (91bg), SNe Ia (Ia), SNe Ib/c (Ib/c), SNe II (II), SNe IIb (SN-like/Other), Superluminous SNe (SLSN), Pair Instability SNe (PISN), Tidal Disruption Events (TDE), Intermediate Luminosity Optical Transients (ILOT), and Kilonovae (KN). The Stochastic group includes the following classes: M-dwarf flare, Dwarf novae, active galactic nuclei (AGN), and gravitational micro lensing events (uLens). The Periodic group includes the following classes: Delta Scuti, RR Lyrae, Cepheid, and Eclipsing Binary.

The ELAsTiCC dataset has six bands and its light-curves contain difference fluxes. In comparison, the alert stream from ZTF [8, 49] classified by the original BHRF model has only two fully-public bands and it offers light-curves in difference magnitudes. In order to deal with this, we modified some of the original features from [110]. All light-curve based features were modified to use fluxes as input instead of magnitudes. The supernova parametric model (SPM) from [110] was modified to better handle the six bands available and the extra information of redshift and Milky Way dust extinction. The fluxes were scaled using the redshift information available and the WMAP5 cosmological model [68, 6], and also deattenuated using the extinction information and the model from [87]. This means that some of metadata information was used in the computation of features. We remove some features from [110] that were not simulated by ELAsTiCC, e.g., the star-galaxy score from the ZTF stream and the color information from ALLWISE. The coordinates of the objects are not used because they were not simulated in a realistic way for each of the astrophysical classes. These engineered features are also used for ATAT in Section 4.5. In the next section we give a comprehensive list of the modified features.

In order to be consistent with ATAT, the Random Forest (RF) models were trained using light-curves trimmed to 8, 128 and 2048 days long. From each astrophysical class 15,000, 9,000 and 3,900 light-curves were sampled to train the transient, periodic and stochastic RF, respectively. Each RF has 500 trees, with the exception of the RF for transient classes, which has 350 trees. The transient RF also uses a minimum impurity decrease value of 0.00003. Both decisions were made to diminish the final size of the model in order to facilitate deployment.

4.4.2 Processed features details

To extract color information from the difference light-curves, on each band we take the absolute value of the flux, compute the percentile 90 and save that value. Following the order ugrizY, we take the value of the percentile 90 previously saved for one band and divide it by the value of the next band. To avoid dividing by zero, we add 1 to the denominator.

Most differences in the features used are related to extracting information from supernova-like light-curves. This is the list of supernova features, which are computed for each band:

- `positive_fraction`: fraction of observations with a positive flux value.
- `dflux_first_det_band`: difference between the flux of the first detection (in any band) and the last non-detection (in the same selected band) just before the first detection.
- `dflux_non_det_band`: same as `dflux_first_det_band`, but instead of using the last non-detection before the first detection, we take all the non-detections before the first detection and compute the median. Later, this median is subtracted from the flux of the first detection (in any band).
- `last_flux_before_band`: flux of the last non-detection (in the selected band) before the first detection (in any band).
- `max_flux_before_band`: maximum flux of the non-detections (in the selected band) before the first detection (in any band).
- `max_flux_after_band`: maximum flux of the non-detections (in the selected band) after the first detection (in any band).
- `median_flux_before_band`: median flux of the non-detections (in the selected band) before the first detection (in any band).

- `median_flux_after_band`: median flux of the non-detections (in the selected band) after the first detection (in any band).
- `n_non_det_before_band`: number of non-detections (in the selected band) before the first detection (in any band).
- `n_non_det_after_band`: number of non-detections (in the selected band) after the first detection (in any band).

As we were not sure if the ELAsTiCC stream would indicate if the observations were alerts or forced photometry (i.e. if the signal was strong enough compared with the noise), for the supernova features we considered an observation as a detection if the absolute value of the difference flux was at least 3 times larger than the observation error.

With respect to the Supernova Parametric Model [SPM, 110], one SPM model per band was fitted to the data, but the optimization was done simultaneously and penalizing the dispersion between the parameters on different bands. The extra term added to the cost function is

$$\left\langle \begin{pmatrix} \widehat{Var}(A)+1 \\ \widehat{Var}(t_0)+0.05 \\ \widehat{Var}(\gamma)+0.05 \\ \widehat{Var}(\beta)+0.005 \\ \widehat{Var}(t_{rise})+0.05 \\ \widehat{Var}(t_{fall})+0.05 \end{pmatrix}, \begin{pmatrix} 0.0 \\ 1.0 \\ 0.1 \\ 20.0 \\ 0.7 \\ 0.01 \end{pmatrix} \right\rangle \tag{4.7}$$

where the variances are estimated over the different bands and the coefficients were found experimentally. The original SPM code was modified to avoid numerical instabilities. To speed up the optimization, the gradient of the cost function is computed using the JAX library [8]. The initial guess and the boundaries for the parameter optimization were tuned for the range of values in the ELAsTiCC dataset.

4.5 Results

4.5.1 Comparison between ATAT and RF-based baseline

Figure 4.3 (a) shows the test-set F1-score of two selected ATAT variants using different data sources with a data augmentation strategy (Masked Temporal Augmentation, MTA, see Section 4.3.5) and the Random Forest baseline (RF, described in Section 4.4.1) as a function of the number of days after the first alert, starting at eight days and progressively increasing in powers of two up to 2048. These ATAT variants outperform the RF model for all light-curve lengths, specially for shorter light-curves. We further compare these models by measuring the F1-score, recall and precision in a per-class basis as shown in Table 4.1. The labels indicate whether the models were trained using the light-curve (LC) data, metadata (MD), engineered features (Features), or combinations of these. In Section 4.5.3, we compare ATAT and RF in terms of confusion matrices. The ATAT's variants surpass the RF in most classes. In particular, the ATAT variant based on LC and MD performs better in the SN subclasses, but the ATAT variants that use all data sources obtain better scores in the periodic

[8] `http://github.com/google/jax`

sub-classes. Three of the four classes where the RF-based baseline outperforms (F1-Score) ATAT, namely KN, CART and M-dwarf are also the ones with fewer examples in the dataset (see Figure 4.2). This may be explained by the differences in the class-balancing strategies, with the RF being more robust to overfitting in the minority classes.

4.5.2 Classification performance of ATAT variants

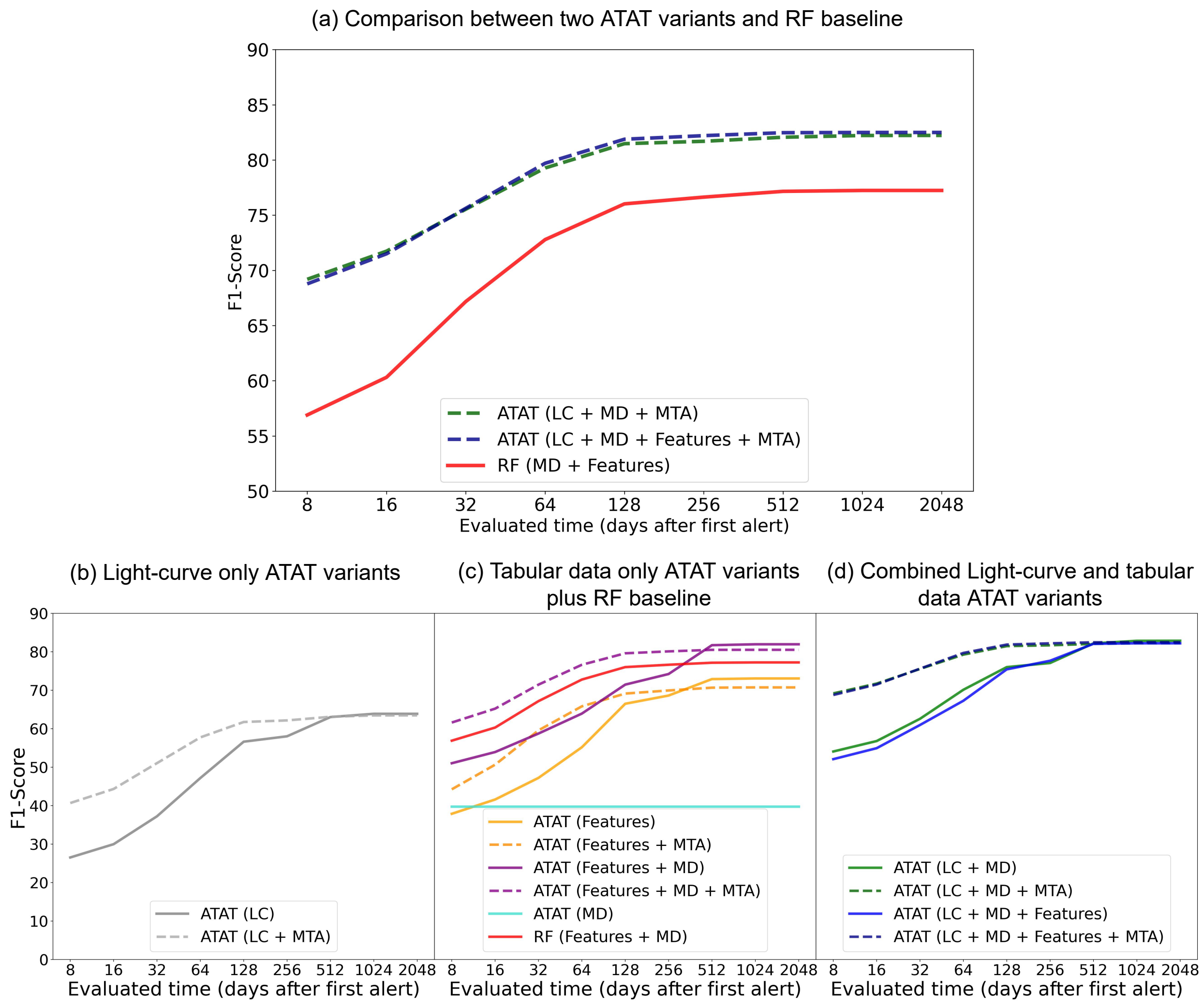


Figure 4.3: F1-score vs time since first alert for a selection of models. We show the better performing ATAT variants and the RF-based baseline (a), the light-curve only ATAT variants (b), the tabular data only ATAT variants (c), and the combined light-curve and tabular data ATAT variants (d). LC/MD/Features refers to models that are optimized using the light-curve, metadata and feature information, respectively. Models can use more than one information source, e.g., LC + MD + Features. Dotted lines refer to models that are optimized with MTA (see Section 4.3.5).

To explore the influence of the number and type of data sources on the classification performance, eleven ATAT variants are compared in Figures 4.3 (b), (c) and (d). In all figures the dashed lines correspond to the cases where the MTA strategy is used.

Figure 4.3 (b) shows the performance of the ATAT variants using only the light-curve as input source, with and without the MTA strategy. The MTA strategy significantly improves the classifier performance at early times, saturating at about 128 days and after that having only marginal increments. This could be related with the majority of the classes in the dataset being transients and with the absence of longer timescales variable objects (e.g., Miras and other LPVs).

Figure 4.3 (c) shows the performance of the ATAT variants trained using only tabular data information. This includes metadata from the first alert and features that are a function of the available light-curve data, where a strategy similar to MTA can also be applied. Note that the feature-based model (yellow line) outperforms the LC-based model (grey line in Figure 4.3 b), although they may not be fully comparable (see Section 4.4). We can also observe that the performance of the feature-based model increases considerably when metadata is incorporated (purple line), and even outperforms the RF-based baseline (red line) when considering MTA. The MTA strategy applied to feature computation has a positive effect in early classification performance in all cases.

Figure 4.3 (d) shows the performance of four ATAT variants trained with both the light-curves and the metadata, with and without features, and with and without the MTA strategy. This figure suggests a low synergy between the light-curve and feature data and that ATAT can extract the most relevant class information using only the light-curve and metadata. Moreover, a comparison with Figure 4.3 (b) suggests a high synergy between the light-curve and metadata, where adding metadata yields a performance improvement between 20% and 30% depending on the length of light-curves. The ATAT variant that uses all information sources (blue solid-line) without the MTA strategy has a worse performance for light-curves shorter than 128 days than the model that uses only light-curve and metadata information (green solid-line). When the MTA strategy is applied, the model that uses all the data sources is only marginally superior.

Summarizing, models combining light-curves and metadata information yield the highest performance (highest synergy). The addition of features improves the classification marginally if using the MTA strategy, but this does not appear to be significant. Additionally, applying the MTA strategy is always beneficial for early classification in the models using light-curves and/or feature data. The two ATAT variants that use light-curves plus metadata and MTA shown in Figure 4.3 (a) were put into production within the pipeline that processes the ELAsTiCC stream. In the future, we will explore class-weighting and/or additional data-augmentation strategies to improve the performance of ATAT in the data-scarce classes.

4.5.3 Confusion matrices

Figure 4.4 shows the confusion matrices of: (a) the ATAT variant that uses LC, metadata and MTA; (b) the ATAT variant that uses LC, features, metadata, and MTA; and (c) the RF-based baseline. These results where obtained by evaluating the light-curves at their maximum length (2048). The ATAT (LC + MD + Features + MTA) model outperformed the RF in 15 out of 20 classes in the dataset. In particular the ATAT performs better in all the SNe subclasses, namely: Iax, 91bg, Ia, Ib/c, II, SLSN, PISN and SN-like/Other. This is specially noticeable for types Ib/c and II where the difference in recall is 37% and 20%, respectively. In the case of transient types, besides the aforementioned SNe subclasses, noticeable differences

Classnames	ATAT (LC + MD)			ATAT (LC + MD + Features)			RF (MD + Features)		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
CART	0.7440	0.4040	0.5200	**0.7520**	0.3980	0.5200	0.5553	**0.5526**	**0.5539**
Iax	**0.6080**	**0.6680**	**0.6360**	0.6040	0.6520	0.6220	0.5434	0.5108	0.5265
91bg	**0.8980**	**0.9220**	**0.9100**	0.8960	0.9000	0.8980	0.7107	0.8752	0.7844
Ia	**0.7520**	**0.8200**	**0.7840**	0.7480	0.8060	0.7760	0.5674	0.7538	0.6474
Ib/c	0.5080	0.6740	**0.5760**	0.4780	**0.6760**	0.5600	**0.5799**	0.3114	0.4052
II	**0.7520**	0.5220	0.6140	0.7480	**0.5440**	**0.6300**	0.6381	0.3450	0.4478
SN-like/Other	**0.5900**	**0.7500**	**0.6600**	**0.5900**	0.7260	0.6500	0.5430	0.4900	0.5151
SLSN	**0.9420**	0.9180	**0.9280**	0.9340	**0.9220**	**0.9280**	0.8865	0.8846	0.8855
PISN	0.9140	**0.9700**	**0.9420**	**0.9180**	0.9660	0.9400	0.8341	0.9578	0.8917
TDE	0.7360	0.9180	0.8140	0.7420	**0.9220**	**0.8200**	**0.8160**	0.7370	0.7744
ILOT	0.9000	0.8300	0.8660	**0.9060**	0.8540	**0.8800**	0.7469	**0.9178**	0.8235
KN	**0.9700**	0.7040	0.8140	**0.9700**	0.6900	0.8040	0.8173	**0.9328**	**0.8712**
M-dwarf Flare	0.9900	0.6840	0.8120	**0.9940**	0.6720	0.8000	0.9529	**0.7890**	**0.8632**
uLens	0.8580	0.9500	0.9020	0.8520	**0.9620**	**0.9040**	**0.9475**	0.8274	0.8834
Dwarf Novae	**0.8820**	0.8740	0.8780	0.8780	**0.8760**	**0.8800**	0.7769	0.8272	0.8012
AGN	**1.0000**	**1.0000**	**1.0000**	0.9980	**1.0000**	**1.0000**	0.9457	**1.0000**	0.9721
Delta Scuti	0.9280	0.9580	0.9440	**0.9680**	**0.9980**	**0.9820**	0.9051	0.9882	0.9448
RR Lyrae	0.9520	0.9540	0.9560	**0.9740**	**0.9900**	**0.9800**	0.9209	0.9836	0.9512
Cepheid	0.9560	0.9820	0.9700	**0.9800**	**0.9980**	**0.9900**	0.9213	0.9878	0.9533
EB	0.8720	**0.9900**	0.9240	0.8840	**0.9900**	0.9340	**0.9282**	0.9752	**0.9511**
Macro avg	0.8376	0.8246	0.8225	**0.8407**	**0.8271**	**0.8249**	0.7769	0.7823	0.7724

Table 4.1: Classification precision, recall and F1-score per class and macro average of the models put into production. We include the two best-performing ATAT variants (with MTA) and the RF-based baseline.

between the models arise. For example the RF-based baseline outperforms ATAT by 24% and 15% in the case of KN and CART, respectively. The former model confused these classes mainly with SNe types Ib/c and Iax. The baseline is also 7% better at detecting ILOT. ATAT confuses this class mainly with TDEs, whereas the baseline does not present such confusion. On the other hand, ATAT outperforms the baseline by 18% and 6% in the case of TDE and uLens, respectively. Cataclysmic types also present interesting differences between models. For example the ATAT outperforms the baseline by 7% for the Dwarf-novae class. On the other hand, the RF-based baseline outperforms the ATAT model by 16% in the case of M-dwarf flares. The latter model has 11% confusion between this class and the EB periodic subtype, whereas the baseline does not present such confusion. Finally, the confusion matrices show that both models achieve almost perfect detection for periodic variable star classes and for the stochastic AGN class, with the proposed model being marginally superior than the baseline. It is worth noting that in three of the four classes where the RF-based baseline outperforms ATAT, namely KN, CART and M-dwarf are also the ones with fewer examples in the dataset (see Figure 4.2). This suggests that the RF is more efficient for highly class-imbalanced datasets than the Transformer-based approach. Additional data-augmentation strategies may be required to improve the performance of ATAT in these data-scarce classes. The ATAT variant that does not use features (LC + MD + MTA) shows a similar performance than the ATAT variant that uses them (LC + MD + Features + MTA), except for periodic classes where the feature-based variant is marginally better.

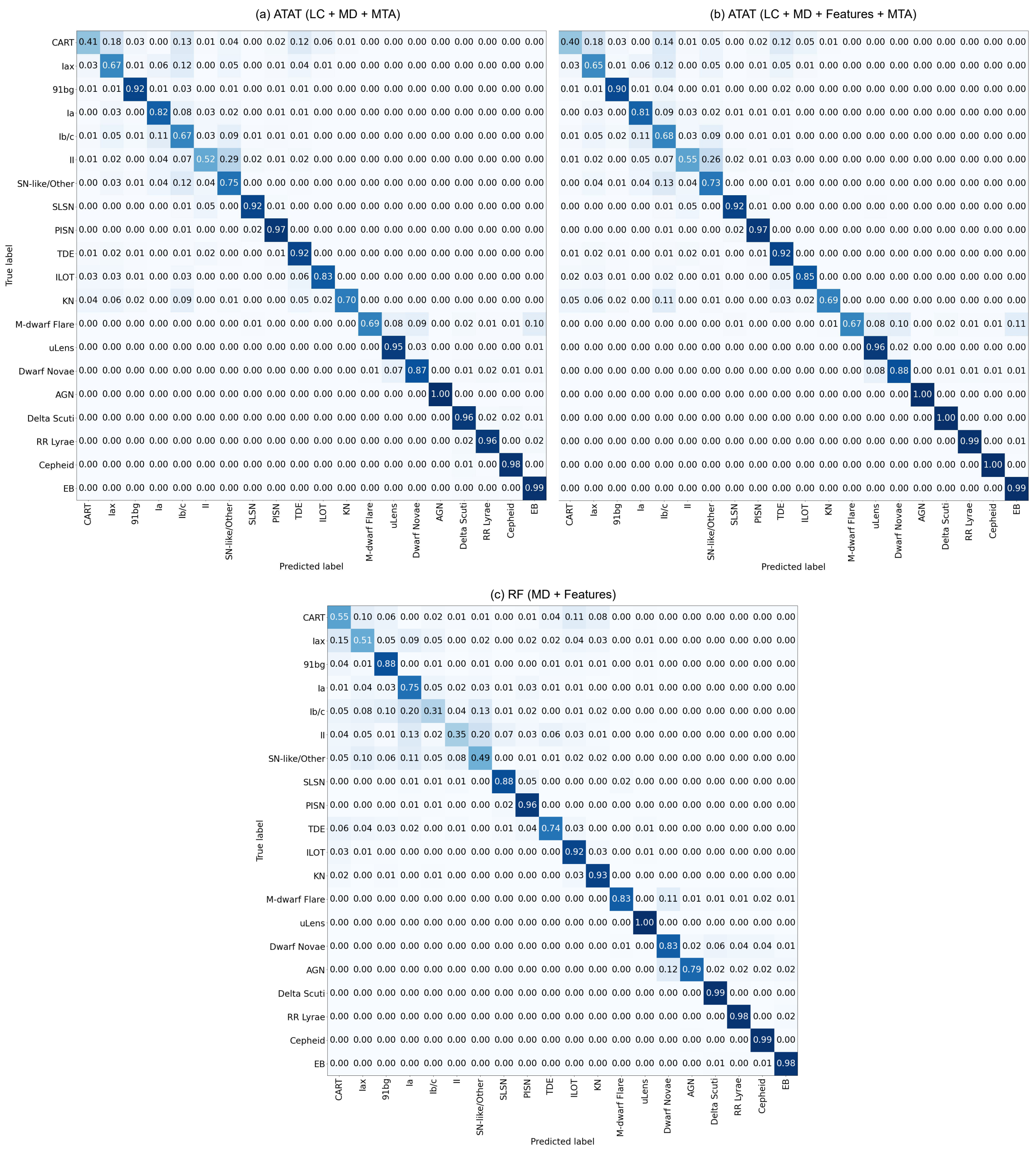


Figure 4.4: Confusion matrix of two ATAT variants, and the Random Forest (RF) baseline. ATAT (LC + MD + MTA) has an F1-Score of 82.25%, ATAT (LC + MD + Features + MTA) has an F1-Score of 82.49%, and RF (MD + Features) has an F1-Score of 77.24%.

4.5.4 Computational time performance

Table 4.2 shows the average computational time to predict the class of a single light-curve[9] with the selected ATAT variants and the RF-based baseline. The table also shows the average time per light-curve to compute the complete set of engineered features. Note that only the RF-based baseline and ATAT (LC + MD + Features + MTA) require features to be computed. From Table 4.2, we observe that the computational time required to perform inference with any of the models is negligible in comparison with the time required to compute features. This means that in total, the ATAT (LC + MD + MTA) variant, is several orders of magnitude faster than the RF-based baseline and feature-based ATAT variants. This sets the LC + MD + MTA variant as a very interesting trade-off, reducing computational time in 99.75% with only a 0.3% decrease in F1-score. We note, however, that some classes are more affected when features are excluded, e.g., periodic variables. As future work we plan to explore which subset of features are more synergistic with the ATAT (LC + MD + MTA) variant. The selection of the best trade-off may also need to be reevaluated as future surveys such as LSST are expected to incorporate some features (e.g., period) in the alert stream.

Inference step	Average time [s]
ATAT (LC + MD + MTA)	$4.75 \cdot 10^{-4}$
ATAT (LC + MD + Features + MTA)	$8.29 \cdot 10^{-4}$
RF (MD + Features)	$2.02 \cdot 10^{-4}$
Feature computation	$1.88 \cdot 10^{-1}$

Table 4.2: Average computational time per light-curve in seconds required to perform the inference step for selected classification models.

4.6 Conclusion

We introduced ATAT, a novel Deep Learning Transformer model that combines time series and tabular data information. The proposed model was developed for the ELAsTiCC challenge that simulates an LSST-like stream, with the objective of testing end-to-end alert stream pipelines. We were able to evaluate both classification and infrastructure performance metrics in the training set provided by ELAsTiCC. Our model was put into production within the ALeRCE broker in preparation for the real-time classification of the LSST alert stream.

Our results show that, using the ELAsTiCC dataset, ATAT outperforms a Balanced Hierarchical RF model similar to the current ALeRCE's light-curve classifier. This RF obtains a macro precision/recall/f1-score of 0.777/0.782/0.772, while ATAT achieves 0.841/0.827/0.825 when using light-curves, metadata and features calculated over the light-curves. Furthermore, if only the light-curves and metadata are considered for ATAT, we achieved values of 0.838/0.825/0.823 for the previous metrics, and about 400 times faster inference times than with the RF. Importantly, our work suggests that it is possible to classify light-curves excluding human-engineered features with no significant loss in performance, and highlights

[9] Averages are estimated using the whole dataset and full-length light-curves.

the importance of including metadata information such as the properties of the host galaxy (e.g., [42]).

The metrics presented in this work, e.g., in Table 4.1 or Figure 4.3, are representative of the dataset provided by ELAsTiCC to prepare machine learning models previous to the end-to-end challenge. The ELAsTiCC simulated stream may not be representative of this dataset (e.g., in the distribution of classes), nor of the real LSST alert stream, and this may result in different performance metrics than those reported in this work. In order to tackle these differences, we suggest the application of fine-tuning and domain adaptation techniques.

To the best of our knowledge, this is the first time that a deep neural network outperforms feature-based tree ensembles in a large, complex and multi-class light-curve classification setting, including very different variability classes. Transformer-based models represent a paradigm shift and we believe that more astronomical applications based on these models will be developed. Particularly, ATAT opens the door for more multi-modal applications, e.g., a third branch for stamps in Fig 4.1. Finally, it is important to note that we won the ELAsTiCC challenge, although we disclosed the results for privacy reasons.

Chapter 5

Self-supervision in time series and tabular data

This thesis mainly focuses on classifying astronomical objects using the most information available. We particularly want to use unlabelled and tabular data to improve the classification. In Chapter 3, we investigated the best architectures for classification when using unlabeled data. We studied different encoders and decoders from a representation learning and semi-supervised learning perspective. Thanks to this research, we could demystify what model is the most promising and flexible for MVLIS time series classification. In Chapter 4, using Transformers, we proposed a new model ATAT that could take advantage of MVLIS time series and also tabular data. The proposed model showed impressive performance allowing us to win the ELAsTiCC Challenge.

Although we explored how to use unlabeled data and tabular data for classification separately, it is important to study a unified approach that can take advantage of unlabeled data and tabular data at the same time. Thanks to our studies, in Chapter 3 and Chapter 4, we can narrow our exploration to MHA models and Transformers. Moreover, we demonstrate that using Transformers is possible to overcome the most widespread approach used in astro-informatics for classification, the Random Forest. But, is it possible to use Transformer with self-supervised approaches for representation learning and semi-supervised learning? In Computer Vision, the literature is vast about it [26, 25, 19]. But, in the astro-informatic literature, this problem is not well studied, and only a couple of works have been developed in this direction [36]. In Chapter 3, we showed various architectures that could be trained with the auto-encoding approaches, a self-supervised technique, but this technique can not be applied to features.

Other self-supervised techniques [48, 58, 47] are more promising to tackle tabular data as input, in addition to time series. These self-supervised techniques have not been widely explored in astro-informatics. However, they are based on Transformers, and they are flexible enough to use other data modalities like time series and tabular data as inputs. In what follows, we will study the most widely self-supervised approaches in Computer Vision and Natural Language Processing. We will obtain their main components, and we will unify them. Finally, we will propose guidelines on how to use these self-supervised approaches and what hyperparameters should be explored. Note, the purpose of this chapter is not to provide experiments or results. Rather, it is to unify under one framework two of the most important self-supervised methods in the literature. This framework can be helpful for future research in self-supervision on tabular and time series data.

5.1 Background of self-supervised methods

In Computer Vision, for representation learning and semi-supervised learning MAE [52] is widely used. In contrast, in Natural Language Processing, BERT [31] and BERT's variation are more popular. Although these methods are from different areas, they are incredibly similar and share many components. Both methods try to reconstruct data that is not present in the input. However, they do this in slightly different manners. While MAE uses encoding and decoding steps, BERT uses only an encoding step. Both methods use tokens that, after being processed by the Transformer, estimate the data that was not present in the input. Let's call these tokens as "predictive tokens". In the case of MAE, the Transformer used to predict missing data is the decoder Transformer, while in the case of BERT, it is the encoder Transformer.

Finally, MAE only uses learnable embeddings as predictive tokens with their corresponding positional encoding, while in BERT, the predictive tokens sometimes were learnable embeddings, sometimes were replaced by tokens of other words, and sometimes predictive tokens were simply the same word token without modifications. Additionally, MAE tries to predict not only missing information but also all the input information. BERT predict only predictive tokens and only reconstructs unmodified inputs occasionally. Note, in this work, we will ignore completely sentence A and sentence B pretraining of BERT. For more information about BERT and MAE, please see Chapter 2.

5.2 Methodology

Let's consider the input pair $(x_j,\ p_j)$, with $j \in [1, \ldots, L]$, being L the maximum length of the sequences in the dataset. This input x can be a time series or tabular data ordered as a sequence or any other sequence of interest. The p_j is the position or time (in case of time series) of input x_j. Additionally, let's consider a mask M_j, which tells us if the input x_j is observed or not. This can occur because we are using matrices and the input can have variable length, see Section 4.3.1 of Chapter 4 for more intuition about this practice. In the case of tabular data, we would have a matrix of ones, unless we also consider missing values. Considering the proposed notation, and how MAE and BERT work, we propose a methodology to test the most important components of these models.

We note MAE reconstructs all the input, while BERT only a 15% of it. Using our notation, we can simulate this behavior by sampling a Bernoulli distribution of the mask M, denoted as $M^{\text{pred}} \sim \text{Bernoulli}(M, \eta^{\text{pred}})$, where only the ones in M are sampled with probability η^{pred} and the zeros are maintained as zeros. Using this notation, we would have $M^{\text{pred}}_{\text{MAE}} \sim \text{Bernoulli}(M, 1)$ for MAE and $M^{\text{pred}}_{\text{BERT}} \sim \text{Bernoulli}(M, 0.15)$. We denote the input that data is not predicted as $M^{\text{base}} = \max(1 - M^{\text{pred}}, 0)$. From the data to be predicted, generally, with a probability $1 - \eta^{\text{orig}}$, the predictive tokens replace the input tokens and the rest are unmodified inputs. This generated the masks M^{mod} and M^{orig}. In practice, this is equivalent to $M^{\text{mod}}_{\text{MAE}} \sim \text{Bernoulli}(M^{\text{pred}}_{\text{MAE}}, 0.75)$ and $M^{\text{orig}}_{\text{MAE}} \sim \text{Bernoulli}(M^{\text{pred}}_{\text{MAE}}, 0.25)$ for MAE (when maintaining only 25% of the image), and $M^{\text{mod}}_{\text{BERT}} \sim \text{Bernoulli}(M^{\text{pred}}_{\text{BERT}}, 0.80)$ and $M^{\text{orig}}_{\text{BERT}} \sim \text{Bernoulli}(M^{\text{pred}}_{\text{BERT}}, 0.10)$. Note that in the case of BERT, there is also a 10% of probability that the input token is modified by the token of another word.

The expressed methodology unifies many components of MAE and BERT and can be used directly in time series and tabular data. However, we observed two differences between MAE and BERT that this methodology doesn't consider. First, MAE adds the predictive tokens in the decoder Transformer; let's call this methodology SSL-M (see Fig. 5.1). While BERT adds the predictive tokens at the encoder's input, let's call this methodology SSL-B (see Fig. 5.1). Second, when BERT modifies the predictive tokens, these are modified into learnable embeddings or are changed by the other token words. To study both behaviors, we separate them. We modify the second behavior because we are using time series and tabular data and not words like BERT. In the case of time series, we didn't consider this step. In the case of tabular, we replace the scalar value with a sample from a normal distribution. This is possible because we are using QFT from Chapter 4. Hence all the tabular data have a normal distribution.

The proposed methodology has four important hyperparameters to search η^{pred}, η^{orig}, if the model uses SSL-B or SSL-M, and, in the case of tabular data, if the predictive tokens are replaced by learnable embeddings or noise from a normal distribution. Using this methodology we propose the Algorithm 1, which formalizes our observations and can help to find the best self-supervised model for tabular data and time series for future work.

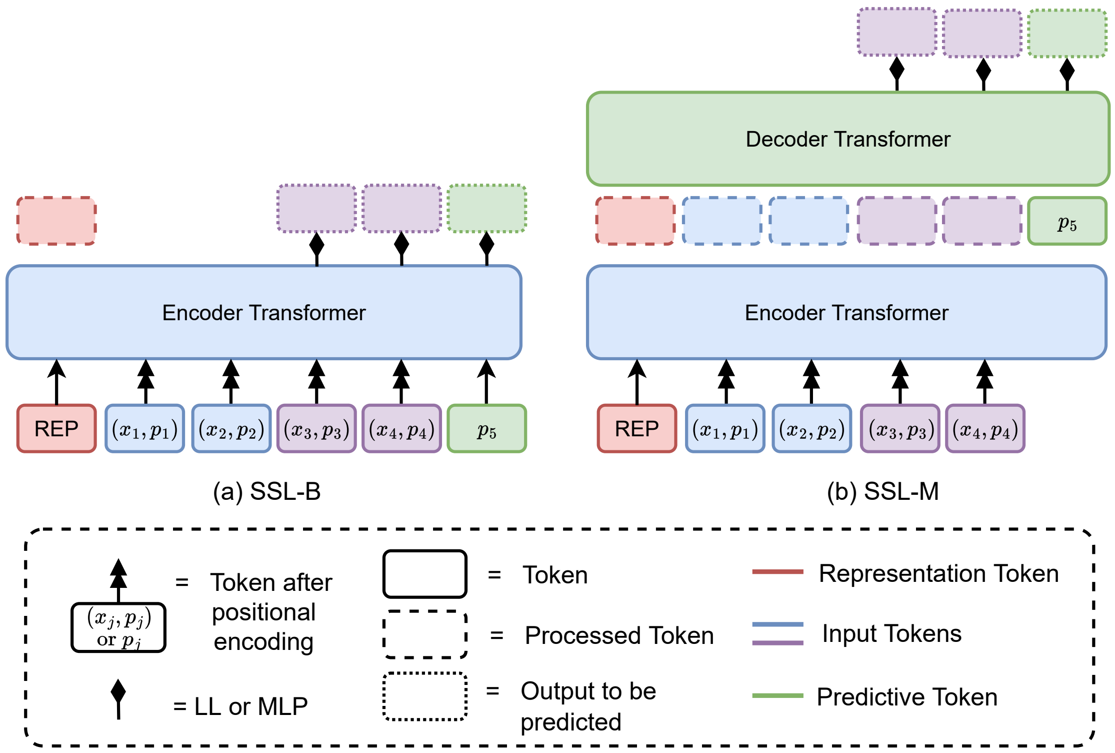


Figure 5.1: Diagram of the types of self-supervision used. We consider two types of self-supervision, Self-B and Self-M. The encoder of Self-B and the decoder of Self-B have the same purpose, process all tokens, so a subset of them (explained later) are reconstructed. However, self-supervision Self-M process non-learnable token with an additional transformer, in this case, the encoder. The red rectangle refers to the representation token. Blue rectangles refer to input tokens that are not used to reconstruct the input. Purple rectangles are input tokens, but they are used to reconstruct the input. Green rectangles are learnable embedding tokens with positional information to reconstruct the data that should be in that position.

Algorithm 1 Algorithm for self-supervision. **Input:** Hyperparameters to explore ($\eta^{\text{pred}}, \eta^{\text{orig}}$), what self-supervision is used (SSL-M or SSL-B), how the predictive token is modified. **Output:** pre-trained model on unlabeled data.

```
1: $train$, $val$, $test \leftarrow$ Set the train, validation, and test set
2: $\eta^{\text{lr}} \leftarrow$ Set learning rates
3: $\eta^{\text{pred}}, \eta^{\text{orig}} \leftarrow$ Set probabilities for masking data
4: SSL $\leftarrow$ Set type of self-supervision, SSL-B or SSL-M
5: $tabular \leftarrow$ Set if using tabular data (True) or time series (False)
6: ptoken $\leftarrow$ Set how the predictive token will be changed, *noise* (only for tabular data) or *embedding*
7: if $tabular$ then
8:     Mod $\leftarrow$ modulator is a pretrained Quantile Feature Transformer
9: else
10:    Mod $\leftarrow$ modulator is Temporal Modulator
11: end if
12: $T_\theta \leftarrow$ Initialize Transformer with parameters $\theta$
13: $tok \leftarrow$ Initialize learnable embedding
14: if SSL == SSL-M then
15:    $T_{\theta_d} \leftarrow$ Initialize decoder Transformer with parameters $\theta_d$
16:    $T_p \leftarrow T_{\theta_d}$                                   ▷ Create auxiliary variable
17:    $\theta_u \leftarrow (\theta, \theta_d)$                        ▷ Create auxiliary variable for parameters
18: else
19:    $T_p \leftarrow T_\theta$                                       ▷ Create auxiliary variable
20:    $\theta_u \leftarrow \theta$                                    ▷ Create auxiliary variable for parameters
21: end if
22: repeat
23:    $X, P, M \sim train$                                           ▷ Obtain input, position and mask matrices
24:    $M^{\text{pred}} \sim \text{Bernoulli}(M, \eta^{\text{pred}})$                ▷ Obtain mask for prediction
25:    $M^{\text{mod}} \sim \text{Bernoulli}(M^{\text{pred}}, 1 - \eta^{\text{orig}})$   ▷ Obtain mask that modify predictive tokens
26:    $M^{\text{base}} = \max(1 - M^{\text{pred}}, 0)$                 ▷ Obtain mask for the Transformer
27:    $\tilde{X} \leftarrow X$ if SSL == SSL-B else $T_\theta(\text{Mod}(X) \odot M^{\text{base}})$   ▷ Create auxiliary input
28:    $ptoken \leftarrow tok$ if ptoken == *embedding* else sample from $\mathcal{N}(0, I)$ with shape of $X$
29:    $\tilde{X} = \text{Mod}(\tilde{X}, P) \odot M^{\text{base}} + \text{Mod}(ptoken, P) \odot M^{\text{mod}}$   ▷ Modify input with predictive tokens
30:    $\hat{X} = T_p(\tilde{X})$                                      ▷ Process auxiliary input with Transformer
31:    $\theta_u \leftarrow \theta_u + \eta^{\text{lr}} \nabla_{\theta_u} \text{MSE}(X \odot M^{\text{pred}}, \hat{X} \odot M^{\text{pred}})$   ▷ Gradient update on network
32: until convergence
```

Chapter 6

Conclusions

In this work, we tackle the challenging problem of classifying astronomical objects, which has various practical difficulties. We consider two of their most fundamental challenges: 1) There exists a massive amount of unlabelled data that could be useful for classification, and that is currently being wasted, and 2) the astronomical objects have various information modalities, they include multivariate variable length and irregular sampling (MVLIS) time series and also tabular data that could be useful for classification. At the same time, tabular data have complex distribution, usually with missing or nan values, and, moreover, specialized architectures are needed to process MVLIS time series.

To deal with the MVLIS time series, we did extensive experiments over encoders and decoders that could manage these time series (Chapter 3). We explored various architectures with a focus on representation learning, which is the core area in Machine Learning that studies models' potential when using unlabeled data. We studied various representation learning metrics and established their connection with semi-supervised learning (Table 3.8). Exploring representation and semi-supervised learning can be useful to take advantage of all the unlabeled data in practice. Additionally, we explore how estimating the aleatoric variance could help representation learning. Thanks to this study, we conclude what models are more robust and flexible. Concluding about what models perform best allows accept or reject Hypothesis 1 which states that representation learning metrics should be dependent on the encoder/decoder architecture used.

We found that decoder architectures are not as relevant for representation learning as in Computer Vision [27]. Although decoders don't significantly affect the representations obtained, there is a big difference in reconstruction error, which could impact forecasting models (see Table 3.5, 3.4). From our explorations, we suggest using a simple MLP for decoding instead of more sophisticated techniques like LSTM or MHA (Chapter 3). Considering these results, we conclude that Hypothesis 1 is not true when measuring representation learning metrics for decoder architectures. However, for reconstruction metrics, the decoder architecture is relevant.

Upon observing the significant impact of encoding architectures, we discovered that LSTMs faced difficulties when dealing with more complex datasets. Surprisingly, interpolation techniques such as InterPred displayed impressive results. However, for encoding MVLIS time series, we opted for MHA due to its consistent performance across various datasets (refer to Table 3.6, 3.7, and 3.8, as well as Sections 3.4.3 and 3.4.4). Additionally, MHA offers greater flexibility as an architecture. These findings confirm Hypothesis 1, indicating that the choice of architecture plays a crucial role in representation learning metrics, primarily observed during the encoding procedure, where high variability exists.

To validate Hypothesis 2, we selected the ELASTiCC dataset, which encompasses both tabular and time series data, offering a more complex and comprehensive evaluation. To effectively handle the substantial volume of data, we employed Transformers with MHA as the primary module. Transformers have established themselves as highly flexible architectures capable of accommodating diverse data modalities, including images, speech, and text. Notably, they have also demonstrated scalability to handle massive datasets, as exemplified in prior studies [11]. In our research, we leveraged Transformers, specifically designed for the two most prevalent modalities in astronomy and astro-informatics, namely, time series and tabular data. By employing distinct modulation techniques for these data types, we effectively utilized the same Transformer architecture to process our astronomical objects. This methodology led to outstanding results, surpassing the conventional RF models, which have been widely adopted for classification purposes in the astronomy literature.

The proposed architecture exhibits remarkable flexibility, enabling us to assess the individual and combined performance of each modality. Fig. 4.3 provides a comparative analysis of the model's performance when utilizing only light-curves, only tabular data, and when leveraging both modalities jointly, depicted in Fig. 4.3(a), Fig. 4.3(b), and Fig. 4.3(c), respectively. Notably, the models that leveraged both modalities outperformed the dedicated time series and tabular classifiers by approximately 20% and 2%, respectively, thereby validating Hypothesis 2. Furthermore, our results demonstrate that hand-crafted features, traditionally employed in astronomy literature for many years, may be unnecessary. The ATAT Transformer we propose exhibited processing times two orders of magnitude faster than the computation of hand-crafted features. This significant improvement has implications for the infrastructure of astronomical object classification brokers. Notably, we applied ATAT in the international challenge ELAsTiCC and emerged as the winners of the classification challenge, further substantiating the effectiveness and practical applicability of our proposed model. While we are unable to present the specific results of the ELAsTiCC challenge in this thesis due to their non-public nature, their successful outcome further validates our model's efficacy.

Finally, in the last Chapter, we give guidelines on how to use Transformers and unlabelled data for classification when considering time series and tabular data. We reviewed the most relevant self-supervised models in Natural Language Processing and Computer Vision literature and selected one model from each area. We identify the essential attributes and hyperparameters of both models and unify them. We provide pseudo-code to test these attributes and to demystify in future work what parameters and hyperparameters are the most important for classification when using unlabelled data with tabular data and time series.

Future work. This work opens the door for several research areas to improve even further the classification of astronomical sources. Our results are promising, and the proposed methodology delivers guidelines on using unlabeled and tabular data for classification. Additionally, we identify three Machine Learning areas that could impact practical applications in classifying astronomical sources. First, although it is extensively explored in ML literature, Domain Adaptation in astro-informatics hasn't been studied to the same extent. Nowadays, there exists a large amount of astronomical data from various surveys that could be helpful to classify new surveys, *e.g.*, the LSST. Second, Active Learning is an ML area that tells the user the most promising data to be labeled. This technique could improve the model classification using the astronomer's time efficiently. Third and final, we think simulated data is fundamental in practical application. Although the before-mentioned techniques allow for improving the models augmenting the labeled data, they will always be restricted by the humans that label the data. Simulated data doesn't have this restriction. For example, in practice, ML models for self-driving cars are trained using simulated data.

Bibliography

[1] Alcock, C, Allsman, R A, Axelrod, T S, Mount Stromlo, Siding Spring Observatories, Australian National University, Weston, ACT 2611, Bennett, D P, Cook, K H, Freeman, K C, Griest, K, Marshall, S L, Peterson, B A, Pratt, M R, Quinn, P J, Rodgers, A W, C W Stubbs, Sutherland, S, Welch, and D L. The macho project lmc variable star inventory. ii. lmc rr lyrae starsemdashpulsational characteristics and indications of a global youth of the lmc. *Astronomical Journal*, 111(3), 3 1996.

[2] Tarek Allam and Jason D. McEwen. Paying attention to astronomical transients: Photometric classification with the time-series transformer, 2021. URL `https://arxiv.org/abs/2105.06178`.

[3] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. *arXiv preprint arXiv:2104.13963*, 2021.

[4] Nicolás Astorga, Pablo Huijse, Pablo A. Estévez, and Francisco Förster. Clustering of astronomical transient candidates using deep variational embedding. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/IJCNN.2018.8489358.

[5] Nicolás Astorga, Pablo Huijse, Pavlos Protopapas, and Pablo Estévez. Mpcc: Matching priors and conditional for clustering. In *European Conference on Computer Vision (ECCV)*, 2020.

[6] Astropy Collaboration, Adrian M. Price-Whelan, Pey Lian Lim, Nicholas Earl, Nathaniel Starkman, Larry Bradley, David L. Shupe, Aarya A. Patil, Lia Corrales, C. E. Brasseur, Maximilian Nöthe, Axel Donath, Erik Tollerud, Brett M. Morris, Adam Ginsburg, Eero Vaher, Benjamin A. Weaver, James Tocknell, William Jamieson, Marten H. van Kerkwijk, Thomas P. Robitaille, Bruce Merry, Matteo Bachetti, H. Moritz Günther, Thomas L. Aldcroft, Jaime A. Alvarado-Montes, Anne M. Archibald, Attila Bódi, Shreyas Bapat, Geert Barentsen, Juanjo Bazán, Manish Biswas, Médéric Boquien, D. J. Burke, Daria Cara, Mihai Cara, Kyle E. Conroy, Simon Conseil, Matthew W. Craig, Robert M. Cross, Kelle L. Cruz, Francesco D'Eugenio, Nadia Dencheva, Hadrien A. R. Devillepoix, Jörg P. Dietrich, Arthur Davis Eigenbrot, Thomas Erben, Leonardo Ferreira, Daniel Foreman-Mackey, Ryan Fox, Nabil Freij, Suyog Garg, Robel Geda, Lauren Glattly, Yash Gondhalekar, Karl D. Gordon, David Grant, Perry Greenfield, Austen M. Groener, Steve Guest, Sebastian Gurovich, Rasmus Handberg, Akeem Hart, Zac

Hatfield-Dodds, Derek Homeier, Griffin Hosseinzadeh, Tim Jenness, Craig K. Jones, Prajwel Joseph, J. Bryce Kalmbach, Emir Karamehmetoglu, Mikołaj Kałuszyński, Michael S. P. Kelley, Nicholas Kern, Wolfgang E. Kerzendorf, Eric W. Koch, Shankar Kulumani, Antony Lee, Chun Ly, Zhiyuan Ma, Conor MacBride, Jakob M. Maljaars, Demitri Muna, N. A. Murphy, Henrik Norman, Richard O'Steen, Kyle A. Oman, Camilla Pacifici, Sergio Pascual, J. Pascual-Granado, Rohit R. Patil, Gabriel I. Perren, Timothy E. Pickering, Tanuj Rastogi, Benjamin R. Roulston, Daniel F. Ryan, Eli S. Rykoff, Jose Sabater, Parikshit Sakurikar, Jesús Salgado, Aniket Sanghi, Nicholas Saunders, Volodymyr Savchenko, Ludwig Schwardt, Michael Seifert-Eckert, Albert Y. Shih, Anany Shrey Jain, Gyanendra Shukla, Jonathan Sick, Chris Simpson, Sudheesh Singanamalla, Leo P. Singer, Jaladh Singhal, Manodeep Sinha, Brigitta M. SipHocz, Lee R. Spitler, David Stansby, Ole Streicher, Jani Šumak, John D. Swinbank, Dan S. Taranu, Nikita Tewary, Grant R. Tremblay, Miguel de Val-Borro, Samuel J. Van Kooten, Zlatan Vasović, Shresth Verma, José Vinícius de Miranda Cardoso, Peter K. G. Williams, Tom J. Wilson, Benjamin Winkel, W. M. Wood-Vasey, Rui Xue, Peter Yoachim, Chen Zhang, Andrea Zonca, and Astropy Project Contributors. The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package. , 935(2):167, August 2022. doi: 10.3847/1538-4357/ac7c74.

[7] Ignacio Becker, Karim Pichara, Márcio Catelan, Pavlos Protopapas, Carlos Aguirre, and Fatemeh Nikzat. Scalable end-to-end recurrent neural network for variable star classification. *Monthly Notices of the Royal Astronomical Society*, 493(2):2981–2995, 2020.

[8] Eric C Bellm, Shrinivas R Kulkarni, Matthew J Graham, Richard Dekany, Roger M Smith, Reed Riddle, Frank J Masci, George Helou, Thomas A Prince, Scott M Adams, et al. The zwicky transient facility: system overview, performance, and first results. *Publications of the Astronomical Society of the Pacific*, 131(995):018002, 2018.

[9] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. URL `http://arxiv.org/abs/1206.5538`. cite arxiv:1206.5538.

[10] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklkeR4KPB`.

[11] Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Big vision. `https://github.com/google-research/big_vision`, 2022.

[12] Kyle Boone. Avocado: Photometric classification of astronomical transients with gaussian process augmentation. *The Astronomical Journal*, 158(6):257, dec 2019. doi: 10.3847/1538-3881/ab5182. URL `https://dx.doi.org/10.3847/1538-3881/ab5182`.

[13] Kirk D Borne. Astroinformatics: data-oriented astronomy research and education.

Earth Science Informatics, 3(1):5–17, 2010.

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1xsqj09Fm`.

[15] Guillermo Cabrera-Vives, Ignacio Reyes, Francisco Förster, Pablo A. Estévez, and Juan-Carlos Maureira. Deep-hits: Rotation invariant convolutional neural network for transient detection. *The Astrophysical Journal*, 836(1):97, feb 2017. doi: 10.3847/1538-4357/836/1/97. URL `https://dx.doi.org/10.3847/1538-4357/836/1/97`.

[16] Guillermo Cabrera-Vives, Zheng Li, Austen Rainer, Dionysis Athanasopoulos, Diego Rodríguez-Mancini, and Francisco Förster. Managing the root causes of "internal api hell": An experience report. In *International Conference on Product-Focused Software Process Improvement*, pages 21–36. Springer, 2022.

[17] Zhaowei Cai, Avinash Ravichandran, Paolo Favaro, Manchen Wang, Davide Modolo, Rahul Bhotika, Zhuowen Tu, and Stefano Soatto. Semi-supervised vision transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=7a2IgJ7V4W`.

[18] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf`.

[19] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[20] Rodrigo Carrasco-Davis, Guillermo Cabrera-Vives, Francisco Förster, Pablo A Estévez, Pablo Huijse, Pavlos Protopapas, Ignacio Reyes, Jorge Martínez-Palomera, and Cristóbal Donoso. Deep learning for image sequence classification of astronomical events. *Publications of the Astronomical Society of the Pacific*, 131(1004):108006, 2019.

[21] Rodrigo Carrasco-Davis, Esteban Reyes, Camilo Valenzuela, Francisco Förster, Pablo A Estévez, Giuliano Pignata, Franz E Bauer, Ignacio Reyes, Paula Sánchez-Sáez, Guillermo Cabrera-Vives, et al. Alert classification for the alerce broker system: The real-time stamp classifier. *The Astronomical Journal*, 162(6):231, 2021.

[22] Tom Charnock and Adam Moss. Deep recurrent neural networks for supernovae classification. *The Astrophysical Journal Letters*, 837(2):L28, 2017.

[23] Chao Chen, Andy Liaw, Leo Breiman, et al. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110(1-12):24, 2004.

[24] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[26] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[27] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016. URL `http://arxiv.org/abs/1611.02731`.

[28] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. 2022.

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL `http://arxiv.org/abs/1412.3555`. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.

[30] Richard Dekany, Roger M. Smith, Reed Riddle, Michael Feeney, Michael Porter, David Hale, Jeffry Zolkower, Justin Belicki, Stephen Kaye, John Henning, and et al. The zwicky transient facility: Observing system. *Publications of the Astronomical Society of the Pacific*, 132(1009):038001, Jan 2020. ISSN 1538-3873. doi: 10.1088/1538-3873/ab4ca2. URL `http://dx.doi.org/10.1088/1538-3873/ab4ca2`.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL `http://arxiv.org/abs/1810.04805`. cite arxiv:1810.04805Comment: 13 pages.

[32] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly notices of the royal astro-*

nomical society, 450(2):1441–1459, 2015.

[33] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *ArXiv*, abs/1907.02544, 2019.

[34] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL `https://openreview.net/forum?id=BJtNZAFgg`.

[35] C Donoso-Oliva, Guillermo Cabrera-Vives, Pavlos Protopapas, Rodrigo Carrasco-Davis, and Pablo A Estévez. The effect of phased recurrent units in the classification of multiple catalogues of astronomical light curves. *Monthly Notices of the Royal Astronomical Society*, 505(4):6069–6084, 2021.

[36] C. Donoso-Oliva, I. Becker, P. Protopapas, G. Cabrera-Vives, Vishnu M., and Harsh Vardhan. Astromer: A transformer-based embedding for the representation of light curves, 2022. URL `https://arxiv.org/abs/2205.01677`.

[37] C. Donoso-Oliva, I. Becker, P. Protopapas, G. Cabrera-Vives, Vishnu M., and Harsh Vardhan. Astromer: A transformer-based embedding for the representation of light curves, 2022. URL `https://arxiv.org/abs/2205.01677`.

[38] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill D. F. Campbell, and Ivor Simpson. Training vaes under structured residuals, 2018. URL `https://arxiv.org/abs/1804.01050`.

[39] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[40] Xuefeng Du, Zhaoning Wang, Mu Cai, and Yixuan Li. Vos: Learning what you don't know by virtual outlier synthesis. *Proceedings of the International Conference on Learning Representations*, 2022.

[41] F Förster, TJ Moriya, JC Maureira, JP Anderson, S Blinnikov, FILOMENA Bufano, G Cabrera-Vives, Alejandro Clocchiatti, T De Jaeger, PA Estévez, et al. The delay of shock breakout due to circumstellar material evident in most type ii supernovae. *Nature Astronomy*, 2(10):808–818, 2018.

[42] Francisco Förster, Alejandra M. Muñoz Arancibia, Ignacio Reyes-Jainaga, Alexander Gagliano, Dylan Britt, Sara Cuellar-Carrillo, Felipe Figueroa-Tapia, Ava Polzin, Yara Yousef, Javier Arredondo, Diego Rodríguez-Mancini, Javier Correa-Orellana, Amelia Bayo, Franz E. Bauer, Márcio Catelan, Guillermo Cabrera-Vives, Raya Dastidar, Pablo A. Estévez, Giuliano Pignata, Lorena Hernández-García, Pablo Huijse, Esteban Reyes, Paula Sánchez-Sáez, Mauricio Ramírez, Daniela Grandón, Jonathan Pineda-García, Francisca Chabour-Barra, and Javier Silva-Farfán. DELIGHT: Deep Learning Identification of Galaxy Hosts of Transients using Multiresolution Images. , 164(5):

195, November 2022. doi: 10.3847/1538-3881/ac912a.

[43] F. Förster, G. Cabrera-Vives, E. Castillo-Navarrete, P. A. Estévez, P. Sánchez-Sáez, J. Arredondo, F. E. Bauer, R. Carrasco-Davis, M. Catelan, F. Elorrieta, S. Eyheramendy, P. Huijse, G. Pignata, E. Reyes, I. Reyes, D. Rodríguez-Mancini, D. Ruz-Mieres, C. Valenzuela, I. Álvarez Maldonado, N. Astorga, J. Borissova, A. Clocchiatti, D. De Cicco, C. Donoso-Oliva, L. Hernández-García, M. J. Graham, A. Jordán, R. Kurtev, A. Mahabal, J. C. Maureira, A. Muñoz-Arancibia, R. Molina-Ferreiro, A. Moya, W. Palma, M. Pérez-Carrasco, P. Protopapas, M. Romero, L. Sabatini-Gacitua, A. Sánchez, J. San Martín, C. Sepúlveda-Cobo, E. Vera, and J. R. Vergara. The automatic learning for the rapid classification of events (alerce) alert broker. *The Astronomical Journal*, 161(5):242, apr 2021. doi: 10.3847/1538-3881/abe9bc. URL `https://dx.doi.org/10.3847/1538-3881/abe9bc`.

[44] Dan Gao, Yan-Xia Zhang, and Yong-Heng Zhao. Random forest algorithm for classification of multiwavelength data. *Research in Astronomy and Astrophysics*, 9(2):220–226, February 2009. doi: 10.1088/1674-4527/9/2/011.

[45] Catalina Gómez, Mauricio Neira, Marcela Hernández Hoyos, Pablo Arbeláez, and Jaime E Forero-Romero. Classifying image sequences of astronomical transients with deep neural networks. *Monthly Notices of the Royal Astronomical Society*, 499(3): 3130–3138, 2020.

[46] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press Cambridge, 2016.

[47] Yu. V. Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *ArXiv*, abs/2106.11959, 2021.

[48] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf`.

[49] Matthew J Graham, SR Kulkarni, Eric C Bellm, Scott M Adams, Cristina Barbarino, Nadejda Blagorodnova, Dennis Bodewits, Bryce Bolin, Patrick R Brady, S Bradley Cenko, et al. The zwicky transient facility: science objectives. *Publications of the Astronomical Society of the Pacific*, 131(1001):078001, 2019.

[50] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=foA_SFQ9zo0`.

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

770–778, June 2016. doi: 10.1109/CVPR.2016.90.

[52] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15979–15988. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01553. URL `https://doi.org/10.1109/CVPR52688.2022.01553`.

[53] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bklr3j0cKX`.

[54] R. Hložek, Kara Ponder, A. Malz, M. Dai, Gowthami Narayan, E. Ishida, Tarek Allam, Anita Bahmanyar, Rahul Biswas, Lluís Galbany, Sourav Jha, David Jones, R. Kessler, Michelle Lochner, A. Mahabal, K. Mandel, J. Martínez-Galarza, J. McEwen, Daniel Muthukrishna, and Christian Setzer. Results of the photometric lsst astronomical time-series classification challenge (plasticc). 12 2020.

[55] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[57] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1558–1567. JMLR.org, 2017. URL `http://dl.acm.org/citation.cfm?id=3305381.3305542`.

[58] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings, 2020.

[59] Pablo Huijse, Pablo A Estevez, Pavlos Protopapas, Jose C Principe, and Pablo Zegers. Computational intelligence challenges and applications on large-scale astronomical time series databases. *IEEE Computational Intelligence Magazine*, 9(3):27–39, 2014.

[60] EEO Ishida, F Mondon, S Sreejith, MV Kornilov, KL Malanchev, MV Pruzhinskaya, AA Volnova, VS Korolev, AA Malancheva, and S Das. Active anomaly detection for time-domain discoveries. *Astronomy and Astrophysics*, 650:A195–A195, 2021.

[61] Željko Ivezić, Steven M Kahn, J Anthony Tyson, Bob Abel, Emily Acosta, Robyn Allsman, David Alonso, Yusra AlSayyad, Scott F Anderson, John Andrew, et al. Lsst: from science drivers to reference design and anticipated data products. *The Astrophysical Journal*, 873(2):111, 2019.

[62] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

[63] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. URL `http://arxiv.org/abs/1812.04948`.

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `http://arxiv.org/abs/1412.6980`. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[65] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[66] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf`.

[67] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf`.

[68] Eiichiro Komatsu, J Dunkley, MR Nolta, CL Bennett, B Gold, G Hinshaw, N Jarosik, D Larson, M Limon, L Page, et al. Five-year wilkinson microwave anisotropy probe* observations: cosmological interpretation. *The Astrophysical Journal Supplement Series*, 180(2):330, 2009.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[70] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

[71] Christian Leistner, Amir Saffari, Jakob Santner, and Horst Bischof. Semi-supervised random forests. In *2009 IEEE 12th International Conference on Computer Vision*, pages 506–513, 2009. doi: 10.1109/ICCV.2009.5459198.

[72] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. *International Conference on Learning Representations (ICLR)*, 2022.

[73] Hanbo Li, Yaqing Wang, Changyou Chen, and Jing Gao. AIM: Adversarial inference by matching priors and conditionals, 2019. URL `https://openreview.net/forum?id=rJx_b3RqY7`.

[74] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL `https://www.science.org/doi/abs/10.1126/science.abq1158`.

[75] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, and Jinqiao Wang. Mst: Masked self-supervised transformer for visual representation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13165–13176. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/6dbbe6abe5f14af882ff977fc3f35501-Paper.pdf`.

[76] Cristóbal Mackenzie, Karim Pichara, and Pavlos Protopapas. Clustering based feature learning on variable stars. *The Astrophysical Journal*, 820, 03 2016. doi: 10.3847/0004-637X/820/2/138.

[77] Ashish Mahabal, Kshiteej Sheth, Fabian Gieseke, Akshay Pai, S George Djorgovski, Andrew J Drake, and Matthew J Graham. Deep-learnt classification of light curves. In *2017 IEEE symposium series on computational intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[78] Thomas Matheson, Carl Stubens, Nicholas Wolf, Chien-Hsiu Lee, Gautham Narayan, Abhijit Saha, Adam Scott, Monika Soraisam, Adam S Bolton, Benjamin Hauger, et al. The antares astronomical time-domain event broker. *The Astronomical Journal*, 161 (3):107, 2021.

[79] Anais Möller, Julien Peloton, Emille E. O. Ishida, Chris Arnault, Etienne Bachelet,

Tristan Blaineau, Dominique Boutigny, Abhishek Chauhan, Emmanuel Gangler, Fabio Hernandez, Julius Hrivnac, Marco Leoni, Nicolas Leroy, Marc Moniez, Sacha Pateyron, Adrien Ramparison, Damien Turpin, Réza Ansari, Jr. Allam, Tarek, Armelle Bajat, Biswajit Biswas, Alexandre Boucaud, Johan Bregeon, Jean-Eric Campagne, Johann Cohen-Tanugi, Alexis Coleiro, Damien Dornic, Dominique Fouchez, Olivier Godet, Philippe Gris, Sergey Karpov, Ada Nebot Gomez-Moran, Jérémy Neveu, Stephane Plaszczynski, Volodymyr Savchenko, and Natalie Webb. FINK, a new generation of broker for the LSST community. , 501(3):3272–3288, March 2021. doi: 10.1093/mnras/staa3602.

[80] Daniel Muthukrishna, Gautham Narayan, Kaisey S Mandel, Rahul Biswas, and Renée Hložek. Rapid: early classification of explosive transients using deep learning. *Publications of the Astronomical Society of the Pacific*, 131(1005):118002, 2019.

[81] Gautham Narayan, Tayeb Zaidi, Monika D. Soraisam, Zhe Wang, Michelle Lochner, Thomas Matheson, Abhijit Saha, Shuo Yang, Zhenge Zhao, John Kececioglu, Carlos Scheidegger, Richard T. Snodgrass, Tim Axelrod, Tim Jenness, Robert S. Maier, Stephen T. Ridgway, Robert L. Seaman, Eric Michael Evans, Navdeep Singh, Clark Taylor, Jackson Toeniskoetter, Eric Welch, Songzhe Zhu, and ANTARES Collaboration. Machine-learning-based Brokers for Real-time Classification of the LSST Alert Stream. , 236(1):9, May 2018. doi: 10.3847/1538-4365/aab781.

[82] Brett Naul, Joshua Bloom, Fernando Perez, and Stéfan van der Walt. A recurrent neural network for classification of unevenly sampled variable stars. *Nature Astronomy*, 2, 02 2018. doi: 10.1038/s41550-017-0321-z.

[83] Mauricio Neira, Catalina Gómez, John F Suárez-Pérez, Diego A Gómez, Juan Pablo Reyes, Marcela Hernández Hoyos, Pablo Arbeláez, and Jaime E Forero-Romero. Mantra: A machine-learning reference light-curve data set for astronomical transient event recognition. *The Astrophysical Journal Supplement Series*, 250(1):11, 2020.

[84] J. Nordin, V. Brinnel, J. van Santen, M. Bulla, U. Feindt, A. Franckowiak, C. Fremling, A. Gal-Yam, M. Giomi, M. Kowalski, A. Mahabal, N. Miranda, L. Rauch, S. Reusch, M. Rigault, S. Schulze, J. Sollerman, R. Stein, O. Yaron, S. van Velzen, and C. Ward. Transient processing and analysis using AMPEL: alert management, photometry, and evaluation of light curves. , 631:A147, November 2019. doi: 10.1051/0004-6361/201935634.

[85] Isadora Nun, Pavlos Protopapas, Brandon Sim, Ming Zhu, Rahul Dave, Nicolas Castro, and Karim Pichara. FATS: Feature Analysis for Time Series. *arXiv e-prints*, art. arXiv:1506.00010, May 2015.

[86] Isadora Nun, Pavlos Protopapas, Brandon Sim, and Wesley Chen. Ensemble learning method for outlier detection and its application to astronomical light curves. *The Astronomical Journal*, 152(3):71, 2016.

[87] James E O'Donnell. Rnu-dependent optical and near-ultraviolet extinction. *The Astrophysical Journal*, 422:158–163, 1994.

[88] Oscar Pimentel, Pablo A. Estévez, and Francisco Forster. Deep attention-based supernovae classification of multi-band light-curves, 2022. URL https://arxiv.org/abs/2201.08482.

[89] Maria V Pruzhinskaya, Konstantin L Malanchev, Matwey V Kornilov, Emille EO Ishida, Florian Mondon, Alina A Volnova, and Vladimir S Korolev. Anomaly detection in the open supernova catalog. *Monthly Notices of the Royal Astronomical Society*, 489 (3):3591–3608, 2019.

[90] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

[91] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

[92] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

[93] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

[94] Danilo Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1278–1286, 2014. URL http://jmlr.org/proceedings/papers/v32/rezende14.html.

[95] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-ODN6SbiUU.

[96] Diego Rodriguez-Mancini, Zheng Li, Camilo Valenzuela, Guillermo Cabrera-Vives, and Francisco Förster. Toward fractal development of data processing-intensive artificial intelligence systems. *IEEE Software*, 39(6):28–34, 2022.

[97] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[98] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal

representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[99] Alexis Sánchez, Guillermo Cabrera, Pablo Huijse, and Francisco Förster. Amortized variational inference for type ia supernova light curves. In *Machine Learning and the Physical Sciences Workshop, 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021. URL `https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_16.pdf`.

[100] P Sánchez-Sáez, Hernan Lira, Luis Martí, Nayat Sanchez-Pi, J Arredondo, FE Bauer, A Bayo, G Cabrera-Vives, C Donoso-Oliva, PA Estévez, et al. Searching for changing-state agns in massive data sets. i. applying deep learning and anomaly-detection techniques to find agns with anomalous variability behaviors. *The Astronomical Journal*, 162(5):206, 2021.

[101] Azusa Sawada, Taiki Miyagawa, Akinori F. Ebihara, Shoji Yachida, and Toshinori Hosoi. Convolutional neural networks for time-dependent classification of variable-length time series, 2022. URL `https://arxiv.org/abs/2207.03718`.

[102] Azusa Sawada, Taiki Miyagawa, Akinori F. Ebihara, Shoji Yachida, and Toshinori Hosoi. Convolutional neural networks for time-dependent classification of variable-length time series, 2022. URL `https://arxiv.org/abs/2207.03718`.

[103] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL `https://www.sciencedirect.com/science/article/pii/S0893608014002135`.

[104] Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1efr3C9Ym`.

[105] Satya Narayan Shukla and Benjamin Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=4c0J6lwQ4_`.

[106] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

[107] K. W. Smith, R. D. Williams, D. R. Young, A. Ibsen, S. J. Smartt, A. Lawrence, D. Morris, S. Voutsinas, and M. Nicholl. Lasair: The Transient Alert Broker for LSST:UK. *Research Notes of the American Astronomical Society*, 3(1):26, January

2019. doi: 10.3847/2515-5172/ab020f.

[108] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[109] Suwon Suh and Seungjin Choi. Gaussian copula variational autoencoders for mixed data. *ArXiv*, abs/1604.04960, 2016.

[110] P. Sánchez-Sáez, I. Reyes, C. Valenzuela, F. Förster, S. Eyheramendy, F. Elorrieta, F. E. Bauer, G. Cabrera-Vives, P. A. Estévez, M. Catelan, G. Pignata, P. Huijse, D. De Cicco, P. Arévalo, R. Carrasco-Davis, J. Abril, R. Kurtev, J. Borissova, J. Arredondo, E. Castillo-Navarrete, D. Rodriguez, D. Ruz-Mieres, A. Moya, L. Sabatini-Gacitúa, C. Sepúlveda-Cobo, and E. Camacho-Iñiguez. Alert classification for the alerce broker system: The light curve classifier. *The Astronomical Journal*, 161(3):141, feb 2021. doi: 10.3847/1538-3881/abd5c1. URL `https://dx.doi.org/10.3847/1538-3881/abd5c1`.

[111] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016. URL `http://arxiv.org/abs/1609.03499`. cite arxiv:1609.03499.

[112] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6527-conditional-image-generation-with-pixelcnn-decoders.pdf`.

[113] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[115] V Ashley Villar. Amortized bayesian inference for supernovae in the era of the vera rubin observatory using normalizing flows. *arXiv preprint arXiv:2211.04480*, 2022.

[116] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[117] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple

fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/wortsman22a.html`.

[118] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[119] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *ArXiv*, abs/2205.01917, 2022.

[120] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017. doi: 10.21629/JSEE.2017.01.18.

[121] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *International Conference on Learning Representations (ICLR)*, 2022.

[122] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJJLHbb0-`.