PELICAN: deeP architecturE for the LIght Curve ANalysis

Johanna Itam-Pasquet, Jérôme Pasquet, Marc Chaumont, Dominique Fouchez

A&A 627, A21 (2019)
https://doi.org/10.1051/0004-6361/201834473
© J. Pasquet et al. 2019

Astronomy & Astrophysics

PELICAN: deeP architecturE for the LIght Curve ANalysis

Johanna Pasquet[1], Jérôme Pasquet[2,3,4], Marc Chaumont[5], and Dominique Fouchez[1]

[1] Aix Marseille Univ., CNRS/IN2P3, CPPM, Marseille, France
e-mail: pasquet@cppm.in2p3.fr
[2] AMIS, Université Paul Valéry, Montpellier, France
[3] TETIS, Univ. Montpellier, AgroParisTech, Cirad, CNRS, Irstea, Montpellier, France
[4] Aix-Marseille Université, CNRS, ENSAM, Université De Toulon, LIS UMR 7020, France
[5] LIRMM, Univ. Nîmes, CNRS, Univ., Nîmes, France

Received 19 October 2018 / Accepted 19 March 2019

ABSTRACT

We developed a deeP architecturE for the LIght Curve ANalysis (PELICAN) for the characterization and the classification of supernovae light curves. It takes light curves as input, without any additional features. PELICAN can deal with the sparsity and the irregular sampling of light curves. It is designed to remove the problem of non-representativeness between the training and test databases coming from the limitations of the spectroscopic follow-up. We applied our methodology on different supernovae light curve databases. First, we tested PELICAN on the Supernova Photometric Classification Challenge for which we obtained the best performance ever achieved with a non-representative training database, by reaching an accuracy of 0.811. Then we tested PELICAN on simulated light curves of the LSST Deep Fields for which PELICAN is able to detect 87.4% of supernovae Ia with a precision higher than 98%, by considering a non-representative training database of 2k light curves. PELICAN can be trained on light curves of LSST Deep Fields to classify light curves of the LSST main survey, which have a lower sampling rate and are more noisy. In this scenario, it reaches an accuracy of 96.5% with a training database of 2k light curves of the Deep Fields. This constitutes a pivotal result as type Ia supernovae candidates from the main survey might then be used to increase the statistics without additional spectroscopic follow-up. Finally we tested PELICAN on real data from the Sloan Digital Sky Survey. PELICAN reaches an accuracy of 86.8% with a training database composed of simulated data and a fraction of 10% of real data. The ability of PELICAN to deal with the different causes of non-representativeness between the training and test databases, and its robustness against survey properties and observational conditions, put it in the forefront of light curve classification tools for the LSST era.

Key words. methods: data analysis – techniques: photometric – supernovae: general

1. Introduction

A major challenge in cosmology is to understand the observed acceleration of the expansion of the universe. A direct and very powerful method to measure this acceleration is to use a class of objects, called standard candles due to their constant intrinsic brightness, which are used to measure luminosity distances. Type Ia supernovae (SNe Ia), a violent endpoint of stellar evolution, are a very good example of such a class of objects as they are considered as standardizable candles. The acceleration of the expansion of the universe was derived from observations of several tens of such supernovae at low and high redshift (Perlmutter et al. 1999; Riess et al. 1998). Then, several dedicated SNe Ia surveys have together measured light curves for over a thousand SNe Ia, confirming the evidence for acceleration expansion (e.g., Betoule et al. 2014; Scolnic et al. 2018).

The future Large Survey Synoptic Telescope (LSST; LSST Science Collaboration 2009) will improve on past surveys by observing a much higher number of supernovae. By increasing statistics by at least an order of magnitude and controlling systematic errors, it will be possible to pave the way for advances in precision cosmology with supernovae.

A key element for such analysis is the identification of type Ia supernovae. However, the spectroscopic follow-up will be limited and LSST will discover more supernovae than can be spectroscopically confirmed. Therefore an effective automatic classification tool, based on photometric information, has to be developed to distinguish between the different types of supernovae with a minimum contamination rate to avoid bias in the cosmology study. This issue has been raised before and led to the launch of the Supernova Photometric Classification Challenge in 2010 (SPCC; Kessler et al. 2010a) to the astrophysical community. Several classification algorithms were proposed with different techniques resulting in similar performance without resolving the problem of non-representativeness between the training and test databases. Nonetheless, the method developed by Sako et al. (2008, 2018) based on template fitting shows the highest average figure of merit on a representative training database, with an efficiency of 0.96 and an SN Ia purity of 0.79.

Since then, several machine learning methods have been applied to classify supernovae light curves (e.g., Richards et al. 2012; Ishida & de Souza 2013; Karpenka et al. 2013; Varughese et al. 2015; Möller et al. 2016; Lochner et al. 2016; Dai et al. 2018). They show interesting results when they are applied on a representative training dataset, but the performance dramatically decreases when the learning stage is made on a non-representative training subset, which represents, however, the real scenario.

We propose to explore in this paper a new branch of machine learning called deep learning, proved to be very efficient for image and time series classification (e.g., Szegedy et al. 2015; He et al. 2016; Schmidhuber et al. 2005). One of the main

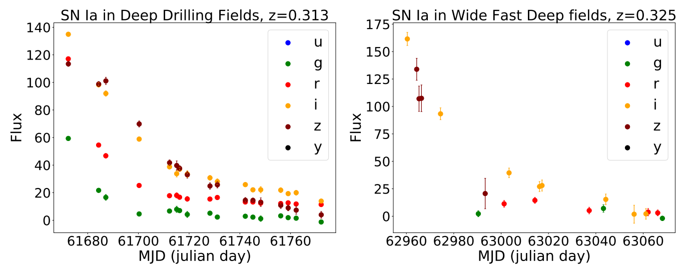


Fig. 1. Example of two light curves of type Ia supernovae observed for a high LSST cadence (Deep Drilling Fields), on the *left* and a low LSST cadence (Wide Deep Fast), on the *right*, at two similar redshifts.

differences of this to the classical machine learning methods is that the raw data are directly transmitted to the algorithm that extracts by itself the best feature representation for a given problem. In the field of astrophysics, deep learning methods have shown better results than the current method applied to images for the classification of galaxy morphologies (Domínguez Sánchez et al. 2018), the classification of transients (du Buisson et al. 2015; Gieseke et al. 2017), and the estimation of photometric redshifts (Pasquet et al. 2019) to name a few. This method has also shown an impressive performance for the classification of light curves (Mahabal et al. 2017; Pasquet-Itam & Pasquet 2018) and especially the classification of supernovae (Charnock & Moss 2017; Brunel et al. 2019).

In this work we develop a complex deep learning architecture to classify light curves. We apply our study to the classification of light curves of supernovae. Unlike the other studies, our method overcomes the problem of non-representativeness between the training and the test databases, while considering a small training database. We apply our method to the SPCC challenge, then on LSST simulated data including a biased and small training database. We also validate our method on real data from the Sloan Digital Sky Survey (SDSS) data. The paper is organized as follows. In Sect. 2, we explain the different issues for the classification of light curves. In Sect. 3, we introduce deep learning concepts that we used and developed in this work. In Sect. 4, we present our architecture named PELICAN (deeP architecturE for the LIght Curve ANalysis). In Sect. 5, we describe the different datasets used in this study. In Sect. 6 we present the experimental protocol that we adapted to make PELICAN robust against the differences of sampling and noise. In Sect. 7 we present our results for different databases. In Sect. 8 we analyze the behavior of PELICAN with respect to a number of light curve properties and observational conditions. Finally we conclude and consider future avenues of research in Sect. 9.

2. Light curve classification issues

Light curves are fundamental signals to measure the variability of astrophysical objects. They represent the flux of an object over time in different photometric bands (e.g., ugriz system). Due to the observational strategy and conditions, light curves have an irregular sampling, often sparse. Therefore a sampling with two different observational cadences presents several differences. Figure 1 shows, as an example, two simulated light curves with two different cadence models (see Sect. 5.3). Compared to an image, such light curves have incomplete and non-continuous information, thus imposing dedicated training algorithms.

The other issue is the non-representativeness between the training and the test databases. As the spectroscopic follow-up used to label the light curves is limited, the coverage of the training database in brightness, redshift, and number is different from

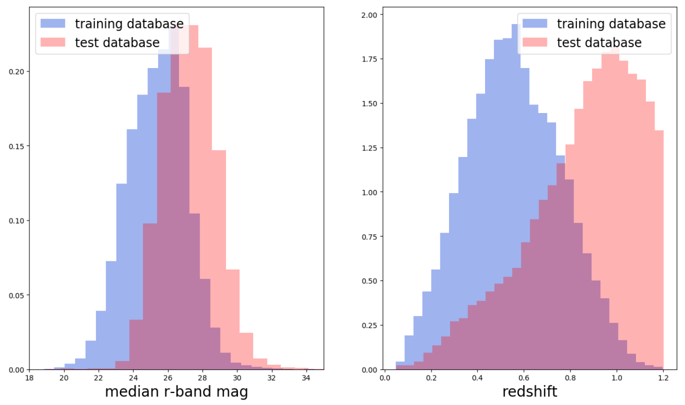


Fig. 2. Distributions of LSST simulated data of the median *r*-band magnitude (*left*) and the simulated redshift (*right*) for the training dataset in blue and the test dataset in red. The mismatch is clearly visible as there is a significant shift between the two distributions.

the test database as shown in Fig. 2. Moreover the absolute magnitude of SNe Ia is correlated with two quantities. First, brighter SNe Ia have wider, slower declining light curves. This variability can be described as a timescale stretch of the light curve (Phillips 1993). In addition brighter SNe Ia are bluer and a color correction has to be applied to standardize them (van den Bergh 1995; Tripp 1998). So due to these correlations, the lower stretch and redder supernovae are fainter and tend to have small recovery efficiency (Malmquist bias) and so are under-represented in the training database, which is limited in brightness. The non-representativeness of the databases, which is a problem of mismatch, is critical for the machine learning process. In general, machine learning methods require a sufficiently large number of training data in order to correctly classify, so the small size of the training database involves another difficulty.

To provide a solution for each of these issues, we have designed a specific architecture. First, light curves from the test database are trained with a non-supervised model without using the knowledge of labels. This allows us to reduce the mismatch between the training and the test databases and provides a solution to the small training dataset, by extracting features from the larger test database. To reduce again the problem of non-representativeness, we performed a second training step to minimize the distances in the feature representation space between bright and faint supernovae of the same label and to maximize the distances of supernovae with different labels. Finally we integrated a regularization term into the training to adapt the model to the sparsity of data. The resulting deep architecture, dedicated to the characterization and classification of light curves, is presented in Sect. 4.

3. Deep learning model

In this section, we present the main deep learning concepts that were used to build our network architecture. Namely the convolution layer network, which describes the basics, the autoencoder, which is a non-supervised module where we detail the notion of loss function, and finally the contrastive approach, where an adapted loss function is defined to improve the clustering of entries with the same label.

3.1. Convolutional neural network

The convolutional neural network (CNN) is a special type of multilayered neural network that is made up of neurons that have learnable weights and biases. The architecture of a CNN is designed to take advantage of the 2D structure of an input image. It takes as input a $h \times w \times c$ image where h is the height, w is the width of the image, and c is the number of channels. As a light curve is a 1D signal, we transform it into a 2D "light curve image" (LCI) as we did in Pasquet-Itam & Pasquet (2018). The width of the LCI is the temporal axis (expressed in days) and the height is the number of photometric bands. For example, if we consider a light curve measured in ugriz bands over 200 days, the corresponding LCI is a 2D array of dimension (5×200) pixels. By treating the band as a second spatial dimension instead of as a channel, we can add the information of the color deeper in the network or even at several levels as we did in Pasquet-Itam & Pasquet (2018). As a light curve is not a continuous signal, the corresponding array is composed of many blank cells that we fill with zero values. A harmful consequence is the overfitting of the position of missing data, which could dramatically degrade the performance. In this case, the model learns the exact position of missing data of the training light curves and is not able to generalize to the unseen test data. To prevent this overfitting and make the model invariant against the position of missing data, we have integrated several techniques that are explained in Sect. 6.

In this work, we developed a CNN instead of a recurrent neural network (RNN), which could, however, seem more suitable for the classification of time series, as the treatment of missing data needs to be considered in a different way. The missing data can be interpolated (e.g., Charnock & Moss 2017), but this preprocessing can add a bias in the classification task. Moreover the interpolation of light curves depends on the observational strategy. In the context of LSST, as there will be two different cadence models, the interpolation of data is not trivial and a common interpolation could degrade the performance of the classification.

3.1.1. Convolution layers

In a convolution layer, each neuron applies a convolution operation to the input data using a 2D map of weights used as a kernel. Then resulting convolved images are summed, a bias is added, and a non-linearity function is applied to form a new image called a feature map. In the first convolution layer, the convolution operation is realized between the input LCI and the set of convolution kernels to form feature maps that are then convolved with convolution kernels in the next convolution layer. For the non-linearity function, we mainly use the most commonly used activation function: the ReLU (Rectified Linear Unit, Nair & Hinton 2010) defined by $f(x) = \max(x, 0)$. The weights of the kernels are updated during the training by backpropagation process.

3.1.2. Pooling layers

The network can be composed of pooling layers, which quantify the information while reducing the data volume. The two most used methods consist in selecting only the maximum or the average value of the data in a local region.

3.1.3. Fully connected layers

Finally, fully connected layers are composed of neurons that are connected to every neuron of the previous layer and perform the classification process with features from previous convolution layers. More details on CNN models can be found in Pasquet-Itam & Pasquet (2018), Pasquet et al. (2019).

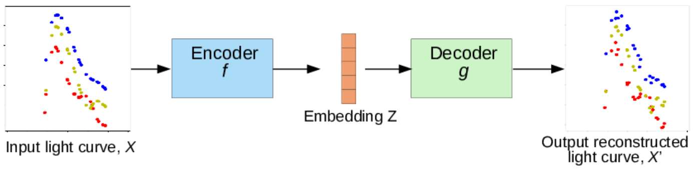


Fig. 3. Schema of the autoencoder process.

3.2. Autoencoder

To benefit from the information of the light curves of the test database and so reduce the mismatch between the training and test databases, we have adapted a non-supervised autoencoder. An autoencoder is an unsupervised learning algorithm that tries to learn an approximation to the identity function such as output should mimic the input. As a consequence the internal layers exhibit a good representation of the input data. The input $X \in \mathbb{R}^D$, with $D = h \times w$, is transformed into an embedding $Z \in \mathbb{R}^K$, often such as $K \ll D$. The mapping from X to Z is made by the encoder, denoted f, which can perform a dimension reduction to finally get a good representation of the data in a compressed format. The reconstruction of the original signal, $X' \in \mathbb{R}^D$, is obtained by the decoder, denoted g, which uses the compressed embedding representation Z (see Fig. 3). The objective of the autoencoder is to minimize the distance function (for example L2 distance), called the loss function, between each input X and each output X'. The learning process of the autoencoder consists in iteratively refining its internal parameters such that the evaluation of the loss function on all the learning set is reduced. The loss function associated to the autoencoder, denoted L_{auto} is defined as

$$L_{\text{auto}} = ||X - g(f(X))||, \tag{1}$$

where X represents the input signal and $||.||$ symbolizes the L2 distance.

The minimization of the loss function that maps from X to Z in order to obtain X' does not guarantee the extraction of useful features. Indeed the network can achieve a perfect reconstruction by simply "copying" the input data and thus obtaining a minimal mapping error. Without any other constraints, the network can miss a good representation of the input data. A strategy to avoid this problem is to constrain the reconstruction criterion by cleaning or denoising partially corrupted input data with a denoising autoencoder (Vincent et al. 2008).

The methodology consists in first applying noise, for example an additive Gaussian noise, on input data to corrupt the initial input X into $\widetilde{X}$. Then the autoencoder maps $\widetilde{X}$ to Z via the encoder f and attempt to reconstruct X via the decoder g (see Fig. 4). Although Z is now obtained by applying the encoder on corrupted data $\widetilde{X}$, the autoencoder is still minimizing the reconstruction loss between a clean X and its reconstruction from $\widetilde{X}$ with the loss function

$$L_{\text{denoising}} = ||X - g(f(\widetilde{X}))||, \tag{2}$$

where $\widetilde{X} = X + \epsilon$, with ϵ an additive uniform noise with the same dimension as X (see Fig. 4).

As the number of parameters of the autoencoder is high, light curves are sparse, and the size of the training database that we will have is small, the overfitting can seriously affect the performance of the network in our case. A solution is to introduce sparsity to the learned weights to avoid learning "noisy" patterns.

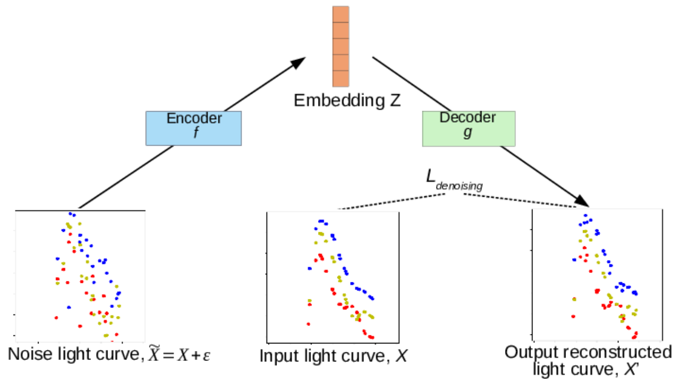

Encoder
f
Embedding Z
Decoder
g
$L_{denoising}$
Noise light curve, $\widetilde{X} = X + \varepsilon$
Input light curve, X
Output reconstructed light curve, X'

Fig. 4. Schema of the denoising autoencoder process.

The sparsity concept makes the assumption that only a few neurons are required in order to reconstruct a complete signal. Let us consider that a neuron is active if its output value is close to one and inactive if its output value is close to zero. We want a very sparse representation such that the neurons of a given layer should be inactive most of the time and so obtain an average activation of neurons close to zero. Therefore the output signal is reconstructed with a very limited number of activated neurons.

We note $\hat{\rho}_j$ the average activation of hidden unit j over the training set. To force neurons to be inactive, we enforce the constraint $\hat{\rho}_j = \rho$, with ρ being a sparsity parameter that is initialized close to zero. Thus the average activation of each hidden neuron has to be close to a small value and so most of the neurons have to be inactive to satisfy this constraint.

In practice, a penalty term is added to the loss function to penalize $\hat{\rho}_j = \rho$ deviating significantly from ρ. One of the most frequently used penalty terms is the Kullback–Leibler (KL) divergence (Hinton 2002) defined as

$$\mathrm{KL}(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \left(\frac{1-\rho}{1-\hat{\rho}_j} \right). \tag{3}$$

KL divergence measures how two distributions are different from one another. It has the property that $\mathrm{KL}(\rho \| \hat{\rho}_j) = 0$ if $\hat{\rho}_j = \rho$ and otherwise it increases monotonically as $\hat{\rho}_j$ diverges from ρ.

The loss function now integrates this penalty term as

$$L_{\text{sparse}} = \|X - g(f(X))\| + \alpha \Omega_L, \tag{4}$$

with $\alpha \in \mathbb{R}$ being a scalar that weights the sparse regularization term Ω_L , which is the sum of the KL divergence term for neurons of a chosen layer noted L defined as

$$\Omega_L = \sum_j \mathrm{KL}(\rho \| \hat{\rho}_j) = \sum_j \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}, \tag{5}$$

$j \in \{1, \ldots, N_j\}$ with N_j the number of neurons of layer L.

3.3. Contrastive loss function

The contrastive loss function was introduced to perform a dimensionality reduction by ensuring that semantically similar examples are embedded close together (Hadsell et al. 2006). It was shown that this method provides invariance to certain transformations on images. The contrastive loss function is computed over pairs of samples unlike traditional loss functions, which are a sum over all the training database. We note $(\mathbf{Z_1}, \mathbf{Z_2})$ a pair of input data and Y a binary label assigned to this pair. If $\mathbf{Z_1}$ and $\mathbf{Z_2}$ have the same label, $Y = 0$, otherwise $Y = 1$. The distance function between $\mathbf{Z_1}$ and $\mathbf{Z_2}$ is learned as the euclidean distance: $D = \|\mathbf{Z_1} - \mathbf{Z_2}\|$. Thus the loss function tends to maximize D if they have dissimilar labels and minimize D if $\mathbf{Z_1}$ and $\mathbf{Z_2}$ have similar labels. So we can write the loss function as

$$L(D, (Y, \mathbf{Z_1}, \mathbf{Z_2})) = (1-Y) L_S(D) + Y L_D(D), \tag{6}$$

with $(Y, \mathbf{Z_1}, \mathbf{Z_2})$ a labeled sample pair, L_S the partial loss function for a pair of similar labels, and L_D the partial loss function for a pair of dissimilar labels. To get low values of D for similar pairs and high values of D for dissimilar pairs, L_S and L_D must be designed to minimize L. We introduce the margin $m > 0$, which defines a minimum distance between $(\mathbf{Z_1}, \mathbf{Z_2})$. Dissimilar pairs contribute to the loss function only if their distance is below this minimum distance so that pairs who share the same label will be brought closer, and those who do not share the same label will be driven away if their distance is less than m. The final contrastive loss function is defined as

$$L_{\text{contrastive}} = (1-Y) \|\mathbf{Z_1} - \mathbf{Z_2}\| + Y \max\left(0, m - \sqrt{\|\mathbf{Z_1} - \mathbf{Z_2}\|}\right)^2. \tag{7}$$

4. Proposed architecture

We developed PELICAN to obtain the best feature-space representation from light curves and perform a classification task. In this work, we apply PELICAN for the classification of supernovae light curves but it can be extended to the classification of other variable or transient astrophysical objects.

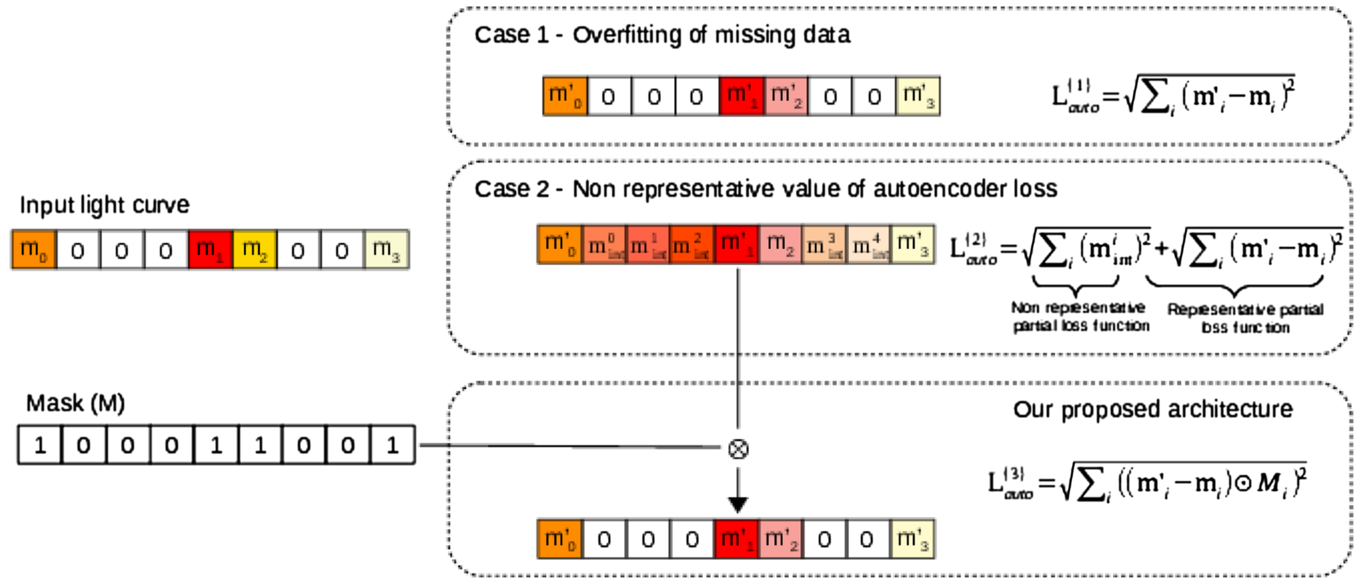
PELICAN is composed of three successive modules (see Fig. 7). Each of them has a specific purpose with a loss function associated. The first module learns a deep representation of light curves from the test database under an unsupervised autoencoder method. The second module optimizes a contrastive loss function to learn invariance features between the bright and fainter supernovae from the training database. Finally, the third module performs the classification task. In this section we explain in more detail the different mechanisms and objectives of the operations related to each module.

4.1. Autoencoder branch

To deal with the low number of examples in the training database, which leads to overfitting and mismatch between the spectroscopic and photometric distributions (see Fig. 2), we propose to train an unsupervised sparse autoencoder method on the test database. In this way we can benefit from the information of light curves in the test database without knowing the label associated to each object.

The autoencoder takes as input a batch of LCIs of size $h \times w$ from the test database, which are encoded and decoded through a CNN architecture. To extract useful features, we applied an uniform noise, which affects differently each magnitude on the light curve by adding a random value $\in [-0.5, 0.5]$ mag, before passing through the encoder (see Fig. 4).

In the first part of the CNN, the encoder, which is composed of nine convolution layers (conv 1 to conv 9 in Fig. 7) and four pooling layers (Pool 1,4, 6, and 8), converts the input noisy LCIs into an embedding representation. Then, the decoder reconstructs the original LCIs from the embedding representation through two fully connected layers (FC10 and FC11) of 5000 neurons. So the output of the autoencoder is a reconstructed LCI with the same size as the input, $h \times w$. As this part is trained on the test database, which contains a sufficiently large quantity of data, it allows us to design a deep architecture with a



Case 1 - Overfitting of missing data
m'0 0 0 0 m'1 m'2 0 0 m'3
$L^{(1)}_{auto}=\sqrt{\sum_i (m'_i - m_i)^2}$
Case 2 - Non representative value of autoencoder loss
Input light curve
m0 0 0 0 m1 m2 0 0 m3
m'0 m0int m1int m2int m'1 m'2 m3int m4int m'3
$L^{(2)}_{auto}=\sqrt{\sum_i (m^i_{int})^2}+\sqrt{\sum_i (m'_i - m_i)^2}$
Non representative partial loss function
Representative partial loss function
Mask (M)
1 0 0 0 1 1 0 0 1
Our proposed architecture
$L^{(3)}_{auto}=\sqrt{\sum_i ((m'_i - m_i)\odot M_i)^2}$
m'0 0 0 0 m'1 m'2 0 0 m'3

Fig. 5. Illustration of the overfitting of the missing data that could appear in the autoencoder process and the solution proposed to overcome it. The input light curve is composed of different magnitudes (m_0, m_1, m_2 m_3) and missing values represented by zero values. In case 1, the algorithm has completely overfitted the missing data by replacing them at the same position on the light curve. So the loss function, $L^{(1)}_{\text{auto}}$, is ideally low. In case 2 the algorithm has completed the missing data by interpolating them. However, as the computation of the loss is made between the new values of magnitudes, (m^0_{int}, m^1_{int}, m^2_{int}, m^3_{int}, m^4_{int}), compared to zero values, the value of the loss $L^{(2)}_{\text{auto}}$ is overestimated. The solution that we provided is to multiply the interpolated light curve by a mask M before the computation of the loss, $L^{(3)}_{\text{auto}}$.

large number of parameters and so learn high-level features. Moreover the first convolution layers are composed of a large kernel size to extract large temporal patterns and capture the maximum of observations, as the light curve is mainly composed of zero values. Then the feature maps are reduced to the size of the convolution kernels to limit the number of parameters. The loss function associated to the autoencoder, called Autoencoder Loss in Fig. 7, minimizes the difference between the original LCI and the reconstructed one. However, we have to pay attention to the overfitting of missing data on the light curve. The problem of sparse data has already been the subject of few studies (Liu et al. 2018; Eldesokey et al. 2018; Hua & Gong 2018). In this work we developed a different approach very specific to the classification of light curves. An illustration of the overfitting problem and the solution we propose is given in Fig. 5. By construction, the input LCI is composed of many zero values (see Sect. 3.1) that are propagated in the network as real data. If we compute a classical autoencoder loss function, two scenarios are possible. In the first case, the model could learn to reconstruct the LCI with the missing data that do not have a physical sense (see case 1 in Fig. 5). In the second case, the model is able to interpolate data. However, the autoencoder loss cannot take into account these interpolated values as they are compared to zero values on the initial LCI, and so lead to a divergence of the loss function (case 2 in Fig. 5). Therefore we propose to define a mask with the same size as the considered original light curve, filling with one if there is an observation on the light curve, and zero otherwise. The reconstructed LCI is then multiplied by the mask before the minimization of the loss function (case 3 in Fig. 5). Equation (1) becomes

$$L_{\text{auto}} = \|X - g(f(X)) \odot M(X)\|, \tag{8}$$

with $M(X)$ being the mask related to the input light curve X.

Finally, we compute the penalty term as defined in Eq. (5), in the second fully connected layer, FC11, and call it sparsity loss. It depends on two hyperparameters: the sparsity parameter ρ and the weight of the sparse regularization α. To determine the best values of ρ and α, we searched the best combination using a 2D grid search among values in the following finite sets: $\{10^{-5}, 5\times10^{-4}, 5\times10^{-3}, 10^{-3}, 5\times10^{-2}, 10^{-2}, 5\times10^{-1}, 10^{-1}\}$ and $\{10^{-3}, 5\times10^{-2}, 10^{-2}, 5\times10^{-1}, 10^{-1}\}$, respectively.

However, the regularization term does not take into account the number of observations on each light curve, which varies significantly. It may cause overfitting as the number of active neurons is then always the same whatever the number of data points on each light curve. So the number of active neurons has to be adapted depending on the number of observations in all filters. Thus, we propose to express the sparsity parameter, ρ, as a linear function depending on the number of observations for each light curve. This contribution allows us to increase the number of active (inactive) neurons when the light curve is densely (poorly) populated with observations. We define a new sparsity parameter $\rho'(l)$ for the specific light curve denoted l as

$$\rho'(l) = \rho_{\text{a}} n_l + \rho_{\text{b}}, \tag{9}$$

with n_l the number of observations on the light curve l; ρ_{a} and ρ_{b} are two hyperparameters. They are determined at the same time as α using a 3D grid search among the same values as ρ.

In this case, the sparse regularization term (see Eq. (5)) of our autoencoder module takes the form

$$\Omega'_L(l) = \sum_j \rho'(l)\log\frac{\rho'(l)}{\hat{\rho}_j} + (1-\rho'(l))\log\frac{1-\rho'(l)}{1-\hat{\rho}_j}. \tag{10}$$

4.2. Contrastive branch

Once the autoencoder training has converged on the test database the weights of its convolution and fully connected layers are fixed. Another strategy is to fine-tune the weights of the autoencoder branch using the contrastive loss function. In our case, this approach has two problems. The first one is to obtain features from the autoencoder module that are less representative of the test database, which does not allow the model to overcome the non-representativeness between the training and test databases. The second problem is an overfitting of the training database due to its small size, which decreases the performance. Then, the output of a chosen layer of the encoder part is given as input to the contrastive branch. This second module is designed to reduce the mismatch between the training (higher magnitudes) and the test (lower magnitudes) databases. This requires a specific contrastive loss function that is minimized through a CNN architecture. So we propose a loss function that minimizes the variations of intra-class light curves and maximizes the variations of inter-class light curves. In this way, we split the training database into four subsets following a cut magnitude m^c in the i-band magnitude.

If we denote $m^{\rm Ia}(l)$ the i-band median magnitude of a type Ia light curve and $m^{\rm non-Ia}(l)$ the i-band median magnitude of a non-Ia-type light curve, a given light curve can belong to one of the four following subsets:

– LC1 : type Ia light curves with $m^{\rm Ia}(l) < m^c$,
– LC2 : type Ia light curves with $m^{\rm Ia}(l) > m^c$,
– LC3 : non-Ia-type light curves with $m^{\rm non-Ia}(l) < m^c$,
– LC4 : non-Ia-type light curves with $m^{\rm non-Ia}(l) > m^c$.

Therefore the goal is to define a loss function that minimizes the variation between intra-class light curves, that is, between the LC1–LC2 and LC3–LC4 sets, and maximizes the variation between inter-class light curves, that is, between LC1–LC3, LC1–LC4, LC2–LC3, and LC2–LC4 sets.

Equation (7) becomes

$$L = \frac{1}{2}\max\left(0, m - \sqrt{\|LC1 - LC3\|}\right)^2 + \frac{1}{2}\max\left(0, m - \sqrt{\|LC1 - LC4\|}\right)^2 + \frac{1}{2}\max\left(0, m - \sqrt{\|LC2 - LC3\|}\right)^2 + \frac{1}{2}\max\left(0, m - \sqrt{\|LC2 - LC4\|}\right)^2 + \|LC1 - LC2\| + \|LC3 - LC4\|. \quad (11)$$

We introduce $\frac{1}{2}$ terms into the formula to weight the inter-class distances so that the inter-class and the intra-class distances have the same weight in the computation of the loss function.

In practice this means that the encoder is fed with sets of four light curves from the training database, with one light curve from each subset. At each iteration light curves are randomly selected. If all light curve subsets have been transmitted, the training database is randomly shuffled and the procedure continues. This procedure allows us also to avoid overfitting as the number of possible pair combinations is larger than the original training database. The learning of the contrastive branch (see Fig. 7) is done without updating the training weights of the autoencoder, which have been adjusted during the non-supervised step on the test database. This step allows us also to solve the problem of asymmetry that exists between the classes as this module takes as input both light curves of type Ia and non-Ia supernovae at the same time. As this part of the network is trained only on the training database, the number of convolution layers is smaller than in the first module of the autoencoder to avoid overfitting. Features from the seventh convolution (conv 7 in Fig. 7) are given as input to the contrastive branch where the training weights are updated. Therefore the minimization of the contrastive loss is made only on the training database. The choice of the seventh convolution layer as input to the contrastive branch was made for several reasons. First of all, as the encoder part of the first module is dedicated to the extraction of relevant features from the test light curves to characterize them precisely, while the decoder part is designed to reconstruct the original light curve,

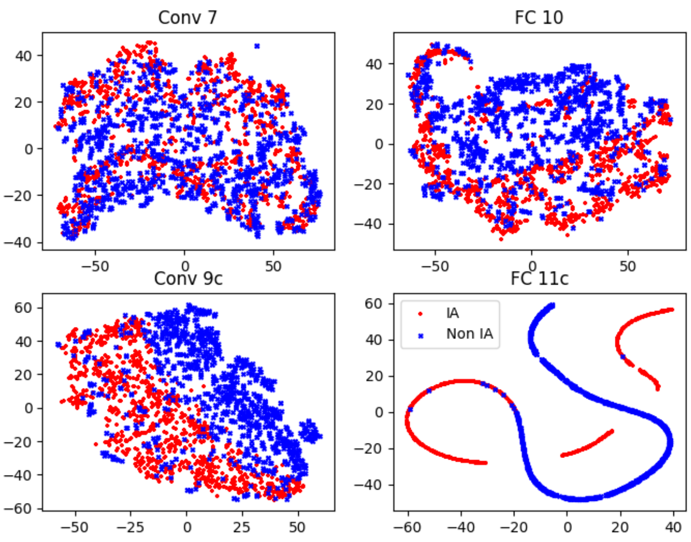


Fig. 6. Representation of the t-distributed stochastic neighbor embedding (t-SNE) projections with features extracted from two layers of the autoencoder module (Conv 7 and FC 10) and from two layers of the contrastive module (Conv 9c and FC 11c).

we decided to extract features from the first part of the autoencoder to reduce the mismatch between the training and the test databases. Figure 6, which represents the t-SNE[1] projections of features, offers a means of better understanding. If the projection of features from the first fully connected layer (FC 10) of the autoencoder part shows a better separation of type Ia and non-Ia supernovae than from the seventh convolution layer, the extraction of these features for the contrastive branch degrades the performance. This means that it is preferable to consider a feature representation space of light curves of high abstraction level rather than a representation apparently more suited for classification in the autoencoder layers, as it allows a significant reduction of the mismatch between the training and the test databases. The last layers of the contrastive module (conv 9c and FC 11c) mark a clear separation between type Ia and non-Ia supernovae (bottom panel of Fig. 6).

4.3. Classification branch

The last module of PELICAN is composed of three fully connected layers (see Fig. 7) with a low number of neurons to reduce overfitting. It takes its input features from the two first modules to perform the classification step. Indeed to make the final classification, this part needs information from the first module that fully characterizes the light curves of the test database and so gives a large variety of features that allows us to reduce the mismatch between the training and test databases. However, this time we extract features from the decoder part as it was shown that it is able to make a separation of the classes that is relevant for this final step (see Fig. 6). Then the classification branch must benefit from features of the second contrastive branch, and particularly the fully connected layer (FC11c), which reduce again the mismatch while marking a separation between classes.

[1] The t-distributed stochastic neighbor embedding (t-SNE, van der Maaten & Hinton 2008) is a non-linear dimensionality reduction technique well suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

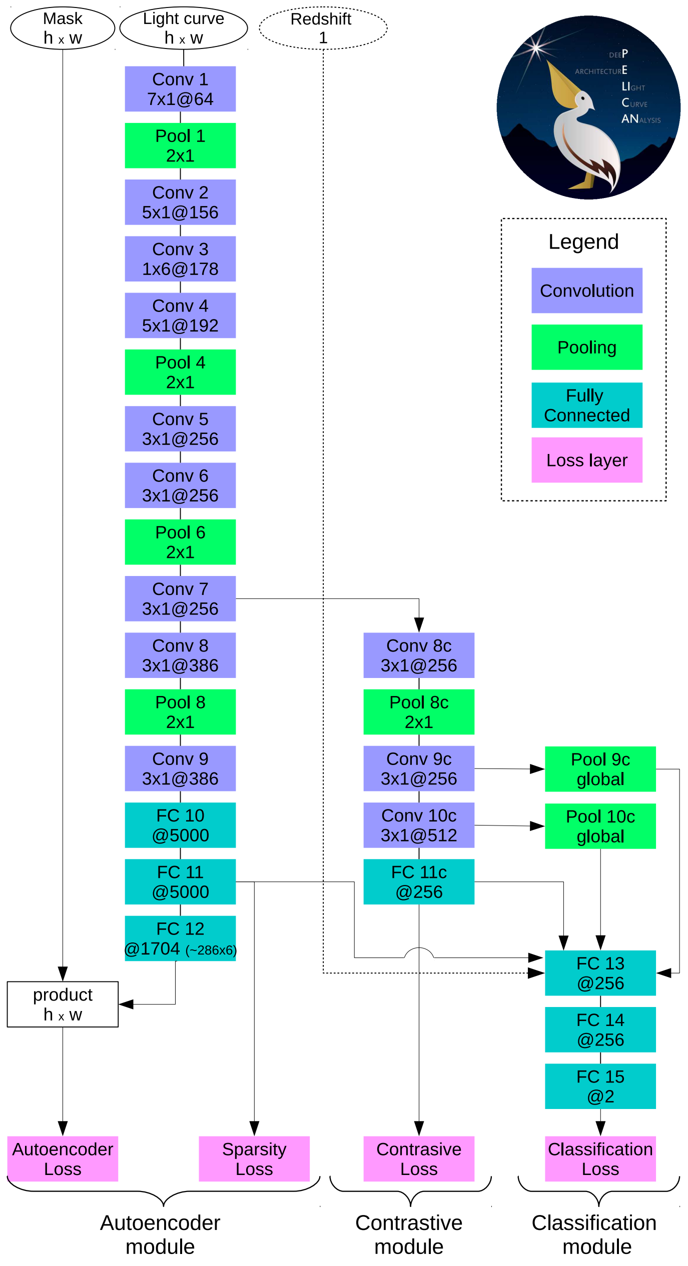


Fig. 7. Representation of PELICAN architecture, which is composed of three modules: the autoencoder, the contrastive, and the classification modules. The first module optimizes the autoencoder loss containing a sparsity parameter (see Eq. (10)). In the second module, the contrastive loss (see Eq. (11)) is optimized to bring the features with the same label together. Finally the third module performs the classification step optimizing a standard classification loss.

Finally to combat the overfitting of missing data, the third module takes also as input, features from the ninth and tenth convolution layers of the contrastive branch (conv 9c and conv 10c). We apply a specific operation, called a global pooling, which allows us to transform a 2D output feature vector of a convolution layer into a 1D feature vector given as input to a fully connected layer. We choose to apply a global max pooling that will select only the maximum value on the 2D output feature maps from the convolution layers, excluding zero values and so missing data. We also make use of a dropout technique (Srivastava et al. 2014) on the two fully connected layers FC13 and FC14 to combat overfitting.

5. Light curve data

We tested and adapted our method on three different databases. First we evaluate the techniques on simulated data from the Supernovae Photometric Classification Challenge (SPCC, Kessler et al. 2010b,a) then on simulated LSST light curves for the main survey and the deep fields. Finally we explore the possibility of making the learning step on simulated light curves and then testing on real data. We apply this last work on SDSS supernovae light curves (Frieman et al. 2008; Sako et al. 2008).

5.1. *The SuperNova ANAlysis software (SNANA)*

Light curves have been simulated using the SNANA simulator (Kessler et al. 2009). This is an analysis package for supernovae light curves that contains a simulation, a light curve fitter and a cosmology fitter. It takes into account actual survey conditions and so generates realistic light curves by using the measured observing conditions at each survey epoch and sky location. First the supernovae properties are generated by choosing a shape-luminosity and color parameters, which are used in addition to other internal model parameters to determine the rest-frame magnitude at each epoch. Then K-corrections are applied to transform rest-frame model magnitudes to observed magnitudes in the telescope system. Finally the ideal above atmosphere magnitudes are translated into observed fluxes and uncertainties. Observed magnitudes are also simulated and that is the input we used for each light curve given as input to the network. Type Ia supernovae light curves are simulated from spectral adaptive light curve template (SALT2; Guy et al. 2007) or multicolor light curve shapes (MLCS) models (Jha et al. 2007; Kessler et al. 2009). However, there are no such models for non-Ia types. So the simulations use a library of spectral templates that give the supernovae flux as a function of epoch and wavelength. Only well-sampled photometric light curves are used because spectral templates are interpolated to cover all wavelengths and epochs. The current library contains composite and individual templates for types Ib, Ibc, IIn, and IIP.

5.2. *Supernova Photometric Classification Challenge data*

The SPCC dataset is composed of simulated light curves of supernovae in griz filters of the Dark Energy Survey (DES). The dataset was subdivided in a spectroscopically confirmed subset of 1103 light curves, which constitutes the training dataset, and a test dataset of 20 216 light curves. However, the training dataset is small and highly biased as it is not representative in brightness and in redshift compared to the test set.

5.3. *Simulated Large Survey Synoptic Telescope data*

As LSST will observe a large number of supernovae, the photometric classification of supernovae types from multiband light curves is necessary. There will be two main kind of cadences. The first one dedicated to the main survey is called the Wide-Fast-Deep (WFD). It will scan a very large area of the sky. The second one, called Deep Drilling Fields (DDF), will focus on small part of the sky with a higher cadence and deeper images. Thus this will correspond to well-measured light curves (see Fig. 1) but for a smaller sample.

To validate our method in the context of the future LSST data, we simulated light curves of supernovae as observed with the WFD and DDF observational strategies, with the minion 1016 baseline model (Biswas et al. 2017). The simulation was realized in the ugriz filters of the LSST. We assume a ΛCDM cosmology with $\Omega_M = 0.3156$, $\Omega_\Lambda = 0.6844$, and $w_0 = -1$. Simulations are made in a finite redshift range, $z \in [0.05, 1.20]$. We consider an efficiency for the image subtraction pipelines reaching 50% around a signal-to-noise ratio (S/N) of ~5. Each object must have two epochs in any band. For the simulation of type Ia light curves, the color and the light curve shape's parameters vary in the following intervals: $c \in [-0.3, 0.5]$, $x_1 \in [-3, 2]$. The simulation of non-Ia types is based on a library of spectral templates for types Ib, Ibc, IIn, and IIP.

Our simulation includes a spectroscopically confirmed sample from the DDF survey. It is based on observations from an 8 m class telescope with a limiting *i*-band magnitude of 23.5. In this work we assume a different allocating time for the spectroscopic follow-up. A reasonable scenario allows a spectroscopic follow-up of 10% of the observed light curves in DDF, that is, 2k spectroscopically confirmed light curves of supernovae. However, we also consider a most restrictive case by assuming that only 500 then 1000 light curves are spectroscopically confirmed. Moreover, we explore two ideal cases for which 5k then 10k supernovae have been followed up. Finally we also consider different numbers of photometric observations of light curves as it is interesting to classify light curves before ten-years observation of LSST. All the configurations are summarized on Table 2.

5.4. Sloan Digital Sky Survey data

As simulated data do not reproduce perfectly the real data, it is interesting to test our method on real data. The ideal strategy is to simulate light curves that correspond to the SDDS survey to train the model and then test on real data. This is a challenging methodology as there is a severe mismatch between the training and the test databases. However, making a model able to remove this kind of mismatch is crucial for future surveys where the spectroscopic follow-up is limited. Therefore we simulated light curves of supernovae that correspond to SDSS data. Then, we extracted light curves in ugriz filters from the SDSS-II Supernova Survey Data (Frieman et al. 2008; Sako et al. 2008). The SDSS-II SN data were obtained during three month campaigns in the Fall of 2005, 2006, and 2007 as part of the extension of the original SDSS. The Stripe 82 region was observed with a rolling cadence. Some spectroscopic measurements were performed for promising candidates depending on the availability and capabilities of telescopes (Sako et al. 2008). A total of 500 SN Ia and 82 core collapse SN were spectroscopically confirmed.

6. Experimental protocol

In this section, we explain the protocol and the different techniques used for the training process.

6.1. Data augmentation

In this classification context, data augmentation is a crucial step. Indeed, in order to make PELICAN robust against the differences between the training and the test databases (i.e., sampling, mean magnitude, noise), it is essential to use different data augmentation techniques. Moreover, when light curves that compose the training and test databases are measured with different observational strategies, the difference in sampling is increased and the data augmentation has to be reinforced. This is the case in the context of LSST if we compare light curves from the WFD survey on the one hand, and light curves from the DDF survey on the other hand. To enable PELICAN to learn on DDF light curves and generalize on WFD light curves, the data augmentation has to be adapted.

Finally as supernovae from the test database are often fainter, errors on their fluxes are often bigger. Therefore the data augmentation also needs to be applied to the errors.

Thus, in addition to the uniform noise applied differently on each magnitude of light curves given as input to the denoising autoencoder, we add two other kinds of noise on magnitudes of light curves:

- an uniform constant noise $\in [-1.0, 1.0]$ mag, which is added to all the magnitudes of the light curve,
- an uniform noise $\in [0.93, 1.07]$, which is multiplied by all the magnitudes of the light curve.

The variation of the noise has been chosen arbitrarily but is large enough to increase the size of the training database and include potential systematic errors that could not have been included in the simulated error model. Then we randomly remove one or several magnitudes or/and all magnitudes for a given band. This process is particularly effective for the classification of light curves of supernovae observed with a WFD strategy based on a training on supernovae light curves from the DDF survey.

Finally, to prevent the PELICAN model from learning the missing value positions on each light curve, we perform random time translations keeping all the data points but varying their positions in time. So the learning becomes invariant to the position of points.

6.2. Setting learning parameters

We used the Adam optimizer (Kingma & Ba 2014) for all the training steps in different modules with a learning rate decreasing by a factor of ten after 25 000 iterations. The batch size during the learning is fixed to 96 light curves. For the autoencoder learning, we optimized the values of the sparsity parameters over one validation base and used them for all the different configurations, as they are not sensitive to the database. We set ρ_a and ρ_b equal to 5×10^{-4} and 0.0, respectively, and α to 0.01.

The cut parameter m^c in the *i*-band magnitude from the second module depends on the database. We chose its value in order to have enough examples on both sides of the cut in magnitude. We set m^c to 23.5 mag, 24.5 mag, and 22.5 mag for the SPCC, LSST, and SDSS databases, respectively. The values of these parameters are not sensitive and a small variation in them did not change the results.

6.3. Ensemble of classifiers

To increase the performance, we trained an ensemble of classifiers as it has been shown to be more accurate than individual classifiers (e.g., Polikar 2006). Moreover the generalization ability of an ensemble is usually stronger than that of base learners. This step involves training N times one model with the same training database but a different initialization of the weights. We chose $N = 7$ and the individual decisions were then averaged out to obtain the final values of probabilities. This step allows us to increase the accuracy by 2% on average.

7. Results

In this section we present the results that we obtained for each dataset.

7.1. Metrics

To evaluate the performance of PELICAN in different contexts, we use several commonly used statistic metrics, which are the accuracy (Acc), the recall (R) or true positive rate (TPR), the precision (P), and the false positive rate (FPR). They are defined from the following confusion matrix:

		Predictive label	
		Ia	Non Ia
True label	Ia	True Positive (TP)	True Negative (TN)
	Non Ia	False Positive (FP)	True Negative (TN)

- $$\text{Acc} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})} \quad (12)$$
- $$\text{R(or TPR)} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (13)$$
- $$\text{P} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \quad (14)$$
- $$\text{FPR} = \frac{\text{FP}}{(\text{FP} + \text{TN})}\cdot \quad (15)$$

As a graphical performance measurement, we use the ROC (receiver operating characteristic) curve, which is plotted with TPR on the y-axis against FPR on the x-axis. It gives an estimation of the performance of a classifier at different threshold settings. The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing the lack of false negatives and false positives. A random guess would give a point along a diagonal line from the left bottom to the top right corners.

From the ROC graphic, we can extract the value of the AUC (area under the curve), which captures the extent to which the curve is up in the upper left corner. The score has to be higher than 0.5, which is no better than random guessing.

7.2. Supernova Photometric Classification Challenge

The first evaluation of PELICAN is made on the SPCC dataset. We trained the model with two different training datasets: a representative training database and a non-representative training dataset. The representative training database is a simplified theoretical scenario, in which there is no limitation in brightness and redshift of the spectroscopic follow-up. It is built by randomly selecting 1103 light curves from the whole dataset. The non-representative training database, which represents the real scenario, is the spectroscopically confirmed subset of 1103 light curves that was proposed for the challenge. This last database is non-representative of the test dataset in brightness and redshift.

As shown in Lochner et al. (2016; noted L16 hereafter) the best average AUC is obtained by extracting SALT2 features and using boosted decision trees (BDTs) as the classifier. Therefore we compared the performance of PELICAN with the BDTs algorithm that takes as input SALT2 features. We tested both methods with and without the information on the redshift inside the training. The ROC curves for both methods are represented in Fig. 8 (on left panels) and the values of statistics are reported in Table 1.

By considering a non-representative training database, without including the redshift during the training, PELICAN obtains an accuracy of 0.856 and an AUC of 0.934, which outperforms the BDTs method, which reaches 0.705 and 0.818. If we train both methods on a representative training database, as expected, the performance increases. The accuracy and the AUC become 0.911 and 0.970 with PELICAN, against 0.843 and 0.905 with the BDTs algorithm. It is interesting to note that the gain in statistics obtained with PELICAN is lower than BDTs values, which means that PELICAN is able to better deal with the problem of mismatch. This ability will be confirmed by the promising results obtained with a non-representative training database composed of LSST light curve simulations (see Sect. 7.3).

The performance of PELICAN does not change by adding the redshift information during the training, which is not the case for the BDTs algorithm, for which the accuracy and the AUC are slightly increased. This might mean that PELICAN is able to extract by itself the redshift information during its training. Figure 8 shows the accuracy as a function of redshift, with the corresponding redshift distributions and BDTs results for comparison. If PELICAN is trained on a representative training database, the accuracy tends to decrease at low redshifts and at redshift above 1.0, as the corresponding redshift bins are poorly populated at these extreme values. A further trend is observed for BDTs, except at redshift above 1.0, and only if redshift values are included as an additional feature for the training. By considering a non-representative training database, the accuracy significantly decreases at high redshift for both methods. As the addition of redshift into the training does not change the tendency obtained by PELICAN, this trend in function of redshift is likely due to the too small number of examples at high redshifts.

7.3. Large Survey Synoptic Telescope simulated light curves

The next step is to evaluate PELICAN on simulated LSST light curves under realistic conditions. In this way, we consider for all the tests a non-representative spectroscopically confirmed database from the DDF survey as represented in Fig. 2. We consider different configurations of the training and test databases. We constrain the number of spectroscopically confirmed light curves to vary between 500 light curves to 10k. Even if the upper bound corresponds to an ideal scenario in which roughly more than 40% of light curves in the DDF have been followed up, it is interesting to compare the performance of PELICAN with a large training sample.

We simulated light curves with the minion 1016 cadence model. This model includes a WFD survey and five DDF (see Fig. 9). It is not certain that a spectroscopic follow-up will be performed on supernovae light curves in WFD fields. So we use a different approach that consists in training PELICAN on DDF light curves and then adapting the pre-trained model to classify supernovae light curves observed in the WFD survey. This strategy allows us to consider the possibility of benefiting from SN Ia candidates from WFD fields to constrain the cosmological parameters, without any spectroscopic follow-up of the main survey.

7.3.1. Classification of Deep Drilling Fields light curves

The results of the different configurations are reported in Table 2 and the ROC curves for some of these configurations in Fig. 10. In addition to the values of the accuracy and the AUC, we compare the recall of SNe Ia by constraining the precision to be higher than 95% and 98%. Such a level of contamination becomes competitive with spectroscopy contamination. Again we compared the performance of PELICAN with the best method highlighted in L16, which is BDTs and SALT2 features.

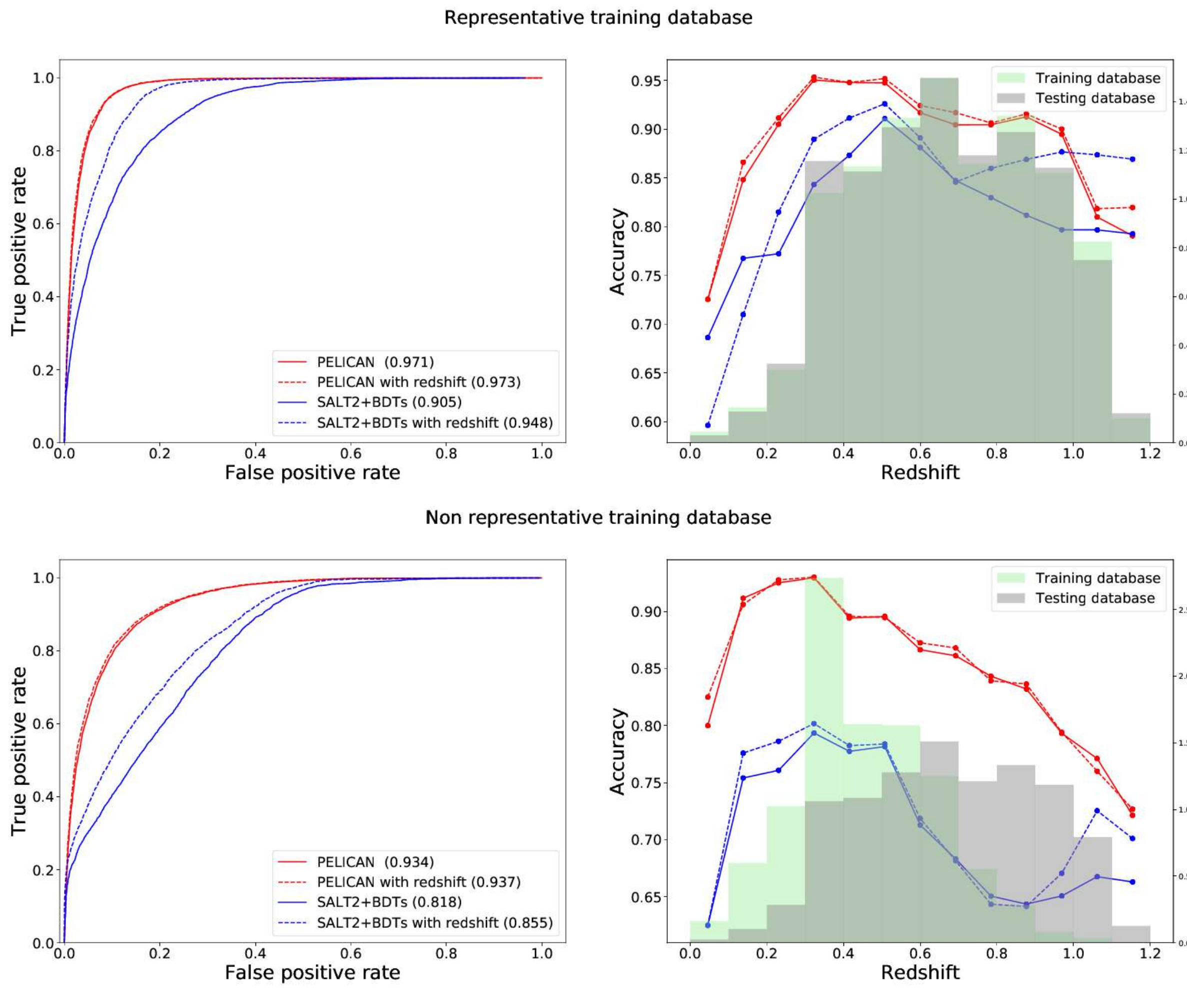


Fig. 8. Comparison of ROC curves with the AUC score in brackets (*left panels*) and the accuracy versus redshift (*right panels*) for PELICAN (in red) and the BDTs method (in blue), with (solid lines) and without (dashed lines) the redshift included in the training. The representative case is on the first line and the non-representative one on the second line.

Table 1. Statistics obtained for SPCC challenge by PELICAN, with BDTs results in parenthesis.

SPCC	Redshift	Training database	Accuracy	AUC
Non-representative training database (SALT2+BDTs)	No	1103 spec	0.856 (0.705)	0.934 (0.818)
	Yes	1103 spec	0.863 (0.713)	0.939 (0.855)
Representative training database (SALT2+BDTs)	No	1103 mix of spec and phot	0.911 (0.843)	0.970 (0.905)
	Yes	1103 mix of spec and phot	0.917 (0.878)	0.971 (0.948)

Notes. The first part reports results for a non-representative training database and the second part for a representative training database. We consider both cases by adding or not the redshift values to the training.

Even if this method was not designed for such training configurations, it allows us to compare a feature-based machine learning method to PELICAN.

If we consider the most constraining configuration composed of only 500 spectroscopically confirmed light curves for the training and 1500 light curves for the test database, PELICAN reaches an accuracy of 0.895, and an AUC of 0.966. Moreover PELICAN is then able to detect 76.9% of SNe Ia with a precision higher than 95% and 60.2% with a precision higher than 98%. These results are quickly improved by considering more examples on both training and test databases. The number of light curves inside the test database is important, especially if the number of examples in the training database is small, as the autoencoder is trained on the test database. Indeed there is about an 8% improvement factor of the recall by going from 1k to 3k light curves in the test database with a fixed number of examples in the training database of 1k light curves. However, this factor becomes negligible if the number of spectroscopically confirmed light curves is sufficient, that is, from 5k examples, with an improvement of around 0.2%. We can see a weak degradation going from 10k total light curves to 24k total light curves for the same number of training examples. However, this effect is low, on the order of 10^{-3}. We can argue that the autoencoder is better able to extract features that represent data well if it is trained on a smaller database (except in the case of an underfitting with a database of 2k). Actually the overfitting of the feature representation of the test database improves the performance.

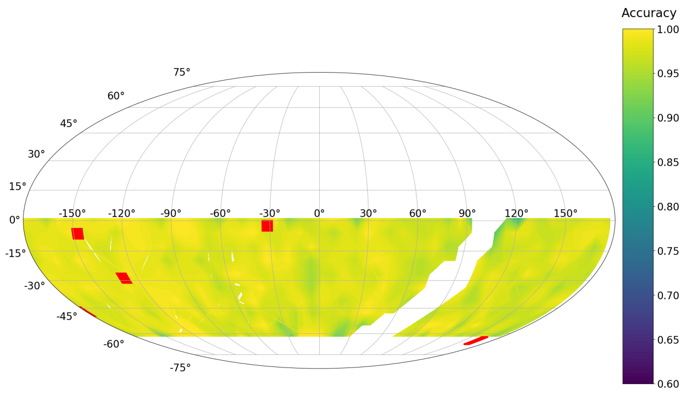


Fig. 9. Spatial distribution of the accuracy obtained with PELICAN for the classification of light curves simulated with minion 1016 cadence model. The Deep Drilling Fields are represented by red squares.

The configuration that seems reasonable after ten years of observations includes a spectroscopic follow-up of 10% of the observed light curves, that is, 2k light curves of supernovae, and a test database of 22k light curves. For this realistic scenario, PELICAN reaches an accuracy of 0.942 and is able to correctly classify 87.4% of SNe Ia with a precision higher than 98%, which constitutes a major result of our study meaning that a large fraction of SNe Ia are well classified by PELICAN, with a precision comparable to a spectroscopy measurement. By considering 10k light curves in the training database, the number of detected SNe Ia is then increased by 9%. All results obtained by PELICAN outperform those obtained by BDTs (the BDTs values are listed in parentheses in Table 2).

The right panel of Fig. 10 shows the accuracy as a function of redshift, with the corresponding redshift distributions on both training and test databases, and the BDTs results for comparison. The accuracy of PELICAN does not depend on redshift until 1.0 where it slightly decreases. This tendency is likely due to the small number of training examples at redshifts higher than 1.0. The BDTs method shows the same behavior at high redshifts.

7.3.2. Classification of light curves in Wide-Fast-Deep survey

The spectroscopic follow-up of SNe Ia candidates is uncertain on the WFD survey. Nevertheless to increase statistics of SNe Ia for cosmological studies, it is interesting to make PELICAN able to classify supernovae light curves from the WFD survey. The strategy consists in training PELICAN on DDF light curves and then testing on light curves observed on WFD fields. However, this methodology leads to another kind of mismatch over and above the existing mismatch between spectroscopically confirmed light curves and unconfirmed ones. Indeed the unconfirmed supernovae from the DDF survey have a different cadence and observational conditions from those of the WFD survey. So the present mismatch is largely increased between the training and test databases. The non-supervised step allows us to reduce the mismatch, as it does for the classification of DDF light curves, but it is not sufficient. The other needed ingredient is the data augmentation to make DDF light curves look like WFD light curves. Thus we performed a severe data augmentation as WFD light curves are about an average of four times more sparse in the u and g bands, and 1.5 times in the r, i and z bands. So we randomly removed 85% of observations on each DDF light curve.

Results are reported in Table 2 and ROC curves in Fig. 11. We consider three configurations of the training and test databases. First we trained PELICAN on a training database of 2k light curves, which could constitute a realistic scenario in which 10% of supernovae in DDF have been spectroscopically confirmed after ten years of observations. We also consider a training database composed of 3k supernovae light curves from DDF as it is still a realistic scenario that includes a spectroscopic follow-up of 12.5% of supernovae in the DDF survey. Finally we trained PELICAN on an ideal training database of 10k supernovae light curves.

With only 2k DDF light curves for the training database and 15k light curves for the test database, PELICAN reaches an accuracy of 0.965. It is able to classify 98.2% of supernovae with a precision higher than 95% and 90.5% with a precision higher than 98%. If we consider 3k light curves for the training database and 40k for the testing database, the percentage of well-classified light curves, with a precision higher than 98%, is 96.4%. With 10k light curves in the training database and 80k in the testing database, the improvement factor is about 1%, where 97.3% of supernovae Ia are correctly classified by PELICAN with a precision higher than 98%. It may seem surprising that the performance is better than for the classification of DDF light curves. This can be explained by the baseline cadence model that we used to simulate data. In this observational model, the supernovae on the main survey (WFD) are observed on several dates often with one or only two different filters for each measurement. However, in the deep fields, supernovae are observed in several filters for a given date but less often over time. The result of this is that a light curve observed in deep fields contains more measurements in all filters but less observations on different days (see Fig. 1 and histograms in the left panels of Fig. 12). In the first module of our architecture, the autoencoder is more efficient at interpolating light curves that contain more observations distributed over time. It means that our method reaches better performance if the number of observations is large over time even if each measurement is not done for each filter. These encouraging results open the possibility of using SNe Ia candidates from the WFD survey, whose the classification precision is comparable to a spectroscopic identification, to constrain cosmological parameters.

The BDTs method obtained poor results for this complex training configuration. This kind of feature-based algorithm has to be adapted to overcome the problem of mismatch, which significantly degrades the performance.

The accuracy as a function of redshift shows a behavior that might seem strange (see Fig. 11). Indeed, the accuracy slightly increases with redshift. This bias is probably due to the mismatch of redshift distributions between the DDF and WFD lights curves. Indeed, during the non-supervised step, low redshift examples are under-represented in the test database, which causes a decrease in the accuracy. This strange behavior is increased for BDTs results. Indeed if the accuracy decreases until redshift 1.0, it increases at redshifts above 1.0. This significant bias is due to the mismatch between the DDF and WFD light curves. Indeed the BDTs algorithm was not designed to deal with this kind of training configuration. Finally, Fig. 9 exhibits no bias across the sky as the accuracy is uniform on WFD survey.

7.4. Classification of real Sloan Digital Sky Survey light curves

The last evaluation of PELICAN is made on real data. The stakes are high as developing a method that can be trained on

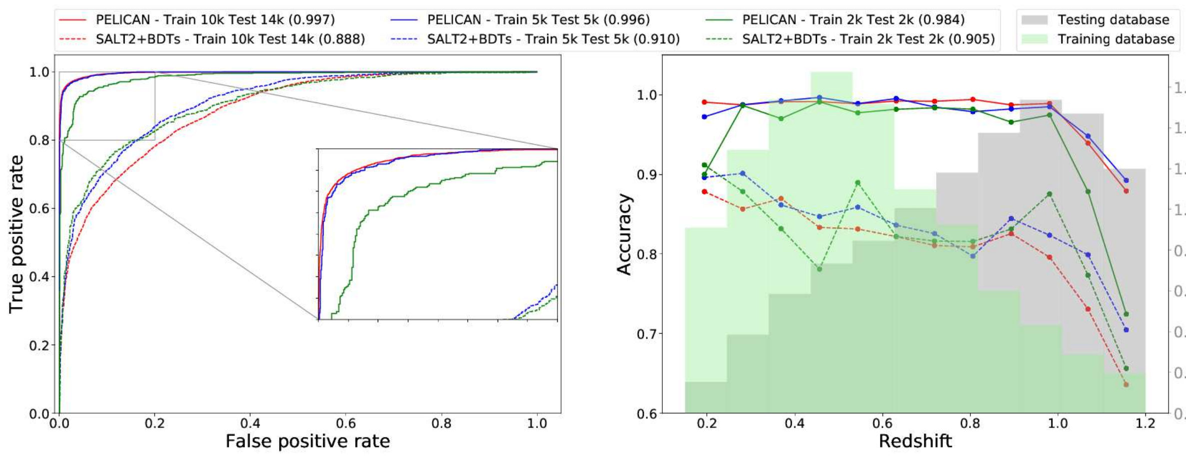


Fig. 10. Comparison of ROC curves for different training configurations of DDF survey, with the AUC score in brackets (*left panel*) and the accuracy versus redshift (*right panel*) for PELICAN (in solid lines) and BDTs method (in dashed lines).

Table 2. Statistics for various training configurations on the DDF survey (first part) and the WFD survey (second part), with BDTs results in parentheses.

	Total (LC)	Training database (Spec. only)	Test database (Phot. only)	Accuracy	$\text{Recall}_{\text{Ia}}$ $\text{Precision}_{\text{Ia}} > 0.95$	$\text{Recall}_{\text{Ia}}$ $\text{Precision}_{\text{Ia}} > 0.98$	AUC
LSST DDF	2000	500	1500	0.895 (0.795)	0.769 (0.382)	0.602 (0.183)	0.966 (0.885)
		1000	1000	0.917 (0.888)	0.827 (0.505)	0.687 (0.353)	0.973 (0.898)
	4000	1000	3000	0.928 (0.850)	0.890 (0.490)	0.764 (0.246)	0.979 (0.895)
		2000	2000	0.938 (0.813)	0.927 (0.594)	0.806 (0.256)	0.984 (0.905)
	10 000	2000	8000	0.946 (0.796)	0.944 (0.496)	0.886 (0.284)	0.989 (0.891)
		3000	7000	0.950 (0.809)	0.950 (0.548)	0.903 (0.285)	0.990 (0.905)
		5000	5000	0.971 (0.818)	0.981 (0.510)	0.959 (0.315)	0.996 (0.910)
	24 000	2000	22 000	0.942 (0.792)	0.940 (0.477)	0.874 (0.209)	0.986 (0.890)
		3000	21 000	0.945 (0.797)	0.937 (0.474)	0.891 (0.254)	0.986 (0.892)
		5000	19 000	0.968 (0.805)	0.978 (0.485)	0.957 (0.228)	0.996 (0.898)
		10 000	14 000	0.971 (0.790)	0.983 (0.465)	0.965 (0.260)	0.997 (0.888)
LSST WFD	17 000	DDF Spec : 2,000	WFD : 15 000	0.965 (0.620)	0.982 (0.041)	0.905 (0.008)	0.992 (0.703)
	43 000	DDF Spec : 3000	WFD : 40 000	0.976 (0.623)	0.995 (0.018)	0.964 (0.000)	0.996 (0.711)
	90 000	DDF Spec : 10 000	WFD : 80 000	0.978 (0.620)	0.995 (0.046)	0.973 (0.000)	0.997 (0.709)

Notes. The different metrics are defined in Sect. 7.1.

simulated data while reaching good performance on real data offers a great opportunity for the future surveys. Indeed by using simulated data, the size of the training database could be unlimited and the problem of mismatch between the training database and the test database could be removed. We evaluate PELICAN only on spectroscopically confirmed supernovae, which corresponds to 500 SNe Ia and 82 core collapse supernovae. In the first step we trained PELICAN only on simulated data and tested on real SDSS light curves, but it reaches poor performance due to the mismatch between simulated and real data (see Table 3). Indeed the sampling and the noise are ideal on simulated data but it is not the case for real ones. Then PELICAN is trained

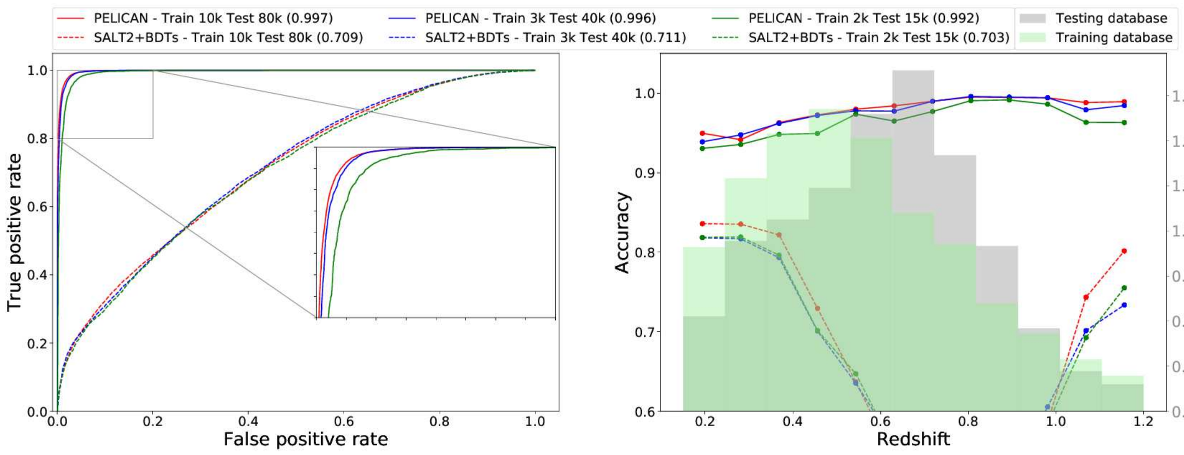


Fig. 11. Comparison of ROC curves for different training configurations of WFD survey, with the AUC score in brackets (*left panel*) and the accuracy versus redshift (*right panel*) for PELICAN (in solid lines) and the BDTs method (in dashed lines).

Table 3. Statistics obtained on real SDSS data.

Training database	Test database	Accuracy	AUC
SDSS simulations: 219,362	SDSS-II SN confirmed: 582	0.462	0.722
SDSS-II SN confirmed: 80	SDSS-II SN confirmed: 502	0.798	0.586
SDSS simulations: 219,362 SDSS-II SN confirmed: 80	SDSS-II SN confirmed: 502	0.868	0.850

Notes. The first line reports results obtained by training PELICAN only on simulated data. In the second line the training and the test databases are only composed of real data. The third line shows an improvement of results by including only 80 SDSS light curves in the training database, which is also composed of simulated light curves.

and evaluated only on real data. The training database is composed of 80 light curves evenly distributed between type Ia and non-Ia-type light curves. The test database is strongly unbalanced and contains mainly type Ia supernovae light curves. In this case, the value of the AUC better captures the performance of the classifier as it takes into account the number of false positives. PELICAN reaches better performance if it is trained only on simulated data with an AUC of 0.722 compared to 0.586 for the classification of only real light curves, as the size of the training database is too small. To improve results, we added 80 real light curves into the training database composed of around 220k light curves. The accuracy and the AUC obtained are 0.868 and 0.850, respectively. This improvement is possible as the architecture of PELICAN overcomes the problem of non-representativeness that appears by mixing real and simulated data. This is a promising result as with only a small subsample of real light curves PELICAN can be trained on simulated data and reaches good performance on real data.

8. Further analysis of the behavior of PELICAN

In this section, we study the impact of characteristics of the input LSST simulated light curves relating to properties or observing conditions.

8.1. Influence of the number of observations

As PELICAN takes only light curves as input, the method should depend on the number of observations. Figure 12 (left panels) shows the correlation between the number of observations on the light curve in all bands, and the accuracy. For the classification of DDF light curves, the accuracy decreases by a small factor of about 9%, as the distributions of the number of observations are the same in both the training and test databases. However, for the classification of WFD light curves, the mismatch is present as PELICAN is trained on DDF light curves that have more observations. So this non-representativeness leads to a further decline of the accuracy, of about 20%.

8.2. Effect of noise

We analyze the impact of S/N, computed as the maximum S/N from all bands, on the accuracy of PELICAN (see middle panels of Fig. 12). For the classification of DDF light curves, the accuracy decreases at small S/N of roughly 10%. For the classification of WFD light curves, this trend has reduced and PELICAN is more robust at low S/N. This is probably due to the first step of non-supervised learning, where PELICAN has "seen" light curves with a low S/N, and the data augmentation that we performed. Indeed by adding different noises on input light curves, PELICAN has learned many noisy examples.

8.3. Peak magnitudes

The right panels of Fig. 12 show the accuracy as a function of the maximum value of peak magnitude from all bands. For the classification of DDF light curves, the accuracy decreases at low magnitudes above 26 due to the low number of examples in the training database in this magnitude range. However, PELICAN is robust at low magnitudes for the classification of WFD light curves. This robustness is due to the non-supervised learning during which PELICAN has learned a light curve representation at low magnitudes. Nevertheless, in this case, the accuracy decreases also at magnitudes below 23. This behavior may be due to the mismatch between DDF light curves that made up the training database and WFD light curves from the test database. Indeed DDF light curves have, on average, brighter magnitudes than light curves in the test database. To reduce the mismatch between the training and test databases, PELICAN performs a first non-supervised training on the test database. Nevertheless

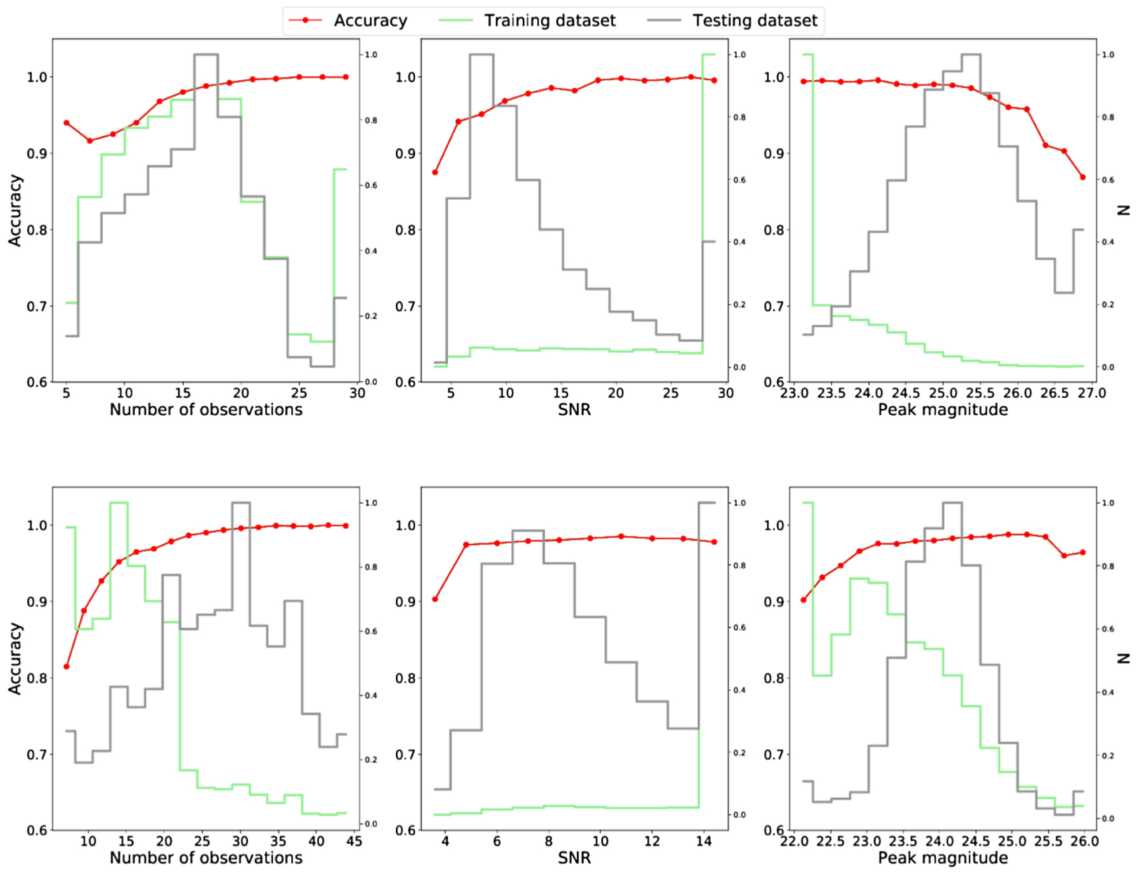


Fig. 12. *Upper panels*: classification of DDF light curves and *lower panels*: classification of WFD light curves. *Left panels*: accuracy as a function of the total number of observations on different dates for all bands. *Middle panels*: accuracy as a function of S/N, which is computed as the maximum S/N from all bands. *Right panels*: accuracy as a function of peak magnitude, which corresponds to the maximum value of peak magnitude from all bands. For each case the distribution of the training database is represented in light green and that of the test database in gray.

this step may cause a bias at bright magnitudes as PELICAN learned a representation of light curves at faint magnitudes from the WFD survey.

9. Summary and discussion

We presented a deep learning architecture for light curve classification, PELICAN. It performs several tasks to find the best feature representation space of light curves and classify them. In this work, we applied PELICAN to the analysis of supernovae light curves, but it can be applied to the analysis of other variable and transient objects. Our model is able to reduce the problem of non-representativeness between the training and the test databases thanks to the development of two modules. The first one uses a non-supervised autoencoder that benefits from light curves of the test set without knowing the labels in order to build a representative feature space. The second module optimizes a contrastive loss function adjusted to reduce the distance into the feature representation space between brighter and fainter objects of the same label.

PELICAN can also deal with the sparsity and the irregular sampling of light curves by integrating a sparsity parameter in the autoencoder module and performing an important data augmentation.

Our model reached the best performance ever obtained for the Supernovae Photometric Classification Challenge with a non-representative training database, with an accuracy of 0.861 and an AUC of 0.937 against 0.713 and 0.855, respectively, obtained by the BDTS algorithm and SALT2 features as shown in Lochner et al. (2016). These kind of feature-based algorithms do not overcome the problem of representativeness. Indeed, even if the features used are relevant, they are not representative of the test database as the spectroscopic follow-up is necessarily limited. Therefore this method offers poor performance in a real scenario such as we consider in this work, and have to be adapted.

In the context of LSST, it is important to confront PELICAN to the observational issues, in particular the uncertainties related to the two main programs of LSST, which are the Wide-Fast-Deep and the Deep Drilling Fields surveys. In this work we addressed several points:

– uncertainties related to the spectroscopic follow-up in the DDF survey. A subsample of light curves should be spectroscopically confirmed in the DDF survey but it might be very limited. PELICAN is able to reach good performance with a small training database (2k light curves) for which it detects 87.4% of SNe Ia with a precision comparable to the spectroscopic one.
– uncertainties related to the spectroscopic follow-up in the WFD survey. It is not certain that a sample of light curves will be spectroscopically confirmed in WFD fields. So it is crucial that PELICAN can classify SNe Ia observed on the WFD survey, with a training composed only of DDF light curves. By considering a training database of 2k–10k light curves, PELICAN is able to classify from 90.5% to 97.3% of SNe Ia with a precision higher than 98%. This result constitutes one of our major contributions as it opens the possibility of using SNe Ia from WFD fields for cosmology studies.

We also found that PELICAN is robust against an S/N above five and magnitudes below 26 for the classification of DDF light curves. The accuracy of PELICAN is very stable until redshift 1.0; above this value the number of examples in the training database is not sufficient, which explains the decrease at high redshifts. However, this tendency is significantly reduced if the training database contains at least 5k light curves. In this case, the accuracy is higher than 90% until 1.2 in redshift.

For the classification of WFD light curves the accuracy decreases at low redshifts and bright magnitudes, due to the mismatch between the training and test databases. Even if the step of non-supervised training on the test database reduces it, PELICAN learns more on low redshifts and faint magnitudes from the test database. It could be possible to reduce this bias by integrating spectroscopically confirmed light curves into the training of the autoencoder but it should be done carefully as DDF light curves have to be transformed to look like WFD light curves to avoid the mismatch. Nevertheless, PELICAN remains robust at low S/N for the classification of WFD light curves.

PELICAN depends on the number of observations on the light curve as it takes only this information as input. Nevertheless the sparsity term in the loss of the autoencoder and the data augmentation help to reduce the bias.

A caveat for the tests done with simulations is the low number of non-Ia templates available, which may underestimate the proportion of non-Ia supernovae that have similar light curves to Ia supernovae. This point will be addressed with more detailed simulators that will be available in the future.

Finally to complete validation of PELICAN, we tested it on real SDSS data. In this case there is a new mismatch that appears as we trained it on simulated data that do not reproduce well real SDSS data. We demonstrated that an additional fraction of 10 % of real light curves inside the training allows PELICAN to reach an accuracy of 86.8%. This is a very encouraging result for the classification of supernovae light curves as the spectroscopically confirmed light curves can be completed by simulated ones to increase the size of the training database and so be less dependant on the costly spectroscopic follow-up.

The success of PELICAN under realistic conditions with a training step on a small and biased database and a testing stage on light curves with different sampling and more noisy measurementss opens very promising perspectives for the classification of light curves of future large photometric surveys. Furthermore it constitutes, up to now, the most appropriate tool to overcome problems of representativeness on irregular and sparse data.

Acknowledgements. This work has been carried out thanks to the support of the OCEVU Labex (ANR-11-LABX-0060). We thank Rahul Biswas and Philippe Gris for useful discussions. We gratefully acknowledge the support of the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. Funding for the creation and distribution of the SDSS Archive has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Aeronautics and Space Administration, the National Science Foundation, the US Department of Energy, the Japanese Monbukagakusho, and the Max Planck Society. The SDSS Web site is (`http://www.sdss.org/`). The SDSS is managed by the Astrophysical Research Consortium (ARC) for the Participating Institutions. The Participating Institutions are the University of Chicago, Fermilab, the Institute for Advanced Study, the Japan Participation Group, The Johns Hopkins University, the Korean Scientist Group, Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy (MPIA), the Max-Planck-Institute for Astrophysics (MPA), New Mexico State University, University of Pittsburgh, Princeton University, the United States Naval Observatory, and the University of Washington.

References

Betoule, M., Kessler, R., Guy, J., et al. 2014, A&A, 568, A22

Biswas, R., Cinabro, D., & Kessler, R. 2017, simlib_minion, DOI: 10.5281/zenodo.1006719

Brunel, A., Pasquet, J., Pasquet, J., et al. 2019, Proceedings of the 2019 IS&T International Symposium on Electronic Imaging (EI 2019)

Charnock, T., & Moss, A. 2017, ApJ, 837, L28

Dai, M., Kuhlmann, S., Wang, Y., & Kovacs, E. 2018, MNRAS, 477, 4142

Domínguez Sánchez, H., Huertas-Company, M., Bernardi, M., Tuccillo, D., & Fischer, J. L. 2018, MNRAS, 476, 3661

du Buisson, L., Sivanandam, N., Bassett, B. A., & Smith, M. 2015, MNRAS, 454, 2026

Eldesokey, A., Felsberg, M., & Shahbaz Khan, F. 2018, ArXiv e-prints [arXiv:`1811.01791`]

Frieman, J. A., Bassett, B., Becker, A., et al. 2008, AJ, 135, 338

Gieseke, F., Bloemen, S., van den Bogaard, C., et al. 2017, MNRAS, 472, 3101

Guy, J., Astier, P., Baumont, S., et al. 2007, A&A, 466, 11

Hadsell, R., Chopra, S., & LeCun, Y. 2006, 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2, 1735

He, K., Zhang, X., Ren, S., & Sun, J. 2016, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770

Hinton, G. E. 2002, Neural Comput., 14, 1771

Hua, J., & Gong, X. 2018, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18 (International Joint Conferences on Artificial Intelligence Organization), 2283

Ishida, E. E. O., & de Souza, R. S. 2013, MNRAS, 430, 509

Jha, S., Riess, A. G., & Kirshner, R. P. 2007, ApJ, 659, 122

Karpenka, N. V., Feroz, F., & Hobson, M. P. 2013, MNRAS, 429, 1278

Kessler, R., Bernstein, J. P., Cinabro, D., et al. 2009, PASP, 121, 1028

Kessler, R., Conley, A., Jha, S., & Kuhlmann, S. 2010a, ArXiv e-prints [arXiv:`1001.5210`]

Kessler, R., Bassett, B., Belov, P., et al. 2010b, PASP, 122, 1415

Kingma, D. P., & Ba, J. 2014, CoRR, abs/1412.6980

Liu, G., Reda, F. A., Shih, K. J., et al. 2018, ArXiv e-prints [arXiv:`1804.07723`]

Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., & Winter, M. K. 2016, ApJS, 225, 31

LSST Science Collaboration (Abell, P. A., et al.) 2009, ArXiv e-prints [arXiv:`0912.0201`]

Mahabal, A., Sheth, K., Gieseke, F., et al. 2017, IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 2757

Möller, A., Ruhlmann-Kleider, V., Leloup, C., et al. 2016, J. Cosmol. Astropart. Phys., 12, 008

Nair, V., & Hinton, G. E. 2010, in Proceedings of the 27th International Conference on Machine Learning (ICML-10), eds. J. Fürnkranz, & T. Joachims (Omnipress), 807

Pasquet-Itam, J., & Pasquet, J. 2018, A&A, 611, A97

Pasquet, J., Bertin, E., Treyer, M., Arnouts, S., & Fouchez, D. 2019, A&A, 621, A26

Perlmutter, S., Aldering, G., Goldhaber, G., et al. 1999, ApJ, 517, 565

Phillips, M. M. 1993, ApJ, 413, L105

Polikar, R. 2006, IEEE Circuit Syst. Mag., 6, 21

Richards, J. W., Homrighausen, D., Freeman, P. E., Schafer, C. M., & Poznanski, D. 2012, MNRAS, 419, 1121

Riess, A. G., Filippenko, A. V., Challis, P., et al. 1998, AJ, 116, 1009

Sako, M., Bassett, B., Becker, A., et al. 2008, AJ, 135, 348

Sako, M., Bassett, B., Becker, A. C., et al. 2018, PASP, 130, 064002

Schmidhuber, J., Wierstra, D., & Gomez, F. 2005, Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI) (Morgan), 853

Scolnic, D. M., Jones, D. O., Rest, A., et al. 2018, ApJ, 859, 101

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, J. Mach. Learn. Res., 15, 1929

Szegedy, C., Liu, W., & Jia, Y. 2015, Computer Vision and Pattern Recognition (CVPR)

Tripp, R. 1998, A&A, 331, 815

van den Bergh, S. 1995, ApJ, 453, L55

van der Maaten, L., & Hinton, G. 2008, J. Mach. Learn. Res., 9, 2579

Varughese, M. M., von Sachs, R., Stephanou, M., & Bassett, B. A. 2015, MNRAS, 453, 2848

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. 2008, in Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08), eds. W. W. Cohen, A. McCallum, & S. T. Roweis (ACM), 1096