A CNN adapted to time series for the classification of Supernovae

Anthony Brunel, Johanna Pasquet, Jérôme Pasquet, Nancy Rodriguez, Frédéric Comby, Dominique Fouchez, Marc Chaumont

HAL Id: lirmm-01968489
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01968489

Submitted on 2 Jan 2019

A CNN adapted to time series for the classification of Supernovae

Anthony BRUNEL[1]*, Johanna PASQUET*[2]*, Jérôme PASQUET*[3]*, Nancy RODRIGUEZ*[1]*,*
Frédéric COMBY[1]*, Dominique FOUCHEZ*[2]*, Marc CHAUMONT*[1,4]

[1] **LIRMM, Université Montpellier, CNRS, France**

[2] **CPPM, Aix Marseille Université, CNRS/IN2P3, France**

[3] **IRSTEA, Université Montpellier, AMIS, Université Paul Valéry, France**

[4] **Université Nîmes, France**

{anthony.brunel, nancy.rodriguez, frederic.comby, marc.chaumont}@lirmm.fr

{pasquet, fouchez}@cppm.in2p3.fr

{jerome.pasquet}@univ-montp3.fr

Abstract

Cosmologists are facing the problem of the analysis of a huge quantity of data when observing the sky. The methods used in cosmology are, for the most of them, relying on astrophysical models, and thus, for the classification, they usually use a machine learning approach in two-steps, which consists in, first, extracting features, and second, using a classifier. In this paper, we are specifically studying the supernovae phenomenon and especially the binary classification "I.a supernovae versus not-I.a supernovae". We present two Convolutional Neural Networks (CNNs) defeating the current state-of-the-art. The first one is adapted to time series and thus to the treatment of supernovae light-curves. The second one is based on a Siamese CNN and is suited to the nature of data, i.e. their sparsity and their weak quantity (small learning database).

1.INTRODUCTION

Cosmologists are facing the problem of the analysis of a huge quantity of data when observing the sky. As an example, in a very close future (2022), the Large Synoptic Survey Telescope (LSST)[1] will produce terabytes of images of the sky per day. Cosmologists thus need automatic analysis algorithms to alert them when a cosmological phenomenon occurs, such as the explosion of a star (supernova). In order to exploit this huge amount of data, cosmologists and computer-science scientists work in collaboration on classification algorithms, and on benchmarking, for example through international competitions such as the Astronomical Classification Challenge form Kaggle[2].

In this paper, we are specifically studying the supernovae phenomenon and especially the binary classification "I.a supernovae versus not-I.a supernovae". The type I.a supernovae produce an extremely bright explosion which can be seen at a very far distance. The uniform intrinsic brightness of this kind of star allows to calculate distances and to understand the universe and the dark energy better. Roughly 10^3 supernovae have been discovered in the history of astronomy. According to [1], the LSST will allow discovering over ten million supernovae during its ten years survey. We thus have to anticipate and find methods that will ease both the processing and analysis of astronomical data.

From a practical point of view, when a zone of the sky is observed by a telescope during the night, an image is produced. The light flux from a region of interest in the image - where a phenomenon is occurring - is computed. Cosmologists repeat this calculation every night. Then they can build a time-series giving the light flux in function of the date. In practice, they are making time-series for specific bands: green, red, near-infrared and infrared.

In this work, and having in mind what will be produced by the LSST, we use these four times-series as input data. The set of four times-series is named a *light-curve*, and its duration is about 100 to 200 days. The Figure 1 gives an example of a supernova I.a light-curve. light-curves have irregular sampling and are sparse. This sparsity is due to the weather conditions which do not allow a regular observation, and also because the observed zone of the sky is not always the same each night.
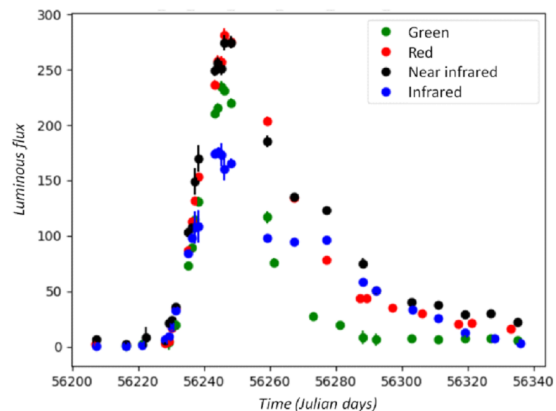
Figure 1. *A light-curve example of type I.a supernova.*

Until the end of 2017, cosmologist methods for classifying a light-curve as a supernova type I.a or not type I.a, are mostly relying on the use of astrophysical models. They usually use a two-steps machine learning approach: feature extraction and classification. Feature extraction is thus a crucial step, but even if features are chosen by experts, they can be incomplete or sub-optimal

---

[1] https://www.lsst.org/

[2] https://www.kaggle.com/c/PLAsTiCC-2018

for the classification step. In this paper, we present two Convolutional Neural Networks (CNNs) to replace the usual two-steps machine learning approaches. The first CNN is adapted to time series and thus to the treatment of supernovae light-curves. The second one is based on a Siamese CNN [2] and is adapted to the nature of data, i.e. their sparsity and their weak quantity (small learning database). We benchmarked our CNNs to two well established state-of-the-art methods, the first one by Lochner *et al.* [3], and the second one using *FATS features* of Nun *et al.* [4]. Additionally, we also evaluated our best CNN to a recurrent neural network (RNN) proposed by Charnock and Moss [5], which is also adapted to time series. Confronted by the strong sparsity of the data, our CNN performs better. Note that we took care to compare the algorithms in fair conditions, at low regime (small learning database), and also with hard condition close to real conditions, by having more than 70% of unavailable data in each time series.

This paper is organized as follows. In Section 2 we present the two state-of-the-art methods of supernovae classification. These machine learning algorithms use different feature descriptors; one relies on supernovae models and the other one on times series. We also present a deep learning method using a recurrent neural network [6]. In Section 3.1 our CNN and Siamese CNN architecture are presented. Then, in Section 4 we detail the databases, the experiments setup, the parameters settings, and analyze results. In Section 5 we conclude and give possible extensions of this work.

2.Existing methods

In this section, we present a small survey of recent machine learning-based methods for supernovae classification. The first method, proposed by Lochner *et al.* [3], uses *SALT2* features [7], and was considered as the state-of-the-art until the end of 2017. The second method relies on the use of a library named "Feature Analysis for Time Series" (FATS) whose features are dedicated to light-curves analysis [4]. Finally, we also present the work of Charnock and Moss [5] which uses a recurrent neural network.

2.1.Boosted decision trees using SALT2 features

In [3], the authors compare the results of different classifiers and features. They found that the boosted decision trees (BDT) with features of "Spectral Adaptive Light-curve Template features 2" (SALT2) model provided the best performance. SALT2 is the most commonly used model for type I.a supernovae. It relies on the following equation for computation of the light-curve's flux in a given band:

$$F(t,\lambda) = x_0 \times [M_0(t,\lambda) + x_1 M_1(t,\lambda) + ... + M_k(t,\lambda)] \times \exp[c \times CL(\lambda)], \qquad (1)$$

where t is the time since the maximum luminosity in B-band (the phase), λ is the wavelength, $M_0(t,\lambda)$ is the average spectral sequence, $M_k(t,\lambda)$ for $k > 0$ are higher order components that describe the variability of supernovae I.a. $CL(\lambda)$ represents the average color correction law. For each supernova, the redshift (variation of the wavelength due to the expansion of the universe) z is also used. The machine learning algorithm uses five parameters to describe the supernovae: z, t_0 (the time of peak brightness in the B-band), the normalization term x_0, x_1 that describes the shape of the light-curve, and c the color at the maximum luminosity in the B-band. Then, BDT are used to classify the supernovae. BDT are machine learning classifiers using multiple decision trees to construct a model. It associates input features to output classes. Multiple decision trees are built on slightly different subsets of data, and the resulting classifications are averaged to provide robustness. Instead of using bagging like in random forest (RF) [8], which selects subsets of data with random replacement, a BDT use boosting such that for each iteration the same data-set is used but with an increase of the weights of incorrectly classified examples, which allows subsequent classifiers to focus on difficult cases. Even if Lochner et al. [3] found that RF and BDT gave almost the same classification results, BDT were usually faster than RF. Moreover, BDT are considered as robust classifiers because of the averaging process [9].

2.2.Boosted decision trees using FATS library

In Lochner *et al.* [3], in addition to SALT2, several feature extraction methods are compared. Among them, there is a method that uses no prior knowledge on supernovae light-curves models, which makes it a generic approach. Nevertheless, similarly to what was observed in [3], our experimental results show that this wavelet-based method is less efficient than the approach of Lochner *et al.*. We thus choose another generic approach based on the FATS library [3] [4], which requires no previous supernovae knowledge and allow to extract many more features than SALT2. FATS (Feature Analysis for Time Series) is a python library which includes a set of features dedicated to time series analysis. FATS contains relevant information for the classification of astrophysical objects, such as the color which is the difference of the flux between two separate bands, the skewness, the mean, ... We then use BDT (presented in 2.1) to perform the classification.

2.3.Recurrent neural network for supernovae classification

Recurrent neural networks (RNN) are a class of deep learning method which exhibit a temporal dynamic behavior for a time sequence [6]. RNN are used for tasks such as speech recognition [10] and language translation [11]. In [5], the authors present a supernovae type I.a versus not-I.a classification algorithm using RNN with Long Short-Term Memory (LSTM) cells [12]. They used LSTM cells because more traditional RNN are unable to manage long-term information. By using the gating process, the LSTM architecture overcomes the vanishing gradient problem, which means that it is more able to learn on long time series while taking better into account the past values. Each LSTM cell is composed of three gates. The first one represents the forget gate that allows the network to remove the information transmitted by the previous cell. The second corresponds to the input gate which processes the input information at a given time. The last gate merges the information from the input gate and the output gate to feed the next cell of the LSTM network with a piece of new information. In [5], the authors tested multiple RNN LSTM architec-

---

[3]FATS is a python library, and it can be found on GitHub https://github.com/isadoranun/FATS

tures with different numbers of LSTM cells per hidden layer. The RNN architecture that gave the best results is unidirectional with two hidden layer and sixteen LSTM cells per layer.

3.NETWORK ARCHITECTURE

In the section, we present the most important characteristics of our convolutional neural network (CNN) and our Siamese network.

3.1.Convolutional neural network

In this section, we present our convolutional neural network (CNN). First, we explain its architecture, and then we detail some of the most important elements.

3.1.1.Network architecture

Our CNN is shown in Figure 2. The source code can be downloaded at `https://github.com/Anzzy30/SupernovaeClassification`. Our CNN takes as input a light-curve which is represented by a matrix of dimension $4 \times T$, where 4 stands for the height and T for the width. There are 4 rows (one for the green, one the red, one the near-infrared, and one for the infrared bands), i.e. 4 time-series, and the duration T of the light-curve is variable. The CNN contains eleven layers, and most of those layers are "inception" modules (described below), where the filters are modified to be 1D temporal filters (performed along the rows of the input). Poolings are also modified to be 1D. A time series is indeed usually represented as a 1D array, and this ensures the extraction of temporal information independently for each time series. 1D convolutions capture the input signals evolution over the time [13]. Layers 2, 5, 9 and 10 are inception modules with 1D convolutions and a stride of two pixels. This enables to divide the input width by two after each inception block.

We also introduce a depth-wise convolution (a 1D convolution in the depth axis, i.e. a convolution in the color axis) in the 7th layer to take into account the correlation between the bands. This color-convolution is done without padding, over the four columns in order to merge the time series and combine the information provided by each time-series. This fusion process is important for a more efficient separability between the two classes supernovae I.a and not-I.a [14]. We indeed observed an increase in the accuracy of the classification by using this color-convolution.

Each convolution operation is followed by a ReLU (Rectified Linear Units) activation function. At the end of last convolution layer, we perform a global max pooling as described in subsection 3.1.3. This "global max pooling" allows treating light-curves of any duration. Then, a fully connected network with 1024 neurons is used for the classification part. The network's output is finally provided by a softmax prediction between the two classes I.a supernovae and not-I.a supernovae.

3.1.2.Inception module

Our network is mainly built by a succession of inception modules [15]. These modules have been adapted to work with time series, i.e. 1D convolution. It is built with 1×1, 1×3 and 1×5 convolution layers and max pooling. It allows extracting information at multiple resolutions. 1×3 and 1×5 convolutions are preceded by 1×1 convolution to reduce the dimensionality and add non-linearity. In our network, each convolution layer is followed by a ReLU activation function [16]. The Figure 3 describes our inception module.
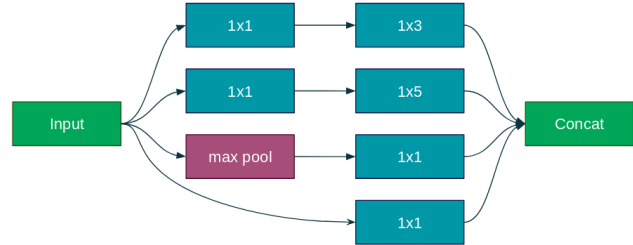

Figure 3. *Our 1D inception module.*

3.1.3.Global max pooling

The time series may have a variable duration, i.e. a variable number of columns for the input matrices. A classical CNN is generally built with a fully connected part, this means with a fixed number of neurons, which impose a fixed dimension for the inputs. Thus, for a classical CNN, a variable length of time series generate feature maps of variable lengths, which cannot be processed by the fully connected layer. To overcome this limitation, we incorporate a global max pooling layer (described in Figure 4) after the last convolution and just before the fully connected layer. This force the features to have a fixed dimension before entering the fully connected layer. Our network can thus classify light-curves of any duration. Another interesting effect of this measure is that during the learning phase, we can now sometimes crop the light-curves from 40% to 80% of their original duration. This allows a data-augmentation, reduces the over-fitting, and increases the generalization abilities.

3.2.Siamese network

The second method proposed in this paper is a Siamese network. This type of neural network was introduced in the 1990s by Bromley et al. [2] and contains at least two sub-networks with shared weights. Each sub-network produces an n-dimensional feature vector. Afterwards, we can compute different metrics like L2-norm between vectors. The learning process for this method is to bring closer, in the features space, elements that have the same label and drive away elements with a different label.

With the Siamese network we search for a solution to the sparsity problem which is extremely present in the data (See Fig. 1). We propose a loss function with the triplet loss presented in [17], and we also propose an adaptation of the triplet loss. First, each triplet is chosen online (online triplet mining) which means
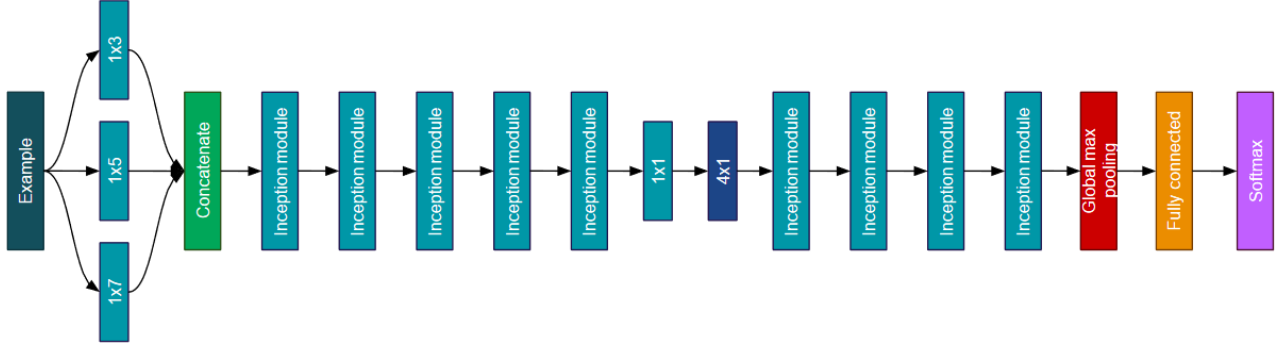
Figure 2. *Our convolutional neural network. There is three parallel temporal convolution followed by a depth concatenation. Then, there are five inception modules, whose two of them have strides of two for the width reduction. Next, there is the color convolution layer with a 1×1 filter, followed by a 4×1 filter. After, there are four inception modules, whose two of them have strides of two. Finally, there is the global max pooling, a 1024 fully connected layer, and the softmax layer which give the classification result.*
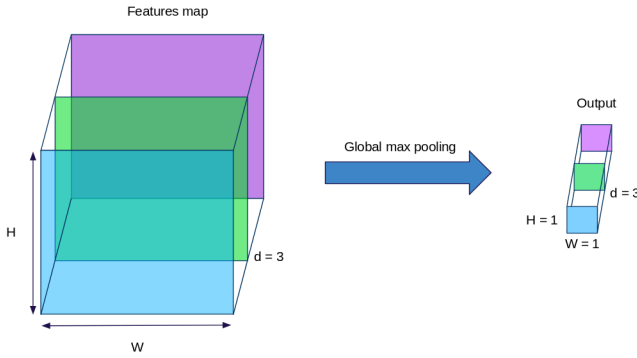

Figure 4. *Scheme of the global max pooling mechanism. Global max pooling perform an operation that take the maximum per features map and produce an 1D vector with known size (number of features map)*

that the useful triplets are computed on the fly. Online triplet mining was introduced in Facenet[4]. Online triplet mining is more efficient than offline regarding computation time and performance.

The loss function is described by Equations 2 and 3. First, we compute the classic triplet loss that allows the network to bring closer elements with the same label. The triplet loss is defined by the Equation 2,

$$L_{triplet}(a, p, n) = \max(d(a, p) - d(a, n) + margin, 0), \quad (2)$$

with d the L2 norm function which takes two vectors as input, $a \in \mathbb{R}^m$ a feature vector of dimension m, named the anchor, $p \in \mathbb{R}^m$ a feature vector named the positive example (it has the same label than the anchor), $n \in \mathbb{R}^m$ a feature vector associated to the negative example (it has a different label as that of the anchor), and the $margin \in \mathbb{R}$. This triplet loss, when minimized, has the effect of pushing the negative examples at a distance of the anchor greater than the margin plus the distance of the anchor to the positive.

During the optimization process, the triplet examples whose loss is greater than zero are the only triplets useful for the mini-

---
[4]http://bamos.github.io/2016/01/19/openface-0.2.0/

mization. The minimization is thus done on the arithmetic mean of the useful triplets; see Equation 3:

$$L_{final\_triplet} = \frac{1}{N-1} \sum_{j=0}^{N-1} L_{triplet_j}, \quad (3)$$

with N the number of useful triplet $L_{triplet_j}$.

In order to better take into account the sparsity of lightcurves, in addition to the triplet loss function, we propose an adaptation of the triplet loss described by Equation 4. The goal of this additional loss function is to amalgamate (= force an identical feature representation) the anchor features vector with the feature vector obtained by a sub-sampling of the anchor light-curve, while keeping a minimum distance with negative examples.

$$L_{triplet'}(a, a', n) = d(a, a') + \max(0, margin' - d(a, n)), \quad (4)$$

where $a \in \mathbb{R}^m$ is a feature vector associated to the anchor example, $a' \in \mathbb{R}^m$ is a feature vector of the sub-sampled anchor lightcurve, $n \in \mathbb{R}^m$ is a feature vector associated with the negative example (which is an example different from the anchor), and the $margin' \in \mathbb{R}$.

Once again, during the optimization process, the triplet examples whose loss is greater than zero are the only triplets useful for the minimization. The minimization is thus done on the arithmetic mean of the useful triplets; see Equation 5:

$$L_{final\_triplet'} = \frac{1}{N-1} \sum_{j=0}^{N-1} L_{triplet'}. \quad (5)$$

The final loss (Equation 6) is the sum of the two previous loss functions described by Equations 3 and 5. Figure 5 illustrates the learning process of the Siamese network.

$$L_{final} = L_{final\_triplet} + L_{final\_triplet'}. \quad (6)$$

To compute the features vectors of each example, we use two networks with shared weights and the same architecture as the convolutional neural network defined in section 3.1 but without the fully connected layer. Once the Siamese network converges, we train a fully connected layer of 1024 neurons followed

by a softmax layer. The input is the feature vector (output of the Siamese network), and the outputs are the two classes to predict (type I.a or not-I.a).
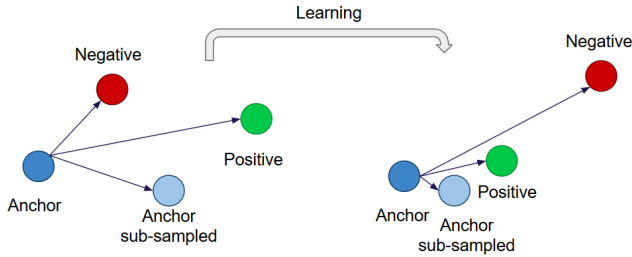

Figure 5. *Scheme of the learning process of our adapted triplet loss function; See the Equation 6. It aims (i) to amalgamate the Anchor feature vector with the Anchor feature vector issued from the sub-sampling of the same light-curve, (ii) to bring closer, without amalgamation, the Anchor vector and the Positive vector, and (iii) to keep distant the Anchor vector and the Negative vector. For example:* Anchor *is a supernova I.a,* Anchor sub-sampled *is the version of the Anchor resultant of a sub-sampling of same the light-curve,* Positive *is another supernova I.a, and* Negative *is a supernova not-I.a.*

4.EXPERIMENTS

In this section, we present the database used for the experiments, the parameter settings, the setup for the experiments, and finally, our results.

4.1.Base

4.1.1.First base

We simulate with the software SNANA [18] light-curves of different types of supernovae. SNANA 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. Each light-curve is composed of 4 time-series. Each time series contains the light flux measure in specific bands taken each day during a certain period of time (around 100 days). The bands are obtained using color band-pass whose filters are green, red, near-infrared, and infrared. In the first experiment, we used 5000 light-curves with 2500 supernovae of type I.a and 2500 of type not-I.a. For the deep learning input, each data is day-sampled and stored in a matrix. Each cell has a specific value of flux. If a value is missing, then we fill it up by zero values. As light-curves of supernovae are very sparse (due to missing values), the matrix contains more than 70% of zero values. We will note this database B1.

4.1.2.Second base

The second database is available on GitHub[5] and contains 21319 light-curves with 5088 I.a supernovae and 16231 not-I.a

supernovae. This database is from the Supernova Photometric Classification Challenge [19]. We used this database to compare our CNN (our first architecture; See Section 3.1) to the results obtained by the RNN LSTM of [5] (See Section 2.3). We will note this database B2.

4.1.3.Data augmentation

For the convolutional neural networks, we used artificial data augmentation. This method allows the networks to get a better representative set for the learning process. Each light-curve gets a chance to be altered during each epoch. The alteration will crop the light-curve and takes a random fraction between 0.4 and 0.8 of the light-curves. Then the network is fed with this new representation. This strategy reduces over-fitting and slightly increases the classification performances.

4.2.Parameter settings

For the machine learning methods, we used the algorithm available on Lochner's GitHub[6] with boosted decision trees, as presented in [3].

We set the number of iteration at 4500 for the convolutional neural network and 9000 for the Siamese network. We set the dropout to 0.4 for the fully connected layer to reduce over-fitting. The learning rate for the 2 deep learning methods varies between 1×10^{-2} and 5×10^{-4} with exponential decay. For the two methods, we used Adam optimizer [20] on a cross entropy loss function. Network's weights are initialized with Xavier uniform initializer algorithm [21] and the batch size is set to 128.

4.3.Experiments setup

We used tensorflow and python to develop our deep learning methods. The training phase is performed with an NVIDIA GTX 1080.

To compare the two methods relying on a two-steps machine learning approach with our convolutional neural network and our Siamese network, we used the Base B1 with k-folds cross-validation with k=4 and 5000 light-curves. The database is thus randomly partitioned into 4 equal sized sub-samples. A single sub-sample is kept for testing the model, and the remaining 3 sub-samples are used as training data. We repeat the process 4 times, with each of the k sub-samples used exactly once as the testing data. It means that each model is trained on only 3750 light-curves and tested on the 1250 other.

To confront our convolutional neural network against the recurrent neural network [5], we followed the same strategy as presented in [5] with the database B2. We thus trained our CNN model (our first architecture; See Section 3.1) on a base composed of 5330 supernovae light-curves took randomly from the database B2. The remaining elements in B2 (15989) are used to evaluate the network. We repeat this process five times to compute the arithmetic mean and the standard deviation of the results. Note that the number of training data is chosen to be close to the training set size used with the database B1.

4.4. Comparison metrics

We use two metrics to compare the different methods. The first is the accuracy defined as the ratio of the number of correct predictions over the total number of predictions. The second one is the Area Under the ROC Curve (AUC). The ROC curve represents the True Positives Rate (TPR) versus False Positives Rate (FPR) when the probability threshold is moved from 0 to 1. TPR and FPR are given by equations 7 and 8.

$$TPR = \frac{TP}{TP+FN} \quad (7) \qquad FPR = \frac{FP}{FP+TN}, \quad (8)$$

where TP are the true positives, FP the false positives, TN the true negatives and FN the false negatives.

4.5. Results and discussion

4.5.1. Database B1: Comparisons of our CNNs and our Siamese network, to SALT2, and FATS

In our first experiment with the database B1 we confront our CNN and our Siamese network to BDT + SALT2 [3], and BDT + FATS [4]. The Figure 6 shows the obtained ROC curves. The convolutional neural network gets the best performances with an AUC equal to 98.4% followed by the BDT with SALT2 features which obtains 97.9%. The Siamese network and the BDT using FATS obtain respectively an AUC equal to 96.3% and 96.4%.
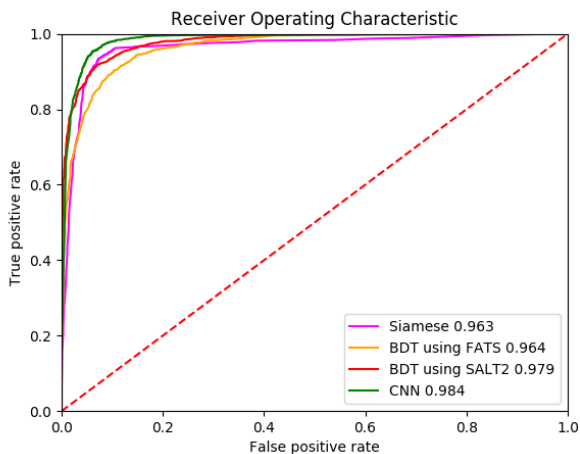

Figure 6. ROC curves of the four methods: Siamese, BDT using FATS, BDT using SALT2, and our CNN.

Another interesting metric is the accuracy. The CNN got the best accuracy of all 4 methods with 94.6% (See Table 1). Siamese network comes with 93.0% that outperforms BDT using SALT2 which obtains 92.3%. BDT using FATS get the lowest accuracy with 90.1%.

The state-of-the-art in two-steps machine learning methods used for supernovae classification [3], [4], perform well with a small learning set (i.e. less than 1 000 learning example), but with the above results, we can clearly say that CNNs are a promising

Model	Training set	AUC	Accuracy
CNN	3750	**0.984**	**94.6**
SALT2	3750	0.979	92.3
FATS	3750	0.964	90.1
Siamese	3750	0.963	93.0

Table 1. AUC and accuracy for the four methods (Siamese, BDT using FATS, BDT using SALT2, and our CNN) on B1 database.

solution for the binary classification of super-novae type I.a versus not-I.a. Indeed, our CNN performs better than two-steps machine learning methods which are based on hand-crafted features, and this with a small number of examples (3 750). A set of 3 750 examples is indeed considered as a very small database for a learning of a binary classification by deep learning methods.

The Siamese network gets good results but is harder to train because of the shared weights between its two networks. Even if its results are not as good as our CNN, we think the results can be improved with a better loss function. Adding some data augmentation, as we do with our CNN, may also improve the performances.

Additionally, we observed that with the increase in the size of the learning set (i.e. with more time series) the performances of our CNN are increasing. Another advantage of our method is that it learns the filters value directly to extract the more relevant features. There is thus no need of prior knowledge as in SALT2 [3] or FATS [4].

When working with a more complex architecture, we have shown that in a more realistic and more complicated situation from a cosmological point of view (extremely small database, a mismatch between learning set and test set), that Deep Learning approaches surpass all the previous known approaches [22].

4.5.2. Database B2: Comparisons of our CNN with an RNN LSTM

In the second experiment with the database B2, we confront our CNN against the recurrent neural network proposed in [5]. The input vector given to the RNN contains the day, the light flux in each band, and the redshift extra-information. The RNN thus uses a little bit more information compared to our CNN. In this experiment, 5330 supernovae light-curves are used for the learning, whereas in the first experiment 3750 light-curves were used. We choose 5330 light-curves (a fraction of 25% of the database) because it is close to the size used in our first experiment. In this experiment, the proportion of I.a and not-I.a is around 1/4, 3/4 whereas in the first experiment with B1 it was 1/2, 1/2. For our CNN the data-augmentation is done by cropping, and it artificially increases the learning data-set by 2 or 3. For the RNN, there is a random interpolation which artificially increases the learning data-set by 5. The data-augmentation for this experiment improves the results for both of the approaches of roughly 1% (with a smaller increase for our CNN). The comparison should then be considered more or less fair, with nevertheless a small disadvantage for our CNN.

The results (see Table 2) show that our CNN is superior to the RNN LSTM of [5] with 98.3% of AUC against 97.5%, and 94.1% of accuracy against 92.9%. Also, note that the standard deviation of the five runs is smaller for our CNNs. In [5] multi-

Model	Training set	AUC	Accuracy
CNN	5330	**0.983 ± 0.0004**	**94.1 ± 0.09**
RNN	5330	0.975 ± 0.003	92.9 ± 0.6

Table 2. AUC and accuracy for our CNN and RNN LSTM on B2 database.

ple experiments with a different fraction of the database are presented. Taking more data, for example, a fraction of 50% of the database, provides better results for both our CNN and the RNN, and the gap between our CNN and the RNN remains the same.

The recurrent neural networks are a great technique to treat time series as it allows to extract dynamic temporal behavior for a time sequence. However, in this work, we showed that our CNN is better. This is maybe due to the trend of over-fitting that is more present in the RNN. The various information of different nature, given to the RNN (day, light flux in each band, redshift), can also make information treatment harder.

Note that when taking less data, for example, a fraction of 5% (= 1065 light-curves) of the database, SALT2 method [3] gets the best results. When the size of the learning set is too small (less than 1000 light-curves), deep learning approaches are suffering from an insufficient number of examples. Measures can nevertheless be taken such as data-augmentation by noise addition, use of transfer learning or curriculum learning, use of cosmological parameters such as the red-shift, use of ensembles, use of multiple classes, etc. (see for example our paper under revision [22]).

5.CONCLUSION

The field of Cosmology is facing great challenges in order to be able to analyze a huge amount of data. One of those challenges is to be able to automatically detect I.a supernovae from not-I.a supernovae. In this paper, we propose a new CNN, adapted to time series (light-curve), that can defeat the state-of-the-art. We also compared our CNN to a Siamese approach and an RNN LSTM. Our CNNs gives the better results, nevertheless, the main conclusion is that all those deep learning approaches are very promising.

We observed strong efficiency improvement of our CNN when the learning database increases in size. We are also aware that using more clever data-enrichment could boost the efficiency of the different deep learning approaches, and especially our CNN, without substantial additional processing.

We believe that this publication will reinforce the use of deep learning in the cosmology and astronomy fields. We put all the necessary files for the deployment of our CNN on an open-access GitHub (`https://github.com/Anzzy30/SupernovaeClassification`), thus giving strong visibility of our work to the cosmology community.

References

[1] LSST Science Collaboration, P. A. Abell, J. Allison, S. F. Anderson, J. Andrew, and et al. Lsst science book, version 2.0. *ArXiv e-prints*, dec 2009.

[2] J. M. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems 6, 7th NIPS Conference*, pages 737–744, 1993.

[3] M. Lochner, J. D. McEwen, H. V. Peiris, O. Lahav, and M. K. Winter. Photometric supernova classification with machine learning. *The Astrophysical Journal Supplement Series*, 225(2):31, 2016.

[4] I. Nun, P. Protopapas, B. Sim, M. Zhu, R. Dave, N. Castro, and K. Pichara. FATS: Feature Analysis for Time Series. *ArXiv e-prints*, may 2015.

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

[6] L.R Medsker and L.C Jain. *Recurrent neural networks: design and applications*. CRC Press, 1999.

[7] J. Guy, P. Astier, S. Baumont, D. Hardin, R. Pain, N. Regnault, S. Basa, R.G. Carlberg, A. Conley, S. Fabbro, D. Fouchez, and others. Salt2: using distant supernovae to improve the use of type ia supernovae as distance indicators. *Astronomy & Astrophysics*, 466(1):11–21, 2007.

[8] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, oct 2001.

[9] T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.

[10] A. Graves, A.R. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[11] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH - Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[12] F. A. Gers, J. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, oct 2000.

[13] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection: A unified approach to action segmentation. In *European Conference on Computer Vision*, volume 9915, pages 255–257. Springer, 2016.

[14] A. Kovcs and I. Szapudi. Stargalaxy separation strategies for wise-2mass all-sky infrared galaxy catalogues. *Monthly Notices of the Royal Astronomical Society*, 448(2):1305–1313, apr 2015.

[15] C. Szegedy, W. andY. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[16] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.

[17] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[18] R. Kessler, J. P. Bernstein, D. Cinabro, B. Dilday, J. A. Frieman, S. Jha, S. Kuhlmann, G. Miknaitis, M. Sako, M. Taylor, and J. Vanderplas. Snana: A public software package for supernova analysis. *Publications of the Astronomical Society of the Pacific*, 121(883):1028, 2009.

[19] R. Kessler, B. Bassett, P. Belov, V. Bhatnagar, H. Campbell, A. Conley, J. A. Frieman, A. Glazov, S. Gonzalez-Gaitan, R. Hlozek, et al. Results from the supernova photometric classification challenge. *Publications of the Astronomical Society of the Pacific*,

122(898):1415–1431, 2010.

[20] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ArXiv e-prints*, 2014.

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256. PMLR, 2010.

[22] J. Pasquet, J. Pasquet, M. Chaumont, and D. Fouchez. Pelican: deep architecture for the light curve analysis. *Astronomy & Astrophysics*, 2019. Submited and under revision.

Author Biography

Anthony BRUNEL received his Master's degree in Computer Science in 2018 from the University of Montpellier, France. He is currently working toward the PhD degree in the LIRMM laboratory of Montpellier. His research interests are localization for vision purpose.

Johanna PASQUET received her PhD in 2016 from the University of Montpellier, France on the search of the baryonic dark matter in our Galaxy. She is currently working toward a postdoc in the CPPM laboratory of Marseille on a deep learning approach to observational cosmology with Supernovae. Her research interests are classification with Deep Learning, photometric redshifts and supernovae cosmology.

Jérôme PASQUET is Assistant Professor in computer science at the Université Paul Valéry Montpellier. He conducted a PhD thesis under the supervision of Marc Chaumont and defended it in November 2016 at LIRMM laboratory of Montpellier. His research interests are deep learning representation applied to the objects detection, steganalysis problems and time series classification.

Nancy RODRIGUEZ is Assistant Professor in Computer Science at the University of Montpellier, France, since 2006. She holds a bachelor's degree in physics and earned her master's degree in computer engineering and physics from Los Andes University, Colombia. She received his PhD in Computer Science from Paul Sabatier University of Toulouse, France. Her research interests include Virtual and Augmented reality, Visualization and Interaction.

Frédéric COMBY received his M.Sc. degree in automatic and microelectronic systems in 1998, and the PhD degree in automatic and signal processing in 2001 from the University of Montpellier, France. He joined the ICAR Team (image and interaction), in the LIRMM (Laboratory of Informatics, Robotics, and Microelectronics of Montpellier) as Assistant Professor in 2003. His current research topics include image processing, vision and multimedia security.

Dominique FOUCHEZ is Research Director at CPPM (Center of Particle Physics of Marseilles) CNRS, AMU (Aix Marseilles University), France. He received his PhD in particle physics in 1993 from Aix Marseilles University. His main research interests are fundamental physics and cosmology.

Marc CHAUMONT is Associate Professor (HDR Hors-Classe) accredited to supervise research since 2013, at the LIRMM laboratory (Laboratory of Computer Science, Robotics and Microelectronic), University of Montpellier and University of Nîmes, in France. He joined the LIRMM in September 2005. He received his PhD in Computer Sciences at IRISA Rennes in 2003. His main research interests are in steganography, steganalysis, digital image forensic, and objects detection with Deep Learning.