# The information of attribute uncertainties: what convolutional neural networks can learn about errors in input data

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 045019 https://doi.org/10.1088/2632-2153/ad0285

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
5 May 2023

REVISED
3 October 2023

ACCEPTED FOR PUBLICATION
11 October 2023

PUBLISHED
27 October 2023



PAPER

# The information of attribute uncertainties: what convolutional neural networks can learn about errors in input data

**Natália V N Rodrigues**[1,*], **L Raul Abramo**[1] **and Nina S T Hirata**[2]

[1] Instituto de Física, Universidade de São Paulo, R. do Matão 1371, CEP 05508-090 São Paulo (SP), Brazil
[2] Instituto de Matemática e Estatística, Universidade de São Paulo, R. do Matão 1010, CEP 05508-090 São Paulo (SP), Brazil
* Author to whom any correspondence should be addressed.

**E-mail:** natalia.villa.rodrigues@usp.br

**Keywords:** machine learning methods, statistical techniques, scientific data analysis

## Abstract

Errors in measurements are key to weighting the value of data, but are often neglected in machine learning (ML). We show how convolutional neural networks (CNNs) are able to learn about the context and patterns of signal and noise, leading to improvements in the performance of classification methods. We construct a model whereby two classes of objects follow an underlying Gaussian distribution, and where the features (the input data) have varying, but known, levels of noise—in other words, each data point has a different error bar. This model mimics the nature of scientific data sets, such as those from astrophysical surveys, where noise arises as a realization of random processes with known underlying distributions. The classification of these objects can then be performed using standard statistical techniques (e.g. least squares minimization), as well as ML techniques. This allows us to take advantage of a maximum likelihood approach to object classification, and to measure the amount by which the ML methods are incorporating the information in the input data uncertainties. We show that, when each data point is subject to different levels of noise (i.e. noises with different distribution functions, which is typically the case in scientific data sets), that information can be learned by the CNNs, raising the ML performance to at least the same level of the least squares method—and sometimes even surpassing it. Furthermore, we show that, with varying noise levels, the confidence of the ML classifiers serves as a proxy for the underlying cumulative distribution function, but only if the information about specific input data uncertainties is provided to the CNNs.

## 1. Introduction

Machine learning (ML) methods are becoming increasingly popular in the analysis of scientific data sets, especially in areas with large volumes of data such as high energy physics and astrophysics—see, e.g. [1–4]. Typically, scientific data consists of individual measurements, each one with an associated uncertainty attached to it—i.e. each input data point is assigned some probability distribution function (PDF) to represent the underlying noise distribution. In astronomy, typical attributes of an object include fluxes, magnitudes and colors, together with their corresponding nominal errors.

Traditional statistical techniques (e.g. Fisherian/frequentist or Bayesian methods) weigh the data according to their specific uncertainties in order to derive constraints and draw conclusions on the basis of those data sets [5, 6]. Template fitting techniques, for example, employ both the data and the uncertainties to find a best-fit model for each object. Nevertheless, despite the deep connections between ML and statistical inference [7], measurement errors are routinely discarded in ML applications, even in the physical sciences [8]. In this paper we show the value of including the information content of the noise in convolutional neural networks (CNNs), by quantifying the improvements in the performance of ML classifiers and comparing them against a baseline maximum likelihood method.

The classification of astrophysical unresolved sources typically relies on the identification of distinguishing features in their spectra—we do not consider here the obvious case of galaxies or other

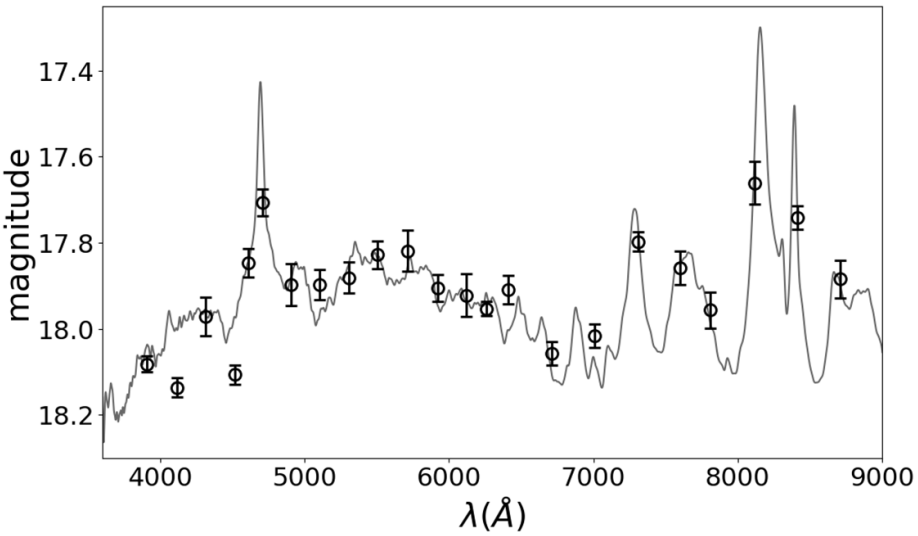


**Figure 1.** Spectrum (gray line) and pseudo-spectrum (black dots) of a galaxy. Each point corresponds to a magnitude at an effective wavelength of a ∼100 Å width bin.

spatially resolved objects which are trivial to classify. Some emission lines, such as H$\alpha$, H$\beta$ and OII, as well as spectral features such as the 4000 Å break, can help differentiating between types of galaxies, for example. In order to bypass the high cost of the acquisition of spectra for large numbers of sources, imaging surveys aim at capturing some of those features using photometry. Some surveys employ broad-band (BB) filters, which yield only a smeared-out version of the spectral features, while other surveys also rely on narrow-band filters [9–15], which allow for a more detailed measurement of those features, resulting in the so called *pseudo-spectra*. Narrow-band survey data avoids some of the loss of information about relevant local features, such as emission or absorption lines, that characterize different astrophysical objects (stars, quasars, emission line galaxies, etc). In all these cases, however, each object is assigned a series of measurements (magnitudes or fluxes at different wavelengths), together with their uncertainties. Figure 1 shows a pseudo-spectrum of a galaxy (black circles with error bars) and the corresponding high-resolution spectrum (gray line, without error bars for clarity). In this example, each point corresponds to measurements of the flux in wavelength intervals of ∼100 Å width. The data points, expressed here as magnitudes, constitute the set of features that characterize the object. Notice, in particular, that the data errors are all different from each other, and cannot be inferred from the values of the data.

The first problem we face when detecting an unknown source is how to classify that source, and in that respect ML methods are becoming increasingly popular. CNNs, in particular, are excellent tools that allow us to extract local features from input data. In this sense, these networks can identify emission and absorption lines in spectra and pseudo-spectra, distinguishing between the different types of sources—see, e.g. [16–19]. However, very often the errors in input data are discarded in those applications. In this paper we show how much information is lost by neglecting those uncertainties, and we present some techniques that can be employed to retrieve that information.

Scientific data sets can be highly complex to analyze, and it is not always clear what is the maximal amount of information that can be extracted from them, or how much of that information is being exploited by ML methods [20]. As a result, it is often difficult to compare the performance of different methods or to assess improvements in the techniques. In order to circumvent this difficulty, we created a toy model that is based on data with noise that is fully described by distributions that are known *a priori*, for each feature of the data set. Moreover, the model parameters (which determine the class of the objects) are also drawn from Gaussian distributions, which means that we are able to quantify exactly the amount of mixing (confusion) between the classes. These underlying distributions form the basis for our comparison between methods, allowing us to quantify the improvements we achieve by providing the uncertainties in the measurements to the CNNs. In this work we focus on the analysis with the toy-model. However the techniques here presented have already been applied in [21] to classify quasars, stars and galaxies using a realistic mock catalog of pseudo-spectra from a photometric narrow-band survey [22].

The main issue we address in this paper is the value of the information contained in the noise associated with data (i.e. irreducible aleatoric uncertainty), and to what extent ML methods can learn about how to incorporate the information content about that noise. Clearly, if all the data points have exactly the same

noise level, $x_i \rightarrow \bar{x}_i \pm \sigma$ (where $\bar{x}_i$ is the true value of that data point and $\sigma$ is the variance of the noise PDF), then there is zero additional information to be gained by explicitly providing that information to an algorithm. However, if different data points have different levels of uncertainties, $x_i \rightarrow \bar{x}_i \pm \sigma_i$, then each point contributes with a different weight to the determination of parameters. The former case is known as homoscedastic errors, while the latter case corresponds to heteroscedastic errors—see, e.g. [23]. It is clear, from both a frequentist or a Bayesian viewpoint, that we should keep track of the different levels of signal and noise in data. The question is, then, what is the value of that information, and to what extent are the different ML techniques able to take it into account?

The issue of attribute uncertainties in ML models has been addressed in different contexts – for recent reviews, see [24, 25]. Some early papers considered situations where heteroscedastic noise is a random variable which can be estimated nonparametrically using ML methods [26, 27]. For more recent applications using random forests, see [28]; for Gaussian processes, see [29]; for support vector machines, see [30]; and for neural networks, see [31].

In this work we investigate the value of including uncertainties in the context of classification with CNNs using as input sequence-like data. It is important to stress that we are not trying to fit a curve. We assume that the error bars from each measurement are known, as in typical astronomical survey catalogs, and we explicitly use these values as inputs to the CNNs, as additional features. We quantify how much information is added by the errors, by using different levels of noise, and we quantify precisely how the output of the CNN changes as the input data transitions from the homoscedastic (input data with identical uncertainties) to the heteroscedastic (different, but known, uncertainties for each data point) regime. We also investigate the case where the noise is sampled from Poisson distribution—and therefore some information about the uncertainty is already present in the measurement itself. In particular, we show that CNNs are able to learn about the context of the information in the specific noise levels of the data, in such a way that they approach (and sometimes even surpass) the performance of the maximum likelihood approach.

The paper is organized as follows. In § 2 we describe our toy model, the smiley–frowny data set. In § 3 we describe how we add Gaussian noise (error-bars) to the smiley–frowny data and perform a Fisher information analysis of the inclusion of these uncertainties. In § 4 we describe the CNNs and the techniques to account for the uncertainties in these ML models. In § 5 we compare the accuracy obtained when classifying the smiley–frowny data with the different CNNs, which do and do not include the uncertainties, with a least squares maximum-likelihood approach and with the Fisher analysis. In § 6 we present a modified version of the smiley–frowny data where we generate noise with Poisson distributions and once again compare the CNNs, evaluating how much information is added by the error-bars. We complement our analysis in § 7 by applying our ML classifiers to the publicly available waveform data set. Finally, in § 8 we discuss our main conclusions and further applications.

## 2. The smiley–frowny data set

Our toy model consists of two simple classes of objects: parabolic curves with positive and negative concavities. Hence, we have a binary classification problem where the positive and negative classes are convex ('smiley', $\smile$) and concave ('frowny', $\frown$), respectively. The basic idea is that each object is represented by a set of $n$ data points (or features), in such a way that each data point has an uncertainty that derives from some known PDF. These uncertainties may be called the 'error bars' of the measurements, which for our purposes can be thought of as the variances (second central momenta) of the PDFs. We discuss these uncertainties in detail in the next section.

Figure 2 shows examples of curves generated with the smiley–frowny model. The smiley and frowny classes represent types of sources to be distinguished between each other, e.g. galaxies, stars, etc. The $n$ points in the curves represent the fluxes or magnitudes measured at $n$ effective wavelengths, i.e. the bins shown in figure 1.

Since both the underlying model (parabolic curves) and the PDFs of the model parameters are precisely known, we are able to classify each object using a statistically rigorous maximum likelihood approach. As discussed throughout this section, the model is constructed in such a way that the classes are uniquely distinguishable with the parameter related with the concavity of the curves. Therefore, the objects can be classified with a least squares optimization method that find the best fit for this parameter. Moreover, we are able to determine exactly the confidence of the least squares classification of smiley and frowny objects by using either an analytical Fisher matrix approach or by employing the likelihood function, which can be determined by means of a Markov chain Monte Carlo (MCMC) exploration of parameter space. The overall performance of the classification, as well as the confidence of the output for each object, can then be compared with ML models which do and do not include the input data uncertainties.

Our task is, therefore, to label curves as a smiley or a frowny in a binary classification scheme:

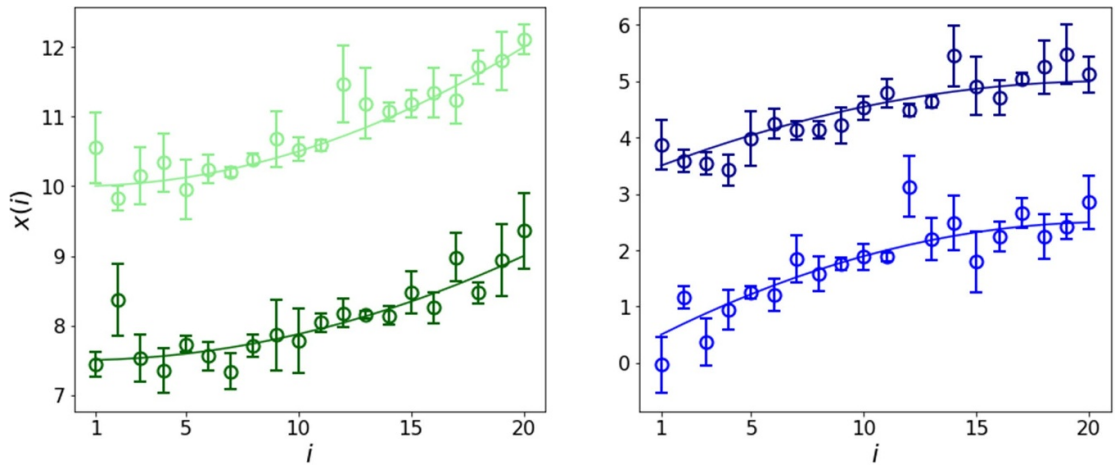


**Figure 2.** Parabolic curves generated according to the smiley–frowny model. The two lines in the left (green and dark green) are 'smiley' objects, and the lines in the right (blue and dark blue) are 'frowny' objects. The vertical shifts (parameter $c$) of the curves shown were fixed for visualization purposes. The noisy measurements of the features of those objects are shown as data points with error bars. Each data point has a different, but known, PDF whose properties are summarized by the variances expressed in the error bars.

- Positive class (1): $\smile$
- Negative class (0): $\frown$.

The parabolic curves are generated according to the model:

$$\bar{x}_{\smile}(i) = a_{\smile}\left(\frac{i}{n}\right)^2 + b_{\smile}\left(\frac{i}{n}\right) + c_{\smile} \tag{1}$$

$$\bar{x}_{\frown}(i) = a_{\frown}\left(\frac{i-n-1}{n}\right)^2 + b_{\frown}\left(\frac{i-n-1}{n}\right) + c_{\frown}, \tag{2}$$

where the indices $i = 1, 2, \ldots, n$, with $n$ being the number of measurements, which are the attributes, or features, that characterize the curves. The parameters $a$, $b$ and $c$ are drawn from normal distributions with means and standard deviations specified in table 1. The random nature of the parameters ensures that we have a variety of objects in each class, and the two PDFs for the curvature parameter $a$ are sufficiently separated (four $\sigma$'s) that the probability that an object sampled from the distribution of one class has the sign of the other class is $3.17 \times 10^{-5}$, which is irrelevant for the purposes of our discussion.

With the definitions of equations (1) and (2), both curves grow by the same amount from start to end: $\bar{x}(i = n) - \bar{x}(i = 1) = b(n-1)/n \pm a(n^2-1)/n^2$, where the plus and minus signs refer to the positive (convex, smiley) and negative (concave, frowny) classes, respectively. Since the distributions of the curvatures are anti-symmetric, $a_{\smile} \leftrightarrow -a_{\frown}$, all curves on average rise by the same amount from $i = 1$ to $i = n$, which further mixes the two classes. This feature of the model ensures that the only significant distinction between smiley and frowny objects is the concavity of the curves. Hence, the least squares method, which fits the model presented in equations (1) and (2) to the data, and classifies the objects based on the value obtained for the parameter '$a$', is incorporating all the available information to distinguish between the classes and can thus be used as a baseline model.

In this work we design our problem as a classification task, where the labels are well known. However, the smiley–frowny toy-model can also be used in a regression problem by, e.g. training a machine to predict the value of the parameter $a$ of each curve, instead of simply assigning classes. This would be more similar to the way we are applying least squares, i.e. to first fit the parameters of a model and than assign a class based on this fit. However, we decided to train the ML model to directly classify the data because, in practical applications, finding a parametric model such as equation (1) is usually unfeasible.

In astronomy we often face situations where the classes are not well defined, for example when the classification is based on BB photometry. One could also use this toy-model to investigate the problem of noisy labels by changing the distributions of $a_{\smile}$ and $a_{\frown}$ in such a way to have a relevant level of overlapping and thus some confusion between the assigned classes. Finally, we stress that here we only add noise (error-bars) in the vertical axis of figure 2, i.e. there is no mixing between the features $i$ (the wavelength bins).

**Table 1.** Parameters of the normal distributions from which the coefficients of the parabolic curves are sampled.

| Coefficient | $a_{\smile}$ | $a_{\frown}$ | $b$ | $c$ |
|---|---|---|---|---|
| $\mu$ | 1 | −1 | 0 | 10 |
| $\sigma$ | 0.25 | 0.25 | 1 | 3 |

## 3. Noise and information in input data: a toy model

In this section we describe how we add uncertainties to the smiley–frowny data and perform a Fisher analysis to evaluate how much information can potentially be added by these uncertainties when distinguishing between the two types of curves.

### 3.1. Adding noise

Scientific data sets are comprised of measurements which are performed with the help of instruments with some nominal uncertainties. However, those uncertainties are usually not fixed for all time: even with the same instrument, some measurements may have higher or lower uncertainties depending on several conditions. Any experiment will carefully assess what those uncertainties are for each data point, taking into account the different circumstances under which those measurements were made—see, e.g. [32].

To be clearer, one can think of two main sources of aleatoric uncertainties. The first is the quality, or the nominal sensitivity, of the apparatus used to perform the measurements. In the context of astronomy, the nominal uncertainty is determined by the size of the telescope's mirror and the sensitivity of the detectors, among other factors. The second source arises from the different conditions under which the measurements are made by the same apparatus. Some nights are brighter than others, some objects appear close to bright sources of light, and so on and so forth, meaning that different images, as well as different parts of the same image, have varying degrees of data uncertainty.

In this work we construct a simple model to reproduce these varying degrees of uncertainty in scientific data. First, we assume that the nominal accuracy of the measuring instrument is given in terms of a parameter $\sigma_0$, meaning that under some 'ideal' conditions for that instrument, the measurements are random numbers that follow a normal distribution with variance ${\sigma_0}^2$. And second, we introduce parameters $g_i$ that follow a uniform distribution, in such a way that the actual measurements $x_i$ (now under varying conditions) have uncertainties given by $g_i\sigma_0$. In other words, the overall (mean) accuracy of the measurements is reflected in the parameter $\sigma_0$, while the variability of individual measurements within that data set is given by the distribution of the $g_i$.

In our data noise model we have, therefore:

$$x_i = \bar{x}_i + \delta x_i, \tag{3}$$

where $\bar{x}_i$ are the true values of the measurements and $\delta x_i$ are random numbers sampled from Gaussian probability distribution functions with zero mean and variance $g_i^2\sigma_0^2$, i.e.:

$$p(\delta x_i) = \frac{1}{\sqrt{2\pi\, g_i^2\,\sigma_0^2}}\, \mathrm{e}^{-\frac{1}{2}\frac{\delta x_i^2}{g_i^2\sigma_0^2}}. \tag{4}$$

Here $g_i$ are numbers that are *known* for each individual measurement: one can think of a label for each data point indicating the degree to which the uncertainties are higher or lower than the nominal ones. In order to simulate the varying conditions under which those measurements are performed, we draw the factors $g_i$ from a uniform distribution in the interval:

$$g_i \in [\bar{g} - \Delta g, \bar{g} + \Delta g]\,,$$

where $\Delta g$ is the noise dispersion parameter, and the expectation value (mean) of that parameter given the uniform distribution is $\langle g_i\rangle_{\mathrm{U}} = \int \mathrm{d}g\, g\, U(g) = \bar{g}$, where $U(g) = 1/(2\Delta g)$ for $\bar{g} - \Delta g \leqslant g \leqslant \bar{g} + \Delta g$, and $U(g) = 0$ if $g < \bar{g} - \Delta g$ or $g > \bar{g} + \Delta g$ (clearly, $0 \leqslant \Delta g < \bar{g}$). For $\Delta g = 0$ we have a homoscedastic dataset, and as this parameter grows, the degree of heteroscedasticity increases. However, we stress the fact that the factors $g_i$ are *known* and, as opposed to the Gaussian random process underlying the noise, the values $g_i$ should not be regarded as stochastic variables in a fundamental sense: they are part of the information of the data set, and can be passed on to the ML methods.

Since the two distributions are uncorrelated by construction, the mean variance of the data errors can be easily computed:

$$\langle \delta x_i^2 \rangle_{\mathrm{U,G}} = \langle g_i^2 \rangle_{\mathrm{U}}\, \sigma_0^2 = \left(1 + \frac{1}{3}\frac{\Delta g^2}{\bar{g}^2}\right) \bar{\sigma}_0^2\,, \tag{5}$$

where $\bar{\sigma}_0 = \bar{g}\,\sigma_0$ denotes the mean nominal noise. This simple result tells us that when there are varying levels of noise in data, as described by this model, then the mean noise of the ensemble is actually *higher* than the mean nominal noise $\bar{\sigma}_0$.

### 3.2. Maximum likelihood and fisher information analysis

Traditional statistical tools for data analysis are naturally equipped to deal with different levels of noise in input data. In particular, the likelihood function is given by:

$$L = N \exp\left[-\frac{1}{2}\sum_i^n \left(\frac{x_i - \bar{x}_i}{g_i\,\sigma_0}\right)^2\right], \tag{6}$$

where $N$ is some normalization and $\bar{x}_i$ is the expectation value of the variable $x_i$ (or, in this context, the 'theory' that we would like to fit to the data). The term inside parenthesis in equation (6) is the usual $\chi^2 = \sum_i (x_i - \bar{x}_i)^2/\sigma_i^2$, where $\sigma_i = g_i\sigma_0$. The least squares solution maximizes the likelihood by minimizing the $\chi^2$. Once the best parameters $a, b, c$ are obtained, we assign the class by evaluating the parameter $a$. If the least squares fits $a > 0$ ($a < 0$), the object is classified as smiley (frowny).

In scientific applications we usually assume the theory to depend on a set of parameters denoted by the vector $\theta^\mu$ through some model, $\bar{x}_i(\theta^\mu)$—in our example, those parameters are $\theta^\mu = \{a, b, c\}$, so $\mu = 1, 2, 3$.

The likelihood function tells us which regions in parameter space are preferred, given the model, the data, and the uncertainties (or, more generically, the data covariance). Although a more thorough exploration of the likelihood function in parameter space is usually carried out using Markov chains generated via a Monte Carlo algorithm (in that respect, see section 5), we can estimate the shape of the likelihood using a Gaussian approximation, in which case the logarithm of the likelihood is a quadratic function. The curvature of that multivariate quadratic function at the peak (maximum likelihood) is the Fisher information matrix, which is computed by means of the Hessian:

$$F[\theta^\mu, \theta^\nu] = -\left\langle \frac{\partial^2 \log L}{\partial\theta^\mu\,\partial\theta^\nu} \right\rangle. \tag{7}$$

The inverse of the Fisher matrix yields an estimate of the parameter covariance, $\mathrm{Cov}[\theta^\mu, \theta^\nu] \to \{F[\theta^\mu, \theta^\nu]\}^{-1}$—see, e.g. [6] for many examples and applications in the physical sciences.

For the likelihood function of equation (6) we obtain the Fisher matrix:

$$F[\theta^\mu, \theta^\nu] = \sum_i^n \frac{\partial_\mu \bar{x}_i\, \partial_\nu \bar{x}_i}{g_i^2\,\sigma_0^2}, \tag{8}$$

where $\partial_\mu(\cdots) = \partial(\cdots)/\partial\theta^\mu$, and we assume that the data is both unbiased, $\langle x_i \rangle = \bar{x}_i$, and that the 'measurements' do not depend on the parameters, $\partial_\mu x_i = 0$ (the *theory*, on the other hand, obviously does: $\partial_\mu \bar{x}_i \neq 0$).

At this point we can take the expectation value over the uniform distribution of the noise dispersion to obtain the mean Fisher matrix:

$$\bar{F}[\theta^\mu, \theta^\nu] = \sum_i^n \left\langle \frac{1}{g_i^2} \right\rangle_U \frac{\partial_\mu \bar{x}_i\, \partial_\nu \bar{x}_i}{\sigma_0^2} = \frac{1}{1 - (\Delta g/\bar{g})^2} \sum_i^n \frac{\partial_\mu \bar{x}_i\, \partial_\nu \bar{x}_i}{\bar{\sigma}_0^2}. \tag{9}$$

We recognize the sum on the right-hand-side as the Fisher matrix for a nominal uncertainty $\bar{\sigma}_0$. Therefore, the amount by which different data points may have different uncertainties (expressed by the noise dispersion parameter $\Delta g$) has the effect of *increasing* the Fisher matrix with respect to the case where the noise is fixed to the nominal value—at least for a uniform distribution of those input uncertainties.

In other words: when each data point has a different (but known) uncertainty, even though the mean noise level is *higher*, the Fisher information actually *increases*. This happens, of course, because the specific noise levels in the data are acting as weights: noisier data are down-weighted, and less noisy data are up-weighted, resulting in higher discriminatory power. Here we are simply restating the fact that we always lose information if we do not distinguish between low-noise and high-noise measurements. This statement remains true when applied to ML techniques.

Our toy model for the smiley/frowny objects has a very simple, analytical Fisher matrix. Using equation (1) into the Fisher matrix of equation (9) we obtain:

$$\bar{F}[\{a,b,c\}] = \frac{1}{\left[1-(\Delta g/\bar{g})^2\right]\bar{\sigma}_0^2}\sum_{i=1}^{n}\begin{pmatrix} (i/n)^4 & \pm(i/n)^3 & (i/n)^2 \\ \pm(i/n)^3 & (i/n)^2 & \pm(i/n)^1 \\ (i/n)^2 & \pm(i/n)^1 & (i/n)^0 \end{pmatrix}, \tag{10}$$

where the plus and minus signs correspond to the smiley and frowny objects, respectively. All terms in this matrix have a closed form, given by the sum rules:

$$\sum_i^n i^0 = n \tag{11}$$

$$\sum_i^n i^1 = \frac{n(n+1)}{2} \tag{12}$$

$$\sum_i^n i^2 = \frac{n\left(2n^2+3n+1\right)}{6} \tag{13}$$

$$\sum_i^n i^3 = \frac{n^2\left(n^2+2n+1\right)}{4} \tag{14}$$

$$\sum_i^n i^4 = \frac{n\left(n^4+15n^3+10n^2-1\right)}{30}. \tag{15}$$

From these expressions we can compute the mean uncertainty in the parameter $a$—which is exactly the same for the two classes, since they are symmetric in the sense that $a_\smile \leftrightarrow -a_\frown$. The inverse of the Fisher matrix is an analytical approximation for the covariance matrix, whose diagonal term corresponding to the parameter $a$ yields the result:

$$\Sigma_a^2 = \left(\bar{F}^{-1}\right)_{aa} = \bar{\sigma}_0^2\left(1-\frac{\Delta g^2}{\bar{g}^2}\right)\frac{180\,n^3}{n^4-5n^2+4}. \tag{16}$$

Here we defined the Fisher-based approximation for the uncertainty of the parameter $a$ by $\Sigma_a$, which should not be confused with the width of the parent distribution for that parameter, $\sigma_a$, that expresses the intrinsic (true) diversity of objects in our classes. As discussed above, the uncertainty in the class of the objects is lower when the input data has varying levels of noise. Furthermore, with only two points ($n=2$) the class is completely undetermined—as it should, since with two points it is impossible to derive the curvature. For larger values of $n$ the uncertainty scales as $\Sigma_a \sim 1/\sqrt{n}$.

The approximate variance obtained above can be used in a normal distribution for the parameter $a$, and integrated for positive and/or negative values, yielding the probability that an object is either in the positive or negative class. E.g. the probability that an object has $a>0$ is given by:

$$\begin{aligned} P(a>0) &= \int_{-\infty}^{\infty} \mathrm{d}a \frac{1}{\sqrt{2\pi\,\sigma_a^2}} \mathrm{e}^{-\frac{1}{2}\frac{(a-\mu_a)^2}{\sigma_a^2}} \int_0^{\infty} \mathrm{d}a' \frac{1}{\sqrt{2\pi\,\Sigma_a^2}} \mathrm{e}^{-\frac{1}{2}\frac{(a'-a)^2}{\Sigma_a^2}} \\ &= \frac{1}{2}+\frac{1}{2}\mathrm{Erf}\left[\frac{\mu_a}{\sqrt{2\left(\sigma_a^2+\Sigma_a^2\right)}}\right], \end{aligned} \tag{17}$$

where $\mathrm{Erf}(z) = 2/\sqrt{\pi}\int_0^z \mathrm{d}t\mathrm{e}^{-t^2}$ is the error function, and $\mu_a$ and $\sigma_a$ are, respectively, the central value and variance of the distribution for the parameter $a$—see table 1. The expression above is therefore an approximate expression for the average confidence of the likelihood-based classification—and, of course, it also expresses the cumulative distribution function at the value $\mu_a$.

In figure 3 we plot the probability of equation (17) for $\mu_a = 1$ (smiley class), as a function of $\Delta g$. In this example we used $\bar{g}=1$ and each object has $n=20$ features (data points). From top to bottom, the curves correspond to increasing values of the nominal input error, $\bar{\sigma}_0 = \sigma_0 = 0.25$, 0.5, and 1.0. For very high uncertainties in the input data ($\sigma_0 \gg 1.0$) the probability approaches 0.5, which means zero confidence in the classification. That confidence grows as we lower the nominal uncertainty and/or if we increase the noise dispersion parameter $\Delta g$.

However, as much as the calculations above are able to provide insights into the problem at hand, we would be misguided if we attempted to use equation (17) to infer the confidence for a likelihood-based

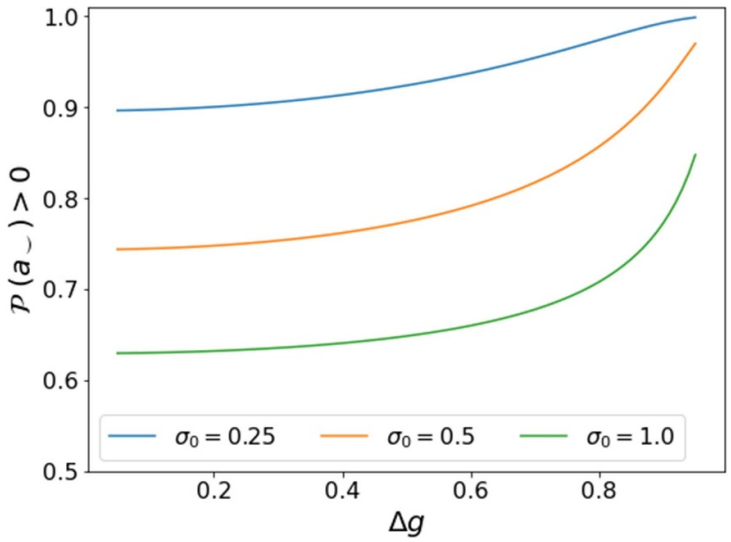


**Figure 3.** Mean probability that smiley objects are correctly classified $\mathcal{P}(a_\smile) > 0$ as a function of the noise dispersion parameter $\Delta g$. From top to bottom, the curves correspond to different standard deviation parameter $\sigma_0 = 0.25$ (blue line), 0.5 (orange) and 1.0 (green), respectively, and we used $n = 20$ features.

classification of individual objects, for two reasons. First, the Cramér–Rao–Fréchet bound [see, e.g, [5]] implies that the Fisher estimator has minimal variance, meaning that the probability expressed by equation (17) is an extreme, limiting case. The second reason is that we approximated the actual Fisher matrix, equation (8), by an average over the (uniformly distributed) noise dispersion parameter, which resulted in equation (9). Due to the random nature of the specific uncertainty $g_i$ in our model, individual objects may have better (less noisy) or worse (noisier) data points, and for those objects the confidence will differ from what is expressed by equation (17). Hence, equation (17) represents an ideal scenario: in practice, applying the likelihood method for a sample of objects results in an average accuracy which is slightly worse than the one that results from using this analytical formula. Therefore, the exploration of the likelihood in parameter space shall be performed object-by-object according to equation (6), using either a least squares optimization method (if one is only interested in the class itself) or an MCMC (if we also need to know the likelihood-based probability of the least squares classification). These results, derived from the likelihood function, form the basis for our comparison with the performance and confidence of the classification using ML methods.

It is instructive, in the context of this section, to also consider an associated linear problem. A linear estimator for the curvature is given by:

$$\hat{a}_\mu = n^2 \sum_{i=1}^{n} M_{\mu i} x_i, \tag{18}$$

where $M_{\mu i} = \frac{1}{2}\delta_{\mu,i+1} + \frac{1}{2}\delta_{\mu,i-1} - \delta_{\mu,i}$, and the indices $\mu = 2, 3, \ldots, n-1$ can be regarded as the $n-2$ intermediate points where we are able to estimate the curvature ($a$) through differences of the neighboring points. It is easy to check that the linear estimator is unbiased, and independent of the model parameters $b$ and $c$: just use the identities $\sum_i M_{\mu i} = \sum_i M_{\mu i}\, i = 0$, and $\sum_i M_{\mu i}\, i^2 = 1$.

The covariance of the linear estimator is given by the expectation value:

$$C_{\mu\nu} = \langle (\hat{a}_\mu - a)\,(\hat{a}_\nu - a) \rangle = n^4 \sum_{i=1}^{n} M_{\mu i} M_{\nu i} g_i^2\, \sigma_0^2\,, \tag{19}$$

which, after averaging over the uniform distribution for the noise dispersion leads to the mean covariance of the linear estimator:

$$\bar{C}_{\mu\nu} = \left(1 + \frac{1}{3}\frac{\Delta g^2}{\bar{g}^2}\right) \bar{\sigma}_0^2\, n^4 \sum_{i=1}^{n} M_{\mu i} M_{\nu i}. \tag{20}$$

Equation (20) leads to another analytical approximation for the variance of $a$. The information from each set of linear estimators is expressed by means of the Fisher matrix of the linear estimator, $F_{\mu\nu} = \bar{C}^{-1}_{\mu\nu}$. Summing

the information from all the estimators corresponds to the total (grand) sum of this Fisher matrix, and the inverse of that sum is the final parameter covariance:

$$\Sigma^2_{a,\mathrm{lin}} = \left[\sum_{\mu\nu} \bar{C}^{-1}_{\mu\nu}\right]^{-1} = \bar{\sigma}_0^2 \left(1 + \frac{1}{3}\frac{\Delta g^2}{\bar{g}^2}\right) \frac{180\, n^3}{n^4 - 5n^2 + 4}, \tag{21}$$

which can be compared with equation (16).

Equation (21) highlights a property that was already apparent in equation (16), which is the fact that the uncertainty in the output (the curvature, $a$) depends on the number of features only through the factor $180\, n^3/(n^4 - 5n^2 + 4)$. The output uncertainty on the pattern of input errors, on the other hand, is encapsulated by its dependence on $\Delta g$. When we include the information about specific noise levels through inverse covariance weighting, as expressed by equation (8), that pre-factor is $1 - \Delta g^2/\bar{g}^2$—i.e. providing the information about noise *improves* the constraints. However, when we neglect the error information and resort to direct estimators such as $\hat{a}_\mu$, then that pre-factor becomes $1 + \Delta g^2/(3\bar{g}^2)$, *increasing* the output uncertainties.

Finally, it is worth pointing out the limitations of tools such as Tikhonov regularization, which are often used to prevent overfitting and to minimize empirical error—see, e.g. [33], as well as related methods such as the one proposed by [31]. Basically, regularization techniques work by effectively imposing a threshold on very small eigenvalues in ill-posed inverse linear problems. However, the associated linear problem presented above is perfectly well-posed, and still it results in a degradation of the output uncertainties when compared with the optimal (inverse covariance weighting) estimator. Furthermore, the covariance of the linear estimator, equation (19), is a positive-definite matrix, $\sum_{\mu\nu} C_{\mu\nu} V_\mu V_\nu \geqslant 0$ for any real-valued vector $V_\mu$, so all the eigenvalues of this covariance matrix are real, non-negative numbers. Consequently, imposing any kind of minimum threshold for those eigenvalues would in fact *increase* the linear estimator uncertainty, equation (21), which means that no amount of regularization can possibly compensate the lack of information about the noise of each input data point.

## 4. ML classifiers

Our problem statement is to classify sets of features that characterize curves which belong to a given class. We can model this as a supervised ML classification task. Our training sample is the set of tuples $\{\mathbf{x}^{(j)}, y^{(j)}\}_{j=1}^{m}$, where $\mathbf{x}^{(j)} \in \mathbb{R}^n$ are the smiley–frowny parabolic curves and $y^{(j)} \in \{0, 1\}$ are the corresponding labels. Additionally, for those models that use the information of the uncertainties, we have $\sigma^{(j)} \in \mathbb{R}^n$ for each $\mathbf{x}^{(j)}$, where $\sigma_i^{(j)} = g_i^{(j)}\, \sigma_0, i = 1, 2, \ldots, n$, as defined in § 2.

Due to the nature of the data, where the attributes are sorted in a significant way, we find it more appropriate to use CNNs since they are able to recognize local features that helps on distinguishing between different types of objects (e.g. emission and absorption lines in spectra). Multiple problems of sequential data analysis are in fact tackled with CNNs and 1D convolutional kernels—see, e.g. [34–37] and for applications in astronomy see, e.g. [16–19]. In this work we implemented the classifiers with `keras` [38] and created the CNN from scratch. We stress the fact that it is not the focus of this work to compare between existing CNN architectures to find which one provides the best fits to our toy-model, but instead to compare between different input data content. Therefore a simple architecture is sufficient for our purposes, as long as we ensure, of course, that the method is not underfitting nor overfitting the data.

Unless noted otherwise, our networks are trained with a sample of $m_{\mathrm{train}} = 2 \times 10^5$, from which 80% are taken to train and the remaining 20% are used to validate the models, i.e. to fix the hyperparameters of the networks and to monitor the bias–variance trade-off. The results shown in the next sections were computed with test sets, which were left completely unbiased by the training procedure and contains $m_{\mathrm{test}} = 10^5$ instances. All the data sets are balanced in terms of classes, i.e. they contain the same amount of smiley and frowny objects.

We created multiple CNNs which differ from each other mainly in the content and shape of the input data. The specifications of each version are described in the next subsections and in appendix C. The general training settings are summarized as follows. We used the binary cross-entropy loss function and the `Adam` [39] optimizer. In all intermediate layers we used the rectified linear unit (ReLU) activation function [40] and in the last layer we used the Softmax activation function so that the scores of both classes sum up to one and, thus, we have a probabilistic interpretation for the output. The convergence of the training was monitored with learning curves for accuracy and for the loss function at each iteration (epoch) for both training and validation sets and we used batches of size 100 to train the networks. We used the `EarlyStopping` callback conditioned to the validation set loss score with `patience` of 16 epochs and the `ReduceLROnPlateau` callback to reduce the learning rate when the validation set loss stagnate for ten

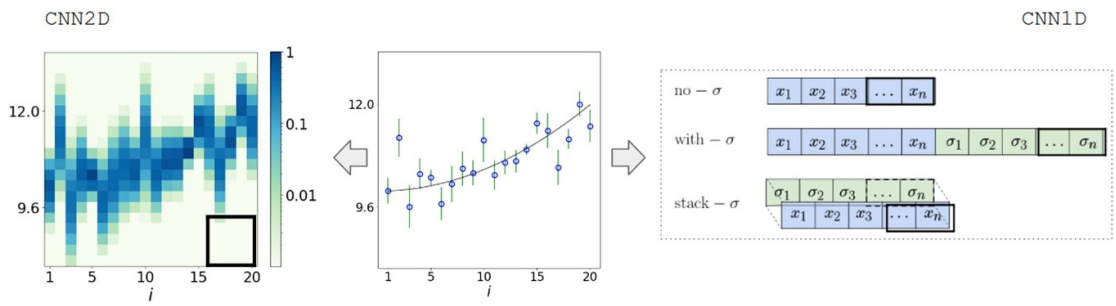


**Figure 4.** Input data illustration. On the right we present the input formats for the three CNN1D networks, $x_i$ are the measurements and $\sigma_i = g_i\sigma_0$ are the uncertainties of each measurement. On the left we show how to convert the input data (center) to a matrix, or image, for the CNN2D network (notice that this is an image with a single channel, where the color gradient indicates the values of each pixel). The black rectangles represent the convolution kernels.

epochs. The final set of weights is the one corresponding to the epoch with best accuracy in the validation set. For more details about the architectures, see appendix C.

We now turn to the description of the different networks that we trained in order to classify the smiley and frowny objects.

### 4.1. CNN1D

In the CNN1D models, the input data shapes are 1D vectors, and, thus, the convolution kernels are also 1D. We compared three versions of CNN1D models (see figure 4):

- **no-$\sigma$**: the input data shape is $(n,\ 1)$, it contains only the $n$ measurements.
- **with-$\sigma$:** the input data shape is $(2\cdot n,\ 1)$ where the $n$ measurements are followed by the $n$ corresponding uncertainties. This is a first approach to include uncertainties, but without making any hypothesis on what is the best way to represent this additional information.
- **stack-$\sigma$**: the input data is the set of measurements and errors arranged in channels. The input shape, therefore, is $(n,\ 2)$. It is identical to CNN1D with-$\sigma$ in terms of the available information, but, in this case, the relation between the measurements and corresponding errors is represented in a straightforward way in this model, providing a context for the uncertainties.

We defined a simple standard network for all three CNN1D versions, which consists of three convolution layers with kernel shapes $(5,)$, $(3,)$ and $(3,)$ with $32, 64, 64$ filters, respectively, one intermediate dense layer with 64 neurons and, finally, the output layer with two neurons. Each convolution layer does *padding*, i.e. the output feature map has the same size as the input feature map, and is followed by a `MaxPooling` layer with kernel size and stride $(2,)$. We also add `BatchNormalization` and `Dropout` layers with dropout rates typically between 0.2 and 0.4. As we vary the levels of noise in the training set, some modifications on the standard architecture might be necessary to ensure the convergence of the models. As the data becomes noisier, the models are more likely to overfit [41], therefore, in some cases a less complex network (with a lower number of layers) or a more regularized network is more appropriate (see appendix C for more details about the architectures).

### 4.2. CNN2D images

Several problems in the physical sciences have benefited from the power of ML methods that were developed for the analysis of images, in particular applications developed for high energy physics [3] or astrophysics [42]. In the CNN2D method we build on the same idea, however we use the additional dimension to represent the uncertainties in input data.

The idea of the CNN2D images method is to represent the complete distribution of the data, given the specific uncertainties. We then organize the data, including the errors, in terms of a matrix whose columns correspond to the features ($i$), and the rows correspond to the values of distribution function of the input data. To be specific, we discretize the range of input values in terms of bins $x_i \to x_i^\rho$, with $\rho = 1, 2, \ldots,$ `n_rows`, and then define values of the pixels of the CNN2D images as:

$$P_{\rho,i} = \frac{1}{\sqrt{2\pi\,\sigma_i^2}}\,\exp\left[-\frac{1}{2}\frac{(x_i - x_i^\rho)^2}{\sigma_i^2}\right]. \tag{22}$$

The left panel of figure 4 illustrates the input data for this model.

The input shape of this model is $(\texttt{n\_rows}, n)$, where the number of rows `n_rows` is one among other hyperparameters of the images that must be chosen. More details about the construction of the images can be found in appendix A. In appendix B, we discuss how the information of the error bars is being used by CNN2D images.

Our standard network consists of three convolution kernels with shapes $(5,5)$, $(3,3)$ and $(3,3)$ with 32, 64, and 64 filters, respectively. The convolution layers are followed by `MaxPooling` layers with kernels with size and stride of $(2,2)$, except for the first layer where the shape and stride of the kernel were chosen to be such that the output shape of this layer is always (10, 10), i.e. it depends on the shape of the input image (see appendix C for more details about the architectures).

The idea of representing a data vector with errors in terms of an image can be generalized for scientific data that is given in terms of pairs $\{x_i \pm \sigma_{x_i}, y_i \pm \sigma_{y_i}\}$. In that case, the multivariate probability distribution associated with the uncertainties $\sigma_{x_i}$ and $\sigma_{y_i}$ mean that each data point $i$ is 'spread out' both in the horizontal (rows, $x$) as well as the vertical (columns, $y$) directions. A CNN where a 2D data set (including uncertainties) is represented by images was used recently to classify supernovas [43].

## 5. ML confronts maximum likelihood

We now present the results for the classification of curves in the two classes (smiley or frowny), in the presence of Gaussian noise in the input data. Throughout this section we compare the accuracy of the CNN classification with that derived on the basis of the likelihood function. In the simplest application of the likelihood, we find the least squares solution for each object in our sample—and in this case, the ratio of objects correctly classified with respect to the total number is called the 'least squares accuracy'. The least squares method is considered as the baseline model in this analysis because, by construction, it incorporates all the information that distinguish between the classes. We can also explore the likelihood as a function of parameter space using an MCMC approach—in that case, for each object we have a probability for the classification which is given by the fraction of points in the chains with the correct class (i.e. we marginalize over the parameters $b$ and $c$.) Finally, we can compute analytically the shape of the likelihood around the minimum (the least squares solution) by means of the Hessian—i.e. the Fisher matrix of equation (17). Notice, however, that the Fisher estimator represents a limiting, ideal scenario, as discussed in § 3.2: it will by design appear to overperform compared with the other methods.

As discussed above, we generate the parabolic curves by sampling random parameters ($a$, $b$ and $c$) for the two classes, and then computing the $n$ features of the curves ($\bar{x}_i$), according to equations (1) and (2). The next step is to add noise to those features, according to equations (3) and (4). It is important to stress that, just like in real experiments, the values $g_i$ are stored, which allows us to keep track of the error bar for each data point, $g_i\sigma_0$. However, for each object neither the underlying 'true' curve, $\bar{x}_i$, nor the random noise, $\delta x_i$, are known *a priori*: we only have access to the measurement, $x_i = \bar{x}_i + \delta x_i$, and the noise levels $g_i\sigma_0$ (the 'error bars').

To classify the smiley–frowny curves, we start with a baseline data set with the parameters described in table 2 and in the following subsections we explore multiple scenarios where we deviate from this baseline data set by varying the values of the parameters $\sigma_0, n, \Delta g$ and $m_{\rm train}$.

### 5.1. Varying the nominal noise $\sigma_0$

We start by investigating how the performance of the CNN and of the least squares classifiers depend on the nominal noise $\sigma_0$—see figure 5. For very high nominal noise all the classifiers eventually fail, with accuracies of 50%, and conversely, for very small noise all models are able to achieve a near-perfect accuracy. Therefore, as one should expect, in both limits ($\sigma_0 \to 0$ and $\sigma_0 \to \infty$) all methods become equivalent, since there is no information in the noise levels.

However, for intermediate levels of noise, the classification methods that make use of the information about the specific noise levels in the input data are able to achieve better performance than the CNN1D no-$\sigma$ method, which does not. Notice that the difference in accuracy between the CNN1D no-$\sigma$ and CNN1D stack-$\sigma$ (and CNN 2D images) is $\sim$0.8 for $\sigma_0 = 0.8$ and drops to $\sim$0.4 for $\sigma_0 = 1.6$. Moreover, for $\sigma_0 \gtrsim 0.3$, the CNNs that take into account the noise levels of the input data are able to reach accuracies which approach those of the least squares classification. This means that when we pass the information about which features are more noisy and should be down-weighted, and which features have less noise and should be up-weighted, we allow the algorithms to learn about how to use those weights for their classifications.

In particular, we note that the CNN1D stack-$\sigma$ method has a performance that is very similar to the CNN2D images method. This is perhaps to be expected, since the noise is Gaussian, and in the absence of skewness or kurtosis the key information about the distribution is already encoded in the standard deviation, $g_i\sigma_0$. For more general PDFs, where it is not possible to summarize the shape of the distribution in terms of a

**Table 2.** Baseline smiley–frowny data set parameters. $m$ is the number of objects in the data set, $n$ is the number of features, i.e. the number of points in the curve. $\sigma_0, \bar{g}, \Delta g$ are the parameters introduced in § 3 to define the noise levels $\sigma_i$ of each measurement, i.e. the standard deviation of the distribution from which the noise $\delta x_i$ is sampled.

| $m$ (training set) | $n$ | $\sigma_0$ | $\bar{g}$ | $\Delta g$ |
|---|---|---|---|---|
| $2 \times 10^5$ | 20 | 0.5 | 0.6 | 0.5 |

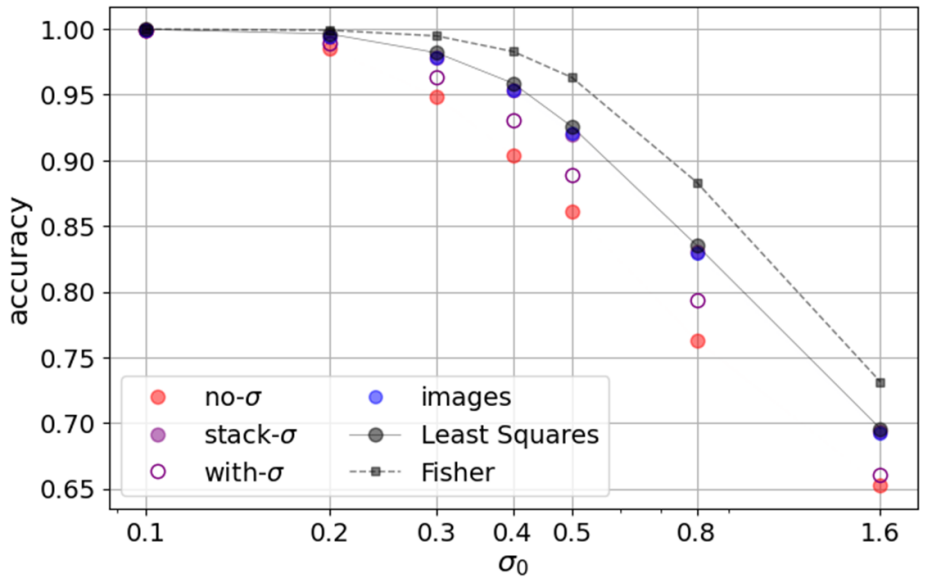


**Figure 5.** Accuracy as a function of $\sigma_0$ for fixed $\bar{g} = 0.6$, $\Delta g = 0.5$ and $n = 20$.

single parameter, using the CNN2D images approach might be an interesting alternative. Alternatively, one could think of generalizing the CNN1D with-sigma models to multiple channels, each one containing the different momenta of the underlying distribution functions.

Still looking at figure 5, we see that all three models that include the errors have better accuracies than CNN1D no-$\sigma$, but CNN1D with-$\sigma$ is slightly worse than the other two. This difference can be explained by a fundamental difference in the form errors are provided to the models. While the point-wise signal-noise association is preserved in the input of CNN1D stack-$\sigma$ and CNN2D images, the same does not happen for CNN1D with-$\sigma$, causing the natural association to be lost. The model may eventually recover such association, but dismissing the association only adds unnecessary difficulties to the model. Therefore, in what follows we opted to discard the CNN1D with-$\sigma$.

### 5.2. Varying the number of features $n$

The typical values of $\sigma_0$ for which we see a significant difference between the models that include or do not include the uncertainties depend on the number of features $n$. If there is an abundance of points that characterize the curve, the classification become easier, and $\sigma_0$ should be larger in order to cause some confusion between the classes. In other words, it is equivalent to either lower the overall level of noise ($\sigma_0$) or to increase the number of features ($n$). This is shown in figure 6, where we evaluate the performance of the models as we increase the number $n$ of features, for a fixed $\sigma_0 = 3.2$. As we grow the number of features, all methods become more efficient, but the CNN1D method without errors clearly underperforms compared with the other methods.

### 5.3. Varying the noise dispersion $\Delta g$

Figure 7 shows the performance of the classifiers as the noise dispersion parameter $\Delta g$ changes. The smaller the value of $\Delta g$, the closer to the homoscedastic regime is the data. This plot shows that, as we increase the relative difference between the noise levels of the data points, that information becomes more critical to the classification. That information is naturally used in the least squares method, and is also learned by the CNNs that are provided the error bars, but the CNN that is only provided the input data is unable to harness that information to improve the classification. Notice that the accuracies for $\Delta g = 0.5$ in figure 7 correspond to the baseline model, i.e. are the same as $\sigma_0 = 0.5$ in figure 5.

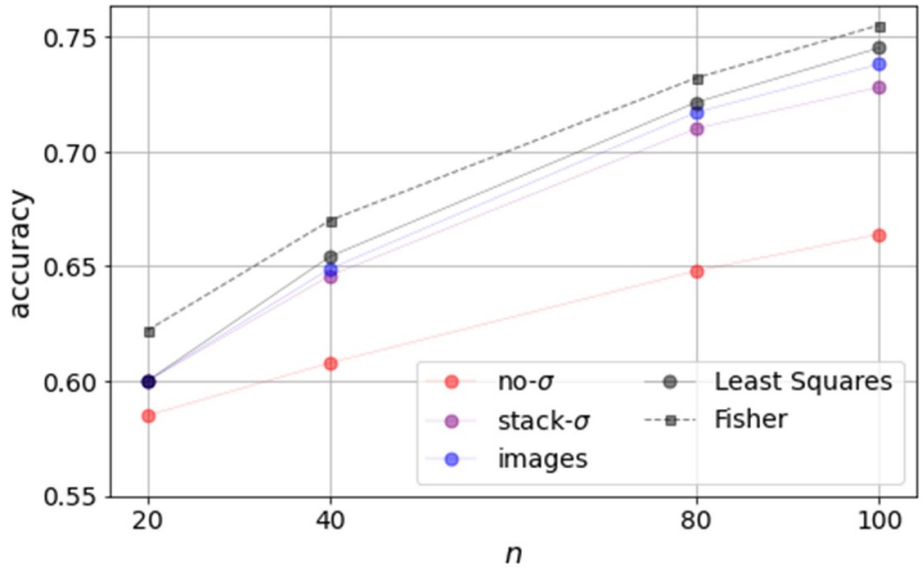


**Figure 6.** Accuracy as a function of the number of features $n$ for fixed $\bar{g} = 0.6, \Delta g = 0.5$ and $\sigma_0 = 3.2$.

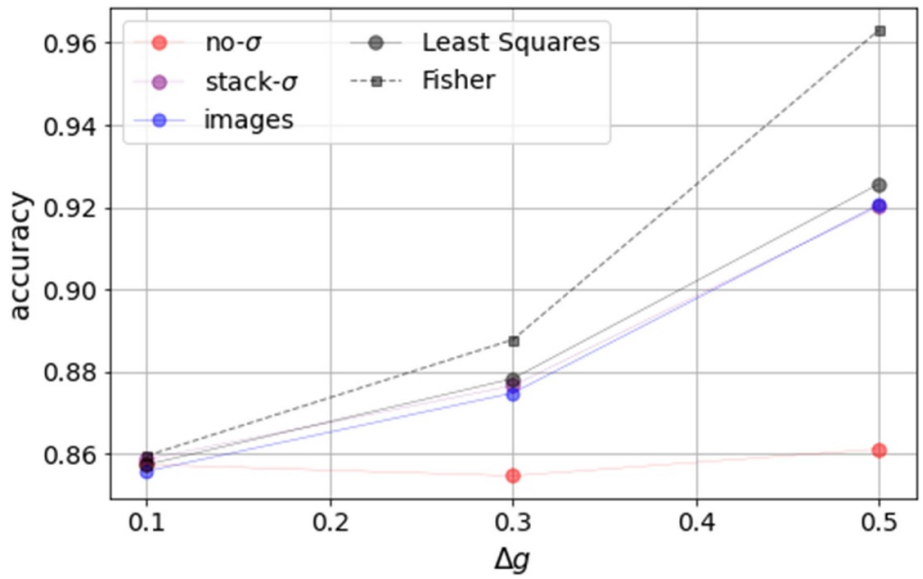


**Figure 7.** Accuracies for $\Delta g = 0.1, 0.3, 0.5$ with fixed $\bar{g} = 0.6$, $n = 20$ and $\sigma_0 = 0.5$.

### 5.4. Probability output

We now address the question about the quality of the ML classifiers as compared with an approached based on the shape of the likelihood function.

In real classification problems, we are not only interested in getting a high accuracy in the predictions, but also to have a reliable estimation of the probability associated to each class. In other words, we expect noisier objects to have lower confidence in their classification, and thus reflect the fact that sometimes we cannot be completely sure if an object is a star or a galaxy, for example. This difficulty is related to the issue of estimating the 'true probability' associated to the class of the object. With the smiley–frowny data set, however, we can estimate these probabilities with the likelihood approach, because the underlying models (the parabolic curves) are known.

For each object we explore the likelihood function, equation (6), by means of MCMC approach. We use flat priors for the parameters, which were free to vary inside the ranges $a \in [-3, 3]$, $b \in [-5, 5]$ and $c \in [-5, 25]$—i.e. more than eight standard deviations for the parameter $a$, and about three standard deviations away from the means for the parameters $b$ and $c$. These priors are sufficiently uninformative to preclude skewing or biasing of the inferred distribution in any significant way.

**Table 3.** MSE of the cases shown in figure 8.

| | CNN1D no-$\sigma$ | CNN1D stack-$\sigma$ | CNN2D images |
|---|---|---|---|
| $\Delta g = 0.1$ | 0.0059 | 0.0052 | 0.0026 |
| $\Delta g = 0.5$ | 0.0386 | 0.0087 | 0.0064 |

The ultimate goal of our MCMC is to compute the posterior probability that the maximum likelihood (least squares) classification for each individual object is correct. In order to compute that probability, we count the fraction of points in the chains that are assigned the correct class of each object, i.e. it is the number of iterations in each Markov chain that give the correct class relative to the total number of all iterations in the chain. Since each object is also assigned a class by the three different ML methods, together with their respective scores, we can plot the ML output values as a function of the MCMC-derived probability (i.e. the posterior) for all objects in the test sample.

This comparison is shown in figures 8 and 9. In figure 8 we show the CNN output value and the MCMC-derived probability for $5 \times 10^4$ objects in the smiley (positive) class, using the baseline data set (we may refer to the CNNs output values as output scores or as the CNN confidence). Objects with MCMC-derived probability $>0.5$ are classified in the correct class, and when the probability is $\leqslant 0.5$ the maximum likelihood classification fails. The same applies for the CNNs: a score of 0.5 marks the threshold between correct and wrong classification.

The most revealing aspect of figure 8 is that, when the noise levels of all input data points are nearly the same (nearly homoscedastic case, $\Delta g = 0.1$, top row), the three ML methods hold a tight correlation between the classification score and the MCMC probability that the classification is correct. However, when the data points have significantly different levels of noise (heteroscedastic case, $\Delta g = 0.5$, bottom row), if those uncertainties are not passed on to the CNN (as is the case of the CNN1D no-$\sigma$ model, left panel), then there is basically no correlation between the ML output value and the MCMC probability. But when the input data uncertainties are part of the information provided to the CNNs, a clear correlation appears between the ML output value and the probability. With a clear relation between the CNN and MCMC output values, we have a model that better resembles the true probability and thus provide more meaningful scores for our purposes. In fact, the ML output is well fitted by the sigmoid function:

$$c_{\mathrm{ML}}(p) \sim \frac{p^2}{p^2 + (1-p)^2}. \tag{23}$$

This function can be easily inverted, which allows us to define:

$$p_{\mathrm{ML}} = \frac{c_{\mathrm{ML}} - \sqrt{c_{\mathrm{ML}} - c_{\mathrm{ML}}^2}}{2c_{\mathrm{ML}} - 1}. \tag{24}$$

A measure of the goodness of this fit can be computed with the mean squared error (MSE) metric:

$$\mathrm{MSE} = \frac{1}{m_{\mathrm{test}}} \sum_{j=1}^{m_{\mathrm{test}}} \left[p_{\mathrm{ML}}(j) - p(j)\right]^2, \tag{25}$$

where $m_{\mathrm{test}}$ is the number of objects in the test sample, and $p(j)$ is the probability (according to the MCMC) that the classification of object $j$ is correct.

We obtain that, for $\Delta g = 0.1$, the MSE for all methods is below 0.01, with the CNN2D images method performing slightly better at 0.0026, compared with 0.0059 for CNN1D no-$\sigma$ and 0.0052 for the CNN1D stack-$\sigma$ model (we use a sample of $m_{\mathrm{test}} = 5 \times 10^5$ objects in order to compute this statistic). When we increase the noise dispersion to $\Delta g = 0.5$, the CNN1D no-$\sigma$ fails to fit the sigmoid, with an MSE of 0.0386, whereas the CNN1D stack-$\sigma$ and CNN2D images methods still performing well, below 0.01 – see table 3.

The results summarized in figures 8 and 9, together with table 3, mean that, when the noise levels of all the input data are the same (or nearly the same), then the ML methods are able to estimate the quality of the classification (the confidence) in a way that works as a proxy for the probability that this classification is correct. On the other hand, if different data points have varying levels of noise, then it becomes essential to pass that information on to the networks. When that information is hidden from the network, as happens for CNN1D no-$\sigma$, then the method loses its ability to provide a score that is significantly correlated with the probability for that classification—see the bottom left panel of figure 8. However, if we pass the noise properties of the data as information to the CNNs, then we allow those networks to reconstruct scores that are tightly correlated with the probabilities for the classification.

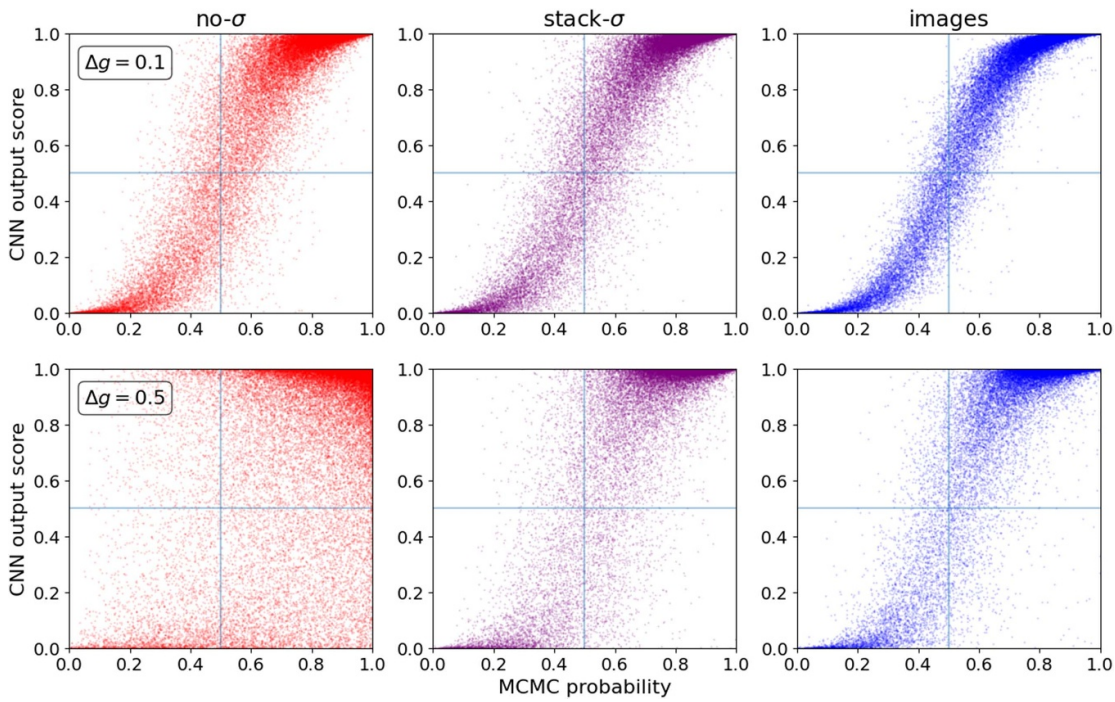


**Figure 8.** CNNs output score *vs* probability derived from the likelihood-based MCMC for the classification of 50 000 smiley objects. The noise models used for this test had $\sigma_0 = 0.5$, $n = 20$, with noise dispersion given by $g \in [0.5, 0.7]$ (nearly homoscedastic case, top), or by $g \in [0.1, 1.1]$ (heteroscedastic case, bottom).

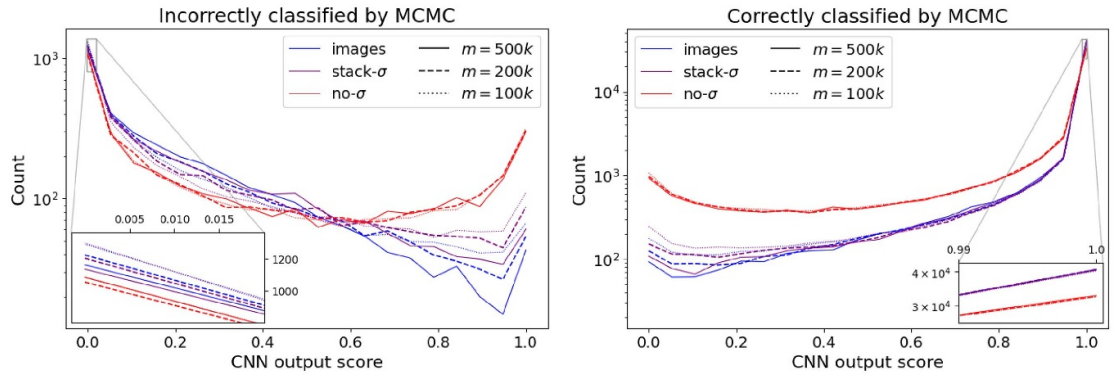


**Figure 9.** CNN output scores in the classification of the smiley objects which were incorrectly (left) and correctly (right) classified by MCMC, in the case where $g \in [0.1, 1.1]$. The model parameters are the same as the ones used for figure 8, but here we also vary the size of the training set. Solid lines correspond to training sets with $m = 500 \times 10^3$ instances, dashed lines $m = 2 \times 10^5$ and dotted lines $m = 1 \times 10^5$. These curves correspond to the mean values over multiple training realizations.

Another way to visualize the predictive power provided by the noise information is to take the points shown in figure 8, and separate them into objects that are correctly classified by the MCMC (probability $>0.5$), and those that are incorrectly classified. In figure 9 we show the resulting distribution of objects as a function of the ML output value for the incorrectly classified (left panel) and correctly classified (right panel) objects. We also show how the output value varies with the size of the training sets ($m = 1,\ 2$ and $5 \times 10^5$ objects). It is immediately clear that the CNN that is blind to the information about uncertainties is unable to pick out the more noisy objects, and as a result it assigns high output scores to objects in the wrong class much more often than the other methods. Furthermore, for the objects that are correctly classified by the MCMC through the least squares solution (right panel), the CNN1D no-$\sigma$ method tends to assign lower probabilities to more objects, which again is a result of those methods being unable to weigh features by their uncertainties. The inset plot in the right-hand side panel shows that the number of objects correctly classified by MCMC with the highest output scores is significantly lower for the CNN1D no-$\sigma$ method. This figure also suggests that augmenting the training set is not sufficient to recover a more reliable output with CNN1D no-$\sigma$ as compared to MCMC's, since even for larger training sets the network is overconfident.

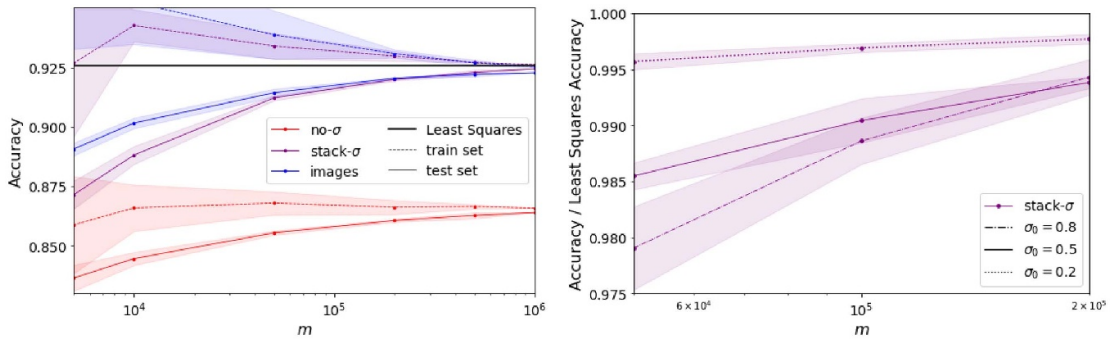


**Figure 10.** Left: accuracies as a function of the training set size $m$. Solid and dashed lines correspond to test and train set, respectively. The curves are the median value of multiple realizations and the shaded region covers the interval median $+/-\sigma_{\rm NMAD}$ 26; right: ratio between the accuracy of CNN1D stack-$\sigma$ and the accuracy of least squares as a function of the number of training set instances for $\sigma_0 = 0.2, 0.5, 0.8$.

### 5.5. Varying the training set size

An important issue that appears as we increase the dimensionality of the system by including additional parameters related to noise is the size of the training set that is needed for the network to converge. We have evaluated the performance of all classifiers as a function of the size of the training set for our baseline model, with $n = 20$, $\sigma_0 = 0.5$, $\bar{g} = 0.6$ and $\Delta g = 0.5$. We froze the same architecture that was used with $m = 2 \times 10^5$ objects in the training set, and re-trained the network with different sizes in order to see how its performance degraded or improved as we lower or increase the number of objects. We also analyze how much sensitive the model becomes to the initial seed as the training set size decreases. For lower number of objects, it is harder for the model to converge. This is not the case for larger training sets, where the model converges to very similar results with different initial seeds. Of course, it is still possible to improve the accuracy for larger/lower $m_{\rm train}$ if one reduces/increases the complexity (number of parameters) of the network. Notice that the least squares is only performed on the test set and, thus, it is not affected by the number of instances in the training set.

The left panel of figure 10 shows the accuracy for both train (dashed lines) and test (solid lines) sets of the models, as we increase the number of training instances. This means that, as we take $m \to \infty$, we have minimized epistemic error for each ML method, and all that remains is the impact of aleatoric errors on the different models. The lines correspond to the median value of multiple realizations, and the error bars (shaded regions) correspond to 'normalized median absolute deviation' $\sigma_{\rm NMAD}$, which is defined by:

$$\sigma_{\rm NMAD} = 1.4826 \times \text{median}\left(|y_i - \text{median}\left(y\right)|\right) . \tag{26}$$

This measure of the width of a PDF reduces to the variance in the case of a mono-variate Gaussian distribution, but is less affected by the tails of the distribution.

We see that, for the training set sizes that we analyzed, no amount of training data is sufficient for CNN1D no-$\sigma$ to come close to the performance of the models that include the information about uncertainties[3]. Notice that our toy model allows us to generate an arbitrarily large number of objects, while data augmentation usually employs some fixed set of objects and then adds an artificial amount of noise to the input data of those objects. Therefore, the larger training sets of figure 10 are composed of objects which behave exactly like the ones in the validation and test sets, while data-augmented training sets are made up of a mix of original objects as well as objects which behave in a fundamentally different way compared with the original ones. As a result, the performance of a model trained on a set of $m$ original objects is always superior to that of a model trained on a set of $m$ objects that was created with the help of data augmentation techniques. Hence, this result also proves that data augmentation techniques cannot possibly overcome the deficit of neglecting the information about uncertainties.

Shy *et al* [44] performed a study of uncertainty inclusion on astronomical data sets by generating additional training instances within the interval given by nominal errors. They argue that by augmenting the training set in this way, i.e. by adding noise, there is no way to overcome the performance obtained with the original training set, which is in agreement with our conclusions.

[3] For extremely small training sets it may be possible that the lower complexity of the methods without noise information makes them superior to the ones that include noise.

**Table 4.** Parameters of the normal distributions from which the coefficients of the parabolic curves are sampled—Poisson version.

| Coefficient | $a_\smile$ | $a_\frown$ | $b$ | $c_\smile$ | $c_\frown$ |
|---|---|---|---|---|---|
| $\mu$ | 1 | −1 | 0 | 3 | 4 |
| $\sigma$ | 0.25 | 0.25 | 1 | 0.5 | 0.5 |

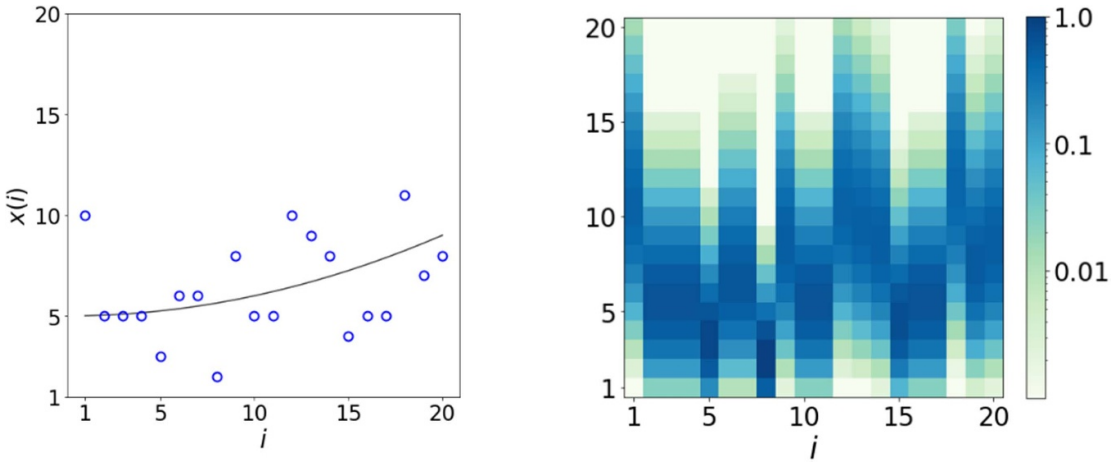


**Figure 11.** Smiley–frowny data generated with Poisson noise.

Both CNN2D images and CNN1D stack-$\sigma$ have similar performances, even though the former presents higher complexity than the latter, especially when considering the intrinsic model variations (shaded regions in figure 10). Moreover, the amount of data required for the models to converge to some accuracy depends on how noisy the data is: when the data is more noisy, the models are more likely to overfit. This is shown in the right panel of figure 10, where we plot the accuracy of the models as a function of size of the training, for different values of $\sigma_0$ (we only show the results for the CNN1D stack-$\sigma$, for clarity, but all methods behave in essentially the same way.) Notice that in this plot we normalized the accuracy of the CNN classification by the accuracy of the least squares (maximum likelihood) classification, in order to highlight the fact that the difference between those accuracies is more pronounced for larger values of the nominal uncertainty $\sigma_0$.

## 6. Poisson noise

The Poisson distribution is particularly interesting because the uncertainty of the measurement can be deduced from the measurement itself. In that sense, it would be redundant to provide the noise information: if a feature has a measured value $x_i$, a good estimator for the noise is already given by $\sqrt{x_i}$. Hence, we expect that the information of the uncertainty is already encoded in the value of the measurement and, therefore, it does not make any difference to use as input only the measurement, with CNN1D no-$\sigma$, or to use the measurement and its associated error bar, like we do with CNN1D stack-$\sigma$ or CNN2D images.

In order to work with Poisson noise, it is convenient to employ measurements that have values closer to one, in order to reinforce the skewness of the distribution. Therefore, we used an adapted version of the smiley–frowny model to study the Poisson case. We lowered the mean value of the parameter $c$ of the parabolic curves to get mean values closer to one, and also reduced its standard deviation to avoid measurements with negative values. The few objects which happened to display any feature with negative values were discarded. Finally, since the noise in this case turns out to be relatively small, we also chose different values of $c$ for smiley and frowny objects, so we could shuffle the curves more efficiently and prevent the machine from distinguishing between both classes by their absolute values instead of the curvature. Similarly to equation (22), we discretize the distribution and define the values of the pixels of CNN2D images as:

$$P_{i,\rho}^{\text{Poisson}} = \frac{\mathrm{e}^{-x_i} x_i^{\rho}}{\rho!}, \tag{27}$$

with $i$ labeling the columns and $\rho = 1, 2, \ldots,$ `n_rows`. Table 4 presents the distribution of parameters used to test the Poisson noise model, and figure 11 shows the input data for CNN2D images with $n = 20$, which represent the measurements.

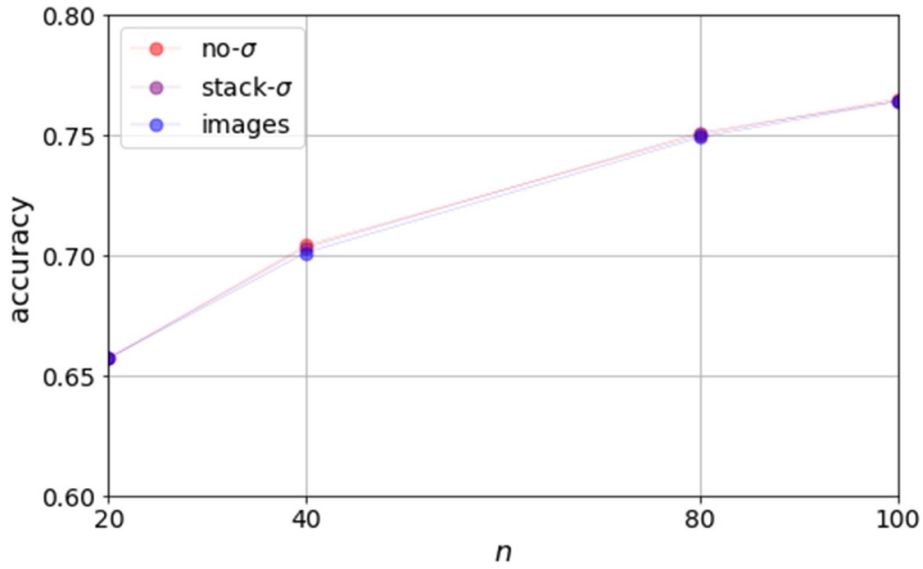


**Figure 12.** Accuracy as a function of the number of features $n$ in the Poisson version of the smiley–frowny model.

In the Poisson case, we can evaluate how the accuracy of different models varies under different noise regimes by increasing the number of features $n$, similarly to § 5.2. Figure 12 shows the dependence of the performance with $n$ for the Poisson case—compare it with figure 6, for the Gaussian noise model.

These results show that the accuracy improves as $n$ increases, as expected, but all models have basically the same accuracy (we verified that small differences are due to fluctuations). Once again, we have adapted the CNNs architectures to best fit the different noise regime data sets and avoid overfitting.

This exercise shows that including or not the uncertainty is irrelevant when the noise is drawn from a Poisson distribution: indeed, in that case there is no additional information in the noise levels (error bars) that is not already present in the measurements (the input data).

## 7. The waveform dataset

We now apply our methods to the publicly available waveform data set[4] [45]. This serves as a complementary analysis, as this data set has been extensively analyzed by the ML community using a variety of methods. That data set consists of three different combinations of three functions:

$$\begin{aligned}\text{class } 1: x_i^1 &= u\,h_i^{(1)} + (1-u)\,h_i^{(2)} + \delta x_i^1\\ \text{class } 2: x_i^2 &= u\,h_i^{(1)} + (1-u)\,h_i^{(3)} + \delta x_i^2\\ \text{class } 3: x_i^3 &= u\,h_i^{(2)} + (1-u)\,h_i^{(3)} + \delta x_i^3,\end{aligned} \tag{28}$$

where $i = 0, 1, \ldots, 20$ labels the $n = 21$ features, and the three 'parent' waves are shown in figure 13—from left to right, $h^{(1)}$, $h^{(3)}$ and $h^{(2)}$. The random variable $u$ is drawn from a uniform distribution, $u \in [0, 1]$. Therefore, it is clear that this is also a problem of assigning classes to sequence-like data.

There are two versions of the waveform dataset available in the UCI repository[5], which provides a data folder with a data generator code written in `C` and also a document with 5000 waveform instances. The versions differ from each other on the available information. The first version contains only the 21 features and the labels. The second version contains, in addition to the 21 features and labels, the 19 noise values $\delta x_i$ for $i \in [1, 20]$. For $i = 0$ and $i = 21$, the value of the wave is zero for all three combinations (classes) (see figure 13), which means that $x_{0,21} = \delta x_{0,21}$. Therefore, explicitly adding $\delta x_i$ to the data set is redundant.

The task is to classify objects according to these three types, using the $n = 21$ measurements and uncertainties. In the original versions, the noises $\delta x_i$ from the waveform data set were generated from the same normal distribution $\mathcal{N}(\mu = 0, \sigma = 1)$. The data set actually included the noise itself, $\delta x_i$, instead of the standard deviation of the Gaussian distribution from which those $\delta x_i$ were sampled. Since the noise is

[4] Competition scores: www.openml.org/t/58.

[5] Link to version 2: https://archive.ics.uci.edu/ml/datasets/waveform+database+generator+(version+2).

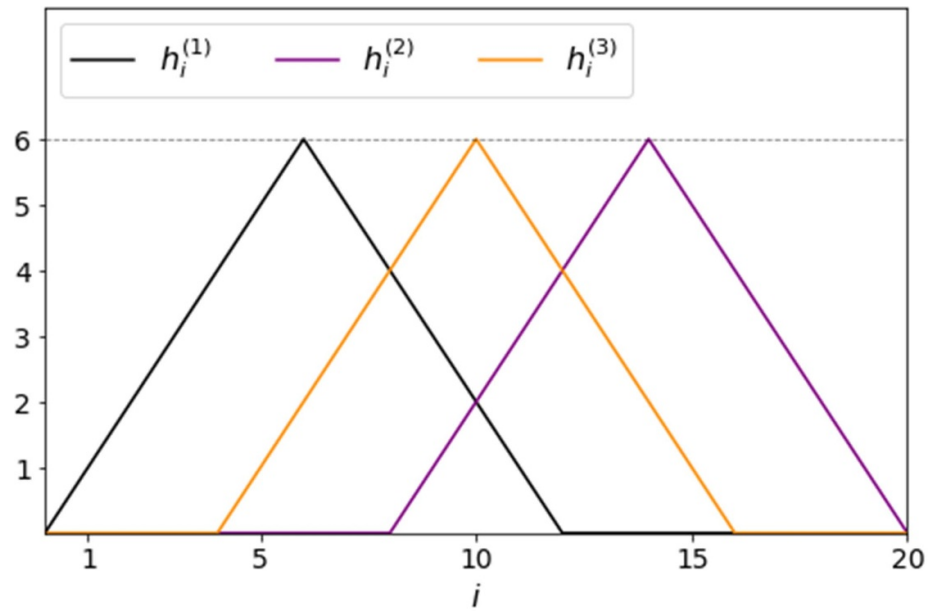


**Figure 13.** $h$ functions of the waveform dataset.

sampled from the same PDF for all points, there is no value in informing the networks about that noise. For this reason, we adapted the code that generates those features in order to have noises with different variances. Just as in the original waveform data set, we draw the uncertainties $\delta x_i$ from normal distributions:

$$\mathcal{N}(\mu = 0,\ \sigma = g_i \cdot \sigma_0)\,, \tag{29}$$

but in our modified version the noise levels $g_i$ are sampled from a uniform distribution with $\bar{g} = 0.8$ and $\Delta g = 0.5$, i.e. $g_i \in [0.3, 1.3]$.

We have computed the performance of the three ML classifiers, CNN1D no-$\sigma$, CNN1D stack-$\sigma$, CNN2D images, as well as the classification using the maximum likelihood[6], in the following situations: first, we kept $\Delta g = 0.5$ fixed, and varied the nominal error parameter $\sigma_0$. Second, we fixed $\sigma_0 = 1.25$ and varied the noise dispersion parameter $\Delta g$.

Here the least squares method is applied as follows. We fit the three models outlined in equation (28) to the data independently. The least squares classification corresponds to the $x_i^1, x_i^2, x_i^3$ with lower $\chi^2$.

The results are shown in figure 14. They resemble closely those shown in figures 5 and 7, and show that the CNNs that take into account the different levels of noise are superior to the method that discards that information, regardless of whether the data is overall less or more noisy (lower or higher values of $\sigma_0$). The right panel of figure 14 shows that, as $\Delta g$ grows and the information about which data points are more or less noisy becomes increasingly relevant, the methods that can account for these differences in the noise levels outperform by far the CNN1D no-$\sigma$ method, which is blind to those distinctions. Notice, in particular, that in the limit $\Delta g \to 0$ we recover the original waveform version, where the noise has a Gaussian distribution with $\mathcal{N}(\mu = 0, \bar{\sigma} = \bar{g} \cdot \sigma_0 = 1)$.

The main difference of this example with respect to the smiley–frowny model is that, while in the latter case the two classes are defined in terms of a continuous parameter, the three waveform types are discrete classes, which does not guarantee that the least squares is optimal as a means to classify the objects into classes. In fact, least squares is an efficient method to estimate the mixture parameter $u$ by fitting the models from equation (28) to the data. But in order to distinguish between the different waveform types, the CNNs model/likelihood can be more efficient than the least squares approach, which simply compares between the individual $\chi^2$'s of the three curves fitted independently.

Nevertheless, even in the case where least squares perform worse than the CNNs, which is likely to be the case in many practical applications, we can still see clearly the uncertainties becoming informative as the data set transits from the homoscedastic to the heteroscedastic regime.

[6] The class was assigned by choosing the function (equation (28)) which fits the data with the lowest $\chi^2$.

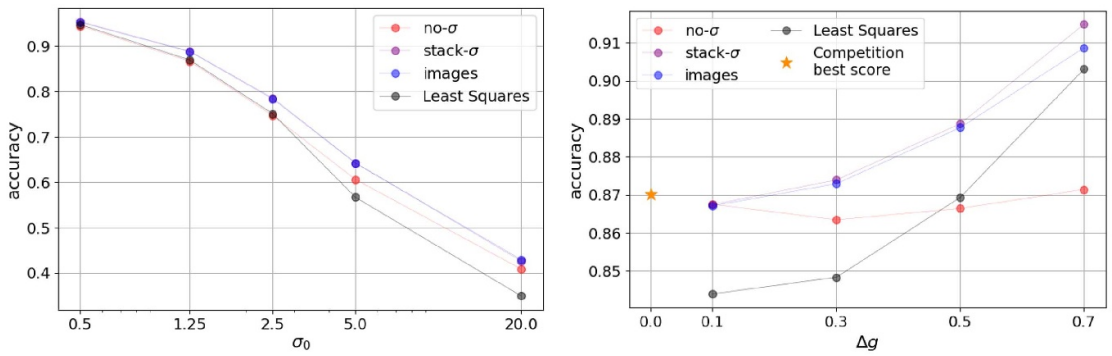


**Figure 14.** Results of the classifications applied to the modified waveform data set. Left: fixed $\Delta g = 0.5$, varying $\sigma_0$; right: fixed $\sigma_0 = 1.25$, vary $\Delta g$. For both cases we take $\bar{g} = 0.8$, thus $\bar{\sigma}_0 = 0.8\,\sigma_0$. The standard waveform data set has $\bar{\sigma}_0 = 1$ and $\Delta g = 0$, which corresponds to extending the curves on the right panel to $\Delta g \to 0$. For comparison, the star symbol in the right panel shows the best accuracy (0.8702) obtained in the waveform classification competition (where $\Delta g = 0$).

## 8. Conclusion

In this paper we address the value of the information about noise in input data for ML methods. We have shown that, when a data set includes not only the (noisy) measurements of the features, but also the information about the underlying distribution functions that generated that noise, CNNs are able to learn about the context of that noise, improving the performance of classification tasks and reaching 'optimality', defined here in terms of a maximum likelihood approach.

In order to prove this statement we created a toy model for two classes (the 'smiley' and 'frowny' parabolic curves), and a model for input data noise that realizes the typical process of measurement. Each object was generated from parameters that obey a random process, allowing us to build arbitrarily large sets that we can use to train, validate and test our methods. Noise, on the other hand, was also generated by means of a random process, but in such a way that each data point (feature) has a noise that is drawn from a different PDF, whose dispersion is known: this is the 'error bar' associated with each feature. This is exactly what takes place in astronomy experiments: not only measurements are taken, but also the uncertainties are assessed—which are typically not all identical.

As a result, not only the objects in our two classes have known underlying distributions, but each object can be classified using a maximum likelihood approach. This creates a standard against which the ML methods can be compared, as well as the concept of an 'optimal' accuracy for the classifiers. Notice that optimality, defined in this sense, has to be used with great care: ML methods can outperform likelihood-based estimators. However, in our toy model we limited the distinction between the two classes to a single parameter—the curvature. It is in that sense that we can define optimality.

Our main result is that, when the information about data noise is passed on to a CNN, it can learn how to use the different levels of noise to weigh the input data. This leads to improvements in the performance of the classifiers, in such a way that the accuracy of the ML classification approaches that of the optimal estimator. In fact, the more the noise levels vary from point to point (as controlled by the noise dispersion parameter $\Delta g$), the better the performance of the CNNs that included the noise level information compared with the CNN that did not.

Moreover, we showed that, when the levels of input data noise are not all identical, the output score of the ML method that is ignorant about those specific noise levels becomes uncorrelated with the underlying cumulative distribution function—see figure 8. However, when the noise levels are provided as additional data inputs to the CNNs, the resulting output values of the classifiers can again be mapped onto the likelihood-based posterior probability for the classification. Although that mapping is noisy, figure 8 shows that the CNNs seem to be using the information about the different levels of noise in the input data to reconstruct what is, in effect, a proxy for the likelihood function. In this work we do not apply specific techniques for uncertainty quantification, such as Bayesian neural networks, which could potentially improve the relation between the ML output scores and the MCMC-derived probability. However, even with such additional ingredients in the models, the uncertainties of the measurements provide unique information to relate between the predictions.

We have further tested CNNs with and without the noise level information using a slightly modified version of the waveform data set [45]. Just as happened for the smiley–frowny model, including the information about the different noise levels improves the accuracy of the classification, by an amount that

becomes larger as we increase the noise dispersion parameter $\Delta g$. We also computed the classification of objects in that data set using the least squares method, however in that case the performance was inferior to the CNNs, which is likely to be the case in many real world applications. Still, the uncertainties become informative to the CNNs as the level of heteroscedasticity increases.

We also checked that the noise levels are only relevant when they provide information that is not already included in the data set itself. In order to show this, we created a modified version of the smiley–frowny model whose features are numbers drawn from a Poisson distribution. In that case, for each feature $x_i$ the noise levels are well approximated by $\sqrt{x_i}$, and therefore there is very little additional information being provided by adding those errors to the data set. And indeed, what we find is that in this case the CNNs with and without error information have basically identical accuracies.

It is important to stress that ignoring the information about the different levels of noise in input data degrades the quality of ML classifiers in a way that cannot be offset by adding objects to the training set—through, e.g. the use of data augmentation techniques. As can be seen in figure 10, increasing the size of the training set, even in an ideal setup such as the one provided by our toy model, is not sufficient to allow the CNN without error information to achieve the accuracy level of the CNNs that include that information. In other words, the noise information is essential, and cannot be substituted or compensated. Moreover, as discussed in section 3, regularization techniques are also insufficient to compensate for the lack of information about the different levels of input data noise.

The smiley–frowny model is a simplification of data from narrow-band photometric surveys. Therefore, the techniques here presented are more appropriate to classify sources based on spectra or pseudo-spectra. Despite being super simplified scenarios, it is still possible to draw general conclusions with toy-models, as we have summarized throughout this section, which may translate to realistic cases. The techniques here presented were applied in [21] to classify quasars, stars and galaxies in the miniJPAS survey [46]. The CNNs were trained with simulated pseudo-spectra from that survey, which is composed by 60 fluxes (measurements) and corresponding uncertainties [22]. The CNNs that use the fluxes and the uncertainties showed better performance than the CNN that discards the uncertainties, especially when classifying fainter sources, which have typically lower signal to noise ratio.

Finally, we ought to remark that our conclusions were drawn in the context of a model inspired by scientific data, where tracking signal and noise are commonplace. However, in all areas of data science the issue of measurement is a key one: some data sets are more robust than others, and some data points are more reliable than others. Furthermore, estimations about the levels of noise in input data are often available to the data analyst not only in the hard sciences, but, e.g. in economic data or in the social sciences as well. What we have shown here is that providing these noise levels to ML methods adds significant information to the algorithms, improving the performance of classification or regression tasks in a way that cannot be compensated by techniques such as data augmentation or regularization.

## Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/nvillanova/smiley_frowny.

## Acknowledgments

We thank the anonymous referees for the numerous and very useful comments. Many thanks to Eloi Pattaro, who first got two of the authors (N R and R A) started on this path. We also thank Laerte Sodré Jr for insightful conversations about machine learning applications in Astronomy. We thank Google Collaboratory for the GPU resources. Financial support for this project was provided by CAPES, CNPq, and FAPESP through Grants 2019/26492-3 (R A), 2017/25835-9 and 2015/22308-2 (N H).

## Appendix A. Describing the images method

In this appendix we discuss the construction of the images in more details. The shape of the images are defined by a set of hyperparemeters, i.e. parameters that must be chosen before training the model. As mentioned in section 4.2, we discretize the features $x_i \to x_i^\rho$ and define the pixels of the images according to equation (22).

To set the height and width of the matrix, we define the variables:

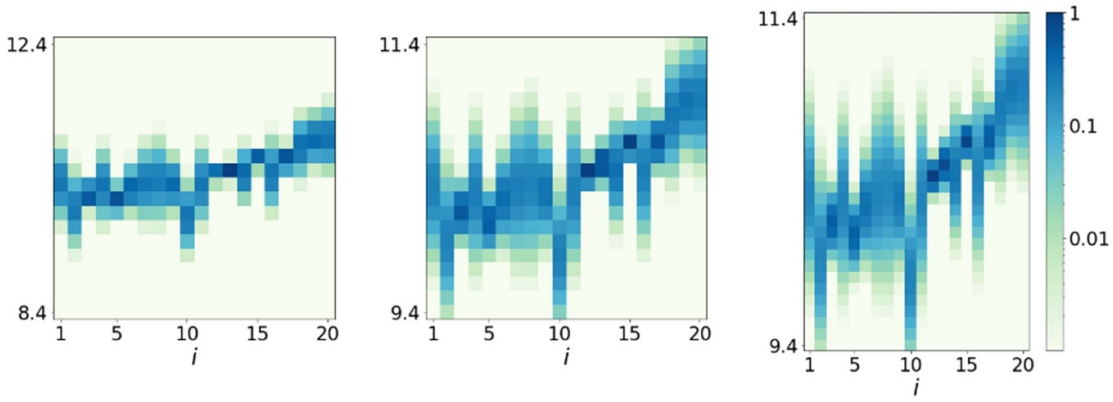


**Figure A1.** Input data of CNN2D images (smiley–frowny, $n = 20, \sigma_0 = 0.2$) with different images hyperparameters $\{\texttt{n\_rows}, \texttt{up\_bound}, \texttt{low\_bound}\}$. From left to right: $\{20, 2, 2\}, \{20, 1, 1\}, \{30, 1, 1\}$ ($\texttt{n\_cols} = n$ in all cases).

$$\texttt{n\_rows} : \text{number of rows}$$
$$\texttt{n}_{\texttt{cols}} : \text{number of columns}.$$

For simplicity, we set $\texttt{n\_cols} = n$, i.e. the number of features, since in this problem we assume there is no mixing in the $x$-axis, i.e. any given measurement, as well as its uncertainty, corresponds to a unique feature $i, i \in [1, n]$.

To define the boundaries of the image of each object $j$ ($j = 1, \ldots, m$) in the data set, we use the variables up_bound and low_bound[7]:

$$\text{upper boundary} = \sum_i^n \frac{x_i^{(j)}}{n} + \texttt{up\_bound}$$
$$\text{lower boundary} = \sum_i^n \frac{x_i^{(j)}}{n} - \texttt{low\_bound}.$$

These variables are useful to better frame the images. If the boundaries were the same for all objects, some curves could be clipped.

The choice of these hyperparameters might have a relevant impact on the performance of the model in some cases because these parameters define how much information of the data is being communicated through this representation. To illustrate this, we show in figure A1(a) smiley curve with $n = 20$ and $\sigma_0 = 0.2$ represented in three images built with different hyperparameters. We see that these parameters control the 'resolution' of the curve, i.e. the number of pixels to bin $x_i$. More specifically, the width of the bins expressing the intervals for the values $x_i^\rho$ are given by:

$$\Delta x = \frac{\texttt{up\_bound} + \texttt{low\_bound}}{\texttt{n\_rows}}. \tag{A.1}$$

## Appendix B. Evaluating the images method

Here we perform a test with CNN2D images model to try to get some intuition on how the model is using the information of the uncertainties. The idea is to evaluate how the accuracy changes as we gradually discard the information of the errors.

In order to do this test, we slightly modify the images by normalizing the columns in such a way that all columns of each image have the largest pixel equal to one:

$$P_{\rho,i} \to \bar{P}_{\rho,i} \equiv \frac{P_{\rho,i}}{\max_i \left(P_{\rho,i}\right)}, \tag{B.1}$$

[7] For the tests with the Poisson version of the smiley–frowny model, we fixed the boundaries to range between 0 and 20 for all the objects.

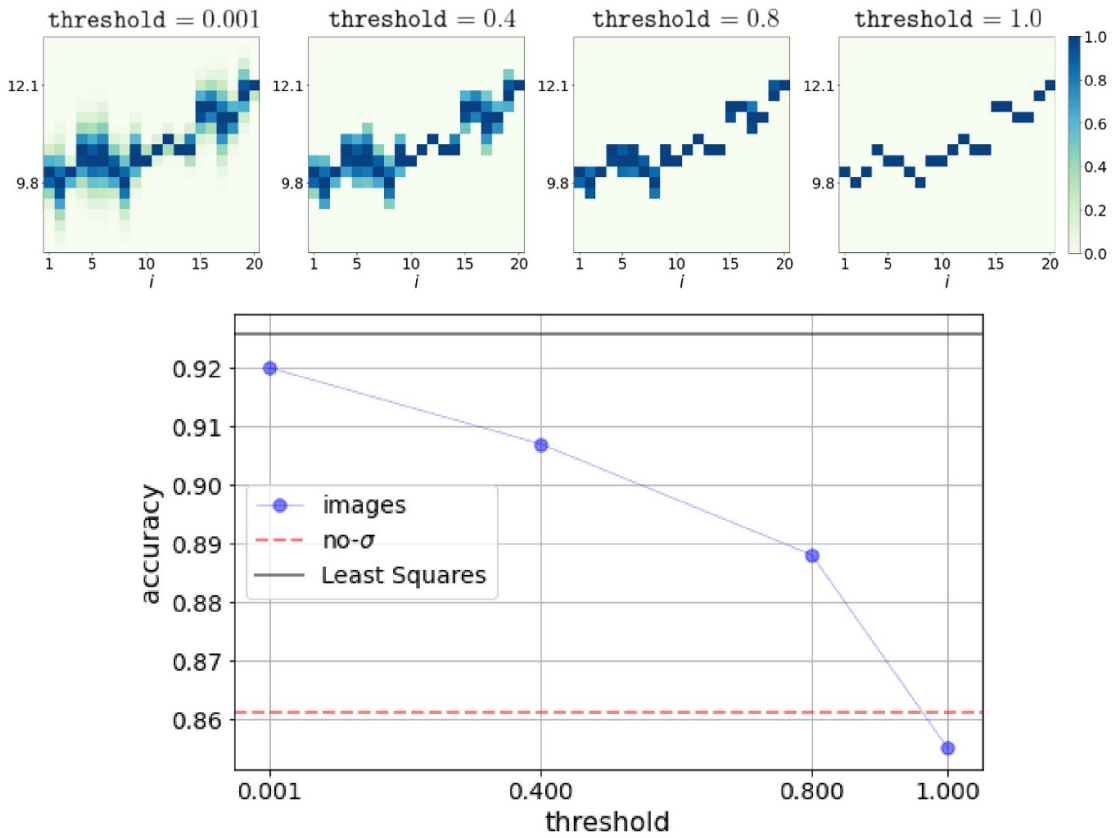


**Figure B1.** *Top:* from left to right: input matrices of CNN images with `threshold` $= 0.001, 0.4, 0.8$ and 1.0 (see equation (B.2)); *bottom:* accuracy as a function of `threshold`.

where $P_{\rho,i}$ is given by equation (22). We also define an additional parameter `threshold` as follows:

$$\begin{aligned} &\text{If} \quad && \bar{P}_{\rho,i} < \texttt{threshold}, \\ &\text{then,} \quad && \bar{P}_{\rho,i} = 0. \end{aligned} \tag{B.2}$$

As we increase the `threshold`, the error bars in the image are 'erased', as shown in the top panel of figure B1. When `threshold` $= 1$, we are left only with the pixels representing the mean value. The bottom panel of figure B1 shows how the accuracy changes as we vary this parameter. We see that the accuracy decreases as the error bars are discarded and, when the image contains only the mean values (`threshold` $= 1$), it approaches the accuracy obtained with CNN1D no-err.

## Appendix C. CNN architectures

In this appendix we discuss the details about the CNN architectures. The architectures developed for the baseline data set (2) are shown in table C1. The ReLU activation function is used in the convolution and dense layers, except in the output dense layer, where we use the softmax function. The pool size and strides in the first MaxPooling layer in the CNN 2D images architecture were chosen in such a way that the output feature map has size (10, 10).

We do not apply the architecture outlined in table C1 for all case studies because, as the data set becomes noisier, we verified that the baseline architecture tends to overfit the data [41]. Therefore, we adapt the model to avoid overfitting by either reducing the number of layers and neurons and/or by adding regularization (increasing the dropout rate). On the other hand, as the data becomes less noisy, the baseline models tend to underfit, so we modified the baseline architecture by increasing the number of neurons and/or reducing the dropout rate. In this way, we use the best CNNs for each case study, which avoid overfitting and underfitting, and thus have a fair application of the models to compare between different noise regimes.

We illustrate the above mentioned tendency of noisier data sets to overfit in figure C1, which shows the learning curves of the CNN 2D images model for the the baseline case (left), i.e. the case where the data generated with the baseline parameters outlined in 2 and the architecture shown in table C1 is applied; and the baseline architecture applied to the $\sigma_0 = 3.2$ data set (right). We see that when the baseline architecture is

**Table C1.** CNN1D and CNN2D images baseline architectures. These are the architectures applied to the baseline data set (see 2).

| CNN 1D | CNN 2D |
|---|---|
| Conv1D(filters = 32, kernel_size = (5,)) | Conv2D(filters = 32, kernel_size = (5, 5)) |
| BatchNormalization() | BatchNormalization() |
| MaxPooling1D(pool_size = 2, strides = 2) | MaxPooling2D(pool_size = (`n_rows`/10, 2), strides = (`n_rows`/10, 2)) |
| | Dropout(rate = 0.2) |
| Conv1D(filters = 64, kernel_size = (3,)) | Conv2D(filters = 64, kernel_size = (3, 3)) |
| BatchNormalization() | BatchNormalization() |
| MaxPooling1D(pool_size = 2, strides = 2) | MaxPooling2D(pool_size = (2, 2), strides = (2, 2)) |
| Dropout(rate = 0.2) | Dropout(rate = 0.4) |
| Conv1D(filters = 64, kernel_size = (3,)) | Conv2D(filters = 64, kernel_size = (3, 3)) |
| BatchNormalization() | BatchNormalization() |
| MaxPooling1D(pool_size = 2, strides = 2) | MaxPooling2D(pool_size = (2, 2), strides = (2, 2)) |
| Dropout(rate = 0.2) | Dropout(rate = 0.4) |
| Flatten() | Flatten() |
| Dense(units = 64) | Dense(units = 64) |
| Dense(units = 2, 'softmax') | Dense(units = 2, 'softmax') |

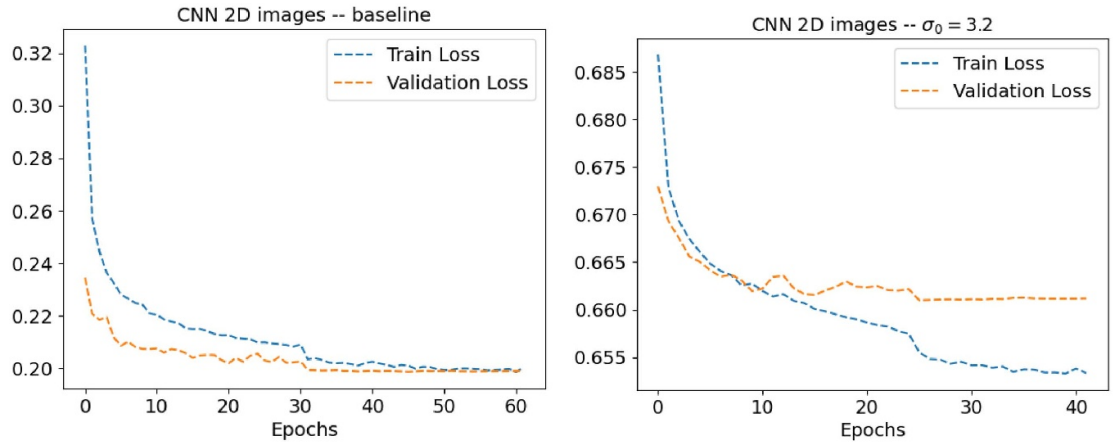


**Figure C1.** Learning curves of the CNN2D images model. Left: baseline data set ($\sigma_0 = 0.5$); right: $\sigma_0 = 3.2$.

applied to the $\sigma_0 = 3.2$, the difference between the training and validation losses is more pronounced. Here we show the case where the data set is noisier due to the increased value of the parameter $\sigma_0$, however the same care must be taken when the number of features $n$ is lower and when the training data set size $m$ is smaller.

## ORCID iDs

Natália V N Rodrigues https://orcid.org/0000-0002-5988-335X
L Raul Abramo https://orcid.org/0000-0001-8295-7022

## References

[1] Storrie-Lombardi M C, Lahav O and Sodre J L 1992 Morphological classification of galaxies by artificial neural networks *Mon. Not. R. Astron. Soc.* **259** 8
[2] Firth A E, Lahav O and Somerville R S 2003 Estimating photometric redshifts with artificial neural networks *Mon. Not. R. Astron. Soc.* **339** 1195–202
[3] Baldi P, Sadowski P and Whiteson D 2014 Searching for exotic particles in high-energy physics with deep learning *Nat. Commun.* **5** 4308
[4] Mehta P, Bukov M, Wang C H, Day A G R, Richardson C, Fisher C K and Schwab D J 2019 A high-bias, low-variance introduction to machine learning for physicists *Phys. Rep.* **810** 1–124
[5] Efron B 1986 Why isn't everyone a Bayesian? *Am. Stat.* **40** 1
[6] Tanabashi M *et al* 2018 Review of particle physics *Phys. Rev.* D **98** 030001
[7] Murphy K P 2012 *The Machine Learning: A Probabilistic Perspective* (MIT Press)

[8] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002
[9] Wolf C, Meisenheimer K, Rix H W, Borch A, Dye S and Kleinheinrich M 2003 The COMBO-17 survey: evolution of the galaxy luminosity function from 25 000 galaxies with $0.2 < z < 1.2$ *Astron. Astrophys.* **401** 73–98
[10] Scoville N *et al* 2007 The cosmic evolution survey (COSMOS): overview *Astrophys. J. Suppl. Ser.* **172** 1–8
[11] Moles M *et al* 2008 The Alhambra survey: a large area multimedium-band optical and near-infrared photometric survey *Astron. J.* **136** 1325–39
[12] Benitez N *et al* 2014 J-PAS: the Javalambre-physics of the accelerated Universe astrophysical survey (arXiv:1403.5237)
[13] Eriksen M *et al* 2019 The PAU survey: early demonstration of photometric redshift performance in the COSMOS field *Mon. Not. R. Astron. Soc.* **484** 4200–15
[14] Cenarro A J *et al* 2019 J-PLUS: the Javalambre photometric local Universe survey *Astron. Astrophys.* **622** A176
[15] de Oliveira C M *et al* 2019 The southern photometric local Universe survey (S-PLUS): improved SEDs, morphologies and redshifts with 12 optical filters *Mon. Not. R. Astron. Soc.* **489** 241–67
[16] Busca N and Balland C 2018 QuasarNET: human-level spectral classification and redshifting with deep neural networks
[17] Cabayol L *et al* 2018 The PAU survey: star-galaxy classification with multi narrow-band data *Mon. Not. R. Astron. Soc.* **483** 529–39
[18] Lovell C C, Acquaviva V, Thomas P A, Iyer K G, Gawiser E and Wilkins S M 2019 Learning the relationship between galaxies spectra and their star formation histories using convolutional neural networks and cosmological simulations *Mon. Not. R. Astron. Soc.* **490** 5503–20
[19] Sharma K, Kembhavi A, Kembhavi A, Sivarani T, Abraham S and Vaghmare K 2019 Application of convolutional neural networks for stellar spectral classification *Mon. Not. R. Astron. Soc.* **491** 2280–300
[20] Chang S, Cohen T and Ostdiek B 2018 What is the machine learning? *Phys. Rev.* D **97** 056009
[21] Rodrigues N V N *et al* 2023 The miniJPAS survey quasar selection—II. Machine learning classification with photometric measurements and uncertainties *Mon. Not. R. Astron. Soc.* **520** 3494–509
[22] Queiroz C *et al* 2023 The miniJPAS survey quasar selection - I. Mock catalogues for classification *Mon. Not. R. Astron. Soc.* **520** 3476–93
[23] Kendall A and Gal Y 2017 What uncertainties do we need in Bayesian deep learning for computer vision? *Proc. 31st Int. Conf. on Neural Information Processing Systems (NIPS'17)* (Curran Associates Inc.) pp 5580–90
[24] Abdar M *et al* 2021 A review of uncertainty quantification in deep learning: techniques, applications and challenges *Inf. Fusion* **76** 243–97
[25] Caldeira J and Nord B 2020 Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms *Mach. Learn.: Sci. Technol.* **2** 015002
[26] Le Q V, Smola A J and Canu S 2005 Heteroscedastic Gaussian process regression *Proc. 22nd Int. Conf. on Machine Learning (ICML '05)* (Association for Computing Machinery) pp 489–96
[27] Nix D and Weigend A 1994 Estimating the mean and variance of the target probability distribution *Proc. 1994 IEEE Int. Conf. on Neural Networks (ICNN'94)* vol 1 pp 55–60
[28] Reis I, Baron D and Shahaf S 2018 Probabilistic random forest: a machine learning algorithm for noisy data sets *Astron. J.* **157** 16
[29] Villacampa-Calvo C, Zaldivar B, Garrido-Merchán E C and Hernández-Lobato D 2020 Multi-class Gaussian process classification with noisy inputs
[30] Bi J and Zhang T 2005 Support vector classification with input data uncertainty *Advances in Neural Information Processing Systems* vol 17, ed L Saul, Y Weiss and L Bottou (MIT Press)
[31] Czarnecki W and Podolak I 2013 Machine learning with known input data uncertainty measure *IFIP Int. Conf. on Computer Information Systems and Industrial Management* vol 8104 pp 379–88
[32] Taylor J R 1997 *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements* (University Science Books)
[33] Bousquet O, Boucheron S and Lugosi G 2013 *Introduction to Statistical Learning Theory* (Springer)
[34] Acquarelli J, van Laarhoven T, Gerretzen J, Tran T, Buydens L and Marchiori E 2016 Convolutional neural networks for vibrational spectroscopic data analysis *Anal. Chim. Acta* **954** 22–31
[35] Ismail Fawaz H, Forestier G, Weber J, Idoumghar L and Muller P A 2019 Deep learning for time series classification: a review *Data Min. Knowl. Discov.* **33** 917–63
[36] Mozaffari M H, Tay L L 2020 A review of 1D convolutional neural networks toward unknown substance identification in portable Raman spectrometer
[37] Kawamura K, Nishigaki T, Andriamananjara A, Rakotonindrina H, Tsujimoto Y, Moritsuka N, Rabenarivo M and Razafimbelo T 2021 Using a one-dimensional convolutional neural network on visible and near-infrared spectroscopy to improve soil phosphorus prediction in Madagascar *Remote Sens.* **13** 1519
[38] Chollet F *et al* 2015 Keras (available at: https://keras.io)
[39] Kingma D P and Ba J 2017 Adam: a method for stochastic optimization *3rd Int. Conf. for Learning Representations* (*San Diego*, *2015*)
[40] Nair V and Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines *Proc. 27th Int. Conf. on Int. Conf. on Machine Learning. ICML'10* (Omnipress) pp 807–14
[41] Abu-Mostafa Y S, Magdon-Ismail M and Lin H T 2012 *Learning from Data* (AMLBook)
[42] Estrada J *et al* 2007 A systematic search for high surface brightness giant arcs in a sloan digital sky survey cluster sample *Astrophys. J.* **660** 1176–85
[43] Qu H, Sako M, Möller A and Doux C 2021 SCONE: supernova classification with a convolutional neural network *Astron. J.* **162** 67
[44] Shy S, Tak H, Feigelson E D, Timlin J D and Babu G J 2022 Incorporating measurement error in astronomical object classification *Astron. J.* **164** 6
[45] Breiman L, Friedman J H, Olshen R A and Stone C J 1984 *Classification and Regression Trees* (Wadsworth and Brooks)
[46] Bonoli S *et al* 2021 The miniJPAS survey: a preview of the Universe in 56 colors *Astron. Astrophys.* **653** A31