Machine Learning Classification of SDSS Transient Survey Images

L. du Buisson\textsuperscript{1,2,}*, N. Sivanandam\textsuperscript{2†}, B.A. Bassett\textsuperscript{1,2,3‡} and M. Smith\textsuperscript{4}

\textsuperscript{1}Department of Mathematics and Applied Mathematics, University of Cape Town, Cross Campus Rd, Rondebosch 7700, South Africa
\textsuperscript{2}African Institute for Mathematical Sciences, 6-8 Melrose Rd, Maitenbergh 7945, South Africa
\textsuperscript{3}South African Astronomical Observatory, Observatory Rd, Observatory 7925, South Africa
\textsuperscript{4}School of Physics and Astronomy, University of Southampton, Southampton, SO17 1BJ, United Kingdom

Accepted XXX. Received XXX; in original form XXX

ABSTRACT

We show that multiple machine learning algorithms can match human performance in classifying transient imaging data from the SDSS supernova survey into real objects and artefacts. This is the first step in any transient science pipeline and is currently still done by humans, but future surveys such as LSST will necessitate fully machine-enabled solutions. Using features trained from eigenimage analysis (PCA) of single-epoch $g$, $r$, $i$-difference images we can reach a completeness (recall) of 95\%, while only incorrectly classifying 18\% of artefacts as real objects, corresponding to a precision (purity) of 85\%. In general the k-nearest neighbour and the SkyNet artificial neural net algorithms performed most robustly compared to other methods such as naive Bayes and kernel SVM. Our results show that PCA-based machine learning can match human success levels and can naturally be extended by including multiple epochs of data, transient colours and host galaxy information which should allow for significant further improvements, especially at low signal to noise.

Key words: methods: data analysis, observational, statistical; techniques: image processing, photometric; surveys

1 INTRODUCTION

The quest to answer the deepest open questions about the cosmos has pushed astronomers and cosmologists to sample larger and larger volumes of the Universe. Current and next-generation of surveys, such as GAIA\textsuperscript{1}, the Dark Energy Survey (DES)\textsuperscript{2}, LSST\textsuperscript{3} and the SKA\textsuperscript{4} will usher in an era of exascale astronomy requiring new machine learning and statistical inference tools.

The LSST, for example, will image the night sky with such depth and frequency that upwards of a million transient alerts are expected every night with at least one million Type Ia supernova (SNIa) candidates detected over a decade of operations (LSST Science Collaboration et al. 2009). This will swamp existing spectroscopic follow-up capabilities, pushing us into the era of photometric transient identification trained on small spectroscopic subsets (Kunz et al. 2007; Campbell et al. 2013; Hlozek et al. 2012; Kessler et al. 2010; Gong et al. 2010; Newling et al. 2011; Knights et al. 2013). Such techniques will always lead to a small set of misidentifications and the danger is that the resulting contamination, unless dealt with in a sophisticated way, will lead to biased results.

However, long before one reaches the final scientific analysis, the data deluge create challenges in the analysis pipeline. For example, difference images are created by subtracting a reference image from the most recent image of a given part of the sky. In the ideal case this will leave a pure noise image unless a real transient such as a supernova, asteroid or variable star exists in the image. In reality there are inevitable artefacts that occur due to instrumental effects: diffraction spikes, CCD saturation and bleeding, registration errors and the like.

We thus need to disentangle the potential objects of interest from the artefacts. Historically this sorting and classification into real objects and artefacts has been done by astronomers scanning the images as soon as possible after the images having been taken. In the case of the SDSS supernova survey, this typically lead to hundreds or thousands of images being scanned each night; a tedious job. Recently it has been shown that this hand scanning can be done effectively by crowdsourcing (Smith et al. 2011); the public correctly identified 93\% of the spectroscopically-confirmed supernovae.

However, using machines to do this classification makes it very difficult to quantify the biases that arise from the subtly different algorithms and internal decision trees in each human scanner’s brain. In addition, the effective decision tree changes with time depending on the mood and tiredness of the hand-scanner \textsuperscript{5} which obviously

\begin{itemize}
\item \textsuperscript{*} E-mail: lisedubuisson@gmail.com
\item \textsuperscript{†} E-mail: navin.sivanandam@gmail.com
\item \textsuperscript{‡} E-mail: bruce.a.bassett@gmail.com
\item \textsuperscript{1} http://sci.esa.int/gaia/
\item \textsuperscript{2} http://www.darkenergysurvey.org/
\item \textsuperscript{3} http://www.lsst.org
\item \textsuperscript{4} http://https://www.skatelescope.org/
\end{itemize}

\textsuperscript{5} This is a well-known problem that affects even the most experienced
cannot be characterised systematically. This human bias was partially dealt with in the SDSS supernova survey by injecting fake SNe into the pipeline yielding an average detection efficiency for each scanner, but it is clearly a fundamental limitation of human hand scanning which is even worse for crowd-sourced classifications. Apart from these biases hand scanning will not be an option for LSST due to the millions of images that will need to be scanned each night.

Replacing humans with machine learning for this transient-artefact classification therefore represents an important frontier in achieving the goals of future transient surveys. Existing work includes the pioneering work of the SNfactory (Bailey et al. 2007; Romano et al. 2006) where features included the position, shape and FWHM of the main object, as well as distance to the nearest object in the reference image.

In contrast we use SDSS data and derive our features from principal component analysis of the Sloan $g$, $r$ and $i$ difference images. We compare a number of different potential machine-learning algorithms such as k-nearest neighbours, artificial neural network (SkyNet – Graff et al. (2014)), naive Bayes and Support Vector Machine (SVM) and show that it is possible to achieve human-levels of classification completeness/recall with limited degradation in purity.

In section 2 we describe the SDSS data used; the testing and performance measures in section 3 and the feature extraction and machine learning algorithms we employed are described in sections 4 and 5.

2 DATA

Our data is drawn from the 2nd and 3rd years of the SDSS supernova survey (Frieman et al. 2008; Sako et al. 2014) which operated in drift scan mode for three months each year from 2005-2007, alternately observing each of the two strips making up the approximately 300 deg$^2$ equatorial Stripe 82, weather permitting. In practise the mean cadence on any patch of sky was about four nights.

The transient detection algorithm consists of image subtraction of the search image from a historical image of the same region of the sky in the Sloan colour bands $g$, $r$, $i$. The difference image is flagged as a candidate for human scanning if it passes certain threshold cuts. However, this process was imperfect and led to a large number of artefacts that constituted 70% of the candidates or more.

A team of about 20 human hand scanners using both the search and difference images classified each candidate object into one of ten distinguishable classes: artefacts, dipoles, saturated stars, moving objects, variables, transients, SN Other, SN Bronze, SN Silver and SN Gold (Frieman et al. 2008). A description of each of these classes, as given by Bassett et al. (2005), can be found in Table 1. For our classification purposes, we re-group these original classes into three new visual classes, based on the observation that many of the classes have very similar visual appearances. These three classes are: real objects, artefacts and dipoles/saturated, illustrated in Fig. 1. Real objects have point-like residuals (convolved with the seeing from the atmosphere), artefacts have diffraction spike-like residuals while the dipoles/saturated class have residuals that are often close to point-like, but typically have negative flux in part of the difference image arising from registration errors or saturated CCD effects. Table 1 shows which of the original classes correspond to each of the three visual classes. This visual classification was used for carrying out PCA by class for feature extraction, as is discussed in section 4. We are ultimately concerned with whether we can correctly predict whether an object is real or not-real (artefact or dipole/saturated), however, and this is therefore the main classification that will be recognised by our classifiers. To see which classes correspond to real objects and which correspond to not-real objects, see Table 1.

The dataset consists of 27,480 images, of which 15,521 are real objects and 11,959 are not-real. These classifications are based on the classifications done by the SDSS hand scanners. About 2500 fake SNe were also inserted into the imaging pipeline in order to provide quality control and to characterise the selection function. Our dataset is the full set including fakes to allow us to compare with the human performance on the fakes.

Fig. 1 shows the three visual classes clearly. Fig. (1a) serves as an example of what high-quality images signal-to-noise ratios (SNRs) above 40, looks like, while Fig. (1b) shows images with SNR below 20, a threshold that represents roughly 80% of the objects in our dataset.

3 TESTING AND PERFORMANCE MEASURES

3.1 Testing

Our basic testing protocol was to withhold 25 percent of the data for method comparison after our various learners have been trained, cross-validated (if necessary) and undergone preliminary testing using the other 75 percent of the data. This 75/25 split was done prior to commencing building our various learning machines and the test data was sequestered until we were ready to perform our final testing.

Some of the classifiers have parameters that can be optimised using cross-validation. In general 30 percent of the training data was kept back to perform the optimisation. We stress here that the sequestering of the true test data is essential for the unbiased comparison of our different classifiers. If one uses the same data to optimize an algorithm as to test there is a strong risk of “training to the test set”, a phenomenon that will almost certainly lead to poor real world results.

3.2 Performance measures

For both testing and for optimisation we have used four measures commonly applied to classification problems. These are the accuracy ($A$), the precision ($P$), the recall ($R$) and the $F_1$ score. They are defined in terms of the number of true/false positives/negatives ($t_p$, $t_n$, $f_p$ and $f_n$) as follows:

\[ A = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \]  
\[ P = \frac{t_p}{t_p + f_p} \]  
\[ R = \frac{t_p}{t_p + f_n} \]  
\[ F_1 = \frac{2PR}{P + R} \]
Table 1. The different classifications for objects in our dataset. A description of each of the ten original classes used by the human hand scanners is given (Bassett et al. 2005), and their corresponding visual classification and main classification (real/not-real) is shown. See section 2 for more detail.

| Original Class       | Description                                                                 | Visual Class       | Real/Not-Real |
|----------------------|------------------------------------------------------------------------------|--------------------|---------------|
| Artefact             | Residuals caused by problems in the image (e.g. diffraction spikes).         | Artefact           | Not-Real      |
| Dipole               | Residuals with roughly equal amounts of positive and negative flux, caused by errors in image registration. | Dipole/Saturated   | Not-Real      |
| Saturated Star       | Residuals of stars that saturate the CCD.                                    | Dipole/Saturated   | Not-Real      |
| Moving               | Anything showing signs of motion between cutouts in different passbands.     | Real               | Real          |
| Variable             | Objects showing a record of long-term variability.                          | Real               | Real          |
| Transient            | Objects with no observation history, no apparent host galaxy in the search image, and no motion. | Real               | Real          |
| SN Other             | Objects that are thought to have a good chance of being SNe, but that don’t fit nicely into any of the above classes. | Real               | Real          |
| SN Bronze            | Point-like residuals at the centre of their host galaxies - most of the objects in this class later turn out to be either quasars (QSOs), active galactic nuclei (AGN) or foreground variable stars, and not SNe. | Real               | Real          |
| SN Silver            | Point-like residuals having no apparent host galaxy - SNe much more luminous than their host galaxies usually fall into this class. | Real               | Real          |
| SN Gold              | Possible SNe identified as point-like residuals that are not at the exact centre of their host galaxies. | Real               | Real          |

For our problem, we define the positive classes to be those corresponding to real objects.

The choice of which of these measures should be used to measure success is rather dependent on the problem at hand. Accuracy can be a misleading measure in situations where the number of positive and negative cases are vastly different, e.g. a classifier that always predicts 1 will have an excellent accuracy on a problem where 99% of cases are positive. For our data the classes are of similar sizes, so the accuracy is a reasonable initial measure. Somewhat better, however, is the $F_1$ score, which is expressed in terms of the precision (the fraction of reported real objects that are so) and the recall (the fraction of true objects that are found by the classifier). The $F_1$ score then is a measure that punishes false negatives and false positives equally, but weighted by their inverse fractional contribution to the full set to account for large class number hierarchies.

In real world applications, we are often more or less concerned with false positives than false negatives. For example, if the final number of items classified as objects is small enough to be easily verified by humans, the key will be to minimise $f_p$, and to maximise the recall. Conversely, if we are met with a data stream too large to be rechecked it may be more important to prevent contamination of data and so minimize $f_p$ and the precision.

We quote all of these statistics for completeness and also include the full confusion matrix, which contains the total numbers of true/false positives/negatives. An archetype is shown in Table 2.

Table 2. The schematic confusion matrix.

| Predicted Class | True Class | Not Object | Object |
|-----------------|------------|------------|--------|
|                 | $t_n$      | $f_p$      | $f_n$  |

Most classifiers can output the probability of an object being in a certain class and as a result we can study the trade-off between false positives and false negatives in a systematic way. The tool of choice for doing this is the Receiver Operating Characteristic (ROC) curve. This is a plot showing the performance of a binary classifier as its discrimination threshold is varied, and is created by plotting the True Positive Rate (TPR), or recall, versus the False Positive Rate (FPR) at various threshold values. The FPR,
also known as the fall-out, can be calculated as shown in Eq. 5.

\[ FPR = \frac{f_p}{f_p + f_n} \]  

(5)

Finally, the area under the ROC curve ("Area Under Curve" or AUC) is a statistic that is often used for model comparison in the machine learning community, and is equal to the probability that a classifier will correctly classify a randomly chosen instance (Hanley & McNeil 1982).

4 FEATURE EXTRACTION

Machine learning algorithms require features that represent the particular samples as their input. In the case of the transient classification problem our raw data are the pixel values for the images. In principle one could use the pixel values themselves as features, but this suffers from two problems. The first is a computational one; three \( 51 \times 51 \) pixel images (for all three passbands) would give 7803 features. The second issue is that the images are highly compressible, so by including all the pixel data we run the risk of swamping the classifier with irrelevant features and suffering from the curse of dimensionality.

To bypass these problems we used Principal Component Analysis (PCA) to extract those features that most faithfully represent the data. PCA involves an orthogonal transformation of the sample data (here the images are represented by their corresponding vectors in pixel space) into uncorrelated variables called principal components or eigenimages. The \( n \)th principal component gives the direction in which the variance of the data is the \( n \)th largest.

Thus we can reduce the number of features by only keeping a small fraction of all the PCA components, and still retain most of the information. In reality, for data with a non-zero mean (such as ours) the first PCA component will (approximately) correspond to the mean. One can remove this from the feature set by subjecting the data to mean subtraction before carrying out the PCA; we do not do this as the mean may contain classification information when the PCA is done on a class by class basis. It should be noted that PCA was done using all three bands (\( g, r \) and \( i \)). For detail on this classifier, see the book by Bishop (2006).

In Fig. (2a) one can see the first six principal components for the \( r \)-band images from the full training set. Beyond the first three eigenimages they appear to be noise but in fact the performance of several of the classifiers we tested improved significantly with the inclusion of higher order components. The optimal number of components to include depends on the classifiers and is, in general, a parameter optimized for each of our classifiers.

One potential issue with PCA for classification problems is that the direction of maximum variance may not align with the boundary between classes, the so-called decision boundary.

There are several ways to alleviate this problem. The simplest is to include more PCA components in the features – as \( n_{\text{components}} \) increases so does the probability that the information required to separate the samples is contained. One can also include components generated from Linear Discriminant Analysis (LDA) along with PCA features. LDA projects data in a direction that maximises the variance between classes while simultaneously minimising the variance within a class. LDA suffers from only being able to generate a number of components \( \leq n_{\text{classes}} - 1 \), so can be insufficient as a class separator if used alone. (See the book by Hastie et al. (2009) for detail on this method.)

A final approach to feature extraction is to apply PCA independently to each of the three visual classes of objects (see Table 1) in the training data, yielding a unique set of eigenimages for each class (see Fig. 2b, 2c and 2d). The feature vector for an image is then obtained by first reconstructing it using the PCA components of each class in turn, and then calculating the error (taken as the Euclidian distance) between each reconstructed image and the original image. The calculated errors are then used as features for the image.

After obtaining our sets of feature vectors from PCA (namely the PCA components and the reconstruction errors) and LDA our final step has been to normalize the data to give all the features a standard deviation of 0.5 and a mean of 0. This is done to improve the efficiency of the various routines that optimize the objective function for the different classifiers. As with the inclusion of the LDA component in the feature set, classifiers were trained both with and without feature normalisation for generality. The PCA and LDA algorithms implemented were taken from the work done by Pedregosa et al. (2011).

The various classifiers we have studied have been optimised by varying the feature set and carrying out intermediate testing. The final test results for a given classifier are reported using whichever feature set gave the best performance.
5 MACHINE LEARNING TECHNIQUES

5.1 Minimum Error Classification

Minimum Error Classification (MEC) is an extremely simple classifier, and is used here in order to form an idea of what the most basic of tools can achieve.

MEC takes only the three error-related features obtained from carrying out a PCA by class (as described in section 4), and assigns an object to the class corresponding to the minimum reconstruction error:

$$\min_i (|\vec{v}_{\text{true}} - \vec{v}_i|^2)$$

(6)

$\vec{v}_{\text{true}}$ is the vector in pixel space corresponding to the original image, and $\vec{v}_i$ is the reconstructed image vector assuming it is a member of the $i$th visual class.

Intuitively, this simply reflects the logic that an image should, on average, belong to the class with the best PCA reconstruction (smallest error). We found that the best results were obtained when three visual classes were used for the PCA-by-class: real objects, artefacts and dipoles/saturated (as discussed in section 2). A sample is then classified as not being an object if the minimum error corresponds with either the artefact class or the dipole/saturated class. It is classified as a real object otherwise.

Because there is no training process involved, MEC was tested directly on the final test data. The results, shown in Table 3, was obtained by using the central $31 \times 31$ pixel subimages and a non-normalised feature set of reconstruction errors.
5.2 Naïve Bayes

Bayesian reasoning dictates that all quantities of interest are controlled by their probability distributions, and that by reasoning about these probabilities while taking observed data into account, we can make optimal decisions. The importance of this reasoning to machine learning lies in its ability to provide a quantitative approach to weighing the evidence for alternative hypotheses (Mitchell 1997).

The Naïve Bayes (NB) classifier (Mitchell 1997) decides on the most probable class \(c_m\) via the computation:

\[
c_m = \arg\max_{c_j \in C} P(c_j | a_1, a_2, ..., a_n)
\]  

(7)

Here \(C\) represents the finite set of class values (real and not-real objects) and \(a_1, a_2, ..., a_n\) are the \(n\) image feature values. We then use Bayes theorem:

\[
P(h | D) = \frac{P(D | h) P(h)}{P(D)}
\]

(8)

where \(h\) is a given hypothesis and \(D\) is the data, and assuming that \(P(a_1, a_2, ..., a_n)\) is independent of \(c_j\), we find

\[
c_m = \arg\max_{c_j \in C} P(a_1, a_2, ..., a_n | c_j) P(c_j)
\]

(9)

Estimating the \(P(c_j)\) priors is easily done by counting the frequency with which each class \(c_j\) appears in the training data. To estimate the \(P(a_1, a_2, ..., a_n | c_j)\) terms in the same fashion, however, requires a very large training set, and is therefore usually not feasible when there are many features due to the exponentially large volume that needs to be sampled.

NB classification rests on the simplifying assumption that, given the class value \(c_j\), the feature values are conditionally independent, leading to the final expression:

\[
c_m = \arg\max_{c_j \in C} P(c_j) \prod_i P(a_i | c_j)
\]

(10)

Now, instead of having to estimate various \(P(a_1, a_2, ..., a_n | c_j)\) terms as before, we need only estimate \(P(a_1 | c_j)\) instead, a much more feasible process. Physically this corresponds to projecting the full posterior onto each of the features sequentially.

The major limiting issue for NB lies in its assumption of the conditional independence of features - something that is rarely true for real life problems and often leads to the degradation of the performance of the classifier.

Applying this method to the various training and validation data sets revealed that using the central \(31 \times 31\) pixel subimages with a non-normalised feature set comprising of an LDA component, 10 PCA weights and three reconstruction error values give the best results. The results obtained when applying NB to the test data can be seen in Table 3.

5.3 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is an exceptionally simple classifier. It finds the K-nearest training objects in feature space to a test point, averages their classes (with a uniform or distance weight) and classifies the test item accordingly (see the book by Hastie et al. (2009)).

The classifier has the advantages of being simple, inherently non-linear and free of intrinsic parameters. Performance can be poor in high-dimensional spaces however, but this was not a problem in our case. Initial training and testing were carried out with a variety of different image sizes and number of PCA components (and both with and without the LDA component). Of the variations, the best result seemed to come from using the central \(31 \times 31\) pixel subimages with 10 nearest neighbours, 53 features (50 PCA components and 3 reconstruction errors) and normalisation of the data. In this case the results for the training and validation sets were similar: accuracy 93%, precision 92%, recall 96% and F1-score 94%.

The results when this model was applied to the test data are given in Table 3. It should be noted that if we optimise for recall rather than accuracy, the recall reaches 95% with a precision of 87%. The KNN algorithm here implemented was taken from work done by Pedregosa et al. (2011).

5.4 Support Vector Machine

The Support Vector Machine (SVM) is a maximum margin classifier – it tries to find a decision boundary such that not only are the classes separated, but also that the separation is as large as possible. In the simplest implementation of an SVM, this decision boundary is a hyperplane in the feature space. The support of the SVM lies, however, in the ease with which the so-called kernel trick can be applied to the data that maps the features into a higher dimensional space in which the classes are well-separated by a hyperplane. An illustrative example of the kernel trick mapping a linear boundary into a curved boundary is shown in Fig. 3, allowing more successful separation of real from not-real objects. For detail on this classifier, see the book by Bishop (2006).

An SVM with a Gaussian (or radial basis function – RBF) kernel function has two parameters that need to be optimized – the width of the Gaussian (expressed in terms of \(\gamma = 1/2\sigma^2\), where \(\sigma\) is the width) and the soft margin parameter (C). The latter sets the trade-off between a smooth decision surface (small C, less biased) and a better fit (large C, more accurate). These parameters are optimised using a grid search on quasi-exponentially increasing parameters and \(n\)-fold cross-validation – the training set is split randomly into \(n\) equal sets, the classifier is trained on \(n-1\) of these, the error on the remaining set is calculated, the process is repeated for the \(n\) choices of the test sets, the average error is calculated and after repeating this procedure for every parameter value pair the C
and \( \gamma \) values that minimise the cross-validation error are selected. We chose to implement an SVM taken from the work done by Pedregosa et al. (2011).

After cross-validation (and trying a variety of different feature sets) the optimal results on the training data came from \( C = 1000 \) and \( \gamma = 0.1 \), using the central \( 31 \times 31 \) pixel subimages with a feature set consisting of 100 normalised components from PCA, one LDA component and three reconstruction errors. The results when this model was applied to the test data can be found in Table 3.

### 5.5 Artificial Neural Network

Loosely inspired by biological neural systems, artificial neural networks (ANNs) form a well-known class of machine learning algorithms. These networks consist of interconnected nodes, each of which receives and processes information before passing the result onwards to other nodes along weighted connections. Generally speaking, ANNs can be of arbitrary structure, but for many machine learning problems only feed-forward ANNs are required - networks that are directed from an input to an output layer of nodes and that have none, one, or many ‘hidden’ layers in between, as shown in Fig. 4. Using training data, the network ‘learns’ a mapping between the inputs and the outputs, and is then able to predict the outputs for new input test data. (Graff et al. 2014)

We used SkyNet as our ANN as it is a well-known and robust neural network training tool developed by Graff et al. (2014). It can train large and deep feed-forward ANNs for use in a wide range of machine learning applications, e.g. regression, classification and dimensionality reduction, to name but a few. SkyNet has a few very advantageous features: It allows the training of an autoencoder, a feed-forward ANN where the inputs are mapped back to themselves - these networks provide an intuitive approach to non-linear dimensionality reduction. It also allows the training of recurrent neural networks (RNNs), a class of ANNs in which connections between nodes form a directed cycle, creating a network of nodes with feedback connections. Furthermore, SkyNet employs an optional pre-training method that obtains network weights and biases close to the global optimum of the network objective function, instead of starting the training process from a random initial state. SkyNet also makes use of convergence criteria that prevents it from overfitting to the training data.

5.5.1 Number of hidden layers and nodes

A universal approximation theorem (Hornik, Stinchcombe & White 1989) says that feed-forward ANNs with at least three layers (one input, one hidden and one output layer), can approximate any continuous function to some given accuracy. This is the case for as long as the activation function is piecewise continuous, locally bounded, and not a polynomial - which is indeed the case for SkyNet.

Furthermore, empirical (Murtagh 1991) and theoretical (Geva & Sitte 1992) considerations suggest that the optimal structure for the approximation of a continuous function is through the use of one hidden layer with \( 2N + 1 \) nodes (\( N \) being the number of input nodes). It was decided that these guidelines would be followed when setting up the ANNs for our classification problem for as long as it remained computationally feasible.

5.5.2 Other network settings and classifier results

For our classification purposes SkyNet was not configured to act as either an autoencoder or a RNN, and the pre-training method on offer was not used. Testing showed that enabling these options did not result in any significant changes in the classification performance for the problem at hand, and they were therefore deemed unnecessary. Furthermore, the inputs to the network were left unwhitened\(^7\), and the average error-squared of the network outputs were used to determine convergence. For more details regarding these settings, see Graff et al. (2014).

We used the central \( 31 \times 31 \) pixel subimages and found that non-normalised feature sets comprising 200 PCA weights and three reconstruction error values did the best for the training and validation phase. The results for the corresponding test set can be seen in Table 3.

For most of the smaller feature sets that were tested, a hidden layer with \( 2N + 1 \) nodes were used, as previously discussed. For the larger feature sets (like the best-performing feature set mentioned above), however, that would result in a computationally unfeasibly large number of hidden nodes. In the case of the winning feature set (above), a hidden layer with 100 nodes was used, and 140 iterations were needed for the algorithm to converge.

6 RESULTS

6.1 Best Performing Classifier

For the supernova application we decided that accuracy and recall would serve as the most important performance metrics; see section 3.2. Because the two main classes in our data (real objects and not-real objects) are of similar sizes, accuracy serves as a good general measure of performance, while recall was chosen because we are more concerned with missing true objects (false negatives) than we are with contaminating our set of predicted objects with false positives since these are easy for humans to weed out.

The Receiver Operating Characteristic (ROC) curves for each of the five classifiers based on their performance on the final test set, is shown in Fig. 5, along with their Area Under the Curve (AUC) values.

\(^7\) Whitening here refers to the normalisation of the inputs to the network - an option when specifying settings in SkyNet. For more detail on how this is done for SkyNet, see Graff et al. (2014).
The ROC curve and Area Under Curve (AUC) for each of the five classifiers. The AUC is indicated in brackets next to each classifier’s name, and classifiers are listed in order from best-performing to worst-performing, based on the AUC statistic. It should be noted that changing the threshold value of KNN, for example, can easily result in a recall (true positive rate) of ∼ 96% with only a slight penalty in the false positive rate, an encouraging sign.

Table 3 gives a summary of the results of the different classifiers, and lists them in order from best-performing to worst-performing based on accuracy, recall and AUC value. It is interesting to note that KNN, a simple nonparametric classifier, performed best in all metrics except recall, closely followed by SkyNet and SVM. Interestingly, the simplest algorithm, MEC, had the best recall out of the box, though it lagged significantly in its true positive rate at small values of the false positive rate; Fig. 5.

How much can we trust the results in Table 3? One approach is to estimate the sample variance errors on the accuracies, recalls and precisions for each of the classifiers by using bootstrap resampling. The results produced through this procedure suggest sample variance errors of about ±0.01 at 95% confidence, and hence our rankings of classifier performances are reasonably robust. However, the differences between the best classifiers (SVM, KNN and SkyNet) are small and it is almost certainly the case that better optimisation of any of the algorithms could change the final ordering we obtained. However, from the point of view of replacing human image scanning, it is encouraging that multiple algorithms did very well.

### 6.2 Inter-classifier Agreement

To form an idea of the extent of inter-classifier agreement in terms of images correctly and incorrectly classified, we calculate the Cohen Kappa coefficient κ (Cohen 1960), a statistical measure of inter-classifier agreement argued to be more robust than a simple percent agreement calculation, as it also takes into account chance agreement. Cohen’s Kappa measures the agreement between two classifiers that each classify N items into C mutually exclusive classes, and is calculated as:

\[
\kappa = \frac{P(a) - P(b)}{1 - P(b)}
\]  

where \(P(a)\) is the observed relative agreement and \(P(b)\) is the hypothetical probability of chance agreement. If the classifiers agree perfectly, \(\kappa = 1\), whereas if there is no agreement between classifiers except for that which would be expected by chance alone, then \(\kappa = 0\). Statisticians differ slightly on the exact interpretation of different values of κ (see the guidelines given by Landis & Koch (1977), Attman (1991) and Fleiss et al. (2003), respectively), but for a rough idea as to what these interpretations are, see Table 4, the guidelines given by Landis & Koch (1977).

Fig. 6 shows the κ-values calculated for all pairs of classifiers, which measures the overlap in performance between classifiers relative to pure chance (which corresponds to \(\kappa = 0\)). These results can be interpreted with the aid of Table 4.

### Table 4. Guidelines to the strength of agreement between two classifiers based on the value of the Cohen’s Kappa coefficient κ (Landis & Koch 1977).

| Kappa Statistic | Strength of Agreement |
|-----------------|-----------------------|
| \(\kappa \leq 0.00\) | None |
| \(0.00 < \kappa \leq 0.20\) | Slight |
| \(0.20 < \kappa \leq 0.40\) | Fair |
| \(0.40 < \kappa \leq 0.60\) | Moderate |
| \(0.60 < \kappa \leq 0.80\) | Substantial |
| \(0.80 < \kappa < 1.00\) | Almost Perfect |
| \(\kappa = 1.00\) | Perfect |

This involves drawing samples (with replacement) from the test set in order to build up an ensemble of test sets.
Table 3. A summary of the performance results of the various classifiers, ordered from best-performing (top) to worst-performing (bottom). The best result for each performance metric is indicated in bold. The true labels are on the vertical side of the confusion matrices, and the predicted labels are on the horizontal side (see Table 2).

| Machine Learning Technique | AUC | Accuracy | Recall | Precision | F1-score | Confusion Matrix |
|----------------------------|-----|----------|--------|-----------|----------|-----------------|
|                            |     |          |        |           |          | Not Object       |
|                            |     |          |        |           |          | Object          |
| KNN                        | 0.94| 0.89     | 0.90   | 0.91      | 0.90     | 2628 363 |
| SkyNet                     | 0.94| 0.88     | 0.89   | 0.90      | 0.89     | 2592 399 |
| SVM                        | 0.93| 0.86     | 0.90   | 0.85      | 0.87     | 2386 605 |
| MEC                        | 0.90| 0.84     | 0.92   | 0.83      | 0.87     | 2237 754 |
| NB                         | 0.81| 0.78     | 0.77   | 0.84      | 0.80     | 2141 577 |
|                            |     |          |        |           |          | Object          |

6.3 Classifier Performance on Different Classes of Images

The performance of the classifiers on the various different visual classes (real objects, artefacts and dipoles/saturated) in the best performing test data sets is shown in Table 5 (note that recall, here, has the same meaning as accuracy). For the calculation of the recall values, it should be noted that true positives ($t_P$) here correspond to the objects in a certain class classified correctly as either real or not-real, while false negatives ($f_n$) correspond to incorrect classifications.

For all classifiers the performance on real objects and artefacts is higher than that on the dipoles/saturated class, the latter therefore being the class that lowers the overall performance of the classifiers. The relatively poor performance on this class can be understood by noting that, firstly, the objects have features that can be quite similar to those of real objects (this can be seen when comparing the PC’s in Fig. 2d to those in Fig. 2b), and will therefore confuse classifiers. On further visual inspection it can be noted that the images in this class (in general) shows a greater diversity in features than real object images do, leading to there not being as strong a common feature to characterise almost all of the images with, as is the case with real object images. When an object in the dipole/saturated class is encountered that looks similar to a real object (this is often the case for noisy images), it can therefore easily be misclassified as strong features of the real object images may approximate the dipole/saturated image more closely than weaker/mixed features from the dipole/saturated class itself. Section 6.4 further discusses misclassification issues.

6.4 Examples of Incorrectly Classified Objects

To give an idea of what causes certain images to get misclassified, we inspected images that were each incorrectly classified by all five classifiers. Fig. 7 shows examples of real objects that were classified as being not-real and Fig. 8 shows examples of not-real objects that were classified as being real.

In Fig. (7a) it is observed that all three passbands have a faint point-like structure at the centre of their images; this is the reason for the object’s ‘real’ label, as these point-like residuals point towards there being a possible SN. The reason for its misclassification as a not-real object is probably due to the much brighter dipole-like residual in the top right corner of all the images, something that would confuse the classifiers since we did not force the classifiers to focus only on the central few pixels. Fig. (7b) and (7c) both show dipole-like structures in all three bands instead of the usual point-like residuals characteristic of real objects - it is no wonder then that all five classifiers misclassified these objects as being not-real. The original classification of these two objects as real might even be a mistake on the part of the hand scanners, which emphasises how errors in the labelling of the training data can propagate through to the test set.

The same sort of problem is evident in Fig. (8a) where human scanners classified an object as an artefact that the classifiers understandably mistook for a real object due to the point-like residuals observed in all bands. Fig. (8b) is an artefact with masked areas in all three bands causing the object to lose most of its spike-like attributes, something very characteristic of the artefact subclass. The
Figure 7. Examples of objects classified as real by the human scanners but classified as not-real by all five machine classifiers; for each image the g-band (top), i-band (middle) and r-band (bottom) image is shown: (a) the dipole-like residuals in the top corner outshining the faint point-like structures probably resulted in misclassification; (b&c) the dipole-like structures observed in the centre of these real object images probably resulted in their misclassification.

fact that all five classifiers classified this object as real is possibly mostly due to the point-like structure at the centre of the i-band image. Fig. (8c) is a dipole, as can be seen by the presence of dipole-like residuals in the i and r-bands. The fact that these images are quite noisy (especially the g-band image) possibly contributed to the fact that all classifiers predicted it to be a real object. Furthermore, the features of the real and dipole/saturated objects are quite similar in some cases, a possible further contribution to the misclassification of the object as real instead of not-real (this problem was discussed in section 6.3).

6.5 Comparison with Human Scanners

As part of the SDSS SN survey fake SNe were inserted into the pipeline to test the efficiency of the hand scanners. The fake SN-tag recall, averaged over scanners, was 0.956 ± 0.010 (Kessler 2007). In comparison, Table 6 shows the classification performance of the various classifiers on the same fake SNe found in the test set. It can be seen that all the algorithms other than Naive Bayes perform comparably to, or even slightly better than, the average human scanner on the fake SNe. Furthermore, when looking at Fig. 5, it can be seen that by changing the discrimination threshold value and inducing only slight penalties (in the form of a larger FPR), a recall (TPR) value of 96% can easily be achieved even when all the test data is used.

Table 6. A summary of the performance results of the various classifiers on the fake SNe found in the test set.

| Machine Learning Technique | Recall |
|----------------------------|--------|
| KNN                       | 0.96   |
| SkyNet                    | 0.96   |
| SVM                       | 0.94   |
| MEC                       | 0.96   |
| NB                        | 0.85   |

neighbours. The impressive success of the linear minimum error classifier suggests that much of the heavy lifting here is done by the judicious use of (class-based) PCA as the basis for feature extraction.

As well as the obvious advantages that machine learning techniques offer in terms of handling large volumes of data, we also stress that automated classifiers have quantified and controllable errors and biases. This latter feature will be essential for building pipelines for data analysis which fully propagate all systematic errors in a reproducible way.

Having illustrated that PCA feature extraction plus a simple classifier provides a robust transient identification solution, the next step would be to build this approach into the data analysis for current and next-generation experiments. This will involve tailoring and optimizing the classification pipeline for further performance enhancements. One option would be to combine classifiers (through e.g. boosting or a weighted voting system): given the fact that the classifiers do not agree that strongly (see Fig. 6) this could be a fruitful avenue to explore. Another strategy would be to try and improve the efficiency for identifying dipole/saturated images (consistently the worst classified category) by augmenting the feature set.

Finally we note that since we have only used difference images we have not used any host galaxy information, nor have we...
used the relative colours of the bands nor multi-epoch data. All of these can be studied and will prove particularly useful for general-purpose surveys that look to classify objects beyond just two or three classes.

ACKNOWLEDGEMENTS

LdB is supported by a South African Square Kilometre Array (SKA) Project postgraduate scholarship; NS is supported by a Claude Leon Foundation fellowship; BB acknowledges funding from the South African National Research Foundation (NRF) and SKA; MS is supported by the South African SKA Project, the NRF and the UK Science & Technology Facilities Council (STFC).

REFERENCES

Altman D. G., 1991, Practical Statistics for Medical Research. Chapman & Hall, London
Bailey S., Aragon C., Romano R., Thomas R. C., Weaver B. A., Wong D., 2007, ApJ, 665, 1246
Bassett B. A., Richmond M., Frieman J., 2005, SDSS-II SN Super-unofficial Hand-Scanning Guide, http://spiff.rit.edu/richmond/sdss/sn_survey/scan_manual/sn_scan.html
Bishop C. M., 2006, Pattern Recognition and Machine Learning. Springer, New York, NY
Campbell H., D’Andrea C. B., Nichol R. C., Sako M., Smith M., Lampeitl H., Olmstead M. D., Bassett B., et al., 2013, ApJ, 763, 88
Cohen J., 1960, Educational and Psychological Measurement, 20, 37
Danziger S., Levav J., Avnaim-Pesso L., 2011, PNAS, 108, 6889
Fleiss J. L., Levin B., Cho Paik M., 2003, Statistical Methods for Rates and Proportions. John Wiley & Sons, Inc., Hoboken, New Jersey & Canada
Frieman J. A., Bassett B., Becker A., Choi C., Cinabro D., Depoy D. L., Dilday B., et al., 2008, AJ, 135, 338
Geva S., Sitte J., 1992, Neural Networks, IEEE Transactions, 3, 621
Gong Y., Cooray A., Chen X., 2010, ApJ, 709, 1420
Graff P., Feroz F., Hobson M. P., Lasenby A., 2014, MNRAS, 441, 1741
Hanley J., McNeil B. J., 1982, Radiology, 143, 29
Hastie T., Tibshirani R., Friedman J., 2009, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY
Hlozek R., Kunz M., Bassett B., Smith M., Newling J., Varughese M., Kessler R., Bernstein J. P., et al., 2012, ApJ, 752, 79
Hornik K., Stinchcombe M. B., White H., 1989, Neural Networks, 2, 359
Kessler R., 2007, Private Communication
Kessler R., Bassett B., Belov P., Bhatnagar V., Campbell H., Conley A., Frieman J. A., Glazov A. a. o., 2010, PASP, 122, 1415
Knights M., Bassett B. A., Varughese M., Hlozek R., Kunz M., Smith M., Newling J., 2013, JCAP, 1, 39
Kunz M., Bassett B. A., Hlozek R. A., 2007, Phys. Rev. D, 75, 103508
Landis J. R., Koch G. G., 1977, Biometrics, 33, 159

© 2014 RAS, MNRAS 000, 1–??