Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **512,** 1696–1709 (2022) https://doi.org/10.1093/mnras/stac480
Advance Access publication 2022 February 25

Deep learning methods for obtaining photometric redshift estimations from images

Ben Henghes [1]★ Jeyan Thiyagalingam,[2] Connor Pettitt,[2] Tony Hey[2] and Ofer Lahav[1]

[1]*Department of Physics & Astronomy, University College London, Gower Street, London WC1E 6BT, UK*
[2]*Scientific Computing Department, Rutherford Appleton Laboratory, Science and Technology Facilities Council (STFC), Harwell Campus, Didcot OX11 0QX, UK*

Accepted 2022 February 17. Received 2022 February 17; in original form 2021 September 6

ABSTRACT

Knowing the redshift of galaxies is one of the first requirements of many cosmological experiments, and as it is impossible to perform spectroscopy for every galaxy being observed, photometric redshift (photo-z) estimations are still of particular interest. Here, we investigate different deep learning methods for obtaining photo-z estimates directly from images, comparing these with 'traditional' machine learning algorithms which make use of magnitudes retrieved through photometry. As well as testing a convolutional neural network (CNN) and inception-module CNN, we introduce a novel mixed-input model that allows for both images and magnitude data to be used in the same model as a way of further improving the estimated redshifts. We also perform benchmarking as a way of demonstrating the performance and scalability of the different algorithms. The data used in the study comes entirely from the Sloan Digital Sky Survey (SDSS) from which 1 million galaxies were used, each having 5-filtre (ugriz) images with complete photometry and a spectroscopic redshift which was taken as the ground truth. The mixed-input inception CNN achieved a mean squared error (MSE) =0.009, which was a significant improvement (30 per cent) over the traditional random forest (RF), and the model performed even better at lower redshifts achieving a $MSE = 0.0007$ (a 50 per cent improvement over the RF) in the range of $z < 0.3$. This method could be hugely beneficial to upcoming surveys, such as Euclid and the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST), which will require vast numbers of photo-z estimates produced as quickly and accurately as possible.

Key words: methods: data analysis – galaxies: distances and redshifts – cosmology: observations.

1 INTRODUCTION

In the past decade, the number of galaxies observed by large sky surveys has been rapidly increasing with hundreds of millions of galaxies being imaged by surveys such as the Dark Energy Survey (DES; DES Collaboration 2016), the Kilo-Degree Survey (De Jong et al. 2013), and Hyper Suprime-Cam (HSC; Aihara et al. 2018). This ever growing number is set to rise even faster with upcoming surveys such as Euclid (Amendola et al. 2018), the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST; Tyson et al. 2003), and the Roman Space Telescope (formerly WFIRST – the Wide-Field Infrared Survey Telescope; Spergel et al. 2015) which will observe many billions of objects. For most cosmological studies in which galaxies are used, the redshift is a key property that is required; however, despite spectroscopic surveys, such as the Dark Energy Spectroscopic Instrument (Flaugher & Bebek 2014; Martini et al. 2018), only a very small fraction of galaxies have an associated spectroscopic redshift. Instead, photometric redshift (photo-z) estimations are necessary.

These photo-z estimates can be obtained using two different methods (or a combination of both): template fitting, or machine learning (ML). The template fitting method uses templates of the spectral energy distribution (SED), and by fitting the observed SED of the galaxy to the template, its redshift can be inferred (Bolzonella, Miralles & Pelló 2000). ML methods instead use a large training set of galaxies with labelled 'true' values for the redshift (the spectroscopic redshift) and learn a mapping from the features of the galaxy data to their redshifts (Collister & Lahav 2004). For traditional ML techniques, such as random forests (RF) or k-Nearest Neighbours (kNN), these features are taken from the photometry, giving magnitudes in different filters, which can be combined into colours. However, for deep learning methods, the image itself can be used as the input, with the pixel values being akin to features (Hoyle 2016).

There are benefits to each method with template fitting being an inherently physical model, and by making use of SED templates that are full spectra, they can be shifted to any redshift and allow for redshift estimates to be obtained in ranges without large spectroscopic data sets (Benitez 2000). ML methods on the other hand require a representative training sample with the same redshift distribution as the targets, and as a result are only valid in that range. Despite this limitation, ML has been increasingly implemented as a faster method, which is able to produce very accurate photo-z estimates where there is a sufficiently large training set (Abdalla et al. 2011).

Recently great strides have been made in the field of deep learning, aided by improving computer architectures and faster graphics processing units (GPUs), far more difficult tasks can now be

★ E-mail: ben.henghes.13@ucl.ac.uk

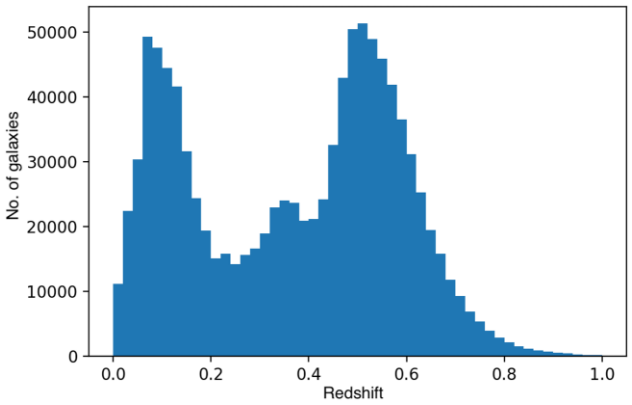

Figure 1. Plot showing the redshift distribution of the 1 million galaxies used in the study. The histogram was plotted with 50 redshift bins between $z = 0$ and $z = 1$ and displays the overall redshift distribution of galaxies in SDSS with the two peaks caused by the difference between the galaxies observed during the main galaxy survey and BOSS as described by Beck et al. (2016).

performed using much larger data sets (Kirk 2007). In industry, many companies have been taking advantage of these methods for tasks which range from creating outfits for fast-fashion (Bettaney et al. 2019), to self driving vehicles (Bojarski et al. 2016). In astronomy, upcoming surveys, such as *Euclid* (Amendola et al. 2018), *LSST* (Ivezić et al. 2019), the *Roman Space Telescope* (Spergel et al. 2015), and the *Square Kilometer Array* (SKA) (Dewdney et al. 2009), will be producing petabytes of data and ML provides a viable solution to the otherwise unimaginable task of data analysis on such a scale.

In addition to the difficult task of processing data on these scales, the photometric redshifts produced must also be highly accurate. Many cosmological analyses, such as weak-lensing-driven cosmology (DES Collaboration 2021; Heymans et al. 2021), rely on having extremely low errors in the redshift estimates to allow for more confident predictions of the cosmological parameters (Hildebrandt et al. 2021; Myles et al. 2021). Indeed, LSST have stated in their science requirements that the error on the mean redshift must be below 0.003 (Mandelbaum et al. 2018). Greater accuracy may naturally be achieved through larger, or novel deep learning methods, but also through increased emphasis on model interpretability and robustness to input error.

There are many other benefits to being able to directly use images rather than photometric features for photo-z estimation, predominantly that the deep learning algorithms could extract far more information than from the magnitudes alone (Pasquet et al. 2019; Schuldt et al. 2020). Indeed previous studies such as Soo et al. (2017) worked hard to include morphological parameters that are implicitly contained in the images and that the deep learning algorithms could extract. Furthermore, other work has been done which found that deep learning methods have been able to produce photo-z estimates which outperform the previously best performing traditional methods based on kNN or RFs (D'Isanto & Polsterer 2018). However, the deep learning algorithms are often much slower and more computationally expensive to run, and there has been little investigation into whether the benefits of these methods is worthwhile in the long-term.

Other studies have also demonstrated methods for estimating probability density functions (PDFs) which can be desirable for cosmological analyses. Cavuoti et al. (2017) introduced a binned method where the input data point is modulated, and the relative frequencies of binned output estimates converted to probability densities. Pasquet et al. (2019) showed that a classifier can be used in a similar manner where each output 'class' is treated as a part of a binned redshift distribution, while the softmax of the output is taken to represent a probability density. We treated this problem of photo-z estimation as a regression problem, where the models produced a point estimate for the redshift. This allowed us to demonstrate how the novel mixed-input methods performed and act as a proof of concept, as well as simplify the comparison between the deep learning algorithms and traditional methods.

Here, we investigate if it is worth using deep learning on images compared to traditional ML using only the magnitudes as features, applying different types of convolutional neural networks (CNNs), as well as mixed-input models, which use both images and magnitudes as inputs. In Section 2, we describe the data collected and used to train and test the ML algorithms, which are outlined in Section 3, along with the metrics and optimization process. We then present the results in Section 4 with discussions about how the mixed-input inception CNN was able to achieve such low errors and outperform all the other algorithms, before concluding in Section 5.

2 DATA

The data used to train and test the different ML algorithms came entirely from the Sloan Digital Sky Survey data release 12 (SDSS DR-12; York et al. 2000; Alam et al. 2015). In this work, we compiled 1059 678 data points each representing a galaxy. We downloaded the Petrosian magnitudes and spectroscopic redshifts, then acquired their corresponding images, each one comprising the five wavelength bands (u, g, r, i, and z). It was a requirement to have spectroscopy performed to be able to use the spectroscopic redshift as the ground-truth, and in order to directly compare methods, the photometric magnitudes were also required to use as features for both the traditional ML algorithms and mixed-input models.

While the total number of galaxies which met the requirement of having an associated spectroscopic redshift was closer to 2 million (Beck et al. 2016), we decided that a training set of 1 million galaxies was sufficient. This decision was made as there would not be a large difference in error performance going from 1 to 2 million galaxies. Furthermore, the scaling of the algorithms would only be visibly different with changing orders of magnitude of the training set , and therefore simulations would have then been required to be able to include more galaxies and reach the next order of magnitude.

The data set used was also kept clean by requiring complete photometry where there were no missing values of any magnitudes which could have negatively impacted the redshift estimations and biased the results. Furthermore, the redshift range was kept to only include galaxies with $z < 1$, with the final distribution of galaxies used shown in Fig. 1. Although this simplified the problem rather than having a larger redshift range, this distribution matched that of the overall SDSS survey as shown by Beck et al. (2016), and as the data used were representative of SDSS, it therefore allowed for valid estimates to be made in this range.

To generate the images used by the deep learning algorithms, we first downloaded the full frames made available by SDSS which each comprised the five flexible image transport system (fits) files for the five different filters. For each galaxy within the frame, we then generated 32 × 32 pixel images by centering the frame on the galaxy and cropping. We chose 32 × 32 pixel images as they were found to be sufficiently large to contain the full galaxy and surrounding sky for even the closer galaxies. When compared with 64 × 64 pixel images

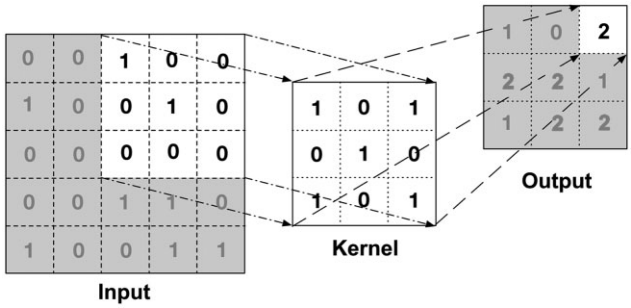

Figure 2. In this figure, we illustrate how a convolutional filter is applied to an input as a part of a CNN. The filter (or kernel) is applied across the input, being shifted by the stride (here =1) each time before calculating the dot product to produce the output.

which other studies had previously used, the smaller galaxy images allowed for much smaller file sizes and more efficient computations, as well as decreasing the incidence of image contamination by other galaxies. Tests were conducted using even smaller 20 × 20 pixel images, however, they resulted in worse performances by the CNNs, and as a result the 32 × 32 images were used throughout.

The final result was a set of five 32 × 32 pixel images for each galaxy which we saved in a numpy array (Harris et al. 2020) with shape (32 × 32 × 5) which could then be used with the deep learning PYTHON package KERAS (Chollet et al. 2015), our chosen interface for TENSORFLOW (Abadi et al. 2015).

3 METHODS

With the magnitude data and images downloaded in a usable format, we were then able to apply the ML algorithms. The algorithms tested were as follows. A CNN and an inception module CNN which both used the image data as the input. A RF and extremely randomized trees (ERT) that had previously been found to be the best performing traditional methods (Henghes et al. 2021) and which only used magnitude features. And two experimental mixed-input models, which combined a CNN or inception module CNN with a multilayer perception to use both the image data and magnitude features as inputs. Each of the algorithms is described in more detail below.

The algorithms were trained multiple times while varying the size of the training set used (up to 1 million galaxies), and tested on a fixed test set of 59 678 galaxies. By doing this we were able to benchmark the different algorithms to determine their scalability as described in Section 3.8. The optimization process is also described in Section 3.7 and the metrics used to evaluate the models' performances are given in Section 3.6.

3.1 Convolutional neural networks

Artificial neural networks are algorithms which aim to mimic the human brain (McCulloch & Pitts 1943). They are made up of many interconnected nodes called neurons which, similar to their biological counterparts, are used to signal and process information. This is done by taking the inputs, which can either be feature values from the data (in this case the pixel values from the images), or outputs from other neurons, and then calculating the weighted sum of of these inputs. The weights are taken from the connections between neurons and are the key property which is varied during the training process to learn the best possible network. A bias is then also added to the weighted sum which is finally passed to an activation function that produces the output.

Neural networks are comprised of multiple layers with a minimum of an input layer and output layer, but typically with at least one 'hidden' layer of nodes in-between, and in the case of deep neural networks many hidden layers are employed (LeCun, Bengio & Hinton 2015). The neurons of each layer are connected to some (or all in the case of fully connected networks) of the neurons from the previous layer. CNNs (Fukushima & Miyake 1982) are a type of deep neural networks that include convolutional layers. They were also inspired by a biological counterpart in the visual cortex where neurons only respond to stimulus in a specific region called the receptive field and the receptive fields of neurons then overlap to cover the entire region (Hubel & Wiesel 1968).

In CNNs, the convolutional layers act to convolve the input in a similar way to the visual cortex. The layer works by using filters (or kernels), which are initially random values that get updated during training to provide more relevant values. Each filter is applied across the volume of the input data as shown in Fig. 2, computing the dot product between the filter values and input values, and produces an activation map (or feature map). This activation map highlights regions of the input, learning features or patterns in the input images, and the maps from all filters are stacked to form the output of the convolutional layer.

As well as being able to learn features directly from the input image data, CNNs allow for much smaller overall network architectures, as filters require far fewer learnable parameters compared with the equivalent fully connected network. This makes them perfectly suited to image data where large images would quickly result in exploding gradients in a traditional neural network. Furthermore, the spatial relationship between features is preserved by the convolutional layers, so for data where spatial information is important (like images), CNNs are a natural choice.

After the convolution, an activation function is used which acts to introduce nonlinearities and decides which neurons fire. The most common activation function is the rectified linear unit (ReLU; Nair & Hinton 2010), which is defined as

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}. \tag{1}$$

While the sigmoid function and hyperbolic tangent have historically been popular choices for the activation function, ReLU sets negative values to 0 and is much faster without decreasing accuracy. Recently an adapted parametric ReLU was introduced (He et al. 2015), which is an example of a 'leaky' ReLU, defined as

$$f(x) = \begin{cases} ax & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}, \tag{2}$$

where the coefficient of leakage, a, is an extra parameter that is also learned.

Another type of layer which is used in CNNs is the pooling layer. Pooling layers, like the one shown in Fig. 3, reduce the size of activation maps by combining the outputs within a set space (typically 2 × 2). The two types of pooling take either the maximum value (max-pooling) or average value (average-pooling) within this space, and by down-sampling, these layers act to further reduce the computational demand of the CNN, as well as reduce overfitting.

The final layers which make up most CNNs are fully connected layers. Unlike the convolutional layer, the neurons of fully connected layers do not share weights and are connected to every other neuron

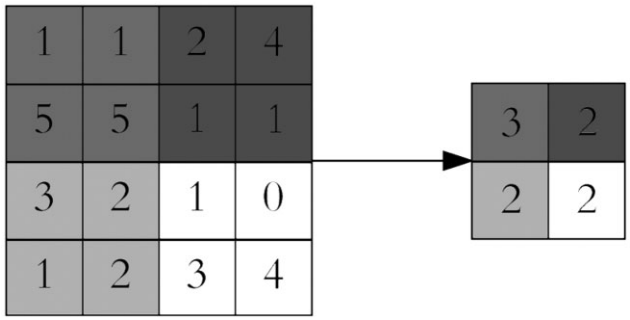

Figure 3. In this figure we show how the down-sampling is performed by a (2×2) average pooling layer (with stride =2), and how it acts to reduce the size of the input.

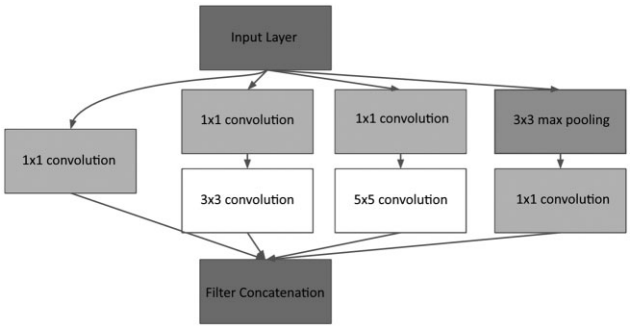

Figure 4. Figure showing a typical inception module which makes up a part of the inception module CNN with each layer and kernel size labelled. The inception module is made of several convolutions, as well as a pooling layer, each applied in parallel with the (1×1) convolutions acting to improve the computational efficiency. The outputs are then concatenated to produce the output of the inception module which can then be fed into the CNN similar to any other layers.

in the previous layer. Typically these final fully connected layers behave in a similar way to a traditional neural network, taking the features extracted by the convolutional layers and further processing the information to reach the final predictions.

The full CNN architecture used in this study is shown in Fig. A1 and made use of two convolutional layers, each followed by an average pooling layer, to extract the information from the images. It then fed the flattened 4096 long array into two dense layers, with 1024 followed by 32 neurons, before a final dense layer was used to give the single value output for the predicted redshift. This architecture was found to be the most effective simple CNN, and while deeper models were also tested, we found that the best performance occurred with only a few layers.

3.2 Inception modules

Many variations of CNNs are possible, one example is the inception module CNN which makes use of inception modules. These are sets of convolutional layers applied in parallel rather than sequentially, as displayed in Fig. 4. Using convolutional layers with a kernel size of (1×1) before then applying larger (3×3) or (5×5) kernels allows for more efficient computation for deeper networks (Szegedy et al. 2015).

In the inception module CNN implemented here (shown in Fig. A3), we used two different sets of inception modules. The first had the same architecture as displayed in Fig. 4, however, after four sets of these inception modules the resulting output was too small to usefully apply a (5×5) kernel to, and instead we simply removed this layer from the inception module. Following this smaller inception module, the flattened 192 long array was fed into two further dense layers each with 1096 neurons before the final dense layer gave the predicted redshift.

Unlike the simple CNNs tested, when testing different configurations of inception module CNNs, we found that the deeper models performed best. As well as changing the depth of the network we also tested different inception modules, changing the size of the kernels and swapping the max pooling layer for an average pooling layer. This resulted in a somewhat different model to that used by Pasquet et al. (2019) while still resembling the original GoogLeNet network.

3.3 Mixed-input models

It is also possible to include multiple inputs in a CNN. This works by defining separate input data and running through networks in parallel before then combining the networks by concatenating similar to how the inception modules are built. In this work we experimented with two mixed-input models which took both SDSS images with shape ($32 \times 32 \times 5$) used by the CNNs, as well as the five magnitudes (u, g, r, i, and z) which were used as the sole features for the RFs. To handle the magnitude features, a simple multilayer perceptron (MLP) was used which included five fully connected layers each with 1024 neurons (for full network architectures used see Figs A2 and A4).

A perfect CNN would theoretically be able to extract all useful information from the galaxy images (including the magnitudes which are obtained from performing photometry on the same images), and hence render the mixed-input models superfluous. However, as we saw from the results, the mixed-input inception CNN was able to outperform the inception CNN it was based off, where the only difference was the magnitude features also being included. This suggested that the CNN was not able to perfectly extract the magnitudes from the images, and by explicitly providing them as additional features to be handled by a MLP, we were able to boost the performance.

In other cases of ML, it can be ill-advised to use features which are highly correlated. This is due to the chance that the model will output results which vary drastically and would not generalize to other data sets (Goldstein 1993). However, due to the difference in extracting information from the images through convolutional layers, which result in more abstract features than the magnitudes, and the fact that the results we saw suggested the explicit inclusion of magnitude features helped rather than hurt the model performance, we concluded that this was not a multicollinearity issue (Garg & Tai 2013).

There are two additional reasons for avoiding using correlated features. The first reason is an increase in model complexity which results in slower models than if fewer, independent features were used. However, in this case the inclusion of a MLP to handle the additional magnitude features had a minimal effect on the overall speed of the model . The second reason is that using correlated features often makes models less interpretative. This point was also less of an issue in this case as the addition of the magnitudes as features was physically motivated and the MLP that handled these features was kept separate to the rest of the CNN.

This allows the mixed-input models to be thought of as a combination of two separate models. This process of taking a combination of different models to give the final prediction is more widely used, and when using neural networks one has the option to implement subnetworks (as we have done here). Subnetworks, such as the CNN and MLP we used, handle the different inputs separately before their

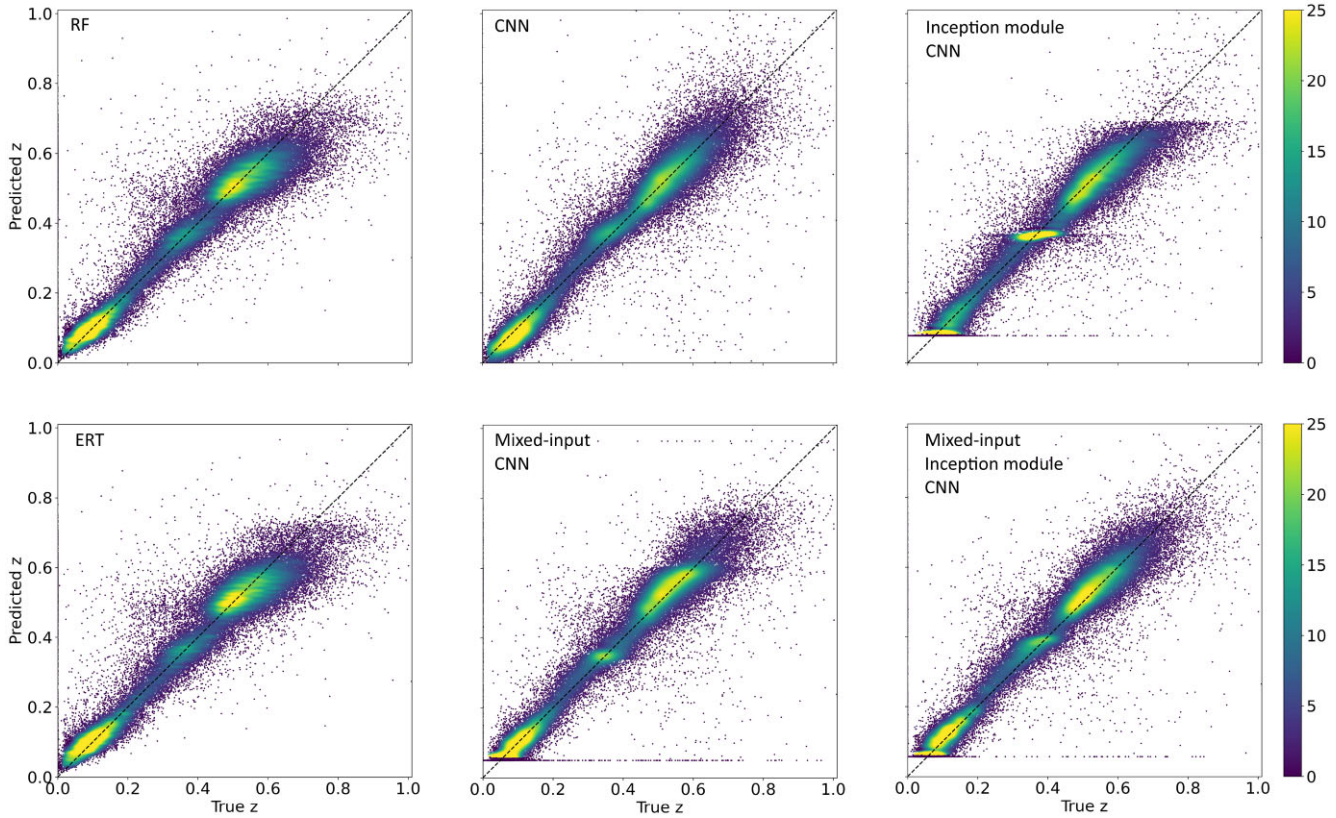

Figure 5. Density-scatter plots of the photometric redshift estimates against the true spectroscopic redshift for each of the algorithms tested. The RF and ERT algorithms both used magnitudes as their input features, whereas the CNN and Inception module CNN used image data, and the mixed-input CNNs used both images and magnitudes as inputs.

Table 1. Results of testing the different ML algorithms, where each algorithm was trained using 1000 000 galaxies, and the best performance for each metric is shown in bold. The RF and ERT both used photometric data with only the magnitudes taken as input features, whereas the CNN and inception module CNN used image data, and the mixed-input CNNs used both images and magnitudes as inputs.

	Random Forest (RF)	Extremely Randomized Trees (ERT)	Convolutional Neural Network (CNN)	Inception Module CNN	Mixed-input CNN	Mixed-input Inception CNN
MSE	0.01253	0.01261	0.01009	0.00956	0.00997	**0.00916**
MAE	0.05003	0.05067	0.04388	0.04310	0.04154	**0.03966**
R^2	0.76154	0.76002	0.80809	0.81810	0.81030	**0.82567**
Bias	0.00498	0.00538	0.00122	**−0.00094**	0.00292	0.00878
Precision	0.03076	0.03170	0.02985	0.02987	0.02764	**0.02588**
Catastrophic Outlier Fraction	0.04722	0.04866	0.03619	0.03309	**0.03048**	0.03075

outputs are concatenated into a single feature vector which can then be used to give the final output prediction (Burkov 2019).

3.4 Random forests

RFs (Breiman 2001) are built from many decision trees, where the predictions of all the individual trees are averaged to give the prediction of the overall forest. Each decision tree is a non-parametric algorithm that works by taking the input features and applying simple if-then-else statements which give decision boundaries to split the data into branches until eventually the data reache a leaf node where there are no further splits. The trees are trained recursively whereby the features splits are chosen to give the most information gain (the largest difference between the two branches).

By averaging the predictions of many individual trees, the resulting RF has a lower variance. This is also achieved by employing extra elements of randomness. First, the data used for each tree is a random subset of the overall training set (Breiman 1996). Secondly, the feature splits used are taken from a random subset of the total possible feature splits rather than always taking the split which resulted in the largest information gain. These random steps result in decision trees with higher errors, but after averaging the predictions these errors cancel out and the overall RF is a much more robust model with lower variance and less overfitting.

3.5 Extremely randomized trees

ERT (Geurts, Ernst & Wehenkel 2006) is an adaptation of the RF with an additional element of randomness. As well as using a random

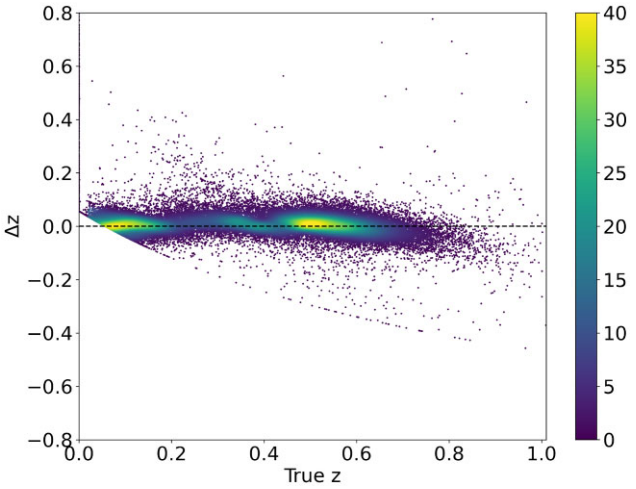

Figure 6. Plot showing the residuals of the mixed-input inception CNN (with lighter colours indicating denser regions). The residuals, Δz, were plotted against the true redshift, z, to show how the errors scaled with redshift. The plot clearly displayed the region where no redshifts would be predicted, as well as there being a slight systematic overestimation at lower spectroscopic redshifts, with underestimation at higher values.

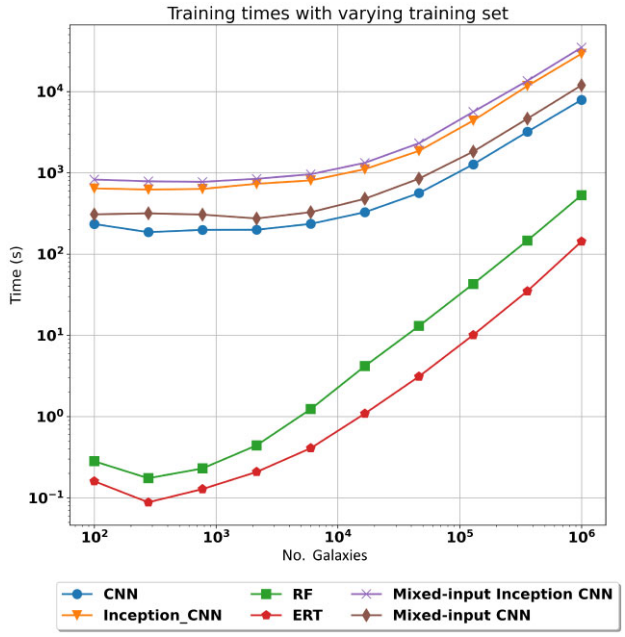

Figure 7. Plot showing how the training time changes with the number of galaxies used in the training set to display how each algorithm scaled.

subset of the possible features, it also takes a random threshold for each of the features before then using the best of the these random thresholds as the decision rule. This results in a model with lower variance at the cost of a slightly greater bias, however, perhaps the biggest improvement is to the efficiency of the model. As shown by the benchmarking performed, the ERT is a much faster algorithm than the RF.

For details of the RF and ERT implemented in this study, we provide the hyperparameters used in Appendix B.

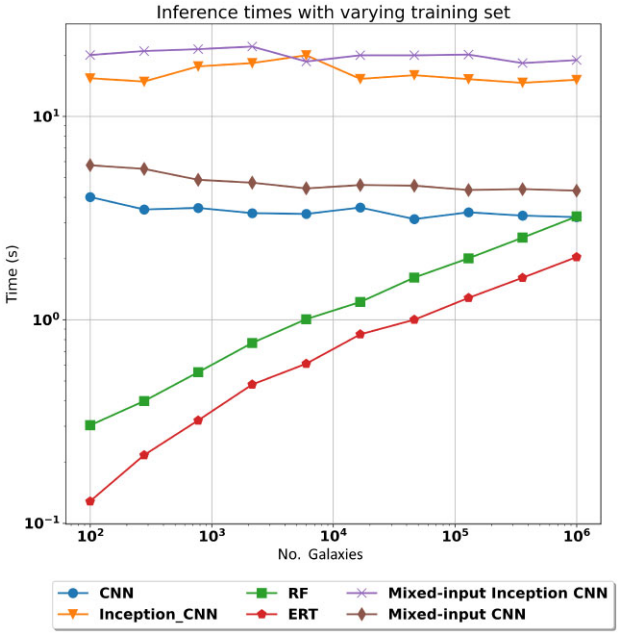

Figure 8. Plot showing how the inference time changes with the number of galaxies used in the training set to display how each algorithm scaled.

3.6 Metrics

One of the most important steps of any ML problem is defining sensible metrics that can be used to evaluate model performance. As we treated this as a regression problem, where we found a single value for the photometric redshift, the natural choice was to calculate the three most commonly used regression metrics: MSE defined as

$$\mathrm{MSE}(z, \hat{z}) = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}-1} (z_i - \hat{z}_i)^2, \tag{3}$$

mean absolute error (MAE) given by

$$\mathrm{MAE}(z, \hat{z}) = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}-1} |z_i - \hat{z}_i|, \tag{4}$$

and R squared score (R^2) defined as

$$R^2(z, \hat{z}) = 1 - \frac{\sum_{i=1}^{n}(z_i - \hat{z}_i)^2}{\sum_{i=1}^{n}(z_i - \bar{z})^2}. \tag{5}$$

The equations for each metric is given above for a data set with n galaxies where for the i^{th} galaxy, $\hat{z}_i$ is the predicted redshift, and z_i is the true spectroscopic redshift.

As well as these standard regression metrics, there are three additional metrics which are commonly used in photometric redshift estimation. These metrics are the bias, precision, and outlier fraction, which are all calculated from the residuals, Δz, defined as

$$\Delta z = \frac{z_{\mathrm{pred}} - z_{\mathrm{spec}}}{1 + z_{\mathrm{spec}}}. \tag{6}$$

The bias is simply defined as the mean of these residuals, given by

$$\mathrm{Bias} = < \Delta z > . \tag{7}$$

The precision (also 1.48 × median absolute deviation (MAD; Ilbert et al. 2006) gives the expected scatter and is defined as

$$\mathrm{Precision} = 1.48 \times \mathrm{median}(|\Delta z|). \tag{8}$$

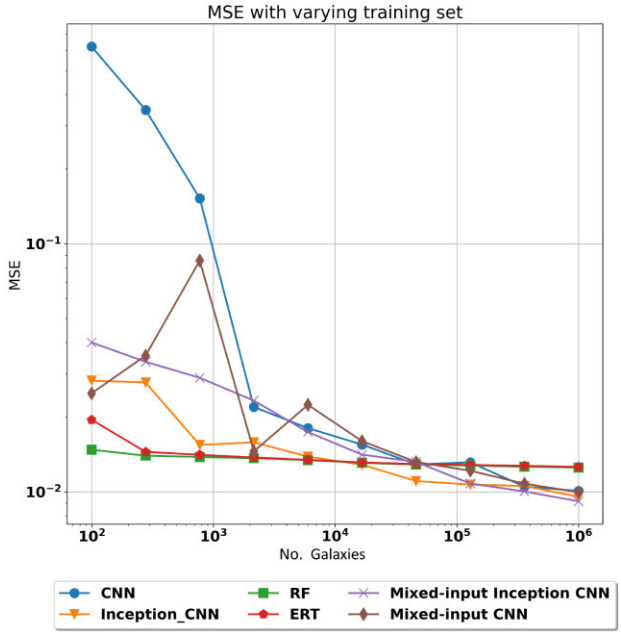

Figure 9. Graphs of the mean squared error (MSE) plotted against the number of galaxies in the training set to show how each algorithm's performance scaled.

Finally, the catastrophic outlier fraction is simply the fraction of predictions with an error greater than a set threshold, here set >0.10, which is given by

$$\text{Catastrophic Outlier Fraction} = \frac{N(|\Delta z|) > 0.10}{N_{\text{total}}}. \quad (9)$$

The results of our tests are given in Table 1 with each of the metrics quoted for the various different algorithms.

3.7 Optimization

Optimization is the process of fine tuning the hyperparameters of ML algorithms to give the best possible predictions. These hyperparameters are parameters that are set previous to the learning process and dictate how the algorithms create the mapping from input data to answer. For traditional methods, such as the RF and ERT, this optimization can be done by specifying a grid of hyperparameters and iteratively testing which combination of parameters gives the best predictions. The process of testing every combination of the specified grid (called brute-force optimization) is very slow, and instead we tested 200 random combinations within the hyperparameter grid (called random optimization) which gave a good estimate of the best possible parameters in much less time.

Neural networks, such as the CNN, inception module CNN, and mixed-input models, cannot be optimized in the same way. As well as being far more computationally expensive, which would make any brute-force optimization impossibly slow, each network has a unique architecture which changes the hyperparameters that need optimizing. In this study, we tested various architectures, and instead of defining grids of hyperparameters to test, we simply went through the hyperparameters which have the greatest impact on the models (such as the number of layers, the number of neurons in each layer, and the kernel size, stride, and padding of convolutional layers) and tested different combinations to find the best preforming models.

3.8 Benchmarking

Benchmarking is the process of running a set of standardized tests to determine the relative performance of an object, in this case iteratively running the training and testing of different ML algorithms. Here, benchmarking was performed in a similar vein to Henghes et al. (2021). We recorded the time taken throughout the ML process and varied the size of the training data set to be able to compare the efficiency of the various models. By combining these measurements with the error results for the photometric redshifts, we were able to better understand the performance of the different algorithms and give more discussions along with plots in Section 4.

Generally the hardware used to test each algorithm should be kept the same, however, in this case as we were comparing CNNs with traditional ML methods, the CNNs were trained using a GPU, whereas the RF and ERT were trained using the central processing unit (CPU). While this did change the nature of the comparison, it was still a valid test as both the GPU and CPU used were simply standard laptop components (an nvidia GTX1050Ti and intel i9-8950hk). Additionally, this highlights one of the key differences between deep learning methods, which are highly parallelizable, and traditional ML methods, which often are not. Even in the case of RFs which are also parallelizable, and indeed were run over multiple CPU cores, they do not benefit to the same extent as CNNs can when run using thousands of CUDA cores (Kirk 2007).

4 RESULTS

The results of our investigation are presented in multiple ways. Table 1 details the error metrics achieved by each of the ML algorithms when using the full 1 million galaxies in the training set. We plotted density-scatter graphs of the predicted photo-z estimates against the true spectroscopic redshift in Fig. 5, as well as plotting the scatter of the error of the best performing mixed-input inception CNN in Fig. 6, and displaying the results of the benchmarks in Figs 7–9 to show how the different algorithms scaled. Finally, the results of running the same algorithms on a smaller redshift range of $z < 0.3$ are discussed in Section 4.1.

Our results showed that the mixed-input inception module CNN was the best performing algorithm in terms of errors, with a *MSE* = 0.009. It also had the best performance in every other metric other than catastropic outlier fraction and bias, where the mixed-input (standard) CNN had a slightly better outlier fraction with both algorithms having just over 3 per cent outliers, and the inception module CNN had a lower bias due to its remarkably symmetric predictions as shown in Fig. 5.

Fig. 5 also displays a potential downfall in the majority of the CNN-based methods, with all but the standard CNN having an initial redshift cut below which they failed to predict any redshifts at all. While this was a result of both the architectures selected and the smaller number of galaxies included in the training set with a redshift $z < 0.05$, one would need to be careful to ensure that any algorithm implemented in a photo-z pipeline could predict redshifts in the full range.

While the mixed-input inception module CNN showed impressive performance, it did come at the cost of being the slowest algorithm tested, which made sense being the most complex model. As seen from Figs 7 and 8, the two inception module based CNNs were the slowest algorithms, with the mixed-input model being only slightly slower in both training and inference than the image only model. Similarly, the mixed-input CNN was only slightly slower compared with the CNN, showing that the addition of the magnitude features to

Table 2. Results of testing the different ML algorithms for the redshift range $z < 0.3$. Each algorithm was retrained using 400 000 galaxies with the RF and ERT both using the photometric magnitudes as input data, whereas the CNN and inception module CNN used images, and the mixed-input CNNs used both images and magnitudes.

	Random Forest (RF)	Extremely Randomized Trees (ERT)	Convolutional Neural Network (CNN)	Inception Module CNN	Mixed-input CNN	Mixed-input Inception CNN
MSE	0.001 40	0.001 62	0.000 72	0.000 70	0.001 69	**0.000 69**
MAE	0.024 72	0.028 55	0.018 51	0.017 73	0.027 85	**0.016 93**
R^2	0.760 04	0.721 96	0.877 32	0.880 74	0.710 23	**0.882 30**
Bias	0.002 92	0.001 28	0.007 13	0.004 47	− 0.006 70	**0.000 70**
Precision	0.022 18	0.028 03	0.018 75	0.016 91	0.026 31	**0.015 43**
Catastrophic outlier fraction	0.022 45	0.023 93	**0.005 57**	0.006 59	0.023 38	0.008 16

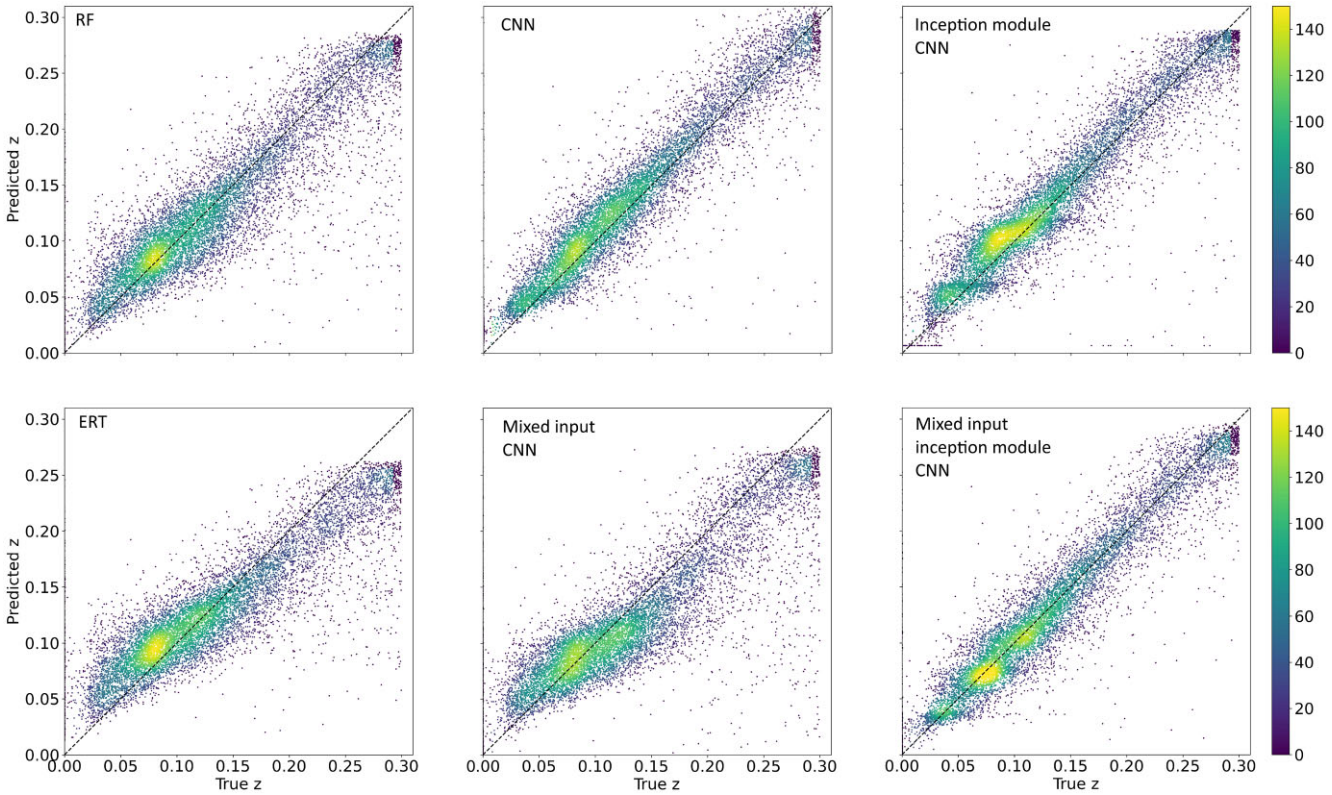

Figure 10. Graphs of photometric redshift estimates against the true spectroscopic redshift for ML algorithms retrained on galaxies within the range $z < 0.3$.

the image-based CNNs means only a small increase in computational requirements.

We also saw that the traditional ML methods, RF and ERT, were significantly faster but also worse in terms of their error performance. While these methods could still be very useful in the absence of image data, the improvements seen by making use of images made the CNNs an exciting alternative. Furthermore, by directly using the image data rather than the magnitude features, one could offset the increased time required to train the algorithms with the time saved due to not needing to previously extract features.

What's more, the RF and ERT also showed the worst scaling of all the algorithms, slowing down at a faster rate as the number of galaxies included in the training set was increased. This was due to the fact that the tree-based algorithms are non-parametric and the complexity increases with the increasing data set, whereas the neural networks have a fixed size. Past 1 million galaxies in the training set the RF was already slower in inference than the CNN, and with large enough data sets it is possible that they could become almost as slow during training. If for the largest data sets CNNs become faster than traditional methods, then their main setback of being slower and more computationally expensive would no longer be of concern.

The performance of the RF and ERT highlighted the improvements possible when including image data rather than using magnitudes alone, with a reduction in errors of around 25 per cent. The experimental mixed-input models also showed good potential to further improve performance; however, it was clear that the CNN network architecture had a greater impact than the addition of the magnitudes as extra features. The improvement from inception module CNN to the mixed-input inception CNN was much less than the improvement from RF or ERT to the CNNs (the improvement from including images), with a further error reduction at just over 4 per cent.

4.1 Lower redshift range

Although the algorithms performed well over the entire data set, we wanted to also test the performance for a smaller region to be able to more directly compare with other studies (such as Pasquet et al. (2019)) and see how much better the performance could be when the problem of estimating redshifts was made easier by only considering the range $z < 0.3$.

The exact same process was carried out using the same six algorithms and the results from the retrained ML algorithms are given in Table 2. We also plotted the redshift estimations against the true spectroscopic redshift in Fig. 10.

From these we saw that in general the algorithms performed much better, reaffirming the fact that photometric redshift estimation becomes an easier problem over a shorter range. The mixed-input inception CNN continued to be the best performing algorithm with an $MSE = 0.0007$, however, there was far less separating the CNN and inception module CNN in the smaller redshift range. In fact, the CNN performed better than every other algorithm when it came to the catastrophic outlier fraction with only 0.56 per cent outliers and one of the best constrained scatter plots (second only to the mixed-input inception module CNN).

Fig. 10 shows how well constrained the redshift estimates of the mixed-input inception module CNN were, with a denser region, along the $z_{\rm pred} = z_{\rm spec}$ line. Furthermore, the algorithm no longer exhibited its previous issue of not predicting redshifts across the full redshift range. Indeed the algorithms which had a redshift cut in the smaller redshift range were the RF, ERT, and mixed input CNN, however, in this case they failed to predict redshifts above a certain value.

There was also a greater disparity between the CNN-based methods and traditional methods for this smaller redshift range. While there was around a 30 per cent improvement going from the RF and ERT to CNNs over the entire data set, this increased to 50 per cent for the smaller redshift range. The image-based CNNs were therefore able to provide even more advantage in this range, suggesting that the additional information extracted from the images is even more beneficial in the smaller range, possibly due to the fact that the galaxies would generally occupy a larger region of the image.

However, this boost in performance for the redshift range of $z < 0.3$ also highlighted a key failing of the algorithms, in that an ideal model would generalize well enough to perform just as well across the entire redshift range. This might not be realistic as by removing a large section of the data there was far less chance of having catastrophic outliers, and the overall problem was made easier.

5 CONCLUSIONS

Processing accurate photometric redshift estimations will remain a vital task of cosmological analyses. Future surveys, such as *Euclid* and *LSST*, aim to observe more galaxies than ever before, and with such strict error requirements, it is of upmost importance that the methods developed and implemented are both effective and efficient.

Here, we have shown how image-based CNN methods compare to traditional tree-based methods that make use of magnitude features from photometry. We found that the additional information the CNNs were able to extract directly from the images of galaxies allowed for a significant reduction in errors. However, as the CNNs were more complex than the RF and ERT algorithms, they were also much slower to run and required far more computational resources.

Our results showed that the experimental mixed-input models in particular had great potential for photo-z estimation. Using 1 million images of galaxies to train the algorithm the mixed-input inception CNN was able to achieve a $MSE = 0.009$. Furthermore, when the problem was simplified to only include galaxies in the range $z < 0.3$, the model achieved an even more impressive $MSE = 0.0007$, outperforming the traditional RF by > 50 per cent.

Further work would include using even more data with tens or hundreds of millions of galaxies and images (which would require the use of large-scale simulations and more powerful computer architectures). The use of more powerful CPUs and GPUs in high perfomance computing systems could allow for better practices in benchmarking and set a standard system. Additionally, by stretching the amount of data further, we could then determine with certainty at what point the CNNs would become faster than the RF and ERT, as well as discover whether increasing the amount of data used in the training set would eventually have no effect on model performance. Finally, the models tested here could also be extended to produce PDFs for the estimated redshifts, and through further optimization (including using custom loss functions) the errors could be reduced even more.

ACKNOWLEDGEMENTS

BH was supported by the STFC UCL Centre for Doctoral Training in Data Intensive Science (grant no. ST/P006736/1). OL thanks All Souls College, Oxford for a Visiting Fellowship. Authors also acknowledge the support from following grants: O.L.'s European Research Council Advanced Grant (TESTDE FP7/291329), STFC Consolidated Grants (ST/M001334/1 and ST/R000476/1), J.T.'s UKRI Strategic Priorities Fund (EP/T001569/1), particularly the AI for Science theme in that grant and the Alan Turing Institute, Benchmarking for AI for Science at Exascale (BASE), EPSRC ExCALIBUR Phase I Grant (EP/V001310/1).

Funding for SDSS-III has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, and the U.S. Department of Energy Office of Science. The SDSS-III website is http://www.sdss3.org/.

SDSS-III is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS-III Collaboration, including the University of Arizona, the Brazilian Participation Group, Brookhaven National Laboratory, Carnegie Mellon University, University of Florida, the French Participation Group, the German Participation Group, Harvard University, the Instituto de Astrofisica de Canarias, the Michigan State/Notre Dame/JINA Participation Group, Johns Hopkins University, Lawrence Berkeley National Laboratory, Max Planck Institute for Astrophysics, Max Planck Institute for Extraterrestrial Physics, New Mexico State University, New York University, Ohio State University, Pennsylvania State University, University of Portsmouth, Princeton University, the Spanish Participation Group, University of Tokyo, University of Utah, Vanderbilt University, University of Virginia, University of Washington, and Yale University.

DATA AVAILABILITY

The data used in this paper came entirely from the SDSS-DR12, and are openly available from: https://www.sdss.org/dr12/.

REFERENCES

Abadi M. et al., 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available at: https://www.tensorflow.org/

Abdalla F. B., Banerji M., Lahav O., Rashkov V., 2011, MNRAS, 417, 1891

Aihara H. et al., 2018, PASJ, 70, S4

Alam S. et al., 2015, ApJS, 219, 12
Amendola L. et al., 2018, Living Rev. Relativ., 21, 2
Beck R., Dobos L., Budavári T., Szalay A. S., Csabai I., 2016, MNRAS, 460, 1371
Benitez N., 2000, ApJ, 536, 571
Bettaney E. M., Hardwick S. R., Zisimopoulos O., Chamberlain B. P., 2019, preprint (arXiv:1904.00741)
Bojarski M. et al., 2016, preprint (arXiv:1604.07316)
Bolzonella M., Miralles J.-M., Pelló R., 2000, A&A, 363, 476
Breiman L., 1996, Mach. Learn., 24, 123
Breiman L., 2001, Mach. Learn., 45, 5
Burkov A., 2019, The Hundred-Page Machine Learning Book. Quebec City, QC, Canada, Andriy Burkov
Cavuoti S., Brescia M., Amaro V., Vellucci C., Longo G., Tortora C., 2016, in IEEE Symposium Series on Computational Intelligence (SSCI), p. 1
Chollet F., et al., 2015, Keras. Available at: https://keras.io
Collister A. A., Lahav O., 2004, PASP, 116, 345
D'Isanto A., Polsterer K. L., 2018, A&A, 609, A111
DES Collaboration, 2016, MNRAS, 460, 1270
DES Collaboration, 2022, Phys. Rev. D, 105, 023520
Dewdney P. E., Hall P. J., Schilizzi R. T., Lazio T. J. L., 2009, Proc. IEEE, 97, 1482
De Jong J. T., Kleijn G. A. V., Kuijken K. H., Valentijn E. A., 2013, Exp. Astron., 35, 25
Flaugher B., Bebek C., 2014, in Ground-Based and Airborne Instrumentation for Astronomy V. SPIE, Bellingham, p. 91470S
Fukushima K., Miyake S., 1982, in Competition and Cooperation in Neural Nets. Kyoto, Japan, Springer, p. 267
Garg A., Tai K., 2013, Int. J. Mod. Identif. Control, 18, 295
Geurts P., Ernst D., Wehenkel L., 2006, Mach. Learn., 63, 3
Goldstein R., 1993, Conditioning Diagnostics: Collinearity and Weak Data in Regression. Taylor & Francis, NY, USA
Harris C. R. et al., 2020, Nature, 585, 357
He K., Zhang X., Ren S., Sun J., 2015, in Proceedings of the IEEE International Conference on Computer Vision. p. 1026
Henghes B., Pettitt C., Thiyagalingam J., Hey T., Lahav O., 2021, MNRAS, 505, 4847
Heymans C. et al., 2021, A&A, 646, A140
Hildebrandt H. et al., 2021, A&A, 647, A124
Hoyle B., 2016, Astron. Comput., 16, 34
Hubel D. H., Wiesel T. N., 1968, J. Physiol., 195, 215
Ilbert O. et al., 2006, A&A, 457, 841
Ivezić Ž. et al., 2019, ApJ, 873, 111
Kirk D., 2007, in Proceedings of the 6th International Symposium on Memory Management, ISMM '07, Association for Computing Machinery. p. 103
LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Mandelbaum R. et al., 2018, preprint (arXiv:1809.01669)
Martini P. et al., 2018, in Ground-Based and Airborne Instrumentation for Astronomy VII. SPIE, Bellingham, p. 410
McCulloch W. S., Pitts W., 1943, Bul. Math. Biophys., 5, 115
Myles J. et al., 2021, MNRAS, 505, 4249
Nair V., Hinton G. E., 2010, in International Conference on Machine Learning, Haifa, p. 807
Pasquet J., Bertin E., Treyer M., Arnouts S., Fouchez D., 2019, A&A, 621, A26
Schuldt S., Suyu S., Cañameras R., Taubenberger S., Meinhard T., Leal-Taixé L., Hsieh B., 2021, A&A, 651, A55
Soo J. Y. H. et al., 2017, MNRAS, 475, 3613
Spergel D. et al., 2015, preprint (arXiv:1503.03757)
Szegedy C. et al., 2015, in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). p. 1
Tyson J., Wittman D., Hennawi J., Spergelb D., 2003, Nucl. Phys. B, 124, 21
York D. G. et al., 2000, AJ, 120, 1579

APPENDIX A: NETWORK ARCHITECTURES

In the following pages, we present plots of the full-network architectures used in the four CNN-based methods tested in this paper.

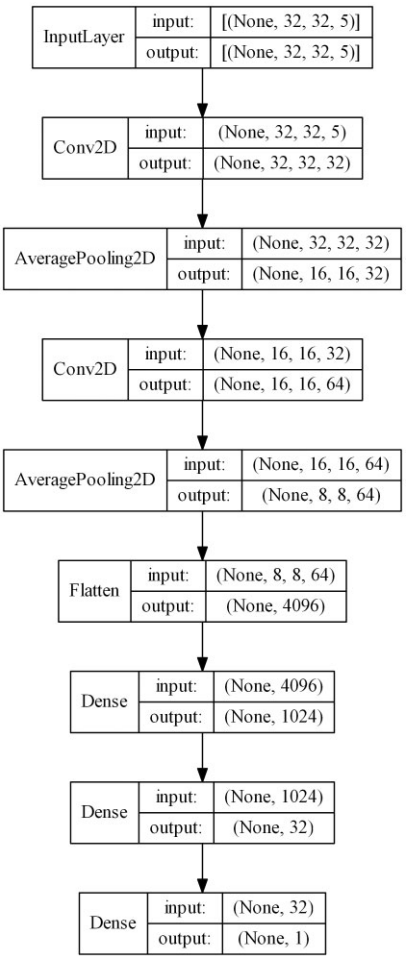

Figure A1. Network architecture of the base CNN tested. The CNN was constructed with two convolutional layers, each followed by an average pooling layer to reduce the dimensionality, before the feature map was flattened to give a 1D feature vector. This could then be handed to the two dense layers (which are the fully connected layers) that process the features before the final, single neuron layer is used to give the value of the predicted redshift.

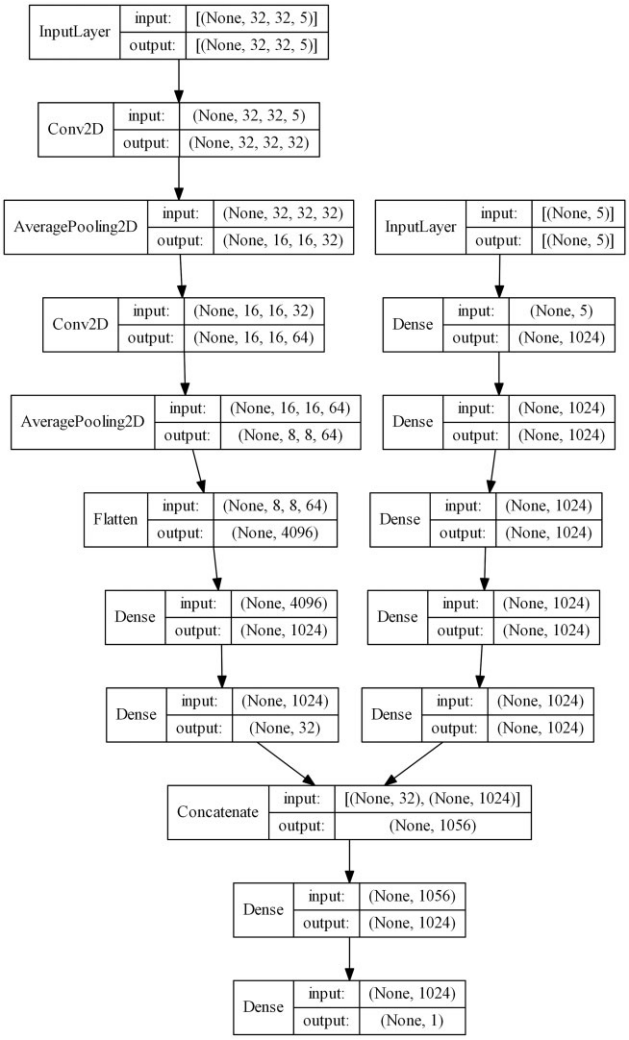

Figure A2. Network architecture of the mixed-input CNN. This model used the same CNN as A1 to handle the images, and added a MLP with five fully connected layers each of 1024 neurons to handle the magnitude data. The outputs of both were then concatenated before being handed to a fully connected layer and finally the single neuron layer which gave the value of the predicted redshift.

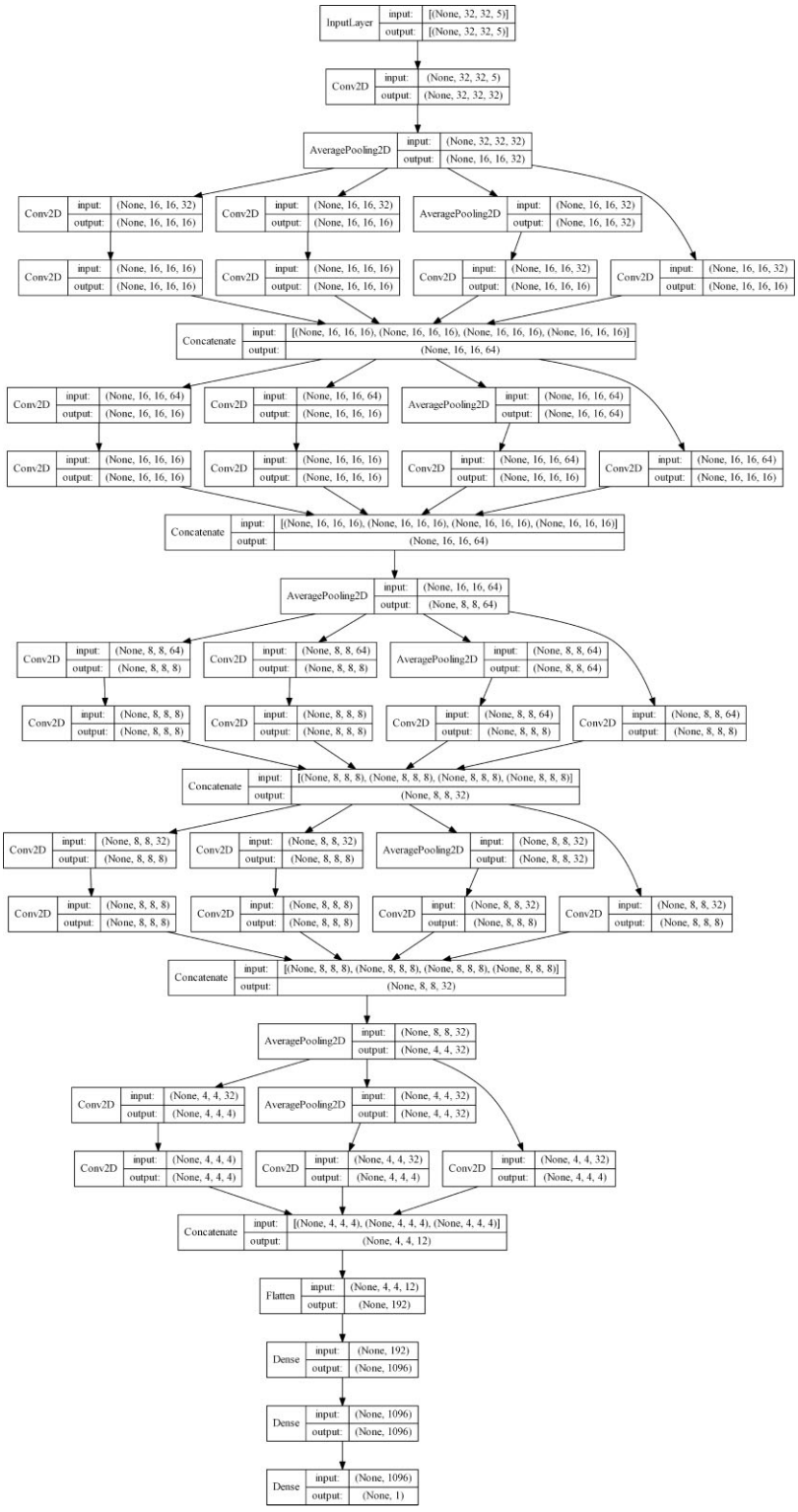

Figure A3. Network architecture of the inception module CNN. This model used a single convolutional layer and average pooling layer before applying five inception modules, where the fifth inception module was a modified version to be smaller and not include a (5×5) kernel. Following the inception modules, the output was flattened to give the feature vector which was processed by two fully connected layers with 1096 neurons, and finally the single neuron layer to give the predicted redshift.

Figure A4. Network architecture of the mixed-input inception CNN. This model used the same inception module CNN as A3 to handle the images, and added the same five layer MLP which was used in A2 to handle the magnitude features. The outputs of both were concatenated and handed to a single fully connected layer before the final single neuron layer gave the predicted redshift.

APPENDIX B: HYPERPARAMETERS

Here, we present a table detailing the hyperpameters used by the RF and ERT methods tested in this investigation.

Table B1. Grids of hyperparameters that were used in the RF and ERT, selected by the random optimization.

Classifier	Hyperparameter	Selected value
RF	'no. estimators'	**200**
	'max. features'	**2**
	'min. samples leaf'	**7**
	'min. samples split'	**3**
	'min weight fraction leaf'	**0**
	'criterion'	**mse**
ERT	'no. estimators'	**147**
	'max. features'	**4**
	'min. samples leaf'	**3**
	'min. samples split'	**87**
	'min weight fraction leaf'	**0**
	'criterion'	**mse**

This paper has been typeset from a TEX/LATEX file prepared by the author.