Full length article

Galaxy morphology classification using neural ordinary differential equations

R. Gupta [a], P.K. Srijith [a], S. Desai [b],*

[a] *Department of Computer Science and Engineering, IIT Hyderabad, Kandi, Telangana 502285, India*
[b] *Department of Physics, IIT Hyderabad, Kandi, Telangana 502285, India*

A R T I C L E   I N F O

A B S T R A C T

We introduce a continuous depth version of the Residual Network (ResNet) called Neural ordinary differential equations (NODE) for the purpose of galaxy morphology classification. We carry out a classification of galaxy images from the Galaxy Zoo 2 dataset, consisting of five distinct classes, and obtained an accuracy between 91%–95%, depending on the image class. We train NODE with different numerical techniques such as adjoint and Adaptive Checkpoint Adjoint (ACA) and compare them against ResNet. While ResNet has certain drawbacks, such as time consuming architecture selection (e.g. the number of layers) and the requirement of a large dataset needed for training, NODE can overcome these limitations. Through our results, we show that the accuracy of NODE is comparable to ResNet, and the number of parameters used is about one-third as compared to ResNet, thus leading to a smaller memory footprint, which would benefit next generation surveys.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

The problem of determining the morphology of a galaxy plays a pivotal role in a large number of fields from galaxy evolution to cosmology. Some of these applications include stellar masses (Bundy et al., 2005), star formation history (Kennicutt, 1998), colour (Skibba et al., 2009), gas and dust content (Lianou et al., 2019), age of the galaxy (Bernardi et al., 2010), various dynamical processes (Romanowsky and Fall, 2012), tests of modified gravity theories (Desmond and Ferreira, 2020) etc. A recent review on various aspects of galaxy morphology and its connections to the rest of astrophysics can be found in Buta (2013).

The very first morphological classification schemes pioneered by Hubble (1926) were based upon visual scanning of galaxies and classifying them into different types such as spirals, ellipticals, lenticulars. With the advent of large area optical surveys, the task of visual classification was outsourced to the Galaxy Zoo project (Lintott et al., 2008). The first incarnation of the project (Galaxy Zoo 1), consisting of a dataset of more than 900,000 images by the Sloan Digital Sky Survey (York et al., 2000), was classified by citizen scientists into four categories: "spiral", "elliptical", "a merger" or "star/don't know" (Lintott et al., 2008). The project enabled the annotation of a million galaxy images within several months. This was superseded by Galaxy Zoo 2 (Willett

et al., 2013), Galaxy Zoo: Hubble (Willett et al., 2017), and Galaxy Zoo: CANDELS (Simmons et al., 2017).

Unfortunately, this manual approach of visual classification does not scale well with the unprecedented pace of data growth due to the large number of metre-class telescopes equipped with multi-CCD imagers, which have been continuously built over the past two decades. Very soon stage IV Dark Energy surveys such as Legacy Survey of Space and Time operated by the Vera Rubin observatory (Abell et al., 2009), Euclid (Laureijs et al., 2011), and Roman Space Telescope (Spergel et al., 2013) are going to produce petabytes worth of data, rendering manual classification impossible.

Therefore, astronomers have turned their attention to automated classification methods. Over the past few decades, a large amount of literature has emerged on such automated methods for measuring galaxy morphology, especially in large observational surveys. These methods range from parametric techniques, which attempt to describe the galaxy light profiles using small sets of parameters (Simard et al., 2002; Sersic, 1963; Odewahn et al., 2001; Lackner and Gunn, 2012), to non-parametric methods that reduce these light distributions to single values such as in the 'CAS' system (Conselice, 2003; Abraham and van den Bergh, 2001; Menanteau et al., 2005), the Gini-M20 coefficients (Lotz et al., 2004; Freeman et al., 2013), etc. Recent reviews of some of these automated methods can be found in de Diego et al. (2020) and Martin et al. (2020).

A major game changer throughout astronomy and astrophysics has been the widespread application of machine learning and

---

* Corresponding author.
E-mail addresses: cs19mtech11024@iith.ac.in (R. Gupta), srijith@iith.ac.in (P.K. Srijith), shantanud@phy.iith.ac.in (S. Desai).

deep learning techniques (Ball and Brunner, 2010; Kremer et al., 2017; Bethapudi and Desai, 2018; Baron, 2019), and galaxy morphology is no exception to this. Applications of machine learning as well as deep learning to galaxy morphology classifications are discussed in Dieleman et al. (2015), Tanoglidis et al. (2020), Tuccillo et al. (2017), Barchi et al. (2020), Khan et al. (2019), Spindler et al. (2020), Bhambra et al. (2021) and Reza (2021).

Deep learning models, known as deep neural networks (DNN), have been widely used for image classification and slowly began to beat human accuracy in these tasks, as soon as large training sets started becoming available (LeCun et al., 2015). DNN models, especially Convolution Neural Networks (CNN) (Krizhevsky et al., 2012), AlexNet, VGGNet and GoogleNet, took the accuracy of DNNs to new heights. With the advent of Residual Networks (ResNet) (He et al., 2015), researchers were able to make these CNN models deeper than ever before, without suffering from additional problems. Among the machine learning techniques, CNNs (Krizhevsky et al., 2012) have become the mainstream method for image classification. However, CNN with a large number of layers suffer from the vanishing gradient problem (Kolen and Kremer, 2001).

In the popular deep learning models such as ResNets, the selection of architecture (depth of the network) and the presence of a large number of parameters can make the training process computationally intractable. Recently, a continuous depth counterpart to ResNets, known as NODE (Chen et al., 2018) was introduced, which could overcome these drawbacks. In our work, we propose to use NODE for the galaxy morphology classification problem. We compare its performance against ResNet, which has also been used in other works (Zhu et al., 2019; Goddard and Shamir, 2020), as that is the state-of-the-art deep learning approach for galaxy morphology classification and demonstrate the benefits of NODE over ResNets.

NODE is inspired by the way ResNet works, where one models the change in the feature maps over layers using a neural network. This can be seen as equivalent to an ordinary differential equation with the derivative modelled as a neural network function. Consequently, the final layer feature map can be obtained using numerical solvers for ODE such as Euler's method and Runge–Kutta method. NODE has certain advantages over ResNet. In NODE, the network depth is implicitly determined by the tolerance parameter of the numerical solver used, rather than being explicitly fixed like in ResNet. Thus, by tuning the tolerance parameter, we can trade-off between the model speed and model accuracy. Another advantage of NODE over ResNet is that the number of parameters in NODE is much less than ResNet. Models with smaller number of parameters require less data to train and do not suffer from over-fitting issues. With new training techniques emerging in this field, like Adaptive Checkpoint Adjoint (Zhuang et al., 2020), NODE architecture is becoming more accurate and faster with time.

The NODE architecture has been applied to a wide variety of fields, such as biomedical imaging, high-energy physics, image and video processing, 3D modelling, economics, etc. (Groha et al., 2020). For example, in the case of biomedical-imaging, it has been used for kidney segmentation (Valle et al., 2019), reconstruction of MRI images (Chen et al., 2020), multi-state survival analysis (Groha et al., 2020), 3-D modelling for accurate manifold generation (Gupta and Chandraker, 2020), small-footprint keyword spotting (KWS) in audio files (Fuketa and Morita, 2020), etc. In the domain of theoretical High-Energy Physics, it has been applied to holographic QCD (Hashimoto et al., 2020). However, to the best of our knowledge, this technique has not been previously applied to any problem in astrophysics.

The organization of this manuscript is as follows. In Section 2, we describe the dataset used to carry out our experiments. Next,

we shed some light on ResNet in Section 3, followed by an in-depth explanation of the working of NODE (Section 4) and its training with the adjoint method. We describe the various pre-processing steps applied to the data, followed by the exact network architecture used in Section 5. Then, in Section 6, we discuss our experimental results. Finally, we conclude in Section 7.

2. Dataset

The dataset used in our experiments is drawn from the Galaxy Zoo Challenge, available on kaggle. Classification labels for the kaggle Dataset (KD) are drawn from Galaxy Zoo 2, and the images used were obtained from SDSS-DR7 (Abazajian et al., 2009). Galaxy images used in this dataset are classified into a total of five classes viz. spiral, edge-on, cigar-shaped smooth, in-between smooth, and completely round smooth. The different morphological types are shown in Fig. 1. Similar to Zhu et al. (2019), we shall use the numerical labels 0, 1, 2, 3, 4 to annotate completely round, in-between, cigar-shaped, edge-on, and spiral galaxies, respectively.

KD consists of around 60,000 images, and each image is divided into five classes, with a classification probability provided for each class. We prune this dataset further and only select those images, which are classified with high probability in their respective classes. After pruning, we are left with a total of 28,790 images, with a single class assigned to each image. We should however point out that there is no absolute ground truth but rather only the truth as estimated via crowdsourcing.

This selection criteria is similar to that described in Willett et al. (2013), in which the galaxy images classified with probabilities higher than a certain threshold (discussed therein), are selected. After these cuts, we have 7806, 3903, 578, 8069, and 8434 images in each class, in the order listed at the beginning of this section.

The size of each image is $424 \times 424 \times 3$ pixels, where the last dimension denotes the number of colour channels viz. RGB. The galaxy of interest is generally located at the centre of the image. We finally split our data randomly in the ratio of 9:1 for the purpose of training and testing, thus assigning 25911 and 2879 galaxies, respectively for each task similar to Zhu et al. (2019). We create multiple random train and test splits and obtain the average and variance across them in order to conduct a more robust evaluation and to obtain error estimates on our machine learning metrics.

3. Residual neural networks

Neural networks are modelled as a series of transformations having discrete number of layers, with each one taking in a previous hidden representation $\mathbf{h}_l$ and producing a new hidden representation $\mathbf{h}_{l+1} = F^l(\mathbf{h}_l)$. We typically consider the transformation as $F(\mathbf{x}) = \sigma(\sum_i W_i x_i)$, where σ is an activation function (e.g. RELU or a sigmoid), and θ is a collection of weight vectors. Recently, many deep learning models were introduced based on the idea that increasing the number of layers can improve the performance. However, this may lead to problems such as vanishing gradients (Kolen and Kremer, 2001), where the initial layer weight vectors cannot be computed correctly through back propagation as the error gradient becomes small.

This problem was addressed by Deep Residual Learning (He et al., 2015). ResNets are a class of DNNs, which try to map the residuals instead of the complete transformation itself in the hidden layer mappings. The idea is to learn a mapping as the difference between the layers (or equivalently adding skip connections): $\mathbf{h}_{l+1} = F^l(\mathbf{h}_l) + \mathbf{h}_l$. In He et al. (2015), they showed

Fig. 1. The five different galaxy morphologies in the Galaxy Zoo-2 dataset. These classes are completely round smooth, in-between smooth, cigar-shaped smooth, edge-on and spiral, from top to bottom. See also Fig. 1 of Zhu et al. (2019) for more examples of different galaxy morphologies from this dataset.

that this simple transformation avoids the vanishing gradient problem due to skip connections and the networks can learn the weights properly. This allowed the development of deep learning models with a large number of layers (e.g. ResNet with 50 and 100 layers).

ResNet and its variants were able to achieve state of the art results for image classification. ResNet won the ILSVRC challenge in 2015. Many other variants of ResNet, achieved state-of-the-art (SOTA) results in other image datasets. ResNet mainly has two types of residual blocks. In the standard block, two 3×3 convolutions are applied, along with a skip connection. In another block, known as the bottleneck block, 1×1 convolutions are applied before and after the 3×3 convolutions, in order to reduce feature space, so that the computational complexity is reduced.

4. Neural ordinary differential equations

4.1. Residual networks

Recently, Chen et al. (2018) and Lu et al. (2017) have shown that continuous depth ResNets, known as NODE, can be developed by relating them to ordinary differential equations. Assuming the mapping function to be the same across all the layers, and letting $\Delta t \in \mathbb{R}$, we can rewrite the hidden representation update of ResNets as state updates at some time t.

$$\mathbf{h}(t+1) = F(\mathbf{h}(t)) + \mathbf{h}(t) = \frac{\Delta t}{\Delta t} F(\mathbf{h}(t)) + \mathbf{h}(t) = \Delta t G(\mathbf{h}(t)) + \mathbf{h}(t) \quad (1)$$

where $G(\mathbf{h}(t)) = F(\mathbf{h}(t))/\Delta t$. This reformulation is the same as the single step of Euler's method for solving ordinary differential equations of the form as observed in Lu et al. (2017).

$$\frac{d\mathbf{h}(t)}{dt} = G(\mathbf{h}(t), t, \theta) \quad (2)$$

As compared to standard differential equations, the derivative is represented by a function parameterized using a neural network G acting on the state $\mathbf{h}(t)$. Here, we have assumed the

G's to depend on t as well as some parameters θ (parameters of the neural network). One can consider G to represent convolution operation when applied to the image data. Considering Eq. (1), the final representation (feature map) of our network is the state $\mathbf{h}(T)$ at time T. This is then fed to a fully-connected neural network (FCNN) to predict the final output, which is a real number for regression problems and a discrete value for classification. For a neural network function G, we can use any off-the-shelf ODE solvers such as Euler and Runge–Kutta (RK4) method to solve and obtain the final representation in an iterative manner.

$$\mathbf{h}(T) = \text{ODESolve}(\mathbf{h}(t_0), G, t_0, T, \theta)$$

4.2. Training process

Training a NODE involves learning the parameters of the neural network function using an appropriate loss function (cross entropy in the case of classification). The representation learned using an ODE solver is fed to the loss function which is optimized with respect to the parameters θ

$$\underset{\theta}{\text{argmin}}\, L(\mathbf{h}(T)) = \underset{\theta}{\text{argmin}}\, L(\text{ODESolve}(\mathbf{h}(t_0), G, t_0, t_1, \theta))$$

The learning of the parameters requires back-propagating through the solver by computing the gradients with respect to the loss, and this step is computationally costly using naive back-propagation. Chen et al. (2018) proposed an adjoint sensitivity method to learn the parameters by running another ODE solver backward in time.

To optimize L and the parameters θ, we need to evaluate the gradients with respect to $\mathbf{h}(t)$ (the state of our system at any time t), and θ the neural network parameters. The adjoint method describes a way to efficiently compute the derivative of the loss with respect to the state. In brief, we define the adjoint state as

$$a(t) = -\partial L / \partial \mathbf{h}(t),$$

which describes the gradient of the loss with respect to some state $\mathbf{h}(t)$. It turns out that the dynamics of the adjoint state can be described using another ODE.

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial G(\mathbf{h}(t), t, \theta)}{\partial \mathbf{h}} \quad (3)$$

The gradient of the loss at the initial state $a(t_0)$ can be computed by running Eq. (3) in the backward direction with initial value as $a(T)$. We can compute the derivative of G with respect to $\mathbf{h}$ easily by computing the gradient through back-propagation in traditional neural networks. Now, the gradient of the loss with respect to parameters $dL/d\theta$ can be computed as

$$\frac{dL}{d\theta} = -\int_{t_0}^{T} a(t)^\top \frac{\partial G(\mathbf{h}(t), t, \theta)}{\partial \theta} dt \quad (4)$$

The approach known as adjoint sensitivity has better memory cost, linear scalability and low numerical instabilities (refer (Chen et al., 2018) for more details).

4.3. NODE Adaptive Checkpoint Adjoint (ACA)

The "standard" NODE technique uses the adjoint method for learning the parameters by efficient back-propagation through the different numerical ODE solvers like Euler, Runge–Kutta, etc. But numerical errors prevail in the computation of the gradient using the adjoint method (Zhuang et al., 2020), sometimes giving lower accuracies than expected. To mitigate this, Adaptive Checkpoint Adjoint (ACA) technique has been introduced, which estimates more robust gradients for NODE. We will refer to the NODE trained with ACA technique as NODE_ACA in our paper. NODE_ACA helps to achieve better accuracy by more accurate

---

Algorithm 1: Numerical Integration algorithm with adaptive step-size used in the forward pass of Adjoint and adaptive checkpoint adjoint approaches.

Input : input data h_0, final time T, first stepsize s_0, error tolerance *etol*
Initialize: h = h_0, s = s_0, error estimate $\hat{e} = \infty$, t = 0
while $t < T$ **do**
    while $\hat{e} > etol$ **do**
        s $\leftarrow$ s $\times$ decay_factor($\hat{e}$)
        $\hat{e}$, $\hat{h}$ = $\psi_s(t, h)$ // $\psi_s(t, h)$ compute the numerical solution at time $t + s$
    end
    t $\leftarrow t + s$, h $\leftarrow \hat{h}$
end

Algorithm 2: Forward and Backward passes of the adaptive checkpoint adjoint (ACA) algorithm. Forward pass uses the adaptive step size (Algorithm 1) for numerical integration and state computation. The state values computed in the forward pass is reused in the backward pass.

Input : initial hidden state h_0, final time T, first stepsize s_0, error tolerance *etol*
Initialize : h = h_0, s = s_0, error estimate $\hat{e} = \infty$, t = 0
ForwardPass :
    1. Perform numerical integration based on Algorithm 1.
    2. Store discretization points $t_0, ...t_{N_t}$ and state values $h_0, h_1, ...h_{N_t}$.
    3. Search for optimal stepsize by deleting local computation graphs

BackwardPass: Initialize $a(T)$, $dL/d\theta$ = 0
for N_t to 1 **do**

    1. Compute $h_{i+1} = \psi_{s_i}(t_i, h_i)$ with stepsize $s_i = t_{i+1} - t_i$
    2. Update $\lambda(t)$ and $dL/d\theta$ based on (3) and (4).
    3. Delete local computation graphs

end

---

gradient calculation and lower computation time by removing the redundancy from the computation graph.

NODE_ACA (Zhuang et al., 2020) saves forward pass and then applies this to backward pass, rather than backward trajectory being calculated independently of the forward pass as in the adjoint case. The adjoint method does not maintain a history of the $h(t)$ computed in the forward pass but remembers the boundary conditions: $h(T)$ and $a(T)$. It then tries to solve $h(t)$ and $a(t)$ backwards in time, i.e. from T to 0 in order to compute the gradient of the loss function (4). However, due to numerical errors accumulated in the forward pass, $h(t)$ computed in the backward pass may not be accurate leading to the inaccurate computation of the gradients as well as the final solution. On the other hand, in NODE_ACA, discretization points t_i and latent states $h_i = h(t_i)$ are recorded in the forward pass and reused in the backward pass to reduce inaccuracies in the gradient computation. This trajectory checkpoint strategy not only reduces numerical errors but also deletes shallow computation graphs. Both the constant and the adaptive stepsize solvers are supported by NODE_ACA. Algorithm 1 provides the details of the adaptive step-size based numerical technique used in the forward pass. Algorithm 1 summarizes the steps in the forward and backward passes of the NODE_ACA approach. NODE_ACA stores the state values computed in the forward pass and uses them in the backward pass to make the gradient computation more accurate.

5. Experimental setup

5.1. Network architecture

The network architecture used for the standard NODE training is as follows. We use a standard convolution block, consisting of two CLN (Convolution, Non-Linearity, Normalization) layers. Each convolution is done with a kernel of size 3×3.

We also downsample the input, before passing it to the ODE network. Downsampling consists of applying 2-D convolutions, while reducing the number of channels. Once the input is downsampled, it passes from the above ODE network, followed by a pooling layer. ODE maps the inputs to some desired latent space, which has the same number of dimensions as input. Similar to a classification task, as our final output has 5 dimensions (equal to the number of classes), we use a fully connected layer at the end. This FC (fully-connected) layer learns a linear mapping from the ODE output to the final output.

For NODE_ACA, we use the same architecture as that for standard NODE. Only difference being that, instead of using the adjoint method for back-propagation, the ACA technique is used.

For ResNet, we use two NLC (Normalization, Non-Linearity, Convolution) layers for the architecture, instead of the CLN layers (as in the standard NODE). This is also referred as Pre-Activation (as the ReLU operation is carried before convolution). The convolution operation used here is again a 3×3 convolution. Finally, the output from these layers is added to the original input (so that these layers only learn the residual). This constitutes one residual block. We take six such blocks, back to back, to form our ResNet. As mentioned above, down-sampling is applied before this network, followed by the pooling operation and FC layer at the end.

5.2. Preprocessing

Standard image processing is done on our image dataset similar to that in Zhu et al. (2019), before it could be fed into the model. This is done so as to ensure that the images carry all the relevant information, needed to accurately train the model.

We mainly apply three image transformations. First, the image is resized from $424 \times 424 \times 3$ pixels to $32 \times 32 \times 3$ pixels, using bilinear interpolation. This makes our training process faster as the number of dimensions in the input image is largely reduced. The transformation applied involves randomly flipping the image horizontally. The final transformation involves image normalization, where all the three channels are normalized according to appropriate values. The data set is randomly split into training and testing set in the ratio 9:1 with 25911 images for training set and 2879 images for testing set as discussed in Section 2. We repeat this procedure 10 times and compute the mean and variance over the evaluation metrics.

5.3. Implementation details

We use mini-batch gradient descent with a batch size of 256. The initial learning rate is set to 0.1 and then decreased by a factor of 10 to 30 K and 60 K iterations. The weight decay is set to 0.0001, dropout probability value to 0.8, and the weights are initialized in the same way as in He et al. (2015).

5.4. Comparison of computational costs

Here, we provide a brief comparison of the computational costs between ResNet, NODE, and NODE_ACA. To keep the comparison simple, as in Chen et al. (2018), let L be the number of ResNet layers, and $\hat{L}$ be the number of forward-passes in NODE. The computational cost depends on L and $\hat{L}$ for ResNet and NODE, respectively. While L is fixed and is a hyper-parameter, $\hat{L}$ is dependent on the error-tolerance we set for NODE and NODE_ACA. If the error-tolerance is high, $\hat{L}$ is comparable with L. If the error tolerance is low, then $\hat{L}$ is much higher than L. More details can be found in Table 1 of Chen et al. (2018).

Similarly, while comparing $\hat{L}$ in the case of NODE and NODE_ACA, $\hat{L}$ is roughly half in case of NODE_ACA as compared to NODE. This is the reason why NODE_ACA is roughly twice as faster than NODE. More details can be found in Table 1 of Zhuang et al. (2020). For our analysis for NODE_ACA, it took about 12 h to run (with about 11 h for training and one hour for the testing) a single network on a NVIDIA dgx server with P100 GPU. For NODE, it took about double the processing time and for RESNET it took about 90 min (80 min for training and 10 min for testing).

6. Results and discussion

6.1. Model accuracy

Standard NODE model (trained with adjoint method) achieves an accuracy of 91%–94%, when trained with the Runge–Kutta method. The accuracy achieved by ResNet on similar architecture is between 89%–94%. With NODE_ACA, we get accuracy between 91%–95%. Thus, we can say that NODE_ACA achieves comparable accuracy to ResNet, while having one-third the number of parameters. For all the three networks, we get poor accuracy for cigar-shaped images (class=2), due to the small number of images available for training. This is also consistent with the results in Zhu et al. (2019).

Table 1 provides the confusion matrix for ResNet for different classes, while Tables 2 and 3 provide the confusion matrix for standard NODE and NODE_ACA, respectively, after averaging over the ten runs. The confusion matrix simply shows the contamination and completeness a particular category was classified into, among all the classes. It gives an idea, with which the other class, model confused a particular class the most. Data shown in the confusion matrices were calculated, when the output of the model led to the maximum correct predictions (or purity), when

Table 1

Confusion Matrix for ResNet (averaged over all the 10 runs), where 0 : Completely round smooth, 1 : In-between smooth, 2 : Cigar-shaped smooth, 3 : Edge-on, and 4 : Spiral.

	0	1	2	3	4
0	757	57	0	0	29
1	22	741	3	4	36
2	0	12	11	27	5
3	0	8	6	363	13
4	7	12	1	13	748

Table 2

Confusion Matrix for NODE (averaged over all the 10 runs), where 0 : Completely round smooth, 1 : In-between smooth, 2 : Cigar-shaped smooth, 3 : Edge-on, and 4 : Spiral.

	0	1	2	3	4
0	800	24	0	0	13
1	36	700	0	8	22
2	0	6	12	31	2
3	0	5	7	365	8
4	10	29	1	22	750

Table 3

Confusion Matrix for NODE_ACA (averaged over all the 10 runs), where 0 : Completely round smooth, 1 : In-between smooth, 2 : Cigar-shaped smooth, 3 : Edge-on, and 4 : Spiral.

	0	1	2	3	4
0	786	36	0	0	12
1	25	721	0	3	16
2	0	4	23	20	1
3	1	4	12	361	5
4	5	36	3	15	747

summed over all the classes. There is no other threshold which needs to be tuned for our problem. On the whole, the results of the completely round, the in-between, the edge-on and the spiral are extremely excellent, except for the cigar-shaped images (class=2). It happens due to the small number of class=2 images for training.

6.2. Parameters discussion

`ResNet` with 6 layers has 0.6 million parameters. Standard NODE on the other hand has total of 0.2 million parameters, for both Euler and Runge–Kutta ODE solving methods. NODE_ACA network also has the same number of parameters as NODE. Thus, NODE and NODE_ACA achieve similar overall accuracy with about one-third of the parameters as Resnet.

6.3. Precision, completeness (recall), and F1

We compare the precision, Completeness (which is referred to as Recall in the Machine Learning Community), and F1 scores of standard NODE, ResNet, and NODE_ACA. The Precision metric is the ratio of the total true positives to the total number of observations labelled positive. The Completeness tries to quantify what proportion of actual positives is correctly classified. While F1 is the harmonic mean of precision and completeness. More detailed definitions of these metrics can be found in Bethapudi and Desai (2018). These three metrics for Resnet, NODE, and NODE_ACA can be averaged over all the ten iterations which can be found in Table 4. We can clearly see that the performance of NODE and NODE_ACA is comparable to that of ResNet.

6.4. ROC curve

ROC curve is an acronym for receiver operating characteristic curve. It plots the true positive against false positive rate, and

Table 4
Accuracy, Precision, Completeness (Recall), F1 scores for ResNet, NODE, NODE_ACA along with error bars using all the 10 runs.

Class	Accuracy			Precision			Completeness			F1		
	ResNet	NODE	NODE_ACA	ResNet	NODE	NODE_ACA	ResNet	NODE	NODE_ACA	ResNet	NODE	NODE_ACA
0	0.897± 0.0061	0.956± 0.0008	0.939± 0.0007	0.962± 0.0029	0.946± 0.0007	0.961± 0.0012	0.897± 0.0061	0.956± 0.0008	0.939± 0.00071	0.928± 0.0029	0.951± 0.0003	0.950± 0.0006
1	0.917± 0.0039	0.914± 0.0011	0.941± 0.0013	0.892± 0.0032	0.917± 0.0017	0.896± 0.0007	0.918± 0.0039	0.914± 0.0011	0.942± 0.0013	0.905± 0.0019	0.916± 0.0009	0.918± 0.0006
2	0.193± 0.0294	0.236± 0.0050	0.468± 0.0092	0.522± 0.0491	0.595± 0.0202	0.596± 0.010	0.193± 0.0294	0.236± 0.0050	0.468± 0.0092	0.281± 0.0356	0.338± 0.0081	0.524± 0.0092
3	0.930± 0.0064	0.948± 0.0007	0.938± 0.0011	0.889± 0.0063	0.859± 0.0020	0.899± 0.0013	0.930± 0.0064	0.948± 0.0007	0.938± 0.0011	0.909± 0.0041	0.901± 0.0013	0.918± 0.0006
4	0.957± 0.0036	0.9244± 0.0011	0.924± 0.0008	0.899± 0.0037	0.944± 0.0007	0.954± 0.0008	0.957± 0.0036	0.924± 0.0011	0.924± 0.0008	0.927± 0.0024	0.934± 0.0006	0.939± 0.0006
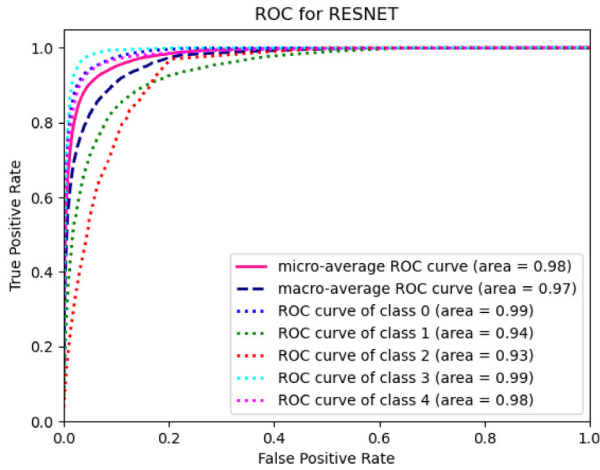
Fig. 2. Average ROC curve for ResNet for the different classes.

Fig. 3. Average ROC curve for standard NODE for the different classes.

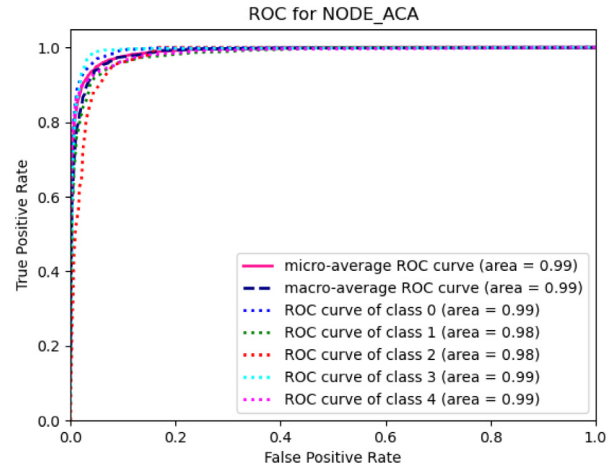**Fig. 4.** Average ROC curve for NODE_ACA for the different classes.

shows how well a model is able to classify. Area under this curve is called AUC. The closer AUC is to one, better is the model in terms of classification. The ROC curves for each class, for ResNet (Fig. 2), standard NODE (Fig. 3) and NODE_ACA (Fig. 4) (after averaging over all the 10 runs) are shown. Micro and macro average for all the classes are also shown in the same figures. Micro-average is calculated by binarizing the output of each label, while macro-average is just the unweighted average of each label. Thus, micro-average takes class-imbalance into account, giving more weightage to bigger classes while macro-average is forced to recognize each class correctly. These averages are well-versed in ML community (Abdar et al., 2021). As we can see, the ROC curves for NODE for both the adjoint and the ACA techniques are very close to those of ResNet, for every image class. In Fig. 5, we plot the average curve (averaged over all the classes), for all the three aforementioned techniques. As we can see, the performance of NODE and NODE_ACA is comparable to ResNet.

7. Conclusions

In this paper, we have used NODE with adjoint training and NODE_ACA technique for the task of galaxy morphology classification, and also compared its performance with ResNet. The dataset used for this purpose is a subset of Galaxy Zoo 2 dataset. We find that the number of parameters in NODE (standard and ACA) is about one-third compared to its counterpart ResNet, and it achieves this without compromising the performance. Accuracy achieved by NODE (with adjoint training) is the same as ResNet, while NODE (with ACA technique) achieves an accuracy of 91%–95%, where the ground truth is determined by the Galaxy Zoo 2 classifications. Also, with both the NODE techniques, we can easily trade-off accuracy for speed, which is not possible for ResNet.
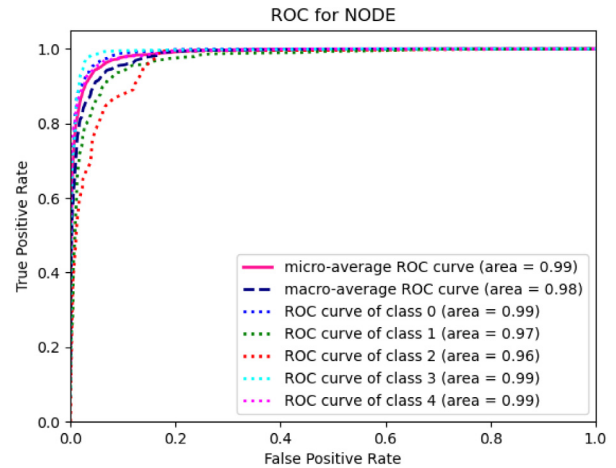
We also compare all the three models using other metrics such as precision, Completeness, F1, and AUC. Through our results, we conclude that, the performance of NODE and NODE_ACA is comparable to ResNet, for all these metrics, while providing all the advantages guaranteed by ResNet. We also illustrate this, by plotting the average performance of standard NODE, ResNet and NODE_ACA, on one graph, as shown in Fig. 5.

From our experiments, we therefore conclude that NODE has several advantages over ResNet and can easily supersede ResNet.
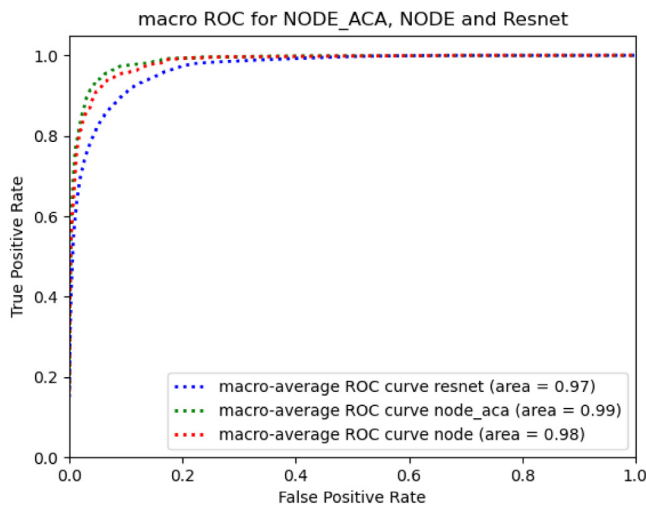
Fig. 5. Average Precision–Recall curves for ResNet, NODE, NODE_ACA.

With large scale astronomical surveys coming up, and more and more data being generated from these surveys, there is a pressing need to replace such classification tasks with robust deep learning models. These emerging deep learning models can not only help speed-up the process of training and classification, but also provide better insights, by breaking down the process in series of small steps. Thus, researchers have better control over the process, and can easily trade-off one parameter (like accuracy) with another (like speed). Our methodology would prove to be beneficial for upcoming large scale astronomical surveys such as Vera Rubin LSST, Euclid, WFIRST etc.

All our codes used for the analysis in this work are publicly available at github.com/rg321/torch_ACA. We also provide some rudimentary guidance on how to use both NODE and NODE_ACA for the supervised classification problem in the appendices.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Appendix A. Instructions for using NODE_ACA within PyTorch

We provide some bare-bones guidelines on how to access and use NODE_ACA for any classification problem. NODE_ACA is available in both TensorFlow and PyTorch. For this work, we have used PyTorch and hence provide some a rudimentary guide on the usage of NODE_ACA in PyTorch. Our full analysis has also been provided in a github link at github.com/rg321/torch_ACA_gz.

Zhuang et al. (2020) provide a PyTorch package at https://github.com/juntang-zhuang/torch_ACA, which can be easily plugged into the existing models, with support for multi-GPU training and higher-order derivative. A simple way to plug NODE_ACA into your existing code is as follows.

```
from torch_ACA import odesolve_adjoint as odesolve
out = odesolve(odefunc, x, options)
```

One then needs to write a custom data-loader in order to load the data into the model. For example, to load the Galaxy Zoo data images into the model for training and testing purposes, a custom data-loader *get_gz_loaders* in file *data_loader.py* is written and used. This loader is written in PyTorch's standard DataLoader style. It fetches the images using PyTorch's ImageFolder function, apply necessary transformations, splits them into training and testing parts and finally returns *train* and *test DataLoader*, which are standard PyTorch objects used for data loading.

Once the data-loaders are in place, rest of the flow is the same as for any other dataset like CIFAR10, ImageNet (Russakovsky et al., 2015) etc. The only thing that needs to be done now is tuning the hyper-parameters in order to get the best accuracy or whatever desired. For example, to run it on galaxy-zoo datset, run the following command -:

```
python train.py --num_epochs 15 --dataset galaxyzoo
--batch_size 64 --test_batch_size 32
```

An example usecase of NODE_ACA in the TensorFlow library can be found in https://github.com/titu1994/tfdiffeq

Appendix B. Instructions for using NODE within PyTorch

After creating the DataLoader as described in above section, use the following command (on the Linux prompt) to use NODE architecture on your data (for example, dataset used is MNIST here) -:

```
python train.py  --data galaxyzoo
--optimizer sgd  --lr 0.1 --solver runge_kutta  --use_ode
```

References

Abazajian, K.N., et al., SDSS Collaboration, 2009. The seventh data release of the sloan digital sky survey. Astrophys. J. Suppl. 182, 543–558, arXiv:0812.0649.

Abdar, M., Salari, S., Qahremani, S., Lam, H.-K., Karray, F., Hussain, S., Khosravi, A., Acharya, U.R., Nahavandi, S., 2021. Uncertaintyfusenet: Robust uncertainty-aware hierarchical feature fusion with ensemble Monte Carlo dropout for COVID-19 detection. arXiv:2105.08590.

Abell, P.A., et al., LSST Science, LSST Project Collaboration, 2009. LSST Science book, version 2.0. arXiv:0912.0201.

Abraham, R.G., van den Bergh, S., 2001. The morphological evolution of galaxies. Science 293 (5533), 1273–1278, arXiv:astro-ph/0109358.

Ball, N.M., Brunner, R.J., 2010. Data mining and machine learning in astronomy. Int. J. Mod. Phys. D 19 (7), 1049–1106, arXiv:0906.2173.

Barchi, P.H., de Carvalho, R.R., Rosa, R.R., Sautter, R.A., Soares-Santos, M., Marques, B.A.D., Clua, E., Gonçalves, T.S., de Sá-Freitas, C., Moura, T.C., 2020. Machine and deep learning applied to galaxy morphology - a comparative study. Astron. Comput. 30, 100334, arXiv:1901.07047.

Baron, D., 2019. Machine learning in astronomy: a practical overview. arXiv e-prints, arXiv:1904.07248.

Bernardi, M., Shankar, F., Hyde, J.B., Mei, S., Marulli, F., Sheth, R.K., 2010. Galaxy luminosities, stellar masses, sizes, velocity dispersions as a function of morphological type. Mon. Not. R. Astron. Soc. 404 (4), 2087–2122, arXiv:0910.1093.

Bethapudi, S., Desai, S., 2018. Separation of pulsar signals from noise using supervised machine learning algorithms. Astron. Comput. 23, 15, arXiv:1704.04659.

Bhambra, P., Joachimi, B., Lahav, O., 2021. Explaining deep learning of galaxy morphology with saliency mapping. arXiv e-prints, arXiv:2110.08288.

Bundy, K., Ellis, R.S., Conselice, C.J., 2005. The mass assembly histories of galaxies of various morphologies in the GOODS fields. Astrophys. J. 625 (2), 621–632, arXiv:astro-ph/0502204.

Buta, R.J., 2013. Galaxy morphology. In: Falcón-Barroso, J., Knapen, J.H. (Eds.), Secular Evolution Of Galaxies. p. 155.

Chen, E.Z., Chen, T., Sun, S., 2020. MRI image reconstruction via learning optimization using neural ODEs. arXiv:2006.13825.

Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D., 2018. Neural ordinary differential equations. CoRR abs/1806.07366, arXiv:1806.07366.

Conselice, C.J., 2003. The relationship between stellar light distributions of galaxies and their formation histories. arXiv:astro-ph/0303065.

de Diego, J.A., Nadolny, J., Bongiovanni, A., Cepa, J., Pović, M., Pérez García, A.M., Padilla Torres, C.P., Lara-López, M.A., Cerviño, M., Pérez Martínez, R., Alfaro, E.J., Castañeda, H.O., Fernández-Lorenzo, M., Gallego, J., González, J.J., González-Serrano, J.I., Pintos-Castro, I., Sánchez-Portal, M., Cedrés, B., González-Otero, M., Heath Jones, D., Bland-Hawthorn, J., 2020. Galaxy classification: deep learning on the OTELO and cosmos databases. Am. Acad. Pediatr. 638, A134, arXiv:2005.07228.

Desmond, H., Ferreira, P.G., 2020. Galaxy morphology rules out astrophysically interesting $f(R)$. doi:10.1103/PhysRevD.102.104060.

Dieleman, S., Willett, K.W., Dambre, J., 2015. Rotation-invariant convolutional neural networks for galaxy morphology prediction. Mon. Not. R. Astron. Soc. 450 (2), 1441–1459, arXiv:1503.07077.

Freeman, P.E., Izbicki, R., Lee, A.B., Newman, J.A., Conselice, C.J., Koekemoer, A.M., Lotz, J.M., Mozena, M., 2013. New image statistics for detecting disturbed galaxy morphologies at high redshift. arXiv:1306.1238.

Fuketa, H., Morita, Y., 2020. Neural ODE with temporal convolution and time delay neural networks for small-footprint keyword spotting. arXiv:2008.00209.

Goddard, H., Shamir, L., 2020. A catalog of broad morphology of pan-STARRS galaxies based on deep learning. arXiv e-prints, arXiv:2010.06073.

Groha, S., Schmon, S.M., Gusev, A., 2020. Neural ODEs for multi-state survival analysis. arXiv:2006.04893.

Gupta, K., Chandraker, M., 2020. Neural mesh flow: 3D manifold mesh generationvia diffeomorphic flows. arXiv:2007.10973.

Hashimoto, K., Hu, H.-Y., You, Y.-Z., 2020. Neural ODE and holographic QCD. arXiv:2006.00712.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385, arXiv:1512.03385.

Hubble, E.P., 1926. Extragalactic nebulae.. Astrophys. J. 64, 321–369.

Kennicutt, J., 1998. Star formation in galaxies along the hubble sequence. Annu. Rev. Astron. Astrophys. 36, 189–232, arXiv:astro-ph/9807187.

Khan, A., Huerta, E.A., Wang, S., Gruendl, R., Jennings, E., Zheng, H., 2019. Deep learning at scale for the construction of galaxy catalogs in the dark energy survey. Phys. Lett. B 795, 248–258, arXiv:1812.02183.

Kolen, J.F., Kremer, S.C., 2001. Gradient flow in recurrent nets: The difficulty of learning LongTerm dependencies. In: A Field Guide To Dynamical Recurrent Networks. pp. 237–243.

Kremer, J., Stensbo-Smidt, K., Gieseke, F., Pedersen, K.S., Igel, C., 2017. Big universe, big data: machine learning and image analysis for astronomy. IEEE Intell. Syst. 32 (2), 16–22.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances In Neural Information Processing Systems, Vol. 25. pp. 1097–1105.

Lackner, C.N., Gunn, J.E., 2012. Astrophysically motivated bulge-disk decompositions of SDSS galaxies. doi:10.1111/j.1365-2966.2012.20450.x.

Laureijs, R., et al., EUCLID Collaboration, 2011. Euclid definition study report. arXiv:1110.3193.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Lianou, S., Barmby, P., Mosenkov, A.A., Lehnert, M., Karczewski, O., 2019. Dust properties and star formation of approximately a thousand local galaxies. Am. Acad. Pediatr. 631, A38, arXiv:1906.02712.

Lintott, C.J., Schawinski, K., Slosar, A., Land, K., Bamford, S., Thomas, D., Raddick, M.J., Nichol, R.C., Szalay, A., Andreescu, D., Murray, P., Vandenberg, J., 2008. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. Mon. Not. R. Astron. Soc. 389 (3), 1179–1189, arXiv:0804.4483.

Lotz, J.M., Primack, J., Madau, P., 2004. A new nonparametric approach to galaxy morphological classification. Astron. J. 128 (1), 163–182, arXiv:astro-ph/0311352.

Lu, Y., Zhong, A., Li, Q., Dong, B., 2017. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. arXiv:1710.10121.

Martin, G., Kaviraj, S., Hocking, A., Read, S.C., Geach, J.E., 2020. Galaxy morphological classification in deep-wide surveys via unsupervised machine learning. Mon. Not. R. Astron. Soc. 491 (1), 1408–1426, arXiv:1909.10537.

Menanteau, F., Ford, H.C., Motta, V., Benitez, N., Martel, A.R., Blakeslee, J.P., Infante, L., 2005. The morphological demographics of galaxies in the ACS hubble ultra deep parallel fields. arXiv:astro-ph/0509759.

Odewahn, S.C., Cohen, S.H., Windhorst, R.A., Philip, N.S., 2001. Automated galaxy morphology: A Fourier approach. arXiv:astro-ph/0110275.

Reza, M., 2021. Galaxy morphology classification using automated machine learning. Astron. Comput. 37, 100492.

Romanowsky, A.J., Fall, S.M., 2012. Angular momentum and galaxy formation revisited. Astrophys. J.s 203 (2), 17, arXiv:1207.4189.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. Imagenet large scale visual recognition challenge. arXiv:1409.0575.

Sersic, J., 1963. Influence of the atmospheric and instrumental dispersion on the brightness distribution in a galaxy.

Simard, L., Willmer, C.N.A., Vogt, N.P., Sarajedini, V.L., Phillips, A.C., Weiner, B.J., Koo, D.C., Im, M., Illingworth, G.D., Faber, S.M., 2002. The DEEP groth strip survey II. Hubble space telescope structural parameters of galaxies in the groth strip. arXiv:astro-ph/0205025.

Simmons, B.D., Lintott, C., Willett, K.W., Masters, K.L., Kartaltepe, J.S., Häußler, B., Kaviraj, S., Krawczyk, C., Kruk, S.J., McIntosh, D.H., Smethurst, R.J., Nichol, R.C., Scarlata, C., Schawinski, K., Conselice, C.J., Almaini, O., Ferguson, H.C., Fortson, L., Hartley, W., Kocevski, D., Koekemoer, A.M., Mortlock, A., Newman, J.A., Bamford, S.P., Grogin, N.A., Lucas, R.A., Hathi, N.P., McGrath, E., Peth, M., Pforr, J., Rizer, Z., Wuyts, S., Barro, G., Bell, E.F., Castellano, M., Dahlen, T., Dekel, A., Ownsworth, J., Faber, S.M., Finkelstein, S.L., Fontana, A., Galametz, A., Grützbauch, R., Koo, D., Lotz, J., Mobasher, B., Mozena, M., Salvato, M., Wiklind, T., 2017. Galaxy zoo: quantitative visual morphological classifications for 48 000 galaxies from CANDELS. Mon. Not. R. Astron. Soc. 464 (4), 4420–4447, arXiv:1610.03070.

Skibba, R.A., Bamford, S.P., Nichol, R.C., Lintott, C.J., Andreescu, D., Edmondson, E.M., Murray, P., Raddick, M.J., Schawinski, K., Slosar, A., Szalay, A.S., Thomas, D., Vandenberg, J., 2009. Galaxy zoo: disentangling the environmental dependence of morphology and colour. Mon. Not. R. Astron. Soc. 399 (2), 966–982, arXiv:0811.3970.

Spergel, D., Gehrels, N., Breckinridge, J., Donahue, M., Dressler, A., Gaudi, B.S., Greene, T., Guyon, O., Hirata, C., Kalirai, J., Kasdin, N.J., Moos, W., Perlmutter, S., Postman, M., Rauscher, B., Rhodes, J., Wang, Y., Weinberg, D., Centrella, J., Traub, W., Baltay, C., Colbert, J., Bennett, D., Kiessling, A., Macintosh, B., Merten, J., Mortonson, M., Penny, M., Rozo, E., Savransky, D., Stapelfeldt, K., Zu, Y., Baker, C., Cheng, E., Content, D., Dooley, J., Foote, M., Goullioud, R., Grady, K., Jackson, C., Kruk, J., Levine, M., Melton, M., Peddie, C., Ruffa, J., Shaklan, S., 2013. Wide-field InfraRed survey telescope-astrophysics focused telescope assets WFIRST-AFTA final report. arXiv e-prints, arXiv:1305.5422.

Spindler, A., Geach, J.E., Smith, M.J., 2020. AstroVaDEr: AStronomical variational deep embedder for unsupervised morphological classification of galaxies and synthetic image generation. arXiv e-prints, arXiv:2009.08470.

Tanoglidis, D., Ćiprijanović, A., Drlica-Wagner, A., 2020. DeepShadows: SEparating low surface brightness galaxies from artifacts using deep learning. arXiv e-prints, arXiv:2011.12437.

Tuccillo, D., Huertas-Company, M., Decencière, E., Velasco-Forero, S., 2017. Deep learning for studies of galaxy morphology. In: Brescia, M., Djorgovski, S.G., Feigelson, E.D., Longo, G., Cavuoti, S. (Eds.), Astroinformatics, Vol. 325. pp. 191–196, arXiv:1701.05917.

Valle, R., Reda, F., Shoeybi, M., Legresley, P., Tao, A., Catanzaro, B., 2019. Neural ODEs for image segmentation with level sets. arXiv:1912.11683.

Willett, K.W., Galloway, M.A., Bamford, S.P., Lintott, C.J., Masters, K.L., Scarlata, C., Simmons, B.D., Beck, M., Cardamone, C.N., Cheung, E., Edmondson, E.M., Fortson, L.F., Griffith, R.L., Häußler, B., Han, A., Hart, R., Melvin, T., Parrish, M., Schawinski, K., Smethurst, R.J., Smith, A.M., 2017. Galaxy zoo: morphological classifications for 120 000 galaxies in hst legacy imaging. Mon. Not. R. Astron. Soc. 464 (4), 4176–4203, arXiv:1610.03068.

Willett, K.W., Lintott, C.J., Bamford, S.P., Masters, K.L., Simmons, B.D., Casteels, K.R.V., Edmondson, E.M., Fortson, L.F., Kaviraj, S., Keel, W.C., Melvin, T., Nichol, R.C., Raddick, M.J., Schawinski, K., Simpson, R.J., Skibba, R.A., Smith, A.M., Thomas, D., 2013. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. Mon. Not. R. Astron. Soc. 435 (4), 2835–2860, arXiv:1308.3496.

York, D.G., et al., SDSS Collaboration, 2000. The sloan digital sky survey: Technical summary. Astron. J. 120, 1579–1587, arXiv:astro-ph/0006396.

Zhu, X.-P., Dai, J.-M., Bian, C.-J., Chen, Y., Chen, S., Hu, C., 2019. Galaxy morphology classification with deep convolutional neural networks. Astrophys. Space Sci. 364 (4), 55, arXiv:1807.10406.

Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., Duncan, J., 2020. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. arXiv:2006.02493.