Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **522,** 292–311 (2023) https://doi.org/10.1093/mnras/stad989
Advance Access publication 2023 April 03

Feature guided training and rotational standardization for the morphological classification of radio galaxies

Kevin Brand [1]★ Trienko L. Grobler [1]★ Waldo Kleynhans,[2] Mattia Vaccari [3,4,5] Matthew Prescott [4] and Burger Becker [1]

[1] *Computer Science Department, Stellenbosch University, Cnr Banghoek Road & Joubert Street, Stellenbosch 7600, South Africa*
[2] *Department of Electrical Electronic and Computer Engineering, University of Pretoria, 0081, Pretoria, South Africa*
[3] *Inter-University Institute for Data Intensive Astronomy, Department of Astronomy, University of Cape Town, 7701 Rondebosch, Cape Town, South Africa*
[4] *Inter-University Institute for Data Intensive Astronomy, Department of Physics and Astronomy, University of the Western Cape, Robert Sobukwe Road, 7535 Bellville, Cape Town, South Africa*
[5] *INAF - Istituto di Radioastronomia, via Gobetti 101, 40129 Bologna, Italy*

Accepted 2023 March 29. Received 2023 February 27; in original form 2022 October 28

ABSTRACT

State-of-the-art radio observatories produce large amounts of data which can be used to study the properties of radio galaxies. However, with this rapid increase in data volume, it has become unrealistic to manually process all of the incoming data, which in turn led to the development of automated approaches for data processing tasks, such as morphological classification. Deep learning plays a crucial role in this automation process and it has been shown that convolutional neural networks (CNNs) can deliver good performance in the morphological classification of radio galaxies. This paper investigates two adaptations to the application of these CNNs for radio galaxy classification. The first adaptation consists of using principal component analysis (PCA) during pre-processing to align the galaxies' principal components with the axes of the coordinate system, which will normalize the orientation of the galaxies. This adaptation led to a significant improvement in the classification accuracy of the CNNs and decreased the average time required to train the models. The second adaptation consists of guiding the CNN to look for specific features within the samples in an attempt to utilize domain knowledge to improve the training process. It was found that this adaptation generally leads to a stabler training process and in certain instances reduced overfitting within the network, as well as the number of epochs required for training.

Key words: radio continuum: galaxies – methods: data analysis – methods: statistical – techniques: image processing.

1 INTRODUCTION

Most, if not all, massive galaxies have a supermassive black hole at their centre. Active galaxies are galaxies that have an Active Galactic Nucleus (AGN), that is formed by the accretion of particles onto their central supermassive black hole (Padovani et al. 2017). Radio galaxies are active galaxies that emit in the radio part of the electromagnetic spectrum, typically via the synchrotron emission produced from electrons that are accelerated by magnetic fields, present in galaxies (Stacy & Vestrand 2003). This emission can most effectively be observed by making use of radio interferometers (Hardcastle & Croston 2020).

The morphological classification of radio galaxies consists of grouping them into classes according to their shapes. This classification process and the definition of a classification system describing the wide variety of observed radio galaxies play a vital role in answering other questions that are fundamental to astronomy, such as how galaxies and galaxy clusters form and evolve over cosmic time (Hardcastle & Croston 2020).

Two important morphological classes, namely class I and II Fanaroff–Riley galaxies (FRI and FRII galaxies), were first identified by Fanaroff & Riley (1974). To differentiate between these classes, the Fanaroff–Riley ratio (FR ratio) was defined as the ratio of the distance between their 'hot spots' (radio bright areas observed on either side of a galaxy's core) to the total distance between the edges of the radio source. Galaxies with an FR ratio less than 0.5 have bright regions close to the galaxy's core and are classified as FRI galaxies, whilst galaxies with an FR ratio greater than 0.5 are edge-brightened and are classified as FRII galaxies.

Some radio galaxies have a more complex warped shape, that is, the two radio jets emanating from their centres are not aligned to form a straight line but are rather at an angle with each other. These radio galaxies are known as bent-tail radio galaxies and they can be separated into two further classes depending on the angle between their jets, namely narrow-angle tailed (NAT, Rudnick & Owen 1976) and wide-angle tailed galaxies (WAT, Owen & Rudnick 1976). For the rest of this paper, these subclasses will be grouped together and referred to as bent galaxies.

The final class of radio galaxies that we will consider in this paper is the FR0 class, also known as the compact class. This class was proposed for the classification of a subset of galaxies that do not fit into the original FR dichotomy very well. These galaxies

★ E-mail: kevinbrand99@gmail.com (KB); tlgrobler@sun.ac.za (TLG)

have a very bright region surrounding the core, but as explained by Baldi, Capetti & Giovannini (2015), they have a much more compact morphology. Galaxies with other, more complex morphologies fall outside the scope of this paper.

With the development of new facilities, such as the Australian Square Kilometre Array Pathfinder (ASKAP, Johnston et al. 2008), the MeerKAT radio telescope (Jonas 2009), the Low-Frequency Array (LOFAR, van Haarlem et al. 2013), and the Murchison Widefield Array (MWA, Beardsley et al. 2019), it has become possible to conduct surveys that will detect much larger numbers of radio galaxies than was previously possible. As the rate of radio source detection grows, it becomes increasingly difficult to manually analyse and classify each source. One approach to solving this problem would be to make use of crowd-sourcing, where large groups of individuals voluntarily classify each source (Banfield et al. 2015). However, this solution relies on the availability of enough individuals that can generate reliable labels, which might not be feasible as the volume of data increases, especially once the construction of the SKA (Dewdney et al. 2009, 2013) has been completed.

An alternative approach has been gaining traction in recent years. This approach consists of automating the analysis and labelling of sources and is much more robust to an influx in the volume of data that is available. Machine learning is a useful tool in this automation process and has been shown to be quite effective (e.g. De La Calleja & Fuentes 2004; Alhassan, Taylor & Vaccari 2018; Hocking et al. 2018).

Initially most of the machine learning techniques that were applied were shallow learning techniques, such as decision trees (Proctor 2003, 2006) and support vector machines (Sadeghi, Javaherian & Miraghaei 2021), which achieved impressive results with respect to the classification of radio galaxies. However, as the field of deep learning continues its rapid growth, a clear shift has been observed towards the use of deep neural networks.

To better understand why this shift is sensible, it is important to understand what some of the limitations are of the shallow techniques and why these limitations are relevant to radio galaxy data. One such limitation is that shallow learning algorithms tend to underperform when compared to deep learning techniques on various high-dimensional data sets (LeCun, Bengio & Hinton 2015). Seeing as radio galaxy data sets tend to consist of relatively large images, this can be problematic. Furthermore, when applying shallow machine learning to high-dimensional data sets, a common approach is to reduce the dimensionality by extracting informative features. However, this approach has certain drawbacks. By only training the shallow models on the extracted features, we are introducing bias into the models and preventing them from finding informative features that we might not be aware of (Janiesch, Zschech & Heinrich 2021). Deep learning has also been applied with great success in applications that utilize large corpora of data (Chen & Lin 2014), whilst shallow learning algorithms have been shown to be inefficient in comparison (Najafabadi et al. 2015). Thus, it is expected that deep learning will play a crucial role in extracting information from future large-scale surveys.

With respect to the morphological classification of radio galaxies, Aniyan & Thorat (2017) found that convolutional neural networks (CNNs) can be very effective. They constructed three CNNs that each played the role of a binary classifier that simply had to differentiate between two classes. The predictions from these binary classifiers were then combined to make a final prediction. This model showed that CNNs could achieve similar performance to manual morphological classification, but that they could do it quicker.

Since then, various new CNNs have been created to perform morphological classification on radio galaxies. Some examples of these CNNs include the single multiclass CNN created by Alhassan et al. (2018) to classify radio galaxies as compact, bent, FRI or FRII galaxies, a multiclass CNN trained on the first data set from the Radio Galaxy Zoo project (Banfield et al. 2015) to differentiate between compact and extended sources (Lukic et al. 2018), CNNs used for the cross-identification of host galaxies (Alger et al. 2018), an augmented Fast Region-based CNN (Fast R-CNN Girshick 2015) that was applied to combinations of corresponding radio and infrared images by (Wu et al. 2018) to simultaneously find and classify radio sources. Ma et al. (2019) made use of a large number of unlabelled samples to pre-train a convolutional auto-encoder (CAE), which is a type of neural network that learns how to encode images into lower dimensional representations. The weights of the encoder section of this network were then fine-tuned, using a set of labelled samples to perform the morphological classification of radio galaxies.

Recently, Bowles et al. (2021) made use of an attention-gating mechanism[1] instead of fully-connected layers in a CNN to show that one could use considerably fewer parameters to achieve performance that is on par with the CNNs previously applied to morphological classification. Another important development was the development of group-equivariant CNNs for radio galaxy classification by Scaife & Porter (2021), who explained that conventional CNNs are not equivariant to certain isometries, such as rotations and reflections, which contribute to intra-class variability. Thus, these CNNs might not be able to accurately classify a sample during inference if its orientation differs from the samples observed during training, because the rotation will have an effect on the outputs of the convolutional layers and will thus affect the classification. Scaife & Porter (2021) addressed this problem by making use of a subspace of conventional convolution kernels that is equivariant to rotations. The use of these kernels led to an improvement in the performance of the CNN, which serves as additional proof that rotations in samples can have an effect on the performance of CNNs.

1.1 Addressing rotational variations in radio sources

Making use of group-equivariant CNNs is not the only way to address rotational variations in radio sources. One can also augment the training data by applying rotations to the training samples, thus increasing the number of training samples and making it possible for CNNs to learn what classes look like at various orientations. This approach is widely adopted in the literature (Aniyan & Thorat 2017; Alhassan et al. 2018; Lukic et al. 2018; Ma et al. 2019; Becker et al. 2021; Bowles et al. 2021), however it has certain flaws. Scaife & Porter (2021) noted that by using data augmentations to address rotational variations, one runs the risk of learning identical kernels for different orientations, which is a waste of some of the computational power of the CNN. Furthermore, they also explain that invariance gained through data augmentation does not guarantee invariance during inference. Aside from the points mentioned by Scaife & Porter (2021), one should also consider that data augmentation can lead to a dramatic increase in the number of samples used during training, which will tend to lead to much longer training times.

Bearing this in mind, this paper investigates a third approach that addresses the rotational variations in radio sources, which attempts to standardize the rotation of all of the radio sources as a pre-processing step. The first step in this process is to construct a matrix that

[1] Attention-gating mechanisms work similarly to human attention mechanisms by identifying and focusing on the features that are the most informative for the given task.

contains the 2D coordinates of the pixels that belong to the galaxy. Principal component analysis (PCA) is then applied to this matrix to determine the principal components of the galaxy. By aligning these principal components with the main axes, we effectively standardize the rotation of the radio sources. A similar approach was applied to optical galaxy images by De La Calleja & Fuentes (2004), but we specifically investigate the effectiveness of this pre-processing step on radio sources and compare it to the commonly used approach of rotational data augmentation.

1.2 Incorporation of domain knowledge during learning

The patterns/features that are deemed to be important by neural networks during training are generally unconstrained, which makes it possible for them to arrive at non-optimal solutions (Lawrence, Giles & Tsoi 1998; Bader, Hölldobler & Marques 2008) or to make decisions that violate known physical constraints within the problem domain (Daw et al. 2021; Chen & Liu 2021). These networks can also overfit on noise or inconsequential patterns within the training data, but due to the 'black box' nature of neural networks (Olden & Jackson 2002; Seidel, Schimmler & Borghoff 2019) it is very difficult to verify whether the network has learned sensible patterns and to determine what those patterns might be (Seidel et al. 2019; Woods, Chen & Teuscher 2019). This also means that neural networks cannot truly contribute to the development of our understanding of the problem domain (Olden & Jackson 2002).

These problems have led to the development of a variety of approaches that attempt to address the 'black box' nature of neural networks. Some of these approaches train neural networks and then develop an understanding of their decision-making process by extracting symbolic rules from the trained models (Huynh & Reggia 2011; Seidel et al. 2019). The use of symbolic rules lies at the core of knowledge-based neural networks (KBNN, Fu 1995; Kolman & Margaliot 2008).

To better understand KBNNs, one should first consider that there are two paradigms of learning, theoretical learning and learning from practical experience. Expert systems follow the theoretical paradigm of learning and utilize expert knowledge of the problem domain during the decision-making process, whilst empirical learning follows the practical paradigm and learns how to solve a task by training on a large number of representative samples from the problem domain (Towell & Shavlik 1994).

Both expert systems and empirical learning systems have been able to achieve incredible results on real-world problems (Goethe & Bronzino 1995; Rahman & Hazim 1996; Sun, Wang & Wang 2008). However, each system has its disadvantages. Expert systems require a large investment of both time and resources to construct complete domain theories. This leads to the use of incomplete theories, which could be detrimental in certain scenarios (Mitchell, Keller & Kedar-Cabelli 1986). Conversely, systems that make use of empirical learning have no domain knowledge, which leads to different problems. For example, without domain knowledge, the system is not aware of any constraints that determine whether results are viable and thus there are no guarantees that they will produce viable solutions (Mitchell et al. 1986). The lack of domain knowledge also means that the performance of the system is entirely dependent on whether the training data is informative with respect to the problem domain. Thus, poor performance is all but guaranteed if the quality of the training data is poor (Schank, Collins & Hunter 1986).

A combination of these two systems might be able to address some of these disadvantages, which is exactly what KBNNs attempt to do. KBNNs combine these two systems by incorporating domain theories into a neural network in the form of symbolic rules. They then make use of empirical learning algorithms to fine-tune the weights of the neural network. This approach has some distinct benefits. For example, the use of empirical learning algorithms and domain samples will refine any incomplete or incorrect domain theories which will lead to better performance. Conversely, if empirical learning leads to results that violate theoretical constraints, domain theories can be used to constrain the network to only produce viable results. Considering these benefits, it should come as no surprise that KBNNs have been shown to be incredibly effective and that they tend to generalize better than both expert and empirical learning systems (Towell & Shavlik 1994).

The implementation details of KBNNs are not relevant to this paper and will not be discussed. However, it is important to note that these networks contain some neurons that are not dependent on domain theory. Towell & Shavlik (1994) showed that the addition of these neurons provides neural networks with the necessary flexibility to refine incomplete or inaccurate domain theories and to address some of the side effects that they cause. The addition of these neurons is relevant to this paper, because this technique will also be used in some of our guided architectures.

Theory-guided data science (TGDS) is a paradigm of data science that one can argue is a continuation of the work done for KBNNs. The goal of TGDS is to produce models that do not exclusively make use of either data or theoretical knowledge, but instead utilize both (Karpatne et al. 2017). This goal is achieved by making use of a wide variety of techniques that introduce domain knowledge into models that are traditionally purely data-driven. One such technique consists of using Lagrangian multipliers to enforce theoretical constraints in the loss functions of models that make use of error-based learning. Other techniques include the use of theoretical knowledge to efficiently initialize parameters, fine-tune model outputs and select probabilistic models that are similar to the known distribution of the problem domain (Karpatne et al. 2017).

A technique that is commonly used in TGDS, which will also be used in this paper, is feature engineering. This technique consists of manipulating and combining existing features to create new features that are expected to be informative when classifying samples. Such features are selected by making use of prior knowledge of the domain and the task at hand (Nargesian et al. 2017). The features that are engineered can be very simple, such as the ratio between two other features (Heaton 2016), or they can be much more complex, such as the results of statistical transformations that were applied to the data. If good features are selected, it has been shown that feature engineering can lead to considerable improvements in the performance of models (Yu et al. 2010; Mulaudzi & Ajoodha 2021).

Based on the results achieved by both KBNNs and TGDS, this paper investigates whether it would be useful to guide neural networks to look for specific features during training. This approach was first tested on a toy example, where a shallow neural network was constructed to emulate the behaviour of the XOR binary operator. Once it had been determined that such a guided neural network was viable, guided CNNs were constructed and applied to the morphological classification of radio galaxies. The performance of the guided CNNs was evaluated and compared to the performance of unguided CNNs with a similar architecture to determine whether there were any considerable benefits that justify guiding neural networks during training.

Our guided neural networks should not be mistaken with Feature-guided Denoising Convolutional Neural Networks (FDCNN Dong, Ma & Basu 2021). FDCNNs make use of an explainable artificial intelligence technique, guided backpropagation, to identify and

Table 1. XOR data set.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

extract important features in ultrasound images. These features are used to improve the performance of a denoising neural network. There are no significant similarities between FDCNNs and our guided networks, other than their name.

Section 2 of this paper presents the data sets that were used, the pre-processing steps that were applied to the data, as well as the process that was followed to extract features for the training of the guided neural networks. This is followed by an explanation of the neural network architectures in Section 3. The analytic process that was used to evaluate and compare the rotational pre-processing steps and the guided networks is discussed in Section 4. We then present and discuss the results of our experiments in Section 5. Finally, we draw our conclusions in Section 6.

2 DATA SETS AND PRE-PROCESSING PROCEDURES

This section discusses the data sets that were used in this paper, the steps involved in the pre-processing procedures, as well as the feature extraction process. Section 2.1 presents the data sets and Section 2.2 discusses the various pre-processing techniques that were applied to the radio galaxy images. Finally, Section 2.3 explains how features were extracted from these images.

2.1 Data sets

2.1.1 XOR binary operator

The XOR binary operator is quite a simple operator. It is also known as the exclusive or operator, which means that it only returns true if one of two binary inputs is equal to one. This operator was used in this paper as a simple example to test whether a certain approach to guiding neural networks was feasible or not. The XOR data set is very small and only consists of four samples that cover the entire domain. These samples can be seen in Table 1.

2.1.2 Radio galaxy data set

Our radio galaxy data set was assembled as follows. We first gathered positional and morphological information from the following well-documented catalogues extensively used in the literature: CoNFIG (Gendre & Wall 2008; Gendre, Best & Wall 2010), GROUPS (Proctor 2011), FR0CAT (Baldi, Capetti & Massaro 2018), FRICAT (Capetti, Massaro & Baldi 2017a), FRIICAT (Capetti, Massaro & Baldi 2017b), and WATCAT (Missaglia et al. 2019). We then downloaded 300 by 300 pixels cutouts from the FIRST survey (Becker, White & Helfand 1995; Helfand, White & Becker 2015) in FITS format using the SkyView online tool (McGlynn, Scollick & White 1998). We then confirmed that the morphology of each radio galaxy as indicated in the different input catalogues was reliable when inspected at the depth and the resolution of the FIRST survey via visual inspection of all radio galaxies by at least three team members. Objects for which consensus was not achieved were collaboratively rechecked.

Table 2. Number of radio sources from different classes within our sample.

Class	Number
Compact	208
FRI	182
FRII	357
Bent	213
Total	960

Where consensus could not be achieved after the recheck, the radio galaxy was dropped from our sample. Our resulting sample, which we called FIRST Radio Galaxy Morphology Reference Catalogue (FRGMRC), consists of a total 960 sources. The breakdown in different source classes is shown in Table 2. The FRGMRC and the supporting FIRST fits cutouts used for our work is publicly available at https://doi.org/10.5281/zenodo.7645530.

2.2 Radio galaxy pre-processing

In this section, we will discuss the various pre-processing techniques that were used in this paper. We will provide an overview of what these techniques consist of and what purpose they serve. The details of when each pre-processing step was used are delayed until the discussion of the investigative procedure in Section 4.2.1.

2.2.1 Normalization

Normalization consists of scaling all pixel values within an image to fall within the range [0, 1]. This pre-processing step is always applied to the radio galaxy images in this paper, irrespective of what they will be used for, because the pixel ranges tend to differ between images, which leads to an unnecessary increase in the intra-class variances. Normalization is achieved by applying equation (1) to each pixel in an image. In this equation, X represents the entire image and x_{ij} represents the individual pixels at row i and column j.

$$x_{ij} = \frac{x_{ij} - \min(X)}{\max(X) - \min(X)} \tag{1}$$

2.2.2 Thresholding

Thresholding is a technique that is frequently used to extract important pixels from an image by removing noise and background pixels. This technique is especially important in this paper, because it improves the quality of the features that are extracted from the images and it assists with determining the principal components of the galaxies.

We use a slightly different thresholding approach in this paper than is commonly used in the literature due to our rotational standardization algorithm's sensitivity to thresholding artefacts. The first step in our approach is to identify how much noise is present in the image. This is done by constructing a histogram of pixel values for each image. The number of bins that are populated with a large number of pixels provides an indication of the amount of noise that is present in the image. If there is noise present in the image there will be a wider range of pixel values, which will lead to more histogram bins being populated.

If there is very little noise present in the image we can simply use a static value to threshold the image. Alternatively, if there is noise present, we make use of a large quantile as a threshold to extract the brightest galaxy pixels. We then make use of morphological dilations to 'grow' these bright pixels to form regions that contain all of the

galaxy pixels. We also make use of morphological operations to find small artefacts in the background and to remove them.

In cases where the noise is severe, we create a new matrix, where each element represents the number of pixel values in the immediate neighbourhoods of the corresponding pixel in the image. We can exploit this information to extract the galaxy pixels, because the pixels along the borders of the galaxy will have background pixels, noise pixels and galaxy pixels in their neighbourhood, which means that the pixel values in their neighbourhoods will be much more diverse. Thus, if we extract the elements with a large range of neighbouring pixel values, we will be able to extract most of the pixels near the border of the radio galaxy. We then apply morphological hole filling, which will fill holes that are surrounded by the pixels that we have extracted. In doing so, we will be able to also extract the galaxy pixels that are not close to the border of the galaxy.

Our approach sometimes includes a small number of pixels from the regions surrounding the galaxy, which might lead to slight distortions of the shape of the galaxy. However, it is very successful in removing thresholding artefacts, which is more important for our algorithms. For more details about our thresholding algorithm, the reader is referred to the Github repository.[2]

2.2.3 Rotational data augmentation

As was mentioned earlier, an approach that is widely used to address rotational variations in radio galaxy data is to augment the training data by applying rotations to each of the training samples. This approach is also implemented in this paper to allow for a comparison with the method of rotational standardization.

Each sample in the FRGMRC data set was rotated in 60° intervals, from 0° to 300°, which creates six copies of each sample, but with different orientations. This makes it possible for the CNNs to learn what each class will look like at different orientations.

Ideally, one would use smaller rotational intervals to ensure that the CNN could recognize the classes at any possible orientation, but this is likely to lead to a significant waste of time and computational resources, because the CNN might not learn much from samples that are too similar with regards to their orientation.

One of the metrics which will be used to evaluate the performance of rotational data augmentation is the amount of time that it takes to train a CNN whilst using this approach. It was determined that with small rotational intervals, data augmentation leads to near-perfect accuracy on the samples in the testing data, which means a large portion of the training time might have been wasted. Thus, a rotational interval was selected that led to a slight reduction in the accuracy of the CNN, because this would ensure that time was not wasted on unnecessary training.

2.2.4 Rotational standardization

To understand the approach that we use to standardize the rotation of the radio galaxy images, it is vital to first understand PCA. The origins of PCA are unclear, but it can be traced back as far as 1829. However, the form of PCA that is commonly used today is attributed to the work done by Hotelling (1933).

PCA is commonly used as a technique for dimensionality reduction. As explained by Rao (1964), PCA consists of finding orthogonal vectors that describe the data the best. To do so, it is necessary to find the vectors along which the variance in the data is maximized. These vectors were established to be the eigenvectors of the covariance matrix of the data and are also known as the principal components of the data set. Thus, the first principal component will be the direction in which the variance is the largest, the second principal component will be the direction with the second largest variance etc. To reduce the dimensionality of the data to q dimensions, each sample can be projected onto the first q principal components, which represent the q vectors along which the variance in the data is the largest and will thus lead to the smallest loss of information.

In this paper, we will not use PCA to reduce the dimensionality of our data, but will instead use it to identify the vectors that represent the directions of maximum spread in our radio galaxies. We do so by first normalizing and thresholding each sample to identify the pixels that belong to the radio galaxies. The quality of PCA is correlated to the quality of this thresholding, so any improvements in the thresholding algorithm will lead to an improvement in the results of PCA.

Once we have identified the galaxy pixels, we construct a $2 \times P$ matrix, X, where P is the number of pixels that were extracted. Each column in X represents the coordinates of one of the pixels. The next step consists of centring the galaxy at the origin of the coordinate system. This is done by subtracting the mean of each row in the matrix from the elements in that row. We then make use of the popular NUMPY library (Harris et al. 2020) to calculate the singular value decomposition (SVD) of the centred matrix. This decomposition is shown in equation (2). For more information regarding the SVD, the reader is referred to Klema & Laub (1980)

$$X = U \Sigma V^T \tag{2}$$

In this decomposition U is a 2×2 orthogonal matrix, Σ is a $2 \times P$ diagonal matrix and V^T is a $P \times P$ orthogonal matrix. We will only need the U matrix, because its columns contain the eigenvectors of the covariance matrix of X, which we know are also the principal components of X. We also know that these principal components maximize the variance of our data.

Seeing as our data is the coordinates of the galaxy pixels, these principal components will maximize the spread of the galaxy pixels. Thus, if we rotate our coordinate system's axes to align with these principal components, we should be able to remove any rotation from our image.

One way to achieve this goal would be to pre-multiply the transpose of U with the coordinate of each pixel in our image. This would provide us with the corresponding coordinates in a coordinate system that is aligned with the principal components. However, this approach would be unnecessarily complicated and very inefficient.

Instead, we start by determining whether U is the identity matrix. If it is, we can determine that no rotation is necessary to align our coordinate system with the principal components. If U is not the identity matrix, we need to calculate the angle with which we have to rotate our image. We know that pre-multiplication by the transpose of the U matrix will lead to a rotation in our data and potentially a reflection. Reflections are possible, because the sign of an eigenvector can be changed and the result will still be an eigenvector. Thus, the sign of any of the columns of U can be changed and the column will still represent a principal component. This means that U^T has to be a combination of an arbitrary number of reflection matrices and a rotation matrix.

Consider that a rotation matrix has the form shown in either equation (3) or equation (4), depending on the direction of the rotation. These matrices are not in their conventional form, because the flipped y-axis in computer graphics changes the direction of rotation.

[2]The repository is available at https://doi.org/10.5281/zenodo.7785574

(a) Original image (b) Image after standardising the orientation

Figure 1. Results of applying the rotational standardization algorithm, which leads to a 41.68° clockwise rotation.

$$R^{\circlearrowleft} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3}$$

$$R^{\circlearrowleft} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4}$$

By utilizing this knowledge, we should be able to extract the angle, θ, from U. We do so by calculating the arcsin of the first element of the second column of U^T. We know that this element might have been multiplied by -1, but seeing as $-\sin(\theta) = \sin(-\theta)$, this should not affect the absolute value of θ.

Once we have calculated the rotation angle, we need to determine whether we should perform a clockwise or an anticlockwise rotation. Before we can determine the direction of the rotation, we will first need to determine whether the sign of any of the columns has been altered.

We know that the sign of θ does not affect the result of the cosine function. Thus, we can use this result to establish whether the sign of one of the columns have been changed. We do so by calculating $\cos(\theta)$ and comparing the result to the first element of the first column and the second element of the second column of U. If the sign of the elements in U differs from the sign of our calculated value, we can assume that the sign of the corresponding column has been flipped. We reverse this reflection by multiplying that column with -1.

This process should address any reflections, meaning that we should be left with only a rotation matrix. To determine whether we need to perform a clockwise or anticlockwise rotation, we have to determine which of the sine elements in U^T is the negative element. We do so by dividing each sine element with $\sin(\theta)$ and inspecting the sign of the results.[3] If the first element of the second column has a negative sign, we know that the rotation matrix is in the form shown in equation (3), otherwise it is in the form shown in equation (4).

[3]In the scenario where the sine elements have the same sign and the cosine elements are zero we will not be able to determine which column's sign has been changed. However, in this scenario the rotation angle has to be 90°, which means the direction of the rotation is not important.

Table 3. Thresholds used for core extraction.

Standard deviation range	Threshold
$0.2 < \sigma$	80th quantile
$0.13 < \sigma <= 0.2$	93rd quantile
$\sigma <= 0.13$	98th quantile

We tested our method of standardizing the orientation of the radio galaxies, by determining whether we could use the rotation angle that we calculated and the assumptions that we made to reconstruct the U^T matrix. We could do so successfully for all of the samples in our training data. For an example of the results that this algorithm can achieve, the reader is referred to Fig. 1.

2.3 Radio galaxy feature extraction

We want to guide the CNNs to look for specific features in the samples that we consider to be informative. This section will discuss the features that we selected, as well as how we extracted them. We extracted two sets of these features. One set was exclusively extracted by algorithms, whilst in the other, two of the features were extracted manually. This was done to determine whether either approach of extracting features resulted in significant improvements in the performance of the guided networks.

We selected four features that we deemed to be informative with respect to the morphological classification of the radio galaxies. These features include the FR ratio, whether the galaxy is bent or not, the number of so-called 'bright spots' that are present in the galaxy, which we will refer to as cores for the rest of the paper, and the ratio of the size of the cores to the total size of the galaxy.

Before extracting any features, the images had to be normalized and thresholded. Once this had been done, the first step consisted of calculating the standard deviation of the galaxy pixels. The cores of the galaxy were then extracted by using a threshold that depended on this standard deviation. A number of thresholds were used during experimentation to find the thresholds that delivered the best results. These thresholds can be seen in Table 3.

The number of pixels larger than the threshold are then counted. This count is divided by the total number of pixels in the galaxy, which leads to the first feature that we want to extract, the ratio of pixels in the cores to pixels in the galaxy. This feature indicates whether the cores dominate the galaxy, which is especially prevalent in compact galaxies.

Once this feature had been extracted we could count the number of cores in the galaxy by counting the number of connected components in the thresholded image. Whilst counting the connected components, we also recorded the coordinates of the pixel with the maximum value in each component. We used these coordinates to represent the center of each component. If there are two cores we can simply use the distance between them to represent the inter-core distance in the FR ratio. If there are more than two cores we calculated the distance between each core and its nearest neighbour and used the average distance to represent the inter-core distance. If there was only one core, it could mean that we are dealing with a compact source, but there is also the possibility that it is an FRI source where the distance between the cores is so small that they look like one core. Thus, we calculate the distance from one side of this core to the other side and use a third of this distance to represent the inter-core distance. This is only an estimate of what the distance would be if cores are overlapping, but it is accurate enough to indicate that the cores are very close to one another.

Before we can continue with feature extraction, it is necessary to determine whether there was a lot of noise present in the sample. The reader might remember that if there is a lot of noise in a sample, the thresholding algorithm tends to include some background pixels or noise that is connected to the galaxy, which complicates the process of identifying potential curves in the galaxy's shape.

Thus, if we detect a lot of noise during thresholding, we use the standard deviation and mean of the extracted galaxy pixels to calculate the z-score. The z-score represents how many standard deviations each pixel is from the mean. We want to get rid of any background pixels that were mistakenly extracted from the original image. These background pixels are generally much darker than the rest of the pixels, which means that they will have a considerably larger z-score than the rest of the pixels. Thus, we only keep pixels with a z-score that is smaller than the median z-score. This might seem to be a very aggressive threshold, but it was found to deliver good results for the next steps of the feature extraction process.

Once we had removed any potential background pixels from the image, we used an approach that is very similar to the approach from Section 2.2.4. The only difference was that we multiplied the U matrix from the SVD directly with the matrix that contains the coordinates of the galaxy pixels. This multiplication produced the coordinates that the pixels would have if we rotated them to align the principal component with the axes. By calculating the difference between the minimum and maximum column value in our new coordinate system, we can determine the approximate distance from one side of the galaxy to the other. The inter-core distance can then be divided by this distance to calculate the FR ratio.

The final step is to determine whether the galaxy is bent. We do this by using the distance between the minimum and maximum column that we just calculated, as well as the distance between the minimum and maximum row. If the galaxy is not bent, one of these distances should be considerably larger than the other, but if the galaxy is bent these distances should be closer to one another.

Unfortunately, some factors complicate the process of identifying curves in practice. These factors include the imperfect identification of galaxy pixels, the small angle between tails in NATs, the similar vertical and horizontal dimensions in certain compact, FRI and FRII galaxies, etc.

To select good thresholds for these values, we took into consideration that bent galaxies are much more likely to have a curve in the galaxy's body, whilst curves in the bodies of other classes of radio galaxies should be much rarer. Thus, we selected thresholds that would separate most of the bent galaxies from the other galaxies. There were no thresholds that could perfectly achieve this separation, but we do not view this as a problem, seeing as we want the feature to indicate whether a curve is present in the galaxy. We do not want it to become a class label for bent-tailed galaxies.

The hyperparameters used in this process are specific to this data set. They can be heavily optimized, because this algorithm will not be applied to unseen samples, meaning that there is no risk of overfitting. However, if this approach is applied to a new data set, it might be necessary to tune these hyperparameters for the new samples.

We also manually determined the number of cores in each galaxy and whether the galaxy was curved in any way. These features were not extracted by an astronomer and thus simply indicate the number of regions in an image that were considered to be cores and whether it seemed like a galaxy was curved in any way. The manually extracted features were used in the additional set of features that were mentioned earlier to determine whether manually extracting features leads to any significant improvements in performance.

3 ARCHITECTURES

For this paper, a few guided, as well as unguided neural networks were constructed. These networks consist of a standard CNN that simply classifies samples, an auxiliary CNN that is used to extract the selected features from samples and three guided CNNs. These guided networks are referred to as the wide, multiheaded and merged neural networks and they are all guided to use information from the extracted features to a certain extent when classifying samples.

This section will explain the architectures of the CNNs that were applied to the FRGMRC data as well as the process of hyperparameter selection. The full details of these architectures can be found in the Github repository. The architectures of the networks that were applied to the XOR data can be seen in Appendix A.

The reader will notice that the architectures in this section contain dense blocks and convolutional blocks. Each convolutional block contains two convolutional layers, followed by a max pooling layer. The dense blocks each consist of a dense layer, which is followed by a dropout layer.

These architectures have not been extensively optimized. This was done purposefully, because we want to be able to identify whether the changes that we make to the network architectures affect overfitting and model performance. If the models were already fine-tuned to the point where there is no overfitting present in the networks and the network performs perfectly, it would be difficult to determine whether our changes affect the models' performance.

Thus, we are of the opinion that it is more important to ensure that the hyperparameters and the general structure remain as similar as possible in the various networks to ensure that there are no factors that could affect network performance, other than the factors that we are investigating.

3.1 Standard CNN

As the name suggests, this network simply represents a standard unguided CNNs as can be seen in Fig. 2. It was used as a baseline with which to compare the performance of the guided CNNs. The standard

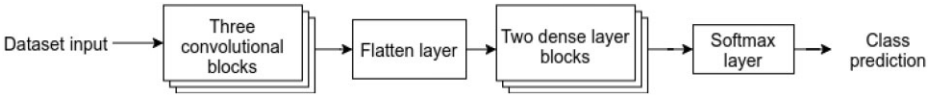


Figure 2. Structure of the SCNNs.

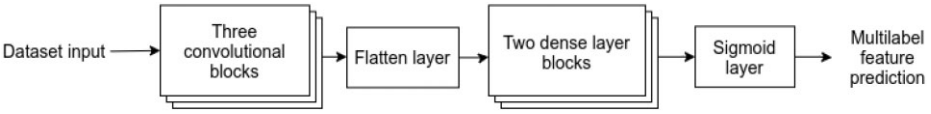


Figure 3. Structure of the ACNNs.

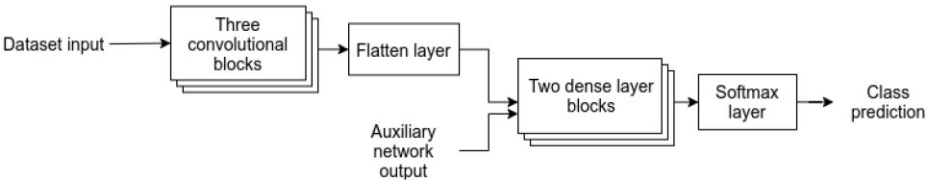


Figure 4. Structure of the WCNNs.

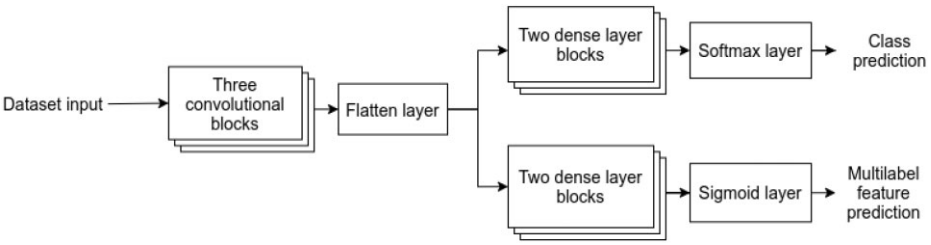


Figure 5. Structure of the MhCNNs.

convolutional neural network (SCNN) was also used to compare our rotational standardization technique to rotational augmentation.

3.2 Auxiliary CNN

A representation of the auxiliary convolutional neural network (ACNN) architecture can be seen in Fig. 3. From this figure it can be seen that the ACNN architecture is very similar to that of the SCNN. The only difference is that the output layer in the ACNN makes use of a sigmoid activation function to produce multilabel feature predictions, whilst the SCNN makes use of a softmax activation function to make a single class prediction. This network is used to engineer new features for the data sets, by determining the presence of the predetermined features within each of the given samples.

3.3 Wide CNN

The wide convolutional neural network (WCNN) was based on the neural network architecture presented by Cheng et al. (2016). Similar architectures have also been applied to radio galaxy data (e.g. Tang et al. 2022). Fig. 4 shows a representation of the modified architecture used for the WCNN in this paper.

These networks have the benefit that they can learn from multiple inputs in parallel. In this paper, the network specifically receives additional inputs just before the first dense layer blocks. These additional inputs are the features engineered by the ACNNs. They are fed into the WCNNs after the convolutional layers, seeing as the convolutional layers are meant for image data and will probably not be very useful if applied to the engineered features.

These networks are the first guided neural networks and were used to establish whether feature engineering using domain knowledge could improve the training process in any way.

3.4 Multiheaded CNN

The multiheaded convolutional neural networks (MhCNN) are based on an existing neural network architecture that has been used for a wide variety of applications. These networks generally have a shared base, which branches out into a number of heads that each performs a specific task. For an example of these multiheaded neural networks, the reader is referred to the work done by Li, Ng & Natsev (2019).

The structure of the multiheaded networks used in this paper is shown in Fig. 5. As can be seen, the MhCNNs split up into two separate heads after the convolutional layers. One head is similar to the ACNN, whilst the other head is similar to the SCNN.

At first glance, it might not be apparent that MhCNNs are guided neural networks. However, the reader should note that the two heads share multiple convolutional blocks. The optimization algorithm will thus adjust the weights according to an aggregate loss function which uses the loss functions of both heads. This means that the auxiliary head will have an effect on parameters in the convolutional layers, which will in turn have an effect on the main head. Thus, during the training process, the auxiliary head will still be able to guide the main head, albeit indirectly. Ideally, the shared convolutional layers will find patterns that are beneficial to both the auxiliary and main heads, which could speed up training.

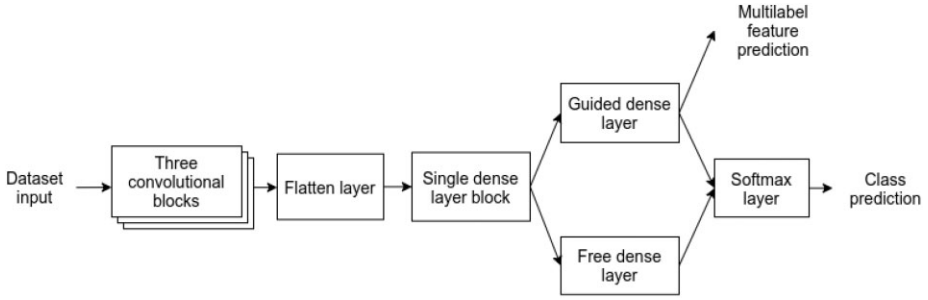


Figure 6. Structure of the MCNNs.

After training, MhCNNs provide the extracted features as an additional output, which can then be used for other tasks. One such task could be to make use of these features to validate the class assigned to each sample. For example if the network did not detect any curves in the body of a sample, but classified it as a bent-tailed galaxy, one might need to double check the result. Such a network could also be used to learn additional labels for the samples, such as the number of peaks and components in each sample, similar to what was done for CLARAN (Wu et al. 2018).

3.5 Merged CNNs

The merged convolutional neural networks (MCNNs) are a continuation of the fundamental concepts used in the Mh and WCNNs. These MCNNs also make use of engineered features to guide the training process, but they construct these features within the hidden layers of the network. They then feed these engineered features into the output layer.

A variant of the merged neural network was first constructed for the XOR binary operator, seeing as the network structure could be kept simple.[4] This made it possible to determine the updated terms from first principles. Doing so proved that backpropagation and stochastic gradient descent were still applicable to networks that made use of the outputs of hidden neurons in their loss function.

Once it had been confirmed that it was possible to construct a merged neural network, MCNNs were constructed for the FRGMRC data set. A representation of the architecture of these networks can be seen in Fig. 6. As can be seen in this figure, the final dense layer is split into two sections. One section consists of 'guided' neurons that are used to extract the selected features from the sample, whilst the other consists of standard 'free' neurons that are free to find any patterns in the given sample that will assist in classification.

MCNNs have similar benefits to those of the MhCNNs. They also learn to extract features from the samples, which can be used for other downstream tasks, such as verifying whether the classification labels are sensible or not. They also have the added benefit that the extracted features are fed back into the dense layer that classifies the samples, which means that the extracted features can play a more direct role in the classification process.

3.6 Hyperparameter selection

Hyperparameters in machine learning models can drastically change model behaviour and thus influence the quality of the output. Thus, it is important to select good values for them.

[4]Refer to Appendix A

Unfortunately, due to the large number of interchangeable, complex elements present in neural networks, they tend to have quite a large number of hyperparameters. This makes it infeasible to explore the entire hyperparameter space in search of good values. Instead, this paper evaluated the networks' performance on a validation set for a few values of each hyperparameter and then selected the values that seem to lead to the best performance.

A Nadam optimizer with a learning rate of 0.0001 was used for all of the networks in this paper. A grid search was conducted that determined the best learning rate for the SCNNs. The same learning rate was used for the other neural networks, seeing as their convergence speed would be compared.

For the hidden layers, ReLU and ELU activation functions were used. ReLU activation functions were used in the convolutional layers and ELU activation functions were used in the dense layers. With respect to weight initialization, He normal weight initialization was used for all of the dense layers and Xavier uniform initialization was used for the convolutional layers. The bias terms were all initialized to zero.

4 EMPIRICAL ANALYSIS

In this section, the analytic process will be discussed that was used to compare the various neural networks. Section 4.1 will discuss the loss functions and performance metrics that were used. This will be followed by a discussion of the procedure that was used to generate reliable performance evaluations for the rotational pre-processing steps as well as each of the neural network architectures.

4.1 Loss functions and metrics

To be able to compare the various neural networks, it was necessary to select a few performance metrics. This subsection will discuss these metrics as well as the loss functions that were used.

4.1.1 Categorical cross-entropy loss

Categorical cross-entropy takes the probability of a sample belonging to a class and compares it with the true label of that sample. It then computes a logarithmic penalty which corresponds to how far away the prediction probability is from the desired output. This calculation can be seen in equation (5). In this equation, N refers to the number of samples in the data set, M refers to the number of classes, and p_{ij} is the output from the neural network corresponding to class j and sample i. Finally, $t_{ij} = 1$ if sample i belongs to class j. Categorical cross-entropy loss is used for models that make a single class prediction. Thus, this loss function was used for the neural networks that classified samples

from the FRGMRC data set to evaluate the class predictions that they made.

$$C = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} t_{ij}\log_2(p_{ij}) \tag{5}$$

4.1.2 Binary cross-entropy loss

The TensorFlow implementation of binary cross-entropy loss is very similar to categorical cross-entropy loss, except that it is used for binary classification. The equation used to calculate this loss is very similar to the equation used for categorical cross-entropy loss, as can be seen in equation (6). The value N still refers to the number of samples, t_i represents the true label for sample i and p_i represents the predicted label. This loss function was used to evaluate how well neural networks were able to extract the bent feature from the FRGMRC data.

$$C = -\frac{1}{N}\sum_{i=1}^{N} t_i\log_2(p_i) + (1-t_i)\log_2(1-p_i) \tag{6}$$

4.1.3 Mean squared error

The mean squared error (MSE) loss function is commonly used to evaluate the performance of neural networks when they are utilized to predict continuous variables. In this paper, it was used to evaluate how well neural networks could extract the continuous-valued features from the radio galaxies, such as the FR-ratio and the number of cores. This loss can be seen in equation (7). N represents the number of samples, t_i is the true value of sample i and p_i is the value predicted by the network.

$$C = \frac{1}{N}\sum_{i=1}^{N}(t_i - p_i)^2 \tag{7}$$

A variant of this loss function was also used to optimize the networks that were trained on the XOR data set. The only difference was that the entire equation was divided by 2. This has no impact on the optimization process, because the weights that minimize C will also minimize $\frac{C}{2}$. This division is generally excluded in the literature, but it has been used here to simplify the derivative of the loss function.

4.1.4 Mean absolute error

The mean absolute error (MAE) is very similar to the MSE. The only difference is that it uses the absolute distance between predictions and true values instead of the squared distance. We also make use of this metric to evaluate how well neural networks could extract the continuous-valued features from the radio galaxies. This metric is included, because it makes it easier to interpret how well the neural network can extract the continuous-valued features. The MAE can be seen in equation (8). N represents the number of samples, t_i is the true value of sample i and p_i is the value predicted by the network.

$$C = \frac{1}{N}\sum_{i=1}^{N}|t_i - p_i| \tag{8}$$

4.1.5 Combined loss function

The MhCNNs and the MCNNs have two outputs, the features that were extracted and the class label of the sample. To ensure that all of the outputs are optimized, the loss function has to incorporate at least one component for each output that can provide an empirical evaluation of the quality of the output.

However, not all of these outputs are necessarily equally important. It might be more important to extract certain features, or it might be more important to classify the sample. Thus, weights were used to indicate how important each component was. The weighted loss function can be seen in equation (9). In this equation, m represents the main component of the loss, the categorical cross-entropy loss, a represents the auxiliary component which indicates how well the networks were able to extract the specified features from the samples, and w_1 and w_2 represent the weights that correspond to each component.

$$C = w_1 m + w_2 a \tag{9}$$

4.1.6 Classification accuracy

The classification accuracy of a model is very easy to calculate. It is simply the number of correct classifications divided by the number of samples in the given data set. This metric gives a good indication of the models' classification performance.

4.1.7 F1 Score

The F1 score is a slightly more intricate performance metric. To understand the F1 score, it is necessary to first understand what is meant by the precision and recall of a network. Precision is calculated as shown in equation (10). In this equation TP is the number of true positives and indicates how frequently the model accurately identifies the presence of the positive class. In a binary classification problem, the positive class is the primary class that one is interested in. The opposite is true for the negative class. FP is the number of false positives and indicates how frequently the model falsely identifies the presence of the positive class.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

Equation (11) shows how the recall of the model is calculated. FN is the number of false negatives and indicates how frequently a model falsely determines that the positive class is not present.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

These two metrics are then combined as shown in equation (12) to create the F1 score. For this paper, the F1 score was used as a metric to evaluate the classification performance of the networks on the FRGMRC data. This was necessary, because there is a slight imbalance in the class representations and the F1 score is less sensitive to these imbalances than the classification accuracy is. In this case, the positive class is simply whether a sample belongs to one of the morphological classes.

$$\text{F1} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

Seeing as the F1 score was applied to problems that were not binary classifications, the calculations had to be adapted. The precision and recall was calculated for each class. The average precision and recall for all of the classes was then used to calculate the overall F1 score. This is commonly known as the macro-average F1 score.

Table 4. Number of samples in training, validation, and test data set.

Training	Validation	Test	Total
702	87	88	877

4.1.8 *Overfitting metric*

We want to be able to determine whether guided neural networks generalize better than unguided neural networks. To be able to compare these networks we need a metric that provides an indication of how much overfitting is present in the network. One indicator that is commonly used, is the difference between the training loss and the validation loss. If overfitting occurs, the training loss will keep improving, whilst the validation loss might plateau or even become worse.

Before we calculate the difference between the training and validation losses, it is important to acknowledge that the losses that are being optimized in the various networks differ from one another and have different scales. Thus, it would not be fair to compare them directly, because losses with a larger range of values will automatically make it seem like the network is overfitting more severely.

To address this problem, we first record all of the training and validation losses for every training run of a network and then use these losses to formulate an approximation of the range of values that we can expect to observe for that loss. We then use this information to normalize the losses, by using an equation similar to equation (1). The only difference is that X will now represent all of the loss values that were recorded and x will refer to one of these loss values.

Once this has been done, all of the losses will have values between zero and one, which makes it possible to draw a fair comparison between the various networks. At this point we can simply calculate the difference between the training loss and the validation loss when training finishes. The larger this difference is, the more overfitting has occurred.

4.2 Investigation procedure

Due to the stochastic nature of neural networks, one cannot expect the training process to deliver consistent results. To get a reliable indication of the performance of the various networks, it is necessary to train each network multiple times and to then use statistical measures to evaluate their performance over all of the runs.

Thus, each of the neural networks in this paper were trained and evaluated over twenty independent runs. The performance metrics were recorded for each run. For the radio galaxy networks this was done by making use of the Tensorboard callback in Tensorflow (Abadi et al. 2015). After all of the runs had finished, the logs were used to determine the average performance of the networks.

To establish how well the networks generalize, the FRGMRC data set was split up into a training, validation, and test data set before the first training run. The sizes of these data sets are given in Table 4. The total number of samples in this table does not align with the total number of samples in Table 2. The difference is due to some compact sources that were added to the FRGMRC data set after experimentation, as well as a few duplicate sources that were dropped at an advanced stage of the project. Neither of these changes affect our results, because the models were all trained and evaluated on the same set of samples.

Recall that we also trained shallow networks for the XOR data set as a feasibility study. These networks were trained for a maximum of 10 000 epochs on the the full data set. This is acceptable, because the XOR data set is exhaustive. The radio galaxy CNNs were trained on the training data set for a maximum of 100 epochs. After each epoch, they were also evaluated on the validation data set, which indicated how well the models were generalizing to unseen data.

To save some time an early stopping mechanism was used to stop the training of the XOR neural networks when the loss function reached a value that was smaller than 0.01. Early stopping was also used to stop the training of the radio galaxy networks if the validation loss had not improved for 5 epochs. This assisted in addressing the overfitting in the network to a certain extent. Once the training of the radio galaxy networks had concluded, the network's state was reset to the state when the minimum validation loss was achieved. The network was then evaluated on the test data set.

4.2.1 *Rotational standardization investigation details*

In this paper, we compare the effectiveness of rotational augmentation and rotational standardization to address rotational variations that are commonly found in the radio galaxy data. In these experiments, SCNNs were trained on the FRGMRC data set after different pre-processing steps were applied.

To evaluate rotational augmentation, each of the training and validation data sets were augmented, using the approach detailed in Section 2.2.3. The test data set was not augmented, because this would lead to an unfair comparison of the various approaches, since they would effectively be evaluated on different data sets. All of the samples were then normalized, after which a SCNN was trained and evaluated on the processed data sets over twenty training runs.

Rotational standardization was evaluated in a similar fashion. The only difference was that instead of augmenting the data, the rotation of each of the samples in the training, validation, and test data sets was standardized, using the approach that was explained in Section 2.2.4.

Finally, we also trained and evaluated an SCNN on a data set where the only pre-processing that was done consisted of normalizing the samples. This SCNN was used as a baseline to determine how important it is to address the rotational variations in a given data set.

4.2.2 *Feature guided training investigation details*

To investigate whether there are any benefits in guiding the networks to look for specific features, three guided CNN architectures were evaluated, as well as one unguided architecture, which served as a baseline with which to compare their performance. The guided architectures included WCNNs, MCNNs, and MhCNNs.

As mentioned earlier, the MCNNs and MhCNNs make use of weights to balance the contribution of the main and auxiliary outputs to the total loss of the network. To get an indication of the effect that these weights have, three MhCNNs and three MCNNs were trained. Each network made use of different weight combinations, which can be seen in Table 5. All of the components of the loss functions had a similar order of magnitude, which is why the weights also have a similar order of magnitude.

Before training and evaluating these networks, each sample was normalized. No other pre-processing steps were applied. We decided not to threshold the images before training, because during experimentation a SCNN performed considerably worse when trained on thresholded images. The SCNN achieved an average accuracy of 86.2 per cent without thresholding and an average accuracy of 82.05 per cent with thresholding over twenty training runs.

Table 5. Weights used to balance loss components in radio galaxy networks.

Network	Main weight	Auxiliary weight
Even MhCNN	0.5	0.5
Main MhCNN	0.75	0.25
Auxiliary MhCNN	0.25	0.75
Even MCNN	0.5	0.5
Main MCNN	0.75	0.25
Auxiliary MCNN	0.25	0.75

5 RESULTS

This section will present and discuss the results from our experiments. The results of our rotational standardization algorithm will be presented in Section 5.1. This will be followed by a discussion of the performance of the guided CNNs on the XOR and FRGMRC data set in Section 5.2 and Section 5.3, respectively.

5.1 Rotational standardization results

Recall that we trained three SCNNs to evaluate our rotational standardization algorithm. One was trained on a data set where no rotational processing was performed, one was trained on a data set that was augmented with rotated versions of the samples and one was trained on a data set where our rotational standardization algorithm was applied as a pre-processing step. The average macro F1, loss, and training time of each of these networks can be seen in Table 6.

This table clearly indicates that approaches that address rotational variations lead to far superior classification results in comparison with approaches that ignore these variations. Such an increase in classification performance validates the claims that rotational variations negatively affect the performance of classifiers.

Upon further investigation of the confusion matrices shown in Fig. 7 and the per-class metrics shown in Table 7, we were also able to establish that these improvements in classification results occurred in all four classes. However, it became clear that addressing rotational variations had a considerably larger impact on the classification results of bent and FRII galaxies. It was observed that CNNs were able to achieve good classification performance on compact and FRI galaxies, even when rotational variations were not addressed. This indicates that CNNs are less sensitive to rotations in these galaxies than rotations in bent and FRII galaxies. We hypothesize that this is due to the simpler, compact nature of the FRI and compact galaxies.

Furthermore, it can be determined from the results in Table 6 and Table 7 that rotational augmentation also leads to better classification performance than rotational standardization. That being said, Table 7 indicates that rotational standardization results in a slightly better recall for bent and compact galaxies, which indicates that CNNs that were trained and evaluated on standardized galaxies are marginally better at identifying these classes.

Upon closer inspection of the samples on which classification errors are made, we found that there was an overlap between the models using augmentation and standardization. However, there were also a number of samples on which only standardization or augmentation led to errors. This could indicate that standardization and augmentation are relying on different features or have different vulnerabilities. Unfortunately, we could not identify any clear similarities between samples on which each technique tended to make mistakes, nor could we establish any clear differences between samples on which only standardization models would make a mistake and samples on which only augmentation models would make a mistake.

We suspect that the reason why augmentation generally leads to a better F1 score than rotational standardization is due to the sensitivity of rotational standardization to the presence of additional radio sources and thresholding artefacts in the background. To clarify what this means, we have included examples in Fig. 8. These sources and artefacts lead to changes in the principal components, which in turn leads to incorrect derotations.

To further investigate the impact of these incorrect derotations, we took a look at their prevalence in our data set. What we found is that 32 samples were incorrectly derotated due to additional radio sources and 6 samples were incorrectly rotated due to thresholding artefacts. Respectively, this corresponds to approximately 3.6 per cent and 0.6 per cent of the total data set. When looking at the class distribution of these incorrect derotations, as shown in Table 8, we can see that these problematic samples only affect the bent, FRI, and FRII classes and that they form a particularly large percentage of the FRI and bent samples. The incorrectly derotated samples might not always lead directly to classification errors, but they will have an indirect effect on the models' ability to classify samples from that class. The models will struggle to extract useful information from these problematic samples, because the information is at a different orientation, which will lead to a loss of training information in an already small data set. It is also entirely possible that these samples will lead to confusion within the models, seeing as they will differ considerably from the other samples that have been standardized and thus increase the intra-class variance. Our suspicions seem to be confirmed by the correlation between the large portion of problematic FRI samples and the large difference in FRI classification performance between data augmentation and rotational standardization that we can observe in Table 7 and Fig. 7. The difference in classification performance for the FRII galaxies is also considerably smaller, which corresponds with the smaller percentage of samples that were incorrectly derotated. However, we should note that this correlation between the difference in classification performance and the percentage of incorrectly derotated samples does not hold for bent galaxies. We suspect that this is due to the fact that bent galaxies have no clear correct orientation, which means that the 'incorrect' derotations will only lead to a marginal increase in intra-class variance.

We hypothesize that if one were to fine-tune the thresholding algorithm to remove these background radio sources, rotational standardization would be able to compete with rotational augmentation in terms of classification performance.

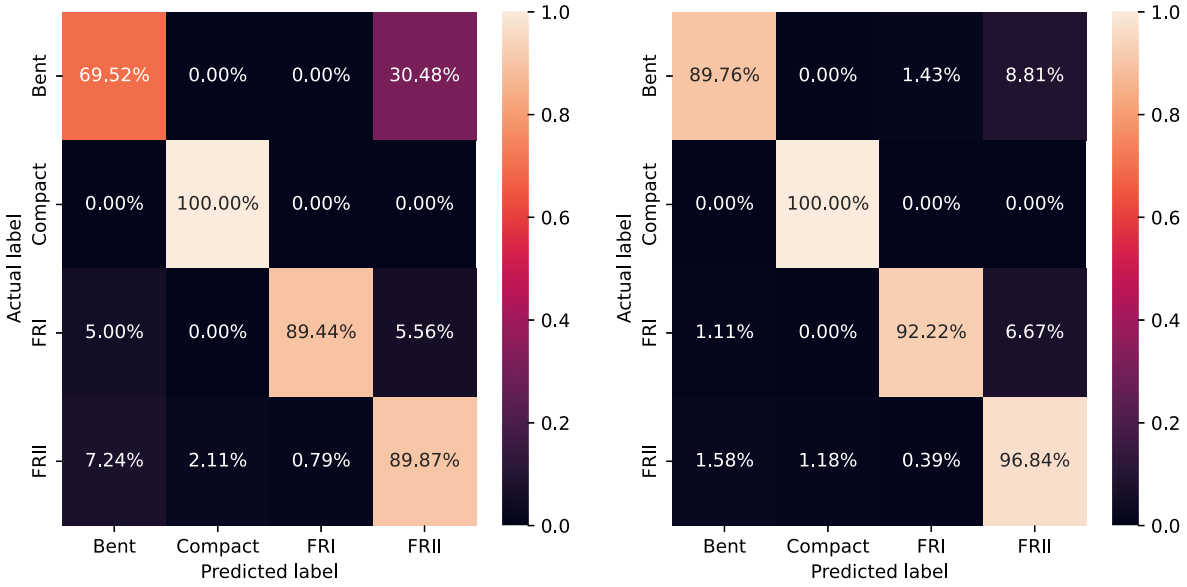
That being said, the true advantage of rotational standardization lies in the amount of time required to train the networks. The average training time for CNNs that were trained on augmented data is almost a factor of five times longer than the times required to train CNNs on samples with a standardized orientation. The training times for augmented data will become even longer if the rotational intervals are reduced. This confirms our hypothesis that data augmentation will be considerably slower due to the consideration of data set size.

What is interesting to note, is that on average, the CNNs also trained faster on the samples with a standardized orientation than the samples where no rotational pre-processing was applied. This result could be due to the fact that conventional CNNs have to learn what a feature looks like at various orientations, as was explained by Scaife & Porter (2021). Thus, if rotational variations are reduced or removed, it is highly likely that less time will be spent on learning these duplicate feature filters, which in turn means that the CNN will be able to learn a variety of informative features in less time.

This reduction in training time is likely to become more prominent as the complexity of the neural networks increases or if larger data sets are used that include more samples or more classes. Thus, in

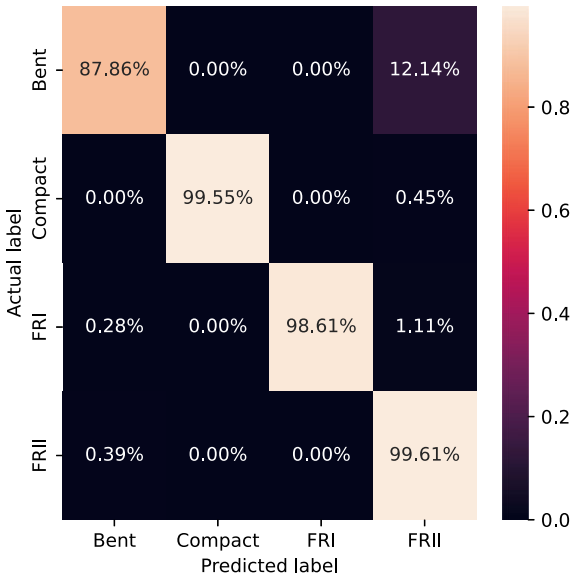
Table 6. Average performance of SCNNs with various rotational pre-processing strategies.

Pre-processing	Average macro F1	Average loss	Average training time (s)
None	0.876	0.34	116
Rotation standardization	0.95	0.18	98
Augmentation	0.971	0.12	476



(a) No preprocessing

(b) Rotational standardisation

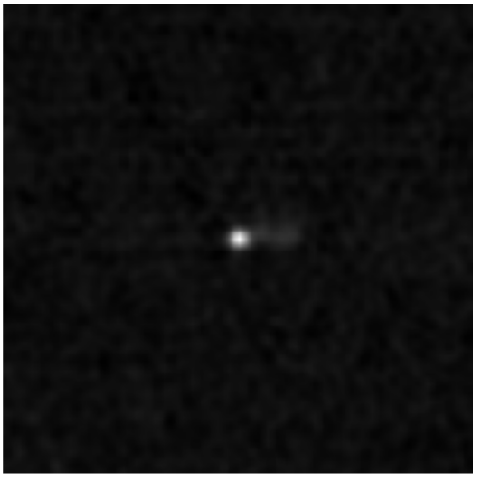


(c) Augmentation

Figure 7. Average confusion matrices of SCNNs with various rotational pre-processing strategies.

Table 7. Average per-class performance of SCNNs with various rotational pre-processing strategies.

	Bent			Compact			FRI			FRII		
Pre-processing	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
None	0.8	0.695	0.743	0.937	1.0	0.966	0.983	0.894	0.936	0.823	0.899	0.858
Standardization	0.961	0.898	0.928	0.963	1.0	0.981	0.975	0.922	0.947	0.924	0.968	0.945
Augmentation	0.99	0.879	0.93	1.0	0.995	0.998	1.0	0.986	0.993	0.932	0.996	0.963



(a) Example of a FRI sample where thresholding produced a background artefact.

(b) The thresholding mask that corresponds to Figure 8a. Notice the artefact in the top right corner.

(c) Example of an additional radio source in the background of a FRI sample.

Figure 8. Samples where rotational standardization failed due to background elements.

Table 8. Distribution of incorrect derotations.

Class	Total incorrect derotations	Percentage of class
Bent	18	8.5 per cent
Compact	0	0.0 per cent
FRI	17	9.3 per cent
FRII	12	3.2 per cent

time-sensitive scenarios it might be beneficial to rather use rotational standardization instead of rotational augmentation.

5.2 Performance of guided neural networks on XOR

The reader might recall that a variant of the merged architecture was evaluated on the XOR data set first to determine whether it was feasible to guide the training of neural networks. For more information regarding the architecture, the reader is referred to Appendix A.

In Fig. 9, one can see violin plots representing the number of training epochs it took for the XOR neural networks to reach a loss smaller than 0.01. These violin plots have a box plot at their

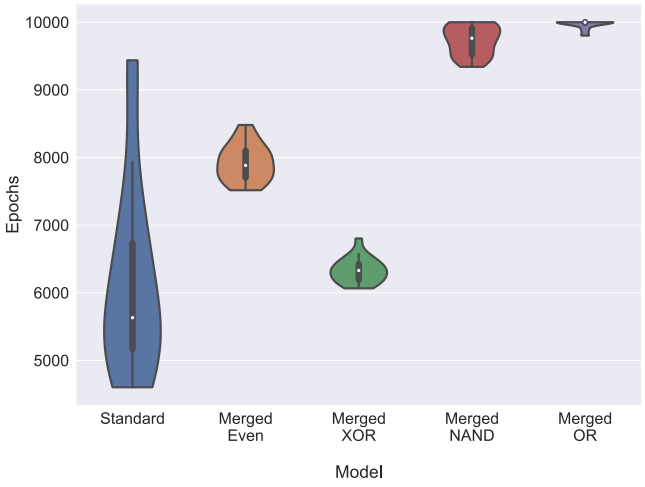


Figure 9. Epochs taken to train the XOR networks. The given labels indicate the network architecture, as well as the weight distribution that was used for each merged network. The reader is referred to Appendix A for clarity on the meaning of the labels.

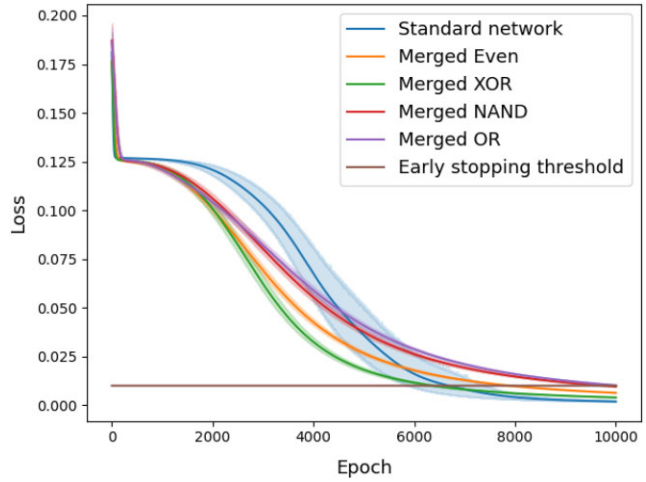


Figure 10. Loss curves of XOR networks during training.

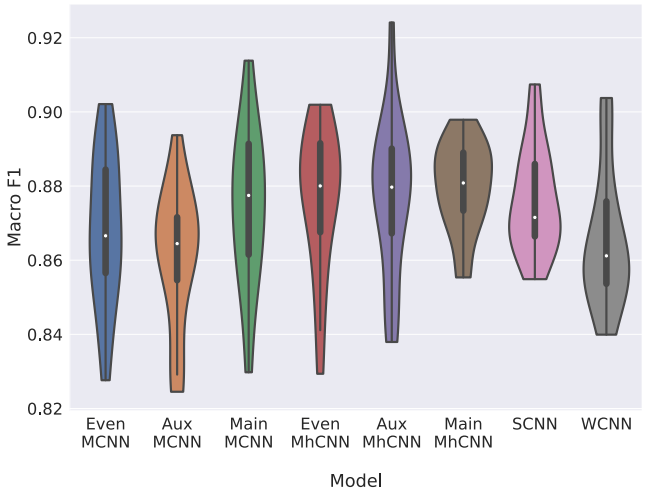


Figure 11. Violin plots of the macro F1 scores from Section 4.1.7 that were observed when evaluating models on the test data set. The reader is referred back to Table 5 for an overview of the weights used in the MhCNNs and MCNNs.

center. The thin lines in the box plot indicate the values smaller than the 25th quantile and larger than the 75th quantile, the thick line indicates the values that are between the 25th and 75th quantile and the white dot indicates the median value. Surrounding this box plot is a density estimation that is based on the observed values. This assists in visualizing the expected distribution of each of the performance metrics. The wider the density estimation is, the more likely we are to observe the values in that region. For this paper, the density estimations were restricted to the range of values observed in our experiments, thus no estimation is done for values outside of the range of observed values. Estimations occurring outside of the box plot is due to the fact that the box plots exclude values that are deemed to be outliers, but the density estimation does not.

These violin plots indicate a large difference between the training process of the guided and unguided XOR neural networks. The training process of the guided networks is clearly much stabler. We can see that all four of the guided networks have a very compact violin plot, whilst the unguided network has a violin plot that spans quite a large range of epochs. This is to be expected, seeing as the guided networks will look for similar patterns during each training run, whilst the unguided network is more likely to explore various different patterns in each run.

We can also observe that the merged network with a heavy XOR weight is clearly the guided network that trains the quickest. This makes sense, seeing as the other guided networks might spend too much time on fine tuning the results of the NAND and OR neurons. Another observation that can be made is that the standard XOR network will train quicker than the guided networks in most runs. However, it can also train considerably longer than some of the guided networks, such as the merged network with a heavy XOR weight.

To investigate why these results were observed, the loss curves of each of the neural networks were plotted after training. These curves can be seen in Fig. 10. The reader should note that these loss curves do not correspond to the training runs for the violin plots, seeing as early stopping had to be disabled to record the entire loss curve over 10 000 epochs. In these loss curves the solid line represents the average loss at each epoch and the shaded region indicates the 95 per cent confidence interval. The horizontal line at the bottom of the figure indicates the early stopping threshold that was used.

These loss curves also indicate that the training process of the merged neural networks is much stabler than that of the standard neural network, seeing as their confidence intervals are considerably more compact. Furthermore, these loss curves indicate that the merged networks perform better than the violin plots suggested. These loss curves suggest that all of the merged neural networks would outperform the standard neural network if the threshold of early stopping was increased. It should also be noted that the merged neural networks will outperform the standard neural network if training was conducted for less epochs. For example, if training was only conducted for 3000 epochs, all of the merged networks would achieve a better loss than the standard network.

Merged neural networks seem to converge much quicker initially, but the convergence slows down in later epochs. This behaviour makes sense, seeing as the merged networks will immediately start learning to extract features and patterns that have been deemed to be informative, whilst the standard neural network will spend more time exploring the search space and is thus likely to take longer to find good features. The quick convergence of these merged networks could be very beneficial for time-sensitive applications.

5.3 Performance of guided CNNs on FRGMRC data

5.3.1 Algorithmically extracted features

To provide an accurate depiction of the performance of the CNNs on the FRGMRC data, we generated a few violin plots that present the observed values of the performance metrics for each run of each of the CNNs. We will focus on the violin plots of the macro F1 scores, the overfitting metric and the number of epochs needed to train the networks, seeing as they were the most informative when evaluating the performance of the CNNs. The plots are shown in Figs 11, 12, and 13, respectively.

When comparing the white dots that represent the median F1 score in Fig. 11 we can observe that all of the MhCNNs, as well as the MCNN with a heavier weight for the main cross-entropy loss managed to achieve a better macro F1 score when classifying each of the radio galaxies. However, these scores are only marginally better than that of the SCNN.

Shifting our focus to Fig. 12, the benefits of the guided networks become more apparent. We can immediately observe that overfitting

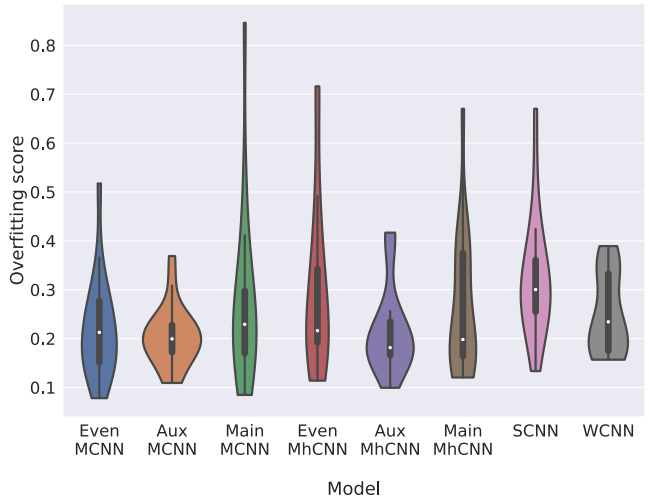


Figure 12. Violin plots of the overfitting scores from Section 4.1.8 that were observed after training the models.

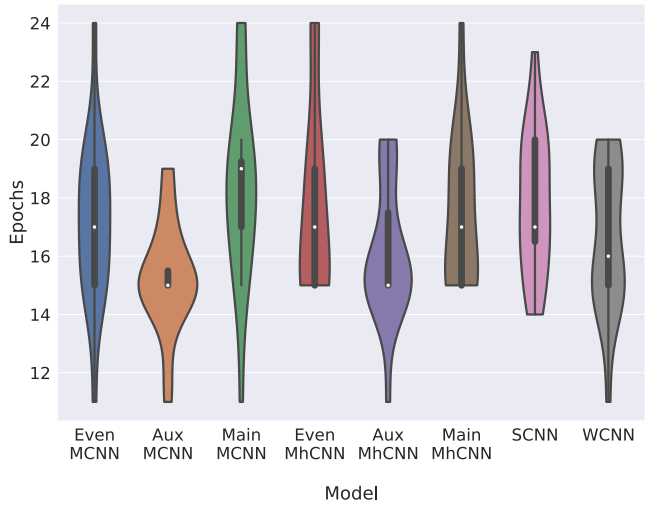


Figure 13. Violin plots of the number of epochs it took to train the CNNs.

is less likely to occur in the MCNNs and MhCNNs that use a larger weight for the auxiliary loss component. We can also observe a similar trend in the WCNN.

These networks all have something in common, they make use of higher quality features to guide the classification of the samples. To understand why this is true for the merged and multiheaded networks, the reader is reminded that these networks made use of a considerably larger weight for the loss that corresponds to the features that are extracted from the samples. During training, the feature extraction will thus be optimized much quicker than the classifications, which means that the network will have access to high quality features much sooner than the other guided networks. The wide network will also have high quality features, because they were extracted by a pretrained ACNN.

It is very likely that it is these high quality features that makes the networks more robust to overfitting. We suspect that the networks are more likely to learn to use these features for classification if they are trustworthy at an early stage of training. If the network makes use of these features, they will not need to find as many other features

Table 9. Average training times of CNNs on galaxy data set.

Network	Average training time (s)
SCNN	116
WCNN	99
Even MhCNN	144
Aux MhCNN	127
Main MhCNN	137
Even MCNN	119
Aux MCNN	104
Main MCNN	119

in the training data, which reduces the risk of overfitting on noise or uninformative features.

From Fig. 13 it also becomes apparent that the auxiliary-weighted MCNNs and MhCNNs, as well as the WCNNs generally require less epochs to train. This is likely due to similar reasons as the reduction in overfitting. Due to the presence of high quality feature vectors early in the training process, we suspect that these networks don't need to spend as much time searching for features that assist in classifying the given samples.

The reader should note that, as shown by Table 9, some of the networks don't necessarily train quicker if they train in less epochs.[5] The MCNNs train in less epochs and less time than the SCNNs, but the MhCNNs have a much more complex structure with more parameters to tune than the other CNNs, which means each epoch takes longer to complete. However, the fact that they train in less epochs still indicates that by guiding the networks to look for specific features, we can help the networks to converge in less training steps. Furthermore, if the code is optimized to train the two heads of the MhCNNs in parallel, it is likely that one will also be able to train the MhCNNs in less time than the SCNNs.

What is also interesting to note is that the box plots of the auxiliary-weighted MCNNs and MhCNNs is much more compact than that of the standard network, which is similar to the behaviour that we observed for the XOR data set. This is especially true for the box plot of the MCNNs. The compact box plots could be another indication that the networks spend less time searching for features and more time optimizing the extraction and use of the features that we proposed. This leads to less variability in the training process and thus the number of epochs required for training becomes more consistent.

We suspect that the MCNNs have a more compact box plot than the MhCNNs, because the selected features play a more direct role in the classification of the samples. Thus, if the selected features are truly informative and the network is able to accurately extract them, there is no need for the network to look for other features. Due to the indirect role that the selected features play in the MhCNNs, it might be necessary for the MhCNNs to spend more time to explore the search space. Seeing as the convolutional layers have been trained to also extract the selected features, the chances are good that the MhCNN will still find these features and make use of them for classification, but it will take more time. The networks are also likely to find other features in the process, which will lead to an increase in the range of epochs that are required for training.

[5]The training time of the WCNNs does not include the time taken to train the ACNNs. This was done to show what the training time would be if these features were already available.

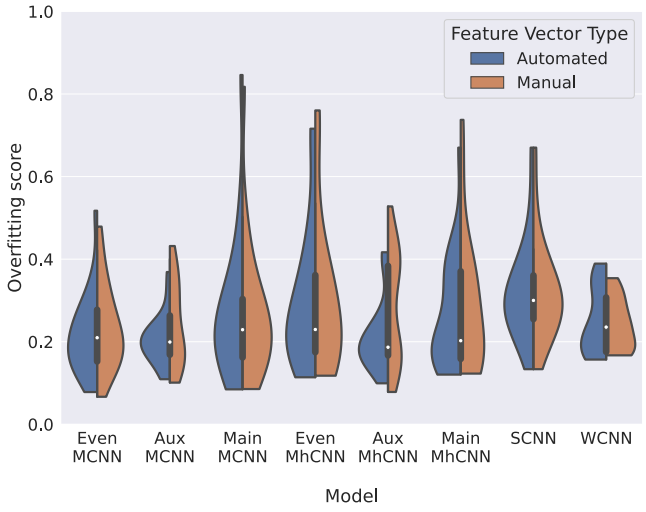


Figure 14. Comparison of overfitting when using manual and algorithmically extracted features.

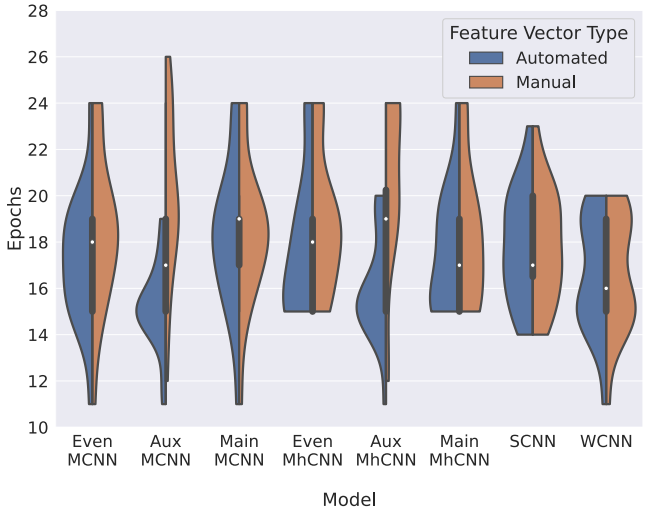


Figure 15. Comparison of epochs taken when using manual and algorithmically extracted features.

5.3.2 Manually extracted features

As it has already been mentioned, we also have a set of feature labels where some of the features were extracted manually from each sample. We evaluated the guided CNNs on this set of features as well, because we wanted to determine whether it would make a significant difference in the networks' performance. In this section we will mainly focus on the violin plots shown in Figs 14 and 15.

From these figures, it becomes apparent that the auxiliary-weighted MhCNNs and MCNNs, as well as the WCNNs, still tend to overfit less. However, the guided networks also seem to perform worse when we make use of the manually extracted features.

This behaviour seems counter-intuitive, but it makes sense after considering that the manually extracted features are not necessarily accurate, especially the core count. The feature that indicates whether the body of the galaxy is curved is likely to be accurate, because it is fairly straightforward to determine if this is true or not when looking at the images. However, there were no clear rules for what should be considered to be a core, which means that the core counts were based on the intuition of the person who labelled them. This could lead to inaccurate core counts, which in turn is likely to lead confusion within the neural networks when they attempt to learn how to extract this feature.

To investigate whether our suspicions were true, we calculated the average MSE of the core counts predicted by the auxiliary-weighted MCNNs after they were trained with the manually and automatically extracted core counts. We found that the MSE was 0.029 when using the automatically extracted core counts and that the MSE was 0.113 when using the manually extracted core counts. This confirms our suspicion that the manually extracted core counts lead to confusion in our guided networks which in turn would negatively affect the overall performance of these networks.

Thus, we suggest that algorithmically extracted features are used, unless one is certain that the features can be accurately extracted manually, using only information that is available in the training data.

6 CONCLUSION

The two goals of this paper were to investigate the benefits of rotational standardization as well as the benefits of guiding neural networks to look for specific features.

In Section 5.1, it was shown that rotational standardization definitely leads to better classification performance than if no attempt is made to address rotational variations. Rotational standardization also managed to achieve this improvement in much less time than rotational augmentation can. That being said, augmentation might still be the better choice in situations where time and computational power is not a problem, seeing as it can achieve better results.

It was also identified that rotational standardization is susceptible to additional radio sources in the images, as well as thresholding artefacts. Thus, if steps can be taken to address these problems in the images, we are of the opinion that rotational standardization will be able to rival rotational augmentation, both in training time and classification accuracy.

When evaluating the performance of guided neural networks on the XOR data set in Section 5.2, we determined that the training process could become stabler and converge much quicker if we guided the neural networks to look for informative features. These results are promising and clearly indicate that the training process can be affected positively by guiding the networks.

After evaluating the guided architectures on the FRGMRC data in Section 5.3.1, we confirmed that it was vitally important that accurate and informative features were used to guide the training process. Furthermore, we confirmed that if these features were accurate and informative, it was possible to reduce overfitting in the networks. It was also possible to reduce the average time needed for the training of the WCNNs as well as the MCNNs with a larger weight for the auxiliary loss. We are of the opinion that these guided architectures will be even more effective if the feature extraction process is fine-tuned.

These results indicate that such guided architectures could be beneficial in time-sensitive applications, in applications where a stable training process is essential, as well as in applications with multiple goals. Specifically, the multiheaded and merged neural networks could be very useful in applications where one wants to identify specific features that could also be informative when classifying samples. As was seen in this paper, these guided neural networks can be trained to achieve two goals in less time than it might take to achieve one, as long as these goals can contribute to one another.

Future work could investigate ways to make the rotational standardization approach more robust to background information in the images. It could also consist of comparing rotational standardization to group-equivariant CNNs or constructing new guided architectures.

ACKNOWLEDGEMENTS

MV acknowledges financial support from the Inter-University Institute for Data Intensive Astronomy (IDIA), a partnership of the University of Cape Town, the University of Pretoria, the University of the Western Cape and the South African Radio Astronomy Observatory, and from the South African Department of Science and Innovation's National Research Foundation under the ISARP RADIOSKY2020 Joint Research Scheme (DSI-NRF Grant Number 113121) and the CSUR HIPPO Project (DSI-NRF Grant Number 121291).

MP acknowledges financial support from the Inter-University IDIA and from the Center of Radio Cosmology (CRC, NRF Grant Number 84156) at the University of the Western Cape.

7 DATA AVAILABILITY

The FRGMRC and the supporting FIRST fits cutouts used for our work are publicly available at https://doi.org/10.5281/zenodo.7645530.

REFERENCES

Abadi M. et al., 2015, TensorFlow: Large-Scale Mach. Learn. on Heterog. Syst. https://www.tensorflow.org/
Alger M. J. et al., 2018, MNRAS, 478, 5547
Alhassan W., Taylor A. R., Vaccari M., 2018, MNRAS, 480, 2085
Aniyan A. K., Thorat K., 2017, ApJS, 230, 20
Bader S., Hölldobler S., Marques N., 2008, in Procs. 4th Intl. Workshop on Neural-Symbolic Learn. and Reason. CEUR Workshop Proceedings, Aachen, Germany
Baldi R. D., Capetti A., Giovannini G., 2015, A&A, 576, 38
Baldi R. D., Capetti A., Massaro F., 2018, A&A, 609, 1
Banfield J. K. et al., 2015, MNRAS, 453, 2326
Beardsley A. P. et al., 2019, PASA, 36
Becker R. H., White R. L., Helfand D. J., 1995, ApJ, 450, L559
Becker B., Vaccari M., Prescott M., Grobler T., 2021, 1828 MNRAS, 503
Bowles M., Scaife A. M. M., Porter F., Tang H., Bastien D. J., 2021, MNRAS, 501, 4579
Capetti A., Massaro F., Baldi R. D., 2017a, A&A, 598, 49
Capetti A., Massaro F., Baldi R. D., 2017b, A&A, 601, 81
Chen X.-W., Lin X., 2014, IEEE Access, 2, 514
Chen J., Liu Y., 2021, Expert Syst. with Appl., 168, 114316
Cheng H.-T. et al., 2016, in Proc. 1st Workshop on Deep Learn. for Recomm. Syst., DLRS. ACM, New York, NY, p. 7
Daw A., Karpatne A., Watkins W., Read J., Kumar V., 2021, preprint (arXiv:1710.11431)
De La Calleja J., Fuentes O., 2004, MNRAS, 349, 87
Dewdney P. E., Hall P. J., Schilizzi R. T., Lazio T. J. L. W., 2009, Proc. of the IEEE, 97, 1482
Dewdney P., Turner W., Millenaar R., McCool R., Lazio J., Cornwell T., 2013, Technical Report, SKA1 System Baseline Design. SKA Program Development Office, Macclesfield, UK
Dong G., Ma Y., Basu A., 2021, IEEE Access, 9, 28272
Fanaroff B. L., Riley J. M., 1974, MNRAS, 167, 31
Fu L., 1995, Knowledge-Based Syst., 8, 299
Gendre M. A., Wall J. V., 2008, MNRAS, 390, 819
Gendre M. A., Best P. N., Wall J. V., 2010, MNRAS, 404, 1719
Girshick R., 2015, in Proc. of the IEEE Int. Conf. on Comput. Vision. IEEE, New York, USA
Goethe J., Bronzino J., 1995, IEEE Eng. in Med. and Biol. Magazine, 14, 776
Hardcastle M. J., Croston J. H., 2020, New A Rev., 88, 101539
Harris C. R. et al., 2020, Nature, 585, 357
Heaton J., 2016, in SoutheastCon. IEEE, Virginia, VA, p. 1
Helfand D. J., White R. L., Becker R. H., 2015, ApJ, 801, L26
Hocking A., Geach J. E., Sun Y., Davey N., 2018, MNRAS, 473, 1108
Hotelling H., 1933, J. Educ. Psychol., 24, 417
Huynh T. Q., Reggia J. A., 2011, IEEE Trans. on Neural Netw., 22, 264
Janiesch C., Zschech P., Heinrich K., 2021, Electron. Mark., 31, 685
Johnston S. et al., 2008, Exp. Astron., 22, 151
Jonas J. L., 2009, Proc. of the IEEE, 97, 1522
Karpatne A. et al., 2017, IEEE Trans. on Knowl. and Data Eng.
Klema V., Laub A., 1980, IEEE Trans. on Autom. Control, 25, 164
Kolman E., Margaliot M., 2008, IEEE Trans. on Neural Netw., 19, 1389
Lawrence S., Giles C. L., Tsoi A. C., 1998, Technical Report, What Size Neural Netw. Gives Optim. Gen.? Converg. Prop. of Backpropagation, Univ. of Maryland, Inst. for Adv. Comput. Stud., https://drum.lib.umd.edu/handle/1903/809
LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Li H., Ng J. Y.-H., Natsev P., 2019, Technical Report, EnsembleNet: End-to-End Optimization of Multi-headed Models. Google
Lukic V., Brüggen M., Banfield J. K., Wong O. I., Rudnick L., Norris R. P., Simmons B., 2018, MNRAS, 476, 246
Ma Z. et al., 2019, ApJS, 240, 34
McGlynn T., Scollick K., White N., 1998, in McLean B. J., Golombek D. A., Hayes J. J. E., Payne H. E., eds, International Astronomical Union Symposia Vol. 179, New Horizons from Multi-Wavelength Sky Surveys. Kluwer Academic Publishers, Norwell, USA, p. 465
Missaglia V., Massaro F., Capetti A., Paolillo M., Kraft R. P., Baldi R. D., Paggi A., 2019, A&A, 626, 8
Mitchell T. M., Keller R. M., Kedar-Cabelli S. T., 1986, Machine Learn., 1, 47
Mulaudzi R., Ajoodha R., 2021, in Chakrabarti S., Nath R., Banerji P. K., Datta S., Poddar S., Gangopadhyaya M., eds, Proc. of the Int. Conf. on Interdiscip. Res. in Technol. and Manag., Chapt. 66. CRC Press, London
Najafabadi M. M., Villanustre F., Khoshgoftaar T. M., Seliya N., Wald R., Muharemagic E., 2015, J. Big Data, 2, 1
Nargesian F., Samulowitz H., Khurana U., Khalil E., Turaga D., 2017, in Sierra C., ed., Proc. of the 26th Int. Joint Conf. on Artif. Intell. Int. Joint Conf. on Artif. Intell., Melbourne, Australia, p. 2529
Olden J. D., Jackson D. A., 2002, Ecol. Model., 154, 135
Owen F., Rudnick L., 1976, ApJ, 205, L1
Padovani P. et al., 2017, A&A Rev., 25, 2
Proctor D. D., 2003, J. Electron. Imaging, 12, 398
Proctor D. D., 2006, ApJS, 165, 95
Proctor D. D., 2011, ApJS, 194, 31
Rahman S., Hazim O., 1996, Electr. Power Syst. Res., 39, 161
Rao C. R., 1964, Sankhyä: The Indian J. of Stat., 26, 329
Rudnick L., Owen F., 1976, AJ, 82, 1
Sadeghi M., Javaherian M., Miraghaei H., 2021, AJ, 161, 94
Scaife A. M. M., Porter F., 2021, MNRAS, 503, 2369
Schank R. C., Collins G. C., Hunter L. E., 1986, Behav. and Brain Sci., 9, 639
Seidel S., Schimmler S., Borghoff U. M., 2019, in Arai K., Kapoor S., Bhatia R., eds, Intell. Syst. and Appl, Adv. in Intell. Syst. and Comput. Springer Int. Publ., Cham, p. 616
Stacy J. G., Vestrand W. T., 2003, in Meyers R. A., ed., Encycl. of Phys. Sci. and Technol.. Academic Press, New York, p. 397
Sun X., Wang H.-F., Wang B., 2008, J. Comput. Sci. Technol., 23, 602
Tang H., Scaife A. M. M., Wong O. I., Shabala S. S., 2022, MNRAS, 510, 4504
Towell G. G., Shavlik J. W., 1994, Artif. Intell., 70, 119
van Haarlem M. P. et al., 2013, A&A, 556, 2
Woods W., Chen J., Teuscher C., 2019, Nat. Mach. Intell., 1, 508
Wu C. et al., 2018, MNRAS, 482, 1211
Yu H.-F. et al., 2010, in KDD 2010. ACM, New York, USA

APPENDIX A: XOR NEURAL NETWORK ARCHITECTURES

In this section, we present the architectures used for the standard unguided neural networks and the merged guided neural networks that were applied to the XOR data set. Section A1 will present the standard network and Section A2 will present the merged network.

A1 Standard XOR neural network

It would not make sense to apply CNNs to the XOR data set. These networks are unnecessarily complex for simple tabular data.

Thus a simple neural network was created for the XOR operator. This network only has an input layer with two input neurons, a hidden layer with three hidden neurons and an output layer with one output neuron. Each neuron makes use of the sigmoid activation function and the MSE variant that was discussed in Section 4.1.3 was used for the loss function. A representation of the architecture can be seen in Fig. A1.

This standard neural network was used as a baseline with which to compare the performance of the guided neural network.

A2 Merged XOR neural network

A representation of the merged network for the XOR operator can be seen in Fig. A2. In this figure b represents bias terms, w represents the weight terms, z represents neuron inputs before applying the activation function, and a represents the inputs after applying the activation function.

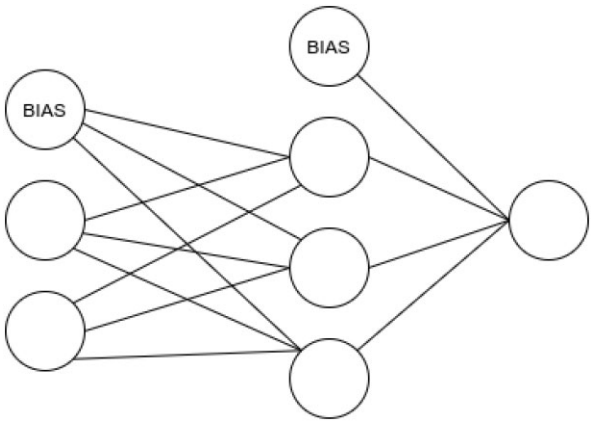


Figure A1. Standard XOR neural network.

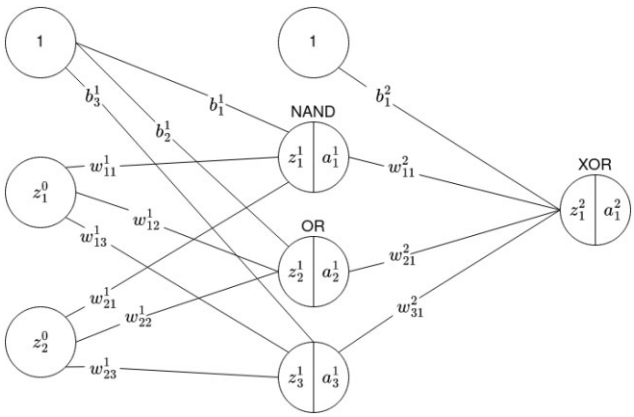


Figure A2. Merged XOR neural network.

Table A1. Weights used to balance loss components in XOR networks.

Network	XOR weight (l_1)	NAND weight (l_2)	OR weight (l_3)
Even merged	0.34	0.33	0.33
XOR merged	0.5	0.25	0.25
NAND merged	0.25	0.5	0.25
OR merged	0.25	0.25	0.5

As can be seen in this figure, two of the hidden neurons are guided to represent the NAND and OR binary operators, which are then fed back into the output neuron that represent the XOR operator. To understand why this was done, consider the fact that the XOR operator can be rewritten in terms of the NAND and OR operators: x_1 XOR $x_2 = (x_1$ NAND $x_2)$ AND $(x_1$ OR $x_2)$.

To ensure that gradient descent does not only optimize the XOR function, a loss function was used that aggregates the error of the XOR, NAND, and OR outputs. This loss function simply calculates the weighted sum of each output's MSE. This weighted sum can be seen in equation (A1).

$$C = \frac{l_1}{2}(y_1 - a_1^2)^2 + \frac{l_2}{2}(y_2 - a_1^1)^2 + \frac{l_3}{2}(y_3 - a_2^1)^2 \tag{A1}$$

The weights that were used for this loss function can be seen in Table A1.

A3 Generalizability of guided architectures

When guided architectures encounter sources with different feature variations or new features, there might be a concern that they will fail to generalize and will perform poorly. In this section, we will shortly discuss this concern and our suggested approach to address it.

The first scenario that one should consider is the presence of new features in our samples, whether in training or in inference. If we encounter new informative features during training, our guided architectures should be able to learn what these features are and how to extract them by using the free neurons within the network, just as a standard CNN would. However, if we only encounter these new features during inference, it is unlikely that our guided architectures will be able to sufficiently utilize them, but this would also hold true for standard CNNs. CNNs in general cannot utilize features during inference that they have not observed during training.

Another scenario to consider would be rare or unusual variations of the galaxies' features. We are of the opinion that the guided architectures will not be affected any worse by these variations than standard architectures, seeing as all of these architectures are unlikely to be able to learn much from these variations if they are rare or uncommon in our data sets. If, however, we have a new data set where these feature variations are no longer rare or unusual, our guided architectures will be able to learn from them by making use of the free neurons. That being said, we do acknowledge that the guided architectures do not attribute as much computational power to finding new features as the standard architectures do. Thus, if the change in feature variations is so severe that the features we selected are no longer informative, it is possible that our guided architectures will perform worse than standard CNNs. We should consider, however, whether we expect such severe changes in our feature space and whether these changes would not represent an entirely new class of radio galaxies instead of an unusual instance of a known class.

To address the concern that we might encounter such unexpected or rare feature variations in future large-scale surveys, we suggest the use of an anomaly detection, or more specifically, a novelty detection system that is trained alongside the classification networks. Such a system could be used in parallel with the classification networks during inference to detect when we encounter rare or new feature variations. This system could assist us in finding samples that could prove to be problematic for our guided architectures. If these samples are rare, we can simply extract them and make use of a human expert to determine what they are and how they affected our classification network. However, if we start to encounter such samples on a frequent basis, it could be an indication of concept drift or that we might have found a new class, in which case we would know that it is a good time to re-evaluate and retrain our models.

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