# Synthetic Training Data For Morphological Classification Of Radio Galaxies With Deep Convolutional Neural Networks

J. E. Morrison,[1]★
[1] *School of Physics and Astronomy, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK*
*Student ID: 1644962*

3 April 2020

**ABSTRACT**
The high resolution and sensitivity of next generation radio surveys will enable the detection of a vast number of radio sources. Deep convolutional neural networks have been proven as an effective method of automating image classification of radio objects. The learning ability of state of the art neural networks designed for morphological classification of radio galaxies is limited by the availability of labelled training data. Labelled data can be especially hard to produce in astronomy due to the finite amount of available image data. In this study we look at classification of radio galaxies into compact, bent-tailed, FRI and FRII morphologies. We present methods for generating synthetic training data using generative adversarial networks (GANs) and a simple geometry based simulation. Augmenting the training set with synthetic data generated by GANs, as well as traditional augmentation techniques of flipped and rotated images, improved the overall classification accuracy of our model from 91% to 92%. Inclusion of synthetic data also reduced the overfitting present in the model by allowing the model to better generalise.

## 1 INTRODUCTION

Extragalactic radio sources, namely radio galaxies, are some of the most powerful and diverse objects in the Universe. Radio galaxies, also known as radio-loud active galactic nuclei (AGN), are powered by the accretion of matter onto a supermassive black hole (Fabian 1999). They can possess jets of highly collimated ionised matter, observed due to the emission of synchrotron radiation, which can extend to great distances away from their host galaxy (Padovani 2017). The interaction of these jets with the surrounding environment (intergalactic medium) and differences in the properties of the AGN and its accretion modes, amongst other factors, leads to a large range of complex radio galaxy morphologies. Classifying and measuring the properties of AGN populations is crucial for understanding their growth and evolution, and for the field of galaxy evolution as a whole. AGN can also be used tracers of the cosmic environment (Croton et al. 2006), dark matter density field, and high red-shift galaxy clusters (Blanton et al. 2000).

Next generation radio telescopes will be able to survey the entire sky at high speed with unprecedented resolution and sensitivity. This will not only generate huge volumes of data, but also produce source catalogs with orders of magnitude more sources. Current and upcoming radio surveys include; LoTSS, EMU and VLASS (Shimwell et al. 2017; Norris et al. 2011; Lacy et al. 2020), leading up to surveys planned with the Square Kilometre Array (SKA, Braun et al. (2015)). The SKA is expected to discover up to as many as 500 million sources. A comparison of the sizes, resolution and sensitivity of a few current and upcoming radio surveys can be seen in table 1. With such huge numbers of sources manual inspection of images will become impractical, hence automated methods of classification are necessary.

When it comes to classifying radio galaxies there are several possible approaches; properties of emission lines, spectral index, properties of the host galaxy, etc. In this study we consider radio galaxy classification in the image domain, so the classification is purely morphological. This also means there is no dependence on ancillary data. The morphology of a radio galaxy is of importance since it correlates with the galaxy's physical properties, such as the total power, surrounding environment, formation and evolution (Gendre et al. 2013).

Radio sources can be compact (point or unresolved sources), or extended (resolved sources). Some classification schemes aim to classify an object by counting the number of unresolved and

| Survey | Date | Size | Resolution (arcsec) | Sensitivity (mJy/beam) | Sources (million) |
|---|---|---|---|---|---|
| NVSS | 2011 | 3/4 | 45" | 2.5 | 1.8 |
| FIRST | 2014 | 1/4 | 5" | 1 | 0.9 |
| LoTSS | 2020 | 1/2 | 5" | 0.1 | 10 |
| VLASS | 2022 | 3/4 | 2.5" | 0.07 | 5 |
| EMU | 2022 | 3/4 | 10" | 0.01 | 70 |
| SKA | 2030 | 3/4 | 0.1" | 0.001 | 500 |

**Table 1.** Comparison of recent and upcoming radio surveys. All table values are approximate, particularly dates which are given as approximate completion dates, but many surveys have staggered data releases with varying resolution and sensitivity. Survey sizes are given as the fraction of the entire sky.

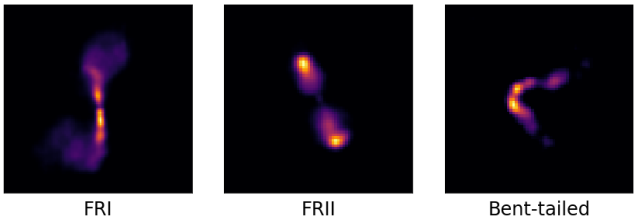


**Figure 1.** Example FIRST image cutouts, showing typical FRI, FRII and bent-tailed radio galaxy morphologies.

resolved components belonging to the object (Lukic et al. 2018; Wu et al. 2019). Another method is to determine the classification based on the distribution of high and low surface brightness regions in the extended components (jets / tails) of the radio galaxy. This is the strategy adopted by the Fanaroff-Riley (FR) classification scheme (Fanaroff & Riley 1974); where radio galaxies are split into two groups, labelled as FRI or FRII. Membership to either group depends on the ratio, $R$, of the distance between the brightest points in each tail and the total extent of the source. $R < 0.5$ corresponds to an FRI, and $R > 0.5$ to an FRII. Typical features for an FRI include diffuse, plume-like jets with the brightest points in the lobes occurring close to the core, hence are often called edge-darkened. On the other hand, FRIIs typically exhibit bright hotspots near the end of lobes, so are often called edge-brightened. Examples of typical FRI and FRII type radio galaxies can be seen in figure 1.

Beyond the typical FRI and FRII morphologies, some extended sources can show more complex bent morphologies. Typically for AGN with jets, two twin jets are emitted from opposite sides of the galactic nucleus (Begelman et al. 1984). These jets can become bent due to ram pressure as they travel through the intergalactic medium (Garon et al. 2019). As such, radio galaxies can also be classified by the bending angle between the jets, for example wide-angled or narrow-angled tails (WATs and NATs). Often the lobes on either side of the central AGN have different appearances due to the different environments they are subject to. Other even more complex morphologies are possible, such as hybrid (a combination of FRI/II characteristics), X-shaped and ringlike (Proctor 2011). However these are largely ignored in this study since their rarity leads to a small sample size of available labelled data.

In this study we aim to classify radio galaxies into the four classifications of; compact, bent-tailed, FRI and FRII. This was chosen since there is the most available labelled data for these classifications, and to be able to compare our results to previous studies using the same classifications (Aniyan & Thorat 2017; Alhassan et al. 2018).

Until recent years, identifying radio object morphology has been done by visual inspection; and most recently through citizen science projects, such as Radio Galaxy Zoo (RGZ, Banfield et al. (2015)). Although these techniques have been adequate in the past, in the anticipation of the huge data volumes provided by next generation radio surveys it is clear that the limitations of manual inspection have been reached.

Recently deep learning approaches to automating classification have been proven to be a successful method; particularly in their ability to cope with complex source morphologies with several discrete components, where simple automated classification methods often fail (de la Calleja & Fuentes 2004). The first application of a CNN classifier for radio galaxy morphology was presented in Aniyan & Thorat (2017), since then several other CNN adaptations have also been implemented (Alhassan et al. 2018; Tang et al. 2019; Walmsley et al. 2020).

Some of the key limitations of deep learning classification approaches found in literature include; a low number of labelled training samples results in a small feature space, and the classification performance is highly sensitive to image pre-processing (Aniyan & Thorat 2017).

However, the main requirement in order for any deep learning approach to work, is the considerable amount of labelled data to train a model. To this end, manual inspection and citizen science will always play an important role. Efforts to reduce the amount of labelled data required when training a model include; data augmentation (discussed in section 4), self organising maps, and convolutional auto encoders (CAEs). Self organising maps aim to generate a set of classification prototypes directly from the raw data, with no labels (Polsterer et al. 2015). CAEs employ a combination of unsupervised and supervised learning methods, to optimise image feature extraction by first learning the best compact latent representation for the data (Ma et al. 2019).

Another technique to reduce labelled data requirements is generating and using synthetic data. In this paper, we investigate the applicability of generating synthetic training data, to reduce the reliance on large quantities of labelled training data, and boost the performance of classification of radio galaxy morphology.

Our neural network models are developed using the TensorFlow deep learning framework. The computations described in this paper were performed using the University of Birmingham's BlueBEAR HPC service, which provides a High Performance Computing service to the University's research community [1]. Full python code and descriptions of the computations performed in this study can be found on the project github [2].

The structure of this paper is as follows. Section 2 includes a brief background on neural networks; more specifically convolutional neural networks (CNNs), and their application in generative adversarial networks (GANs). Section 3 discusses source sample selection from available catalogs. Section 4 contains the image preprocessing and data augmentation techniques used to obtain training and validation datasets. Section 5 describes the techniques used for synthetic data generation. Section 6 describes the model architecture and parameters used for the CNN classifier, followed by section 7 containing the model performance results and discussion. To finish, in section 8 we summarise our findings, and discuss potential directions for further research.

## 2 DEEP LEARNING

Machine learning is a subset of artificial intelligence which focuses on creating algorithms that are able to process information to inform future decisions, but without being specifically programmed to do the task at hand. Deep learning is a further subset of machine learning, which explores how we can actually extract the useful pieces of information needed to inform future decisions (Schmidhuber 2014). In deep learning successive layers are used to progressively extract higher and higher level features from the raw data. Traditional machine learning algorithms typically try to extract information using hand-engineered features or rules about a dataset. Manually defining features and how to extract them can be time consuming, brittle

[1] http://www.birmingham.ac.uk/bear
[2] https://github.com/jxm1028/Y4-Project

and not scalable in practice. On the other hand, in deep learning the underlying features of a dataset are learnt directly from the data.

Deep learning is currently a rapidly growing topic, expanding both in terms of what is capable and possible applications. Although deep learning algorithms have existed for decades, there has been a recent surge in advancements (Shrestha & Mahmood 2019). This is in part due to the large volumes of data available today, which are required by the algorithms, and the ease of collecting new data. Deep learning algorithms are also massively parallelisable, so are able to benefit from modern GPU architectures and hardware acceleration. Today there are also many great open source deep learning toolboxes available, such as TensorFlow, which make building and deploying models extremely streamlined. They also make the field much more accessible to non machine learning experts, sparking a vast variety of applications across many disciplines.

### 2.1 Artificial neural networks

Most modern deep learning models are based on artificial neural networks (ANNs). ANNs, inspired by biological neurons, are a network of connected nodes / neurons. Typically a single neuron takes in many (numerical) inputs, multiplies each by an individual 'weight' value, sums the results, adds on a 'bias' value, and finally applies an activation function to output a single value. This process can be represented mathematically by,

$$y = f\left(\sum_{j=1}^{n} w_j x_j + w_0\right) \tag{1}$$

where $y$ is the output of a single neuron, $x_j$ are the inputs to the neuron, $w_j$ are the weights for each input, $w_0$ is the bias, and $f$ is the activation function. An image representation of the function of a single neuron can also be seen in figure 2.

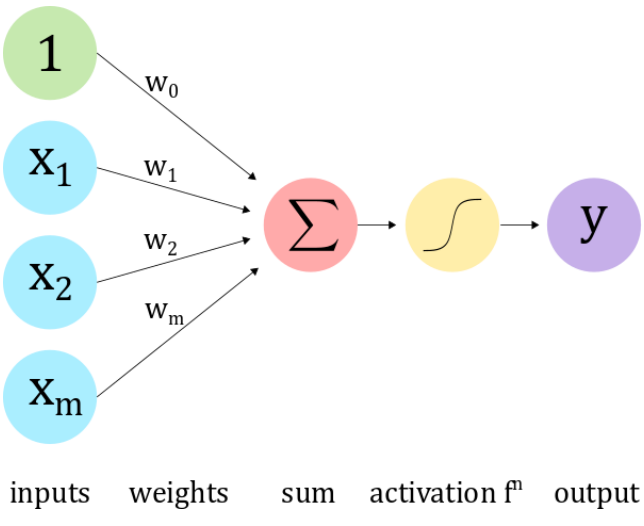


**Figure 2.** Pictoral representation of how a single neuron receives and processes inputs to produce an output.

When many such neurons are combined in an interconnected network, although only consisting of simple functions, they are able to approximate complex non linear functions. A simple example of an ANN with only a few neurons can be seen in figure 3. The simplest arrangement of neurons is in fully connected (a.k.a dense) layers, where the neurons are arranged into layers with the outputs from neurons in one layer feeding in as the inputs to the neurons in the next layer (Rosenblatt 1958). In this setup each neuron in a layer is connected to every neuron in the previous and next layers. The layers in-between the input and output of a network are often called 'hidden layers'. This is because none of the parameters of these layers are directly enforced, but are instead learned by the network itself during training.

A model composed of a sequence of layers is called a sequential model. Neural networks can be described as 'deep' or 'shallow'. Although there is no distinct cut off point between when a model transitions from shallow to deep, typically to create a deep neural network involves adding more layers to create a more hierarchical model. This means the output is computed by going deeper and deeper into the feature representation of the input data. Increasing the depth of a model also increases the model's 'capacity' (Donier 2019). The capacity of model is an informal term describing how much information about a dataset the model can hold. The higher the capacity of a model, the more relationships between more variables it would be able to successfully fit. In order to perform best, and avoid over fitting, the capacity of a model should be proportional to the complexity of its task.

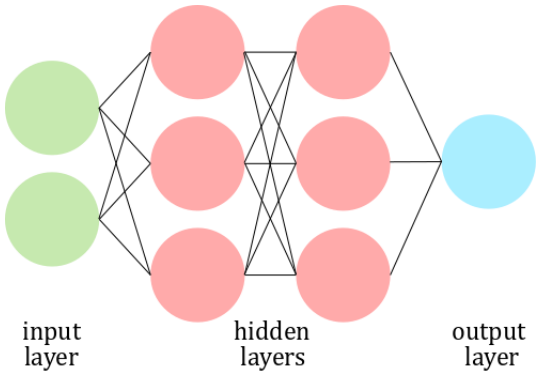


**Figure 3.** Simple sequential ANN model architecture, consisting of an input layer, two hidden dense layers, and a dense output layer. This image shows layers with only a few nodes for simplicity, but in practise dense layers can have hundreds or thousands of nodes.

Applying an activation function before the output of each neuron is particularly important. This is because these (non-linear) activation functions introduce non-linearities into the network, and non-linearities are what allow the network to approximate arbitrarily complex functions (Tianping Chen & Hong Chen 1995). Without an activation function the output signal of a neuron would be a linear function, making the network a simple linear regression model. While easy to solve, linear equations are limited in their complexity and have less power to learn complex functional mappings from data. The most common examples of activation functions include; logistic sigmoid, tanh and rectified linear activation unit (ReLU). The sigmoid and tanh functions map the outputs from a neuron into a finite range. For sigmoid this range is (0, 1), hence it is especially useful when wanting the network to output a probability and often used in the output layer of a network. ReLU is equivalent to $max(0, x)$, so effectively just sets negative outputs to zero. Although the simplest function, ReLUs have been proven outperform other activation functions, resulting in faster and more reliable convergence on the optimal network parameters during training (Changpinyo

et al. 2017). As a result they have been widely adopted by most deep learning models used today.

When data is input into a neural network the information is processed and propagated forward through each subsequent layer in the network until reaching the output layer, hence is often called forward propagation.

Although able to approximate complex functions, when first initialised a network won't know anything about the data so will give a random, undesirable output. In order to correctly map the given input data to the desired outputs the network first needs to learn the best weights and biases for the problem. These variable weights and biases are known as the model parameters. In order to learn the optimal model parameters for a desired task a loss function needs to be defined. Then, during training the model aims to find the optimal parameters that minimise the total loss of the training data. For a binary classification problem a commonly used loss function is binary softmax cross entropy loss (Janocha & Czarnecki 2017). Whereas for a regression problem a common loss function is a mean square error loss. For each data example passed into a network a loss can be calculated, typically depending on how close the output from the network is to the true data label. The average loss over many examples is called the empirical loss.

A model learns to minimise the loss function using an algorithm called gradient descent (Amari 1993). In gradient descent, the gradient of the loss function with respect to each of the weights is calculated. This gives the direction of maximal ascent. Then a small step is taken in the opposite direction (downhill) giving a new value for the weights. Repeating this process eventually leads to a local minima of the loss function.

Gradients with respect to the model parameters are calculated using back-propagation (Hecht-Nielsen 1989). Back-propagation utilises the chain rule, enabling computation of the gradient of the loss function with respect to each weight. Gradients are computed one layer at a time, iterating backwards from the output layer to input layer.

Overall the training loop of an ANN involves the forward propagation of some input data through the network, then using the output to calculate the loss and back-propagate through the network to update the weights to reduce the loss. This process can be repeated for lots of different input training data examples. Each time all of the data in a training set is fed through the network is called an epoch. Over many epochs an ANN hopefully learns and improves to be able to successfully achieve the desired task. In practise, training optimisation can be very difficult, in part due to the high complexity of the loss function (Li et al. 2018).

One model parameter that can be changed to help optimise training is the learning rate. The learning rate is how large of a step is taken when updating the weights during gradient descent. Stable learning rates aim for smooth convergence on the loss minima. Too small learning rates result in getting stuck in local minima during training, whereas if too large the loss function can diverge. It can be tricky to test many learning rates to find one which is just right for a problem, so one approach is to use an adaptive learning rate (Kim et al. 1996). This allows the learning rate to change depending on the situation, for example the size of the gradient, or how fast learning is happening. Popular gradient descent algorithms, often referred to as 'optimizers' in deep learning, include Adam and RMSProp.

Another way to help optimise model training is using mini-batches (Cotter et al. 2011). Here instead of calculating the gradient of the loss function for all training data, which can be computationally expensive for large datasets, gradients are estimated by a subset of the data. Overall this achieves faster training iterations in trade for a lower convergence rate. Mini-batches also allow to parallelize the computation, since batches can be split up onto multiple GPUs.

One of the main obstacles in any machine learning task is the problem of over-fitting. Over-fitting can occur when a model is too complex, which allows it to memorise certain aspects of the training data, resulting in poor generalisation to unseen data. Similarly, under-fitting is when a model doesn't have the capacity to fully learn the features of a dataset. The aim is the middle ground between the two regimes where the model is not too complex that it can memorise all of the training data, but it is still able to generalise when it sees new data. Regularization is a technique aimed at discouraging complex information from being learned, to ensure that a model is able to generalise well to new data (Smirnov et al. 2014).

An important regularization technique is dropout (Srivastava et al. 2014). During training a dropout layer randomly sets some of the hidden neurons to zero. This allows the network to lower its capacity, and stops it building memorisation channels through the network where it just tries to memorise the training data. This forces the network to generalise better, and have multiple channels through the network.

Early stopping is another regularization technique (Yao et al. 2007). Often when training a network the available data is split into training and validation samples. The network is never trained on the validation data, so it can be used as a good measure of well the network is able to generalise to new data. The aim of early stopping is to identify the point where the validation loss starts to increase and diverge from the training loss, as this is an indication that the network is starting to over-fit. The training can simply then be stopped at this point.

### 2.2 Convolutional neural networks

Given above are the fundamentals of how neural networks work. However solely using dense layers can have some limitations, particularly when working with image data. When using dense layers an image must first be flattened, this looses all of the 2D spatial structure in the image. Images can also have very high dimensionality, resulting in a very large number of model parameters. Instead of flattening the input image data we could connect patches of the input image to neurons in the hidden layer, hence retaining spatial structure. This is the key concept behind convolutional layers. The conservation of spatial structure, while also doing local feature extraction using convolutions is at the core of CNNs for computer vision tasks (Fukushima 2007).

Convolutional layers employ filters of weights to extract local features. The same filter is applied to all patches in an image to produce a feature map, and multiple filters can be used to extract different features from the image. A pictorial example of a convolution operation on an image can be seen in figure 4. Mathematically a convolution involves element wise multiplication of a filter with each patch of an image. The sum of each element wise multiplication is then equal to the new pixel values for the resulting feature map.

The output from a convolutional layer for a given input image is a volume of images, with the depth dimension corresponding to all of the different filters / features that you want to detect. Typically CNNs start with low depth convolutional layers, since there are not many different low level features as they are common to many images. Then the depth progressively increases throughout the network as higher level features are extracted, which many only correspond to a specific subset of the image dataset (Masci et al. 2011).

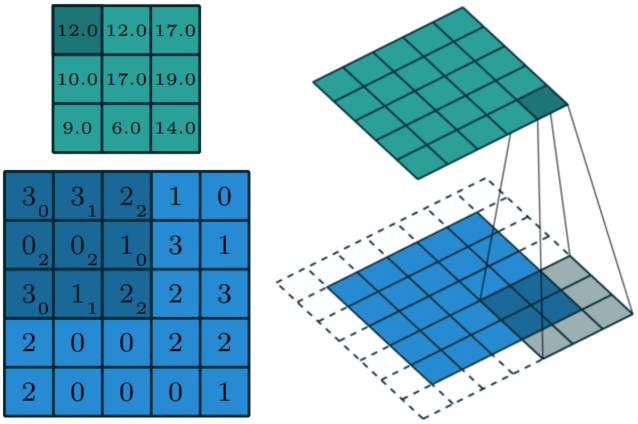


**Figure 4.** Two representations of a convolution operation on an input image to produce a feature map. The left image shows the weight values of the filter and input, showing the calculation taking place. The right image includes padding of the input image, preserving the shape of the output image. In both cases the kernel size is (3, 3) and stride length is 1. Graphics from Dumoulin & Visin (2016).

Some extra parameters than can be specified for each convolutional layer are the stride and kernel size (Dumoulin & Visin 2016). The stride of a convolutional layer corresponds to how many pixels the filter is moved by between each image patch. Strides greater than one can be used to reduce the dimensionality of the output feature map. The kernel size is the desired dimension for the filters, e.g. (3x3). Larger kernel sizes means each convolution looks at a larger patch of the input image.

Another type of layer that is often used in CNNs is a pooling layer (Dumoulin & Visin 2016). Pooling layers downsample the spatial resolution of the input image. This can be useful when dealing with multiple scales of features within an image. The most common example of a pooling operation is max pooling. Here the largest pixel value within the kernal size is used, with the rest being discarded.

CNNs are able to layer convolution and pooling operations to learn a hierarchy of features which we want to detect the presence of in the image data. After the convolutions we can take the learned features and use them to make some classification about the image, using dense layers with sigmoid activation functions to output a probability. A simple example showing the typical recipe for building a CNN for classification is shown in figure 5.

One thing that makes CNNs such a powerful computer vision

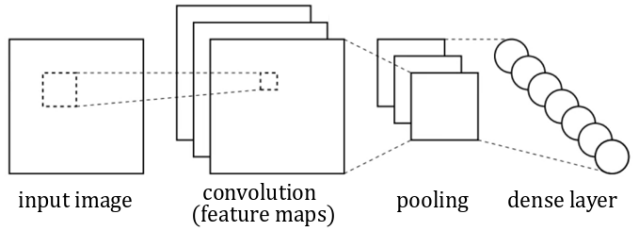


**Figure 5.** Example CNN structure showing the typical flow of data from input, through convolution and pooling layers, to a fully-connected output for classification. The multiple images shown in the convolution layer correspond to the number of different filters used, i.e. the depth of the convolution layer.

tool is their ability to cope well with variation in the input images (Li et al. 2014). This includes variations such as; occlusion, deformation, scale variation, background clutter, viewpoint variation and intra-class variation.

### 2.3 Deep generative modelling

The previous sections describe how neural networks can be used to learn patterns within data. However there are many different and inventive model architectures and layers that can be used to adapt neural networks for different tasks. One such use of neural networks is to generate brand new data based on the learnt patterns and information (Xu et al. 2015). This is a new and emerging field within deep learning.

Section 2.1 described how networks can be trained using supervised learning, when training data comes with labels. In that case the aim was to learn a functional mapping from the data to the labels, where the functional mapping is described by a neural network. In contrast, unsupervised learning requires only the data, without labels. In that case the aim is to learn the hidden / underlying structure in the data. This can help to give an insight into what, at the foundational level, the explanatory factors are behind the data, and to generate brand new examples that resemble the true data distribution. Generative modelling is an example of unsupervised learning (Sanger 1989).

In generative modelling the goal is to take input training samples from some distribution, and learn a model that represents that parent distribution. For a given dataset we may not know the exact distribution of features within it, and the data could in fact be very biased to particular features that are over-represented. For example, in a dataset of faces some skin tones, genders, orientations, etc, could be over-represented. Generative models can learn features to find over and under represented parts of the data to create fair, more representative data sets to train an unbiased classification model.

Generative models can also be used to help detect outliers from the distribution (Aggarwal 2015). Outliers include new or rare data, and are important in training as to avoid unpredictable behaviour in a classifier.

In this study we focus on the class of generative model; generative adversarial networks (GANs), first proposed in Goodfellow et al. (2014). Another similar class is the variational autoencoder (VAE; Kingma & Welling (2013)), which is not explored in this work. These are both examples of latent variable generative models. A latent variable is an unobserved variable, but one which can be inferred from the observed variables. These are the most fundamental variables, which carry the most important / compressed information of a dataset. Latent variable generative models aim to find the latent variables of a dataset, and approximate their distribution.

In GANs the aim isn't to explicitly model the underlying distribution of some dataset, but instead to just generate new instances that are similar to the seen data (Dosovitskiy et al. 2015). This can be an effective solution when the dataset has a very complex distribution that cannot be directly learnt in an efficient manner. As the name suggests, GANs make use of two competing neural networks. A 'generator' network gets input random noise, with the aim to output the training distribution. To achieve this a 'discriminator' network is used to classify the fake / generated data from the real data. Loss functions for each network are then defined by how good the generator is at fooling the discriminator; with the generator aiming to have its data classified as real, and the discriminator aiming to be able to tell the difference between real and fake data. The

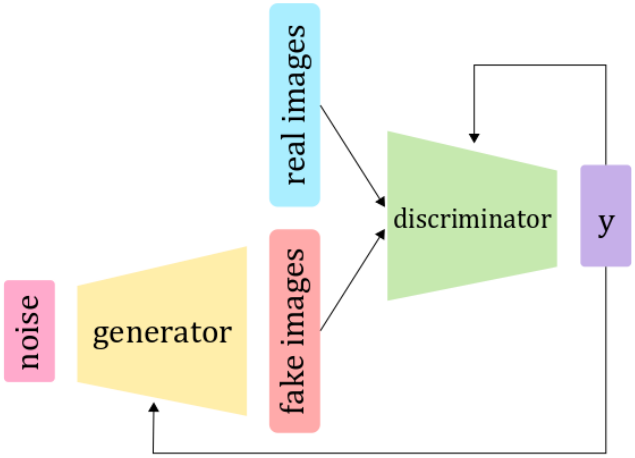


**Figure 6.** Representation of the training loop for a GAN. The generator and discriminator networks are shown as wedges, indicating up-sampling or down-sampling from the input to output. Arrows flow from the discriminator output, y, back to the two networks indicating training of the networks via back-propagation using loss functions calculated from y.

competition between the two networks forces each to improve and learn together, eventually resulting in a generator that can produce very realistic fakes. The structure of a GAN can be seen in figure 6.

When aiming to generate image data, both the generator and discriminator networks are based on a CNN architecture, hence the technique is often referred to as deep convolution GAN (DCGAN, Fang et al. (2018)). The discriminator uses a typical CNN structure; where the input into the model is an image, and throughout the network the image down-sampled using convolution and pooling layers to extract higher and higher level features from the image, until a final dense output layer to predict whether the image is real or fake. On the other hand, the generator needs to achieve a reverse process. Input into the generator is a low-dimensional vector, which then needs to be up-sampled to output an image. To achieve up-sampling to increase the dimensionality of the output, transposed convolutions are used (Dumoulin & Visin 2016).

## 3 TRAINING SAMPLE SELECTION

When constructing our sample of labelled compact, bent-tailed, FRI and FRII radio galaxy sources, we limited our search to only include images from the FIRST (Becker et al. 1995) radio survey. This was chosen because it contained the most available labelled data, and was the most wide and sensitive survey to date. It is also beneficial to use data from only one survey, in order to best explore the impact of synthetic data. Data looks different from each survey, hence synthetic data generation also depends on the survey.

Requirements for each source are the coordinates and its classification label. The sample of radio galaxies was selected using a similar methodology to Aniyan & Thorat (2017) and Alhassan et al. (2018). Since there is no single source catalog which is sufficiently large, we have combined several catalogs when creating the training sample for each morphology. Each used source catalog is described below.

(i) **CoNFIG:** The Combined NVSS and FIRST Galaxies catalogues (CoNFIG) (Gendre & Wall 2009; Gendre et al. 2010) contain 859 sources in total. The catalog was created to address to lack of FRI/II samples in literature. These were classified into FRI, FRII and compact sources (as well as sources of uncertain morphology which were not used in this study). In the catalog the FRI/II type radio galaxies have a associated flag describing the confidence in the classification as either 'confirmed' or 'possible'. Only the 'confirmed' sources were selected for the training sample, because to ensure high classification accuracy it is best to include high quality training samples over a large number of poorly labelled samples. The final catalog contains 50 confirmed FRIs and 390 confirmed FRIIs. Compact sources were identified as being smaller than 3 arcsec, and similarly to FRI/IIs, the only compact sources used were the ones with an associated 'confirmed' flag.

(ii) **FRICAT:** The FRICAT catalog (Capetti et al. 2017a) of FRI type radio galaxies is a sub-sample of the Best & Heckman (2012) catalog. Constraints were imposed on the redshift and size of the sources to ensure high resolution, and objects were only included in the catalog if at least two out of the three authors agreed on its classification. The catalog contains a further 219 FRIs.

(iii) **FRIICAT:** An equivalent catalog for FRII type radio galaxies is the FRIICAT catalog (Capetti et al. 2017b). Similar selection criteria were used, resulting in a catalog of 122 FRIIs.

(iv) **Proctor:** In Proctor (2011) FIRST radio sources have been classified based on their morphology using a combination of pattern recognition techniques and visual inspection. Sources were classified into different bent morphologies, including WATs and NATs, as well as other more complex bent morphologies. When creating our sample of bent-tailed galaxies, we only used the WATs and NATs, totalling 412, to be consistent with Alhassan et al. (2018).

From the initial sample selection, cross-matching the coordinates of sources from each catalog we found some duplicate sources, which were removed. Some CoNFIG FRI/II sources included bent-tailed galaxies, so these sources were re-labelled as bent or discarded. Upon manual inspection of the sources we also excluded the source images which; had strong artefacts, contained multiple sources, were too large to fit in a 150x150 pixel cutout, or had unclear morphologies. The initial selection and final number of sources of each morphology used in our sample is summarised in table 2.

**Table 2.** Summary of the sample selection process. Image based cuts refer to images excluded due to the presence of artefacts, multiple sources, weak structural information, or very large source size. The morphology based cut counts images excluded due to uncertainty in the source morphology / classification.

| | Initial sample | Image cut | Morphology cut | Final sample |
|---|---|---|---|---|
| FRI | 269 | 14 | 35 | 220 |
| FRII | 513 | 18 | 90 | 405 |
| Bent | 412 | 65 | 121 | 226 |
| Comp | 284 | 0 | 41 | 243 |
| Total | 1478 | 97 | 287 | 1094 |

## 4 IMAGE PRE-PROCESSING AND AUGMENTATION

Source images were first downloaded as 4.5 arcmin squares, corresponding to 150x150 pixels, from the FIRST image cutout server

[3]. The pixel values of the images correspond to brightness in units of mJy/beam. How the images are prepared before being fed into a network can have a large impact on the model performance (Aniyan & Thorat 2017).

First it is important to remove noise and background from the images. This allows the CNN to focus on identifying features of the sources themselves, not background or imaging artefacts that can come as a result of the telescope. To achieve this the sigma-clipped statistics [4] of each image are calculated, which gives an estimation of the background noise distribution. Pixel values above a chosen noise threshold can then be discarded, leaving only the pixels belonging to the object remaining. In this study, pixel values below 3 sigma above the background median were set to zero. This value for sigma-clipping was identified as the most suitable in Aniyan & Thorat (2017).

At this point it is then possible to crop or down-sample (resize) the images. While it is not preferable to crop or resize images smaller than necessary since information about the image is lost, it can be useful to speed up training, or required when generating GAN data as discussed in section 5. In this study images are down-sampled to 56x56 images in order to match the size of generated synthetic images. While this is a significant loss of information, preliminary tests showed that it actually had minimal impact on the classification accuracy of the network. However future studies should endeavour to retain as much information in the input images, perhaps through generating larger synthetic images. Image resizing is done via interpolation of the pixel values to down-scale the image.

Cropping of the image to remove background could be used to reduce image size and increase training speed. It would also reduce the amount of source information lost through resizing. However, since the radio sources are of varying sizes and locations in the images we were not able to crop the images further than the initial 150x150 images, without first calculating source size and location properties. Hence no cropping was applied to the images.

After resizing, the pixel values of an image were normalised to a [-1,1] range. Normalisation is important as it ensures the network only learns features on a morphological basis, rather than based on the source brightness. It also helps maintain the homogeneity of the sample space, helping the network to generalise learnt features.

Before augmenting the data to boost the number of training samples, the dataset must be split into a training and validation subsets. As discussed in section 2.1, validation samples are never shown to the network during training. That way they can be used as a measure of how well a trained model can generalise to unseen data, or if the model has over-fit to the training data. For the best model performance it is important to train the model with as much data as possible, however validation data is needed to fairly test the performance. In this study, we split the data into training and validation samples by using 40 images from each class as the validation set. This ensures an equal representation of classes in the validation set, and roughly corresponds to a training to validation ratio of 80:20.

The performance of any machine learning model is always impacted by the size and quality of the training data. More data allows a model to generalise better by identifying features consistent across the most data samples. Typical sizes for deep learning applications are to the order 10,000 data samples (Hestness et al. 2017). Our original dataset contains a total of 1094 images, 20% of which are required as validation samples. In this case data augmentation is clearly necessary.

It is fortunate that in astronomy image data has high symmetry, in the sense that the source properties of an image are not altered by rotations, translations, flips or zooms. This means that data augmentation can be very effective, and a large dataset of valid images can be generated from a relatively small initial sample. In this study, data augmentation was done using random rotations and flips of the original images. This augmentation also helps to teach the network rotational invariance when identifying image features. Additionally, augmenting each class separately allows to fix class imbalances by generating an equal training sample for each class.

Table 3 details the number of images of each class in the original sample, and in the final augmented training and validation samples. The different pre-processing steps used as also summarised in figure 7.


**Figure 7.** The different stages of image pre-processing when creating the training and validation datasets. Original images are 150x150 pixels. After sigma clipping the images are resized to 56x56, this is to match the size of the synthetic images generated by the GAN (since all the inputs to a CNN need to be the same shape).

**Table 3.** Summary of how the original samples were split into test and training subsets, and how much the training sample was augmented to, using flips and rotations.

| | Original sample | Test sample | Training sample | Augmented training sample |
|---|---|---|---|---|
| FRI | 220 | 40 | 180 | 6,000 |
| FRII | 405 | 40 | 365 | 6,000 |
| Bent | 226 | 40 | 186 | 6,000 |
| Comp | 243 | 40 | 203 | 6,000 |
| Total | 1094 | 160 | 934 | 24,000 |

## 5 SYNTHETIC DATA GENERATION

The main aim of this study is to explore the impact of training a CNN, for radio galaxy classification, with fake / synthetic image data. We explored two methods of generating synthetic data; the first being a simple geometric simulation program using our knowledge of the image domain, and the second using the deep generative approach of GANs, as discussed introduced in section 2.3.

[3] https://third.ucllnl.org/cgi-bin/firstcutout
[4] https://docs.astropy.org/en/stable/api/astropy.stats.sigma_clipped_stats.html#astropy.stats.sigma_clipped_stats

### 5.1 Geometric simuation

For the simple geometric simulation an empty array was first created, in which pixel values would be added. The main image features of a typical AGN with jets are a central peak corresponding to the AGN, with two additional peaks for each jet. In the simulation the peaks were approximated by a 2D Gaussian, but before adding them to the image the central locations needed to be defined. A random grid location near the centre of the image was chosen to be the central AGN coordinate. Most AGN are roughly symmetric, meaning each jet peak is a similar distance away from the central AGN. To implement this a random distance was chosen, within the constraints of the total image size, and using this along with a random bending angle, the two coordinates of the jet peaks could be defined. Approximating the peaks by 2D Gaussian functions allowed for flexibility in the x and y spread of the peaks, as well as the angle of rotation. Elliptical Gaussian functions were created at each peak given by the general function,

$$f(x, y) = A \exp\left(-(a(x - x_0)^2 + 2b(x - x_0)(y - y_0) + c(y - y_0)^2)\right)$$

where $x_0, y_0$ is the central peak location, and the a, b, c coefficients are defined as,

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2}, \qquad b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2},$$

$$c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2}$$

which allow for specifying the x and y spread of the peak ($\sigma_x, \sigma_y$), as well as the angle of rotation, $\theta$, of the Gaussian. Calculating the values of the functions for each pixel coordinate, and summing the results for each peak gives the simulated image.

In order to create more realistic FRI/II type radio galaxies, asymmetric Gaussians were created with different $\sigma$ values on either side of the peak coordinate. This allowed the simulation of longer, more diffuse plumes leading away from the jet peaks. The asymmetry parameter also allowed for a distinction between simulated FRI and FRII type radio galaxies. FRIs typically have a larger spread from the jet peak leading away from the central AGN, with the reverse being true for FRIIs.

Extra Gaussian functions with slightly different parameters could also be added to each peak to create more complex structures, such as 'halos'. The addition of random noise to the images also helped increase how realistic the synthetic images looked. With careful consideration of the range of random values that each parameter can take, a dataset of 6000 randomly generated synthetic images was created for each classification. Compact sources consisted only a single central peak, bent-tailed sources were created with bending angles of between 20 and 180 degrees, and FRI/II type sources were created with bending angles less than 10 degrees. A definition for a radio galaxy's bending angle is given in Garon et al. (2019). Examples of generated images using this approach can be seen in figure 8.

This approach allows for image generation of any size. We generated 56x56 images to match those generated by GANs, and the real data.

While this is a very simple approach to synthetic data generation, a similar but more sophisticated approach could be to use the results of physical simulations based on the predicted physical processes behind the creation of AGN, such as presented in Barniol Duran et al. (2017).

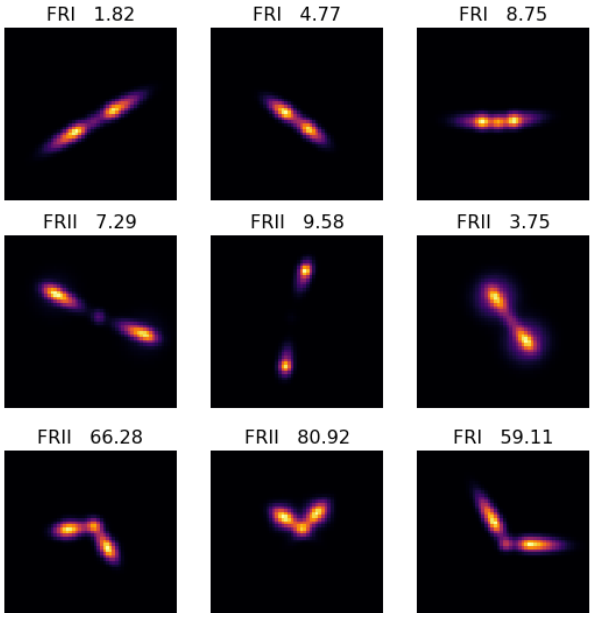


**Figure 8.** Example simulated images created using the simple geometric simulation. Above each image is the source morphology (FRI/II) and bending angle. The first two rows aim to mimic regular FRI/II sources, with low bending angles, and the third row shows high bending angle morphology.

### 5.2 GAN simulation

The above geometric simulation relies on knowledge about the image domain trying to be classified, meaning that it can be time-consuming to create and might not transfer well to new object classifications or new surveys. On the other hand GANs don't rely on any prior knowledge of the data, and the same approach can be applied for many different datasets, or object classifications.

The process of training a generator and discriminator by forcing them to compete against each other was outlined in section 2.3. When implementing the process to generate synthetic training images, there needs to be a good balance in the learning rates of each network. If either network trains faster than the other it can inhibit the training of both networks, resulting in the generator not being able to produce good fake images that resemble the training data. To achieve this we found that the best network architectures were quite simple CNNs, in comparison to the CNN architecture used for the classification model outlined in the next section (6). The exact model architectures used for the generator and discriminator networks are shown in appendix A.

Training GANs can be very computationally expensive, especially with large images. They can also struggle when it comes to creating accurate, high resolution images. To ease some of this computational pressure, and to generate more realistic images, in this study we only create images of 56x56 pixels. Whilst it is certainly possible to create higher resolution images, it can prove more difficult, particularly when designing the generator's model architecture. A novel progressive growing technique to stably increase the resolution of images is given in Karras et al. (2017).

To train the GAN the same pre-processing and data augmen-

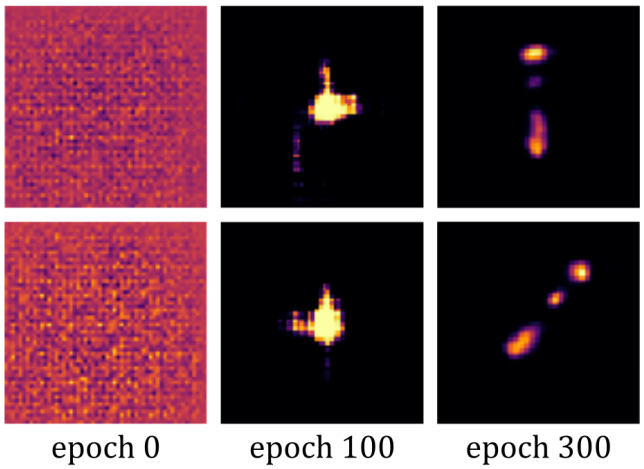


**Figure 9.** Each row shows the fake images produced by the generator during training for a fixed input vector, showing the improvement in the generators ability to create realistic fake images. Before training, at epoch 0, the images can be seen to be random noise. The images then become more realistic until convergence after around 300 epochs.

tation technique was used as described in section 4 to create the training image sample. The images from each classification were augmented using rotations and fips to 60,000, to match the size of the mnist dataset on which the GAN architecures were tested (LeCun & Cortes 2010). Since GANs are an unsupervised technique with no image labels, a separate GAN was trained for each classification (FRI, FRII, bent-tailed and compact). Each GAN was trained for 500 epochs, after which the trained generator model was saved. GAN training can also be visualised by looking at the generated fake images for some fixed input vectors after each epoch, these

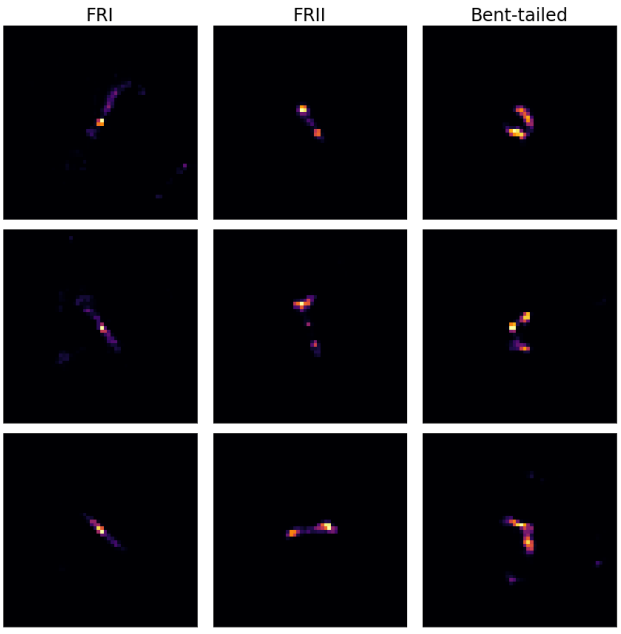


**Figure 10.** Random examples of the final synthetic images produced using GANs. Each column shows images belonging to each class; FRI, FRII and bent-tailed respectively.

images can be used to create a training GIF, snapshots of which can be seen in figure 9. From the training GIFs it could be seen that the generated images started to converge to a solution after 300 epochs, hence training for 500 epochs was an appropriate choice to ensure convergence.

After training a generator for each classification, we used them to produce training data we could use to train a classification CNN. The output of the generator is an approximation of the real training data distribution. This distribution is sampled using random input vectors to create a dataset of synthetic images. A dataset of 6000 synthetic images was created for each classification, to match the size of the augmented training set of real images. Examples of the synthetic images created for each classification are shown in figure 10. The generated images from a GAN vary in quality, with some being more realistic than others.

## 6 CLASSIFICATION CNN NETWORK ARCHITECTURE

All of the building blocks for creating a CNN model were outlined in section 2. Neural network model design is often considered a hyper parameter optimisation problem (Smithson et al. 2016). Network architecture hyper parameters include; the number and order of convolution, pooling and dense layers, the type and learning rate of the optimizer, the kernel sizes, strides and nodes for each convolutional layer, the batch size, the number of epochs to train for, etc. There are no strict rules to determine the optimal values of these parameters, and sometimes no obvious reasons why some parameters work better than others. Typically the main factor to take into consideration when designing a network architecture for a task is the complexity of the data.

One popular and powerful CNN architecture for complex classification tasks is AlexNet, which is a deep 12-layered model (Krizhevsky et al. 2012). Deep models are favoured when looking at transfer learning applications (Tang et al. 2019). Shallower networks tend to have less learnable parameters and expressive ability, which makes their transfer learning prospects limited. Aniyan & Thorat (2017) based their model off this architecture and were able to achieve classification accuracies above 90%. However this high accuracy was only able to be achieved through the use of a fusion classifier, where the output of several binary classification models were combined. Alhassan et al. (2018) took a different approach, using a simpler 8-layered network architecture. This was motivated by the belief that current radio images can be described by a small number of features. Using a single trained classification model they were able to achieve accuracies of around 97%.

In this study we are not concerned with transfer learning applications, so inspired by Alhassan et al. (2018) we adopt a similar network architecture, see figure 11. The model consists of 3 convolutional layers, each with a 3x3 kernal size, with an increasing number of filters from 32 to 194 going deeper into the network. The convolutional layers are interlaced with pooling layers, and each uses a ReLU activation function. After the final pooling layer, signalling the end of feature extraction, two dense layers are used for object classification. The first, with 192 nodes and a ReLU activation function, followed by the final output layer of 4 nodes (one for each classification) and a softmax activation function, to be able to interpret the outputs as probabilities. In-between the two dense layers is a dropout layer, with a dropout rate of 0.5, used for more regularization in the model. We use the adaptive mini-batch optimizer, Adam, to minimise the loss, with a batch size of 128 and an initial learning rate of 0.001. The weights and biases in the network

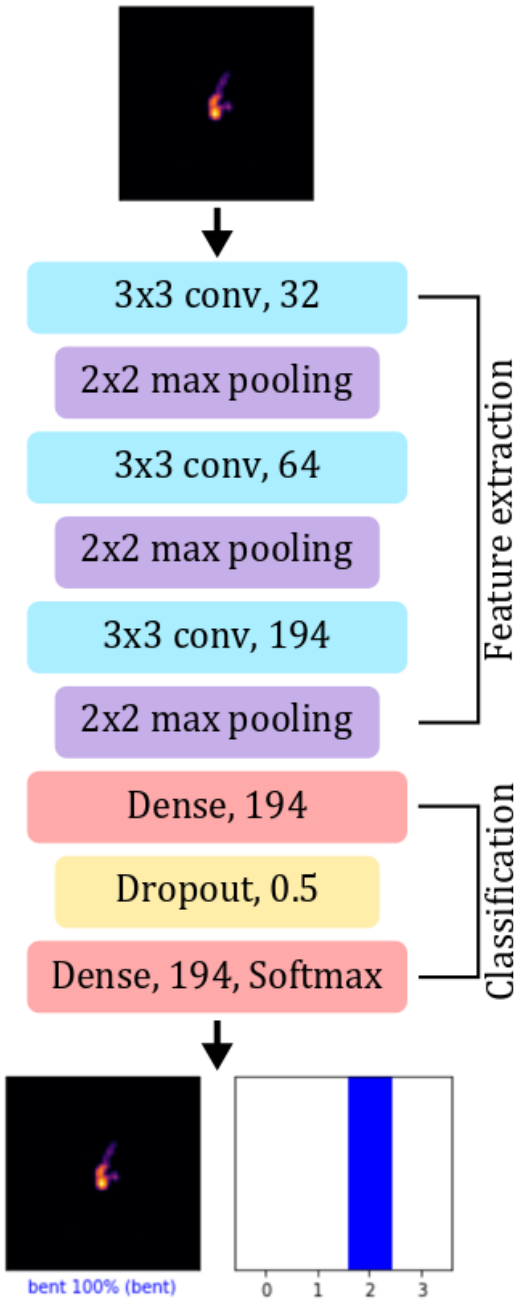


**Figure 11.** Deep CNN model architecture for radio galaxy classification. Each convolutional and dense layer has a ReLU activation, except for the final softmax layer. Input into the model is a 56x56 image, and outputs are four probabilities corresponding to each class. In the above image classes 0, 1, 2, 3 correspond to the classes FRI, FRII, bent-tailed and compact respectively. The shown image was correctly classified as bent-tailed with a prediction probability of 100%.

were initialised to random Gaussian values. A full description of each layer in the model is given in appendix B.

## 7 MODEL TRAINING AND EVALUATION

In order to examine the influence of augmenting the training data with synthetic data, we trained the same classification CNN model for different datasets:

(a) The real image data (Augmented as described in section 4)
(b) A combination of the real and geometric simulation data
(c) A combination of the real and GAN generated data

The above datasets were each used as the training data for the model, while the validation data was kept the same for each different training set. The validation data set only contained real images, since this is ultimately what we aim to be able to classify accurately, not the synthetic data. Validation samples were also augmented using rotations and flips to 1,500 per class, this allows for a more accurate test of model performance. Models were trained with each datset for 100 epochs, since this was long enough for the models to start converging. To measure the performance of the model when trained on each dataset, plots for the loss and accuracy of the model at each epoch were calculated, as shown in figure 12. These metrics were calculated for both the training data set, as well as the validation dataset. The accuracy of the model is the fraction of images which are correctly classified (i.e. the class with the highest predicted probability is the correct class). The loss is the value of the loss function of the dataset, here this is calculated by the categorical cross entropy. The categorical cross entropy for a model with 100% accuracy and complete certainty in each of its class predictions is zero.

Plot (a) shows that when only using real image data the validation accuracy converged to 91% (calculated as an average of the final 20 epochs for consistency). This is then the benchmark value for this problem, which we should aim to improve through the use of synthetic training data. Although 91% is a lower accuracy than achieved with a similar network and dataset in Alhassan et al. (2018), this is most likely a result of the image down-sampling to 56x56. However the main aim of this study is to assess the classification improvements from synthetic training data, not to achieve the highest possible classification accuracy. When trained with only real image data, dataset (a), it can also be seen that the training accuracy of the model is much higher than the validation accuracy, converging on a high accuracy of 99%. This is an indication that the model is over-fitting. A further sign of over-fitting can be seen in the loss plot, where the validation loss starts to increase while the training loss continues to decrease. The minimum training loss is in fact reached within the first 20 epochs of training. Although the validation loss starts to increase, this trend is not mirrored by the validation accuracy decreasing. Instead the validation accuracy continues to level off. The validation loss increasing while the accuracy remains relatively constant implies that as the model continues to train it is still able to classify objects correctly, but the certainty in its classifications decreases.

The model performance for datasets augmented with synthetic data can be seen in plots (b), and (c). Dataset (b) shows very similar performance to dataset (a). This is quite surprising considering how simple and different from the real data the geometric simulation data can sometimes appear. The inclusion of the geometric simulation data gave no performance increase to the CNN, however nor did it hinder the CNN in any way. On the other hand, combing the real data with the GAN generated data, dataset (c), had major impacts on the model's performance. Most importantly, dataset (c) showed an increase in the validation accuracy of the model from 91%, with dataset (a), to 92%. While a 1% increase may sound insignificant, it is in fact a considerable improvement for a machine learning model, particularly when the model was already performing at such high accuracy.

Another key difference in the model performance for dataset (c)

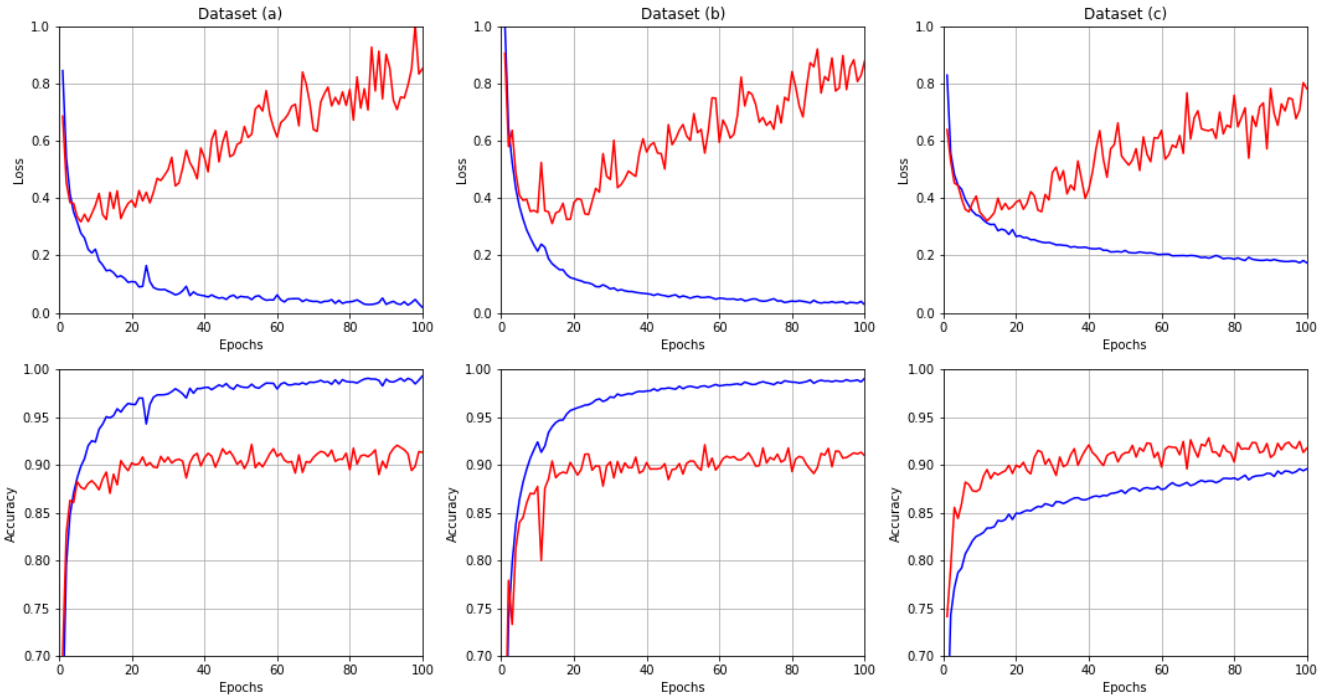


**Figure 12.** Plots showing how the loss and accuracy of the model changed throughout training. Each column of plots corresponds to model training with datasets (a), (b) and (c) respectively. The blue curves show the loss and accuracy for the training set, and the red curves for the validation set.

is that the training accuracy is considerably lower, only eventually reaching an accuracy of 90% after the 100 epochs, compared with the 99% accuracy of dataset (a). In fact, for dataset (c) the validation accuracy is consistently higher than the training accuracy. A low training accuracy is not necessarily a bad thing as a model's ability to classify new, unseen data is the only thing that realistically matters. However, the low training accuracy does suggest that the quality of the GAN synthetic data is not good enough to be comparable to real data. Despite this, the increase in the validation accuracy implies that the GAN synthetic data still provides the network with useful information for generalising better to real data. A final difference in model performance is shown in the loss plot, where the validation loss increases slower than for dataset (a). This implies that the inclusion of GAN synthetic data helps to reduce over-fitting in the model, as well as increase the classification accuracy.

As well as plotting loss and accuracy metrics of a network, another way to show classification performance is with a confusion matrix. Confusion matrices compare the true class labels to the labels predicted by the trained model. This allows us to see the individual classification accuracies for each class, as well as which class images are predicted to be when not predicted as the correct class. Figure 13 shows the confusion matrices for (a) the model trained on only real data, and (c) the model trained with a combination of real and GAN generated synthetic data. For both datasets the plots show that the model is able to very successfully classify compact sources (accuracy roughly 100%), which should be expected as they are the simplest and most difference from the other classes. The class the model struggles to identify the most is FRI-type radio galaxies, which it incorrectly classifies as all three of the other classifications (around 5% each). A possible reason for this is that FRIs tend to be samller than FRII and bent-tailed galaxies, so when downsampling the images FRIs could lose the most source information. FRIs are also typically the faintest sources, so they could lose more of their structure during sigma-clipping.

The confusion matrices show that despite overall validation accuracy increasing with the inclusion of GAN synthetic images, the classification accuracy of FRIs in fact decreases from 86% to 82%. This suggests that the quality of the FRI images generated by the GAN are worse than that of the other classes, and in fact hinder the classification accuracy of the network rather than improve it.

## 8 CONCLUSIONS

Upcoming radio surveys will generate vast amounts of data, rendering visual inspection methods impractical, and motivating the need for automatic source classification. Current source catalogs only contain a small number of labelled radio sources, however deep learning methods require large sample sizes to be able to generalise. This paper explored the impact of training a CNN for radio-galaxy morphology classification with a training set augmented by synthetic data. Two approaches were used for generating synthetic data; a simple geometric simulation, and an application of the deep generative modelling technique, GANs.

Our model aimed to classify sources into four classes; FRI, FRII, bent-tailed, and compact radio galaxies. Labelled training data was collected from several FIRST source catalogs. The architecture of the classifier was a simple CNN consisting of 3 convolutional layers, chosen as it had proved to be capable of high accuracy in previous literature (Alhassan et al. 2018).

Augmenting a training set with synthetic images generated using the geometric simulation method had little effect on the performance of the classification model. However when augmented with GAN generated synthetic images the overall classification ac-

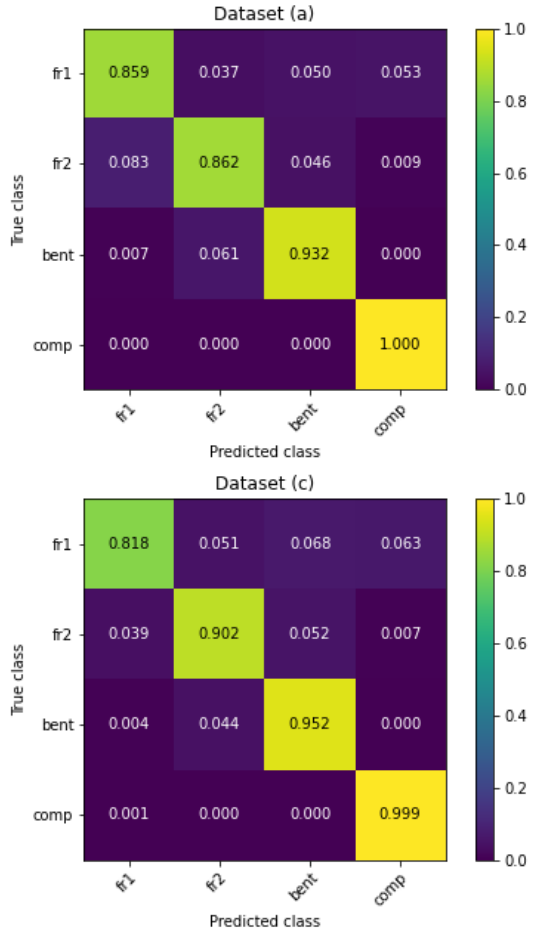


**Figure 13.** Normalised confusion matrices for datasets (a) and (c). Class predictions were made for images in the validation set, a comparison of these predictions to the true data labels of the images is shown by the confusion matrix. Each row is normalised so values represent the fraction of images which are predicted as each class. The matrix for dataset (b) was very similar to that for dataset (a) so not included.

curacy of the model was increased from 91% to 92%. Using these synthetic images also reduced the over-fitting present in the model. Thus we have shown that synthetic images are a viable method of improving the performance of a CNN for the task of radio galaxy classification.

In this study the size of images generated by the GAN were 56x56, hence all other images had to be down-sampled from 150x150 to this size. This resulted in the loss of a lot of source information. Despite still being able to achieve > 90% accuracies at this size, future work should aim to generate higher resolution GAN images so that no source information need be lost, and higher classification accuracies can be achieved. With higher resolution GAN images, we are confident that a greater increase in classification accuracy will be able to be achieved, such as presented in Sandfort et al. (2019).

There were also clear signs of over-fitting present in the model when trained with all datasets. So more follow up work could include a more rigorous optimisation of the CNN architecture and hyper-parameters for this dataset, particularly the inclusion of more regularization in the model, such as batch normalisation.

## REFERENCES


Aggarwal C. C., 2015, in Data mining. pp 237–263
Alhassan W., Taylor A. R., Vaccari M., 2018, MNRAS, 480, 2085
Amari S.-i., 1993, Neurocomputing, 5, 185
Aniyan A. K., Thorat K., 2017, ApJS, 230, 20
Banfield J. K., et al., 2015, MNRAS, 453, 2326
Barniol Duran R., Tchekhovskoy A., Giannios D., 2017, MNRAS, 469, 4957
Becker R. H., White R. L., Helfand D. J., 1995, ApJ, 450, 559
Begelman M. C., Blandford R. D., Rees M. J., 1984, Reviews of Modern Physics, 56, 255
Best P. N., Heckman T. M., 2012, MNRAS, 421, 1569
Blanton E. L., Gregg M. D., Helfand D. J., Becker R. H., White R. L., 2000, ApJ, 531, 118
Braun R., Bourke T., Green J. A., Keane E., Wagg J., 2015, in Advancing Astrophysics with the Square Kilometre Array (AASKA14). p. 174
Capetti A., Massaro F., Baldi R. D., 2017a, A&A, 598, A49
Capetti A., Massaro F., Baldi R. D., 2017b, A&A, 601, A81
Changpinyo S., Sandler M., Zhmoginov A., 2017, CoRR, abs/1702.06257
Cotter A., Shamir O., Srebro N., Sridharan K., 2011, in Advances in neural information processing systems. pp 1647–1655
Croton D. J., et al., 2006, MNRAS, 365, 11
Donier J., 2019, arXiv e-prints, p. arXiv:1902.08572
Dosovitskiy A., Tobias Springenberg J., Brox T., 2015, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 1538–1546
Dumoulin V., Visin F., 2016, arXiv e-prints, p. arXiv:1603.07285
Fabian A. C., 1999, Proceedings of the National Academy of Science, 96, 4749
Fanaroff B. L., Riley J. M., 1974, MNRAS, 167, 31P
Fang W., Zhang F., Sheng V. S., Ding Y., 2018, CMC: Comput. Mater. Continua, 57, 167
Fukushima K., 2007, Scholarpedia, 2, 1717
Garon A. F., et al., 2019, AJ, 157, 126
Gendre M. A., Wall J. V., 2009, MNRAS, 394, 1712
Gendre M. A., Best P. N., Wall J. V., 2010, MNRAS, 404, 1719
Gendre M. A., Best P. N., Wall J. V., Ker L. M., 2013, MNRAS, 430, 3086
Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014, in Advances in neural information processing systems. pp 2672–2680
Hecht-Nielsen 1989, in International 1989 Joint Conference on Neural Networks. pp 593–605 vol.1
Hestness J., et al., 2017, arXiv e-prints, p. arXiv:1712.00409
Janocha K., Czarnecki W. M., 2017, arXiv preprint arXiv:1702.05659
Karras T., Aila T., Laine S., Lehtinen J., 2017, arXiv e-prints, p. arXiv:1710.10196
Kim H. B., Jung S. H., Kim T. G., Park K. H., 1996, Neurocomputing, 11, 101
Kingma D. P., Welling M., 2013, arXiv preprint arXiv:1312.6114
Krizhevsky A., Sutskever I., Hinton G. E., 2012, in Advances in neural information processing systems. pp 1097–1105
Lacy M., et al., 2020, PASP, 132, 035001
LeCun Y., Cortes C., 2010
Li W., Zhao R., Xiao T., Wang X., 2014, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 152–159
Li H., Xu Z., Taylor G., Studer C., Goldstein T., 2018, in Advances in Neural Information Processing Systems. pp 6389–6399
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
Masci J., Meier U., Cireşan D., Schmidhuber J., 2011, in International conference on artificial neural networks. pp 52–59
Norris R. P., et al., 2011, Publ. Astron. Soc. Australia, 28, 215
Padovani P., 2017, Nature Astronomy, 1, 0194
Polsterer K. L., Gieseke F., Igel C., 2015, Automatic Galaxy Classification via Machine Learning Techniques: Parallelized Rotation/Flipping INvariant Kohonen Maps (PINK). p. 81
Proctor D. D., 2011, VizieR Online Data Catalog, p. J/ApJS/194/31

Rosenblatt F., 1958, Psychological review, 65, 386
Sandfort V., Yan K., Pickhardt P. J., Summers R. M., 2019, Scientific reports, 9, 1
Sanger T. D., 1989, Neural networks, 2, 459
Schmidhuber J., 2014, arXiv e-prints, p. arXiv:1404.7828
Shimwell T. W., et al., 2017, A&A, 598, A104
Shrestha A., Mahmood A., 2019, IEEE Access, PP, 1
Smirnov E. A., Timoshenko D. M., Andrianov S. N., 2014, Aasri Procedia, 6, 89
Smithson S. C., Yang G., Gross W. J., Meyer B. H., 2016, in Proceedings of the 35th International Conference on Computer-Aided Design. pp 1–8
Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, The journal of machine learning research, 15, 1929
Tang H., Scaife A. M. M., Leahy J. P., 2019, MNRAS, 488, 3358
Tianping Chen Hong Chen 1995, IEEE Transactions on Neural Networks, 6, 911
Walmsley M., et al., 2020, MNRAS, 491, 1554
Wu C., et al., 2019, MNRAS, 482, 1211
Xu J., Li H., Zhou S., 2015, IETE Technical Review, 32, 131
Yao Y., Rosasco L., Caponnetto A., 2007, Constructive Approximation, 26, 289
de la Calleja J., Fuentes O., 2004, MNRAS, 349, 87

## APPENDIX A: GAN ARCHITECTURE

**Table A1.** Generator model architecture. This model is trained to produce images which are able to fool the discriminator into thinking they are real. The input into this network is a random noise vector of size 100. Activation functions are listed as separate layers since they are applied after batch normalization layers, instead of after dense or convolutional layers. After the first leaky ReLU layer the outputs are resized to (7, 7, 512) before being fed into the following transposed convolution layer. Parameters in batch normalization layers are not trainable, but required to aid convergence during back-propagation. The total number of trainable parameters for the model is 6,862,272.

| Layer type | Nodes | Kernel size | Stride | Output shape | # parameters |
|---|---|---|---|---|---|
| Dense | 7*7*512 | - | - | 25088 | 2508800 |
| Batch normalization | - | - | - | 25088 | 100352 |
| Leaky ReLU | - | - | - | (7, 7, 512) | 0 |
| Transposed convolution | 256 | (5, 5) | 1 | (7, 7, 256) | 3276800 |
| Batch normalization | - | - | - | (7, 7, 256) | 1024 |
| Leaky ReLU | - | - | - | (7, 7, 256) | 0 |
| Transposed convolution | 128 | (5, 5) | 2 | (14, 14, 128) | 819200 |
| Batch normalization | - | - | - | (14, 14, 128) | 512 |
| Leaky ReLU | - | - | - | (14, 14, 128) | 0 |
| Transposed convolution | 64 | (5, 5) | 2 | (28, 28, 64) | 204800 |
| Batch normalization | - | - | - | (28, 28, 64) | 256 |
| Leaky ReLU | - | - | - | (28, 28, 64) | 0 |
| Transposed convolution | 1 | (5, 5) | 2 | (56, 56, 1) | 1600 |

**Table A2.** Discriminator model architecture. This model is trained to output a 1 for real images a 0 for fake images. Input image shape is (56, 56, 1). The total number of trainable parameters in the network is 231,681.

| Layer type | Nodes | Kernel size | Stride | Dropout rate | Output shape | Activation | # parameters |
|---|---|---|---|---|---|---|---|
| Convolution | 64 | (5, 5) | 2 | - | (28, 28, 64) | Leaky ReLU | 1664 |
| Dropout | - | - | - | 0.3 | (28, 28, 64) | - | 0 |
| Convolution | 128 | (5, 5) | 2 | - | (14, 14, 128) | Leaky ReLU | 204928 |
| Dropout | - | - | - | 0.3 | (14, 14, 128) | - | 0 |
| Dense | 1 | - | - | - | 1 | Linear | 25089 |

## APPENDIX B: CNN ARCHITECTURE

.

**Table B1.** Classification CNN model architecture. Input to the model is a (56, 56, 1) image. The final max pooling layer is flattened before the following dense layer, hence the output shape of 9506. The dropout layer has a dropout rate of 0.5, and the strides of all of the convolutional layers is 1. The total number of trainable model parameters is 1,975,892.

| Layer type | Nodes | Kernel size | Output shape | Activation | # parameters |
|---|---|---|---|---|---|
| Convolution | 32 | (3, 3) | (56, 56, 32) | ReLU | 320 |
| Max pooling | - | (2, 2) | (28, 28, 32) | - | 0 |
| Convolution | 64 | (3, 3) | (28, 28, 64) | ReLU | 18496 |
| Max pooling | - | (2, 2) | (14, 14, 64) | - | 0 |
| Convolution | 194 | (3, 3) | (14, 14, 194) | ReLU | 111938 |
| Max pooling | - | (2, 2) | 9506 | - | 0 |
| Dense | 194 | - | 194 | ReLU | 1844358 |
| Dropout (0.5) | - | - | 194 | - | 0 |
| Dense | 4 | - | 4 | Softmax | 780 |

## APPENDIX C: EXAMPLE IMAGES

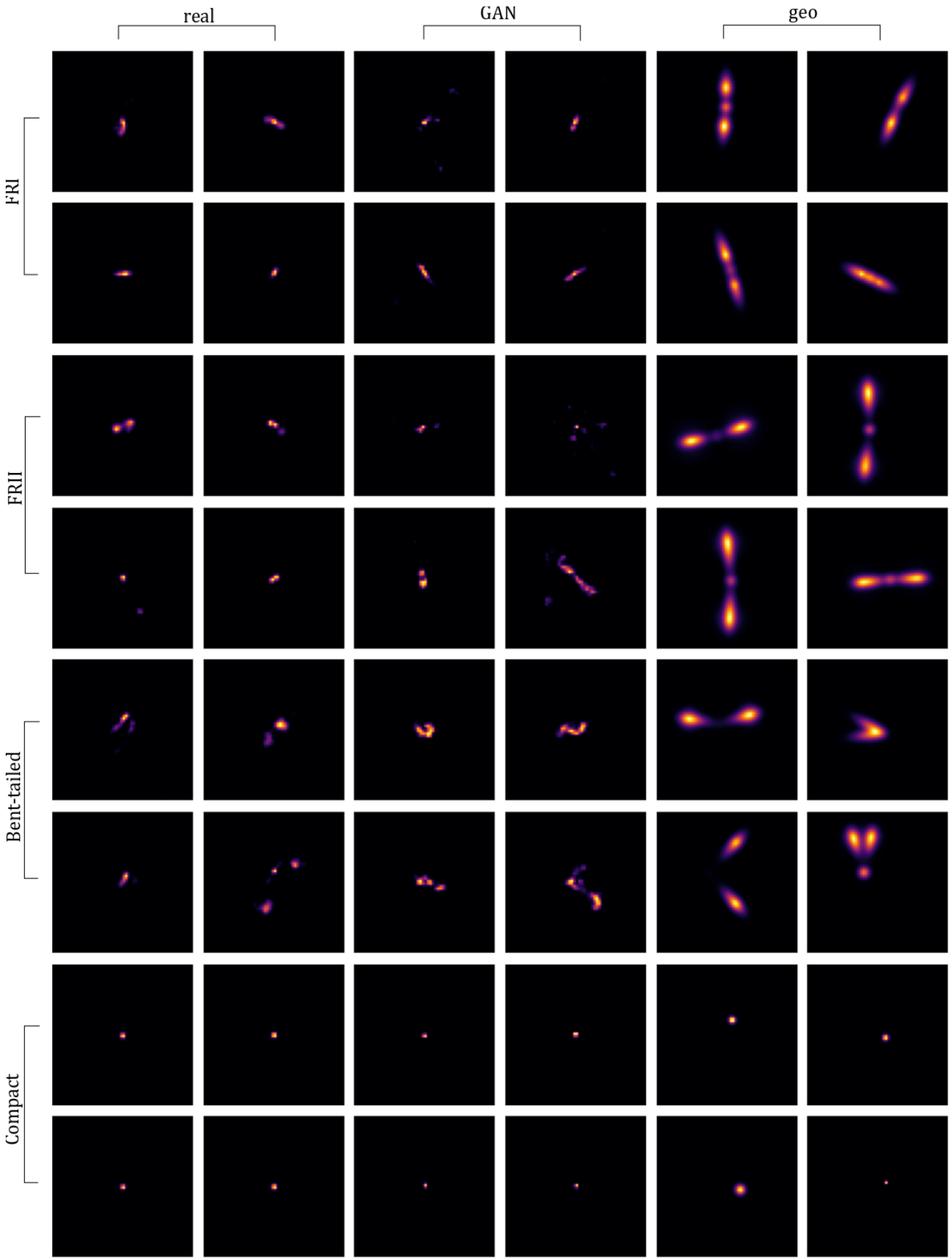


**Figure C1.** Random example images of real data, GAN generated data and geometric simulation generated data, for each classification.