Deep learning-based imaging in radio interferometry

K. Schmidt1*, F. Geyer1**, S. Fröse1, P.-S. Blomenkamp1, M. Brüggen2, F. de Gasperin2,3, D. Elsässer1 and W. Rhode1

1 Lehrstuhl für Experimentelle Physik V, TU Dortmund University, Otto-Hahn-Straße 4a, 44227 Dortmund, Germany
2 Hamburger Sternwarte, University of Hamburg, Gojenbergsweg 112, 21029 Hamburg, Germany
3 INAF - Istituto di Radioastronomia, via P. Gobetti 101, 40129 Bologna, Italy

Received ; accepted

ABSTRACT

Context. The sparse layouts of radio interferometers result in an incomplete sampling of the sky in Fourier space which leads to artifacts in the reconstructed images. Cleaning these systematic effects is essential for the scientific use of radio-interferometric images. Aims. Established reconstruction methods are often time-consuming, require expert knowledge, and suffer from a lack of reproducibility. We have developed a prototype deep learning-based method that generates reproducible images in an expedient fashion. Methods. To this end, we take advantage of the efficiency of convolutional neural networks to reconstruct image data from incomplete information in Fourier space. The neural network architecture is inspired by super-resolution models that utilize residual blocks. Using simulated data of radio galaxies that are composed of Gaussian components, we trained deep learning models whose reconstruction capability is quantified using various measures. Results. The reconstruction performance is evaluated on clean and noisy input data by comparing the resulting predictions with the true source images. We find that source angles and sizes are well reproduced, while the recovered fluxes show substantial scatter, albeit not worse than existing methods without fine-tuning. Finally, we propose more advanced approaches using deep learning that include uncertainty estimates and a concept to analyze larger images.


1. Introduction

With radio interferometry, it is possible to obtain images of radio sources with angular resolutions of up to milliarcseconds (Reinhard et al. 2010). Achieving such high angular resolutions was made possible by the advent of very long baseline interferometry (VLBI) (Broten et al. 1967), which exploits large distances between two telescopes. Most current radio telescopes are radio interferometers, such as the Very Long Array (VLA), Low Frequency Array (LOFAR), or MeerKAT. Furthermore, data from radio interferometers play a decisive role in multiwavelength studies (Linhoff et al. 2020; Ghirlanda et al. 2019).

Radio interferometers record information about the sky in Fourier space, also called visibility space. The relation between the measurement and the specific brightness distribution of the source is described by the van Cittert-Zernike theorem (Ostrovsky et al. 2009; Thompson et al. 2001c), which states that the two-point correlation function of the electric field measured by two antennas of a radio interferometer is the Fourier-transformed intensity distribution of the source. As the number of antennas in a radio interferometer array is limited, the sampled Fourier space always remains incomplete. By applying the inverse Fourier transformation to the data, artifacts dominate the reconstructed image. For this reason, data first need to be "cleaned" by the astronomer to use them for scientific analyses (Clark 1980; Thompson et al. 2001a).

Established cleaning software such as DIFMAP (Shepherd et al. 1994) or the Common Astronomy Software Application (CASA) (Jaeger 2008) partly require human intervention during the analysis, for example marking the area of the sky where emission is expected ("masking"). This process is iterative, slow, and generates nonreproducible results because the reconstructed images depend on the user’s experience. More recent approaches such as w-stacking clean (WSCLEAN) (Offringa et al. 2014) perform the imaging faster and with a higher degree of automation. Nevertheless, the analysis is still time-consuming since several parameters have to be adjusted in iterative cleaning runs in order to determine the best-suited parameter set for the given data quality. With increasing data rates of modern radio interferometers such as LOFAR (van Haarlem et al. 2013) and the Square Kilometre Array (SKA) (Oberoi et al. 2017), fast solutions are necessary to analyze observations on reasonable timescales.

With increasing computing power, deep learning-based analysis strategies become more widely used in astronomy and astroparticle physics. Neural networks are commonly used because they are successful in other domains, their application is reasonably fast, and they generate reproducible results. First attempts of application to data from radio interferometers have already been tried in Morningstar et al. (2018) and Morningstar et al. (2019). However, they often remain black boxes and their output is not always easy to interpret.

In this work, we use convolutional neural networks (CNNs) because these networks have been proven to be efficient tools for image tasks. More precisely, we propose a CNN built from elements used in the context of super-resolution applications. Super-resolution networks have the purpose of converting low-resolution images into high-resolution versions by upsampling...
Specific Intensity / a.u.

0.00
0.05
0.10
0.15
0.20
Specific Intensity / a.u.

0.00
0.05
0.10
0.15
0.20

Fig. 1. Image of a radio galaxy from the simulated data set in image space. The source is composed of two-dimensional Gaussian distributions, which are blurred with a Gaussian kernel.

and reconstructing the fine-scale structures (Yang et al. 2019). These networks use the available information and enhance it to perform their assignments. This is possible because the convolution kernels use values of the neighboring pixels to determine a value for the pixel to be estimated. This way, the missing information can be reconstructed and corrupted pixel values corrected. When analyzing data from radio interferometers, we face a similar problem since the visibility space is incompletely sampled. This procedure is comparable to the reconstruction of additional pixels in the upsampling process. Thus, we aim to solve the reconstruction problem with an architecture inspired by Ledig et al. (2016).

We show that our approach offers a fast, reproducible way to generate clean radio images by evaluating the reconstruction quality using simulated data. We developed a radionets framework (Schmidt et al. 2019) that is available for the astronomical community as an open-source package. We have provided a helpful tool to speed up the imaging process in radio interferometry, which does not rely on user input during the training and the application process.

Finally, we propose more advanced strategies to improve the simulations and the cleaning of the data. We upgrade our simulations by including point sources in the images creating more complex data sets. Moreover, we present ideas to how to deal with noise and larger images. By adding uncertainty estimates for the reconstructed source images, one can attain more meaningful results. Moreover, source finders based on deep learning networks can help to speed up the imaging process in general.

Sec. 2 motivates and describes the simulations that were used to create the training data. The architecture and parameters for the training of the neural networks are laid out in Sec. 3. In the next section, the results are evaluated. In Sec. 4 we compare a network trained on clean and a network trained on noisy input data to the results of the established cleaning method wsclean. Sec. 5 lists additional approaches and ideas for further analysis. In Sec. 7 we present our conclusions.

Fig. 2. Amplitude (top) and phase (bottom) of the simulated radio galaxy. In the first step, the representation of the radio galaxy in Fourier space was calculated by Fourier transform. Then the complex values were transformed to amplitude and phase utilizing Euler’s formula. To mimic an observation with a radio interferometer, the images were sampled using the (u, v) coverage of a simulated observation with the VLBA. The remaining information is displayed in color, all dropped information is grayed out.

2. Simulations

In order to train deep learning models, we created synthetic data with Monte-Carlo simulations of known ground truths (Hastie et al. 2009). Since the creation of realistic simulations is a time consuming and elaborate process, we start with simple simulations of radio galaxies. The goal is to extend and improve the simulations within the ongoing process of this project, see subsection 5.1.

2.1. Extended radio galaxies

One of the targets of VLBI observations are radio galaxies, which are a subclass of active galactic nuclei. Radio galaxies typically consist of a bright core and two jets that emerge from the central black hole region (Bridle & Cohen 2012). Because of relativistic boosting, these jets can also appear one-sided, depending on the angle of observation (Urry & Padovani 1995).
According to the model of Blandford and Königl (Blandford & Königl 1979; Königl 1980), active galactic nuclei host narrow conical jets fueled by continuous plasma inflows. Using these so-called jet components for the source representation is a common simplification in the analysis of VLBI data, see [Lister et al. 2013, 2019], for example. Hence, we simulate radio galaxies by two-dimensional Gaussian distributions. Fig. 1 shows an example of such a source where the brightest component, called the core component, is always placed in the center of the image. For variations within the data set, the following source parameters are randomly drawn: First, the number of one-sided jet components was chosen between 3 and 6. This ensures that the jet can reach the edge of the image, but not beyond. An angle for the jet orientation was randomly drawn from a range between 0° and 360°. A straight line starting from the core component and moving toward the edge of the image was used to place the centers of the jet components. The distance between two jet components was fixed to a value of 5 pixels. Furthermore, the components were initialized with random amplitudes. The core components’ peak amplitudes range between $10^{-3}$ and $10^{4}$ and were drawn randomly from a uniform distribution. Jet components’ peak amplitudes were logarithmically reduced by calculating $A_{\text{core}}/\exp(n_{\text{comp}})$. Additionally, we used randomized standard deviations with increasing $\sigma_x$ and $\sigma_y$ for components further outside the jet. In the case of a two-sided jet, the selected jet components were mirrored at the core component, leading to a total number of components between 4 and 13.

### 2.2. Observations with radio interferometers

Radio interferometers measure complex values in $(u, v)$ space, which offers the possibility to reconstruct the brightness distribution of the source using the inverse Fourier transform (Thompson et al. 2011). Using Euler’s formula (Needham 1999) to transform complex data into amplitudes and phases limits the ranges in the data, which helps in training deep learning models. For illustration, Fig. 2 shows amplitudes and phases for an example of a simulated radio galaxy.

Limited by the antenna configuration, the entire $(u, v)$ plane is never completely sampled. In the case of the VLBA, the ten antennas lead to 90 baselines (Napier et al. 1994). Each baseline provides one complex value per snapshot. Utilizing the Earth’s rotation helps to fill the $(u, v)$ plane as densely as possible (Taylor et al. 1999). Before correlating the data, one has to compensate for the arrival time difference between individual antennas caused by the geometric layout of the interferometer. This is done by defining a reference point on a plane perpendicular to the source direction. All arrival times of the signals are adjusted to form a two-dimensional plane containing all antennas. Owing to the Earth’s rotation, baselines change with respect to the source direction, which in turn provides additional data points in $(u, v)$ space.

Fig. 3 shows the $(u, v)$ coverage of a simulated single channel observation in which large areas of the $(u, v)$ space remain unsampled. We created such sampling masks for the complete $(u, v)$ plane to simulate the incomplete coverage by the radio interferometer. Thus, data sets can be created with a variety of $(u, v)$ coverages. It is straightforward to change different observation parameters such as the starting hour angle of the source and the length of the observation. The maximal baselines correspond to resolutions of approximately 2 pixels and the smallest baselines allow to resolve scales on the order of approximately 8 pixels. We note that the scales given here represent values in units of the image size of 64 pixels and do not refer to absolute angles. In the simulation of the $(u, v)$ space, instrumental effects of the radio interferometer were ignored. The resulting $(u, v)$ data are comparable to already gridded visibility data $V_{\text{noiseless}}$. In the following, we refer to frequels as gridded data points in Fourier space, in the same way as pixels are used to describe the source distributions in image space. Simulations of un-gridded data that include antenna characteristics can be performed using the RIME formalism (Smirnov 2011), which is discussed in subsection 5.1.

Fig. 2 shows the amplitude and phase distribution for the example source, where all information lost in the sampling process are grayed out and set to zero for the next data processing steps. This example illustrates the underlying problem of reconstructing missing radio interferometric data. If these incomplete $(u, v)$ spaces are used to reconstruct the source’s brightness distribution by inverse Fourier transformation, so-called “dirty images” dominated by artifacts are generated illustrated in Fig. 4. The goal of our convolutional neural network is to clean up these dirty images or to reconstruct the incomplete data before applying the inverse Fourier transformation. The colored parts of the $(u, v)$ space shown in Fig. 2 are the input data for the neural networks, and the complete Fourier planes, as shown in gray, represent the target images.

#### 2.3. Uncorrelated noise

In order to mimic noise, we smeared the pixel values of the simulated radio galaxies in image space by adding offsets for every pixel. These offsets are randomly drawn numbers from a standard normal distribution:

$$g(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

(1)
Fig. 4. Result of the inverse Fourier transformation applied to the sampled noiseless visibilities (left) and to the noisy visibilities with additional white noise (right), called “dirty images”. The appearance of background artifacts is visible in both cases. Compared to the true brightness distribution shown in Fig. 1, the brightness is underestimated. These reconstruction errors are caused by incomplete data in Fourier space.

Fig. 5. Image of a radio galaxy from the simulated data set in image space with added noise corruption $I_{\text{noisy}}$. The source is composed of two-dimensional Gaussian distributions, which are blurred with a Gaussian kernel.

Before adding the random noise, the values got scaled by a factor of 5% of the peak source flux resulting in noisy images given by

$$I_{\text{noisy}}(l,m) = I(l,m) + I_{\text{max}} \cdot g(x \mid \mu, \sigma)_{lm}.$$  \hspace{1cm} (2)

Here, $I(l,m)$ represents the image intensity at pixel $(l,m)$, $I_{\text{max}}$ stands for the peak intensity of the image, and $g(x \mid \mu, \sigma)_{lm}$ for a new randomly drawn number at pixel coordinate $(l,m)$ with $\mu = 0$ and $\sigma = 1$. This procedure results in corrupted Fourier data $V_{\text{noisy}}$ when calculating the Fourier transform. Fig. 5 shows an example of a simulated radio galaxy with added noise.

To increase the complexity of our data and to simulate possible measurement effects in $(u,v)$ space, we added additional white noise to the visibilities. The white noise was produced by drawing random values from a Gaussian distribution with zero mean and a standard deviation of 1 resulting in corrupted visibilities

$$V_{\text{noisy + white noise}}(u,v) = V_{\text{noisy}}(u,v) + g(x \mid \mu, \sigma)_{uv}.$$ \hspace{1cm} (3)

Again, $g(x \mid \mu, \sigma)_{uv}$ is a new random number for every frequency with $\mu = 0$ and $\sigma = 1$. Fig. 4 illustrates the dirty image created from the noisy visibilities with additional white noise on the right. Additional artifacts caused by the noise corruption are visible. In subsubsection 4.1.2, $V_{\text{noisy}}$ and $V_{\text{noisy + white noise}}$ are both used separately as input data for our neural network and the reconstruction results are compared. This noise is uncorrelated while real observations can suffer from substantial correlated noise. In the future, we plan to consider also correlated noise as described in subsection 5.1.

3. Model training

In this section, we describe our neural network. Specifically, we illustrate the architecture and its components, describe the input data and the data augmentation applied to it, explain our loss function and present the optimizer function, which is used for the minimization process.

3.1. Architecture

The convolutional layers exploit the spatial correlation in images by passing a kernel with specified weights over the image. For most applications, the result is a down-sampled version of the input image, which contains some features extracted from the original image. In our case, we chose the parameters of our convolutional layers such that the image size does not change. Thus we use the available information from the sampled amplitude and phase maps, which serve as our input images, to reconstruct values for the missing information. This idea is sketched in Fig. 6.

In super-resolution applications, high-resolution images are produced from low-resolution input images by up-sampling and reconstructing the fine-scale structures. Using a combination of
Schmidt et al.: Deep learning-based imaging in radio interferometry

**Residual Block (×8)**

**Fig. 6.** Schematic representation of the architecture employed in this paper. Sampled amplitude and phase distributions serve as input. After a preconvolution followed by a PRelu activation (He et al. 2015b), the main part of the architecture is composed of eight residual blocks (He et al. 2015a; Gross & Wilber 2016). One block consists of two convolution batch norm pairs with a PRelu activation in between. The input data is bypassed and added element-wise to the output for every residual block. The last residual block is followed by a convolution batch norm pair. Another skip connection is placed around the main part of the architecture. After a final convolution the reconstructed amplitude and phase distributions are the output of the architecture. Diagram reproduced from (Ledig et al. 2016).

**Table 1.** Overview of the used architecture

<table>
<thead>
<tr>
<th>stage</th>
<th>layer</th>
<th>channel in</th>
<th>channel out</th>
<th>kernel size</th>
<th>stride</th>
<th>padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre Block</td>
<td>Convolution</td>
<td>2</td>
<td>64</td>
<td>(9 × 9)</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>PReLU</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Residual Block (×8)</td>
<td>Convolution</td>
<td>64</td>
<td>64</td>
<td>(3 × 3)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Batch Norm</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PReLU</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Convolution</td>
<td>64</td>
<td>64</td>
<td>(3 × 3)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Batch Norm</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Elementwise Sum</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Post Block</td>
<td>Convolution</td>
<td>64</td>
<td>64</td>
<td>(3 × 3)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Batch Norm</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Elementwise Sum</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Final Block</td>
<td>Convolution</td>
<td>64</td>
<td>2</td>
<td>(9 × 9)</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: For each layer, the number of input and output channels is given. In the case of convolutional layers, the kernel size, stride, and padding are specified.

Convolutional layers, a pixel value is determined based on the values of the neighboring pixels. Applying this concept to the \((u, v)\) plane, we built a network following Ledig et al. (2016) using the residual block layout investigated by He et al. (2015a) and adapted by Gross & Wilber (2016). In this setup, every residual block consists of five operations. In the first step, the input data passes through a convolutional layer with 64 input and 64 output channels, a kernel size of \((3 × 3)\) pixels, a stride of 1, and a padding of 1. Kernel size refers to the extent of the kernel which is used to scan the image. The stride is the number of pixels, which lie between the centers of two individual convolutions. Furthermore, padding is used to add pixels to the edges of the images, which are filled with zeros to keep the same size for the output image. Note that with these settings the image size remains the same as the input image size. The second and third operation consist of batch normalization and a parametric rectified linear unit (PReLU) (He et al. 2015b) as nonlinearity. Here, the negative part of the function is not constant, but its coefficient is learned by the network. The block ends with another convolution with the same settings as described above and an additional batch norm layer. In parallel, the input data is bypassed through a skip connection and added element-wise to the output of the residual block. This changes the underlying mapping function \(F\) which the neural network has to learn to map the input \(x\) to the output \(y\):

\[
F(x) = y - x.
\] (4)

Here, the network must only learn to map the difference between input and output, meaning that each block predicts only a residual. Predicting only the residual led to a faster convergence of the network and made it more robust to outliers. Fig. 6 illustrates the layout of the residual blocks, which is highlighted by the gray box, as well as the complete architecture. In the following, we describe the properties of the different stages.

The neural network starts with a preconvolution taking two input channels and extending them to 64 output channels with a \((9 \times 9)\)-kernel. Using a padding of 4 keeps the image size the same. For this convolutional layer, we are using a group setting, which divides the parameters into two parts. Thus, the output results in 32 channels dedicated to the amplitude, and 32 for the phase, both with individual filter weights. The central part of the architecture consists of 8 residual blocks. This part follows for...
an additional post-block with a (3×3) convolution and a batch norm layer. The chosen settings keep the channel and image size the same. An additional skip connection between the preconvolution and the final convolution accelerates the convergence of the network since again only the residual has to be learned. The final convolutional layer has the same settings as the preconvolution. This time, it takes the 64 channels as input and makes two channels out of it. Again, 32 channels are dedicated to the amplitude and 32 channels to the phase by enabling the parameter grouping. A summary of the architecture including the settings for the different layers can be found in Tab. 1.

### 3.2. Training

For the network training, we used a data set that was simulated using the procedure described in Sect. 2. For every simulated source in image space, we generated a different observation resulting in a new sampling mask for the corresponding (u, v) space. This approach created variation in the data set and served as a form of data augmentation. More importantly, the use of different sampling masks allowed for a more realistic training process. Furthermore, our experience showed that when using different sampling masks, the loss drops more than when using a static mask. As stated in subsection 2.2, we transformed the complex data into amplitude and phase maps using Euler’s formula. In contrast to the real part, the amplitude distribution only has positive values. Furthermore, the transformation of the imaginary part had a big impact on the training. The phase distribution is limited to a range from −π to π, which is an enormous reduction of the parameter range to be learned, since the imaginary part distribution would span several orders of magnitude for the entire data set. This resulted in smoother training processes, faster convergence of the models, and improved fine-scale reconstructions. In general, the limitation of the parameter space has been proven to be a great advantage when training neural networks. The properties of the different data sets are as follows: 50 000 training amplitude and phase maps, 10 000 validation amplitude and phase maps, and 10 000 test amplitude and phase maps. In the input data, frequels with missing information were set to a value of 0 resulting in a range between 73% and 82% of all input frequels, which varies depending on the simulated observation. During the training process we used data augmentation to prevent an overfitting of the network. In the batch creation step, input and target maps were rotated by a random multiple of 90 degrees which further increased the number of individual training images.

Contrary to most other approaches, we trained our network in Fourier space. The neural network was used to reconstruct missing data in the visibility space, such that a clean image can be generated using the Fourier transformation afterwards. The convolution filters used information from sampled frequels to calculate values for neighboring frequels with missing information, which were marked with zero values. This procedure is visualized in Fig. 7. The advantage of marking frequels with missing information with zeros is that these frequels do not distort the result of the convolutional layers. At the same time, zero pixels have a large offset to the target value of the simulated true amplitude and phase maps, which causes large losses. As a result, the network learned to reconstruct these pixels in a prioritized manner, as good reconstructions rapidly reduced the calculated loss. Step by step, the convolutional layers could use existing and newly filled information to calculate values for all frequels. In this way, it is possible to fill the complete amplitude and phase maps with continuous depth of the architecture. A direct advantage of this approach is that it is not necessary to switch between Fourier space and image space, as is the case between every iteration when using the CLEAN algorithm. In our case, no flux components were extracted from the dirty image to create a model of the radio source, but the information in Fourier space was used directly to reconstruct the missing information. Fur-

Fig. 7. Schematic illustration how our neural network reconstructs incomplete amplitude and phase maps. Frequels with missing information have values of zero. Convolutions inside the architecture use information of neighboring frequels to calculate a new value for the center frequel. A convolution with a (9×9) kernel is shown on the left (dark-blue). An input amplitude map is shown in the middle. The part of the map used for the convolution example is marked with dark-blue. The part marked in light-blue is used to demonstrate the reconstruction progress after every stage. The amplitude map is reconstructed up to the edges when it has passed through the Post Block. For simplicity, only the influence of the convolutions is taken into account. A re-scaling of the output of the different stages was performed for better visualization.
thermore, it was not necessary to convolve the reconstructed images with a telescope beam, as we did not perform a point source extraction. Another advantage of reconstructing in Fourier space is the ability to transfer knowledge gained on smaller amplitude and phase maps to be able to reconstruct larger ones with little additional effort, see subsection 5.2. This is possible because the field of view of the clean image is directly related to the spacing of the samples in Fourier space. Finally, working in Fourier space allows for an estimation of an uncertainty in the reconstructed values. This feature will be available in a future version of our network. The quantification of uncertainties also makes it possible to compute an uncertainty map for the clean image of the radio source. First tests have already been performed, see subsection 6.2 for details.

First, we trained a network with noiseless input data \( V_{\text{noiseless}} \) and then we investigated the influence of noise. To this end, we trained another network with noisy input data \( V_{\text{noisy}} \), as described in subsection 2.3. Both networks were trained for 300 epochs with a batch size of 64 and a learning rate of \( 2 \times 10^{-4} \). This learning rate selection followed the evaluation of the result from the learning rate finder created by Howard & Gugger (2020). For the loss function, we used an adapted L1 loss

\[
\text{Loss} = \text{L1} \left( x_{\text{amp}}, y_{\text{amp}} \right) + \text{L1} \left( \text{HardTanh}(x_{\text{phase}}), y_{\text{phase}} \right),
\]

with \( \text{L1}(x, y) = | x - y | \),

\[
\text{HardTanh}(x) = \begin{cases} 
\pi, & \text{if } x > \pi \\
-\pi, & \text{if } x < -\pi \\
0, & \text{otherwise}
\end{cases}
\]

where \( x \) is the predicted output from the network and \( y \) is the true image. The HardTanh\((x)\) function restricted the prediction for the phase to a parameter range between \( -\pi \) and \( \pi \), which is illustrated in Fig. 8. This avoided problems that can arise from the phase’s periodic nature when reconstructing the values. We chose the L1 loss over the MSE loss because our experiments showed that the reconstruction of fine details and small scales were less accurate when using the MSE loss. In our case, the slightly adapted L1 loss as shown in Eq. 5 led to improvements in the reconstructed \( (u, v) \) spaces which in turn resulted in more detailed and cleaner reconstructed source images.

We used the ADAM optimizer (Kingma & Ba 2017) to update the weights during training, which outperformed stochastic gradient decent (SGD) (Amari 1993) in convergence time at the cost of more readily learned parameters.

Our training time was about 170 seconds for one epoch on computer specifications described in Tab. C.1. The complete training of 300 epochs thus took just over 14 hours. The application time of the trained neural network was much lower, on the order of milliseconds per image and will be further discussed in Sect. 4.

The loss curves shown in Fig. 9 illustrate the learning process of the network without noise (top) and with added noise (bottom). After a sharp drop at the beginning of the training, a period with more spikes in the validation loss occurs. However, this has no negative impact on the reconstruction quality of the final network since the training process continues smoothly for the last 100 epochs. Both curves show a similar behavior for the individual training sessions, meaning that the training converges well even for noisy input data.

4. Model evaluation

In this section, we test the ability of the trained models to reconstruct incomplete \((u, v)\) data. We apply the models described in Sect. 3 to images from a dedicated test data set. Next, we calculate the deviation of reconstructed amplitude and phase from the true distributions. Then we compare the reconstructed source images, which are generated by the inverse Fourier transformation, with the true images. Finally, we investigate the reconstructed jet.
angles, source areas, and the specific intensity of the core components are evaluated.

4.1. Reconstruction of Fourier data and brightness distributions

The deep learning models allow for the reconstruction of missing information in incomplete Fourier data. Their execution time is on the order of milliseconds per image, which, compared to cleaning algorithms implemented in standard software such as DIFMAP or WSCLEAN, constitutes a significant speed advantage.

For a first assessment of the reconstructions, we compare the reconstructed input data to the true distributions. By calculating the difference, we can identify areas with reconstruction problems. We summarize the results in Fig. 10. In these figures, the first row shows the amplitude and the second row the phase. We use the predicted (u, v) spaces to create reconstructed source images, called clean images, by applying the inverse Fourier transformation. The comparison with the simulated brightness distribution provides a first indication of the quality of the reconstruction and helps to identify areas with reconstruction problems. Finally, we calculate the jet angle, \( \alpha \), using a principal component analysis (PCA). In PCA, a new basis is searched within the data. The goal is to maximize the information contained when projecting onto this basis. In the case of radio galaxy images, the axis through the jet forms a basis which can then be used to determine the jet angle. This is accomplished by computing the covariance matrix of the image and determining the eigenvalues and vectors of this matrix. The searched angle is then calculated by a tangent relation between the two eigenvectors.

Fig. 10. Exemplary reconstruction for the training session with noiseless input data \( V_{\text{noiseless}} \). Visualization of prediction (left), true distribution (middle) and the difference between both (right). Results are shown for amplitude (top) and phase (bottom).

Fig. 11. Source reconstruction with the predicted amplitude and phase distributions for noiseless input data \( V_{\text{noiseless}} \). Resulting clean image (left), simulated brightness distribution (middle) and difference between both (right). The jet angle \( \alpha \), which was calculated using a PCA, is given for both source images.
Fig. 12. Exemplary reconstruction for the training session with noisy input data $V_{\text{noisy}}$. Visualization of prediction (left), true distribution (middle) and the difference between both (right). Results are shown for amplitude (top) and phase (bottom).

Fig. 13. Source reconstruction with the predicted amplitude and phase distributions for noisy input data $V_{\text{noisy}}$. Resulting clean image (left), simulated brightness distribution (middle) and difference between both (right). The jet angle $\alpha$, which was calculated using a PCA, is given for both source images.

To investigate the model’s ability to handle noise, we evaluated the results of two different training sessions. The first session was trained with noiseless input data, the second session was trained with noisy data. To compare our model to established imaging software, we performed the analysis of a dedicated data set using wsclean. In the following, we present results produced with the example source shown in Sect. 2.

4.1.1. Noiseless input data

Fig. 10 shows the reconstructed amplitude and phase for a training session with noiseless input data and different sampling masks. Overall, there is good agreement between prediction and truth for, both, amplitude and phase, except for small areas in the central part of the amplitude map. The trained model even enables the reconstruction of small-scale structures. The deviations, that is to say the difference between prediction and truth, confirm this. The agreement between prediction and truth can also be seen in the clean image in Fig. 11. At first glance, one can hardly detect any differences between estimated and true brightness distribution. Calculating the difference between both reveals a small underestimation of the specific intensity in the source region. Additionally, artificial structures parallel to the central source region appear. The good reconstruction of amplitude and phase leads to differences in the two brightness distributions that are an order of magnitude smaller than the simulated specific intensity. The remaining background structures can be further constrained by comparing the contours of prediction and truth, visible in Fig. B.1. In this image, the red boundary contains all specific intensity above 10% of the source’s peak intensity. Since this boundary does not include any of the background structures, the missing specific intensity is smaller than 10% of...
the source’s peak intensity. The source area defined in this way is in good agreement with the true source area, with a ratio of 0.98. The jet angles for prediction and truth are also consistent as their deviation is 0.23°.

4.1.2. Noisy input data

Fig. 12 shows the prediction for the same example described above, but with noisy input data. It is evident that the results do not differ much from the results with noiseless input data. The predicted amplitude misses some more nuances in the center of the image. The reconstruction of the small-scale structures still works reliably. For the phase, deviations between prediction and truth become larger in the entire image. Therefore, noisy input data leads to losses in the reconstruction of the small-scale structures. Still, the differences between both distributions are relatively small. This result can be seen in the reconstructed clean image shown in Fig. 13. Again, a visual comparison between reconstructed and true brightness distribution does not show large deviations. The calculation of the difference reveals a small underestimation of the specific intensity in the central part of the source. The deviations are about an order of magnitude smaller than the simulated specific intensities. The agreement between prediction and truth is also evident in Fig. B.2. The outer red boundary shows that the background structures make up less than 10% of the source’s peak intensity. Not only are the images well reconstructed, but noise added to the input data is cleaned up in the reconstruction process. The calculated jet angles for prediction and truth are also consistent as their deviation is 1.37°.

To further increase the complexity of our data and to simulate possible measurement effects in (\(u, v\)) space, we added additional white noise directly to the visibilities. For this, random values were drawn from a Gaussian distribution with a mean of 0 and a standard deviation of 0.05. Afterwards, the random sample was added to the real and imaginary part of the data in Fourier space to create noise corrupted visibilities \(V_{\text{noisy}}\). As described in subsection 2.3, due to the characteristics of the Fourier transformation, both types of noise result in distinct artifacts in the dirty images, see Fig. 4. The white noise in \((u,v)\) space leads to a sensitivity limit below which the information about the source area is lost. With the selected settings, this sensitivity limit is a specific intensity of \(5.47 \times 10^{-7}\).

4.1.3. Processing with \texttt{wsclean}

Table 2. Overview of the parameter settings utilized to create the clean images using \texttt{wsclean}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>64 pixels</td>
</tr>
<tr>
<td>scale</td>
<td>0.39 masec</td>
</tr>
<tr>
<td>mgain</td>
<td>0.3</td>
</tr>
<tr>
<td>gain</td>
<td>0.005</td>
</tr>
<tr>
<td>niter</td>
<td>50 000</td>
</tr>
</tbody>
</table>

In order to compare our reconstructions against data that have been cleaned using established imaging methods, we created a stack of clean images using \texttt{wsclean}. To this end, we used the same simulated brightness distributions, as described in Sect. 2. As \texttt{wsclean} uses measurement sets as input format, the second part of the simulation had to be adjusted. Firstly, we could no longer use a sampling mask to mimic an observation since ungridded visibilities are needed as input data. To calculate these visibilities for each baseline, we used a simple RIME formulation like the one described in Smirnov (2011). Here, we only applied the direction-independent phase delay term for a better comparability with the input data for the neural network models. The phase delay term corresponds to the Fourier kernel and does not introduce any further noise corruption. Thus, the following equation for determining the uncorrupted visibility of the antenna pair \(pq\) is obtained:

\[
V_{pq} = \sum_{m} K_p(l,m)B(l,m)K^*_q(l,m),
\]

with the source brightness \(B(l,m)\) and the phase delay Jones matrix \(K(l,m) = \exp(-2\pi i(u + vm))\). Here, \(u\) and \(v\) describe the coordinates of the current baseline corresponding to antenna pair \((p,q)\) in direction cosines. The \(\pi\)-term is neglected as \(\sqrt{1 - \xi^2 - m^2} \approx 1\) is valid in our case. Furthermore, \(H\) denotes the conjugate transpose operation. In order to ensure that the intensities of the visibilities are comparable with the flux density of the simulated bright distribution, the summed flux density of the images was normalized to one.

In the next step, the simulated complex visibilities were written to FITS files. Afterward, we used the \texttt{casa importuvfits} task to convert to measurement set format. The obtained data was fed to \texttt{wsclean} to create clean images utilizing the cleaning parameters summarized in Tab. 2.

Fig. 14 shows a clean image of a source with an one-sided jet generated using \texttt{wsclean} (left) and the corresponding simulated brightness distribution of this source convolved with the clean beam calculated by \texttt{wsclean} (middle). The comparison of both images shows that the general structure of the source is well reconstructed in the clean image. However, in this example the flux density is slightly overestimated as the reconstructed source is more blurred. Especially in the core region this becomes obvious. In addition to the difference between the simulated and the reconstructed viewing angle, the reconstruction of the individual jet components works well in general. For reference, the reconstruction of our deep learning approach is shown on the right, which for a better comparison is convolved with the clean beam calculated via \texttt{wsclean}. Also in the case of two-sided jets \texttt{wsclean} is able to reconstruct individual components.

The example in Fig. 15 illustrates that the jet structure in the clean map (left) is reconstructed well. However, there is a difference between the simulated and the reconstructed jet angle. Additionally, faint background artifacts are visible, which occur more frequently in the case of two-sided jets. This is due to the fact that it is impossible to find a set of cleaning parameters for 10 000 sources that gives optimal results for all sources. The results show that, in the case of one-sided sources, the used parameter set leads to more accurate results. Furthermore, the flux density of the reconstructed source is slightly underestimated. This can be caused by several reasons: First, the cleaning might not have been performed deeply enough. In this case, tuning of cleaning parameters can be a possible solution. Secondly, the difference in flux density may be caused by the large information loss during sampling. Especially in the central region of the \((u,v)\) plane, many data points are missing. These data points contain a lot of information about the large-scale brightness distribution of the source image.

4.2. Advanced evaluation methods

In this section, we compare the reconstructions of the deep learning models with results using \texttt{wsclean}. For deep learning, the
Fig. 14. Clean image of a source with an one-sided jet created using wsclean (left) and simulated brightness distribution of the source convolved with the clean beam calculated via wsclean (middle). For comparison, a reconstruction of the same source using our deep learning approach trained with noiseless input data is shown (right), which is also convolved with the clean beam calculated via wsclean.

The full-width half-maximum clean beam sizes are shown on the lower left.

Fig. 15. Clean image of a source with a two-sided jet created using wsclean (left) and simulated brightness distribution of the source convolved with the clean beam calculated via wsclean (middle). For comparison, a reconstruction of the same source using our deep learning approach trained with noiseless input data is shown (right), which is also convolved with the clean beam calculated via wsclean.

The full-width half-maximum clean beam sizes are shown on the lower left.

Fig. 16. Histogram of the source area ratios between prediction and truth. Results for a deep learning model trained with noiseless (orange), one trained with noisy input data (blue) and clean images generated with wsclean (green) are displayed. It becomes clear, that the distributions match well, which is supported by the mean and the standard deviation. In the case of wsclean a small overestimation of the area is visible.

Fig. 17. Histogram of the jet offsets for a deep learning model trained with noiseless (orange), one trained with noisy input data (blue) and clean images generated with wsclean (green). The offset range is capped from -25° to 25° for visibility reasons. Only small differences are present between the two distributions obtained from the deep learning models. Reconstructed jet orientations fit the true values well, which is supported by the means and standard deviations. Small pixel offsets in the cleaning process can already cause large offsets of the jet angle. This is a possible explanation for the larger deviations occurring in the case of wsclean.
transform. In the case of wsclean, the visibilities stored in measurements sets were used to create the clean images. Thus, two dedicated test data sets were created each consisting of 10 000 images.

In order to compare the reconstructed and true source areas, we set an outer source boundary of 10% of the source’s peak intensity. Afterwards, we measured the source areas using Leibniz’ Sector formula (Walter 1990), which provides a relation between the path integral and the area of an enclosed region. The enclosure of the source areas was obtained by calculating the contour levels of the source using matplotlib. Then, the path integral was numerically approximated. The resulting values were used to calculate the source areas with the help of the Leibniz’ Sector formula. Then, we computed the area ratio between the reconstructed and the true brightness distributions. In the ideal case, the area ratio is one. Smaller values correspond to an underestimate of the reconstructed source area. Larger predicted source areas lead to ratios above one. With this method, we obtain an estimate of the number of background artifacts. Images with reconstruction errors show more background artifacts and less specific intensity in the source region at the same time. Therefore, they have area ratios below one. Fig. 16 shows the area ratios for the complete test data set for the deep learning model trained with noiseless input data (orange), the deep learning model trained with noisy input data (blue), and for the images created with wsclean (green). All three distributions peak around the optimal value of one. A small deviation between predicted and true source areas is confirmed by the mean and standard deviation of \(0.964 \pm 0.117\) (noiseless) and \(0.992 \pm 0.128\) (noisy). The smearing to values above 1 on the right side of the peak in the case of wsclean indicates that the area of sources that can be reconstructed is overestimated. Some source distributions cannot be reconstructed by any of the models, which is represented by the peaks at a ratio of zero. These instances occur more frequently in the case of wsclean. This results in mean and standard deviations of \(0.869 \pm 0.372\) (wsclean). Both reasons lead to a standard deviation of the distribution obtained with wsclean that is 3 times larger than that of the deep learning results. For all distributions, the optimal value is within the uncertainty boundaries. Comparing the two deep learning models, the mean values for clean and noisy input data differ by only 0.028, which indicates that the architecture can reach the same reconstruction level for both input types.

In a second experiment, we investigated the reconstruction of the jet angles. In Fig. 17 three histograms containing the differences between the predicted and the true jet angles for 10 000 test images are shown. The orange histogram represents the deep learning model trained with noiseless input data, while the blue histogram represents the deep learning model trained with noisy input data. The results are similar, with the reconstruction using noisy input data faring slightly better, which is supported by the mean and standard deviation of \((-0.016 \pm 1.429\) (noiseless) and \((-0.014 \pm 1.333\) (noisy) for the distributions. In both cases, there are only slight deviations between the jet angles calculated from the reconstructed and the true images, which is shown by the small standard deviations. As one can see, the reconstruction with noisy input data performs slightly better than the one with noiseless input data. The distribution of the jet angle offsets obtained from the images cleaned using wsclean is shown in green. It is evident that there are only a few cases where a deviation of more than 15 degrees occurs. Nevertheless, two maxima appear at offsets of around \(-12^\circ\) and \(12^\circ\), which exceed the peak at zero in height. This may be caused by a systematic problem in the reconstructions with wsclean. Since in our analyses small image sizes are used, small deviations in the cleaned maps lead to large differences in the jet angle. A deviation of one pixel can already lead to an offset of several degrees. Additionally, we note that wsclean is not optimized for clean images of this size. The evaluation of the complete test set results in a mean jet offset of \((2.037 \pm 27.441)^\circ\) (wsclean). Where the large standard deviation is caused by the two maxima at \(-12^\circ\) and \(12^\circ\).

Finally, we compare the predicted and the true specific intensities of the core components. For this purpose, the Gaussian components of the simulated images were identified by the blob detection algorithm inside the scikit-image (van der Walt et al. 2014) package. Afterward, the pixels with the brightest specific intensities, in our case the core component, were averaged for both truth and prediction, and then compared. In this way, the relative deviation of the mean true specific intensity in the first component was computed. Fig. 18 visualizes the resulting values. The mean specific intensity deviation for the training session with noiseless input data (orange) and the training session with noisy input data (blue) is close to the optimal value of zero for both deep learning models. The standard deviation, however, is higher due to some outliers whose flux is over- or underestimated. In the case of wsclean, the mean reconstructed flux densities are underestimated to a larger extent. This is also reflected in the mean and standard deviation of the distribution which is \((-13.645 \pm 24.312)\%\) (wsclean). Again, reconstruction can be improved by tuning the cleaning parameters in wsclean for the different sources and we wish to reiterate that wsclean is not designed to be used with the same settings on 10 000 different samples.

In order to evaluate the performance of the model applied to data affected by white noise, all three methods were applied to the new data set. Mean and standard deviations of the resulting distributions are summarized in Tab. 3. The reconstruction of the source area and the reconstruction of the jet angle do not change remarkably. The reason that the source area does not change that much despite the above-mentioned sensitivity limit is that only around 30 of our 10 000 image test data set are below this limit. In the case of the mean intensity deviation of the core
component an increase of the mean and the standard deviation is visible. This happens due to the sensitivity limit resulting from the white noise in \((u, v)\) space. Sources with particularly weak brightness distributions can no longer be reconstructed and appear as outliers in the distributions, leading to increased standard deviations.

To test the dependence on the masks (i.e., \((u, v)\) coverage), we evaluated three data sets with different fillings for the masks on the model trained with noisy input data (see Sect. 4). The fillings were 20\%, 50\% and 70\%. To achieve such high sampling densities we added the option to simulate multichannel data. An example of a simulation with four channels is shown in Fig. 19. Here, the number of samples was increased by a factor of 4. This resulted in more information collected per frequency, which increased the sensitivity of the simulated observation as the signal to noise ratio was improved. Table 3 summarizes the reconstruction results which were evaluated with the methods introduced in this chapter. Mean and standard deviations of the distributions show that the success of the reconstruction is correlated with the filling of the mask. The evaluation of the jet offsets and the mean intensity deviation illustrates that the reconstruction quality is greatly affected by small sampling densities. The values for the source area ratios and their standard deviations improve with larger sampling rates, as expected. The behavior of the mean intensity deviations is opposite, as larger sampling densities lead to an overestimation of the specific intensity. This leads to larger values for mean and standard deviations. In future work, we will use a wider range of sampling rates in the training to further improve the robustness of our model.

To conclude, the sampling density is directly correlated with the model’s ability to reconstruct the characteristics of the simulated jet. This confirms the robustness of our model, which made use of data with a sampling density of around 30\%.

### Table 3. Overview of the mean results of the three evaluation methods with clean, noisy and noisy plus additional white noise input maps and for input maps with different mask fillings

<table>
<thead>
<tr>
<th>Jet offset [°]</th>
<th>Intensity deviation [%]</th>
<th>Source area ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noiseless input (V_{\text{noiseless}})</td>
<td>-0.02 ± 1.43</td>
<td>-4.35 ± 6.66</td>
</tr>
<tr>
<td>Noisy input (V_{\text{noisy}})</td>
<td>-0.01 ± 1.33</td>
<td>0.61 ± 12.40</td>
</tr>
<tr>
<td>Noise &amp; white noise (V_{\text{noisy&amp;white\ noise}})</td>
<td>0.05 ± 1.96</td>
<td>3.49 ± 12.26</td>
</tr>
</tbody>
</table>

### Table 4. Comparison of run-times to reconstruct the data with our trained radionets neural network model and wsclean for different image sizes

<table>
<thead>
<tr>
<th>Image size [pixels]</th>
<th>radionets run-time [s]</th>
<th>Evaluation run-time [s]</th>
<th>wsclean run-time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2.29 ± 0.04</td>
<td>0.0031 ± 0.0001</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>128</td>
<td>2.31 ± 0.08</td>
<td>0.0081 ± 0.0002</td>
<td>0.63 ± 0.04</td>
</tr>
<tr>
<td>256</td>
<td>2.48 ± 0.06</td>
<td>0.0269 ± 0.0008</td>
<td>1.03 ± 0.04</td>
</tr>
<tr>
<td>512</td>
<td>2.65 ± 0.13</td>
<td>0.0995 ± 0.0043</td>
<td>2.42 ± 0.09</td>
</tr>
<tr>
<td>1024</td>
<td>3.20 ± 0.06</td>
<td>0.4070 ± 0.0052</td>
<td>9.00 ± 0.34</td>
</tr>
</tbody>
</table>

Note: The programs are run for 100 times on one image. The presented values are the mean run-times with standard deviations.

Fig. 19. Exemplary \((u, v)\) coverage for a simulated observation with 50 time steps and 4 frequency bands. The additional channels lead to a slightly improved coverage of amplitude and phase frequels. However, the greater advantage is the improved sensitivity of the simulation since the signal-to-noise ratio is improved by the increased number of data points.

### 4.3. Execution times

Assuming that we can apply our network to a large number of similarly obtained data sets, the time for training the model becomes less important than the execution time on a single data set. To evaluate this, we have summarized the run-times of our model and of wsclean for different image sizes in Table 4. For the deep learning model, the run-time is given once for the...
whole radionets framework, which includes the loading of the model, the loading of the test data, and the saving of the clean image. Additionally, the pure execution time of the model is shown. In order to determine the run-times of wsclean, the cleaning settings from Tab. 2 were used. The size parameter was adjusted here for the different image sizes. The results show that the radionets framework generates clean images faster than wsclean for larger image sizes. Considering the pure reconstruction time of the deep learning model, our models reconstruct the input data faster for all image sizes.

5. Further analysis

5.1. Improved simulations

The current simulations mimic observations of radio galaxies built from Gaussian components. More realistic simulations of radio interferometric observations require further adjustments. The most important aspects are the beams of the individual telescopes inside the interferometer array, the effect of sidelobes, and the influence of Gaussian noise per baseline. In the future, we will directly simulate visibilities using the radio interferometer measurement equation (RIME) to describe the individual components of the observation (Smirnov 2011) including the array responses. Furthermore, it is modularly expandable to add effects of the ionosphere or noise corruption by telescope receivers. Together with advanced radio sky simulations, this will provide the path to train deep learning networks that can be applied to real data.

First, we created a data set containing extended as well as point sources and trained a new network for 300 epochs with 50 000 training and 10 000 validation images. This has the advantage that we can test how the chosen architecture behaves on data that differs from what we have tested so far. To evaluate the performance of the network, we created a new test data set containing simulated point-like Gaussian sources with different sizes besides the extended radio galaxies. Each image covers a randomly drawn number of these additional point-like sources between one and six. The images have a size of 64×64 pixels and thus are comparable to subimages of a larger sky survey. Again, the methods described in Sect. 2 were used to simulate observations with a radio interferometer. In this way, we created a data set consisting of 10 000 test images. The sampled amplitude and phase distributions served as input for the neural network.

In Fig. 20, we show an example of a source reconstruction image that comes from the mixed data set. It is apparent that the positions of all point sources, even the ones with a very small specific intensities, are correctly reconstructed in the predicted image. The extended source in the lower right half of the image can also clearly be seen. For all sources, some intensity is missing as evident from the difference plot on the right side, but the maximal difference is just around 10 % of the maximal intensity. In addition, the background is predominantly reconstructed to zero, no major artifact can be seen. In summary, our network is able to reconstruct a mixed data set as well as a data set containing only extended sources.

For the evaluation of these mixed images, we compared the mean specific intensity of each source in the image in the same way as in Fig. 18. For the extended sources, we summed up the specific intensity over the whole source area. The results are summarized in a histogram shown in Fig. 21 separately for point and extended sources. It turns out that most of the sources’ intensity is underestimated, which is confirmed by the mean values of ≈−6 % and ≈−14 %, respectively. We note that there are some outliers with large positive deviations but their numbers are very small.

Moreover, we evaluated a possible relation between the linear extent of the sources and the mean specific intensity deviation. The linear extent for the point sources was calculated via the standard deviation of the Gaussian kernel that was used to smear out the sources. For the extended sources, this was accomplished via the distance between the most distant blobs for an extended source plus the sigma values for these blobs to account for the spreading. The corresponding plot is shown in Fig. 22. While the extended sources are generally underestimated, the point sources have more outliers in the overestimate region, but the majority is still underestimated, as Fig. 21 shows. Furthermore, we find no correlation between the linear extent and the intensity deviation.

For a more detailed look into the specific intensity, we show the relative deviations for the different intensities in Fig. 23. For small intensity values, the estimates are in accordance with the true values, except for the very smallest values. With increasing mean specific intensity the relative deviations increase, as expected from Fig. 21. The uncertainty of the relative deviations are similar for all intensities, except for the smallest and the largest intensities.

In summary, our network is able to reconstruct a mixed data set containing point and extended sources on a similar level as the simple data set consisting of extended sources only.

5.2. Independence of image size: reconstruction of larger images with the same architecture

A feature of the proposed architecture is not to resize the input data, which means that the image size does not vary throughout the network, and thus any image size can be handled without the need for resizing or similar operations.

For illustration, we performed the training of a basic network on a data set with small-scale amplitude and phase maps, such as the maps presented in the previous sections. Here, training on a big data set on short timescales is possible. Afterward, we fine-tuned this network by training it with a data set consisting of larger amplitude and phase maps for another 20 epochs. This data set consisted of 5 000 training maps and 1 000 validation maps. For these maps, we simulated a larger field of view and larger baselines at the same time, which resulted in higher spatial resolutions. The fine tuning took around 30 min. This technique of transferring the learning process enables a fast convergence for the network to perform well on the new data set. The fine-tuning is necessary to adapt the new scaling effects of the ionosphere or noise corruption by telescope receivers from what we have tested so far. To evaluate the performance of the network, we created a new test data set consisting of extended as well as point sources and trained a new network for 300 epochs with 50 000 training and 10 000 validation images. This has the advantage that we can test how the chosen architecture behaves on data that differs from what we have tested so far. To evaluate the performance of the network, we created a new test data set containing simulated point-like Gaussian sources with different sizes besides the extended radio galaxies. Each image covers a randomly drawn number of these additional point-like sources between one and six. The images have a size of 64×64 pixels and thus are comparable to subimages of a larger sky survey. Again, the methods described in Sect. 2 were used to simulate observations with a radio interferometer. In this way, we created a data set consisting of 10 000 test images. The sampled amplitude and phase distributions served as input for the neural network.

In Fig. 20, we show an example of a source reconstruction image that comes from the mixed data set. It is apparent that the positions of all point sources, even the ones with a very small specific intensities, are correctly reconstructed in the predicted image. The extended source in the lower right half of the image can also clearly be seen. For all sources, some intensity is missing as evident from the difference plot on the right side, but the maximal difference is just around 10 % of the maximal intensity. In addition, the background is predominantly reconstructed to zero, no major artifact can be seen. In summary, our network is able to reconstruct a mixed data set as well as a data set containing only extended sources.

For the evaluation of these mixed images, we compared the mean specific intensity of each source in the image in the same way as in Fig. 18. For the extended sources, we summed up the specific intensity over the whole source area. The results are summarized in a histogram shown in Fig. 21 separately for point and extended sources. It turns out that most of the sources' intensity is underestimated, which is confirmed by the mean values of ≈−6 % and ≈−14 %, respectively. We note that there are some outliers with large positive deviations but their numbers are very small.

Moreover, we evaluated a possible relation between the linear extent of the sources and the mean specific intensity deviation. The linear extent for the point sources was calculated via the standard deviation of the Gaussian kernel that was used to smear out the sources. For the extended sources, this was accomplished via the distance between the most distant blobs for an extended source plus the sigma values for these blobs to account for the spreading. The corresponding plot is shown in Fig. 22. While the extended sources are generally underestimated, the point sources have more outliers in the overestimate region, but the majority is still underestimated, as Fig. 21 shows. Furthermore, we find no correlation between the linear extent and the intensity deviation.

For a more detailed look into the specific intensity, we show the relative deviations for the different intensities in Fig. 23. For small intensity values, the estimates are in accordance with the true values, except for the very smallest values. With increasing mean specific intensity the relative deviations increase, as expected from Fig. 21. The uncertainty of the relative deviations are similar for all intensities, except for the smallest and the largest intensities.

In summary, our network is able to reconstruct a mixed data set containing point and extended sources on a similar level as the simple data set consisting of extended sources only.
5.3. Additional source shapes

Machine learning algorithms learn from examples in the training data set and they use the acquired knowledge to make predictions for new examples. One of the limitations of machine learning is its inherent difficulty to generalize their acquired knowledge.

To test our network on untrained source shapes, we used the network from subsubsection 4.1.1 and applied it to more complex simulated radio galaxies. In these more complex examples, the jets of the sources are bent and individual components start to overlap as additional rotations of the jets are taken into account. Fig. 26 and Fig. 28 show the reconstruction of new jet shapes for a one- and a two-sided source, respectively. In both cases, the specific intensity is underestimated, while the general source shape is reconstructed well.

Again, a fine-tuning of the existing deep learning network helps to increased the reconstruction quality. For this reason, we trained the network for additional 40 epochs on a data set consisting of the new source shapes. This data set contained 10 000 training images and 2 000 validation images. The sampled amplitude and phase maps served as input for our network. The fine tuning took around 40 min. Fig. 27 and Fig. 29 visualize the reconstruction of new jet shapes for the one- and the two-
Fig. 24. Clean image resulting from the reconstructed amplitude and phase distributions generated by a basic network trained on (64 × 64)-pixel maps without additional fine-tuning. Resulting clean image (left), simulated brightness distribution (middle) and difference between both (right). The jet angle $\alpha$, which was calculated using a PCA, is given for both source images.

Fig. 25. Clean image resulting from the reconstructed amplitude and phase distributions generated by a basic network trained on (64 × 64)-pixel maps, which was fine-tuned on the new data set with (128 × 128)-pixel maps for another 20 epochs. Resulting clean image (left), simulated brightness distribution (middle) and difference between both (right). The jet angle $\alpha$, which was calculated using a PCA, is given for both source images.

Fig. 26. Reconstruction of a one-sided jet with kinks and overlapping Gaussian components, which was not used in the training procedure. Resulting clean image (left), simulated brightness distribution (middle) and difference between both (right).

sided source reconstructed by the fine-tuned network, respectively. The reconstruction quality increased significantly as the intensity deviation between reconstruction and simulation decreased for both cases. Especially, fainter parts of the source are reconstructed better. This comes at the cost of more background artifacts.

For our network, this means that source shapes which were not used in the training process are less likely to be well-reconstructed. One advantage of radio interferometry is that many sources appear as two-dimensional Gaussian distributions or can be composed of several two-dimensional Gaussian distributions. Since the networks we have trained are capable of reconstructing Gaussian sources, they can do so to some extent on Gaussian sources with new shapes. However, when the jet components become very diffuse, the reconstruction quality drops significantly.
Schmidt et al.: Deep learning-based imaging in radio interferometry

6. Perspectives

6.1. Improved data preparation: noise estimations

Analysis methods based on deep learning often have difficulties with very noisy input data. This problem mainly occurs because of the difficulty to describe the effect of noise on observations correctly in the simulations. This can be done by improving noise simulations or by improving the data preparation before the data enters the neural network.

Neural networks built from residual blocks show a reasonable capability for handling Gaussian noise. Nevertheless, further development and testing is necessary since noise on radio interferometer data leads to large uncertainties in the reconstructions and is a known problem in the conventional analysis. To address this problem, we suggest a noise estimation and correction step already before the reconstruction step. A possible solution is the application of another deep learning network in the data preparation step.

6.2. Uncertainty estimate

A disadvantage of conventional imaging strategies is their inability to quantify the uncertainties in the reconstructed clean images. To enable the deep learning network to quantify errors, we suggest an adjustment of the loss function, comparable to the one proposed in Abbasi et al. (2021). We assume a value origi-
initiating from a Gaussian distribution with the parameters $\mu$ and $\sigma$ for every pixel. The negative log-likelihood of this Gaussian distribution, is

$$-\mathcal{L} = 2\log(\sigma) + \frac{(x - \mu)^2}{\sigma^2},$$

and serves as a minimization function for the neural network. Now, the network predicts two values for every pixel, where $\mu$ is the reconstructed value for this pixel, and $\sigma$ corresponds to the estimated uncertainty for this value. $x$ represents the true value for the specific pixel. This approach has its limits, as the pixels are unavoidably correlated. This loss function is an extension of the mean squared error (MSE), which results in the special case of $\sigma$ being the same for all pixels.

Estimates for $\mu$ and $\sigma$ make it possible to vary the reconstruction obtained by the neural network. The one-dimensional Gaussian distribution obtained for each pixel permits the sampling of several values for a specific pixel. As a result, we obtain $n$ versions of the reconstructed amplitudes and phases. The pixel-wise comparison of these reconstructions helps to quantify uncertain regions by calculating the standard deviation of the $n$ different versions. Furthermore, these $n$ versions of the reconstructions allow for creating $n$ reconstructed clean images which enable the quantification of uncertainties in image space.

The first tests show how regions with high uncertainties in the resulting clean images can be identified. In the future, we plan to extend this approach such that the quality of the reconstructions can be estimated even if the real brightness distribution is unknown.

### 6.3. Source finder and source list prediction

In radio astronomy one may not always want to obtain an image but rather extract parameters such as the source position and the flux density. Hence, alternative approaches may be more suitable for the analysis of images of large areas of the sky. The cleaning process of such images puts high demands on computer memory. We suggest the use of deep learning networks to estimate source parameters directly without obtaining an image first. Architectures developed for object detection, for example the SSD300 architecture (Liu et al. 2016), can help to solve this task. These networks are able to infer source positions and, simultaneously, can classify the sources. A loss function using the object’s bounding boxes enables the estimation of an arbitrary number of sources in the input image (Erhan et al. 2013).

Finally, this approach allows for the improvement of conventional imaging software. During the reconstruction process, flux from source regions gets extracted iteratively in the form of point sources. In this way, common imagers create a model consisting of multiple point sources for the source. Replacing the existing routines with an approach based on a deep learning network can help to improve the speed and the accuracy of the currently used methods.

### 7. Conclusions

A new generation of radio interferometers with enormous data rates requires analysis strategies that generate reproducible results on short timescales and with affordable computing resources. A way to achieve this goal is to apply deep learning-based analysis methods directly on the $(u,v)$ data.

Our simulation is designed to quickly generate images based on the appearance of cleaned radio interferometry measurements. The $(u,v)$ coverage of the simulated sources are sampled by $(u,v)$ masks based on observations with the VLBA. Thus, we generate incomplete $(u,v)$ spaces as in realistic radio interferometric observations. This data serves as input for the neural network.

We have shown that our approach of using neural networks to reconstruct incomplete $(u,v)$ spaces can generate reproducible results quickly and reliably in the case of radio galaxies made from Gaussian components. Deviations between reconstructed and simulated images vary depending on the quantity, we are looking at. Especially the jet angles and the source area ratios, as well as the mean specific intensity deviations of the core components are constructed fairly reliably. The corresponding mean values are summarized in Tab. 3. The results in the lower part of Tab. 3 show that larger sampling densities in $(u,v)$ space lead to better reconstructions by the neural network. Furthermore, our network can deal with noisy input data. This result is supported by the example images and histograms presented in subsection 4.1.2.

For comparison with an established cleaning method, we processed our data using wsclean. The results of subsection 4.2 suggest that the reconstruction with wsclean does not perform as well as the deep learning networks. At this point we would like to emphasize that the direct comparison is not completely fair. The networks we have trained are designed to reconstruct a wide range of similar input data. This is made possible by the large statistics during the training process, which takes several hours, as described in Sect. 3.

When looking at derived quantities such as jet angles, it turns out that our neural network performs better than wsclean.

The big advantage of wsclean is that there is no need to train a network first. After finding suitable cleaning parameters, the creation of the clean image takes only a few seconds. The disadvantage is that the quality of the reconstructed images depends strongly on the selected cleaning parameters and these differ for different input data. This makes it difficult to easily apply wsclean to a large data set even if the data quality of the input data hardly differs, see subsection 4.2. Furthermore, finding suitable cleaning parameters can take many iterative approaches.

Another difference between the two methods is that our deep learning reconstructions are not convolved with a clean beam. The input data is reconstructed directly in Fourier space and the clean image is created by the inverse Fourier transformer of the fully filled $(u,v)$ space. In case of wsclean a point source model is created which is then convolved with the theoretical clean beam. This results in smearing of the reconstructed brightness distributions. This is partly due to technical obstacles, such as small image sizes, and partly due to the fact that our chosen parameters from wsclean are only optimized for a handful of images and therefore do not perform as well as our network on a 10 000 image data set. Hence, our approach may have advantages over wsclean when it comes to fast reconstructing a large number of images.

In subsection 5.3, we discuss the performance of our network applied to a more complex data set with kinked and diffuse jet sources. The results show that our network is able to reconstruct these examples reasonably well.

In conclusion, we have made a proof-of-concept that deep learning methods can be applied to reconstruct incomplete Fourier data in radio interferometric imaging. Even input data with uncorrelated noise produces results that match those created with noisecless input data.

Our analysis framework radionets (Schmidt et al. 2019) is made available as an open-source package. In future releases, we will improve our simulations by utilizing RIME and the Jones calculus. Thus, we enable the consideration of additional com-
lications such as the point spread function of the radio telescopes, the influence of side-lobes, or multi-channel data. In combination with the ability to handle larger image sizes, this will open the way to train deep learning networks applicable to real data. Furthermore, neural networks can be used to quantify uncertainties in the reconstructed clean images. Uncertainty maps will help to evaluate the reconstruction results. More advanced segmentation techniques enhance the location of source positions. The segmentation maps can be used to improve the performance of established imaging software.

Acknowledgements. This work is supported by Deutsche Forschungsgemeinschaft (DFG) – project number 124020371 – within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, DFG project number 124020371. SFB project C3. We acknowledge support from the BMBF (Verbundforschung). MB acknowledges support from the Deutsche Forschungsgemeinschaft under Germany’s Excellence Strategy - EXC 2121 "Quantum Universe" - 390833036, and the support and collaboration with the Center for Data and Computing in Natural Sciences (CDCS). We thank Kai Brügge for his initial motivation and discussions that formed the foundation for this work. We thank Richard Wiemann for fruitful discussions and helpful ideas.

References

Abbasi, R., Ackermann, M., Adams, J., et al. 2021, A Convolutional Neural Network based Cascade Reconstruction for the IceCube Neutrino Observatory
Amaro, S. I. 1993, Neurocomputing, 5, 185
Gross, S. & Wilber, M. 2016, Training and investigating Residual Nets
He, K., Zhang, X., Ren, S., & Sun, J. 2015b, in Proceedings of the IEEE International Conference on Computer Vision (ICCV)
Howard, J. & Gugger, S. 2020, Deep Learning for Coders with fastai and PyTorch (O’Reilly Media, Inc.), 205–207
Hunter, J. D. 2007, Computing in Science Engineering, 9, 90
Königl, A. 1980, The Physics of Fluids, 23, 1083
Office, M. 2011-2018, Cartopy
Python Software Foundation. 2020, Python

Article number, page 19 of 21
Appendix A: Used software and packages

In this work, we used PyTorch (Paszke et al. 2019) as the fundamental deep learning framework. It was chosen because of its flexibility and the ability to directly develop algorithms in the programming language Python (Python Software Foundation 2020). In addition to PyTorch, we used the deep learning library fast.ai (Howard et al. 2018) which supplies high-level components to build customized deep learning algorithms in a quick and efficient way. For simulations and data analysis, the Python packages NumPy (Oliphant 2006), Astropy (The Astropy Collaboration et al. 2013, Price-Whelan et al. 2018), Cartopy (Office 2011-2018), scikit-image (Van der Walt et al. 2014), and Pandas (McKinney et al. 2010) were used. The illustration of the results was done using the plotting package Matplotlib (Hunter 2007). A full list of the used packages and our developed open-source radionets framework can be found on github: https://github.com/radionets-project/radionets

Appendix B: Flux distributions

Fig. B.1. Exemplary contour plot for a training session with clean input images. Resulting clean image (left) and simulated brightness distribution (right) are shown. For both images, the contour levels are based on the peak flux density of the simulated brightness distribution. The ratio of 0.98 is calculated between the 10% boundary of the prediction and the truth.
Fig. B.2. Exemplary contour plot for a training session with noisy input images. Resulting clean image (left) and simulated brightness distribution (right) are shown. For both images, the contour levels are based on the peak flux density of the simulated brightness distribution. The ratio of 1.12 is calculated between the 10% boundary of the prediction and the truth.

Appendix C: Computer setup

Table C.1. Computer specifications for the setup used in the training process

<table>
<thead>
<tr>
<th>Part</th>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>Nvidia GeForce RTX 2080</td>
<td>8 GB</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel Core i7-8700k SSD</td>
<td>12 Cores @ 3.7 GHz</td>
</tr>
<tr>
<td>Hard drive</td>
<td>SSD</td>
<td>512 GB</td>
</tr>
</tbody>
</table>