Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

# VT2 | MSE Data Science

## Application of Deep Learning to Source Classification in Radio Astronomy

| | |
|---|---|
| **Author** | Manuel Weiss |
| **Advisor** | Philipp Denzel |
| **Date** | 31.01.2024 |

Zürcher Fachhochschule www.engineering.zhaw.ch Studium

# Abstract

Modern radio telescope surveys, capable of detecting millions of galaxies, have made manual morphological classification impracticable. This applies in particular when the Square Kilometre Array (SKA) becomes operable in 2027. With the SKA, researchers hope to close an important gap in our understanding of the Universe: the Cosmic Dawn, i.e. the time when the first stars and galaxies were formed. Since light travels through the cosmos from different points in spacetime, an image captured by a radio telescope includes galaxies from several epochs. To analyse the origins of the Universe, more recent radio emissions must therefore be removed from the record, which is why the classification of celestial objects is crucial. This work presents a benchmark of state-of-the-art object detection and classification architectures on a six-class Radio Galaxy Zoo dataset. The focus of the model selection was on the comparison between CNN- and transformer-based algorithms. The experiments include the investigation of different scaling techniques, the selection and training of two object detection and three classification algorithms, an uncertainty estimation by an ensemble analysis and a verification of the results on samples of the GaLactic and Extragalactic All-Sky MWA Survey (GLEAM) dataset. The best classification performance of $89.67\%$ top-1 and $97.47\%$ top-2 accuracy was achieved by using a ResNet50 architecture trained on ZMZStack scaled images and by applying basic augmentation. Using an ensemble of three models further increased performance to $90.68\%$ top-1 and $98.58\%$ top-2 accuracy. The final evaluation of six GLEAM images analysed with an ensemble of $30$ ResNet50 models showed a probability of misclassification above $5\%$ and below $32\%$ for the five complex samples. However, the image crops consist of a significantly lower resolution, and an analysis of GLEAM images with higher resolution had to be postponed to future work due to external factors.

# Contents

# Chapter 1

# Introduction

## 1.1 Problem Statement

Investigating the origin of the Universe is a fundamental field of research in astronomy[1;2]. The current construction of the Square Kilometre Array (SKA) radio interferometer is the most promising human initiative to study the history of the Universe[3;4]. More specifically, the exploration of the Universe at the time of the *Cosmic Dawn* is a key science objective of the prospective SKA telescope observatory[2]. The Cosmic Dawn describes an era when the first stars were formed and the Universe was only a few 100 million years old[2;5]. Contrary to optical telescopes, radio telescope can analyse light emission at much lower frequencies, in the radio band, which allows sources to be studied at much greater distances[6]. Due to the enormous size of the SKA and the corresponding increase in resolution, the SKA allows researchers to investigate a larger timespan[7]. However, the farther reach of the SKA imposes new challenges[8].

Since the Universe can be regarded as four-dimensional with three spatial and one timelike dimension, the resulting image of the SKA at a given frequency is two-dimensional[9;7]. The third and fourth dimension are reduced, like squeezing a cone. In other words, since light travels at constant speed, light rays emanating from far away objects in earlier epochs and rays originating more recently, closer to Earth are recorded simultaneously[7]. Consequently, if a radio telescope has a sufficient resolution, the captured image features sources from several epochs[7]. In order to investigate the Cosmic Dawn with the results recorded by the SKA, the emissions from before this time must be filtered. More recent emissions can therefore be regarded as noise in the signals from the Cosmic Dawn. In an image these foreground sources must first be localized and characterized, resulting in a radio source catalogue. Astronomers can then cleanly identify the types of radio sources which can be omitted to get a denoised view of the Cosmic Dawn and earlier epochs.

The unprecedented extent and efficiency of the SKA allows coverage of a wide and deep field of the sky, yielding an image containing millions of radio sources[1]. This large quantity of sources makes it infeasible to process the emissions manually and requires an automated approach[8;10]. In recent years, advances in deep learning (DL) have produced various new methods and algorithms for object recognition and image classification[11;12;13;14;15;16]. Particularly in computer vision, DL algorithms such as convolutional neural networks are being used successfully in various areas of research[17]. Their success in dealing with high-dimensional data and a low signal-to-noise ratio (SNR) has led to a widespread use of DL algorithms to identify radio sources or astronomy in general[8;18;19;20].

This work mainly focuses on comparing and training different image classification and object detection models for the automated identification of radio sources. This encompasses testing various preprocessing techniques, conducting ensemble experiments and fine-tuning the most promising architecture from a model benchmark on a dataset of radio galaxies captured in several last-generation surveys. As a final test, the trained model is applied to samples from the GLEAM survey to evaluate its transferability to another radio source dataset obtained by a precursor project to the SKA[21].

In recent years, various works with remarkable outcomes on training classification or object detection models for radio source determination can be found in the literature[22;8;23;24;25;26]. However, the majority investigate or develop a specialized architectures. Therefore, this work aims to scrutinise whether prevalent state-of-the-art (SOTA) architectures can obtain comparable results to tailored approaches which require much more effort and development time. Furthermore, this review examines if the complexity of recent DL-based architectures is an important criterion for the rather simple task of radio source classification or detection.

## 1.2 Related Work

Since the discovery of radio source emissions by James Stanley Hey in 1942, an important task in radio astronomy was to identify or classify the captured objects[6]. The term *classification* in this context refers to the process of assigning a uniquely describing label to a located radio source. One of the main motivations for these classification efforts is the creation of catalogues and maps of the Universe, which was initially done manually by experts[6]. Thanks to technological advances, future radio telescopes may detect light waves from millions of undiscovered radio sources over the time of their operation[8]. This is especially true for a planned survey of the SKA observatory, the so-called Evolutionary Map of the Universe (EMU) with a sensitivity of 10 mJy/beam, encompassing the entire southern sky, as well as an estimated number of 70 million detected radio sources[27]. This enormous inflow of data requires rethinking of old analysis pipelines and the development of new ideas.

The development of automated algorithms for radio source classification and detection conveniently aligns with advances in computer vision, machine learning and more recently, deep learning algorithms. With the first larger catalogues, researchers applied classical machine learning approaches to the present classification problem. Fayyad, Weir and Djorgovski[28] as well as Ball et al.[29] applied Decision Trees, Zhang[30] and Sadeghi[31] used Support Vector Machines (SVM) and Alger[32] or Cheng[33] compared the performance of Logistic Regression as well as Random Forests to neural architectures like Convolutional Neural Networks (CNNs). Due to their data efficiency and the smaller number of samples required for training[34], classical machine learning architectures are still widely used for these tasks. However, most benchmarks reveal a clear advantage when using neural networks like CNNs in comparison to classical machine learning approaches[33;34;32].

Statistical methods for radio source classification are a staple for astronomy and astrophysics alike. E.g., AEGEAN, a method proposed by Hancock, Trott and Hurley-Walker, benefits from spatially correlated data as well as from a varying background and noise across the sky to fit a source finding model[7]. Algere et al.[35] use PySE, a Python software package for radio source identification through brightness peaks and noise reduction[36], on a dataset acquired with the LOFAR telescope[37] and characterized sources using a cross-matching technique with a binary classifier.

Lately however, neural-based approaches have gained traction in radio source detection or classification. The first architectures which notably exceeded the performance of classical machine learning methods were CNNs. Therefore, researchers analysed well-known CNN architectures like AlexNet[38] or Google's LeNet[39] as well as classical CNN implementations[40;41] with specific adjustments like sigma clipping layers as in Lukic et al.[42]. Finally, Becker et al.[43] performed extensive comparisons of CNN architectures.

In addition to the preprocessing steps discussed in this report (chapter 4.4.2), augmentation is often used to tackle the problems with imbalanced datasets or strengthen a model's robustness[40;44;45]. Therefore, simple methods such as the rotation[40;44] up to more advanced ones such as the generation of synthetic data with generative adversarial networks (GAN), as in the work of Hosenie[46], have yielded great performance improvements.

More recent investigations go beyond classical supervised learning methods into the domain of semi-supervised or unsupervised learning approaches[47;48;49;10;50]. Whereas Polsterer, Gieseke and Igel[47] as well as Galvin et al.[48] use a flipping invariant extension of self-organising maps (SOM), called PINK, apply Slijepcevic[10] and Hossain[50] an algorithm called Bootstrap Your Own Latent (BYOL) together with Contrastive Learning (SimCLR) to overcome the shortage of inaccurately labelled data.

In the field of radio source detection, various other architectures have also been studied in the literature. However, the latest SOTA architectures designed for the use with natural images still remain largely untested up to now. A frequently used algorithm are versions of R-CNN or Masked R-CNN, as in the papers of Riggi et al.[22;8] or Mostert et al.[23]. Another widely used model family is You Only Look Once (YOLO), e.g.[51] which is a precursor version of the current YOLOv8 architecture[11]. Whereas Zhang, Jiang and Zhang[24] applied a basic implementation of the YOLOv5 model, Wang et al.[25] added a supplementary attention mechanism to the architecture. At the time of this project, Sortino et al.[26] provides the only comparison available with multiple recent models examined for radio source detection. In the extensive benchmark, representatives of Masked R-CNN, Detectron2, DETR, YOLOv4, YOLOv7, YOLO S and EfficientDet were compared with promising results above the 90% accuracy level on an open source radio galaxy dataset. Sortino et al.[26] therefore gives an indication that SOTA object detection models work efficiently with radio source images.

After two-stage region proposal networks (RPN) like faster R-CNN[52] had dominated the field for years, faster and more performant one-stage models like YOLO emerged and became the prevalent architectures for generic object detection in industrial sectors as well as academia[53]. With the outstanding successes of transformers in Natural Language Processing (NLP) applications[13], their architecture also received attention by the computer vision community, especially for generative models[54]. The first end-to-end transformer-based architecture for object detection was proposed by Carion et al.[55], called Detection Transformer or DETR. Many adjustments have been made to the original implementation to address particular shortcomings like a slow training convergence, as for the encoder-only DETR[56]. Other studies focus on a deeper understanding of decoder queries which are often associated with spacial positions of different perspectives[12]. From these efforts new architecture variations like DAB-DETR[57], denoising-improved DETR models such as DN-DETR[58], or DINO[12] emerged which further raised training speed, stability and ultimately performance on many popular datasets like COCO[59].

Ultimately, to our knowledge, there has not yet been a study that applied the latest SOTA object detection or classification architectures for radio source datasets.

## 1.3 Goals

The main objective of this (and subsequent) work is to enable automatic classification of radio sources on unseen or newly generated datasets. For this purpose, a selected number of neural-based image classification and object detection architectures was trained and compared on a curated, open-source dataset of radio galaxies. However, due to limited resources, not arbitrarily many architectures can be compared, which is why a configurable training pipeline for multiple architectures was set up to minimize overhead. As the SKA is still in construction and not yet operable, a second key objective is to test the trained model against the GaLactic and Extragalactic All-sky MWA Survey (GLEAM) catalogue[4;21]. In summary, the objectives of this work can be expressed as follows:

1. Evaluating different preprocessing steps for the selected radio source dataset.
2. Implementing a configurable multi-architecture object detection and classification pipeline.
3. Conducting a benchmark of selected SOTA image classification and object detection models.
4. Selecting and fine-tuning the most promising model out of the architecture benchmark.
5. Carrying out ensemble experiments for selected models to estimate uncertainties.
6. Testing the obtained model on samples of the GLEAM[21] catalogue.

# Chapter 2

# Theoretical Introduction

## 2.1 Datasets

This section introduces the terminology of radio source datasets and provides an overview of the datasets used in this project.

### 2.1.1 Radio Source Datasets

Radio source telescopes are commonly operated by intergovernmental organizations, often partially owned by universities. Creating a large image or map of radio sources, known as a survey in astronomy, is an involved project that costs time and money. On the one hand, this means that only a limited number of datasets are available, on the other hand it also supports the fact that the datasets are made publically available to the scientific community, albeit after an embargo phase of one to two years. For this work, two datasets were considered which were publicly available at the time of the project. The two datasets, namely Radio Galaxy Zoo (RGZ) and GaLactic and Extragalactic All-sky MWA Survey (GLEAM), are presented in the following subsections.

### 2.1.2 Radio Galaxy Zoo (RGZ)

The Radio Galaxy Zoo (RGZ) of Banfield et al.[60] is one of the most widely used radio galaxy classification dataset in the literature. It contains $170'000$ (and counting) radio source images taken from the Faint Images of the Radio Sky at Twenty Centimeters (FIRST)[61] and the Australia Telescope Large Area Survey (ATLAS) Data Release 3[62] datasets. The data labelling was organised in a citizen science project. In a first version (DR1), around $12'000$ citizens contributed to the dataset by labelling roughly $75'000$ sources (Wong et al., in prep.).

However, in order to train an object detection model, not only a dataset with labelled classes must be available, but also annotated bounding boxes, which specify the coordinates of the recognised objects in an image. Therefore, this project uses a subset of the original RGZ DR1 dataset, produced by Wu et al.[63]. We will call this subset Radio Galaxy Zoo Object Detection (RGZ OD) dataset to make a distinction to the original RGZ dataset.

Contrary to the traditionally applied Fanaroff - Riley (FR)[64] categorization for radio sources, the RGZ DR1 dataset is separated into classes in terms of number of peaks and number of components (Wong et al., in prep.)[63]. The number of components is defined as the number of discrete radio components that a source encompasses, identified at the $4\sigma$ flux-density threshold[42;63]. The number of peaks refers to the amount of bright or conspicuous illumination peaks in a radio source that are detected by an automatic pipeline processor[42;32]. This results in classes that are named after the combination of these two characteristics and written as follows:

$[Number\ of\ Components]\_[Number\ of\ Peaks] \rightarrow$ e.g $1\_1$.

Subsequently, an example made by Wu et al.[63]: A double-lobed radio galaxy with small angular extent and no radio core may be identified as a source with one component-two peaks (1C_2P) or a two component-two peaks (2C_2P) if the two lobes appear disconnected in the radio image. The classes used in this work are defined within the next paragraphs.

The RGZ OD dataset is the result of two filter rules that were applied to the RGZ DR1 dataset. First, the RGZ OD contains only samples that exceed a user-weighted consensus level (CL) of $\geq 0.6$ in the original RGZ DR1 dataset[63]. According to Wu et al.[63] this should ensure that most radio sources are morphologically human-resolvable. Second, only samples with less than four components and four peaks were selected[63]. This to reduce the effect of a highly imbalanced dataset[63]. After applying these two rules, the resulting six classes of the RGZ OD dataset, together with their number of occurrences are shown in table 2.1 and examples are visualised in figure 2.1.

| **Classes** | **1_1** | **1_2** | **1_3** | **2_2** | **2_3** | **3_3** |
|---|---|---|---|---|---|---|
| Occurrences | 5'300 | 1'331 | 1'412 | 1'251 | 1'208 | 1'334 |
| **Total** | **11'836 Samples** | | | | | |

Table 2.1: Class Occurrences Radio Galaxy Zoo Object Detection Dataset (RGZ OD)[63]

RGZ-OD images are available in PNG or FITS (Flexible Image Transport System) file format[63]. For the latter, image pixel intensities can be stored in an arbitrary range from negative to positive values depending on the unit in which the image was captured or processed[65]. Note that negative values occur due to the primary beam corrections applied in the image generation from the raw telescope data. Additional metadata such as the measurement unit or the telescope from which the file originates can be stored in the header attributes of the file, which are similar to a key-value store[65].

In a second automated step, squared bounding boxes with respect to location and size were generated by considering physical meta attributes defined in the RGZ DR1 dataset[63]. To define a bounding box's centre, Wu et al.[63] used the central location of a sample defined in the $RA$ (Right ascension of the phase centre) and $DEC$ (Declination of the phase centre) header attributes[65]. Additionally, the sky coordinates $S_c$ of the box's four corners are calculated using the RGZ DR1 $max\_angular\_extent$ parameter, which is an estimate of the source's angular size for all RGZ consensus sources as detailed in Banfield et al.[60] and Wong et al. (in preparation)[63]. Finally, the sky coordinates $S_c$ are converted into pixel coordinates $P_c$ by `Python` imaging processing libraries[63].

For this project, the images were used in FITS file format together with the annotation information for a sample[63]. This in order to retain the accuracy of the intensity values before preprocessing.

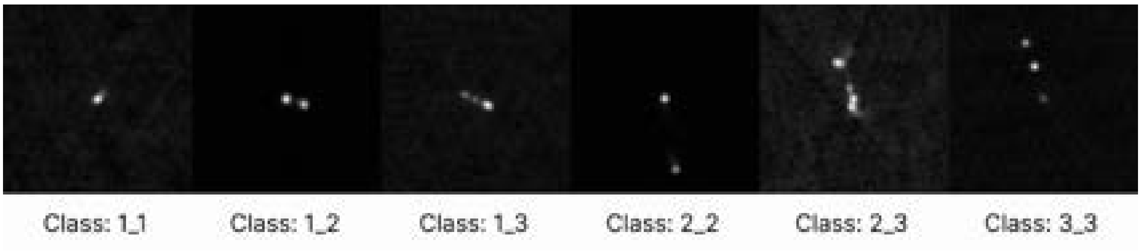


Figure 2.1: Class Samples RGZ OD Dataset[63]

### 2.1.3 GaLactic and Extragalactic All-sky MWA Survey (GLEAM)

Starting in August 2013, the two-year sky observation made with the Murchison Widefield Array (MWA)[66] was source for the GaLactic and Extragalactic All-sky MWA Survey (GLEAM) encompassing the entire radio sky south of declination $+25°$[67;68]. Driven by the goal to measure radio emissions from high-redshift neutral hydrogen during the Epoch of Reionisation (EoR), which are expected to lie in the frequency band between $50$ - $200$ MHz, the GLEAM Survey collected frequencies between $72$ and $231$ MHz[69;70;67]. In the two years of recording, observations were made with a frequency resolution of $40$ kHz and a time resolution of $0.5$s in the first year, respectively $10$ kHz and 2s in the second year. A summary of the recording parameters of the GLEAM Survey is listed in table 2.2.

The final resolution depends on the measured frequency[67]. Operating at $154$ MHz, the imaging capabilities yield a resolution of around $\frac{2.5\times2.2}{\cos(\delta+26.7°)}\ arcmin$, with the sensitivity to detect structures up to an angular size of approximately $10°$[67]. The resulting catalogue contains $307'455$ radio sources[68]. However, some areas were excluded, such as the Magellanic Clouds[68].

Similar to the intention of the GLEAM Survey investigators, the catalogue is used to perform experiments prior the SKA era[67]. The GLEAM Survey gives us the closest pre-SKA test case to verify results obtained by models trained on other datasets, such as the RGZ OD. Detailed explanations about the experiments performed on the GLEAM catalogue can be found in chapter 3.4.

| Parameter | Values |
|---|---|
| Pointing Declinations (deg) | $-72$, $-55$, $-40.5$, $-26.7$, $-13$, $+1.6$, $+18.3$ |
| Central Frequencies (MHz) | $87.68$, $118.4$, $154.24$, $184.96$, $215.68$ |
| Frequency Resolution (kHz) | $40$ (first year), $10$ (second year) |
| Time Resolution (s) | $0.5$ (first year), $2$ (second year) |

Table 2.2: GLEAM Survey Recording Parameters[67]

## 2.2 Model Architectures

Deep Learning architectures are evolving rapidly in the past recent years. As this work evaluates classification and object detection architectures, this chapter aims to offer brief historical and theoretical background on the applied model architecture in this work and provides further references.

### 2.2.1 Classification Models

**Residual Network (ResNet)**

At the time of its conception by He et al.[14] in 2015, ResNet significantly outperformed existing architectures by enabling deeper models to be trained faster and more robustly. Moreover, today still, ResNets are used for many feature extracting backbones in modern architectures[71;72]. In the years around 2015, a debate was held if a model's depth is the primary driver of performance[14]. However, deeper models require a longer training time and are often faced the problem of vanishing or exploding gradients as well as degradation[73]. For instance, the degradation problem can be identified in an error curve, which shows a rapid degradation after starting to converge[14].

Different strategies exist to tackle vanishing or exploding gradients, such as input normalisation, a specific weight initialisation[74;75] or batch normalisation[76]. In comparison, at its essence, ResNet aims to tackle the degradation problem by introducing residual blocks of which an example is shown in 2.2. Instead of the usual sequential stack of layers and activation functions within a block, a residual block also consists of a so-called residual mapping, a residual connection or a skip connection[14]. Such skip connections have mainly two benefits[14]. First, residual blocks convey the original input information to later layers, allowing larger gradients for early stages during optimisation[14]. Second, they retain gradient information when back propagating it to early layers of the network by flowing directly through the skip connections rather than through the entire chain of blocks of a deep network[14]. Nowadays, ResNets are no longer considered state-of-the-art for classification tasks, but they are used in this work to establish a baseline and to validate the classification task and the dataset itself.

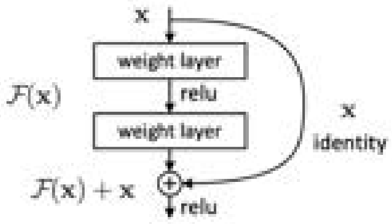

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.2: Residual Block[14]

### EfficientNet

Essentially, EfficientNet, as the name suggests, is all about the efficiency of a model, or more intuitively, how to achieve the desired accuracy with the least amount of parameters[15]. In the original paper of Tan and Le[15], the main focus therefore is on defining a method to efficiently scale CNN architectures.

Considering the development and performance improvements achieved by AlexNet, Google's LeNet and SenNet, it can be assumed that increasing the number of layers in a network makes the greatest contribution to model performance. However, empirical studies have shown that simply increasing the depth of a model leads to a saturation of the performance. A major finding made in the paper was recognising, that there exist a relationship between scaling the depth (#layers), the width (#channels) and the resolution of the input image to a network. Therefore, Tan and Le[15] propose a *compound coefficient* for the network depth, width and the resolution $\phi$, which scales the network more efficiently. Referring to the authors, the compound scaling can be understood intuitively[15]. If the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns[15].

To understand compound scaling, first some notation needs to be introduced, adopted from Tan and Le[15]. Let a convolutional layer be defined as $Y_i = \mathcal{F}_i(X_i)$, where $\mathcal{F}_i$ is an operator, $Y_i$ the output tensor and $X_i$ the input tensor of shape $\langle H_i, W_i, C_i \rangle$, where $H_i$ and $W_i$ are the spatial dimensions of the input image and $C_i$ the number of channels. An entire convolutional neural network (ConvNet) is composed of multiple layers and can be defined as $\mathcal{N} = \mathcal{F}_k \odot ... \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{i=1,...,5} F_i(X_i)$[15]. Today's architectures often use multiple blocks or stages in their definition, with similar convolutional layer types within a block to make the scaling easier. An example of such stages are residual blocks. Hence, a ConvNet can also be defined as:

$$\mathcal{N} = \bigodot_{i=1,...,s} \mathcal{F}_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle}) \tag{2.1}$$

where $\mathcal{F}_i^{L_i}$ denotes layer $\mathcal{F}_i$ is repeated $L_i$ times in stage $i$. Along that notation, the scaling factors $d, w, r$ for depth, width and resolution can be introduced which scale the network as follows:

$$\mathcal{N} = \bigodot_{i=1,...,s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i}(X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w\hat{C}_i \rangle}) \tag{2.2}$$

Here, $\hat{F}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$ are predefined parameters of a baseline network, which were empirically evaluated in favour of performance[15]. The idea of EfficientNet is to scale the *EfficientNet-B0* base network efficiently with a *compound coefficient*.

The *compound coefficient* $\phi$, defined by Tan and Le[15], uniformly scales the network's depth, width, and resolution as follows:

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.3}$$

$\alpha, \beta, \gamma$ are constants that can be determined by grid search. The evaluation of these parameters by Tan and Le[15] lead to $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$. The *compound coefficient* $\phi$ is user defined and controls how many more resources in terms of FLOPS are available, whereas $\alpha$, $\beta$, $\gamma$ determine how these extra resources are assigned to the network in terms of scaling.

Soon after the introduction of EfficientNetV1, Tan an Le proposed EfficientNetV2[77], which aims to further improve training speed while retaining the parameter efficiency. Specifically, EfficientNetV2 tries to address the following limitations of EfficientNetV1: (1) Large image sizes lead to slow training[78]. (2) Scaling up every stage similarly is suboptimal for training speed and parameter efficiency[77]. (3) Depth wise convolutions are slow in early layers but effective in later stages[79;77].

With EfficientNetV2, these shortcomings were addressed as follows: (1) Images sizes and regularisation are dynamically adjusted during training[77]. Early epochs use smaller images sizes with less regularisation (dropout and augmentation), later epochs use larger image sizes with more regularisation[77]. (2) The compound scaling was retained with slight optimisations. The maximum inference image size is limited to 480 pixels and instead of increase layer stages equally, in EfficientNetV2, layers in later stages are added with a gradual increase[77]. (3) Instead of solely using MBConv (inverted residual blocks) layers, EfficientNetV2 uses Fused-MBConv layers in early stages and usual MBConv layers in later stages[79;77].

All these mechanisms were tuned in terms of training speed and efficiency with a neural architecture search based on reinforcement learning[77]. Finally, the empirically optimised results led to a new base network called EfficientNetV2-S, which is used in this work[77].

### Vision Transformer (ViT)

Transformers, as the model of choice in natural language processing (NLP)[13;80;81;82;83], were the first time successfully implemented for image classification problems by Dosovitskiy et al. in 2021 and called Vision Transformer (ViT)[84]. However, compared to existing CNN-based architectures of similar size, such as ResNet, the transformer-based architecture requires a larger amount of data during training[84]. In their paper, Dosovitskiy et. al. mention that transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality[84]. This shortcoming leads to poor generalisation when not trained on a sufficient amount of data[84].

The main goal of Dosovitskiy et al.[84] was to keep the design of the ViT as closely as possible to the original transformer[13], to make use of the simple setup and scalability of transformer models, as well as using contemporarily efficient accelerators. At its core, a ViT is a Transformer encoder-only system with a multi-layer perceptron (MLP) head for classification[84]. An overview of the ViT architecture, along a basic encoder layer (or stage), can be seen in figure 2.3[84]. Today, ViT laid out the basis for many descendants and extension of the architecture, such as BEiT[85] and DeiT[86].

To apply the ViT in computer vision, an image $x \in \mathbb{R}^{H \times W \times C}$ has to be cropped and flattened into 2D patches of size $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$[84]. As in the previous section, $H, W, C$ describe the resolution and the number of channels of an image, $(P, P)$ is the resolution of a patch, and $N = \frac{HW}{P^2}$ is the resulting number of patches which also defines the length of the input sequence to the transformer[84].

In a second step, the patches get embedded to a constant vector size $D$ through a linear projection layer[84].

To the patch embeddings, an additional learnable embedding is prepended, whose final state at the output decoder serves as the image representation $y$. This practice can be compared to the $[class]$ token of BERT[87]. Moreover, similar as in classical transformer models, 1D positional embeddings are added to the patch embeddings to retain positional information[13].

Within the Transformer encoder, depending on the size of the ViT, a defined number of $L$ decoder block are building the backbone of the network[84]. A transformer encoder stage consists of alternating layers of multiheaded self-attention and MLP blocks. Additionally, a layer normalisation (LN) is performed before every block and a residual connection added after every block[88;14].

In the end, on top of the network, an MLP head is applied for final classification with an cross entropy loss. In this work, the ViT is used to evaluate a basic transformer model for radio source classification.

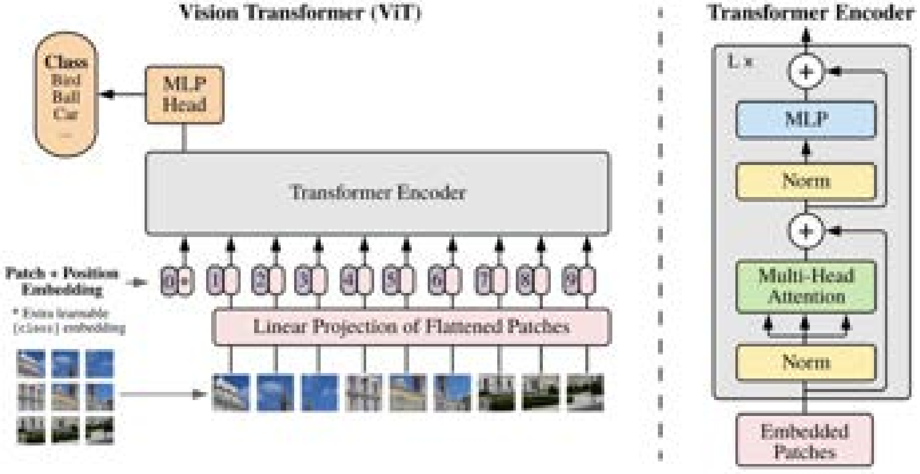


Figure 2.3: Vision Transformer (ViT) Architecture[84]

### EfficientFormer

Beside the larger amount of data required when using transformer-based architectures like ViT[84], they are also often slower (during inference) in comparison to their CNN-based counterparts[16] due to their massive number of parameters. Therefore, the main objective of Li et al.[16] was to create a transformer-based architecture which runs at MobileNet-like speed. To achieve this, Li et al.[16] first performed an in-depth latency analysis of the existing vision transformers. The main bottlenecks that were identified are large kernel sizes when generating the patch embeddings which were shrunk to $3 \times 3$ convolutions for the EfficientFormer. Second, repeated reshaping from 4D to 3D representations within the imagers contributes the most to latency, which is why a dimensionally consistent network is proposed to avoid frequent reshaping operations[16]. Third, Conv-BN (Convolutional Batch Normalisation) is used in early stages, which showed latency gains in comparison to layer normalisation (LN) used in the original ViT[16]. However, empirical analyses have shown that LN leads to better performance when implemented in later phases. Accordingly, LN is used in the final Meta Transformer Blocks (MB)[16]. Lastly, Li et al.[16] concluded that the selection of the activation function should be made hardware and compiler dependent. Tested on an IPhone 12, Li et al. propose to use GeLU in the original implementation of the EfficientFormer network[16]. The final architecture of an EfficientFormer is shown in figure 2.4.

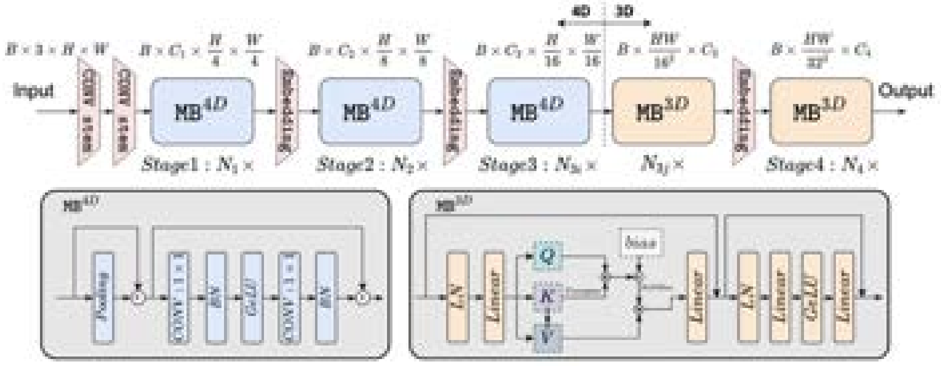


Figure 2.4: EfficientFormer Architecture[16]

### 2.2.2 Object Detection Models

In this project, two object detection models are evaluated on radio source datasets. This chapter aims to give high-level information about the utilised architectures.

#### YOLOv8

YOLO (You Only Look Once) is a popular model family with the major focus on real-time object detection[53]. In comparison to other object detection algorithms, YOLO has proven to have a favourable balance between detection speed and accuracy[99]. Essentially, YOLO reframed object detection tasks from two-stage region proposal systems to a single regression problem[53;52]. In other words, it predicts bounding box coordinates and class probabilities directly from image pixels[53].

Since 2015, the models have been further developed in several iterations from version 1 to version 8 in 2023 to eliminate limitations and improve performance[99]. In version 1, YOLO divides an image into a grid of $S \times S$ cells and predicts $B$ bounding boxes along a confidence score $C$ for all classes in a dataset and for each grid cell[53;99]. A bounding box is defined by five values: $[Pc, bx, by, bh, bw] = BBox$ where $Pc$ indicates the confidence of the model about the accuracy of that particular bounding box, and that it contains an object. $bx$ and $by$ represent the coordinates of the box centre relative to the bounds of the grid cell, and $bh$ and $bw$ describe the width and height of the box, relative to the entire image[53]. The model allows an image with its corresponding dimensions as input and outputs a tensor of shape $S \times S \times (B \cdot |BBox| + |C|)$. In case of RGZ OD with six classes $|C| = 6$, we would use $B = 2$ and $S = 6$, which results in $22$ pixel wide grid cells $(132/6 = 22)$, for which YOLOv1 would output a tensor of shape $6 \times 6 \times 16$. Finally, to remove redundantly predicted bounding boxes, YOLO models rely on a simple but expensive postprocessing step or algorithm called *Non-maximum Suppression (NMS)*[94].

From version 2 to version 7, YOLO models use the concept of prior boxes, better known as default boxes or anchor boxes[99;100;101;102;51;103;104]. Anchor boxes are starting-point boxes for the network with predefined shapes and are used to match prototypical shapes of the objects in the target image[105]. The goal of a network is then to predict the translation offset to this predefined boxes and the class probability[105]. Defining viable anchor boxes supports the network to predict accurate bounding boxes[99]. To evaluate supporting anchor boxes, often k-means clustering[106] is applied in advance. This was incorporated in the algorithm for YOLOv5 onwards and called AutoAnchor[51]. However, these days the technique is more and more seen as hindering and outdated.

YOLOv3, an incremental improvement, introduced an additional *objectness score* for each bounding box using logistic regression[101]. This score should be $1$ if a bounding box belonging to a prediction overlaps a ground truth object by more than any other bounding box prior[101]. In addition, YOLO models from version 3 use predictions at multiple scales through a spatial pyramid pooling plane (SPP) or, in other words, predictions at multiple grid sizes to better account for large or small objects in an image[101].

Beside the introduction of AutoAnchor in version 5, several other augmentation techniques were added to the architecture like Mosaic, MixUp, HSV and translational augmentations[51]. Experiments have shown that the techniques are beneficial for the grid sensitivity and make the model more stable to runaway gradients[99].

YOLOX was the first representative of the YOLO family, which was an anchor-free model[107]. This architecture change reduced the model complexity and improved performance[107]. Besides that, YOLOX decoupled its head module into two heads. One for detection and one for classification. Experiments have indicated that this further improved the performance of the model as well as convergence speed[107].

Beside the mentioned major developments in the previous paragraphs, every evolutionary step included various improvements to the backbone, neck, and head networks, or individual layers in terms of efficiency, robustness, and performance for which we refer to the original papers[53;100;101;102;51;103;104;107;11].

YOLOv8, the version utilised in this work, brought additional enhancements. First, YOLOv8 changed the main YOLO version line to be anchor free[11]. This eliminates the process of optimising anchors before training with a clustering algorithm like k-means[99]. Additionally, anchor-free detection reduces the number of predicted boxes and therefore speeds up the calculation of non maxima suppression (NMS)[11;94]. Second, a C2f module (cross-stage partial bottleneck with two convolutions) was introduced which combines high-level features with contextual information to improve the detection accuracy[11;99]. The YOLOv8 architecture uses a decoupled loss for classification and detection to optimise these tasks separately[11]. To optimise bounding boxes, it uses a Complete IoU (CIoU) loss[108] as well as a Distribution Focal Loss (DFL)[109] and for classification a binary cross entropy loss. The final architecture of YOLOv8 is shown in 2.5.

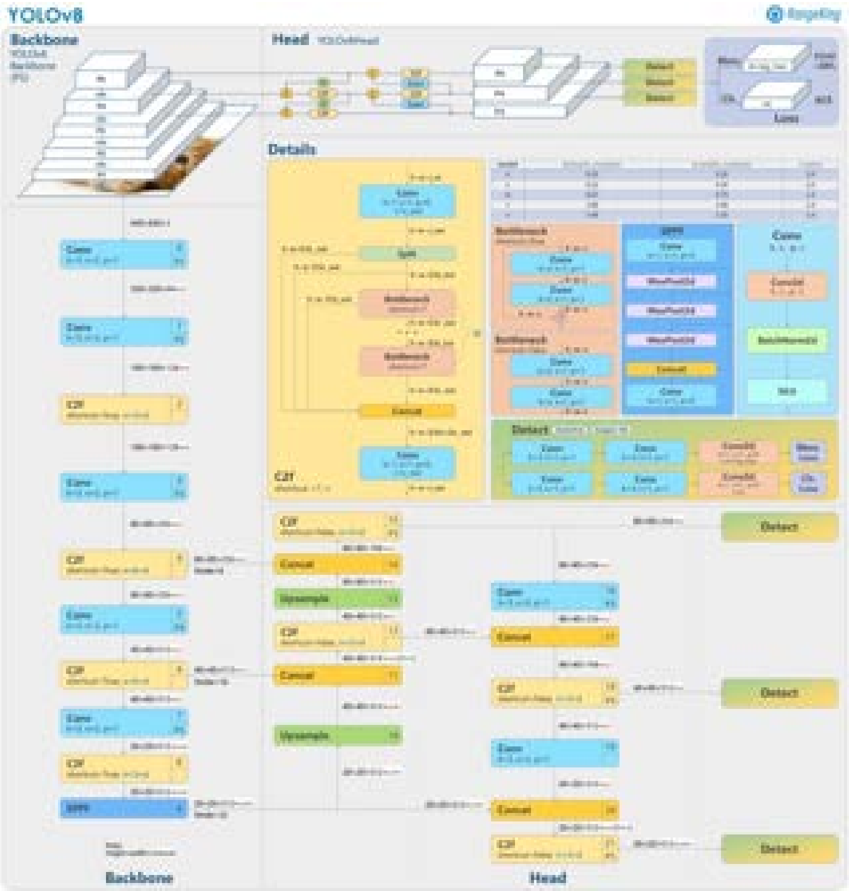


Figure 2.5: YOLOv8 Architecture[11]

### DINO: DETR with Improved deNoising anchOr boxes

As introduced in the previous sections, impressive progress has been achieved with various classical, convolutional architectures during the recent years, like Faster-RCNN, or the YOLO family[80;52;53;89]. After the introduction of the transformer architecture[13] and the successful application of transformers in image classification tasks with ViT, transformers also found the way into other computer vision tasks[90]. Inspired by the original ViT, the same ideas were adapted to object detection problems with the paper by Beal et al.[80] who proposed the ViT-FRCNN network.

In recent years, attention-based architectures with representatives such as Dynamic Heads (DyHead)[91], Swin[92], or HTC++[93] have been developed rapidly and conquered the first places in the COCO test-dev leader board[59]. However, all of these architectures have a need for hand-designed components like anchor generation[25] or non-maxima suppression (NMS)[94;12]. The first transformer-based end-to-end object detection architecture, called Detection Transformer (DETR) proposed by Carion et al.[55], made these additional steps redundant[12]. At the time of writing, DETR-based architectures are among the most performant object detection models[59].

In this project, an optimised version of the original DETR model is fitted, called DINO, which stands for **D**etection Transformer with **I**mproved de**N**oising anch**O**r boxes[12]. DINO is a combination of different improvement attempts for DETR regarding training efficiency, training robustness, and performance in general[12]. DINO follows DN-DETR[58] which itself follows the DAB-DETR[57] architecture to improve training efficiency. Moreover, DINO includes several techniques of deformable DETR[95] such as the deformable attention mechanism to achieve better performance.

Additionally to the advances of previous DETR descendants, the DINO architecture proposes three new methods[12]. The first method, called *contrastive denoising training*, helps the model avoiding duplicate outputs of the same target[12]. The second is a *mixed query selection* method which improves the initialisation of the queries; the third is a *look forward twice* mechanism to improve training efficiency by optimising parameters of early layers with the gradient information of later layers[12].

In the original paper of Zhang et al.[12], the backbone of the DINO architecture to generate multi-scale features builds either a CNN-based architecture, for instance a ResNet[74], or a transformer-based architecture like Swin[92]. In order to obtain a comparable result of the transformer-based architecture with CNN-based recognition models, only DINO with a Swin backbone is considered in this work. Furthermore, the DINO head implementation makes use of a focal loss[96] as classifier loss, an L1 loss[97] as bounding box loss, and a generalised IoU loss (GIoU-Loss)[98] as IoU loss. Proposed by Lin et al.[96], focal loss as an extension of the standard cross-entropy is advantageous to apply for one-stage detection models on class imbalanced datasets. Finally, the architecture of the DINO model can be viewed in figure 2.6.

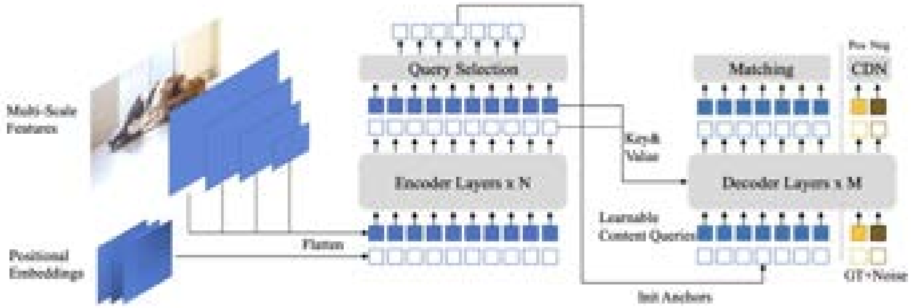


Figure 2.6: DINO Architecture[12]

## 2.3 Metrics

In the interest of comparability and reproducibility of the experiments, it is essential to define the metrics for test evaluations and their rationale[110]. There are several opinions in the literature on how to select expressive metrics for supervised learning classification and object detection tasks. On the one hand, a metric should accurately assess and describe the real performance of a model. On the other hand, it should ensure the intuitive interpretability of the results for humans[111;112;113]. Ng[111] suggests using a single metric for an efficient comparison between consecutive experiments, whereas Doshi-Velez and Kim[113] emphasise the inability of a single metric, such as the accuracy, to describe all facets of a performance result[113]. In this report, a combination of both approaches is applied. First, a single, primary metric is identified for a straightforward and elementary comparison between the experiments. Second, additional metrics should be computed to allow a more in-depth analysis of a change in performance, the differences between classes of a dataset or the sensitivity of a model to hyperparameter changes. The aim of this section is to present possible metrics that can be used in the experiments conducted in this project.

### 2.3.1 Accuracy

Accuracy is the most commonly used performance measure in classification problems, due to its simple interpretability and straightforward calculation[114;115]. However, it is often criticised for not being able to capture all facets of a model's classification performance and for failing with imbalanced datasets[116;117]. Accuracy is defined by[118;117]:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can also be defined as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

### 2.3.2 Precision, Recall, F1 Score

To mitigate the shortcomings of accuracy, precision or recall, the F1 score can be used. This especially applies for imbalanced datasets[114]. Whereas precision is a measure of relevance, recall describes how many relevant results are returned[114]. The F1 score can be interpreted as a harmonic mean between precision and recall and is commonly used as it combines both metrics[119;111].

Because precision and recall are originally defined for a binary classification problem, an extended version of the metrics is required for a multi-class classification problem. Mainly two methods exists. First, the macro-averaged form, in which the metrics are assessed in a one-against-all approach for each class, followed by taking the mean over the partial results[114]. Second, the weighted average of the metrics. This form works similarly, but the distribution of the classes is also taken into account to avoid an overestimation of certain classes in an imbalanced dataset[119]. To have an alternative to accuracy in experiments with unequally distributed classes, only the weighted form of the metrics is considered for the experiments of this project. Formally, the three metrics are defined by[120]:

**Precision**

$$\text{Weighted Precision} = \frac{\sum_{i=1}^{n} |y_i| \frac{TP_i}{TP_i + FP_i}}{\sum_{i=1}^{n} |y_i|}$$

**Recall**

$$\text{Weighted Recall} = \frac{\sum_{i=1}^{n} |y_i| \frac{TP_i}{TP_i + FN_i}}{\sum_{i=1}^{n} |y_i|}$$

**F1 Score**

$F1_i = 2 \times \frac{PRE_i \times REC_i}{PRE_i + REC_i}$

With $F1_i$ = F1 score for class $i$, $PRE_i$ = Precision for class $i$ and $REC_i$ = Recall class $i$

Weighted F1 Score $= \frac{\sum_{i=1}^{n} |y_i| \times F1_i}{\sum_{i=1}^{n} |y_i|}$

### 2.3.3 Confusion Matrix

In machine learning, the confusion matrix is typically used to visually evaluate the performance of a model in a supervised classification problem[121;122]. Especially in a multi-class environment, the confusion matrix can help to understand the performance of the model on different classes and allows identifying the classes on which the model performs well and those on which it discriminates poorly (is confused)[121]. This is fairly impossible by only considering a single metric such as accuracy, or requires much more effort to study precision and recall for every single class. Moreover, all three metrics can easily be derived from a confusion matrix[121]. Figure 2.7 shows an example of a confusion matrix presented in this project. Note that for all confusion matrices calculated in this work, the predicted labels are shown on the x-axis, while the true labels are shown on the y-axis.

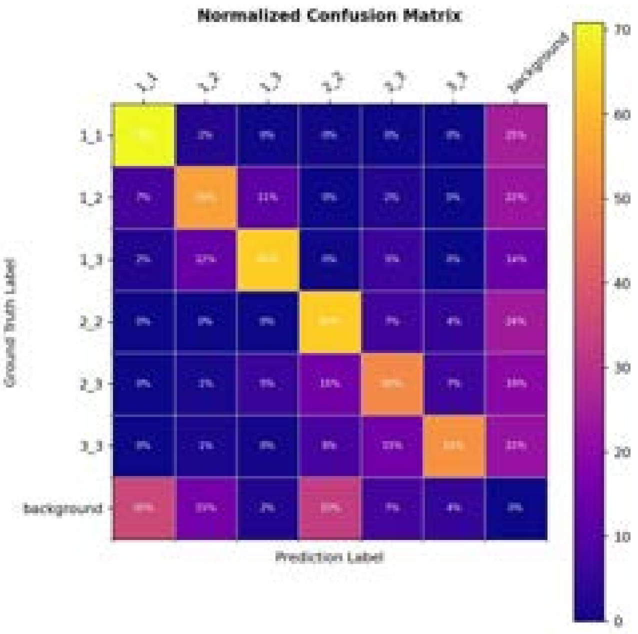


Figure 2.7: Sample Confusion Matrix[123]

### 2.3.4 Precision-Recall (PR) Curve

Computing the precision-recall curves can have two main advantages for us. First, it provides a supplementary visualisation on how well a model performs by visualizing the trade-off between precision and recall (also called sensitivity)[124;125]. Specifically, the PR curves in this project describe the performance metric for a single class[124] such as $1\_1$ for which an example is shown in figure 2.8. Definitions for Precision and Recall can be found in the previous sections. Second, it offers the area-under-the-precision-recall-curve (AUPRC) score as an additional metric to accuracy and F1 score[125]. The value of the AUPRC is ranging from zero to one, with $< 0.5$ to be a fully random prediction and $1$ as the most preferable outcome[126].

The PR curves used in this work have different layers to allow a more in-depth error analysis. The layers shown in figure 2.8 correspond to the definitions of the official COCO[59] metrics documentation and are specified as follows[127;128]:

1. C75 (White): PR at $IoU = 0.75$ (AP at strict IoU), area under curve corresponds to $AP^{IoU} = 0.75$ metric.
2. C50 (White): PR at $IoU = 0.50$ (AP at PASCAL IoU), area under curve corresponds to $AP^{IoU} = 0.50$ metric.
3. Loc (Blue): PR at $IoU = 0.10$ (localisation errors ignored, but not duplicate detections). All remaining settings use $IoU = 0.10$.
4. Sim (Red): PR after super category false positives (fp) are removed. Specifically, any matches to objects with a different class label but that belong to the same supercategory don't count as either a fp (or tp). Sim is computed by setting all objects in the same supercategory to have the same class label as the target class and setting their ignore flag to 1. Note that person is a singleton supercategory so its Sim result is identical to Loc.
5. Oth (Green): PR after all class confusions are removed. Similar to Sim, except now if a detection matches any other object, it is no longer a fp (or tp). Oth is computed by setting all other objects to have the same class label as the class in question and setting their ignore flag to 1.
6. BG (Purple): PR after all background (and class confusion) fps are removed. For a single category, BG is a step function that is 1 until max recall is reached then drops to 0 (the curve is smoother after averaging across categories).
7. FN (Orange): PR after all remaining errors are removed (trivially AP=1).

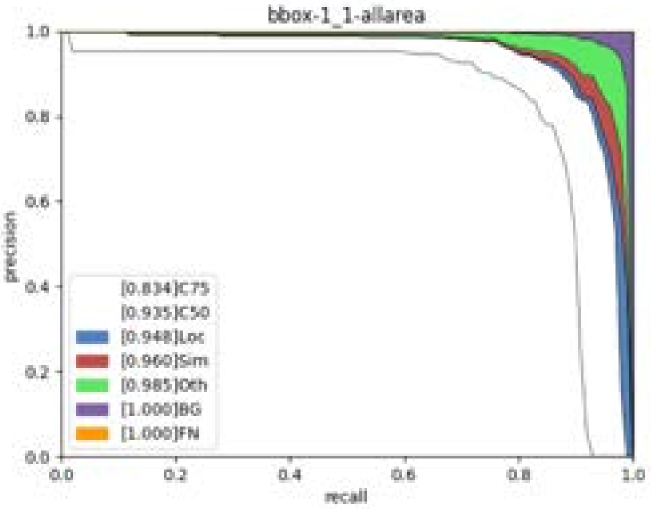


Figure 2.8: Sample ROC Curve[127;128]

### 2.3.5 Intersection over Union (IoU)

Intersection over Union (IoU) is the most prominent evaluation score to verify the detection performance of object detection models in literature[129].

The metric is evaluated by calculating the ratio between the intersection of the predicted with the ground truth bounding box and the union of the two boxes. A bounding box is thereby defined by its location and size. The IoU score can be formulated as follows[129]:

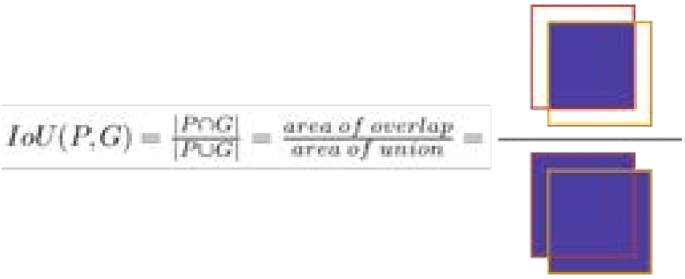

$IoU(P,G) = \frac{|P \cap G|}{|P \cup G|} = \frac{area\ of\ overlap}{area\ of\ union} =$

Figure 2.9: Intersection over Union (IoU)[129]

Where $P$ is the predicted bounding box and $G$ is the ground truth bounding box. In this work, both a separate IoU value for each class and a global, weighted average value for all classes are evaluated on the validation and test dataset.

### 2.3.6 Mean Average Precision (mAP)

If a single metric for the evaluation of an object detection method is required, the mean average precision (mAP) is a widely used strategy[130]. The mAP is a combination of many evaluation concepts for machine learning models[130]. Specifically, it combines the concepts of the confusion matrix, the IoU score and the precision recall curve with the associated AUPRC score. To evaluate the mAP, the average precision (AP) needs to be evaluated first. Unlike the name suggests, the AP can be considered as a way to calculate the AUPRC. To reduce the computational efforts to calculate an integral, different alternatives exists[130]. In this example, the $11$-point $AP_{11}$ is considered. The $AP_{11}$ is calculated by taking an average of the maximum precision scores above 11 equally spaced recall thresholds: $R = [0.1, 0.2, ..., 1]$

$AP_{11} = \frac{1}{11} \sum_{r \in R} P_{interp}(r)$

where

$P_{interp}(r) = \max\limits_{\tilde{r}:\tilde{r} \geq r} P(\tilde{r})$

As mAP is a metric to rate the accuracy of a model in a multi-class problem, the last step to calculate the mAP is to average AP over all classes in a dataset.

$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$

with $N$ as the number of classes.

Because the primary goal of this work is to train a model which can classify different sources within a radio source dataset, mAP is taken as the primary metric for the evaluation of detection models.

## 2.4 Preprocessing Techniques

Radio source images exhibit a significant amount of noise, show a narrow distribution of distinct intensity values and in case of the RGZ dataset, are represented in mJy beam$^{-1}$[42]. To ensure a stable and fast training and possibly even achieve an increase in performance, appropriate preprocessing of radio source images is essential in this project[131]. Moreover, some libraries applied in this project require a PNG or JPG file format. This chapter aims to briefly introduce the compared preprocessing methods within the dataset conversion pipeline. Finally, the impacts of the subsequently described techniques on the intensity distribution of the radio source images are visualised in appendix A.2.2.

### 2.4.1 Scaling Techniques

Scaling is an important preprocessing step to allow efficient and robust training and to prevent exploding or vanishing gradients[131]. Moreover, scaling is used to retain values within defined boundaries, for example between zero and one[131]. Finally, scaling can also be used to arrange the data values in a desired distribution, such as the normal distribution[131]. This work makes use of two well-known scaling techniques which are described in the next sections.

#### Z-Scale

The z-scale algorithm was originally developed by the National Optical Astronomy Observatory and contained in the IRAF Framework[132]. It is designed to display the intensity values near the median image[132]. According to Tody[132], this is especially valuable for astronomical images, as they typically exhibit a very peaked histogram in comparison to the background sky.

If the contrast is not zero, the sample pixels are ranked in brightness to form the function $I(i)$ where $i$ is the rank of a pixel and $I()$ returns the intensity value[132]. Generally, the median is very near the peak of the image histogram and there is a well-defined slope about the median which is related to the width of the histogram[132]. At the ends of the intensity distribution ($I(i)$), there are a few very bright and dark pixels due to objects and defects in the field[132]. To determine the slope, a linear function is fit with iterative rejection[132].

```
I(i) = intercept + slope * (i - median)
```

#### Min-Max Scale

In image preprocessing, min-max scaling has shown advantageous if the boundaries of the original images are known as it keeps the scale of the input data[133]. Min-Max scaling is often called normalisation and can be achieved by using the following formula:

$Y = \frac{X - X_{min}}{X_{max} - X_{min}}$

### 2.4.2 Sigma Clipping

Although deep neural networks should be able to cope with noisy data, a common method used in astronomy to reduce noise during pre-processing is sigma clipping[42]. With sigma clipping, all values that deviate more from a defined centre function than a certain number of standard deviations are essentially clipped or discarded. In this work, either the median or the mean is used as centre function. Finally, the following rules are applied to evaluate the values out of bounds.

```
data < centre - (sigma_lower * std)
```

```
data > centre + (sigma_upper * std)
```

Where `sigma_lower` and `sigma_upper` is a scalar number.

### 2.4.3 Stretching Techniques

Because the dynamic range can be narrow in astronomical images, it can be advantageous to stretch images[134;135;136]. This also accounts for cases when the distribution of intensity values is different across samples of a dataset, that is, when some images are visually black and for others the radio source can be recognised by eye. If this distinction is not a discriminative factor, it can hinder the performance of a model. One family of methods to widen the distribution of intensity values across a defined interval is called stretching methods. Table 2.3 shows the investigated stretching methods in this project.

| **Name** | **Square Root Stretch** | **Power Stretch** | **Power Dist Stretch** |
|---|---|---|---|
| **Definition** | $Y = \sqrt{X}$ | $Y = X^a$ | $Y = \frac{a^X - 1}{a-1}$ |
| **Parameters** | - | $a = 0.62\ [a > 0]$ | $a = 1000.0\ [a >= 0]$ |
| **Name** | **Linear Stretch** | **Log Stretch** | **Sinh Stretch** |
| **Definition** | $Y = m \cdot X + q$ | $Y = \frac{\log(aX+1)}{\log(a+1)}$ | $Y = \frac{sinh(\frac{X}{a})}{sinh(\frac{1}{a})}$ |
| **Parameters** | $m = 1.6\ ;\ q = 0$ | $a = 1000.0\ [a > 0]$ | $a = \frac{1}{3}\ [0 < a <= 1]$ |

Table 2.3: Overview Stretching Techniques[134;135;136]

## 2.5 Augmentation Techniques

Data augmentation is a widely used strategy when data is scarce, imbalanced, or when there is explainable variance within the data which can be generated synthetically[137]. Moreover, data augmentation techniques are often used to mitigate the problem of overfitting[137]. However, data augmentation should only be used if it is unfeasible to gather alike samples from the real data distribution[137]. The main idea behind data augmentation is to apply label preserving transformations to an input dataset in order to add more invariant examples[138].

Formally, to a sample-label pair $(x,y)$ a transformation function $\Phi(x)$ with a certain probability $p_{aug}$ is applied *a priori*[139]. If a model is trained accordingly, it should theoretically output the same probability distribution for the augmented input $P(y|\Phi(x))$ as for the same input sample without the applied transformation $P(y|x)$[139].

As the goal of this work is to detect and/or classify radio sources, our intuition is, only a few, out of the well-known basic augmentation techniques do not change the sample out of the real distribution. Since each detected source is cropped such that it lies approximately in the centre of the image, and since the existing distance between the sources in the large original image is large enough compared to the source size, most samples contain only one type of radio source in an image (besides the fact that classes already encompass multiple components). However, as specific augmentation techniques were leading to an increased performance in some of the applied model architectures in this work[15;77;11], this intuition should be verified. Therefore, the following augmentation techniques were applied to either improve model performance or to prove that the technique has a negative effect for radio source detection or classification.

First, rotation or flipping is applied with limited rotation angles of $90°$ steps. The second and third evaluated augmentation techniques are translation and scaling, bounded by a minimum and maximum translation respectively scaling ratio. The last and most vague verified method is shearing, initialised by setting a maximum shearing degree.

Used in the original EfficientNetV2 paper, RandAugment lead to significant performance improvements[77]. The primary goal of RandAugment is to remove a separate augmentation search phase to ease and speed up training[140]. The technique can be applied for any set of any per-image based augmentation technique[140]. The simplification is achieved by reducing the number of hyperparameters to two parameters[140]. One parameter $N$ describes the number of randomly chosen techniques out of the predefined set[140]. A second parameter $M$ defines the magnitude of the applied method within an interval of $(0,10)$. For both values, the regularisation also increases when the values are increased[140].

# Chapter 3

# Methods

This chapter aims to introduce the different experiments performed in this project and further sets the scope of possible experiments within the given time frame.

## 3.1 Preprocessing Experiments

The first type of experiments conducted in this work are preprocessing experiments. In this type of experiments, the preprocessing techniques presented in chapter 2.4 are verified in a grid search in order to determine whether a suitable scaling mechanism has a significant influence on the performance result of a trained model. For the evaluation, a basic YOLOv3[101] model is used with the default settings proposed by the `MMDetection` framework and with no additions to the pipeline like augmentation. In the end, the two most promising scaling and preprocessing settings are used in the subsequent model benchmark experiments.

## 3.2 Benchmark Experiments

The second and main kind of experiments performed are the model benchmark experiments. Hereby, a selected set of models are trained and evaluated with basic adjustments of the training pipeline. Furthermore, the benchmark experiments aim to provide a statement on the performance of modern transformer based architectures in detecting and classifying radio sources compared to CNN-based models, especially with the limited number of samples available in the RGZ OD dataset[63]. The selected object detection architectures in this project are YOLOv8[11] as CNN and DINO[12] with a Swin[92] backbone as transformer based representative. For classification, ResNet50[14] and EfficientNet[15] constitute the selected CNN based architectures, and Vision Transformer[84] as well as EfficientFormer[16], the transformer based models. In the basic benchmark experiments, no upscaled model representations of the architectures were used, as the resolution of the images is relatively low at $132 \times 132$ pixels and the dataset size is rather limited with approximately $12'000$ samples. This to mitigate the risk of overfitting. However, this hypothesis needs to be verified in the extending tuning experiments.

The basic adjustments to the pipeline configuration of the basic benchmark experiments include min-max scaling and z-scaling, the decision whether to use pre-trained weights on ImageNet[141] for classification models and on COCO[59] for object recognition models, and finally the distinction between trained models on the entire unbalanced dataset or the reduced balanced dataset with undersampling. This procedure allows drawing a general conclusion about the efficacy of the model architecture for radio source datasets with a limited number of samples. In the end, the combination of all settings and model architecture leads to $48$ performed basic benchmark experiments.

## 3.3 Ensemble Experiment

To examine the uncertainty of the model or epistemic uncertainty depending on the data quantity and quality at hand, the ensemble method is applied in this work[142;143;144;145]. There are multiple methods how to calculate the ensemble for a classification task, as well as calculating the uncertainty between the models. As definition for the ensemble prediction, this work simply uses the average prediction of all trained models. Further, it assumed that the networks have a SoftMax-Layer as final prediction layer.

$$p(y|x,\theta_0,...,\theta_M) = \frac{1}{M}\sum_{i=0}^{M} f(x,\theta_i) = \frac{1}{M}\sum_{i=0}^{M} y_i = \overline{y} \tag{3.1}$$

The epistemic uncertainty can then be calculated by using the variance or standard deviation between the predictions of the different models[145] or by utilizing the entropy of the probability distribution[144;142;146]. For the matter of this work, using the method of calculating the standard deviation is sufficient to evaluate the epistemic uncertainty $U_{ep}$, which is defined by:

$$Var_y = \sigma_y^2 = \frac{1}{M}\sum_{i=1}^{M}(y_i - \overline{y})^2 \tag{3.2}$$

$$U_{ep} = \sigma_y = \sqrt{Var_y} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \overline{y})^2} \tag{3.3}$$

The ensemble experiment is performed on the most promising training result and setting of the benchmark experiments with $M = 30$.

## 3.4 GLEAM Experiments

GaLactic and Extragalactic All-sky MWA Survey (GLEAM) is an extragalactic catalogue containing $307'455$ radio sources, measured across $72 - 231$ MHz by the Murchison Widefiled Array (MWA)[68]. The GLEAM experiments of this work aim to use the catalogue as first verification of the transferability of the most promising model evaluated in the benchmark experiments to a new or unseen dataset. As Murchison Widefiled Array is part of the SKA-Low1[68], it can offer a first impression about how state-of-the-art models could be applied for future catalogues and surveys obtained by SKA and what performance could be expected[3;4;2]. However, it is not within the scope of the project to perform in-depth analysis of the test runs performed on the GLEAM catalogue and rather let to future work.

## 3.5 Tuning Experiments

The final set of experiments conducted in this project are model tuning experiments to improve the results obtained with the present datasets. To reduce the number of required experiments within the scope of this work, only the most promising model of the benchmark experiment is further fine-tuned. The experiments encompass pipeline adjustments in terms of preprocessing, model scaling adjustments with respect to the number of parameters as well as hyperparameter tuning.

More specifically, a further preprocessing method to convert a greyscale input image $x_g \in R^{H \times W \times 1}$ into a colour image of dimension $x_c \in R^{H \times W \times 3}$, called dynamic weights equations[147] is applied, which showed positive result in the preprocessing method evaluation. In respect to augmentation, a limited number of reasonable techniques are verified, which were introduced in section 2.5. Further it is investigated, whether the model achieves better performance when trained with the original input image size of $132 \times 132$, or with resizing to the model's pre-trained image size of $640 \times 640$ for the COCO[59] and $224 \times 224$ in case of the ImageNet[141] dataset. All the applied models offer different sizes in terms of the number of parameter, with respect to depth, width and resolution. Therefore, additional tests are conducted, whether stronger models lead to better performance or end up in overfitting the training data. Finally, standard hyperparameter tuning is performed regarding optimiser parametrisation and model specific parameters.

# Chapter 4

# Implementation

A main goal of this project is to set up a training pipeline that can be flexibly applied to different model architectures and datasets. The focus of this chapter is to describe the technical details about the pipeline setup and should indicate what kind of adjustments to the pipeline can be made to reproduce or even extend the performed experiments of this work.

## 4.1 Libraries

This project depends on several libraries, of which the most important should be introduced in this section. Images in astronomy related fields are often stored in the Flexible Image Transport System (FITS) file format[65]. In order to load and process this file format, the `Astropy` library was used[135]. Moreover, `Astropy` offers several methods to convert, preprocess or visualise FITS images such as scaling or conversion between different coordinate systems[135].

Allowing a simple dataset transportation between different servers, the HDF5 file format was utilised to store the images as well as class and bounding box annotations, together with additional meta data attributes in a single file[148]. The HDF5 then builds the source to generate a specific dataset for either an object detection model or a classification model. To work with HDF5 files in `Python`, at the time of this project, the `H5Py` library is the most complete library for this task.

To train multiple architectures with a single pipeline, a flexible toolbox or framework is required to simply switch between different existing state-of-the-art object detection or classification models. For that reason, the project pipeline is based on the `OpenMMLab` algorithm system called `MMEngine`[149]. `MMEngine` is `OpenMMLab`'s foundational library to train deep learning models based on `PyTorch`[149;150]. It embodies the basic pipeline called runner for a variety of subprojects of the `OpenMMLab` suite[149]. The subprojects are mostly structured depending on their application or problem they want to solve. There exists, subprojects for classification, object detection, pose estimation, optical character recognition (OCR), video object tracking and many more[149]. In this project, `MMDetection`[123] is applied for object detection, `MMPretrain`[151] for classification and `MMYolo`[152] for the YOLO object detection architectures. The reason to use a separate project for the YOLO family was, that the current `MMDetection` version only supports YOLOv3 and YOLOX[123]. In order to train more up to date YOLO models like YOLOv8, it was necessary to include support for `MMYOLO` in training pipeline[151].

## 4.2 Requirements

For the implementation of the training pipeline `Python 3.10.x` was used as primary programming language. All basic required dependencies can be found in the `requirements.txt` in the root folder of the project. Additionally, there are requirements which depend on the availability of GPUs or CPUs and can be found in the requirements file specified with either cpu or gpu suffix (e.g. requirements-gpu.txt). Additional information about the environment and setup is given in the README.md file.

After running the pipelines, all resulting checkpoints, log files and plots are stored in the project's work_dirs folder. Beyond the visual plots, the project depends on the platform Weights & Biases (WandB) for metrics and progress logging[153]. However, it is not mandatory to use WandB to run the project and can rather be configured in a dedicated configuration file. For security reasons, the credentials to connect to the WandB project are stored in a separate .env file, which is not checked in to the version control system. The required environment variables to successfully connect to WandB are also explained within the README.md file.

## 4.3 Project Structure

Primarily the project is structured into configuration files, executable python scripts, services and utilisation modules as well as the framework home directories of the OpenMMLab project. This section should give a high-level overview about the structure and reason of the different modules.

First, every OpenMMLab subproject obtained a folder. Cloning the dedicated git repository of any subproject is the recommended installation method over a direct installation via PIP package manager. By design, this cloned project folder would constitute the project home folder for a classic OpenMMLab project. Because it was the goal in this project, to have a single repository unifying all model architecture in a single pipeline, it was not possible to follow this setting. Every subproject folder contains a config folder in which the predefined Python configuration files for OpenMMLab Runner can be found. The configuration files act as starting point for any custom configuration implementation of a training pipeline and are designed in a tree like inheritance structure. Unlike MMDetection, MMPretrain and MMYolo contain similarly named package folders. To unify the different subprojects, it was required to move the module folder to the top level of the repository and renaming the subproject's home folders to mmpretrain-scripts and mmyolo-scripts.

Second, the project contains an own config folder. The configuration folder comprises a tree like structure of YAML files, orchestrated by the Hydra framework[154]. The idea of the additional configuration files is to have a central place of configurations for all supported scripts and pipelines. Possible configurations exist for the dataset preprocessing, the model training and fine-tuning as well as the settings for logging like the configurations for WandB[153].

The main executable Python scripts are located in the src folder of the project. There are four main type of scripts in the src directory. The dataset folder contains scripts to generate the base dataset collection. After the generation of the HDF5 dataset collection, combining images of the RGZ, FIRST and Mira Best dataset, the conversion scripts can be used in the similarly named folder. The scripts allow converting any base dataset into a COCO[59] formatted, in a basic classification dataset referenced by filenames or by annotation files[59]. The actual training script is located in the src/train folder, together with a Dockerfile that can be used to create and run an executable Docker container for the training pipeline. Finally, the adjusted Python configuration files which overwrite the default OpenMMLab training pipeline configurations are located in src/models.

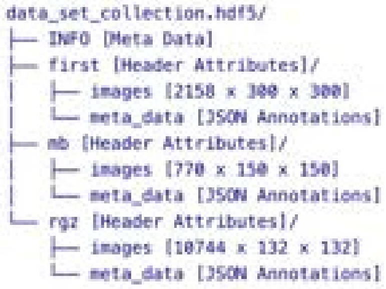
## 4.4 Pipelines

The project is based on three pipelines. The base dataset collection pipeline, the dataset conversion pipeline and the training and evaluation pipeline. This chapter aims to introduce the most important concepts and components of these pipelines.

### 4.4.1 Dataset Collection

The main goal of the base dataset collection pipeline is to unify and combine different datasets into a common dataset format from which it can be converted in different target dataset formats. For the unification of multiple datasets, the pipeline makes use of the HDF5 file format. It stores the dataset in a hierarchical tree structure which is shown in figure 4.1. Within each of the three included datasets FIRST (first), Mira Best (mb), Radio Galaxy Zoo (rgz), there are two arrays stored. One for all images and another for all metadata information such as annotations, which are stored in a list of JSON objects. An example of an annotation JSON can be found in figure A.1 of the appendix.

During the generation of the base dataset collection, only basic preprocessing is applied to ensure the validity of the HDF5 arrays. This includes converting non-number-values into zeros, floating point values into $32$-bit floating point values and crop or fill deviating image dimension with zeros. This practice could be applied without further adjustment of annotations or later impacts, because the maximum deviation of dimension in all datasets was three pixels. Ultimately, the collection contains $13'672$ images and annotations with a size of $630$ MB.



```
data_set_collection.hdf5/
├── INFO [Meta Data]
├── first [Header Attributes]/
│   ├── images [2158 x 300 x 300]
│   └── meta_data [JSON Annotations]
├── mb [Header Attributes]/
│   ├── images [770 x 150 x 150]
│   └── meta_data [JSON Annotations]
└── rgz [Header Attributes]/
    ├── images [10744 x 132 x 132]
    └── meta_data [JSON Annotations]
```

Figure 4.1: Tree-Structure Dataset Collection

### 4.4.2 Preprocessing

The main preprocessing is achieved during the generation of the target dataset, depending on the requirements of the models applied. As an overview, the preprocessing pipeline, shown in figure 4.2, performs the following tasks. Based on the HDF5 dataset collection which is described in section 4.4.1, the images and annotations can be converted into a target dataset format depending on the source dataset. This means that, for example, the RGZ dataset containing images and bounding box annotations can be converted into a COCO[59] formatted dataset for object detection, in a classification dataset based on annotation files or based on a class specific directory tree. Within the preprocessing pipeline, two filters can optionally be applied. One to balance the dataset by undersampling. A second to reduce the samples to those with only one radio source per sample, which allows training a single label classification model. Further, the pipelines offer configuration based offline preprocessing steps, which can be either performed globally on the entire dataset or sample based. A list of supported preprocessing mechanisms can be found in appendix A.2.

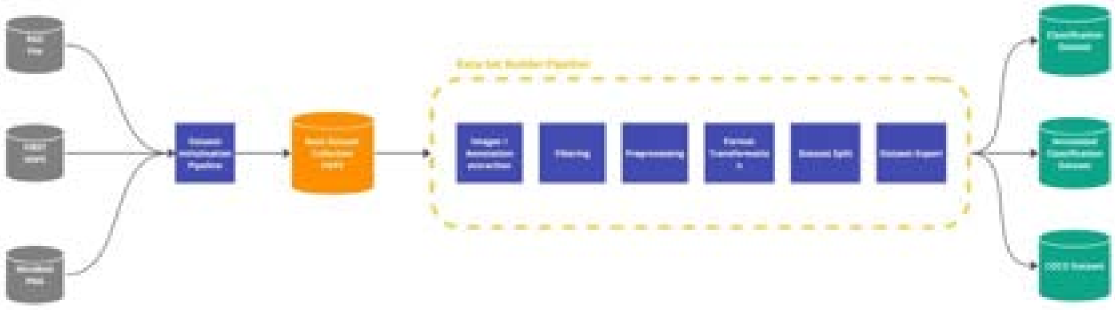


Figure 4.2: Preprocessing Pipeline

### 4.4.3 Training and Evaluation

The training pipeline orchestrates the experiment configuration, the model training, the model evaluation and the experiment tracking into one single executable pipeline. A visualisation of the components of the training pipeline is shown in figure 4.3. As input to the pipeline, three artefacts are required. A Hydra[154] `YAML` pipeline configuration file, a `MMengine`[149] Python configuration file and the configured dataset. Depending on the configuration of an experiment, the pipeline either starts a Weights and Bias (WandB)[153] run on itself or initialises a managed model fine-tuning with a sweep over specified hyperparameters. An experiment can either be started in production or development mode. Conditioned by the model family, the pipeline selects the specific `MMLab` sublibrary to run the training of the model. Evaluation on the validation set is performed in an interval of a specified number of epochs, while the evaluation of the test set is performed after the model training in a dedicated sub-pipeline. Further pipeline steps are executed to generate result plots such as precision recall curves and confusion matrices. For a brief comparison of the conducted experiments, all metric, plots and samples of the validation and test set are uploaded to (WandB) along the log files and the model weights. Moreover, all results and files are stored on the computing instance on which the training was performed, which can be used for a more in-depth analysis described in the next chapter.

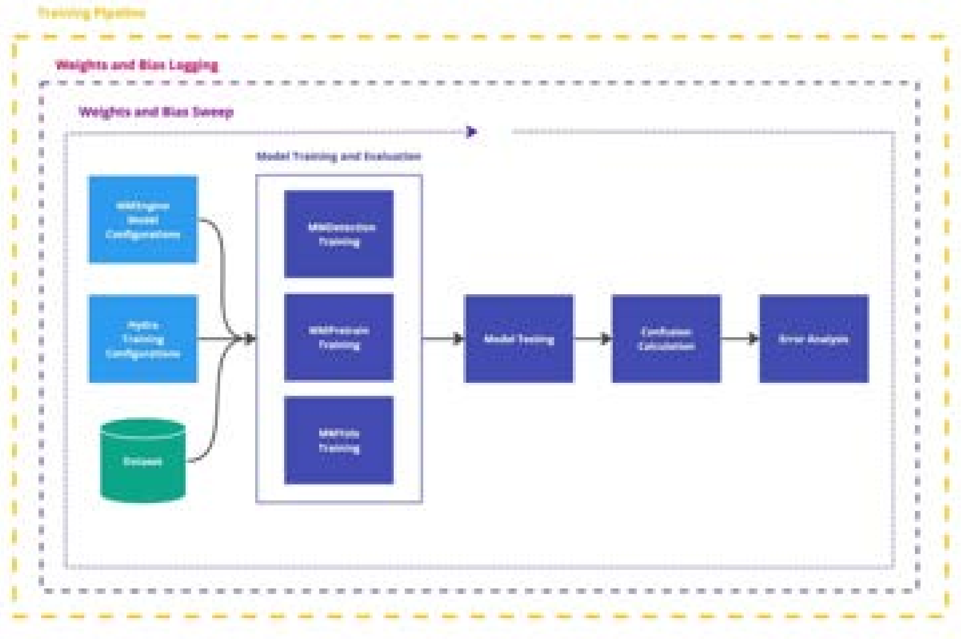


Figure 4.3: Training Pipeline

## 4.5 Analysis Notebooks

Beside the main pipelines, the project encompasses supporting analysis notebooks. The dataset exploration notebook (`dataset_exploration.ipynb`) is used to investigate the different dataset sizes, class distributions and evaluates how the filters affect the number of samples of a dataset. A second notebook (`image_preprocessing.ipynb`) is used to investigate the effects of the various preprocessing mechanisms listed in the appendix A.2 with regard to the change in the intensity value distribution. Lastly, a third notebook (`error_anaylsis.ipynb`) is applied to perform an error and ensemble analysis of trained models, as well as comparing the errors made between different training experiments.

# Chapter 5

# Results

Numerous experiments were conducted according to the experiment types introduced in chapter 3. The aim of this chapter is to summarise the results of performed experiments whereas the detailed plots to the experiments can be found in the appendix.

## 5.1 Preprocessing Experiments

To verify possible preprocessing techniques introduced in chapter 3.1, $24$ experiments were performed with a lightweight YOLOv3 model[101]. Table 5.1 verifies that the biggest influence on the performance is obtained by using the z-scaling technique. Further, the experiments show that sigma clipping either by mean or by median can have a positive effect. Also, dynamic weights conversion can have an advantageous influence, however, regarding the results also simply replicating the channels, together with initialising the weights randomly does not negatively influence the performance. On the opposite, the bunch of stretching methods examined often do not lead to a favourable result, even if those methods increase the dynamic range within the image. In the end, for tuning purposes, z-scaling, sigma clipping and dynamic weights conversion should be considered for continuous experiments.

| Model | Preprocessors | IOU | Loss | Accuracy | MaP |
|---|---|---|---|---|---|
| YOLOv3 | ZScale, SigmaMedianClip, ReplicateChannels | 0.76 | 24.55 | 0.76 | 0.49 |
| YOLOv3 | SigmaMeanClip, ZScale, ReplicateChannels | 0.75 | 25.78 | 0.73 | 0.47 |
| YOLOv3 | ZScale, DynamicWeightsConversion | 0.78 | 24.00 | 0.75 | 0.45 |
| YOLOv3 | ZScale, ReplicateChannels | 0.76 | 26.18 | 0.75 | 0.45 |
| YOLOv3 | PowerStretch, ZScale, ReplicateChannels | 0.77 | 25.82 | 0.74 | 0.44 |
| YOLOv3 | SqrtStretch, ZScale, ReplicateChannels | 0.77 | 24.91 | 0.72 | 0.43 |
| YOLOv3 | SqrtStretch, ZScale, DynamicWeightsConversion | 0.77 | 25.53 | 0.73 | 0.42 |
| YOLOv3 | ZScale, SigmaMeanClip, ReplicateChannels | 0.76 | 25.20 | 0.72 | 0.42 |
| YOLOv3 | SigmaMedianClip, ZScale, ReplicateChannels | 0.74 | 24.94 | 0.73 | 0.41 |
| YOLOv3 | PowerStretch, ZScale, DynamicWeightsConversion | 0.78 | 25.21 | 0.73 | 0.37 |
| YOLOv3 | PowerStretch, MinMaxScale, ReplicateChannels | 0.74 | 25.71 | 0.67 | 0.33 |
| YOLOv3 | SinhStretch, ZScale, ReplicateChannels | 0.74 | 26.57 | 0.64 | 0.31 |
| YOLOv3 | LinearStretch, ZScale, ReplicateChannels | 0.75 | 27.31 | 0.64 | 0.31 |
| YOLOv3 | MinMaxScale, ReplicateChannels | 0.74 | 27.22 | 0.67 | 0.31 |
| YOLOv3 | MinMaxScale, SigmaMedianClip, ReplicateChannels | 0.72 | 26.70 | 0.66 | 0.30 |
| YOLOv3 | SigmaMedianClip, MinMaxScale, ReplicateChannels | 0.72 | 28.11 | 0.65 | 0.30 |
| YOLOv3 | SqrtStretch, MinMaxScale, ReplicateChannels | 0.73 | 25.86 | 0.65 | 0.30 |
| YOLOv3 | SinhStretch, MinMaxScale, ReplicateChannels | 0.74 | 27.89 | 0.65 | 0.28 |
| YOLOv3 | LinearStretch, MinMaxScale, ReplicateChannels | 0.73 | 27.02 | 0.67 | 0.27 |
| YOLOv3 | MinMaxScale, DynamicWeightsConversion | 0.72 | 28.96 | 0.65 | 0.27 |
| YOLOv3 | PowerDistStretch, MinMaxScale, ReplicateChannels | 0.68 | 33.08 | 0.58 | 0.22 |
| YOLOv3 | PowerDistStretch, ZScale, ReplicateChannels | 0.43 | 137.95 | 0.16 | 0.00 |
| YOLOv3 | LogStretch, ZScale, ReplicateChannels | 0.43 | 135.35 | 0.12 | 0.00 |

Table 5.1: Results Preprocessing Experiments

## 5.2 Benchmark Experiments

This section considers the results of the benchmark experiments for the classification as well as object detection models. As the results of this different kind of models cannot be compared directly, the results are presented in different subsections.

However, what remains similar for all tested architectures are the distinct configurations of benchmark experiments performed. Table 5.2 shows the legend of abbreviations used for further reference of the experiments.

| Key | Name | Description |
|---|---|---|
| ResNet | Residual Network | Residual Network 50 with default parameters. |
| EffNet | Efficient Network V2 | Efficient Network V2 S with default parameters. |
| EffFor | Efficient Former | Efficient Former L1 with default parameters. |
| ViT | Vision Transformer | Vision Transformer base p32 with default parameters. |
| DINO | DINO | DINO 5s Swin with default parameters. |
| YOLO | YOLOv8 | YOLOv8 S with default parameters. |
| MM | Min Max Scale | Min-Max scaling technique performed offline. |
| ZS | Z-Scale | Z-Scale technique performed offline. |
| PT | Pre-trained | Using pre-trained weights for the applied model. |
| NPT | Not Pre-trained | Trained the model from scratch without pre-trained weights. |
| US | Undersampling Filter | Applied undersampling to balance the classes of the dataset. |

Table 5.2: Abbreviations Legend Benchmark Experiments

### 5.2.1 Classification Models

The results of the classification model benchmark experiments show a relatively clear tendency of suitable approaches. All models were trained over $300$ epochs and a batch size of $32$ images. Table 5.3 presents the three out of eight best working configurations of each examined model, whereas table A.2 of the appendix lists the result summary of all $32$ performed experiments. To account for imbalanced classes in the RGZ OD dataset, the performance metrics are macro-weighted.

| **Model** | **Configuration** | **Acc. Top-1** | **Acc. Top-2** | **F1 Score** | **Precision** | **Recall** |
|---|---|---|---|---|---|---|
| ResNet | ZS, NPT | **83.38** | **94.53** | **79.60** | 79.67 | **79.71** |
| ResNet | ZS, NPT, US | 81.42 | 93.90 | 79.55 | **80.06** | 79.37 |
| ResNet | ZS, PT, US | 81.42 | 94.47 | 79.21 | 79.36 | 79.26 |
| EffNet | ZS, NPT | 79.23 | 92.30 | 74.00 | 74.41 | 73.72 |
| EffNet | ZS, PT | 78.42 | 92.71 | 72.29 | 72.90 | 71.93 |
| EffNet | ZS, PT, US | 76.74 | 90.21 | 74.77 | 76.10 | 74.15 |
| EffFor | ZS, NPT | 76.60 | 89.46 | 69.64 | 70.10 | 69.38 |
| EffFor | ZS, PT | 75.48 | 89.56 | 68.69 | 69.24 | 68.45 |
| EffFor | ZS, NPT, US | 73.05 | 90.64 | 71.02 | 71.62 | 70.62 |
| ViT | ZS, PT | 59.473 | 77.91 | 42.33 | 55.34 | 36.41 |
| ViT | ZS, NPT | 58.46 | 77.41 | 37.19 | 60.42 | 32.63 |
| ViT | MM, PT | 52.99 | 70.72 | 19.12 | 36.71 | 18.50 |

Table 5.3: Summary Results Benchmark Experiments | Classification Models

First, the results are consistently ordered by the differently investigated models. Surprisingly, ResNet50 surpasses the performance of EfficientNet S notably, with a maximum top-1 accuracy performance of $83.38\%$. In addition, the intuition that on the limited RGZ OD dataset, transformer based architectures cannot use their large number of parameters as an advantage has proven to be true.

Furthermore, for all applied models, z-scaling showed to be more suitable for radio source images in comparison to min-max-scaling. When training with a sufficient number of epochs, the experiments indicate that re-training the models is more beneficial than using pre-trained weights. That is reasonable, as the radio source images are significantly different from the samples of the ImageNet[141] dataset.

All models could not benefit from applying an undersampling filter on the imbalanced dataset. Even though the undersampling filter appears in the summary of the three best results, it can be assumed that z-scaling and the use of pre-trained weights have a stronger effect than balancing the classes.

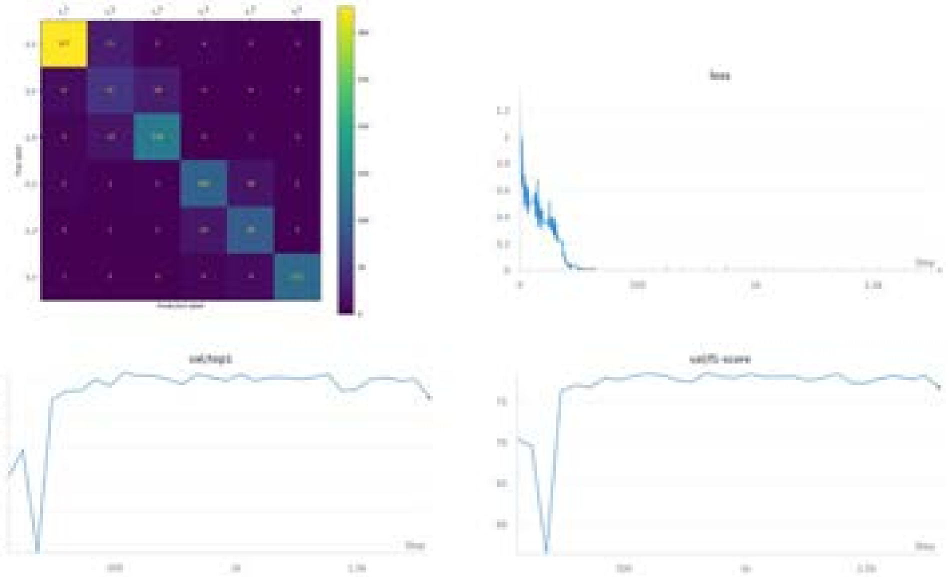


Figure 5.1: ResNet50 | Z-Scale | Not Pre-Trained | Performance Metric Summary

When considering the confusion matrix in figure 5.1, evaluated on the test set by using a ResNet on the z-scaled dataset without pre-trained weights, it can be observed that most confusion exists between class 1_2 and 1_3 as well as 2_2 and 2_3. This result coheres with the discriminative capability of humans on such a classification task and verifies the valid learning process of the model.

Finally, if the learning curves of the model are considered, we can also agree that the model has been trained for a sufficient number of epochs and that the problem of overfitting is only marginal due to the stagnating accuracy. The plots of all runs mentioned in table 5.3 can be viewed in appendix A.3.1.

### 5.2.2 Object Detection Models

Similar to the classification model benchmark experiments, the object detection models were trained on the RGZ OD dataset, with one difference in the preprocessing. In order to fulfil the requirements of standard classification models, samples with multiple classes were filtered out. However, these samples are retained in the dataset for the benchmark experiments on object detection. The performance results of the object detection benchmark experiments are listed in table 5.4.

The results show a different characteristic to the classification benchmark experiments. Firstly, in terms of mean average precision, the YOLO[11] based models could not outperform the transformer based models such as DINO[12]. However, YOLOv8 achieved the best performance on the Intersection over Union (IOU) metric. In other words, this means that the performance of the DINO model could surpass that of its CNN-based counterparts for the combination of object detection and the classification of the radio sources. But if only the detection performance is observed, the YOLOv8 model could obtain a higher accuracy. Although, just to consider the IOU for object detection performance could be misleading for the RGZ OD dataset. This because the bounding boxes were generated automatically and always with a quadratic shape. Lower IOU values can therefore also result from a deviation between inaccurate ground truth (GT) labelling and predicted boxes that are well-matched to the actual object.

Secondly, the architectures were able to benefit from pre-trained weights, which was not the case for the classification models. This could be explained by the higher complexity of the object detection models and the larger possible generalisation of a detection task per se. In fact, the use of pre-trained weights was the most influential factor leading to a higher mAP. Surprisingly, and in contrast to the classification results, training the DINO models on images with min-max scaling resulted in a higher performance than when using z-scaling. Lastly, DINO could not benefit from balancing the dataset with undersampling, which differs from the results of YOLOv8.

| Model | Configuration | mAP | IOU | Accuracy | mAP 50 | mAP 75 |
|---|---|---|---|---|---|---|
| DINO | MM, PT | **70.92** | 66.19 | **80.28** | **83.90** | **79.90** |
| DINO | ZS, PT | 68.88 | 65.73 | 79.11 | 80.80 | 77.40 |
| DINO | MM, PT, US | 67.80 | 64.83 | 76.60 | 80.90 | 77.00 |
| DINO | ZS, PT, US | 66.02 | 64.76 | 74.47 | 78.30 | 74.90 |
| DINO | MM, NPT | 64.71 | 67.32 | 78.70 | 79.20 | 74.70 |
| DINO | ZS, NPT | 64.20 | 68.42 | 78.86 | 78.10 | 73.90 |
| DINO | ZS, NPT, US | 63.05 | 70.88 | 73.01 | 76.80 | 72.70 |
| DINO | MM, NPT, US | 61.18 | 67.52 | 72.21 | 75.10 | 70.10 |
| YOLOv8 | ZS, PT, US | 55.75 | **88.08** | 71.68 | 69.80 | 66.00 |
| YOLOv8 | ZS, PT | 52.72 | 89.08 | 70.09 | 66.10 | 62.00 |
| YOLOv8 | MM, PT, US | 49.96 | 85.66 | 67.02 | 63.90 | 58.50 |
| YOLOv8 | MM, PT | 47.41 | 85.97 | 69.09 | 62.40 | 55.50 |
| YOLOv8 | ZS, NPT | 43.27 | 79.60 | 68.50 | 57.30 | 50.00 |
| YOLOv8 | MM, NPT | 0.00 | 48.18 | 32.75 | 0.00 | 0.00 |
| YOLOv8 | MM, NPT, US | 0.00 | 40.35 | 19.55 | 0.00 | 0.00 |
| YOLOv8 | ZS, NPT, US | 0.00 | 39.12 | 19.55 | 0.00 | 0.00 |

Table 5.4: Summary Results Benchmark Experiments | Object Detection Models

The confusion matrix of the best DINO configuration visualised in figure 5.2 indicates a similar result than for the classification models, however, with a higher confusion between class 1_2 and 1_3 as well as between 2_2 and 2_3. Moreover, the matrix shows that the model often missed some objects, by classifying the area as background when there was actually an emission and similarly for empty regions it predicted 1_1 sources. This effect describes the model's difficulty in dealing with noise and makes appropriate scaling of images essential when working with radio sources. Looking at the curves of the loss and performance metrics, one can conclude that the models have been trained for a sufficient number of epochs for this evaluation, but a longer training may lead to a better result. The screening of the training set images shows that the model was able to detect the objects properly, and that the lower performance results from the misclassified classes. Additional, representative examples are included in appendix A.3.1. When considering the results obtained with YOLOv8 model, there is a tendency for the model to additionally predict sub-emission with a smaller number of components and peaks for larger emissions. An example is visualised in figure A.19.

After discussing the results in the project team, it was concluded that only the best performing classification model out of the architecture benchmark should be further evaluated in the subsequent experiments of this work. This is primarily due to the significantly lower classification performance of the detection models. Moreover, it is not expected that the performance of the detection architectures can be raised to the level of the classification models without a considerable amount of tuning and architectural investigations. For this project, classification capabilities are more highly valued, as the location of the radio sources can either be assessed by using the meta-information of the dataset or with a lightweight peak or blob finder. However, this decision should not connote that neural based object detection models are not well suited in the domain of radio source detection. DINO in particular achieved promising results, and it is recommended to perform extended experiments in a subsequent study.

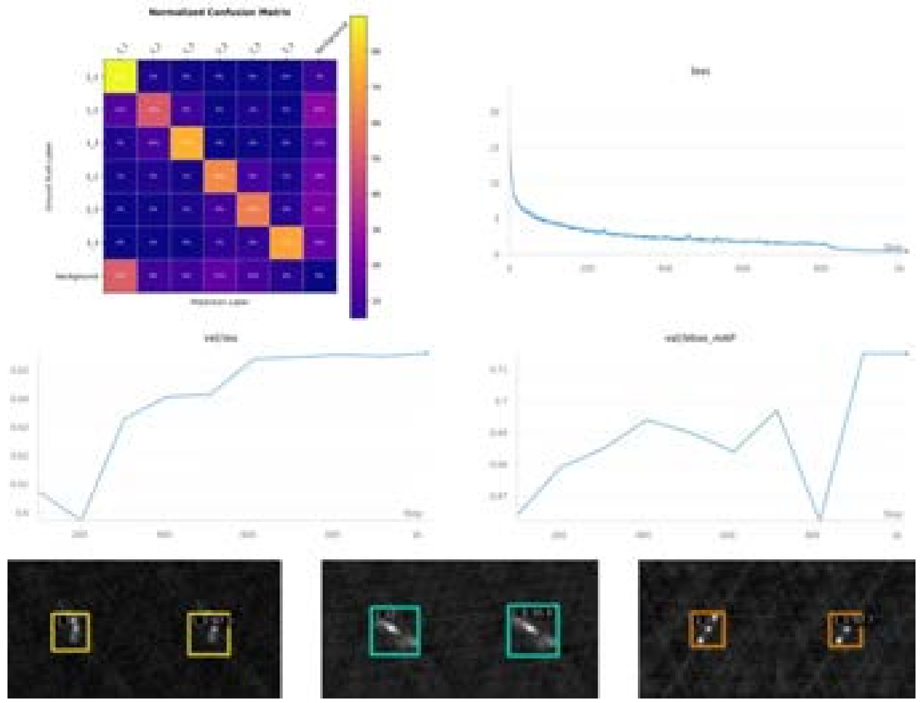


Figure 5.2: DINO | Min-Max Scale | Pre-Trained | Performance Metric Summary

## 5.3 Ensemble Experiments

For the ensemble experiment, the best performing classification model evaluated during the benchmark experiment in section 5.2 was further analysed. In more detail, $30$ ResNet50 models were trained on z-scaled radio sources and without pre-trained weights. To measure the uncertainty in terms of standard deviation $\sigma$, the models were initialised with random weights. All other settings and hyperparameters were retained, similar to the ones of the benchmark experiments.

Table 5.5 shows the evaluated resulting standard deviation per class as well as on average. Only class 3_3 features a significantly smaller standard deviation. For all other classes, the result is approximately similar, which could be an indication for a regression to the mean problem by considering $987$ samples in the test set. Therefore, items with a particularly high variance are further analysed.

| **Classes** | **1_1** | **1_2** | **1_3** | **2_2** | **2_3** | **3_3** |
|---|---|---|---|---|---|---|
| Uncertainty | 0.0407 | 0.0622 | 0.0521 | 0.0508 | 0.0514 | 0.0225 |
| **Average Uncertainty** | **0.0466** | | | | | |

Table 5.5: Uncertainty ResNet50 | Z-Scale | Not Pre-trained | Per Class and on Average

As a second analysis, the number of standard deviations of the difference between the mean value of the predictions and the ground truth was calculated. Hereby, the mean of the predictions refers to the mean over the predicted scores (SoftMax values) by the $30$ trained models. Table 5.6 lists the distribution of the number of standard deviations, and the diagrams in figure 5.3 show the distribution in relation to the distance between the mean prediction and the ground truth. The histograms show that $83.08\%$ of the samples are within one standard deviation. However, there are $45$ samples that make up $4.6\%$ of the test set that are above the six-sigma interval. An excerpt of those samples is displayed in figure 5.4, whereas the left side shows the utilised z-scaled representation and the left side the unscaled image. Moreover, additional examples are visualised in appendix A.3.2.

| Sigma Interval | # of Samples |
|---|---|
| [0, 1) | 820 |
| [1, 3) | 87 |
| [3, 6) | 35 |
| [6, ∞) | 45 |
| **Total** | **987** |

Table 5.6: Uncertainty Distribution by the Number of Standard Deviations $\sigma$

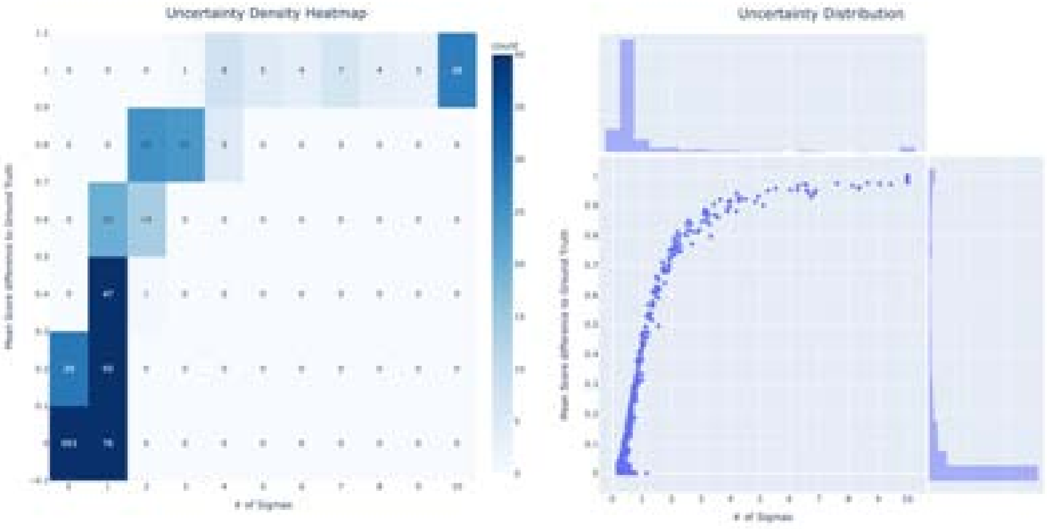


Figure 5.3: Uncertainty Distribution by the Number of Standard Deviations $\sigma$

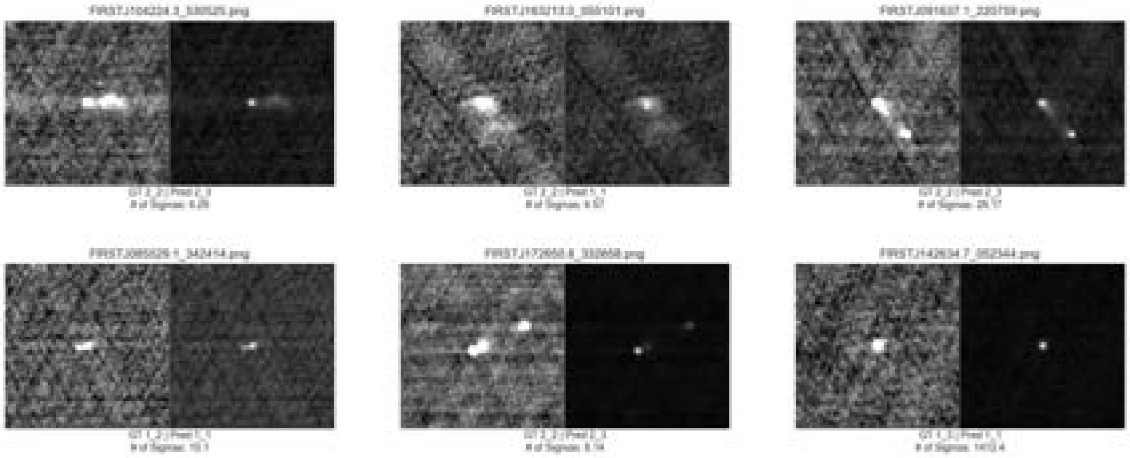
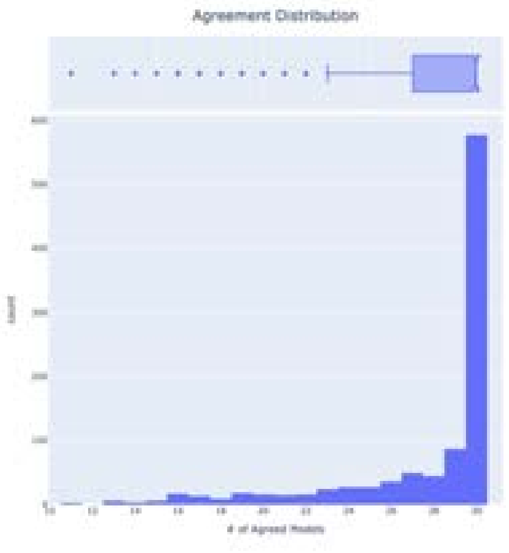


Figure 5.4: Ensemble ResNet50 - High Variance Examples

Over all samples, on average, $27.66$ models out of $30$ agree on the predicted class with a standard deviation of four models. The distribution of agreement levels over the examples of the test set is pictured in figure 5.5. The diagram shows that most models coincide in their predictions. If now the results of figure 5.4 are considered, which shows samples with a high number of standard deviations from the ground truth mean, two phenomena can be observed. The first cluster of samples are true misclassifications of the model. However, if the z-scaled representation on the left is compared with the unscaled representation on the right, it can be reasoned, that some confusion of the model is attributable to different scaling methods. Therefore, z-scaling seems to be advantageous for most of the samples, but not for all. The second cluster of misclassified images could be mislabelled examples or at least difficult cases to which even humans cannot precisely assign a unique class. These two clusters of errors can especially be observed on the samples on which all 30 models agreed on the same different class from the ground truth. An excerpt of those examples is shown in appendix A.3.2. As the RGZ OD dataset is a citizen science project, it is likely that not all designations are correct. However, comments from experts on the RGZ dataset also indicate that, especially between the classes $1\_2$ and $1\_3$ as well as $2\_2$ and $2\_3$, even people disagree on the actual classes, which emphasises the difficulty of radio source classification tasks. In order to determine the extent of possible misclassifications, table 5.5b shows the agreement and success rates for the samples in the test series.



(a) Agreement Distribution

| **Name** | **Value** |
|---|---|
| Full Success Rate | 56.64% |
| Full Success | 559 Samples |
| Partial Success Rate | 40.43% |
| Partial Success | 399 Samples |
| Full Error Rate | 2.94% |
| Full Error | 29 Samples |
| Agreed Full Error Rate | 1.93 |
| Agreed Full Error | 19 |
| Majority Success Rate | 85.21% |
| Majority Success | 841 Samples |
| **Total** | **987 Samples** |

(b) Quantitative Results of Agreement Analysis

Figure 5.5: Agreement Analysis

It can be seen that with $30$ randomly initialised models, all models predicted an incorrect class in only $3\%$ of the examples. Further, with only about $2\%$ of the data, all models predicted the same class, which differs from the ground truth. However, also in only $56\%$ of the test set, all models agreed on the similar class as the ground truth. Lastly, if the scores are evaluated in a classical majority vote, the ensemble could further rise the performance to $85\%$ top-1 accuracy. Indeed, $30$ models would not be applicable in a practical use case. Therefore, each ensemble size was analysed by performing the top-1 accuracy assessment in $100$ runs for each size with randomly selected models from the $30$ trained ensemble experiment models. The result, represented by the mean value as well as the lower and upper bounds, can be seen in Figure 5.6 as well as in table 5.7.

Ultimately, for further verification purposes, the ensemble experiments were additionally performed on $30$ ResNet model of depth $50$ with the fine-tuned parameters of the tuning experiments. The results of this analysis can be viewed in appendix A.3.3. The analysis shows that with an ensemble of three individually trained models a possible top-1 accuracy of $90.68\%$ and a top-2 accuracy of $98.58\%$ can be achieved.

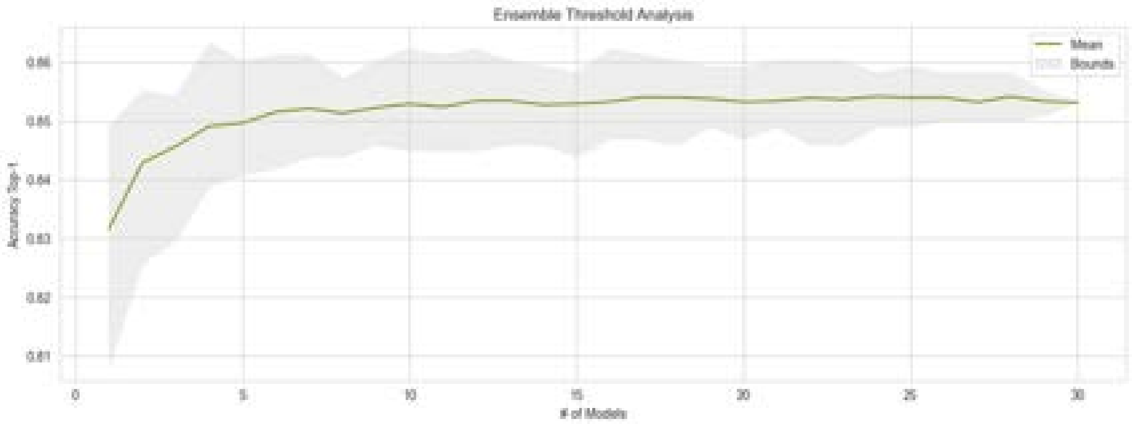


Figure 5.6: Ensemble Size Analysis

| Key | Value |
|---|---|
| Maximum mean accuracy | 85.41% |
| Maximum mean accuracy at # of models | 26 |
| Maximum accuracy | 86.42% |
| Maximum accuracy at # of models | 3 |
| Lowest mean accuracy | 83.22% |
| Lowest mean accuracy at # of models | 1 |
| Lowest accuracy | 81.46% |
| Lowest accuracy at # of models | 1 |

Table 5.7: Ensemble Size Analysis Top-1 Accuracy

## 5.4 Tuning Experiments

The main reason for the tuning experiments was to further increase the classification performance of the most promising architecture evaluated during the benchmark and ensemble experiments. Specifically, the tuning experiments include scaling, augmentation and hyperparameter tuning of the evaluated ResNet[14] architecture.

The analysis of the pre-processing, benchmark and ensemble experiments revealed that scaling has a significant impact on the performance of the classifier. Examples were identified where the model classified noise as an additional component or peak due to the increase in illumination from z-scaling, or sidelobes of weak intensity could not be distinguished from noise by the classifier without scaling. To remedy this shortcoming, two simple approaches were tested. Firstly, tuning the contrast parameter of the z-scale implementation and secondly, since the original sample is a single-channel greyscale image, the use of a z-scaled and a min-max scaled representation in different channels. The resulting pre-processed sample then contained three channels, two z-scaled and one min-max scaled in the centre. In this paper, this method is referred to as ZMZStack, where a comparison with z-scaled and min-max scaled images is visualised in figure 5.7. Subsequently, the highest performance increase was achieved by using ZMZStack, which increased the top-1 accuracy from $83.38\%$ to $86.12\%$ without further tuning.

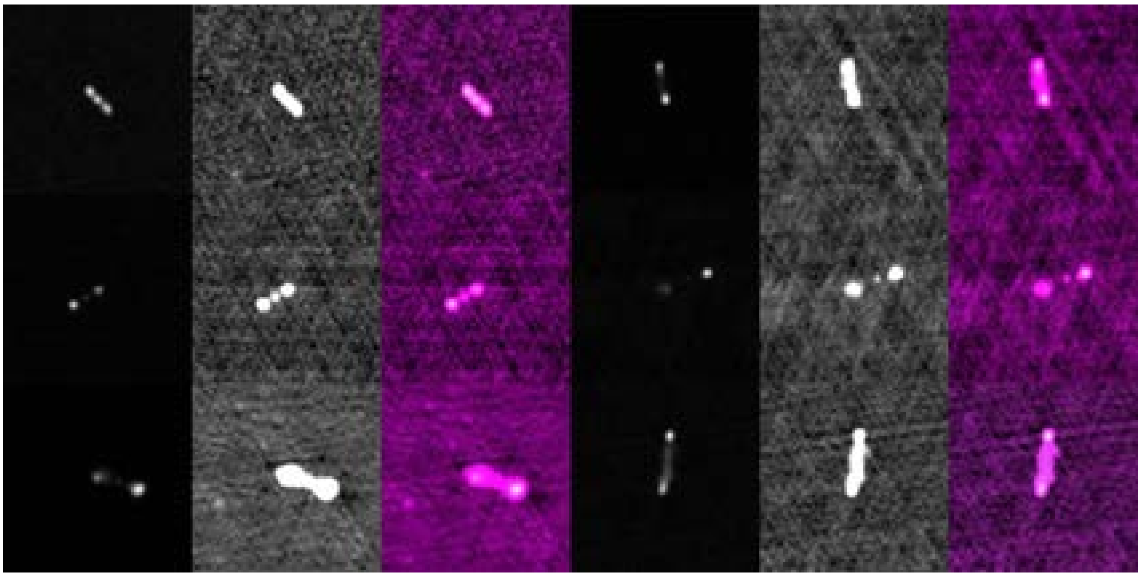


Figure 5.7: Visual Comparison of Min-Max Scale, Z-Scale and ZMZStack | LTR

Basic augmentation techniques were evaluated as the second mean of fine-tuning. To account for possible distortions of the real distribution, rotation, scaling, translation and shearing were applied by a random factor $p$ between $0$ and $0.5$. Each method was analysed both individually and in combination. While scaling and translation were able to further increase performance, rotation had a neutral effect and shearing even a negative one. Ultimately, the highest improvement was achieved by a combination of scaling and translation with $p = 0.25$, which further increased the top-1 accuracy to $88.35\%$.

Fine-tuning the optimiser by strategy (SGD[155], AdamW[156], RMSProp[157]), learning rate and learning rate scheduler could supplementary increment the performance level to $89.32\%$. In particular, training the model for a longer period of time before the learning rate decays had a positive effect.

Finally, two promising methods investigated during the preprocessing experiments were evaluated. While sigma median clipping could sightly raise the performance from $89.36\%$ to $89.67\%$ top-1 accuracy, led dynamic weights conversion to a lower result. However, in terms of top-2 accuracy, the performance declined to $97.47\%$ when using sigma median clipping. Sigma median clipping was thereby only applied to the z-scaled channels of ZMZStack. Most likely, the model lacks of the different pperspectivesön the radio source when using dynamic weights conversion, since it does not combine different scaling methods, but weights a particular technique across the three channels.

To summarise, fine-tuning the ResNet architecture improved the top-1 accuracy from $83.38\%$ to $89.67\%$ and the top-2 accuracy from $94.53\%$ to $97.47\%$. A résumé of the incrementation steps is shown in table 5.8.

| Model | Configuration | Acc. Top-1 | Acc. Top-2 | F1 Score | Precision | Recall |
|---|---|---|---|---|---|---|
| ResNet | Sigma Median Clipping | **89.67** | 97.47 | **86.89** | **87.52** | **85.82** |
| ResNet | Optimiser | 89.36 | **97.57** | 86.24 | 87.40 | 85.44 |
| ResNet | Augmentation | 88.35 | 96.86 | 86.01 | 86.58 | 85.67 |
| ResNet | ZMZStack | 86.12 | 94.43 | 82.37 | 82.74 | 82.06 |
| ResNet | Benchmark | 83.38 | 94.53 | 79.60 | 79.67 | 79.7 |

Table 5.8: Results Tuning Experiments | ResNet50

## 5.5 GLEAM Experiments

In the GLEAM experiments, the finally fine-tuned models were verified on samples from the GLEAM survey[68]. As the official GLEAM API was not available at the time of the project, the verification could only be conducted on samples with the lowest wide-field resolution and need therefore be viewed with caveat.

For the experiments, six radio galaxy samples were manually cut out of a larger image containing hundreds of sources. Such a source image is shown in the appendix A.32. The resulting crops have a resolution between $34 \times 34$ and $50 \times 50$ pixels, and are therefore even difficult for humans to classify. Consequently, it was decided for this work to not assess the classes of the samples conclusively, but to comment the results provided by an ensemble of $30$ randomly initialised ResNet50 models.

The samples were evaluated using the mean value of the $30$ individual model predictions for each class, as well as the standard deviation. The results for all six samples investigated are displayed in figure 5.8 and table 5.9.

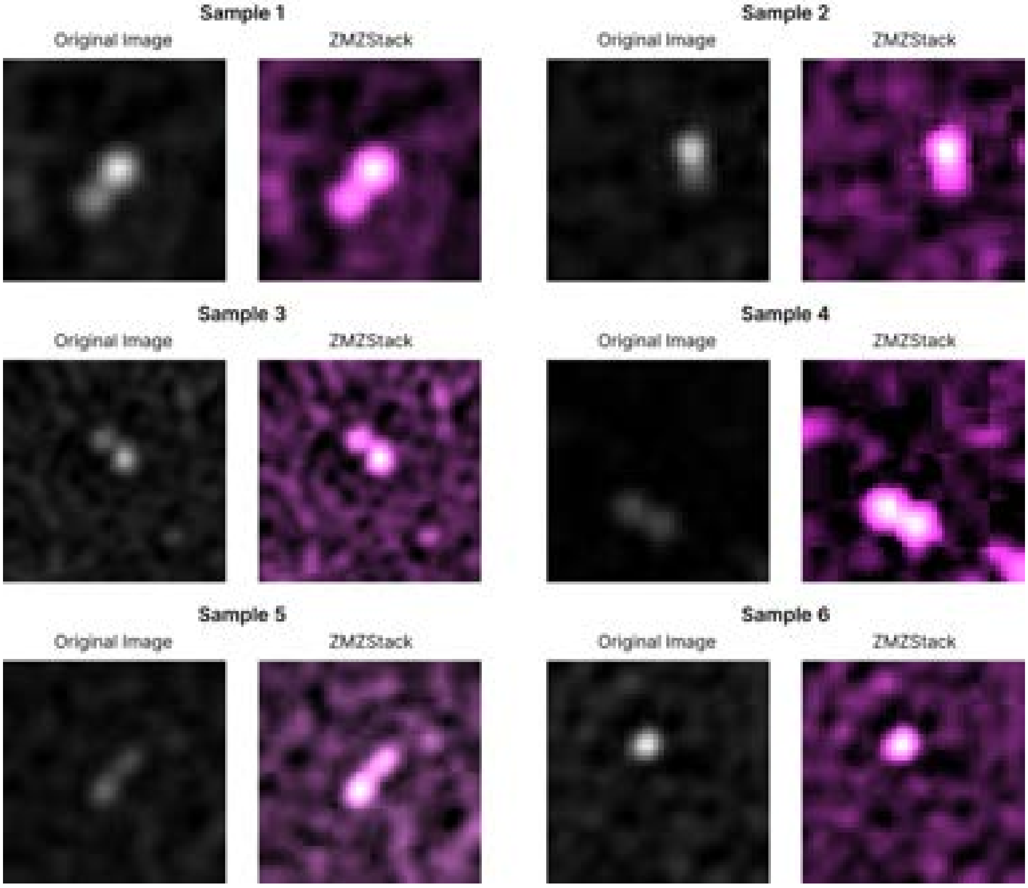


Figure 5.8: Evaluated GLEAM[68] samples

| **Metric / Classes** | **1_1** | **1_2** | **1_3** | **2_2** | **2_3** | **3_3** |
|---|---|---|---|---|---|---|
| | | | **Sample 1** | | | |
| Mean Score | **0.6186** | 0.0373 | 0.0339 | 0.1203 | 0.1319 | 0.0582 |
| Standard Deviation | 0.2775 | 0.0284 | 0.0435 | 0.1232 | 0.1844 | 0.0879 |
| | | | **Sample 2** | | | |
| Mean Score | 0.3946 | 0.0037 | 0.0023 | **0.4268** | 0.0596 | 0.1129 |
| Standard Deviation | 0.2978 | 0.0056 | 0.0056 | 0.2628 | 0.0519 | 0.1288 |
| | | | **Sample 3** | | | |
| Mean Score | 0.1634 | **0.3290** | 0.3097 | 0.0658 | 0.1299 | 0.0022 |
| Standard Deviation | 0.1736 | 0.1708 | 0.1921 | 0.1309 | 0.1811 | 0.0038 |
| | | | **Sample 4** | | | |
| Mean Score | 0.0530 | 0.0430 | 0.0005 | 0.1852 | **0.5156** | 0.2027 |
| Standard Deviation | 0.1947 | 0.1789 | 0.0017 | 0.1844 | 0.3503 | 0.2980 |
| | | | **Sample 5** | | | |
| Mean Score | 0.2436 | 0.1765 | **0.3651** | 0.0474 | 0.1603 | 0.0071 |
| Standard Deviation | 0.2036 | 0.0698 | 0.2082 | 0.0523 | 0.1929 | 0.0155 |
| | | | **Sample 6** | | | |
| Mean Score | **0.8144** | 0.0053 | 0.0022 | 0.1341 | 0.0108 | 0.0332 |
| Standard Deviation | 0.2180 | 0.0056 | 0.0025 | 0.1473 | 0.0202 | 0.0837 |

Table 5.9: Prediction Results GLEAM[68] Experiments

When considering the individual samples in general, it can be seen that the ensemble shows a substantial level of uncertainty for almost all images. For instance, sample $1$. The average prediction value peaks at class $1$, which represents a radio source with one component and a single peak. However, the corresponding standard deviation reveals that class $1\_1$ overlaps with the scores of class $2\_2$ and $2\_3$ within the two-sigma confidence interval, which indicates a probability of more than $5\%$ for misclassification. This signifies, that despite the high score for class $1\_1$, the models of the ensemble are uncertain in their predictions.

The same conclusion can be drawn for samples $2$ to $4$. Only for sample $6$ shows the model ensemble a confidence interval above the two-sigma level, which indicates a confidence level of over $5\%$ for the correctness of class $1\_1$. This result is reasonable, since sample $6$ certainly contains the least complex radio source component and can also most likely be assigned to class $1\_1$ by humans.

The evaluation of the selected GLEAM[68] examples shows that the fine-tuned ResNet50 models are likely to work satisfactorily on different datasets. However, the examination of the samples indicates that the ensemble is particularly uncertain for more *complex* and less characteristic radio sources. In this context, the term *complex* refers to emissions apart from simple $1\_1$ sources, which, admittedly, account for the vast majority of emissions recognisable in the source image. But, these more complex samples are also difficult to categorise conclusively for humans. Therefore, on the basis of the data at hand, it cannot be decisively verified, if the promising results of the trained and fine-tuned ResNet50 models can be transferred to other datasets or to a measurement of the future operable SKA observatory.

## 5.6 Conclusion

Various types of experiments were carried out as part of the project. The preprocessing experiments showed that the scaling of the original RGZ OD FITS images can have a significant impact on the detection and classification performance. Of the manifold methods tested, z-scaling produced the best results when using CNN-based backbone models such as the evaluated classifiers and the YOLO-based object detection architectures. This is in contrast to the results of the trained DINO detection model, which is based on a Swin backbone and for which a basic min-max scaling was sufficient.

The subsequent error analysis showed that the model probably misclassified some radio sources due to a too strong z-scaling or a too weak min-max scaling. Since the images must have three input channels for the algorithm framework applied, a stack of a z-scaled, a min-max scaled and a z-scaled greyscale representation was used instead of a simple replication of a uniquely scaled channel. By using this method called ZMZStack, the classification performance could be notably improved.

The benchmark showed that relatively small networks are sufficient for the classification task of the RGZ OD dataset. Of the evaluated models, the best classification performance was achieved with a top-1 accuracy of $83.38\%$ and a top-2 accuracy of $94.53\%$ with a ResNet of depth $50$. Classification models based on transformers were unable to utilise their full potential, possibly due to the small size of the dataset. For the object detection models, however, a transformer-based DINO model achieved promising results with a mAP of $70.92\%$ and an accuracy of $80.28\%$, as well as an IOU of $66.19\%$. Since classification performance was the key metric in this work, all subsequent experiments were performed with the ResNet architecture.

On the one hand, the ensemble experiments have shown that the uncertainty of the 30 trained models is low for about $83\%$ of the samples, which is indicated by a standard deviation between the predictions below $1\sigma$. On the other hand, they showed that the performance can be further improved by using an ensemble of two to ten models, with a suggested number of five models. The analysis also made it possible to identify ambiguous samples for the model, as well as incorrectly labelled samples. For the ambiguous radio sources, more samples could be collected from the entire FIRST catalogue[61], and for the limited number of mislabelled samples of up to $1.5\%$, it would be worth having the samples corrected by a subject matter expert.

In the final fine-tuning of the evaluated ResNet architecture through preprocessing methods, augmentation techniques and optimiser adjustments, it was possible to increase the classification performance from a top-1 accuracy of $83.38\%$ and a top-2 accuracy of $94.53\%$ to a top-1 accuracy of $89.67\%$ and a top-2 accuracy of $97.47\%$. Additionally, it was evaluated that deeper models, for instance a ResNet101 or a ResNet152, cannot efficiently raise the model performance in comparison to the increased resources they require.

The GLEAM ensemble experiments revealed a possible misclassification rate of more than $5\%$ for complex radio source samples with very low resolution. However, for less complex examples, the models can predict with high confidence, the most likely class. Consequently, the analysis raised confidence that the models will perform superiorly on higher resolution examples and that the results from the RGZ OD dataset can be transferred either to higher resolution GLEAM radio sources or to initial surveys of the SKA observatory.

# Chapter 6

# Discussion and Future Work

The chapter is divided into two sections. First, the results are discussed and extensions to the existing solutions are proposed while the reflection. Secondly, possible future work should be prioritised.

## 6.1 Result Discussion

The novelty of this work can be summarised in the combination of the different experiments performed. It combined the analysis of SOTA object detection and classification models for the overall goal of identifying and classifying radio sources. Existing preprocessing techniques have been extensively verified, allowing suggestions for their use with common computer vision libraries. In addition, the project not only trained and fine-tuned promising architectures, but also evaluated the uncertainty of the trained model and considered possible errors in the dataset through a detailed error analysis. Finally, the trained model was used to classify detected objects in the GLEAM[21] dataset to examine the model's transferability to new datasets. This chapter discusses and summarises the findings and experience gained.

One of the key findings from the preprocessing experiments was that scaling is one of the most influential factors for the classification performance of the model. It could be verified that z-scaling[132] has the largest positive effect among the methods and combinations analysed, as it makes hidden radio source components visible. However, when considering the results of the error analysis, examples can be identified, where z-scaling connects blobs that were originally separated without scaling. This effect was one reason for misclassifications made by the model. The combination of these two findings led to further investigations by stacking two z-scaled channels and one min-max scaled channel into a three-channel image, which we called ZMZStack. Various experiments have shown that this method is favourable for the classification performance of the model. The intuition could be that by applying both scaling techniques, the model obtains information about hidden components as well as about the separation between emissions. As a conclusion, the right preprocessing can have a significant influence on the model performance. Moreover, it is expected that additional fine-tuning of the z-scale parameters can further improve the results. However, due to the time constraints of this project, these efforts are led to follow-up studies.

When viewing at the results of the benchmark experiments, it is important to distinguish between the object detection and classification experiments. First, the object detection models could not benefit from z-scaling and pre-trained weights lead to a better result than by retraining the entire model. However, this result needs further differentiation. By considering the learning curve in appendix A.3.1, it can be seen that DINO models trained with pre-trained weights first experience a slight drop in mAP performance followed by a sharp increase. In comparison, the DINO models trained from scratch show a more gradual increase in mAP, whereas the YOLOv8 models could not be trained in a stable process. Even an increase in model complexity did not lead to a significant increase in performance, and should be viewed sceptically whether the longer training period and higher resource consumption are justifiable. This is reasonable, as the complexity of radio sources and the number of classes are not enormous for the RGZ OD dataset. Unsurprisingly, the detection performance of both evaluated models is higher than the classification performance. Despite the high level of noise in some images, connected components still contrast the background, which is also explained by the fact that basic blob detection algorithms were applied successfully. Considering the resulting accuracy of roughly $80\%$ without further fine-tuning is a positive result, however, this performance is also attributable to an over-representation of class 1_1.

Interestingly, the comparison of the results of the evaluated detection and classification models shows a profound difference. In pure classification, CNN-based architectures outperformed transformer-based models, while the opposite is true for object detection. One can argue, that in case of classification, the dataset size of approximately $10'000$ images, of which $4000$ samples are from class 1_1, is too small to train a powerful transformer. Especially the slow improving and wiggly loss curves of the ViT support this assumption. But even with significantly smaller transformers like the EfficientFormer, without specific fine-tuning and architecture adjustments, the model could not surpass a basic ResNet. One reason for this could be that in order to fulfil the sequential input for a transformer, the image is cut into patches, which can destroy the semantics of a small or stretched radio source, even with positional encoding.

Ultimately, the benchmark experiment showed that modern neural architectures can achieve a remarkable result on an appropriately scaled and limited radio source dataset without extensive fine-tuning. This means that it is not always necessary to invent a customised model for radio astronomy and often standard, modern architectures are sufficient. In addition, it may be worth investing more time in preparing the dataset with favourable preprocessing than in extensive model design, even if feature engineering has become less important with the use of deep learning.

With the ensemble experiments of $30$ models, it could be verified that for the majority of the samples ($83\%$), the model uncertainty is relatively low ($< 1\sigma$). Further, two types of issues could be identified by analysing the samples on which all models failed by predicting the same wrong class. First, actual misclassification of the model and second, might wrongly labelled data by the RGZ citizens science project. By analysing the random test set, those errors could account for up to $1.5$ percent of the dataset. But to be fair, these are also those samples, on which subject-matter experts disagree as well. However, for those samples, when using an ensemble of three models, the top-2 performance could be raised to $98.58\%$. When taking some mislabelled samples into account, the models can predict with high confidence between which two classes an expert needs to decide on as well as on which samples the model is less certain. This approach could be used for future labelling projects or for the automatic analysis of radio source catalogues.

Essentially, the fine-tuning showed two outcomes. First, that the hyperparameters chosen for a ResNet trained on ImageNet were not far from those fine-tuned for a ResNet trained on the RGZ OD dataset. Secondly, preprocessing and augmentation can further accelerate performance and robustness. Interestingly, the use of augmented samples by rotation had no significant positive influence on the result. This is possibly due to the fact that the radio sources in the training dataset were already arranged in several directions. On the other hand, scaling augmentation has increased performance. This comes close to analyse the radio sources at different frequencies and was expected to confuse the model, as small objects become similar to noise and noise itself could grow to 1_1 sources. Finally, it is plausible that the translation augmentations have a positive effect. This is because most of the RGZ OD samples are located at the centre of the image, and translation should make the model more robust to shifted radio sources.

For five out of six manually cropped samples of the GLEAM source images, an ensemble of $30$ ResNet50 models showed a probability of more than $5\%$ for a misclassification of the predicted radio source class. However, the reason for the high uncertainty is most likely attributable to the significantly lower resolution of the samples compared to those used during training. One possible approach to reduce uncertainty could be to further augment the RGZ OD dataset with more significantly lower scaled samples. But even if the certainty level could be notably increased, manual validation by the human eye is still difficult with images of such low resolution, and the true classes can only be guessed. In addition, it cannot be ruled out that individual side lobes or components of emissions may be lost due to the low resolution of the source images. It is also possible that due to the proximity of the emissions, two actually individual sources in the image are assessed as class $2\_2$. To mitigate this risk, either the resolution has to be increased or the length of the light rays needs to be taken into account. Ultimately, it would be worth considering repeating the GLEAM experiments with higher-resolution images as soon as the public GLEAM API is back online.

## 6.2 Future Work

This section prioritises and summarises possible future work that has emerged from the discussions in this thesis. The first proposed work package is to profounder analyse scaling. With ZMZStack, only two of three possible channels are occupied from a different scaling. There is a chance that an additional, cleverly scaled representation of the third channel could be advantageous for the model. This would finalise the preprocessing investigations and could form the basis for many future projects and tasks. To ensure an efficient evaluation of additional techniques, the existing pipelines for preprocessing, training and evaluation would have to be integrated into one modular pipeline. This additional effort was the main reason why this work package was transferred to future work due to the time constraints of this project.

As a second priority, more attention should be paid to object recognition models. This includes a profound fine-tuning of the DINO architecture in combination with an error analysis in order to better understand the reasons for incorrectly classified objects. The results of this project have provided evidence for the successful application of transformer-based detection models for radio sources, and it is likely that the performance level can be significantly increased with moderate fine-tuning. If the classification performance of the object detection model could not be raised to the level of the investigated classification architectures, another idea could be queuing a detection model sequentially with a classification model. This because the detection performance of the trained models was already satisfactory. The recognised objects would then be cropped and serve as input to the classifier. Furthermore, the additional classification of the detection model would provide a further level of confidence in the predicted class.

The third priority is to further automate the training pipeline in order to accelerate the evaluation of new models with the findings from this project. At this stage of the project, the pipeline combines three `Open-MMLab` frameworks with numerous architecture supported by each library. The problem, each model has slightly different configuration requirements. The next steps of automation would therefore consist of the following tasks. Firstly, the creation of an additional abstraction level of configurations to harmonise all available models by a few settings. Secondly, to speed up the training process, an extension of the pipeline should be implemented to support distributed training on multiple workers and GPUs. Thirdly, the execution of Slurm jobs should be further automated, such that a single script can be run on a local or hosted machine. Fourthly, the configurations should be made via a user interface (UI), as it is difficult to memorise all possible configurations without a guided process in a YAML file tree. In this way, an arbitrary formatted dataset can be verified and fine-tuned on all `Open-MMLab` supported SOTA classification and recognition models without effort and only depends on the available computational resources.

Because this final work package depends on third-party suppliers, it is listed at the end of this section. However, it should be performed with the highest priority as soon as the public GLEAM API is back online. As noted in the discussions, to be able to draw further conclusions about the transferability of the results obtained in this work and to possible future surveys of the SKA telescope, the trained models should be evaluated on higher resolution GLEAM images. Due to external factors, this validation was not possible within the period of this project but would finalise the investigations of this work.

# Chapter 7

# Directories

# Bibliography

[1] Tao An. Science opportunities and challenges associated with ska big data. *Science China Physics, Mechanics and Astronomy*, 62:989531, 3 2019. ISSN 1869-1927. doi: 10.1007/s11433-018-9360-x. URL `https://doi.org/10.1007/s11433-018-9360-x`.

[2] Robert Braun, Tyler L Bourke, James A Green, Evan Keane, and Jeff Wagg. Advancing astrophysics with the square kilometre array. volume 215, page 174, 2015.

[3] *Astronomy and Astrophysics in the New Millennium: Panel Reports*. National Academies Press, 2001.

[4] Peter E Dewdney, Peter J Hall, Richard T Schilizzi, and T Joseph L W Lazio. The square kilometre array. *Proceedings of the IEEE*, 97:1482–1496, 2009. doi: 10.1109/JPROC.2009.2021005.

[5] Hayato Shimabukuro, Kenji Hasegawa, Akira Kuchinomachi, Hidenobu Yajima, and Shintaro Yoshiura. Exploring the cosmic dawn and epoch of reionization with a 21cm line. *Publications of the Astronomical Society of Japan*, 75:S1–S32, 12 2022. ISSN 2053-051X. doi: 10.1093/pasj/psac042.

[6] James Stanley Hey and J S Hey. *The evolution of radio astronomy*, volume 1. Science History Publications New York, 1973.

[7] Paul J. Hancock, Cathryn M. Trott, and Natasha Hurley-Walker. Source finding in the era of the ska (precursors): Aegean 2.0. *Publications of the Astronomical Society of Australia*, 35:e011, 3 2018. ISSN 1323-3580. doi: 10.1017/pasa.2018.3. URL `https://www.cambridge.org/core/product/identifier/S1323358018000036/type/journal_article`.

[8] Simone Riggi, Cristobal Bordiu, Daniel Magro, Renato Sortino, Carmelo Pino, Eva Sciacca, Filomena Bufano, Thomas Cecconello, Giuseppe Vizzari, Fabio Vitello, and Giuseppe Tudisco. Radio source analysis services for the ska and precursors, 2023.

[9] Teun Koetsier. *Minkowski: The Universe Is a 4-Dimensional Manifold*, pages 303–318. Springer Nature Switzerland, 2024. ISBN 978-3-031-39872-8. doi: 10.1007/978-3-031-39872-8_18.

[10] Inigo V Slijepcevic, Anna M M Scaife, Mike Walmsley, Micah Bowles, O Ivy Wong, Stanislav S Shabala, and Sarah V White. Radio galaxy zoo: Towards building the first multi-purpose foundation model for radio astronomy with self-supervised learning, 2023.

[11] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics, 1 2023. URL `https://github.com/ultralytics/ultralytics`.

[12] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[15] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[16] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed, 2022.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 5 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

[18] Emma Torres Chickles. Applications of convolutional neural networks to problems in astronomy and planetary science, 2021.

[19] Andrew Davies, Stephen Serjeant, and Jane M Bromley. Using convolutional neural networks to identify gravitational lenses in astronomical images. *Monthly Notices of the Royal Astronomical Society*, 487:5263–5271, 12 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz1288. URL https://doi.org/10.1093/mnras/stz1288.

[20] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450:1441–1459, 12 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv632. URL https://doi.org/10.1093/mnras/stv632.

[21] N Hurley-Walker, J R Callingham, P J Hancock, T M O Franzen, L Hindson, A D Kapińska, J Morgan, A R Offringa, R B Wayth, C Wu, Q Zheng, T Murphy, M E Bell, K S Dwarakanath, B For, B M Gaensler, M Johnston-Hollitt, E Lenc, P Procopio, L Staveley-Smith, R Ekers, J D Bowman, F Briggs, R J Cappallo, A A Deshpande, L Greenhill, B J Hazelton, D L Kaplan, C J Lonsdale, S R McWhirter, D A Mitchell, M F Morales, E Morgan, D Oberoi, S M Ord, T Prabu, N Udaya Shankar, K S Srivani, R Subrahmanyan, S J Tingay, R L Webster, A Williams, and C L Williams. Galactic and extragalactic all-sky murchison widefield array (gleam) survey – i. a low-frequency extragalactic catalogue. *Monthly Notices of the Royal Astronomical Society*, 464: 1146–1167, 12 2016. ISSN 0035-8711. doi: 10.1093/mnras/stw2337. URL https://doi.org/10.1093/mnras/stw2337.

[22] S Riggi, D Magro, R Sortino, A De Marco, C Bordiu, T Cecconello, A M Hopkins, J Marvil, G Umana, E Sciacca, F Vitello, F Bufano, A Ingallinera, G Fiameni, C Spampinato, and K Zarb Adami. Astronomical source detection in radio continuum maps with deep neural networks, 2022.

[23] Rafaël I J Mostert, Kenneth J Duncan, Lara Alegre, Huub J A Röttgering, Wendy L Williams, Philip N Best, Martin J Hardcastle, and Raffaella Morganti. Radio source-component association for the lofar two-metre sky survey with region-based convolutional neural networks. *arXiv preprint arXiv:2209.14226*, 2022.

[24] Zhen Zhang, Bin Jiang, and Yanxia Zhang. Automatic detection and classification of radio galaxy images by deep learning. *Publications of the Astronomical Society of the Pacific*, 134:064503, 2022. ISSN 1538-3873. doi: 10.1088/1538-3873/ac67b1. URL https://dx.doi.org/10.1088/1538-3873/ac67b1.

[25] Xingzhu Wang, Jiyu Wei, Yang Liu, Jinhao Li, Zhen Zhang, Jianyu Chen, and Bin Jiang. Research on morphological detection of fr i and fr ii radio galaxies based on improved yolov5. *Universe*, 7, 2021. ISSN 2218-1997. doi: 10.3390/universe7070211. URL https://www.mdpi.com/2218-1997/7/7/211.

[26] Renato Sortino, Daniel Magro, Giuseppe Fiameni, Eva Sciacca, Simone Riggi, Andrea DeMarco, Concetto Spampinato, Andrew M Hopkins, Filomena Bufano, Francesco Schillirò, Cristobal Bordiu, and Carmelo Pino. Radio astronomical images object detection and segmentation: a benchmark on deep learning methods. *Experimental Astronomy*, 56:293–331, 2023. ISSN 1572-9508. doi: 10.1007/s10686-023-09893-w. URL https://doi.org/10.1007/s10686-023-09893-w.

[27] Ray P Norris, A M Hopkins, J Afonso, S Brown, J J Condon, L Dunne, I Feain, R Hollow, M Jarvis, M Johnston-Hollitt, and et al. Emu: Evolutionary map of the universe. *Publications of the Astronomical Society of Australia*, 28:215–248, 2011. doi: 10.1071/AS11021. Publisher: Cambridge University Press.

[28] Usama M. Fayyad, Nicholas Weir, and S. George Djorgovski. SKICAT: A machine learning system for automated cataloging of large scale sky surveys. In Paul E. Utgoff, editor, *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 112–119. Morgan Kaufmann, 1993. doi: 10.1016/B978-1-55860-307-3.50021-6. URL `https://doi.org/10.1016/b978-1-55860-307-3.50021-6`.

[29] Nicholas M. Ball, Robert J. Brunner, Adam D. Myers, and David Tcheng. Robust machine learning applied to astronomical data sets. i. star-galaxy classification of the sloan digital sky survey dr3 using decision trees. *The Astrophysical Journal*, 650(1):497, oct 2006. doi: 10.1086/507440. URL `https://dx.doi.org/10.1086/507440`.

[30] Zhang Y. and Zhao Y. Automated clustering algorithms for classification of astronomical objects. *A and A*, 422:1113–1121, 2004. doi: 10.1051/0004-6361:20040141. URL `https://doi.org/10.1051/0004-6361:20040141`.

[31] Mohammad Sadeghi, Mohsen Javaherian, and Halime Miraghaei. Morphological-based classifications of radio galaxies using supervised machine-learning methods associated with image moments. *The Astronomical Journal*, 161:94, 1 2021. doi: 10.3847/1538-3881/abd314. URL `https://dx.doi.org/10.3847/1538-3881/abd314`.

[32] M J Alger, J K Banfield, C S Ong, L Rudnick, O I Wong, C Wolf, H Andernach, R P Norris, and S S Shabala. Radio galaxy zoo: machine learning for radio source host galaxy cross-identification. *Monthly Notices of the Royal Astronomical Society*, 478:5547–5563, 8 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty1308. URL `https://doi.org/10.1093/mnras/sty1308`.

[33] Ting-Yun Cheng, Christopher J Conselice, Alfonso Aragón-Salamanca, Nan Li, Asa F L Bluck, Will G Hartley, James Annis, David Brooks, Peter Doel, Juan García-Bellido, David J James, Kyler Kuehn, Nikolay Kuropatkin, Mathew Smith, Flavia Sobreira, and Gregory Tarle. Optimizing automatic morphological classification of galaxies with machine learning and deep learning using dark energy survey imaging. *Monthly Notices of the Royal Astronomical Society*, 493:4209–4228, 4 2020. ISSN 0035-8711. doi: 10.1093/mnras/staa501. URL `https://doi.org/10.1093/mnras/staa501`.

[34] B Becker and T Grobler. Classification of fanaroff-riley radio galaxies using conventional machine learning techniques. pages 1–8, 2019. doi: 10.1109/IMITEC45504.2019.9015881.

[35] Lara Alegre, Jose Sabater, Philip Best, Rafaël I J Mostert, Wendy L Williams, Gülay Gürkan, Martin J Hardcastle, Rohit Kondapally, Tim W Shimwell, and Daniel J B Smith. A machine-learning classifier for lofar radio galaxy cross-matching techniques. *Monthly Notices of the Royal Astronomical Society*, 516:4716–4738, 11 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac1888. URL `https://doi.org/10.1093/mnras/stac1888`.

[36] D. Carbone, H. Garsden, H. Spreeuw, J. D. Swinbank, A. J. van der Horst, A. Rowlinson, J. W. Broderick, E. Rol, C. Law, G. Molenaar, and R. A.M.J. Wijers. Pyse: Software for extracting sources from radio images. *Astronomy and Computing*, 23:92–102, 4 2018. ISSN 2213-1337. doi: 10.1016/J.ASCOM.2018.02.003.

[37] M P van Haarlem, M W Wise, A W Gunst, G Heald, J P McKean, J W T Hessels, A G de Bruyn, R Nijboer, J Swinbank, R Fallows, M Brentjens, A Nelles, R Beck, H Falcke, R Fender, J Hörandel, L V E Koopmans, G Mann, G Miley, H Röttgering, B W Stappers, R A M J Wijers, S Zaroubi, M van den Akker, A Alexov, J Anderson, K Anderson, A van Ardenne, M Arts, A Asgekar, I M Avruch, F Batejat, L Bähren, M E Bell, M R Bell, I van Bemmel, P Bennema, M J Bentum, G Bernardi, P Best, L Bîrzan, A Bonafede, A.-J. Boonstra, R Braun, J Bregman, F Breitling, R H van de Brink, J Broderick, P C Broekema, W N Brouw, M Brüggen, H R Butcher, W van Cappellen, B Ciardi, T Coenen, J Conway, A Coolen, A Corstanje, S Damstra, O Davies, A T Deller, R.-J. Dettmar, G van Diepen, K Dijkstra, P Donker, A Doorduin, J Dromer, M Drost, A van Duin, J Eislöffel, J van Enst, C Ferrari, W Frieswijk, H Gankema, M A Garrett, F de Gasperin, M Gerbers, E de Geus, J.-M. Grießmeier, T Grit, P Gruppen, J P Hamaker, T Hassall, M Hoeft, H A Holties, A Horneffer, A van der Horst, A van Houwelingen, A Huijgen,

M Iacobelli, H Intema, N Jackson, V Jelic, A de Jong, E Juette, D Kant, A Karastergiou, A Koers, H Kollen, V I Kondratiev, E Kooistra, Y Koopman, A Koster, M Kuniyoshi, M Kramer, G Kuper, P Lambropoulos, C Law, J van Leeuwen, J Lemaitre, M Loose, P Maat, G Macario, S Markoff, J Masters, R A McFadden, D McKay-Bukowski, H Meijering, H Meulman, M Mevius, E Middelberg, R Millenaar, J C A Miller-Jones, R N Mohan, J D Mol, J Morawietz, R Morganti, D D Mulcahy, E Mulder, H Munk, L Nieuwenhuis, R van Nieuwpoort, J E Noordam, M Norden, A Noutsos, A R Offringa, H Olofsson, A Omar, E Orrú, R Overeem, H Paas, M Pandey-Pommier, V N Pandey, R Pizzo, A Polatidis, D Rafferty, S Rawlings, W Reich, J.-P. de Reijer, J Reitsma, G A Renting, P Riemers, E Rol, J W Romein, J Roosjen, M Ruiter, A Scaife, K van der Schaaf, B Scheers, P Schellart, A Schoenmakers, G Schoonderbeek, M Serylak, A Shulevski, J Sluman, O Smirnov, C Sobey, H Spreeuw, M Steinmetz, C G M Sterks, H.-J. Stiepel, K Stuurwold, M Tagger, Y Tang, C Tasse, I Thomas, S Thoudam, M C Toribio, B van der Tol, O Usov, M van Veelen, A.-J. van der Veen, S ter Veen, J P W Verbiest, R Vermeulen, N Vermaas, C Vocks, C Vogt, M de Vos, E van der Wal, R van Weeren, H Weggemans, P Weltevrede, S White, S J Wijnholds, T Wilhelmsson, O Wucknitz, S Yatawatta, P Zarka, A Zensus, and J van Zwieten. Lofar: The low-frequency array. *Astronomy and Astrophysics*, 556:A2, 7 2013. ISSN 1432-0746. doi: 10.1051/0004-6361/201220873. URL http://dx.doi.org/10.1051/0004-6361/201220873. Publisher: EDP Sciences.

[38] H Tang, A M M Scaife, and J P Leahy. Transfer learning for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 488:3358–3375, 9 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz1883. URL https://doi.org/10.1093/mnras/stz1883.

[39] Devina Mohan, Anna M M Scaife, Fiona Porter, Mike Walmsley, and Micah Bowles. Quantifying uncertainty in deep learning approaches to radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 511:3722–3740, 4 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac223. URL https://doi.org/10.1093/mnras/stac223.

[40] A K Aniyan and K Thorat. Classifying radio galaxies with the convolutional neural network. *The Astrophysical Journal Supplement Series*, 230:20, 2017. ISSN 0067-0049. doi: 10.3847/1538-4365/aa7333. URL https://dx.doi.org/10.3847/1538-4365/aa7333.

[41] Wathela Alhassan, A R Taylor, and Mattia Vaccari. The first classifier: compact and extended radio galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 480:2085–2093, 10 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty2038. URL https://doi.org/10.1093/mnras/sty2038.

[42] V Lukic, M Brüggen, J K Banfield, O I Wong, L Rudnick, R P Norris, and B Simmons. Radio galaxy zoo: compact and extended radio source classification with deep learning. *Monthly Notices of the Royal Astronomical Society*, 476:246–260, 5 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty163. URL https://doi.org/10.1093/mnras/sty163.

[43] Burger Becker, Mattia Vaccari, Matthew Prescott, and Trienko Grobler. Cnn architecture comparison for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 503:1828–1846, 5 2021. ISSN 0035-8711. doi: 10.1093/mnras/stab325. URL https://doi.org/10.1093/mnras/stab325.

[44] Viera Maslej-Krešňáková, Khadija El Bouchefry, and Peter Butka. Morphological classification of compact and extended radio galaxies using convolutional neural networks and data augmentation techniques. *Monthly Notices of the Royal Astronomical Society*, 505:1464–1475, 7 2021. ISSN 0035-8711. doi: 10.1093/mnras/stab1400. URL https://doi.org/10.1093/mnras/stab1400.

[45] H Tang, A M M Scaife, O I Wong, and S S Shabala. Radio galaxy zoo: giant radio galaxy classification using multidomain deep learning. *Monthly Notices of the Royal Astronomical Society*, 510:4504–4524, 3 2022. ISSN 0035-8711. doi: 10.1093/mnras/stab3553. URL https://doi.org/10.1093/mnras/stab3553.

[46] Zafiirah Banon Hosenie. Source classification in deep radio surveys using machine learning techniques. 2018.

[47] Kai Lars Polsterer, Fabian Gieseke, and Christian Igel. Automatic galaxy classification via machine learning techniques: Parallelized rotation/flipping invariant kohonen maps (pink). *Astronomical data analysis software and systems XXIV (ADASS XXIV)*, 495:81, 2015. Publisher: Citeseer.

[48] T J Galvin, M T Huynh, R P Norris, X R Wang, E Hopkins, K Polsterer, N O Ralph, A N O'Brien, and G H Heald. Cataloguing the radio-sky with unsupervised machine learning: a new approach for the ska era. *Monthly Notices of the Royal Astronomical Society*, 497:2730–2758, 9 2020. ISSN 0035-8711. doi: 10.1093/mnras/staa1890. URL `https://doi.org/10.1093/mnras/staa1890`.

[49] Inigo V Slijepcevic, Anna M M Scaife, Mike Walmsley, Micah Bowles, O Ivy Wong, Stanislav S Shabala, and Hongming Tang. Radio galaxy zoo: using semi-supervised learning to leverage large unlabelled data sets for radio galaxy classification under data set shift. *Monthly Notices of the Royal Astronomical Society*, 514:2599–2613, 8 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac1135. URL `https://doi.org/10.1093/mnras/stac1135`.

[50] Mir Sazzat Hossain, Sugandha Roy, K M B Asad, Arshad Momen, Amin Ahsan Ali, M Ashraful Amin, and A K M Mahbubur Rahman. Morphological classification of radio galaxies using semi-supervised group equivariant cnns. *Procedia Computer Science*, 222:601–612, 2023. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2023.08.198. URL `https://www.sciencedirect.com/science/article/pii/S1877050923009638`.

[51] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, Lorenzomammana, Tkianai, YxNONG, AlexWang1900, Laurentiu Diaconu, Marc, Wanghaoyang0106, Ml5ah, Doug, Hatovix, Jake Poznanski, Lijun Yu, Changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, Hopesala, Pritul Dave, and Yzchen. ultralytics/yolov5: v3.0, 8 2020. Version Number: v3.0.

[52] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[54] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Comput. Surv.*, 54, 9 2022. ISSN 0360-0300. doi: 10.1145/3505244. URL `https://doi.org/10.1145/3505244`.

[55] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. pages 213–229. Springer International Publishing, 2020. ISBN 978-3-030-58452-8.

[56] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection, 2021.

[57] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr, 2022.

[58] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising, 2022.

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[60] J K Banfield, O I Wong, K W Willett, R P Norris, L Rudnick, S S Shabala, B D Simmons, C Snyder, A Garon, N Seymour, E Middelberg, H Andernach, C J Lintott, K Jacob, A D Kapińska, M Y Mao, K L Masters, M J Jarvis, K Schawinski, E Paget, R Simpson, H.-R. Klöckner, S Bamford, T Burchell, K E Chow, G Cotter, L Fortson, I Heywood, T W Jones, S Kaviraj, Á R López-Sánchez, W P Maksym, K Polsterer, K Borden, R P Hollow, and L Whyte. Radio

galaxy zoo: host galaxies and radio morphologies derived from visual inspection. *Monthly Notices of the Royal Astronomical Society*, 453:2326–2340, 11 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1688. URL https://doi.org/10.1093/mnras/stv1688.

[61] Robert H Becker, Richard L White, and David J Helfand. The first survey: Faint images of the radio sky at twenty centimeters. 450:559, 9 1995. doi: 10.1086/176166.

[62] T M O Franzen, J K Banfield, C A Hales, A Hopkins, R P Norris, N Seymour, K E Chow, A Herzog, M T Huynh, E Lenc, M Y Mao, and E Middelberg. Atlas – i. third release of 1.4 ghz mosaics and component catalogues. *Monthly Notices of the Royal Astronomical Society*, 453: 4020–4036, 11 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1866. URL https://doi.org/10.1093/mnras/stv1866.

[63] Chen Wu, Oiwei Ivy Wong, Lawrence Rudnick, Stanislav S Shabala, Matthew J Alger, Julie K Banfield, Cheng Soon Ong, Sarah V White, Avery F Garon, Ray P Norris, Heinz Andernach, Jean Tate, Vesna Lukic, Hongming Tang, Kevin Schawinski, and Foivos I Diakogiannis. Radio galaxy zoo: Claran – a deep learning classifier for radio morphologies. *Monthly Notices of the Royal Astronomical Society*, 482:1211–1230, 1 2019. ISSN 0035-8711. doi: 10.1093/mnras/sty2646. URL https://doi.org/10.1093/mnras/sty2646.

[64] B L Fanaroff and J M Riley. The morphology of extragalactic radio sources of high and low luminosity. *Monthly Notices of the Royal Astronomical Society*, 167:31P–36P, 4 1974. ISSN 0035-8711. doi: 10.1093/mnras/167.1.31P. URL https://doi.org/10.1093/mnras/167.1.31P.

[65] Eric W Greisen. Aips fits file format. *AIPS Memo*, 117, 2012.

[66] S J Tingay, R Goeke, J D Bowman, D Emrich, S M Ord, D A Mitchell, M F Morales, T Booler, B Crosse, R B Wayth, C J Lonsdale, S Tremblay, D Pallot, T Colegate, A Wicenec, N Kudryavtseva, W Arcus, D Barnes, G Bernardi, F Briggs, S Burns, J D Bunton, R J Cappallo, B E Corey, A Deshpande, L Desouza, B M Gaensler, L J Greenhill, P J Hall, B J Hazelton, D Herne, J N Hewitt, M Johnston-Hollitt, D L Kaplan, J C Kasper, B B Kincaid, R Koenig, E Kratzenberg, M J Lynch, B Mckinley, S R Mcwhirter, E Morgan, D Oberoi, J Pathikulangara, T Prabu, R A Remillard, A E E Rogers, A Roshi, J E Salah, R J Sault, N Udaya-Shankar, F Schlagenhaufer, K S Srivani, J Stevens, R Subrahmanyan, M Waterson, R L Webster, A R Whitney, A Williams, C L Williams, and J S B Wyithe. The murchison widefield array: The square kilometre array precursor at low radio frequencies. *pasa*, 30:e007, 1 2013. doi: 10.1017/pasa.2012.007.

[67] R B Wayth, E Lenc, M E Bell, J R Callingham, K S Dwarakanath, T M O Franzen, B.-Q. For, B Gaensler, P Hancock, L Hindson, N Hurley-Walker, C A Jackson, M Johnston-Hollitt, A D Kapińska, B McKinley, J Morgan, A R Offringa, P Procopio, L Staveley-Smith, C Wu, Q Zheng, C M Trott, G Bernardi, J D Bowman, F Briggs, R J Cappallo, B E Corey, A A Deshpande, D Emrich, R Goeke, L J Greenhill, B J Hazelton, D L Kaplan, J C Kasper, E Kratzenberg, C J Lonsdale, M J Lynch, S R McWhirter, D A Mitchell, M F Morales, E Morgan, D Oberoi, S M Ord, T Prabu, A E E Rogers, A Roshi, N Udaya Shankar, K S Srivani, R Subrahmanyan, S J Tingay, M Waterson, R L Webster, A R Whitney, A Williams, and C L Williams. Gleam: The galactic and extragalactic all-sky mwa survey. *Publications of the Astronomical Society of Australia*, 32, 2015. ISSN 1448-6083. doi: 10.1017/pasa.2015.26. URL http://dx.doi.org/10.1017/pasa.2015.26. Publisher: Cambridge University Press (CUP).

[68] N Hurley-Walker, J. R. Callingham, P. J. Hancock, T. M. O. Franzen, L Hindson, A. D. Kapińska, J Morgan, A. R. Offringa, R. B. Wayth, C Wu, Q Zheng, T Murphy, M. E. Bell, K. S. Dwarakanath, B For, B. M. Gaensler, M Johnston-Hollitt, E Lenc, P Procopio, L Staveley-Smith, R Ekers, J. D. Bowman, F Briggs, R. J. Cappallo, A. A. Deshpande, L Greenhill, B. J. Hazelton, D. L. Kaplan, C. J. Lonsdale, S. R. McWhirter, D. A. Mitchell, M. F. Morales, E Morgan, D Oberoi, S. M. Ord, T Prabu, N Udaya Shankar, K. S. Srivani, R Subrahmanyan, S. J. Tingay, R. L. Webster, A Williams, and C. L. Williams. Galactic and extragalactic all-sky murchison widefield array (gleam) survey - i. a low-frequency extragalactic catalogue. 464:1146–1167, 1 2017. doi: 10.1093/mnras/stw2337.

[69] Miguel F Morales and J Stuart B Wyithe. Reionization and cosmology with 21-cm fluctuations. *araa*, 48:127–171, 9 2010. doi: 10.1146/annurev-astro-081309-130936.

[70] Steven R Furlanetto, S Peng Oh, and Frank H Briggs. Cosmology at low frequencies: The 21cm transition and the high-redshift universe. *Physics Reports*, 433:181–301, 10 2006. ISSN 0370-1573. doi: 10.1016/j.physrep.2006.08.002. URL http://dx.doi.org/10.1016/j.physrep.2006.08.002. Publisher: Elsevier BV.

[71] Sihan Li, Jiantao Jiao, Yanjun Han, and Tsachy Weissman. Demystifying resnet, 2017.

[72] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures, 2016.

[73] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:31, 1991.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[75] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9, pages 249–256. PMLR, 5 2010. URL https://proceedings.mlr.press/v9/glorot10a.html.

[76] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[77] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training, 2021.

[78] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. 6 2020.

[79] Suyog Gupta and Mingxing Tan. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl. *Google AI Blog*, 2, 2019.

[80] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection, 2020.

[81] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. Publisher: OpenAI.

[82] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[83] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[84] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[85] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2022.

[86] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention, 2021.

[87] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[88] Ye Wang, Zhenyi Liu, and Weiwen Deng. Anchor generation optimization and region of interest assignment for vehicle detection. *Sensors*, 19, 2019. ISSN 1424-8220. doi: 10.3390/s19051089. URL `https://www.mdpi.com/1424-8220/19/5/1089`.

[89] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[90] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[91] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. pages 7373–7382, 6 2021.

[92] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. pages 10012–10022, 10 2021.

[93] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. 6 2019.

[94] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. 7 2017.

[95] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021.

[96] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. 10 2017.

[97] Hongyu Zhai, Jian Cheng, and Mengyong Wang. Rethink the iou-based loss functions for bounding box regression. volume 9, pages 1522–1528, 2020. doi: 10.1109/ITAIC49862.2020.9339070.

[98] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. 6 2019.

[99] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond, 2023.

[100] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. 7 2017.

[101] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[102] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[103] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[104] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. pages 7464–7475, 6 2023.

[105] Yuanyi Zhong, Jianfeng Wang, Jian Peng, and Lei Zhang. Anchor box optimization for object detection. 3 2020.

[106] James MacQueen et al. Some methods for classification and analysis of multivariate observations. volume 1, pages 281–297. Oakland, CA, USA, 1967. Issue: 14.

[107] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021.

[108] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation, 2021.

[109] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, 2020.

[110] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H. Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: A systematic review. *Journal of Neural Engineering*, 16, 8 2019. ISSN 17412552. doi: 10.1088/1741-2552/ab260c.

[111] Andrew Ng. *Machine Learning Yearning*. Online Draft, 2017.

[112] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8, 2019. ISSN 2079-9292. doi: 10.3390/electronics8080832. URL https://www.mdpi.com/2079-9292/8/8/832.

[113] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. URL https://arxiv.org/abs/1702.08608.

[114] Brendan Juba and Hai S Le. Precision-recall versus accuracy and the role of large data sets. volume 33, pages 4039–4048, 2019. Issue: 01.

[115] I Bratko. Machine learning: Between accuracy and interpretability. pages 163–177. Springer Vienna, 1997. ISBN 978-3-7091-2668-4.

[116] Charles X Ling, Jin Huang, and Harry Zhang. Auc: A better measure than accuracy in comparing learning algorithms. pages 329–341. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-44886-0.

[117] Jason Brownlee. Failure of classification accuracy for imbalanced class distributions, 1 2021. URL https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/.

[118] Classification: Accuracy, 7 2022. URL https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[119] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[120] Classification: Precision and recall, 7 2022. URL https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[121] Olivier Caelen. A bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence*, 81:429–450, 12 2017. ISSN 1573-7470. doi: 10.1007/s10472-017-9564-8. URL https://doi.org/10.1007/s10472-017-9564-8.

[122] David M W Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2020. doi: 10.48550/ARXIV.2010.16061. URL https://arxiv.org/abs/2010.16061. Publisher: arXiv.

[123] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[124] Brice Ozenne, Fabien Subtil, and Delphine Maucort-Boulch. The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases. *Journal of Clinical Epidemiology*, 68:855–859, 2015. ISSN 0895-4356. doi: https://doi.org/10.1016/j.jclinepi.2015.02.010. URL https://www.sciencedirect.com/science/article/pii/S0895435615001067.

[125] Zhe Hui Hoo, Jane Candlish, and Dawn Teare. What is an roc curve?, 2017. Issue: 6 Pages: 357–359 Publication Title: Emergency Medicine Journal Volume: 34.

[126] Jonathan Cook and Vikram Ramadas. When to consult precision-recall curves. *The Stata Journal*, 20:131–148, 3 2020. ISSN 1536-867X. doi: 10.1177/1536867X20909693.

[127] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. pages 740–755. Springer International Publishing, 2014. ISBN 978-3-319-10602-1.

[128] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. pages 340–353. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33712-3.

[129] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. pages 234–244. Springer International Publishing, 2016. ISBN 978-3-319-50835-1.

[130] R Padilla, S L Netto, and E A B da Silva. A survey on performance metrics for object-detection algorithms. pages 237–242, 2020. ISBN 2157-8702. doi: 10.1109/IWSSIP48289.2020.9145130.

[131] Vinod Sharma. A study on data scaling methods for machine learning. *International Journal for Global Academic and Scientific Research*, 1:23–33, 2 2022. doi: 10.55938/ijgasr.v1i1.4. URL https://journals.icapsr.com/index.php/ijgasr/article/view/4.

[132] Doug Tody. The iraf data reduction and analysis system. volume 627, pages 733–748. SPIE, 1986.

[133] S Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage, 2015.

[134] Richard L White, David J Helfand, Robert H Becker, Eilat Glikman, and Wim de Vries. Signals from the noise: Image stacking for quasars in the first survey. *The Astrophysical Journal*, 654:99, 2007. ISSN 0004-637X. doi: 10.1086/507700. URL https://dx.doi.org/10.1086/507700.

[135] Astropy Collaboration. The astropy project: Sustaining and growing a community-oriented open-source project and the latest major release (v5.0) of the core package. 935:167, 8 2022. doi: 10.3847/1538-4357/ac7c74.

[136] Florin Dumitrescu, Bogdan Ceachi, Ciprian-Octavian Truică, Mihai Trăscău, and Adina Magda Florea. A novel deep learning-based relabeling architecture for space objects detection from partially annotated astronomical images. *Aerospace*, 9, 2022. ISSN 2226-4310. doi: 10.3390/aerospace9090520. URL https://www.mdpi.com/2226-4310/9/9/520.

[137] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:60, 7 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0. URL https://doi.org/10.1186/s40537-019-0197-0.

[138] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. pages 1542–1547, 2018. doi: 10.1109/SSCI.2018.8628742.

[139] Cédric Rommel, Joseph Paillard, Thomas Moreau, and Alexandre Gramfort. Data augmentation for learning predictive models on eeg: a systematic comparison, 2022. URL https://arxiv.org/abs/2206.14483.

[140] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[141] J Deng, W Dong, R Socher, L.-J. Li, K Li, and L Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[142] Rahul Rahaman and alexandre thiery. Uncertainty quantification and deep ensembles. volume 34, pages 20063–20075. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/a70dc40477bc2adceef4d2c90f47eb82-Paper.pdf.

[143] Göran, Karlsson Alexander, Mathiason Gunnar Ståhl Niclas, and Falkman. Evaluation of uncertainty quantification in deep learning. pages 556–568. Springer International Publishing, 2020. ISBN 978-3-030-50146-4.

[144] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[145] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[146] Amanda Olmin. On uncertainty quantification in neural networks: Ensemble distillation and weak supervision. 2022.

[147] Shymaa Akram Hantoush Alrubaie and Ahmed Hasan Hameed. Dynamic weights equations for converting grayscale image to rgb image. *JOURNAL OF UNIVERSITY OF BABYLON for Pure and Applied Sciences*, 26:122–129, 10 2018. doi: 10.29196/jubpas.v26i8.1677. URL https://www.journalofbabylon.com/index.php/JUBPAS/article/view/1677.

[148] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[149] MMEngine Contributors. Mmengine: Openmmlab foundational library for training deep learning models. 2022. URL https://github.com/open-mmlab/mmengine.

[150] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[151] MMPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark, 2023. URL https://github.com/open-mmlab/mmpretrain.

[152] MMYOLO Contributors. Mmyolo: Openmmlab yolo series toolbox and benchmark, 2022. URL https://github.com/open-mmlab/mmyolo.

[153] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[154] Omry Yadan. Hydra - a framework for elegantly configuring complex applications, 2019. URL https://github.com/facebookresearch/hydra.

[155] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[156] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[157] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4:26–31, 2012.

[158] Shymaa Akram Hantoush Alrubaie and Ahmed Hasan Hameed. Dynamic weights equations for converting grayscale image to rgb image. *Journal of University of Babylon for Pure and Applied Sciences*, 26:122–129, 2018.

# List of Figures

# List of Tables

# List of Abbreviations

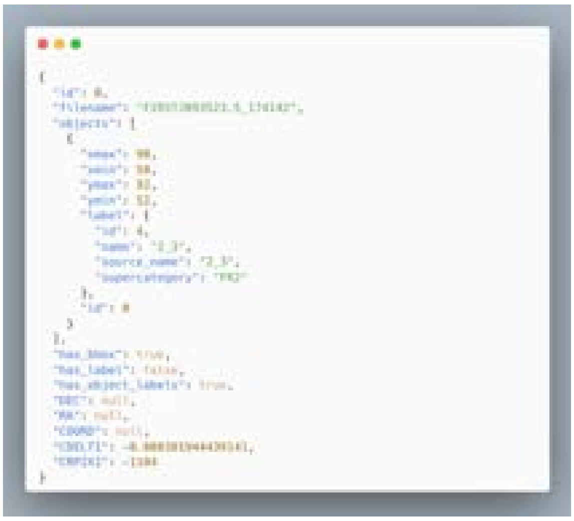
| Abbr | Abbreviation |
|---|---|
| **AP** | Average Precision |
| **ATLAS** | Australia Telescope Large Area Survey |
| **AUPRC** | Area Under the Precision-Recall Curve |
| **BYOL** | Bootstrap Your Own Latent |
| **CIoU** | Complete Intersection over Union |
| **CL** | Consensus Level |
| **CNN** | Convolutional Neural Network |
| **Conv-BN** | Convolutional Batch Normalisation |
| **ConvNet** | Convolutional Network |
| **DETR** | Detection Transformer |
| **DFL** | Distribution Focal Loss |
| **DL** | Deep Learning |
| **DR** | Data Release |
| **EMU** | Evolutionary Map of the Universe |
| **EoR** | Epoch of Reionisation |
| **FIRST** | Faint Images of the Radio Sky at Twenty Centimeters |
| **FITS** | Flexible Image Transport System |
| **FR** | Fanaroff - Riley |
| **GAN** | Generative Adversarial Network |
| **GIoU-Loss** | Generalised IoU Loss |
| **GLEAM** | GaLactic and Extragalactic All-sky MWA Survey |
| **IoU** | Intersection over Union |
| **LN** | Layer Normalisation |
| **mAP** | Mean Average Performance |
| **MB** | Meta Transformer Block |
| **MBConv** | Inverted Residual Block |
| **ML** | Machine Learning |
| **MLP** | Multi-Layer Perceptron |
| **MWA** | Murchison Widefield Array |
| **OCR** | Optical Character Recognition |
| **PRC** | Precision-Recall Curve |
| **NAS** | Neural Architecture Search |
| **NLP** | Natural Language Processing |
| **NMS** | Non-maximum Suppression |
| **ResNet** | Residual Network |
| **RGZ** | Radio Galaxy Zoo |
| **RPN** | Region Proposal Network |
| **SimCLR** | Simple Framework Contrastive Learning |
| **SKA** | Square Kilometre Array |
| **SNR** | Signal-To-Noise-Ratio |

| | |
|---|---|
| **SOM** | Self-Organising Maps |
| **SOTA** | State-Of-The-Art |
| **SPP** | Spatial Pyramid Pooling |
| **SVM** | Support Vector Machine |
| **ViT** | Vision Transformer |
| **YOLO** | You Only Look Once |

# Appendix A

# Appendix

## A.1 Dataset Collection



```
{
  "id": 0,
  "filename": "[illegible]",
  "objects": [
    {
      "xmax": 99,
      "xmin": 58,
      "ymax": 82,
      "ymin": 52,
      "label": {
        "id": 4,
        "name": "2_3",
        "source_name": "2_3",
        "supercategory": "FR2"
      },
      "id": 0
    }
  ],
  "has_bbox": true,
  "has_label": false,
  "has_object_labels": true,
  "DEC": null,
  "RA": null,
  "COORD": null,
  "CDELT1": -0.[illegible],
  "CRPIX1": -1184
}
```

Figure A.1: Dataset Collection Annotation Example

## A.2 Preprocessing

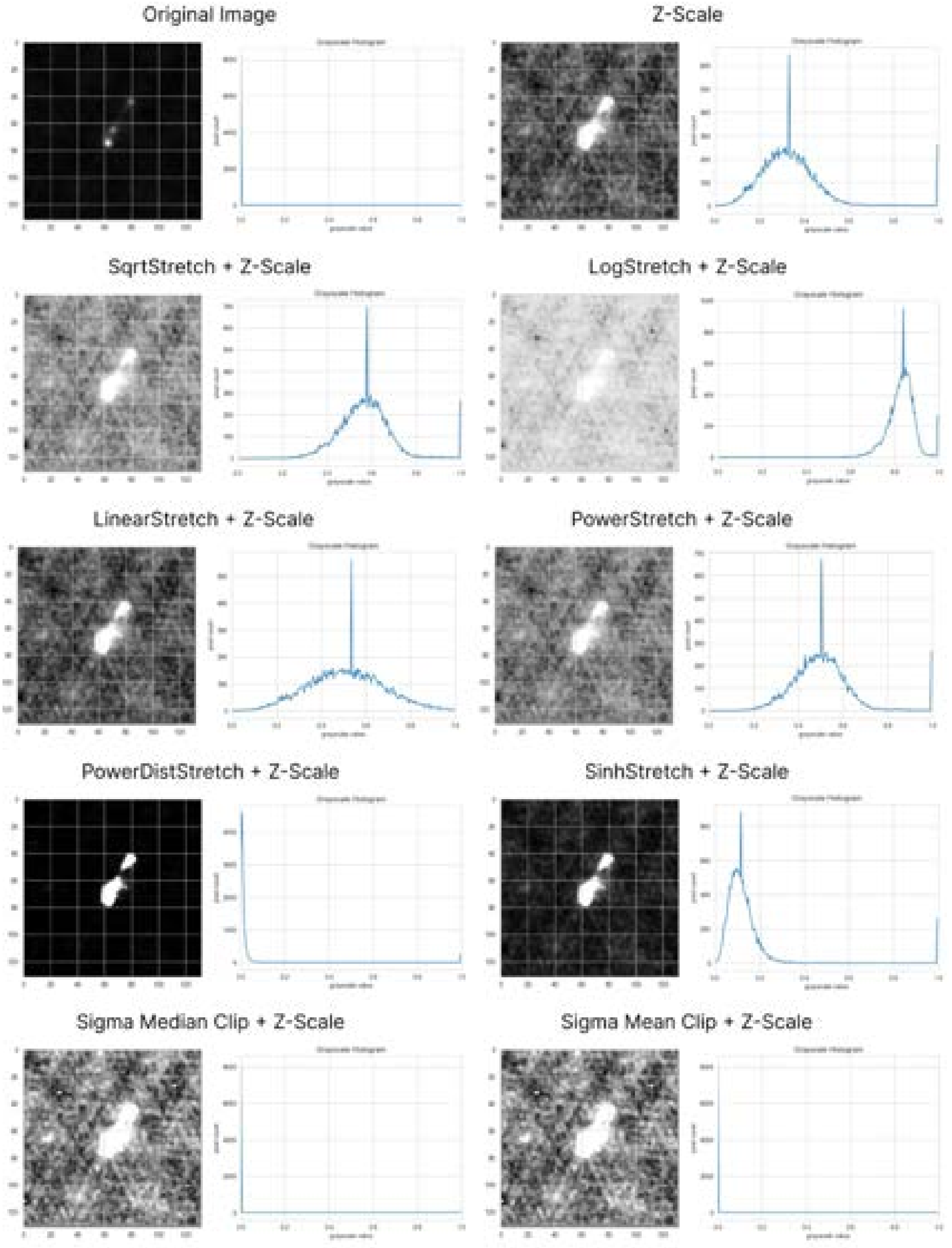
### A.2.1 Supported Preprocessing Techniques

| Name | Config Key | Description |
|---|---|---|
| Otsu's Method | OtsusThreshold | Applies Otsu's Thresholding to a given sample to eliminate noise. |
| Z-Scaling | ZScale | Applies Z-Scaling based on IRAF's Z-Scale to a given sample. See section 2.4. |
| Min-Max-Scaling | MinMaxScale | Applies Min-Max-Scaling to a given sample based on the minimum and maximum value in the data. See section 2.4. |
| Hyperbolic Sin Stretch | SinhStretch | Applies a hyperbolic sin stretch to a given sample. See section 2.4. |
| Square Root Stretch | SqrtStretch | Applies a square root stretch to a given sample. See section 2.4. |
| Linear Stretch | LinearStretch | Applies a linear stretch with a slope and offset. See section 2.4. |
| Power Stretch | PowerStretch | Applies a power stretch to a given sample. See section 2.4. |
| Power Distribution Stretch | PowerDistStretch | Applies a power distribution stretch to a given sample. See section 2.4. |
| Sigma Mean Clipping | SigmaMeanClip | Clips outlier values depending on a deviating number of standard deviation of the mean. See section 2.4. |
| Sigma Median Clipping | SigmaMedianClip | Clips outlier values depending on a deviating number of standard deviation of the median. See section 2.4. |
| Replicate Colour Channels | ReplicateChannels | Converts a grayscale image to a three channel colour image by replicating the channel to all channels. |
| CV2 Gray to Cloour | GrayToColour | Uses the CV2 cvtColor method to convert a grayscale to a three channel colour image. |
| Dynamic Weight Conversion | DynamicWeightsConversion | Takes the mean, standard deviation and skew of an image to dynamically weight a grayscale colour channel to three colour channels[158]. |
| Uint 8 Conversion | ConvertToUint8 | Converts a floating point image with values within (0, 1) to an uint8 image with values within (0, 255). |

Table A.1: Supported Preprocessing Techniques through the Preprocessing Pipeline

### A.2.2 Intensity Distributions Preprocessing Techniques

The figures in this section are showing the intensity distributions of the preprocessing techniques analysed in this project based on either z-scaling and min-max scaling.



Original Image

Z-Scale

SqrtStretch + Z-Scale

LogStretch + Z-Scale

LinearStretch + Z-Scale

PowerStretch + Z-Scale

PowerDistStretch + Z-Scale

SinhStretch + Z-Scale

Sigma Median Clip + Z-Scale

Sigma Mean Clip + Z-Scale

Figure A.2: Intensity Distributions Preprocessing Techniques based on Z-Scale

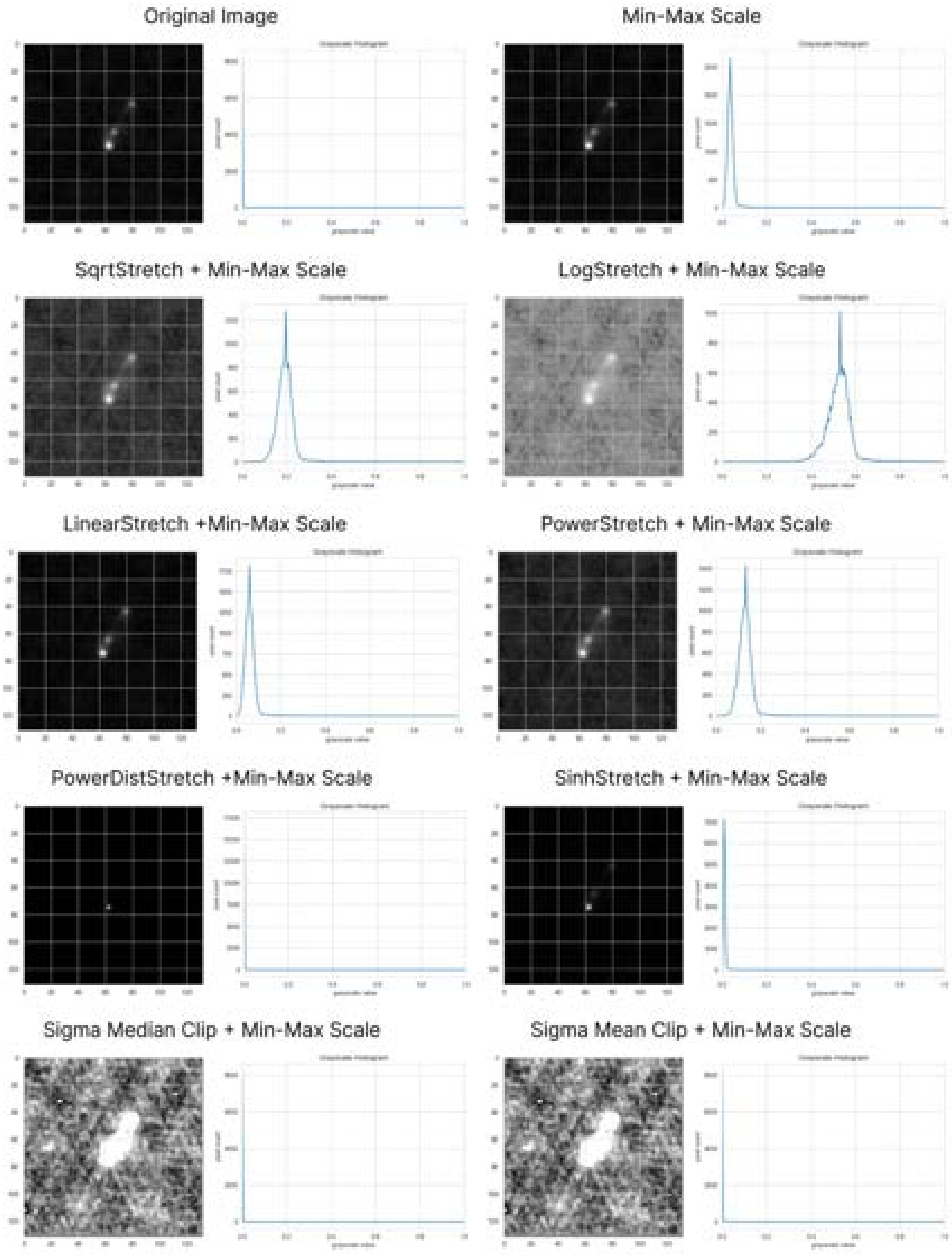


Figure A.3: Intensity Distributions Preprocessing Techniques based on Min-Max Scale

## A.3 Results

### A.3.1 Benchmark Experiments

**Classification Models**

| Model | Config | Top 1 | Top 2 | F1 Score Macro | Precision Macro | Recall Macro |
|---|---|---|---|---|---|---|
| ResNet | ZS, NPT | **83.38** | **94.53** | **79.60** | 79.67 | **79.71** |
| ResNet | ZS, NPT, US | 81.42 | 93.90 | 79.55 | **80.06** | 79.37 |
| ResNet | ZS, PT, US | 81.42 | 94.47 | 79.21 | 79.36 | 79.26 |
| ResNet | ZS, PT | 81.16 | 93.52 | 77.13 | 77.30 | 77.26 |
| EffNet | ZS, NPT | 79.23 | 92.30 | 74.00 | 74.41 | 73.72 |
| EffNet | ZS, PT | 78.42 | 92.71 | 72.29 | 72.90 | 71.93 |
| EffNet | ZS, PT, US | 76.74 | 90.21 | 74.77 | 76.10 | 74.15 |
| EffFor | ZS, NPT | 76.60 | 89.46 | 69.64 | 70.10 | 69.38 |
| ResNet | MM, PT | 75.684 | 90.48 | 68.75 | 69.62 | 68.00 |
| EffFor | ZS, PT | 75.48 | 89.56 | 68.69 | 69.24 | 68.45 |
| EffNet | MM, PT | 74.57 | - | 66.98 | 68.59 | 65.87 |
| ResNet | MM, NPT | 73.76 | 90.58 | 67.50 | 68.05 | 67.08 |
| EffNet | ZS, NPT, US | 73.19 | 93.62 | 71.76 | 72.19 | 71.83 |
| EffFor | ZS, NPT, US | 73.05 | 90.64 | 71.02 | 71.62 | 70.62 |
| EffFor | MM, PT | 72.644 | 89.06 | 65.48 | 65.77 | 65.34 |
| EffFor | MM, NPT | 71.834 | 88.86 | 64.78 | 65.37 | 64.34 |
| ResNet | MM, NPT, US | 71.77 | 89.22 | 71.00 | 71.44 | 70.79 |
| EffNet | MM, NPT | 71.631 | - | 64.00 | 65.62 | 62.70 |
| EffFor | ZS, PT, US | 69.22 | 90.07 | 66.75 | 67.30 | 66.39 |
| EffNet | MM, PT, US | 68.23 | 87.94 | 67.62 | 68.64 | 66.88 |
| ResNet | MM, PT, US | 66.81 | 86.53 | 66.89 | 67.92 | 66.08 |
| EffFor | MM, NPT, US | 64.97 | 85.11 | 64.02 | 64.70 | 63.70 |
| EffFor | MM, PT, US | 63.12 | 86.38 | 61.86 | 62.30 | 61.55 |
| EffNet | MM, NPT, US | 62.84 | 83.97 | 62.31 | 63.05 | 61.80 |
| ViT | ZS, PT | 59.473 | 77.91 | 42.33 | 55.34 | 36.41 |
| ViT | ZS, NPT | 58.46 | 77.41 | 37.19 | 60.42 | 32.63 |
| ViT | MM, PT | 52.99 | 70.72 | 19.12 | 36.71 | 18.50 |
| ViT | MM, NPT | 50.56 | 71.83 | 15.70 | 22.21 | 16.60 |
| ViT | ZS, NPT, US | 49.79 | 75.04 | 36.11 | 51.99 | 31.28 |
| ViT | ZS, PT, US | 46.95 | 72.766 | 44.53 | 50.33 | 41.06 |
| ViT | MM, PT, US | 36.31 | 61.56 | 7.55 | 26.85 | 5.39 |
| ViT | MM, NPT, US | 21.14 | 39.86 | 0 | 0 | 0 |

Table A.2: Results Benchmark Experiments | Classification Models

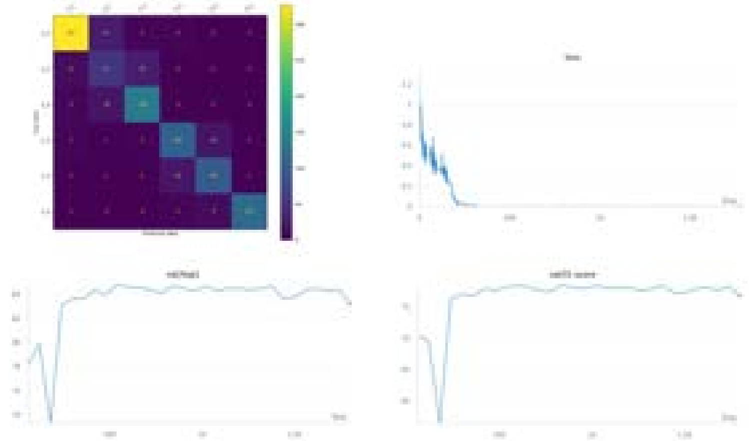


Figure A.4: ResNet50 | Z-Scale | Not Pre-Trained | Performance Metric Summary

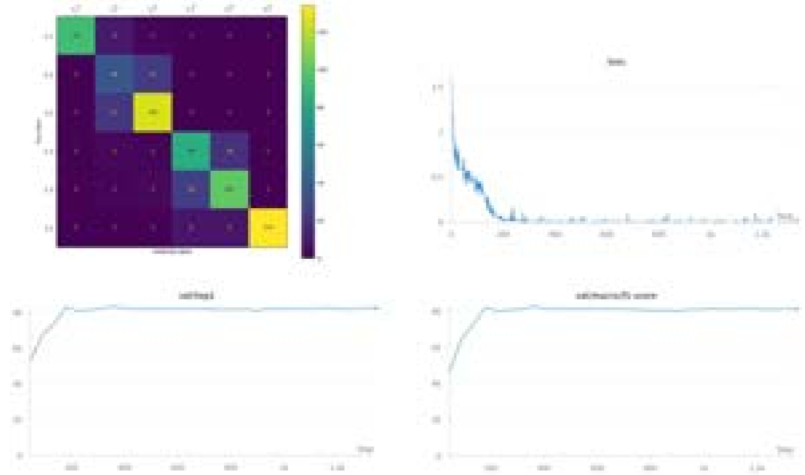


Figure A.5: ResNet50 | Z-Scale | Not Pre-Trained | Undersampling | Performance Metric Summary

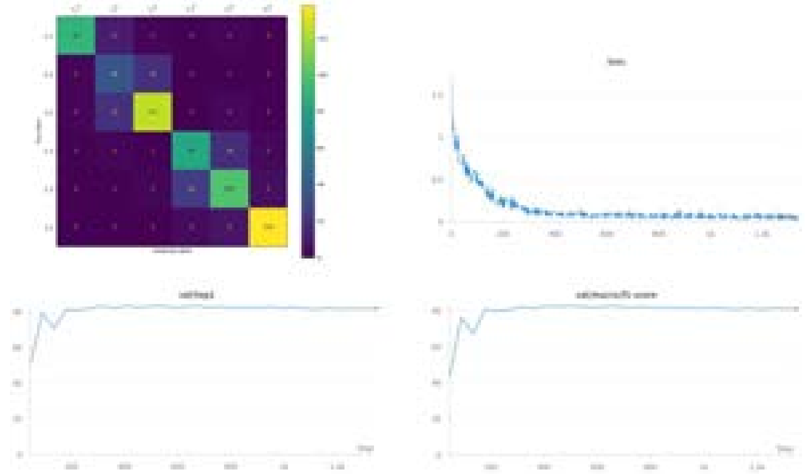


Figure A.6: ResNet50 | Z-Scale | Pre-Trained | Undersampling | Performance Metric Summary

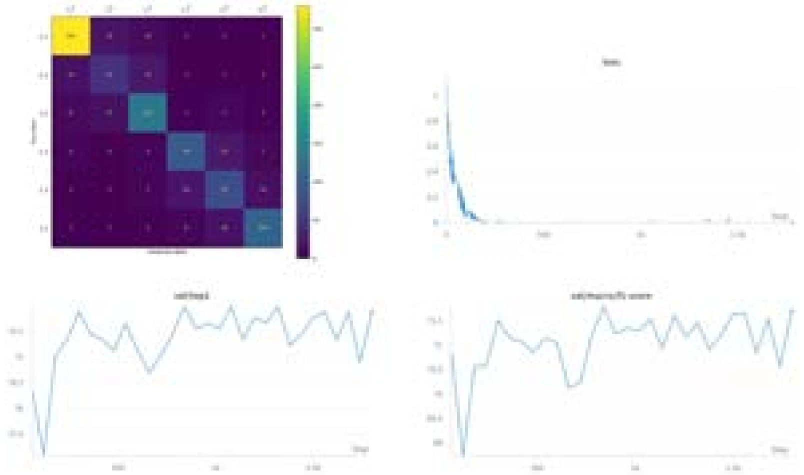


Figure A.7: EfficientNet V2 | Z-Scale | Not Pre-Trained | Performance Metric Summary

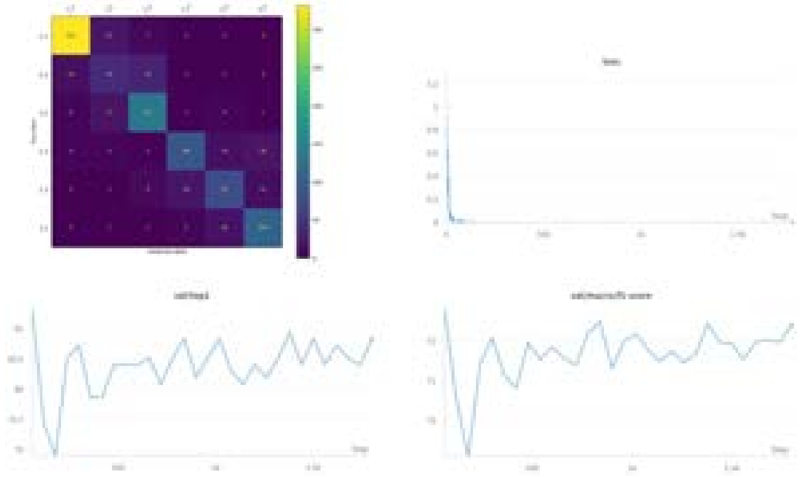


Figure A.8: EfficientNet V2 | Z-Scale | Pre-Trained | Performance Metric Summary

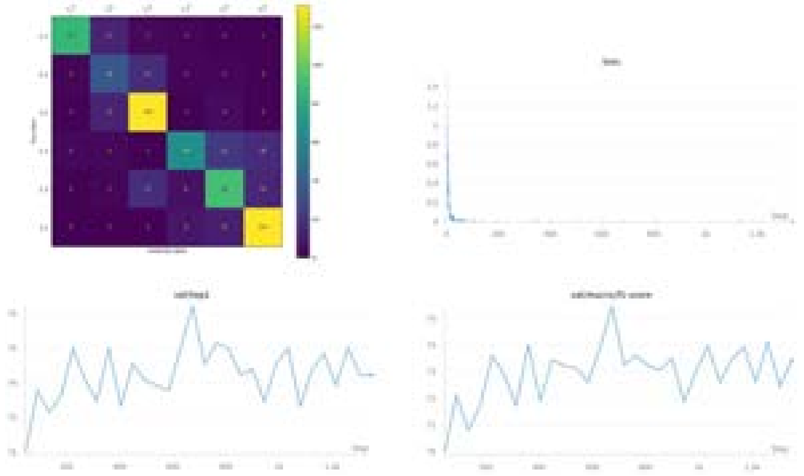


Figure A.9: EfficientNet V2 | Z-Scale | Pre-Trained | Undersampling | Performance Metric Summary

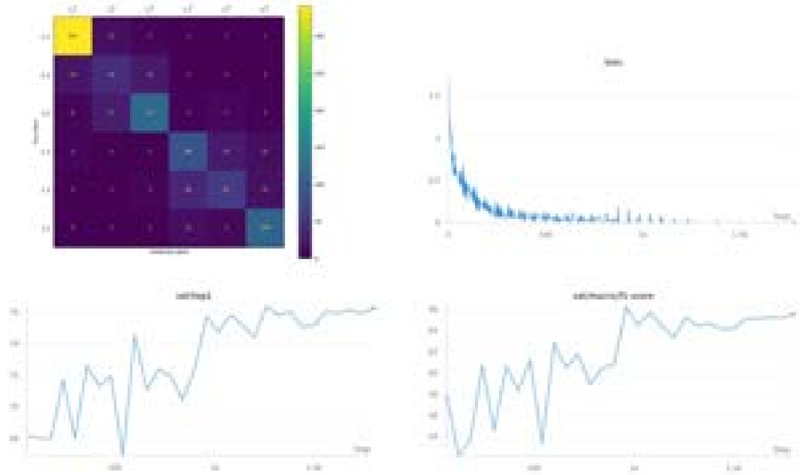


Figure A.10: EfficientFormer | Z-Scale | Not Pre-Trained | Performance Metric Summary

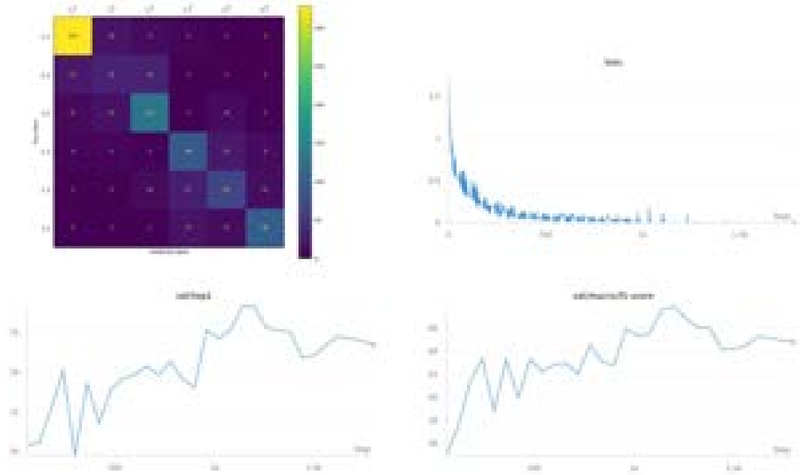


Figure A.11: EfficientFormer | Z-Scale | Pre-Trained | Performance Metric Summary

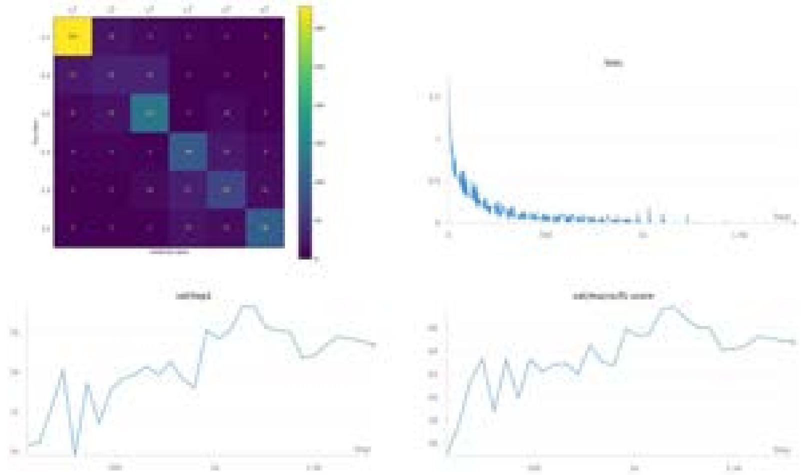


Figure A.12: EfficientFormer | Z-Scale | Not Pre-Trained | Undersampling | Performance Metric Summary

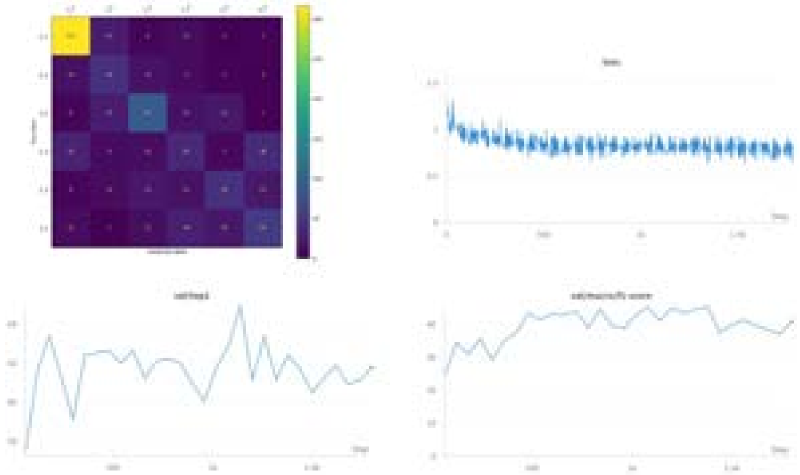


Figure A.13: Vision Transformer | Z-Scale | Pre-Trained | Performance Metric Summary

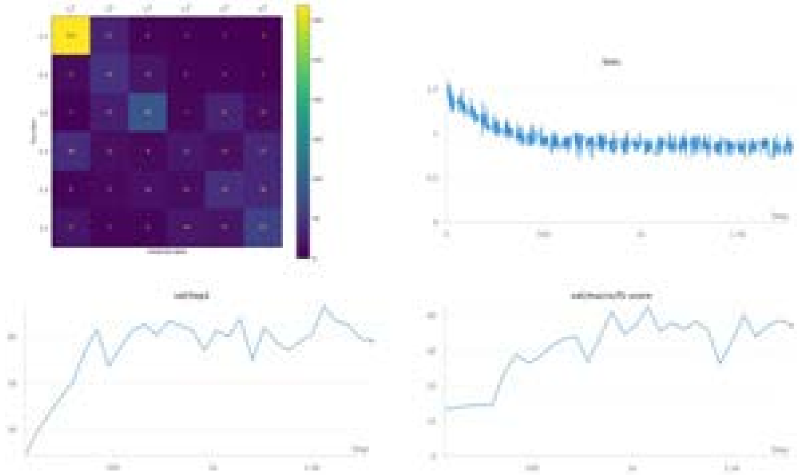


Figure A.14: Vision Transformer | Z-Scale | Not Pre-Trained | Performance Metric Summary

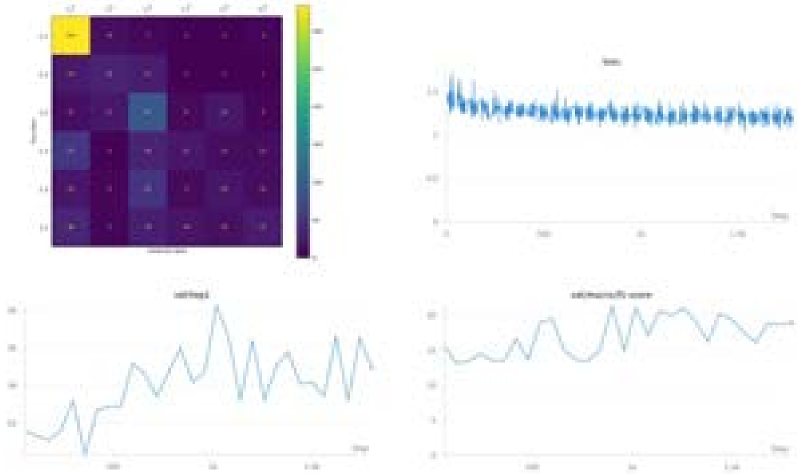


Figure A.15: Vision Transformer | Min-Max Scale | Pre-Trained | Performance Metric Summary

### Detection Models

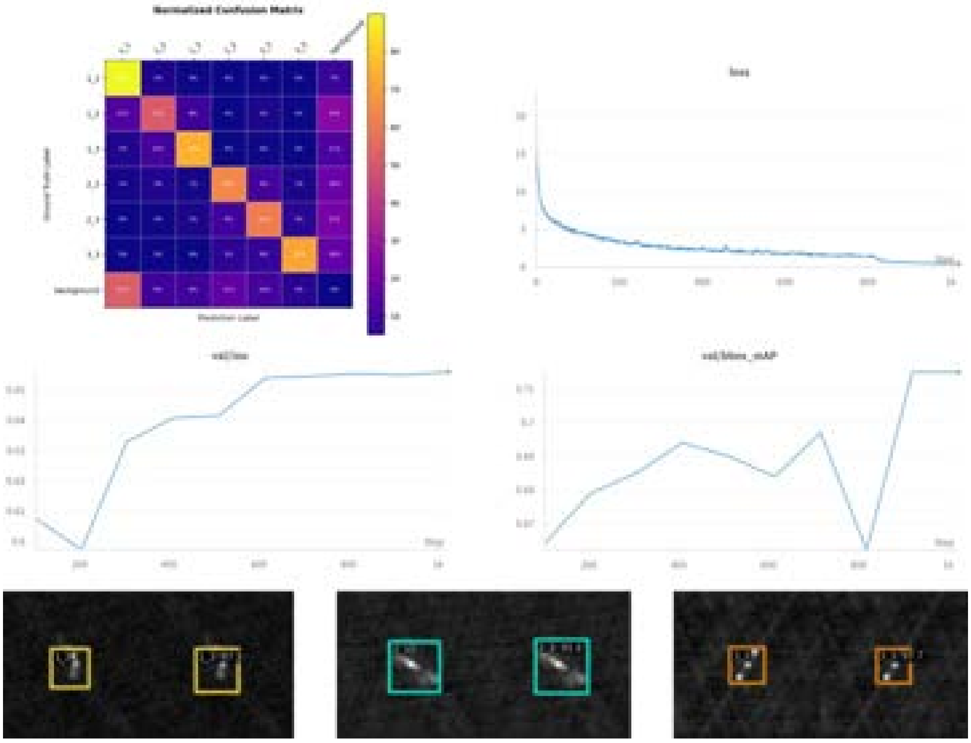


Figure A.16: DINO | Min-Max Scale | Pre-Trained | Performance Metric Summary

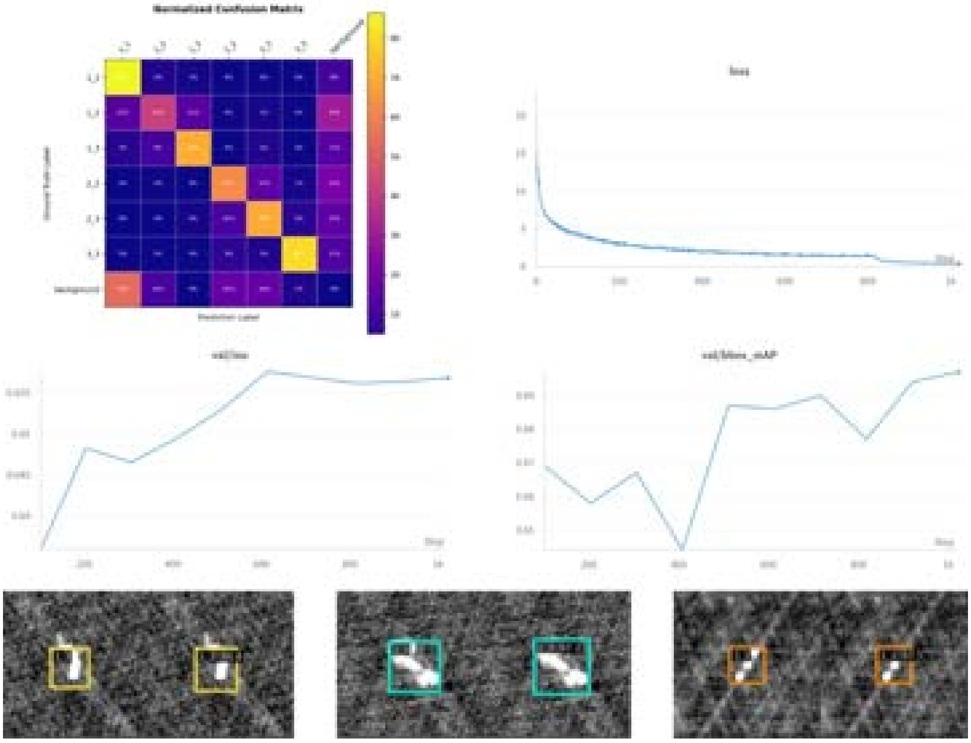


Figure A.17: DINO | Z-Scale | Pre-Trained | Performance Metric Summary

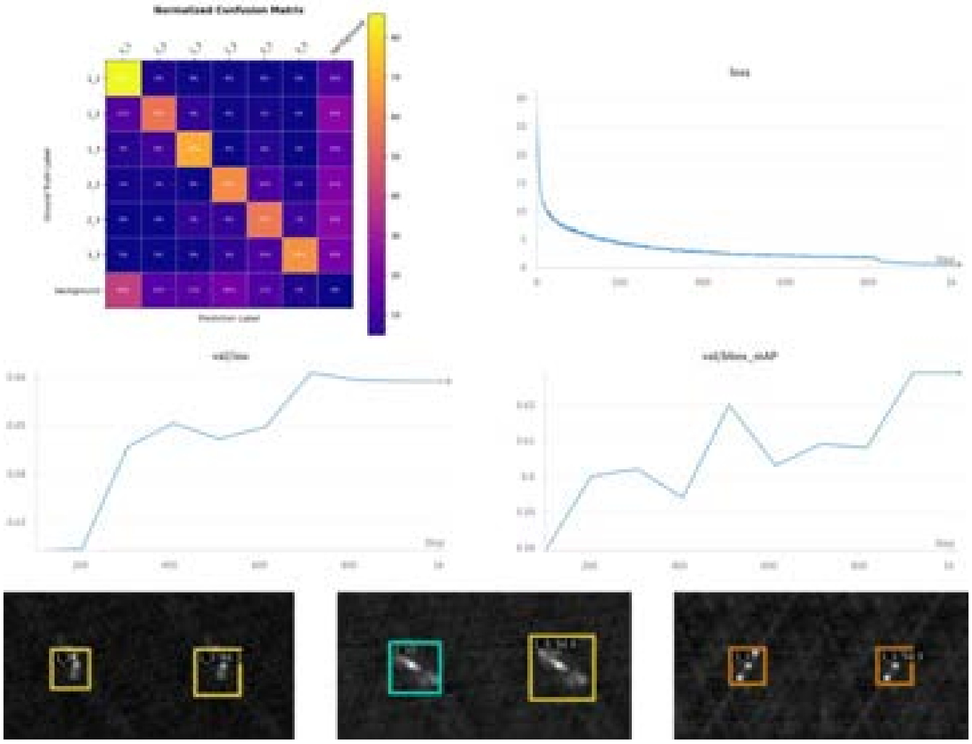


Figure A.18: DINO | Min-Max Scale | Not Pre-Trained | Performance Metric Summary

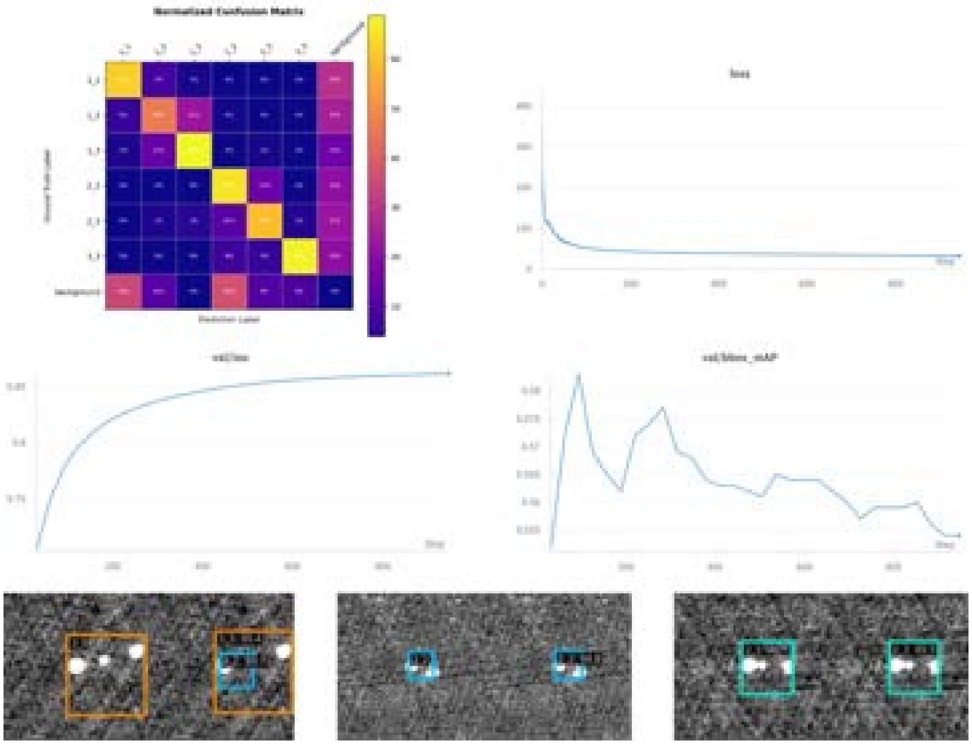


Figure A.19: YOLOv8 | Z-Scale | Pre-Trained | Undersampling | Performance Metric Summary

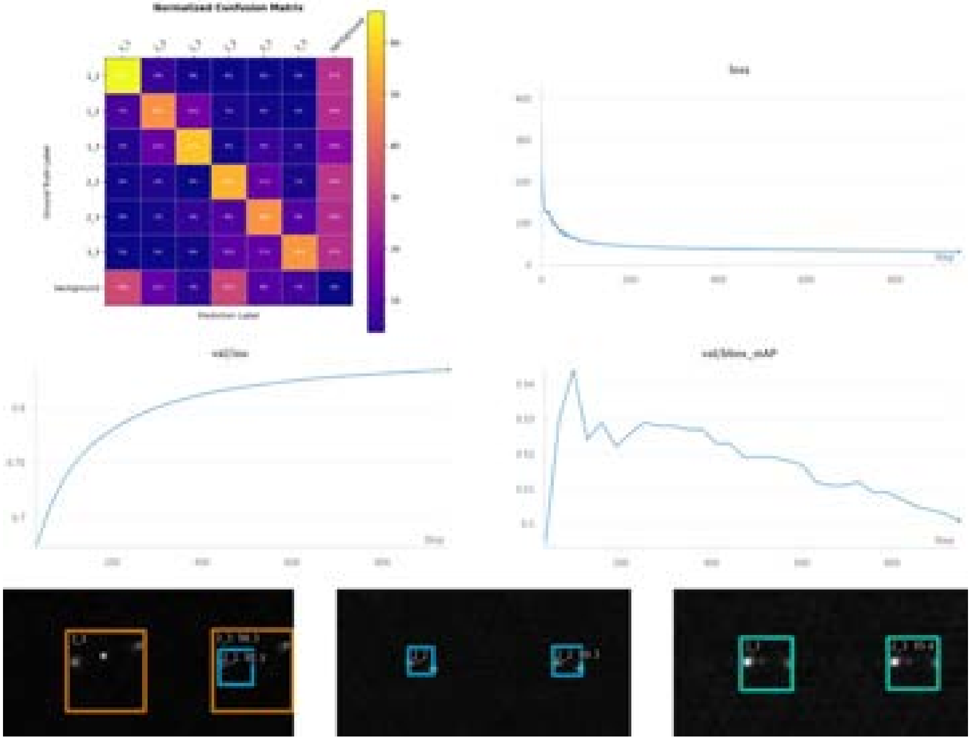


Figure A.20: YOLOv8 | Min-Max Scale | Pre-Trained | Undersampling | Performance Metric Summary

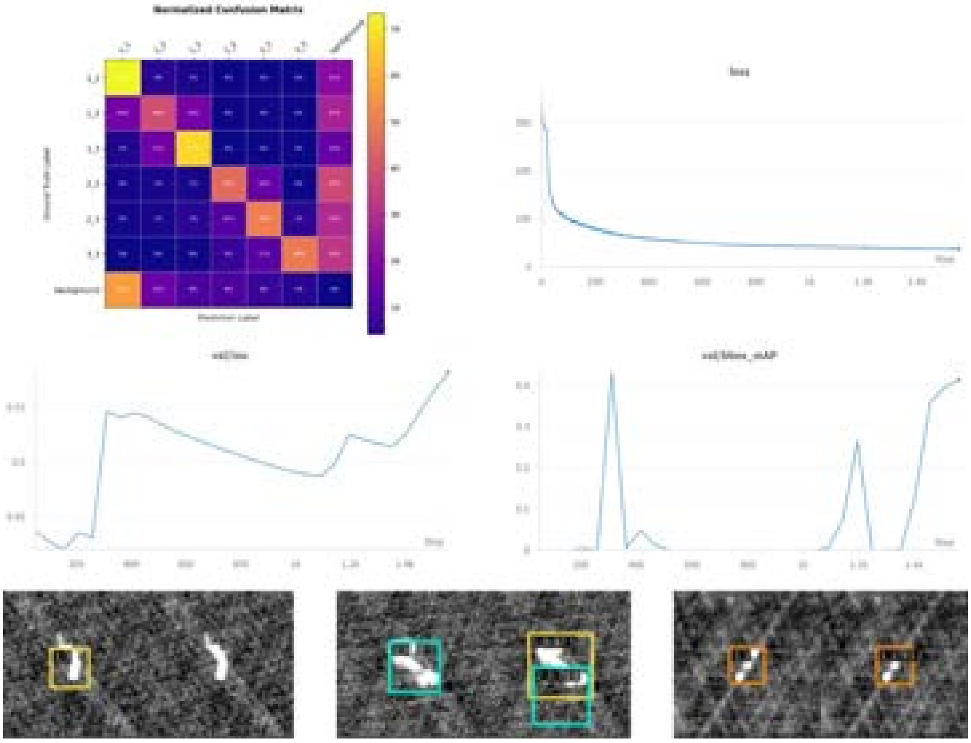


Figure A.21: YOLOv8 | Z-Scale | Not Pre-Trained | Performance Metric Summary

## A.3.2 Ensemble Experiments

### ResNet50 Six Sigma Deviation

Examples of samples which deviates more than six sigma from the distribution mean obtained by 30 ResNet50 models.

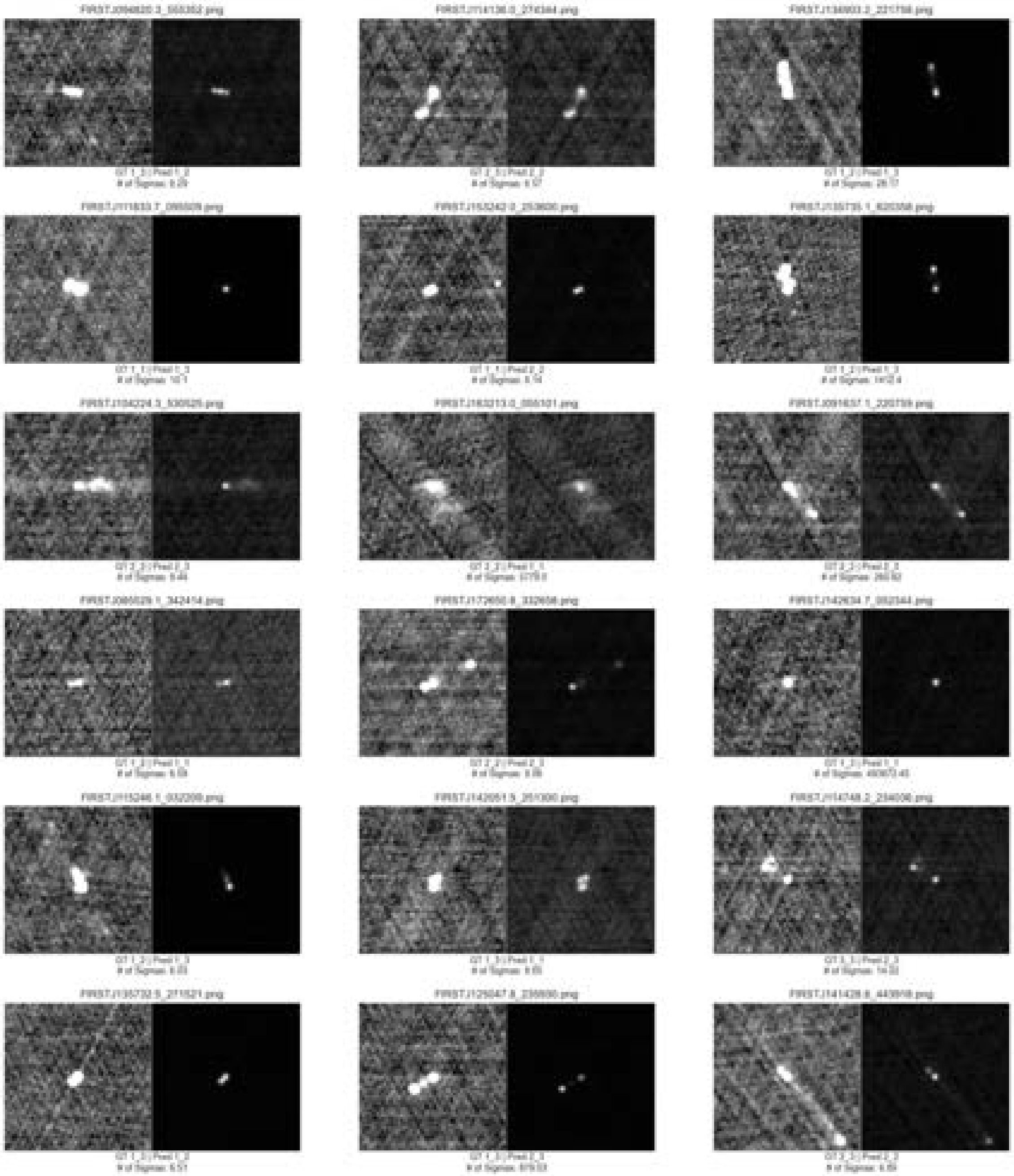


Figure A.22: Ensemble Six Sigma Deviation Results

### Full Error List ResNet 50 Ensemble

Examples of samples on which all 30 models failed to predict the ground truth class.

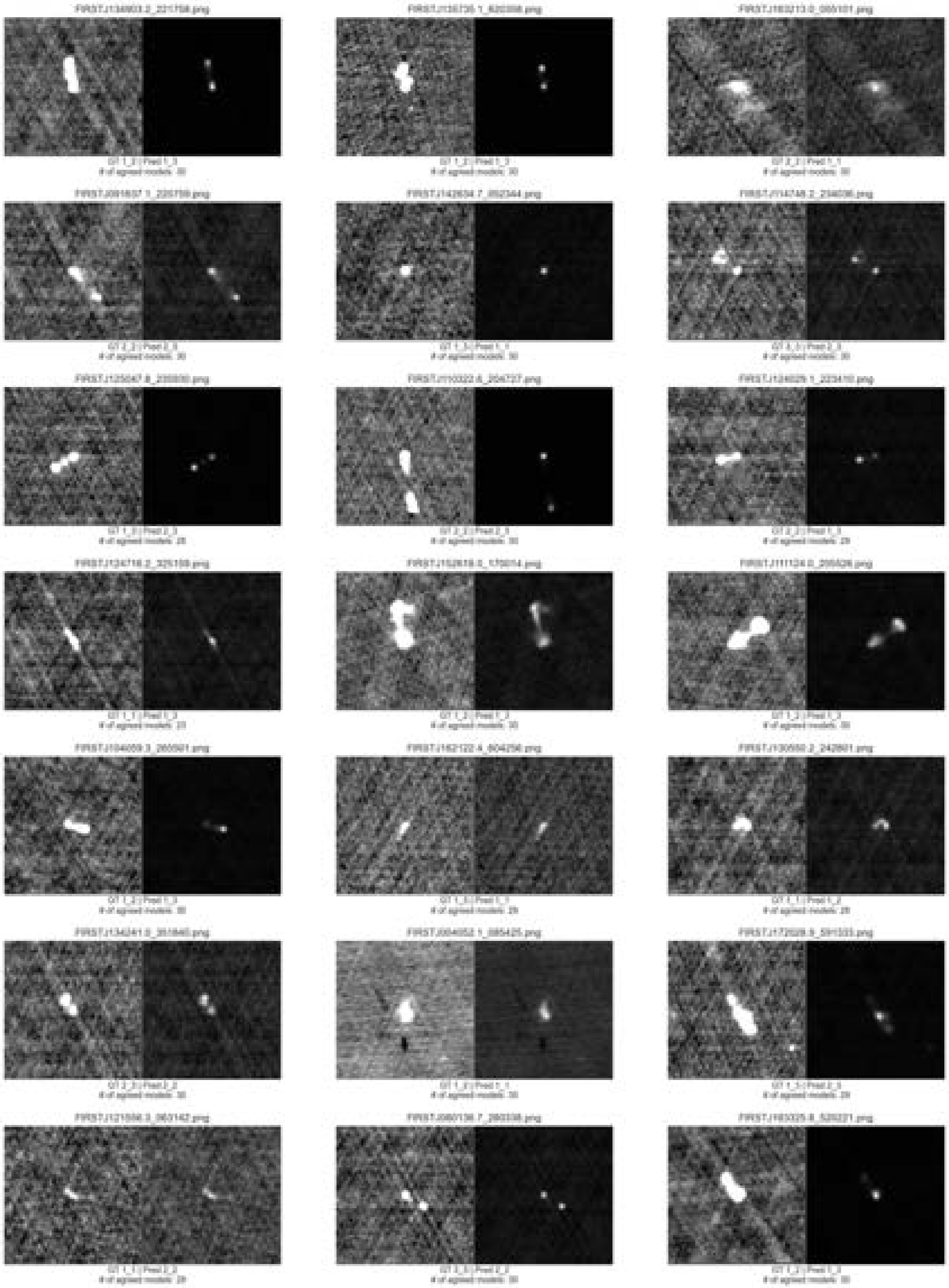


Figure A.23: Resnet50 Ensemble Full Error List

### Agreed Full Error List ResNet 50 Ensemble

Examples of samples on which all 30 models failed by predicting the identical wrong class.

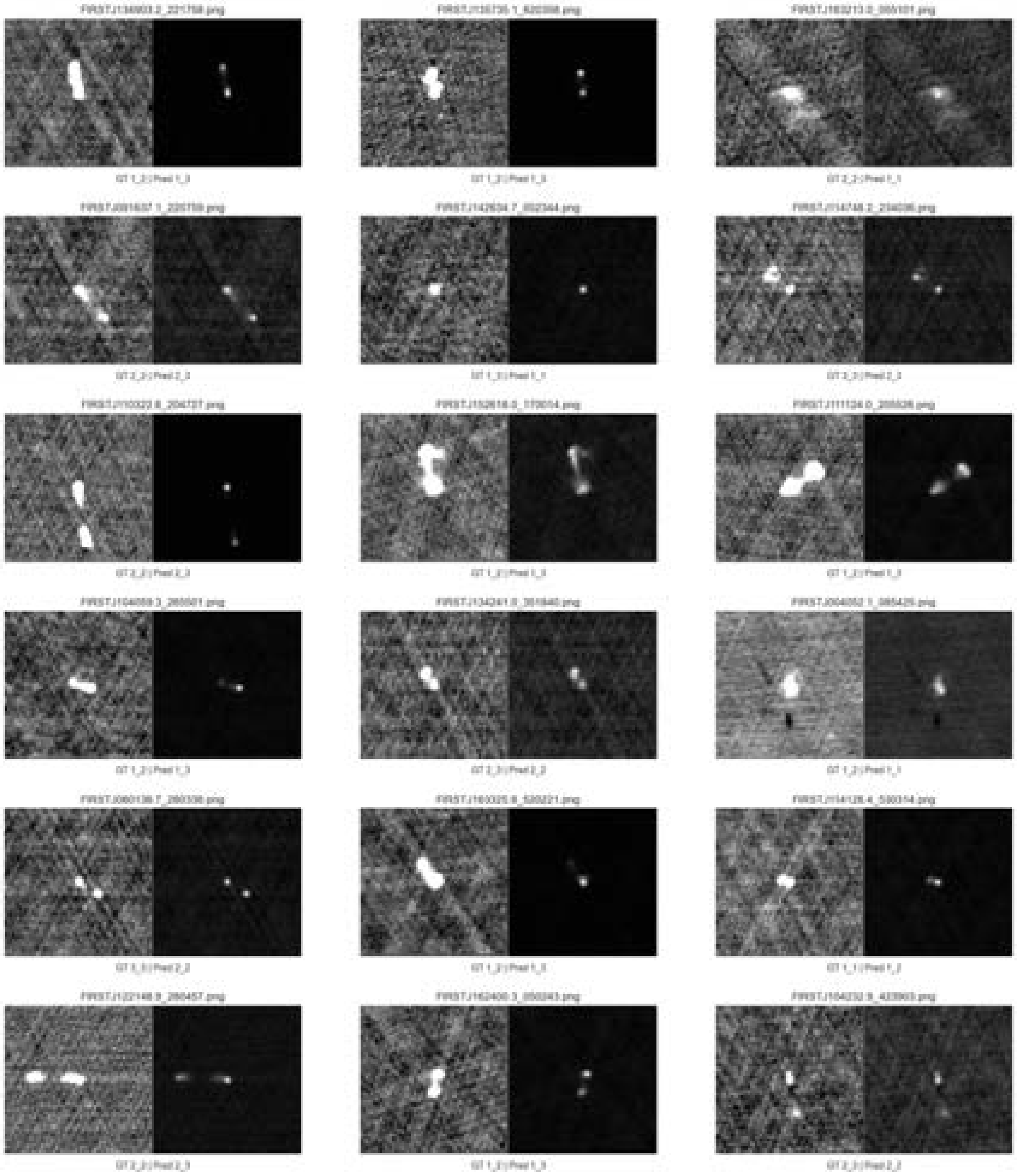

FIRSTJ134903.2_221758.png
GT 1_2 | Pred 1_3
FIRSTJ135735-1_620358.png
GT 1_2 | Pred 1_3
FIRSTJ163213.0_055101.png
GT 2_2 | Pred 1_1
FIRSTJ091637.1_220759.png
GT 2_2 | Pred 2_3
FIRSTJ142634.7_002344.png
GT 1_3 | Pred 1_1
FIRSTJ114748.2_234036.png
GT 2_3 | Pred 2_2
FIRSTJ110322.8_204727.png
GT 2_2 | Pred 2_3
FIRSTJ152618.0_170014.png
GT 1_2 | Pred 1_3
FIRSTJ111124.0_205526.png
GT 1_2 | Pred 1_3
FIRSTJ104059.3_265501.png
GT 1_2 | Pred 1_3
FIRSTJ134241.0_351840.png
GT 2_3 | Pred 2_2
FIRSTJ004052.1_085425.png
GT 1_2 | Pred 1_1
FIRSTJ080136.7_260336.png
GT 1_3 | Pred 2_2
FIRSTJ160325.8_520221.png
GT 1_2 | Pred 1_3
FIRSTJ114128.4_530314.png
GT 1_1 | Pred 1_2
FIRSTJ122148.9_260457.png
GT 2_2 | Pred 2_3
FIRSTJ162400.3_050243.png
GT 1_2 | Pred 1_3
FIRSTJ104232.9_423903.png
GT 2_3 | Pred 2_2

Figure A.24: Resnet50 Ensemble Agreed Full Error List

### A.3.3 Fine-Tuned Ensemble Experiments

#### Standard Deviation Analysis per Class

| **Classes** | **1_1** | **1_2** | **1_3** | **2_2** | **2_3** | **3_3** |
|---|---|---|---|---|---|---|
| Uncertainty | 0.0199 | 0.0293 | 0.0262 | 0.0242 | 0.0267 | 0.0148 |
| **Average Uncertainty** | **0.0466** | | | | | |

Table A.3: Uncertainty Fine-Tuned ResNet50 | ZMZ-Stack | Not Pre-trained | Per Class and on Average

#### Uncertainty Distribution Analysis

For this analysis the uncertainty was analysed in terms of number of standard deviations to the ground truth.

| **Sigma Interval** | **# of Samples** |
|---|---|
| [0, 1) | 341 |
| [1, 3) | 561 |
| [3, 6) | 49 |
| [6, ∞) | 36 |
| **Total** | **987** |

Table A.4: Uncertainty Distribution by the Number of Standard Deviations $\sigma$

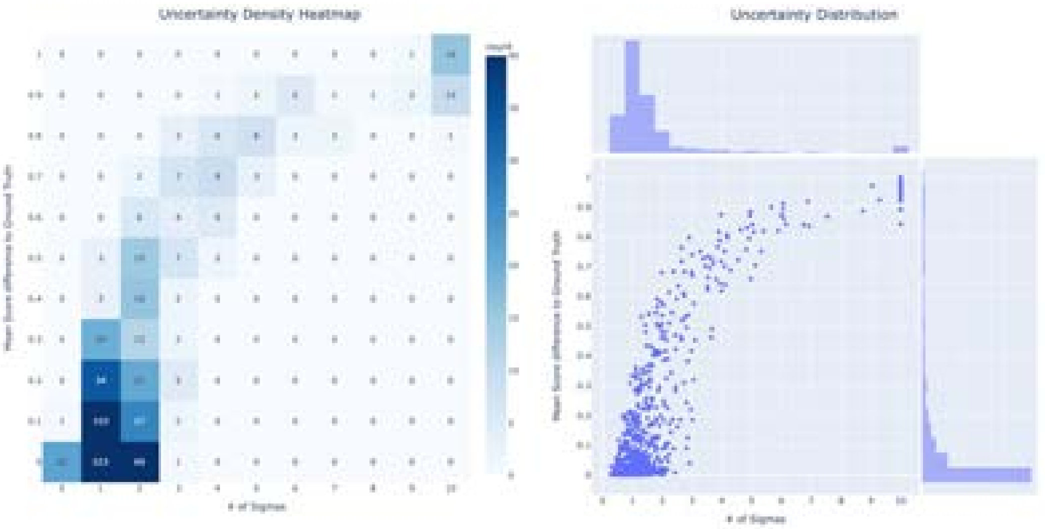
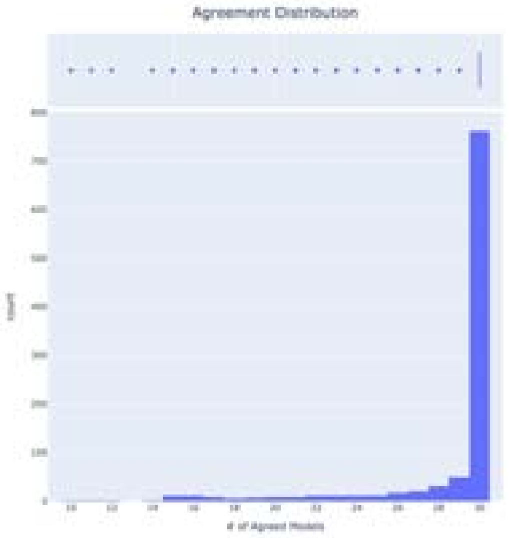


Figure A.25: Uncertainty Distribution by the Number of Standard Deviations $\sigma$

### Agreement Analysis

The following table shows the resulting metrics of the agreement analysis performed on 30 randomly initialised ResNet50 models by using the evaluated fine-tuned hyperparameters.



| Name | Value |
|---|---|
| Full Success Rate | 74.87% |
| Full Success | 739 Samples |
| Partial Success Rate | 22.19% |
| Partial Success | 219 Samples |
| Full Error Rate | 2.94% |
| Full Error | 29 Samples |
| Agreed Full Error Rate | 2.33 |
| Agreed Full Error | 23 |
| Majority Success Rate | 89.67% |
| Majority Success | 885 Samples |
| **Total** | **987 Samples** |

(a) Agreement Distribution (b) Quantitative Results of Agreement Analysis

Figure A.26: Agreement Analysis

### Fine-Tuned Ensemble Threshold Analyses

This analysis is performed to see the performance gains of using an ensemble of models from 1 to 30 models applied on the test set. Whereas in figure A.27 the top-1 accuracy was considered, analysed figure A.28 the top-2 accuracy. Moreover, the corresponding most important numbers are listed in table A.5 and A.6 respectively.

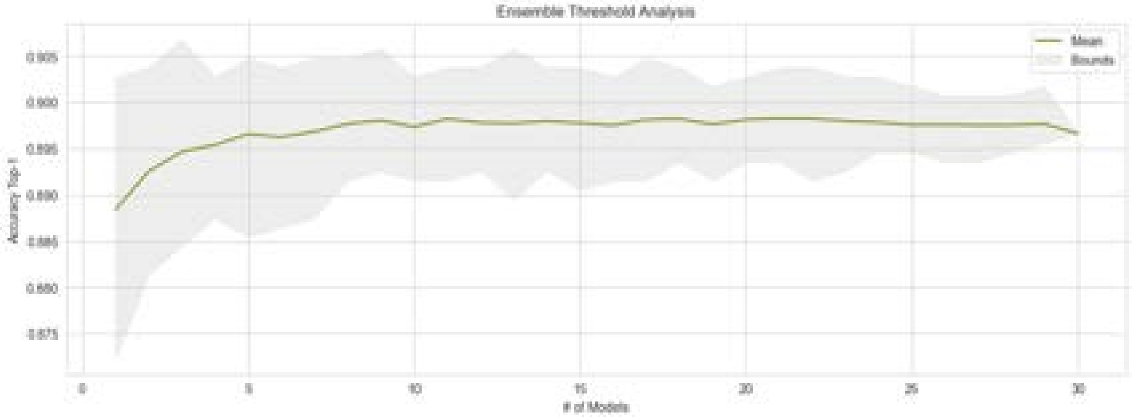


Figure A.27: Fine-Tuned Ensemble Size Analysis Top-1 Accuracy

| Key | Value |
|---|---|
| Maximum mean accuracy | 89.83% |
| Maximum mean accuracy at # of models | 21 |
| Maximum accuracy | 90.68% |
| Maximum accuracy at # of models | 3 |
| Lowest mean accuracy | 88.84% |
| Lowest mean accuracy at # of models | 1 |
| Lowest accuracy | 87.23% |
| Lowest accuracy at # of models | 1 |

Table A.5: Fine-Tuned Ensemble Size Analysis Top-1 Accuracy

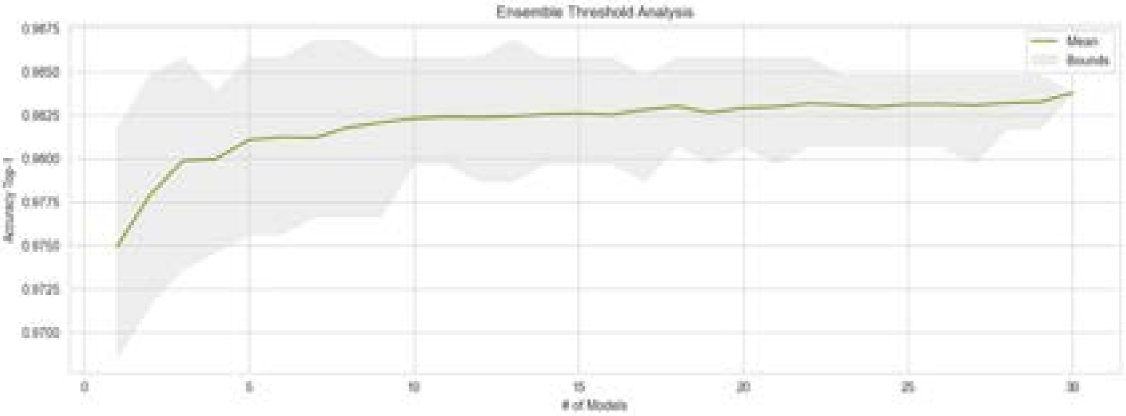


Figure A.28: Fine-Tuned Ensemble Size Analysis Top-2 Accuracy

| Key | Value |
|---|---|
| Maximum mean accuracy | 98.38% |
| Maximum mean accuracy at # of models | 30 |
| Maximum accuracy | 98.68% |
| Maximum accuracy at # of models | 7 |
| Lowest mean accuracy | 97.49% |
| Lowest mean accuracy at # of models | 1 |
| Lowest accuracy | 96.86% |
| Lowest accuracy at # of models | 1 |
| Maximum accuracy with three models | 98.58% |

Table A.6: Fine-Tuned Ensemble Size Analysis Top-2 Accuracy

### Fine-Tuned ResNet50 Six Sigma Deviation

Examples of samples which deviates more than six sigma from the distribution mean obtained by 30 fine-tuned ResNet50 models.

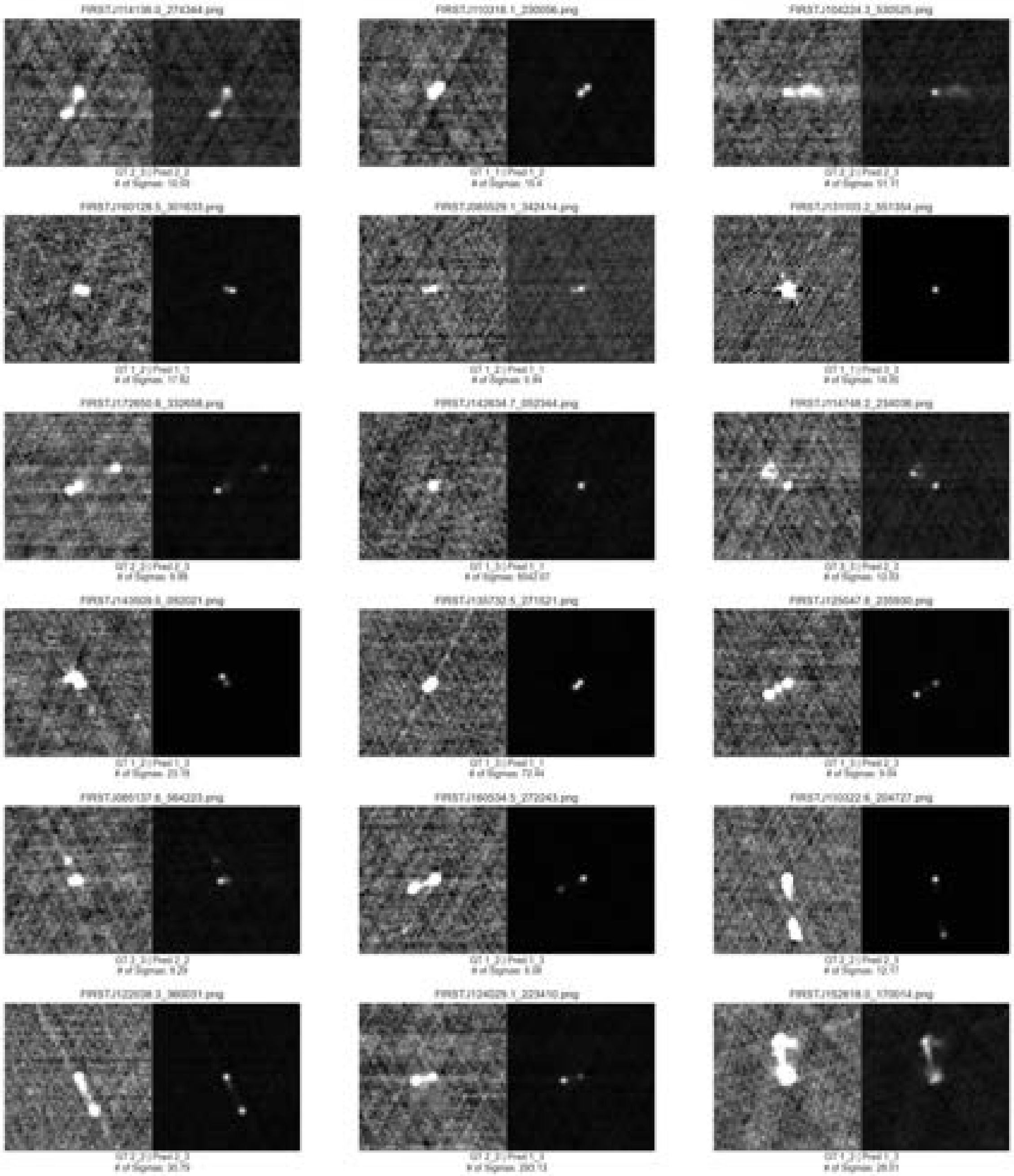


Figure A.29: Fine-Tuned ResNet50 Ensemble Six Sigma Deviation Results

### Fine-Tuned ResNet50 Ensemble Full Error List

Examples of samples on which all 30 models failed to predict the ground truth class.

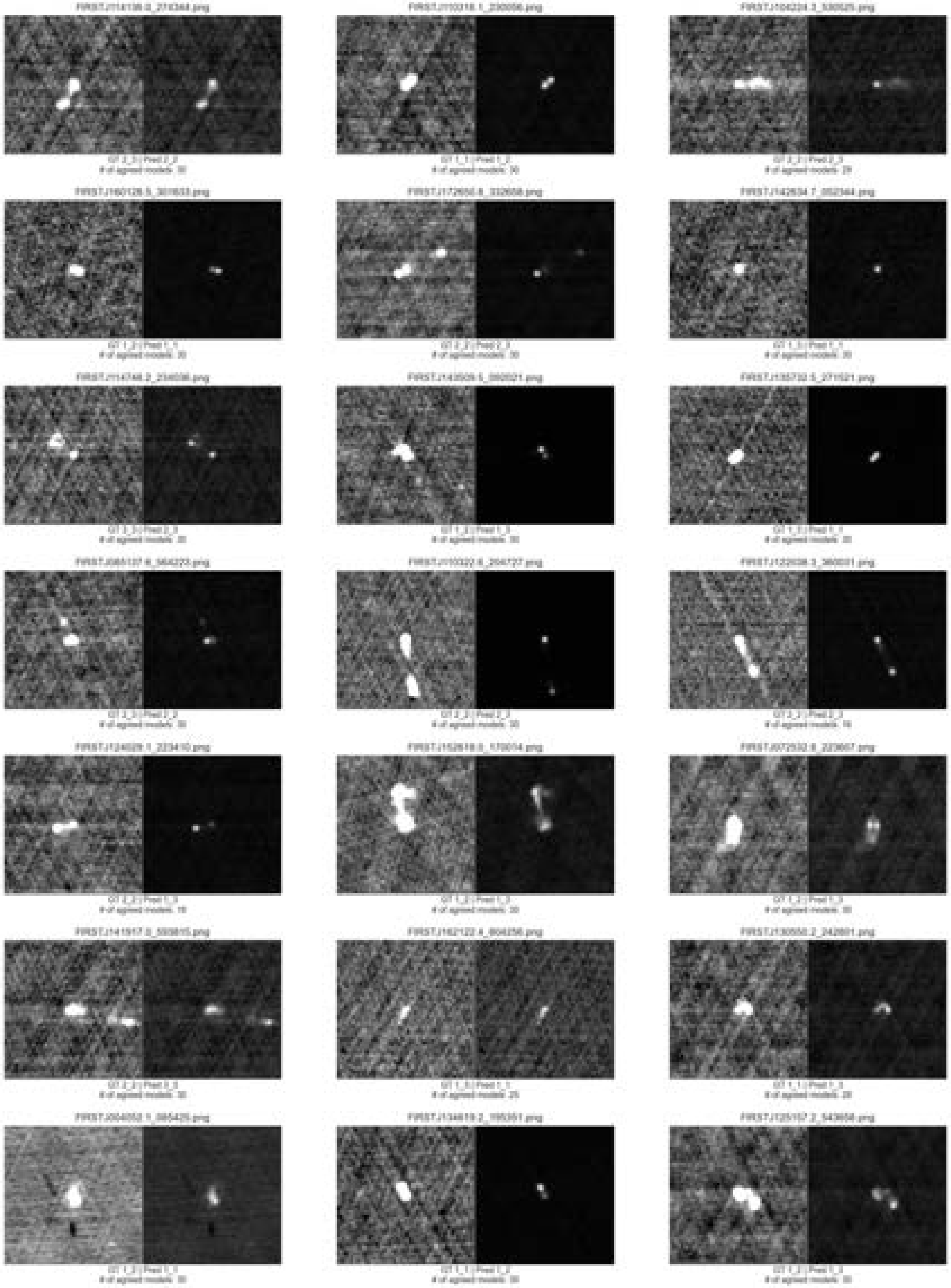


Figure A.30: Fine-Tuned ResNet50 Ensemble Full Error List

### Fine-Tuned ResNet50 Ensemble Agreed Full Error List

Examples of samples on which all 30 models failed by predicting the identical wrong class.

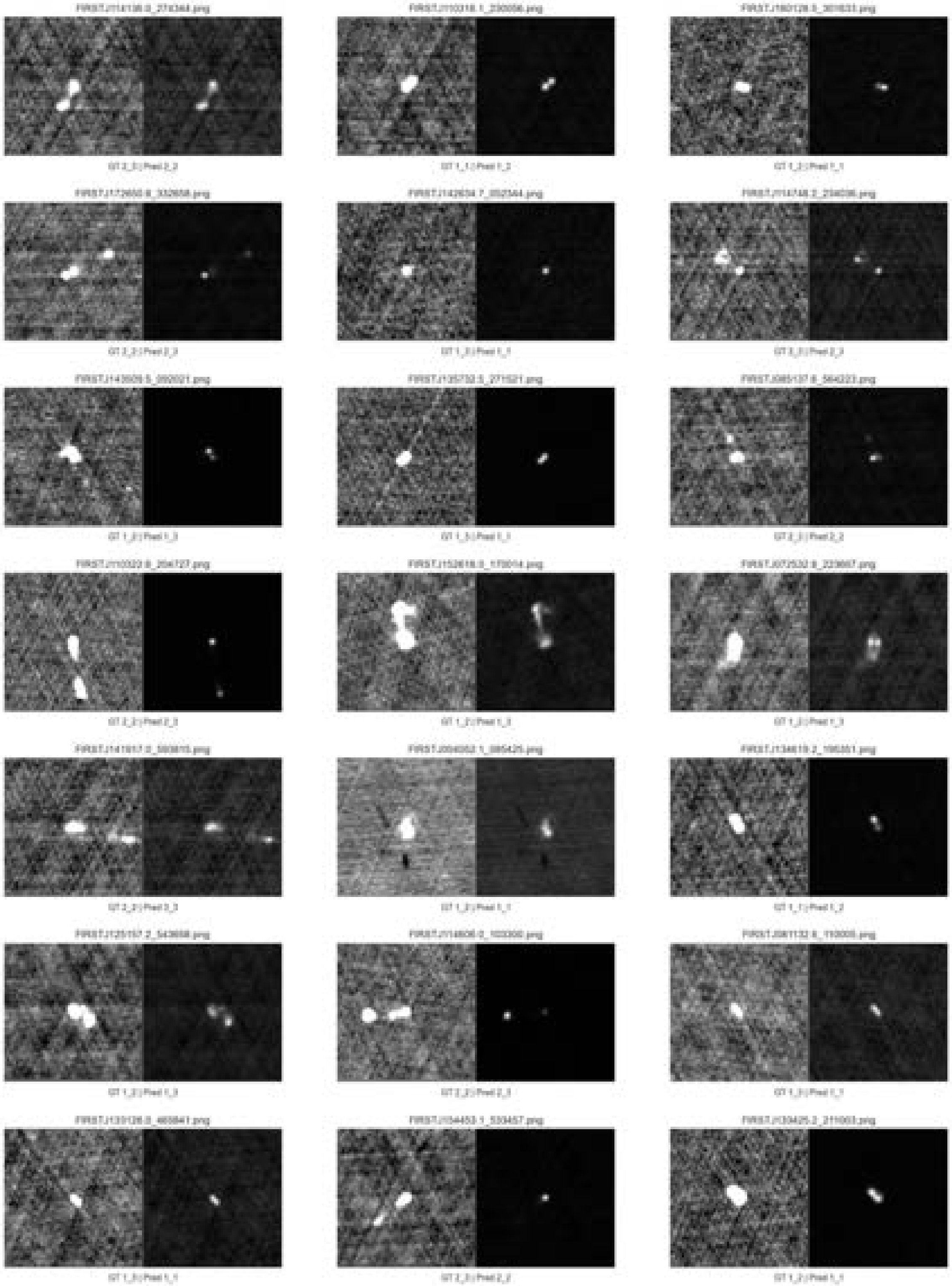


Figure A.31: Fine-Tuned ResNet50 Ensemble Agreed Full Error List

### A.3.4 GLEAM Experiments

The following image shows an example of a GLEAM source image of which six samples were manually cropped for further analyses with an ensemble of $30$ randomly initialised ResNet50 models trained on the RGZ OD dataset.

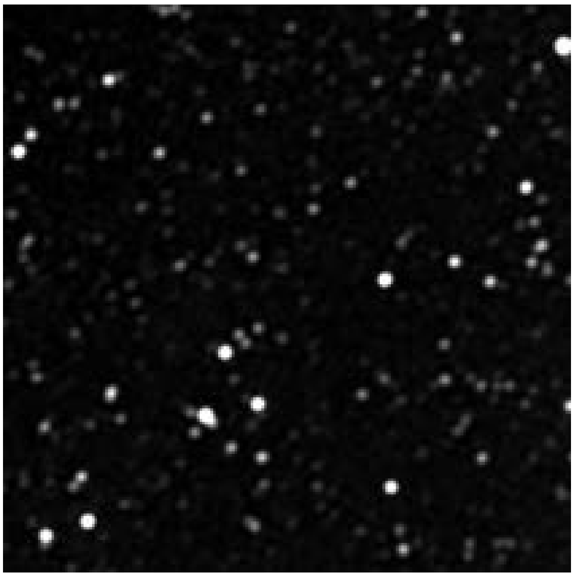


Figure A.32: GLEAM[68] source image