University of Groningen,
the Netherlands

Master's Thesis
Artificial Intelligence

Training Deep Neural Networks with Soft Loss for Strong Gravitational Lens Detection

Quintin van Lohuizen
s2701782

Internal Supervisor:
Dr. M. Wiering (Artificial Intelligence, University of Groningen)
External Supervisors:
Prof. Dr. L.V.E. Koopmans (Kapteyn Astronomical Institute, University of Groningen),
Dr. G.A. Verdoes Kleijn (Kapteyn Astronomical Institute, University of Groningen)

April 13, 2021

Abstract

This thesis explores a binary classification problem using convolutional neural networks (CNNs) applied to gravitational lensing. We try to find strong gravitational lenses in the Kilo-Degree Survey (KiDS) dataset, where we lack large labelled training sets and have noisy data. It can lack sufficient resolution and is severely imbalanced. Currently, astronomers spend a significant amount of time manually labelling images with a large portion of false positives. CNNs can reduce the strain on astronomers and speed up the process by rejecting obvious negative cases. This work investigates various loss functions and their influence on the trade-off between precision and recall. We consider the following loss functions: binary cross-entropy, F_1-, F_1 double- and F_β soft loss. Our results show that binary cross-entropy reaches the highest accuracy, while F_β reaches the highest precision and F_1 soft loss the highest recall. The loss function choice can influence how much time an astronomer would need manually labelling false positives. We used simulated lensing features to address a lack of real-world labelled data and merged them with real luminous red galaxies (LRGs). Merging lens and source enables us to understand the relation between misclassification and the lensing parameters' values. Our findings suggest that low brightness and size cause most false-negative cases and not other galaxies in the image's background. A Grad-CAM visualization algorithm highlights areas in the image containing evidence favouring a strong gravitational lens, enabling more insight into misclassifications. Finally, we perform experiments with a dynamic ensemble of CNNs, each trained on a different loss function. A separate CNN determines the weights attributed to each member based on the input image. This study has shown that training an ensemble with six members and the Nelder-Mead optimization algorithm can improve performance on high decision thresholds. An exciting prospect would be a newly created high purity candidate strong lenses set, based on the best performing ensemble with data from the upcoming Euclid and LSST mission.

Acknowledgement

I thank the supervisors Marco Wiering, Leon Koopmans and Gijs Verdoes Kleijn, for their support throughout the project and for being a great source of inspiration and knowledge. Next, I would like to thank Carlo Enrico Petrillo for laying the foundations of this research. A special thanks goes out to Christian Roest, a good friend and always listening to crazy new ideas and always being there (on Telegram). I would also like to thank Thijs van der Knaap, Remco van Buijtenen and Priscilla Mulder for their insights and daily walk in the park, which were a great relief throughout this endeavour. I also want to acknowledge the importance of the Center for Information Technology of the University of Groningen to provide access to the Peregrine high performance computing cluster. Without their resources, these large scale computational experiments would not be possible. A final thank you goes out to my parents, for providing motivation and financial support throughout my studies.

Contents

1 Introduction **1**

1.1 Gravitational Lensing . . . 2

1.2 Deep Learning . . . 4

1.3 Research Questions . . . 4

1.4 Thesis Structure . . . 5

2 Strong Gravitational Lensing **6**

2.1 Background . . . 6

2.2 Strong Lensing Applications . . . 7

3 Neural Networks **8**

3.1 Convolutional Neural Networks . . . 9

3.2 Residual Neural Networks . . . 9

3.3 Regularization . . . 11

3.3.1 Early Stopping . . . 11

3.4 Loss Functions . . . 12

3.4.1 Binary Cross-Entropy . . . 12

3.4.2 F_1 Soft Loss . . . 12

3.4.3 F_1 Double Soft Loss . . . 15

3.4.4 F_β Soft Loss . . . 15

3.5 Model Selection . . . 16

3.5.1 F_β Metric . . . 16

3.5.2 F_β Soft Metric . . . 17

3.6 Ensemble Learning . . . 17

3.6.1 Nelder-Mead - Weighted Average Ensemble . . . 18

3.6.2 Input Dependent Ensemble . . . 18

3.7 Gradient-Weighted Class Activation Mapping . . . 18

4 Experimental Setup **20**

4.1 The KiDS Survey . . . 20

4.2 Preprocessing . . . 21

4.2.1 Data Augmentation . . . 23

4.3 Network Architecture . . . 23

4.4 Loss Functions . . . 24

4.5 Grad-CAM 24
4.6 Ensembles 25
4.7 Max-Tree Segmentation 26
4.8 Specifications 28
4.8.1 Hardware 28
4.8.2 Software 28

5 Results **29**
5.1 Loss Functions 29
5.2 Gradient-Weighted Class Activation Mapping 32
5.3 Information Content 34
5.4 Ensembles 42
5.5 Max-Tree Segmentation 46
5.6 Discussion 47

6 Conclusion **49**
6.1 Summary of Results 49
6.2 Recommendations for Future Research 50

7 Appendix A **57**
7.1 Model Predictions 57
7.2 Gradient-Weighted Class Activation Mapping 57
7.3 Information Content 57

List of Figures

1.1 Galaxy Morphology Classification Example 1
1.2 Three Known Lenses 3

2.1 Strong Gravitational Lens Example 7

3.1 CNN Schematic Overview 10
3.2 Residual Block 11

4.1 Mock Lens Superimposition 20
4.2 Lens Contaminants (Negative Examples) 22
4.3 ResNet18 Architecture 23

5.1 Loss Function Experiment Results 30
5.2 Prediction Distributions 31
5.3 Grad-CAM True Positive 11 33
5.4 Grad-CAM True Negative 5 34
5.5 Grad-CAM True Negative 4 35
5.6 Grad-CAM False Positive 143 36
5.7 Grad-CAM False negative 328 37
5.8 Grad-CAM False negative 339 38
5.9 Hexbin TPR - Binary Cross-Entropy 39
5.10 TPR vs. Source Information - Binary Cross-Entropy 39
5.11 Hexbin TPR - F_β Soft Loss 40
5.12 TPR vs. Source Information - F_β Soft Loss 40
5.13 Hexbin TPR - F1 Soft Loss 41
5.14 TPR vs. Source Information - F1 Soft Loss 41
5.15 F_β Graph for Ensemble NM_3 43
5.16 F_β Graph for Ensemble NM_6 44
5.17 F_β Graph for Ensemble IDE_3 44
5.18 F_β Graph for Ensemble IDE_6 45
5.19 Max-Tree Segmentation Before and After 46
5.20 F_β Graph Max-Tree 47

7.1 Model Predictions Binary Cross Entropy 57
7.2 Model Predictions F_β metric 58

7.3 Model Predictions F_β soft loss metric 58
7.4 Model predictions F_β soft loss as loss function 59
7.5 Model predictions F_1 soft loss 59
7.6 Model predictions F_1 double soft loss 60
7.7 Grad-CAM True Positive 17 60
7.8 Grad-CAM True Positive 18 61
7.9 Grad-CAM True Positive 93 62
7.10 Grad-CAM True Negative 11 63
7.11 Grad-CAM True Negative 141 64
7.12 Grad-CAM True Negative 28 65
7.13 Grad-CAM True Negative 51 66
7.14 Grad-CAM True Negative 550 67
7.15 Grad-CAM False Positive 22 68
7.16 Grad-CAM False Positive 313 69
7.17 Grad-CAM False Positive 59 70
7.18 Grad-CAM False negative 190 71
7.19 Grad-CAM False negative 215 72
7.20 Grad-CAM False negative 232 73
7.21 Grad-CAM False negative 324 74
7.22 RMS Above Noise vs Prediction 74
7.23 Model Prediction vs. Einstein Radius - Binary Cross Entropy . . 75
7.24 Model Prediction vs. Einstein Radius vs. Brightness - Binary Cross Entropy 75
7.25 RMS Above Noise vs Prediction 76
7.26 Model Prediction vs. Einstein Radius - F1 Soft Loss 76
7.27 Model Prediction vs. Einstein Radius vs. Brightness - F1 Soft Loss 77
7.28 RMS Above Noise vs Prediction 77
7.29 Model Prediction vs. Einstein Radius - F_β Soft Loss 78
7.30 Model Prediction vs. Einstein Radius vs. Brightness - F_β Soft Loss 78

List of Tables

3.1 Confusion Matrix 13

5.1 Loss Function Performance Results 30
5.2 Ensemble Results 45
5.3 Max-Tree Segmentation Result 46

Chapter 1

Introduction

Gravitational lenses can give us tremendous insight into the geometry and history of our universe. Traditionally astronomers looked up at the sky with their telescopes and found objects interesting enough to study. However, Moore's law and automated imaging techniques increased the observational data by multiple orders of magnitude, while the number of astronomers did not keep pace. This 'data avalanche' and the prospect of future, deep, high-resolution wide surveys necessitates fast algorithms and image processing techniques to process all of this data to keep pace. The increased resolution of imaging techniques and the depth at which observational data can be taken [1] has taken astronomers down a road where they need help from computer science, computer vision, and artificial intelligence (AI) to make sense of this 'big data'.

Astronomical imaging techniques mostly aid morphology and spectroscopy studies [2, 3, 4] of objects of interests, such as planets, stars, galaxies, galaxy clusters and the overall structure of our universe. In an exciting study [5], a Convolutional Neural Network (CNN) predicts the class of a galaxy from the Galaxy Zoo 2 dataset (Fig. 1.1), based on its morphology.

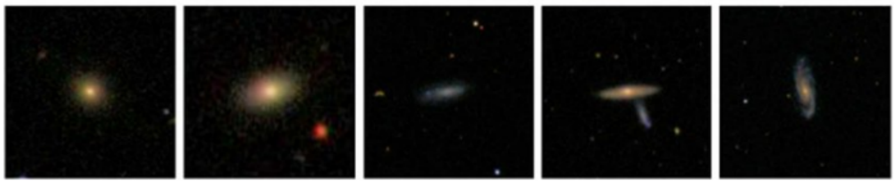


Figure 1.1: Example galaxy images from the Galaxy Zoo 2 dataset (28790 images in total). Each column represents a class. Their respective labels are: completely round smooth, in-between smooth, cigar-shaped smooth, edge-on and spiral. These examples illustrate the kind of features that can be captured from astronomical data and extracted through CNNs.

From studies such as [5, 2, 3, 4], astronomers can infer astrophysical understanding by classifying each galaxy into a predefined category, later to be used to model their physics and relate that to their observational morphology and spectra.

Traditionally astronomers relied upon instruments making specific measurements and constructing features from them. However, the recent developments in AI can automate feature extraction and construction and apply astronomical expert knowledge encoded in algorithms in a more automated fashion.

The aforementioned galaxy morphology classification study is an example of a multi-class classification algorithm in the field of deep learning (DL). DL is a subfield of machine learning (ML) concerned with a broader topic, where vastly different algorithms learn from data. DL has experienced rapid advances in the past years. The invention of CNNs [6, 7] and Residual Neural Networks (ResNets) [8] are of particular interest for image-based classification and morphology-based gravitational lens detection studies. Deployment of such algorithms can aid astronomers in combing through all the deep, high-resolution wide surveys existing now and those that will come in the future. CNNs are of great importance for image classification due to their architecture and spatial feature construction properties.

These deep neural networks (DNNs) have proven to outperform experts in many domains and raise the bar for algorithms on problem-solving in general [9], such as object detection [10] and playing games [11].

1.1 Gravitational Lensing

A gravitational lens is a visual effect observed from the Earth if the light from a distant source is deflected by a high-mass distribution between the source and observer. From Earth, we observe the source as being distorted and displaced from its actual location. To understand this effect, we must go back to Einstein, the theory of General Relativity (GR) [12]. Which essentially states that mass deforms the fabric of space-time in the vicinity of mass, giving space-time a curvature around high-mass objects. Galaxies or galaxy clusters have a high-mass distribution, creating an ideal region in space-time to observe these effects if a source is located behind it. The observed lensing features depend entirely upon the physical parameters of the source, deflector and observer. Fig. 1.2 depicts three examples of strong lenses. A significant challenge to find these lenses is the rarity of the event. We can look at the dataset used by [2], where non-lenses outnumber lens-galaxies, approximately a thousand to one. This sample dataset is constructed because the galaxies in this sample are more likely than other galaxies to be part of a gravitational lensing system. The number is likely to be higher in other datasets, making it even more challenging to find them. There are three classes of gravitational lenses: micro-, weak- and strong lensing. The distinction between the three depends on the source's relative location, lens, observer, and mass of the lensing system. In *micro* lensing, the magnification can be considerable. However, the event's angular scale is much smaller than galaxy-scale lenses, therefore mostly unresolvable by current telescopes. In contrast, *weak* lensing distorts the source object slightly, which can be measured by averaging over many sources. Strong lensing has additional discriminative features beyond magnification and displacement, such as doubling, quadrupling and arching effects (Fig. 1.2).

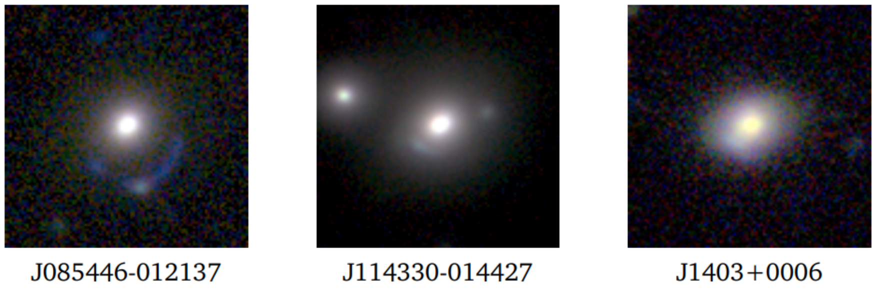


Figure 1.2: These are RGB reconstructed images, from GRI colored image data. These are three known strong gravitational lenses in the luminous red galaxy sample dataset. These are 20 by 20 arcsec images.

An essential application of strong lensing is to help put constraints on the universe's expansion rate [2]. In [13], lensed quasars were used to measure time delays between images of the same lensed quasar. Determining an accurate measurement of the lensing geometry means that an accurate estimation of the Hubble Constant (H_0) is possible. It refers to the speed by which objects seem to recede away from us, based on their distance from us. Differing methodologies and assumptions in measurement techniques lead to incompatible estimations of H_0 [14]. In [15], the value of H_0 is determined by measuring three lensed quasars' geometry. However, all three measurements indicated a different Hubble constant. It might be the case that it may depend on location, time and the direction we look at in the universe. This study shows the importance of sufficiently large datasets containing strong lensing systems to decrease model fitting uncertainties. Strong lensing can aid astronomers in studying very distant galaxies. They can act as galaxy scale telescopes, enabling astronomers to peek beyond the current imagery resolution and into a distant past of the universe [16, 17, 18].

Throughout the years, strong lenses have been found serendipitously or by visual inspecting surveys [2]. Finding them depends on the specific lensing configurations and situations. The process of confirming a strong lens has been to follow up a manual visual inspection by spectroscopy. It discerns between the light of the source and lens by measuring various wavelengths. When finding strong lenses through a binary classifier, scaling up with more data leads to a significant portion of unwanted false positives due to the severe data imbalance. Each positively predicted strong lens needs to be visually inspected by experts in the field, which hinders finding strong lenses in an automated fashion. Additionally, some lenses images are hard to visually identify because they are noisy and have a relatively low resolution. A lack of information in the data could limit the correct classification of strong lenses. A candidate strong lenses set produced by a ResNet in [3] had its true positives outnumbered by false positives by a factor of 40. Methods in Deep Learning (DL) could reduce this factor lower than 40 by taking the severe data imbalance into account.

1.2 Deep Learning

The field of Machine Learning (ML) is concerned with fitting a model onto data. It has a wide range of algorithms, with one of the most important ones, the artificial neural network (ANN). It is loosely based on biological neurons in order to facilitate learning for complex problems. Through successive layers of linear or non-linear transformations, the input data transforms into various representations. Many parameters define an ANN: input dimensionality, number of hidden layers, hidden layer neuron count, output dimensionality, activation functions, an optimizer, batch size and learning rate. The difference between ML and DL is the number of intermediate layers in an ANN and the input data's complexity. DNNs are typically bigger in order to properly deal with the high-dimensional input space. They are driven by the backpropagation algorithm that tries to minimize an *objective function* through gradient descent [6]. In most classification problems, the objective function is defined by a differentiable error rate function. A CNN is a hierarchical feature detector, where features are constructed based on lower level features detected earlier in the network. Each layer applies its activation function, which can be linear or non-linear, resulting in transforming the input.

The field of DL is concerned with complex model fitting onto data to achieve good generalization error. CNNs are a recent invention [7] aimed at rotational-, translational- and scale-invariant spatial feature extraction. Successfully training a DNN consists of hyper-parameter tuning and rigorous testing when training terminates. Since the discovery of the CNN, many architectures have been proposed that perform well on a wide range of tasks, such as Residual Neural- [8], VGG- [19] and Inception networks [20]. CNNs are of particular interest to strong lens detection due to the data being spatial. CNNs utilize the spatial characteristics of the features in the data hierarchically for inference. The field of DL is accelerated by open-source machine- and deep learning libraries, such as Keras [21], TensorFlow [22], Pytorch [23], and the emergence of Graphics Processing Unit (GPU).

1.3 Research Questions

In this thesis, we investigate how we can improve detection rates of DNNs concerning strong gravitational lens detection. Four existing loss functions will be considered and compared against each other based on the F_1 score, F_β score, and accuracy. The following research questions are investigated:

- **Which loss functions perform best with respect to precision and/or recall?** Performance is measured in terms of F_β score, which can be used to adjust the importance of precision relative to recall. This performance metric is meaningful in heavily imbalanced classification problems. The under-representation of strong lenses requires careful tuning of a models prediction threshold and the resulting true positive-, and false-positive rate. We will detail the consequences of loss function choice and selection metric during training, and its implications on dataset purity, if applied to real data.

- **Which lensing parameters explain the performance of the DNNs?** Here we visualize *where* a DNN looks while making an inference. We use partially simulated data because there is not enough real data for DL methods. Therefore we know what kind of data goes into the DNN and its information content. A simulated source will be merged with a real lens to create a mock lens. Consequently, we could calculate how much information the DNNs need to make a reliable inference.

- **Does an ensemble of DNNs perform better than individual DNNs, if members are trained on various loss functions or selected for different performance metrics?** Here we compare the performance of an ensemble of DNNs to the performance of individual DNNs. Here we also test the performance of weighted ensemble methods, where each member has high performance on either accuracy, precision, recall or F_β score.

- **Does image segmentation result in better classification performance?** For this question, a Max-Tree image segmentation algorithm is used and compared to a baseline according to accuracy and training time.

1.4 Thesis Structure

The thesis structure is as follows: Chapter 2 will discuss the theoretical background of strong gravitational lensing and its applications and prospects. Chapter 3 will explain the theoretical background of neural networks and their application to strong gravitational lens detection. Chapter 4 describes the experimental setup and explains the data. Chapter 5 depicts the results followed up by a discussion. Lastly, Chapter 6 concludes this thesis, presenting a discussion of the prospects of the new methods applied to the Euclid mission and ending with a recommendation for future research.

Chapter 2

Strong Gravitational Lensing

2.1 Background

Einstein found out with his theory of General Relativity (GR) that gravity is not a force, but a deformation of space-time itself, caused by the energy-density situated in it [12]. This theory changed our perspective of space-time drastically, from Newtonian physics to GR. In Newtonian physics, space is viewed as a static object that could be assigned coordinates and where time flows at the same rate at each of these coordinates [24]. GR tells us that even photons, which consist of energy with no rest-mass, can affect space-time geometry. Because they can affect space-time, the inverse holds that space-time affects it. Therefore high mass objects such as galaxies cause photons to follow a curved space-time around them. An illustrative example of this is a massive galaxy cluster. Therefore they deform space-time around it significantly. An approaching photon follows a straight line; however, this path becomes curved in the vicinity of mass. From the photon's perspective, it has always travelled in a straight line and continuous to do so. However, according to an outside observer, the photon describes a curved path around the galaxy cluster. The difference in perspective between the object and the outside observer is essential to understanding GR.

Another example of GR is light deflected by a considerable mass, such as a giant galaxy or cluster of galaxies. Such a case is called gravitational lensing due to the visual similarity with optical systems. The deflected light comes from an entirely different source than the deflector (lens system), for example, another galaxy. The visual imagery depends on the alignment of the telescope, lensing-, and source object. Gravitational lensing can produce single, multiple, magnified and distorted images of the source object (See Fig. 2.1). In this thesis, we will focus on galaxy-scale strong gravitational lenses only.

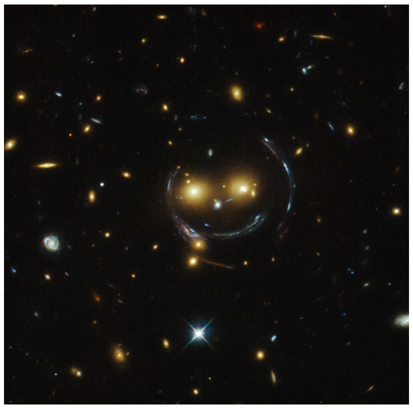


Figure 2.1: These are two luminous galaxies distorting the light from a distant blue galaxy, creating extended blue light arcs. This system is called the smiling lens due, to its similarity to a smiling face. (Image credit: ESA/Hubble & Nasa).

2.2 Strong Lensing Applications

Finding strong lenses is essential because they are a valuable tool for many scientific objectives. The lensing magnification effect can shed light on otherwise unobservable structures below the current imagery resolution. Strong lenses are particularly interesting for studying high redshift galaxies and their substructures. These lenses are used in cosmology, where we try to understand the universe's expansion rate. Estimating the value of the cosmological Hubble Constant (H_0) is possible through strong lensing studies. They can aid in H_0 estimation by providing multiple images of the same object travelling through different geometric paths through the universe. It provides a faithful reconstruction of the lensing system, allowing a reasonable estimation of H_0. Another example of the application of strong lenses was illustrated by [25], where 58 strong lenses were used to estimate the density slope inside one effective radius of massive early-type galaxies. Finding strong lenses can aid studies into galaxy formation and studies about galaxy evolution through cosmic time. Strong lenses can probe dark-matter substructures of distant galaxies [26, 27]. Studying dark-matter substructures of high redshift galaxies can aid our understanding of dark-matter distribution throughout the universe.

Finally, it is common in astronomy to determine the distance of a galaxy relative to Earth by measuring redshift z. When light travels through an expanding space-time, the wavelength of photons get stretched. If light travels through space-time for an extended time, the wavelength of light stretches towards redder colours, hence the name redshift. By measuring the redshift, we can calculate how long the light has travelled and the distance of an object that we are observing, where a larger redshift also implies a longer look-back time.

Chapter 3

Neural Networks

One of the simplest forms of a neural network (NN), which we refer to as a *model*, is a multilayer perceptron (MLP). It is an important, feedforward ML model since it is a standard building block of many modern architectures, such as Residual Neural Networks (ResNets). To train an MLP in a *supervised learning* fashion, we need a labelled dataset, a *loss function* and an *optimizer*. The labelled dataset is run through the MLP in an iterative training process of stacked non-linear transformations from input data to the predicted output label. Through this process, the MLP defines a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of all learnable parameters, $\mathbf{x}$ the input data and $\mathbf{y}$ the output prediction [28]. An MLP is a *universal function* approximator due to non-linear activation functions, such as the Relu and Sigmoid (Eq. 3.1 and Eq. 3.2).

$$\sigma(x) = \max(0, x) \tag{3.1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

The *forward pass*, the computation of the *error* using the loss function and the *backward pass* with backpropagation [29] are three essential training steps.

Input data is transformed at each layer via activation functions, such as *Relu* or *Sigmoid*, during the forward pass. Each layer has its own set of learnable parameters, referred to as *learned features*.

The second step needed to complete a training iteration is the computation of the loss over a given sample or batch. The error for each input sample is computed according to a loss function, for example, $e = y - \hat{y}$, where y is the label and $\hat{y}$ the predicted label. However, training a NN is usually done in *batches* or *mini-batches* to ensure better training stability. The loss function needs to change accordingly, to calculate the loss over a given batch of data, for example, the Mean Square Error: $mse = \sqrt{(\frac{1}{n}) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$, where n is the number of samples in the batch. Essential to a loss function is the ability to be differentiable with respect to $\boldsymbol{\theta}$. Otherwise, the backpropagation step is not possible. There are many types of loss functions, such as *binary cross-entropy*, *categorical cross-entropy* or F_1-soft loss, all with their advantages and disadvantages for various problems.

To conclude one iteration of the training process, we need to perform the backpropagation step [29]. To learn parameters $\boldsymbol{\theta}$, gradients need to be calculated for the whole batch. The gradient calculation over a random subset of data instead of calculating the gradient over the entire dataset makes the algorithm stochastic and reduces the computational cost. The following equation defines stochastic gradient descent:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L_n(\boldsymbol{\theta}) \quad \text{for } i = 1...n \tag{3.3}$$

The $L_n(\boldsymbol{\theta})$ term refers to the loss function on the weight vector $\boldsymbol{\theta}$, n the number of batches, η the *learning rate*. The newly calculated weight vectors for each layer can now be used for the next training iteration of stochastic gradient descent. Training NNs can take a long time, and the search for optimal hyperparameters can be cumbersome. However, they are an essential part of ML and DL. The recent developments in GPU training acceleration have helped the field speed of scientific progress drastically. A GPU can accelerate training by multiple orders of magnitude compared to using a CPU for training.

3.1 Convolutional Neural Networks

After discovering CNNs, image recognition tasks have considerably higher performance than before [30]. Due to the hierarchical stacking of visual features in images, complex patterns can be learned and classified. Low-level visual features such as edge detectors are successively stacked into higher-level features. A CNN consists of an input layer equal to the visual data dimensions, a *convolutional base*, and a classifier see Fig. 3.1, for a schematic overview.

The convolutional base learns visual features relevant to the classification problem. These layers usually consist of convolutional layers for feature detection and max-pooling layers for dimension reduction.

The bottom of a NN is the input, while the top is the output. The last dense layers on top of the convolutional base are the classifier and resemble a simple MLP. The top classifier considers all detected features from the convolutional base and predicts the input data class label based on these features. The MLP is most related to the problem at hand because the dimensionality of the output layer directly relates to the number of classes in the classification problem.

3.2 Residual Neural Networks

A ResNet is architecturally similar to a CNN except for skip connections, see Fig. 3.2. DNNs suffer from the vanishing gradient problem [31, 32], where lower layers in the network barely receive any gradient updates. The gradient

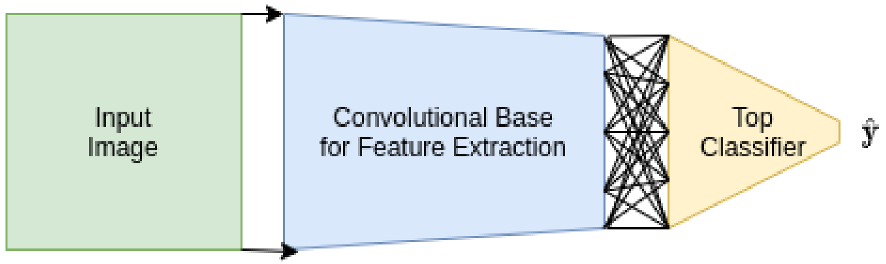


Figure 3.1: There is an input image on the left, with a convolutional base in the middle that performs features extraction and a top classifier (MLP) on the right.

decreases iteratively with each layer, reaching the computer's numerical precision and effectively becomes zero. A ResNet addresses this problem by having higher layer activations passed down to lower layers directly, skipping intermediate layers. The gradient's direct flow through a skip connection ensures that some significant gradient update is passed down to the network's bottom layers during backpropagation. Skip connections combat the vanishing gradient problem because the application of activation functions are skipped. Activation functions such as the Sigmoid have flat regions in their function curve that contribute to the vanishing gradient problem. In recent years, deeper networks have reached higher performances than their shallower counterparts due to these skip connections [8].

Iteratively adding layers to a network has diminishing returns up until the point where the classifier's performance saturates and then drops rapidly [8, 33, 34]. Such a network has too much capacity for the problem at hand. It means that the number of learnable parameters is too great for the problem and the data volume. Therefore reducing the number of learnable parameters increases performance. If a network has learned good features in bottom layers and has too much capacity, it still needs to pass the acquired information through multiple top layers before a final prediction. These higher layers apply non-linear transformations to the representations of the bottom layers. However, the desired learned mapping function should be an identity mapping because the earlier layers have already learned good representations. Learning the identity mapping function is non-trivial and reduces a NN's performance because more training and data are needed to learn this function. Instead, the correctly learned representations from bottom layers should be passed through a skip connection. The identity mapping is inherent in a ResNet's architecture due to these skip connections and does not have to be learned by top layers. Adding layers with skip connections does not hurt performance but can only increase it with diminishing returns. Diminishing returns refers to each added layer adding some amount to the performance metric, but each successive layer less than the previous. In a network with too much capacity and skip connections, the top layers learn to apply the identity mapping via the skip connections without hurting performance [8].

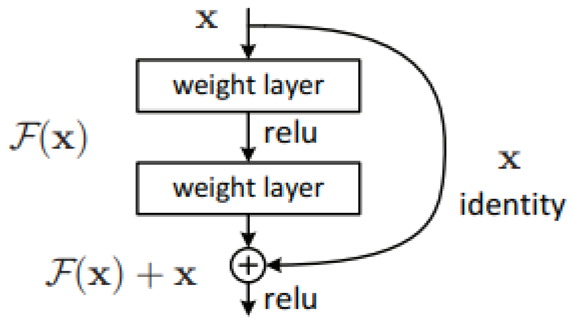


Figure 3.2: This is a Residual Block, where incoming activations **X** are passed through standard weight layers, such as convolutional- or max-pooling layers. Moreover, **X** is added to these layers' output via a skip connection using a vector/matrix addition. Because of the addition operation, the dimensionality of activations before and after the skip connection must equal. The activations running through the skip connection do not pass through any learnable parameters. In these networks, a typical downscaling factor is 2. Therefore a stride of 2 is sufficient to match the dimensions in case of dimension mismatch in a shortcut connection.

3.3 Regularization

Regularization concerns methods to keep a model from *overfitting*. If a model trains on training data too extensively, it will remember specific data instances. An overfitted model will perform worse on unseen data than a model that does not. To find a model that does not overfit, first find a model that does overfit. Once the *epoch* of overfitting is known, a model should be trained up until that epoch. An inherent feature of the ResNet architecture is the Batch Normalization layer [35], which standardize the inputs to a layer, reducing training time and aiding overall generalization.

3.3.1 Early Stopping

Before training, one must decide upon a maximum number of epochs needed for a NN to converge. Usually, one chooses a number too large and stops training based on the desired error rate or loss value. However, this choice is of less importance due to *early stopping*, where the algorithm stops trying to fit the data if the validation- and training-loss start to diverge. The early stopping parameter is usually called *patience* and signifies the number of epochs the algorithm waits until training is brought to a halt.

3.4 Loss Functions

In this section, we will go into more detail regarding loss functions. A loss function or *objective function* needs to be differentiable and minimizable. Loss functions have already been briefly explained at the beginning of Sec. 3. It defines how the NN will steer itself to a minimum value in weight space during optimization by measuring its performance. A loss function can be defined in the following way:

$$L(\mathbf{D};\boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{N} l(\mathbf{D}(i),\boldsymbol{\theta}) \tag{3.4}$$

It is an arbitrary average loss function over some data $\mathbf{D}$, given network parameters θ. During training, the loss function is optimized to be as low as possible on the training dataset. The function $l(\mathbf{D}(i),\boldsymbol{\theta}$ is the loss over a single data point.

3.4.1 Binary Cross-Entropy

If faced with a binary classification problem, many would rely on the loss function, binary cross-entropy. It measures the distance between two probability distributions, the positive and negative class. The following equation defines cross-entropy for two classes [36]:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i)\cdot\log(1-p(y_i)) \tag{3.5}$$

The y_i is the label of sample i, N the number of samples and $p(y_i)$ the probability of the sample being a positive class member.

3.4.2 F_1 Soft Loss

Finding strong gravitational lenses in a real galaxy sample is challenging due to a heavy data imbalance. Lens-galaxies are usually outnumbered by normal galaxies by a factor of one to a thousand [2]. The result of this imbalance is that any real galaxy set will have its false-positive rate (FPR) much greater than its true positive rate (TPR). A CNN trained in [2] has its sample set dominated by false positives (FPs) by a factor of ~ 40 when tested on a subset of KiDS-DR3. They suggested that it is critical to reducing the number of false positives due to the need for a manual visual inspection of the output sample set. We will address this problem by utilizing a different loss function than the standard binary cross-entropy for a binary classification problem. In this section, we will review a different loss function, namely F_1 *Soft Loss*.

In the field of ML, it is common to talk about the *precision* of a classifier. Precision is defined by the percentage of the predicted positives that are true positives; see Eq. 3.6. Precision is associated with a false alarm rate (false identification of a strong gravitational lens). The higher the number, the less 'false alarms' the classifier raises.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.6}$$

FP and TP refer to a raw count of FPs and TPs predicted by a classifier on some data.

The recall of a classifier can be thought of as the percentage of true positives that were predicted positive, see Eq. 3.7. It signifies the factor of true positives the classifier could recall from the data. A false negative (FN) is an example that would be predicted as positive in an ideal case; however, it is classified as a negative under the current decision-making threshold. Therefore we add the FN count in the recall formula because the model should have recalled this example.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.7}$$

To determine whether an example is classified correctly, we need to define a *decision-making threshold*, also referred to as p-threshold. The value of this threshold determines the value counts in the confusion matrix; see Tab. 3.1.

	Labelled Positive	Labelled Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Table 3.1: This is a Confusion Matrix [37] of a classifier under a decision-making threshold. The decision-making threshold defines the confusion matrix and, therefore, the F_1-score of a classifier.

Fine-tuning recall and precision is difficult because one must find a balance between the two metrics. Taking the harmonic mean of the two results in the following equation:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.8}$$

In this equation, precision and recall receive the same amount of importance because neither is favoured over the other. In an extreme case, where one would only consider recall to be significant, a classifier could learn to predict the positive case consistently. The recall formula would result in a value of precisely 1.0 to avoid rejecting any object of interest. This kind of behaviour is undesirable; therefore, it cannot be used as loss functions.

To incorporate the $F1$-score as a loss function for a classifier, we need to make the function differentiable. Instead of adding integer values to the TP count, we add fractional or also called *soft* values, using the following definitions:

$$\mathrm{TP} = \hat{y} \cdot y \tag{3.9}$$

$$\mathrm{FP} = \hat{y} \cdot (1 - y) \tag{3.10}$$

$$\mathrm{FN} = (1 - \hat{y}) \cdot y \tag{3.11}$$

$$\mathrm{TN} = (1 - \hat{y}) \cdot (1 - y) \tag{3.12}$$

y Refers to the label of the example and $\hat{y}$ to the predicted label by the classifier. Without these definitions, precision and recall are discrete numbers and do not have a derivative. These definitions make it possible to construct a loss function that considers precision and recall as continuous variables.

The difference between hard and soft labels [38] is that hard labels are binary in how membership to a class is assigned. The data point either belongs to the positive class or the negative class, either 1 or 0. A soft label assigns a probability of belonging to a class instead of assigning membership in a binary fashion.

For example, if a positive example gets the prediction 0.4, and the decision-making threshold is 0.5, the model was only off by 0.1 for correct classification. If a soft loss function is applied instead of a hard loss function, the model is not penalized too much for making an almost correct prediction.

The soft F_1-score is defined by the following equation [37] obtained by substituting Eq. 3.6 and Eq. 3.7 into Eq. 3.8 and adding a small fractional value for numerical stability:

$$F_{1Soft} = \frac{2TP}{2TP + FN + FP + 10^{-16}} \tag{3.13}$$

10^{-16} Is added to ensure that division by zero does not occur. In order to obtain a minimizable function, we need to adjust the F_{1Soft}-score function to a loss function for the positive class:

$$F^1_{1Soft} = 1 - (\frac{2TP}{2TP + FN + FP + 10^{-16}}) \tag{3.14}$$

When training NNs, it is common to train on batches or mini-batches. In such cases, the loss is calculated over a whole batch instead of on a single example. The batch TP, FP and FN, definitions would change only slightly:

- TP $= \hat{\mathbf{y}} \cdot \mathbf{y}$
- FP $= \hat{\mathbf{y}} \cdot (1 - \mathbf{y})$

- FN $= (1 - \hat{\mathbf{y}}) \cdot \mathbf{y}$
- TN $= (1 - \hat{\mathbf{y}}) \cdot (1 - \mathbf{y})$

Eqs. 3.13 and 3.14 do not change, except that we use vectors instead of scalars. The resulting soft F1 loss function has been turned into macro soft $F1$ loss due to the loss being calculated over a batch instead of a single example.

3.4.3 F_1 Double Soft Loss

The Soft-$F1$ Loss function only considers the loss from the positive class' perspective (being a gravitational lens). The (Macro) Soft-$F1$ Loss can be modified to incorporate the negative class' viewpoint (TN). Modifying Eq. 3.14 in order to incorporate this consists of replacing the TP with the TN values:

$$F^0_{1Soft} = 1 - (\frac{2TN}{2TN + FN + FP + 10^{-16}}) \tag{3.15}$$

TNs are negative examples that have been predicted negatively by the classifier. The following equation defines how to calculate the value of TN in a 'soft' fashion:

$$\text{TN} = (1 - \hat{y}) \cdot (1 - y) \tag{3.16}$$

Both Eq. 3.14 and Eq. 3.15 need to be combined into one differentiable loss function to be able to use minimization techniques. It is done by taking the mean loss value of both equations:

$$F_{1DoubleSoft} = \frac{F^0_{1Soft} + F^1_{1Soft}}{2} \tag{3.17}$$

To obtain one loss value over a batch of examples, we compute the mean Double Soft-$F1$ Loss and call it the *Macro Double Soft-$F1$ loss*.

3.4.4 F_β Soft Loss

In order to adjust the importance of recall relative to precision, the following equation is defined [39]:

$$F_\beta = (1 + \beta^2)\frac{\text{Pr} \cdot \text{Re}}{(\beta^2 \cdot \text{Pr}) + \text{Re}} \tag{3.18}$$

In this equation, precision and recall values are calculated with 'soft' values for TP, FP, and FN (Eq. 3.9, 3.10 and 3.11). It allows optimizing for precision more strongly rather than recall if β is lower than 1.0. Due to Eqs. 3.9, 3.10 and 3.11 being continuous, give the function a derivative. In order to make the function minimizable, we can subtract 1 with the F_β-score:

$$F_\beta = 1 - ((1 + \beta^2)\frac{\text{Pr} \cdot \text{Re}}{(\beta^2 \cdot \text{Pr}) + \text{Re}}) \tag{3.19}$$

3.5 Model Selection

A DNN is either trained in batches or in mini-batches over which multiple performance metrics can be calculated. Based on the value of the loss function, the network parameters are adjusted per batch. At the end of training on a single batch, one could save the current model parameters based on some performance metric, such as accuracy, precision, recall, binary cross-entropy, F_1-score, F_β-score, or some custom made performance metric for the problem at hand. The formula for accuracy for a binary classifier is defined as:

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{3.20}$$

The current model parameters are selected if the newly calculated performance metric on a batch of data is better than before. A metric can either be minimized or maximized during the selection process. For example, the validation loss needs to be minimized, while the F_1-score is a metric that should be maximized. The process of selecting a model from a set of models is called *model selection*.

3.5.1 F_β Metric

It might be a good idea to select models, during the training process, based on F_β-score instead of validation binary accuracy or validation loss.

The goodness of a model in terms of F_β-score can be defined as having a high area under the curve (AUC). However, this curve is not a continuous function. Therefore integration over the F_β function does not work. An approximation or indicator value of the AUC can be used to assess the performance of a model.

Given a batch of data, the precision and recall change with respect to the decision threshold p. To approximate the AUC for the F_β curve, we need to discretize the p-threshold range to obtain an F_β-score per decision threshold. Once this is performed, we can sum over the individual F_β values for each decision threshold and divide by the number of decision thresholds. The average F_β value of the F_β curve is an indicator of its AUC; the higher, the better the model performs concerning F_β.

The justification of β having the value 0.17 comes from [39]. In this comparative study, many different binary classifiers were tested against each other using the F_β metric.

Eq. 3.18 must be calculated for each decision threshold, where the number of thresholds is the number of discrete and equally spaced steps between the range [0, 1].

The following equation calculates the indicator value of the AUC:

$$I_{AUC} = \frac{1}{S}\sum_{i=0}^{S}(1+\beta^2)\frac{\mathrm{Pr}\cdot\mathrm{Re}}{(\beta^2\cdot\mathrm{Pr})+\mathrm{Re}} \tag{3.21}$$

Precision and recall are calculated for a decision threshold from a linear range [0, 1] with a step size determined by the value of S. The later can be interpreted as the parameter giving resolution to the F_β curve.

3.5.2 F_β Soft Metric

The 'soft' in F_β soft metric refers to calculating values of the confusion matrix (Tab. 3.1, to be calculated similarly to F_1 soft loss in Sec. 3.4.2. Consequently, we calculate TP, FP, and FN according to a class's probabilities instead of an absolute count. These formulas are clearly described in Sec. 3.4.2. The soft F_β metric is not an indicator value of the AUC of the F_β curve, as we have seen in the previous section (Sec. 3.5.1). With the soft equations of TP, FP and FN (see Eqs. 3.9, 3.10 and 3.11) in mind, we can calculate the precision and recall on a batch of data in terms of probabilities. This probabilistic approach to TP, FP and FN does not lead to any changes to Eq. 3.18. The equation for the soft F_β metric is the same as the loss function: Eq. 3.19. This model selection metric results in a model optimized for high F_β soft values.

3.6 Ensemble Learning

Ensembling methods can be used to obtain state-of-the-art performance for many classification problems. They pool together individual predictions of multiple trained models into a single prediction called *model ensembling* [40]. Their performance has been tested rigorously, particularly on Kaggle, where huge ensembles outperform almost all single models trained on the same problem. Ensembles mostly rely on each member bringing a unique perspective of the problem into the ensemble. An ensemble has a method of combining these perspectives into a single inference. Each member has different initial weight initialization and should look at the data at least slightly different. By pooling these predictions together, you can get a far more accurate picture of the data, which reduces the variance of the ensemble relative to the individual models. The challenging part of model ensembling is, finding a method to combine individual predictions into a single prediction. The simplest baseline form of model ensembling is, using the individual predictions to calculate an average prediction at inference time. This is not a good baseline if some members are significantly better than others. However, it does provide a result to compare other methods with. The key part to correctly apply model ensembling is the diversity in ensemble members [40]. If all models are trained on the same data and have the same architecture, none of them provides a unique insight into the problem. To obtain a strong baseline, we could assign more weight to members that perform better. This is called a *weighted average ensemble*, where the weights are optimized on validation data. The following equation defines the final prediction of the ensemble, according to an ensemble weight vector $\vec{w}$, where each member gets their own associated weight for output prediction:

$$\vec{y}_{pred} = \mathbf{X} \cdot \vec{w} \tag{3.22}$$

$\mathbf{X}$ refers to the prediction matrix of size $N \times M$, where N is the number of samples in the data or batch and M the number of models in the ensemble. The vector $\vec{w}$ stands for a weight vector of size M.

3.6.1 Nelder-Mead - Weighted Average Ensemble

To search for suitable ensemble weights in Eq. 3.22, an optimization algorithm such as Nelder-Mead (NM) [41] can be used. It is an iterative *downhill simplex* method commonly applied to numerical, multidimensional data to find a minimum or maximum value of an objective function. In the case of finding ensemble weights, the number of members determines the objective function's dimensionality. A commonly applied objective function for classification is the mean squared error (MSE). The NM algorithm shrinks an $(M+1)$ dimensional polytope towards a minimum value in an M-dimensional space. When optimization terminates, we have obtained a suitable weight vector $\vec{w}$ that can be applied to Eq. 3.22 to obtain an ensemble prediction $\vec{y}_{pred}$ on some data $\mathbf{X}$.

3.6.2 Input Dependent Ensemble

We investigate an ensemble method that determines which member is most likely to make the best prediction based on an input image. Such an ensemble method will further be referred to as an *input dependent ensemble* (IDE). An image from the data will be fed into a separate DNN, with the last layer having a dimensionality equal to the number of ensemble members (M). Such a DNN will be referred to as a Dynamic Ensembler (DE_m), with size M. During the training of a DE_m, predictions need to be converted into one-hot encoding. One-hot encoding is a label vector with all values set to 0.0 except for one index, where it is 1.0. It is used often in multiclass classification problems combined with a softmax activation function. Training a DE_m starts with predicting $\hat{y}$ for each member in the ensemble for an input image resulting in a $\vec{y}_{pred}$ vector of size M. This part of the problem can be interpreted as a multiclass classification problem, where the DE_m has to predict which target class (which member) performs best on the input image. The resulting softmax output $\vec{y}_{pred}$ vector is converted to one-hot encoding, based on which member was closest with its prediction to the target label (y_{true}). The resulting one-hot encoding is treated as the label for the DE_m. Determining final inference is calculated via the same method as a weighted average ensemble. The DE_m predicts $\vec{w}_i$ in Eq. 3.23 for each input sample of $\mathbf{X}_i$.

$$\{\forall i \in |\mathbf{X}| \;\big|\; \hat{y}_i = \mathbf{X}_i \cdot w_i\} \tag{3.23}$$

3.7 Gradient-Weighted Class Activation Mapping

Visual explanations of CNNs can aid users in understanding why it makes the prediction that it makes. Gradient-weighted Class Activation Mapping (Grad-CAM) [42] uses the gradients of a target class to create a coarse heat map of the image regarding the evidence favouring the target class c. Grad-CAM makes use of the activations of the final convolutional layer concerning c. As stated in [42], it can help users discern a 'stronger' learner from a 'weaker' one, even when both make identical predictions.

CNNs are perfectly suited for visualization techniques because the feature maps in each layer retain spatial information. Visualizing activations is most useful and interpretable in the last layer because high-level concepts such as entire target classes are represented visually [43, 44]. If we can localize the region in the input image with the target concept, then we can better understand why the network makes a certain inference. This method can be especially insightful in misclassification cases, where we can localize regions in the input image introducing confusion or concepts that interfere with correct predictions.

To get the class discriminative localization map Grad-CAM $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$, where u is width and v is the height for class c, we need to compute the gradient for class c. We compute y^c before the final activation function concerning the feature maps A^k of a convolutional layer. The backwards flowing gradients are global-averaged-pooled in order to obtain the neuron importance weights α^c_k:

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{3.24}$$

The weight α^c_k captures how important the feature map k is with respect to target class c. The higher the value, the more feature map k contributes to a prediction of the positive class. Performing a weighted combination of activation maps and weighted neuron importance and the application of ReLU results in:

$$L^c_{Grad-CAM} = \text{ReLU}(\sum_k \alpha^c_k A^k) \tag{3.25}$$

The application of ReLU ensures that only feature maps with positive influence with regards to class c are captured, "*i.e.* pixels whose intensity should be increased in order to increase y^c." [42]. Negative pixels are likely to belong to other classes than class c. The heat map dimensions are determined by the dimensions of the feature maps in the convolutional layer used to obtain $L^c_{Grad-CAM}$. To aide users in feature localization in the input image for class c, the input image and the heat map should be superimposed. However, the heat map needs upscaling to match the input image's dimensions for superimposition to work. If the heat map is coarse, then a bicubic interpolation method might lead to misleading superimposition. It could be interpreted by the user as if the model is paying the most attention to the brightest parts. However, this is only an effect of upscaling. Using nearest neighbour as upscaling technique results in superimposition where the area with the positively contributing features concerning c is highlighted without attributing weights to individual pixels.

Chapter 4

Experimental Setup

4.1 The KiDS Survey

The Kilo-Degree Survey (KiDS) [45] is a publicly available data release designed to shed light on multiple topics, such as weak lensing shear tomography, large scale matter distribution in the universe, constraining the equation-of-state of Dark Energy, galaxy evolution, Milky Way structure, white dwarf detection and high-redshift quasars detection. KiDS is an ESO public survey carried out by the VLT Survey Telescope (VST) and OmegaCAM camera. The camera has imaged 1500 square degrees in four filters (u, g, r, i).

The KiDS has had four data releases over the years. Data release 3 (KiDS-ESO-DR3) or (DR3) has: "calibrated, stacked images and their weights, as well as masks and single-band source lists for 292 survey tiles not previously released." This thesis will use DR3 because Petrillo has used it [2] to determine a candidate set for strong gravitational lenses. Petrillo's study was followed up by a manual visual inspection of the lens candidates by four experts in the field (astronomers), resulting in a positive and negative candidate set. The positives seem convincing, and the negatives mostly contain images of spiral galaxies and other contaminants (Fig. 4.2).

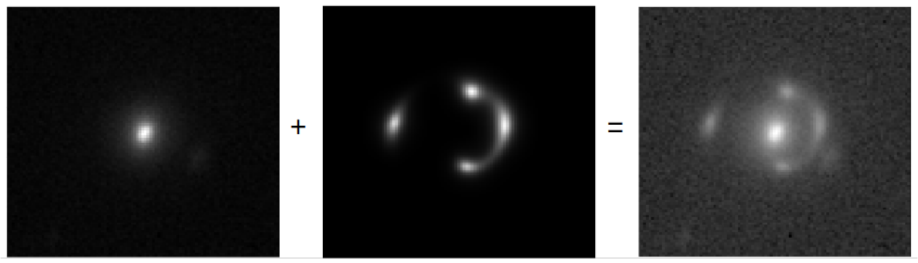


Figure 4.1: Lens (left), Source (center) and Mock Lens on the (right)

From DR3, we select a sub-sample, being Luminous Red Galaxies (LRGs) [46]. The likelihood of finding strong lenses among this subset is higher than average because these galaxies are usually very massive and, therefore, more likely to show lensing features. In [47], it is stated that the likelihood of a spiral galaxy showing lensing features is small. Meanwhile, the likelihood of elliptical galaxies showing these features is much greater. For this reason, LRGs have been chosen as selection criteria for finding strong gravitational lenses.

4.2 Preprocessing

There are a couple of steps in the preprocessing phase. The first step is to load all data (.fits files) into random-access memory (RAM). Each image will be normalized, according to Eq. 4.1.

$$f(x) = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

Sources (the artificial lensing features) need to be convolved with a point spread function (PSF) of the r-band to obtain realistic-looking sources. The PSF of the r-band for the KiDS data is the same as in [4]. One example of a source convolved with the PSF of the r-band is depicted in Fig. 4.1. Before we move to the lens and source merging step, we shuffle all the data.

Due to not having enough real-world positive data (real strong gravitational lenses) for deep learning, we have to perform experiments with simulated sources superimposed on lenses, creating mock lenses. To have a representative sample of source parameter space, we need a sufficiently large sources dataset. An order of magnitude 10^5 sources seems a reasonable sample for the sources parameter space.

The data is split into training-, validation- and test data 80/10/10, respectively. The training data is composed in the following way:

- 4.411 lenses
- 80.000 sources
- 4.867 negatives

If we combined sources with lenses in every possible combination, we would end up with approximately 353 million mock lenses. However, there are a couple of reasons not to use all combinations. First of all, the storage requirement would be huge, in the order of magnitude 10^4 GBs. Storing this amount of data is infeasible for the scope of this thesis and not necessary. Lastly, an ANN does not need every possible combination to learn what entails a strong gravitational lens. A representative sample covering the parameter space is sufficient.

During the training phase, sources and lenses are merged into mock lenses, which we refer to as positives. An equally sized set of negatives is added to the positives set, creating a *chunk* of data. Preliminary experiments have shown that a chunk size of 4096 results in stable and fast learning.

Mock lenses: Creating a mock lens is carried out by the following procedure. Firstly, randomly selecting k random samples with replacement from the lens- and source population, where k is half the chunk size. Furthermore, we rescale the source's peak brightness between 2% and 30% of the lens's peak brightness. This specific percentage further referred to as α, is drawn from a uniform distribution and applied to each mock lens. Eq 4.2 shows how a lens and source are merged into a mock lens [4]. By multiplying the source by α, we consider that lensing features are typically lower in luminosity than the lens. Lastly, a square-root stretch is performed in order to give more contrast to lower-luminosity features. A result of this procedure is depicted in Fig. 4.1. The lenses could already contain gravitational lensing features. However, it seems reasonable to assume that most do not due to their rarity in this dataset [4]. See Eq. 4.2 for the synthesis have a mock lens (m) based on the lens (l), source (s) and α-scaling.

Negatives: A negative sample does not show lensing features, or it depicts a non-lensing system that has features resembling a strong lens. Some clear examples of these contaminants are depicted in Fig. 6. This set does not require much preprocessing except for normalization. It contains 80% contaminants and 20% LRGs [3]. The contaminants show a wide variety of structures that were classified as being gravitational lenses in [3]. The idea behind this ratio is that the network should learn what does not constitute a strong gravitational lens. If we would only use LRGs as negative samples, the network would only pay attention to any structure near the lensing system's centre instead of considering its entire morphology. The negatives also get square-root stretched to emphasize lower luminosity features, just like the positives.

$$m = l + \frac{s \cdot \max(l) \cdot \alpha}{\max(s)} \tag{4.2}$$

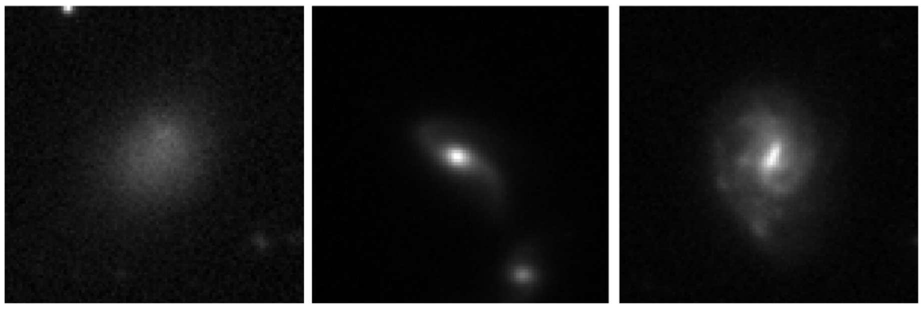


Figure 4.2: Three examples of lens candidates proposed by [2] but labelled as 'negative' after visual inspection by an expert. The negative set contains various spirals and other contaminants. Only the r-band is depicted in this figure.

4.2.1 Data Augmentation

DNNs have the potential to overfit drastically. Overfitting can be addressed by regularization of which data augmentation is one. It applies transformations to the images in order to cover more of the mock lens parameter space. To detect strong gravitational lenses, we employ the following data augmentations [4]:

- A uniform, random zoom range between 100% and 105%.
- A uniform, random rotation range between 0 and 2π.
- A uniform, random width and height shift of a maximum of 4 pixels.
- A 50% probability of a horizontal flip.

For the first three data augmentation techniques, we use a pixel interpolation method called **nearest neighbour**. Data augmentation is performed during training on the GPU in order to reduce training time. It is a sequence of matrix operations, which the GPU can calculate quickly.

4.3 Network Architecture

Our experimental setup is built around ResNet18, which is an 18-layered residual neural network [8] used by [4]. The last layer (fully connected layer) is changed to one neuron with a sigmoid activation function, which is often the case for binary classification problems. The input dimensionality is set to (101, 101, 1) because we only work with the red colour channel. A schematic overview of the architecture is depicted in Fig. 4.3. More details about ResNets can be found in Sec. 3.2.

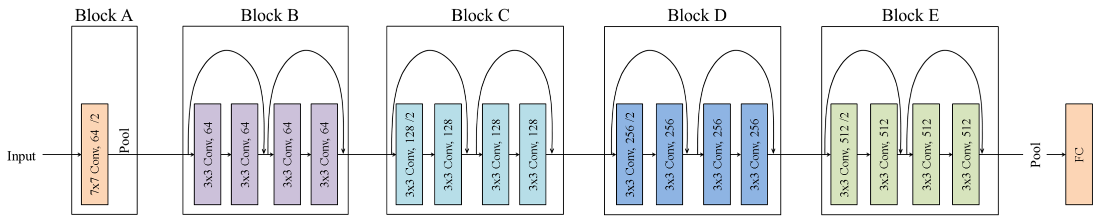


Figure 4.3: Image credit: [48]. The architecture depicts 8 skip connections and 5 blocks with their own associated feature map depth. The number of feature maps does not change within these blocks. Downsampling occurs at the beginning via 2D Max Pooling and strides at the beginning of each block. The last downsampling is 2D Average Pooling before the fully connected layer at the top.

4.4 Loss Functions

In an attempt to answer the first research question: **Which loss functions perform best with respect to precision and/or recall?** We limit ourselves to the loss functions described in Sec. 3.4. Additionally, we will assess the performance of models selected for F_β score and F_β- soft score.

As a baseline comparison model, we will train ResNet18 with binary cross-entropy as a loss function, similar to Petrillo's approach [3]. However, this loss function does not consider that high false-positive rates are undesirable. Therefore we train models with F_1-, F_1 double- and F_β soft loss as loss functions. Each model (ResNet18) is trained with the following hyperparameters:

- Number of chunks: 8000
- Chunk size: 4096
- Early stopping (patience: 1200 chunks)
- Batch size: 64
- Learning rate: 0.0001
- Optimizer: Adam [49]

Validation is conducted after training on a chunk of data. The validation chunk always has the same data and has a size of 1102, with equal positive and negatives.

Model selection is the process of selecting a model during training based on other functions than its loss function. In our case, a model is selected for F_β score or F_β soft score based on the validation data. If a measured score is better than a previous score, then the model is stored.

In order to have reproducible and stable results, experiments are performed in groups of 10. It means that we can assess the accuracy, F_β score, precision and recall as a mean of 10 runs with a standard deviation. Each loss function will be compared against the others based on these performance metrics.

4.5 Grad-CAM

Grad-CAM visualisations can be used to give users insight into classification decisions made by a DNN. This thesis will look at some TP, FP, FN and TN case studies for each trained model described in the previous section (Sec. 4.4). We will not be covering all misclassifications but rather look at interesting cases that seem like a trend for a given model. It would be interesting to see whether a DNN looks at noise objects in the image's background when predicting a FP. It can also give us a non-quantitative observation of lensing feature clarity, such as its apparent brightness and size. However, not all strong lenses can be identified by experts [3].

In [42], it is stated that deeper layers differentiate between high-level visual features for all target classes the most. Less deep layers have a higher chance of having overlapping features and are therefore less suitable for Grad-CAM visualisation. However, in a binary classification problem, we have fewer classes and fewer overlapping features. For this reason, we could visualise gradients of less deep convolutional layers. We will visualise the layers named add_x, where x is 4, 5, 6 and 7. The benefit of visualising earlier layers is that the produced heat map has a higher resolution than deeper layers due to having bigger feature maps. The reason for visualising the 'addition' layers in ResNet18 is that they capture both gradients of the previous convolutional layer and the skip connection gradient.

Training based on the red colour channel yields better performance than training on GRI composite images [3]. Training on the red channel is equivalent to training on morphology alone. In this thesis, only the red channel is considered and can be interpreted as greyscale. The created heat map can be viewed as a different colour and be superimposed with the input image if upscaling is applied first. As explained in Sec. 3.7, each heat map cell does not have a unique weight per pixel but equally weighs each pixel in a cell. To visualise equal weight per pixel, we employ nearest neighbour interpolation on the heat map and assign it the blue colour channel of an RGB image. The resulting superimposed image has its red channel assigned to the input image, the blue channel to the heat map and its green channel left empty.

4.6 Ensembles

To be able to answer the third research question: "**Does an ensemble of DNNs perform better than individual DNNs if members are trained on different loss functions or selected for different performance metrics?**" we compare two distinct ensemble methods, the *Nelder-Mead* (NM) method and an *Input Dependent Ensemble* (IDE) method (Sec. 3.6.2). Finding the appropriate members and ensemble size can be time-consuming if we would consider each combination. Instead, we will limit ourselves to the best performing models from Sec. 4.4 on each mentioned performance metric. Additionally, we will create an ensemble with each loss function and model selection criteria mentioned in Sec. 3.6. The following models trained on loss functions or selected for a specific metric are added to the ensemble:

- F_1 soft loss
- F_1 Double soft loss
- F_β soft loss
- Binary Cross-Entropy (loss)
- F_β score metric
- F_β soft score metric

We will consider the following ensembles for the NM ensemble method, NM$_x$ and NM$_6$, where x is the number of best performing models, on each metric, from Sec. 4.4. The number of models for NM$_x$ is not known before the first research question is answered because in an unlikely case, there could be only one model that performs best on accuracy, F_β score, precision and recall at the same time. We will construct an F_β score curve and compare it with individual models relevant to the first research question.

The objective function that NM will minimize is MSE. The algorithm will start with random weight initialization and iteratively decrease the MSE of the train data with respect to their label ($\vec{y}_{true}$). The algorithm terminates with a tolerance in the output of 10^{-6} or when the maximum number of iterations has been reached. This value is by default set to $M * 200$, where M is the number of members in the ensemble.

To train IDE_x and IDE_6, we need to choose an architecture for the Dynamic Ensembler (DE_m, where m are the number of members in the ensemble). Due to the input being an image, we need a CNN architecture. The DE_m needs a fine understanding of visual features in the input image. In [3], it was found that ResNet18 performs well and can capture most discriminative lensing features present in the input data. For this reason, it seems reasonable for the DE_m to have the same architecture, except for the last layer. The last layer will be changed into m neurons with a softmax activation function. This model's loss function should change accordingly to *categorical cross-entropy*, often the case in multiclass classification. Training the DE_m proceeds with the following hyper-parameters:

- Number of chunks: 2000
- Chunk size: 4096
- Learning rate: 0.0001
- Loss function: Categorical Cross Entropy
- Last activation function: Softmax
- Optimizer: Adam

The number of chunks is set to a high value to ensure that the DE_m converges. However, we will use a model checkpoint that keeps track of the best validation score during training and will only retain the model if the validation score has increased. The calculation of the DE_m targets is explained in Sec. 3.6.2.

4.7 Max-Tree Segmentation

During preliminary research, it was hypothesized that misclassifications could occur due to unrelated objects in the lens' field. Since they have no relation to the lensing system, the classifier should ignore them. To test this hypothesis, we use a segmentation algorithm that segments the lens and lensing features out of the image, which essentially rids the image of most noise objects in the background. This research has chosen a Max-Tree (MT) segmentation algorithm [50] with some additional post-processing.

The MT algorithm [51] is used to find connected components in a greyscale image and apply binary attribute filters to reconstruct a filtered image. It is a *rooted uni-directed* tree encoding set of nested peak components with regards to grey pixel intensities. A connected component can be computed according to two possible connectivity rules, 4- or 8- way connectivity on a grid. Each connected component C has to adhere to the *maximality condition*, which states: "there is no other connected set that is a superset of C that adheres to the same connectivity rules." [51]. Converting an image to greyscale and integer values between 0 and 255 is necessary before the MT algorithm can be applied. Each greyscale level is associated with its own depth h in the MT with the black background (pixel value 0) as its root. Every node is a set of components for which exists a unique mapping to the peak component. The leaves of the tree are regional maxima corresponding to the brightest pixels. If no binary filters are applied, then the resulting superimposed MT image is identical to the input image. A *binary connected operator* or binary filter accesses all pixels and returns the foreground if the operator returns the connected component it belongs to or an empty set (background) otherwise. Here are some examples of binary connected operators: area, rectangularity, compactness and intensity.

To segment the lensing system out of the image, we convert the normalized input image to 8-bit integer values, after which we apply a 5×5 Gaussian kernel for smoothing. This problem's connectivity rule is an 8- way connectivity kernel due to the circular features in the image. After smoothing, the MT can be constructed with the defined connectivity kernel, and the following filters can be applied to each connected component C with a bounding box (BB):

- Discard if surface area (a) < 45
- Discard if $BB_{dx} < 3$ or $BB_{dx} > 60$
- Discard if $BB_{dy} < 3$ or $BB_{dy} > 60$
- Discard if rectangularity ratio $RR < 0.45$, where RR is defined as: $\frac{a}{dx*dy}$

After the binary connected operators are applied, the MT is superimposed on itself to get a segmented image. However, this image can still contain segmented noise objects at the edges of the image that need to be filtered out. We apply an exponential decay formula on the pixel intensities (I) to get rid of these objects based on distance from the image's centre. The following formula is applied to each pixel:

$$I_{new} = I * e^{-d/\alpha} \tag{4.3}$$

Where d is the Euclidean distance from the centre of the image and $\alpha = 20$ an empirically determined slope decay rate. Decreasing this value will result in a further reduction of pixel intensity while increasing it will result in pixel intensity retention. After intensity scaling, a flood fill operation with a tolerance of 20 is performed to rid the image of faint structures remaining from pixel intensity scaling. When this process is complete, we should end up with an image where most distant noise objects are segmented out of the image, while most lensing features are retained due to filtering in the MT and pixel intensity scaling. Most lensing features are retained due to their proximity to the centre of the image.

4.8 Specifications

4.8.1 Hardware

Each experiment is performed on hardware from a node of the Peregrine cluster from the University of Groningen:

- **CPU**: 1x Intel Xeon Gold 6150 @ 2.70GHz (virtualized)
- **GPU**: 1x NVIDIA V100 (32GB VRAM)
- **RAM**: 128GB

Each experiment on loss functions lasted for about 6 to 8 hours. In contrast, experiments on max-tree segmentation could take between 22 up to 26 hours each.

4.8.2 Software

Experiments are build in Python 3, using Keras with a Tensorflow 2 backend. The code is available on Github at `https://github.com/Quintin1995/Strong_Gravitational_Lens_Detection_2.0`

Chapter 5

Results

This chapter will present results that pose to answer the research questions described in section 1.3. To reiterate, the research questions are:

- **Which loss functions perform best with respect to precision and/or recall?**
  This question is addressed by the results in Sec. 5.1.
- **Which lensing parameters explain the performance of the DNNs?**
  This question is addressed in Sec. 5.2 and Sec. 5.3.
- **Does an ensemble of DNNs perform better than individual DNNs if members are trained on different loss functions or selected for different performance metrics?**
  This question is addressed in Sec. 5.4
- **Does image segmentation result in better classification performance?**
  This question is addressed in Sec. 5.5.

5.1 Loss Functions

To assess the performance of different loss functions and model selection methods on a model, we need a common method for comparing prediction results. This can be achieved using the equation of F_β (eq. 3.18). In this equation, we can give precision relative importance with regards to recall. The average F_β curves per loss function or model selection method are depicted in Fig. 5.1. In Tab. 5.1, the accuracy, F_β-score, precision and recall are averaged over 10 identical runs (except for random weight initialization).

According to Tab. 5.1, a model with binary cross-entropy has the highest accuracy but not with a large margin with respect to a model selected for F_β or soft F_β performance metrics.

Model	Accuracy	F_β-score	Precision	Recall
Binary Cross Entropy	$\mathbf{0.937 \pm 0.005}$	0.947 ± 0.006	0.948 ± 0.007	0.924 ± 0.016
F_β metric	0.929 ± 0.005	0.930 ± 0.009	0.930 ± 0.009	0.927 ± 0.013
Soft F_β metric	0.935 ± 0.005	0.947 ± 0.009	0.948 ± 0.009	0.921 ± 0.014
F_β Soft Loss	0.849 ± 0.008	$\mathbf{0.969 \pm 0.007}$	$\mathbf{0.979 \pm 0.008}$	0.713 ± 0.014
F_1 Soft loss	0.892 ± 0.015	0.854 ± 0.029	0.852 ± 0.030	$\mathbf{0.951 \pm 0.015}$
F_1 Double Soft loss	0.917 ± 0.008	0.927 ± 0.012	0.927 ± 0.012	0.904 ± 0.013

Table 5.1: All performance metrics are calculated with the decision threshold set at 0.5. The second and third-row are models selected for the respective metric (F_β and F_β soft). These models are optimized with binary cross-entropy as the loss function.

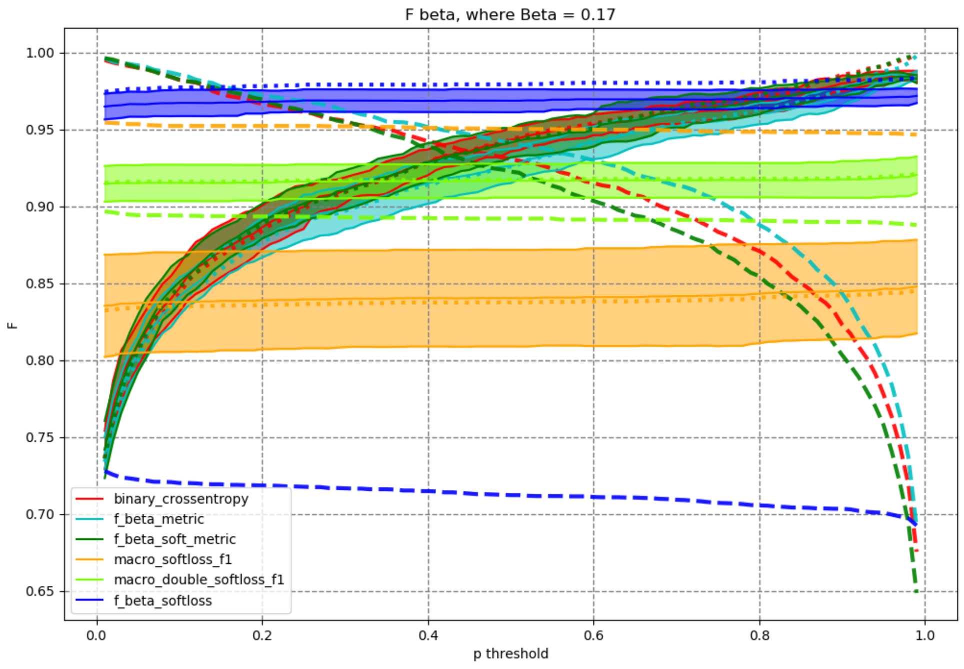
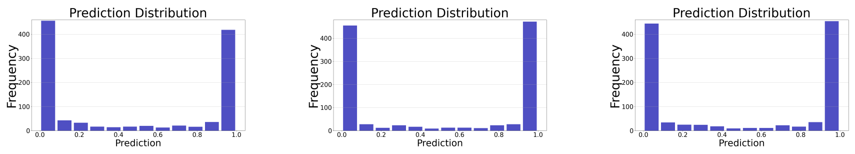


Figure 5.1: These curves represent the average F_β-score per model type. The average F_β-score is calculated over 10 runs on a test set with the same hyperparameters to average out the effect of different initial model weight initialization. Around each curve, the standard deviation is plotted. The long dashed lines represent recall, and the short dashed line represents precision. The p-threshold is the decision threshold of the classifier.

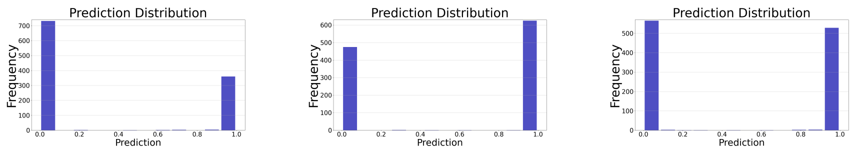
From Fig. 5.1, we observe that a model trained with F_β soft loss, performs really well on all decision thresholds. However, we can observe from Tab. 5.1 and Fig. 5.1 that the increased F_β score comes at the cost of a low recall of approximately 0.72 at a decision threshold of 0.5. On the other hand, this model's precision is better (Tab. 5.1) than the other models. High precision indicates a low false-positive rate, creating a strong lenses candidate set with high purity. The downside of a low recall is that the number of false negatives goes up, decreasing overall accuracy.

From Fig. 5.1, we observe that we have to choose different models depending on the performance metric of interest. If a researcher wants a high purity dataset, then the model optimized for F_β soft loss is the best choice. *Purity* is defined as a dataset having high TP and TN count, with low FP and FN count. The result of a low recall and high precision is that we end up with a model stringent in its prediction due to a high rejection rate. A high rejection rate for this model can be observed in Fig. 5.2. **b** on the left side. Relative to other models, it is far more likely to reject a sample than to accept it. Another interesting observation is the shape of the prediction distribution. They are heavily polarized for models trained with soft loss functions. Row **a** in Fig. 5.2 (also found in App. 7.1) shows that polarized models have a higher count in the histograms at 0.0 and 1.0 due to predictions between these values barely occurring.

Models trained with soft loss functions depicted in Fig. 5.1 are relatively flat because their output predictions are polarized. Changing the decision threshold does not result in a significant difference in performance concerning precision and recall. Because the models do not predict values between zero and one, but rather exactly zero or one (see Fig 5.2).



(a) Binary cross-entropy models. (Left: baseline, binary cross-entropy. Middle: F_β metric selection. Right: Soft F_β metric.)



(b) Left: F_β soft loss. Middle: F_1 soft loss. Right: F_1 double soft loss.

Figure 5.2: Prediction distributions on the test set, for 6 different models.

Recall defines the percentage of strong lenses recalled from all strong lenses in the dataset. If a researcher is interested in a model that returns as many true positives from the dataset as possible (not necessarily a high purity), then the model trained with F_1 soft loss might yield the best results. This can be observed in Tab. 5.1, where it has the highest recall.

The answer to the research question: "**Which loss functions perform best with respect to precision and/or recall?**" depends on what recall and precision are required by the research objective you are interested in. If the interest is a high purity (high precision and F_β score), then a model trained with the F_β soft loss function is the best choice. However, if an application requires high recall, a model trained with F_1 soft loss is a good fit. Lastly, training a model with binary cross-entropy as a loss function results in a model with the highest accuracy.

5.2 Gradient-Weighted Class Activation Mapping

This section will address the Gradient-weighted Class Activation Mapping (Grad-CAM) of some models listed in Tab. 5.1. It will try to answer the second research question (**Which lensing parameters explain the performance of the DNNs?**).

A Grad-CAM shows where a DNN looks while making inference about a class. First, we will address the Grad-CAM of a model trained with binary cross-entropy as the loss function. Next, we will look at Grad-CAMs from a model trained with F_β soft loss. However, we will not be looking at Grad-CAMs for all models listed in Tab. 5.1 because it is reasonable to assume that a model trained with binary cross-entropy responds the same as a model selected for F_β or soft F_β score. The same holds for models trained with soft loss: F_β soft loss, F_1 soft loss and F_1 double soft loss.

From Fig. 5.3, we can observe that the DNN seems to be looking at the correct features in the input image. They are the lensing features surrounding the central galaxy (lens). If the model did not consider the lensing features, only the central galaxy would probably be highlighted, not the lensing features. For a model to discriminate between a strong lens and a non-strong lens, the lensing features should be considered while making an inference. Fig. 5.3 is a true positive case, and most other true positive cases for this model seem to follow the same kind of Grad-CAM pattern, where the layers add_4 and add_5 show the most diversity. Layers add_6 and add_7 are similar across true positive cases. The central four regions of the Grad-CAMs are activated significantly and indicate that the DNN does look at the lensing features present in the input image. App. 7.2 shows three more true positive cases.

In Fig. 5.4, a Grad-CAM of a true negative is depicted. The model is clearly not convinced by the luminous sources near the lens because it predicts this image belongs to the negative class. The Grad-CAM algorithm (Sec. 3.7) states that it can only visualize evidence favouring the positive class. From Fig. 5.4, we can see this effect (column b) because the Grad-CAMs, do not focus

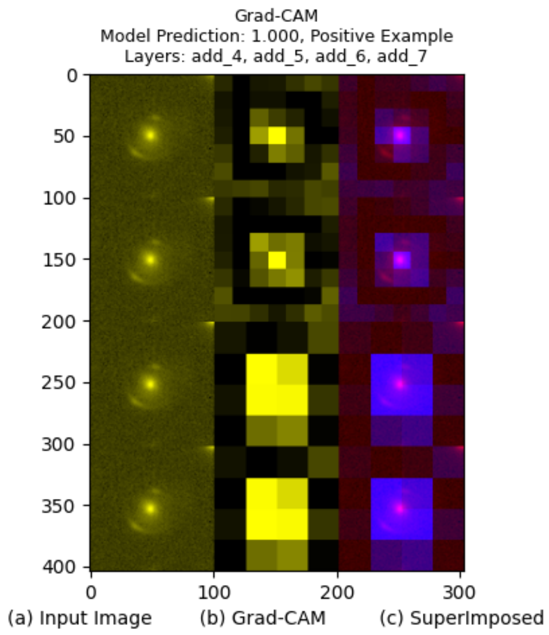


Figure 5.3: This is a true positive Grad-CAM on a test image. (a) Is the input image. (b) Is the Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

on any object except the edges of the image and a slight focus on the luminous object above the right side of the lens.

In Fig. 5.5, we observe empty Grad-CAMs. The interesting observation about this figure is the fact that the prediction is correct. The superimposed image's red colour is due to the Grad-CAMs being empty and assigned a colour channel. In the discussion section, we will discuss the possibilities, why this occurs.

Fig. 5.6 illustrates an example of a false positive. The model sees the spiral arms, around the central galaxy, as lensing features. In the second superimposed image in Fig. 5.6, a bar-like structure represents evidence for the positive class. Spiral galaxies take up a significant fraction of the negative class because they are considered a frequently occurring contaminant in the real world. Spiral galaxies such as this one are often classified as positive due to the spiral arms' similarity to lensing features.

A false negative is a member of the positive class, while the model predicts it is negative. Fig. 5.7 and Fig. 5.8 show examples of FNs, visualized with Grad-CAMs. We observe from Fig 5.7 that the model does not look at the middle of the image or its area around it when making an inference. The edges

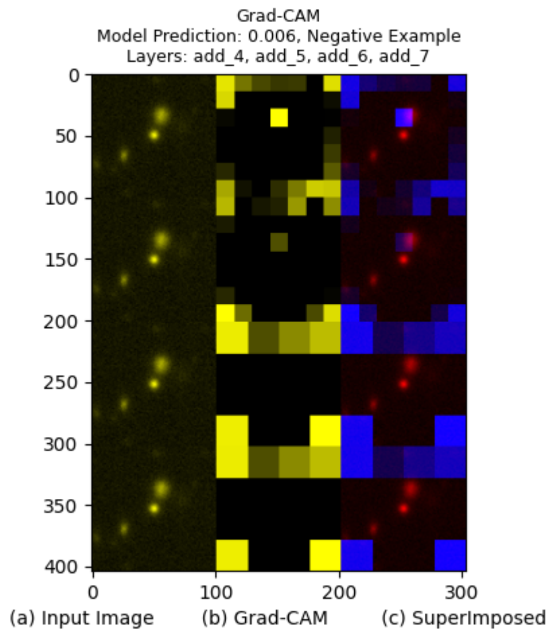


Figure 5.4: This is a true negative Grad-CAM on a test image. (a) Is the input image. (b) Is the Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

of the Grad-CAMs have non-zero values, as also observed for true negatives (for example, Fig. 5.4). Interesting about Fig. 5.7 is the apparent non-brightness of the lensing features, which could be why the inference is incorrect. Fig. 5.8 presents a lens with multiple noise objects around it (random galaxies in the background), with one lensing feature on the left side. The second row's superimposed image in Fig. 5.8 indicates that the model does see the left side of the lens as evidence for being a member of the positive class. However, the prediction (0.175) indicates that the model is not convinced adequately by the evidence. This fact be seen in the superimposed images of row 3 and 4 in Fig. 5.8. Here the Grad-CAMs are not active and do not overlap the noise galaxies and lensing feature.

5.3 Information Content

In this section, we will analyse the information content of a mock lens to get insight into the second research question: "**Which lensing parameters explain the performance of the DNNs?**" Fig. 5.7 from the previous section

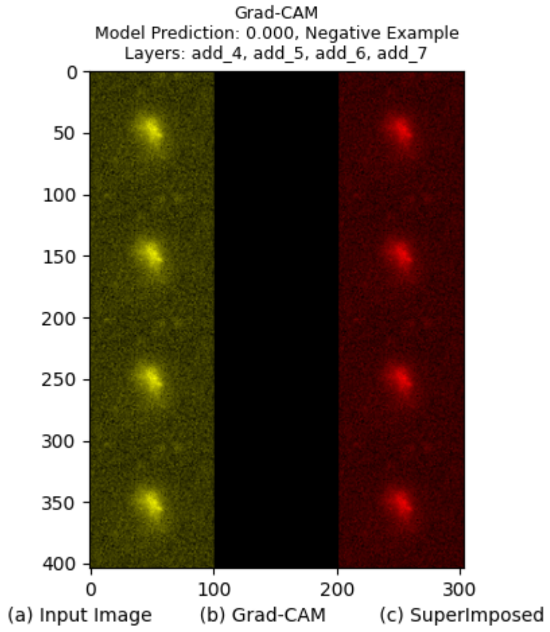


Figure 5.5: This is a true negative Grad-CAM on a test image. (a) Is the input image. (b) Is the Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

(Sec. 5.1) hinted at the lensing features not being bright enough. Additionally, the random selection of source and lens may result in a mock lens with lensing features overlapping the lens itself. This process could create mock lenses without visible lensing features.

To address the size of the lensing features and apparent brightness with respect to true positive rate (TPR) and false-negative rate (FNR), we need visualisation of performance relative to these features. Fig. 5.9 depicts a hexagonal tessellation of source brightness relative to the lens's peak brightness versus the source's Einstein radius. This figure is based on 551 test images of the positive class. When generating 'fake' sources, the parameters that make up the source are stored in its metadata. Therefore, the Einstein radius and some other parameters are available to us. The negative class does not have sources. Therefore we cannot create figures like Fig. 5.9 for the negative class. The TPR determines the colour-magnitude of a hexagon. The brightness scaling of the source relative to the lens's peak brightness (α) comes from a uniform distribution between 0.02 and 0.30. We record each α for all positive examples in the test set. Due to Einstein radii being generated according to an exponential probability distribution [3], where low Einstein radii are selected more often

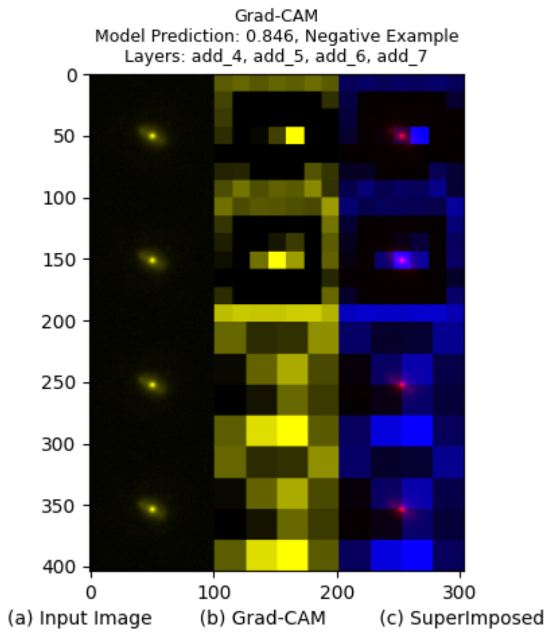


Figure 5.6: This is a false-positive Grad-CAM on a test image. (a) The input image. (b) The Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

than high values, a simple figure of Einstein radii versus relative source brightness does not suffice. It would seem like the model makes more mistakes on low Einstein radii because they are more abundant in our dataset. The hexagonal tessellation with TPR as colour-magnitude averages this problem out.

Binary cross entropy: From Fig. 5.9, we can observe that the most error comes from low α values and not necessarily from low Einstein radii. There seems to be a small effect of Einstein radii on TPR, where low Einstein radii result in a slight TPR decrease. Empty hexagons indicate that there is no data available within that data range. Fig. 5.9 depicts a black hexagon (TPR=0.0) near an Einstein radius of 3 and α between 0.20 and 0.25 because it is an average over one wrongly classified sample.

To investigate the TPR and FNR further, we can take a look at Fig. 5.10. This figure clearly illustrates the importance of high α values. This plot shows FNR and TPR per binned α range. From this figure, we can learn that mistakes made by a model trained with binary cross-entropy can be avoided if α is higher than 0.043.

Fig. 7.22 shows the fraction (image area with regards to the whole image area) of the source image, above two times the Root-Mean-Square (RMS) esti-

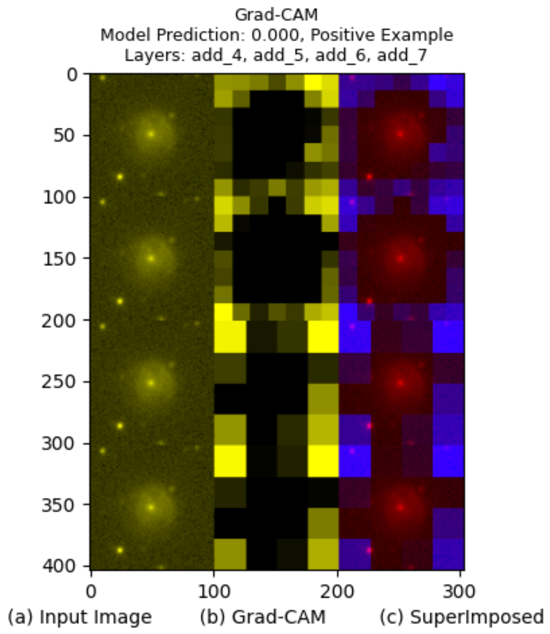


Figure 5.7: This is a false-negative Grad-CAM on a test image. (a) Is the input image. (b) Is the Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

mation of the lens's noise plotted versus the model prediction. We can observe that misclassifications only come from low fractions. A low fraction indicates that there might not be enough information in the mock lens for correct classification. However, many low fractions are correctly classified by the model. If we combine information from Fig. 5.10 and Fig. 7.22, we observe that low α and small Einstein radii are associated with higher misclassification rates.

F$_\beta$ Soft Loss: From Fig. 5.11, we observe a different effect than from Fig. 5.9. This effect can be attributed to a low recall (0.651), as is depicted in Tab. 5.1. This model could be perceived as a stringent model that rejects an example rather than accept it if the α and Einstein radius are low.

From Fig. 5.12, it becomes clear that α must be higher than 0.074 to get 100% TPR without regards for the Einstein radius. The model shows high TPR for Einstein radii above approximately 2.5 (see Fig. 5.11). From this figure, we observe a gradient from left to right, where the TPR increases with the Einstein radius. Keep in mind that these statistics are calculated over a test set containing positives only. Therefore we cannot say anything from these figures about the negative class rejection rates.

App. 7.3 has some interesting figures (Fig. 7.28 and Fig. 7.29), which show

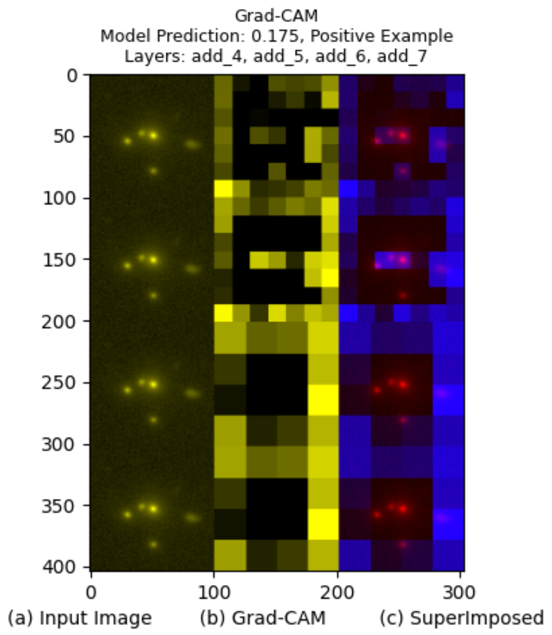


Figure 5.8: This is a false-negative Grad-CAM on a test image. (a) Is the input image. (b) Is the Grad-CAM response of layers add_4, add_5, add_6 and add_7, respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other using colour channels.

the area fraction of the source above two times the RMS estimation of noise in the lens. This figure indicates that the model can make mistakes only if the fraction is below 0.07. From Fig. 7.29, we observe that Einstein radii have less effect on performance due to mistakes being present along the whole range of Einstein radii.

If we compare Fig. 7.23 and Fig. 7.29, we see that a model trained with F_β soft loss has a much higher rejection rate than a model trained with binary cross-entropy.

F_1 Soft Loss: In Fig. 5.13, most hexagons have a TPR of 1.0, indicating that misclassifications do not occur regularly regarding the positive class. However, from Tab. 5.1, we observe that the model trained with F_1 soft loss has the lowest accuracy of all described models. Therefore, most misclassifications must come from the negative class, where negatives are predicted to be positives (FPs). From Fig. 5.1, we can observe that in terms of F_β score, the model does not perform better than the other models. However, this model performs well if the desired candidate strong lenses set should have a high TPR and that the FPR is less important.

The F_1 soft loss model is less sensitive to low α values than models trained

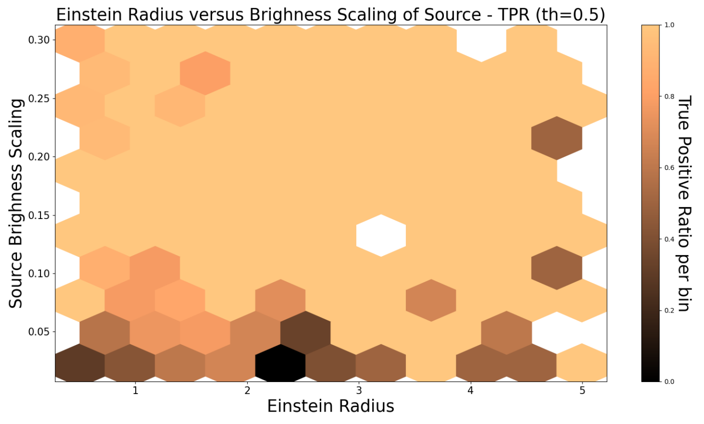


Figure 5.9: Model: **Binary Cross-Entropy**. TPR plotted versus Einstein radius and source brightness on 551 test images. Each hexagon contains a set of data points, which can express a TPR. The TPR is interpreted as the colour-magnitude of the hexagon.

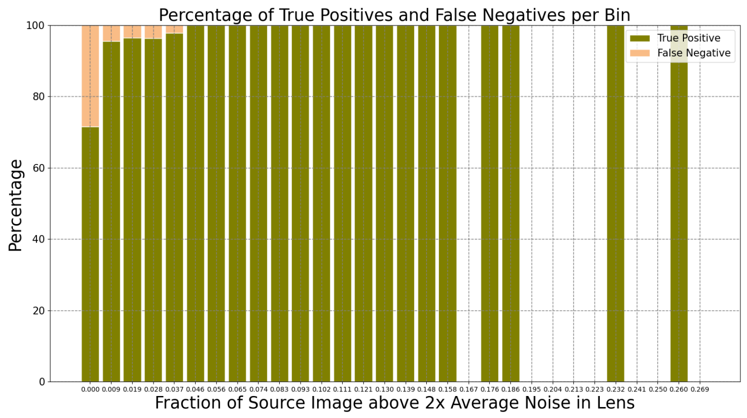


Figure 5.10: Model: **Binary Cross-Entropy**. The fraction of the source image area above 2x the unnormalised lens's RMS noise level on 551 test images. Each bar represents a binned value. If no bar is shown, then there are no data points within that range.

with binary cross-entropy or F_β soft loss. In Fig. 5.14, we observe that α should be above 0.017 for the TPR to be 100%. However, the TPR is not solely

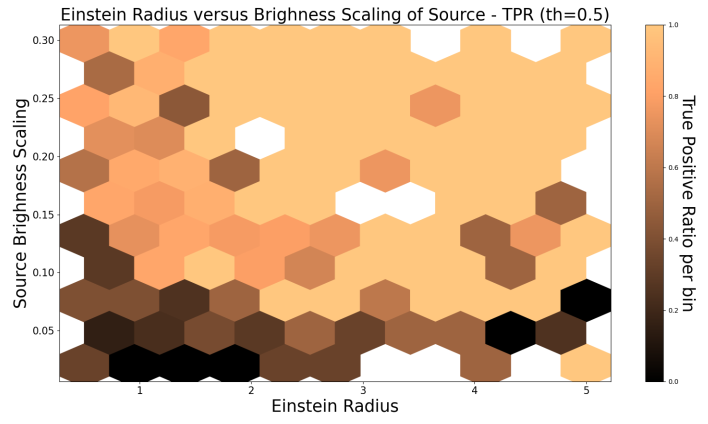


Figure 5.11: Model: $\mathbf{F}_{\beta}$ **Soft Loss**. TPR plotted versus Einstein radius and source brightness. Each hexagon contains a set of data points, which express a TPR and is interpreted as the colour-magnitude.

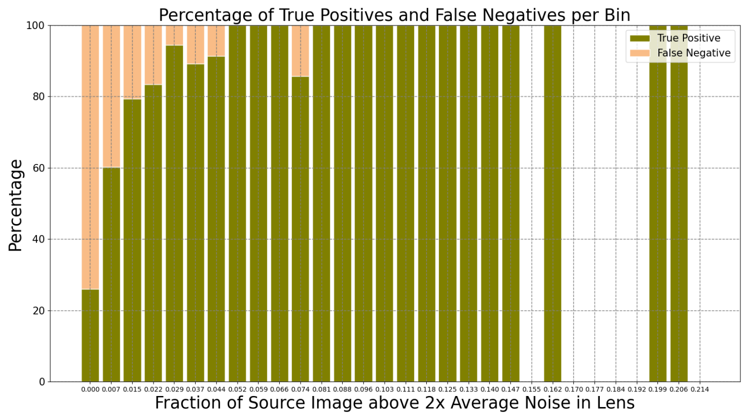


Figure 5.12: Model: $\mathbf{F}_{\beta}$ **Soft Loss**. The fraction of the source image area above 2x the RMS noise level of the unnormalised lens. Each bar represents a binned value. If no bar is shown, then there are no data points within that range.

determined by α but also by Einstein radii, as observed for the other models. This effect can be seen in Fig. 5.13 to a slight degree for Einstein radii near the value 1.0.

The answer to the research question: "**Which lensing parameters ex-**

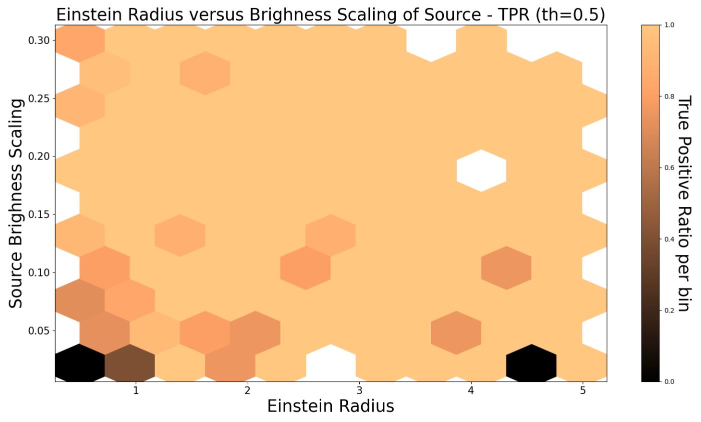


Figure 5.13: Model: $\mathbf{F_1}$ **Soft Loss**. TPR plotted versus Einstein radius and source brightness. Each hexagon contains a set of data points, which define a TPR and is interpreted as the colour-magnitude.

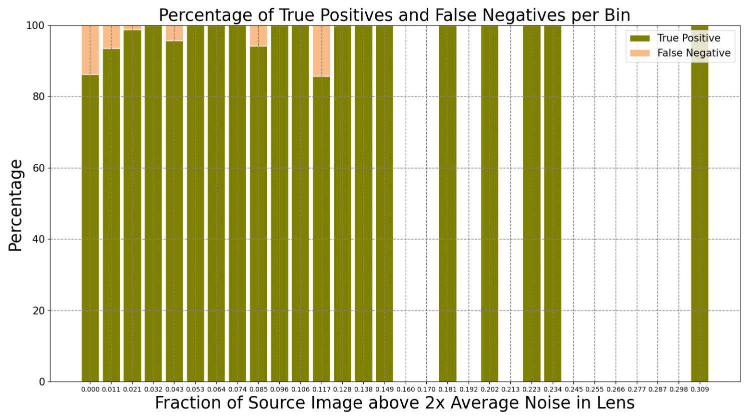


Figure 5.14: Model: $\mathbf{F_1}$ **Soft Loss**. The fraction of the source image area above 2x the RMS noise level of the unnormalised lens. Each bar represents a binned value. If no bar is shown, then there are no data points within that range.

plain the performance of the DNNs?" depends on the trained model. First of all, a model's performance trained with binary cross-entropy is explained by the source's brightness relative to the lens's peak brightness (α) and Einstein radii. Most errors occur if α is too low, and there must be enough area in the image above the noise level (> 0.05 from Fig. 7.22) representing the lensing features. Secondly, the F_β soft loss model is prone to errors if α and Einstein radius are low. There seems to be a strong correlation between both features having low values and poor classification results. The most important feature explaining the model's performance seems to be the value of α (see Fig. 5.11). The FPR of this model is low due to the model rejecting many potential positives. It displays the same behaviour as a model trained on binary cross-entropy regarding the area of the source above 2x the noise level. Misclassification seems only to occur if the image's information content (see Fig. 5.12) is below 0.07. Lastly, the performance of a model trained with F_1 soft loss is explained mostly by information content and class type. If information content reaches a value below 0.017, misclassifications start to occur regarding the positive class. The performance is less correlated with α and Einstein radii than the other models. Compared to a binary cross-entropy model, this model's low accuracy can mostly be ascribed by the performance on the negative class, where many negatives are predicted to belong to the positive class.

5.4 Ensembles

This section will address the third research question: "**Does an ensemble of DNNs perform better than individual DNNs, if members are trained on different loss functions or selected for different performance metrics?**".

Fig. 5.15 depicts the F_β score on a range of decision thresholds for the Nelder-Mead (NM_3) ensemble with members trained with the following loss functions: binary cross-entropy, F_β- and F_1 soft loss. Performance can be assessed through different metrics, which are listed in Tab. 5.2. It has become apparent that the best loss function for training a model depends on the research objective. With this in mind, we observe that neither the first nor second ensemble (see Tab. 5.2) performs better on any metric than models listed in Tab. 5.1 on a decision threshold of 0.5. For each metric defined in Tab. 5.2, there is a better performing model instance in Tab. 5.1. On the other hand, we observe an increase in F_β score for decision thresholds above approximately 0.75, indicating that the ensemble is better than its members if evaluated on this decision threshold. Comparing ensembles with their members based on F_β score can be performed on any value between 0.0 and 1.0. However, it must be noted that if this threshold had been different ($p > 0.75$, for example). The conclusion would be that the ensemble performs better on F_β score and precision but lower on recall (see Fig 5.15). The ensemble consists of members relatively insensitive to the decision threshold and a member that is not. Because we deal with a weighted average ensemble, it is possible to get an ensemble performing worse or better than its best performing member, depending on the preferred decision threshold. From the weight vector of the NM_3 ensemble in the caption from Fig. 5.15, we observe that most weight is attributed to the binary cross-entropy model, with equal weight for both F_β- and F_1 soft loss models.

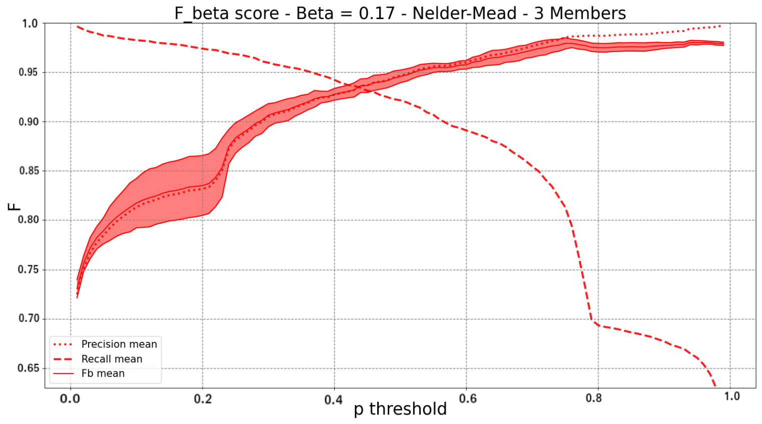


Figure 5.15: This figure depicts the F_β-score with standard deviation of the NM_3 ensemble with members trained on the following loss functions: binary cross-entropy, F_β- and F_1 soft loss, with respective mean model weight vector: $[0.55, 0.22, 0.23]$, calculated over 10 ensemble runs.

In Fig. 5.16, we observe a similar F_β curve as in Fig. 5.15. Its shape seems to follow the same kind of curvature as for models trained with binary cross-entropy, as shown in Fig. 5.1, except for low and high decision thresholds. On the low end approximately ($F_\beta = 0.2$), we observe a decrease in F_β score. However, on the high end of the decision thresholds (approximately $p > 0.75$), we observe an increased F_β score if we compare it with the NM_3 on the same decision threshold range. If we compare the F_β score on the high end, we see it is higher than models trained with binary cross-entropy, F_1-, F_1 double- and F_β soft loss (see Fig. 5.1). The weight vector from the NM_6 ensemble (see caption of Fig. 5.16) has the most weight attributed to the binary cross-entropy model and models selected for F_β soft-, and the F_β metric. This indicates that less weight is attributed to heavily polarized models that are trained with soft loss functions. We will detail why both the NM_3 and NM_6 ensembles have dips in their F_β score on low decision thresholds in the discussion section.

Studying Fig. 5.17 and Fig. 5.18 and the IDE_x ensembles from Tab. 5.2 results in the observation that these ensembles do not perform better on F_β score and precision relative to the NM_x ensembles and individual models summarized in Tab. 5.1 on a decision threshold of 0.5. However, we observe a substantial effect on ensemble variance between IDE_3 and IDE_6, where the latter has a lower standard deviation from the mean F_β score. Both curves' general shape is relatively flat relative to their NM_x counterparts. From observing the recall curves and Tab. 5.2 for both NM_x and IDE_x, we see that the IDE_x ensembles enjoy a higher overall recall.

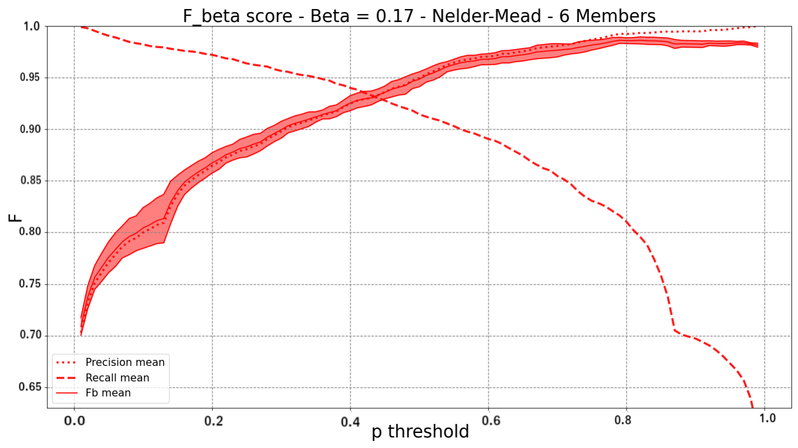


Figure 5.16: This figure depicts the F_β-score with standard deviation of the NM_6 ensemble with members trained on the following loss functions or selected for a metric: binary cross-entropy, F_β-, F_1 soft loss, F_β metric, F_1 double soft loss and F_β soft metric with respective mean model weight vector: $[0.20, 0.13, 0.13, 0.20, 0.13, 0.19]$, calculated over 10 ensemble runs.

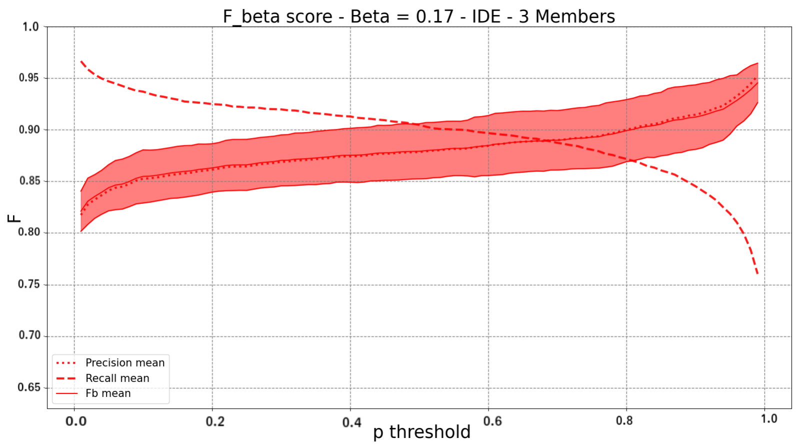


Figure 5.17: This figure depicts the F_β-score with standard deviation of the IDE_3 ensemble with members trained on the following loss functions: binary cross-entropy, F_β- and F_1 soft loss, calculated over 10 ensemble runs.

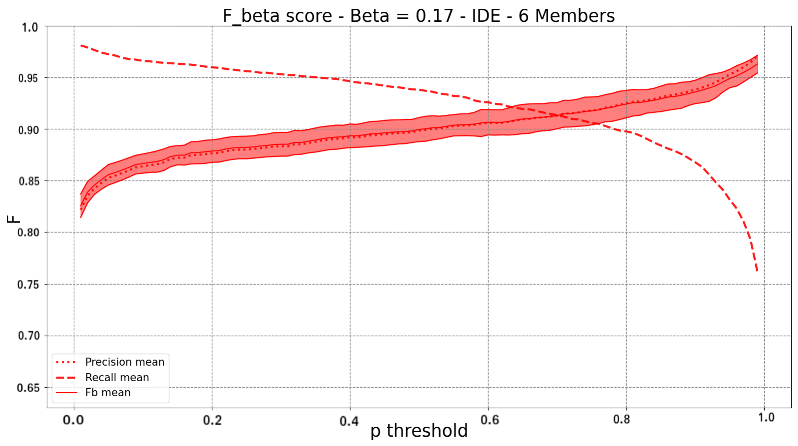


Figure 5.18: This figure depicts the F_β-score with standard deviation of the IDE$_6$ ensemble with members trained on the following loss functions or selected for a metric: binary cross-entropy, F_β-, F_1 soft loss, F_β metric, F_1 double soft loss and F_β soft metric, calculated over 10 ensemble runs.

Model	Accuracy	F_β-score	Precision	Recall
NM_3	0.890 ± 0.043	0.946 ± 0.005	0.947 ± 0.006	0.921 ± 0.014
NM_6	0.903 ± 0.037	$\mathbf{0.957 \pm 0.005}$	$\mathbf{0.959 \pm 0.005}$	0.915 ± 0.009
IDE_6	$\mathbf{0.912 \pm 0.014}$	0.913 ± 0.010	0.913 ± 0.011	0.928 ± 0.008
IDE_3	0.904 ± 0.014	0.898 ± 0.022	0.897 ± 0.023	$\mathbf{0.935 \pm 0.016}$

Table 5.2: All metrics are calculated with the decision threshold set at 0.5. The NM$_3$ and IDE$_3$ ensembles have the following members: binary cross entropy, F_β- and F_1 soft loss. The NM$_6$ and IDE$_6$ ensembles have the following members: binary cross-entropy, F_β-, F_1-, double F_1 soft loss and models selected for F_β metric and F_β soft metric. These statistics have been calculated over 10 ensemble runs.

The third research question can now be answered with respect to the evidence. An ensemble of DNNs can outperform its members on high decision thresholds based on F_β score. However, the ensemble does not necessarily perform better on all thresholds. On high decision thresholds (approximately $p > 0.75$) one should prefer an ensemble, however on lower thresholds an individual model is preferred.

5.5 Max-Tree Segmentation

The final research question: "**Does image segmentation result in better classification performance?**" is assessed through means of a Max-Tree image segmentation algorithm. During preliminary experiments, it was hypothesized that error could occur due to random noise objects (other galaxies in the background) being present in the lens image. This hypothesis's validity is assessed by segmenting the lens and lensing features out of the image (see Fig. 5.19).

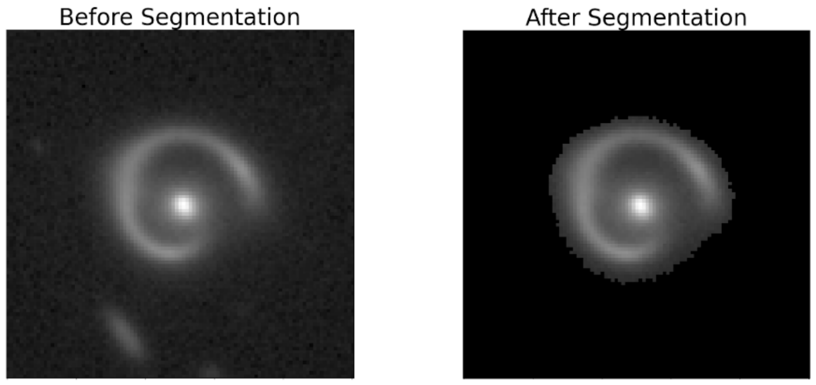


Figure 5.19: Before Max-Tree image segmentation (left) and after (right).

From Fig. 5.20, we observe a similar F_β curve as in Fig. 5.1 for models trained with binary cross-entropy. However, from Tab. 5.3, we can conclude that using this Max-Tree image segmentation algorithm does not increase performance on any aforementioned performance metric. Therefore, the last research question's answer is that this specific set of hyper-parameters for the Max-Tree image segmentation algorithm does not increase performance. The results from Sec. 5.3 suggests that the problem with performance comes from a lack of information content and not from noise objects. A visual inspection of the segmented image in Fig. 5.19 indicates that the algorithm works well.

Model	Accuracy	F_β-score	Precision	Recall
Max-Tree Seg.	0.898 ± 0.008	0.918 ± 0.014	0.919 ± 0.015	0.874 ± 0.015

Table 5.3: All metrics are calculated with the decision threshold set at 0.5. These models are optimized with binary cross-entropy as the loss function.

The last research question, "**Does image segmentation result in better classification performance?**" can be answered. Using Max-Tree image segmentation as preprocessing step does not improve accuracy, F_β score, precision and recall, as can be read from Tab. 5.3. Results from Sec. 5.3 also indicate

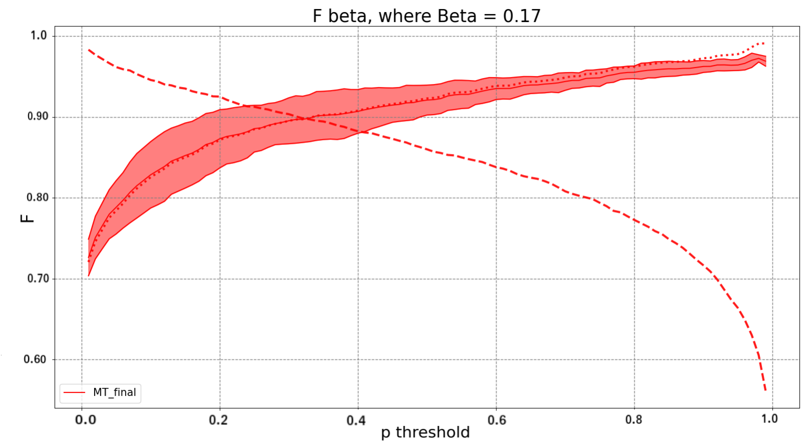


Figure 5.20: This figure depicts the F_β-score of a model trained with binary cross-entropy as the loss function. However, the data is preprocessed using a Max-Tree segmentation algorithm to segment the central galaxy (lens), and lensing features out of the image.

that misclassification occurs due to other effects than noise objects somewhere in the image, as was hypothesized earlier.

5.6 Discussion

In this section, we will discuss the findings of Chap. 5.

The heavy polarization of the prediction distribution depicted in the bottom row of Fig. 5.2 might be attributed to the learner being less hesitant than its binary cross-entropy counterpart in the top row. All of these models are associated with soft loss functions. They have learned not to hesitate in output prediction by predicting either 0.0 or 1.0. If the model would predict any value between the labels (0.0 or 1.0), then there would always be a loss value. The model might learn to predict either 0 or 1 because any value in between would result in a loss. This behaviour can also explain why the shape of the F_β curve for models trained on soft loss functions is relatively flat. The F_β score does not change significantly when varying the decision threshold p. The advantage of models trained with soft loss functions is that a search for a suitable decision threshold p is less critical because performance remains relatively constant for an arbitrary value of p in $0 < p < 1$.

A TN Grad-CAM is depicted in Fig. 5.4, in which the model is not convinced by the proposed evidence favouring the positive class. The evidence is considered to be the luminous source above the right side of the lens. This could indicate, that the model does not merely look for any luminous object near the lens but probably collects evidence from multiple locations around the central lens. Since no other evidence around the lens has been found, it is reasonable to assume that the model assigns this example to the negative class.

Fig. 5.5 depicts empty Grad-CAMs. This effect could be due to the average pooled gradients summing to less than zero because the pooled gradients' multiplication iteratively sets it to zero. If one feature map of all these feature maps has an average pooled value of zero, the entire product becomes zero. Empty Grad-CAMs occur with greater frequency on models trained with soft loss functions. During the algorithm's implementation, a slight change to the algorithm was tried, changing the product into a sum, circumventing the effect of multiplying by 0. This change did result in non-empty Grad-CAMs indicating that the change seems to have a positive effect. However, this contrasted the algorithm's inner workings and theory and decided that this change will not be a part of this research.

Interesting to note is the fact that models selected for performance metrics such as F_β- or F_β soft scores perform the same as models trained with binary cross-entropy as the loss function. During experimentation, these models seemed to perform better because they showed higher scores on validation data. However, this increased score was not present when evaluated on a test set.

An interesting effect of using Nelder-Mead weighted average ensembles with polarised members and members that are not (Fig. 5.15 and Fig. 5.16) is the bump in performance under two different regions of decision thresholds. The first bump is located on the lower end of the decision thresholds, at or below 0.2. This effect can be attributed to the use of a weighted average ensemble. It seems justifiable to assume that on the low end of p, the binary cross-entropy model does not have enough influence on the average F_β score. Therefore the shape of the curve is mainly influenced by the polarized models in these regions. This can also be supported by the fact that the polarized models do not predict much values, if any, between 0.0 and 1.0. Models trained on soft loss functions greatly influence the average weighted F_β curve on low and high ends of the curve because their predictions are situated in these regions.

Earlier it was hypothesized that misclassifications could occur due to noise objects, such as random other galaxies being present in the image's background. Thus, it would make sense to segment the lens and lensing features out of the image and use that as training data. However, the results (see Sec. 5.5) indicated that the Max-Tree image segmentation algorithm and preprocessing steps did not increase any performance metric. Suppose we combine these findings with the results from Sec. 5.3, where it was concluded that most misclassifications occur due to low luminosity of the lensing features and small feature sizes. In that case, it seems reasonable to assume that other segmentation algorithms or other values of the hyperparameters will not increase performance. It is likely that the DNNs could consistently ignore noise objects and that segmentation algorithms will hurt the training data by being too aggressive in pruning. Additionally, the computational cost of computing Max-Trees and filtering on them is severe, making examining them more cumbersome and less interesting.

Chapter 6

Conclusion

6.1 Summary of Results

This thesis began by asking four questions regarding the general increase in strong gravitational lens detection performance through DNNs and how their performance is explained. The first research questions asks which loss function to use with regards to precision and/or recall. In this research, we found that if high performance on accuracy, F_β-score, precision and recall is needed all at once, then training a DNN with binary cross-entropy as the loss function is sufficient. However, it is rarely the case in astronomy that all of these metrics are of equal importance. For strong gravitational lens detection, where the problem is severely imbalanced towards the negative class, it is often desirable to favour precision over recall, which F_β can express. Hence, it is good to train a model with F_β soft loss as the loss function because of its high F_β score and precision. On the other hand, if a particular research objective requires high recall, then training a model with F_1 soft loss results in high performance.

The second research question relates to the origin of misclassifications in the DNNs for this problem. First, we found out by a non-quantitative study with Grad-CAMs that most error seems to come from lensing features low in apparent luminosity and small Einstein radii. An exciting finding was that the DNNs seemed insensitive to noise objects in the background of the images, indicating that they ignored them during inference. Secondly, we quantitatively studied the amount of information content in the positive class images using RMS noise estimation, resulting in the conclusion that a combination of low lensing feature luminosity and Einstein radii resulted in most misclassifications. The lensing features' low brightness was deemed essential for misclassification due to the DNNs not recognising discriminative features.

The third research question aims to combine the strengths of DNNs trained on various loss functions into an ensemble. We found that an NM_6 ensemble performs better on high decision thresholds than the other ensembles and better than its members individually for F_β score and precision. The NM_3 ensemble also performs well on high decision thresholds but does not outperform the NM_6 ensemble. If the desired metric is accuracy, then an ensemble does not outperform a single model trained on binary cross-entropy. If a research objective requires recall, then a single model trained on F_1 soft loss is still a better choice.

Finally, we can conclude with the answer to the fourth research question regarding Max-Tree image segmentation. Here we answer whether we can improve performance if we segment the lens and lensing features out of the image and use that as training data. The findings suggest that Max-Tree image segmentation does not improve performance relative to a model trained on unsegmented data. The DNNs can ignore most noise objects in the background, therefore rendering image segmentation ineffective. We can conclude that image segmentation can hurt performance due to segmenting out parts of the image's essential lensing features.

6.2 Recommendations for Future Research

Future research could focus on multiple aspects concerning this research. First of all, a broader search for ensembling methods and which members to use could yield an ensemble performing even better than NM_6. Adding members with different DNN architectures or even machine learning algorithms could be beneficial. Secondly, an interesting topic could be the exploration of Cascade Ensembles, where the ensemble is created incrementally, and the output of previous members are fed to the inputs of subsequent members. Incremental ensemble cascade development can halt at a point of diminishing performance return. This method could yield good performance due to taking decisions from preceding models into consideration for inference at a later stage in the cascade. In [3], it was found that using all three colour channels (GRI) resulted in worse performance than a model trained on just the morphology (red colour channel). However, the output of models trained on GRI data with various loss functions could be valuable knowledge for a member later in the cascade. It would also be interesting to see how the soft loss models and the high performing NM_6 ensemble performs on real data (such as data from Euclid and LSST) and the effect of time spend manually visually inspecting the produced candidate sets. That research could also study the explanation of misclassifications and compare them with the explanation of this research. It would be an exciting inquiry because these datasets will have higher spatial resolution and be significantly larger in size.

Bibliography

[1] S. V. W. Beckwith, M. Stiavelli, A. M. Koekemoer, J. A. R. Caldwell, H. C. Ferguson, R. Hook, R. A. Lucas, L. E. Bergeron, M. Corbin, S. Jogee, and et al., "The hubble ultra deep field," *The Astronomical Journal*, vol. 132, no. 5, p. 1729–1755, Sep 2006. [Online]. Available: http://dx.doi.org/10.1086/507302

[2] C. Petrillo, C. Tortora, S. Chatterjee, G. Vernardos, L. Koopmans, G. Verdoes Kleijn, N. Napolitano, G. Covone, P. Schneider, A. Grado, and J. McFarland, "Finding strong gravitational lenses in the kilo degree survey with convolutional neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 472, 02 2017.

[3] C. Petrillo, C. Tortora, S. Chatterjee, G. Vernardos, L. Koopmans, G. Verdoes Kleijn, N. Napolitano, G. Covone, L. Kelvin, and A. Hopkins, "Testing convolutional neural networks for finding strong gravitational lenses in kids," *Monthly Notices of the Royal Astronomical Society*, vol. 482, pp. 807–820, 01 2019.

[4] C. E. Petrillo, C. Tortora, G. Vernardos, L. V. E. Koopmans, G. Verdoes Kleijn, M. Bilicki, N. R. Napolitano, S. Chatterjee, G. Covone, A. Dvornik, and et al., "Links: discovering galaxy-scale strong lenses in the kilo-degree survey using convolutional neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 484, no. 3, p. 3879–3896, Jan 2019. [Online]. Available: http://dx.doi.org/10.1093/mnras/stz189

[5] X.-P. Zhu, J.-M. Dai, C.-J. Bian, Y. Chen, S. Chen, and C. Hu, "Galaxy morphology classification with deep convolutional neural networks," *Astrophysics and Space Science*, vol. 364, 04 2019.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Advances in neural information processing systems," *Neural Information Processing Systems Foundation*, vol. 1269, 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[9] A. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, pp. 1–5, 01 2020.

[10] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." *Neural Networks*, vol. 32, pp. 323–332, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/nn/nn32.html#StallkampSSI12

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, jan 2016.

[12] A. Einstein, "Die Grundlagen der Allgemeinene Relativitätstheorie. (German) [The foundations of the Theory of General Relativity]," vol. 354, no. 7, pp. 769–822, 1916.

[13] V. Bonvin, F. Courbin, S. H. Suyu, P. J. Marshall, C. E. Rusu, D. Sluse, M. Tewes, K. C. Wong, T. Collett, C. D. Fassnacht, and et al., "H0licow – v. new cosmograil time delays of he 04351223:h0to 3.8 per cent precision from strong lensing in a flat cdm model," *Monthly Notices of the Royal Astronomical Society*, vol. 465, no. 4, p. 4914–4930, Nov 2016. [Online]. Available: http://dx.doi.org/10.1093/mnras/stw3006

[14] I. Jee, S. H. Suyu, E. Komatsu, C. D. Fassnacht, S. Hilbert, and L. V. E. Koopmans, "A measurement of the hubble constant from angular diameter distances to two gravitational lenses," *Science*, vol. 365, no. 6458, p. 1134–1138, Sep 2019. [Online]. Available: http://dx.doi.org/10.1126/science.aat7371

[15] G. C.-F. Chen, C. D. Fassnacht, S. H. Suyu, C. E. Rusu, J. H. H. Chan, K. C. Wong, M. W. Auger, S. Hilbert, V. Bonvin, S. Birrer, and et al., "A sharp view of h0licow: H0 from three time-delay gravitational lens systems with adaptive optics imaging," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 2, p. 1743–1773, Sep 2019. [Online]. Available: http://dx.doi.org/10.1093/mnras/stz2547

[16] M. Limousin, H. Ebeling, C.-J. Ma, A. M. Swinbank, G. P. Smith, J. Richard, A. C. Edge, M. Jauzac, J.-P. Kneib, P. Marshall, and et al., "Macs j1423.8+2404: gravitational lensing by a massive, relaxed cluster of galaxies atz= 0.54," *Monthly Notices of the*

Royal Astronomical Society, Mar 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2966.2010.16518.x

[17] T. Kitching, A. Amara, M. Gill, S. Harmeling, C. Heymans, R. Massey, B. Rowe, T. Schrabback, L. Voigt, S. Balan, and et al., "Gravitational lensing accuracy testing 2010 (great10) challenge handbook," *The Annals of Applied Statistics*, vol. 5, no. 3, p. 2231–2263, Sep 2011. [Online]. Available: http://dx.doi.org/10.1214/11-AOAS484

[18] C. A. Mason, T. Treu, K. B. Schmidt, T. E. Collett, M. Trenti, P. J. Marshall, R. Barone-Nugent, L. D. Bradley, M. Stiavelli, and S. Wyithe, "Correcting thez 8 galaxy luminosity function for gravitational lensing magnification bias," *The Astrophysical Journal*, vol. 805, no. 1, p. 79, May 2015. [Online]. Available: http://dx.doi.org/10.1088/0004-637X/805/1/79

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[21] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[24] I. Newton, “Philosophiae naturalis principia mathematica,” vol. Vol. 1, 1726.

[25] L. V. E. Koopmans, A. Bolton, T. Treu, O. Czoske, M. W. Auger, M. Barnabè, S. Vegetti, R. Gavazzi, L. A. Moustakas, and S. Burles, “THE STRUCTURE AND DYNAMICS OF MASSIVE EARLY-TYPE GALAXIES: ON HOMOLOGY, ISOTHERMALITY, AND ISOTROPY INSIDE ONE EFFECTIVE RADIUS,” *The Astrophysical Journal*, vol. 703, no. 1, pp. L51–L54, sep 2009. [Online]. Available: https://doi.org/10.1088/0004-637x/703/1/l51

[26] S. Vegetti and M. Vogelsberger, “On the density profile of dark matter substructure in gravitational lens galaxies,” *Monthly Notices of the Royal Astronomical Society*, vol. 442, 06 2014.

[27] A. M. Nierenberg, T. Treu, S. A. Wright, C. D. Fassnacht, and M. W. Auger, “Detection of substructure with adaptive optics integral field spectroscopy of the gravitational lens B1422+231,” *Monthly Notices of the Royal Astronomical Society*, vol. 442, no. 3, pp. 2434–2445, 06 2014. [Online]. Available: https://doi.org/10.1093/mnras/stu862

[28] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-propagating Errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: http://www.nature.com/articles/323533a0

[30] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” *Neural Information Processing Systems*, vol. 25, 01 2012.

[31] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[32] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10). Society for Artificial Intelligence and Statistics*, 2010.

[33] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” *CoRR*, vol. abs/1412.1710, 2014. [Online]. Available: http://arxiv.org/abs/1412.1710

[34] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *CoRR*, vol. abs/1505.00387, 2015. [Online]. Available: http://arxiv.org/abs/1505.00387

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[36] B. Amos and D. Yarats, "The differentiable cross-entropy method," *CoRR*, vol. abs/1909.12830, 2019. [Online]. Available: http://arxiv.org/abs/1909.12830

[37] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, "Thresholding classifiers to maximize f1 score," 2014.

[38] A. Galstyan and P. R. Cohen, "Empirical comparison of "hard" and "soft" label propagation for relational classification," in *Inductive Logic Programming*, H. Blockeel, J. Ramon, J. Shavlik, and P. Tadepalli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 98–111.

[39] R. Metcalf, M. Meneghetti, C. Avestruz, F. Bellagamba, C. Bom, E. Bertin, R. Cabanac, F. Courbin, A. Davies, E. Decencière, R. Flamary, R. Gavazzi, M. Geiger, P. Hartley, M. Huertas-Company, N. Jackson, C. Jacobs, E. Jullo, J. Kneib, L. Koopmans, F. Lanusse, C. Li, Q. Ma, M. Makler, N. Li, M. Lightman, C. Petrillo, S. Serjeant, C. Schäfer, A. Sonnenfeld, A. Tagore, C. Tortora, D. Tuccillo, M. Valentín, S. Velasco-Forero, G. Verdoes Kleijn, and G. Vernardos, "The strong gravitational lens finding challenge," *Astronomy astrophysics*, vol. 625, no. May 2019, May 2019.

[40] F. Chollet, *Deep Learning with Python.* Manning, Nov. 2017.

[41] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[42] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[43] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012. [Online]. Available: http://arxiv.org/abs/1206.5538

[44] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *CoRR*, vol. abs/1512.02017, 2015. [Online]. Available: http://arxiv.org/abs/1512.02017

[45] J. T. A. de Jong, G. A. V. Kleijn, T. Erben, H. Hildebrandt, K. Kuijken, G. Sikkema, M. Brescia, M. Bilicki, N. R. Napolitano, V. Amaro, and et al., "The third data release of the kilo-degree survey and associated data products," *Astronomy Astrophysics*, vol. 604, p. A134, Aug 2017. [Online]. Available: http://dx.doi.org/10.1051/0004-6361/201730747

[46] D. J. Eisenstein, J. Annis, J. E. Gunn, A. S. Szalay, A. J. Connolly, R. C. Nichol, N. A. Bahcall, M. Bernardi, S. Burles, F. J. Castander, and et al., “Spectroscopic target selection for the sloan digital sky survey: The luminous red galaxy sample,” *The Astronomical Journal*, vol. vol. 122, no. 5, p. 2267–2280, Nov 2001. [Online]. Available: http://dx.doi.org/10.1086/323717

[47] M. Fukugita, T. Futamase, M. Kasai, and E. L. Turner, “Statistical properties of gravitational lenses with a nonzero cosmological constant,” *Astronomical Journal*, vol. vol. 393, pp. 3–21, July 1992.

[48] S. Ghassemi and E. Magli, “Convolutional neural networks for on-board cloud screening,” *Remote Sensing*, vol. 11, p. 1417, 06 2019.

[49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[50] R. Souza, L. Rittner, R. Machado, and R. Lotufo, “iamxt: Max-tree toolbox for image processing and analysis,” *SoftwareX*, vol. 6, pp. 81–84, 12 2017.

[51] G. Ouzounis, “The max-tree data structure,” 02 2018.

Chapter 7

Appendix A

7.1 Model Predictions

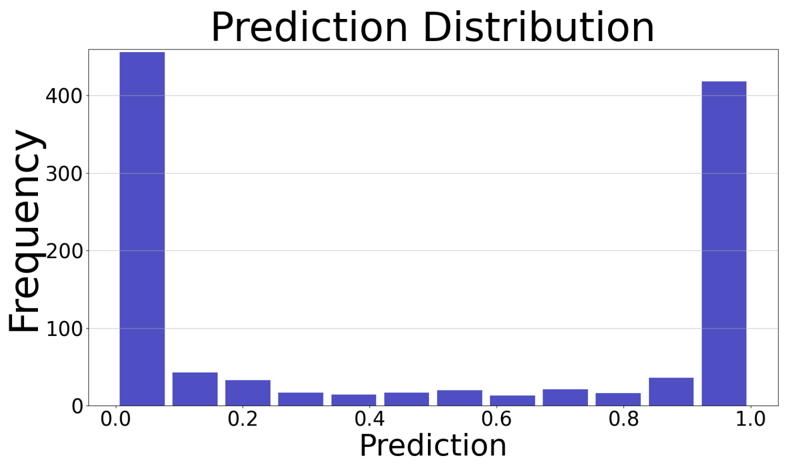


Figure 7.1: Prediction results on test set of model trained with binary cross entropy as loss function.

7.2 Gradient-Weighted Class Activation Mapping

7.3 Information Content

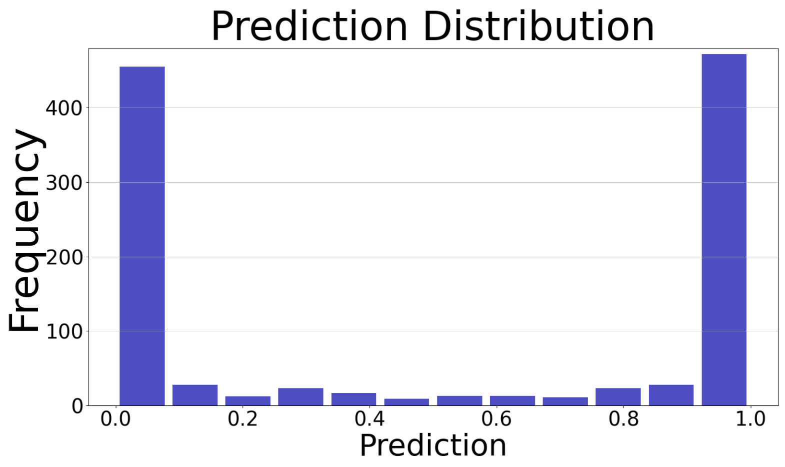


Figure 7.2: A model trained with binary cross entropy as loss function, but selected on the F_β metric.

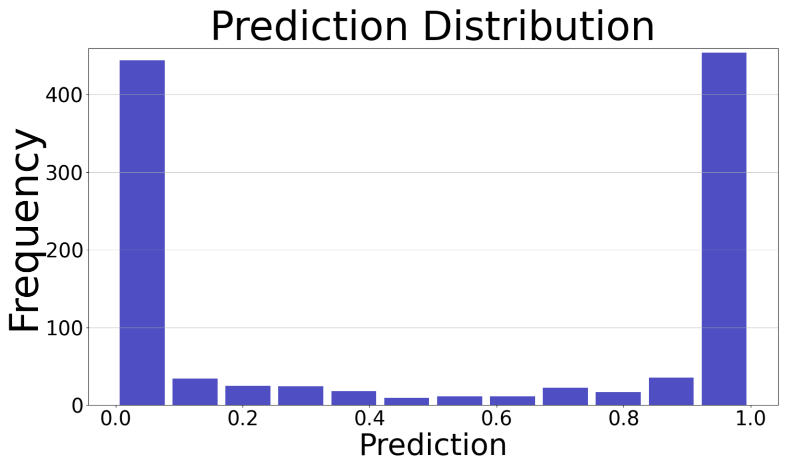


Figure 7.3: A model trained with binary cross entropy as loss function, but selected on F_β soft loss as metric.

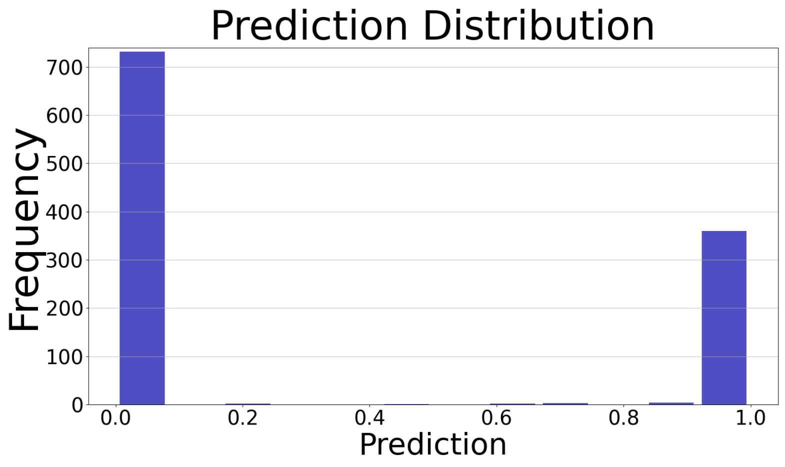


Figure 7.4: A model trained with F_β softloss as loss function.

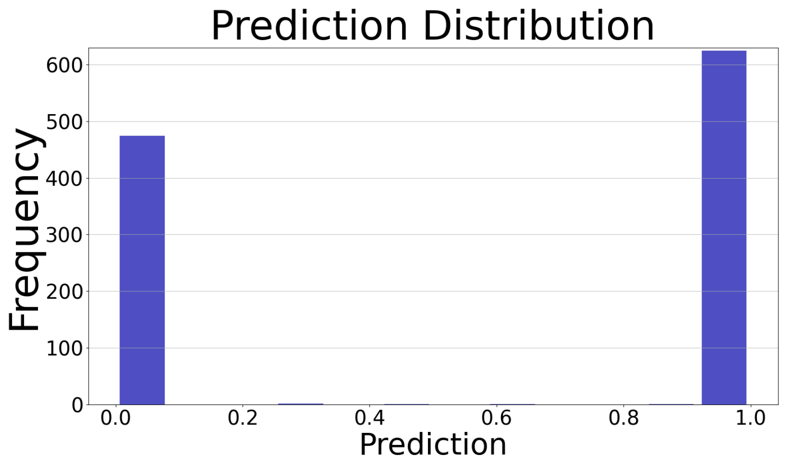


Figure 7.5: A model trained on the F_1 soft loss function.

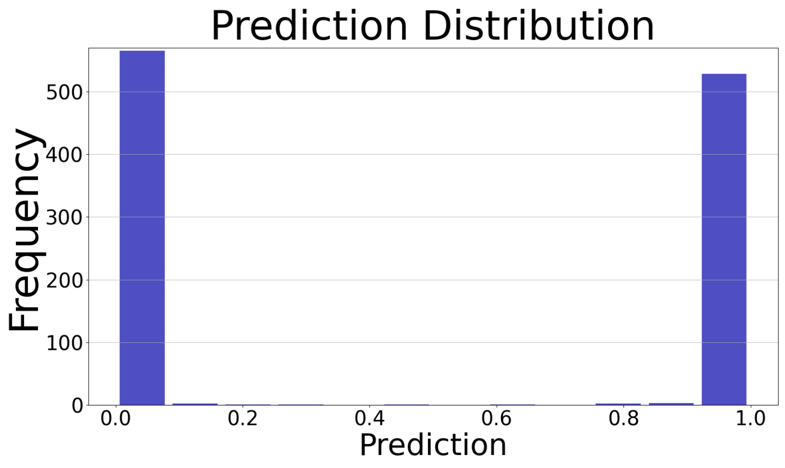


Figure 7.6: A model trained on the F_1 double soft loss function.

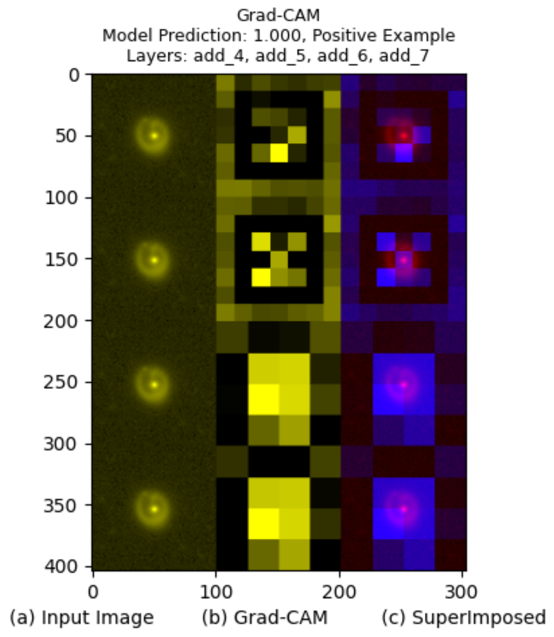

Grad-CAM
Model Prediction: 1.000, Positive Example
Layers: add_4, add_5, add_6, add_7
(a) Input Image
(b) Grad-CAM
(c) SuperImposed

Figure 7.7: True positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

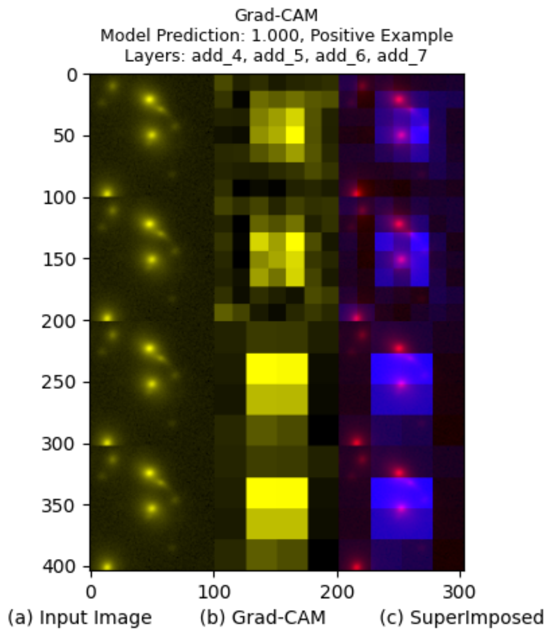


Figure 7.8: True positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

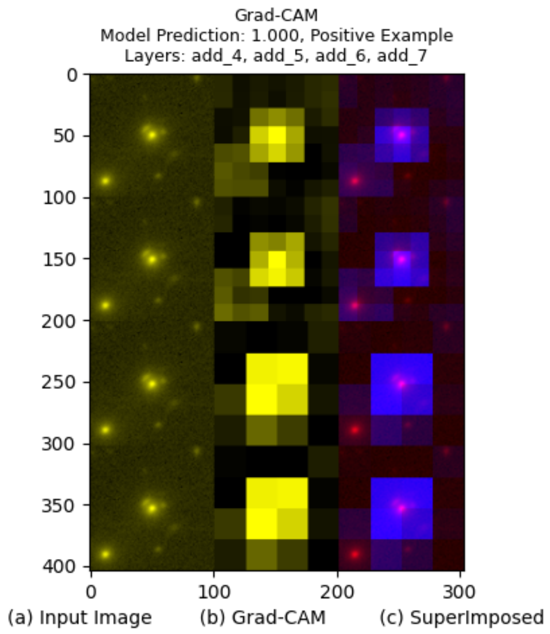


Figure 7.9: True positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

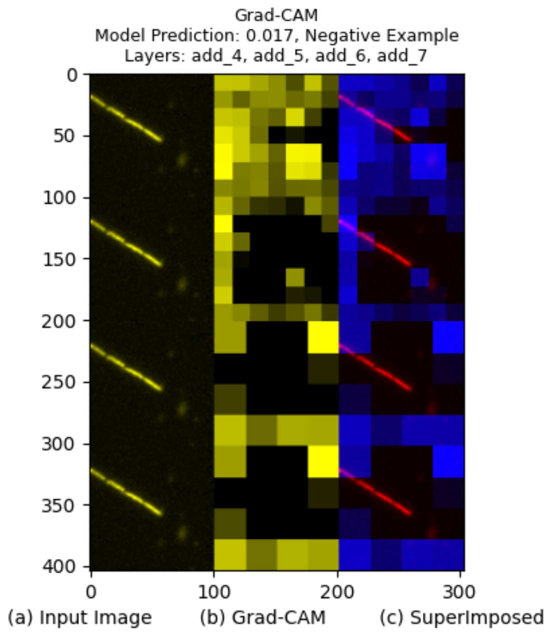


Figure 7.10: True negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

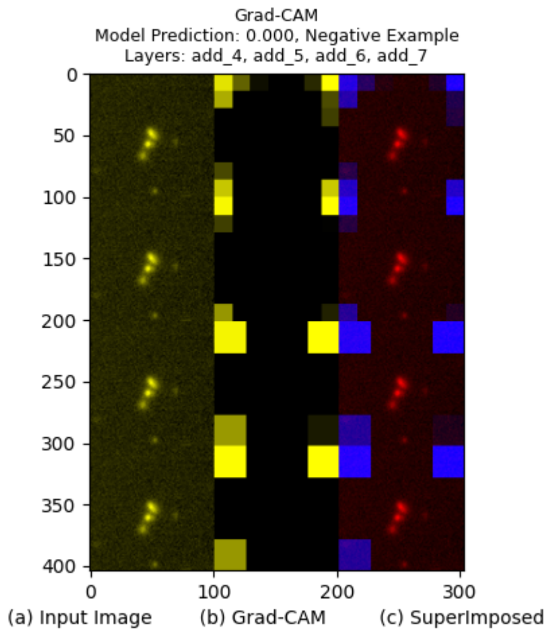


Figure 7.11: True negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

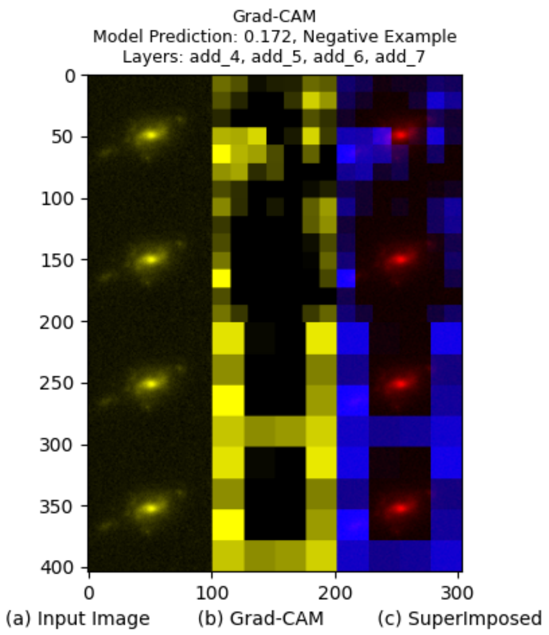


Figure 7.12: True negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

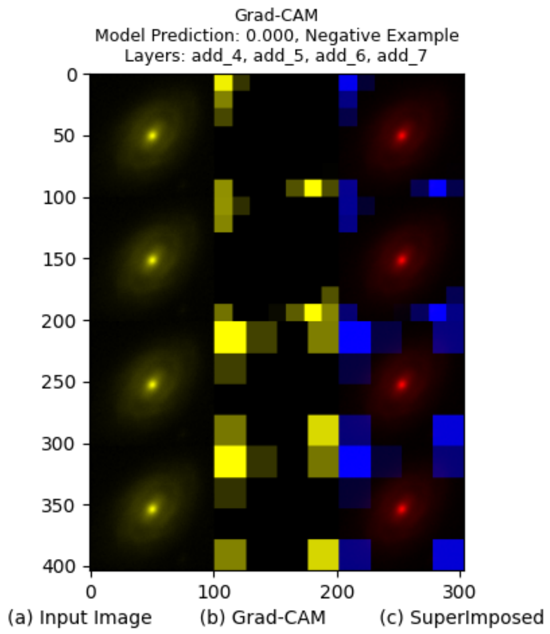


Figure 7.13: True negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

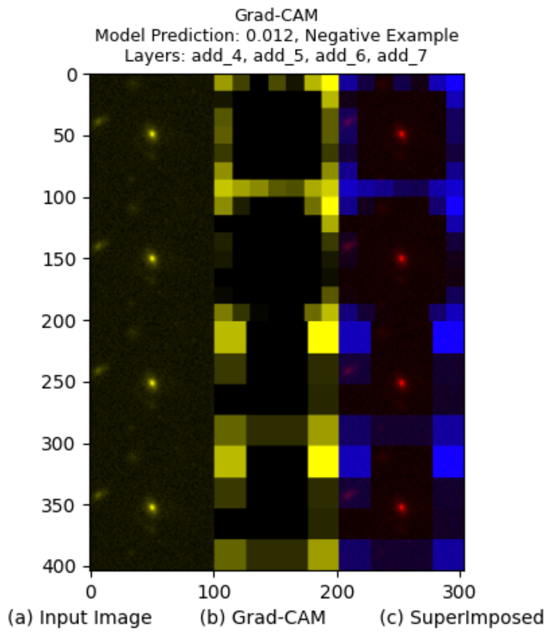


Figure 7.14: True negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels. This is a model trained with binary cross entropy as loss function.

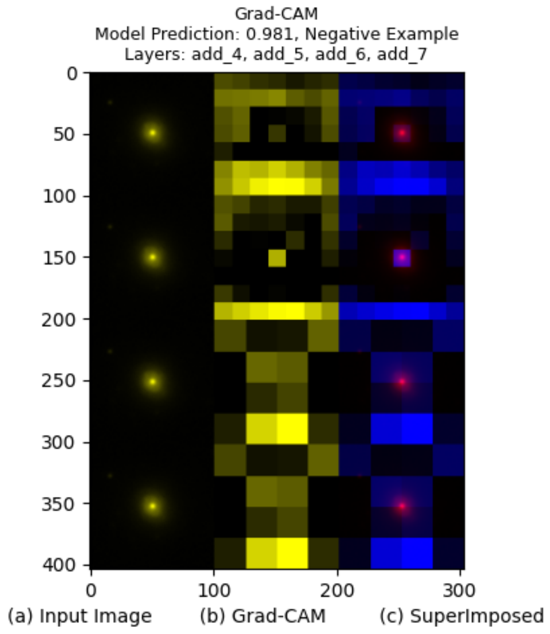


Figure 7.15: False positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

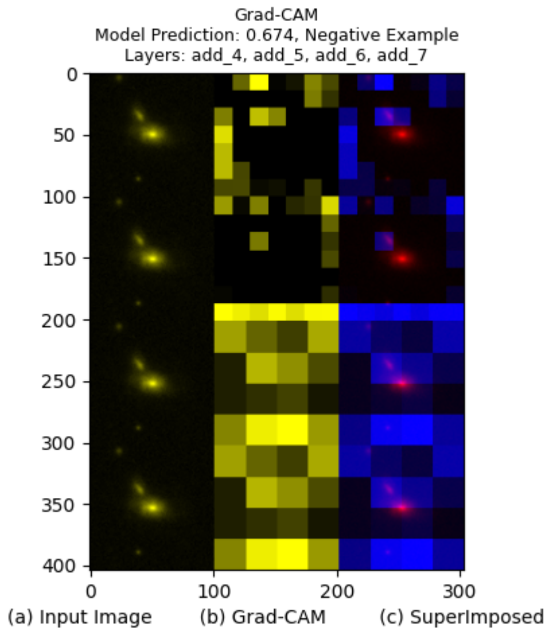


Figure 7.16: False positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

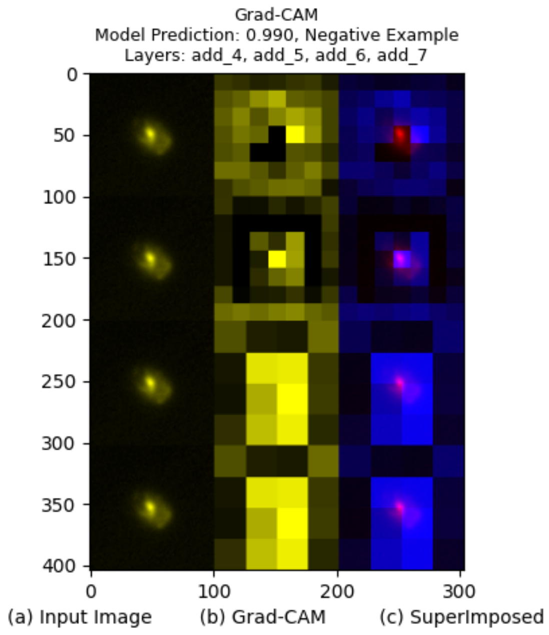


Figure 7.17: False positive grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

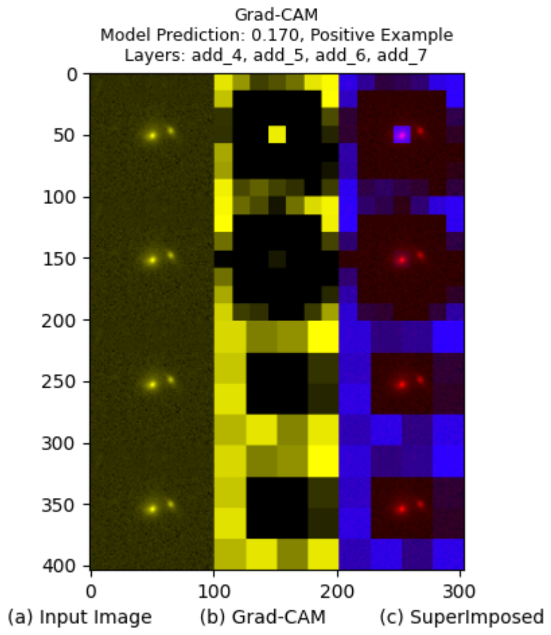


Figure 7.18: False negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

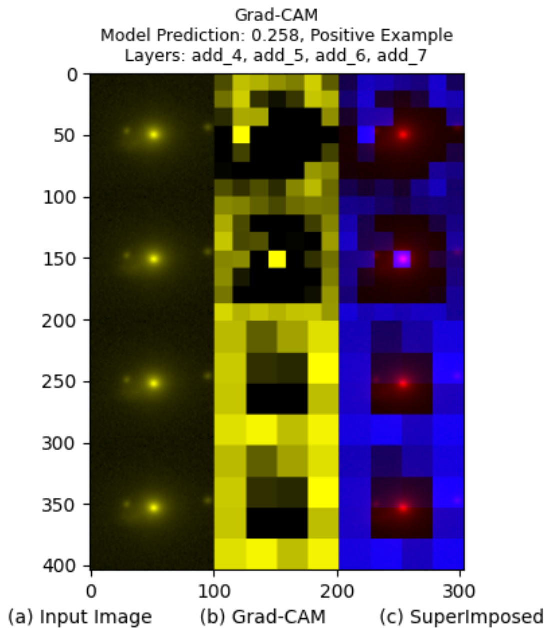


Figure 7.19: False negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

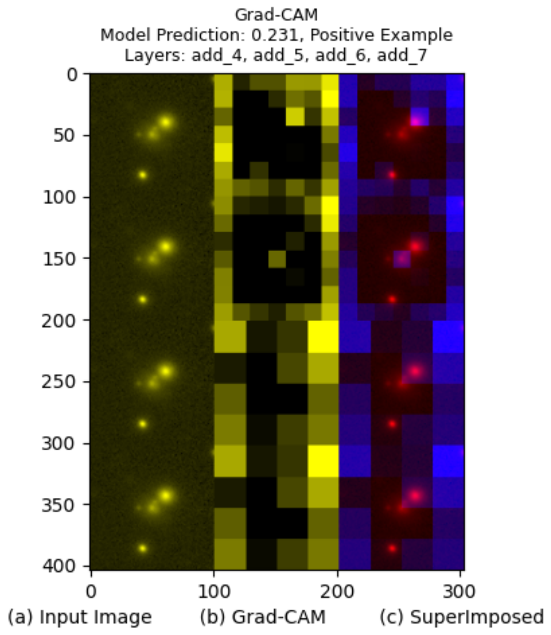


Figure 7.20: False negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

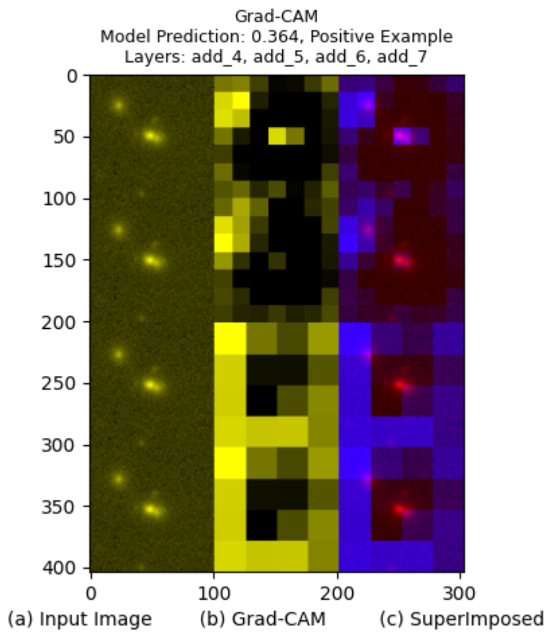


Figure 7.21: False negative grad-CAM on a test image. (a) Is the input image. (b) Is the Gradient-weighted Class Activation Mapping response of layers add_4, add_5, add_6 and add_7 respectively (top to bottom). (c) are both (a) and (b) superimposed on top of each other, by means of using color channels.

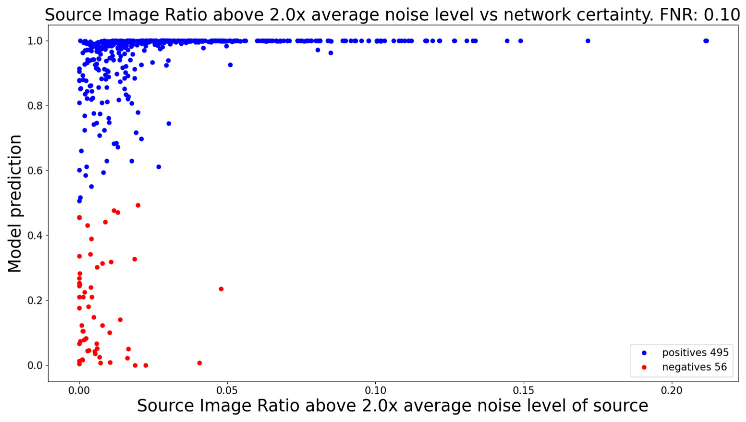


Figure 7.22: Model: **Binary Cross Entropy**. Root-Mean-Square (RMS) noise level estimation. The x-axis represents the fraction of the source image above 2x the RMS estimation of the noise. The decision threshold is set to 0.5.

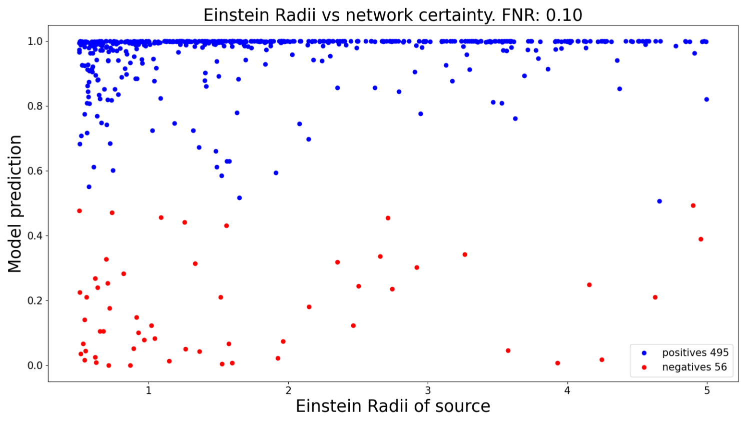


Figure 7.23: Model: **Binary Cross entropy**. Model certainty plotted versus the Einstein Radius of the Source image.

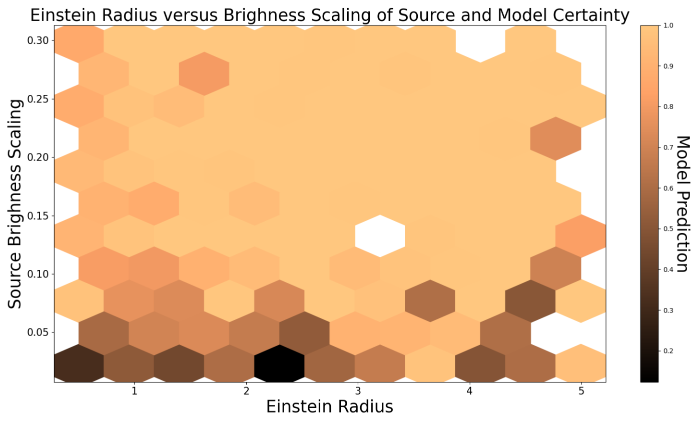


Figure 7.24: Model: **Binary Cross Entropy**. Model certainty plotted versus Einstein Radius and source brightness. Each hexagon contains a set of data points, which are averaged in order to obtain the color magnitude of the hexagon.

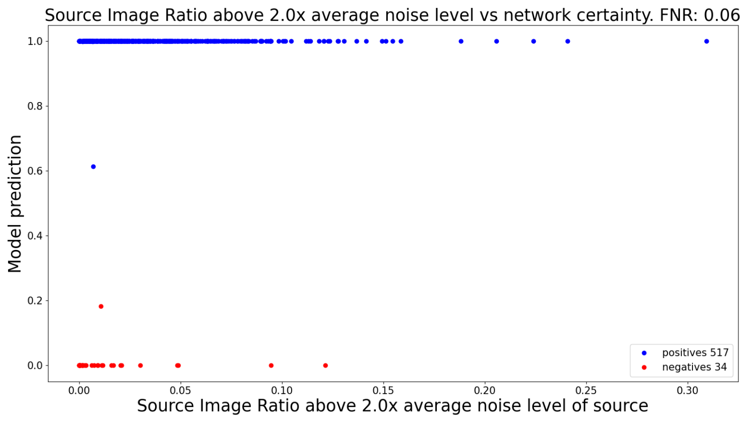


Figure 7.25: Model: $\mathbf{F_1}$ **Soft Loss**. Root-Mean-Square (RMS) noise level estimation. The x-axis represents the fraction of the source image above 2x the RMS estimation of the noise. The decision threshold is set to 0.5.

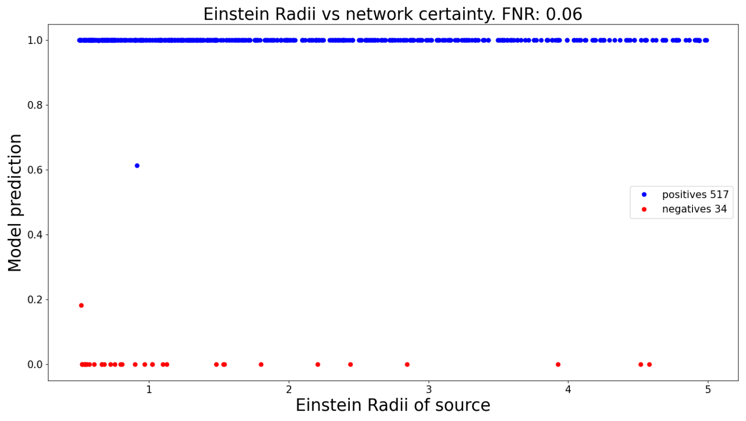


Figure 7.26: Model: $\mathbf{F_1}$ **Soft Loss**. Model certainty plotted versus the Einstein Radius of the Source image.

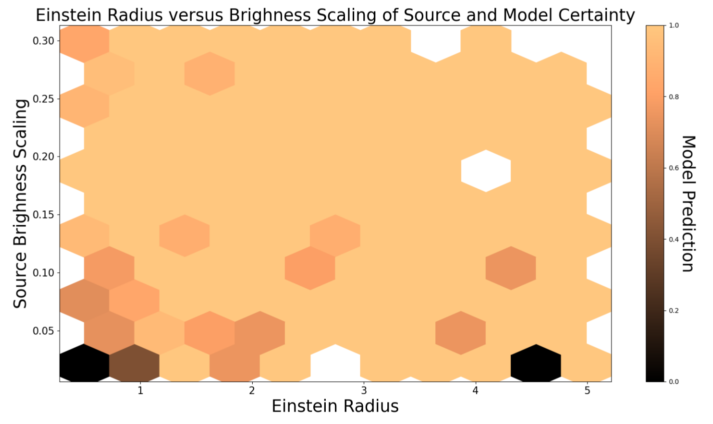


Figure 7.27: Model: $\mathbf{F_1}$ **Soft Loss**. Model certainty plotted versus Einstein Radius and source brightness. Each hexagon contains a set of data points, which are averaged in order to obtain the color magnitude of the hexagon.

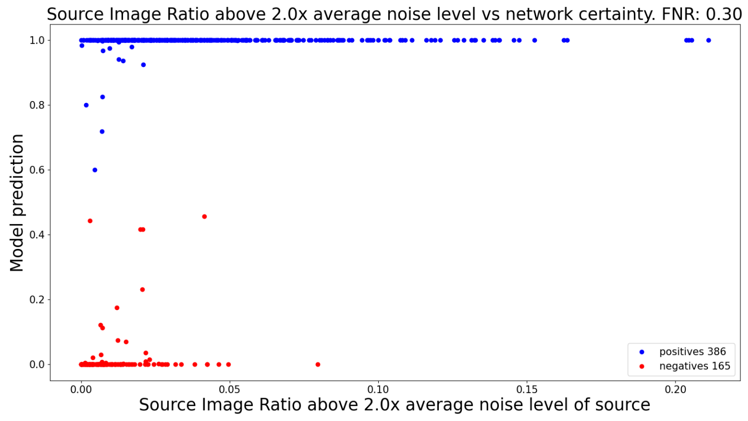


Figure 7.28: Model: $\mathbf{F}_\beta$ **Soft Loss**. Root-Mean-Square (RMS) noise level estimation. The x-axis represents the fraction of the source image above 2x the RMS estimation of the noise. The decision threshold is set to 0.5.

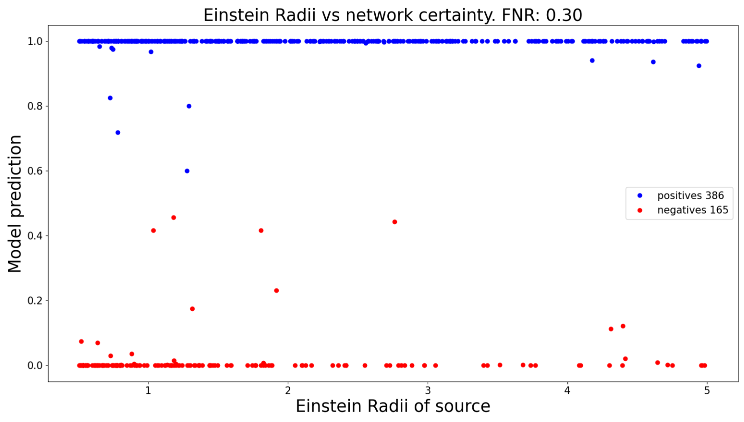


Figure 7.29: Model: $\mathbf{F}_{\beta}$ **Soft Loss**. Model certainty plotted versus the Einstein Radius of the Source image.

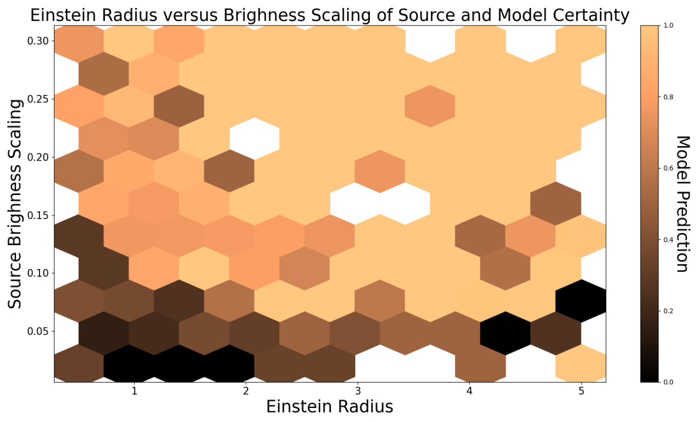


Figure 7.30: Model: $\mathbf{F}_{\beta}$ **Soft Loss**. Model certainty plotted versus Einstein Radius and source brightness. Each hexagon contains a set of data points, which are averaged in order to obtain the color magnitude of the hexagon.