
UNIVERSITÄT
LEIPZIG

Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

# A Study on the Impact of Class Imbalance on CNNs for Bee Health Detection

Bachelor thesis

vorgelegt von:
Noah Rasp

Matrikelnummer:
3738544

Betreuer:
Prof. Dr. Erhard Rahm
Lucas Lange

## Abstract

This work covers the implications of class imbalance on a Convolutional Neural Network designed for bee health detection. With domestic bee populations facing a diverse range of possibly deadly conditions, a computer-based warning system would improve their survivability by reducing monitoring overhead for beekeepers. Class imbalance is a common issue in health-related datasets, with healthy examples being abundant, while examples of disease are more uncommon. Such imbalance tends to skews the performance of a neural network towards the healthy examples. This thesis studies the effect of this problem and compares multiple possible solutions to it. A CNN for bee health detection was constructed on a public dataset, and the approaches of oversampling, undersampling and cost sensitive classification together with their combinations were compared. The experiments showed oversampling performs very well, although its performance comes with the large downside of overfitting. Cost sensitive classification boosts the minority classes, while losing some performance on the majority classes with no visible downside. Overall, it produced the most desirable outcome, with the option to better the performance of another technique by working in conjunction with it. Undersampling falls short on this dataset due to the large amount of lost information hampering its performance.

Noah Rasp
3738544

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Bees are vital for our Earth's natural ecosystem [1]; by facilitating pollination, they ensure the reproduction of a wide range of plant species. The studies of [2] show that western honey bees(Apis mellifera) alone account for 13% of all registered floral visits, and 5% of accounted for plant species were exclusively visited by them. Furthermore, bees are important for agriculturally relevant plants such as strawberries, sunflowers and cole crops [1]. Thus they contribute to preserving our quality of life. In recent times native bee populations have seen a worrying decline, which can be attributed to a wide variety of causes for example climate change, habitat loss, invasive species, pesticides and parasites [3]. This can be combated for domestic bee colonies, but it requires extensive monitoring by a beekeeper. As [4] mentions, beekeepers can lessen the impact of parasites or diseases through various means, but this can only come into effect if the problem was detected. The digitization of this monitoring process could yield several benefits, including workload reduction on the side of the beekeeper, earlier detection of parasite or fungal infestations and reduction of human-related stress for the bees. For this kind of use case a neural network is well-suited, since it can extract the features from the data itself. Specifically, Convolutional Neural Networks (CNN) are the current industry standard in image recognition, with most top scores in competitions such as ImageNet Large Scale Visual Recognition Challenge 2017 [5] containing some form of CNN. However, with this performance comes the requirement of a sufficiently sized dataset [6]. Especially in health-related data, it is quite common that there are ample examples of healthy classes, but a deficient amount of diverse examples for the diseases. This can lead to either overfitting, or the network not learning from the disease data at all. Solving this would enable the network to perform better in real scenarios, since the cost of misclassifying as healthy are way higher than a false ill diagnosis. This class imbalance can be addressed by a wide variety of methods, which can be classified into either algorithm level adjustments or data level methods. [7] split data related approaches into oversampling or undersampling. Oversampling refers to the replication of examples of the minority classes, until the class distribution is balanced. Meanwhile, undersampling involves removing data from the majority class, to achieve a balanced dataset. As for algorithm level methods there is cost-sensitive classification, which adjusts the loss function in training by multiplying the output with a cost matrix, in [7] it is referred to as threshold moving. There are also combinations of the approaches from both categories. This list is by no means complete, as the issue is a long-standing one and many different proposals to its solution were made. The showcased methods are rather elementary and intuitively understandable, thus their impact on the classifier shows what can be done to alleviate class imbalance with simple methods. The next section covers related work, after which background information on the topic is given. Then the conducted experiments are presented and their results are discussed, ending with a conclusion.

# 2. Related Work

Bees being eusocial insects, combined with their importance for the human condition, places them as a topic of interest within the scientific community. There is a wealth of bee-related AI research, with one example being the research into markerless tracking of honey bees in a hive, presented in [8]. Tracking bee health via a neural network supported architecture is another common application, although there is a wide range of such research the concrete attributes being looked for are quite diverse. An interesting example is [9], where the sound profile of the beehive is being analyzed to detect the presence or absence of a queen bee. The authors achieve this by testing a diverse set of audio feature extraction techniques, with the resulting sound profiles then being fed into a CNN for classification. The most common approach is using bee images to train a CNN for classification. Such images mostly come from live video feeds of bee hives and are either labelled and segmented semi-automatically or by hand to ensure the quality. Even with this procedure, there are different attributes available as indicators for bee health. This can directly be designated as the target. Another possible target is the pollen bearing of bees, which is a sign of the health of the bee hive. [10] builds a CNN used to discern if bees are carrying pollen, and as mentioned in the paper, the presence and amount of bees bringing pollen is important for the survival of the hive, as pollen is used to feed the larva. Other works specifically build a CNN for certain disease types such as [11], which focuses on the varroa mite. This paper also introduces a novel segmentation approach, which locates the mite on the body of the bee via semantic segmentation. Certain publications select multiple diseases for a multiclass classification problem. A fitting dataset for such a publication would be [12] from Kaggle, as it includes 5172 images across 6 classes. However it contains major class imbalance, with the majority class making up around 65.4%, while the minority class only contributes 0.56% of the dataset. [13] examines the impact of mathematical morphology and image filters on the accuracy of a CNN, using [12] as its dataset. There are no mentions of class imbalance in the paper, so it is safe to assume there were no adjustments in that regard. Furthermore, the metrics used in the paper are not split up into the different classes, and thus the model performance with regard to the minority class is unknown. The baseline model accuracy was reported to be close to 85%, due to the real-time requirement of their work, a smaller model is used to achieve a faster processing time. Furthermore, they performed hyperparameter optimization to achieve a final accuracy of 95.2%. As a second objective, the authors constructed a segmentation neural network, which crops bees out of an image, thus conceptualising a pipeline for a complete bee health monitoring system using neural networks. This pipeline would take pictures from a live video feed, cropping the bees out via the segmentation neural network and then inputting them into the health classification network to arrive at an output. Another work using this dataset is [14], which focuses on using pre-trained models via transfer learning to try and achieved optimal performance. The two different networks of VGG-16 and DenseNet-121 are being compared in their performance on this dataset. Class imbalance is addressed by removing the minority class from the dataset and introducing data augmentation such as flipping and rotations of the images. Another measure used to combat imbalance is stratifying the training and test sets according to the original class distribution. The author paid some attention to the outcome in relation to different classes, as there are F1-scores and confusion matrices to further split the result, showing the performance of the model in regard to singular classes. [15] is an interesting baseline notebook available on

Kaggle. It implements CNNs for both possible targets within the datasets, namely bee subspecies and health. There are also useful visualisations for the way a CNN works, with pictures detailing the different transformations an image goes through in the network. Data augmentation was performed to prevent overfitting. To address the imbalance, undersampling in conjunction with oversampling was employed to achieve a sample size of 500 images per class. The sampling process was performed on a random basis. The result includes a detailed breakdown of the performance for each class. Class imbalance is a well-researched topic with a long history in regard to machine learning as a whole. The oldest freely available paper referencing sampling techniques and the class imbalance problem is [16], dating back to 2000, but it is safe to assume that studies go back to at least the 1990s. The wealth of research even warrants papers to provide an overview of the available methods such as [17]. For CNNs specifically, a lot of techniques are harder or impossible to implement, as image data is not as easily modified as data points are. Methods to counter imbalance within CNNs were already compared once in [18], although thresholding was used instead of cost sensitive classification. The previous paper does not include a potentially important tool for combating class imbalance in data augmentation. In [19] data augmentation was utilized to create variations of existing data for low population classes, specifically, noise was added to the pictures. There is a wide variety of possible transformations, and the application thereof can increase the robustness and overall performance of the classifier. The available transformations for the kind of image data being used here include random amounts or enabling of cropping, zooming, flipping horizontally or vertically, brightness or contrast changes and rotating.

# 3. Background

The following section will provide the necessary knowledge and terminology to understand the context of the experiments conducted in this work. It is recommended to read this chapter in the case of an incomplete understanding of the basic concepts of machine learning and more specifically computer vision via neural networks.

## 3.1. Basic concepts and terminology

A dataset used for machine learning consists of the data or input itself, which could take many forms, including but not limited to pictures, numerical values or nominal attributes. For each such datapoint there exists a label commonly referred to as output [20]. This label refers to the class the data point belongs to in the case of classification [21], which is the topic of interest here. The process behind machine learning is using the available data to train a model to accurately predict the outcome based on a certain input [21], for classification it would be the correct target class, with the resulting model's ability to predict unseen data potentially varying depending on the data and techniques used in its creation. A phenomenon that adversely affects this ability is the so- called overfitting. Overfitting is the inability of a model to perform well on unseen data, even though the performance on the training set was high, as defined in [22]. This effect can be visualised by splitting

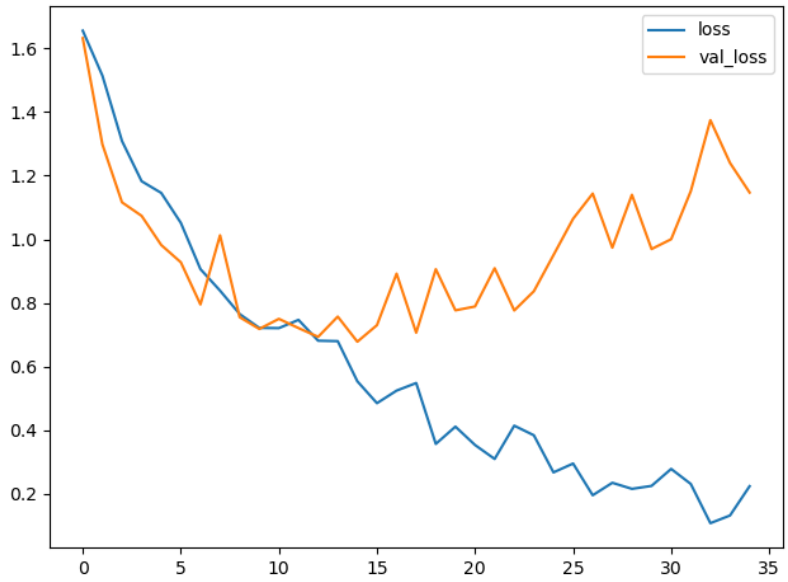



Figure 3.1.: Showcase of overfitting

the dataset into different sections: a training set and a validation set. The training set is used for training ,and after each iteration, also known as an epoch, the current model is tested against the validation set. This is done to observe if the performance of the classifier on the unseen validation set negatively deviates from the performance development on the training set, as this would be a clear sign of overfitting. Figure 3.1 shows the increase of the validation loss, referred to as val_loss compared to the improvement of the training loss depicted as loss. loss is one of the metrics used

to judge the quality of a model. loss refers to the output value of the loss function, which can be how close the model's result was to the actual solution; in theory, the loss of a completely correct answer would be 0. This in turn turns loss machine learning into an optimization problem [23], with loss being the variable to minimize. There is a wide variety of different available loss functions each with their respective upsides, and the choice of the loss function directly impacts the resulting model. For multiclass classification, categorical cross entropy is a common choice. [24] presents a list of other popular classification metrics such as accuracy, which is defined in said paper as dividing the amount of correctly classified examples by the amount of all examples. Accuracy alone is not indicative of the performance of the classifier regarding a specific class, for this purpose two new metrics are introduced, namely precision and recall. Recall is calculated as the division of the number of correctly classified examples of a target class over the number of examples of said class. Recall penalises the classifier for not correctly classifying examples of the target class, but does not consider the number of examples from other classes wrongly classified as the target class. Precision is defined as the number of correctly classified examples of a target class divided by the number of examples classified as said class by the model. Precision discourages examples to be wrongfully classified as the target class. Recall and precision cover entirely different aspects of class performance of a classifier, calculating the harmonic mean of these 2 metrics leads to the F1-score, which is a competent benchmark for the single class performance of a classifier.

## 3.2. Structure of a CNN

A CNN adds two new types of layers to the conventional neural network repertoire: these are the eponymous convolutional layer and the pooling layer. Commonly the structure of a CNN consists of a number of convolutional layers each followed by a pooling layer, resulting in a structure referred to as the base of the network [25]. The outcome of this base is a number of feature maps. To feed them into the classification part of the network flattening is used to turn this multidimensional input into a one dimensional array, which can then be used by the dense layers for classification [26]. Typically an architecture-dependent number of fully-connected layers is used for the final classification, which is reminiscent of a standard neural network [27].

### 3.2.1. Convolutional Layer

The functionality of the convolutional layer is based on filters also known as kernels, which are matrices that manipulate the pixel values of an image that they are applied to. Figure 3.2 shows the concept that filters work on. The middle of the kernel matrix is first aligned with a pixel within the image.Following this, each pixel value is multiplied with its respective value within the kernel, after which all values are summed up and the final result is stored in the original middle pixels' position [25]. It is of note that this operation does not impact the original picture, since the resulting values are stored in a new picture as to not skew the following application of the filter. A parameter of this operation is called stride, and it dictates how much the center pixel moves after each application. For example, referring to Figure 3.2, a stride value of 2 would cause the next pixel to not be the 0 to the right of the current pixel, but rather the 1 which is 2 steps to

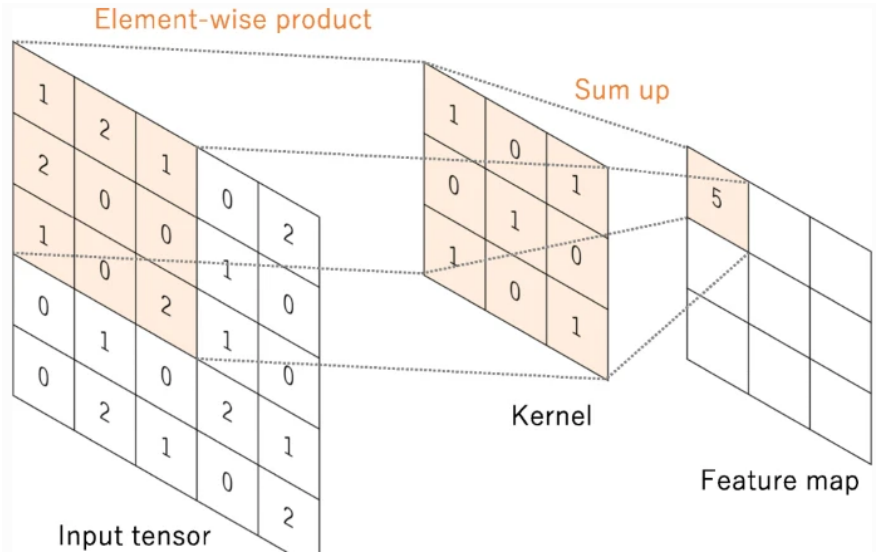
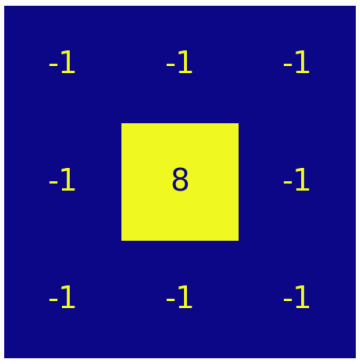



Figure 3.2.: The Image stems from [25] and is licensed under the Creative Commons license [1].

the right [28]. The default value for the stride is one. The size of the filter is variable, although uneven values are preferred, since this allows a symmetric alignment with the middle pixel, thus 3x3, 5x5 and 7x7 are typical [25]. Another important parameter is the padding, which decides how the filters deal with the edges of the image where the kernel would face unavailable values. The two most common options are valid padding and zero padding. Valid padding discards these edge cases, and thus reduces the image size with each convolution [29]. Zero padding fills the missing values with zeros, keeping the dimensions of the image the same after the convolution [28]. The last customisable parameter is the number of filters the CNN learns for this convolutional layer. Figure

(a) Base Picture



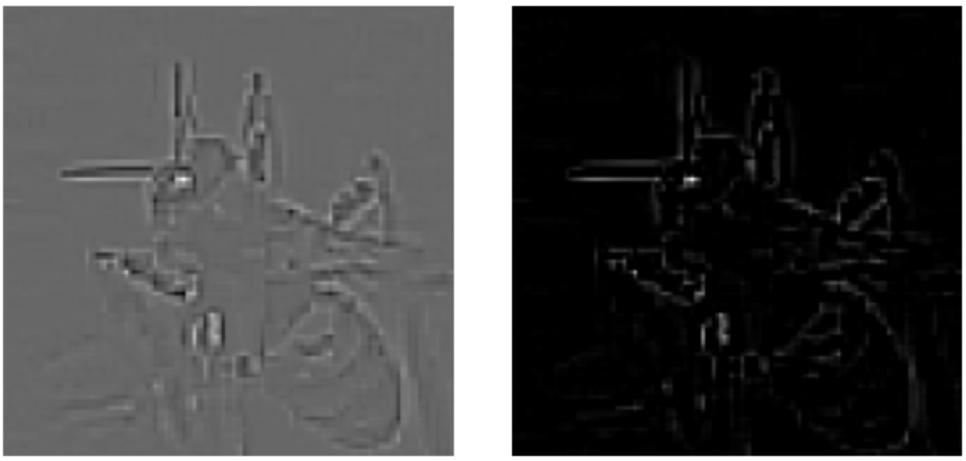

(b) Filter used

Figure 3.3.: Data used for the example

3.3.a shows the original picture for an example regarding image filters, while 3.3.b is the kernel used for the convolution. This kernel is the so-called ridge detection, which is a quite common tool used for creating outlines of a picture. It is important to note that the filters used in a CNN are learned in the training phase [28]. These learned filters, especially in later convolutions, often look

[1] http://creativecommons.org/licenses/by/4.0/

rather indecipherable to a human, and so only the context is explained. Figure 3.4.a shows the



(a) Filter applied on Base Image (b) Relu applied to filtered Image

Figure 3.4.: Filter example

outcome of the kernel. It can be noted that there is a large amount of grey space present. This is due to very small negative values appearing on the image, as zero values would be shown as black. The configuration used for this convolution are a stride of one and valid padding. To introduce non-linearity, an activation function is generally on the output of the convolution, with the most common examples being sigmoid, tanh and relu. Of these, Relu is the preferred choice in recent times due to a simpler gradient, better performance on deep structures and the attribute of setting all negative numbers to 0, thus creating an easier depiction of the relevant areas [28].

### 3.2.2. Pooling

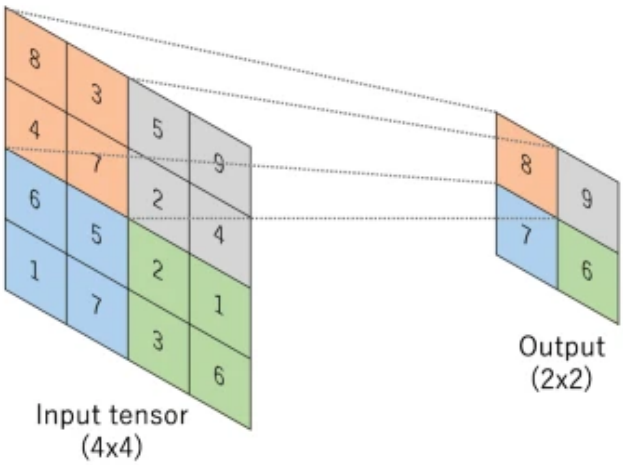



Figure 3.5.: The Image stems from [25] and is liscensed under the Creative Commons liscense.

After the convolution layer, there is usually a pooling layer. The most commonly used type of pooling is max pooling, but other types such as average pooling exist. Pooling reduces the size of the filter maps, which decreases the computational requirement of subsequent layers due to less parameters being present, while also establishing translational invariance for the classifier [25]. Max

pooling shares most parameters with the convolutional layer, so that stride and padding do not need to be introduced again. Regarding window size, there arises the difference that uneven numbers are not preferred, rather the most common one is 2x2 [28]. Figure 3.5 depicts an example of the max pooling process. The maximum value of the pooling window is selected to be put into the new image. Generally the pooling window and stride are aligned so that pooling windows do not overlap in any position, but in some cases overlapping can increase the performance of the classifier [28].
Figure 3.6 depicts the application of pooling onto the previously shown filtered image after the

Figure 3.6.: Pooling the Example from Figure 3.4

activation function. The parameters selected were a pooling window of 2x2 and a stride of 2, resulting in no overlap, as valid padding was chosen. The image is significantly more pixelated, but it still retains all features present in Figure 3.4.b, while having reduced dimensions which bring the previously mentioned upsides.

## 3.3. Bee diseases

This section will briefly describe some of the possible diseases that are found within the datset used here. The varroa mite is a parasite which only recently switched host from the eastern honey bee to the western honey bee, which is muss less resistant to the parasite as its eastern counterpart [30]. Varroa mite infections are often fatal for their hosts and can even contribute to the destruction of a hive by placing additional stress on the colony. Another dangerous species for bee populations are small hive beetles, which began to spread worldwide from their native habitat in sub-Saharan Africa in recent times [31]. These hive beetles infest bee colonies and steal their food sources, and using the hive to protect themselves from environmental hazards [31]. This invasion process often causes destruction of the hive, but an early detection can reduce the chances of this with the intervention of a beekeeper. A different danger to bee hives are hive robberies. Carried out by bees from other hives in times of scarce food, this is often a cause for the collapse of a hive[32]. A vital factor for the survival of a bee hive is their queen, as the colony is solely reliant on her for reproduction. A hive losing their queen without a fast replacement is certain to die [33]. Early detection can help a Beekeeper insert a replacement queen to attempt to rescue the hive.

# 4. Experiments

## 4.1. Setup

All experiments were performed on an AMD Ryzen 7 5800X 8 core CPU at 3801 MHz, provided with 16 GB of RAM. The code and a list of necessary libraries are accessible via a gitlab repository[2]. For the sake of repeatability, it was important to seed all elements of randomness within the project. It became apparent that it was necessary to seed all elements separately, as simply setting the TensorFlow seed does not impact random layers such as Dropout or RandomFlip.

## 4.2. Data

As previously stated the dataset is taken from [12]. It is important to note that this dataset can change over time if the creator chooses to add more pictures or even classes[3]. It is comprised of 5172 RGB images and a CSV file, which assigns attributes to each image, such as health, location, time, date and subspecies. For the task at hand, only the health attribute is of interest, which will be used to create a classifier, but the distribution of those classes is heavily imbalanced.

Table 4.1.: Class names will be abbreviated by their class number, for example healthy is class 4.

| Class | Amount | Percentage of dataset | Class number |
|---|---|---|---|
| varrao, small hive beetles | 472 | 9.1 | 1 |
| ant problems | 457 | 8.8 | 2 |
| few varrao, hive beetles | 579 | 11.2 | 3 |
| healthy | 3384 | 65.4 | 4 |
| hive being robbed | 251 | 4.9 | 5 |
| missing queen | 29 | 0.5 | 6 |

As seen in Table 4.1, the least represented class only accounts for 0.5% of the entire dataset, which is insufficient to build a reliable classifier, considering the total amount of images is not very large either. The data is processed by first reading in the CSV file, then loading all pictures in respect to the CSV. All pictures are decoded from the PNG format to a tensor of floating point values. Afterwards, the image gets resized to standardize the input size to 128 pixels by 128 pixels. Through all these transformations a tensor of the shape (128,128,3) is acquired. The data is then split into training, validation and test set with a ratio of 70%/20%/10%. These sets are stratified, preserving the relative class distribution from the original dataset. Each time an image from the training set is fed into the neural network, it goes through two additional transformations, being a random contrast change up to one and a random horizontal flip. This is used to help against overfitting and to generalise the classifier.

[2] https://git.informatik.uni-leipzig.de/nr27lomi/Bachelorarbeit
[3] For reference the dataset was downloaded on the 15th of June 2022

## 4.3. Baseline

The baseline is the point of reference for all experiments conducted, as it presents a typical CNN that might be built to solve such a problem. This means that it does not include any adjustments to address the issue of class imbalance.

### 4.3.1. Architecture

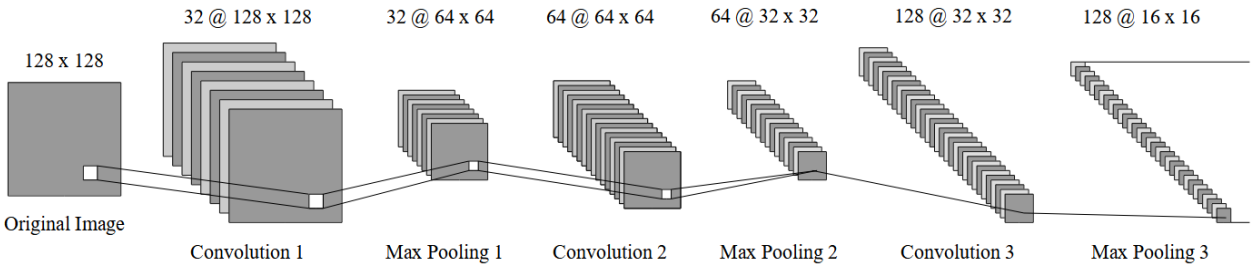



Figure 4.1.: Model Base of the CNN

Figure 4.1 shows the structure of the model base, which deals with extraction of the features that are later used for classification. It contains three convolutional layers and three max pooling layers. Each subsequent convolutional layer contains increasingly elaborate filters. Max pooling layers are used to decrease computational effort, while also providing subsequent filters with an increasing scope of the surrounding without increasing filter size [25]. The number of filters in the later convolutional layers increases for a better possible combination of high level features. This increase does not affect the computational requirement much, due to the size reduction of the image caused by the max pooling layers. A flatten layer reduced the dimensionality of its input to

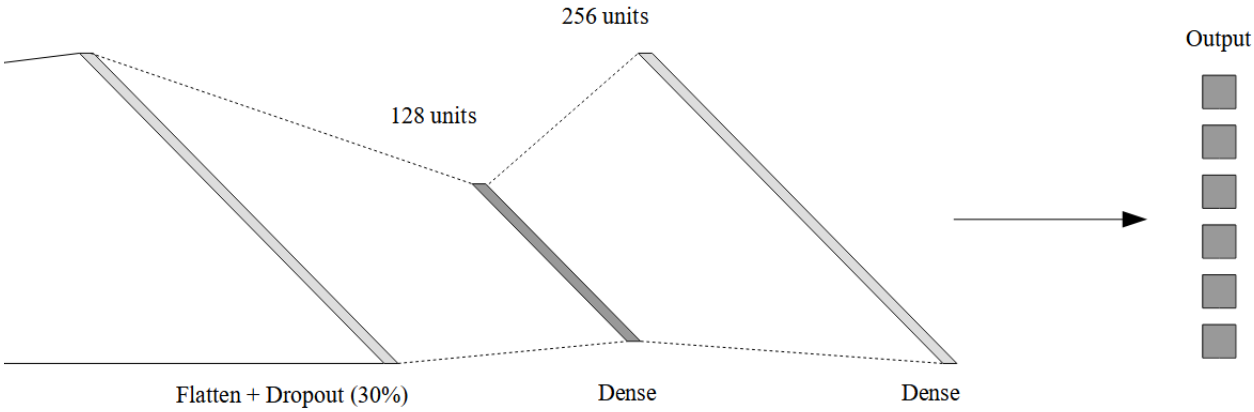


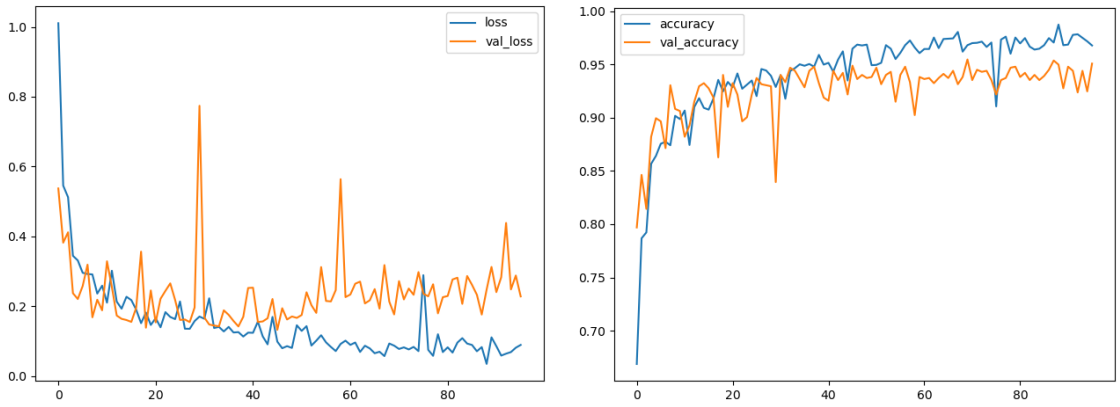

Figure 4.2.: Model Head of the CNN

one, making it processable for the following dense layers. A dropout layer is interposed between flatten and dense layers, to reduce overfitting by randomly disabling input units during each step while training. Dense layers are used to associate the extracted features with the classes, this so called model head, seen in Figure 4.2, learns what features imply which class. The last dense layer is used to determine the final class prediction, using the softmax function for activation. For all other layers relu is chosen as the activation function, due to it being a well-researched industry standard as stated in [34]. The Adam optimizer was selected in this case, based on the research

from [35] showing its high effectiveness, in comparison to other popular optimizers. As for the loss function, categorical cross entropy is widely accepted as the standard for multiclass classification, with [36] providing a deeper explanation of the reasoning for its success. To avoid overfitting, a mechanism called early stopping is implemented. This tracks the best result in terms of validation loss, and after a user defined amount of periods, learning stops, with the option to restore the weights associated with the best validation loss. This method is further explored in the paper [37]. The number of epochs is chosen to be large, since early stopping is used to decide when the model has learned all useful information and terminates the process accordingly.

### 4.3.2. Initial Result and Confusion Matrix

The architecture described above was used to train a classifier. The training history can be seen in Figure 4.3. After Epoch 40 overfitting started as validation loss stagnated, while the training loss continued to increase. Accuracy has a similar development. After the training finished, the previously discussed test set was used to check the ability of the network to generalise on unseen data, producing the result of loss equaling 0.1367 and an accuracy of 95.17%. As a last evaluation the entire data set was predicted by the CNN, resulting in a loss of 0.0524 and an accuracy of 98.11%. All of these numbers do not give any insight into the performance of the network in regard




(a) loss while training

(b) accuracy while training

Figure 4.3.: Training process of the Baseline

to the minority class. To address this, two new metrics were introduced, namely the confusion matrix and F1-score. A confusion matrix displays the network's predictions and the ground truth in a matrix form, with the number of correct predictions for each class along the main diagonal of the matrix. The F1-score is the harmonic mean of accuracy and recall with a maximum value of 1. It is a popular metric for binary as well as multiclass classification [38]. Both of these methods were applied to the network, giving results shown in Table 4.2.

This outcome is rather unexpected since the class imbalance should lead to an inferior classifier for the classes with a lower sample size, yet the F1-score is almost perfect at 0.986 and 0.982 for classes 5 and 6 respectively. To give a quick overview of the performance, a technique called macro-averaged F1-score is applied, which calculates the arithmetic mean of all the single class F1-scores.

Table 4.2.: Confusion matrix and F1-score of the Baseline

| | predicted class | | | | | | |
|---|---|---|---|---|---|---|---|
| actual class | 1 | 2 | 3 | 4 | 5 | 6 | F1-score |
| 1 | 442 | 1 | 29 | 0 | 0 | 0 | 0.916 |
| 2 | 0 | 457 | 0 | 0 | 0 | 0 | 0.999 |
| 3 | 50 | 0 | 522 | 7 | 0 | 0 | 0.921 |
| 4 | 1 | 0 | 3 | 3378 | 2 | 0 | 0.997 |
| 5 | 0 | 0 | 0 | 4 | 247 | 0 | 0.986 |
| 6 | 0 | 0 | 0 | 0 | 1 | 28 | 0.982 |

For this classifier the outcome is 0.967. Macro-averaged F1-score will henceforth be abbreviated as macroF1.

### 4.3.3. K-fold Validation

K-fold cross validation describes the act of splitting the dataset into K equally sized sets, of which K-1 are used to create a classifier, while the remaining one is used to test the classifier's performance [39]. This process is repeated until K classifiers have been created, with all performances on the respective left out sets then averaged. In this work, the folds(sets) were stratified and K was selected to be 5. The number 5 was chosen due to it being close to a division of 29, being the number of samples in the minority class, with one fold receiving an example less. K is relatively small because the left-out fold has a greater number of examples from the minority class and it additionally reduces required computing power. The arithmetic mean of each metric was calculated to see if the prior observations were just a statistical abnormality or if the observed performance of the classifier is accurate. The following values were acquired by predicting the entire dataset after training is completed. The average loss and accuracy are 0.0842 and 96.72% respectively, comparing this to the Baseline values shows that the performance does not deviate too much from the mean. Furthermore the mean F1-scores of classes 5 and 6 are 0.975 and 0.941, which means they did decrease by a considerable amount, but not enough to be called within expectations. Everything considered, the classifier still performs too well, what could be the reason for this deviation from expectation?

## 4.4. Influence of the Background Colour

If a classifier is producing heavily unexpected results, it is common practice to closely examine the underlying dataset. A dataset can have certain flaws that potentially induce strange outcomes. For example, a small watermark is present in every image of a certain class, but not in the data of any other, which skews the performance and perception of the network heavily. After reviewing the dataset in question manually, it became apparent that there exist background objects which consistently appear in examples of classes 5 and 6. In the case of class 6, all pictures feature a very distinct shade of yellow as can be seen in Figure 4.4, which does not seem to appear elsewhere in the dataset. Class 5 has a similar issue with a different shade of yellow.

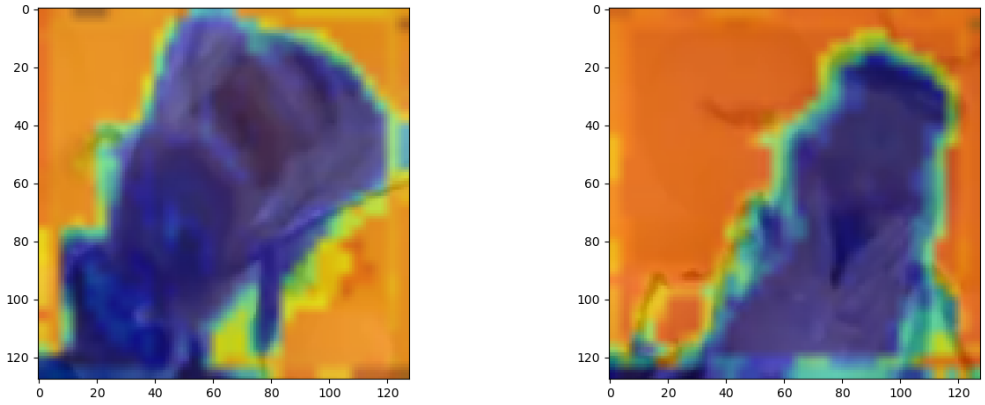
(a) Image number: 034_024

(b) Image number: 034_036

Figure 4.4.: Examples from the missing queen class

### 4.4.1. Grad-CAM

To confirm these suspicions, a technique called Grad-CAM is employed. It has the goal of providing a map, highlighting the areas on a picture which produce the most neuron activations. Grad-CAM achieves this by backpropagating the output of the network, concerning a specific class, to the last convolutional layer. This acquires backpropagated feature maps, which then get globally average pooled and multiplied with the original feature maps. The outcome is scalars that are added together, after which relu is applied to them, creating the desired activation heat map, for further insight [40] provides an in depth explanation. To accommodate for this technique, the architecture has to be changed slightly by replacing the flatten layer with a global average pooling layer. The performance of the model does not significantly get altered. Implementation was modelled after the example provided in [41]. Figure 4.5 shows the result of applying Grad-CAM, proving the



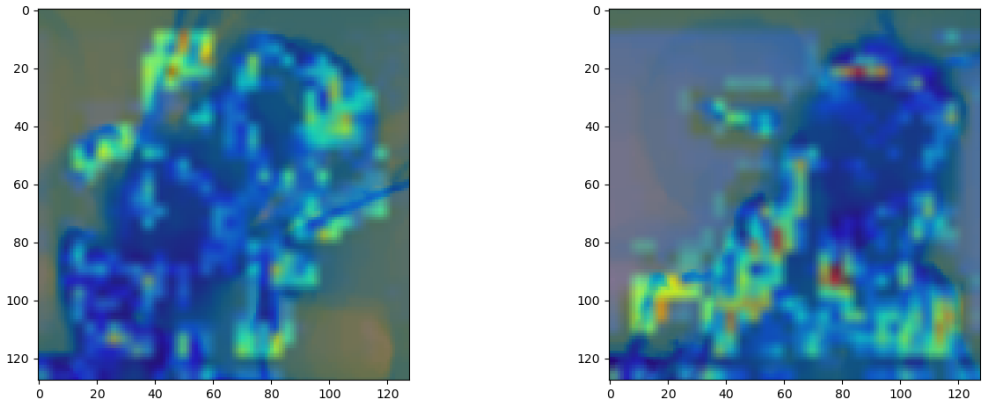

(a) (b)

Figure 4.5.: Grad-CAM on the previously shown examples from class 6

hypothesis of the background being used for classification instead of the actual bee. Heat maps show regions of strong activity in a yellow to red tone and regions of close to no activity in a blue tone. To continue with the study, what can be done to alleviate this glaring flaw in the dataset?

### 4.4.2. Grayscale

Multiple methods to remove this heavy bias were tested, such as brightness and contrast changes, but these did not alleviate it. A more radical approach was chosen to present a quick way of making this dataset useable. The method chosen was grayscaling all images in the preprocessing stage, which removes a large amount of information from the dataset along with a sizeable portion of the bias. Bias was not completely eliminated as can be seen in Figure 4.6, but the activations are now also happening on the bee portion of the image. A better approach could be, to train a segmentation model that crops the bee out of the picture, before it is inputted into the CNN. This concept goes beyond the scope of this work but leaves room for further exploration. This changes



(a) (b)

Figure 4.6.: Grad-CAM for grayscaled previous examples

the performance of the baseline model drastically. F1-scores for the respective classes, along with accuracy and loss values, can be seen in Table 4.3. Overall, the observed values are much closer to expectation than the previous version, especially in regard to the minority classes. The Values in Table 4.3 were measured, when the classifier was tested for the entire dataset after training was complete. These will serve as the baseline values, to be compared against the upcoming methods to tackle class imbalance.

Table 4.3.: Metrics of the grayscaled baseline CNN

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.841 |
| ant problems | 0.925 |
| few varrao, hive beetles | 0.787 |
| healthy | 0.954 |
| hive being robbed | 0.639 |
| missing queen | 0.286 |
| macroF1 | 0.739 |
| loss | 0.266 |
| accuracy in % | 90.58 |

## 4.5. Countering Imbalance

### 4.5.1. Undersampling

Undersampling reduces or eliminates class imbalance by removing data from the majority class. Depending on how imbalanced the dataset is, this could remove a lot of data for training. In the current situation fully balancing this dataset using only undersampling would leave only 29 images in each class, because the minority class only contains 29 pictures. A CNN can hardly operate with this small of a dataset. Another option would be to simply reduce the imbalance by undersampling the majority class, to be closer in number to the minority class. The eliminated pictures were selected randomly from the majority class until the target value of 500 was reached. The number 500 was chosen due to the population of classes 1, 2 and 3 being very close to it, furthermore, class number 3 was also brought down to 500 examples from 579.

Table 4.4.: Metrics for undersampling

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.751 |
| ant problems | 0.906 |
| few varrao, hive beetles | 0.695 |
| healthy | 0.873 |
| hive being robbed | 0.502 |
| missing queen | 0.577 |
| macroF1 | 0.717 |
| loss | 0.539 |
| accuracy in % | 80.82 |

The outcome is presented in Table 4.4, showing that in all aspects undersampling is inferior to the baseline model, apart from the F1-score of class 6, where an increase of around 0.3 can be seen. This lack of performance compared to the baseline classifier possibly stems from the information lost due to undersampling. Even though in most metrics the result appears significantly worse, macroF1 only loses 0.022, which can be attributed to the gain of class 6.

### 4.5.2. Oversampling

Oversampling is the act of resampling the minority class until the class distribution of the dataset is balanced. With how large the imbalance is in this dataset, this will lead to a heavy amount of repeating examples within the dataset, and thus the classifier is quite likely to overfit on those examples. Class number 4 has 3384 pictures compared to the 29 pictures of class 6, meaning that those 29 images are repeated on average 116 times to equalise the numbers. As an experiment, this was tested and it reported a loss of 0.0314 and an accuracy of 99.25%, with F1-scores above 0.95 in every category. This result is skewed, due to excessive overfitting and is most likely not applicable for unseen data. Instead, each class will be resampled to at least 500 examples, if they contain fewer pictures. The Table 4.5 shows the resulting F1-scores after training.

Table 4.5.: Metrics for oversampling

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.818 |
| ant problems | 0.970 |
| few varrao, hive beetles | 0.785 |
| healthy | 0.966 |
| hive being robbed | 0.828 |
| missing queen | 0.951 |
| macroF1 | 0.886 |
| loss | 0.212 |
| accuracy in % | 92.77 |

Both minority classes saw a significant increase in F1-score, especially class 6, as seen in Table 4.5. There is probably a reasonable amount of overfitting for class 6, seeing as the increase is this large. The total performance of this classifier is impressive, it can be seen as a satisfactory increase compared to the baseline. MacroF1 improves heavily aswell, with an increase of 0.147.

### 4.5.3. Cost sensitive Classification

Cost sensitive Classification, sometimes referred to as class weights, is implemented here by using the weight argument in the fit function of the neural network. It essentially changes the weights used in the loss function while training is in progress. The weights are acquired by calculating inverse class frequencies with the scikit-learn library [42]. This is a rather naive approach, with a deep grid search probably being able to find better values. The computational requirements for such an endeavour are rather steep and go beyond the scope of this work.

Table 4.6.: Metrics for class weights

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.797 |
| ant problems | 0.939 |
| few varrao, hive beetles | 0.777 |
| healthy | 0.950 |
| hive being robbed | 0.727 |
| missing queen | 0.780 |
| macroF1 | 0.828 |
| loss | 0.296 |
| accuracy in % | 90.20 |

Table 4.6 shows that class 6 saw a sharp increase in F1-score and 5 received a slight gain. Classes 1 and 3 receeded somewhat, while 2 and 4 remained the same. The overall performance of the model

is very similar to the baseline. This method can be seen as sacrificing some performance elsewhere to boost the ability of the network in predicting the minority classes. The improvement on the minority classes strongly affects the macroF1, causing an increase of 0.089.

### 4.5.4. Undersampling + Class Weights

The class weights were calculated with the adjusted class distribution given by undersampling. In comparison to the results of solely undersampling, this method improves loss, accuracy and

Table 4.7.: Metrics for undersampling combined with class weights

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.838 |
| ant problems | 0.923 |
| few varrao, hive beetles | 0.741 |
| healthy | 0.902 |
| hive being robbed | 0.598 |
| missing queen | 0.563 |
| macroF1 | 0.761 |
| loss | 0.436 |
| accuracy in % | 85.25 |

macroF1. Although there have been large improvements, it still falls short compared to the baseline CNN in every category apart from the F1-score of class 6 and macroF1, as Table 4.7 shows.

### 4.5.5. Oversampling + Class weights

Table 4.8.: Metrics for oversampling combined with class weights

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.884 |
| ant problems | 0.952 |
| few varrao, hive beetles | 0.860 |
| healthy | 0.962 |
| hive being robbed | 0.820 |
| missing queen | 0.921 |
| macroF1 | 0.900 |
| loss | 0.206 |
| accuracy in % | 93.46 |

With the previous success of cost sensitive classification in mind, is it able to enhance the highly performant oversampling even further? Table 4.8 shows improvements to every metric apart from the F1-score of classes 5 and 6, which suffered a slight decrease. This classifier is quite outstanding

when compared it to the baseline, although overfitting remains as a point of concern, as is the case for every method involving oversampling. It also shows the best macroF1 seen so far with 0.9.

### 4.5.6. Oversampling + Undersampling

Table 4.9.: Metrics for oversampled majority classes and undersampled minority classes

| Class | F1-score |
|---|---|
| varrao, small hive beetles | 0.816 |
| ant problems | 0.887 |
| few varrao, hive beetles | 0.747 |
| healthy | 0.874 |
| hive being robbed | 0.546 |
| missing queen | 0.906 |
| macroF1 | 0.796 |
| loss | 0.568 |
| accuracy in % | 82.62 |

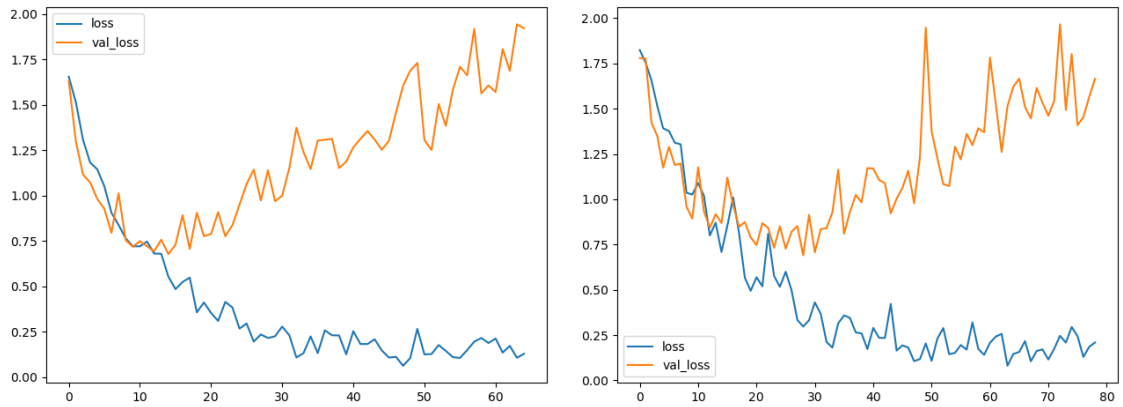
This last method combines oversampling the minority classes and undersampling the majority classes. The outcome is a perfectly even class distribution, where every class has 500 examples in the dataset. The reason why there is no variation with class weights is that in an even class distribution the weight for every class would be one, meaning it does not affect the loss function at all. The performance this method achieves is noticeably worse than baseline in most aspects, apart from the F1-score of class 6 and macroF1 as seen in Table 4.9. Although the overall performance is worse, due to the extreme gain for class 6 the macroF1 increased by a decent margin compared to baseline.

# 5. Results

Table 5.1.: Abbreviations: BL: BaseLine, US: UnderSampling, OS: OverSampling, CSC: Cost Sensitive Classification

| Class specific F1-score | BL | US | OS | CSC | US+CSC | OS+CSC | OS+US |
|---|---|---|---|---|---|---|---|
| varrao, small hive beetles | 0.841 | 0.751 | 0.818 | 0.797 | 0.838 | 0.884 | 0.816 |
| ant problems | 0.925 | 0.906 | 0.970 | 0.939 | 0.923 | 0.952 | 0.887 |
| few varrao, hive beetles | 0.787 | 0.695 | 0.785 | 0.777 | 0.741 | 0.860 | 0.747 |
| healthy | 0.954 | 0.873 | 0.966 | 0.950 | 0.902 | 0.962 | 0.874 |
| hive being robbed | 0.639 | 0.502 | 0.828 | 0.727 | 0.598 | 0.820 | 0.546 |
| missing queen | 0.286 | 0.577 | 0.951 | 0.780 | 0.563 | 0.921 | 0.906 |
| macroF1 | 0.739 | 0.717 | 0.886 | 0.828 | 0.761 | 0.900 | 0.796 |
| loss | 0.266 | 0.539 | 0.212 | 0.296 | 0.436 | 0.206 | 0.568 |
| accuracy in % | 90.58 | 80.82 | 92.77 | 90.20 | 85.25 | 93.46 | 82.62 |

Table 5.1 presents an overview of the experimental results. At first glance, oversampling as well as oversampling + cost sensitive classification stand out as vastly superior compared to the other approaches. In every metric, those two methods are better than every other approach, especially in regard to the F1-score of the minority class "missing queen" also known as class 6. What these numbers do not convey is the possibility of overfitting to the specific attributes of the resampled examples from the minority class. This is difficult to test since there is only data from the same location for class 6 present in the dataset. It is safe to assume that there will be some amount of overfitting present, even though data augmentation steps are in place, as the results for class 6 are almost perfect with an F1-score of 0.951, 0.921 and 0.906. Undersampling overall did not perform well, possibly due to the validation set containing images of the undersampled class 4 that were taken in a different location with different backgrounds and bee species. This is possibly due to two random factors: the undersampling and the separation into training and test set. For both undersampling and undersampling with class weights validation loss worsens rather quickly, while training loss continued to improve, as seen in Figure 5.1.




(a) loss progression of US

(b) loss progression of US + CSC

Figure 5.1.: loss history from approaches including undersampling

All of this could lead to the classifier not being able to fully learn the required features for better results. This could be seen as either something that has to be paid special attention to on the implementation side or as a general weakness of undersampling with regard to losing information from reducing sample size. Undersampling combined with oversampling boasts a strong result for the minority class 6, but it falls short with low general accuracy and comparatively bad performance for the majority class 4. The downsides of both oversampling and undersampling combine for an unfavourable outcome, as the only strong suit of this technique is likely to be overfitted.For undersampling aswell as oversampling the chosen resampling numbers may not be optimal, so there is still space for further optimization. As for the cost sensitive classification, in its standalone form accuracy and loss are similar to baseline, but looking at the F1-scores of class 6 and macroF1 reveals a large improvement. The original goal of a better result for minority classes is fulfilled, without the downsides of overfitting or major performance loss in other areas. Cost sensitive classification also has a strong showing when working in tandem with other methods. In conjunction with undersampling, class weights boost the performance in all metrics by a sizeable amount, but this combination still suffers from lost information due to undersampling, and so remains worse than all options not involving undersampling. Together with oversampling, cost sensitive classification produces the best classifier in the experiment if one were to only look at the metrics of macroF1, loss and accuracy. In terms of the F1-score of the least represented class, it is slightly behind oversampling. It is important to note, that the class weights chosen may not be optimal, since the simple approach of inverse class frequencies was used. Thus, class weights could be a good target for hyper parameter search, further improving the possible results. Overall, cost sensitive classification is a technique with no visible downsides. In a vacuum, it simply shifts the performance of the classifier more towards the minority classes. Furthermore, class weights can work well in conjunction with other techniques to better or shift the performance of the resulting classifier.

# 6. Conclusion

In this work a basic CNN architecture for bee health detection was constructed, studying the effect of the class imbalance present in the public dataset. During the process, a fundamental flaw within the minority classes of said dataset became apparent. The problem pertained to the background colours present in certain classes, leading to the network not deriving judgements from the bees but rather solely from the background colour of the picture. Through applying grayscale to all images, this effect was significantly reduced, as confirmed by the inspection of Grad-CAM images of the minority classes. After this, three elementary approaches to handle class imbalance and their combinations were tested against the newly established baseline. These approaches are oversampling, undersampling and cost sensitive classification. Additionally, all non trivial combinations were tested to attempt to further improve results. The outcome suggests that undersampling underperforms quite heavily in a dataset with large class imbalance such as the one used in this study, while oversampling performs well, due to the large imbalance that must be combated, resulting in a high risk of overfitting. Cost sensitive classification on the other hand works well in conjunction with other approaches and enhances their performance. As a standalone, it slightly shifts the performance away from the majority class and onto the minority classes, resulting in a more desirable classifier from the viewpoint of this work. The bias of the background is not fully removed; in future work this could be addressed by training a segementation network to crop the pictures even further before they are used for training of the classification CNN. Another approach could be generating synthetic training data with varied backgrounds via a Generative Adversarial Network (GAN). Depending on the quality of the created images this could address the background issue aswell as the class imbalance present in the dataset.

# Bibliography


[1] V. J. Tepedino. "THE IMPORTANCE OF BEES AND OTHER INSECT POLLINATORS IN MAINTAINING FLORAL SPECIES COMPOSITION". In: *Great Basin Naturalist Memoirs* 3 (1979), pp. 139–150. ISSN: 0160239X, 23312742. URL: http://www.jstor.org/stable/23376607 (visited on 08/10/2022).

[2] Keng-Lou James Hung et al. "The worldwide importance of honey bees as pollinators in natural habitats". In: *Proceedings of the Royal Society B: Biological Sciences* 285.1870 (2018), p. 20172140. DOI: 10.1098/rspb.2017.2140. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspb.2017.2140. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspb.2017.2140.

[3] Rachael Winfree. "The conservation and restoration of wild bees". In: *Annals of the New York Academy of Sciences* 1195.1 (2010), pp. 169–197. DOI: https://doi.org/10.1111/j.1749-6632.2010.05449.x. eprint: https://nyaspubs.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1749-6632.2010.05449.x. URL: https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-6632.2010.05449.x.

[4] Giorgio Sperandio et al. "Beekeeping and honey bee colony health: A review and conceptualization of beekeeping management practices implemented in Europe". In: *Science of the Total Environment* 696 (2019), p. 133795.

[5] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[6] Rohit Keshari et al. "Learning Structure and Strength of CNN Filters for Small Sample Size Training". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9349–9358. DOI: 10.1109/CVPR.2018.00974.

[7] Shaza M Abd Elrahman and Ajith Abraham. "A Review of Class Imbalance Problem". In: *Journal of Network and Innovative Computing* 1.2013 (), pp. 332–340.

[8] Katarzyna Bozek et al. "Markerless tracking of an entire honey bee colony". In: *Nature communications* 12.1 (2021), pp. 1–13.

[9] Alessandro Terenzi et al. "Comparison of Feature Extraction Methods for Sound-Based Classification of Honey Bee Activity". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), pp. 112–122. DOI: 10.1109/TASLP.2021.3133194.

[10] Tomyslav Sledevič. "The Application of Convolutional Neural Network for Pollen Bearing Bee Classification". In: *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*. 2018, pp. 1–4. DOI: 10.1109/AIEEE.2018.8592464.

[11] Stefan Schurischuster and Martin Kampel. "Image-based Classification of Honeybees". In: *2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 2020, pp. 1–6. DOI: 10.1109/IPTA50016.2020.9286673.

[12] Jenny Yang. "The BeeImage Dataset: Annotated Honey Bee Images". In: (). URL: https://www.kaggle.com/datasets/jenny18/honey-bee-annotated-images (visited on 08/19/2022).

[13] Diogo Braga et al. “An intelligent monitoring system for assessing bee hive health”. In: *IEEE Access* 9 (2021), pp. 89009–89019.

[14] Shruti Chawane. “Image based bee health classification”. MA thesis. University of Twente, 2022.

[15] Dmitry pukhov. “Honey Bee health detection with CNN”. In: (). URL: https://www.kaggle.com/code/dmitrypukhov/honey-bee-health-detection-with-cnn (visited on 09/14/2022).

[16] Nathalie Japkowicz et al. “Learning from imbalanced data sets: a comparison of various strategies”. In: *AAAI workshop on learning from imbalanced data sets*. Vol. 68. AAAI Press Menlo Park, CA. 2000, pp. 10–15.

[17] D Ramyachitra and Parasuraman Manikandan. “Imbalanced dataset classification and solutions: a review”. In: *International Journal of Computing and Business Research (IJCBR)* 5.4 (2014), pp. 1–29.

[18] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. “A systematic study of the class imbalance problem in convolutional neural networks”. In: *Neural networks* 106 (2018), pp. 249–259.

[19] Juan F Ramirez Rochac et al. “A data augmentation-assisted deep learning model for high dimensional and highly imbalanced hyperspectral imaging data”. In: *2019 9th International Conference on Information Science and Technology (ICIST)*. IEEE. 2019, pp. 362–367.

[20] Amandalynne Paullada et al. “Data and its (dis) contents: A survey of dataset development and use in machine learning research”. In: *Patterns* 2.11 (2021), p. 100336.

[21] G. Kesavaraj and S. Sukumaran. “A study on classification techniques in data mining”. In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. 2013, pp. 1–7. DOI: 10.1109/ICCCNT.2013.6726842.

[22] Xue Ying. “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168 (Feb. 2019), p. 022022. DOI: 10.1088/1742-6596/1168/2/022022. URL: https://doi.org/10.1088/1742-6596/1168/2/022022.

[23] Qi Wang et al. “A comprehensive survey of loss functions in machine learning”. In: *Annals of Data Science* 9.2 (2022), pp. 187–212.

[24] Alaa Tharwat. “Classification assessment methods”. In: *Applied Computing and Informatics* (2020).

[25] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into imaging* 9.4 (2018), pp. 611–629.

[26] Nyoman Abiwinanda et al. “Brain tumor classification using convolutional neural network”. In: *World congress on medical physics and biomedical engineering 2018*. Springer. 2019, pp. 183–189.

[27] Keiron O’Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. DOI: 10.48550/ARXIV.1511.08458. URL: https://arxiv.org/abs/1511.08458.

[28] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[29] Aldi Wiranata et al. "Investigation of Padding Schemes for Faster R-CNN on Vehicle Detection". In: *2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*. 2018, pp. 208–212. DOI: 10.1109/ICCEREC.2018.8712086.

[30] Le Conte, Yves, Ellis, Marion, and Ritter, Wolfgang. "Varroa mites and honey bee health: can Varroa explain part of the colony losses?*". In: *Apidologie* 41.3 (2010), pp. 353–363. DOI: 10.1051/apido/2010017. URL: https://doi.org/10.1051/apido/2010017.

[31] Peter Neumann, Jeff S Pettis, and Marc O Schäfer. "Quo vadis Aethina tumida? Biology and control of small hive beetles". In: *Apidologie* 47.3 (2016), pp. 427–466.

[32] Rahman Tashakkori, Gurney B. Buchanan, and Luke M. Craig. "Analyses of Audio and Video Recordings for Detecting a Honey Bee Hive Robbery". In: *2020 SoutheastCon*. 2020, pp. 1–6. DOI: 10.1109/SoutheastCon44009.2020.9249684.

[33] Robert Brodschneider et al. "Preliminary analysis of loss rates of honey bee colonies during winter 2015/16 from the COLOSS survey". In: *Journal of Apicultural Research* 55.5 (2016), pp. 375–378. DOI: 10.1080/00218839.2016.1260240.

[34] Johannes Lederer. "Activation functions in artificial neural networks: A systematic overview". In: *arXiv preprint arXiv:2101.09957* (2021).

[35] Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. "Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers". In: *Plants* 9.10 (2020), p. 1319.

[36] Ahmet Demirkaya, Jiasi Chen, and Samet Oymak. "Exploring the role of loss functions in multiclass classification". In: *2020 54th annual conference on information sciences and systems (ciss)*. IEEE. 2020, pp. 1–5.

[37] Lutz Prechelt. "Early stopping-but when?" In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[38] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. *Thresholding Classifiers to Maximize F1 Score*. 2014. DOI: 10.48550/ARXIV.1402.1892. URL: https://arxiv.org/abs/1402.1892.

[39] Juan D Rodriguez, Aritz Perez, and Jose A Lozano. "Sensitivity analysis of k-fold cross validation in prediction error estimation". In: *IEEE transactions on pattern analysis and machine intelligence* 32.3 (2009), pp. 569–575.

[40] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[41] François Chollet. *Grad-CAM class activation visualization*. URL: https://keras.io/examples/vision/grad_cam/ (visited on 08/29/2022).

[42] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

# Erklärung

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

*A Study on the Impact of Class Imbalance on CNNs for Bee Health Detection*

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 26.09.2022

_____

Noah Rasp