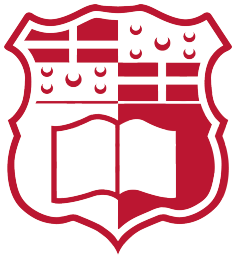


Exploring How Weak Supervision Can Assist the Annotation of Computer Vision Datasets

Enhancing image annotation workflows via CAMs

Andrea Abela

Supervised by Dr Dylan Seychell

Co-supervised by Mr Mark Bugeja

Department of Artificial Intelligence

Faculty of ICT

University of Malta

October, 2022

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.

WWW.UM.EDU.MT

First edition, Saturday 8th October, 2022

To my family and friends

For supporting me throughout this monumental milestone in my life

Acknowledgements

I would like to thank my supervisors, Dr Dylan Seychell and Mr Mark Bugeja, for providing me with unparalleled guidance throughout this study. I would also like to thank all the participants who participated in this study; it would be an understatement to say that their contributions were important. Finally, I would also like to express my gratitude to my parents and friends, as this study would not have been realised without their support.

Abstract

Current artificial intelligence (AI) workflows depend on researchers performing laborious annotation work. In the case of computer vision (CV), annotators would need to produce bounding boxes and labels in tasks requiring localisation. Since this process needs to be iterated over thousands of images, this can possibly cause low-quality annotations to emerge. Techniques like crowdsourcing can mitigate the effects of this issue, but can still possibly yield a sub-par dataset due to the inexperience of the annotators on either AI or the application itself. However, through existing weakly supervised frameworks that utilise techniques like heuristic rules and other notable methods, the general public can provide even more trustworthy annotations.

This paper proposes a basis for an image dataset annotator helper that combines a weakly supervised technique known as class activation maps (CAMs) with convolutional neural networks (CNNs). This produces weakly supervised object localisers that could further improve human image annotation performance. In addition to this, surveys were carried out to extract information that helped either create primary data or verify the results further.

Comparing these models with primary crowdsourcing data revealed that the models can annotate better than humans by 9.7% when measuring the localisation error (LE) while taking into account both false positives (FPs) and false negatives (FNs). Moreover, the models can also save up to 36% of the time required to perform manual image annotation. This confirms that there is potential within CAM-empowered models to further improve the image annotation experience.

Contents

List of Figures

List of Tables

List of Abbreviations

1

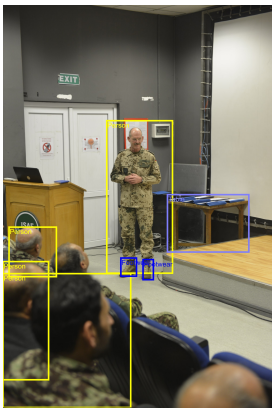
Introduction

Artificial intelligence (AI) development can sometimes prove to be quite a complex assignment. This is due to multiple factors such as its mathematical basis and intricate high-level processes. Such factors are required because of AI's primary element - data. Indeed, there is a myriad of techniques to preprocess data for AI [1]. In computer vision (CV), for example, image classification requires image-level labels for each image [2]. Meanwhile, object detection requires a set of bounding boxes, each containing their own label instance [3]. Furthermore, AI performance usually scales with the amount of preprocessed input data available. Consequently, these processes can exponentially grow in terms of both time and effort. This can discourage researchers and annotators, possibly reducing the overall quality of the dataset.

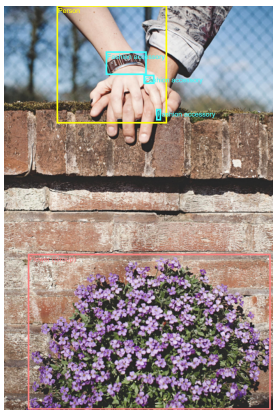
1.1 | Problem Definition

Annotating datasets usually requires a monumental endeavour from a team of annotators. In large-scale annotation projects, the sheer magnitude of the workload can cause the annotations to possibly exhibit inaccuracies or have debilitating limitations. ImageNet, for instance, has multiple identical classes and only one label per image [2, 4]. The Open Images dataset is also prone to mislabelled annotations, even though it is professionally annotated. This is evident in Figure 1.1. In addition to this, there are studies that have improved on these datasets. Indeed, Schumann et al. [6] improved on Open Images' "person" label by adding 100,000 missing instances. This study notes how dataset annotation methodologies could be inconsistent even though they serve similar purposes [7]. Such inaccuracies are usually the consequences of human error; however, other causes could be at fault. The dataset known as Conceptual Captions reported that 97% of its original content was filtered due to the images not being suit-

able for their task [8]. So not only does such a dataset require large amounts of effort to carry out data annotation, but high quality source material may also be scarce. Due to the immense effort required to craft a high quality dataset, researchers have resorted to methods like crowdsourcing to alleviate these issues. Contextually, crowdsourcing is the act of employing the general public as dataset annotators. This would ease the researchers' burden, allowing them to focus on other tasks. Even so, the general public is not a subject expert on either AI or the subject itself. This sparked concerns that the annotations they produced would be subpar. Yet, through the usage of digital aids, crowdsourcing proved to be acceptable [9, 10, 11]. Another instance where digital systems are used to aid annotators involves a method known as weak supervision. In weak supervision, cheap labels are used to help participants with data annotation [12, 13]. Regardless, in weak supervision frameworks, such as Snorkel and Snuba, participants

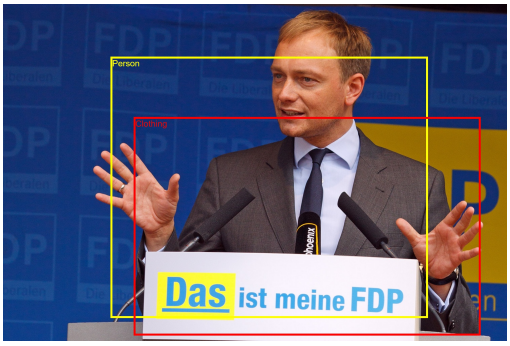


(a) Two men were not annotated



(b) Rightmost person not included

(c) Whole image is a "person" annotation



(d) Inaccurate bounding box

(e) Worker not included

(f) Inconsistent "person" annotations

Figure 1.1: Inconsistent "person" annotations found in Open Images [5]

would still need to learn how to program their conditions [14, 15]. While this method does provide an overall time advantage, this cements the problem that obtaining high quality annotations requires a large amount of effort. This remains true for both the crowd workers and the researchers involved. In CV, there exists a technique known as class activation maps (CAMs). These CAMs highlight model attention when analysing an the result of an inference. While primarily an AI explainability technique, it can be adapted to a weakly supervised learning context. In further detail, it can be used to infer a rudimentary localisation of the target class. Therefore, it has potential as an object localiser, and in turn, as a dataset annotator helper for both regular data annotation workflows and crowdsourcing projects. Further information about this technique can be reviewed in Section 3.3.4.4.

1.2 | Motivation

Granted that data annotation is a very important and lengthy process of AI workflows, it is still considered as one of the most neglected aspects in terms of data quality. This is a significant issue, especially considering that AI is being continuously more integrated in critical applications [16]. Enlisting the usage of tools like crowdsourcing can aid in enhancing the data annotation workflow, both in a user-centric and a data-centric manner. This is especially true in image annotation for CV tasks like object detection, which require painstakingly annotating each individual instance with bounding boxes and labels over thousands of images [2, 5, 8]. Even though crowdsourcing does help mitigate these issues, ideally it would still use some form of digital facilitator to aid the participants in their task. This can be done by implementing some form of dataset annotator helper through the usage of techniques like weakly supervised learning. Aiding the crowdsourcing participants can help in increasing data annotation quality, which in turn, reduces the chances for lacklustre performance in the final AI models [9, 10].

1.3 | Research Question

Among these digital systems, weakly supervised learning has been used for similar purposes as seen in previous studies [13, 15]. Thus, some weakly supervised techniques were explored while comparing their strengths and weaknesses. Ultimately, CAMs were presented as the recommended option as explained in Section 3.3.4.4. With all this known, this study answers the question:

Should humans working on image annotation make use of CAMs to be more effective?

When considering the inner workings of weakly supervised learning in this context, two points arise from this question. Since CAMs only require the user to provide the image-level labels, then both localisation time and weakly supervised accuracy are important. This is because these factors are the most direct approach to measure the weakly supervised technique's effectiveness in this context. If weak supervision does not meet these requirements, then this method would not be ready for this application.

1.4 | Hypothesis

The question presented in Section 1.3 and the arguments discussed in the literature on the subject (see Section 3.3.4.4) indicate that CAMs show potential as a dataset annotator helper. Subsequently, a hypothesis was formulated:

Dataset annotators can save on both time and effort while achieving comparable performance by incorporating CAMs within their workflow.

1.5 | Aims and Objectives

The aim of this study is to explore how weak supervision can assist and improve human annotation in CV datasets. For this aim to be achieved, the following objectives have been set:

- Extract human image annotation performance through a tailor-made crowdsourcing web application.
- Evaluate machine-generated and man-made annotations via various metrics like accuracy, F1-score, localisation error (LE) and time in seconds.
- Verify the results to remove human bias through a form that tests a human's ability to discern machine-generated annotations from man-made ones.

In view of the issues described in Section 1.1, this study investigates the utility of digital systems acting as facilitators to data annotators. Various types of digital systems are reviewed, with a clear focus on minimising effort.

1.6 | Proposed Solution

To achieve the goals outlined in Section 1.5, various methodologies are required. The first objective could be achieved by implementing a custom web application that emulates crowdsourcing. Through various auxiliary functions, it would be able to anonymously track user performance. These measurements would then be used to model human annotation behaviour objectively. This information concerning human annotation behaviour would then be analysed to determine the effectiveness of humans within such work. To further evaluate this primary data, image classifiers based on various AI architectures would be trained to output localisations. This would be done by applying a weakly supervised technique like CAMs. By comparing the man-made and machine-generated annotations through metrics like LE, the localisation performance of both types of annotations would be quantified. However, this could lead to arguably subjective results. Since the ground truth is also man-made, then the results would be biased. Thus, to verify the previous results, further primary data would need to be gathered via a web form so as to quantify the consistency of the general public via the previous results. By asking the participants to discern which annotations are man-made, their observational abilities would be tested. If the participants fail to determine which annotations are man-made, then it can be confirmed that humans are unreliable in image annotation work. Additionally, this would also confirm that machine-generated annotations are comparable to man-made annotations. In that case, a digital aid would definitely be needed to ascertain their work.

1.7 | Document Structure

This report is organised into different chapters, which present their own respective types of content. Such chapters are introduced with the goal of prefacing their purpose within this document. Therefore, an overview of each chapter is presented in the following subsections.

1.7.1 | Introduction

The introductory chapter gave a brief overview of the premise of this study. The premise was then supported with sufficient motivation and further details, which described the general incentive of this study. Such details included the defined problem, in addition to multiple deliverables in the form of the aim, objectives, and contributions. These

separated the incentive into tangible goals, further strengthening the purpose of this study.

1.7.2 | Background

The Background chapter serves as a preamble to the technical concepts tackled in this study. It contains relevant knowledge concerning updated techniques and how they evolved into their current state. More specifically, this chapter contains information about the AI and CV research areas. It also could be considered as an introduction to the Literature Review.

1.7.3 | Literature Review

The Literature Review chapter extends the knowledge presented in the Background chapter by involving arguments and discussions from several methodologies so as to determine which techniques are ideal for achieving this study's goals. Just as in the Background chapter, the evolution of the methodologies considered is discussed by considering the latter's importance within this study. For example, supervised methods are discussed before determining why weakly supervised techniques are the superior option for the current use case.

1.7.4 | Methodology

The Methodology chapter segments this study's objectives into different modules within the implementation. This gives rise to a pipeline, which describes the process of the entire methodology of the study. This chapter also describes what technologies were used in view of the discussions presented in the Literature Review and the Background chapters. More specifically, it presents the datasets, tools, techniques, and experiments utilised to achieve the outlined goals. Additionally, sections describing the nature of the research, such as its scope, expectations and ethics are also included.

1.7.5 | Results

The Results chapter compiles the outputs of the experiments described in the Methodology chapter to derive information relevant to this study's goals. The results are showcased in an objective manner via different equations, metrics, and functions that objectively showcase these outputs. Just as in the Methodology chapter, these discussions

are segmented by the study's objectives so that the results are presented in an organised manner while also providing detailed insights into how such results determined the outcome.

1.7.6 | Conclusion

The Conclusion chapter combines the insights derived from the results to discuss whether this study's goals were achieved. Thus, these insights are assessed in view of the aim, objectives, research question, and hypothesis presented in the Introduction chapter. The outcomes of these deliverables detail the potential of such methods when applied in the current context. Moreover, possible improvements and expansions to the methodology are discussed to truly show how such research could evolve in the future.

2

Background

To ensure a comprehensive study, various related topics were investigated with the goal of understanding the theoretical information behind the main research areas tackled. The literature consulted included brief overviews of AI techniques, such as machine learning (ML) and deep learning (DL), and how AI integrates with CV. The discussions presented in this chapter are supported with examples illustrating practical applications of such technologies within the medical, transport, retail, and security fields. The underlying tasks supporting these applications, such as image classification, object localisation, object detection and image segmentation are also inspected briefly. These discussions served as the basis for further reviews to determine which technologies would be the most appropriate for this study.

2.1 | Artificial Intelligence

AI is a well-known area of study that explores the ability of machines to rationalise information and predict a result from a given piece of data [17, 18]. As with other research areas, AI is segmented into various topics such as ML [19, 20, 21], natural language processing (NLP) [22, 23], CV [24, 20, 25], and robotics [26, 27]. These topics brought forth multiple techniques, which further improved performance within their relevant applications. Since this study deals with CV, the relevant topics are reviewed in detail to showcase algorithms that provide specific advantages or fill a gap. This chapter then presents ML's well-known advancement known as DL, which revolutionised ML to become more accessible and powerful.

2.1.1 | Machine Learning

ML is a subset of AI where algorithms are fitted to a sample of data to infer results from an input in a generalised manner. Therefore, these algorithms would be transformed into models where they differ in their predictions depending on their architecture and training data [17, 18]. ML is most useful in applications where conventional programming solutions are infeasible. Khan et al. [19], in fact, used ML to predict stock market trends, which can be vague and unpredictable. Another example is when Caruso and Gattone [21] took advantage of ML to extract waste recycling habits through distinct variables. To implement such ML solutions, there exists a common workflow. Firstly, sample data must be acquired from the population of data relevant to the application. In the case of Khan et al. [19], they used public sentiment and political articles gleaned through search engines and public repositories to achieve this. The difficulty of this step is proportionate to the required final accuracy of the model and the inaccessibility of the population of data. Secondly, the data are sanitised to remove any unnecessary or erroneous information through an algorithm. This process can be seen in this study, where an algorithm known as histogram of oriented gradients (HOG) was utilised to extract features from multispectral images for weed detection in crops [24]. Finally, the ML algorithm is trained and evaluated through a training session. This step can vary considerably, depending on what type of ML algorithm is utilised. For instance, decision trees generate conditions through trials to dynamically fit the sample data inputted

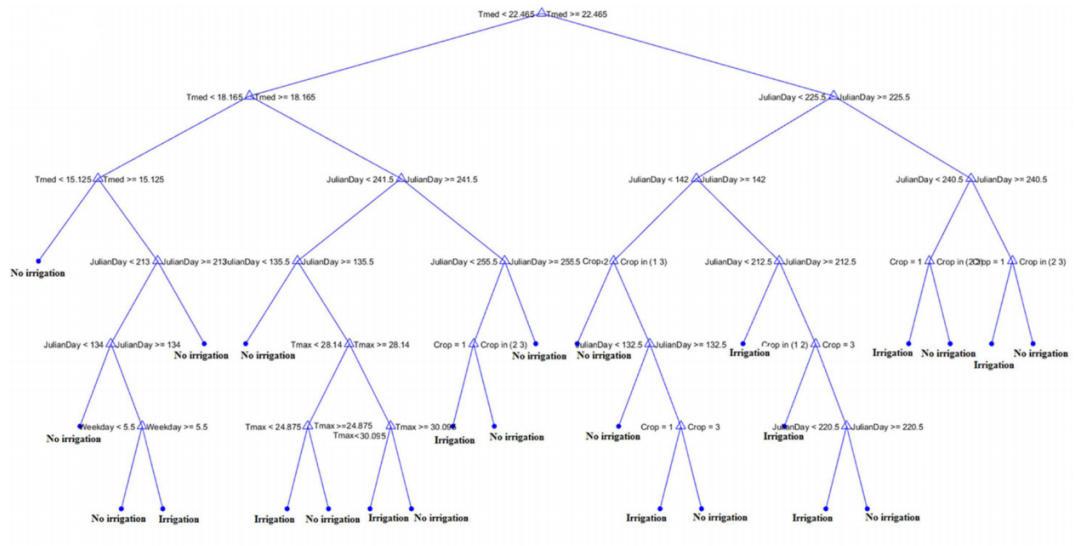


Figure 2.1: Decision tree based on best practices for irrigation [28]

[28, 29]. An example of a decision tree is illustrated in Figure 2.1. Meanwhile, support vector machines (SVMs) attempt to find the optimal regression or classification model within the sample data represented as a hyperplane [30, 31]. Another notable technique involves what is known as a neural network (NN). NNs are trained by altering their individual weights to adhere to the expected results dictated by the sample data [32, 33].

2.1.2 | Deep Learning

Data sanitisation within the ML workflow is significantly problematic. In most cases, the feature extraction process is inadequate to properly remove all redundancies and errors. This is usually mitigated via manual intervention by humans in a process known as feature selection. For instance, Suhandy and Yulia [34] used a process known as principal component analysis (PCA) to reduce the dimensionality of the annotations within wine data. A visualisation of this process can be seen in Figure 2.2. Such instances of excessive human intervention reduce the accessibility of AI research, stunting its development and advancement. This significant issue within ML's workflow spawned efforts to mitigate it, leading to the research topic known as DL. DL is an extension of ML that eliminates the feature selection prerequisite. This was done by extending the functionality of the NN [17]. For example, it is common within the CV area to integrate AI via convolutional neural networks (CNNs). CNNs are simply NNs

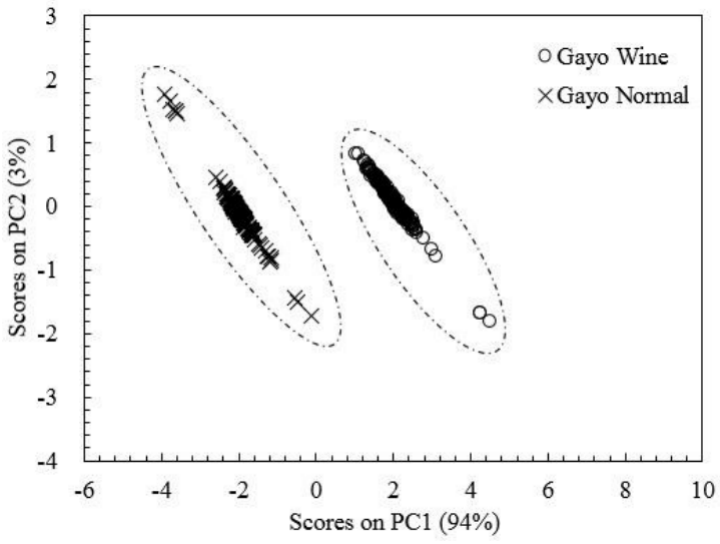


Figure 2.2: Wine annotations processed via PCA to a 2D plot where data can be visualised easier [34]

combined with a filtering process designed for images [35, 36, 37, 38]. These filtering processes automatically extract the features of the input data, removing the need for feature selection [17]. Another instance of DL can be found within recurrent neural networks (RNNs). RNNs are essentially NNs that can evolve over time through their own inferences, showcasing competent memory retention abilities [39]. Overall, DL revolutionised ML and the AI research area as a whole due to simplifying the data sanitisation process. This is further supported by the notion that DL algorithms require much more training data; since much more training data are utilised, DL models are considerably more accurate when compared to ML models. Consequently, DL models became renowned for their reliable performance.

2.1.3 | Metrics

Various metrics may be used depending on the scenario these models are applied in. The most popular of these metrics is known as accuracy, which uses the true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs) to calculate model performance. Equation 2.1 reveals the inner workings of the accuracy metric, where: tp represents the TP count, fp represents the FP count, tn represents the TN count, and fn represents the FN count. Variable a denotes the outputted accuracy value.

$$a = \frac{tp + tn}{tp + fp + tn + fn} \tag{2.1}$$

These result types vary depending on their application. In the case of Equation 2.1, this is mostly relevant to classification. Accuracy is calculated by dividing the true results by all the results. Initially, this calculation seems straightforward; however, it is not predictive due to how it takes into account all results. Therefore, this measurement is unable to foresee the strengths or weaknesses of different models. To mitigate this issue, predictive metrics could be used to evaluate models more accurately. The precision of a model is reflected through its positive predictive value (PPV). In other words, precision is the measurement that represents how many of a model's inferences are actually true. As the designation PPV entails, it calculates a model's performance in view of the FPs. Precision is determined by following Equation 2.2, with the same denotations as accuracy and where p represents the end precision value.

$$p = \frac{tp}{tp + fp} \tag{2.2}$$

Precision is not the only measurement that can be utilised as a predictive metric. Commonly used in conjunction with precision, recall is also a prominent predictive metric

in model performance calculations. Arguably, recall is the inverse of precision. Where precision takes into account the FPs, recall takes into account the FNs. Therefore, recall measures the model's ability to detect all the ground truths. Similar to precision, recall is alternatively known as the true positive rate (TPR). To quantify a model's hit rate through recall, Equation 2.3 can be utilised with the same denotations as Equation 2.2, the only difference being that, here, r represents the final recall result.

$$r = \frac{tp}{tp + fn} \tag{2.3}$$

While both precision and recall are competent metrics, they still fail to provide a complete outlook on model performance. Both metrics have specific uses, which might differ in importance depending on the use case. Furthermore, they are two different metrics, which may complicate a result set. To limit the effects of this issue, researchers use a metric known as the F-measure. The F-measure combines precision and recall to create a singular predictive metric. Additionally, the F-measure is formulated in a configurable manner. This is because of a variable known as β, which is used within the F-measure, as seen in Equation 2.4:

$$f_\beta = (1 + \beta^2) \times \frac{p \times r}{(\beta^2 \times p) + r} \tag{2.4}$$

This β variable is a positive real number that represents the importance of recall compared to precision. This is useful because different scenarios may treat precision and recall differently. For example, medical studies might prioritise recall over precision. This is because obtaining an FN in a medical test is usually much more severe than obtaining an FP within the same medical test. Conversely, content recommendation models might present a more precision-focused study. This is because failing to present a correct instance of content is less critical than presenting the incorrect type of content. Formally, $\beta > 1$ is used when recall is more important, while $0 < \beta < 1$ is used when precision is more important. There might also be instances where neither precision nor recall have prevalence, such as in generic datasets. Such situations require that $\beta = 1$. In such scenarios, the F-measure represents the harmonic mean of precision and recall. This measurement is alternatively known as the F1-score. Simplifying the F-measure by setting $\beta = 1$ gives rise to Equation 2.5:

$$f_1 = 2 \times \frac{p \times r}{p + r} \tag{2.5}$$

In tasks where a localisation component is present, such as object localisation and object detection, Equations 2.7 and 2.6 are used. Equation 2.6 is known as the intersection over

union (IoU) or the Jaccard index.

$$i(a,b) = \frac{r(a) \cap r(b)}{r(a) \cup r(b)} \tag{2.6}$$

To evaluate this, the area of a rectangle must be calculated. This can be done through Equation 2.7, where x_l represents the length of rectangle x, while x_w denotes the width of rectangle x. Following this, variables a and b are two different bounding boxes that are defined by their co-ordinates.

$$r(x) = x_l \times x_w \tag{2.7}$$

2.1.3.1 | Localisation Error

As the name implies, IoU quantifies the overlap of a set of bounding boxes. In CV tasks such as object detection, a threshold for this result determines whether a prediction matches with the ground truth [40, 41, 42]. Object localisation uses this function in the same manner. According to the ImageNet guidelines, a localisation match is determined when the IoU value reaches a 50% threshold [2]. In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), IoU is used as follows in a localisation context. Localisation serves as an extension to classification, where the model must classify the image as well as locate objects within it via bounding boxes. The ILSVRC evaluates the localisation ability of models by implementing a metric known as LE. However, LE contains many components. Indeed, LE combines a classification evaluation with a localisation evaluation, these being the two main components constituting it. The classification evaluation is very simple. It essentially takes the same form as the accuracy metric shown in Equation 2.1. When observing an LE entry, the classification is evaluated by simply checking whether the classes match. However, LE does present a difference. This difference can be observed in more detail in Equation 2.8. Variable c denotes the actual classification, while the subscripts p and t reveal whether it is related to the prediction or to the ground truth.

$$c(c_p, c_t) = \begin{cases} 0, & \text{if } c_p = c_t \\ 1, & \text{otherwise} \end{cases} \tag{2.8}$$

As the equation shows, LE returns either a 0 or 1 based on the computed comparison. These binary values become relevant further on when combining the two components together. In contrast to the classification component, the localisation component is more complex. It consists of two main functions that convert a set of bounding boxes into an

evaluation. As described in Section 2.1.3, the IoU in conjunction with a threshold can be used to determine whether a predicted bounding box can match with the ground truth. When integrating this localisation functionality within the LE metric, this gives rise to Equation 2.9. Variable b represents a bounding box while the p and t subscripts determine whether it is a prediction or a ground truth. Function i references Equation 2.6, that calculates the IoU.

$$l(b_p, b_t) = \begin{cases} 0, & \text{if } i(b_p, b_t) >= 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{2.9}$$

Similar to Equation 2.8, this function returns either a 0 or 1. In truth, these binary values simply determine whether the classification or localisation was correct in arithmetic. This is important due to how the final equation evaluates the two functions. To formulate the finalised algorithm, sets P and T represent a list of both classification and bounding box elements. P represents the predictions, while T represents the ground truth. This produces Algorithm 1[1]. In summary, LE is produced by comparing each

Algorithm 1 LE representation [43]

$m \leftarrow [\dots]$
for $c_p, b_p \in P$ **do**
  $n \leftarrow [\dots]$
  for $c_t, b_t \in T$ **do**
    $c \leftarrow c(c_p, c_t)$
    $l \leftarrow l(b_p, b_t)$
    $n \overset{+}{\leftarrow} \max(c, l)$
  end for
  $m \overset{+}{\leftarrow} \min(n)$
end for
$e \leftarrow \min(m)$

prediction with the ground truth. Firstly, the predictions' shared label is compared with the ground truth label. If the label is identical, then $c = 0$; otherwise, $c = 1$. Secondly, the current prediction's bounding box is compared with each ground truth bounding box. If any of the combinations produce an IoU of 50% or greater, then $l = 0$; otherwise, $l = 1$. Variables c and l are then compared to return the maximum of the two. Whatever value is returned suggests the prediction and ground truth combination's result.

[1] https://www.kaggle.com/c/imagenet-object-localization-challenge

2.2 | Computer Vision

CV is a research area dedicated to digital imaging. This ranges from hardware converting analogue visuals into digital information to the compression of digital images and videos to a smaller size. CV is also occasionally integrated with AI due to its sheer usefulness within this sector. CV is usually used within AI to offer machines the ability to understand the visual features of a digital image [17]. Such processes are treated as applications consisting of tasks, like image classification and object detection.

2.2.1 | Applications

Being one of the five essential human senses, applying vision to digital systems opens a myriad of possibilities. In medical imaging, CV can be used to detect serious illnesses and conditions. Indeed, Kumar et al. [44] utilised a CNN architecture known as U-Net to accurately outline unhealthy masses through CV. This is also not the only instance where U-Net had been applied in medical applications [45]. They obtained a dataset comprised of mammograms and annotated the different types of breast masses. CV has also seen some use within transport, for instance, in self-driving vehicles. Fujiyoshi et al. [46] surmised that CNN architectures are a popular choice for such applications. They even mentioned that a technique known as CAMs could be used in an end-to-end system to explain the models' predictions. Such models have also seen use within more niche applications, such as when Higa et al. [47] applied CV models to planetary rovers to predict fuel consumption by taking into account Mars' terrain. CV and AI have also been used in retail to automate checkouts. Santra and Prasad Mukherjee [48] surveyed trends within such applications and listed multiple vision-based methods that were being utilised in such situations. This was then extended by cross-referencing the result to the pricing, allowing the customer to automatically check out their products. Another similar example of this is the Amazon Go system[2]. In relation to retail, CV could also be utilised in security. Automating such applications could reduce unnecessary human intervention in areas that need constant surveillance. Meanwhile, Jain et al. [49] used CNNs to ensure safety within establishments by detecting weaponry. They utilised two different datasets, where one was pre-annotated while the other was tailor-made for the application in question. Another application in the same domain was created by Morales et al. [50], who used CNNs to detect robberies within establishments. They did so by modifying an image classifier to extract features instead. These features were then

[2]https://www.amazon.com/b?ie=UTF8&node=16008589011

fed to long-short term memory (LSTM) layers to process a result via the frame sequence. They also used a pre-annotated dataset known as UNI-Crime.

2.2.2 | Tasks

These applications are industry-level products that wrap different tasks into a tangible system. These tasks are the underlying logic that dictates CV's usefulness within such applications. Therefore, these tasks are briefly reviewed to ensure that our understanding of them guides this study to the correct outcome.

2.2.2.1 | Image Classification

Image classification is arguably the most straightforward utilisation of CNNs within CV. The goal of image classification models is to convert an image into an appropriate label. In other words, this task can be treated as a visual categorisation method. This is similar to how us humans can recognise different types of objects by simply looking at them [17]. While this task is simple in nature, it has multiple important use cases. For example, Abubakar et al. [51] used image classification to help identify severe burns via CV. They claimed that categorising these burns through CNNs reduced medical costs whilst being more time-efficient. Meanwhile, Kaur and Gandhi [52] and Mathew and Anto [31] used image classification to efficiently determine whether a magnetic resonance imaging (MRI) instance shows a healthy brain. These use cases confirm that image classification is best suited when singular images are the typical input. Moreover, the input data should contain a clear view of the target object, eliminating the need for any localisations.

2.2.2.2 | Object Localisation

Object localisation is an extension to image classification, where the localisation abilities of image classifiers are reviewed [2]. In addition to the label provided in the training set, bounding box data are supplied to the algorithm. Therefore, the outputted model is able to provide both a label and bounding boxes when inferring its predictions. Object localisation models are most useful in situations where the model is localising instances of a singular class or if the class detected by the model is unimportant. An instance of such functionality can be seen within this dataset [3]. While typically used within object detection scenarios, the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) dataset offers subsets of its own dataset for specific categories. The KITTI dataset provides a pedestrian-only dataset for person detection. Since only one label is

utilised, object localisation is an applicable method for the task. Additionally, Sharma et al. [25] used object localisation to decode license plates. Since only license plates were the label of focus, object localisation was a logical method to implement.

2.2.2.3 | Object Detection

Object detection is similar to object localisation, but without the one label per image limitation. This empowers models to detect multiple instances of different objects within the same image. This significant upgrade allowed CV to be applied in even more different scenarios [53]. The aforementioned KITTI dataset is used to train traffic-related AI models for self-driving vehicles [3]. It contains labels related to pedestrians, vehicles, and traffic signs. Each image contains annotations of all instances of those objects rather than of just one. This changes the problem definition from finding a specific object to essentially understanding a scene. There are other instances of transport-related studies that utilise object detection as well [54, 55, 56, 57]. Therefore, in instances where multiple objects of interest are found within media, object detection is used to effectively decode their locations.

2.2.2.4 | Image Segmentation

Image segmentation is used when the localisations provided by either object localisation or object detection are not detailed enough. Image segmentation differs from object localisation or object detection in that it classifies each pixel under a label rather than classifying each bounding box. Thus, apart from simply classifying malicious breast masses, Kumar et al. [44] also used image segmentation to outline them. While image segmentation is quite a computationally expensive task, it has also been applied in a similar way as object detection. Indeed, by integrating image segmentation with video sequences, Chen et al. [58] were able to segment entire urban scenes. From this, it can be inferred that image segmentation is most useful when the scene must be understood entirely or when the exact shape of the detection is important.

2.3 | Summary

This chapter clarified the scope of this study by briefly introducing the AI and CV research areas. This was done by segmenting CV into its respective tasks. It can be seen that DL is superior to ML both in ease-of-use and performance. This is shown through the critical nature of DL's applications, which include medicine [44, 39] and transport

[56, 57]. Since this study focused on CV, these differences are to be further explored in that same scope.

3

Literature Review

In this chapter, the CV tasks that were discussed in Chapter 2 are examined in further detail. Image classification and object detection architectures are extensively studied to discuss their contributions. Each iteration is compared to a successor either via the reported metrics, their unique features, or their applications. Supervised techniques are also discussed. Multiple methods, such as fully supervised learning, unsupervised learning, self-supervised learning, and weakly supervised learning, are briefly reviewed. Weakly supervised learning is, however, discussed in further detail as an AI explainability method known as CAMs could alternatively be used for weakly supervised object localisation. Since CAMs are also commonly applied with CV, they presented potential towards this study's goals. Consequently, further investigations were carried out on similar works that attempted to improve human annotation performance. These studies helped with determining the most prominent aspects of such digital systems, shaping the end prototype of this study.

3.1 | Image Classification

Expanding on the information presented in Chapter 2, the following subsections review multiple image classification techniques to evaluate their strengths and weaknesses.

3.1.1 | Machine Learning

In ML, image classification is usually carried out by extracting the visual features of the image data and then using those same features as input data for a classifier. This classifier would then output the appropriate value which represented the predicted label

[32, 33]. Similar to other ML methodologies, this complicated the annotation process making AI more inaccessible compared to DL.

3.1.1.1 | K-Nearest Neighbours

One solution to perform image classification would be gathering similar images by their features and showing them as clusters. This is how the k-nearest neighbours (KNN) algorithm can be used for this purpose. This study reviewed multiple ML methods to classify facial expressions [59]. Since the KNN algorithm is only a classifier, various feature selection methods were used to prepare the input data. These included HOG for extracting the features and PCA for reducing the dimensionality of those features. Additionally, KNNs do not require any labels within the training data. Even so, Dino and Abdulrazzaq [59]'s classifier achieved almost 80% accuracy under such conditions. Another popular use case for KNNs is optical character recognition (OCR). Hazra et al. [60] used KNN in this fashion, extracting the features of handwritten text via similar methods. Other more applied uses for KNNs can be seen within plant health [61] and transport [62].

3.1.1.2 | Support Vector Machine

KNN is usually used in either clustering or association tasks. This is due to the lack of labels presented within the training data. When labels are provided, a classifier can operate within a supervised scenario. An example of such a classifier is the SVM. SVMs are algorithms that operate well in classification or regression tasks on data with a high number of dimensions. These data can be considered as a Euclidean space of n dimensions, with each data point in the space being represented by an n-dimensional Cartesian co-ordinate. Consequently, SVMs are trained by finding the optimal equation for that graph, which is known as its hyperplane. The hyperplane is determined by maximising the distance between the nearest data points from each label. In CV, SVMs are also used in a similar manner. An image can be deconstructed into its features by using feature extractors like HOG and PCA [59]. These features are usually represented as high dimensional features, which are then classified through the SVM. Despite being a dated ML method, SVMs were still being used in recent years in medical use cases such as this study by Mathew and Anto [31]. In this case, Mathew and Anto [31] used wavelet transforms to extract the image features to be inputted to the SVM. Additionally, the model was also able to perform segmentation. Other use cases include remote sensing [63] and material classification [30].

3.1.1.3 | Multilayer Perceptron

Another fully supervised classifier architecture is the multilayer perceptron (MLP). The MLP is an NN-based architecture that uses fully connected layers. This is what is known as a feed-forward network, as the network processes an image's features, which are calculated by transforming n-dimensional inputs via non-linear operations, through all the layers once in sequence [65]. In its simplest configuration, an MLP could contain as little as three layers: an input layer, a hidden layer, and an output layer. The hidden layers can be increased to theoretically improve performance. Figure 3.1 shows an MLP with four hidden layers. Each of these layers contains a set of artificial neurons, which are the basis of NN architectures. Each neuron can have an input, an activation function, and an output. In the case of a fully connected NN, like the MLP, the input of a neuron usually consists of all the weights obtained from the previous neurons' output. The activation function of a neuron is simply a mapping of the summed values of the neuron input's weights. When using a non-linear activation function, such as rectified linear units (ReLU), the result for the outputted value is $max(0, x)$, where x is the neuron's input [66]. This result also represents the output of the neuron, which is typically passed to all the next layer's neurons in the MLP. To train an NN-based architecture like the MLP, the weights are updated via an algorithm known as backpropagation. A batch of images are passed through the network in a forward pass. The predictions are then compared with the ground truths, where an error value is calculated via a loss function. This error is then used to update the neuron weights in a backward pass. This is done to minimise the loss function in the following forward pass. This cycle is known as an

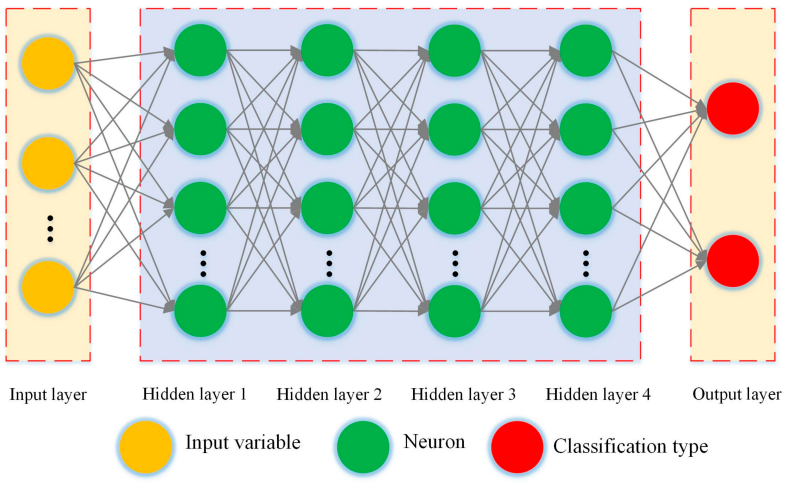


Figure 3.1: The MLP architecture utilised for a surface water extraction algorithm [64]

epoch. Epochs are processed until all training batches have passed through the network [66]. Despite their simplicity, MLPs have been used in many complex scenarios. In CV, they have been used for medical problems such as in the study by Ulloa et al. [65], where brain MRI instances were classified to detect schizophrenia in patients. To improve the classification rate, different types of MRI were used to create a multimodal dataset. Therefore, one label contained more than one type of sample. Ellen et al. [66] too used an MLP in their study; they experimented with classifying planktons using various types of classifiers, including CNNs. The MLP reported almost 80% accuracy over 27 different classes. However, the researchers also commented that a CNN provides better performance and ease of use when using large datasets such as theirs.

3.1.2 | Deep Learning

DL improved the ML workflow by removing the need to perform manual feature extraction techniques such as the likes of the HOG [59], PCA [59], and wavelet transforms [31]. In CV, DL is primarily applied via CNNs, which brought about the implementation of multiple CNN architectures. The latter have provided unique techniques which helped improve the state of the art of image classification.

3.1.2.1 | AlexNet

Earlier implementations of CNNs were trained using central processing units (CPUs). Due to the large amount of features that were extracted from images, this CPU prerequisite caused the training of DL algorithms to be cumbersome. However, a model known as AlexNet, which was presented in the ILSVRC of 2010, showed an alternative. AlexNet was trained by using graphical processing units (GPUs), which drastically accelerated its training. When using a variant of this model in ILSVRC 2012, Krizhevsky et al. [67] also achieved first place by presenting a low top-5 classification error (CE) rate of 15.3%. This specific CNN architecture used five convolutional layers, each followed by a max pooling (MP) layer. The model then ended with three fully connected dense layers as the classifier [67]. Despite being considered a classic model, AlexNet was recently used in prominent fields like medicine. Indeed, Lu et al. [68] used AlexNet as an automatic feature extractor for brain MRI. There are also instances of AlexNet being used in retail; Seker [69], in fact, used AlexNet to classify damaged fabric. In the same vein, AlexNet was modified by Samir et al. [70] to both classify and localise forgeries of art pieces.

3.1.2.2 | Visual Geometry Group

AlexNet is relatively small in its architecture, it being comprised of only five convolutional layers. However, each convolutional filter is quite large, resulting in AlexNet containing 650,000 individual neurons [67]. In ILSVRC 2014, Simonyan and Zisserman [35] implemented a CNN architecture known as Visual Geometry Group (VGG). VGG is an experimental architecture that was implemented to discuss the effects of increasing the depth of a CNN rather than the size of its receptive fields like AlexNet. VGG brought about an algorithm in which the smallest configuration has eight convolutional layers. The most complex configuration, which has 16 convolutional layers, achieved a top-5 CE rate of only 7.3%. Additionally, VGG was also implemented with localisation in mind. It won the localisation challenge of ILSVRC 2014 with 25.3% top-5 LE. Simonyan and Zisserman [35] also approved of VGG being used to further research within CV, which prompted the creation of various implementations and applications of VGG. Kaur and Gandhi [52], in fact, used VGG to classify brain MRI instances of different neurological conditions. In transport, a modified VGG was used by Bi et al. [71] to determine whether it was capable of inferring traffic sign classifications accurately in real time. VGG was also applied to forgery detection, such as in the study by Chang et al. [72], to detect deepfakes. Chang et al. [72] handled this problem by presenting a slightly modified version of VGG that was trained on a deepfake classification dataset for celebrities. In addition to this, VGG was also applied to waste classification [73].

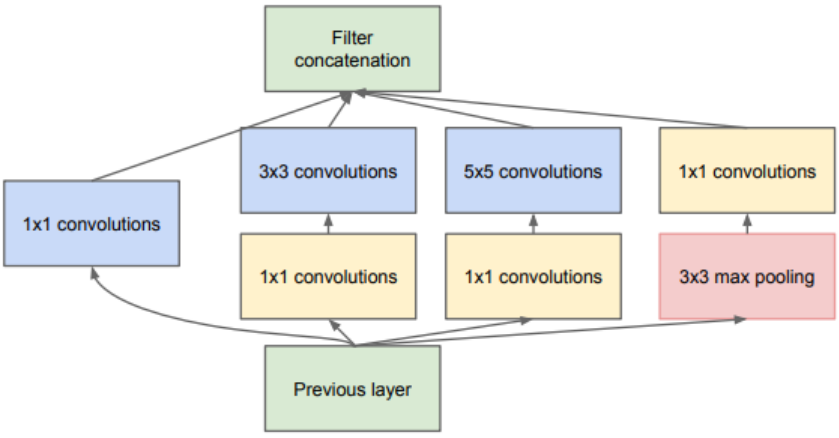


Figure 3.2: Inception module [74]

3.1.2.3 | Inception

Both AlexNet and VGG are relatively complex models for their size, boasting 60 million and up to 144 million individual parameters, respectively. This is due to the increase of connections as the network deepens in layers. While hardware does continuously become more computationally powerful over time, such large numbers of parameters do prolong the training times and hinder the inference speeds of these architectures. Another architecture that attempted to resolve this issue was presented in ILSVRC 2014. This architecture, which is known as Inception, mitigated the CNN efficiency problem by introducing a component called the "Inception module", which is illustrated in Figure 3.2. This module sets convolutions of different sizes within the same layer to be concatenated. The number of parameters is consequently reduced while still preserving performance [74]. Moreover, Inception was subjected to further optimisations, such as the inclusion of a global average pooling (GAP) layer. This was done to avoid unnecessarily increasing the parameters like VGG. VGG flattens the image features for the classifier, forcing high dimensional features to be determined as a singular array [35]. This increases the computational complexity required to process the predictions; however, Inception successfully avoids this issue. Inception performed better than VGG by 0.7% in terms of CE, taking first place in ILSVRC 2014. Additionally, Inception uses fewer parameters than AlexNet by 12 times, leading to inference speeds that are up to three times faster. Inception proves that optimising CNNs does have a quantifiable benefit, both in efficiency and accuracy. The Inception architecture has been applied to environmental scenarios, such as in the study by Feng and Tang [75], who used an updated version of Inception to classify different types of garbage within an office space. Gao et al. [76] used the same updated version of Inception as an automatic feature extractor for their flower classifier. Medically, Inception has seen use in mammogram classification to detect breast cancer in patients [39].

3.1.2.4 | ResNet

Early CNN implementations, such as AlexNet, suffered from a phenomenon known as the "vanishing gradient". This problem emerged from the training technique known as backpropagation. Even though the backward pass would update the weights via the loss function, the error would barely change if the the weight value is small. In severe cases, this would prevent the algorithm from converging [77]. A study that presented an algorithm known as ResNet to ILSVRC 2015 attempted to solve this issue. The study suggested that simply adding layers to the architecture would not force the model to perform better. Instead, it was discovered that performance would inevitably degrade

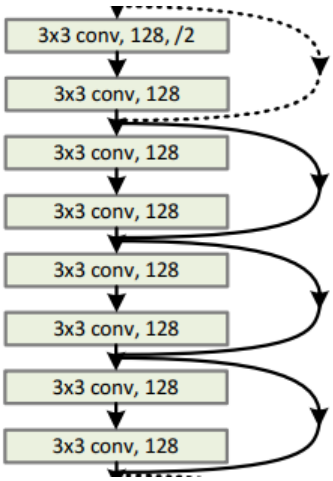


Figure 3.3: Skip connections in ResNet-34 [77]

the deeper a model is [77]. ResNet resolved this problem by implementing residual learning. This technique was primarily applied via "skip connections". These skip connections are integrated throughout the architecture in blocks. They provide an alternative path for the information to flow without adding any parameters. In turn, this presents an identity function that preserves the gradient while preventing the weights from becoming inconsequentially small, allowing a truly deep architecture to improve [77]. Figure 3.3 shows a representation of these skip connections. ResNet is a family of architectures that presents variations that have between 18 and 152 total layers [36, 77]. In ILSVRC 2015, the 101-layer variation presented a CE of 6.1% and an LE of 10.6% when combined with a region proposal network (RPN) and regions with convolutional neural networks (R-CNN). This novel architecture has been used for medical diagnoses, for instance, to classify burns [51]. Additionally, unmanned aerial vehicles (UAVs) too have benefited from this architecture as Seidaliyeva et al. [78] used ResNet to classify the status of such vehicles. ResNet has also been used for synthetic-aperture radar (SAR) imaging to classify sea ice [79]. Another study used ResNet to classify waste [73].

3.1.2.5 | Inception-ResNet

Inception-ResNet is an experimental architecture that explores the effect of skip connections when integrating them with the updated InceptionV3 architecture. The creation of this architecture was inspired from the significant performance increase yielded by residual learning. While Inception-ResNet only improved ResNet's performance by 0.2% top-5 CE, the training time significantly decreased in comparison to the standard InceptionV3, implying that CNNs were becoming even more accessible to researchers.

This specific architecture has been implemented in industrial use cases such as, vehicle classification [80] and satellite imagery [81]. Additionally, it has also been employed in chemistry where, Zhang et al. [82] used Inception-ResNet to detect a fish allergen known as parvalbumin.

3.1.2.6 | Xception

As with Inception-ResNet, Xception was implemented by using Inception as a basis. However, Xception applies depthwise separable convolutions rather than Inception modules. This block separates the red, green, blue (RGB) channels of the input features and calculates their result individually. This is implemented by using a standard depthwise convolutional layer followed by a pointwise convolutional layer. Instead of multiplying all the convolutions together, the channels are calculated separately and then simply added together, reducing the computational complexity of the algorithm whenever this block is applied [84]. This block can be seen in further detail in Figure 3.4. Despite this major change, the model achieved a slight improvement of 0.05% in top-5 classification accuracy over InceptionV3 on the ImageNet validation subset. Further investigations revealed, however, that Xception performed even better than InceptionV3 on a different dataset known as FastEval14K. FastEval14K is another dataset that is used for image captioning, where an image has multiple labels rather than just one. FastEval14K, in fact, presents an average of 36.5 labels per image. Xception reported a relative improvement of 4.3% mean average precision (mAP)@100 over InceptionV3 when using this dataset [84]. Applying mAP to the top-100 predictions in image captioning scenarios is logical, as multiple labels would always be predicted. Additionally, FPs are more likely than FNs due to the large number of predicted labels. Therefore, Xception presents a marginal improvement in generalisation over InceptionV3. Consequently, Xception has been applied to very niche use cases, for instance, in microbiology [85, 86], as well as a

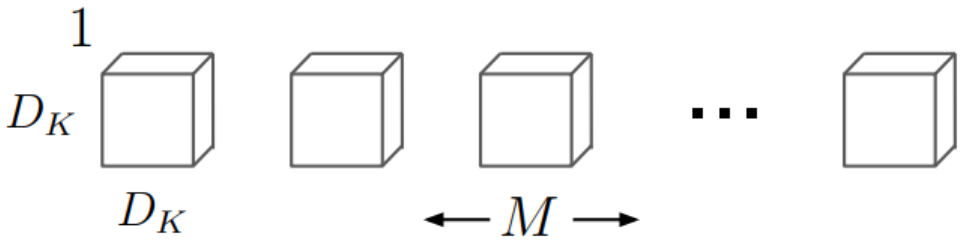


Figure 3.4: Depthwise separable convolution in MobileNet [83]

rudimentary form of image captioning to classify entire scenes [87].

3.1.2.7 | MobileNet

MobileNet is a CNN that was implemented with embedded systems and mobile devices in mind. This resulted in an architecture that is deep but also efficient in both training time and inferences. It contains 13 million parameters with 28 total layers [83]. This efficiency was achieved via the same depthwise separable convolutions used in Xception [84]. Despite being primed for efficiency, MobileNet achieved 70.6% top-1 classification accuracy while VGG achieved 71.5% top-1 classification accuracy [35]. Considering the 91% decrease in parameters, MobileNet proved to be an effective architecture. Additionally, MobileNet was also developed with the intention of integrating it with object detection and image segmentation algorithms [83, 88]. This shows its versatility and effectiveness within other CV tasks. MobileNet has been used in many scenarios where efficiency is usually a requirement, such as environmental [73, 89], security [90], and medical [91] applications.

3.1.2.8 | NASNet

Previous architectures were designed to advance the state of the art or fill a research gap. This was achieved by manually implementing new architectures or by improving on their predecessors. NASNet is an experimental architecture that was automatically implemented by a digital system using an RNN. The RNN is tasked to discover the op-

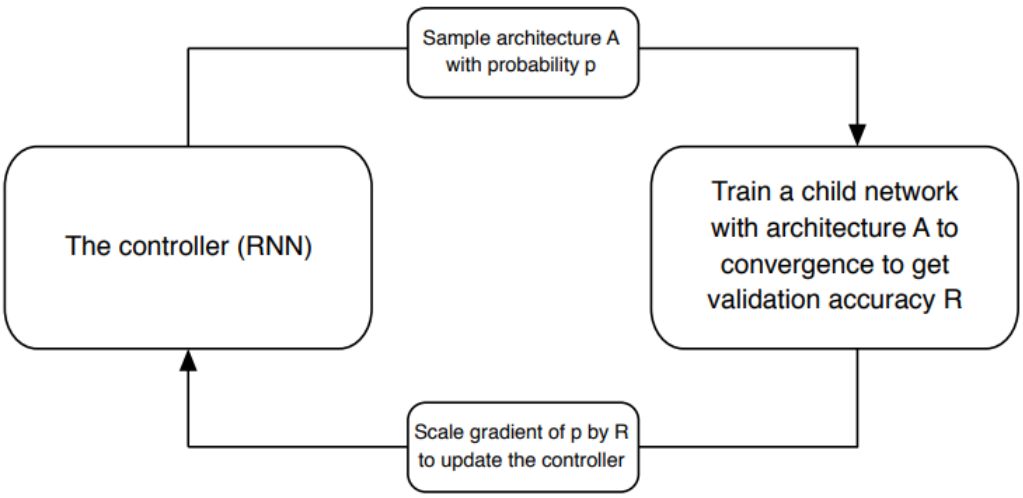


Figure 3.5: NAS flowchart [92]

timal layers on a small dataset, such as CIFAR-10. Once found, this baseline architecture is stacked onto other similarly acquired layers, which are then evaluated on other larger datasets, such as ImageNet [92]. This process, which is shown in Figure 3.5, is known as neural architecture search (NAS). Repeating this process yielded various NASNet architectures. Despite not being a handcrafted solution, NASNet achieved 0.2% higher top-5 classification accuracy than Inception-ResNet. While this is only a meagre improvement, the architecture presented a 28% decrease in computational complexity [92]. This further supports the argument that CNNs can improve in accuracy even if efficiency is prioritised. While not as popular as other examples, NASNet has been used for security applications. Indeed, this study by Radhika et al. [93] used a modified NASNet to perform biometric recognition. NASNet was modified by further optimising some of the layers. Meanwhile, Bakkali et al. [94] trained NASNet on a multimodal dataset to classify physical documents both by their appearance and text. Additionally, NASNet has been used in remote sensing applications [95].

3.1.2.9 | EfficientNet

Usually, CNN architectures are designed with a specific goal or improvement in mind. ResNet was implemented to resolve the "vanishing gradient" problem [77] while MobileNet was optimised for low-powered devices [83]. While this mentality produced competent algorithms, it also reduced their flexibility. As a result, these models suffer in performance when they are applied outside of their original scope. This is what a study known as EfficientNet attempted to solve. EfficientNet is more an ideology than simply a family of architectures. The study implemented a principle that helps with the scaling of CNNs to quickly adapt the architecture to any set of requirements [96]. This is done by using a "compound coefficient" that serves as a basis to determine the width, depth and resolution of an architecture. Different combinations of these attributes can contribute to an increase in accuracy, efficiency, or both. As a demonstration, Tan and Le [96] first used NAS to develop a baseline architecture called EfficientNet-B0. Then, they applied the principle in order to create variations of the baseline network for different use cases. This generated the architectures EfficientNet-B1 to EfficientNet-B7, which increase in complexity by ascending order. The principle proved true, as accuracy increased with each scale. EfficientNet-B7 outperformed models like NASNet by 1.4% top-1 classification accuracy while using 26% fewer parameters [96]. The baseline architecture, EfficientNet-B0, outperformed ResNet-50 in top-1 classification accuracy by 1.1%. Additionally, the baseline architecture is even more efficient than ResNet-34 with only 11 million trainable parameters. This novel algorithm has been used in various

fields such as agriculture [97], transport [98], and even astronomy [99].

3.2 | Object Detection

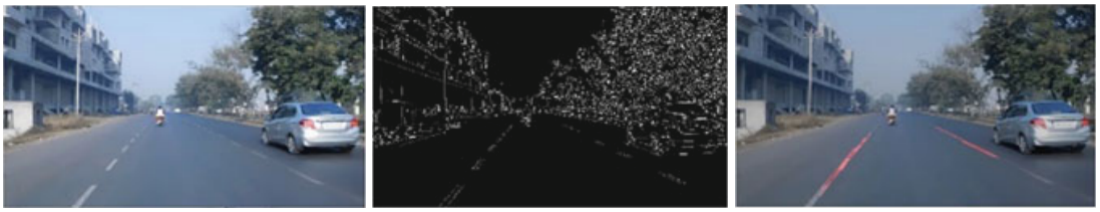
Just as in the image classification section, different object detection architectures are reviewed in this section to discuss their contributions to CV. In ML, these object detectors are mostly used as feature extractors, which complicate human annotation efforts. In DL, some of the CNNs are used in conjunction with deep image classifiers to convert them to deep image localisers. Additionally, some of these CNNs use image classification architectures as base networks for their detection capabilities. A small portion of the reviewed object detectors also use unique annotation methods that could have increased their performance.

3.2.1 | Machine Learning

In ML, feature extractors, such as the HOG, are integral for object detection. Other ML techniques, such as edge detection and Haar cascades, have been used for similar purposes and served as the basis for modern object detection.

3.2.1.1 | Edge Detection

Edge detection is a method that is commonly found within CV applications due to its simplicity as an image filter. Despite its limited use, edge detection could prove to be an effective solution to various problems. Wang et al. [100], for instance, proposed an ML solution to detect landslide hazards. They used canny edge detection in conjunction with other image processing algorithms like Otsu's threshold to monitor cracks in the ground. When it comes to transport applications, Hasabnis et al. [20] used canny edge detection to detect lanes. Since edge detection is only an image filter, it exhibited low computational complexity. This allowed it to infer its results in real time, which is a



(a) Original image (b) Edge detection output (c) Final result

Figure 3.6: Edge detection algorithm applied to lanes [20]

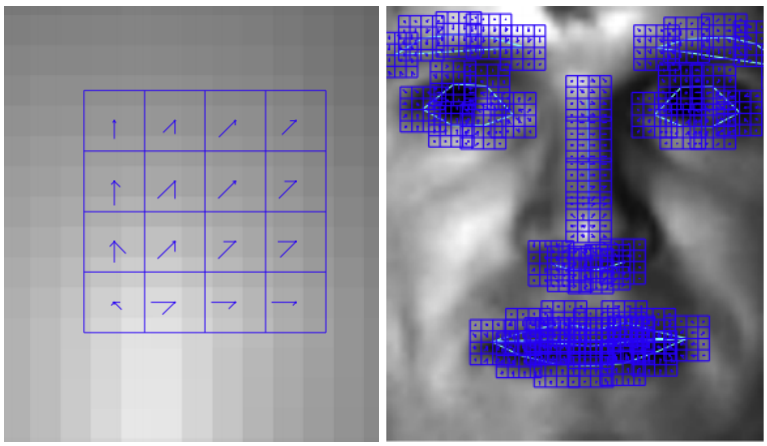
requirement when applied to self-driving vehicles. Figure 3.6 shows sample images of the process of edge detection when applied to lane detection.

3.2.1.2 | Histogram of Oriented Gradients

The HOG could be considered as an extension to edge detection. This is because the HOG is also able to provide the edge direction. This reveals differences in luminance within an image, leading to the implementation of concrete object detection applications. Déniz et al. [101] used HOG as a face detector, which is a popular application for it. Figure 3.7 illustrates the HOG and how it represents facial features. Another effective application for HOG includes person detection, as seen in Watanabe et al. [102]'s study. Machine damage detection [103] and weed detection [24] are some other use cases involving the HOG.

3.2.1.3 | Haar Cascades

When compared to other feature extractors like Haar cascades, the HOG is quite inefficient. Haar cascades use various types of features, such as edges, lines and the four rectangle features. Like the HOG's gradients, these features describe the image contents. Haar cascades use these features to detect patterns that fit the target object (see Figure 3.8). Its most prominent application is the Viola-Jones face detector [105], an early example of real time object detection. Haar cascades have also been applied to detect other objects that presented discriminative features. Reiser de Melo et al. [104] used Haar



(a) Zoomed HOG sample (b) HOG output on face

Figure 3.7: HOG face visualisations [101]

cascades to create an efficient sign language interpreter. It must be noted, however, that when the target object varies in presentation, Haar cascades suffer in performance [106].

3.2.2 | Deep Learning

CNNs have enhanced the performance of object detectors. As a result, many researchers have designed novel CNNs due to their ease of use. Some of these networks have extended the functionality of classifiers to become localisers. Other deep object detectors have presented alternative image annotation techniques. This section discusses these object detectors in further detail.

3.2.2.1 | OverFeat

OverFeat is a CNN architecture that can handle multiple CV tasks. Sermanet et al. [107] designed OverFeat for image classification, object localisation, and object detection; it was itself a candidate for ILSVRC 2013. To integrate localisation and detection functionality, Sermanet et al. [107] used the sliding window approach. This empowered OverFeat to produce bounding boxes. It won both the localisation and detection challenges at ILSVRC 2013. Despite it being a dated architecture, it was still employed in some relatively recent CV-based applications. Indeed, Biswas et al. [108] used OverFeat as a vehicle counter in traffic scenes. Similarly, Rad et al. [109] applied OverFeat for waste detection in the streets.

3.2.2.2 | Regions with Convolutional Neural Networks

Girshick et al. [40] introduced R-CNN as an object detector. When evaluating the model on ImageNet, R-CNN presented a 30% increase in mAP when compared to OverFeat. Instead of using the sliding window approach, Girshick et al. [40] opted to use region proposals. However, the original R-CNN model presented up to 2,000 region proposals.

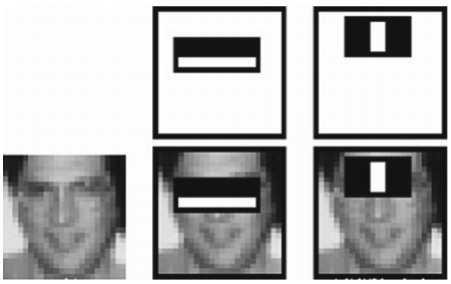


Figure 3.8: Haar-like features on face [104]

This prevented the model from inferring its results efficiently. In fact, it was typical for R-CNN inferences to take up to 13s per image on a GPU. This issue inspired Girshick [110] to implement Fast R-CNN. Fast R-CNN managed to improve inference speeds by replacing the extraction of 2,000 region proposals with a feature map. This minimised the search space for Fast R-CNN, reducing its computational complexity. The feature map was then processed by a modified deep image classifier. In this case, Girshick [110] used VGG16 as the localiser. Consequently, this reduced its prediction times. Fast R-CNN was able to output detections in under 3s per image [110]. While still not suitable for real time object detection, the architecture exhibited a significant improvement in efficiency. Ren et al. [38] further contributed to the R-CNN family by introducing Faster R-CNN. Instead of the selective search used in Fast R-CNN, Ren et al. [38] applied an RPN in Faster R-CNN, this being a dedicated network used to efficiently determine regions of interest (RoIs). Figure 3.9 illustrates this network's design. This reduced the inference times of Faster R-CNN down to 0.2s. Thus, Faster R-CNN is suitable for non-critical real-time applications. Many applications used an R-CNN model within their implementation. Dhungel et al. [111], for example, applied the original R-CNN architecture to detect malicious masses within mammograms. Meanwhile, Ahmad et al. [112] used Fast R-CNN to detect faults in manufacturing processes. Su et al. [113] used Faster R-CNN to detect lung nodules via CV. On a much larger scale, Mao et al. [114] used Faster R-CNN to detect cracks in dam structures for safety.

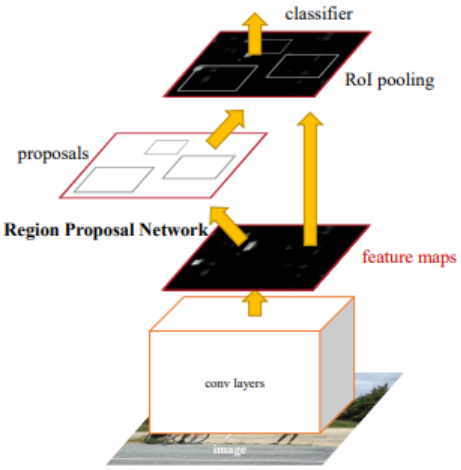


Figure 3.9: RPN diagram [38]

3.2.2.3 | You Only Look Once

The R-CNN family of architectures uses what is known as a two-stage approach. In a two-stage approach, the model proposes the regions to be classified. Then, the model classifies those regions as part of either the foreground or the background. CNNs like the You Only Look Once (YOLO) family of architectures, however, use a one-stage approach [41, 116, 117]. The original implementation of YOLO splits the image into a grid. The resulting crops are then classified and accompanied with a confidence score. If the confidence score meets a specific threshold, the bounding box is verified. While YOLO is inferior to Faster R-CNN by 10% mAP, it can achieve speeds of 45 frames per second (FPS) [41]. The sheer speed of this model indeed makes it very attractive for real time applications. Additionally, YOLO also uses a deep image classifier as a base network. While the YOLO architectures primarily use a tailor-made classifier known as DarkNet, the latter can be replaced with VGG16. The general architecture of YOLO is shown in the diagram depicted in Figure 3.10. Over time, YOLO was improved with more iterations. YOLO9000 updated its DarkNet classifier with a higher resolution and multiscale features, which improved its accuracy while preserving its speed [116]. Redmon and Farhadi [117] applied many experimental changes, retaining the ones that yielded a positive result. One such change involved the replacement of the DarkNet-19 classifier with DarkNet-53. Despite the larger base network, YOLOv3 is still able to infer its predictions at 45 FPS while improving its localisation capabilities. The performance exhibited by these networks attracted many researchers to apply them to

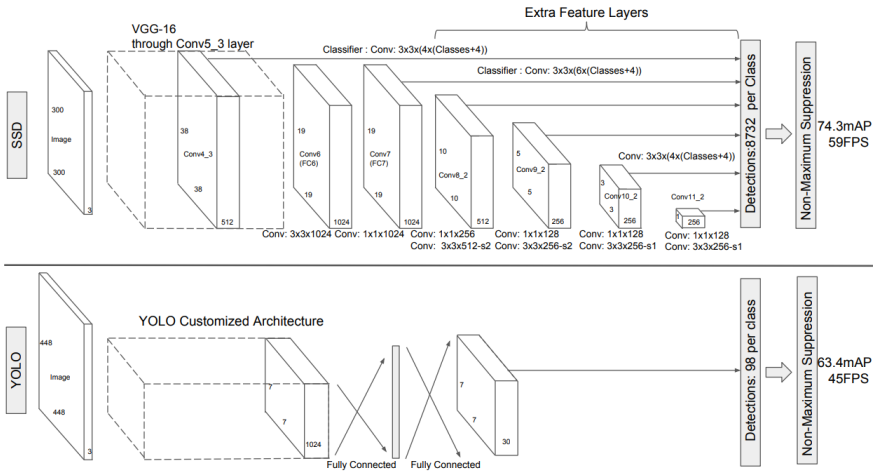


Figure 3.10: SSD and YOLO architectures visualised for comparison [115]

their own studies. Primarily, such researchers were working on applications requiring a quick model. YOLO has, in fact, been employed in self-driving vehicle applications, such as for traffic light detection [118], license plate recognition [119], pedestrian detection at night [57, 120], and vehicle detection [121]. Other time-sensitive applications involving YOLO include weapon detection [122] and building rubble detection [123].

3.2.2.4 | Single Shot Detector

Single Shot Detector (SSD), which is visualised in Figure 3.10, is a one-stage object detector like YOLO. First, SSD extracts the feature maps of the image via a deep image classifier. Liu et al. [115] used VGG16 to perform this step. Since this can be replaced with any deep image classifier, SSD is a modular architecture [88]. To efficiently predict the location of a target object, SSD uses small convolutional filters rather than an RPN. Furthermore, SSD detects objects of various sizes by applying these filters at different scales. Liu et al. [115] evaluated SSD as well as the original YOLO model and Faster R-CNN using VGG16 as the base network, where SSD outperformed the other two architectures. Indeed, SSD can achieve up to 76.8% mAP on the PASCAL VOC 2007 dataset, whereas Faster R-CNN and YOLO achieve 66.4% and 73.2% mAP, respectively. Additionally, SSD reported inference speeds of 22 FPS when using its highest resolution. While YOLO is faster, SSD is more accurate. Consequently, despite the slower inference speeds, there were studies that used SSD in time-sensitive scenarios. Li et al. [124] applied SSD to survey train tracks for either landslide debris or unauthorised personnel. Another study used SSD to monitor important assets at construction sites [125]. Self-driving vehicles can also benefit from SSD, as shown by Chen and Shin [126], who analysed its effectiveness on multi-spectral pedestrian detection.

3.2.2.5 | Feature Pyramid Network

While both YOLO and SSD use multiscale features, they still proved inadequate in detecting small objects. Lin et al. [127] developed and implemented an object detection technique known as a feature pyramid network (FPN) with the aim of tackling this issue. FPNs take an image as input and output individual feature maps in various scales. These can be represented as a pyramid comprised of differently sized layers. Similar to the concept of RPNs, FPNs solve this issue by creating a dedicated network. Additionally, Lin et al. [127] developed FPNs to be modular. In their study, Lin et al. [127] used an FPN-empowered Faster R-CNN with ResNet-101 as the full solution's backbone. On a dataset known as Common Objects in Context (COCO), the FPN-empowered model achieved 36.2% average precision (AP). The same Faster R-CNN architecture without

the FPN gained 34.9% AP. Moreover, the FPN-empowered model inferred its predictions quicker than the original Faster R-CNN by 14%. This shows that FPNs provide a considerable advantage to object detectors. Various studies have employed this method in instances where the target object is small or easily missed. Chen et al. [128] integrated an FPN with a modified RPN to detect faraway ships via SAR. Meanwhile, Ou et al. [129] integrated an enhanced version of an FPN with Inception-ResNet to produce a competent drug pill localiser.

3.2.2.6 | RetinaNet

While one-stage object detectors provide competent architectures, they often perform worse than two-stage object detectors. This can be seen when comparing the performances of SSD and YOLO with R-CNN. Lin et al. [130] investigated this issue in great detail and discovered that one-stage detectors are prone to FPs. This is because algorithms like SSD generate many localisation proposals that do not fit with any label. To resolve this issue, they developed a loss function known as focal loss [130]. To demonstrate this technique, they implemented a one-stage object detector called RetinaNet. Like other one-stage object detectors, RetinaNet uses a deep image classifier as a base network for classifications. For the purpose of their work, Lin et al. [130] used ResNet-101. RetinaNet uses a bounding box regressor combined with an FPN to increase efficiency. When evaluated on COCO, RetinaNet achieved 39.1% AP, which was 2.3% better than the state of the art [130]. This proves that the focal loss technique improves the performance of one-stage object detectors, so much so, that the model's performance might exceed that of two-stage object detectors. Several studies have also used this novel architecture within their applications. Liu et al. [131] used RetinaNet to detect pneumonia through x-rays, while Santos et al. [132] used RetinaNet to develop a storm drain detector to improve urban planning procedures.

3.2.2.7 | CenterNet

In the object detection domain, there exist several annotation methods. Usually, these annotations would comprise four values that draw a box. This box contains the object of interest and is a ground truth sample. However, there is an annotation method that uses keypoints instead. In the object detection architecture known as CornerNet, these keypoints represent only two corners of a bounding box, as shown in Figure 3.11. CornerNet managed to reduce the effort required for annotation; however, it produces subpar results. Duan et al. [133] implemented a possible solution to this issue; they improved on CornerNet by integrating an efficient method to analyse the centre of the

bounding box. This was done by using an extra point - the geometric centre of the object. Additionally, they integrated two modules: centre pooling and cascade corner pooling. These modules enriched the features detected by CornerNet at the relevant keypoints, verifying its own predictions. This architecture is called CenterNet [133]. CenterNet achieved 47% AP on COCO, indicating a 4.9% improvement on CornerNet. Furthermore, CenterNet infers its predictions at 3.7 FPS, which is faster than CornerNet [133]. Many studies have used CenterNet to apply keypoint-based annotations to their applications. Wang et al. [134] implemented a helmet detection application for enhanced safety at construction sites. Meanwhile, Liu et al. [135] used CenterNet with the goal of creating an efficient hand-raise detector for the classroom.

3.2.2.8 | EfficientDet

EfficientDet is a modular object detector that uses the EfficientNet family as the deep image classifier base network. As with EfficientNet, Tan et al. [42] implemented a principle to effectively scale the entire architecture, which could complement the compound scaling applied to the base network. To further improve object detection efficiency, they also proposed a bi-directional feature pyramid network (BiFPN). This is an improved version of the original FPN, which combines both a top-down and bottom-up approach when generating feature maps. Additionally, the feature maps are weighted to further balance the detections and avoid unnecessary overhead, thus presenting more effective feature fusion, which reduces feature complexity. When evaluated on COCO,

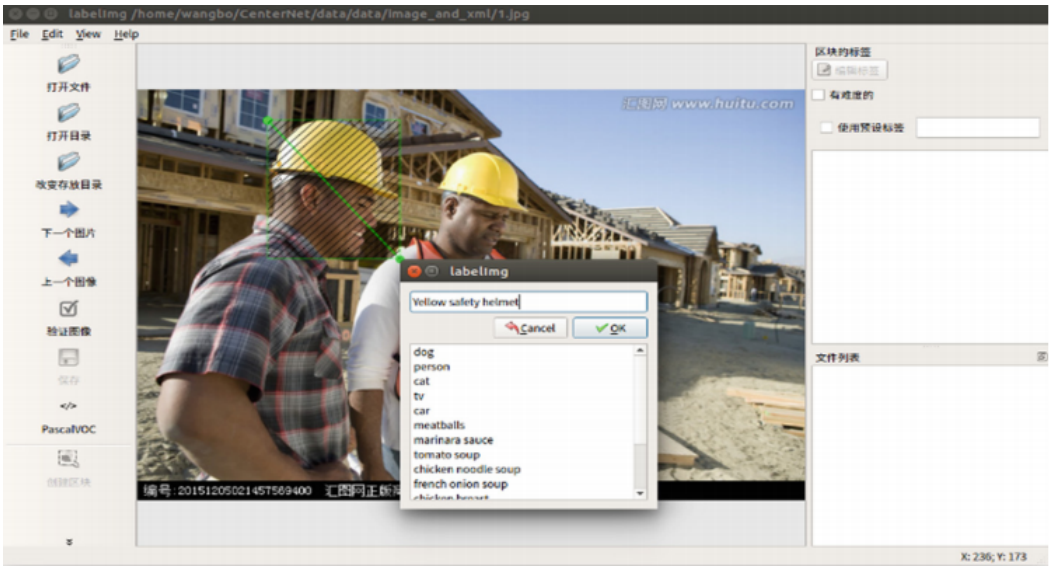


Figure 3.11: Image annotation via corner keypoints [134]

EfficientDet-D0 outperforms YOLOv3 by 0.8% mAP while performing 28 times less floating point operations per second (FLOPS). Researchers have used EfficientDet and its variations in many use cases, presenting it as an effective architecture in transport-related scenarios [55, 56]. Furthermore, by combining it with a modified version of an LSTM, Gupta et al. [136] implemented a visual question answering (VQA) application via EfficientDet. You et al. [137] also used EfficientDet to train a garbage classification model via UAVs.

3.3 | Supervised Techniques

Researchers train AI models by using a dataset as their source of information. The structures of these datasets can vary. In image classification, the annotations present themselves as labels. This supporting set of annotations is also known as the "ground truth". Their goal is to provide the correct learning material for the AI model. Therefore, they supposedly represent a perfect example of their label.

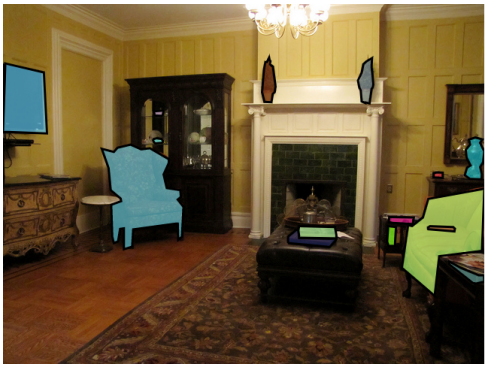
3.3.1 | Fully Supervised Learning

Fully supervised learning produces a dataset that is perfectly suited for its task. Most of the architectures discussed in Sections 3.1.2 and 3.2.2 use fully supervised learning. In object detection, annotations even extend to individual instances within images [107, 40, 41]. This is taken a step further in image segmentation due to it requiring pixel-level annotations, as Figure 3.12 illustrates. Annotating each individual image requires considerable human resources. Thus, creating fully supervised annotations for CV datasets is a rather time-consuming task. Additionally, the amount of effort required might discourage annotators from providing high-quality annotations. Sambasivan et al. [16] showed how such low-quality annotations could have real consequences, further strengthening the argument that annotators should use a digital facilitator for their work. This is especially true in the case of fully supervised learning.

3.3.2 | Unsupervised Learning

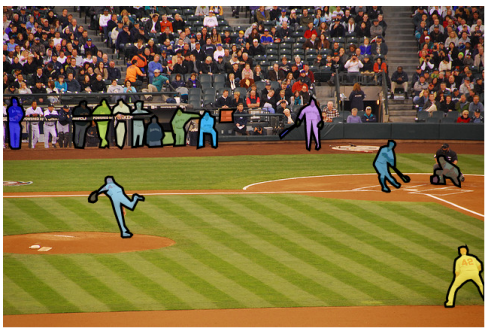
Unsupervised learning is the absence of any metadata within a structured dataset. Therefore, it would contain no annotations. Unsupervised learning is frequently applied in clustering or association scenarios. This study by Caruso and Gattone [21], for instance, used KNN to discover waste recycling trends between locales. In other words, unsupervised learning is most useful when researchers need to extract relationships from

(a) Sample containing a person and an animal



(b) Sample containing multiple types of furniture

(c) Sample containing various airline items



(d) Sample containing people in baseball

Figure 3.12: COCO instance segmentation annotations [138]

unlabelled data. However, since there is a lack of annotations, most of the work lies with data acquisition [21]. Thus, it can be concluded that unsupervised learning is not truly applicable to this study. Additionally, its applications differ from ones that use fully supervised learning. In fact, CV applications that use unsupervised learning fall under standard image processing. Studies have used unsupervised learning for image denoising [139], reconstruction [140] and compression [141]. This further shows how unsupervised learning has limited CV applications.

3.3.3 | Self-Supervised Learning

Self-supervised learning could be considered as an extension of unsupervised learning. Researchers apply it to train AI models for classification without any annotations. This supervised technique is generally used within NLP. In fact, recent NLP models such, as BERT and GPT-3 are self-supervised models [142, 143]. However, there are CV applications that use self-supervised learning. Jiao et al. [144] used self-supervised learning on

fetal ultrasound footage as medical annotations are expensive. They developed a self-supervised model that could reorder footage and segment images. Similarly, Noroozi and Favaro [145] shuffled images to train a self-supervised model, the goal of which was to learn the object's features while reordering the image. Despite the absence of any annotations, such methods can still be unstable. Jiao et al. [144] reported that their self-supervised model achieved a 75.7% F1-score on visual saliency. Meanwhile, Noroozi and Favaro [145] reported that their model achieved only 68% classification accuracy. Additionally, the same model achieved 53% detection accuracy and 38% segmentation accuracy. Self-supervised models are also computationally expensive to train, despite the lesser amount of annotations compared to fully supervised learning. In fact, Noroozi and Favaro [145] trained their model for three days on an NVIDIA Titan X GPU.

3.3.4 | Weakly Supervised Learning

Weakly supervised learning aims to match fully supervised model performance through cheap labels [12]. These types of labels can present themselves in different ways. Lee et al. [146] used this method to reveal the words that contributed to the overall sentiment of a document. Another study by Marino et al. [147] aimed to perform image segmentation via image-level labels only. Therefore, a dataset's annotations are considered as "weak" when it doesn't provide all the necessary detail for fully supervised training. This study lists and discusses a number of weakly supervised learning paradigms.

3.3.4.1 | Semi-Supervised Learning

Researchers are often limited in their data annotation efforts. Such limitations are especially true if they are performing an independent study. These limitations usually bring about either small or incomplete datasets. A weakly supervised technique known as semi-supervised learning aims to resolve this issue by using the already labelled data to infer the annotations of the unlabelled samples. Yoon et al. [148] used semi-supervised learning to enhance their stock investment model with fewer data. Semi-supervised learning simplified data acquisition as stock investment data can be scarce or vague. Some researchers have also applied this supervised technique to CV. Chen et al. [58] used a semi-supervised model to segment urban scenes. Reducing the number of annotations reduced the overall effort required from the researchers to annotate their dataset [58]. Despite the reduced effort, however, this supervised method can also still present instability, especially due to performance degradations, which are often overlooked by the metrics used [149].

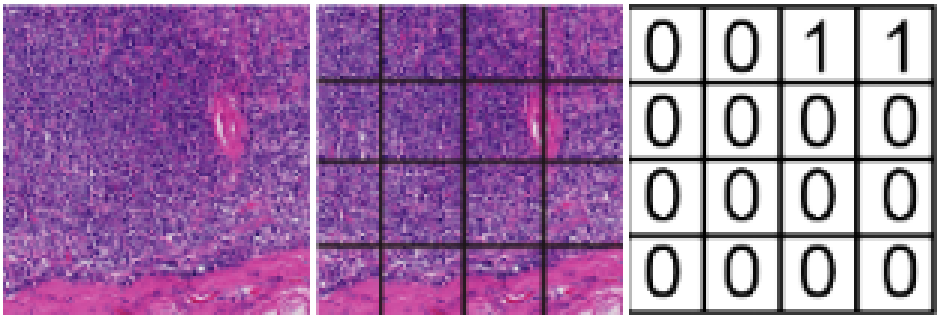
3.3.4.2 | Heuristic Rules

There exist algorithms that are too complex to solve in a reasonable amount of time. An example of such an algorithm is the travelling salesman. This problem describes that a travelling salesman must find the shortest route to visit all the settlements to sell their wares [150]. Therefore, it can also be considered as a route optimisation problem. One method to reduce the amount of time required to solve such problems is to use heuristic rules. These are solutions to such problems that sacrifice either precision, optimisation, or completeness. This is similar to the concept of weakly supervised learning; in fact, weakly supervised learning benefits from heuristic rules. Weakly supervised learning may use heuristic rules to further support the annotations included. This study by Wolfson et al. [151], for instance, used a set of heuristic rules to map NLP to Structured Query Language (SQL). The subject expert then uses these rules to develop their statement. The overall focus of these methods is to simplify annotation efforts. In fact, Snorkel is a framework centred around weakly supervised data annotation [14]. Similar frameworks also extend to CV, as seen with Snuba [15]. Despite these powerful frameworks, they still require a subject expert to program the rules and conditions needed. Therefore, there is still a significant amount of effort required. Furthermore, the annotator could also make mistakes, reducing the accuracy of the annotations [14].

3.3.4.3 | Multiple Instance Learning

Another weakly supervised technique is multiple instance learning (MIL). This technique's methodology alters the traditional AI workflow by introducing the following three steps:

- Organise different sets of unlabelled examples into bags



(a) Original image (b) Extracted patches (c) Patches designated labels

Figure 3.13: MIL annotation process in medical imaging [152]

- Label each bag as positive if it contains the required classification
- Label the rest as negative

When applied to CV, these bags are analogous to image crops. Xu et al. [152] employed MIL to segment cancerous regions in the body, as illustrated in Figure 3.13. MIL has also been used as a dataset annotator helper. Since MIL lattices images, the crops can be treated as bounding boxes, which in turn, provide both positive and negative examples of the label [13]. As a result, the dataset would be fully supervised, albeit with imprecise and unnecessary annotations. Despite its use within CV, MIL is ideal for binary classification. This is because the workflow requires the model to learn from a singular label [153]. Since this study may require multiple classes, MIL is infeasible.

3.3.4.4 | Class Activation Maps

CNNs work like black boxes. It is impossible to answer why a trained model infers a specific prediction using conventional debugging procedures. This issue brought about the topic known as AI explainability. While not primarily used for weakly supervised learning, some AI explainability methods can be used as such. Since CNNs are usually deterministic, they present the same result for the same input. Therefore, it is theoretically possible to interpret an AI model's results. CAMs, for instance, can be utilised for CV explainability [154]. A model can generate a CAM by:

- analysing each label prediction's weight from the final layer;

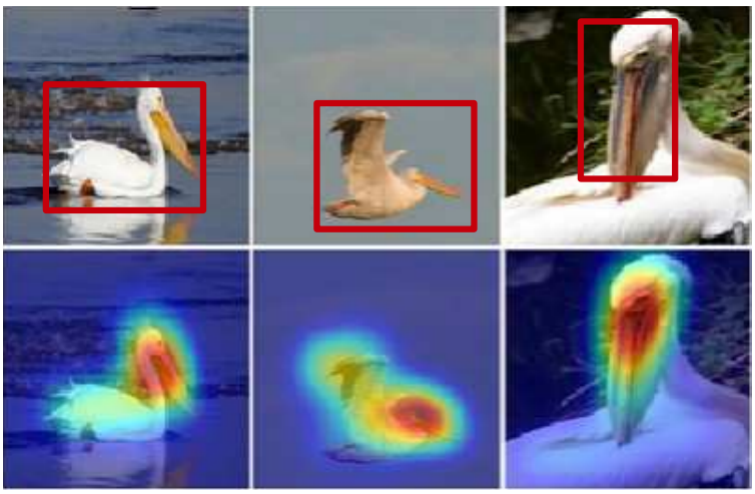


Figure 3.14: A standard CAM visualising a model's focus on some pelicans' distinct features [154]

- producing a feature map from the GAP or global max pooling (GMP) layer;
- calculating the dot product of the feature map and the prediction scores;
- upsampling the attention map to image resolution; and
- superimposing the upsampled data to produce a heatmap.

If a CNN was trained correctly, the model should present a "hot" area on the distinct features of the label. An application can use CAMs as a weakly supervised learning technique by further processing the heatmap. Marino et al. [147], in fact, used the activated pixels of the heatmap to segment potato defects. Then, a thresholding image process is applied to produce the annotation [155]. According to Zhou et al. [154], there was only an increase of up to 16.2% top-5 LE on one of the compared architectures. Despite being a simple weakly supervised method, CAMs do present some drawbacks. Since they show the distinct features of a prediction, it is likely that they present inaccurate localisations. Figure 3.14 shows this scenario more clearly. Standard CAMs also require the usage of a GAP layer as that is how a feature map is generated. This might force changes within architectures like VGG, which do not use a GAP layer. This change would then warrant the re-training of a model, wasting time. These issues inspired many researchers to improve on CAMs, leading to the creation of many variations of the same method. The following paragraphs include a review of these variations as well as their contributions.

Grad-CAM Standard CAM methodologies might be infeasible due to their GAP or GMP layer requirements. If researchers use CAMs on models like VGG16, the models would need to be re-trained as VGG16 flattens its features instead of pooling them. Grad-CAM is a relatively recent AI explainability method that removes this requirement [156]. Grad-CAM implements this feature by taking advantage of backpropagation. Backpropagation produces a gradient which Grad-CAM uses for its attention map generation process by tracking the predicted label's gradients via a backward pass. These gradients are then multiplied with the feature map from the last convolutional layer to produce a heatmap. According to Selvaraju et al. [156], this process allows Grad-CAM to obtain more semantic information. In addition to this, Grad-CAM achieved a 0.7% reduction in top-1 LE when compared to standard CAM methodologies. Selvaraju et al. [156] evaluated these results via VGG16. Since no architectural changes were required, VGG16 did not need to be re-trained. Furthermore, Selvaraju et al. [156] designed Grad-CAM for other tasks, such as image captioning and VQA. In fact, Acharya et al. [157] used Grad-CAM within their VQA application.

Grad-CAM++ Despite its improvements, Grad-CAM still presents some drawbacks. Since Grad-CAM is primarily an AI explainability technique, it is quite discriminative, causing the heatmaps to focus on distinct features as in standard CAM. Furthermore, Grad-CAM is less likely to detect multiple instances of the same label. Grad-CAM++ is an incremental improvement of Grad-CAM that mitigates these issues [158]. The main benefit that Grad-CAM++ provides is that it generalises CAMs even further, allowing it to be a viable technique for both weakly supervised learning and AI explainability. Chattopadhay et al. [158] also claimed that this could be achieved without a significant cost in efficiency. Their main contribution was to introduce pixelwise weighting to the calculation of the gradient, allowing Grad-CAM++ to effectively quantify the contributions of a pixel to the classification. Since each pixel is considered, more exact heatmaps are outputted, as represented in Figure 3.15. This requires the use of higher-order derivatives, which are evaluated via automatic differentiation [158]. To test their method, Chattopadhay et al. [158] used a metric called average drop percentage (ADP). ADP is responsible for quantifying the irrelevance of the heatmap to the predicted label. Therefore, lower ADP represents better performance. Grad-CAM++'s ADPs was shown to be 35.5% lower than Grad-CAM's, showing that Grad-CAM++ is superior to Grad-CAM.

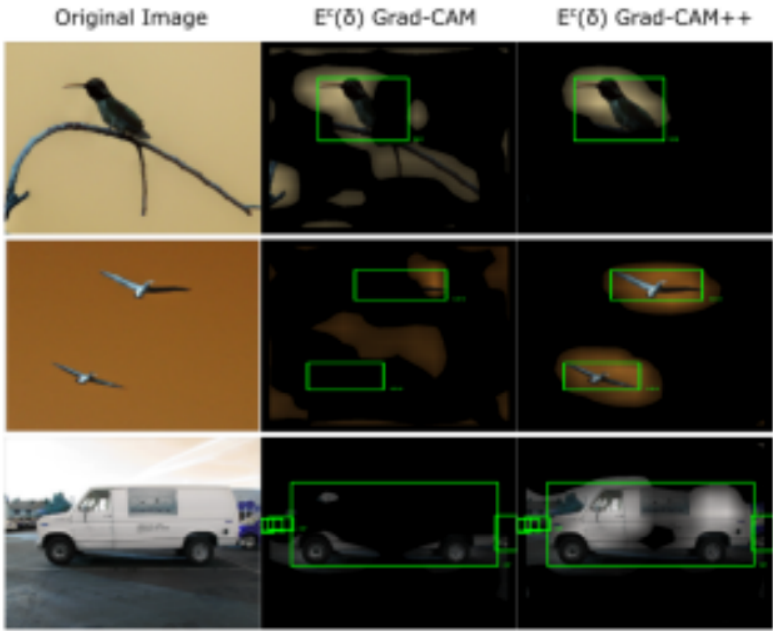


Figure 3.15: Grad-CAM++ improvements over Grad-CAM [158]

Score-CAM Score-CAM is another variation of the CAM methodology. However, it presents some key advantages over both Grad-CAM and Grad-CAM++ [159]. Grad-CAM and Grad-CAM++ use the gradients generated from the backpropagation process. However, according to the researchers behind Score-CAM, this is an unreliable technique. This is because the gradients are susceptible to noise from loss functions like Sigmoid [159]. Noise clouds the CAM and builds an ineffective heatmap. The same researchers also claimed that gradient-based CAM methods present false confidence. Since the feature map is generated via the weights, there could be instances where attention is either under- or over-valued [159]. This can lead to the discriminative issue described when discussing CAM methods. Score-CAM is an experimental CAM technique that abstains from using gradients. Instead, it only requires forward passes to produce a heatmap. Score-CAM does this by gathering the weights of each layer's activation map. This is done with respect to the predicted class. It then produces the final heatmap by linearly combining the weight and activation map values. Score-CAM produces more focused results with fewer inconsistencies within the heatmap. Figure 3.16 shows this on a few samples. Like Chattopadhay et al. [158], Wang et al. [159] also used ADP to evaluate Score-CAM. Score-CAM achieved 14% less ADP than Grad-CAM++. Despite its higher accuracy, Score-CAM presents another issue: as a consequence of not using gradients, Score-CAM upsamples a substantial amount of activation maps throughout the process. This increases its computational complexity, making it relatively inefficient. This issue can be mitigated by limiting the number of activation maps gathered. This is what led to the development of Faster Score-CAM, which improves

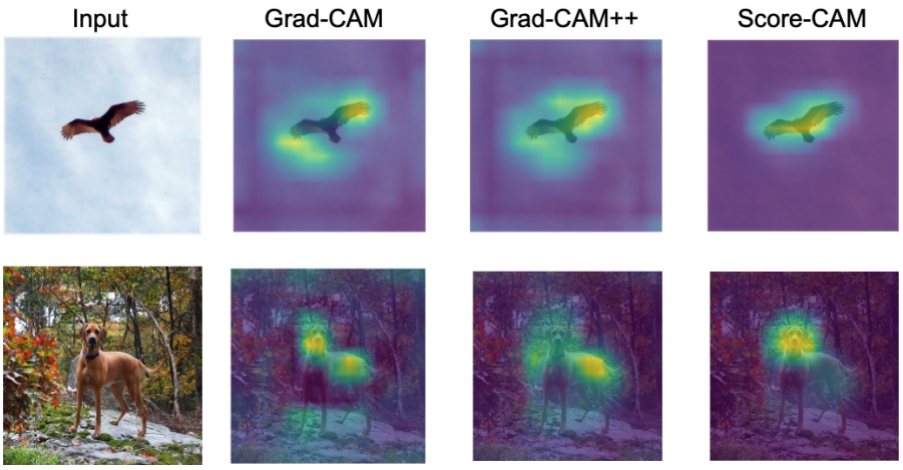


Figure 3.16: CAM comparisons between Grad-CAM, Grad-CAM++, and Score-CAM [159]

Score-CAM's efficiency at the cost of its accuracy[1].

3.4 | Related Works

Handcrafted annotations are the current standard for supervised techniques. Since this implies a considerable amount of effort, human errors are likely. This led to the development of digital systems that can improve human performance. Some systems use AI, while others apply simpler solutions.

3.4.1 | Weakly Supervised Annotation Frameworks

Some dataset annotator helpers use weakly supervised learning as their core functionality. Since it infers results from cheap labels, it can be used to infer annotations. Like this study's goals, these systems do not aim to replace human annotators, but rather, to serve human annotators by facilitating their work and improving their performance.

3.4.1.1 | Snorkel

Ratner et al. [14] argued that high-quality training data are both scarce and expensive. To resolve this, Ratner et al. [14] developed a system to combine information gathered from multiple data sources. Based on their framework's usage, Snorkel can be used to maximise the usage of data from differing sources. Ideally, the relevant information would be in a structured format, such as a database. However, this is not always the case. Medical data, for example, usually requires the intervention of a subject expert, like a doctor [144]. Combining these different sources together increases the likelihood of obtaining better samples. In the case of NLP, Snorkel can output a context hierarchy for a sentence. Then, users can define heuristic rules that produce a label matrix. The latter is used to train a generative model that predicts the label data. These annotations are then fed to a standard CNN for supervised training. This model should use a noise-aware loss function, which might complicate development as it would be a tailor-made solution for the model used. Ratner et al. [14] performed a number of experiments both within and outside of CV. To perform annotations for CV datasets, however, Snorkel is applied in a cross-modal setting. This complicates development as the image data would need to be accompanied by text data to define heuristic rules. In fact, Ratner et al. [14] commented on the difficulty of using their framework on image data. Despite this, Ratner et al. [14] reported that their weakly supervised models come within 3.6%

[1] https://github.com/tabayashi0117/Score-CAM

predictive performance of fully supervised ones. Additionally, Snorkel was found to present a time advantage. When creating the fully supervised sample dataset, Ratner et al. [14] recorded the labelling time to be 10s per image. Considering that the annotations were performed 2.8 times faster, Snorkel possibly presents a worthwhile tradeoff. Nevertheless, Snorkel's users spent the majority of their time programming heuristic rules. Thus, while there is a significant time advantage, Snorkel exponentially increases the effort required for data annotation.

3.4.1.2 | Snuba

Like Ratner et al. [14], Varma and Ré [15] acknowledged the sheer difficulty of obtaining high-quality annotations. There was, in fact, a desire to reduce annotation costs via a digital system. Snuba operates similarly to semi-supervised learning as the framework's main use is to infer labels from a small subset to a larger unlabelled subset. Unlike semi-supervised learning, however, Snuba is almost completely automated. Varma and Ré [15] designed an iterative process that automatically creates the heuristic rules by itself. This process is separated into different components that contribute to automatic annotation. The synthesizer generates a heuristic rule. This rule is evaluated by a pruner on a subset of the training data. If it performs better than the previous heuristic rule of the same nature, the new heuristic rule replaces it. The verifier is a generative model, which is used to determine the end condition for Snuba. Varma and Ré [15] implemented the verifier to ensure that the probabilistic labels are not over-saturated. Otherwise, this event would reduce end model performance. Snuba was tested on a vast range of applications, including medical, generic and social media-related applications. The medical datasets in question were actually CV datasets as they used x-rays or mammograms, making Snuba very relevant to this study's goals. Snuba is superior by up to 14.35 F1 points when compared to standard semi-supervised learning approaches. Additionally, Snuba also outperforms user-defined heuristics, such as the ones presented with Snorkel [14]. Despite the experimental and successful multiclass example, Snuba is mostly primed towards binary classification scenarios. Additionally, Snuba has no localisation applications. This makes it unsuitable for bounding box annotation scenarios, which are common within CV. Despite being mostly automated, Snuba still requires some initial user-defined heuristics for the small, labelled subset. In the case of medical CV datasets, the users took days or even weeks to output the results. So, Snuba still requires a lot of effort from data annotators to perform their work.

3.4.2 | Tool-Assisted Crowdsourcing Systems

Crowdsourcing employs the general public or professional data annotators to annotate a dataset. This is done with the aim of reducing the burden of annotation from the researchers. The users' annotations, however, may be of low quality; since users are typically not subject experts, it is likely that their inexperience might interfere. Several studies have developed solutions that help reduce the drawbacks of such crowdsourcing systems. In other words, digital systems were leveraged to improve annotation quality.

3.4.2.1 | Revolt

Revolt is a web-based crowdsourcing application that builds on traditional methods by exploiting contrary user opinions. In their study, Chang et al. [10], performed a crowdsourcing experiment to evaluate Revolt. Traditional systems only require users to choose a category or draw a bounding box. Revolt asks for further details from its participants on why or how they deduced that answer [10]. It does so by splitting the crowdsourcing process into three stages: voting, explaining and categorising. In their experiment, voting consisted of asking the users whether an image included a "cat" or not. Users were also supplied with a "not sure" option, if necessary. Revolt does this to avoid forcing users to perform annotations they are not confident about. The explaining stage incorporated a text area for users to reveal the logic behind their choice. The final stage presented a user with several image and category combinations from other users. Additionally, explanations from other users were also included to inform the relevant user of the reasoning behind others' decisions. From there, the user was able to either create their own category or choose one of the other users' categories [10]. These statements were also performed in real time; in other words, users were labelling their images and sharing their opinions as a group at the same time. This minimised the time taken for data collection [10]. To further their investigation, Chang et al. [10] developed variations of Revolt to analyse the consequences that different labelling conditions could bring. For example, both a standard crowdsourcing solution and a non-real time application were developed to effectively compare results. Chang et al. [10] reported that Revolt improved the accuracy of annotations by up to 7.1% on a separate vehicle dataset. However, this result did come at an increased time and monetary cost as Revolt users took approximately an additional 7s per entry [10]. Moreover, the complex nature of their system presented a high dropout rate. In their initial prototype, users exited at a rate of 50%. Implementing several improvements, such as progress indicators and clearer instructions, reduced the rate to 5% [10]. This shows the importance of user

interface (UI) and user experience (UX) design.

3.4.2.2 | The Cure

Cancer is a severe illness that has affected many lives worldwide. Gathering medical data on different types of cancer, such as breast cancer, can improve its treatment. Good et al. [9] used crowdsourcing to collect as much information as possible that could aid with breast cancer prognosis. Crowdsourcing is usually seen as a task that requires effort, so Good et al. [9] converted the crowdsourcing system to an online, web-based game to reduce the user's notion that they are using up their free time on a work task that does not benefit them. Instead, the users played the game to have fun or pass the time. Initially, the players were asked to register. The registration process included short questions that were used to build the demographics of the player population. The demographic questions helped segment the players, revealing trends corresponding to their gameplay. For example, 57.1% of the players knew some biology and had a basic understanding of cancer. The game put the players against an automated opponent to choose cards from a board. This board was shared, so if the opponent took a card, the player could not use it anymore. Each card represented a gene, and the player's goal was to build a set of genes before their opponent did. During gameplay, the player's choices served to build a random forest classifier. This classifier was then evaluated to determine the player's score. These gene scores were then used to build a life expectancy dataset for breast cancer patients. Therefore, this game represents an interactive method of data annotation. To evaluate their dataset, Good et al. [9] used an SVM with the goal of correctly predicting whether a specific gene set suggests a 10-year survival. This was done to report the players' contributions in comparison to information recorded in other works. After aggregating the results, the best performing model trained from the end dataset came within 1% of the best performing classifier. It must be noted that, this classifier was trained on all of the entries. Therefore, both experts and novices in biomedicine contributed to producing the best classifier. This shows that non-experts are useful for crowdsourcing, regardless of what the application involves. Additionally, this reiterates the importance of good UI and UX in a web application.

3.5 | Summary

This chapter listed a range of dated and state-of-the-art techniques to impartially determine the ideal methods to apply within the methodology. In large-scale image classification and object detection, ML techniques like SVMs are inferior to DL models like CNNs

in terms of performance and ease of use [66]. This is also applicable when considering data maintenance, as ML models require feature selection, albeit that they are more easily explainable than DL models. In view of their effectiveness, many relevant CNNs were reviewed to list their features and compare their performance. Notably, most of the object detectors discussed use image classifiers as their backbone, alluding to the importance of classification. This is especially true when considering that CAMs can be integrated with such classifiers. Incorporating CAM techniques in image classifiers produces weakly supervised object localisers. Such models can provide localisations even without them existing in the original annotations. This reduces the effort required from both the dataset annotators and the AI researchers to develop a CNN model.

4

Methodology

A set of techniques described in Chapter 3 were used to develop the proposed solution. As already stated, this study used a pipeline, thus presenting the solution via different modules based on the study's objectives. Therefore, each module produced a set of results related to its respective objective. The modules are:

- **Crowdsourcing emulation**: This study developed a web-based crowdsourcing application to extract human annotation behaviour.
- **Annotations comparison**: CAM-empowered weakly supervised object localisers are trained to be evaluated together with the crowdsourcing participants.
- **Human reliability evaluation**: By using a separate web form, the results achieved in the previous module were verified.

4.1 | Solution

The proposed solution tested the hypothesis in Section 1.4 by answering the research question posed in Section 1.3. To this end, a pipeline guided the implementation which was eventually split into three different modules. This pipeline is presented in Figure 4.1. Evidently, the pipeline diagram showcases all the modules in sequence. Each module is discussed in further detail in Sections 4.1.1, 4.1.2, and 4.1.3. The pipeline also reveals the most prominent information that is passed between each module. Due to the image annotation focus of this study, most of the data required came from the original dataset utilised.

4.1.1 | Crowdsourcing Emulation

This study retrieved information concerning human annotation behaviour by collecting image annotations via an interactive web application that allowed its participants to label images and draw bounding boxes around them. This application used images from ImageNet due to its generic nature [2]. ImageNet itself is also a crowdsourced dataset, making it a prime specimen for this study [2]. Additionally, part of the implementation consisted of simplifying ImageNet to produce clearer labels. This ensured a more legible label choice for the participants. To process the results, the web application recorded various timestamps, which were used to determine many aspects of the participants' behaviour.

4.1.1.1 | Preprocessing ImageNet for Crowdsourcing

ImageNet was crafted upon the existing WordNet dataset [2]. WordNet is a repository of text organised in synsets [160]. ImageNet was built on these synsets to aid in the organisation of the labels used. For example, all dog breeds are found under the synset "dog". However, each dog breed is also its own synset. ImageNet itself does not contain the entirety of WordNet; while ImageNet uses 1,000 categories, WordNet uses 100,000 synsets.

Simplification effort Despite ImageNet's crowdsourced origins, the likelihood of confusing the participants was high. Therefore, the implementation preprocessed ImageNet to increase its labels' legibility. The least invasive way of conducting such a process is by performing superclassing. Section 4.1.1.1 mentioned that synsets help organise ImageNet. Since these synsets organise the dataset by exposing label relationships,

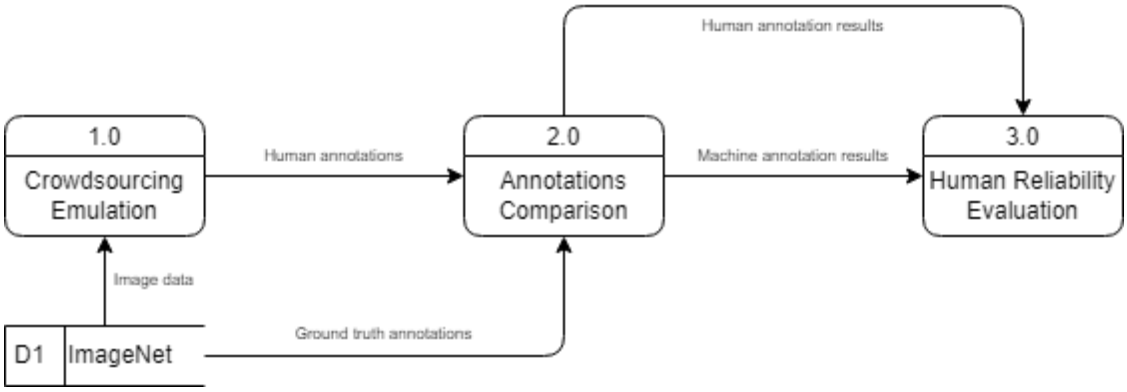


Figure 4.1: Basic pipeline split into each individual module

they can be used for this purpose. Using a library called Robustness[1], the parent of an overly specific label was uncovered through the information provided by the relevant synset. Robustness achieves this by providing presets which detail the final parent categories. Among the presets available, this study selected the "big_12" superset due to its sensible variety of superclasses. The big_12 superset contains the following classes: dog, structure, bird, clothing, vehicle, reptile, carnivore, insect, instrument, food, furniture, and primate. In total, the "big_12" superset provides 128,763 training images.

4.1.1.2 | Interactive Web Survey

Replicating a crowdsourcing workflow is essential to uncover image annotation behaviour. Hence, this study incorporated a web application as a survey to emulate crowdsourcing. Examples of such systems include Amazon Mechanical Turk[2] and Revolt [10]. Care was taken to design the website with appropriate UI and UX, as guided by previously discussed studies [9, 10]. Besides image annotation, the application recorded additional details relating to user behaviour through various time intervals. This supporting set of results helped identify possible weaknesses within human performance.

Participants As seen within Good et al.'s study, even non-subject experts can provide valuable data [9]. Therefore, it was decided to not selectively choose the participants based on their data annotation experience. But rather, employ the efforts of annotators of all experiences. This would ensure a good distribution of annotation submissions. Just like ImageNet's crowdsourcing process, many participants will also be able to annotate many images. Each image can have multiple annotation submissions from many people, providing a diverse range of answers [2].

Load balancing Section 4.1.1.1 established that a library known as Robustness was utilised to simplify ImageNet. This was achieved this by superclassing ImageNet into a more legible dataset. While the the number of classes decreased substantially, the image count was still too large to include within the survey in its entirety. Therefore, the web application used only a small sample from the validation subset for the survey. Furthermore, this sample of images was equally divided between the big_12 categories. This even distribution of labels ensured a balanced subset. The validation subset also served as the testing subset, as described further in Section 5.3.1.2. In total, the survey consisted

[1] https://robustness.readthedocs.io/en/latest/
[2] https://www.mturk.com/

of 120 images. While the number of images used was acceptable, this also presented an obstacle as most participants would be discouraged from annotating that many images at once. Therefore, the web application implemented a method known as load balancing. As the data flow diagram (DFD) in Figure 4.2 shows, load balancing ensures a fair distribution of records by analysing the weight of each record type. Functionally, this technique prioritises images that have fewer entries than others. As a result, the individual participant did not need to annotate the entire survey. Additionally, if a participant dropped out of the survey, their incomplete set of annotations did not bias the results to a particular image.

Classification segment Section 1.3 implied that both labelling and localisation times must be recorded. Theoretically, image-level labels take less time to annotate than localisation annotations. While this assumption is reasonably sound, the scale of this difference is unknown. Hence, it was decided that this survey would be made up of two parts. The first part required participants to annotate image-level labels, while the second part required users to annotate localisations. The web application recorded the time taken to perform both annotation types separately. The time the user spent deliberating their category was also recorded as this could provide insight into a user's behaviour when labelling images.

Localisation segment As discussed, in the second part of the survey, the participants performed object localisation. The components forming this section were essentially

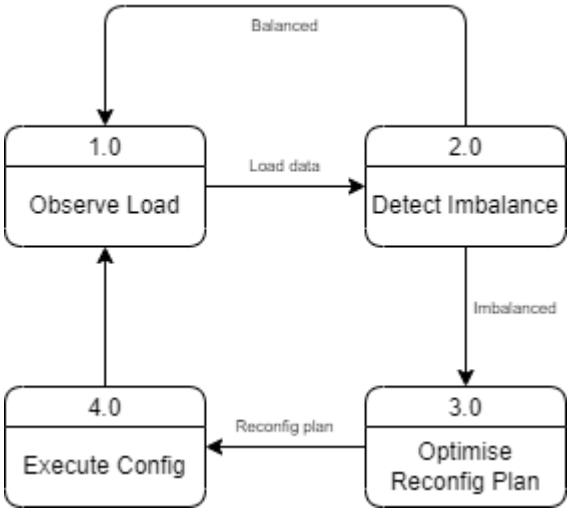


Figure 4.2: Generic load balancing DFD [161]

identical to those constituting the classification section; however, the former also required users to draw bounding boxes. As in object detection, these boxes surround the objects of interest. Furthermore, the time taken to perform these localisations was recorded to further analyse human annotation behaviour when it comes to bounding boxes.

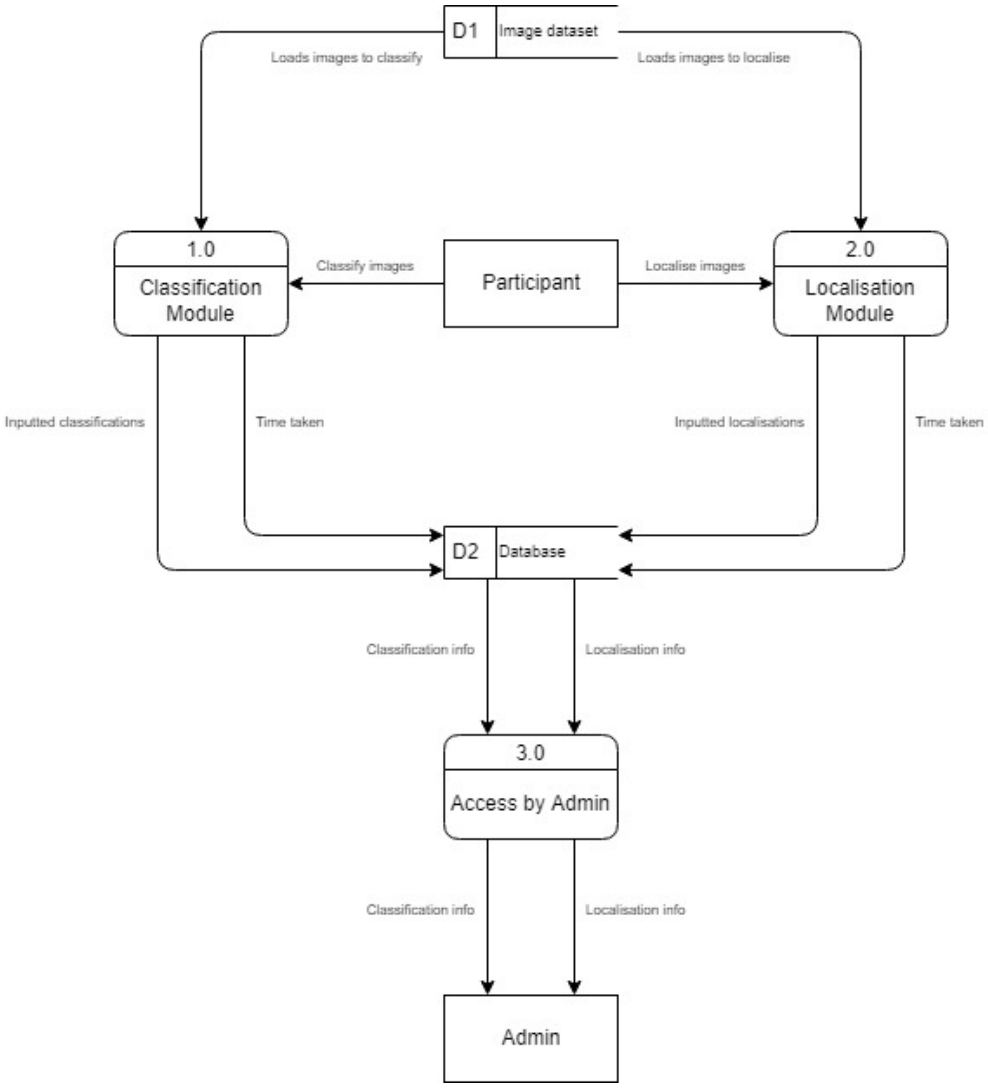


Figure 4.3: Interactive web survey DFD

Application As per the requirements, this web application's functionality was more complex than a standard survey's. For instance, Google Forms[3] and SurveyMonkey[4] are popular choices for publishing surveys; however, they specialise in simple forms or questionnaires. Advanced functionalities, such as user tracking and load balancing, are unavailable in off-the-shelf solutions. Figure 4.3 visualises this application in further detail.

Database Additionally, the web application incorporated a database. This database stored a variety of information, including the user timestamps. Moreover, the database was normalised to efficiently store these data. Figure 4.4 shows the database split into five tables. This study designed these tables so that they could be easily modifiable, yet still accurately representing the survey data.

- The Images table was responsible for storing each image that a participant can

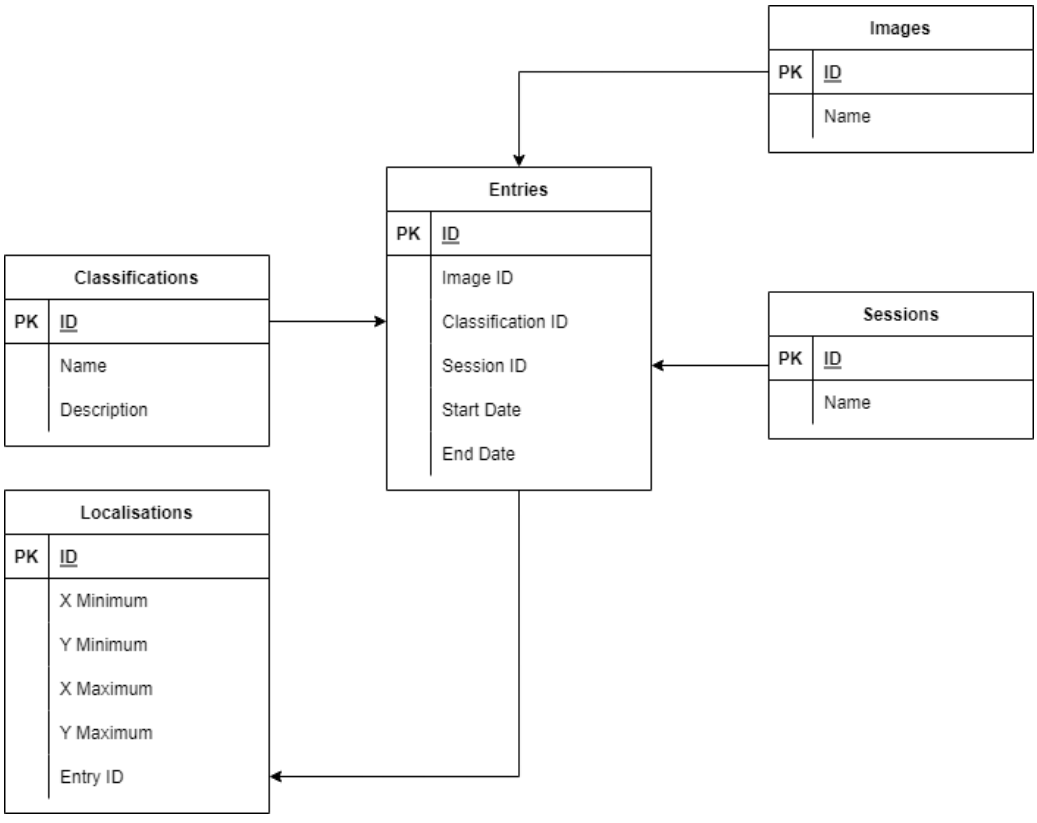


Figure 4.4: Interactive web survey database diagram

[3]https://docs.google.com/forms/u/0/
[4]https://www.surveymonkey.com/

view. This is applied to both the classification and localisation segments. Furthermore, the table used references to the images rather than the images themselves so as to simplify development.

- The Classifications table stored all the possible labels that could be chosen for the survey subset. More specifically, these were the big_12 categories detailed in Section 4.1.1. Moreover, this table also provided the "none of the above" option. This was implemented to allow the participants to express their disbelief at the categories' fitness to an image or their confusion towards an image's contents. This was inspired by the reasoning behind a similar option in Chang et al. [10]'s crowdsourcing application.

- The Sessions table was used to anonymously identify a user's attempt via their browser session cache. This was mainly used to roughly estimate the number of people who actually participated in the survey. However, it was also useful for other auxiliary tasks. For example, it was able to reveal how many participants truly completed the survey.

- The Entries table joined all the previous tables together. It was used to save each participant's individual entry. Each record showed the individual image loaded, the time taken for a participant to choose a category, and even whether the entry was considered as ground truth for evaluation purposes.

- One of the survey's main requirements was that classification entries and localisation entries need to be separated. To do so, the Localisations table was developed to identify whether a specific entry contained bounding boxes. If it met this requirement, then it was treated as a localisation entry. The rest were considered as classification entries. Additionally, when applicable, this table also saved the actual bounding boxes associated with the entry. This design also allowed the participants to create more than one bounding box per localisation entry.

4.1.2 | Annotations Comparison

Following the data collection process, the human annotations were compared with the machine annotations. The implementation generated these machine annotations from CAM-empowered weakly supervised CNN object localisers. This study evaluated these models by subjecting them to the same survey subset. Subsequently, both types of annotations were evaluated based on the ground truth.

4.1.2.1 | Preprocessing ImageNet for Models

Much like how preprocessing was required for the survey as detailed in Section 4.1.1.1, the images used for the algorithms needed to be preprocessed as well. Nevertheless, the survey sample was too small to use for this purpose. Since CAMs require a trained model, the training subset was preprocessed instead. Afterwards, the models used this subset for their training sessions. The implementation further preprocessed the images due to the architectures' requirements. Each algorithm has a specific method of preprocessing an image for inference. For example, VGG16 requires images to be mean averaged [35]. Meanwhile, MobileNetV2 [88] requires images to be normalised. Fittingly, the images were preprocessed according to which algorithm was being used at the time. This assured that the inference provided the most accurate prediction possible.

4.1.2.2 | Models

Section 3.2.2 reviewed many distinct model architectures. Each architecture has its own qualities, which serve to improve the state of the art. Thus, it is logical to consider them prime candidates for CAMs. To determine the ideal models, a number of conditions were presented. The primary requirement was that the models had been implemented as localisers. Therefore, models presenting a localisation score meet this prerequisite. Due to the dated nature of LE, models that were used as base networks for object detectors were also considered. Additionally, the models must have been pre-trained on ImageNet because the test set consisted of ImageNet samples. These conditions pruned the eligible classifiers to the following models:

- **VGG16**: The algorithm has been documented to be effective for a vast range of tasks, including image classification, object localisation, and object detection. While the algorithm is quite dated, it is one of the few algorithms that present a localisation score [35, 40, 115, 41].
- **MobileNetV2**: It has been widely used in applications where inference speed is prioritised or where edge devices are required. In other words, it is quite efficient and fast. It has also been used in object detection as a component in SSD [88].
- **EfficientNetB0**: A relatively recent and effective algorithm that, in fact, used CAMs to evaluate its effectiveness. Furthermore, it was also applied to object detection as the backbone for EfficientDet [96, 42].

This was the final selection of algorithms that were trained for this study. Most importantly, they all presented the potential to be effective in weakly supervised learning.

Furthermore, various CAM techniques were combined with these models to produce the best possible results. Figure 4.5 shows how the models were included in the digital system.

4.1.2.3 | Class Activation Maps

In Section 3.3.4, some weakly supervised methods were presented in view of this study's goal. Ultimately, CAMs were deemed to be the superior choice due to their flexibility and practicality. EfficientNet, for example, used CAMs to observe the effects of customising the algorithm's architecture [96]. They are also easy to use; indeed, they involve plugging the technique into the model after inference. The only exception to this is the original CAM method, which requires a GAP layer for it to work, rendering some

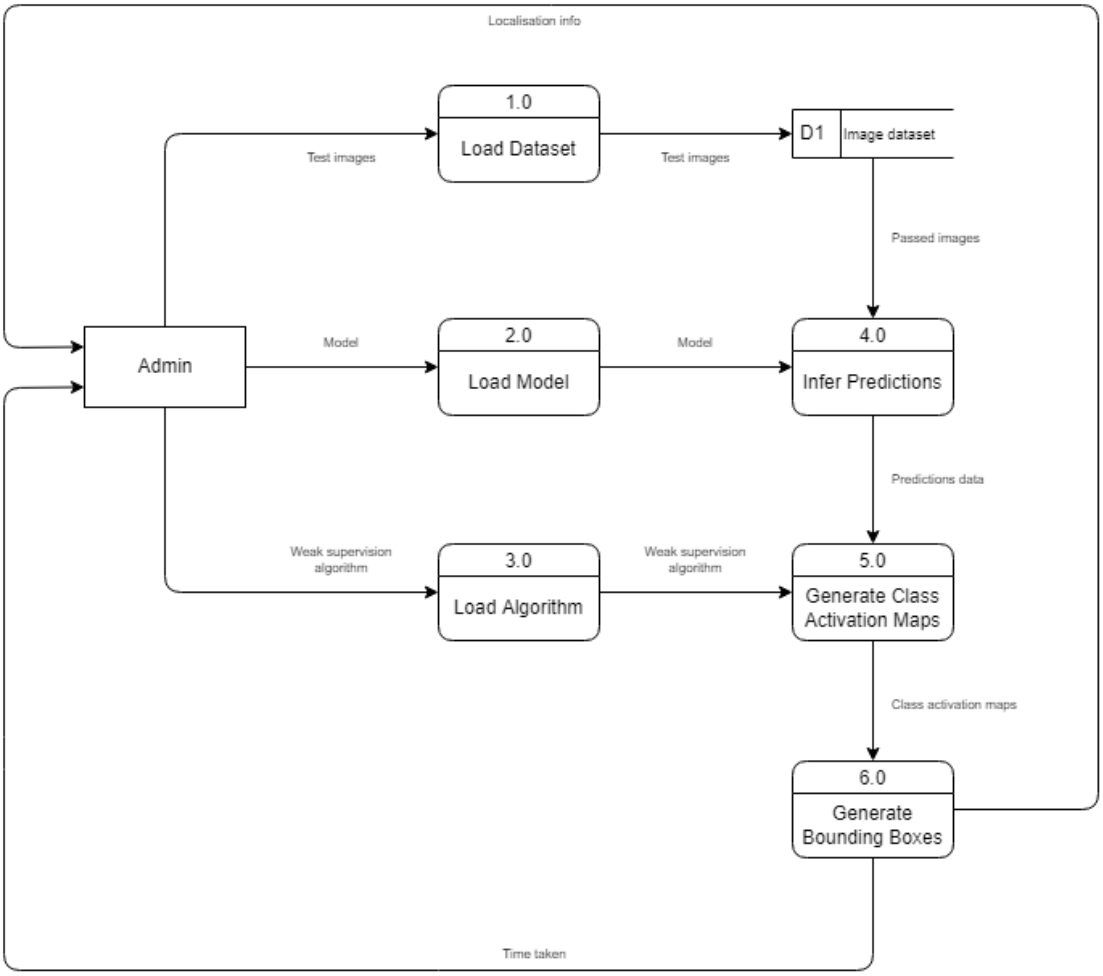


Figure 4.5: Weak supervision model inference DFD

models, such as VGG16, incompatible in their default state [35]. Consequently, the techniques chosen include all the reviewed CAM methodologies except the original one:

- **Grad-CAM**: uses model gradients to improve the generalisation of the result while also removing the need for GAP layers [156]
- **Grad-CAM++**: claims to improve Grad-CAM's accuracy by incorporating pixel-wise weighting [158]
- **Score-CAM**: removes the gradients in favour of a score-based approach [159]
- **Faster Score-CAM**: identical to Score-CAM [159] with the exception of limiting the extraction of effective maps[5]

This selection of CAM methodologies provided a variety of techniques, thus increasing the likelihood of achieving an effective weakly supervised object localiser. Moreover, all these CAM methodologies have no prerequisites. The fact that they also need no extra training steps, simplified the development and further highlighted their utility. To combine the models and CAMs, each model architecture was branched to provide two results. The first branch outputted the predicted label, while the second branch outputted the input's features. The input's features were then fed to the relevant CAM technique to output a heatmap based on the model's predicted label. This heatmap was then converted to bounding box data by using Otsu's threshold [162].

4.1.2.4 | Hyperparameters

Section 2.1.1 detailed that CNNs have their weights altered to fit a training sample. These training sessions were crucial for the final performance of the algorithm. If the training session had been performed incorrectly, model performance would have suffered. Hyperparameters are the variables that dictate the configuration of the training session. To ensure optimal performance, they were determined with the utmost precaution by using their original configurations as a basis and a helper tool to further optimise them.

Workstation The training sessions and other evaluation software were executed on the main workstation. The following are the main workstation's most relevant specifications:

- GPU: EVGA GeForce GTX 1070 Superclocked Gaming ACX 3.0 (8 Gb)

[5] https://github.com/tabayashi0117/Score-CAM

- CPU: Intel Core i7-6700 (3.4 Ghz)
- RAM: G.Skill Ripjaws V DDR4 (2 $\times$ 8 Gb)

Due to computational power concerns, online notebook services like Kaggle[6] and Google Colab[7] were considered. Nevertheless, these were not used due to the data required to train the algorithm. As mentioned in Section 4.1.2.1, the training subset of the simplified ImageNet was used. The subset alone contains over 120,000 images, exhibiting a large computational size. It would have been infeasible to upload such a large dataset to one of these services. Even alternative methods like uploading to Google Drive[8] would have been impractical. Consequently, this study's methodology settled on using the main workstation. The experiments were carried out via the GPU, rather than the CPU. However, further care was taken when determining some of the hyperparameters, for instance, the image resolution and the batch size. The image resolution determines the final array of pixels that enter the algorithm. For this study, the common 224 $\times$ 224 pixel size was used due to it being the main resolution used by all the algorithms mentioned in Section 4.1.2.2. The batch size used was 16. This meant that during training, the images were supplied to the algorithm in groups of 16. Such a large payload taxed the workstation's memory. If the batch size was larger than the specified amount, the experiment would have run out of memory during execution.

Models Since this study made use of pre-trained models, it attained better model performance. Moreover, these algorithms were researched thoroughly by considering their features. Such configurations also included the hyperparameters used for ImageNet training. The optimiser is one of the most prominent of these configurations. As mentioned in Section 2.1.1, NNs use weights. These weights can be represented through a function. The goal of the training session is to minimise this function's error by finding the ideal set of weight values [163]. However, this process is computationally complex. Altering each weight for each batch for many epochs would take an exorbitant amount of time. To mitigate this, architectures use the aforementioned optimisers. Aptly named, optimisers hasten an algorithm's training session with little repercussions. While there are many available optimisers [163], only the original models' optimisers were considered as they presented the best model performance at the time, thus offering a good likelihood of optimal model performance:

- VGG16: Stochastic gradient descent (SGD) [35]

[6] https://www.kaggle.com/
[7] https://colab.research.google.com/
[8] https://drive.google.com/

- MobileNetV2: RMSprop [88]
- EfficientNetB0: RMSprop [96]

Combining these optimisers with the pre-trained models served to boost model performance. To load the architectures, the pre-trained AI models' weights were applied so as to avoid having the framework randomly allocate weight values. Altogether, these precautions substantially improved the models' training. Simplifying the training of these models, further showed the practicality of CAMs.

Hyperparameter optimisation Effective CNNs take a considerable amount of time to train. Due to the abundance of training images, this study's training sessions were lengthy. Consequently, experimenting with the hyperparameters would have exhausted all available resources. Hence, this experiment employed several techniques to shave off any unnecessary time. Since the optimiser is simply a function to efficiently find a function's minima, it has its own parameters. The chosen optimisers consisted of a common set of hyperparameters, including the learning rate and the momentum. These hyperparameters are both numerical, meaning that both of them have an extremely large number of possible values. To avoid finding their ideal values via trial and error, systematic hyperparameter selection was explored. Eventually, this effort yielded an appealing tool known as KerasTuner, a hyperparameter tuning tool that uses an oracle system [164]. The tool's purpose is to automate hyperparameter tuning. In turn, this would ease the usual burdens of the process. Thus, this study used KerasTuner to efficiently discover the optimal hyperparameters. The oracles included in this tool use either Bayesian optimisation, Hyperband, or random search. After defining the search space, a researcher can use an oracle to find the optimal hyperparameter combination. Random search indiscriminately tests a combination from the search space for a set amount of trials. While this method can be useful in most applications, it can still lead to less than ideal results. Meanwhile, Bayesian optimisation is a well-known method in AI research. While it is a competent method for expensive, inexact methods, it does require some trial and error to be effective. For example, the Bayesian optimisation class in KerasTuner exhibits an "alpha" parameter. This parameter modifies the value of the kernel applied during the process. Finally, there is the Hyperband method, which is an optimisation method implemented specifically for hyperparameter optimisation. It uses a novel bandit-based technique to find the optimal hyperparameters by executing trials with the goal of stopping them early on to preserve resources. It then prioritises the methods that show the most prospects. This accelerates the process as it would not need to wait until a full session is complete to move on. Due to this key advantage, this study used Hyperband

for hyperparameter tuning. Furthermore, this study created a new subset specifically for this process. The tuning subset is a sample that is a tenth of the size of the training subset. Since the tuning subset still had over 12,000 samples, it still provided good generalisation. Upon implementation, the Hyperband oracle was set to run at a maximum of only 10 epochs per model. This experiment used such a low epoch value due to the already trained nature of the pre-trained models. Additionally, since the chosen dataset used similar samples to ImageNet, the models did not require much processing to adapt to the new dataset. Other parameters, such as "factor" and "full iteration", were left at 3 and 1, their respective default setting. After this, KerasTuner was executed on all the models. This process is discussed in further detail in Section 5.3.1.2.

4.1.2.5 | Other Training Options

Upon the hyperparameter optimisation's process completion, this study trained each image classifier. The optimiser consisted of each model's ideal hyperparameters. While KerasTuner did efficiently discover the ideal hyperparameters, this study carried out further precautions by fine-tuning the models, for example. Since the experiment applied pre-trained models, they had already generalised ImageNet. The image samples used were from the same dataset with the only difference lying in the classifications. Thus, the models' feature extractor was "frozen". This process converted the model to a classifier. In other words, the training session only altered the classifier weights. Since this reduced the models' parameters, it also reduced the models' overall complexity while also saving time. In addition to this, pre-trained models that have a similar objective should not have their feature extractors re-trained [165]. This is because any high-level features would most likely be similar. Therefore, re-training the entire model would run the risk of losing the already suitable weights. The final precaution consisted of applying callbacks, which are high-level functions offered by the framework. They are usually applied to perform changes to the training session in real time. For example, the experiment used a callback that defined a termination condition. If the model failed to improve after a number of epochs, the training session stopped. This ensured that the final AI model presented minimal overfitting. In this case, the callback made use of the validation subset's loss value. If the error did not decrease after three epochs, then the training session would have ceased. The experiment also employed another callback that saved the weights of the model to auxiliary storage. This callback offered similar functionalities to the previous callback; it could be configured to only save the best performing model. By using the same condition as the previous callback, the best model checkpoint was obtained without any manual intervention. Afterwards,

the saved models were applied for further evaluation.

4.1.2.6 | Evaluation

As discussed in Section 2.1.3, the ILSVRC LE metric is essentially a combination of classification accuracy and IoU. This metric was used to evaluate the weakly supervised object localisers as the ImageNet dataset was used with its object localisation bounding boxes as the ground truth. Algorithm 1 showed the inner workings of the LE metric considering both its classification and localisation components, producing a result that combines classification and localisation performance. Delving deeper into the logic behind Algorithm 1 would reveal Equation 4.1. Since some of prediction and ground truth combinations that are iterated throughout the algorithm may be illogical, all the possible results are compared. The minimum error of these possibilities is returned. However, the results that would return are specifically for each prediction bounding box. According to the official ImageNet guidelines, a match is obtained when the predicted bounding boxes match at least once [2, 43]. If such a match is found through the minimum error of the predictions, then $e = 0$; otherwise, $e = 1$. This seems partial, however, as the metric would be ignoring some possible FPs and FNs. This equation is also identical to Equation 4.1, which is applied in the same manner as Equation 2.1:

$$e = \left\lfloor \frac{fp + fn}{tp + fp + tn + fn} \right\rfloor \tag{4.1}$$

LE is essentially an inverse version of accuracy. In addition to this, it removes any incorrect answers by flooring the outputted value. Flooring the outputted value ensures that a wrong prediction only exists when no TPs exist. Avoiding inaccuracies in this manner is prone to hiding inconsistencies within the results. Therefore, modifications were applied to LE to improve on its weaknesses. The flooring function was removed, for instance, as presented in Equation 4.2. This newly modified metric represented the performance of the models more accurately.

$$e = \frac{fp + fn}{tp + fp + tn + fn} \tag{4.2}$$

Additionally, removing the relevant type of error could help reveal further information. Removing FPs from the equation produces a recall-focused result, whereas removing FNs reveals the error in a similar manner as precision. These variations of the same equation were utilised when evaluating the results. Nevertheless, it was decided that the original LE should be preserved. This would enable this study to provide comparable and reproducible results.

4.1.3 | Human Reliability Evaluation

This study's final experiment determined the reliability of human annotations. Thus, a method had to be established to further verify the results. Indeed, verification of the previous experiment was required due to a contradiction. While the results generated from the previous experiment were objective, this could only be true when assuming that the ground truth is accurate. It could be argued, however, that such an assumption is unreasonable, as that ground truth had also been annotated by humans, thus giving rise to a paradox as the experiment used human annotations to evaluate both humans and machines. To resolve this bias, this final experiment consisted of a web form to gather more primary data. The web form included various annotation samples from both humans and machines. This experiment required the participants to choose the annotations they thought were man-made. If the participants were more likely to choose machine annotations over human ones, then it would have been proven that humans are inconsistent. Therefore, their annotations would be unreliable, and they should be aided by some form of a digital system.

4.1.3.1 | Web Form

As discussed, more primary data were gathered via a web form to resolve the aforementioned paradox. Since this form only required a few multiple-choice questions based on the web application's entries, this study used off-the-shelf solutions. The service known as Google Forms[9] proved to be acceptable for this purpose because it excels at simple form creation. Other options like SurveyMonkey[10] were considered, but Google Forms was deemed more user-friendly.

Participants The web form's main priority is to evaluate individual submissions from both the interactive web survey's participants and the model-CAM's submissions. With this known, the same philosophies used within Section 4.1.1.2 were applied to this section of the study as well. Specifically, the reasoning that even non-subject experts can provide valuable and possibly significant discoveries [9]. Therefore, this set of participants was also not selectively chosen so as to evaluate their performance as representatives of the general public.

[9]https://docs.google.com/forms/u/0/
[10]https://www.surveymonkey.com/

4.1.3.2 | Requirements

The new web form had to be a structured solution. Primarily, the form needed to address the subjectivity issue of the previous experiment, the main problem being that the ground truth was also annotated by humans [2]. Therefore, the annotation samples were the most important aspect to be determined. Since the ground truth is the core of this issue, the participants could not know the "accuracy" of the annotation. This was circumvented by shifting this experiment's focus; indeed, the form tested the participants instead of having them offer their opinion. If the participants were more likely to choose the machine annotations, then the results would imply that machines can offer human-like performance. Otherwise, human annotations would have been both distinct and consistent. Additionally, the form consisted of more questions. The first set of additional questions served to extract the participants' demographics, while the second set asked the participants for their opinion on the form. To summarise, the web form was divided into three sections: the participant demographic questions, the annotation choices, and the sentiment-oriented questions.

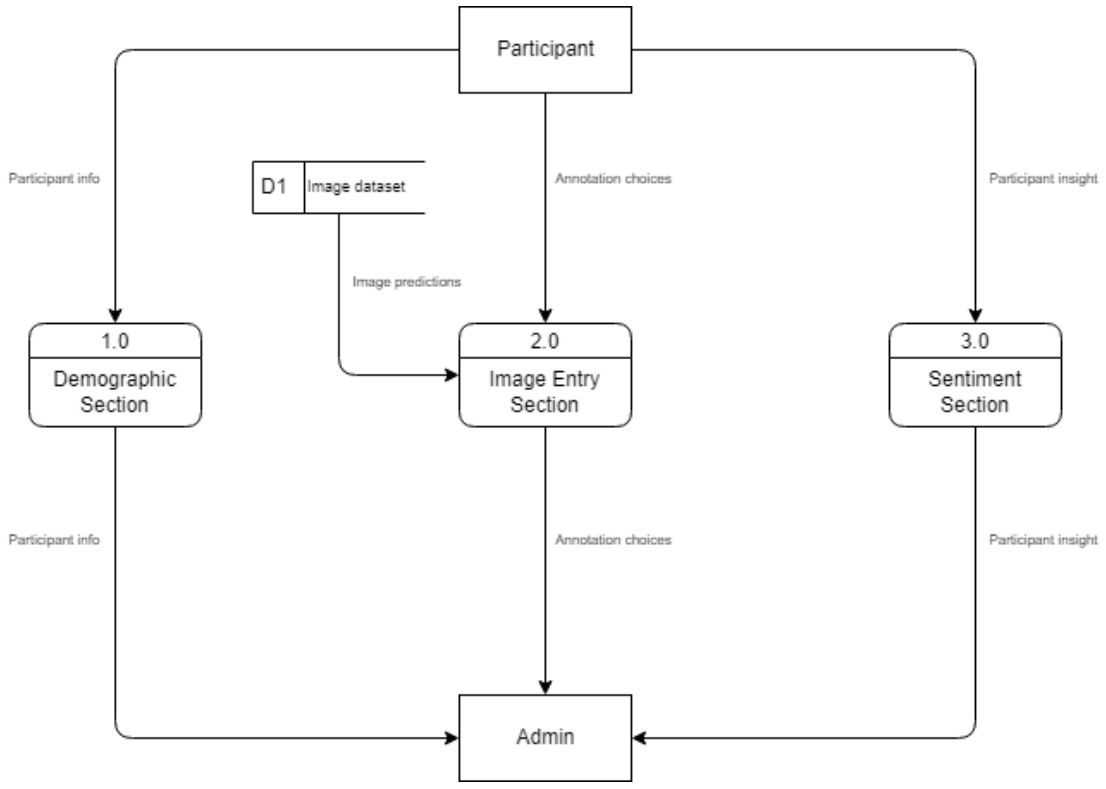


Figure 4.6: Web form sections DFD

Participant demographics The first questions of the form dealt with the auxiliary goal determined in Section 4.1.3.2, which involved gaining insight into the participants themselves. This was done by asking a few demographic questions, concerning the participants' age and gender, for example. Additionally, the form asked the participants to rate their own image annotation abilities:

> "How highly do you think of your own image annotation abilities?"

The goal of this question was to categorise the participants by their confidence, as this could have divulged further information on how users perform based on their self-assurance. Furthermore, this question gauged the general public's overall expertise in data annotation.

Organic vs artificial annotations The second section presented the main focus of the form. The participants were required to distinguish the human annotations from their machine counterparts. To reduce the frequency of dropout rates, this experiment did not use the full survey subset. Instead, the number of questions corresponded to the number of big_12 labels, ensuring that the form presented a good selection of images. Since this study used the simplified ImageNet dataset, the number of questions added up to 12. Since there were more annotation samples than labels, a selection process was carried out. The experiment performed multiple steps to ensure an unbiased set of annotations:

- Annotations from both the survey database and models were listed in auxiliary storage.
- For each label, an eligible image was randomly selected. Eligible images were pruned by assuring that the image had at least one:
  - correct survey annotation;
  - inaccurate survey annotation;
  - correct machine annotation; and
  - inaccurate machine annotation.
- For each image, a random set of eligible annotations was selected.
- For each annotation, an image was generated with only the predictions visible.

To determine whether an annotation was correct or not, this study used a metric. Section 5.3.2 presents more information about this. For each question, there were a total of

four images. Half the image annotations were man-made, while the rest were machine-generated. The form required the participant to choose exactly two of these images to progress. This ensured that the test was impartial. Subsequently, the questions under this section assumed the following template:

> "Which of these 2 (out of 4) images do you think were localised / labelled by human annotators?"

With this, the participants were able to fairly showcase their observational abilities. Combining the results from this section with the demographic questions revealed further information. Additionally, this newly acquired primary data provided a clearer outlook on human annotation behaviour.

Sentiment questions The final section of the form presented questions related to the participants' opinions. The first question asked them about their experience with the previous section of the form. The participants answered this question by quantifying the perceived difficulty via a range:

> "On a scale from 1 to 5, how difficult was it determining your choices in the previous section?"

This question could potentially offer information concerning two factors. Firstly, it could have helped reveal a relationship between the participants' performance and their confidence. Secondly, the results could have also revealed a trend between confidence and perceived difficulty. Additionally, this range was also used to gauge the participants' expected performance. Since the form asked its users to choose two out of four answers, each question had a 50% chance of being answered correctly. If required, the confidence rating question could have been used to weigh the responses. The second question of this section was a direct enquiry about this study's purpose, asking for the user's opinion, experienced or not, on whether they were willing to use a dataset annotator helper:

> "Would you use an AI-empowered image annotation tool that annotates less effectively than humans by n%, but saves m% of the time required to annotate l images?"

Variable n represented the percentage difference in annotation performance as indicated by the metric. The second variable, m, represented the time advantage as a percentage. Finally, l showed the number of images these values were applicable to. In terms of responses, this question had four choices in total. Two responses alluded to a positive

notion, while the other two alluded to the opposite. The following are the different answer choices:

- Yes, I'd use it as is to completely automate the annotation process
- Yes, I'd use it, but by manually improving its annotations later on
- No, but I'd consider it if the accuracy-time tradeoff is better
- No, I'd rather take that much longer creating my own annotations manually

Each set of two choices represented the individual participant's sentiment towards a dataset annotator helper. The aim of this question was to reveal further information while avoiding the usage of text input to help reduce dropout likelihood. The responses to this question could also be used to categorise the results. This could reveal whether or not the participants' responses reflected their performance. If a positive sentiment exhibited itself, it would further legitimise this study.

4.2 | Scope

This study was focused on testing weakly supervised models as dataset annotator helpers. Therefore, it was not within this study's scope to extend the applied techniques. This meant that the models were localisers and not detectors. Section 6.2.2.1 presents further information about this matter. Moreover, this study approached the aim through an experimental context. Due to the ImageNet simplification effort, it was difficult to compare the results to similar methods such as Snuba [15]. This issue is discussed in more detail in Section 6.2.1.2.

4.3 | Ethics

Both surveys discussed in Sections 4.1.1 and 4.1.3 were completely anonymous in nature. Indeed, security measures were taken while developing the web application to ensure the integrity of the connection. Such measures included practising proper web hosting and following responsible development procedures. Additionally, the ImageNet dataset was acquired through the official sources, thus ensuring that it was procured and processed in an ethical manner. Therefore, no ethical laws were broken or abused to reach this study's aim.

4.4 | Expectations

Although CAMs can potentially act as dataset annotator helpers, this is only a possibility. As mentioned in Section 3.3.4.4, CAMs are primarily used for AI explainability. In their official papers, they have been presented as such [154, 156, 158, 159]. The researchers behind them presented their localisation potential as a more hypothetical application. This study exploited this to observe their performance as localisation components. Consequently, it was expected that CAMs could be largely inferior compared to human annotations; however, if this was the case, this would not imply that CAMs are useless for this application. It could very well mean that they are simply not ready yet for such an application. Further research into the CAM technique could yield better results.

4.5 | Summary

This chapter described each module's process in detail, keeping their respective objective in mind. Each individual objective was represented as an experiment through a pipeline. The first module discussed the functions of a web application. In addition to emulating crowdsourcing, the web application tracked user performance anonymously. This was done via UI event timestamps. Upon the experiment's completion, these were processed to evaluate the time intervals in question. The second module detailed the training processes of the AI models. Both the survey's and the models' performance were compared through small programs. Finally, the third module verified these results to avoid a paradox. Since the ground truth is man-made, then a bias was present. To resolve this, this study published a web form to gather more primary data to verify the results acquired. Altogether, these modules presented the steps required to achieve this study's goals.

5

Results

Chapter 4 presented the designed experiments that were employed to achieve this study's goal. As discussed in Section 1.5, this study's aim was to test the basis of a dataset annotator helper system. To do so, human annotations were compared with the annotations created by weakly supervised object localisers. By using various metrics, this study yielded a detailed result set. For example, a metric known as LE revealed the error rate of both humans and models. Since LE is not predictive, this study employed more metrics, such as the F1-score. From there, the findings were discussed in view of this study's objectives.

5.1 | Criteria

This study clearly defined the evaluation criteria based on the objectives provided in Section 1.5. Each module yielded a different result set. Therefore, each experiment outputted a result set in accordance with its respective objective. The following subsections briefly summarise the criteria considered so as to provide sufficient preamble to the discussions that follow.

5.1.1 | Crowdsourcing Emulation

The first objective required this study to observe human annotation behaviour. This was achieved via a crowdsourcing experiment in the form of a web application. In addition to annotations, the web application recorded various timestamps related to the users' actions. These timestamps produced behavioural information based on the time taken to create annotations. Section 5.2 delves into further detail about these discoveries.

5.1.2 | Annotations Comparison

The second objective required empirical evidence concerning both human and model annotation performance. This was obtained by comparing the model predictions with the human annotations. This evaluation quantified both annotation sets' performance via metrics like LE. Additionally, predictive metrics like precision and recall were also used. Such metrics helped with further analysing the results. Applying these metrics in different ways also revealed further information, for example, by analysing each bounding box rather than an image as a whole. This revealed annotation performance on a per-annotation basis rather than on a per-entry basis. These results are presented and discussed within Section 5.3.

5.1.3 | Human Reliability Evaluation

The third objective concerned the verification of the previous results. Section 4.1.3 described a contradiction that could have invalidated this study. To further support the acquired results, this study used a web form. This web form gathered more annotation performance data from the general public, as human reliability could be quantified by asking the participants to pick the man-made annotations. Furthermore, demographic and sentiment-oriented questions were presented to categorise the participants. These categorisations revealed trends that could yield further information. Section 5.4 presents the results of the final web form in more detail.

5.2 | Crowdsourcing Intervals

The web application presented in Section 4.1.1 was an integral component of the study. This is because it served as the basis for the results in Sections 5.3 and 5.4. After accumulating 5606 image annotation entries, various intervals were extracted via the recorded timestamps. These intervals were used to analyse participant performance throughout the survey. For example, the average time taken for both labelling and localising was compared.

5.2.1 | Labelling vs Localising

As articulated in Section 4.1.1.2, the web application was equipped to track user performance. Particularly, the application recorded the time taken for a participant to complete individual entries.

5.2.1.1 | Specific Conditions

When focusing on the full entry times, the participants were recorded as follows:

```
start_date = image.onload() { return datetime.current() }
end_date = form.onsubmit() { return datetime.current() }
total_time = end_date - start_date
```

The web application saved the initial time once the image had loaded rather than when the page loaded. This is an important distinction to note as otherwise, the recorded time would have been biased; no participant would have been able to appropriately identify the category of an image before it had even loaded. Even if such an event happened, this would have only meant that the participant had guessed at random. This line of logic was also applied to the entry's end condition. While there would always be a delay when storing the results of an entry in the database, this was of no consequence to the user. This delay could also differ depending on the programming practices, tools, and frameworks used. The participant finalised their entry by confirming their submission.

5.2.1.2 | Raw Times

As discussed in Section 4.1.2, the implementation focused on comparing human and machine annotations. This step included aggregating the database of the survey into legible result sets. This study achieved this by implementing a small number of modular programs that could present organised results through specific conditions. For example, the programs could have been configured to output a table averaged by each survey image while:

- excluding "none of the above" entries;
- excluding labelling entries; or
- showing only "none of the above" localisation entries.

Through these conditions, the programs generated the results for evaluation. Table 5.1 represents the averaged full entry time results. The presented results show the expected outcome. In almost all instances, image labelling took less time than object localisation. Based on these results, there is a difference of 7s when excluding "none of the above" entries. Initially, this value may seem small considering that this is the average; however, data annotators carry out their work on thousands of images. Therefore, that time difference would have a considerable effect, as seen in Section 5.3.2.4. When including all entries, there was a general increase in the time taken of around 1s. This was

Configuration	Labelling	Localising
E	$\approx$ 17s	$\approx$ 23s
$E - U$	$\approx$ 15s	$\approx$ 22s
U	$\approx$ 31s	$\approx$ 30s

Table 5.1: Interactive web survey full entry times. Set E represents all individual and identifiable full entry times. Set U contains all individual and identifiable full entry times for entries marked with "none of the above".

expected as those participants that chose "none of the above" likely took their time to come to that conclusion. This is further supported when considering that typically, the participants took 30s to complete their entry with "none of the above". In summary, these results provide two key pieces of information:

- Image labelling usually takes less time than object localisation.
- Humans are liable to second-guess their choices, even if it comes at a quantifiable cost.

This alone alludes to the potential of CAMs. CAMs only require the image-level labels to be manually annotated. Therefore, a substantial amount of both time and effort can be saved. Additionally, annotators are likely to express less doubt as there would be fewer annotations to perform.

5.2.2 | Human Deliberation

The time taken for the participants to make their choices was also recorded. Depending on the survey segment, different conditions were used to record them. Since the participants spent time navigating the webpage, alternative intervals were used. These alternative timestamps, which represented the participants' optimised times, excluded the time taken to perform unnecessary actions like:

- the time taken to start performing the first annotation; and
- the time taken between finishing the last annotation and submitting the entry.

5.2.2.1 | Adjustments Performed

In the classification segment, a select element informed the participants of their label choices. Analysing the select element's events was integral to recording these times. The following pseudocode is presented to further describe this logic:

```
start_cls_date = select.onfocus() { return datetime.current() }
end_cls_date = select.onunfocus() { return datetime.current() }
cls_time = end_cls_date - start_cls_date
```

This pseudocode retrieved the exact time taken for the participants to lock in their choice. This was recorded for two main reasons. Since the participants were human, they were expected to present periods of inactivity like looking at the image before performing any annotations, for example. Excluding these unnecessary moments optimised the participants' entries. The same logic was applied to the localisation segment. However, the latter allowed users to draw bounding boxes through a canvas element and a library known as fabricJS[1]. In addition to the previous pseudocode, the following logic was also added to the localisation part:

```
start_loc_date = canvas.oncreate() { return datetime.current() }
end_loc_date = canvas.onmodify() { return datetime.current() }
loc_time = end_loc_date - start_loc_date
```

To effectively calculate the localisation time, the dominant time type was negated from the full entry time. The programs achieved this by using the earliest and latest entries, no matter whether they clicked on the label select element or the canvas element first. Further information about these calculations can be seen in Section 5.3.2.4.

5.2.2.2 | Optimised Times

The implementation presented in Section 4.1.2 also catered for tasks other than finding the full entry times. For example, the optimised entry times were recorded through it as well. These optimised entry times removed the participants' unnecessary movements and delays, thus enhancing their performance. Table 5.2 presents the optimised entry times. It can be observed that there is a relatively noticeable decrease in time when optimising the entries. For all the entries and the entries excluding the unknown answers,

Configuration	Only labelling	Only localising
E	≈10s	≈17s
$E - U$	≈9s	≈14s
U	≈18s	≈23s

Table 5.2: Interactive web survey optimised entry times. Set E represents all individual and identifiable full entry times. Set U contains all individual and identifiable full entry times for entries marked with "none of the above".

[1] http://fabricjs.com/

there is a 6s - 8s reduction in the time taken. For the unknown entries, there is an average decrease of 10s between labelling and localising. Such small differences might seem inconsequential at first, but this small gap would exponentially increase on a large image set. Furthermore, it can be seen that the participants took a relatively long amount of time to choose, implying that humans spend a relatively long time deliberating their choices. This further supports the claim made in Section 5.2.1.2, where it was stated that humans are prone to second-guess their choices, this highlighting the need for dataset annotator helpers - since there would be fewer annotations, less time would be wasted.

5.2.3 | Participant Inactivity

The participants seemed to present moments of doubt within their annotations. This was revealed when the optimised survey times were calculated. Subsequently, this study further investigated the nature of their deliberations. This effort revealed that the participants spent a substantial amount of time not annotating.

5.2.3.1 | Extraction

These periods of inactivity were calculated by reducing the full entry time from the optimised entry time. This revealed the amount of time that was negated via the optimisations. Quantifying these periods showed how much time manual annotations can waste. Like the previous conditions, these were calculated differently depending on the survey segment. The localisation segment's remaining time proved to be more complex to calculate. This is because the localisation segment contained two inputs rather than one.

5.2.3.2 | Inactivity Rates

These periods of inactivity were recorded to display how human nature can affect annotation performance. As seen in Table 5.3, these readings were averaged and then presented as percentages of the average full entry time. Percentages were used instead of absolute values to show the relative amount of wasted time. It emerges that the participants were not performing annotation work for at least 30% of the time throughout all entries. This rate of inactivity is significant when considering the large number of samples processed in annotation work. This points towards yet another advantage of having dataset annotator helpers; since they are only a digital procedure, they present minimal computational delays. Further details on this can be seen in Section 5.3.2.4.

Configuration	Not labelling	Not localising
E	41%	**30%**
$E - U$	40%	36%
U	**38%**	31%

Table 5.3: Interactive web survey idling times in relation to full entry times. Set E represents all individual and identifiable full entry times. Set U contains all individual and identifiable full entry times for entries marked with "none of the above". Bold denotes the smallest rate.

5.2.4 | Annotation Preferences

Due to the different inputs required in the localisation part, further information on user annotation behaviour could be extracted. Since both inputs' events were recorded, it was possible to discover what the participants preferred to fill in first. Such information could provide insight into what was most prominent for the participants.

5.2.4.1 | Purpose

Extracting what the participants preferred to perform first could potentially reveal further information. Therefore, further investigations were carried out to discover the participants' annotation preferences. To find out which input a participant clicked on first, the start dates recorded in Section 5.2.2 were reused. The earliest start date indicated the initially selected input. From there, scores were assigned to quantify the likelihood of a participant drawing bounding boxes first. The participants started their annotations in this manner for approximately 66% of the entries that the participants started their annotations in this manner. Indeed, it was expected that most of the participants would start with the bounding boxes due to the visual stimulus caused by the altering of images; the participants would have naturally shifted their focus towards the task at hand. However, the fact that just over a third of the localisation entries did not follow this trend is even more interesting. The remaining 34% of localisation entries had their label selected first. This implies that the participants may have been confused by the image. They then reminded themselves of what categories were available to finalise their choice. Another plausible scenario is that the participants were sure of their answer and immediately chose their label. In other words, both scenarios consist of the participants basing their bounding boxes on that label. This is remarkably similar to the inner workings of CAMs; as explained in Section 3.3.4.4, CAMs base their heatmaps on the results of the image classifier.

5.2.5 | Discoveries

The previous sections described the discoveries made with the web application. The extracted time intervals revealed critical information about human annotation behaviour, prompting further discussions in view of this module's respective objective. In summary, it was discovered that humans take less time labelling images than localising objects within them. This improves the prospects of CAM-empowered weakly supervised models as they only require image-level labels. Additionally, exclusively annotating images via image-level labels reduces the complexity of the task. Furthermore, the participants performed sub-optimally; indeed, they wasted 30% of their entry times. Dataset annotator helpers would be advantageous in this context; since programs simply execute, no time would be wasted except for computational delays. Additionally, more than a third of the entries were made similarly to how CAMs operate. In other words, some participants annotated the label before adding the bounding boxes. In conclusion, a dataset annotator helper might provide a quantifiable time benefit. Additionally, using only image labels reduces the overall amount of effort required.

5.3 | Human vs Machine Metrics

The second objective detailed in Section 1.5 guided this study to evaluate the human and machine annotations. This was done by using the modules discussed in Section 4.1.2. These are also the small and modular evaluation programs used within Section 5.2. The survey programs processed the survey results and averaged them, revealing the expected performance when utilising the general public as data annotators. Meanwhile, the CAM-empowered models' program generated the results based on the machine annotations, which were produced from model inferences. Weakly supervised object localisers were created from different combinations of model and CAM techniques. Additionally, auxiliary details concerning the training sessions of the AI models were recorded as well.

5.3.1 | Models

The models listed in Section 4.1.2.4 were chosen to represent the machine-generated annotations. Integrating them with the CAM techniques mentioned in Section 3.3.4.4 produced a total of 12 combinations. This experiment used these models to generate all the machine annotations. The images used for evaluation were the same ones used for the survey. Further details about the models' hyperparameter tuning process and train-

ing sessions are discussed in the following paragraphs. These details serve as extensions to the discussions presented in Section 4.1.2.4.

5.3.1.1 | Tuning

Before the training sessions in Section 4.1.2.4 were performed, a hyperparameter tuning process was executed via KerasTuner [164]. KerasTuner executed pseudo-training sessions for the AI architectures, gaining insight into how they perform for each combination of values. The data used for this process included a dedicated tuning subset. KerasTuner was operated by specifying the hyperparameter search space, these being the learning rate and the momentum in this study's use case. The specific values given to KerasTuner were typical values seen in AI training sessions [35, 88, 96]. In the following equation, set LR represents all the possible learning rate values, while set M represents all possible momentum values:

- $LR = \{1\times10^{-2}, 1\times10^{-3}, 1\times10^{-4}, 1\times10^{-5}\}$
- $M = \{0, 0.9\}$

These values were applied to all models during their respective hyperparameter tuning sessions. After the session had been completed, the optimal hyperparameter setup was saved for each model. Table 5.4 presents the final setups for the models' training sessions. The chosen momentum for all the algorithms was 0.9. Even with pre-training, the models still required a higher value to reach optimal performance. Meanwhile, MobileNetV2 required a slower learning rate than EfficientNetB0. Therefore, its hyperplane was more orderly. Lower learning rates are suitable in such scenarios as they would be less likely to encounter local minima. Another scenario could be that the model's pre-trained state was closer to the minima than EfficientNetB0. During hyperparameter tuning, VGG16 took the longest to complete. This was expected due to the optimiser used; indeed, SGD is quite simple as an optimiser and slower than RMSprop [163]. Nevertheless, all algorithms managed to reach their respective optimal settings. Without KerasTuner, these settings would have been discovered in a trial and error fashion.

Model	Optimiser	Learning Rate	Momentum
VGG16	SGD	1×10^{-3}	0.9
MobileNetV2	RMSprop	1×10^{-4}	0.9
EfficientNetB0	RMSprop	1×10^{-3}	0.9

Table 5.4: Hyperparameter tuning results with Hyperband algorithm

5.3.1.2 | Training

After obtaining the optimal hyperparameters, the AI models were trained. In addition to this, the other training options presented in Section 4.1.2.4 were applied. The architectures were all trained on the simplified ImageNet training subset. The validation subset was used as a benchmark to limit model overfitting. The training sessions consisted of a maximum of 50 epochs. In Table 5.5, one can find the reports presented by the models. Additionally, the training graphs of the models can be viewed in Figures 5.1 and 5.2. The relative increase in epochs was recorded by using Equation 5.1. In this instance, n represents the smallest amount of epochs, while e represents the largest amount of epochs. The result i denotes the final percentage increase.

$$i = \frac{e - n}{\mid n \mid} \times 100 \tag{5.1}$$

VGG16 took the longest out of the three models, exhibiting a 1050% increase in epochs compared to EfficientNetB0. VGG16 also seemed to have the most successful training session as it reported more than a 1% performance increase than the others. However, it also exhibits the largest discrepancy in accuracy between training and validation. This suggests that VGG16 was on the verge of overfitting. With thanks to the model callbacks detailed in Section 4.1.2.4, the training was stopped on time. Meanwhile, EfficientNetB0 seemed to provide the best generalisation. This is because it achieved the highest validation accuracy and the lowest validation loss. However, accuracy is generally not a good indicator on model performance, as discussed in Section 2.1.3 with Equation 2.1. To mitigate this issue, this study applied further metrics to extract more details on each model's performance. The precision of a model calculates its performance in view of the FPs. Precision is determined by Equation 2.2. Applying this function to the models' validation subset produced the results shown in Table 5.6. This table showcases all the models' validation precision, indicating their ability to handle FPs on new data. Evidently, the most precise model proved to be EfficientNetB0, garnering an over 1% in-

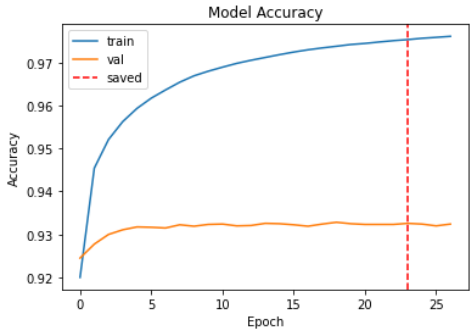
Model	Epochs	Accuracy (Tr./Vld.)	Loss (Tr./Vld.)
VGG16	23	**97.7**%/93%	**1.645**/1.689
MobileNetV2	3	96.1%/93.2%	1.659/1.689
EfficientNetB0	2	95.2%/**94.7%**	1.667/**1.671**

Table 5.5: The image classification models' training output. Accuracy is separated into their training and validation versions. The same applies to loss. Bold denotes the best performance. Note the large increase in epochs on VGG16 due to the different optimiser.

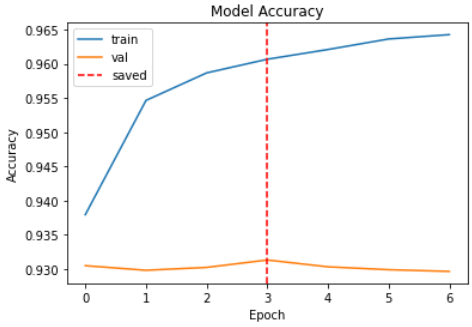
Model	Precision
VGG16	93.5%
MobileNetV2	93.3%
EfficientNetB0	**94.8%**

Table 5.6: Classification precision by model. Bold denotes the best performance.

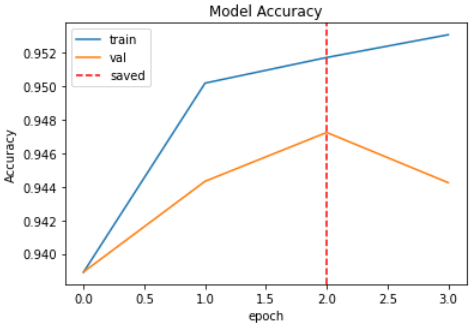
crease in precision over the other models. This implies that EfficientNetB0 was the least prone to classifying an instance of a label as another. This also suggests that EfficientNetB0 has the best PPV. By applying Equation 2.3 to the current results, model recall



(a) VGG16 training epoch accuracy visualisation. Note how VGG16 took longer due to the SGD optimiser.



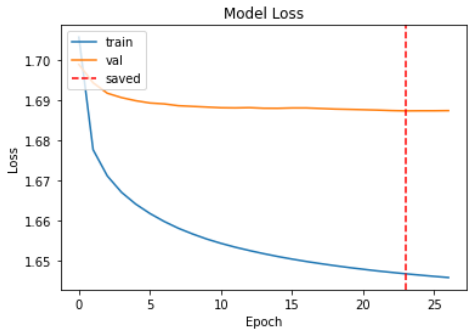
(b) MobileNetV2 training epoch accuracy visualisation. Note the decrease in time taken to end training due to the RMSprop optimiser.



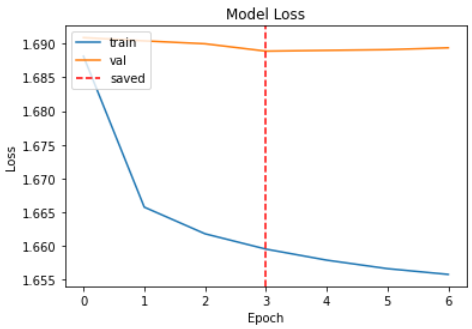
(c) EfficientNetB0 training epoch accuracy visualisation. Note the decrease in time taken to end training due to the RMSprop optimiser.

Figure 5.1: Model accuracy visualisations. The x-axis represents the epoch while the y-axis represents the accuracy. Blue denotes training accuracy while orange denotes validation accuracy.

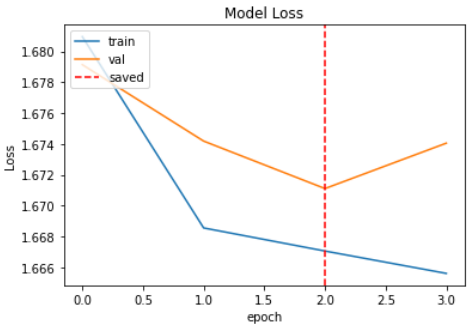
was also quantified. EfficientNetB0 seems to still be the best performing model, as seen in Table 5.7. As discussed in Section 2.1.3, precision and recall can be combined to create the F-measure, which in turn, can be configured to provide the F1-score. This can be seen in further detail in Equations 2.4 and 2.5. In this study, the dataset used was quite generic; therefore, no real priority was given to either precision or recall. Such situations



(a) VGG16 training epoch loss visualisation. Note how VGG16 took longer due to the SGD optimiser.



(b) MobileNetV2 epoch loss visualisation. Note the decrease in time taken to end training due to the RMSprop optimiser.



(c) EfficientNetB0 epoch loss visualisation. Note the decrease in time taken to end training due to the RMSprop optimiser.

Figure 5.2: Model loss visualisations. The x-axis represents the epoch while the y-axis represents the loss. Blue denotes training loss while orange denotes validation loss.

Model	Recall
VGG16	93.2%
MobileNetV2	93.1%
EfficientNetB0	**94.7%**

Table 5.7: Classification recall by model. Bold denotes the best performance.

require that $\beta = 1$. By using the precision and recall values presented in Tables 5.6 and 5.7, respectively, the F1-scores of each model could be calculated. Applying these values produced Table 5.8, which displays each model's respective F-measure where $\beta = 1$. As expected, the F1-score reflects the same rankings that were exhibited in Tables 5.6 and 5.7; EfficientNetB0 presented the best overall performance in a predictive setting, while MobileNetV2 presented the worst odds. Nevertheless, all the models performed well in terms of classification. All the metrics yielded results over 90% on a validation subset containing 12,000 images. It must be noted, however, that classification is not the only factor that determines the usefulness of such models within a study such as this. These classification results simply provide a rough estimate of the base performance and the prospects of these models.

5.3.2 | Results by Metric

To test the hypothesis in Section 1.4, human annotation performance had to be compared with the models' performance. The results presented in Section 5.3.1 only detail the models' classification performance. At that point, they had not been combined with the CAM techniques mentioned in Section 3.3.4.4. By integrating a CAM method with a model, a weakly supervised object localisation model was created. Additionally, by extracting the arithmetic mean of each entry, the general public was represented as a singular "model". These measures were taken to establish an impartial comparison while also preserving the expected performance. This performance was then quantified through various metrics, such as LE, classification accuracy, and bounding box accuracy.

5.3.2.1 | Localisation Error

Considering that this study simplified ImageNet, the ILSVRC's challenges were reviewed to analyse their evaluation methods. The ILSVRC is split into three main categories: classification, localisation, and detection [2]. Since this study's models were used as weakly supervised object localisers, they did not use the ILSVRC bounding boxes for their training session. The ILSVRC evaluates the localisation ability of models by imple-

Model	F1-score
VGG16	93.3%
MobileNetV2	93.2%
EfficientNetB0	**94.7%**

Table 5.8: Classification F1-score by model. Bold denotes the best performance.

menting a metric known as LE. However, LE contains many components which should be reviewed before presenting the final LE equation. Indeed, LE combines a classification evaluation with a localisation evaluation, these being the two main components constituting it. In contrast to the classification component, the localisation component is more complex. It consists of two main functions that convert a set of bounding boxes into an evaluation. As described in Section 2.1.3, the IoU in conjunction with a threshold can be used to determine whether a predicted bounding box can match with the ground truth. From the eligible architectures, this metric was used only by VGG16 [35]. This is most likely due to the fact that localisation was largely replaced with detection in this context. Object detection replaced object localisation because it outputs more than one class per image. However, since weakly supervised localisers are extensions to image classifiers, they could only produce one class per image. Consequently, LE is still a viable metric compared to object detection metrics like mAP. More information about this limitation can be viewed in Section 6.2.2.1. As discussed in Section 4.1.2.6, the LE metric was used and modified to improve on its weaknesses, further ensuring a comprehensive result set.

Survey By using both the original and improved variations of LE, the results for the survey participants were generated. Table 5.9 presents the LE results of the participants separated by sets E and U as referenced in Tables 5.1, 5.2, and 5.3. Evidently, the participants were quite confused during the survey. Removing the "none of the above" entries and not accounting for partially incorrect answers still indicated that the participants gave incorrect answers to a third of the localisation entries. The rounded average results represent the participants' results categorised as either a 1 or a 0. More specifically, the most common trend of each entry by image was used as a representation of a prediction. This allowed the survey results to exhibit themselves more as if the general public was a singular model. When using all the entries and not rounding the average results, the

Configuration	LE	LE_{fp}	LE_{fn}	LE_{both}
E	43.9%	53.5%	50.5%	58.9%
$E - U$	37.5%	48.2%	44.8%	54.2%
$\approx E$	34.2%	44.2%%	42.5%	49.2%
$\approx (E - U)$	**31.7%**	**42.5%**	**40%**	**48.3%**

Table 5.9: Survey localisation metrics with various configurations. Set E represents all individual and identifiable entries. Set U contains all entries marked with "none of the above". The best results are marked in bold.

participants presented 43.9% LE. When introducing the LE variations, even worse performance ensued. Additionally, the participants appeared more likely to present FPs rather than FNs, implying that they were more likely to present an entry of an object as another. When taking into account both error types, the participants presented between 48.3% and 58.9% LE_{both}. Therefore, the participants clearly suffered in localisation performance. The set U by itself was removed from the calculations as otherwise, the participants would have always presented 100% error on all metric variations. Based on the reported metrics, it is clear that the general public is quite average in terms of annotation performance. However, such metrics are not detailed enough to offer insight into where the participants struggled the most. In summary, more information was required on this performance.

Models The models were subjected to similar processes as the survey. What differentiated the processes was that the survey had multiple entries per image. Since the general public was then treated as a singular model, the averages had to be rounded to provide a consistent result. Meanwhile, since the models were on their own, they already provided a consistent result set. Consequently, the processing of the models' results was much simpler. Table 5.10 presents the error results by each model and CAM combination. VGG16 performed both the best and the worst in terms of LE, implying that whatever CAM technique is utilised would have a considerable effect on the final result. However, EfficientNetB0 seemed to provide more consistent results. The results indicate only around a 4% difference in LE within EfficientNetB0 and CAM combinations. Nevertheless, it still didn't perform well compared to the best-performing model-CAM combinations. Most of the models performed worse than the survey participants with an overall LE of 58.8%. However, each model-CAM combination was treated as an individual model as per the reasons specified in Section 4.1.2. Table 5.10 shows the equivalent of the E results. The AI models were also subjected to their own "none of the above" option. Further information about this can be found in Section 5.3.2.5. Table 5.10 also presents why the newly presented LE modifications were important. According to standard LE, the best performing model was VGG16 integrated with Grad-CAM++. When introducing the other variations of LE, however, VGG16 with Grad-CAM++ exhibited an almost 20% increase in error at its worst. In contrast, MobileNetV2 with Faster Score-CAM presented an under 4% increase in error at its worst, causing its error to become lower than that of VGG16 with Grad-CAM++. If only standard LE was utilised in this study, a subpar model would have been used to present the results. However, through the LE variations, FPs and FNs were also taken into account within the overall result. Nevertheless, more information could be extracted from the models via predictive met-

rics.

Comparisons It is clear that both the survey and the models seem to be inaccurate. Assuming that the current ground truth is correct, their performance seems to range between average to lacking. When observing all the $E - U$ entries, MobileNetV2 with Faster Score-CAM outperformed the survey entries by more than 4% LE_{both}. However, when introducing the survey entries as a singular model via the rounding operation, the participants outperformed this model by around 1% LE_{both}. In short, the LE variations provided important insight into both survey and model performance. Without these, this study would have overlooked critical information. The survey participants seemed to be more prone to providing FPs rather than FNs, implying that they were more likely to predict an object in place of another one in a bounding box. Meanwhile, the models presented differing results. Most models, like EfficientNetB0 with Score-CAM, MobileNetV2 with Score-CAM, and MobileNetV2 with Faster Score-CAM, exhibited the same type of performance. In contrast to the survey participants, however, they were prone to FNs, implying that model-CAM combinations were more prone to missing ground truths with their predictions. While this result set proved useful to determine the hypothesis presented in Section 1.4, more information was required to analyse the performance of these models and the survey participants.

Model	CAM	LE	LE_{fp}	LE_{fn}	LE_{both}
EfficientNetB0	Grad-CAM	68.3%	71.7%	71.7%	74.2%
EfficientNetB0	Grad-CAM++	64.2%	69.2%	67.5%	71.7%
EfficientNetB0	Score-CAM	65.8%	69.2%	68.3%	71.7%
EfficientNetB0	Faster Score-CAM	65.8%	66.7%	70.8%	70.8%
MobileNetV2	Grad-CAM	60%	65%	63.3%	67.5%
MobileNetV2	Grad-CAM++	55%	58.3%	58.3%	60.8%
MobileNetV2	Score-CAM	54.2%	57.5%	58.3%	60.8%
MobileNetV2	Faster Score-CAM	46.7%	**46.7%**	49.2%	**49.2%**
VGG16	Grad-CAM	74.2%	82.5%	78.3%	85%
VGG16	Grad-CAM++	**40.8%**	55.8%	**48.3%**	60%
VGG16	Score-CAM	44.2%	55%	53.3%	60.8%
VGG16	Faster Score-CAM	56.7%	62.5%	61.7%	66.7%

Table 5.10: Model-CAM results by LE variations, which take into account either FPs (LE_{fp}), FNs (LE_{fn}), or both (LE_{both}). The best results are marked in bold.

Configuration	Accuracy
E	77.9%
$E - U$	89.1%
$\approx E$	87.5%
$\approx (E - U)$	**92.5%**

Table 5.11: Survey classification accuracy results. Set E represents all individual and identifiable entries. Set U contains all entries marked with "none of the above". Bold denotes the best performance.

5.3.2.2 | Classification Accuracy

In Section 5.3.1.2, the original image classifiers were evaluated through their training session. This produced their classification accuracy. The survey entries were also subjected to the same process, producing a classification accuracy result set for the entire survey. This revealed whether the participants or the models were better for determining the image-level labels. In this case, classification accuracy was calculated by applying Equation 2.1. Since accuracy is not predictive, other metrics were also applied to aid in extracting further details about classification performance.

Survey In terms of accuracy, Table 5.11 shows participant performance when considering only image-level labels. As expected, classification accuracy improved when removing the "none of the above" entries. In fact, the survey participants seemed to perform moderately well when determining image-level labels, implying that the issue with the average LE_{both} might not lie with the labels. Nevertheless, 20% of the already filtered entries had been classified inaccurately. Table 5.12 shows predictive metrics, including precision, recall, and F1-score, which revealed where the aforementioned errors specifically occurred. To obtain these results from the survey entries, a macro-average was used. The classification metrics revealed that the participants were more prone to providing FNs rather than FPs. This is due to the lower reported recall values when compared to precision. These results imply that the participants provided the opposite predictive performance of all LE variations. This further suggests that whatever caused the participants to perform lacklustrely extends to more than simply incorrect labelling. Conversely, the rounded results improved the overall metrics. This indicates that most participants classified each entry correctly. This further improved the overall rounded result.

Models The models' overall classification performance was already presented in Table 5.5. Arguably, that table is a more accurate representation of the models' performance. However, to provide a suitable set of results, the models were re-evaluated onto the survey subset. This was done to formally compare the metrics with the survey results. Table 5.13 presents the acquired results. All the models performed quite well in determining the ground truth classes. None of them was less than 90% accurate. All the accuracy values also coincided with the full validation subset results. This was due to the fact that the values shown in Table 5.5 come within 0.5% of each other, suggesting that the highly limited survey subset still provided enough generalisation. Nevertheless, the best performing image classifier still remained VGG16. Despite this, VGG16 was not the best weakly supervised model, as seen in Table 5.10. This suggests that training an accurate classifier is not enough to develop an accurate weakly supervised object localiser. The key factor most likely lies within the model architecture and CAM technique combination. Regardless, it can be concluded that the models' lacklustre performance exhibited in Table 5.10 was not because of improper classifications. On the contrary, the models proved to be competent image classifiers. Similar to the survey classifications, the macro-averages of precision, recall, and the F1-score revealed further details, which are presented in Table 5.14. As expected, VGG16 presented the best performance out of all the models. Just as with the survey classifications, the models seemed to handle FPs better than FNs. However, both precision and recall seemed to

Configuration	Precision	Recall	F1-score
E	82.3%	71.9%	76.1%
$E - U$	89.2%	88.4%	88.6%
$\approx E$	91%	81.5%	85.2%
$\approx (E - U)$	**92.8%**	**92.5%**	**92.4%**

Table 5.12: Various macro-averaged survey classification results. Set E represents all individual and identifiable entries. Set U contains all entries marked with "none of the above". Bold denotes best performance.

Model	Accuracy
EfficientNetB0	93.3%
MobileNetV2	93.3%
VGG16	**95%**

Table 5.13: Classification accuracy by model. Bold denotes best performance.

Model	Precision	Recall	F1-score
EfficientNetB0	86.3%	86.2%	86.1%
MobileNetV2	86.3%	86.2%	86.1%
VGG16	**88%**	**87.7%**	**87.7%**

Table 5.14: Macro-averaged classification metrics by model. Bold denotes the best performance.

be quite similar. The F1-score indicates that the predictive performance of these models was worse than their reported accuracy. Such a scenario shows why predictive metrics are important. Despite this, all the models performed moderately well, indicating that they are potent classifiers and that they were trained correctly. However, further investigation was required to discover the models' lacking performance with LE.

Comparisons Both the participants and the models performed moderately well to excellent, suggesting that both humans and machines are competent image classifiers. Moreover, both the participants and the models exhibited comparable performance. When considering the $E - U$ entries, the models barely reached the same performance as the participants. The largest distinction in terms of F1-score is that of 10%, when the E entries are compared with VGG16. Therefore, the models may have a slight advantage, depending on the survey configuration applied. Within the scope of this study, however, such a comparison might be paradoxical. Firstly, the AI models were trained on image-level labels. Therefore, some might consider such a comparison to be partial, giving rise to the notion that these results are completely subjective. This is due to the ground truth correctness assumption. Secondly, this study's aim was to help users create object localisation datasets through image-level labels. Therefore, comparing these performances might seem illogical as users still have to provide image-level labels. The only fact that can be gleaned from these results is that care must be taken when performing crowdsourcing. As indicated by the original entry results shown in Table 5.11, the general public can be quite inattentive and thus may potentially provide lacking annotations.

5.3.2.3 | Bounding Box Accuracy

Since both the humans and models presented accurate labelling, further investigations were conducted on the localisations instead. The LE metrics already accounted for the localisations, but only in conjunction with the class labels. To truly test the accuracy of the localisations, IoU was utilised by itself. Just as with the LE metrics, the localisation

was considered accurate if the IoU value was 50% or over. Notwithstanding, bounding box accuracy was determined as if each class was simply a positive instance. In summary, Equation 2.1 was still utilised to determine the accuracy of the bounding boxes; however, it was not applied in a multiclass scenario.

Survey By calculating bounding box accuracy, an accurate representation of the localisation performance from the survey was obtained. It is important to note that bounding box accuracy was calculated via both entries and individual instances. This means that each metric was produced by either processing a whole image or each bounding box in that image. Such a variation was introduced to delve deeper into bounding box performance. These results were not applied in the case of LE as that specific metric revolved around each image entry only. This is also true for the individual classifications. Table 5.15 presents the results separated by the types of entries as in the previous survey-related tables. Evidently, the participants performed quite sub-optimally when it came to localisation. When observing the $E - U$ entries, the participants drew under 50% bounding boxes accurately. This implies that the root of the average LE metrics is due to bounding box performance. The results also indicate that humans find it more difficult to perform localisations than classifications. This is because while the participants performed moderately well in categorising the contents of the image under a specific label, their performance suffered when asked to indicate where the basis of their results lay. The U entries could also be evaluated exclusively in this scenario. This is due to the label not being required when evaluating Equation 2.1. The entries that included such an option came with rather poorly drawn bounding boxes. Under 20% of such entries were correct. This was to be expected, but it does provide important insight - it can be inferred that bounding box accuracy is dependent on classification accuracy.

Configuration	Accuracy_{entry}	Accuracy_{box}
E	45.9%	26.1%
$E - U$	48.8%	28.4%
U	19.9%	9.5%
$\approx E$	**54.2%**	29.9%
$\approx (E - U)$	**54.2%**	**30%**
$\approx U$	32.8%	15%

Table 5.15: Survey bounding box accuracy presented by either each entry or per bounding box. Set E represents all individual and identifiable entries. Set U contains all entries marked with "none of the above". Bold denotes the best performance.

Configuration	Precision$_{entry}$	Recall$_{entry}$	F1-score$_{entry}$
E	46.1%	76%	55.1%
$E - U$	48.3%	83.5%	60.1%
U	19.9%	100%	33.2%
$\approx E$	53%	85.6%	61.3%
$\approx (E - U)$	**53.4%**	**86.2%**	**63.6%**
$\approx U$	32.8%	100%	49.4%

Table 5.16: Macro-averaged survey bounding box metrics by entry. Set E represents all individual and identifiable entries. Set U contains all entries marked with "none of the above". Bold denotes the best performance. U recall performance was ignored as there were no "none of the above" ground truth labels.

When the participants failed to determine what the appropriate label should be, they were not able to properly identify its location either. This information also corroborates what was implied in Section 5.2.4; since 34% of the localisation entries were determined via the label first, it can be concluded that the label is a deciding factor in localisations. Additionally, bounding box performance suffered when viewing the metric through individual instances. Indeed, there is a 20% difference between the non-rounded entries' accuracy variations. The only instance where the participants performed averagely was within the $\approx (E - U)$ entries. Despite the already negative results, further information was extracted by applying predictive metrics. Precision, recall, and the F1-score were applied to determine the causes of localisation failures. Tables 5.16 and 5.17 present the obtained results split between entry and instance variations. As expected, the instance version of the metrics reported worse performance than the entry version, indicating

Configuration	Precision$_{box}$	Recall$_{box}$	F1-score$_{box}$
E	41.8%	42.7%	41.3%
$E - U$	45.6%	47.3%	45.3%
U	18.8%	16%	17.3%
$\approx E$	46%	48.7%	46.2%
$\approx (E - U)$	**46.8%**	**49.3%**	**46.6%**
$\approx U$	26.4%	25.8%	26.1%

Table 5.17: Macro-averaged survey bounding box metrics by instance. Set E represents all individual and identifiable full entry times. Set U contains all individual and identifiable full entry times for entries marked with "none of the above". Bold denotes the best performance.

Model	CAM	Accuracy_{entry}	Accuracy_{box}
EfficientNetB0	Grad-CAM	25.8%	12.2%
EfficientNetB0	Grad-CAM++	30%	14.7%
EfficientNetB0	Score-CAM	30%	13.8%
EfficientNetB0	Faster Score-CAM	29.2%	14.8%
MobileNetV2	Grad-CAM	32.5%	16.4%
MobileNetV2	Grad-CAM++	40.8%	20.4%
MobileNetV2	Score-CAM	40.8%	21.3%
MobileNetV2	Faster Score-CAM	**51.7%**	**26.4%**
VGG16	Grad-CAM	15%	8%
VGG16	Grad-CAM++	40%	24.4%
VGG16	Score-CAM	39.2%	23.5%
VGG16	Faster Score-CAM	33%	18.7%

Table 5.18: Model bounding box accuracy presented by either each entry or per bounding box. Bold denotes the best performance.

that there were more incorrect specific instances in all the entries. None of the bounding box metric variations reported over 50% precision, recall or F1-score. This further supports the argument that the participants struggled to localise correctly. Further investigations revealed that the participants struggled with FPs more than with FNs. This is due to all the entry recall metrics being over 80%. Nevertheless, the F1-score reveals that the participants' predictive performance was quite lacking. In conclusion, the participants suffered in localisation, this being the main cause of the average LE metrics presented previously.

Models Using similar methods, the bounding box accuracy for all the models was also obtained. Table 5.18 showcases the results for each model-CAM combination. As seen in Table 5.18, these models performed quite ineffectively. The best performing model outputted around 52% bounding box accuracy per entry. When viewing the instance variation of the metric, performance dropped by just under half of the previous value. Other instance results that achieved similar performance include some model-CAM combinations incorporating VGG16. This corroborates previous VGG16 performance, where it performed slightly worse than MobileNetV2 with Faster Score-CAM. In addition to accuracy, the standard predictive metrics were applied to further analyse bounding box performance. Tables 5.19 and 5.20, which were produced via precision, recall, and the F1-score, reveal such details. When viewing the per-entry results, the models presented

Model	CAM	Precision$_{entry}$	Recall$_{entry}$	F1-score$_{entry}$
EfficientNetB0	Grad-CAM	23.7%	76.7%	35%
EfficientNetB0	Grad-CAM++	27.7%	81.8%	39.8%
EfficientNetB0	Score-CAM	27.9%	82.7%	39.1%
EfficientNetB0	Faster Score-CAM	27.3%	79.2%	38.8%
MobileNetV2	Grad-CAM	29.9%	77.6%	41.6%
MobileNetV2	Grad-CAM++	37.6%	83.1%	50.7%
MobileNetV2	Score-CAM	37.6%	81.7%	50.4%
MobileNetV2	Faster Score-CAM	**47.6%**	**83.2%**	**59.2%**
VGG16	Grad-CAM	13.9%	60%	21.6%
VGG16	Grad-CAM++	36%	79.4%	48.5%
VGG16	Score-CAM	35.3%	80.5%	47.5%
VGG16	Faster Score-CAM	30.8%	78.4%	42.4%

Table 5.19: Macro-averaged model bounding box metrics by entry. Bold denotes the best performance.

quite optimistic results. Particularly, the recall results ranged from average to good. Since the F1-score is a harmonic mean of both precision and recall, the metric also produced better than expected results. The high recall value implies that when viewing the

Model	CAM	Precision$_{box}$	Recall$_{box}$	F1-score$_{box}$
EfficientNetB0	Grad-CAM	21.5%	20.1%	20.4%
EfficientNetB0	Grad-CAM++	26.7%	24.3%	25%
EfficientNetB0	Score-CAM	24.9%	23.8%	23.8%
EfficientNetB0	Faster Score-CAM	28.8%	19%	22.2%
MobileNetV2	Grad-CAM	30.4%	25.5%	27.2%
MobileNetV2	Grad-CAM++	39.3%	30%	33.2%
MobileNetV2	Score-CAM	41.2%	30.6%	34.3%
MobileNetV2	Faster Score-CAM	**49.5%**	**35.9%**	**40.5%**
VGG16	Grad-CAM	12.4%	16%	13.6%
VGG16	Grad-CAM++	40%	37.6%	38%
VGG16	Score-CAM	39.6%	33.8%	35.5%
VGG16	Faster Score-CAM	34.2%	27.4%	29.2%

Table 5.20: Macro-averaged model bounding box metrics by instance. Bold denotes the best performance.

entries, there was a minimal amount of misses per class. Conversely, the instance variation of the metrics presented a different conclusion. Recall suffered immensely when compared to the entry metrics. In fact, precision outperformed recall in almost every model-CAM combination. The models were more prone to missing a bounding box rather than producing too many of them. Such results conclusively determine that the models' average LE is also because of subpar bounding box performance.

Comparisons Between the survey and model entries, the most important results were quite comparable. However, the participants seemed to present slightly better general bounding box performance. When comparing the survey F1-score$_{box}$ with the relevant MobileNetV2 with Faster Score-CAM result, the participants outperformed the model-CAM combination by 6%. However, this result was generated from the $\approx (E - U)$ entry set, which were artificially modified by rounding the entries into one per image. Despite this, there are instances where MobileNetV2 with Faster Score-CAM outperformed the participants. When observing the F1-score$_{entry}$ results presented by the same model-CAM combination, it surmounted the survey participants by 4%. Meanwhile, the models seemed to suffer in recall when evaluating individual instances, showing that the models were more effective at handling FPs rather than FNs. The opposite was true in the case of the survey entries; it was more likely for the participants to provide too many bounding boxes rather than too few.

5.3.2.4 | Times

Section 5.2 presented and reviewed various types of entry times. Originally, this was done to obtain more information on human annotation behaviour as per the first objective; however, the entry times' usage can also be extended to the current objective being discussed, particularly, they can be used to compare whether humans or machines are faster in annotation. Such results also indicate whether it is worthwhile to consider CAM-empowered models as dataset annotator helpers. To estimate these values, pseudo-session periods were calculated based on the dataset to help envision the expected performance from such models. As described in Section 4.1.1.1, the simplified ImageNet training set contains exactly 128,763 images. This amount, now represented as n, was applied in Equations 5.2 and 5.3 to discover such times.

Survey As indicated in Section 5.2, the time periods of interest for these calculations were the full entry times and the optimised entry times. These can be found in Tables 5.1 and 5.2, which represent the typical and improved performance of the general public,

Configuration	Full	Optimised
E	$\approx$ 823h	$\approx$ 608h
$E - U$	$\approx$ **787h**	$\approx$ **501h**

Table 5.21: Time to annotate $n = 128,763$ images based on different survey time types. Set E represents all individual and identifiable full entry times. Set U contains all individual and identifiable full entry times for entries marked with "none of the above". Bold denotes the shortest times.

respectively. Thus, these are the most viable values when computing the time required for humans to annotate images, which can be calculated by evaluating 5.2. Apart from n, the equation also includes variable b, which represents whatever average localising entry time was utilised to generate the result, as well as t, which represents the total time required to annotate the dataset, based on the crowdsourcing intervals.

$$t = n \times b \tag{5.2}$$

Applying these entry times provided the total amount of time required for the general public to annotate the simplified ImageNet training set. Table 5.21 shows these results in terms of hours. The fact that the smallest amount of time possible was approximately 21 full days' worth of work confirms that manual annotation is an extremely laborious task. This is especially true when considering the original ImageNet dataset, which contains over 1 million training images [2]. Moreover, the importance of even a few seconds shows itself in Table 5.21. In Tables 5.1 and 5.2, the difference between the relevant times only amounts to between 1s and 3s. Similarly, the time taken was reduced substantially when using the optimised entries. By applying Equation 5.1, a percentage decrease was calculated to represent this time advantage. It resulted that there was a 39% decrease in the time taken when considering e as the full time including unknown entries and n as the optimised time excluding unknown entries. Summarily, the results objectively show that manual image annotation is a monumental effort while also demonstrating how an effective dataset annotator helper could be of great help in this endeavour.

Models In Equation 5.3, m represents the training time of the model. This tallied up to roughly 48h, including the hyperparameter tuning session. Variable l stands for any relevant labelling time, while variable w represents the inference time of the model-CAM combination used as the dataset annotator helper. It must be clarified that the training time was only applicable for VGG16, the model that took the longest to fit.

Model	CAM	Full	Optimised
EfficientNetB0	Grad-CAM	$\approx$ 671h	$\approx$ 361h
EfficientNetB0	Grad-CAM++	$\approx$ 674h	$\approx$ 364h
EfficientNetB0	Score-CAM	$\approx$ 806h	$\approx$ 496h
EfficientNetB0	Faster Score-CAM	$\approx$ 684h	$\approx$ 374h
MobileNetV2	Grad-CAM	$\approx$ 669h	$\approx$ 359h
MobileNetV2	Grad-CAM++	$\approx$ 670h	$\approx$ 360h
MobileNetV2	Score-CAM	$\approx$ 794h	$\approx$ 483h
MobileNetV2	Faster Score-CAM	$\approx$ 683h	$\approx$ 373h
VGG16	Grad-CAM	$\approx$ **668h**	$\approx$ **358h**
VGG16	Grad-CAM++	$\approx$ 671h	$\approx$ 361h
VGG16	Score-CAM	$\approx$ 753h	$\approx$ 443h
VGG16	Faster Score-CAM	$\approx$ 676h	$\approx$ 366h

Table 5.22: Time to annotate $n = 128,763$ images based on different model times. Bold denotes the shortest times.

Despite this, this specific value was applied to all models.

$$t = (n \times l) + m + (n \times w) \tag{5.3}$$

Iterating this equation over each model-CAM combination produced Table 5.22, where the largest and smallest possible times are shown. Similar to the survey times, a semi-automated annotation workflow would have still taken a significant amount of time. While no model took significantly longer than the rest, certain CAM methodologies did. For example, Score-CAM always caused whatever model it was integrated with to take substantially longer. By applying Equation 5.1, it was discovered that Score-CAM would have taken 16.4% longer than the other CAM techniques. As a result, none of the models that used Score-CAM performed well. Therefore, such an increase might not have an impact when determining the best result. Moreover, the difference between the full and optimised times is quite substantial. Both instances were presented to envision the range of possible times that a semi-automated annotation workflow could make. Such ranges turned out to be quite large and should be taken into account. This further suggests that a "snowballing" effect can occur depending on the performance of the annotator. Therefore, it can be conclusively determined that a semi-automated annotation workflow would still take a substantial amount of time to perform.

Comparisons From Tables 5.21 and 5.22, it can be surmised that both manual and partially automated annotation would have taken a substantial amount of time. Such results suggest the utter significance of image annotation in CV tasks. Nevertheless, the models would have taken less time in almost every combination. Comparing the longest optimised model time with the shortest manual annotation time reveals a small difference of about 5 hours' worth of work. While not a substantial amount, the models objectively would have taken less time to perform annotations, even in the worst-case scenario. When considering the full entry times, the best manual annotation time outperformed the MobileNetV2 with Score-CAM's annotation time by 7h. This was the only model-CAM combination that fell short of the manual annotation times. This suggests that there is a quantifiable benefit in time and effort when using CAM-empowered object localisers as dataset annotator helpers. In fact, MobileNetV2 with Faster Score-CAM can save between 17% and 36% of the time taken to perform manual annotation.

5.3.2.5 | Uncertainty

The survey participants were given the "none of the above" option as a choice. This option was available in both the classification and localisation segments. This study decided that modelling uncertainty is a suitable technique for evaluating participant performance. Thus, this concept of uncertainty was also extended to the AI models. Doing so helped provide an impartial comparison of confidence for all types of entries.

Survey To avoid forcing the participants to choose a label, the web application offered a "none of the above" option. This represented an individual participant's confusion about the categories. While not a standard concept for crowdsourcing, it did help determine the participants' confidence in their answers. This is especially true when the participants had to localise an image. By dividing the number of unknown entries by the total amount of entries, the rate of "none of the above" entries was calculated. Such entries were also separated by their entry type, giving rise to Table 5.23. The rate of unknown choices was higher in image labelling than object localisation. This suggests that the participants struggled more when providing image-level labels. When asked to perform a localisation however, they searched the image contents thoroughly. This effort

Configuration	Labelling	Localisation
$\lvert U \rvert / \lvert E \rvert$	14.3%	10.2%

Table 5.23: Relative "none of the above" choice rates by annotation type in the survey

increased their chances to find a suitable object for the category in question. Additionally, it was made evident that the participants made their annotations apprehensively in at least one out of 10 entries. This rate implies that human participants should be taken precariously when using crowdsourcing.

Models To evaluate a model's uncertainty factor, a model feature was utilised. Since all models used label encoding, a confidence score could be extracted from any predicted class. This was used to test whether model confidence reached a specific threshold. The confidence threshold c was represented as $c = 0.5$. This is identical to the threshold utilised for LE in ImageNet [2]. Any confusion or uncertainty exhibited by the model would have been detected by their failure to reach that threshold. However, even with this threshold set in place, none of the AI models returned a "none of the above" prediction. This is logical since the AI models achieved high classification scores in all metrics. Assuming that the models were trained correctly, CAM-empowered object localisers appeared to be utterly confident when applied to this workflow.

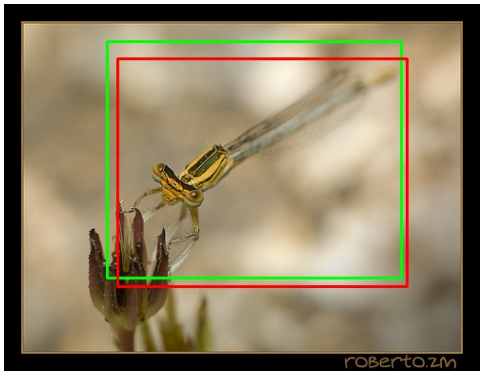
Comparisons When assuming that $c = 0.5$, the models outperformed the survey participants in terms of confidence. In this context, the weakly supervised models completely outmatched the survey participants. However, this is only applicable if the models are trained properly. If the performance of the models was subpar, their confidence scores would have also suffered as well. Therefore, these results reiterate the importance of good CNN training practices. These were achieved through the application of pre-trained models and extensive hyperparameter tuning.

5.3.2.6 | Discoveries

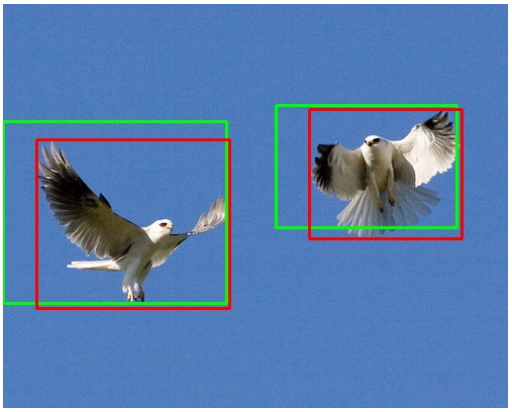
Through the various assessments that were utilised in this module, in-depth information about participant and model performance was revealed. When reviewing the various LE metrics applied, it was discovered that both the models' and the participants' performance was from inadequate to average. Despite this, the results suggest that the participants and models are comparable. Labels and bounding boxes were analysed individually to report on the possible cause for such performances. In terms of classification, both the participants and the models performed moderately well, implying that the issue lied with the bounding boxes. Bounding box performance was, in fact, quite low on both a per-entry and per-annotation basis, thus proving that the problem mostly stemmed from the localisations. The results also highlight how these models performed quite similarly to their human counterparts. In fact, the best performing model came

within 1% LE_{both} of the best survey entries. Further investigations were carried out to envision a semi-automated workflow via a weakly supervised model. It was discovered that models save about 17% - 36% of the time required to annotate manually. Finally, the models' uncertainty was modelled based on a similar system found within the survey, which helped reveal that apart from saving time, the models also presented their results more confidently than the participants. Indeed, it was found that crowdsourcing participants were unsure of their choice for one in every 10 entries. In conclusion, the models provided comparable performance to humans while still saving time. Since they do not suffer from doubt and perform accurately, they can act as competent dataset annotator helpers.

(a) Survey participant localised "instrument" when the truth label was "instrument", referring to the banjo



(b) Survey participant localised "insect" when the truth label was "insect", referring to the dragonfly



(c) EfficientNetB0 with Grad-CAM++ localised "bird" when the truth label was "bird", referring to both birds

(d) MobileNetV2 with Faster Score-CAM localised "vehicle" when the truth label is "vehicle", referring to the bumper car

Figure 5.3: Various correct annotation samples from both the survey participants and the models. The ground truth is denoted in red, while green denotes the prediction.

5.4 | Annotation Choices Assessment

As discussed in Section 5.3.2.2, the metrics utilised in 5.3.2 were subjective. Since no true ground truth could ever be determined, this issue could not be solved in its entirety. Nevertheless, this study attempted to further support the results by utilising a secondary survey. This web form was designed to evaluate human reliability. This evaluation consisted of testing more participants to observe whether they could distinguish human annotations from machine annotations. After accumulating the responses of 120 participants, the results were reviewed.

5.4.1 | Survey Demographic

Extracting the typical participant was important for the premise of this survey. Indeed, trends can be revealed based on the type of participant. Therefore, introductory questions concerning the individual participant's nature were made.

5.4.1.1 | Participant Information

Participants from a broad range of ages responded to the form. While about 30% of the participants were between 26 and 35 years old, similar distributions of about 20% were found for the following age brackets: 18 - 25, 36 - 45, and 46 - 55. The least popular age range was that of between 56 and 65, which only made up about 3% of the entire age demographic. Additionally, about two-thirds of the respondents identified themselves as male. Most of the rest identified themselves as women, while a marginal amount did not disclose this information. Such information could be utilised to categorise the results and could determine trends based on a target demographic.

5.4.1.2 | Annotation Confidence

This section also established the participants' confidence in their image annotation abilities as well as their self-assurance on their observation skills. Figure C.5 presents the exact distribution of annotation confidence ratings. Almost 50% of the participants had total confidence in their image annotation abilities. By extracting the arithmetic mean of this question as a normalised value, it was revealed that a typical participant was 84% confident in their abilities. This finding is important as it can be used to reveal any change in confidence within the participants once the survey was completed. Consequently, when evaluating the image choices presents, one would be able to deduce whether or not their confidence was unfounded.

5.4.2 | Image Choices

The second section of the form presented the participants with a set of image choices, which were determined as described Section 4.1.3.2. These choices tested the participants' ability to recognise human annotations. If the participants were unable to choose the correct answers, it would suggest that annotation accuracy stems further than simply evaluating metrics based on ground truths.

5.4.2.1 | User Performance

By averaging each individual participant's entry, the participants were only able to recognise a human annotation 36.8% of the time. This suggests that participants were more likely to choose a machine annotation as a human annotation. This argument is further strengthened by the fact that the participants were asked to choose two images. This is important to consider, as it suggests that the machine annotations are more human-like than the actual human annotations. To further investigate the results, this experiment also evaluated whether the participants answered more than 50% of the questions correctly. Overall, it transpired that only 12.5% passed the multiple-choice test, further showing that humans are unreliable in their image annotation skills, as they are not able to recognise another human's handiwork. This further alludes to the importance of a digital aid within such work.

5.4.2.2 | Trends

Analysing the individual questions in a qualitative manner revealed further trends about the participants, which, in turn, could be used to obtain more information on how the participants came to their answers.

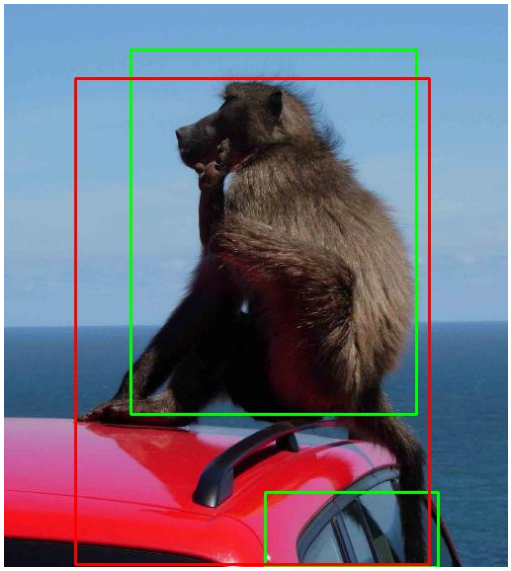
Perceived correctness Most of the participants chose the annotations with the most correct bounding boxes. Such bounding boxes were also considered correct by the ground truth. This is an important finding as it can be inferred that for most participants, a human annotation is equal to an infallible annotation. Instances of such scenarios can be observed in Figure 5.3.

Alternative answers While most of the participants seemed to recognise which images were considered correct, there were specific instances where the participants deliberately chose the incorrect answer. For example, Figure 5.4 shows incorrect or missing annotations. These instances were chosen frequently by the participants as human annotations. While in this case, they were correct, it was not immediately obvious why

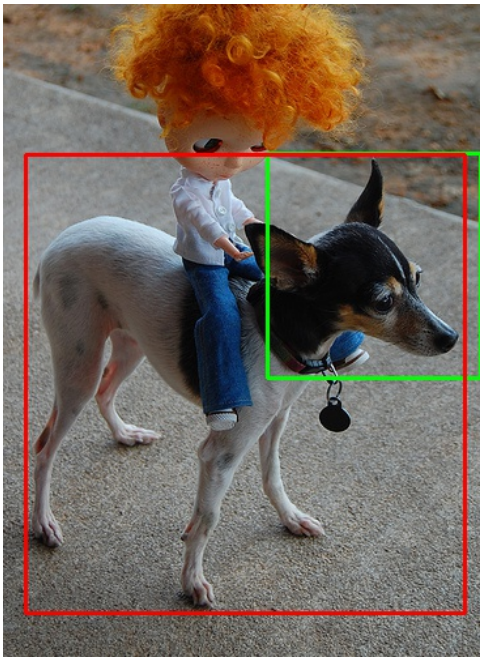
they did this. The most likely scenario is that the participants viewed the uniqueness of such instances as the alternative option. For example, the mislabelled "clothing" instance was technically correct; the ground truth of the image was labelled as "furniture",

(a) Survey participant localised "clothing" when the truth label is "furniture", referring to the entire bed

(b) Survey participant localised "none of the above" when the truth label was "reptile"



(c) VGG16 with Grad-CAM++ localised "primate" when the bounding box truth outlined the entire monkey in only one bounding box



(d) VGG16 with Grad-CAM localised "dog" when the bounding box truth outlined the entire dog rather than only its face

Figure 5.4: Various incorrect annotation samples from both the survey participants and the models. The ground truth is denoted in red, while green denotes the prediction.

but the original entry classified it as "clothing" due to the bounding box surrounding the sock. This could also be considered as proof of the subjectivity issue presented by the original ground truths.

By elimination Further alternative behaviour was observed in "obviously" mishandled annotations, as showcased in Figure 5.4. The disorganised nature of such images led the participants to believe that these were not man-made annotations. This further strengthens the theory that most of the form's participants evaluated an annotation as man-made if it seemed correct. If they saw something illogical, then it was less likely to be chosen as a machine annotation.

5.4.3 | Participant Sentiment

The final section of the form asked the participants some sentiment-based questions. These questions mainly provided insight into the form and its basis. These answers could reveal further information about the participants' behaviour in the previous section. Inexperienced participants also rudimentarily observed what annotation work consists of. This provided context for the final question, which led to clearer sentiments.

5.4.3.1 | Perceived Difficulty

Based on the results, the participants perceived the previous section's questions as moderately difficult. Determining the arithmetic mean of their ranges indicated that the participants found the second section to be 65% difficult. This is quite low when considering their performance in the previous section. Moreover, if one were to consider this difficulty rating as another representation of the participants' confidence, the participants' overall confidence reduced by the end of the form. This puts into perspective a change of opinion, as they were not as confident as they originally were by the end of the form.

5.4.3.2 | Opinions on Annotation Facilitators

The final question was reserved to the end to ensure that the participants were more qualified to answer it. Since they were part of the general public, it was likely that they were not professional data annotators. Aggregating the results revealed that 85% of the participants expressed a positive sentiment towards dataset annotator helpers. Furthermore, 38.2% of that substratum of participants were even willing to trust such a system

entirely, with no human intervention. Such results suggest that humans either acknowledge their own faults or they appreciate a digital aid's convenience. The larger segment of this substratum reveals that the participants would be willing to use up the the time saved from manual annotation to further improve model annotations. This would align with the intended use of a dataset annotator helper. In contrast, 15% of the participants indicated that they would not like to use such a system in their work. However, two-thirds of this substratum revealed that they were not willing to use them due to not seeing the tradeoff as worthwhile. Therefore, if better models are utilised, their opinion would most likely change. The specific distribution of annotation helper opinions can be seen in Figure C.19. In conclusion, this further implies that such digital systems are not only objectively recommended via the metrics but also subjectively recommended by other humans.

5.5 | Summary

This study designed all the experiments based on its objectives. The implementation consisted of small, modular programs that converted data to information. The first experiment outputted data related to the general public's performance in image annotation. Comparing the intervals revealed that the participants were relatively laid back, especially when considering the high inactivity rates calculated by negating the full entry times from the optimised entry times. Indeed, the participants were not performing annotations for at least 30% of the full entry time. Additionally, the participants started 34% of the entries in the localisation segment by selecting the label first. This indicates that a significant amount of participants operated similarly to CAMs, further supporting the argument that the label is a defining factor in localisation. The second experiment consisted of training the CAM-empowered object localisers. This was done by using KerasTuner to optimise the hyperparameters. The participants and models were compared by using a metric known as LE. However, LE presented many oversights. Further work was carried out by developing variations of LE. These novel metrics showed a substantial difference in performance between them and the original LE metric. The best performing model, MobileNetV2 with Faster Score-CAM, outperformed the survey participants' standard entries by 9.7% LE_{both}. After artificially improving the survey entries via an optimisation procedure, the survey participants outperformed the same model by only 0.9% LE_{both}. The same model presented a 25.5% reduction in the time taken when compared to manual image annotation. Since LE is similar to accuracy, predictive metrics were applied to the solution. These were used to determine the

cause of the subpar performance exhibited by both the humans and the models. It was discovered that both humans and models lack a good understanding of how to create bounding boxes. When evaluating each bounding box instance as a positive instance, they only achieved a 46.6% and 40.9% F1-score, respectively. This shows that both humans and models suffer in localisation. Finally, the web form revealed that humans are truly inconsistent in their annotations. Despite the fact that the typical participant was 84% confident in their image annotation abilities, 61.8% of the questions were answered incorrectly, indicating that the participants were more likely to choose a machine annotation than a human annotation. It also proves that the models outputted human-like annotations. Additionally, 85% of the participants responded positively to the idea of using a dataset annotator helper. This result further legitimises the reasoning behind this study.

6

Conclusion

This study's purpose was to investigate weakly supervised techniques as dataset annotator helpers and thus trained CAM-empowered object localisers with this aim in mind. To evaluate the models, they were compared with primary data obtained from the web application via common metrics. Since the ground truth was also human-annotated, an additional web form was published to verify the acquired results. Its results ascertained that humans are inconsistent with their annotations, further suggesting the potential of this methodology within image annotation.

6.1 | Discussion

This study designed a pipeline to segment the implementation. Consequently, development was split into three modules, each of which generated experiments that produced a result set. In turn, each result set corresponded to a specific objective. Then, these results were analysed to extract relevant information regarding the objective in question. This information helped determine whether this study achieved its goals.

6.1.1 | Crowdsourcing Intervals

By processing the timestamps recorded via the web application, human annotation behaviour was quantified. In Section 5.2, it was determined that the general public is inefficient in data annotation workflows due to their high inactivity rates when annotating images. This result clarified the need for a digital aid, especially in crowdsourcing systems, leading to the argument that a weakly supervised model workflow could be of help. This is because it was discovered that image labelling takes less time than object localisation, which increased the prospects for CAM-empowered object localisers as

they only require image-level labels. Since this would also reduce manual intervention, fewer annotations would be susceptible to human error. Therefore, it can be concluded that this study successfully analysed human annotation behaviour in detail.

6.1.2 | Human vs. Machine Metrics

Training the eligible architectures and combining them with various CAM techniques brought on a host of model-CAM combinations. Through LE and other predictive metrics, this study obtained a detailed result set. In Section 5.3, it was concluded that the humans and the models produced a comparable set of results; however, some model-CAM combinations performed better than specific survey configurations. MobileNetV2 with Faster Score-CAM achieved 9.7% less LE_{both} than the unprocessed survey entries. When removing the unknown entries and rounding the remaining entries, humans outperformed the same model-CAM combination by only 0.9%. Furthermore, the models seemed to infer their predictions much faster than human annotators. Altogether, this study confirmed that machines are better at annotating than humans. Therefore, applying CAMs as dataset annotator helpers can further aid humans in their work.

6.1.3 | Annotation Choices Assessment

The fact that ImageNet's ground truths were also created by humans gave rise to a paradox. Human annotations and machine annotations could not be evaluated on man-made answers. To resolve this bias, this study published a web form to obtain more primary data. The participants had to distinguish the human annotations from the machine annotations. These results were analysed in detail in Section 5.4, which primarily discussed their accuracy and extracted any trends. This experiment revealed that the participants were inconsistent as they were more likely to choose a machine annotation as a human annotation, thus highlighting the general public's unreliability in image annotation. Additionally, 85% of the participants responded positively to using a dataset annotator helper, further supporting the argument that a digital aid would be helpful in this scenario.

6.1.4 | Verdict

Overall, it can be concluded that by satisfying the aforementioned objectives, this study has successfully reached its aim. Both humans and models presented comparable annotation performance as they were both subpar. Despite this, some model-CAM combinations did perform better than the human participants. Most weakly supervised models

also saved time when compared to manual human annotation. With all this known, this study confirmed the hypothesis presented in Section 1.4, implying that AI models integrated with CAMs could be viably used as dataset annotator helpers.

6.2 | Future Work

Apart from proving that CAMs are competent dataset annotator helpers, this study also outlines possible future work. This study categorises such work as either improvements or expansions.

6.2.1 | Improvements

This section presents possible improvements that can be made when conducting further research within this methodology and its related research topics.

6.2.1.1 | Web Application

Since the web application was tailor-made, future work can include improving its underlying logic. For example, determining the participants' confidence in their submission could be more intuitive than a simple "none of the above" option; for instance, the participants could quantify their confidence in the form of a range. Whatever number is chosen from the range would represent the participants' confidence in their annotation. This concept is similar to how the participants rated their own image annotation abilities in the web form. Using scales would streamline the comparisons of human and machine annotations by quantifying humans' confidence more similarly to how models present their confidence. Additionally, the web application's load balancing component could be extended. Rather than load balancing the images only, future work could also extend this functionality to the application's parts. This would help mitigate bias when conducting further experiments on different annotation types.

6.2.1.2 | Models

In addition to including more models and CAMs, future work in this area could also entail custom solutions. This study offered a vast range of result sets that future work can extend via tailor-made model-CAM techniques. Other custom image datasets could also be tested. Additionally, certain model-CAM combinations might offer different performances in other applications. Furthermore, future work can apply more in-depth

hyperparameter optimisation. This can be done through KerasTuner in various ways, for example by:

- expanding the possible learning rates and momentum values;
- including other hyperparameters, such as batch size and layer filter sizes;
- including pre-trained and randomly initialised model variations; and
- using other oracle algorithms, such as Bayesian optimisation and random search.

6.2.2 | Expansions

The following expansions feature possible extensions to the established pipeline. These can range from possible prototypes to technical considerations.

6.2.2.1 | Localisation vs Detection

Despite their performance, model-CAM combinations still present an inconvenience. Since the models are deep image classifiers, they can only produce one label per class. This led to the usage of the LE metric to quantify their localisation potential. However, object localisation is quite dated as it has largely been replaced by object detection due to the fact that object detection uses fully supervised models that can localise multiple labels per image. This limits the scope of the current methodology considerably. However, future work can resolve this issue. As Selvaraju et al. [156] detailed, CAMs can be used for image captioning. Image captioning can convert image classifiers from single-label to multilabel predictors. With the multilabel output, CAMs can be used to produce a heatmap based on multiple classes. This act converts weakly supervised object localisers to weakly supervised object detectors. Then, more popular metrics such as mAP could be applied to further extend the result set.

6.2.2.2 | Prototype Application

Since CAMs are suitable for dataset annotator helpers, future work can develop a prototype application. Rather than using individual modules, a framework can be designed to support an end-to-end solution. This final application can be similar to the works presented in Section 3.4.2. Other possible inspirations can include enterprise systems like Microsoft Azure[1] or domestic applications like LabelImg[2].

[1]https://azure.microsoft.com/en-us/
[2]https://github.com/tzutalin/labelImg

6.3 | Closing

In conclusion, this study analysed the effectiveness of state of the art CV techniques. This was done by highlighting crowdsourcing and fully supervised learning products and their maintenance issues. Such issues were mitigated via the weakly supervised technique applied, which is known as CAMs. Through various experiments, it was determined that CAMs could be effective as digital aids. Therefore, continuously improving this methodology would serve to not only improve the overall image annotation experience, but also provide higher quality AI models within CV.

A

Media Content

The implementation consisted of different modules that helped realise this study. Each module was designed through either large projects or small, modular programs. These files are presented here for further review, if necessary. Each section represents a folder.

A.1 | Web Application

File / Folder	Comment
css	Folder containing the CSS files that determined page styling
images	Folder housing the survey images
js	Folder storing all of the website's front-end logic
utility	Folder storing purely back-end code (e.g. load balancing)
annotation.php	Localisation part of the web application
error.php	Dedicated error page for when something went wrong
finish.php	Page that informed the participants that they were done
footer.php	Footer content
head.php	Common page metadata
header.php	Header content
index.php	Participant starting point
instructions.php	Instructions for participants
labelling.php	Classification part of the web application
maintenance.php	Prevented access to the survey when required

Table A.1: Web application file structure

A.2 | Evaluation Programs

File / Folder	Comment
AutoUnzip.ipnyb	Used to uncompress ImageNet dataset
calc_time.py	Calculated pseudo-session times
camutils.py	Stored all CAM logic
datasetsizer.py	Created the hyperparameter tuning subset
Eval.ipynb	Used to generate metrics for deep image classifiers
resultutils.py	Defined result types and metrics
Robustness.ipynb	Created the big_12 superset
Starter.ipynb	Used to tinker with CAMs
survey2_imgs.py	Generated the web form's annotation choice images
survey2_results.py	Calculated the participants' web form performance
survey2_survey_bbs.py	Stored survey entries for survey2_imgs.py
survey2_wsl_bbs.py	Stored model localisations for survey2_imgs.py
surveyresults_both.py	Calculated all LE results of survey entries
surveyresults_cls.py	Calculated classification metrics of survey entries
surveyresults_loc.py	Calculated bounding box metrics of survey entries
surveyresults_time.py	Determined Objective 1's results
Train.ipynb	Trained the deep image classifiers
weaksupervision.py	Generated bounding boxes from heatmaps
wslresults_both.py	Calculated all LE results of the models
wslresults_cls.py	Calculated classification metrics of survey models
wslresults_loc.py	Calculated bounding box metrics of survey models
wslresults_time.py	Determined the inference times of models
xmltosql.py	Generated SQL to insert ground truth to database

Table A.2: Evaluation files. The files "AutoUnzip.ipynb" and "Robustness.ipynb" used the "wnid" environment. The rest of the files used the "thesis" environment.

A.3 | Environments

File / Folder	Comment
thesis.yml	Evaluation programs environment
wnid.yml	Robustness environment

Table A.3: Environment files

A.4 | Results

File / Folder	Comment
enb0	Trained EfficientNetB0 checkpoint folder
mnv2	Trained MobileNetV2 checkpoint folder
vgg16	Trained VGG16 checkpoint folder
form	Web form annotation images by question
responses.csv	Web form responses
websurvey.json	Interactive web survey exported database

Table A.4: Result files

B

Implementation

In Appendix A, the code related to the implementation of this study was listed. This code was supported by the use of various environments and frameworks, which allowed this study to take shape. This supporting set of packages are briefly discussed in the following sections.

B.1 | Environments

In Table A.3, two environments were listed. These environments organised and managed various libraries, frameworks and helpers that aided this study's development. Both environments were created, managed and exported through the Anaconda package manager[1]. Some packages, like Robustness, were not available through Anaconda. To install these missing packages, this study used the Pip package manager[2] in conjunction with Anaconda.

B.2 | Development

In this section, some specific packages are highlighted to provide further implementation details.

B.2.1 | Web Application

The web application was implemented by using the following web technologies:

[1]https://www.anaconda.com/
[2]https://pip.pypa.io/en/stable/

- Front-end: HTML, CSS, and JavaScript
- Back-end: PHP and SQL

In addition to the aforementioned technologies, various tools and libraries were used to simplify development:

- Database: phpMyAdmin[3]
- Libraries: Bootstrap[4], jQuery[5], and fabricJS[6]

B.2.2 | Scripts

The training and evaluation programs used various packages to create the models and generate the results. The base programming language utilised is Python, due to its vast support. In fact, Jupyter Notebooks were used to segment the algorithm training code into different cells. To train the models and implement the CAM techniques, the TensorFlow framework[7] was used in conjunction with its Keras API. OpenCV[8] and Pillow[9] were used to process the images, for instance, to generate the bounding boxes from the heatmap. Other packages, such as Matplotlib[10], were used to generate the visualisations of the model training sessions. The rest of the packages used can be observed by importing the supplied .yml files through Anaconda or any other package manager into an environment.

[3]https://www.phpmyadmin.net/
[4]https://getbootstrap.com/
[5]https://jquery.com/
[6]http://fabricjs.com/
[7]https://www.tensorflow.org/
[8]https://opencv.org/
[9]https://pillow.readthedocs.io/en/stable/
[10]https://matplotlib.org/

C

Supplement

This appendix provides extra content related to the modules. Mainly, the content includes sample images of the web application and the web form.

C.1 | Web Application

The web application used various technologies to ensure good UI and UX design. Some sample images of the web application can be found in Figures C.1 and C.2.

C.2 | Web Form

This section presents multiple images of the web form's statistical summary. Figures C.3 to C.19 serve to visualise the participants' demographic, performance and sentiments.

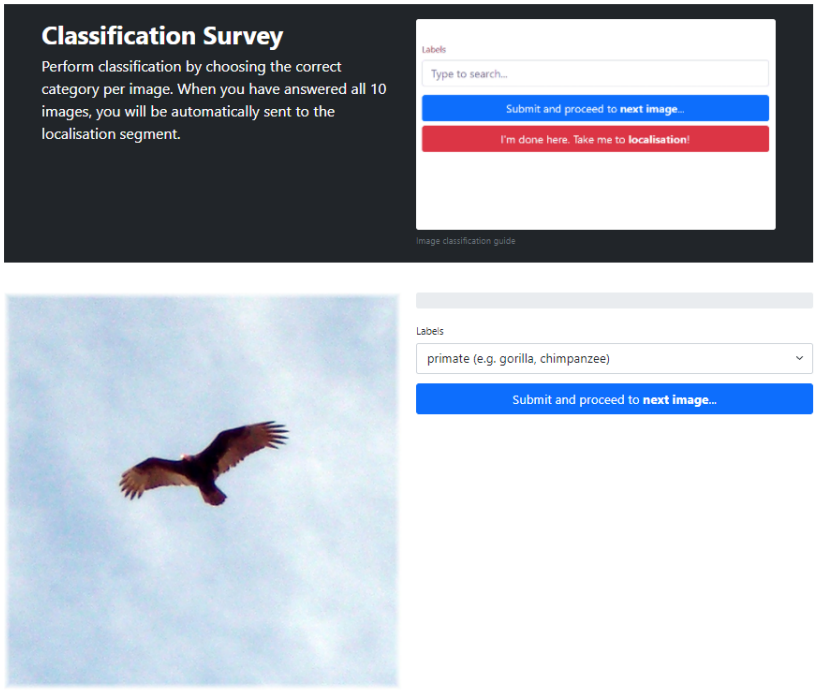


Figure C.1: Classification segment sample

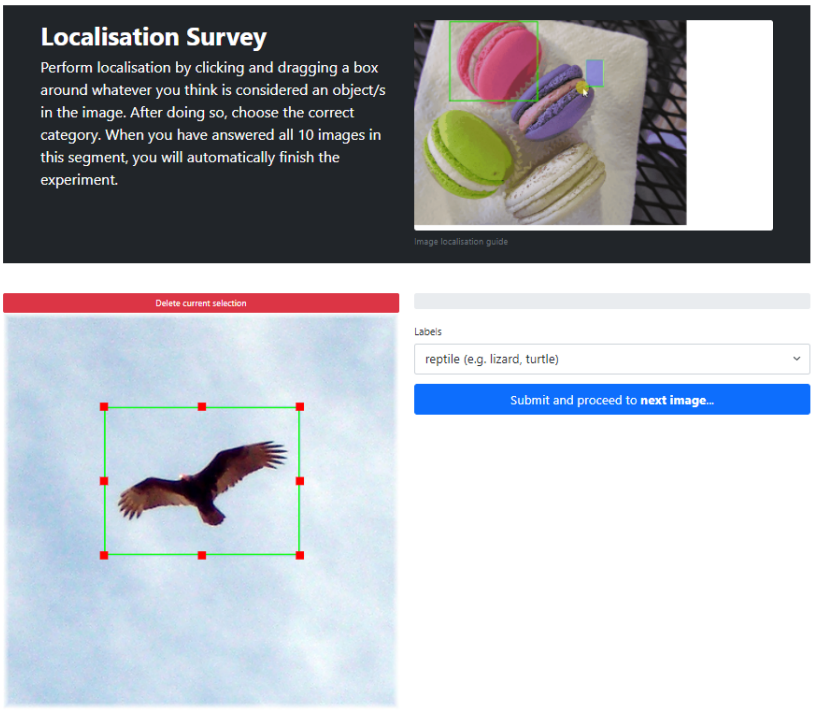


Figure C.2: Localisation segment sample

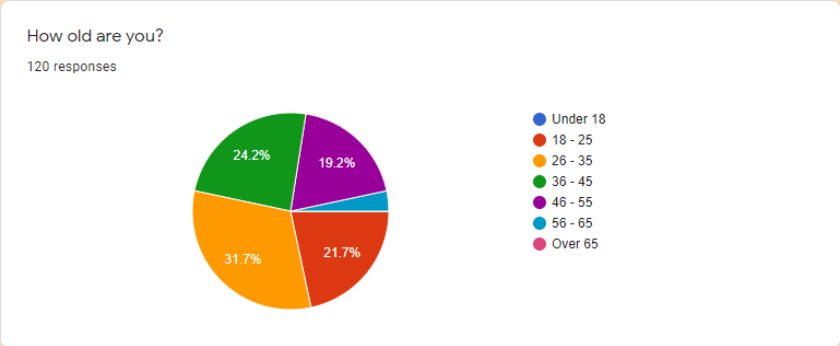


Figure C.3: Age question summary

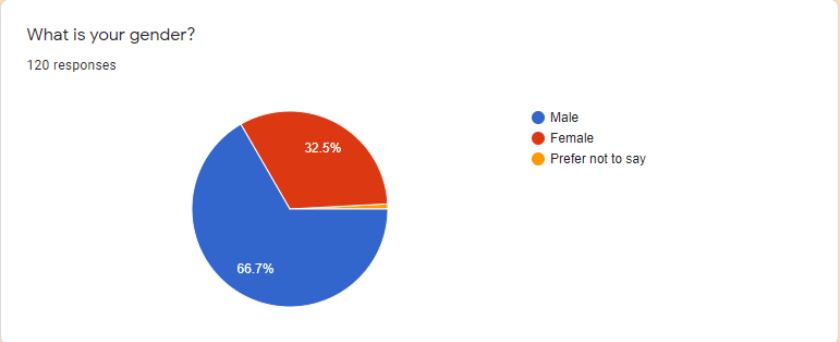


Figure C.4: Gender question summary

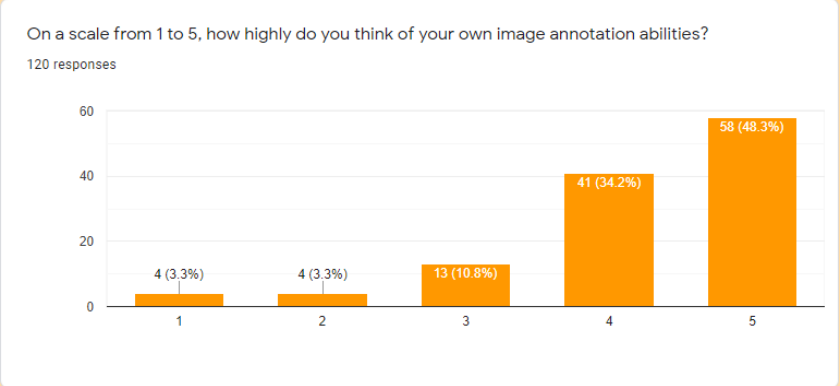


Figure C.5: Annotation confidence question summary

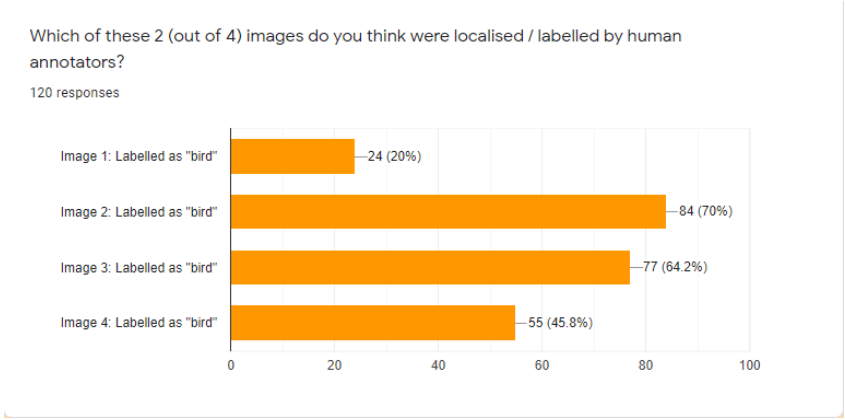


Figure C.6: Annotation question 1 summary. Man-made annotations were options 1 and 2.

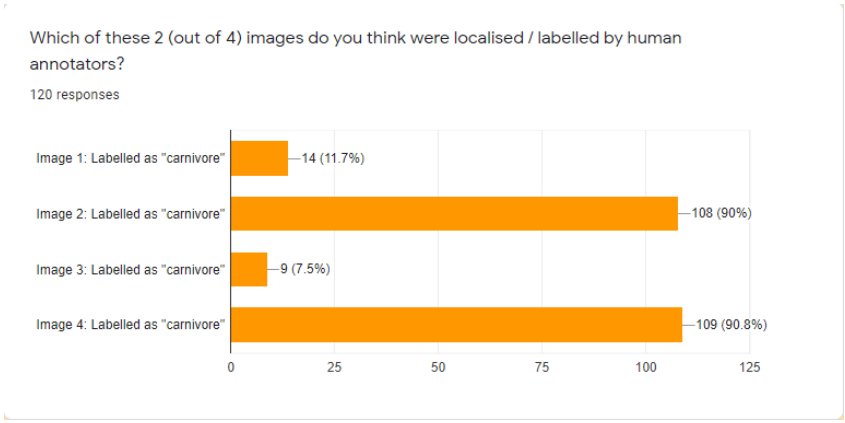


Figure C.7: Annotation question 2 summary. Man-made annotations were options 1 and 2.

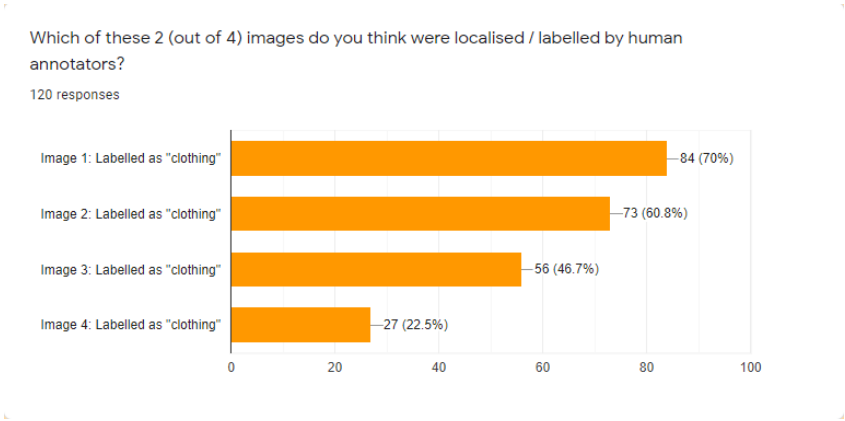


Figure C.8: Annotation question 3 summary. Man-made annotations were options 1 and 4.

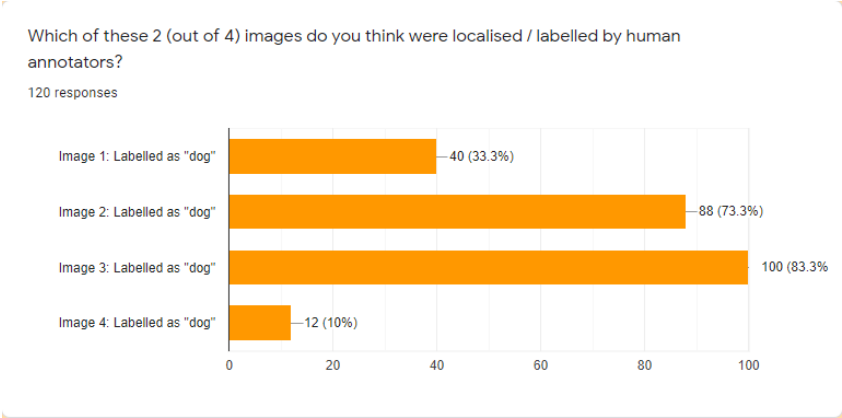


Figure C.9: Annotation question 4 summary. Man-made annotations were options 3 and 4.

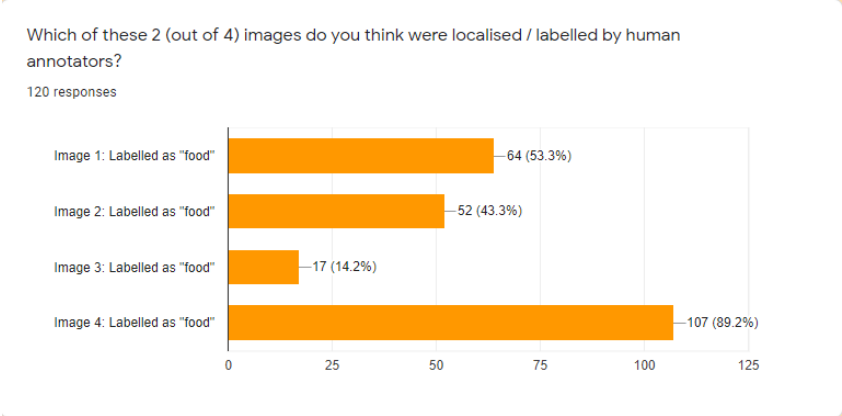


Figure C.10: Annotation question 5 summary. Man-made annotations were options 2 and 4.

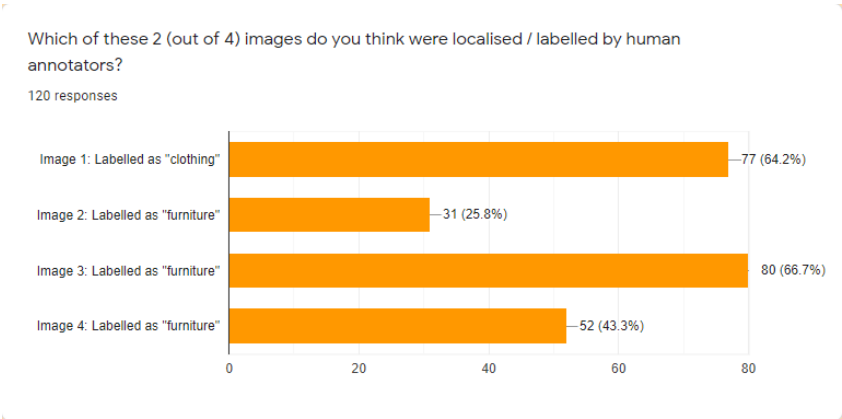


Figure C.11: Annotation question 6 summary. Man-made annotations were options 1 and 4.

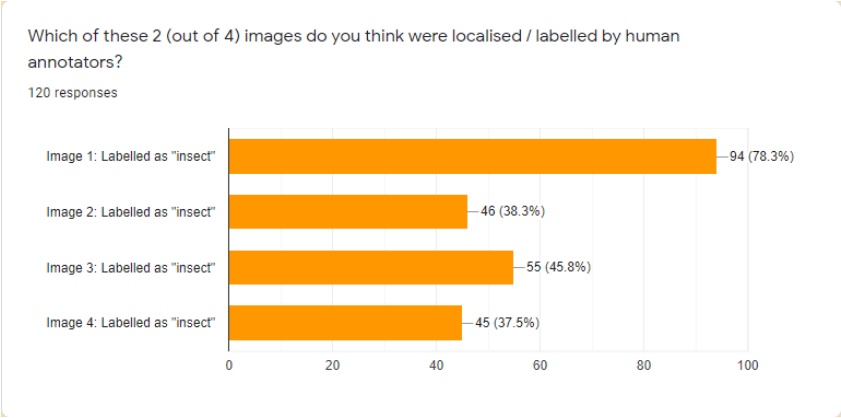


Figure C.12: Annotation question 7 summary. Man-made annotations were options 1 and 4.

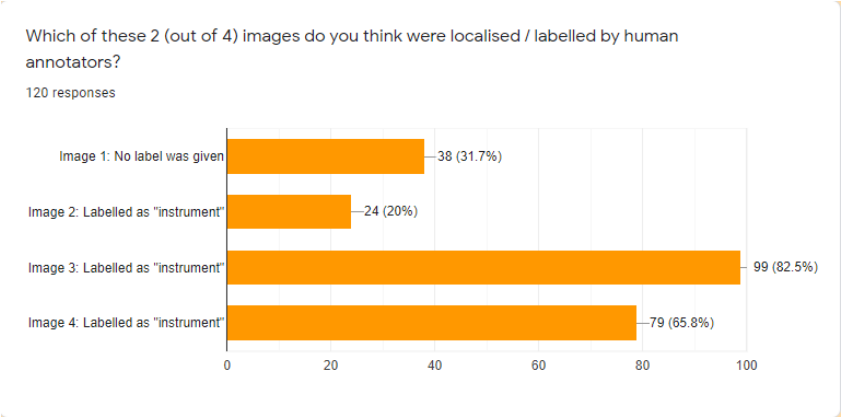


Figure C.13: Annotation question 8 summary. Man-made annotations were options 1 and 3.

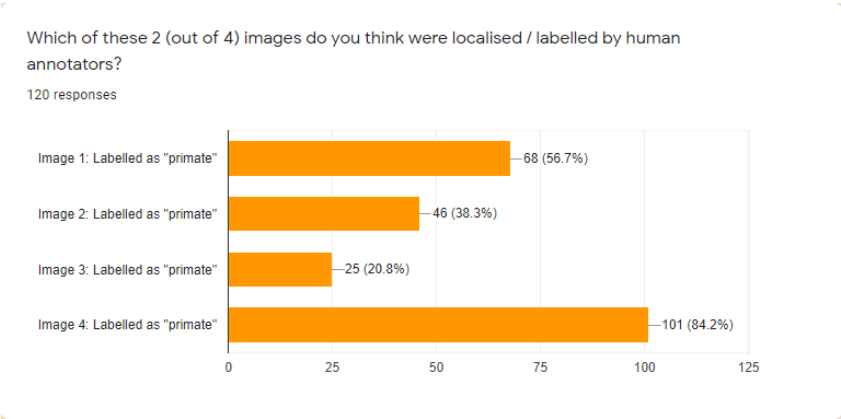


Figure C.14: Annotation question 9 summary. Man-made annotations were options 3 and 4.

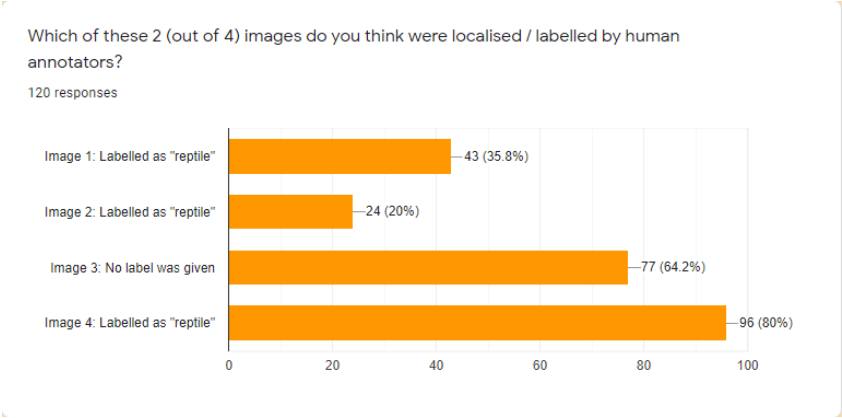


Figure C.15: Annotation question 10 summary. Man-made annotations were options 3 and 4.

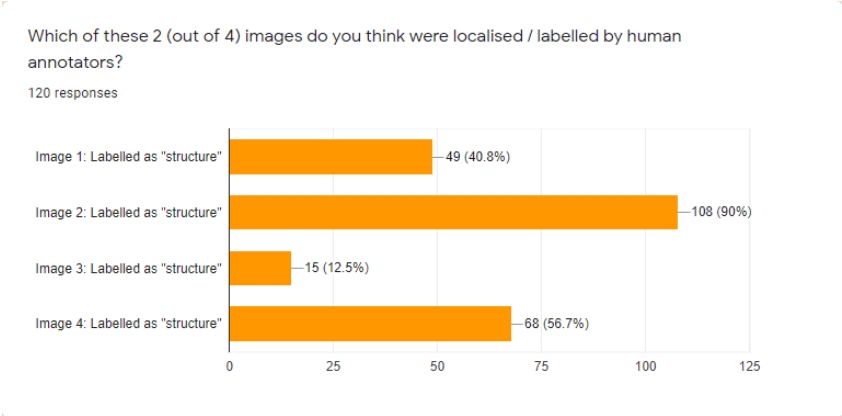


Figure C.16: Annotation question 11 summary. Man-made annotations were options 1 and 2.

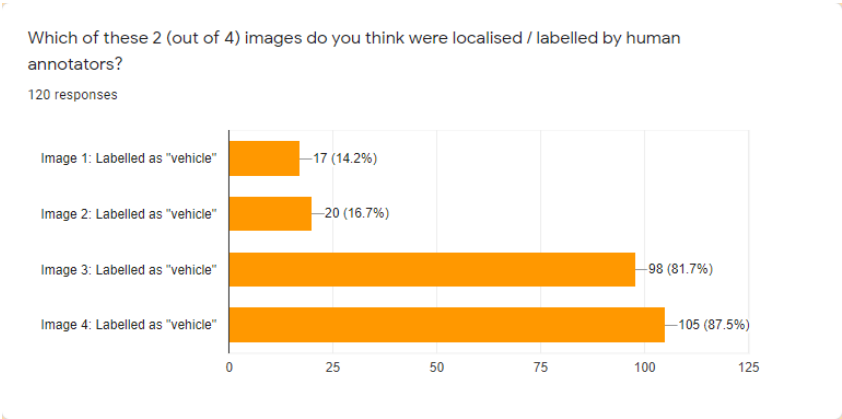


Figure C.17: Annotation question 12 summary. Man-made annotations were options 2 and 3.

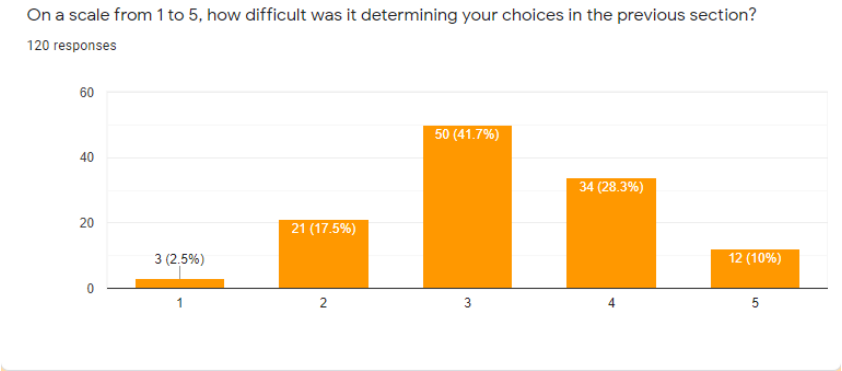


Figure C.18: Difficulty question summary

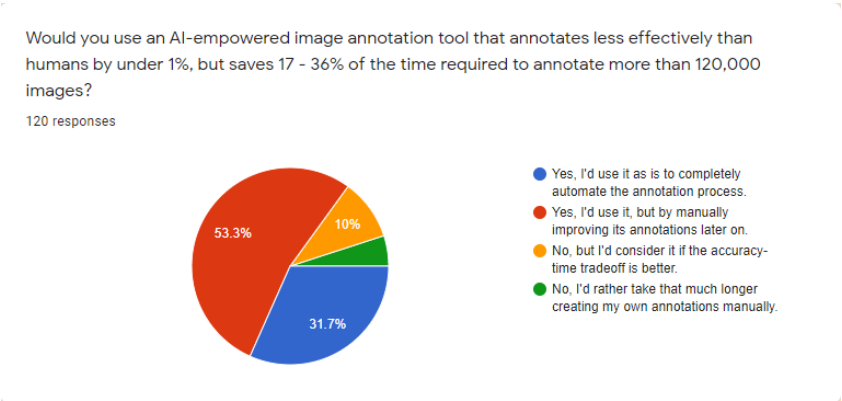


Figure C.19: Dataset annotation helper opinion question summary

References

[1] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 10 2017, pp. 843–852. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Sun_Revisiting_Unreasonable_Effectiveness_ICCV_2017_paper.html

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: Constructing a large-scale image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 08 2013. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/0278364913491297

[4] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord, "Are we done with imagenet?" arXiv:2006.07159 [cs], 06 2020. [Online]. Available: https://arxiv.org/abs/2006.07159v1

[5] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *Int J Comput Vision*, vol. 128, pp. 1956–1981, 07 2020. [Online]. Available: https://link.springer.com/article/10.1007%2Fs11263-020-01316-z

[6] C. Schumann, S. Ricco, U. Prabhu, V. Ferrari, and C. Pantofaru, "A step toward more inclusive people annotations for fairness," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 07 2021, pp. 916–925. [Online]. Available: https://arxiv.org/abs/2105.02317v1

[7] D. Seychell, C. J. Debono, M. Bugeja, J. Borg, and M. Sacco, "Cots: A multipurpose rgb-d dataset for saliency and image manipulation applications," *IEEE Access*, vol. 9, pp. 21 481–21 497, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9340352

[8] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 07 2018, pp. 2556–2565. [Online]. Available: https://aclanthology.org/P18-1238/

[9] B. M. Good, S. Loguercio, O. L. Griffith, M. Nanis, C. Wu, and A. I. Su, "The cure: Design and evaluation of a crowdsourcing game for gene selection for breast cancer survival prediction," *JMIR Serious Games*, vol. 2, p. e7, 07 2014. [Online]. Available: https://games.jmir.org/2014/2/e7?newDesign

[10] J. C. Chang, S. Amershi, and E. Kamar, "Revolt: Collaborative crowdsourcing for labeling machine learning datasets," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association

for Computing Machinery, 05 2017, pp. 2334–2346. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3025453.3026044

[11] V. S. Sheng and J. Zhang, "Machine learning with crowdsourcing: A brief summary of the past research and future directions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9837–9843, 07 2019. [Online]. Available: https://www.researchgate.net/publication/335221485_Machine_Learning_with_Crowdsourcing_A_Brief_Summary_of_the_Past_Research_and_Future_Directions

[12] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *Natl. Sci. Rev.*, vol. 5, pp. 44–53, 01 2018. [Online]. Available: https://academic.oup.com/nsr/article/5/1/44/4093912

[13] J. Correia, I. Trancoso, and B. Raj, "Automatic in-the-wild dataset annotation with deep generalized multiple instance learning," in *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, 05 2020, pp. 3542–3550. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.435/

[14] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, pp. 709–730, 05 2020. [Online]. Available: https://link.springer.com/article/10.1007/s00778-019-00552-1

[15] P. Varma and C. Ré, "Snuba: Automating weak supervision to label training data," *Proceedings of the VLDB Endowment*, vol. 12, pp. 223–236, 11 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6879381/

[16] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. Aroyo, ""everyone wants to do the model work, not the data work": Data cascades in high-stakes ai," in *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 05 2021, pp. 1–15. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3411764.3445518

[17] N. Buduma and N. Locascio, *Fundamentals of Deep Learning : Designing Next-generation Artificial Intelligence Algorithms*. O'Reilly Media, Inc., 2017. [Online]. Available: http://perso.ens-lyon.fr/jacques.jayez/Cours/Implicite/Fundamentals_of_Deep_Learning.pdf

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: from Theory to Algorithms*. Cambridge University Press, 07 2014. [Online]. Available: https://www.cambridge.org/core/books/understanding-machine-learning/3059695661405D25673058E43C8BE2A6

[19] W. Khan, U. Malik, M. A. Ghazanfar, M. A. Azam, K. H. Alyoubi, and A. S. Alfakeeh, "Predicting stock market trends using machine learning algorithms via public sentiment and political situation analysis," *Soft Comput*, vol. 24, pp. 11 019–11 043, 08 2020. [Online]. Available: https://link.springer.com/article/10.1007/s00500-019-04347-y

[20] C. Hasabnis, S. Dhaygude, and S. Ruikar, "Real-time lane detection for autonomous vehicle using video processing," in *ICT Analysis and Applications*, vol. 93. Springer Singapore, 2020, pp. 217–225. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-0630-7_21

[21] G. Caruso and S. A. Gattone, "Waste management analysis in developing countries through unsupervised classification of mixed data," *Social Sciences*, vol. 8, p. 186, 06 2019. [Online]. Available: https://www.mdpi.com/2076-0760/8/6/186

[22] K. Sailunaz and R. Alhajj, "Emotion and sentiment analysis from twitter text," *J. Comput. Sci.*, vol. 36, p. 101003, 09 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1877750318311037

[23] P. Mehndiratta, S. Sachdevai, and D. Soni, "Detection of sarcasm in text data using deep convolutional neural networks," *Scalable Computing: Practice and Experience*, vol. 18, 09 2017. [Online]. Available: https://www.scpe.org/index.php/scpe/article/view/1302

[24] A. E. Puerto Lara, C. Pedraza, and D. A. Jamaica-Tenjo, *Weed Estimation on Lettuce Crops Using*

Histograms of Oriented Gradients and Multispectral Images, ser. Pattern Recognition Applications in Engineering. IGI Global, 2020, pp. 204–228. [Online]. Available: https://www.igi-global.com/chapter/weed-estimation-on-lettuce-crops-using-histograms-of-oriented-gradients-and-multispectral-images/247798

[25] P. S. Sharma, P. K. Roy, N. Ahmad, J. Ahuja, and N. Kumar, "Localisation of license plate and character recognition using haar cascade," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, vol. 6, 03 2019, pp. 971–974. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8991267

[26] J. Yang, R. Wang, X. Guan, M. M. Hassan, A. Almogren, and A. Alsanad, "Ai-enabled emotion-aware robot: The fusion of smart clothing, edge clouds and robotics," *Future Gener Comp Sy*, vol. 102, pp. 701–709, 01 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19305643

[27] E. Asadi, B. Li, and I.-M. Chen, "Pictobot: A cooperative painting robot for interior finishing of industrial developments," *IEEE Robot. Autom. Mag.*, vol. 25, pp. 82–94, 06 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8360499

[28] R. González Perea, E. Camacho Poyato, P. Montesinos, and J. Rodríguez Díaz, "Prediction of irrigation event occurrence at farm level using optimal decision trees," *Comput Electron Agr*, vol. 157, pp. 173–180, 02 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0168169918309062

[29] B. Yeo and D. Grant, "Predicting service industry performance using decision tree analysis," *Int J Inform Manage*, vol. 38, pp. 288–300, 02 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0268401217304127

[30] J. Gola, J. Webel, D. Britz, A. Guitar, T. Staudt, M. Winter, and F. Mücklich, "Objective microstructure classification by support vector machine (svm) using a combination of morphological parameters and textural features for low carbon steels," *Comp Mater Sci*, vol. 160, pp. 186–196, 04 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0927025619300060

[31] A. R. Mathew and P. B. Anto, "Tumor detection and classification of mri brain image using wavelet transform and svm," in *2017 International Conference on Signal Processing and Communication (ICSPC)*, 07 2017, pp. 75–78. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8305810

[32] P. Hou, B. Zhao, O. Jolliet, J. Zhu, P. Wang, and M. Xu, "Rapid prediction of chemical ecotoxicity through genetic algorithm optimized neural network models," *ACS Sustainable Chemistry and Engineering*, vol. 8, pp. 12 168–12 176, 07 2020. [Online]. Available: https://pubs.acs.org/doi/pdf/10.1021/acssuschemeng.0c03660

[33] W. Halecki, D. Młyński, M. Ryczek, E. Kruk, and A. Radecki-Pawlik, "Applying an artificial neural network (ann) to assess soil salinity and temperature variability in agricultural areas of a mountain catchment," *Pol J Environ Stud*, vol. 26, pp. 2545–2554, 11 2017. [Online]. Available: http://www.pjoes.com/Applying-an-Artificial-Neural-Network-ANN-to-Assess-Soil-Salinity-and-Temperature,70925,0,2.html

[34] D. Suhandy and M. Yulia, "The classification of arabica gayo wine coffee using uv-visible spectroscopy and pca-da method," *MATEC Web of Conferences*, vol. 197, p. 09002, 01 2018. [Online]. Available: https://www.matec-conferences.org/articles/matecconf/abs/2018/56/matecconf_aasec2018_09002/matecconf_aasec2018_09002.html

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, 04 2015. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision – ECCV 2016*. Springer International Publishing, 07 2016, pp. 630–645. [Online]. Available: https://arxiv.org/abs/1603.05027v3

[37] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv:2004.10934 [cs, eess], 04 2020. [Online]. Available: https://arxiv.org/abs/2004.10934v1

[38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[39] M. Z. Alom, C. Yakopcic, M. S. Nasrin, T. M. Taha, and V. K. Asari, "Breast cancer classification from histopathological images with inception recurrent residual convolutional neural network," *J. Digit. Imaging*, vol. 32, pp. 605–617, 02 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10278-019-00182-7

[40] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," arXiv:1311.2524 [cs], 10 2014. [Online]. Available: https://arxiv.org/abs/1311.2524v5

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2016, pp. 779–788. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

[42] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2020, pp. 10 781–10 790. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.html

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *Int J Comput Vision*, vol. 115, no. 3, pp. 211–252, 4 2015.

[44] V. Kumar, J. M. Webb, A. Gregory, M. Denis, D. D. Meixner, M. Bayat, D. H. Whaley, M. Fatemi, and A. Alizad, "Automated and real-time segmentation of suspicious breast masses using convolutional neural network," *PLoS One*, vol. 13, p. e0195816, 05 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5955504/

[45] N. Micallef, D. Seychell, and C. J. Bajada, "Exploring the u-net++ model for automatic brain tumor segmentation," *IEEE Access*, vol. 9, pp. 125 523–125 539, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9530684

[46] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*, vol. 43, pp. 244–252, 12 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0386111219301566

[47] S. Higa, Y. Iwashita, K. Otsu, M. Ono, O. Lamarre, A. Didier, and M. Hoffmann, "Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics," *IEEE Robot. Autom. Lett.*, vol. 4, pp. 3876–3883, 10 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8764007

[48] B. Santra and D. Prasad Mukherjee, "A comprehensive survey on computer vision based approaches for automatic identification of products in retail store," *Image Vision Comput*, vol. 86, pp. 45–63, 03 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0262885619300277

[49] H. Jain, A. Vikram, Mohana, A. Kashyap, and A. Jain, "Weapon detection using artificial intelligence and deep learning for security applications," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 07 2020, pp. 193–198. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9155832

[50] G. Morales, I. Salazar-Reque, J. Telles, and D. Díaz, "Detecting violent robberies in cctv videos using deep learning," in *AIAI 2019: Artificial Intelligence Applications and Innovations*, 2019, pp. 282–291. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-19823-7_23

[51] A. Abubakar, H. Ugail, A. M. Bukar, A. A. Aminu, and A. Musa, "Transfer learning based histopathologic image classification for burns recognition," in *15th International Conference on Electronics, Computer and Computation (ICECCO)*, 12 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9043205

[52] T. Kaur and T. K. Gandhi, "Automated brain image classification based on vgg-16 and transfer learning," in *2019 International Conference on Information Technology (ICIT)*, 12 2019, pp. 94–98. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9031952

[53] M. Schembri and D. Seychell, "Small object detection in highly variable backgrounds," in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 09 2019, pp. 32–37. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8868719

[54] Z. Zhao, Z. Zhang, X. Xu, Y. Xu, H. Yan, and L. Zhang, "A lightweight object detection network for real-time detection of driver handheld call on embedded devices," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–12, 12 2020. [Online]. Available: https://www.hindawi.com/journals/cin/2020/6616584/

[55] H.-H. Nguyen, D. N.-N. Tran, and J. W. Jeon, "Towards real-time vehicle detection on edge devices with nvidia jetson tx2," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 11 2020, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9277463

[56] S. Naddaf-Sh, M.-M. Naddaf-Sh, A. R. Kashani, and H. Zargarzadeh, "An efficient and scalable deep learning approach for road damage detection," in *2020 IEEE International Conference on Big Data (Big Data)*, 12 2020, pp. 5602–5608. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9377751

[57] D. Heo, E. Lee, and B. C. Ko, "Pedestrian detection at night using deep neural networks and saliency maps," *Electronic Imaging*, vol. 2018, pp. 060 403–1–060 403–9, 01 2018. [Online]. Available: https://www.ingentaconnect.com/content/ist/ei/2018/00002018/00000017/art00015

[58] L.-C. Chen, R. G. Lopes, B. Cheng, M. D. Collins, E. D. Cubuk, B. Zoph, H. Adam, and J. Shlens, "Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 695–714. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58545-7_40

[59] H. I. Dino and M. B. Abdulrazzaq, "Facial expression classification based on svm, knn and mlp classifiers," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 04 2019, pp. 70–75. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8723728

[60] T. K. Hazra, D. P. Singh, and N. Daga, "Optical character recognition using knn on custom image dataset," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, 08 2017, pp. 110–114. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8079572

[61] N. Krithika and A. G. Selvarani, "An individual grape leaf disease identification using leaf skeletons and knn classification," *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–5, 03 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8275951

[62] E. Arboleda, K. Rabe, and R. Dellosa, "Fuzzy logic based vehicular congestion estimation monitoring system using image processing and knn classifier," *International Journal of Scientific and Technology Research*, vol. 8, pp. 1377–1380, 08 2019. [Online]. Available: https://www.researchgate.net/publication/335421548_Fuzzy_Logic_Based_Vehicular_Congestion_Estimation_Monitoring_System_Using_Image_Processing_And_KNN_Classifier

[63] K. Uma Maheswari and S. Rajesh, "A novel qim-dct based fusion approach for classification of remote sensing images via pso and svm models," *Soft Comput*, vol. 24, pp. 15 561–15 576, 10 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1007/s00500-020-04884-x

[64] W. Jiang, G. He, T. Long, Y. Ni, H. Liu, Y. Peng, K. Lv, and G. Wang, "Multilayer perceptron neural network for surface water extraction in landsat 8 oli satellite images," *Remote Sensing*, vol. 10, p. 755, 05 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/5/755

[65] A. Ulloa, S. Plis, and V. Calhoun, "Improving classification rate of schizophrenia using a multimodal multi-layer perceptron model with structural and functional mr," arXiv:1804.04591 [cs], 04 2018. [Online]. Available: https://arxiv.org/abs/1804.04591v1

[66] J. S. Ellen, C. A. Graff, and M. D. Ohman, "Improving plankton image classification using context metadata," *Limnol. Oceanogr. Methods*, vol. 17, pp. 439–461, 08 2019. [Online]. Available: https://aslopubs.onlinelibrary.wiley.com/doi/full/10.1002/lom3.10324

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 01 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[68] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on alexnet and transfer learning," *J. Comput. Sci.*, vol. 30, pp. 41–47, 01 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750318309116

[69] A. Seker, "Evaluation of fabric defect detection based on transfer learning with pre-trained alexnet," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 09 2018, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8620888

[70] S. Samir, E. Emary, K. El-Sayed, and H. Onsi, "Optimization of a pre-trained alexnet model for detecting and localizing image forgeries," *Information*, vol. 11, p. 275, 05 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/5/275

[71] Z. Bi, L. Yu, H. Gao, P. Zhou, and H. Yao, "Improved vgg model-based efficient traffic sign recognition for safe driving in 5g scenarios," *International Journal of Machine Learning and Cybernetics*, vol. 12, pp. 3069–3080, 08 2020. [Online]. Available: https://link.springer.com/article/10.1007/s13042-020-01185-5

[72] X. Chang, J. Wu, T. Yang, and G. Feng, "Deepfake face image detection based on improved vgg convolutional neural network," in *2020 39th Chinese Control Conference (CCC)*, 07 2020, pp. 7252–7256. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9189596

[73] A. Abela and T. Gatt, "Using class activation maps on deep neural networks to localise waste classifications," in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 01 2021, pp. 000 143–000 148. [Online]. Available: https://ieeexplore.ieee.org/document/9378662

[74] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2015, pp. 1–9. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

[75] J.-w. Feng and X.-y. Tang, "Office garbage intelligent classification based on inception-v3 transfer learning model," *J. Phys.: Conf. Ser.*, vol. 1487, p. 012008, 03 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1487/1/012008/meta

[76] Z. Gao, M. Li, W. Li, and Q. Yan, "Classification of flowers under complex background using inception-v3 network," in *Proceedings of the 2020 4th International Conference on Deep Learning Technologies (ICDLT)*. Association for Computing Machinery, 07 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3417188.3417192

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2016. [Online]. Available: https://arxiv.org/abs/1512.03385v1

[78] U. Seidaliyeva, M. Alduraibi, L. Ilipbayeva, and N. Smailov, "Deep residual neural network-based classification of loaded and unloaded uav images," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, 11 2020, pp. 465–469. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9287925

[79] W. Song, M. Li, W. Gao, D. Huang, Z. Ma, A. Liotta, and C. Perra, "Automatic sea-ice classification of sar images based on spatial and temporal features learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, pp. 9887–9901, 12 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9332239

[80] A. Thomas, H. P. M., P. P., and V. P. Gopi, "Moving vehicle candidate recognition and classification using

inception-resnet-v2," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 07 2020, pp. 467–472. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9202415

[81] S. Voinov, F. Heymann, R. Bill, and E. Schwarz, "Multiclass vessel detection from high resolution optical satellite images based on deep neural networks," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 07 2019, pp. 166–169. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8900506

[82] X. Zhang, Y. Li, Y. Tao, Y. Wang, C. Xu, and Y. Lu, "A novel method based on infrared spectroscopic inception-resnet networks for the detection of the major fish allergen parvalbumin," *Food Chem*, vol. 337, p. 127986, 02 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0308814620318483

[83] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861 [cs], 04 2017. [Online]. Available: https://arxiv.org/abs/1704.04861v1

[84] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," arXiv:1610.02357 [cs], 04 2017. [Online]. Available: https://arxiv.org/abs/1610.02357v3

[85] R. Plonus, J. Conradt, A. Harmer, S. Janßen, and J. Floeter, "Automatic plankton image classification - can capsules and filters help cope with data set shift?" *Limnol. Oceanogr. Methods*, vol. 19, pp. 176–195, 01 2021. [Online]. Available: https://aslopubs.onlinelibrary.wiley.com/doi/full/10.1002/lom3.10413

[86] M. F. Wahid, M. J. Hasan, and M. S. Alom, "Deep convolutional neural network for microscopic bacteria image classification," in *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, 09 2019, pp. 866–869. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8975588

[87] H. Chen, Y. Yang, and S. Zhang, "Learning robust scene classification model with data augmentation based on xception," *J. Phys.: Conf. Ser.*, vol. 1575, p. 012009, 06 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1575/1/012009/meta

[88] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018, pp. 4510–4520. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[89] S. L. Rabano, M. K. Cabatuan, E. Sybingco, E. P. Dadios, and E. J. Calilung, "Common garbage classification using mobilenet," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 11 2018, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/8666300

[90] B. Yassine, G. Larbi, and L. Hicham, "Human detection in surveillance videos using mobilenet," in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 10 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9257662

[91] F. Wu, Y.-C. Lin, and Y.-H. Wu, "Cnn-based medical device design for the smart pillbox with face recognition," in *2020 International Conference on Culture-oriented Science & Technology (ICCST)*, 10 2020, pp. 607–611. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9262822

[92] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018, pp. 8697–8710. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html

[93] K. Radhika, K. Devika, T. Aswathi, P. Sreevidya, V. Sowmya, and K. P. Soman, *Performance Analysis of NASNet on Unconstrained Ear Recognition*, ser. Nature Inspired Computing for Data Science. Springer International Publishing, 2020, pp. 57–82. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-33820-6_3

[94] S. Bakkali, Z. Ming, M. Coustaty, and M. Rusinol, "Cross-modal deep networks for document image classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, 10 2020, pp. 2556–2560. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9191268

[95] A. Bahri, S. G. Majelan, S. Mohammadi, M. Noori, and K. Mohammadi, "Remote sensing image classification via improved cross-entropy loss and transfer learning strategy based on deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, pp. 1087–1091, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8844264

[96] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97. PMLR, 06 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[97] J. Liu, M. Wang, L. Bao, and X. Li, "Efficientnet based recognition of maize diseases by leaf image classification," *J. Phys.: Conf. Ser.*, vol. 1693, p. 012148, 12 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012148

[98] A. Noor, B. Benjdira, A. Ammar, and A. Koubaa, "Driftnet: Aggressive driving behavior classification using 3d efficientnet architecture," arXiv:2004.11970 [cs], 04 2020. [Online]. Available: https://arxiv.org/abs/2004.11970

[99] S. Kalvankar, H. Pandit, and P. Parwate, "Galaxy morphology classification using efficientnet architectures," 08 2020. [Online]. Available: https://arxiv.org/abs/2008.13611

[100] H. Wang, D. Nie, X. Tuo, and Y. Zhong, "Research on crack monitoring at the trailing edge of landslides based on image processing," *Landslides*, vol. 17, pp. 985–1007, 04 2020. [Online]. Available: https://link.springer.com/article/10.1007%2Fs10346-019-01335-z

[101] O. Déniz, G. Bueno, J. Salido, and F. De la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recogn Lett*, vol. 32, pp. 1598–1603, 09 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167865511000122

[102] T. Watanabe, S. Ito, and K. Yokoi, "Co-occurrence histograms of oriented gradients for human detection," *IPSJ Transactions on Computer Vision and Applications*, vol. 2, pp. 39–47, 2010. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjtcva/2/0/2_0_39/_article/-char/ja/

[103] C. G. Dias, L. C. da Silva, and W. A. Luz Alves, "A histogram of oriented gradients approach for detecting broken bars in squirrel-cage induction motors," *IEEE Trans. Instrum. Meas.*, vol. 69, pp. 6968–6981, 09 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9005202

[104] J. G. Reiser de Melo, L. Paes de Souza, D. Frossard, and R. R. de Oliveira, "Sign language interpreter detection method for live tv broadcast content," in *HCI International 2020 – Late Breaking Papers: Universal Access and Inclusive Design*, vol. 22. Springer International Publishing, 2020, pp. 318–332. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-60149-2_25

[105] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 02 2001, p. 1. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/990517

[106] C. Rahmad, R. A. Asmara, D. R. H. Putra, I. Dharma, H. Darmono, and I. Muhiqqin, "Comparison of viola-jones haar cascade classifier and histogram of oriented gradients (hog) for face detection," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 732, p. 012038, 01 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/732/1/012038/meta

[107] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," arXiv:1312.6229 [cs], 02 2014. [Online]. Available: https://arxiv.org/abs/1312.6229v4

[108] D. Biswas, H. Su, C. Wang, J. Blankenship, and A. Stevanovic, "An automatic car counting system using overfeat

framework," *Sensors*, vol. 17, p. 1535, 06 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/7/1535

[109] M. S. Rad, A. von Kaenel, A. Droux, F. Tieche, N. Ouerhani, H. K. Ekenel, and J.-P. Thiran, "A computer vision system to localize and classify wastes on the streets," in *International Conference on Computer Vision Systems*, 10 2017, pp. 195–204. [Online]. Available: https://www.researchgate.net/publication/320301873_A_Computer_Vision_System_to_Localize_and_Classify_Wastes_on_the_Streets

[110] R. Girshick, "Fast r-cnn," arXiv:1504.08083 [cs], 09 2015. [Online]. Available: https://arxiv.org/abs/1504.08083v2

[111] N. Dhungel, G. Carneiro, and A. P. Bradley, "Automated mass detection in mammograms using cascaded deep learning and random forests," in *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 11 2015, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7371234

[112] S. Ahmad, N. Enshaei, F. Naderkhani, and A. Awasthi, "Integrated deep learning and statistical process control for online monitoring of manufacturing processes," in *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 06 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9187046

[113] Y. Su, D. Li, and X. Chen, "Lung nodule detection based on faster r-cnn framework," *Comput. Methods Programs Biomed.*, vol. 200, p. 105866, 03 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0169260720316990

[114] Y. Mao, J. Chen, P. Ping, and H. Chen, "Crack detection with multi-task enhanced faster r-cnn model," in *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, 08 2020, pp. 193–197. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9179493

[115] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

[116] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 07 2017, pp. 7263–7271. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.html

[117] ——, "Yolov3: An incremental improvement," arXiv.org, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767v1

[118] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 05 2017, pp. 1370–1377. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7989163

[119] Hendry and R.-C. Chen, "Automatic license plate recognition via sliding-window darknet-yolo deep learning," *Image Vision Comput*, vol. 87, pp. 47–56, 07 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0262885619300575

[120] S. Galea, D. Seychell, and M. Bugeja, "A survey of intelligent transportation systems based modern object detectors under night-time conditions," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 12 2020, pp. 265–270. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9316076

[121] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, 02 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8658300

[122] F. Xiang, G. Ding, J. Su, W. Zhang, X. Gu, and J. Wu, "Dangerous target recognition of massive image and video based on deep learning," in *2019 Chinese Automation Congress (CAC)*, 11 2019, pp. 454–459. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8996657

[123] H. Ma, Y. Liu, Y. Ren, and J. Yu, "Detection of collapsed buildings in post-earthquake remote sensing

images based on the improved yolov3," *Remote Sensing*, vol. 12, p. 44, 12 2019. [Online]. Available: https://www.mdpi.com/2072-4292/12/1/44

[124] Y. Li, H. Dong, H. Li, X. Zhang, B. Zhang, and Z. Xiao, "Multi-block ssd based on small object detection for uav railway scene surveillance," *Chinese J Aeronaut*, vol. 33, pp. 1747–1755, 06 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936120301126

[125] V. Thakar, H. Saini, W. Ahmed, M. M. Soltani, A. Aly, and J. Y. Yu, "Efficient single-shot multibox detector for construction site monitoring," in *2018 IEEE International Smart Cities Conference (ISC2)*, 09 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8656929

[126] Y. Chen and H. Shin, "Multispectral image fusion based pedestrian detection using a multilayer fused deconvolutional single-shot detector," *J. Opt. Soc. Amer. A*, vol. 37, pp. 768–779, 04 2020. [Online]. Available: https://www.osapublishing.org/josaa/abstract.cfm?uri=josaa-37-5-768

[127] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 07 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Lin_Feature_Pyramid_Networks_CVPR_2017_paper.html

[128] P. Chen, Y. Li, H. Zhou, B. Liu, and P. Liu, "Detection of small ship objects using anchor boxes cluster and feature pyramid network model for sar imagery," *Journal of Marine Science and Engineering*, vol. 8, p. 112, 02 2020. [Online]. Available: https://www.mdpi.com/2077-1312/8/2/112

[129] Y.-Y. Ou, A.-C. Tsai, X.-P. Zhou, and J.-F. Wang, "Automatic drug pills detection based on enhanced feature pyramid network and convolution neural networks," *IET Comput. Vision*, vol. 14, pp. 9–17, 02 2020. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-cvi.2019.0171

[130] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 10 2017. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html

[131] M. Liu, Y. Tan, and L. Chen, "Pneumonia detection based on deep neural network retinanet," in *2019 International Conference on Image and Video Processing, and Artificial Intelligence*, R. Su, Ed., vol. 11321. SPIE, 11 2019, pp. 86–92. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11321/113210F/Pneumonia-detection-based-on-deep-neural-network-Retinanet/10.1117/12.2539633.short?SSO=1&tab=ArticleLink

[132] A. Santos, J. Marcato Junior, J. de Andrade Silva, R. Pereira, D. Matos, G. Menezes, L. Higa, A. Eltner, A. P. Ramos, L. Osco, and W. Gonçalves, "Storm-drain and manhole detection using the retinanet method," *Sensors*, vol. 20, p. 4450, 08 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/16/4450

[133] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10 2019. [Online]. Available: https://arxiv.org/abs/1904.08189v3

[134] B. Wang, Q. Zhao, Y. Zhang, and J. Cheng, *A Detection Method of Safety Helmet Wearing Based on Centernet*, ser. The 10th International Conference on Computer Engineering and Networks. Springer Singapore, 01 2021, vol. 1274, pp. 1128–1137. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-8462-6_129

[135] T. Liu, F. Jiang, and R. Shen, "Fast and accurate hand-raising gesture detection in classroom," in *Neural Information Processing*, H. Yang, K. Pasupa, A. C.-S. Leung, J. T. Kwok, J. H. Chan, and I. King, Eds., vol. 1332. Springer International Publishing, 2020, pp. 232–239. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-63820-7_26

[136] R. Gupta, P. Hooda, Sanjeev, and N. Kumar Chikkara, "Natural language processing based visual question answering efficient: an efficientdet approach," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 05 2020, pp. 900–904. [Online]. Available: https://ieeexplore.ieee.org/abstract/

document/9121068

[137] T. You, W. Chen, H. Wang, Y. Yang, and X. Liu, "Automatic garbage scattered area detection with data augmentation and transfer learning in suav low-altitude remote sensing images," *Math Probl Eng*, vol. 2020, pp. 1–13, 10 2020. [Online]. Available: https://www.hindawi.com/journals/mpe/2020/7307629/

[138] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 740–755. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48

[139] J. Cui, K. Gong, N. Guo, C. Wu, X. Meng, K. Kim, K. Zheng, Z. Wu, L. Fu, B. Xu, Z. Zhu, J. Tian, H. Liu, and Q. Li, "Pet image denoising using unsupervised deep learning," *Eur. J. Nucl. Med. Mol. Imaging*, vol. 46, pp. 2780–2789, 08 2019. [Online]. Available: https://link.springer.com/article/10.1007%2Fs00259-019-04468-4

[140] S. Kuanar, V. Athitsos, D. Mahapatra, K. Rao, Z. Akhtar, and D. Dasgupta, "Low dose abdominal ct image reconstruction: An unsupervised learning based approach," in *2019 IEEE International Conference on Image Processing (ICIP)*, 09 2019, pp. 1351–1355. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8803037

[141] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen, "Unified binary generative adversarial network for image retrieval and compression," *Int J Comput Vision*, vol. 128, pp. 2243–2264, 09 2020. [Online]. Available: https://link.springer.com/article/10.1007/s11263-020-01305-2

[142] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805 [cs], 05 2019. [Online]. Available: https://arxiv.org/abs/1810.04805v2

[143] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," arXiv:2005.14165 [cs], 07 2020. [Online]. Available: https://arxiv.org/abs/2005.14165v4

[144] J. Jiao, R. Droste, L. Drukker, A. T. Papageorghiou, and J. A. Noble, "Self-supervised representation learning for ultrasound video," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 04 2020, pp. 1847–1850. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9098666

[145] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 69–84. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46466-4_5

[146] G. Lee, J. Jeong, S. Seo, C. Kim, and P. Kang, "Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network," *Knowledge-Based Systems*, vol. 152, pp. 70–82, 07 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950705118301710

[147] S. Marino, P. Beauseroy, and A. Smolarz, "Weakly-supervised learning approach for potato defects segmentation," *Eng Appl Artif Intel*, vol. 85, pp. 337–346, 10 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0952197619301630

[148] B. Yoon, Y. Jeong, and S. Kim, "Detecting a risk signal in stock investment through opinion mining and graph-based semi-supervised learning," *IEEE Access*, vol. 8, pp. 161 943–161 957, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9184877

[149] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach Learn*, vol. 109, pp. 373–440, 02 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10994-019-05855-6

[150] A. H. Halim and I. Ismail, "Combinatorial optimization: Comparison of heuristic algorithms in travelling salesman problem," *Arch Comput Method E*, vol. 26, pp. 367–380, 2019. [Online]. Available:

https://link.springer.com/article/10.1007/s11831-017-9247-y

[151] T. Wolfson, J. Berant, and D. Deutch, "Weakly supervised mapping of natural language to sql through question decomposition," arXiv:2112.06311 [cs], 12 2021. [Online]. Available: https://arxiv.org/abs/2112.06311

[152] G. Xu, Z. Song, Z. Sun, C. Ku, Z. Yang, C. Liu, S. Wang, J. Ma, and W. Xu, "Camel: A weakly supervised learning framework for histopathology image segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10 2019. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Xu_CAMEL_A_Weakly_Supervised_Learning_Framework_for_Histopathology_Image_Segmentation_ICCV_2019_paper.html

[153] W. Zhu, Q. Lou, Y. S. Vang, and X. Xie, "Deep multi-instance networks with sparse label assignment for whole mammogram classification," in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2017*. Springer International Publishing, 2017, pp. 603–611. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-66179-7_69

[154] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2016, pp. 2921–2929. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Zhou_Learning_Deep_Features_CVPR_2016_paper.html

[155] H. Kang, H.-m. Park, Y. Ahn, A. Van Messem, and W. De Neve, "Towards a quantitative analysis of class activation mapping for deep learning-based computer-aided diagnosis," in *Medical Imaging 2021: Image Perception, Observer Performance, and Technology Assessment*, F. W. Samuelson and S. Taylor-Phillips, Eds., vol. 11599. SPIE, 02 2021, p. 13. [Online]. Available: https://biblio.ugent.be/publication/8694842

[156] 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 *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 10 2017, pp. 618–626. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.html

[157] M. Acharya, K. Kafle, and C. Kanan, "Tallyqa: Answering complex counting questions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8076–8084, 07 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4815

[158] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 03 2018, pp. 839–847. [Online]. Available: https://ieeexplore.ieee.org/document/8354201

[159] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 06 2020, pp. 24–25. [Online]. Available: https://openaccess.thecvf.com/content_CVPRW_2020/html/w1/Wang_Score-CAM_Score-Weighted_Visual_Explanations_for_Convolutional_Neural_Networks_CVPRW_2020_paper.html

[160] C. Fellbaume and G. Miller, *WordNet: An Electronic Lexical Database*. MIT Press, 1998. [Online]. Available: https://ieeexplore.ieee.org/book/6267389

[161] D. Kunkle and J. Schindler, "A load balancing framework for clustered storage systems," in *High Performance Computing*. Springer-Verlag, 2008, pp. 57–72. [Online]. Available: https://dl.acm.org/doi/10.5555/1791889.1791900

[162] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, pp. 62–66, 01 1979. [Online]. Available: https://cw.fel.cvut.cz/b201/_media/courses/a6m33bio/otsu.pdf

[163] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, pp. 3668–3681, 08 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8903465

[164] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, and K. Team, "Keras tuner," KerasTuner, 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[165] D. Soekhoe, P. Putten, and A. Plaat, "On the impact of data set size in transfer learning using deep neural networks," in *International Symposium on Intelligent Data Analysis*, 10 2016, pp. 50–60. [Online]. Available: https://www.researchgate.net/publication/313065901_On_the_Impact_of_Data_Set_Size_in_Transfer_Learning_Using_Deep_Neural_Networks