TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Joanna Rose Castillon del Mar 177278IVCM

Automated Photo Categorization for Digital Forensic Analysis Using a Machine Learning-Based Classifier

Master's Thesis

Supervisors: Hayretdin Bahşi, PhD
Center for Digital Forensics and Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Leo Mršić, PhD
AlgebraLAB Director
Algebra University
Zagreb, Croatia

Krešimir Hausknecht, M.Sci.
Head of Digital Forensics Department
INsig2 d.o.o.
Zagreb, Croatia

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia Teaduskond
Tarkvarateaduse Instituut

Joanna Rose Castillon del Mar 177278IVCM

Fotode automatiseeritud liigendamine küberkriminalistiliseks analüüsiks masinõppel põhineva liigendaja abil

Magistritöö

Juhendaja: Hayretdin Bahşi, PhD
Center for Digital Forensics and Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Leo Mršić, PhD
AlgebraLAB Director
Algebra University
Zagreb, Croatia

Krešimir Hausknecht, M.Sci.
Head of Digital Forensics Department
INsig2 d.o.o.

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Joanna Rose Castillon del Mar

13.05.2019

Abstract

Categorizing the content of seized devices for potential evidentiary value, particularly photos, is inherent in a forensic investigation. This use-case can be re-formulated as a machine learning classification problem. The performance of neural networks in solving image classification is unparalleled, some already exceeding human-level accuracy. These state-of-the-art neural networks are built from powerful computer vision models pre-trained in various categories from benchmarked datasets, and freely available online.

Using design science, this paper demonstrates how pre-trained computer vision models can be utilized for developing a tool that is usable, easy to use and effective, addressing a genuine gap in the tools available to forensic examiners. A prototype classifier for gun and not-gun categories for automatic categorization of photos is developed in Flask and Python 3 with a Keras-TensorFlow architecture.

Model selection and evaluation are completed using synthetically-generated realistic datasets. During selection, the InceptionV3 model outperformed Xception, ResNet and VGG16 models, and was used as the final model for the prototype. For evaluation, it is tested against two unbalanced datasets containing only 1% of gun pictures while the web- and console-based prototype is tested for usability, learnability, and effectiveness by five respondents. The usability and learnability attributes, measured using the System Usability Scale (SUS) approach, resulted in a rating of “Acceptable”. Effectivity is measured by comparing the speed of completing tasks between the manual method versus the prototype. Hypothesis testing to determine the significant difference of the tool’s performance was found to be neither worse nor better (no significant difference). However, this could be attributed to the low number of respondents, resulting in low statistical power. Additionally, alternatives to InceptionV3 (Magnet Axiom software and the Xception model) were also investigated, and the results are promising.

This thesis is written in English and is 108 pages long, including 6 chapters, 76 figures, 33 tables, and 15 equations.

List of Abbreviations

AUC	Area Under the Curve
CPU	Central Processing Unit
CSAM	Child Sexual Abuse Material
CSV	Comma-Separated Values
EXIF	Exchangeable Image File Format
FPR	False Positive Rate
GPU	Graphical Processing Unit
MCC	Matthew's Correlation Coefficient
PR curve	Precision-Recall curve

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

In the field of computer vision research, machine learning has gained significant traction in different industries such as medicine, entertainment, and commercial products and services. Similarly, in digital forensics, re-formulation of traditional questions into either a regression or a classification problem is already being studied such as classifying fingerprint quality in forensic analysis [1], object identification in police photographs for evidence recognition [2], and classifying tampered images in digital image forensics [3]. This extension of computer vision to applications in *digital image forensics* (sometimes known as *multimedia forensics*) is a natural progression, attributed to the ubiquitous use of digital pictures and videos, and hence, increasing potential evidentiary value. The need for automation becomes even more apparent with under-staffed police forces with financial constraints and institutions with limited resources.

The idea for this research study came from discussions with forensic examiners in a renowned digital forensics firm during an internship in Croatia. During interactions with the team, it was revealed that although analysing photos is such an integral part of forensic analysis, there is no automated way to categorise the images using non-proprietary tools.

It is estimated that a 64GB iPhone could hold at least 35,000 images based on SanDisk's [4] estimation on the number of JPEG[1] files that can be stored. During a forensic acquisition, thousands of photos are often extracted and analysed for processing. Without access to proprietary tools that provide media classification (e.g. Magnet Forensics' Magnet Axiom), the forensic examiner must manually review these photos by hand to search for *artefacts* - "items of interest that help an investigation move forward" [5]. The trivial task of categorizing photos manually consumes the examiner's time especially when there is a substantial number of devices acquired waiting to be processed. The forensic examiners who were interviewed lamented the lack of open-source tools that could (1) automate categorisation, as well as (2) provide information that would be relevant in a forensic analysis.

[1] The most common file format in consumer cameras

The implementation of neural networks to solve image recognition problems was introduced in 2012 when the first team using a neural network framework called AlexNet [6], won the *ImageNet* competition (ILSVRC) [7] by a significant margin, and the first team to achieve a remarkable error rate of below 25% since the competition started in 2010. *ImageNet*[1] is a dataset of benchmarked high-resolution images with around 22,000 categories. The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)[2] uses only a subset of ImageNet with 1000 categories of around 1000 images in each category. Since AlexNet, the performance of neural networks in solving image classification and detection problems is unparalleled. In 2016, such models already exceeded human-level accuracy [8]. Technology and algorithm development have improved significantly that we see automated classification deployed in applications such as Pinterest [9], Google Photos [10], and Facebook [11] for quite some time now.

To improve the performance of models, rather than going through the process of designing, developing and training a model, the winning entries usually perform a transfer of learned representations approach (transfer learning). That is, a pre-trained computer vision model with pre-learned capabilities from some popular benchmarked datasets like ImageNet, is modified to serve in a different context or answer a different question from what it was originally created for. To illustrate, a base computer vision model trained in different categories (birds, animals, buildings, plants) can be fine-tuned to detect different species of flowers. The overwhelming simplicity and robustness of *transfer learning* allow data science enthusiasts of all levels to achieve remarkable results. The performance of the winning entries can be attributed to the already powerful base models that were pre-trained in many categories, and these models are freely available online.

Then, if these models are indeed powerful, robust and more importantly - freely available, can we use these pre-trained computer vision models to create a tool specifically for digital forensic usage? Can forensic examiners use this tool to aid in their investigations?

[1] http://image-net.org/
[2] http://image-net.org/challenges/LSVRC/

The open-source development of a tool using such *state-of-the-art*[1] neural network-based classifiers to meet the needs of forensic examiners is one of the goals of this research. Therefore, a non-trivial choice for a *category* or a *class* (output response of a classifier) to measure its feasibility in the forensics field must be considered. Consideration is also given to the available classes from which the models had been trained on.

Magnet Forensics' Magnet Axiom[2] automatically categorises the following: Possible weapons, card/ID/paper documents, vehicles(cars/trucks/vans/buses), militants, buildings(exterior), child abuse content, drones/UAVs, money, screen captures, nudity, and drugs. *Weapon classification* is the most practical option because of the availability of benchmarked data available from ImageNet. However, the term "weapons" encompasses a broad array of categories from firearms and explosives to knives. In this prototype, only firearms such as handguns and rifles are considered.

1.1 Research Questions

To address the genuine gap in the tools available to forensic examiners, this research endeavoured to answer the following questions:

- Is it possible to adopt a state-of-the-art pre-trained machine learning model to create an open-source tool to aid forensic examiners in their forensic investigation?

- Using gun classification as a prototype, how does the chosen model perform with *realistic* forensic data?

- How *usable* is the prototype to forensic examiners and potential forensic examiners? Can the prototype be used with ease (learnability)? Once the users learned the design, can they perform the task efficiently (speed of performance)?

[1] The level of development reached at the time of writing this paper that resulted out of common methodologies, research and development of neural networks employed for classification
[2] https://www.magnetforensics.com/

1.2 Open-source in Digital Forensics

In the courtroom, the reliability of the forensic tools used is paramount when presenting results as digital evidence. The historical discussion between proprietary (closed-source) versus open-source is a contentious battle waged from different contexts and circumstances. In the realm of forensics, those advocating for proprietary software raise the issue of security, while those in favour of open-source, cost. When the source of the digital evidence is questioned, software reliability, whether proprietary or open source, takes on an immediate and profound meaning that the extent to which these tools meet the legal requirements surrounding the admissibility of digital evidence must be scrutinised. The dichotomy was comprehensively discussed by Kenneally in [12], in which the article concluded that open-source code had the significant advantage because it could be validated and substantiated. Carrier [13] examined how some open source technology may fare better than their closed-code counterparts in clarity and comprehension.

As such, the drive for this research project is fuelled by the idea of working with open-source technologies that can be easily cross-examined and validated by software developers. Within the developer ecosystem, the Python programming language has seen a remarkable rise in usage since 2014 according to Stack Overflow [14]. Perhaps due to its efficient syntax and readability, extension to web development and data science applications, the enthusiastic adoption of Python within the developer community seems to be the trend in the next few years, and thus, a smart choice for the programming language used in the prototype.

1.3 Data Corpus

The need to acquire real-life data is a challenge in any scientific study. Digital forensics had always suffered from the lack of standardised data suitable for experimental research and even for comparative edification. We were not able to procure a real-life case data of photos extracted from devices acquired from an arms deal, for example. Hence, for evaluation, *realistic* data was used instead. This research has two evaluation parts: (1) computer vision model selection and (2) testing of the prototype and its model. The data corpus used for model selection is simulated by randomly downloading images from the internet, generating thumbnails as well as using an available test dataset[1] from Olmos, Tabik and Herrera's paper: *Automatic Handgun Detection Alarm in Videos* [15]. The data corpus used for testing the prototype and its core model is laboratory-generated from a mobile phone dump, a hard-disk dump, and downloaded images with manufactured EXIF information.

Due diligence had been taken to ensure that images for selection and testing had not been downloaded[2] from ImageNet since the computer vision models were pre-trained from the ImageNet dataset. Using even one training image as part of the test set is tantamount to cheating as the performance of a model should be evaluated on data that the model has never seen before.

1.4 The Prototype

The prototype's user interface is inspired by Google Photos' content-based image searching functionality, and most forensic software's information and GPS coordinates display. It uses Python with a Keras-based neural network open-source library running on a TensorFlow abstraction backend. It runs from the console to predict the output classes for each image in the directory while a web interface is provided for parsing and visualising the results. The source code called *cbis* (content-based image search), installation and documentation guide and sample datasets are fully available in GitHub [16] [17].

1.4.1 The Pre-trained Neural Network

For prediction, different *state-of-the-art* computer vision models pre-trained in ImageNet were evaluated: InceptionV3 [18], Xception [19], ResNet [20] and VGG16 [21]. The InceptionV3 model gave the most promising results based on the model selection criteria and was, therefore, used as the basis for the prototype.

[1] https://sci2s.ugr.es/weapons-detection
[2] http://image-net.org/download.php

1.5 Usability, Learnability and Efficiency

In a survey on digital forensic tools [22], the paper concluded how the existing tools investigated[1] have usability issues such as the lack of intuitive interfaces, the lack of user-friendliness in reporting and graphics, and a limited collaborative environment among other forensic tools.

In the context of computer forensic analysis tools, usability is understood to be "a characteristic of the interaction between the forensic investigator and the computer system they are utilising to effect an investigation" [23]. ISO 9241-11:2018 defined usability as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [24] . Nielsen [25] viewed usability as a multi-faceted acceptability of a system, tool or device. According to him, usable systems should be easy to learn (Learnability), efficient (Efficiency), easy to remember (Memorability), free from errors (Errors) and results in a pleasant experience for users (Satisfaction). However, he stated that improving all attributes is only ideal as they are non-orthogonal - improving one attribute such as learnability might adversely affect efficiency.

The research study employs Nielsen's view on usability by focusing measurement on the user's perceived usability, learnability, and efficiency in operating the prototype to complete a task. Learnability is how well a new user can learn the system while efficiency measures the effort of completing a required task [25]. In this study, usability and learnability were measured using the System Usability Scale (SUS) approach, while efficiency was measured by the speed in which tasks were completed manually and through the prototype. Tasks were designed to elicit a semblance of interaction between the forensic examiner (user) and the prototype (tool) in solving an investigation, in line with the definition of usability in [23].

1.6 Hypothesis and Contributions

This research addresses a genuine and important gap in the tools available to forensic examiners in the performance of their forensic activities. This research hypothesizes that

[1] The tools used in the survey were FTK, EnCase, Autopsy, Sleuth Kit

an open-source tool can be created based on a pre-trained machine learning-based classifier and that this resulting tool can be deployed to aid real-life forensic investigations.

As technology advances, the need for content classification in forensics becomes evident with the growing amount of raw data that require processing during an investigation. Proprietary tools incur huge licensing costs. Cellebrite Analytics, for example, costs around €5,000 – €15,000 for the first year with a discounted rate for subsequent updates[1]. While user-friendly, convenient and reliable, the functionalities are also available from some open-source options and are not less reliable and less effective than their proprietary counterparts [26]. The resulting prototype from this research project provides an open-source option written in Python, creating a viable way for forensic examiners to categorize through a vast number of digital photos from a forensic acquisition.

This paper demonstrates the approach using Design Science methodology using both balanced and unbalanced datasets. Balanced dataset has an equal or nearly equal number of positive (guns) and negative (not guns) classifications. The unbalanced dataset is the attempt to simulate dataset distribution occurring in most forensic acquisitions where the ratio of relevant pictures against the non-relevant pictures is meager. This research provides insight into the performance of pre-trained classifiers against unbalanced datasets.

1.7 Importance and Applicability

The contribution of this paper is both immediate and applicable to other situations, even beyond forensics. The usefulness of machine learning in many different aspects of practical life can be seen in the way we make use of tools and applications. Technology and the community have allowed a data-intensive approach to thrive and its prevalence raised the bar on how we expect things to work. The growing maturity and the avid support of developers and organisations surrounding the field of computer vision and machine learning make it easier for data science enthusiasts to transition from theoretical

[1] Forensic experts were asked about the cost of tools because prices are not published online

knowledge of machine learning architectures to implementation of applications, tools or functionalities that are relevant in their specific fields.

In digital forensics, content-based searches have unlimited potential applications. Searching for content can be expanded to searching for similar photos from other cases (cross-case relation), evidence detection based on photos and videos, determining make and model of specific objects such as firearms and explosives, extracting and determining content from screenshots, triaging based on photos and video categorization, and many more. Such applications save resources – time, cost and human resources. The tedious task of categorization is minimised or even eliminated - allowing the examiner to focus on essential investigative tasks. Aside from resources, there is also potential to improve the mental health of examiners. In a child exploitation case, for instance, the potential to fully automate the search and detection of child sexual abuse material (CSAM) has a tremendous impact on the general psyche of the examiner tasked to review such images and videos.

One of the implications of automated classification of objects leveraging powerful pre-trained models is the development of time- and cost-efficient tools without the need for modelling new neural networks. Employing this transfer learning approach saves development and training time for models, cutting short the time-to-deploy process. The possibilities are exciting as architecture platforms, such as TensorFlow [27], continually evolve to create easy-to-use, powerful, and robust backend structures with flexible, comprehensive, cross-platform and freely-available ecosystem of tools and libraries.

This research hopes to inspire the development of similar machine learning-based tools with practical applications, in this case, in forensic investigations.

1.8 Novelty of the Study

Although content-based image classification is not particularly novel, nor is gun classification, this research focuses on the open-source development of pre-trained classifiers evaluated to meet the requirements of forensic examiners systematically. The development of a prototype that showcases the integration of machine learning with a real-world tool that can be validated and even modified to suit different needs addresses a genuine gap in the tools available to forensic examiners. Hence, this paper is a first foray into the digital forensics domain that attempts to develop a tool for content image search based on a pre-trained classifier using the Design Science methodology.

As of our present knowledge, there is no academic literature that performs image classification with specific applications in digital forensics. Moreover, limited academic works on the *Design Science* methodology were found, and interestingly, none that tackles both digital forensics and machine learning. The Related Works section identifies some of the papers that (1) answer a classification problem using machine learning in the digital forensics' domain, and (2) perform image classification for handgun images and videos.

2 Background of the Study

2.1 Machine Learning Essentials

2.1.1 Variables and Data

In machine learning notation, the inputs, typically denoted as X, are often called *observations*, *predictors*, *independent variables*, *features, attributes* or *variables*. The output, denoted as Y, is sometimes called the dependent variable or the response. In classification problems, qualitative predictors are also called *factors*; the possible values for these factors are also known as *levels*. *Quantitative* values take on a *numerical* value such as age, price and income. *Qualitative* values take on descriptive values also known as *labels*, *classes* or *categories* such as gender, brand of product purchased, or a cancer diagnosis. Data are characterized as either *continuous* or *discrete*, *nominal* or *ordinal*. *Discrete* data are countable such as the number of bedrooms in a house. *Continuous* data are non-countable, such as age and time. *Nominal* data are named values which take no order or hierarchy, such as marital status (single, married, divorced, widowed). Data that only take two values such as gender (male or female) are called *dichotomous*. *Ordinal* data takes an order or hierarchy based on a position, for example, temperature on a scale or size of a house in square meters (small, medium, large).

In this research, the predictors are the image parameters and features while the output is the class or object labels.

2.1.2 Supervised versus Unsupervised

The two major categories of machine learning approaches are: Supervised and Unsupervised. Other categories such as Active Learning and Reinforced Learning are just some forms and variations of supervised and unsupervised approaches.

Supervised learning means that for every observation or input, there is an associated response. The goal of supervised learning is to *fit a model* that relates the labelled data instances to its response (which is known *a-priori* during training), with the aim of (1) accurately predicting the response for future and unseen data, or (2) a better understanding of the relationship between the input and response. In machine learning, *fitting a model* means choosing the best and most generalised model that predicts the responses or infer relationships that are as close as possible to the input observations. When there are more observations or data from which the model learns from, the model becomes better at generalisation. An *underfit* model has poor accuracy in both training and test data while an *overfit* model has the best accuracy in training but performs worse in test. A general model (neither underfitting nor overfitting) performs well in both training and test data. The best model, therefore, is expected to perform well with unseen and future data which is the objective of all machine learning solutions.

In contrast, unsupervised learning offers no associated output response during training. The lack of response to supervise the prediction or inference makes this approach and its corresponding analysis, challenging.

This research paper uses the traditional supervised approach. It brings its own challenges such as the accurate labelling of ground truth data by a domain expert. In this case, ground truth labelling is crucial to the process of model selection because the evaluation of the performance metrics occurs on the model's predictive capacity versus the correctly represented labels of the images. In this research, the ground truth labeller is the researcher in which a gun is classified as an actual gun when it is detectable to the human eye.

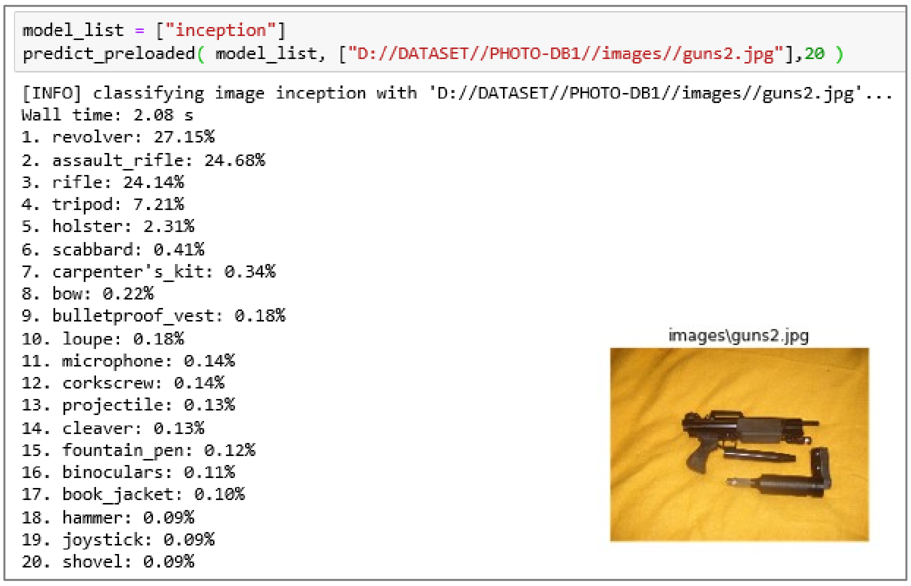
2.1.3 Regression vs Classification

Machine learning problems can be formulated as either a regression or a classification problem. A regression problem typically involves quantitative or continuous dataset as an output, such as predicting the house selling price based on features like location and floor area. A classification problem, on the other hand, is used for responses expecting qualitative or categorical values such as determining if today's stock price goes up or down. A model that attempts to answer a classification problem is called a *classifier*.

However, this distinction is not almost always clear. Classification models may behave like regression models because they first predict the probability of each category, that is, the output is a bunch of quantitative values, as the basis of making a classification. This research paper follows this approach. The neural network used in this research behaves as a regression model, but the resulting prototype is a classifier.

Binary classifiers are classifiers that produce only two output responses (e.g. yes or no, gun or not gun). Multi-class classifiers produce more than two categorical outputs.

In this paper, the pre-trained neural network produces multiple labels with a corresponding probability for each response class. In Figure 1, the InceptionV3 model predicts the first 20 possible labels for the *guns2.jpg* image. Additionally, the web application creates a visualisation that outputs the binary responses of both gun and non-gun labels as seen in Figure 2.



```
model_list = ["inception"]
predict_preloaded( model_list, ["D://DATASET//PHOTO-DB1//images//guns2.jpg"],20 )
```

```
[INFO] classifying image inception with 'D://DATASET//PHOTO-DB1//images//guns2.jpg'...
Wall time: 2.08 s
1. revolver: 27.15%
2. assault_rifle: 24.68%
3. rifle: 24.14%
4. tripod: 7.21%
5. holster: 2.31%
6. scabbard: 0.41%
7. carpenter's_kit: 0.34%
8. bow: 0.22%
9. bulletproof_vest: 0.18%
10. loupe: 0.18%
11. microphone: 0.14%
12. corkscrew: 0.14%
13. projectile: 0.13%
14. cleaver: 0.13%
15. fountain_pen: 0.12%
16. binoculars: 0.11%
17. book_jacket: 0.10%
18. hammer: 0.09%
19. joystick: 0.09%
20. shovel: 0.09%
```

Figure 1 Input Image and Multi-class Output Responses (Predictions)

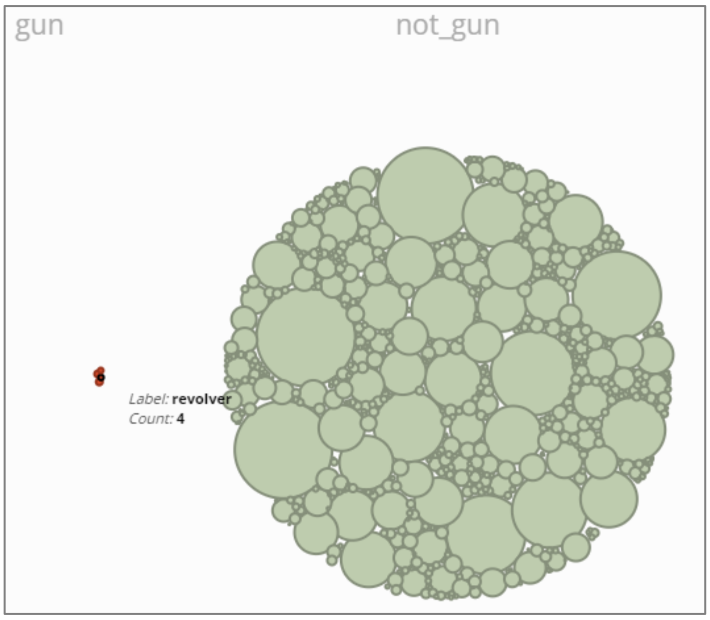


Figure 2 Prototype's Classification for Guns and Not Gun labels

2.1.4 Classification, Detection and Recognition

In the field of computer vision, the terms *classification*, *detection* and *recognition* are often encountered. It is important to establish the distinction between terms in this paper.

To have a standardized definition, this paper borrowed the definitions from Russakovsky *et al.*'s paper [28]. *Image (object) classification* is the identification of object classes, for example, a person, and determining whether the image contains an instance of that class. *Object detection* is the localization or identification of the location of learned objects in the image. For example, detection implies determining the bounding box coordinates within the image to identify where the person is. Figure 3 is taken from [29] and clearly shows the distinction between classification and detection algorithms and the expected outcomes from the same image. *Object recognition* is a term that encompasses both classification and detection tasks.

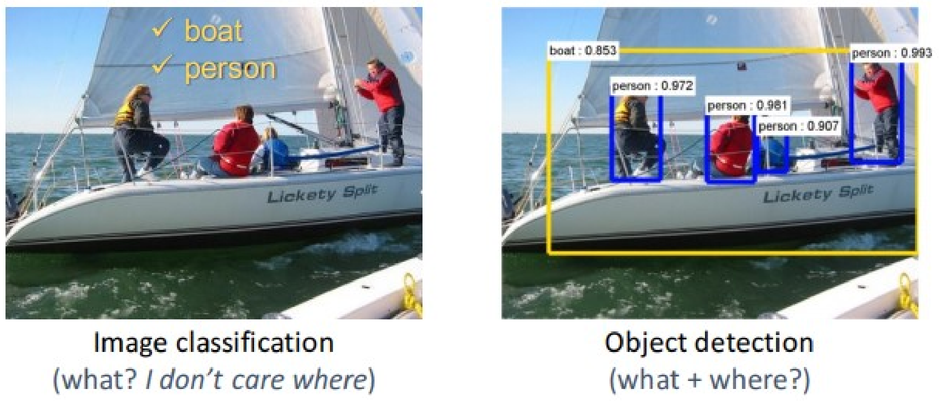


Figure 3 Classification versus Detection from [29]

2.1.5 Models

Machine learning algorithms are typically chosen based on the type of problem: regression or classification. For example, predictions expecting quantitative responses typically employ regression-based algorithms while predictions involving label or categories use classifier-based techniques.

Again, this distinction is not always clear. The Logistic Regression model, despite its name, is used as a classification method. Other models such as K-Nearest Neighbour (KNN) and Decision Trees, can be used in both quantitative or qualitative responses.

In solving image classification problems, a traditional machine learning pipeline involves two modules: feature extraction and classification. Feature extraction is the retrieval of higher-level information from raw pixel values that capture the distinction or differences between the categories in the training set. Such features may be regions, edges, corners or areas. Examples of feature detectors include Histogram of Oriented Gradients (HoG), Difference of Gaussians (DoG), and Speeded Up Robust Features (SURF). After the features are extracted, the classification module is trained with the extracted features and their corresponding labels. Traditional classification models using feature extraction include Logistic Regression, Decision Trees or Support Vector Machine (SVM).

The main drawback of traditional image classification is in the feature extraction module. It is challenging to achieve a level of generalisation of features that can represent all the categories with an accuracy that is close to human level.

With neural networks, there is no hard-coded feature extractor built in the model. The network is the combination of a feature extractor and a classifier; the network learns to discriminate the representations of the images (feature extractor) and identify them based on supervised data (classifier).

As shown in Chapter 3.1, the best performing models evaluated for digital forensic triage classifiers are shown to be traditional machine learning techniques such as Bayesian Networks (Bayes Net and Naïve Bayes), Decision Trees and KNN. Chapter 3.2, on the other hand, shows Neural Networks dominating the gun classifier category.

2.2 Neural Networks

One of the most popular algorithms in machine learning, neural networks take centre stage in computer vision research. Inspired by the human brain’s architecture, the basic building block of a neural net is a *neuron*, which takes some input and fires some form of output. A neuron is a mathematical function termed as an *activation* function. There are five major activation functions used today: step, sigmoid, tanh and ReLu (Rectified Linear Unit). These functions introduce non-linearity in the modelling capabilities of the network. The collection of these mathematical functions (neurons) is called a *layer*.

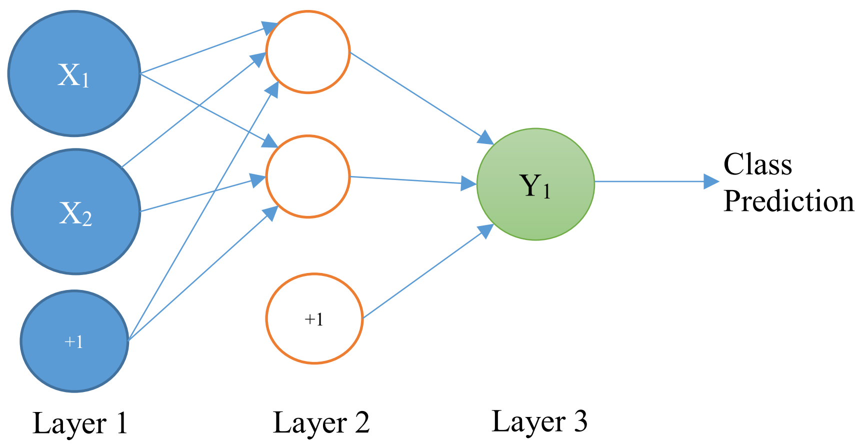


Figure 4 A Regular Neural Network

Figure 4 shows a regular neural network. Nodes are also called units and layers are the collection of units. A *bias* unit is a node that is not affected by the other nodes but directly influences the output. Bias can be compared to the "y-intercept" in the linear expression for a line y = mx + b, where *m* is the *slope* and *b* is the *y-intercept*. Layer 1 is called the input layer with 2 input units X_1 and X_2 (does not include the bias unit, +1). Layer 3 is the output unit with only 1 node (one node for each class prediction). The middle layer is the hidden layer because the values are not transparent during training. All neural networks have one input and one output layer. The number of hidden layers is dependent on the complexity of the problem.

During training, each neuron learns *weights* at every layer using *forward* and *backward* propagation based on some metric such as a loss function. Weights are just the coefficients assigned to each node representing the strength of connection of each neuron. The weights determine how much each node influences the output. The ideal is to find the weights where the loss (or error) is minimum. *Gradient descent* is another technique in which the weights are adjusted in small increments (learning rates) by the calculation of the derivative (gradient) of the loss function. The derivative tells the algorithm in which direction to *descend* to reach the global minimum (where the error is at minimum). In practice, *Stochastic Gradient Descent (SGD)* is widely used as this is done in several batches of shuffled data in successive iterations (epochs).

2.2.1 Convolutional Neural Network

In a regular neural network, every layer is made up of a set of neurons where every node is fully connected to all the neurons before it (except for the bias nodes). The last fully-connected layer is the output layer that represents the predictions or class outputs. On the other hand, a *convolutional* neural network does not have all neurons connected to the layer before and after it. The hidden layers typically consist of convolutional (CONV) layers, pooling (POOL) layers and fully-connected (FC) layers.

Convolution is a mathematical operation on two functions (or images) to produce a third function that expresses how the shape of one modifies the other. This is essentially a dot product of two functions. In image processing, the input image is one function while the other function (called a kernel) is a filter that passes over the image function and changes the value of each pixel as it passes by.

Pooling is a sample-based discretization process, that is, it reduces the spatial dimensionality of the input image. Down-sampling using this method reduces the parameters, thus reducing computational resources while controlling the risk of overfitting the data.

Taken from [30], the interaction between the input image matrix and the output matrix in the convolution layer is shown in Figure 5 while the down-sampling in the pooling layer is shown in Figure 6. These layers (convolution and pooling) perform operations to automatically detect features (feature extraction), for example, detecting the barrel, trigger and the grip for a gun image. The fully connected layers serve as the classifier that assigns a probability for the object on the image.

Taken from [31], a sample of the convolutional neural network architecture is shown in Figure 7. Here, the input image of a car is subjected to a network of convolutions and pooling layers including ReLu activations, while the output classifier is represented by the fully connected layer that displays the predicted classes (car, truck, airport, etc) and the probabilities.

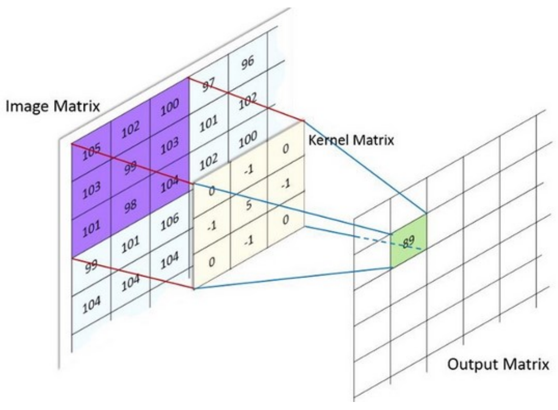


Figure 5 Convolution Layer from [30]

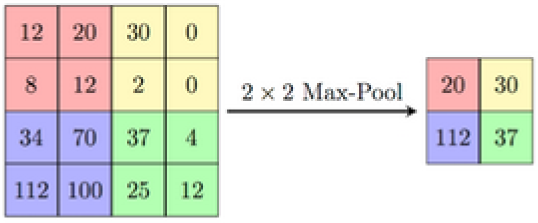


Figure 6 Pooling Layer from [30]

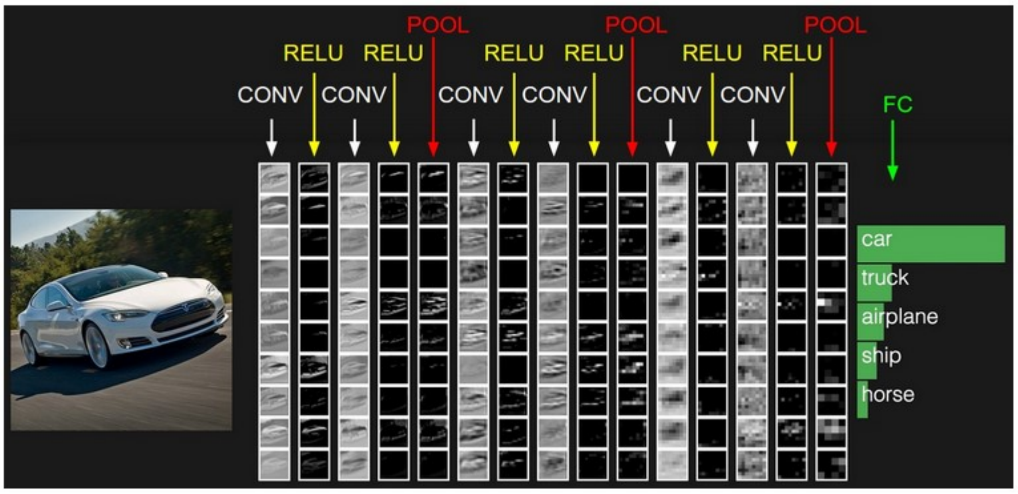


Figure 7 A Sample Convolutional Neural Network from [31]

2.3 Machine Learning in Digital Forensics

Machine learning has been proposed on areas of digital forensics such as email forensics, network forensics, data analysis and file fragment classification. In email forensics, for example, authorship identification is a forthcoming field of interest especially for tracing identity in digital investigations. This research paper focuses on digital image forensics. The research potential and interest in this field had long been discussed [32] [33] and research is ongoing with Adobe joining the fray [34].

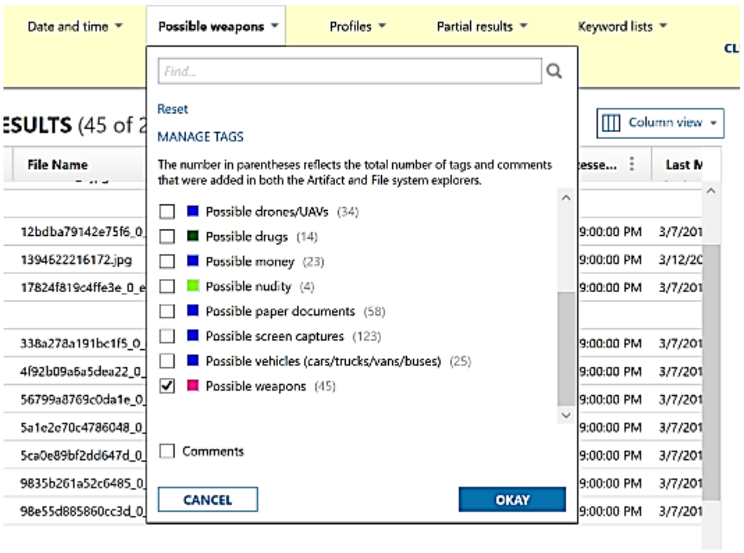
2.3.1 Photos as Digital Evidence

Part of the forensic investigation is the systematic search for evidence. Guidelines such as ACPO 's Good Practice Guide [35], and DOJ's Seize and Seizure Manual [36] advocate the formulation of a forensic or search strategy to speed up the investigation by focusing on relevant information. The same guidelines stressed the need to consider the nature and purpose of the investigation in which an initial triage or review of digital evidence to identify priority might be necessary. DOJ's Guide to First Responders [37] classifies pictures as a data source with potential evidential value for certain categories of crime such as child exploitation and harassment. In practice, the identification of the types of potential evidence based on the nature of the investigation is one of the considerations for formulating a forensic strategy.

To illustrate, in a child exploitation case, media files (Child Sexual Abuse Material or CSAM) are one of the first sources of data that are extracted and analysed. Although provenance[1] of digital evidence is equally crucial in the investigation [33], identifying the existence of certain types of graphic content is primary in determining interesting artefacts for further analysis. This use-case can be re-formulated as a machine learning question, specifically a classification problem.

[1] Record or chronology of ownership and origin, custody or location of digital evidence

Analysing the content of seized devices such as photos is inherent in a forensic investigation that almost all digital forensic toolkits have some form of photo analysis functionality. Aside from photo carving which is fundamental in computer forensics, recent versions of forensic tools have included advance analysis software such as explicit content detection and media classification with machine learning integration.



The number in parentheses reflects the total number of tags and comments that were added in both the Artifact and File system explorers.

- [ ] 🟦 Possible drones/UAVs (34)
- [ ] 🟩 Possible drugs (14)
- [ ] 🟦 Possible money (23)
- [ ] 🟩 Possible nudity (4)
- [ ] 🟦 Possible paper documents (58)
- [ ] 🟦 Possible screen captures (123)
- [ ] 🟦 Possible vehicles (cars/trucks/vans/buses) (25)
- [✓] 🟥 Possible weapons (45)
- [ ] Comments

File Name	esse...	Last M
12bdba79142e75f6_0	9:00:00 PM	3/7/201
1394622216172.jpg	9:00:00 PM	3/12/20
17824f819c4ffe3e_0_e	9:00:00 PM	3/7/201
338a278a191bc1f5_0	9:00:00 PM	3/7/201
4f92b09a6a5dea22_0	9:00:00 PM	3/7/201
56799a8769c0da1e_0	9:00:00 PM	3/7/201
5a1e2e70c4786048_0	9:00:00 PM	3/7/201
5ca0e89bf2dd647d_0	9:00:00 PM	3/7/201
9835b261a52c6485_0	9:00:00 PM	3/7/201
98e55d885860cc3d_0	9:00:00 PM	3/7/201

Figure 8 Magnet Axiom's Photo Categorisation - Weapon Tagging

Figure 8 shows the machine learning-based classification functionality of Magnet Axiom from Magnet Forensics[1]. During processing of the acquired device, each photo or video detected by the software is tagged with possible categories: drones, drugs, money, nudity, weapons, etc. The examiner can then search for images and categorise them based on these tags. A CSV report based on the search results can also be retrieved.

[1] https://www.magnetforensics.com/

3 Related Works

The increasing amount of data that needs to be processed has outpaced the effectiveness of traditional digital forensic methods and the desire for automated methods of detecting and classifying content is a natural step further. Research on machine learning and deep learning (neural networks) techniques are prevalent, extending far beyond image classification. However, limited works can be found dealing with image classification for use in the digital forensics' domain. The classification for our choice of the output class, gun, had had extensive research but these works were formulated to address data science questions such as architecture evaluation and model performance.

This section divides these related works into two parts: (1) classifiers used in Digital Forensic Tools, and (2) hand-gun classifiers. Dataset generation, evaluation and performance metrics used during the evaluation of the models are of particular importance for this research study.

3.1 Works on Classifiers in Digital Forensic Tools

2009, Grillo *et al.*, *Fast user classifying to establish forensic analysis priorities* [38], proposed a methodology called Five-Minute-Forensics (5MF) and a classifier to support computer user profiling as a triage tool. The authors generated the dataset themselves, extracted and processed using SleuthKit[1]. For the classifier validation, they used ten-fold cross validation test on 25 feature vectors, five per user category (occasional user, chat-internet user, office worker user, experienced user, hacker user). The study experimented with different models: Bayesian algorithms BayesNet and Naïve Bayes, and Decision Tree (J48) with preliminary results for accuracy ratings of 100%, 92%, and 84%, accordingly.

2010, Garfinkle *et al.*, *An Automated Solution to the Multiuser Carved Data Ascription Problem* [39], proposed a machine learning-based approach to ownership attribution for carved information on disk drives and storage media. The tool *fiwalk*[2] (file and inode

[1] http://www.sleuthkit.org/autopsy/
[2] http://www.forensicswiki.org/wiki/Fiwalk

walker) had already been integrated into SleuthKit. The classifier is auditable and appropriate for use in court cases as the accuracy can be verified using cross-validation and manual inspection. Reliability (error rate) specific for the disk on which it was run is also reported. The datasets came from realistic (laboratory-generated data) and real data (acquired from the secondary computers from second-hand markets). For validation, they used leave-one-out and ten-fold validation. Accuracy rate was used as a performance metric with the Decision Tree (C.45) classifier achieving 96.47% against K-NN ($k = 1$) at 75.45%.

2011, Marturana *et al.*, *A Quantitative Approach to Triaging in Mobile Forensics* [40], worked on a study dedicated to the evaluation of triage mobile data in paedophilia cases. The study evaluated BayesNet, WEKA Decision Tree (J48) and Locally Weighted Learning (LWL) using three performance indictors: precision, recall and f-measure. BayesNet and J48 achieved 100% on all metrics. Note that accuracy is not an evaluation metric in this study.

2012, Gomez, *Triage in-Lab: case backlog reduction with forensic digital profiling* [41], presented a triage model using an automated predictive classifier focused on child exploitation and intellectual property theft. The classifiers C.45, Naïve Bayes, K-NN, SVM and Classification Tree were evaluated against classification accuracy and its area under the ROC curve (AUC). K-NN achieved the best accuracy at 90% with an AUC of 94.44%.

2013, Marturana and Tacconi, *A Machine Learning-based Triage methodology for automated categorization of digital media* [42], presented a model for both live and post-mortem mobile device investigation for copyright infringement and child exploitation cases. The dataset was taken from real-life investigations. The following metrics were calculated for the infringement dataset: accuracy, mean absolute error, root mean squared error, precision, recall and f-measure while only precision, recall and f-measure were used for the exploitation set. Different classifiers outperform others depending on the scenario. BayesNet outperformed other classifiers: SVM, decision tree, LWL (Locally Weighted Learning), by achieving a weighted accuracy rating of 99%, 93.5%, 89.5% and 78.5% accordingly for the copyright infringement dataset. Decision Tree outperforms BayesNet, and LWL with 68%, 64.44%, 55.3% for precision, 64.4%, 63.2%, 57.9% for recall, and 64.4%, 63.6% and 56% for f-measure, respectively.

The methods discussed relied on crime-related templates, implying that examiners had to know *a-priori* what artefacts to search for as part of a forensic search strategy (ACPO Guide [35] and DOJ's Seize and Seizure Manual [36]). To improve accuracy and reduce mis-triage incidents, Marturana and McClelland built on the former work of Marturana and Tacconi [42] by introducing the feature manipulation approach to address the generality recommended in methodologies dealing with incident response. *A Digital Forensics Triage methodology based on feature manipulation techniques* [43] aimed to build a comprehensive Digital Forensic Triage methodology using automatic weighing for classification of mobile devices. The ground truth weighted features were generated manually from forensic experts using survey and interview. Data is taken from real-life investigations from mobile devices and hard-drives. Only the child exploitation dataset exhibited improved performance of 19.2% from the baseline (Naïve Bayes) of 55.6% accuracy rate.

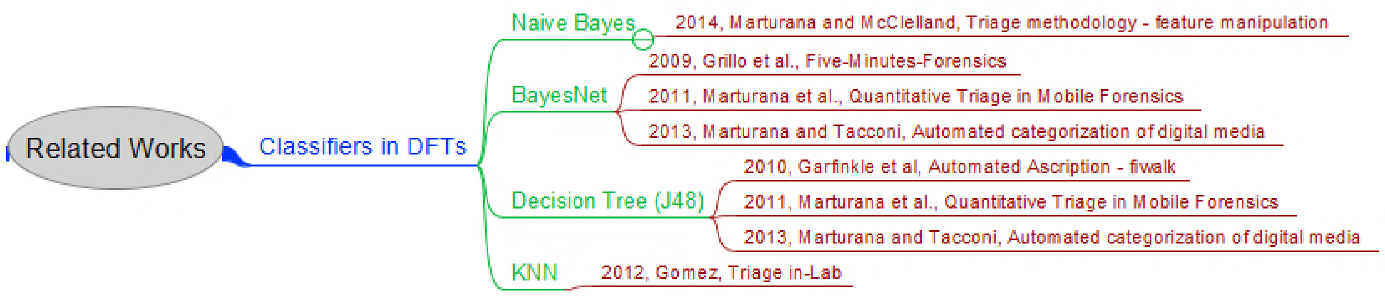

Related Works
Classifiers in DFTs
Naive Bayes
2014, Marturana and McClelland, Triage methodology - feature manipulation
BayesNet
2009, Grillo et al., Five-Minutes-Forensics
2011, Marturana et al., Quantitative Triage in Mobile Forensics
2013, Marturana and Tacconi, Automated categorization of digital media
Decision Tree (J48)
2010, Garfinkle et al, Automated Ascription - fiwalk
2011, Marturana et al., Quantitative Triage in Mobile Forensics
2013, Marturana and Tacconi, Automated categorization of digital media
KNN
2012, Gomez, Triage in-Lab

Figure 9 Related works – Digital Forensic Tools

Figure 9 summarises the different digital forensic tools mentioned in this section. The diagram shows that out of six papers, there are four best models selected after evaluation: Naïve Bayes, BayesNet, Decision Tree (J48) and KNN. This tells us that traditional machine learning techniques were demonstrably applicable for some classification tasks in forensics.

3.2 Hand-gun Classifiers

2013, Lach and Sieradzki, *Automated Recognition of Firearms in Surveillance Video* [44], used sensitivity and specificity as metrics. Classification accuracy was not disclosed.

2015, Tiwari and Verma, *A Computer Vision based Framework for Visual Gun Detection using SURF* [45], was based on true positive rate and false positive rate metrics. The proposed model claimed to be rotation, scale and shape invariant.

2017, Verma and Dhillon, *A Handheld Gun Detection using Faster R-CNN Deep Learning* [46], dealt with gun classification taken from the Internet Movie Firearms Database using Faster R-CNN. The classifier performance was based on accuracy, true positive rate and false positive rate. Accuracy was 93.1% for the Faster R-CNN while 92.6% was achieved for the SVM model.

2017, Olmos, Tabik and Herrera, *Automatic Handgun Detection Alarm in Videos* [15], aimed on minimizing the number of false positives as well as reaching near real-time detection. In this paper, the measures used to evaluate the classifiers were accuracy and speed. True positive was evaluated based on a 50% bounding box threshold. Ground truth was based on human eye recognition. Classification accuracy was 90.6% while Recall is at 100%.

The researchers of [15] were contacted regarding the dataset used to test and achieve the metrics specified in the study. The availability of this dataset[1] made comparative edification possible – that is, a sound comparison can be conducted in the context of binary gun classification. See the section on Model Selection for a detailed explanation on how the dataset was utilized in this study.

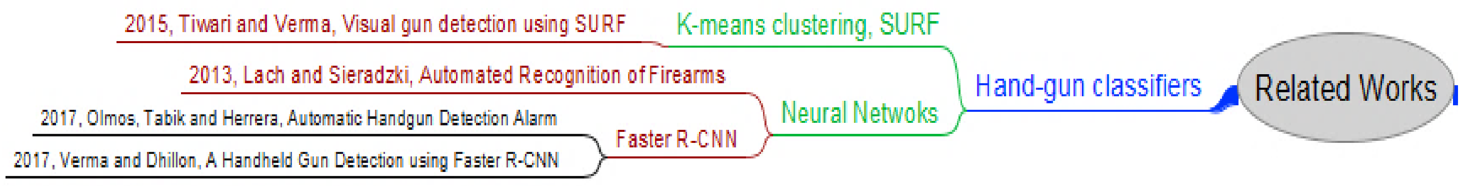

2015, Tiwari and Verma, Visual gun detection using SURF
K-means clustering, SURF
2013, Lach and Sieradzki, Automated Recognition of Firearms
2017, Olmos, Tabik and Herrera, Automatic Handgun Detection Alarm
2017, Verma and Dhillon, A Handheld Gun Detection using Faster R-CNN
Faster R-CNN
Neural Netwoks
Hand-gun classifiers
Related Works

Figure 10 Related Works - Gun Classifiers

Figure 10 summarises papers related to gun classification. It is apparent that neural networks are overwhelmingly employed in image classification: 3 out of 4 papers use neural nets to perform gun classification.

[1] https://sci2s.ugr.es/weapons-detection#testset

3.3 Human classifier

The level of human classifier accuracy is included in this section to understand just how far state-of-the-art classifiers have come in creating and developing architecture close or even surpassing this baseline.

The highly influential paper by Russakovsky *et al*. [28] did not only describe the creation of ImageNet's benchmarked dataset but also the measurement of the accuracy of a human person in classifying multiple objects. The trained human annotator - Andrej Karpathy [47], achieved 5.1% error rate, outperforming GoogLeNet (the precursor of the Inception architecture) at that time by 1.7%, a statistically significant difference.

The exhaustive AI Progress Measurement project [48], a collaborative and on-going effort, summarizes the state of the accuracy of classifier algorithms in relation to the human-level annotator mentioned in [28]. Figure 11, taken from [48] , tells us that in the ImageNet validation datasets, machine learning algorithms have already exceeded human-level accuracy since 2016.

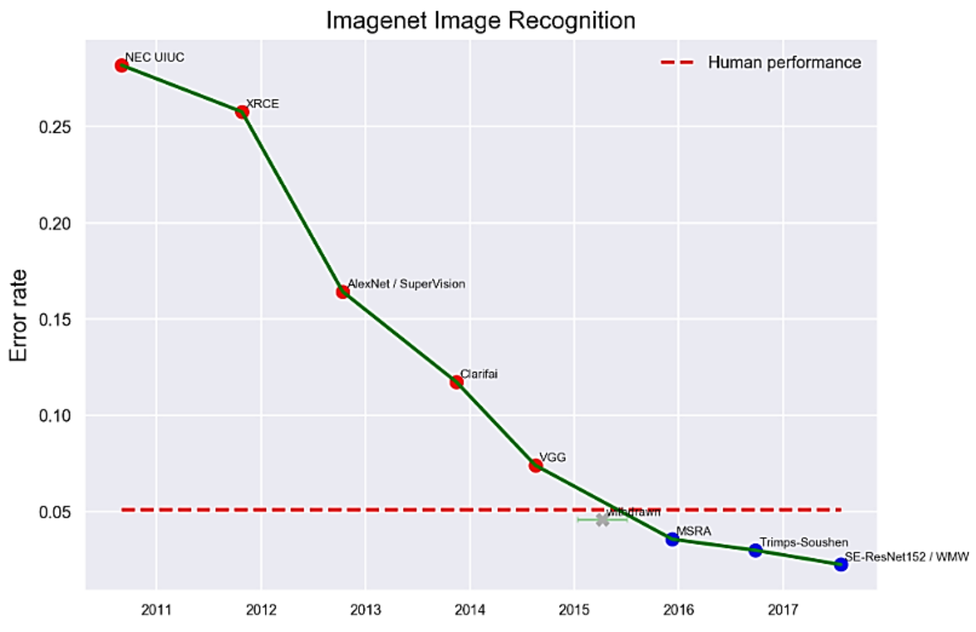


Figure 11 ImageNet Performance, taken from [48]

3.4 Summary of Related Works

Surpassing the accuracy of human-level classification is undoubtedly one of the milestones of machine learning algorithm development. The visualisation in Figure 12 summarises the performance metrics of the preceding related literature in the context of the accuracy achieved by the human annotator (dashed line). The green markers indicate the classifiers with published accuracy results exceeding the human-level classifier; the red ones, otherwise.

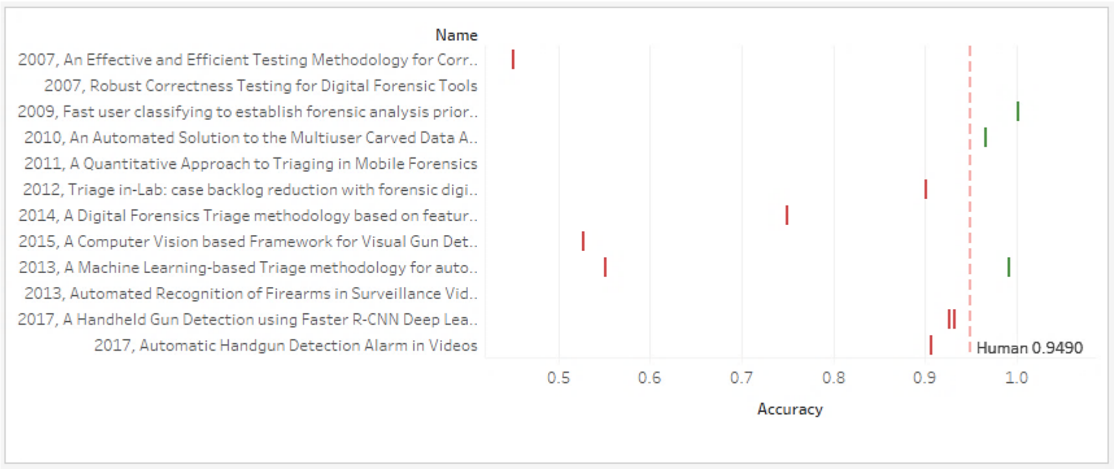


Figure 12 Related Works - Performance Metrics (Accuracy)

Figure 13 shows Precision and Sensitivity values for the same related works. No human-level precision and recall could be accounted for because the paper [28] did not disclose the confusion matrix for the human classifier. The arithmetic mean (solid dark line) is

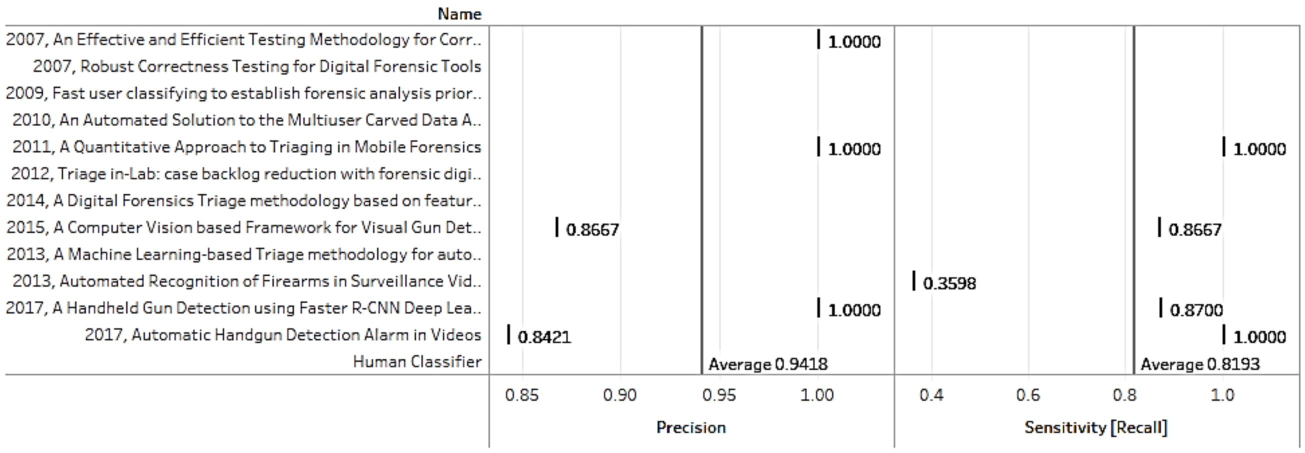


Figure 13 Related Works - Performance Metrics (Precision and Recall)

used to summarize the data. The choice of arithmetic mean as a central tendency to describe the results is arbitrary.

The models, dataset and metrics used by each paper and presented in this section are not intended to suggest a general error rate or create baseline out of the findings but to understand the level of accuracy achieved from these studies in the context of the following factors:

- the dataset used (realistic or real dataset)
- the size of the dataset used for training, validation, test data
- the hyperparameters used to achieve the “best” results (*e.g.* the K-value that gives the optimum performance)
- the type of errors that may be introduced and the biases introduced during training and testing
- the purpose of the classifier (context)
- the type of input data (e.g. files, images, videos)
- the complexity of the methodology
- the type of classifiers tested

Placing these papers side-by-side is not meant as an apple-to-apple comparison in the mere fact that the input and output formats, dataset size, and the rigorous testing and validation processes of each paper is different. Care must be exercised in order not to be misled by seemingly high values of test results as this should be evaluated based on the target function or a performance requirement. For machine learning classifiers where emphasis on accuracy is encouraged, it should be noted that accuracy alone as the basis of performance is limited and even misleading without considering the domain and context for which the model is used for.

The factors raised above is a clear reminder that the persistent call for standardized digital corpora in the realm of digital forensics ([49], [50], [51], [52]) remains a necessity if consistent and sound comparisons are to be made.

4 Methodology

This research study used Design Science as it is well suited to the creation of a viable artefact (prototype) that is grounded on solid research principles in the information systems domain. The work of Peffers *et al*. [53] demonstrated a six-step process for design science: problem identification and motivation, definition of the objective for a solution, design and development, demonstration, evaluation and communication. The process is outlined in Figure 14.

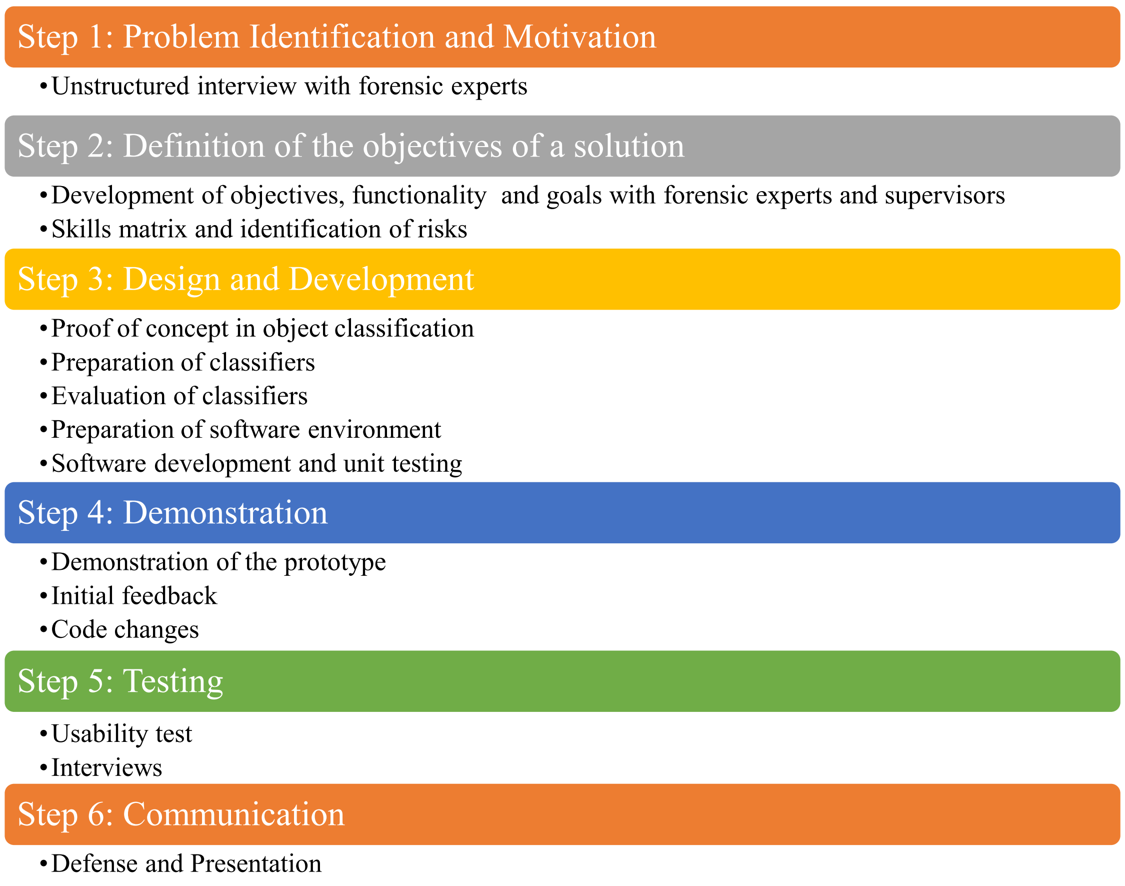

Step 1: Problem Identification and Motivation
•Unstructured interview with forensic experts
Step 2: Definition of the objectives of a solution
•Development of objectives, functionality and goals with forensic experts and supervisors
•Skills matrix and identification of risks
Step 3: Design and Development
•Proof of concept in object classification
•Preparation of classifiers
•Evaluation of classifiers
•Preparation of software environment
•Software development and unit testing
Step 4: Demonstration
•Demonstration of the prototype
•Initial feedback
•Code changes
Step 5: Testing
•Usability test
•Interviews
Step 6: Communication
•Defense and Presentation

Figure 14 Design Science Process

As the research study aimed to create a digital forensic prototype that is innovative, purposeful, and evaluated in the proper scientific process, the Design Science methodology is applicable and therefore has been adapted. The design science approach used in this study is described in detail in the next section.

4.1 Problem Identification and Requirements

The first activity for this research is an unstructured interview with forensic experts to find out gaps especially in tools used for forensic investigations. Figure 15 shows the respondents' distribution of practical experience in the industry: three junior examiners and two senior consultants.

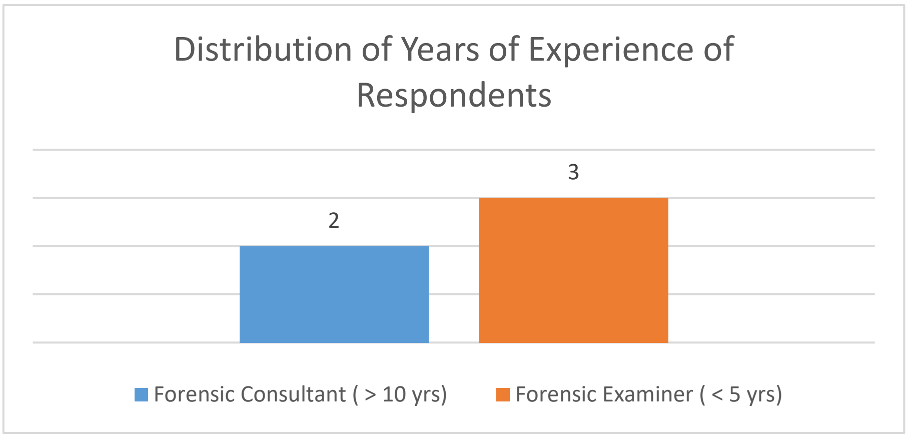


Figure 15 Unstructured Interview Respondent Distribution

Figure 16 summarizes the discussion in a mind map while Appendix 8 – Unstructured Interview – Raw Data compiles the data gathered during the interview.

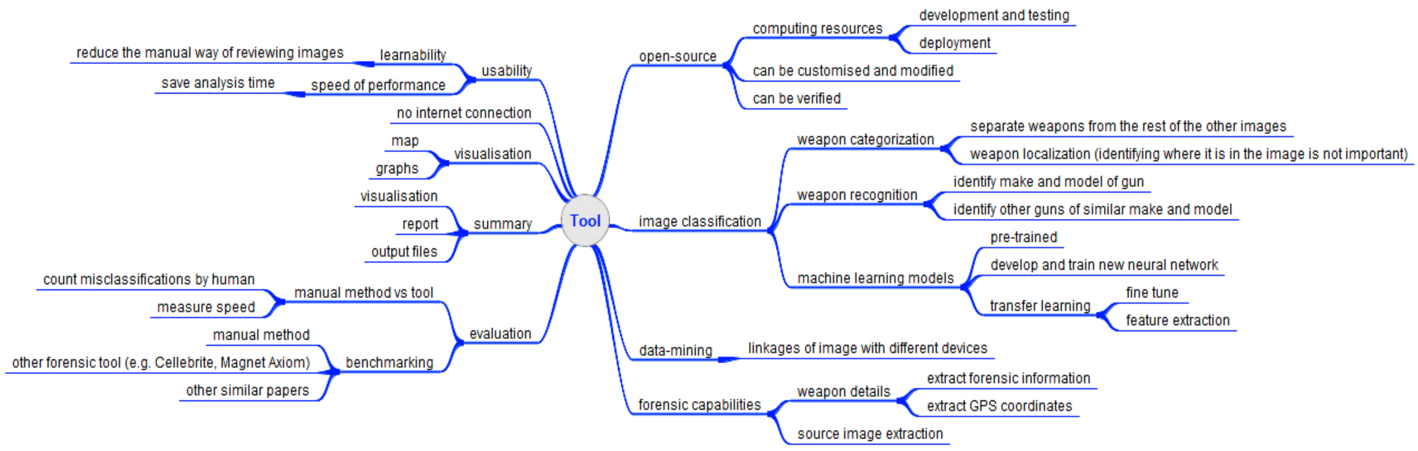


Figure 16 Unstructured Interview Discussion

An open-source forensic tool with the discussed capabilities in Figure 16 would be a remarkable feat. However, the development of a fully functional tool with the described functionalities is not possible within the timeframe of the study. However, since the research aims to investigate the feasibility of using pre-trained classifiers, a prototype with the following minimum requirements would nevertheless be useful in a forensic investigation:

- Open-source tool that can be customized, modified and improved
- Automated categorisation of images
- Meta-data extraction and maps
- File export
- Visualisation
- Weapon categorisation

Creating test cases and scenarios meant to cover the minimum functionalities is part of the development of the unit test and design of the usability test.

4.1.1 The Binary Classifier Dilemma for Forensic Use

The prototype attempts to create a binary classifier for weapons, specifically guns. A binary classifier would have to detect a gun against all the other non-gun objects in the wild. To illustrate, Figure 17 shows that a binary classifier needed for this study requires successfully differentiating a gun, such as a revolver, rifle or a pistol against a non-gun object which could be a plant, animal, building, or person.

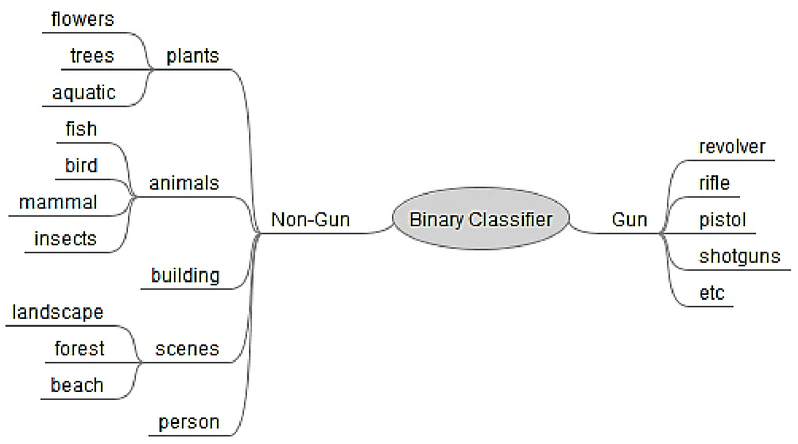


Figure 17 Gun Classifier

Take a cat vs dog classifier, for example. The machine learning algorithm is trained with images from *cat* and *dogs*. The machine is trained to tell them apart by learning the distinct features of a *cat* and a *dog* to differentiate the two output classes. During testing, the algorithm is fed with images containing only a cat or a dog and then classifies them accordingly. However, in the context of this study, in a gun detection scenario, the non-gun labels are infinite. The algorithm would have to differentiate a gun from the rest of the other objects such as an animal or a plant or a building or a person – and the extent of the categories is vast. A more practical solution is to use a multi-label a classifier before a binary classifier as seen in Figure 18.

The design and development of the prototype's classifier is based on this approach.

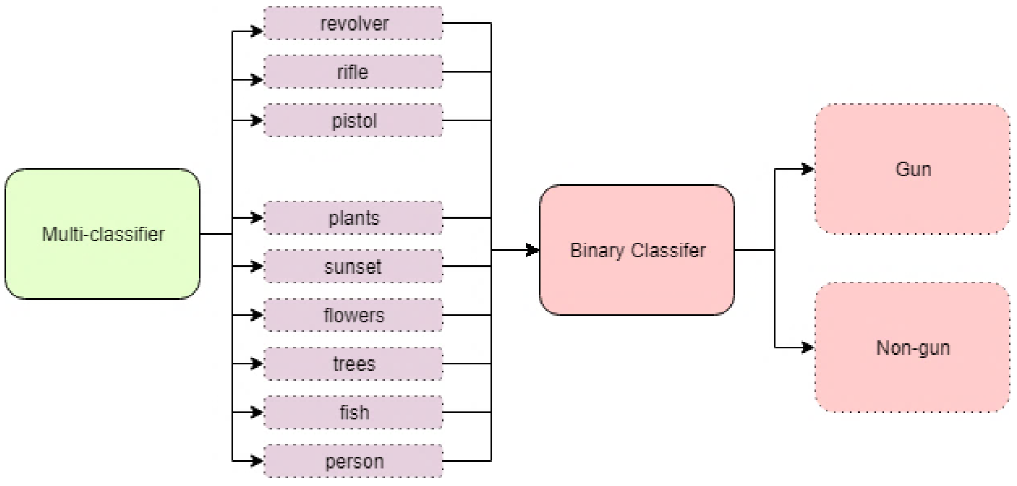


Figure 18 Multi-class Classifier

4.1.2 Algorithm for the Prototype

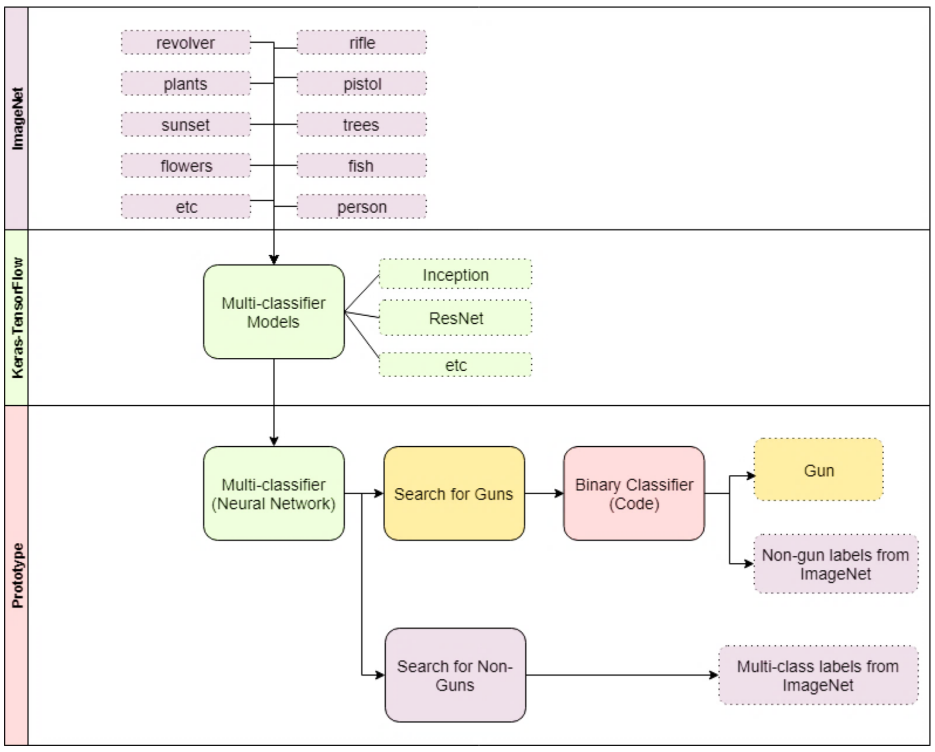


Figure 19 Prototype Process Diagram

Figure 19 shows the proposed algorithm for the prototype. The prototype aims to leverage the extensive ImageNet pre-training done on the neural networks. Appendix 1 – ImageNet Classes contains the full list of the output classes the models were trained on. These multi-classifier models are available for download from the Keras library. The best model based on some evaluation criteria is taken and used in the prototype as the classifier or predictor. The binary classifier is a module that takes into account the search criteria, the output labels of the predictor and threshold values if supplied.

4.2 Design and Development

4.2.1 Performance Estimation

The design and development of this research study is centred on performance estimation. The predictive performance of a model with unseen or future data is the goal of machine learning applications or the development of new algorithms. A typical machine learning process involves training a model, validating a model (tuning hyperparameters and estimating performance) and testing the model with unseen data. In this study, the models were already trained in ImageNet and the (published) accuracy has been estimated using ImageNet's validation dataset. The validation set contains the 1000-class output labels while our study only requires the gun classification accuracy. Therefore, one of the questions we attempt to answer in estimating performance: *Does the published accuracy of the selected models hold when evaluated and tested against our dataset in our context (gun classification)*?

How do we choose the best model? The study is interested in selecting the best-performing model from a pre-defined set of pre-trained models downloaded from Keras by ranking them against each other based on some criteria. To achieve this, the approach illustrated in Figure 20 is implemented. During model evaluation, four models are evaluated based on some performance metrics on a specific dataset (WEAPON-DB3). The best model is selected based on performance. The performance of the best model is estimated by running the model against a few more datasets. If the model's performance is acceptable, the final model is used as the multi-label classifier for the prototype.

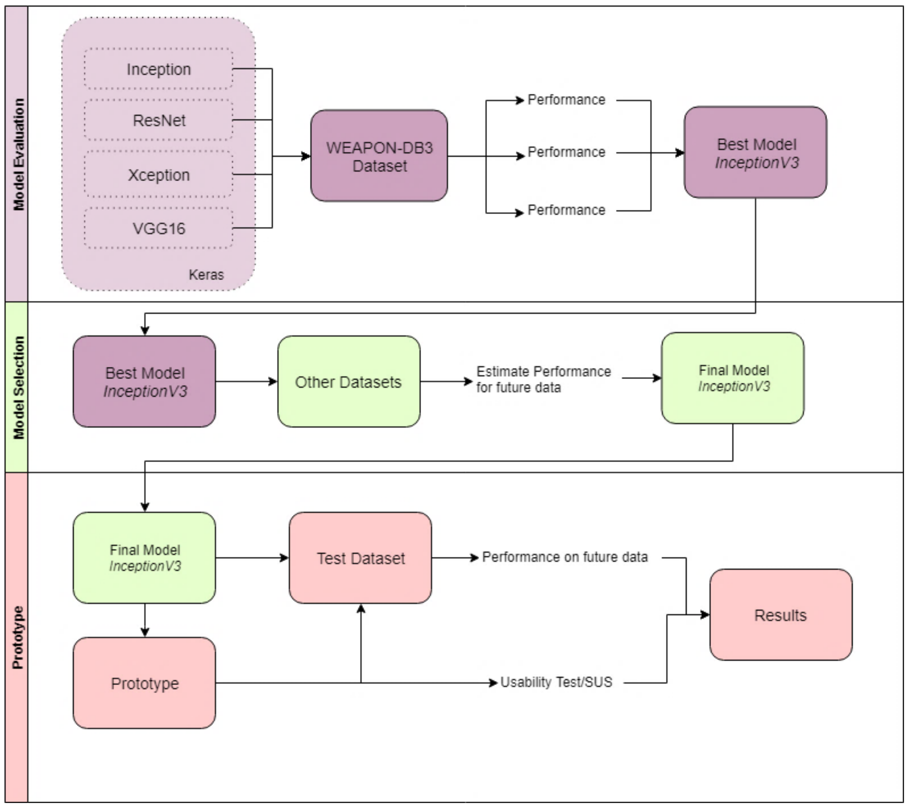


Figure 20 Performance Estimation Approach

Based on the approach outlined in Figure 20, the design and development flow can be divided into five major steps: Data Collection and Pre-processing, Model Evaluation and Selection, and Prototype Development. This five-step process is illustrated in Figure 21 and discussed in detail in the next sections.

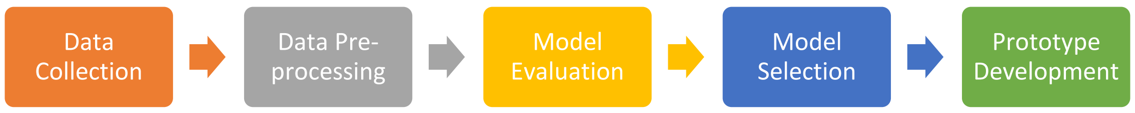


Figure 21 Design and Development Approach

4.2.2 Data Collection

Data acquisition is required for two processes: (1) model evaluation and selection, and (2) prototype testing. The evaluation dataset is used to select the best computer vision model (best and final model) based on the performance metrics. Once the model is selected, the prototype is developed using the final model as the predictor or classifier. The testing dataset is then used to evaluate the performance and usability of the prototype.

No evaluation and testing images should be downloaded from the ImageNet database[1] because the pre-trained computer vision models had been trained from this set.

Dataset for Model Evaluation and Selection

The following dataset sources are used to evaluate the computer vision models and select the best performing model for the prototype:

(1) Dataset[2] from Olmos, Tabik and Herrera's paper: *Automatic Handgun Detection Alarm in Videos* [15]. The dataset is taken from the test folder[3] with 304 images of guns and non-guns. This dataset is named "*WEAPON-DB3*" or "*WEAPON-DB3: Ugr.Es*" in this paper.

(2) Randomly downloaded data from Google images searching for pistol and revolvers with 3-channel images (coloured)

(3) Gun images viewed from Windows Explorer, retrieved from the Window's thumbnail cache[4] and exported using ThumbCache viewer open source tool [54]

(4) Partially carved gun images such as one shown in Figure 22

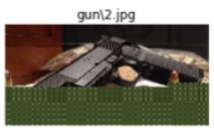


Figure 22 Carved and Partial Gun Image

Dataset for Testing

(1) Exported images from an Android mobile dump with some random gun images;

(2) Exported images using FTK from a Windows hard disk dump with some random gun images

(3) Gun images with modified EXIF information for testing

[1] http://image-net.org/synset?wnid=n03948459
[2] https://sci2s.ugr.es/weapons-detection
[3] https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/WeaponsDetection/BasesDeDatos/Test.zip
[4] C:\Users\{UserName}\AppData\Local\Microsoft\Windows\Explorer

Dataset Description

Datasets are either *balanced* or *unbalanced*. Balanced dataset indicates a relatively similar percentage of guns and not-gun in the distribution. In digital forensics, it is highly unlikely that a balanced distribution is acquired. It would be oversight if the models and the prototype were not tested against the skewed cases.

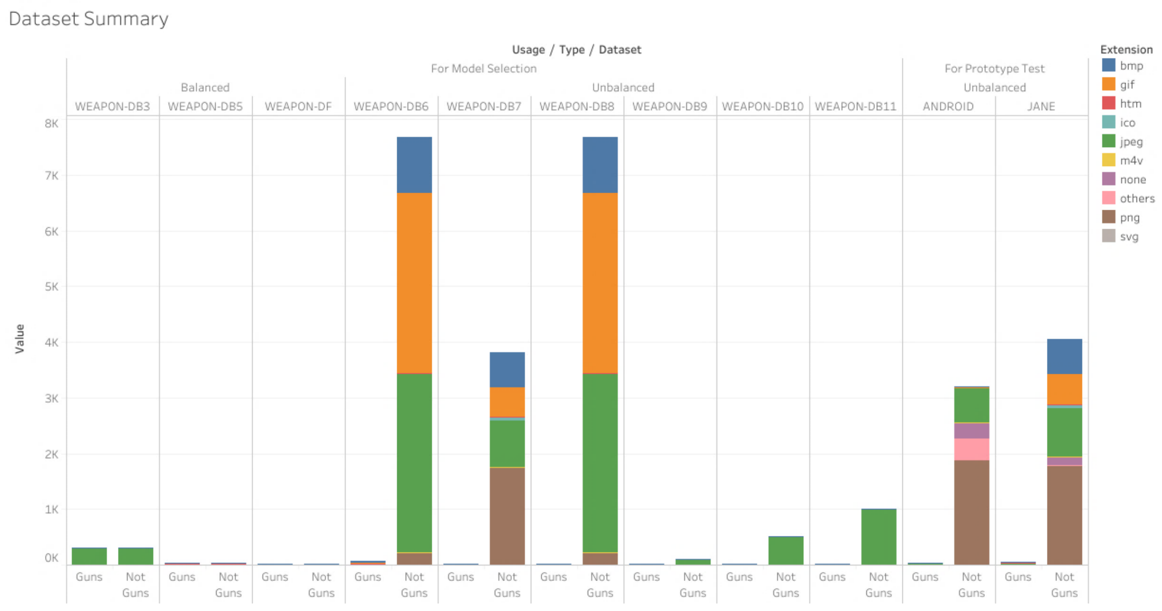
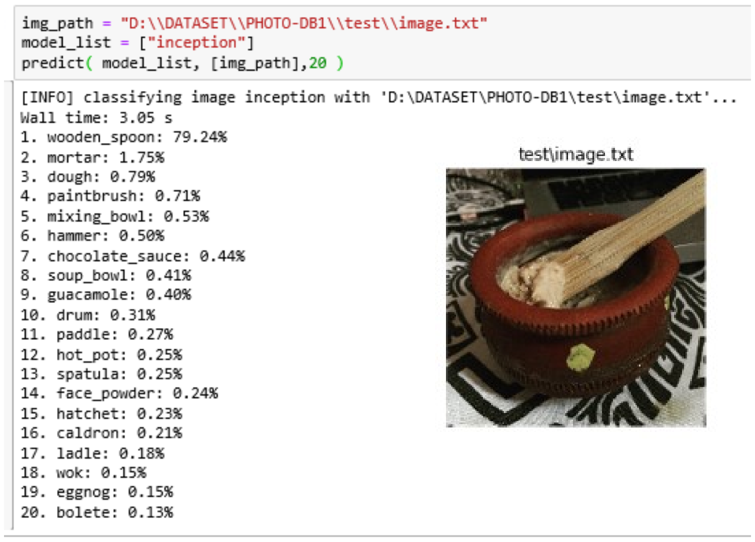


Figure 23 Frequency Distribution of Datasets

Figure 23 describes the frequency distribution of the common image formats for each dataset. It also summarises the number of guns versus the not-gun images in the dataset. From the figure, we can see that the evaluation and test sets are dominated by unbalanced datasets with a negatively skewed distribution, that is, an overwhelming number of not-guns are present. This is intentional because we want to create datasets that are close to real (realistic) as possible. Although the datasets do not guarantee that the diversity of the file formats is truly representative of the real-world case data, due diligence was employed in simulating realistic data in the evaluation and test datasets, i.e. random gun pictures downloaded from the Internet in various file formats. This enables us to determine the performance of the model when tested in the context of forensics.

Also, it is worth noting that the Keras image libraries do not rely on the file extensions to predict images. It verifies the content of the file object. Figure 24 shows the exploratory testing done on a mismatched file extension “image.txt” to show that the Keras-based model can process and display misnamed files.



```
img_path = "D:\\DATASET\\PHOTO-DB1\\test\\image.txt"
model_list = ["inception"]
predict( model_list, [img_path],20 )
```

```
[INFO] classifying image inception with 'D:\DATASET\PHOTO-DB1\test\image.txt'...
Wall time: 3.05 s
1. wooden_spoon: 79.24%
2. mortar: 1.75%
3. dough: 0.79%
4. paintbrush: 0.71%
5. mixing_bowl: 0.53%
6. hammer: 0.50%
7. chocolate_sauce: 0.44%
8. soup_bowl: 0.41%
9. guacamole: 0.40%
10. drum: 0.31%
11. paddle: 0.27%
12. hot_pot: 0.25%
13. spatula: 0.25%
14. face_powder: 0.24%
15. hatchet: 0.23%
16. caldron: 0.21%
17. ladle: 0.18%
18. wok: 0.15%
19. eggnog: 0.15%
20. bolete: 0.13%
```

Figure 24 Prediction on a mismatched extension (image.txt)

All images collected have file extensions representing the file format of the images (no mismatched file extensions). Figure 25 summarises the distribution of file formats (based on file extensions) for the datasets used in this study. JPG/JPEG images remain the largest distribution of test images because it is the most common image file format in the wild.

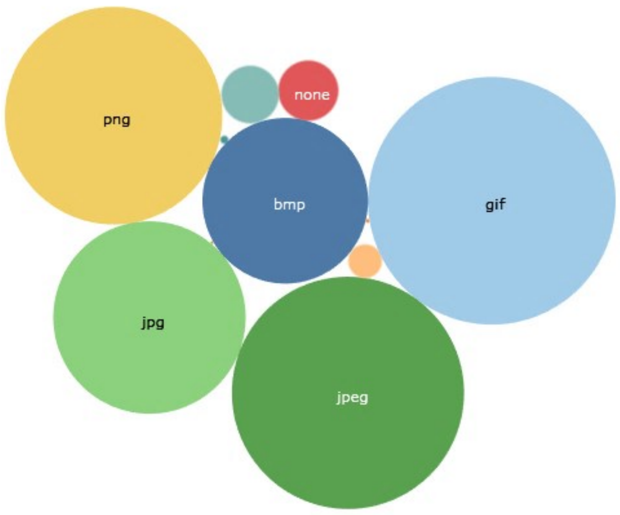


Figure 25 Frequency Distribution for Images containing guns

Dataset Summary

The complete list of datasets collected and used is summarised in Table 1.

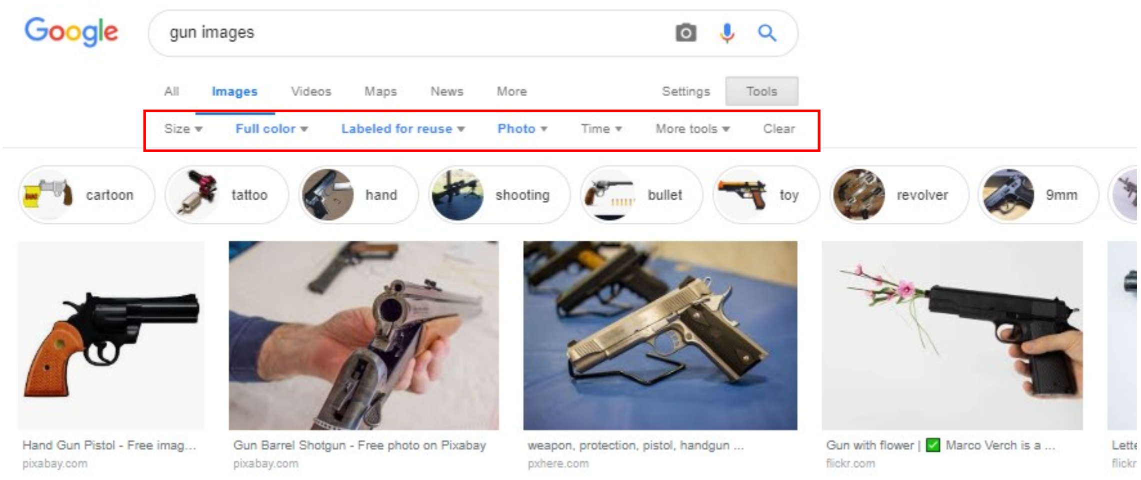
Table 1 List of Datasets Used

Dataset	% of guns	Total	Guns	Not Guns	Unprocessed
Evaluation Dataset for Model Selection					
WEAPON-DB3 (Ugr.Es)	*50.00%*	608	304	304	0
WEAPON-DF	*50.00%*	40	20	20	0
WEAPON-DB5	*50.62%*	81	41	39	1
WEAPON-DB6	*0.79%*	7746	61	7426	259
WEAPON-DB7	*0.63%*	3836	24	3740	72
WEAPON-DB8	*0.31%*	7709	24	3740	3945
WEAPON-DB9	*19.35%*	124	24	99	1
WEAPON-DB10	*4.49%*	534	24	509	1
WEAPON-DB11	*2.33%*	1030	24	27	2
Testing Dataset for Prototype Evaluation					
ANDROID	*1.17%*	3236	38	3198	214
JANE	*1.05%*	4101	43	4058	112

4.2.3 Data Pre-processing

4.2.3.1 Download of Images

Gun images are downloaded from Google Images and filtered by usage rights and colour as shown in Figure 26.


gun images

Size ▾ | Full color ▾ | Labeled for reuse ▾ | Photo ▾ | Time ▾ | More tools ▾ | Clear

Google
All
Images
Videos
Maps
News
More
Settings
Tools
cartoon
tattoo
hand
shooting
bullet
toy
revolver
9mm
Hand Gun Pistol - Free imag...
pixabay.com
Gun Barrel Shotgun - Free photo on Pixabay
pixabay.com
weapon, protection, pistol, handgun ...
pxhere.com
Gun with flower | Marco Verch is a ...
flickr.com

Figure 26 Google Image search for gun pictures

Although nowadays, people upload enhanced photos using filters that distort the contrast and the colour distributions of the image, filters are not checked when downloading the pictures from the internet. Hence, images may or may not contain enhanced photos uploaded by users.

4.2.3.2 Extraction of Images

Since the datasets are retrieved by simulating realistic data by taking all exported photos from a hard-disk dump and mixing some gun images in the folder, some file formats are uncommon while some cannot be processed by the model. This is to simulate an unbalanced dataset by using the export function from FTK without filtering or cleaning the data (such as removing uncommon file formats). This also means that the prototype can be directly supplied with the directory of freshly extracted photos from any forensic tool. Figure 27 shows a scenario wherein the forensic tool (*e.g.* FTK) extracts the photos from the acquired device (*e.g.* mobile) to a directory. This directory is supplied to the prototype even without filtering the images (full-sized, thumbnails, icons, carved, etc). Although in a real case, some form of filtering would have been employed.

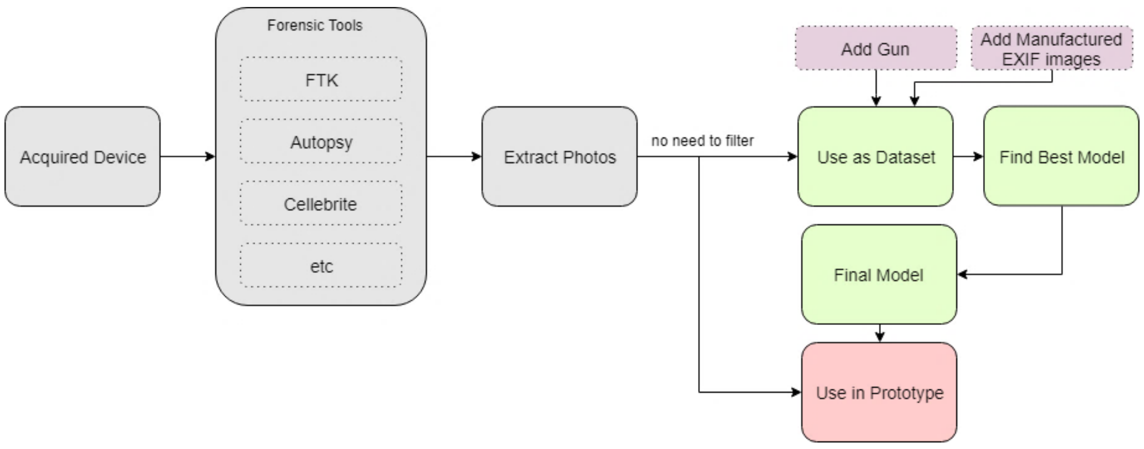
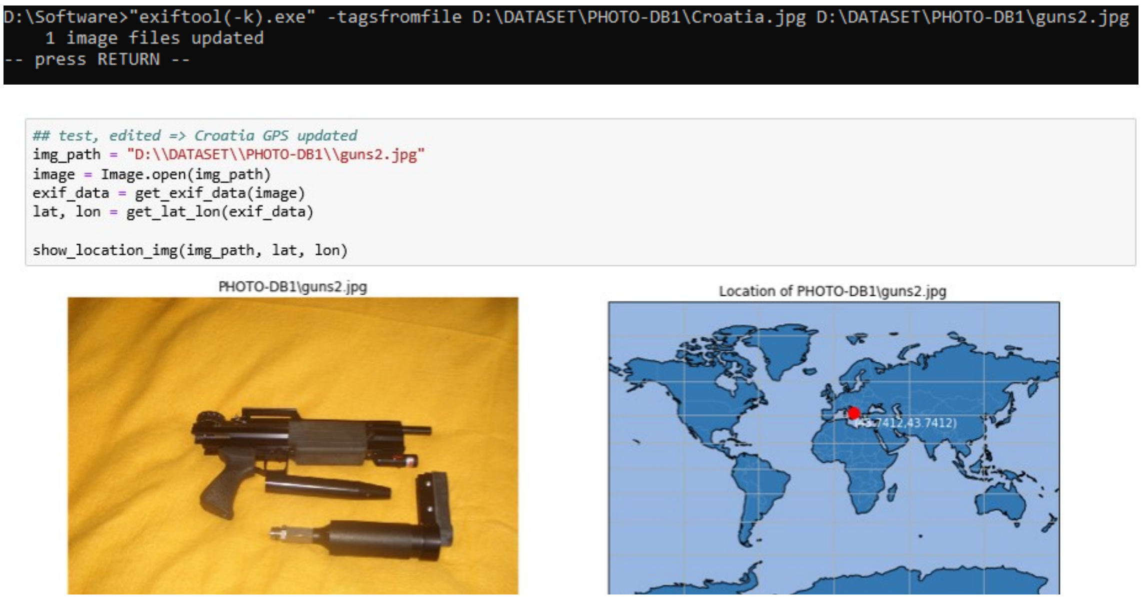


Figure 27 Photo Extraction

4.2.3.3 EXIF Information Modification

The tool used to retrieve, modify or erase the EXIF information is taken from [55]. For the testing scenario, the modification of EXIF information is necessary to simulate a gun picture with GPS Coordinates. The steps in manufacturing EXIF information is shown in Figure 28. Since the tool allows transfer of all tags from one picture to another, the tags of a picture with Croatia's GPS coordinates, "Croatia.jpg" is transferred to a gun image, "guns2.jpg".



```
D:\Software>"exiftool(-k).exe" -tagsfromfile D:\DATASET\PHOTO-DB1\Croatia.jpg D:\DATASET\PHOTO-DB1\guns2.jpg
    1 image files updated
-- press RETURN --
```

```
## test, edited => Croatia GPS updated
img_path = "D:\\DATASET\\PHOTO-DB1\\guns2.jpg"
image = Image.open(img_path)
exif_data = get_exif_data(image)
lat, lon = get_lat_lon(exif_data)

show_location_img(img_path, lat, lon)
```

Figure 28 Gun with manufactured EXIF

4.2.3.4 Ground Truth Labelling / Annotation

In classification tasks, ground truth labelling is often an arduous activity because of the overwhelming amount of data that needs to be annotated. For training images, the annotation itself takes the bulk of the researcher's time. Annotation must be conducted with care because the performance of a classifier hinges on the ground truth having minimal error. The ImageNet dataset contains approximately 0.3% of incorrectly annotated ground truth labels [28], noting that 5 out of 1500 images from the ILSVRC[1] test set were found to be wrongly classified.

In this study, we are the *perfect classifier* for the ground truth data. If a gun is detectable by the human eye, then it is classified as a gun. The images for model selection (all

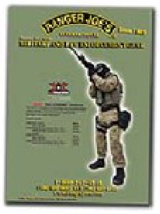
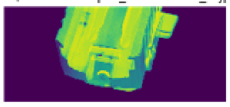
[1] Large Scale Visual Recognition Challenge (ILSVRC)

datasets named *WEAPON-*)* did not pose any problems during ground truth labelling. The images with guns are easy enough to identify. *WEAPON-DB3: Ugr.Es*, in particular, was already annotated by the paper's researchers[1]. However, some challenges are encountered when trying to annotate the test dataset (*ANDROID* and *JANE* datasets). We tried to overcome these challenges using optimistic labelling.

ANDROID and JANE: Optimistic Labelling

The test images for testing the prototype are taken from disk dumps from data that we, the human ground truth annotator, have not seen before. Although humans tend to detect better than machines in classifying images by visual saliency (i.e. classification based on an object that is more highlighted and pronounced than the rest) [28], the common-sense approach (human-eye detectable) does not help much when faced with images from movie posters or images taken in poor lighting conditions. To resolve the dilemma, *optimistic* classification is used. This means, for the unsure images (is this a gun or not a gun?), we have labelled them as guns. Table 2 shows two images where optimistic labelling has been performed.

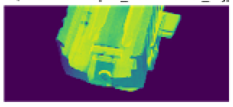
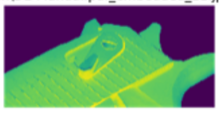
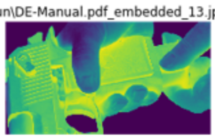
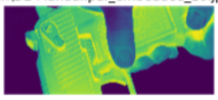
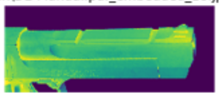
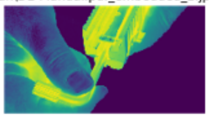
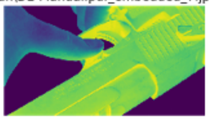
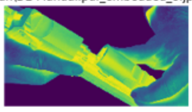
Table 2 Examples of Optimistic Classification

Images	Notes
exported\Carved [314].jpeg 	Can this be classified as a gun? Is the assault rifle in this image too small to be recognized? Perhaps the context and background of the picture help reinforce that the person is holding something that resembles a rifle? *Classify as a gun, anyway.*
gun\DE-Manual.pdf_embedded_1.jpg 	Can this be classified as a gun? The x-ray image does look like it is a gun but only when viewed with similar images. *Classify as a gun, anyway.*

During evaluation, some of the optimistic classifications ended up as a mis-detection (false negative). Forensic examiners have been asked during the evaluation phase if these mis-detections are acceptable. The images that are labelled as guns by the annotator but can be contested are shown in Table 3 with the top-5 predictions from InceptionV3.

[1] https://sci2s.ugr.es/weapons-detection

Table 3 Optimistic Ground Truth Labelling of Test Dataset

Image	**Prediction by InceptionV3**	**Ground Truth Annotation**
Folder: Jane Doe exported\Carved [314].jpeg 	1. book_jacket: 30.17% 2. binder: 20.26% 3. web_site: 6.52% 4. comic_book: 4.57% 5. packet: 3.72%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_1.jpg 	1. switch: 80.35% 2. radiator: 4.34% 3. panpipe: 3.32% 4. picket_fence: 1.55% 5. space_heater: 1.03%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_11.jpg 	1. plate_rack: 15.94% 2. teapot: 13.97% 3. water_jug: 7.17% 4. pitcher: 3.48% 5. coffeepot: 3.22%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_13.jpg 	1. mask: 32.53% 2. toilet_tissue: 15.16% 3. paper_towel: 8.46% 4. packet: 3.05% 5. matchstick: 3.03%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_16.jpg 	1. iron: 12.62% 2. shower_curtain: 8.09% 3. shoe_shop: 2.89% 4. toilet_tissue: 2.37% 5. electric_guitar: 2.27%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_18.jpg 	1. printer: 49.46% 2. cradle: 6.77% 3. grand_piano: 3.58% 4. toilet_tissue: 3.44% 5. toilet_seat: 2.41%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_2.jpg 	1. syringe: 70.25% 2. switch: 1.28% 3. bonnet: 1.22% 4. panpipe: 1.00% 5. quill: 0.86%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_7.jpg 	1. syringe: 48.23% 2. sewing_machine: 5.17% 3. can_opener: 3.28% 4. modem: 2.48% 5. dial_telephone: 2.16%	Gun
Folder: Android gun\DE-Manual.pdf_embedded_8.jpg 	1. syringe: 87.88% 2. screw: 0.79% 3. switch: 0.24% 4. lotion: 0.23% 5. upright: 0.20%	Gun

4.2.4 Model Evaluation

The core of the prototype is the computer vision model. Selecting the right model is crucial in creating a working and usable prototype. The selection of the best performing model in machine learning requires a sound comparison of appropriate metrics. The following section describes some of the metrics used in this study.

4.2.4.1 Metrics

For regression-based algorithms, performance can be based on residuals (errors) or deviations between predicted and ground truth values such as the root mean squared error (RMSE); the model of choice is the algorithm with the minimal RMSE. Classifiers, on the other hand, are evaluated based on either a class or a probability output. In the context of a binary classification problem, the values are by convention either positive (+) or negative (-). This classifier can make two types of errors: (1) it can incorrectly classify an image with no gun under the gun category (false positive or Type I error), and (2) it can incorrectly classify an image with a gun under the not_gun category (false negative or Type II error). It is easier to determine and visualise these misclassifications by creating a *confusion matrix* or sometimes called a *contingency* or *confounding table*.

Confusion Matrix

The performance of a classifier is evaluated based on a baseline classification, ideally a perfect classifier such as the Bayes classifier[1] but in practice an output of another gold standard test[2]. For this study, we are the ground truth baseline. The resulting cross-tabulated results of predicted and actual values is called a *confusion* matrix as shown in Table 4.

Table 4 Confusion Matrix

Confusion Matrix		**Ground Truth / True, Actual or Target**	
		Positive	Negative
Predicted / Classified by the Model	Positive	TP	FP
	Negative	FN	TN

[1] The Bayes classifier is an unattainable gold standard for classification because in real life, there is no way to know the conditional distribution of Y given X for the population. [56]

[2] For example, large-scale image classification employed by ImageNet is done by leveraging crowdsourcing strategy via Amazon Mechanical Turk (see [89])

TP (True Positive) – correctly classified as Positive, also called a “hit”
TN (True Negative) – correctly classified as Negative
FP (False Positive) – misclassified as positive but is negative; also called “false alarm”
FN (False Negative) – misclassified as negative but is positive; also called a “miss”

Table 4 shows the confusion matrix: test prediction results are on the vertical axis and baseline or ground truth baselines are on the horizontal axis. The underlying performance evaluation criteria used throughout this study is derived from this matrix.

Accuracy and Precision

Common in regression problems, accuracy can also be used to measure how well the classifier identifies or excludes a condition. Also called Rand Accuracy or Classification Accuracy, it takes the ratio of correctly classified predictions against the total number of samples. The formula for accuracy is taken from [56] and shown in Equation 1:

$$\boldsymbol{Accuracy\ (ACC) = \frac{TP + TN}{TP + TN + FP + FN}}$$

Equation 1 Classification Accuracy

Good accuracy results may not always be relevant. For a model that aims to detect a gun, for example, the cost of a miss (false negative, i.e. a gun classified to be not a gun) may be unacceptable.

Precision is called the Positive Predictive Value (PPV) or True Positive Accuracy. It is defined as the number of true positives that the model deemed to be positive. The formula for precision is taken from [57] and shown in Equation 2:

$$\boldsymbol{Precision\ (PPV) = \frac{TP}{TP + FP}}$$

Equation 2 Precision

Sensitivity and Specificity

Sensitivity or *Recall* is the ratio of true predictions from the target positive classifications. The term sensitivity and recall are used interchangeably in this paper. This is also called the Hit Rate, True Positive Rate or the Probability of Detection. This value reflects the number of relevant positive cases that the classifier was able to detect. In e-discovery, this measure is not highly regarded because the relevance of the documents that were not retrieved cannot be determined. However, in medicine, this measure is primary as the desired goal is to identify as much positives as the model can. The formula for recall is taken from [57] and shown in Equation 3:

$$\boldsymbol{Recall\ or\ Sensitivity} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}}$$

Equation 3 Recall

The inverse of sensitivity or specificity is the proportion of true negatives from the total target negative cases. It is also called True Negative Rate. The formula for specificity is taken from [56] , [57] and shown in Equation 4:

$$\boldsymbol{Specificity} = \frac{\boldsymbol{TN}}{\boldsymbol{FP + TN}}$$

Equation 4 Specificity

As the model's sensitivity increases, specificity decreases. To illustrate, a model that is sensitive means that it classifies any firearm (rifle, grenade launcher, rocket, machine guns) as a hand-gun (more false positives) while a highly specific model will only classify a pistol but not a revolver as a hand-gun (more false negatives). The delicate art of finding the right balance between sensitivity and specificity drives many research applications today.

False Negative Rate and False Positive Rate

The proportion of target positives that are classified otherwise is called a miss or formally, the False Negative Rate. The formula for false negative rate is taken from [57] and shown in Equation 5

$$\boldsymbol{False\ Negative\ Rate} = \frac{\boldsymbol{FN}}{\boldsymbol{TP + FN}}$$

Equation 5 False Negative Rate

The complementary form for the false negative rate is called a fallout, false alarm rate or the false positive rate. The formula for false positive rate is taken from [57] and shown in Equation 6:

$$\boldsymbol{False\ Positive\ Rate\ (FPR)} = \frac{\boldsymbol{FP}}{\boldsymbol{FP+TN}} \text{ or } \mathbf{1} - \boldsymbol{Specificity}$$

Equation 6 False Positive Rate

The definitions of the metrics are translated into questions to easily understand their relevance in the context of gun classification in this paper. The metrics and corresponding questions are tabulated in Table 5.

Table 5 Summary of Metrics

Metrics	**Formula**	**Questions**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Out of all pictures in the directory, how many have been correctly classified?
Precision	$\frac{TP}{TP+FP}$	Out of all pictures classified as guns, how many are actual guns?
Sensitivity or Recall	$\frac{TP}{TP+FN}$	Out of all the actual gun pictures, how many are correctly classified?
Specificity	$\frac{TN}{FP+TN}$	Out of all the actual pictures without guns, how many are correctly classified?
False Negative Rate	$\frac{FN}{TP+FN}$	Out of all the actual gun pictures, how many are wrongly classified? How many guns pictures did the model miss?
False Positive Rate	$\frac{FP}{FP+TN}$	Out of all the actual pictures without guns, how many are wrongly classified? How many no- gun pictures did the model miss?

In practice, evaluating trade-offs is unavoidable. The trade-off between accuracy and precision should be reasonable and practical in real life. For example, if the goal is to fit a model that can detect all the gun images (100% recall rate, no false negatives) regardless of the false positives, then it might return too many false positives that the model has now become unbearably useless.

F-score

There are attempts to summarize the confusion matrix into a single number. One such attempt is the *F-score* (sometimes called *F-measure)* which is the harmonic average of precision and recall. The formula is taken from [57] and shown in Equation 7:

$$F_\beta = (1 + \beta^2)\frac{(Precision * Recall)}{(\beta^2 * Precision + Recall)}$$

Equation 7 F-score

The most common measure is F_1 which gives equal weight to precision and recall at $\beta = 1$. The formula is taken from [56], [57] and shown in Equation 8:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Equation 8 F_1-score

If more weight is given to recall, the score is called F_2, and $F_{0.5}$ score is meant to give less weight to recall than precision.

Oftentimes, this measure is enough to evaluate the performance of a model. However, if true negatives are equally important, one major drawback is this calculation's inability to take the true negatives into account. For unbalanced datasets, where true negatives far outweigh true positives, the F-measure might provide misleading results.

Matthew's Correlation Coefficient (MCC)

The *Matthews correlation coefficient* (MCC) is regarded as one of the best measures that summarizes the confusion matrix into a single probability [57]. Among other measures such as F_1 and accuracy, MCC is more informative as it is easily interpretable, robust to changes in the prediction goal [58], and takes the size of the matrix into consideration to adjust for unbalanced datasets [59] . Unfortunately, this value is not reported among any of the literature reviewed in this paper. The MCC is given in Equation 9:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Equation 9 Matthew's Correlation Coefficient

MCC returns a value from -1 to 1. 1 indicates a perfect prediction, -1 indicates an unmatched prediction versus observation values and 0 indicates random guess. A value of 1 is ideal. This study reports this metric instead of the F-measure; models with MCC as close to 1 as possible are regarded as "good" classifiers.

Receiver Operating Characteristics (ROC) Curve

As mentioned, reporting the performance of a classifier using accuracy is misleading and inaccurate. As such, most dichotomous responses usually employ ROC curves to express Sensitivity and Specificity across a range of threshold values. In most papers, the ROC curves are plotted using the False Positive Rate[1] instead of Specificity as it gives a more intuitive feel, up is good versus down is good for the latter. Figure 29 shows two representations of the ROC graph: Specificity vs TPR (left) and 1- Specificity vs TPR (right). 1 – Specificity is also called the False Positive Rate. The graph on the right is the more commonly seen version of the ROC because it is more intuitive to choose better models in an upward trend as opposed to the left graph; using the downward trend is counter-intuitive. The ideal regions are highlighted; the perfect classifiers are in (1,1) and (0,1) for left and right graphs respectively.

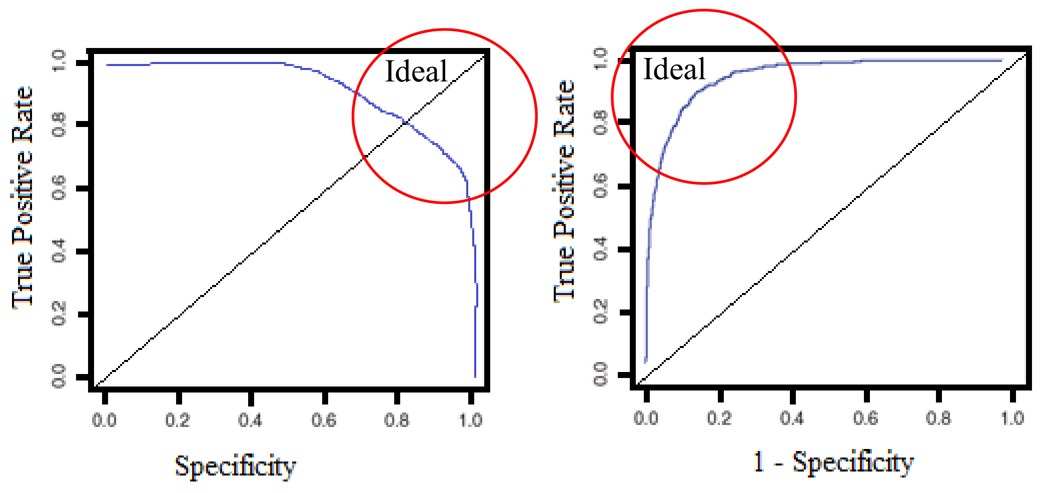


Figure 29 ROC Curves

Visualising Trade-offs

In a paper by Provost *et al*. [60], the authors recommended the use of ROC to show the trade-off between correctly classified results (recall or sensitivity) and falsely classified results or false alarm rates. In [61], the ROC space is shown to evaluate the performance of a binary classifier.

[1] FPR = 1 – Specificity

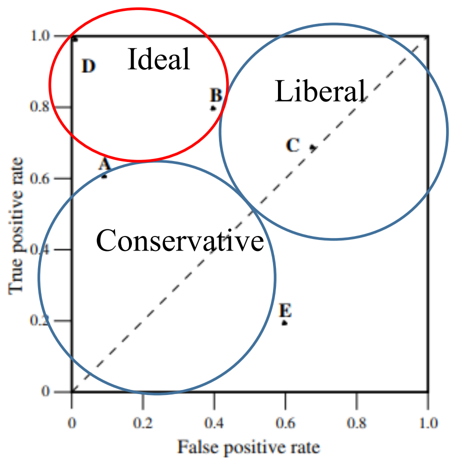


Figure 30 ROC Space

Figure 30 shows the ROC graph taken from Fawcett's work[61], but in which we had superimposed the classifier performance within the ROC space as explained by Fawcett in the same paper. The point (0,1), where classifier D is located, represents a perfect classifier. It has virtually no false positives. This is the area that the model aims to achieve. The point (0,0) represents an extreme strategy of issuing no positive classification and therefore, no false alarms. Classifiers along this area are deemed conservative: a positive prediction is only given when there is strong evidence. Thus, classifiers along this space also yield low positive classifications as well as low false alarms. The point (1,1) issues the opposite strategy by classifying all observations as positive. Classifiers around this area tend to be generous or liberal with their predictions resulting in high positive rate at the cost of false alarms. The dashed diagonal line ($y = x$) represents the strategy of randomly guessing an output class having a probability of 0.50[1]. Classifiers near or along this line are said to be near to "guessing" the positive classification.

Decision or Probability Threshold

For probabilistic classifiers[2] such as Naïve Bayes, Linear Discriminant Analysis model or a Neural Network, the output is an instance probability (a continuous numeric score) that represents the degree to which the instance is a member of a class. The classification is implemented by using a decision threshold (or probability threshold) to produce a

[1] Bayes classifier uses a threshold of 50% for the probability of assigning a random observation to the positive or negative class.
[2] Classifiers that output probabilities instead of categorical data

discrete (binary) classifier: if the output is above the threshold it is positive, else, negative [56].

The ROC curve below is shown by thresholding the predictions. Figure 31 shows two ROC graphs with different threshold values. The left graph is taken from [61] and the right graph from [56]; a smoother curve is achieved by running the function on multiple thresholding values as seen in the right graph.

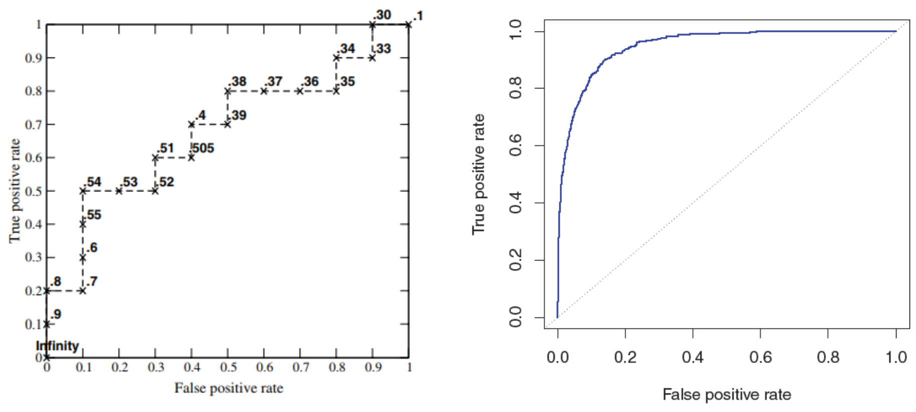


Figure 31 ROC Curve with Threshold

Although classifiers along a higher true positive rate but lower false positive rates are deemed to be good performing classifiers and is ideal, determining the appropriate threshold value depends on domain knowledge, that is, it depends on the problem and the industry for which the algorithm will be used for [56]. Hence, a certain degree of balance is crucial in finding a “good” classifier in this context. The tuneable parameters and ideal values are not intrinsic in the test or validation methodology; these should be determined by the context on which the test is applied.

Area Under the Curve (AUC)

To quantify the performance of the ROC in terms of numerical values, analysts use the Area Under the Curve (AUC) measure. In machine learning, the ROC AUC is often used to measure model performance. The AUC of a random classifier is 0.5 [56]. A model with an AUC that is greater than 0.50 means it performs better than a random model (see Equation 10). A perfect classifier has an AUC of 1.00 (see Equation 11).

$$0.50 < AUC < 1.00$$

Equation 10 AUC of a classifier that is better than a “random” guess

$$AUC = 1.00$$

Equation 11 AUC of a perfect classifier

However, there are drawbacks in using the ROC and AUC, such as sensitivity to noise, ignores the relevance of Type I and II errors, and unreliable in unbalanced datasets [62], and these considerations must be taken into account when using this measure as a performance indicator.

Precision-Recall Curve

Many real-world applications such as digital forensics are dominated by an unbalanced amount of negative sets. For example, when searching for CSAM[1], the number of images without explicit content is higher than the number of target images. As such, the ROC curve is deemed inappropriate [56].

For highly skewed datasets, the cost curve or more formally, the precision-recall curve is recommended instead of the ROC curve [63]. The figure below is taken from the work of Davis and Goadrich [64] that discusses the relationship between and ROC and a PR curve.

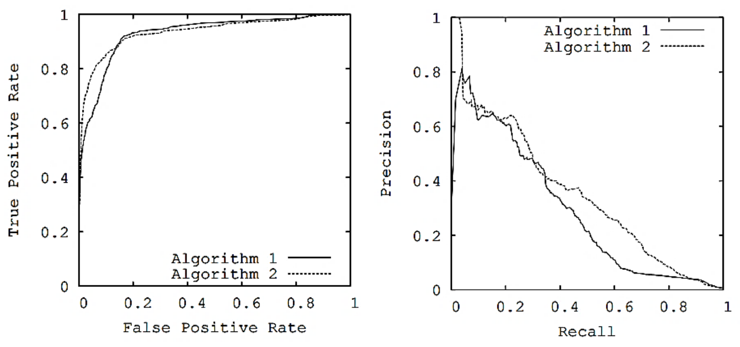


Figure 32 ROC curve (left) and PR curve (right) [64]

Taken from [64], Figure 32 shows that in an ROC curve (left), the differences between algorithms is not immediately apparent, but Algorithm 2 clearly performs better when plotted against the PR curve (right). The difference is due to a larger number of true negative observations compared to true positives. In the ROC curve, a huge increase in

[1] Child Sexual Abuse Material, CSAM

false positives does not necessary have a huge effect on the rate[1] as the true negative values are significantly larger. This scenario is common in many real world domains such as digital forensics. In a PR curve, instead of false positives, the comparison is done on the actual positives by using precision[2]; the performance difference now becomes apparent.

4.2.4.2 Models Used in this Study

As mentioned in the preceding sections, the computer vision models are taken from Keras[3]. Keras is an open-source neural network API written in Python and capable of running in open-source deep-learning toolkits such as TensorFlow[4], Microsoft's Cognitive Toolkit (CNTK) [5] and Theano[6]. This research study used the TensorFlow platform. The modularity, flexibility and user-friendly APIs of Keras allow for a painless and fast prototyping of machine learning models for research, experimentation or tool development. It also runs seamlessly with GPU and CPU deployments.

The Keras website lists a summary[7] of accuracy results for each of the 12-computer vision models available on the platform. We picked the classifiers with more than 90% top-5 accuracy. Tabulated in Table 6 are the reported classification accuracies for the four selected models (Xception, VGG16, ResNet50 and Inception) with published Top-1 and Top-5 values evaluated using the ImageNet validation set.

Table 6 Model Accuracy based on ImageNet Validation Set

Model	Size	Top-1 Accuracy	Top-5 Accuracy
Xception [19]	88 MB	0.79	0.945
VGG16 [21]	528 MB	0.713	0.901
ResNet50 [20]	98 MB	0.749	0.921
InceptionV3 [18]	92 MB	0.779	0.937

The accuracy values published for these models do not guarantee the same set of accuracies for gun classification using our dataset. In performance estimation, the

[1] FPR = FP / (FP + TN)
[2] Precision = TP / (TP + FP)
[3] https://keras.io/
[4] https://www.tensorflow.org/
[5] https://docs.microsoft.com/en-us/cognitive-toolkit/
[6] http://www.deeplearning.net/software/theano/
[7] https://keras.io/applications/#models-for-image-classification-with-weights-trained-on-imagenet

question, "*Does the published estimated accuracy of the selected models hold when evaluated and tested against our dataset*?*"* is answered during model evaluation and selection in Final Model: InceptionV3 Performance.

The classifiers for selection are described briefly below.

VGG16

This deep learning method is one of the first attempts of adding depth to improve classification accuracy. The best performance is achieved in 16-19 weight layers. In this study, VGG16 is evaluated instead of VGG19 because the accuracy published is relatively the same, 0.901 and 0.90 respectively, but VGG19 is more computationally expensive than its -16 counterpart. The full paper [21] is published online.

InceptionV3

While VGG achieved phenomenal accuracy on the ImageNet dataset compared to AlexNet's winning entry in 2012, it is resource-intensive. GoogLeNet, the first Inception version, devised an "inception" module optimising resource by growing "wider" instead of "deeper". The filters (*kernel*) of multiple sizes operate on the same level. This is also an answer to the problem of saliency[1] variation in which different objects will have various variations depending on the location of the object in the picture. A closeup of an image requires a larger *kernel* (global filter) than an object taken at the far end of the room (local filter). Choosing the correct *kernel* size to reflect either global or local variations to perform the convolution is not straightforward and the inception module was built to improve this bottleneck. The version 3 of this network added upgrades to the optimisation and regularization of the network to avoid overfitting. The full paper [18] is published online.

ResNet50

Yet another winner, this model secured the first place in both ImageNet and COCO competitions in 2015. This model introduced a deep residual training network ("Res" in ResNet is Residual) that showed that it is relatively easy to train residual networks of

[1] classification based on an object that is more pronounced than the rest

great depth. The paper also showed that gains in accuracy can be achieved by increasing the level of depth of the network. The full paper [20] is published online.

Xception

The Xception model is based on InceptionV3 model. Based on the ImageNet validation set, this model outperforms ResNet, InceptionV3 and VGGNet architecture. The full paper [19] is published online.

4.2.5 Model Selection

During the model evaluation phase, the four models from the Keras application are evaluated on the" WEAPON-DB3: Ugr.Es" dataset taken from [15]. This dataset contains 304 images of guns and 304 images of non-guns (faces, objects, etc). The model with the best performing metrics (*i.e.*, Recall, Precision, False Positive Rate and Matthew's Correlation Coefficient) is chosen. The InceptionV3 model scored the highest recall rate and therefore, is the model of choice (best model). The InceptionV3 is then evaluated through other datasets to identify if the recall rate remains consistent (around 100%) among the rest of the remaining datasets (final model). The final model is then used as the basis for the prototype. This process is summarised in Figure 33 and the full approach was described in the preceding Performance Estimation section.

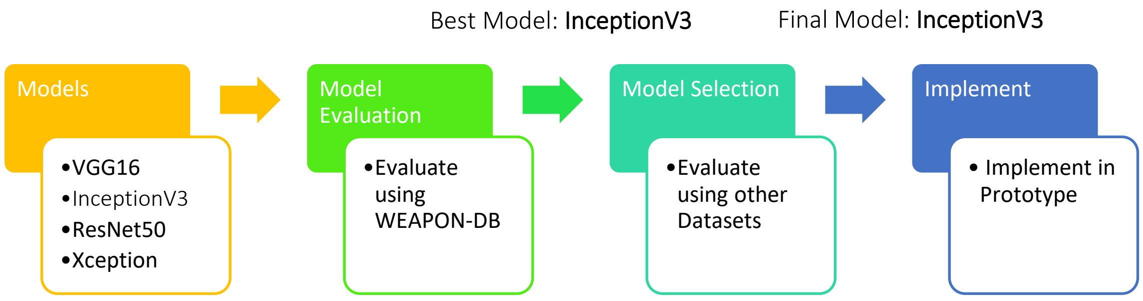


Figure 33 Model Selection Process

During the evaluation of the computer vision models, the main tool used is the Jupyter[1] notebook in an Anaconda-Python 3 environment with Keras and TensorFlow in the backend. Popular among data-science enthusiast, the Jupyter notebook lets users interact

[1] https://jupyter.org/

with the python code, save and load machine learning models, and run simulations and visualizations.

4.2.6 Prototype Development

The prototype is powered by Python 3 using Flask web development framework. The prototype uses web technologies because of the ease of which to build and customise powerful visualisations, make changes to the code, and host a working demo online for usability testing. Figure 34 shows the technical overview of the prototype architecture.

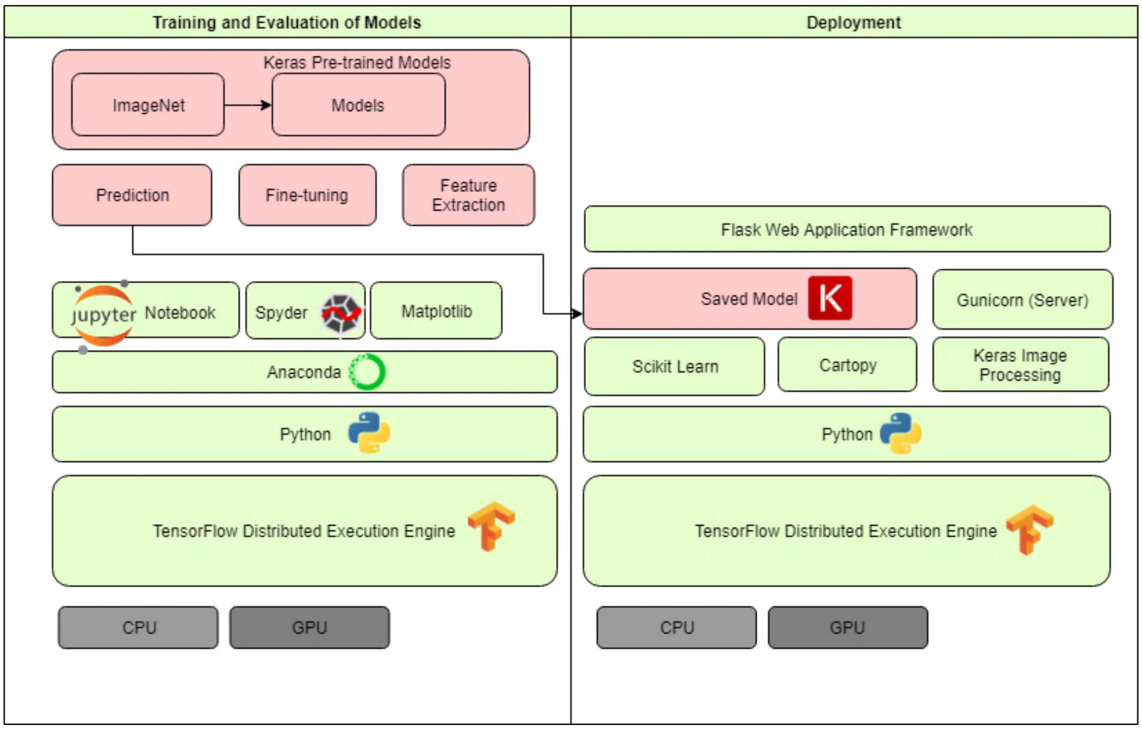


Figure 34 Technical Overview

Technologies Used

Anaconda[1] is a toolbox in Python and R programming languages that can perform data science and machine learning tasks in an easy way. It can manage libraries and dependencies, develop and train machine learning models using scikit-learn, and the TensorFlow backend, and visualise results using Matplotlib[2] and seaborn[3] visualisation libraries.

[1] https://www.anaconda.com
[2] https://matplotlib.org/
[3] https://seaborn.pydata.org/

Keras.io[1] was initially developed as a research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Operating System). Now a full-grown neural network library, it is written in Python with state-of-the-art deep learning models available for prediction, finetuning, and feature extraction. For this project, the ImageNet pre-trained library is used directly for prediction. The simple APIs and exhaustive documentation and tutorials make this deep learning framework a popular choice among data analyst and machine learning enthusiasts.

TensorFlow[2] is an end-to-end ecosystem for machine learning and deep learning created by Google's Brain Team. It uses Python as a convenient front-end API for creating and building applications and neural networks while executing those applications in the back-end using high performance C++ binaries. The TensorFlow architecture can run on any target distribution such as a CPU, GPU and even Android and iOS devices.

Flask[3] is a microframework (*i.e.* a minimalistic web application framework) written for Python. Since the prototype required in this research needs only the core functionalities of a web application without the extensions of most full-blown web frameworks such as database abstractions or server configurations, Flask is the ideal choice. *Gunicorn*[4] (Green Unicorn) is a Python Web Server Gateway Interface HTTP server for the UNIX environment that is light and fast, appropriate for development and prototyping activities. It is compatible with several web frameworks such as Flask, and in this project, the prototype runs in gunicorn.

[1] https://keras.io/models/about-keras-models/
[2] https://www.tensorflow.org/
[3] http://flask.pocoo.org/
[4] https://gunicorn.org/

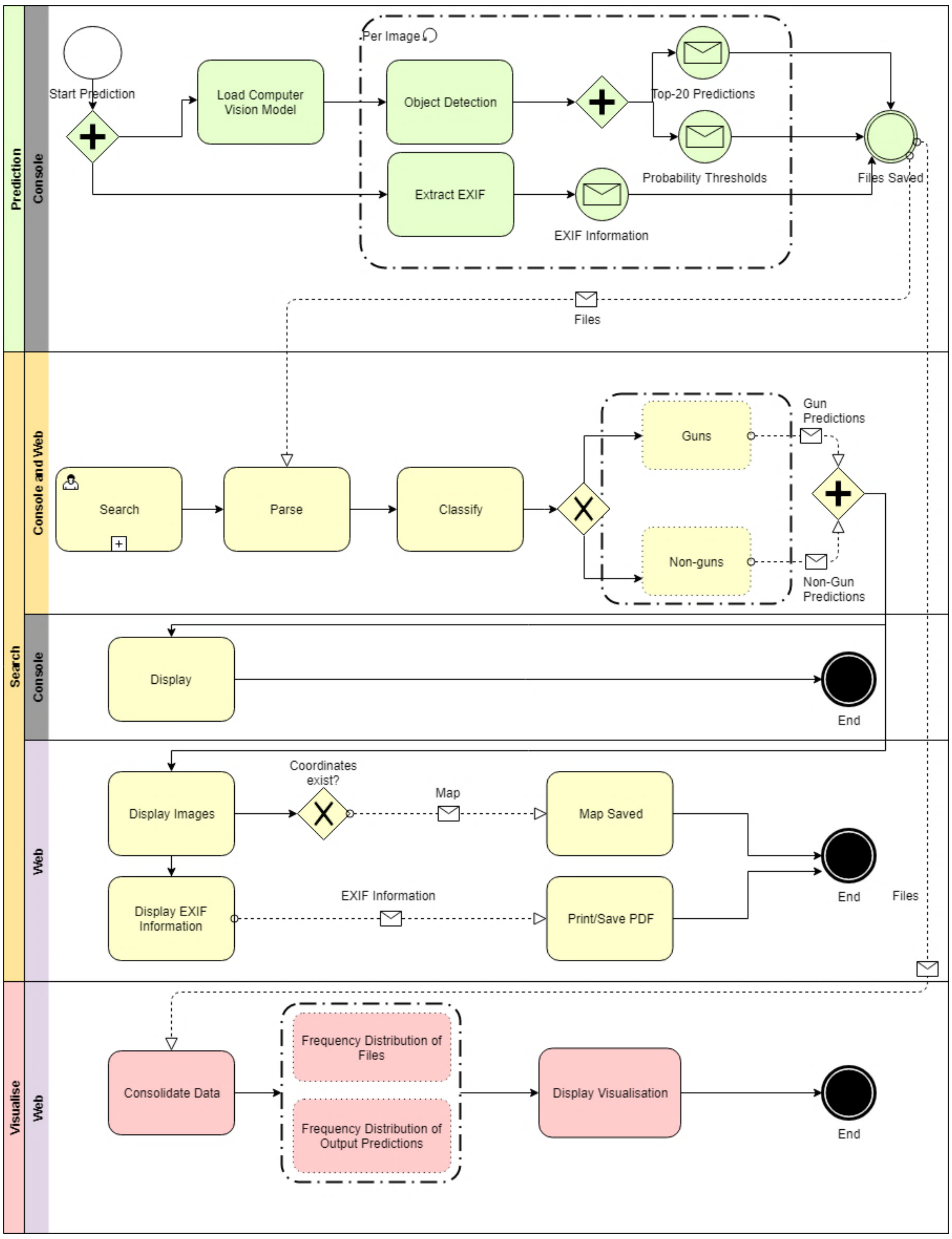
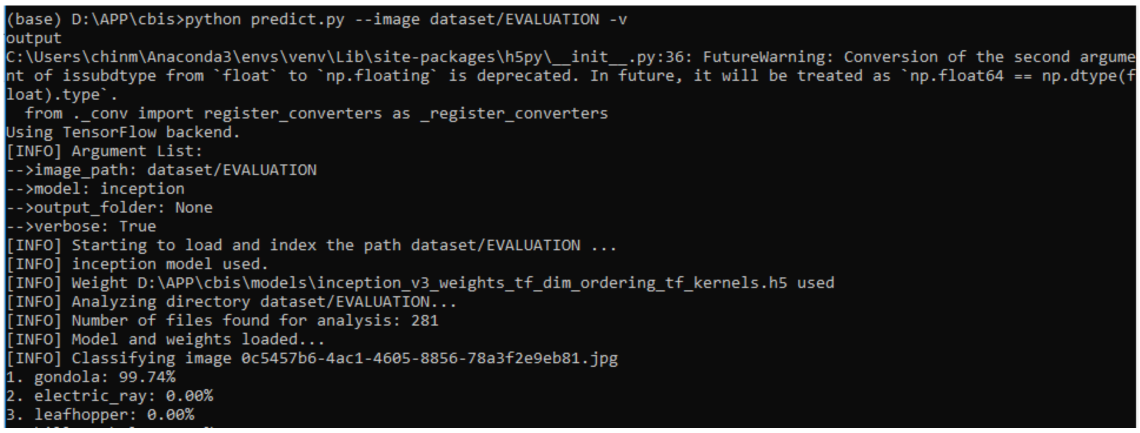


Figure 35 Process Diagram

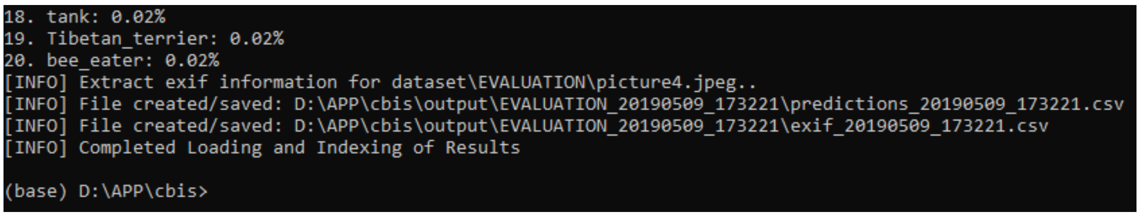
Figure 35 shows the process diagram for the prototype. There are three main functionalities of the prototype: prediction, search and visualization.

Prediction is the core function of the prototype. This is console-based where the directory of the images is supplied to the command prompt. It has 2 main modules: (1) prediction of output classes and the (2) extraction of EXIF data. During prediction, the desired computer vision model is loaded (see Figure 36). This part of the code can be easily modified to allow other neural network models to be used instead of InceptionV3 (the final model chosen in the study). The model predicts each image from the supplied directory including the probabilities for each output class. A CSV file is generated at the end of the process. In the extraction of EXIF data, each image is subjected to an EXIF extractor module; an output file is also generated (this can be seen in Figure 37).



```
(base) D:\APP\cbis>python predict.py --image dataset/EVALUATION -v
output
C:\Users\chinm\Anaconda3\envs\venv\Lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argume
nt of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(f
loat).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
[INFO] Argument List:
-->image_path: dataset/EVALUATION
-->model: inception
-->output_folder: None
-->verbose: True
[INFO] Starting to load and index the path dataset/EVALUATION ...
[INFO] inception model used.
[INFO] Weight D:\APP\cbis\models\inception_v3_weights_tf_dim_ordering_tf_kernels.h5 used
[INFO] Analyzing directory dataset/EVALUATION...
[INFO] Number of files found for analysis: 281
[INFO] Model and weights loaded...
[INFO] Classifying image 0c5457b6-4ac1-4605-8856-78a3f2e9eb81.jpg
1. gondola: 99.74%
2. electric_ray: 0.00%
3. leafhopper: 0.00%
```

Figure 36 Prediction Module - Loading of Model

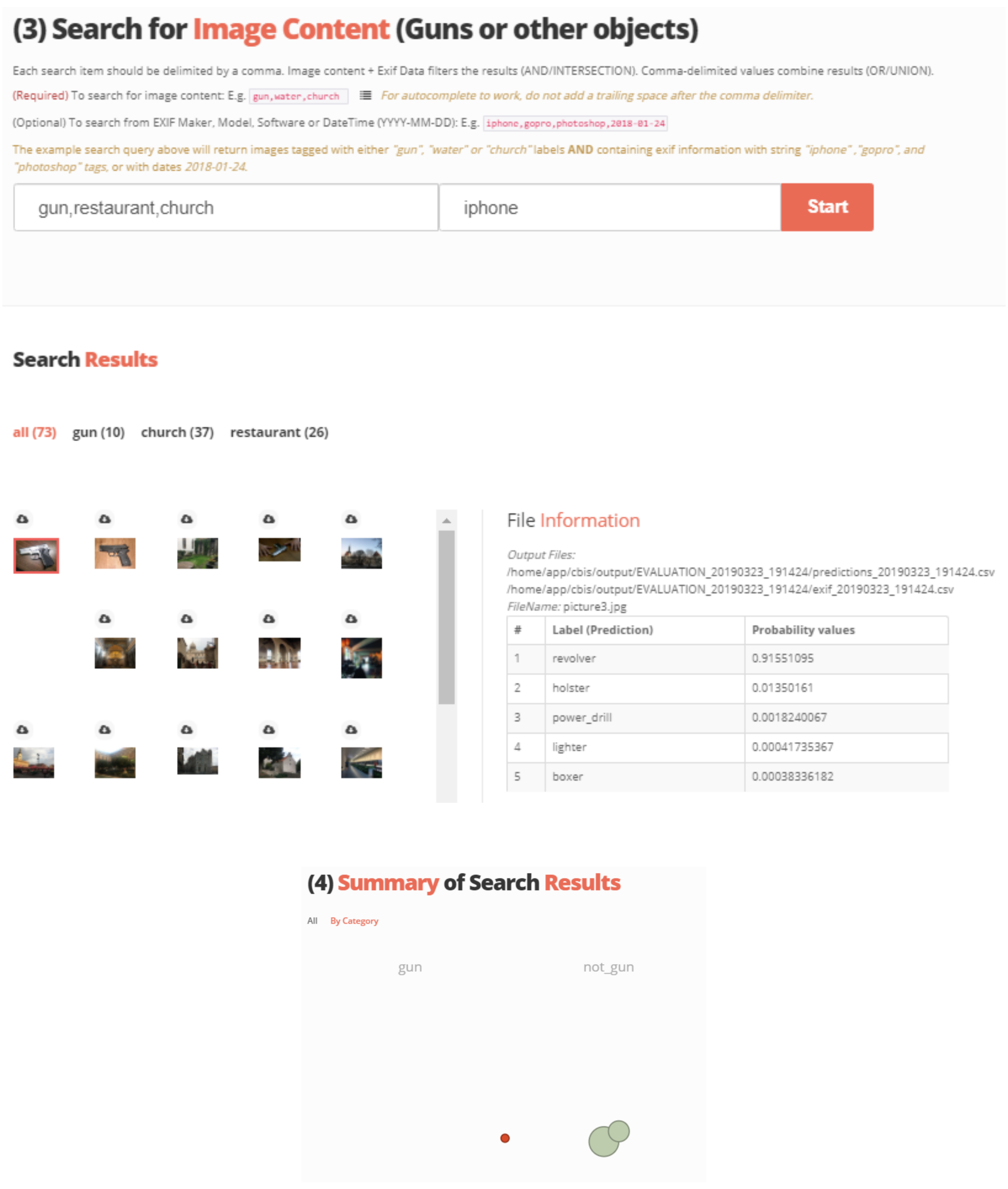


```
18. tank: 0.02%
19. Tibetan_terrier: 0.02%
20. bee_eater: 0.02%
[INFO] Extract exif information for dataset\EVALUATION\picture4.jpeg..
[INFO] File created/saved: D:\APP\cbis\output\EVALUATION_20190509_173221\predictions_20190509_173221.csv
[INFO] File created/saved: D:\APP\cbis\output\EVALUATION_20190509_173221\exif_20190509_173221.csv
[INFO] Completed Loading and Indexing of Results

(base) D:\APP\cbis>
```

Figure 37 Prediction Module – Saving the Results

The prototype's main user function is the *search*, which deals with returning the images based on the keywords searched by the user. The CSV files generated by the prediction function is parsed for content, top-*k* predictions, probability thresholds and EXIF information (Maker, Model, Software, and Date created) based on the keyword search by the user (see “Search for Image Content” in Figure 38). Additional classification is performed for gun and non-gun content from the web interface (see “Summary of Search Results” in Figure 38). For the console, only textual information is displayed (see Figure 39).



(3) Search for Image Content (Guns or other objects)

Each search item should be delimited by a comma. Image content + Exif Data filters the results (AND/INTERSECTION). Comma-delimited values combine results (OR/UNION).

(Required) To search for image content: E.g. `gun,water,church` *For autocomplete to work, do not add a trailing space after the comma delimiter.*

(Optional) To search from EXIF Maker, Model, Software or DateTime (YYYY-MM-DD): E.g. `iphone,gopro,photoshop,2018-01-24`

The example search query above will return images tagged with either *"gun", "water" or "church"* labels **AND** containing exif information with string *"iphone", "gopro", and "photoshop" tags,* or with dates *2018-01-24.*

Search Results

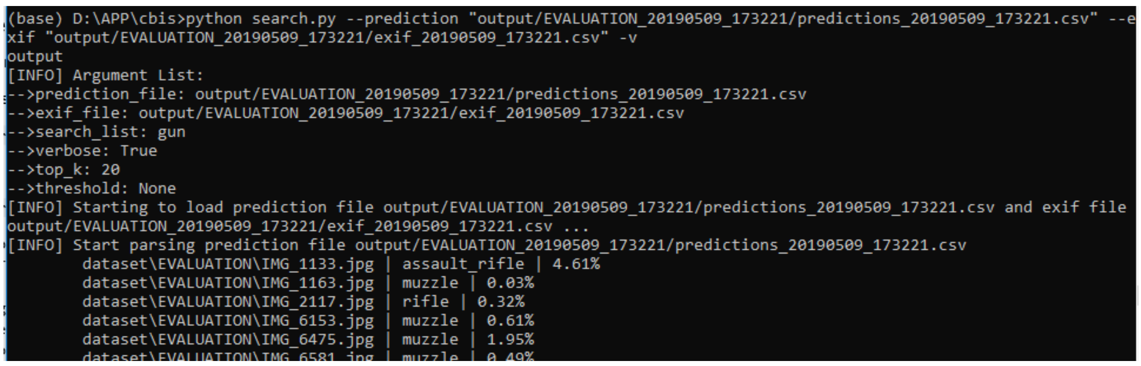
File Information

Output Files:
/home/app/cbis/output/EVALUATION_20190323_191424/predictions_20190323_191424.csv
/home/app/cbis/output/EVALUATION_20190323_191424/exif_20190323_191424.csv
FileName: picture3.jpg

#	Label (Prediction)	Probability values
1	revolver	0.91551095
2	holster	0.01350161
3	power_drill	0.0018240067
4	lighter	0.00041735367
5	boxer	0.00038336182

(4) Summary of Search Results

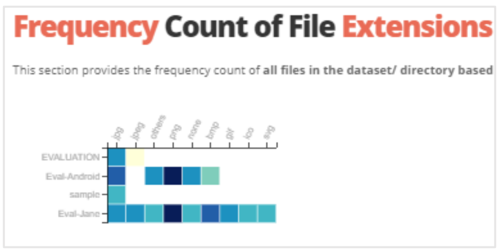
Figure 38 Search - Web Interface



```
(base) D:\APP\cbis>python search.py --prediction "output/EVALUATION_20190509_173221/predictions_20190509_173221.csv" --e
xif "output/EVALUATION_20190509_173221/exif_20190509_173221.csv" -v
output
[INFO] Argument List:
-->prediction_file: output/EVALUATION_20190509_173221/predictions_20190509_173221.csv
-->exif_file: output/EVALUATION_20190509_173221/exif_20190509_173221.csv
-->search_list: gun
-->verbose: True
-->top_k: 20
-->threshold: None
[INFO] Starting to load prediction file output/EVALUATION_20190509_173221/predictions_20190509_173221.csv and exif file
output/EVALUATION_20190509_173221/exif_20190509_173221.csv ...
[INFO] Start parsing prediction file output/EVALUATION_20190509_173221/predictions_20190509_173221.csv
        dataset\EVALUATION\IMG_1133.jpg | assault_rifle | 4.61%
        dataset\EVALUATION\IMG_1163.jpg | muzzle | 0.03%
        dataset\EVALUATION\IMG_2117.jpg | rifle | 0.32%
        dataset\EVALUATION\IMG_6153.jpg | muzzle | 0.61%
        dataset\EVALUATION\IMG_6475.jpg | muzzle | 1.95%
        dataset\EVALUATION\IMG_6581.jpg | muzzle | 0.49%
```

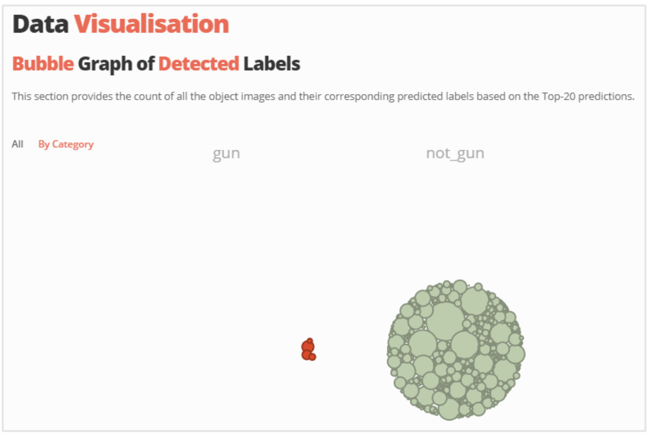
Figure 39 Search - Console

For the *visualization* module, this is only available from the web interface. Parsed from the CSV files from the prediction module, the data is consolidated to display a summarized visual form of the data such as a Frequency Count and a Bubble Graph in Figure 40 and Figure 41 respectively.


Frequency Count of File Extensions

This section provides the frequency count of **all files in the dataset/ directory based**

Figure 40 Visualisation - Frequency Count


Data Visualisation

Bubble Graph of Detected Labels

This section provides the count of all the object images and their corresponding predicted labels based on the Top-20 predictions.

Figure 41 Visualisation - Bubble Chart

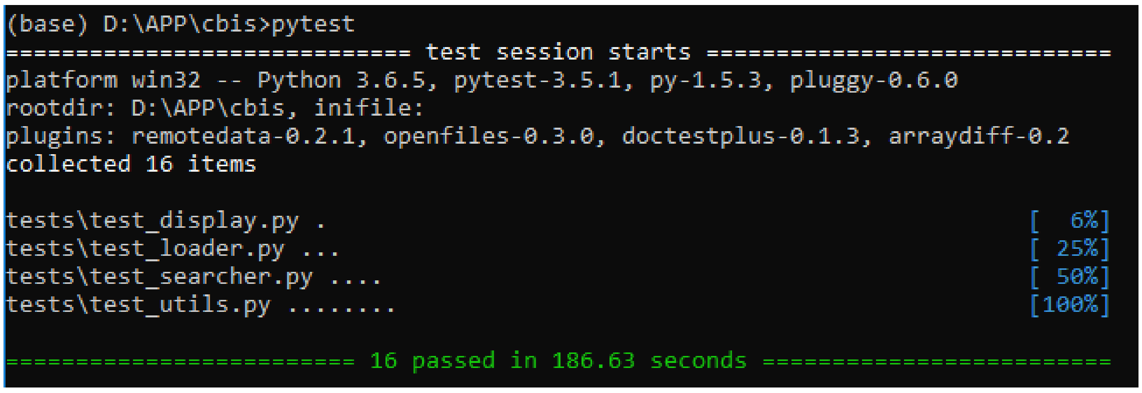
The functionalities: prediction, search and visualization, cover the minimum requirements specified during the discussion (see Problem Identification and Requirements). The Requirements and the implementations are tabulated for clarity in Table 7.

Table 7 Requirements Implemented

Requirements	**Status**	
Open-source tool that can be customized, modified and improved	Implemented	Developed in Python and released in GitHub
Automated categorization of images	Implemented	Search for guns and other image objects
Metadata extraction and maps	Implemented	EXIF extraction
File Export	Implemented	CSV file output for predictions and EXIF data
Visualization	Implemented	Visual graphs from web application
Weapon categorization	Implemented	Image categorization using the best computer vision model during selection

Unit testing

Unit testing was done using Python's built-in *pytest* module. The classic code-and-test approach is employed in the development.



```
(base) D:\APP\cbis>pytest
============================ test session starts =============================
platform win32 -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: D:\APP\cbis, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 16 items

tests\test_display.py .                                                  [  6%]
tests\test_loader.py ...                                                 [ 25%]
tests\test_searcher.py ....                                              [ 50%]
tests\test_utils.py ........                                             [100%]

========================= 16 passed in 186.63 seconds =========================
```

Figure 42 Unit Testing - PyTest

4.3 Demonstration and Testing

Although the instructions and evaluation forms are hosted online, a demonstration of the working prototype was provided prior to commencing the test. Feedback is received via email or face-to-face as some respondents readily gave their insights regarding the tool right after testing was completed.

Part of this research study measures the overall perceived usability of the resulting prototype in terms of usability, learnability and efficiency with a series of tasks in which the forensic users and potential users attempt to solve. The tasks are designed to simulate questions that might arise from some investigative work such as: "*The suspect uploaded a photo in Instagram of a restaurant in India on July 25, 2017 in the afternoon. Investigators speculated that he stayed the night in this area. Where do you think the suspect could have stayed?*"

The perceived usability of the respondents is taken from the overall mean score of the System Usability Scale (SUS) test. Learnability is taken from item-level results of the same test while efficiency is measured by the speed of which tasks are completed.

4.3.1 Design of Usability Test

The test is designed as a classic test-and-measure approach. A series of tasks (tests) were designed to measure the following research questions:

1) *Usability* – Is the prototype usable in a forensic investigation?
2) *Learnability* – Can the prototype be used with ease?
3) *Efficiency* – Can the users perform the task efficiently?

A properly designed usability test ensures that the above measures are covered. To measure *efficiency*, the speed of completing the tasks is measured before and after using the tool. Usability and learnability are measured using the System Usability Scale (SUS) test [65] [66].

Usability testing has been a well-established evaluation method that measures the perceived usability of users or potential users of a system, device or application [67]. According to [68], valid usability tests have been mentioned to have six characteristics: (1) usability as the focus, (2) respondents are end-users or potential end-users, (3) there must be a system or a tool to evaluate, (4) performance of tasks by respondents, (5) responses are recorded and analysed, and (6) the results are communicated.

The following sections explains how these characteristics are implemented in this study.

4.3.1.1 Focus on Usability

The evaluation test conducted online is divided into four sections in a test-and-measure paradigm. It tests for the usability of the system and some of its attributes such as learnability.

4.3.1.2 End-Users

A valid usability test includes respondents who are part of the target market for the tool. A user profile of relevant characteristics should be built prior to the release of the evaluation form as suggested in [68]. The objective of a user profile is to ensure that the range of users targeted for the tool are clearly defined. This provides insight and understand to whom this prototype is built for. The user profile for this research study is described in Table 8.

Table 8 User Profile

User Category	User Profile
Student	Digital Forensic Students Taken the Systems Forensics Course in Taltech in Spring 2018 Experience Level: 0 - 2 years Location: Tallinn Technical University
Entry Level Users	Forensic Examiners Experience Level: 1 – 5 years Location: Anywhere
Mid-Level Users	Forensic Examiners Experience Level: 5 – 10 years Location: Anywhere
Senior Users	Forensic Examiners Experience Level: 10+ years Location: Anywhere

Although the evaluation form, instructions and codes are online and no personal information are gathered, due diligence is performed in ensuring that only examiners and potential examiners fitting the user profile are accepted.

4.3.1.3 Product for Evaluation

Prototypes can be categorised as low-fidelity or high-fidelity prototypes. Fidelity encompasses the breadth of features, the degree or complexity of the functionality implemented, or the style of interaction. Low-fidelity prototyping could be a pen-and-paper approach or a static HTML mock-up while a high-fidelity prototype can be a running demo website. In this study, a high-fidelity testing is performed as the range of functionalities available are mature and almost *complete*. Completeness of functionality implies that the requirements gathered during the elicitation phase had been implemented (refer to Table 7 for the list of requirements and the corresponding functionalities implemented).

4.3.1.4 Performance of Tasks

It is important to emphasize that the tasks probe the usability of the prototype and not the skill of the respondent. Hence, our interaction with respondents are crucial in ensuring the tasks are well understood and feedback is timely. The tasks tap into the core functionalities of the tool: categorisation of image (prediction), detection of guns (search), and visualisation. The complete list of questions is found in Appendix 2 – Evaluation Questions and Answer Key.

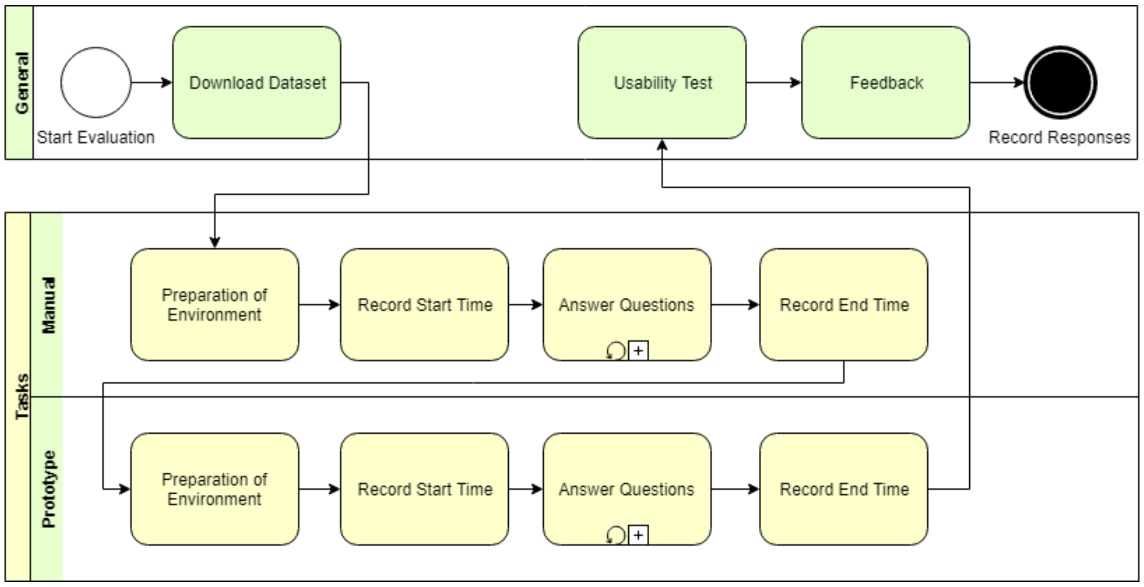


Figure 43 Evaluation Flow

Figure 43 shows how the test is designed. The respondent is requested to download the dataset (data collection) and prepare the environment before proceeding with the tasks. The time to complete each task is recorded manually. After answering the set of questions, a 10-question usability test is presented including space for the respondent to fill-in the feedback. The resource links that are supplied to the respondents are given in Table 9.

To explain the process flow in detail:

(1) Data Collection: Respondents are asked to download three datasets (Evaluation, Android and Jane) for evaluation. The datasets folders contain random gun images with manufactured EXIF information.

(2) Preparation of Environment for Manual Categorisation: The first part of the test is to evaluate the dataset using manual or existing tools that the respondents have already been using such as Window Explorer or a downloadable EXIF tool extractor [55] from Phil Harvey. They had to download the necessary tools before proceeding to answer the tasks.

(3) Recording of Time: Respondents are asked to record the start and end time to complete the tasks.

(4) Answer Questions: Respondents answer the questions without using the prototype and using the tools in step (2).

(5) Preparation of Environment for Prototype Use: This requires that the respondents answer the same set of questions using the prototype. There were three ways to gain access to the prototype:

> (a) installation from source code - a guide to install in Ubuntu 18.04.1 LTS fresh machine is provided,
> (b) use of a portable VirtualBox virtual machine (~.ova file) with pre-installed dependencies and pre-loaded GitHub source code, or
> (c) use of the demonstration website online

(6) Recording of Time: Respondents are asked to record the start and end time to complete the tasks using the prototype.

(7) Answer Questions: The respondents are asked to answer the same set of questions using the prototype.

(8) Answer Usability Questions: The respondents answer the Usability test using the System Usability Scale (SUS) approach. This is explained in detail in the System Usability Scale (SUS) section.

(9) Feedback: Feedback is gathered at the end of the test via email, form or face-to-face.

Table 9 Resource Links during Evaluation

Description		Links
Evaluation Instructions	Google Doc	http://bit.ly/2WmLAf6
How to Use the Prototype	Online	https://cbis.jrdelmar.dev/help
Source Code	GitHub	https://github.com/jrdelmar/cbis
Dataset	GitHub	https://github.com/jrdelmar/cbis-dataset
VirtualBox virtual machine	One drive	http://bit.ly/2HFXnkv
Demo Website	Online	https://cbis.jrdelmar.dev/
Evaluation Tasks and Usability Test	Google Form	https://forms.gle/pbZyky4XBSCVRTc9A

Table 9 lists the online resource links released for evaluation by potential forensic examiners and digital forensic professionals.

4.3.1.5 Recording and Communication of Responses

The forms are created via Google Forms; therefore, responses are stored after the respondents click the submit button. The responses are then analysed and communicated in the Results and Discussion section of this paper.

4.3.2 System Usability Scale (SUS)

The usability questionnaire used the System Usability Scale (SUS) method. SUS is a well-known questionnaire in usability research for various kinds of applications such as websites, devices, systems, hardware, software and tools. Developed by John Brooke in 1986 [69], it has demonstrated validity and reliability and has now become an industry standard with references in over a thousand articles and publications [67]. Even with small sample sizes, even recommending only three to five users [70] for early-phase usability tests, SUS can provide reliable results.

This study can be considered as an early-phase usability test; therefore, five respondents are deemed enough to perform preliminary analysis.

The SUS questionnaire consists of ten five-level Likert-scale questions, half worded positively, and half worded negatively. The respondents are then asked to score these items in a scale of 1 – 5 from Strongly Agree to Strongly Disagree, with 3 as Neutral. The typical SUS questionnaire has standardised questions, but this study had to revise some of the statements for clarity.

4.3.2.1 SUS Questions

The list of SUS questions including the changes against the standard questions (italicised for emphasis) are shown:

1. I think that I would like to use this system frequently if I have to deal with categorizing images and getting EXIF information.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need *assistance (technical or non-technical)* to be able to use this system.
5. I found the various functions (console prediction, search, display EXIF information, report screen) in this system were well integrated.
6. I thought there was too much inconsistency in this system. It would be difficult to deploy the tool in the real world.
7. I *think* that most *forensic examiners* would learn to use this system very quickly.
8. I found the *tool inconvenient* to use.
9. I felt very confident using the *tool*.
10. The learning curve to use this tool is steep. I needed to learn a lot of things before I could use the tool.

4.3.2.2 Interpreting scores

Overall Mean SUS Score

The scoring for SUS is not straightforward. Each question is converted into a new number; the score for each odd number is deducted by 1 point (x – 1) while the score for each even number will be deducted from 5 points (5 – x). The minimum score is 0 while the maximum score is 100, but this does not convey percentages, only percentile ranking. This unit-less measurement does not convey any meaning, but some studies had proposed an adjective rating scale (Worst imaginable, Awful, Poor, Ok, Good, Excellent, Best Imaginable) by Bangor et.al [65] and letter grades (A to F) by Lewis and Sauro [66].

Figure 44 is taken from [65] and illustrates the proposal from Bangor *et al.*; it is shown that the acceptable SUS scores are 70 and above which is the upper 25% of the distribution.

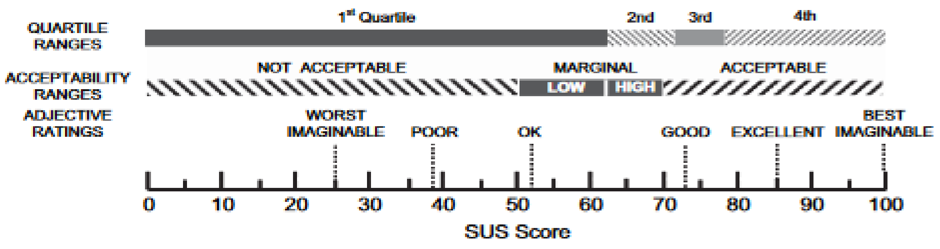


Figure 44 Adjective-based scoring from Bangor *et al.*

However, the more recent study of Lewis and Sauro [66] suggests that the widely adopted adjective-adjusted scoring system by Bangor *et al.* [65] is inaccurate and the curved method based on a linear regression model is proposed. They showed that in 241 industrial usability studies, the published norms for overall SUS mean scores have an average grade of 68 and yet "it is becoming a common industrial goal to achieve a SUS of 80 as evidence of an above average user experience." [66]

Figure 45 is taken from [71] and illustrates the revised scoring method proposed in [66]. The overall SUS average of 68 means 68% of the maximum score is reached but it belongs to the 50th percentile with a grade of C.

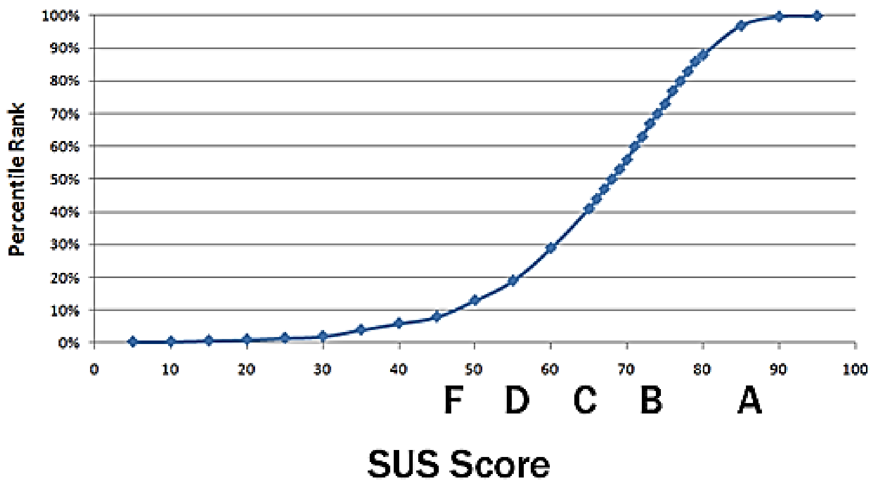


Figure 45 The *curved* scoring function

Overall SUS Benchmark

The average SUS rating of 68 and its corresponding letter grade of “C” is the benchmark applied in this study. This benchmark determines the minimum acceptable level of perceived usability of the prototype.

Individual Item Scores

Along with the curved scoring system, item-level benchmarking from individual SUS questionnaires was introduced in the same paper by Lewis and Sauro [66]. The paper proposed item-level benchmarking to measure other user experience attributes such as: perceived complexity (question 2), perceived ease-of-use (question 3), perceived consistency (question 6), perceived learnability (question 4, 10) [1], and confidence-in-use (question 9). The questions can be found in the preceding section: SUS Questions. In [72], Sauro proposed a two-factor assessment of SUS using Usability and Learnability. Sauro [71] graphed learnability versus usability using learnability scores for questions 4 and 10 and usability – for the rest of the 8 questions. Thus, to answer the research question on usability and learnability, the same approach is employed.

Item-level SUS Benchmark

The same benchmark for the overall SUS rating is adopted for item-level scores. That is, the average SUS rating of 68 and its corresponding letter grade of “C” is the benchmark applied in this study.

[1] Question 7 (“I would imagine that most people would learn to use this system very quickly”) is not considered because it refers to other people’s skills and not the user’s [72]

5 Results and Discussion

5.1 Model Selection

There are two kinds of performance metrics that are reviewed in the Related Works section of this paper: (1) metrics to evaluate classifiers as digital forensic tool, and (2) metrics to evaluate gun classifiers. It is apparent that there is no standard consensus on the minimum performance requirement of a classifier. Thus, the findings are not evaluated against any baseline but only among the results achieved. The baseline throughout the study serves only as a sanity-check as suggested in [73].

The use of accuracy alone in evaluating performance is insufficient and even misleading. Class-specific metrics common across the reviewed literature are: Accuracy, Precision and Recall. However, this study also considers other metrics such as Matthew's Correlation Coefficient and False Positive Rate.

Recall Figure 20 in Performance Estimation section; the first part of the process diagram presents the model selection and evaluation process and is explored in more depth in the next section.

5.1.1 Model Evaluation

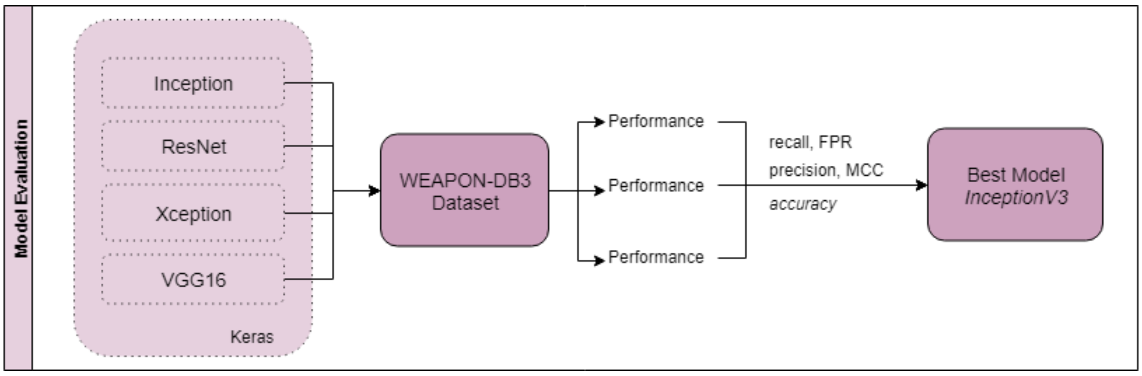
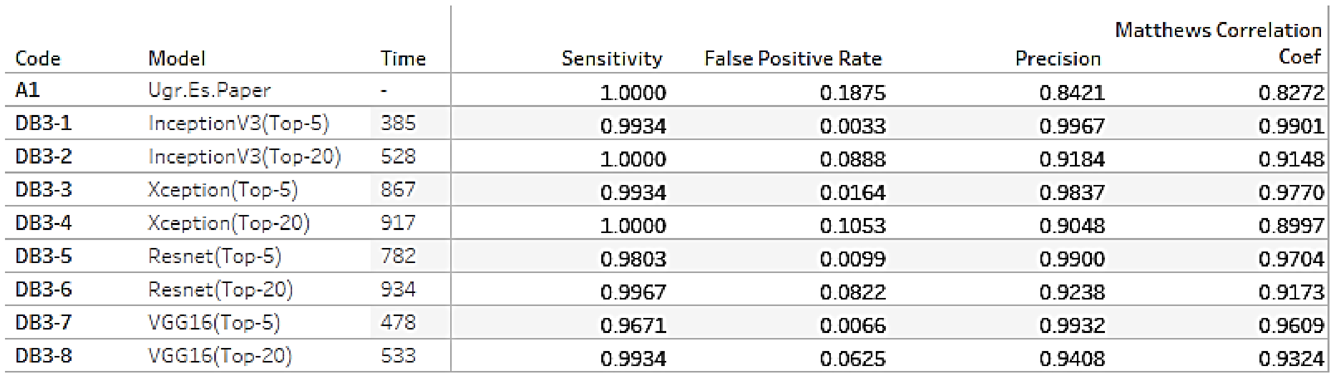


Figure 46 Model Evaluation Approach

Figure 46 shows the model evaluation process employed in this study. The four models: *InceptionV3*, *Xception*, *Resnet*, and *VGGG16* are evaluated using the dataset used in Olmos *et al.*'s Handgun detection paper [15]. There are two advantages of using an existing dataset. One is to remove any possibility of error that we could have introduced due to misclassification introduced during ground truth labelling; second, to have a baseline.

For selecting the best model among the four models in the list, the following metrics are evaluated: Sensitivity(recall), False positive rate (1 - specificity), Precision, Matthew's correlation coefficient (MCC) and Processing Time (in seconds per image). Relegating accuracy over recall during model selection implies that this study prioritises models that can detect more guns (ideally all guns, no false negatives) rather than accuracy which takes all predicted values (false positives, false negatives) into account. The model is willing to tolerate mistakes in detecting a non-gun as a gun (false positives, Type I error) than detecting a gun as a non-gun (false negatives, Type II error).

Table 10 shows the complete results of the four selected models during evaluation. Top-5 indicates that only the first five output labels are considered when tabulating the confusion matrix while Top-20 takes into account all the output labels provided by the model (the models output a maximum number of 20 labels per image). Probability thresholds that filter labels (*e.g.* more than 35% probability) are not introduced here. Time is measured in seconds. Sensitivity, precision and MCC have ideal values of 1.00 while false positive rate has an ideal value of 0.



Code	Model	Time	Sensitivity	False Positive Rate	Precision	Matthews Correlation Coef
A1	Ugr.Es.Paper	-	1.0000	0.1875	0.8421	0.8272
DB3-1	InceptionV3(Top-5)	385	0.9934	0.0033	0.9967	0.9901
DB3-2	InceptionV3(Top-20)	528	1.0000	0.0888	0.9184	0.9148
DB3-3	Xception(Top-5)	867	0.9934	0.0164	0.9837	0.9770
DB3-4	Xception(Top-20)	917	1.0000	0.1053	0.9048	0.8997
DB3-5	Resnet(Top-5)	782	0.9803	0.0099	0.9900	0.9704
DB3-6	Resnet(Top-20)	934	0.9967	0.0822	0.9238	0.9173
DB3-7	VGG16(Top-5)	478	0.9671	0.0066	0.9932	0.9609
DB3-8	VGG16(Top-20)	533	0.9934	0.0625	0.9408	0.9324

Table 10 Cross-tabulated results for selecting the model

5.1.1.1 Recall versus False Positive Rate

To visualise the trade-off between sensitivity (recall) and specificity for each model, we created a Recall versus False Positive Rate (1 – Specificity) graph as shown in Figure 47.

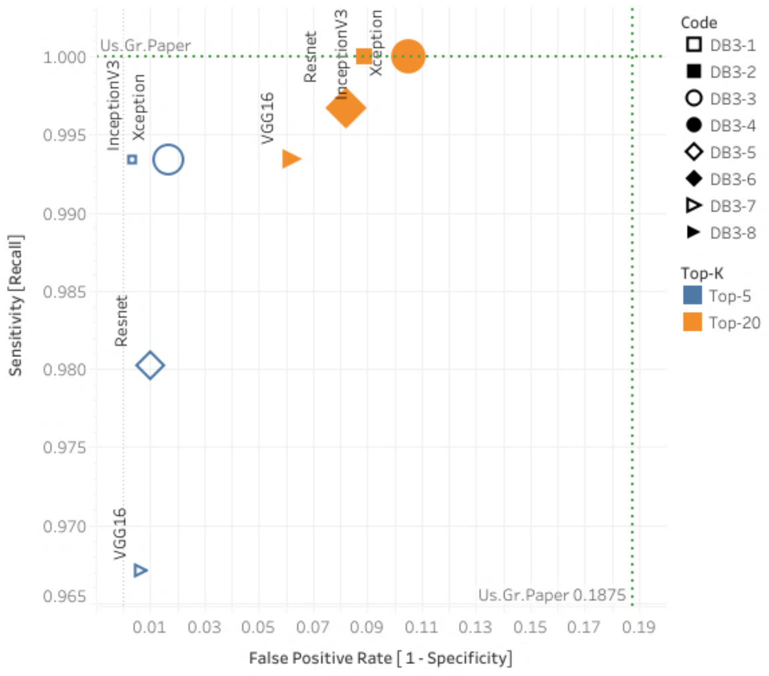


Figure 47 Model Evaluation: Recall vs False Positive Rate on WEAPON-DB3 dataset

The ROC graph in Figure 47 shows that the metrics from Olmos *et al.*'s paper [15] is used as the baseline and indicated by the dashed lines with text "Us.Gr.Paper". The baseline values are 0.1875 and 1.0 for false positive rate and recall, respectively.

The colour and filling indicate the Top-K predictions when performing the predictions. Blue unfilled markers are the Top-5 while orange filled markers are Top-20. Most of the Top-5 predictions have lower false positives, they can filter out most of the Type I errors at the expense of lower recall rate. Top-20 predictions, on the other hand, achieve higher and even perfect recall (as in the case of Inception and Xception models) at the expense of false positives.

The size of the markers is indicative of the processing time measured during prediction, that is, the time to complete the prediction divided by the number of pictures processed. To measure time, we used Python's `%timeit` function. This method is recommended when timing functions for benchmarking programs as it abstracts the intricacies of the different timing functions in Python.

Recalling Figure 30 in Visualising Trade-offs section, the ideal area should be closer to coordinates (0,1) which indicates a lower false positive detection rate but a higher recall rate. As seen from Figure 47, the InceptionV3 and Xception models for the Top-20 predictions have 100% recall rate, and therefore are the only choices for further consideration. The result is not surprising as Xception is based on the Inception architecture. Table 10 shows that InceptionV3 has the lowest false positive rate score, higher MCC value and faster in processing speed than Xception for the WEAPON-DB3 dataset.

5.1.1.2 Total Processing Time

Figure 48 shows the total time the function completes processing each dataset. Here, InceptionV3 model for the Top-20 is faster by almost seven minutes for the same set of 608 (gun and not gun) images. For an estimated 10,000 images, the InceptionV3 model would take 2.5 hours while Xception will take 4 hours in an x-64-based Intel Core i5 2-core CPU with 16-GB RAM.

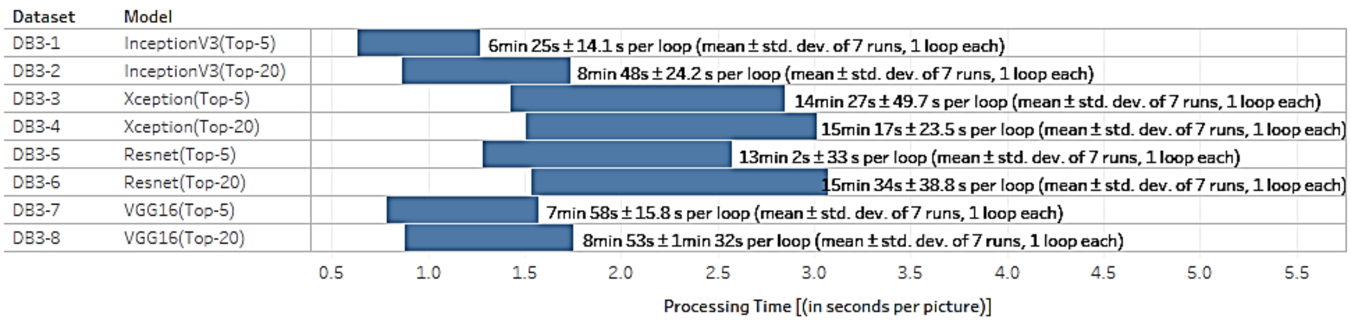


Figure 48 Total Completion Time

5.1.1.3 Precision and Recall

In the Precision-Recall or PR Graph, the ideal spot would be at the coordinates (1,1) for the maximum value of precision and recall rate. Figure 49 shows that InceptionV3 outperforms Xception for this dataset.

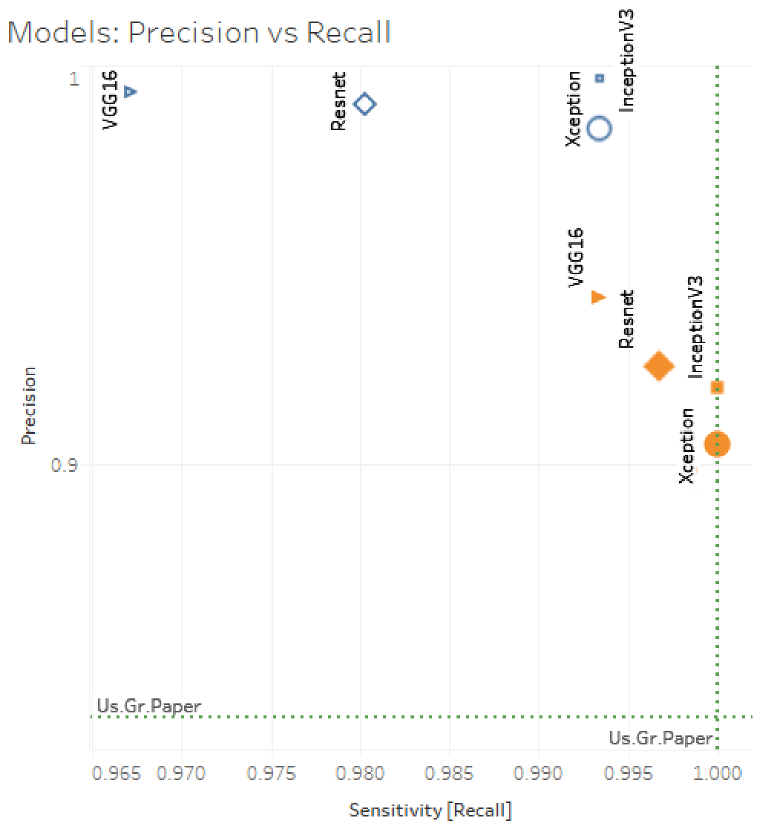


Figure 49 Precision and Recall for models tested in WEAPON-DB3 dataset

5.1.1.4 Accuracy among Models

In this study, the accuracy metric is not chosen as a criterion in model selection since in forensics, the recall rate holds more importance. However, it played a role in selection since two of the models achieved a recall rate of 100%.

Figure 50 shows the accuracy values of Top-5 Published, Top-5 and Top-20 Gun Evaluation for each model. *Top-5 Published* are accuracy values taken from their published papers (see Models Used in this Study section) and are tested against the ImageNet validation set. This is only used as a benchmark. Top-5 and Top-20 are the accuracy values achieved when running the prediction models on the WEAPON-DB3 dataset.

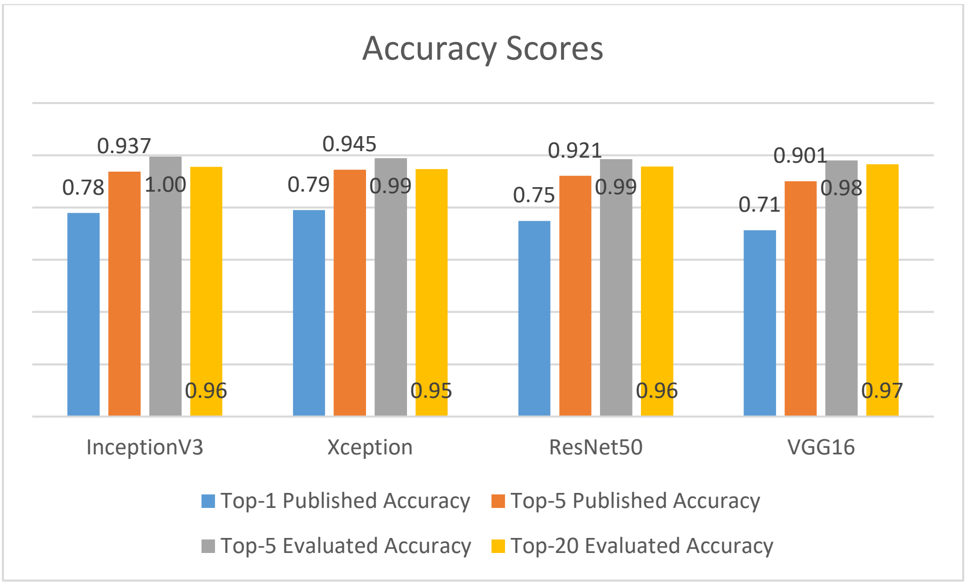


Figure 50 Accuracy Scores

Interestingly, although the InceptionV3 model only ranks second to Xception in the ImageNet validation set, it scores the highest among the four models in the Top-5 gun category. The Top-20 accuracy ratings of both InceptionV3 and Xception drop significantly compared to ResNet and VGG16 due to the increased number of false positives. The accuracy rating for InceptionV3 for both Top-5 and Top 20 gun predictions tested on the WEAPON-DB3 dataset is higher than the Xception model.

5.1.1.5 Ranking of Results

To decide the best model, the models are ranked against the metrics. Table 11 summarises the ranking results. A checkmark indicates that the model evaluated has the highest score.

Table 11 Model Ranking based on Results

Model	**Recall***	**Accuracy (Top-5)**	**Precision**	**False Positive Rate**	**Processing Time**	**MCC**
InceptionV3	1 ✓	1✓	3	3	1 ✓	4
Xception	1 ✓	2	4	4	3	3
ResNet	2	3	2	2	4	2
VGG16	3	4	1	1	2	1

* takes precedence over the other metrics

Table 12 Decision Matrix

Model	Recall*	False Positive Rate	Precision	Proc Time	Accuracy	MCC	Results P * Rank	Decision
Priority Ranking (P)	1	2	3	4	5	4		
InceptionV3	1	3	3	1	1	4	41	✓
Xception	1	4	4	3	2	3	55	
ResNet	2	2	2	4	3	2	51	
VGG16	3	1	1	2	4	1	40	

* takes precedence over the other metrics

Table 12 shows the decision matrix based on the ranks and priorities set out for this study. The priority rankings are based on our interpretation of the requirements during the identification stage. The priority rankings are multiplied by the individual rank scores to get the total results. The Recall metric takes precedence among any other metric. The other metrics (FPR, Precision, Processing Time, Accuracy, MCC) are considered in the decision making process after the recall pre-requisite is satisfied. Thus, only InceptionV3 and Xception vie for the best model. The lowest value is taken; in this case, InceptionV3 is the choice.

It can be inferred that the published accuracy of computer vision models, although tested in a multi-class setting such as ImageNet, is insufficient grounds to assess its performance when used for a different domain and dataset. Testing the models again on some dataset, as this study has done, showed that sometimes a simpler model might offer better results. Therefore, the best model selected to be used for the prototype is the computer vision model: *InceptionV3*.

5.1.1.6 InceptionV3's Accuracy among Related Works

Figure 51 and Figure 52 give an overview of the performance metrics from Related Works sections against the InceptionV3 model. The InceptionV3 values are indicated by dashed lines. These are not meant to compare results; they merely show how the InceptionV3 model perform relative to other forensic and gun classifier literature in terms of class-specific metrics commonly used in forensics such as accuracy, precision and recall.

The central tendency used is the median (black solid line) which is the middle value of the distribution and is not sensitive to outliers. Both Figure 51 and Figure 52 indicate that the resulting findings in this study for the InceptionV3 model (indicated by the dash lines) lie within the upper 50% of the distribution.

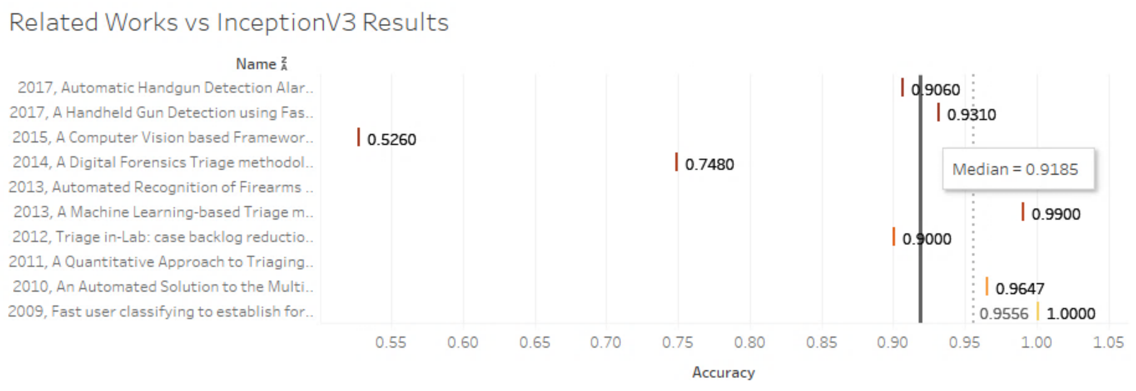


Figure 51 Related Works and InceptionV3 Accuracy Results

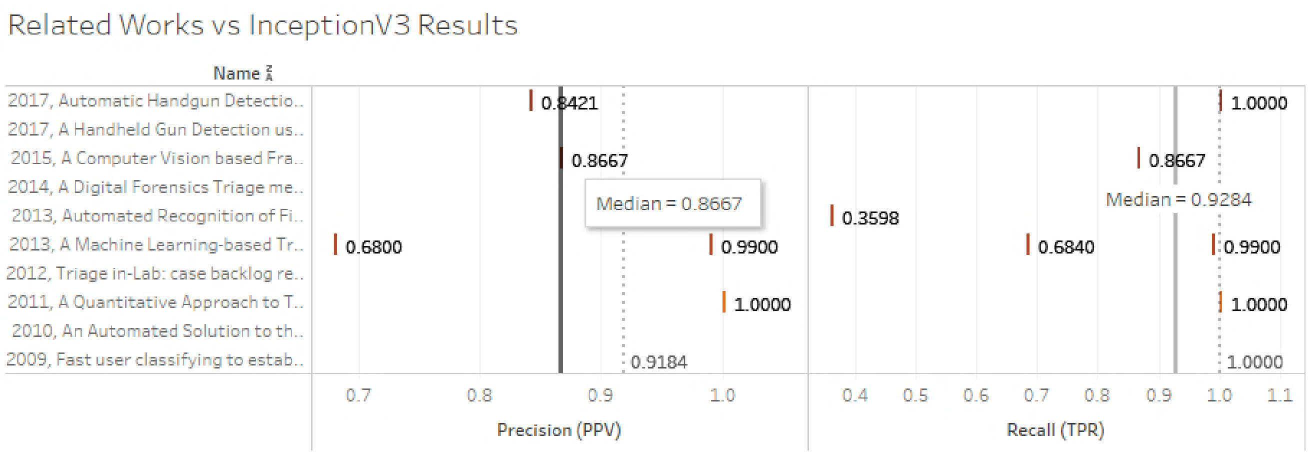


Figure 52 Related Works and InceptionV3 Precision-Recall Results

5.1.2 Model Selection

Figure 53 describes that after the best model is selected, in this case InceptionV3, it is evaluated through a series of evaluation datasets to determine its performance outside the “WEAPON-DB3” dataset.

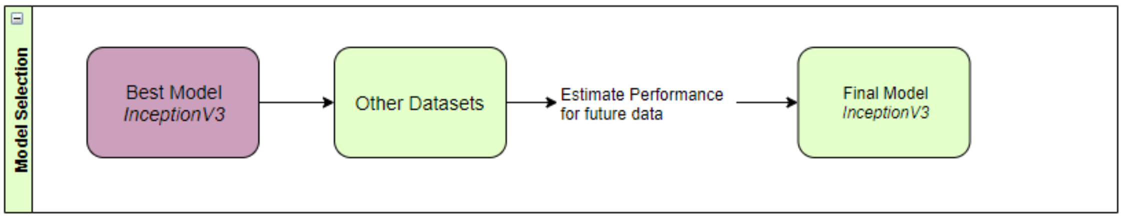


Figure 53 Model Selection: InceptionV3

5.1.2.1 Recall versus False Positive Rate with Threshold

Figure 54 shows the recall versus FPR graph including the probability threshold values. Superimposed on the graph is the suggested ROC space (recall Figure 30 in Visualising Trade-offs section). Images with prediction probabilities equal to or above the threshold are classified accordingly. The threshold values, indicated by colours in this graph, are in the range of 0.05 to 0.95. The size of the markers indicates the total number of images processed.

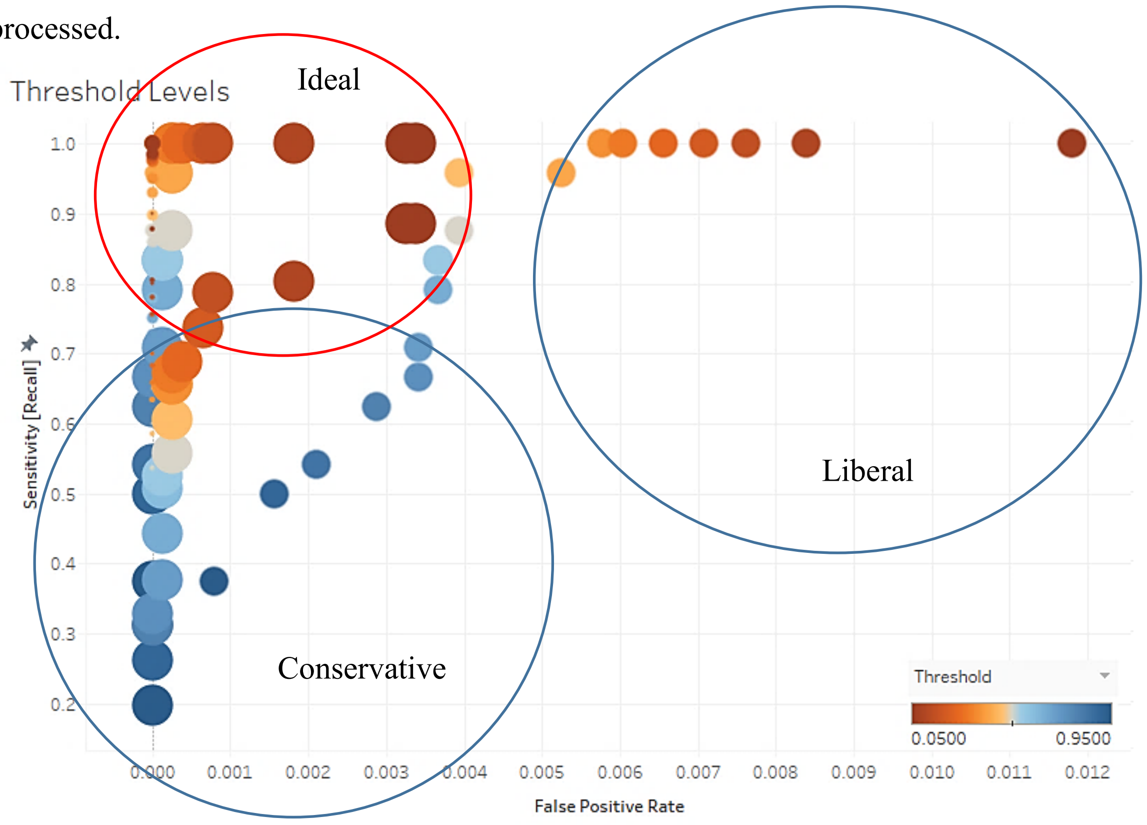
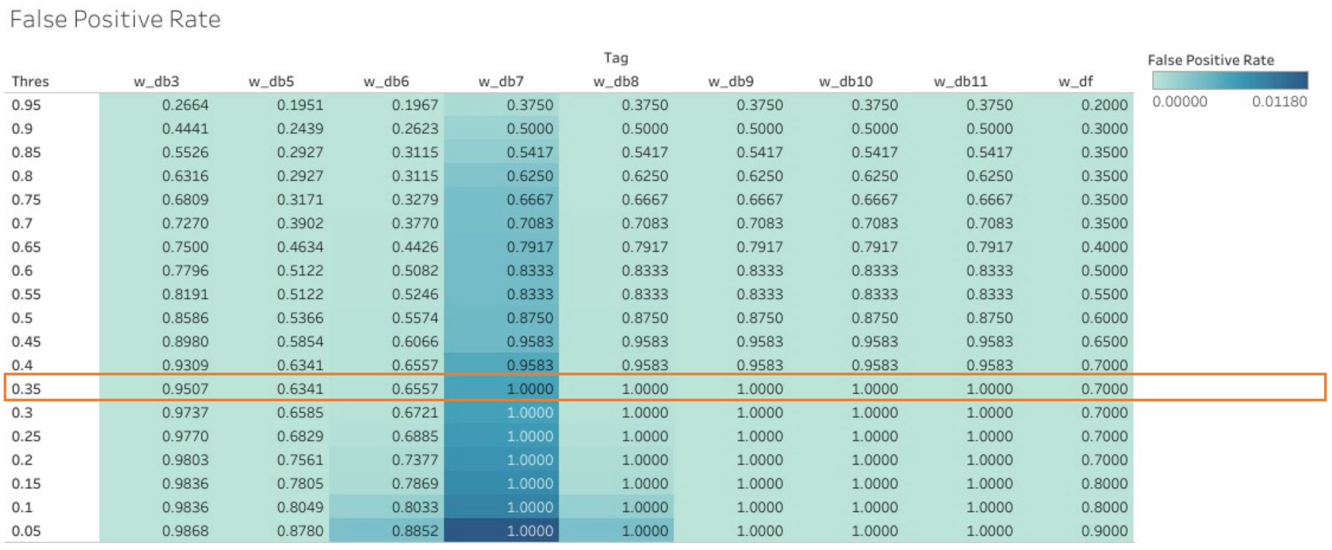


Figure 54 Recall vs False Positive Rate of InceptionV3 with Other Datasets

For the Recall-FPR graph, the ideal points should be as close as possible to the coordinates (0,1) where there are virtually no false positives and no false negatives. In a typical machine learning use-case, this ideal point remains a "gold standard" because the training dataset of models only account for a small subset of data in the infinite possibilities of images found in the real world. However, our graphs seem to indicate promise as observed from the number of points congregating around the area of high sensitivity and low false positive rates. These points have a threshold of below 0.50 as indicated by the red colours prevalent in these points. We also observe that the lower the thresholds set, the higher the sensitivity value.

Figure 55 is a matrix showing threshold (row), sensitivity (value in cell) and false positive rate (colour) mappings per dataset (column). Despite the intuition that lower thresholds result in higher false positive rates (i.e. a lower threshold tolerates more mistakes), this doesn't seem to be the case for most datasets (light-coloured cells). The intuition holds true only on w_db6, w_db7 and w_db8 datasets (indicated by darker colours as the threshold decreases). The labels on each cell are the recall rate (sensitivity) based on the number of true positives (gun pictures identified as guns) against the total actual gun images (actual + missed). As the threshold decreases, the number of missed pictures also decreases, but the sensitivity increases. This is observed in all datasets. In some datasets (in w_db7, w_db8, w_db9, w_db10), the ideal sensitivity of 1.00 is achieved with minimum 0.05 probability threshold.


False Positive Rate

	Tag								
Thres	w_db3	w_db5	w_db6	w_db7	w_db8	w_db9	w_db10	w_db11	w_df
0.95	0.2664	0.1951	0.1967	0.3750	0.3750	0.3750	0.3750	0.3750	0.2000
0.9	0.4441	0.2439	0.2623	0.5000	0.5000	0.5000	0.5000	0.5000	0.3000
0.85	0.5526	0.2927	0.3115	0.5417	0.5417	0.5417	0.5417	0.5417	0.3500
0.8	0.6316	0.2927	0.3115	0.6250	0.6250	0.6250	0.6250	0.6250	0.3500
0.75	0.6809	0.3171	0.3279	0.6667	0.6667	0.6667	0.6667	0.6667	0.3500
0.7	0.7270	0.3902	0.3770	0.7083	0.7083	0.7083	0.7083	0.7083	0.3500
0.65	0.7500	0.4634	0.4426	0.7917	0.7917	0.7917	0.7917	0.7917	0.4000
0.6	0.7796	0.5122	0.5082	0.8333	0.8333	0.8333	0.8333	0.8333	0.5000
0.55	0.8191	0.5122	0.5246	0.8333	0.8333	0.8333	0.8333	0.8333	0.5500
0.5	0.8586	0.5366	0.5574	0.8750	0.8750	0.8750	0.8750	0.8750	0.6000
0.45	0.8980	0.5854	0.6066	0.9583	0.9583	0.9583	0.9583	0.9583	0.6500
0.4	0.9309	0.6341	0.6557	0.9583	0.9583	0.9583	0.9583	0.9583	0.7000
0.35	0.9507	0.6341	0.6557	1.0000	1.0000	1.0000	1.0000	1.0000	0.7000
0.3	0.9737	0.6585	0.6721	1.0000	1.0000	1.0000	1.0000	1.0000	0.7000
0.25	0.9770	0.6829	0.6885	1.0000	1.0000	1.0000	1.0000	1.0000	0.7000
0.2	0.9803	0.7561	0.7377	1.0000	1.0000	1.0000	1.0000	1.0000	0.7000
0.15	0.9836	0.7805	0.7869	1.0000	1.0000	1.0000	1.0000	1.0000	0.8000
0.1	0.9836	0.8049	0.8033	1.0000	1.0000	1.0000	1.0000	1.0000	0.8000
0.05	0.9868	0.8780	0.8852	1.0000	1.0000	1.0000	1.0000	1.0000	0.9000

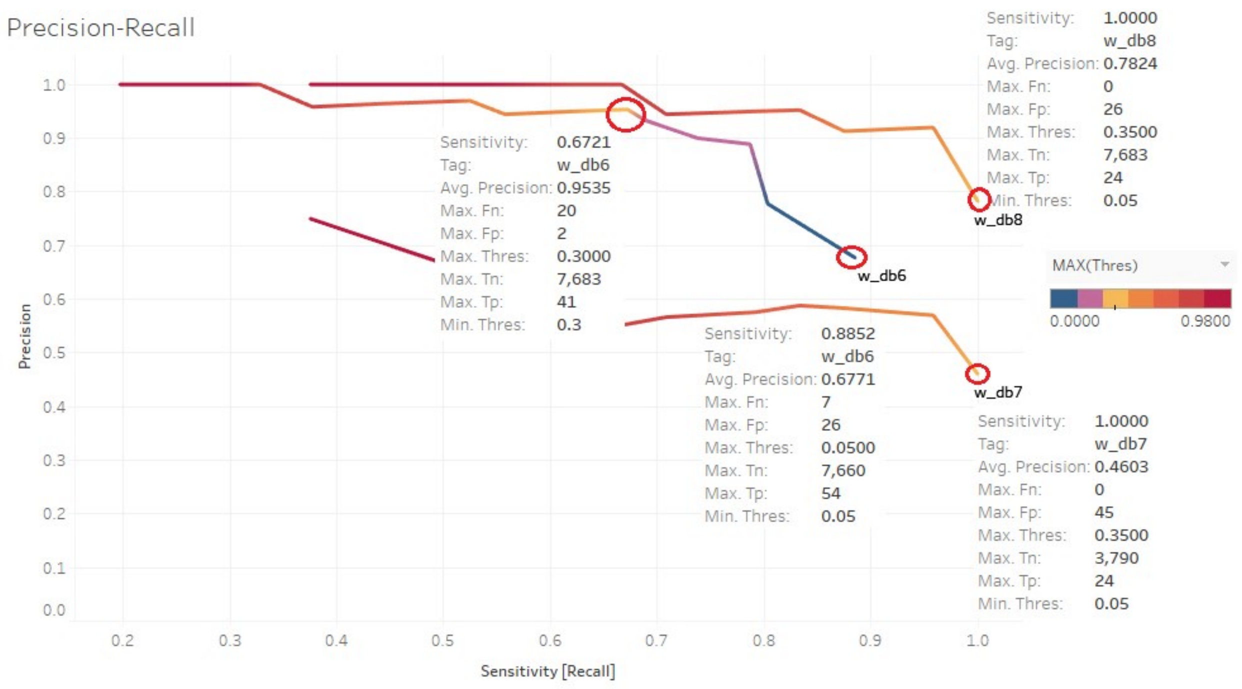
False Positive Rate
0.00000 0.01180

Figure 55 Thresholds with False Positive Rates

A good threshold value is observed to be in the 0.35 probability threshold for majority of the datasets. On this threshold, the lowest sensitivity is 63.41% in db5. However, it is reasonable to grant flexibility to the tool according to the dataset and user preference by offering adjustments to the threshold or even a choice to ignore the threshold option altogether. This observation is also supported by one of the feedbacks from the user (see Appendix 9 –Raw Feedback Responses during Evaluation).

5.1.2.2 Precision and Recall

The aim is to get a value of recall as close to 1.0 as possible since the goal is to detect as many gun images in the dataset. The trade-off between precision and recall is visualised better in a PR curve. Supporting the findings summarised in Figure 55, Figure 56 shows a PR curve for datasets db6, db7 and db8; colours indicate threshold levels (the slider is marked at 0.35). It is observed that the highest level of recall is achieved in w_db8 and w_db7 datasets with 0.35 decision threshold at the expense of precision. We can also observe that as recall increases, there is a corresponding increase in false positives (indicated by deeper colours in Figure 55) and a decrease in precision (Figure 56). Thus, depending on the context of the application, trade-offs between false positive values, precision and sensitivity should be considered. Allowing users to adjust these parameters from their end had already been suggested by one of the respondents.



Sensitivity:	1.0000
Tag:	w_db8
Avg. Precision:	0.7824
Max. Fn:	0
Max. Fp:	26
Max. Thres:	0.3500
Max. Tn:	7,683
Max. Tp:	24
Min. Thres:	0.05

Sensitivity:	0.6721
Tag:	w_db6
Avg. Precision:	0.9535
Max. Fn:	20
Max. Fp:	2
Max. Thres:	0.3000
Max. Tn:	7,683
Max. Tp:	41
Min. Thres:	0.3

Sensitivity:	0.8852
Tag:	w_db6
Avg. Precision:	0.6771
Max. Fn:	7
Max. Fp:	26
Max. Thres:	0.0500
Max. Tn:	7,660
Max. Tp:	54
Min. Thres:	0.05

Sensitivity:	1.0000
Tag:	w_db7
Avg. Precision:	0.4603
Max. Fn:	0
Max. Fp:	45
Max. Thres:	0.3500
Max. Tn:	3,790
Max. Tp:	24
Min. Thres:	0.05

Figure 56 Precision-Recall with Thresholds for db6, db7 and db8

To summarise the findings, Figure 57 shows the maximum values of sensitivity and precision based on previous figures (Figure 55 and Figure 56) regardless of threshold values using a scatter plot with label: *False Positive Value (False Negative Value)*. The results are promising, however there are 3 images detected as false negatives for dataset: WEAPON-DB5 and WEAPON-DB6.

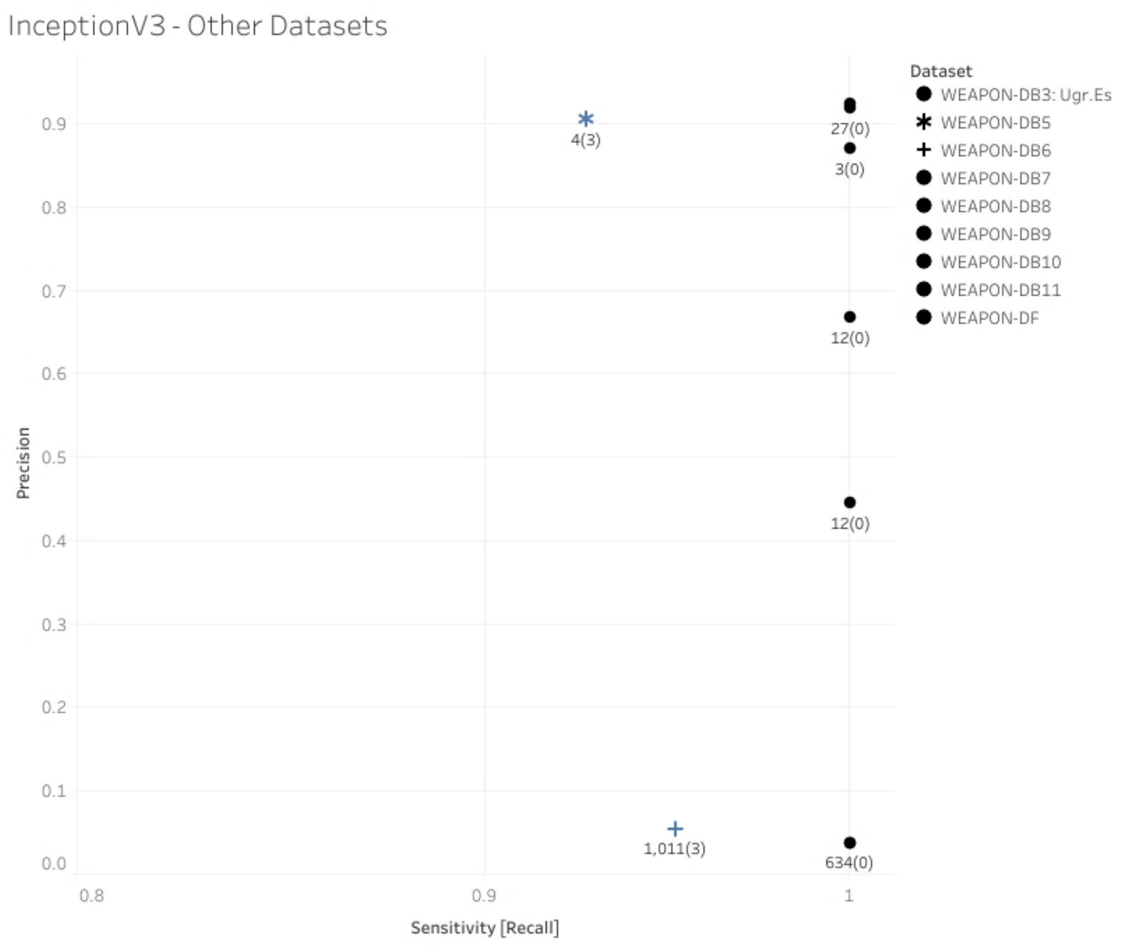


Figure 57 Precision – Recall with no thresholds

Analysing the false negatives shows that both w_db5 and w_db6 have the same images missed by the model. Figure 58 lists the false negatives.

This seems to be acceptable and therefore, InceptionV3 is the final model selected.

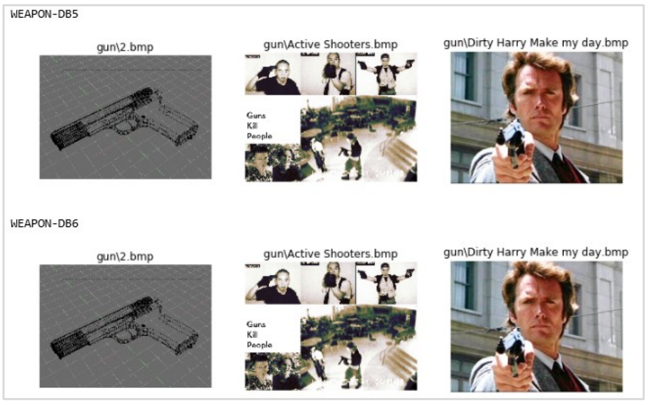


Figure 58 InceptionV3 Missed Guns

5.1.2.3 Final Model: InceptionV3 Performance

Table 13, Table 14 and Table 15 show the performance of the InceptionV3 model for the evaluation datasets (WEAPON-DB3, WEAPON-DF, WEAPON-DB5, WEAPON-DB6, WEAPON-DB7, WEAPON-DB8, WEAPON-DB9, WEAPON-DB10, WEAPON-DB11). Although the performance mean is better expressed as a confidence interval, the sample data with only nine datasets is inconclusive to draw conclusions about the population (images in the wild used during investigations). In performance estimation detailed in the next section (Section 5.2), conclusions are drawn based on the central tendencies: mean, median and mode from the sample data, and reported in Table 13, Table 14 and Table 15. Individual results from which these values are derived can be seen in Appendix 10 – Raw Data for Metrics Computation.

Table 13 InceptionV3 - Evaluation Results - 1

	Accuracy	**Sensitivity**	**Specificity**	**Precision**
Mean	0.92	0.99	0.90	0.54
Median	0.93	1.00	0.90	0.67
Mode	0.83	1.00	0.83	0.04

Table 14 InceptionV3 - Evaluation Results - 2

	False positive rate	**False discovery rate**	**False negative rate**	**MCC**[1]
Mean	0.10	0.46	0.01	0.62
Median	0.10	0.33	0.00	0.81
Mode	0.17	0.96	0.00	0.17

Table 15 InceptionV3 - Evaluation Results - 3

	Processing Time (second/picture)
Mean	2.03
Median	1.42
Mode	-

[1] Matthew's correlation coefficient

Table 16 shows the accuracy findings from the nine datasets against the model's published values (recall Table 6). The recorded evaluation accuracy is based on Top-20 output labels.

Table 16 Evaluation Accuracy vs Published Accuracy

Model	**Top-5 Accuracy**	**Top-20 Accuracy**	**Published Top-1 Accuracy**	**Published Top-5 Accuracy**
InceptionV3 Mean	0.98	0.92	0.779	0.937

One of the questions raised in the preceding section 4.2.4.2: Models Used in this Study is: *Does the published accuracy of the selected models hold when evaluated and tested against our dataset in our context (gun classification)?* From Table 16, we can observe that this paper's accuracy value for Top-5 labels is higher by 4.3 percentage points, promising good performance estimation for unseen and future data.

5.2 Final Model – Evaluation of Results

5.2.1 InceptionV3 model

Table 17 shows the results for comparative analysis of the "evaluation" set (recall Table 13), which is referred to here as the "InceptionV3 Mean Values", against the results gathered for the "unseen test set" (Android and Jane datasets).

Validating the test result findings is best expressed by reporting the overall mean within a confidence interval. A confidence interval with a significance level of 0.05 means that if this sampling (evaluation dataset) is repeated, the true mean of the population (gun images in the wild) resides within the interval 95% of the time.

Unfortunately, although sample means are best expressed as a confidence interval, the underlying assumption of normality does not hold (unless normality tests are conducted). But, using a box-plot, we can verify visually if our model's performance on unseen data are within reasonable range[1].

[1] 1.5±IQR (Inter Quartile Range)

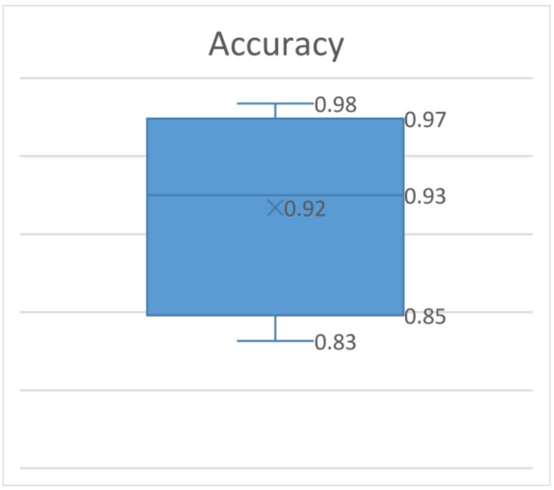
Table 17 marks the metrics and values that are visibly outliers based on the boxplots drawn.

Table 17 Comparison of Results from Evaluation Dataset

Dataset	Accuracy	Sensitivity	Precision	FPR	MCC	Time
InceptionV3 Mean Values	0.92	0.99	0.56	0.10	0.94	4.26
Android	0.83	0.78*	0.06	0.17	0.18	0.74
Jane Doe	0.84	0.95	0.08	0.16	0.22	0.62*

* outlier

For Accuracy and Precision (see Figure 59), False Positive Rate and MCC (see Figure 60), the values from the evaluation dataset are not outliers.



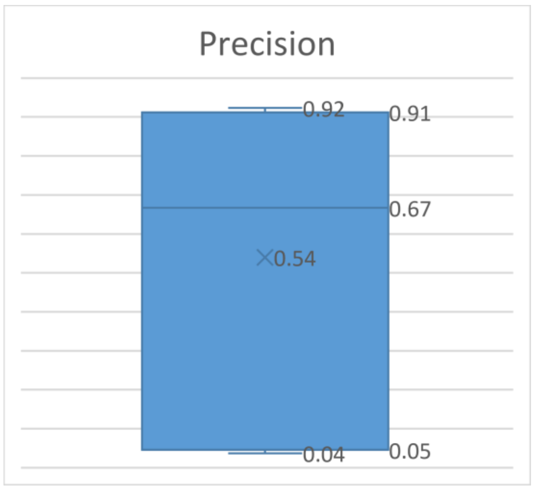
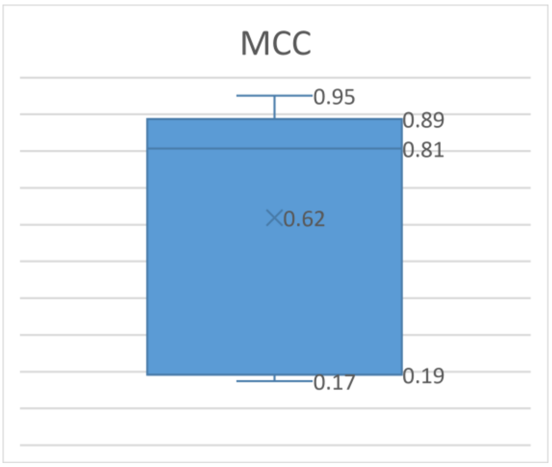


Figure 59 Box Plots - Accuracy, Precision – no outliers



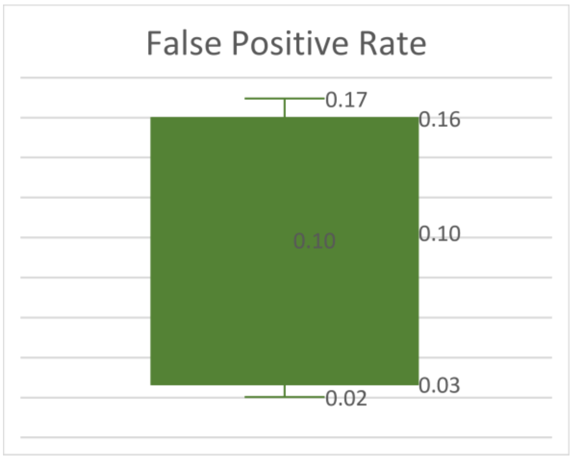


Figure 60 Box Plots – MCC, False Positive Rate – no outliers

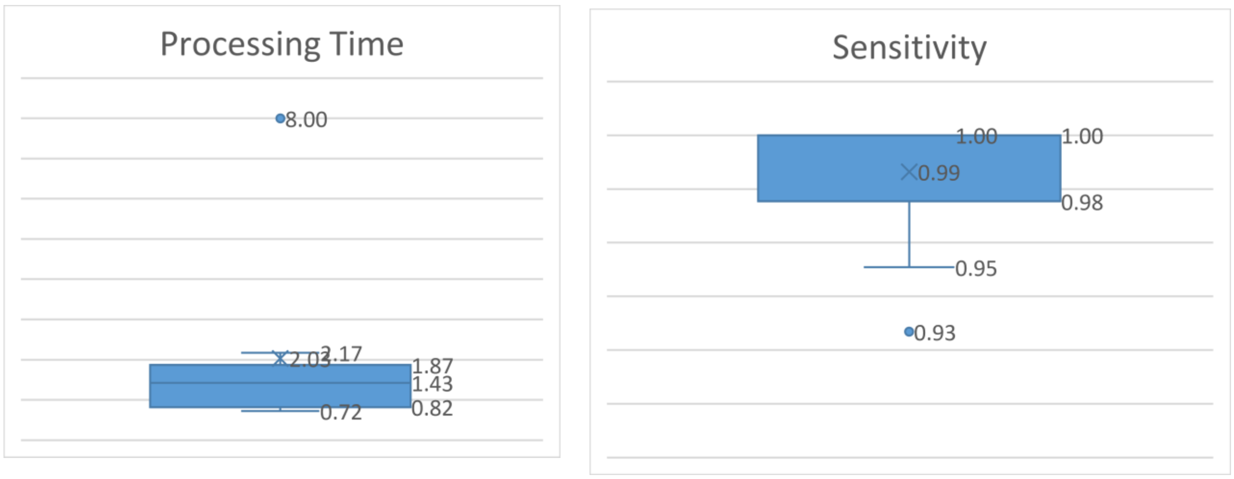
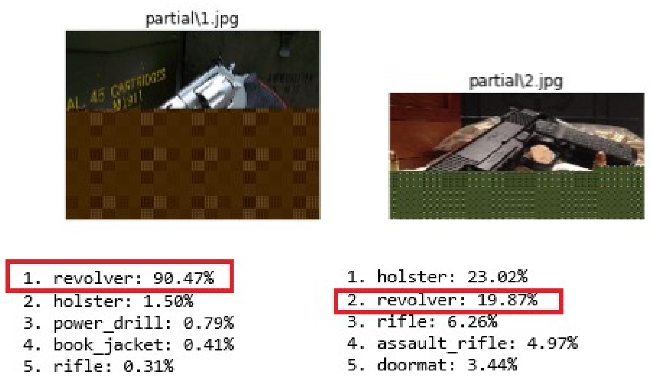


Figure 61 Box Plots – Processing Time, Sensitivity – with outliers

Figure 61 shows that sensitivity has an outlier of 0.78, and processing time has an outlier of 0.62 for the Android and Jane datasets, respectively.

We try to analyse the reasons for the outlier values. As seen from Equation 3 Recall, sensitivity is most affected by ground truth labelling mistakes. We initially suspected a level of misclassification of ground truth labels for the Android dataset; earlier, our approach to labelling is optimistic. However, if we remove four images that were occluded[1] and some partial images using the pessimistic approach, there is a positive increase in value from 0.62 to 0.89. However, this new value is still considered an outlier. Therefore, the misclassification factor could not be the main cause.

Another factor could be that the model could not properly detect occluded images. However, this can be refuted since the InceptionV3 model was able to detect and predict partial pictures successfully.



```
1. revolver: 90.47%
2. holster: 1.50%
3. power_drill: 0.79%
4. book_jacket: 0.41%
5. rifle: 0.31%
```

```
1. holster: 23.02%
2. revolver: 19.87%
3. rifle: 6.26%
4. assault_rifle: 4.97%
5. doormat: 3.44%
```

Figure 62 Partial images detected by InceptionV3

[1] When an object is hidden but itself or by some other object

The only logical conclusion we could surmise is that these models were not fully trained on filtered occluded and partial images. However, additional empirical tests should be conducted to support this observation. Regardless, filtered images should be taken into consideration when planning to re-train models to improve accuracy.

Although part of the criteria in model selection, processing time outlier could be safely ignored as speed in processing images is not a class-specific metric and therefore not the focus of this study. Speed is mentioned merely to illustrate the average speed of models when processing images and offer comparisons.

5.3 Prototype - Evaluation of Results

The test and measure approach on usability is completed with five respondents. The respondents are asked to test out the prototype by answering tasks in both manual method and in using the prototype. Then the usability is measured using SUS.

5.3.1 Distribution of Respondents

Sixty percent of the users who evaluated the prototype are forensic professionals each from Estonia, Germany and Croatia, two have extensive forensic experience. Two MSc Cyber Security second-year students who took the Systems Forensics course in Spring 2018 from Tallinn University of Technology made up the remaining 40% of the distribution.

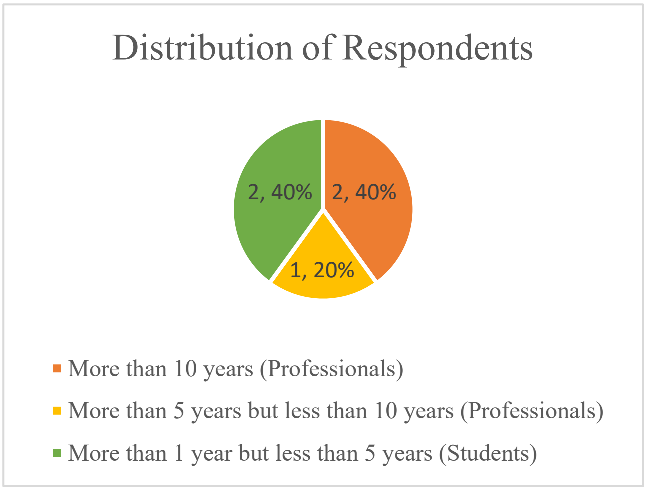


Figure 63 Distribution of Respondents

5.3.2 Efficiency - Speed of Performance

One of the research questions of this study is: *Once the users learned the design, can they perform the task efficiently?* This is answered by determining if the analysis time (duration) is reduced when using the prototype (tool). The duration in completing tasks by manually searching for content versus using the tool is analysed for statistical difference. Using hypothesis testing, we reformulate the question as: "Is there a *significant difference* in speed using the tool compared to the existing manual methods of searching for image content?".

In hypothesis testing, a parametric approach is taken when the distribution is known, for example, *t-test* if the distribution is normal or approximately normal. When normality does not hold, a non-parametric approach is employed. For both approaches, the sample size plays a central role in the interpretation of results in relation to statistical power and significance level.

Assuming Non-Normality: Wilcoxon Rank Signed Test

Testing for normality is usually the first step in statistical analysis as most statistical procedures have an underlying assumption of normality. There are four commonly used formal normality tests that can be conducted: Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. These "normality tests" test the following hypothesis:

$$H_0: Data\ follows\ a\ normal\ distribution$$
$$H_1: Data\ does\ not\ follow\ a\ normal\ distribution$$

Equation 12 Normality Hypothesis testing

However, in this paper, the sample size is low (n = 5). In [74], experiments show that all four tests are not reliable when it comes to small sample sizes (n < 10) due to low statistical power. The statistical power is the ability to correctly identify a significant difference if there is one, that is, reject the null hypothesis if the alternative hypothesis is true. This means that the likelihood of detecting the distribution to be "not normal" when it is true is low. For this reason, the test for normality is not conducted and we assume that the data distribution violates the assumption of normality.

Using the Wilcoxon signed rank test is the equivalent of a paired t-test, and this is the non-parametric test used in this paper.

There are key assumptions in using this test: the differences of the set of pair-wise values are continuous and symmetric around the mean [75]. Our data comes from matched pairs as respondents performed both tasks, the duration is a continuous variable and lastly, the data shows relative symmetry as illustrated by the box plot in Figure 64.

Table 18 Task Duration with Differences for each Participant

P	M	T	M - T
1	21	57	-36
2	8	9	-1
3	29	16	13
4	39	10	29
5	20	35	-15

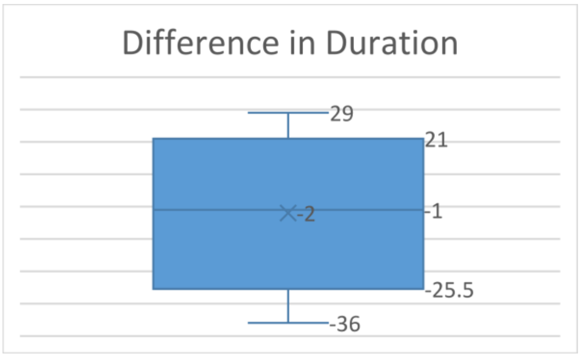


Figure 64 Symmetry Check using a Box Plot

Table 18 shows the time of completion of tasks for individual participants in using manual methods (M) and in using the tool (T). Figure 64 shows the corresponding box-plot taken from the values of the differences (M – T). Data distribution is seen to be symmetrical along the mean (identified by the "x" mark in the boxplot). Therefore, key assumptions specified in [75] are not violated, and the Wilcoxon signed rank test can be conducted.

We assume a null hypothesis (H_0) that both samples have no statistical difference (mean between the two samples is zero: $\mu_{\text{manual}} - \mu_{\text{tool}} = \mu_d = 0$. We want to prove that the tool allows the users to perform the tasks faster, that is, our alternate hypothesis (H_1) is μ_d is greater than zero.

$$H_0: M_d = 0$$

$$H_1: M_d > 0$$

Equation 13 Wilcoxon Signed Rank Test Hypothesis Testing

The calculations (outlined in Appendix 11 – Speed: Hypothesis Testing) show that the critical value for the significance level α=0.10 is w = 0 and the null hypothesis is rejected at $W \leq 0$. Since in this case, W is 6 and ($W \geq w \mid H_0$), the null hypothesis is not rejected. *There is no statistical difference in using the manual method against using the prototype in completing the tasks.*

Assuming Normality: T-test

According to Sauro [76], the average task time should be reported using the geometric mean and expressed as a confidence interval. Since task time is positively skewed, his paper suggests running statistical analysis on the task time by log-transforming the data first to fit it into a normal distribution. Using his research as the basis for the assumption of normality for the distribution of task times, we conducted a one-tailed paired t-test. *One-tailed* because we want to determine a positive difference between the two means of the distribution, *paired* because we are measuring one group with paired observations (before using the prototype and after using the prototype) and *t-test* because of the small number of samples (n = 5) and the unknown variances of our distribution.

The hypothesis is tested similarly to a Wilcoxon signed rank test earlier. We assume a null hypothesis (H_0) that both samples have no statistical difference. We want to prove that the tool allows the users to perform the tasks faster; our alternate hypothesis (H_1) is an upper-tailed test where μ_d is greater than zero. We used the level of significance, $\alpha = 0.05$.

$$H_0: \mu_{\text{manual}} = \mu_{\text{tool}}$$
$$H_1: \mu_{\text{manual}} > \mu_{\text{tool}}$$

Equation 14 t-test Hypothesis Testing

Table 19 Task Duration: t-test

Log Transformed Data	Manual (log-trans)	Tool (log-trans)
Mean	20.69952485 (minutes)	19.57316596 (minutes)
t Stat	0.133474971	
P(T<=t) one-tail	0.450131797	
t Critical one-tail	2.131846786	

Table 19 shows the results of the t-test using log-transformed data for each participant's task duration. From the results, the p-value calculated is $P(T \geq t \mid H_0)$ with $\alpha=0.05$ which means that our null hypothesis is not rejected. *There is no statistical difference in using the manual method against using the prototype in completing the tasks.*

Both parametric and non-parametric tests support the same conclusion.

Expressing the duration as a confidence interval based on the geometric mean as suggested in [76], we can express the time to complete the tasks to be within (18.6, 22.79) and (16.86, 22.28) minutes with 95% confidence for manual method and using the tool, respectively. This is tabulated in Table 20.

Table 20 Task Duration - Confidence Intervals

	Manual (minutes)	**Tool (minutes)**
Geometric Mean	20.69952485	19.57316596
Confidence Interval at 95%	20.70 ± 2.096	19.57 ± 2.71

Inspection of the data might be misleading as it shows that the tool seems to perform better: the mean is lower for the tool, having a margin of error larger by only 0.6 seconds. However, the statistical power of this test is low because the number of respondents is low ($n = 5$), therefore a larger difference is expected to be able to conclude a significant difference between the two methods.

The complete solutions are found in Appendix 11 – Speed: Hypothesis Testing

Real-Case Estimations

Perhaps a contributing factor to the conclusion of finding no difference between using the tool against existing manual methods might be the number of test data examined. Based on an estimated real case load, we estimated that a case would have a minimum of one personal computer (PC) and one mobile device. For an organised crime case, minimum would be around five of each pair (PC and mobile). A pair roughly has 3,000 images (irrelevant images from the system are filtered out) while the organised crime case would have a minimum of 30,000 images to go through. However, requesting users to go through 3,000 to 30,000 images requires significant amount of time and dedication. As a compromise, the users are asked to go through a total of 7618 images with two-thirds (4 out of 6) of the tasks dedicated to searching for image content from only 281 images. In comparison with the minimum real-case estimates and the tasks involved, the proportion of test images for one pair is 47% and 9.4% for organised crime. The test cases barely cover half of the minimum estimated case load and only a tenth for organised crime.

Perhaps if we had used large-scale image content search in our usability tests as is estimated, the results would have been more reflective of an actual case load.

Correct Answers vs Duration

In this study, the efficiency of the tool is measured by the difference in the speed of performance between manual methods and the tool. While the preceding section measures the task time, this section analyses the time duration of completing the task and the number of correct answers as shown in Figure 65.

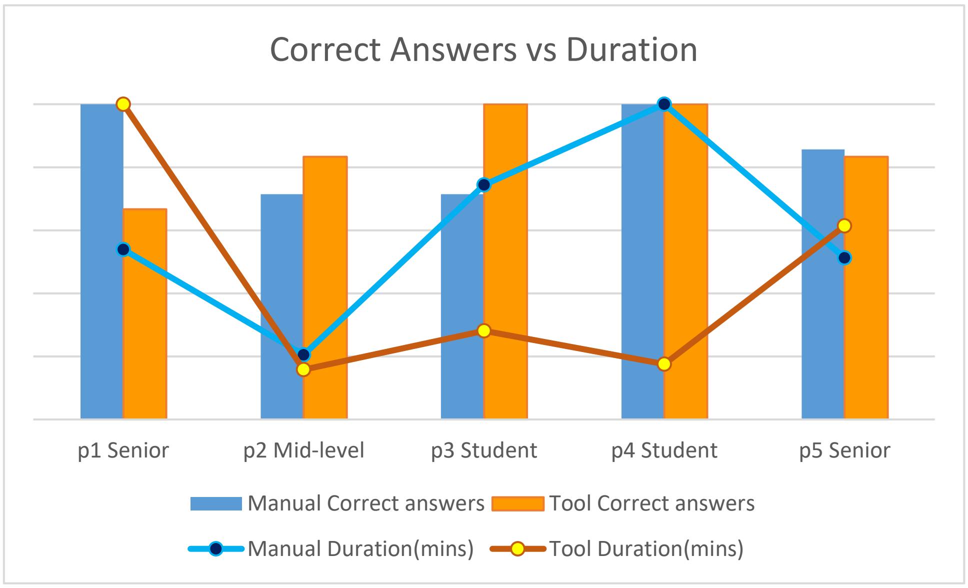


Figure 65 Summary of Correct Answers and Duration

All the students, who got all the answers right, were able to solve the tasks faster using the tool than manually doing it, while the experienced (senior) forensic respondents had more correct answers using the manual method. Perhaps this could be attributed to the examiner's adept usage of manual methods and the student's unfamiliarity of the built-in image viewers as a tool in forensic investigations. No one used specialised software for classification of images; the respondents used default built-in tools such as Windows Explorer, and image viewers in iOS and Ubuntu.

A demo of the prototype was conducted for participant 1, 3 and 4 while others evaluated the tool using the online installation instructions. Two out of five respondents (40%) got more correct answers after using the tool while one student got all the answers correct for both methods.

Correlation of Variables

Analysing the relationship between two variables such as correct answers and duration, is best done by computing for the Pearson Correlation Coefficient (*r*) to indicate the linear relationship between variables. The interpretation of the scores is taken from [77] (see Appendix 5 – Rule of Thumb for Analysing Size of a Correlation Coefficient). Correlation does not imply causation (*i.e.* one variable *causes* an effect - increase or decrease, on another variable). This coefficient implies an association between variables, for a Pearson coefficient, it checks for a *linear* relationship between the variables.

Correct Answers vs Duration

For the manual method, duration and number of correct answers have a moderate linear relationship ($r = 0.5$, p-value=0.39), that is, an increase in the correct answers correspond to an increase in duration. This seems to be logical, as the number of tasks is increased with a constant processing time, the total task time also increases. For the prototype, there is a strong indication of negative correlation for correct answers and speed – increase in correct answers correspond to a decrease in duration ($r = -0.82$, p-value=0.08, $\alpha = 0.05$). 67% of the variation in duration, denoted by the coefficient of determination[1] (r^2), is explained by the correct answers. This seems contrary to intuition – users with a high number of correct answers spend less time on the task. One plausible explanation could be that the processing time per image decreases because of the growing experience in using the tool.

However, the p-value tells us that the associations of the variables (manual and tool) would not be considered statistically significant at 95% confidence. Although the correlation seems to be strongly indicative of a negative relationship, the results are inconclusive. We need to establish an improved testing process or increase the number of respondents to increase the statistical power of our test.

The correlation table and p-values are in Appendix 6 – Correlation Solution.

5.3.3 Usability Score

Sauro [78] has consolidated the different methods from different papers [65] [66] in an attempt to interpreting the raw scores. His findings are tabulated in Table 21.

[1] explains the percentage of variation in Y that can be predicted from X.

The mean SUS score gathered for the five respondents is 69. This raw score falls under C and within the 50% percentile, getting an 'OK' rating (highlighted row). Based on the SUS results, it can be concluded that the prototype is acceptable. However, the SUS itself is non-diagnostic. Problems with the user interface could only be discovered during the interview and/or feedback.

Table 21 SUS Score Interpretation

Grade	SUS	Percentile range	Adjective	Acceptable
A+	84.1-100	96-100	Best Imaginable	Acceptable
A	80.8-84.0	90-95	Excellent	Acceptable
A-	78.9-80.9	85-89	Good	Acceptable
B+	77.2-78.8	80-84	Good	Acceptable
B	74.1 – 77.1	70 – 79	Good	Acceptable
B-	72.6 – 74.0	65 – 69	Good	Acceptable
C+	71.1 – 72.5	60 – 64	Good	Acceptable
C	65.0 – 71.0	41 – 59	OK	Acceptable
C-	62.7 – 64.9	35 – 40	OK	Marginal
D	51.7 – 62.6	15 – 34	OK	Marginal
F	25.1 – 51.6	2– 14	Poor	Unacceptable
F	0-25	0-1.9	Worst Imaginable	Unacceptable

Figure 66 shows the SUS scores per participant, the overall mean score (average) is indicated by an orange dashed line and the baseline of 68 is a black solid line.

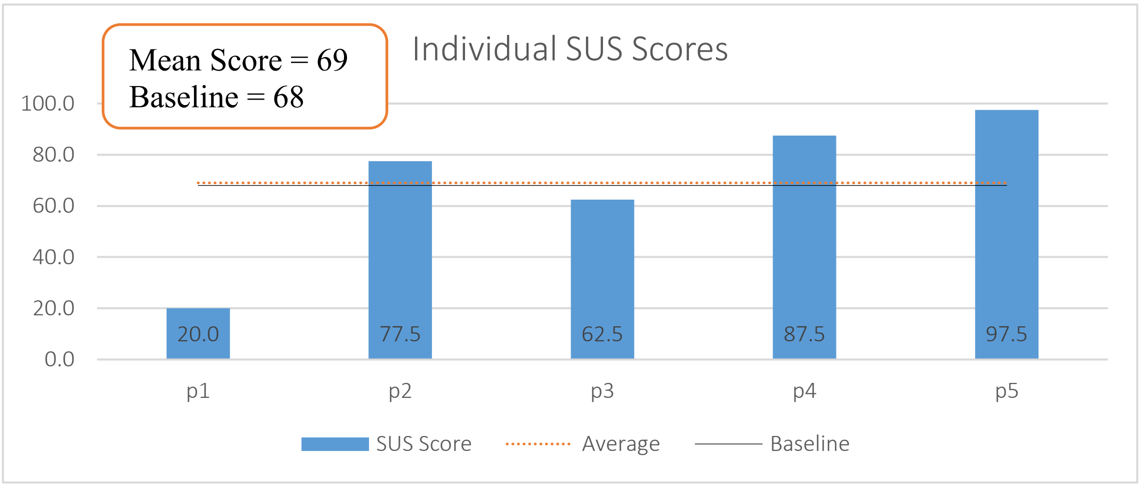


Figure 66 Individual SUS Scores

5.3.4 Learnability

For measuring learnability, we employ the method used in [66] in benchmarking results for item-level SUS items. Table 22 is taken from [66] and tabulates the usability attribute of each question and the corresponding item score (as a range) for the associated target SUS level (average and good).

Table 22 Item-level benchmarks

SUS Item Question	Usability Attribute	Target for Average Score (SUS = 68)	Target for Good Score[1] (SUS = 80)
1		≥ 3.39	≥ 3.80
2	Complexity	≤ 2.44	≤ 1.85
3	Ease of Use	≥ 3.67	≥ 4.24
4	Learnability	≤ 1.85	≤ 1.51
5		≥ 3.55	≥ 3.96
6	Consistency	≤ 2.20	≤ 1.77
7	Learnability	≥ 3.71	≥ 4.19
8		≤ 2.25	≤ 1.66
9	Confidence	≥ 3.72	≥ 4.25
10	Learnability	≤ 2.09	≤ 1.64

The score for each item is derived from the regression equation given in [66]. The results can either be Good, Average or Poor. *Good* means that a SUS of 80 is achieved, *Average* is at least 68 and *Poor* means neither of the targets were achieved.

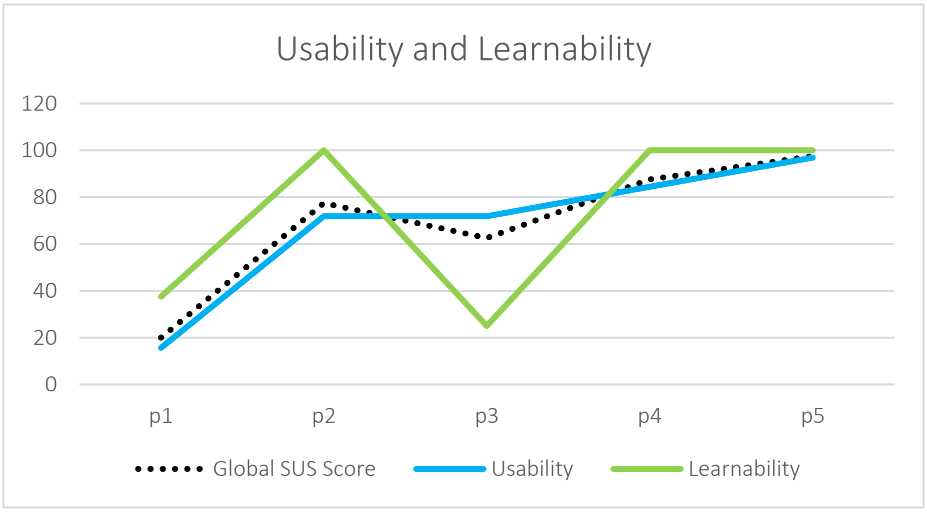


Figure 67 Usability and Learnability Graph

The sub-scale computations for usability and learnability against the overall SUS score are taken from Sauro [66], [72]. The results are presented in Figure 67. Although the results from participant 3 (p3) shows a different trend, in general, the learnability aspect

[1] Standard industrial goal for a good user experience [66]

of the tool outperforms both global and usability scores. This can be an indication that the tool can be used with ease when deployed in practice.

The numerical basis (from which Figure 67 is taken) and its corresponding rating is tabulated in Table 23. Here, we can clearly see that the SUS findings for learnability show an *Average* and *Acceptable* rating.

Table 23 SUS Item-level Results

SUS Item	Score	Usability Attribute	Results
1. I think that I would like to use this system frequently if I have to deal with categorizing images and getting EXIF information.	3.421583	*Usability*	Average - C
2. I found the system unnecessarily complex.	2.39837835	Complexity	Average - C
3. I thought the system was easy to use.	3.72199864	Ease of Use	Average - C
4. I think that I would need *assistance (technical or non-technical)* to be able to use this system.	1.82251156	Learnability	Average - C
5. I found the various functions (console prediction, search, display EXIF information, report screen) in this system were well integrated.	3.58101901	*Usability*	Average - C
6. I thought there was too much inconsistency in this system. It would be difficult to deploy the tool in the real world.	2.16144182	Consistency	Average - C
7. I *think* that most *forensic examiners* would learn to use this system very quickly.	3.74977867	*Usability*	Average - C
8. I found the *tool inconvenient* to use.	2.19662174	*Usability*	Average - C
9. I felt very confident using the *tool*.	3.75991896	Confidence	Average - C
10. The learning curve to use this tool is steep. I needed to learn a lot of things before I could use the tool.	2.05625717	Learnability	Average - C

5.3.5 Acceptability of the Prototype as a Forensic Tool

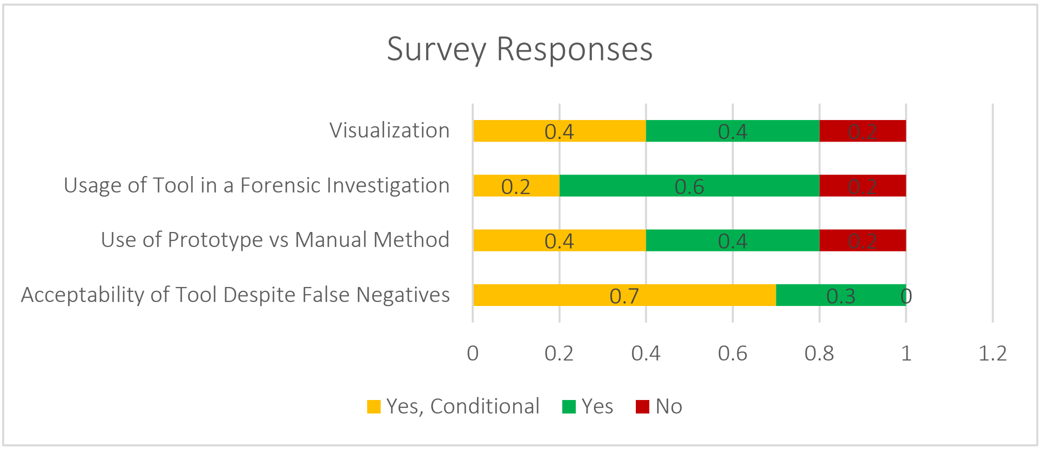


Figure 68 Survey Responses

Figure 68 presents the survey responses (in a stacked bar graph in percentages) categorised by important areas: visualisation, usage of tool, prototype vs manual method and acceptability of tool. When asked about the acceptability of the false negative rate of the model, *"Acceptability of Tool Despite False Negatives",* all respondents answer positively (Yes and Yes with Potential for Improvement).

When asked for the possibility of using the tool in an investigation, "*Usage of Tool in a Forensic Investigation*", one respondent answers "No". However, his feedback clarifies that his forensic work does not involve object classification or content search.

On performance speed, statistical analysis shows that there is no significant difference in using manual methods and the tool. However, when asked about preference of using the tool over the manual methods, "*Use of Prototype vs Manual Method*", and "*Visualisation*", only one professional preferred the manual method over the tool.

5.3.6 Summary of Evaluation

The summary of evaluation findings in the context of the requirements (recall Problem Identification and Requirements section) is given in Table 24.

Table 24 Requirements and User Feedback

Requirements	Status	User Feedback
Open-source tool that can be customized, modified and improved	Implemented	-
Automated categorization of images	Accepted	Majority (70%) - Yes, with further improvements

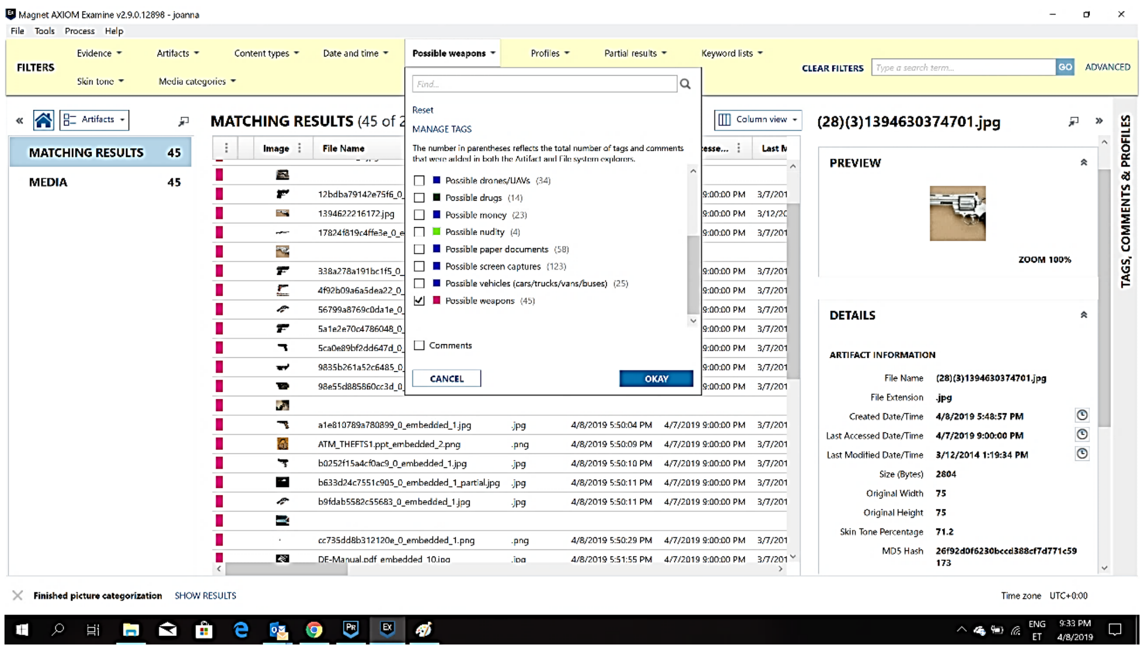
Metadata extraction and maps	Implemented	-
File Export	Implemented	-
Visualization	Accepted	80% - Yes, visualization is useful and can be improved (e.g. bar graph or frequency chart)
Weapon categorization	Accepted	Despite the false negatives, all users said the prototype can be used and improved.

The SUS test is not a diagnostic tool, it does not give insights on the functionality that should be improved or the areas in the tool that need to be addressed. The SUS is supplemented by a feedback either in email or face to face discussions. The responses are recorded in Appendix 9 –Raw Feedback Responses during Evaluation.

5.4 Alternatives to InceptionV3

5.4.1 Forensic Software

The purpose of this section is to evaluate the neural network-based categorisation feature of existing forensic software against the pre-trained models used by the prototype. *Magnet Axiom* is used for comparison. The screenshot of the weapons categorization function in Magnet Axiom is displayed in Figure 69.



The number in parentheses reflects the total number of tags and comments that were added in both the Artifact and File system explorers.

- [ ] Possible drones/UAVs (34)
- [ ] Possible drugs (14)
- [ ] Possible money (23)
- [ ] Possible nudity (4)
- [ ] Possible paper documents (58)
- [ ] Possible screen captures (123)
- [ ] Possible vehicles (cars/trucks/vans/buses) (25)
- [✓] Possible weapons (45)
- [ ] Comments

```
12bdba79142e75f6_0_                                              9:00:00 PM   3/7/201
1394622216172.jpg                                                9:00:00 PM   3/12/20
17824f819c4ffe3e_0_e                                             9:00:00 PM   3/7/201

338a278a191bc1f5_0_                                              9:00:00 PM   3/7/201
4f92b09a6a5dea22_0_                                              9:00:00 PM   3/7/201
56799a8769c0da1e_0_                                              9:00:00 PM   3/7/201
5a1e2e70c4786048_0_                                              9:00:00 PM   3/7/201
5ca0e89bf2dd647d_0_                                              9:00:00 PM   3/7/201
9835b261a52c6485_0_                                              9:00:00 PM   3/7/201
98e55d885860cc3d_0_                                              9:00:00 PM   3/7/201

a1e810789a780899_0_embedded_1.jpg          .jpg   4/8/2019 5:50:04 PM   4/7/2019 9:00:00 PM   3/7/201
ATM_THEFTS1.ppt_embedded_2.png             .png   4/8/2019 5:50:09 PM   4/7/2019 9:00:00 PM   3/7/201
b0252f15a4cf0ac9_0_embedded_1.jpg          .jpg   4/8/2019 5:50:10 PM   4/7/2019 9:00:00 PM   3/7/201
b633d24c7551c905_0_embedded_1_partial.jpg  .jpg   4/8/2019 5:50:11 PM   4/7/2019 9:00:00 PM   3/7/201
b9fdab5582c55683_0_embedded_1.jpg          .jpg   4/8/2019 5:50:11 PM   4/7/2019 9:00:00 PM   3/7/201

cc735dd8b312120e_0_embedded_1.png          .png   4/8/2019 5:50:29 PM   4/7/2019 9:00:00 PM   3/7/201
DE-Manual.pdf_embedded_10.jpg              .jpg   4/8/2019 5:51:55 PM   4/7/2019 9:00:00 PM   3/7/201
```

(28)(3)1394630374701.jpg

ARTIFACT INFORMATION

File Name	(28)(3)1394630374701.jpg
File Extension	.jpg
Created Date/Time	4/8/2019 5:48:57 PM
Last Accessed Date/Time	4/7/2019 9:00:00 PM
Last Modified Date/Time	3/12/2014 1:19:34 PM
Size (Bytes)	2804
Original Width	75
Original Height	75
Skin Tone Percentage	71.2
MD5 Hash	26f92d0f6230bccd388cf7d771c59173

Time zone UTC+0:00

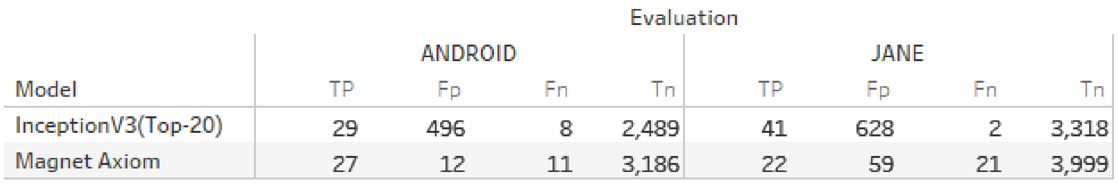
ENG ET 9:33 PM 4/8/2019

Figure 69 Magnet Axiom - Weapons Categorization

Magnet Axiom[1] is a forensic software capable of acquiring and recovering digital evidence from multiple sources (mobile, laptop, cloud) including categorization of multi-media. A trial version of the software was downloaded to categorise pictures with gun. Unlike the prototype, the categories in Magnet Axiom are fixed (e.g. possible weapons, card/ID/paper documents, vehicles(cars/trucks/vans/buses), etc). The equivalent search for guns in Magnet Axiom is "Possible Weapons" categorization.

5.4.1.1 Comparison of InceptionV3 and Magnet

The class-based performance metrics of both InceptionV3 and Magnet Axiom are recorded in Figure 70.



Evaluation

	ANDROID				JANE			
Model	TP	Fp	Fn	Tn	TP	Fp	Fn	Tn
InceptionV3(Top-20)	29	496	8	2,489	41	628	2	3,318
Magnet Axiom	27	12	11	3,186	22	59	21	3,999

Figure 70 Confusion Matrix between InceptionV3 and Magnet

From the confusion matrix in Figure 70, Magnet Axiom produces higher accuracy because of the fewer false positives. However, the number of missed guns in the test set is higher in Magnet Axiom, thrice that of InceptionV3 results.

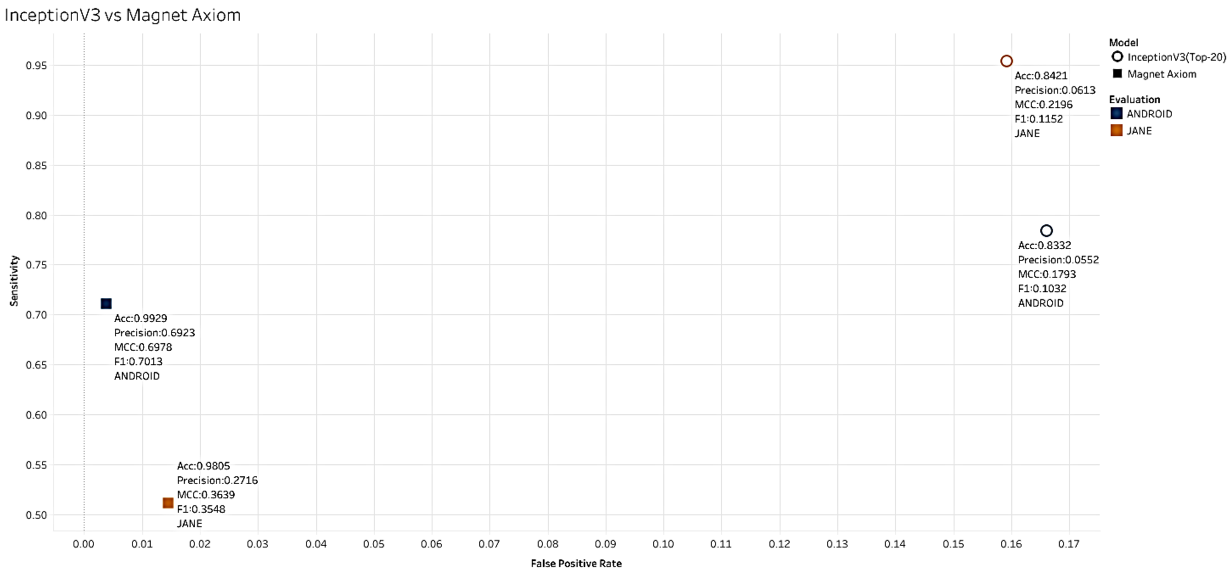


Figure 71 Inception vs Magnet

[1] https://www.magnetforensics.com/products/magnet-axiom/

In Figure 71, we can observe that Magnet Axiom has a higher accuracy and precision due to lower false positives at the expense of sensitivity. The performance for sensitivity in the Jane dataset is lowest at around 51%. This value is 12 percentage points lower than the sensitivity observed for the observed optimal threshold of 35% for the sample datasets (see Recall versus False Positive Rate with Threshold section) using the InceptionV3 model.

5.4.2 Other Pre-trained Networks

There are other pre-trained models available in the Keras framework. Table 25 shows the models supported in Keras along with the published accuracy ratings at the time of this writing. Parameters are the weights of the neural connections and depth is the number of topological layers such as activation and normalisation layers. The highlighted rows indicate the models used during the selection process (recall Models Used in this Study).

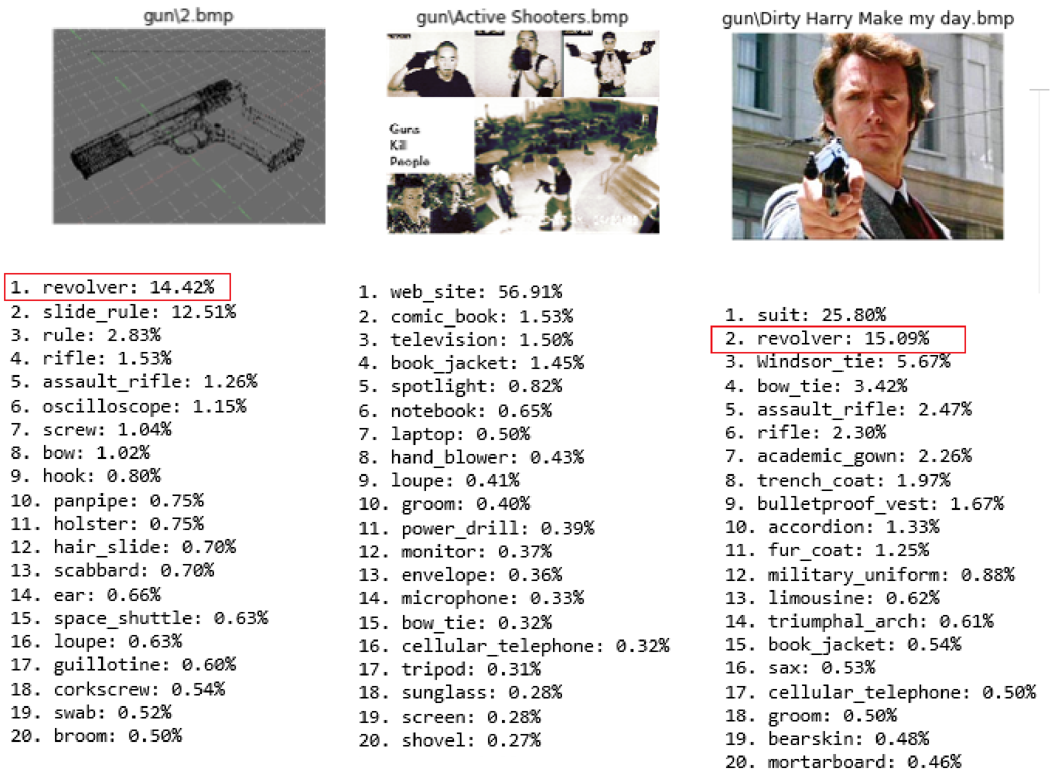
Table 25 Pre-trained Models available in Keras[1].

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.79	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.9	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.76	0.93	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.78	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.75	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.96	88,949,818	-

[1] https://keras.io/applications

During Model Evaluation, the InceptionV3 and Xception models had had the highest recall rating (100%), but InceptionV3 offered better accuracy, precision, false positive rate and MCC on the evaluated “WEAPON-DB3” dataset. However, the model selection only evaluated the models on one dataset – the “WEAPON-DB3” from Olmos’ paper [15]. Although the published accuracy ratings do not guarantee the same accuracy against any non-ImageNet dataset, it serves as a benchmark and sanity check.

Since the Xception model results were promising during the model selection phase, we analyse the performance of the Xception model against the rest of the evaluation dataset.



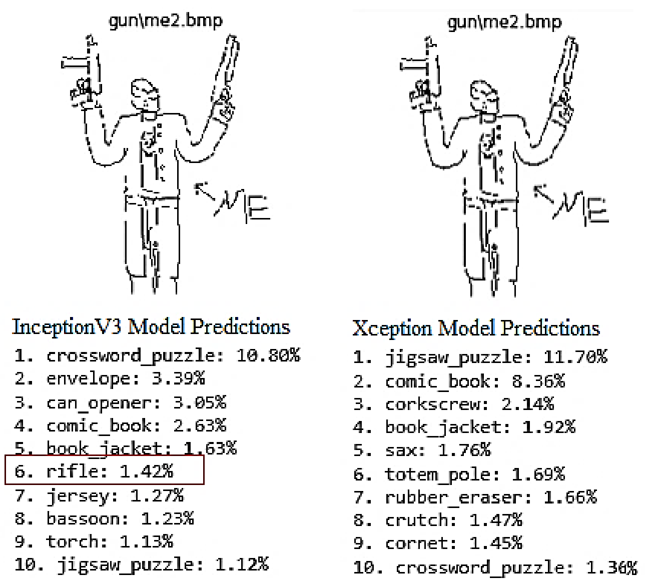
```
1. revolver: 14.42%
2. slide_rule: 12.51%
3. rule: 2.83%
4. rifle: 1.53%
5. assault_rifle: 1.26%
6. oscilloscope: 1.15%
7. screw: 1.04%
8. bow: 1.02%
9. hook: 0.80%
10. panpipe: 0.75%
11. holster: 0.75%
12. hair_slide: 0.70%
13. scabbard: 0.70%
14. ear: 0.66%
15. space_shuttle: 0.63%
16. loupe: 0.63%
17. guillotine: 0.60%
18. corkscrew: 0.54%
19. swab: 0.52%
20. broom: 0.50%
```

```
1. web_site: 56.91%
2. comic_book: 1.53%
3. television: 1.50%
4. book_jacket: 1.45%
5. spotlight: 0.82%
6. notebook: 0.65%
7. laptop: 0.50%
8. hand_blower: 0.43%
9. loupe: 0.41%
10. groom: 0.40%
11. power_drill: 0.39%
12. monitor: 0.37%
13. envelope: 0.36%
14. microphone: 0.33%
15. bow_tie: 0.32%
16. cellular_telephone: 0.32%
17. tripod: 0.31%
18. sunglass: 0.28%
19. screen: 0.28%
20. shovel: 0.27%
```

```
1. suit: 25.80%
2. revolver: 15.09%
3. Windsor_tie: 5.67%
4. bow_tie: 3.42%
5. assault_rifle: 2.47%
6. rifle: 2.30%
7. academic_gown: 2.26%
8. trench_coat: 1.97%
9. bulletproof_vest: 1.67%
10. accordion: 1.33%
11. fur_coat: 1.25%
12. military_uniform: 0.88%
13. limousine: 0.62%
14. triumphal_arch: 0.61%
15. book_jacket: 0.54%
16. sax: 0.53%
17. cellular_telephone: 0.50%
18. groom: 0.50%
19. bearskin: 0.48%
20. mortarboard: 0.46%
```

Figure 72 Xception Predictions for InceptionV3’s false negatives (missed guns)

In the InceptionV3 model, there are three images missed out in the DB5 and DB6 datasets. When predicted using the Xception model, two out of three images are detected; see Figure 72 for the images and predictions. However, Figure 73 shows there is one image, “me2.bmp”, that the Xception missed but InceptionV3 detected.



```
InceptionV3 Model Predictions
1. crossword_puzzle: 10.80%
2. envelope: 3.39%
3. can_opener: 3.05%
4. comic_book: 2.63%
5. book_jacket: 1.63%
6. rifle: 1.42%
7. jersey: 1.27%
8. bassoon: 1.23%
9. torch: 1.13%
10. jigsaw_puzzle: 1.12%
```

```
Xception Model Predictions
1. jigsaw_puzzle: 11.70%
2. comic_book: 8.36%
3. corkscrew: 2.14%
4. book_jacket: 1.92%
5. sax: 1.76%
6. totem_pole: 1.69%
7. rubber_eraser: 1.66%
8. crutch: 1.47%
9. cornet: 1.45%
10. crossword_puzzle: 1.36%
```

Figure 73 me2.bmp InceptionV3 and Xception

Table 26 displays the tabulated mean values of InceptionV3 and Xception models. The dispersion for the Xception model is lower compared with the Inception model as observed by a narrower inter-quartile range in Figure 74. The narrower bands from the Xception results indicate that we could expect a more stable (less variation or fluctuation from the average values) and possibly higher accuracy than InceptionV3.

Table 26 Comparison of Means between InceptionV3 and Xception

Model	accuracy	sensitivity	specificity	Precision	FPR	MCC	Time*
InceptionV3	0.92	0.99	0.90	0.54	0.10	0.62	2.03
Xception	0.95	0.99	0.93	0.58	0.07	0.67	1.65

*processing time in second per picture

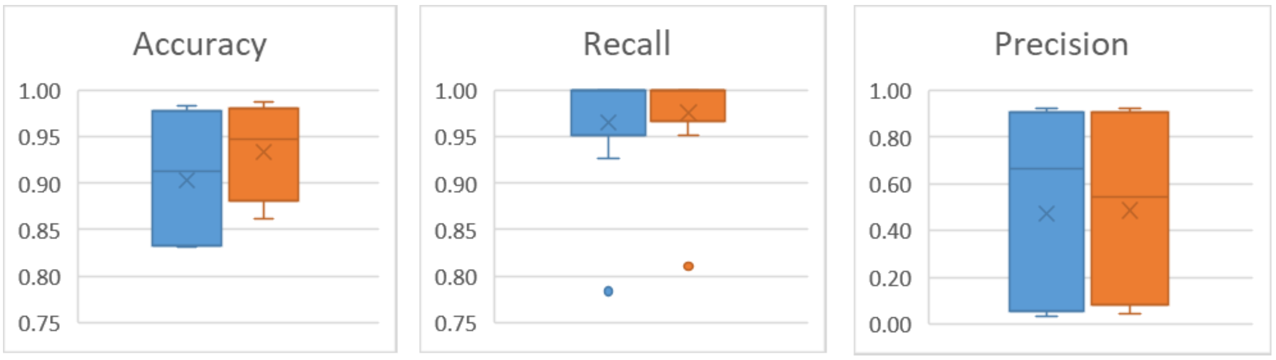


Figure 74 Box plot results - InceptionV3 vs Xception

Table 27 Inception vs Xception Test Results

Dataset	Model	Accuracy	Sensitivity	Specificity	Precision	FPR[1]
ANDROID	InceptionV3	0.83	0.78	0.83	0.06	0.17
ANDROID	Xception	0.86	0.81	0.86	0.07	0.14
JANE	InceptionV3	0.84	0.95	0.84	0.06	0.16
JANE	Xception	0.88	1.00	0.88	0.08	0.12

Table 27 shows that running the results through the testing dataset (Android and Jane datasets) tells us that InceptionV3 model is outperformed by Xception but not by a large margin. However, in the Jane dataset, an exceptional recall rate of 100% was achieved by the Xception model.

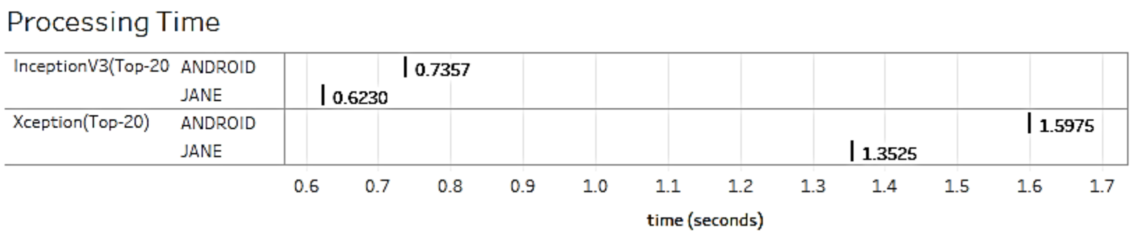


Figure 75 Processing Time: InceptionV3 and Xception

At the expense of accuracy and sensitivity, the processing time per image increases. Processing time is seen as the ratio of the time it takes to complete the classification against the total number of processed images. As seen in Figure 75, in the Jane dataset, the InceptionV3 model completed 10 minutes earlier. InceptionV3 completed in 1 hour 9 minutes and 55 seconds while Xception completed in 1 hour 29 minutes and 55 seconds.

Identifying the best performing model cannot be easily identified even by intuition or graphical methods. A statistical non-parametric test called McNemar’s test is employed to verify whether the two models have a statistical difference. This method is based on the accuracy of the models.

5.4.2.1 Comparing Models using McNemar’s Test

A suggested method to compare models using hypothesis testing is called McNemar’s test which is suggested by Kohavi [79] to establish goodness of fit between two classifier

[1] False positive rate

models. McNemar’s test is a non-parametric[1] statistical test for comparing the accuracy of two models to determine if the models differ (reject the null hypothesis) or if they are statistically equal (accept the null hypothesis). It is important to note that this model does not check for accuracy as metric, but it checks the skewedness of the disagreements between the two models. The 2x2 contingency matrix for McNemar’s Test is constructed as a comparison for each model. Like the confusion matrix for a classifier, instead of Ground Truth, the second model is used as a comparison. The table for McNemar’s is constructed in Table 28.

Table 28 McNemar's Contingency Table

		Model 2	**Model 2**
		Correct	Wrong
Model 1	Correct	**A**	**B**
Model 2	Wrong	**C**	**D**

Let **B** be the number of images that the InceptionV3 model predicts correctly while Xception has it wrong. Let **C** be the number of images that Xception predicts correctly and InceptionV3 has it wrong.

If the null hypothesis that both models have the same performance are true, then B = C and the test statistics follow the chi-square distribution with 1 degree of freedom using the continuity correction [79]. To compute for the chi-square test statistic, we use Equation 15.

$$\chi^2 = \frac{(|\mathrm{B} - \mathrm{C}| - 1)^2}{B + C}$$

Equation 15 Chi-squared Test Statistic

The contingency table is constructed based on the predicted output labels and not reconstructed from each classifier’s confusion matrix values. Since the computation highlights the differences, the values where the models do not differ (A and D) do not need to be computed. Table 29 shows the McNemar’s table populated with results from Android and Jane datasets.

[1] No assumptions on the underlying data distribution (e.g. does not assume normality holds)

Table 29 McNemar's Contingency Table with Values

		Xception	
		Correct	Wrong
InceptionV3	Correct	-	385
	Wrong	628	-

As seen in Table 29, intuition tells us that model 2 (Xception) is the better model because it was able to predict 628 images correctly while model 1 (InceptionV3) got 385 wrong. However, we can substantiate the claim by running a statistical hypothesis test using McNemar's method.

For this test, we hypothesize (null hypothesis, H_0) that the probabilities of p(B) and p(C) are the same (no significant difference between the two models). The alternate hypothesis (H_1) supports that the probability distribution differs. The chosen significance level is $\alpha = 0.05$, if the computed p-value is lower than the critical chi-squared level, then we accept our null hypothesis that the two models are equal. The critical value is $\chi^2_{crit} = 3.84$, for probability (p-value) 0.05 and 1 degree of freedom. The computed chi-squared statistic based on Equation 15Equation 15 is $\chi^2 = 57.81$. Since $3.84 < 57.81$, we reject the null hypothesis: *there is a statistical difference between the accuracy values of InceptionV3 and Xception models*.

5.4.3 Summary of Metrics

The Table 30 and Table 31 summarise the class-performance metrics taken from calculating the results of InceptionV3 and Xception, and the forensic software Magnet Axiom evaluated from the Android and Jane datasets.

Table 30 Inception, Xception and Magnet - Confusion Table

Dataset	Model	TP	FP	FN	TN
ANDROID	InceptionV3	29	496	8	2489
ANDROID	Magnet Axiom	27	12	11	3186
ANDROID	Xception	30	409	7	2576
JANE	InceptionV3	41	628	2	3318
JANE	Magnet Axiom	22	59	21	3999
JANE	Xception	43	475	0	3471

Table 31 Inception, Xception and Magnet

Dataset	Model	Accuracy	Sensitivity	Precision	FPR
ANDROID	InceptionV3	0.83	0.78	0.06	0.17
ANDROID	Magnet Axiom	0.99	0.71	0.69	0.00
ANDROID	Xception	0.86	0.81	0.07	0.14

JANE	InceptionV3	0.84	0.95	0.06	0.16
JANE	Magnet Axiom	0.98	0.51	0.27	0.01
JANE	Xception	0.88	1.00	0.08	0.12

Comparing Models using AUC

We can verify which of the models perform better using the area under the curve (AUC) metric. Table 32 shows AUC values for each row. Since AUC's ideal value is 1, the best value among the three is Xception. Based on the results, Xception model outperforms the rest of the models, and this conclusion has been supported in the preceding chapter using McNemar's test of significant difference and the values in Table 31. Figure 76 is the graphical representation of all three AUCs.

Table 32 AUC values

Model	**AUC**
InceptionV3	0.856
Magnet Axiom	0.851
Xception	0.892

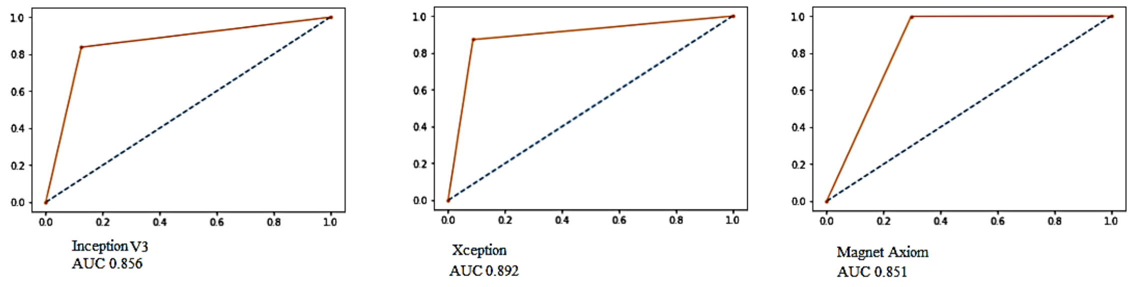


Figure 76 ROC-AUC Summary

The computations are given in Appendix 7 – Area Under the Curve Computation.

Based on the above findings, this tells us that generally, the tested computer vision models (InceptionV3 and Xception) that are pre-trained in the ImageNet dataset outperform a forensic software tool, Magnet Axiom with a trained neural network core, in ensuring less missed classified guns, i.e. the models have a higher recall rate. Although Magnet Axiom has remarkable accuracy, the study uses Sensitivity (Recall) as the deciding factor in evaluating models.

This is promising because having created an open-source tool - that we could improve core machine learning models, perhaps even cater for running multiple models and apply the best model for use in practice.

5.5 Future Work

There is a growing interest in machine learning applications in cybersecurity and forensics. The adoption of machine learning techniques in forensic investigations is not new but the creation of open-source tools employing machine learning theories for practical implementation is noteworthy. This study can be used on exploring visualisation on content-based searches for e-discovery applications, employ *transfer learning* to enhance neural network models by fine-tuning or feature extraction to improve accuracy and precision and further evaluate models based on real data from real cases.

In digital forensics, content-based searches have unlimited potential applications. Searching for content can be expanded to searching for similar photos from other cases, evidence detection based on photos and videos, determining make and model of specific objects such as firearms and explosives, extracting and determining content from screenshots, triaging based on photos and video categorization, and many more.

Regarding this study, most of the statistical analysis done in this research has been severely limited by the low number of respondents during the evaluation phase of the tool. As the prototype is still in its infancy, future research could focus on photo categorization of real cases (with anonymised data), including more filtered images in training and test set, and testing by at least ten users. It is also encouraged to get respondents from different users and potential users – that is, from forensic students, technical examiners and senior consultants with a varying level of expertise. The constructive feedback of the professional forensic community is invaluable in encouraging the development and improvement of similar open-source tools.

Finally, work is underway for the integration of this study to one of the mainstream open-source tools - SleuthKit.

6 Conclusion

This study demonstrated the feasibility of adopting a state-of-the-art machine learning model and create an open-source tool to aid forensic examiners in forensic investigations. In this study, we had taken the *gun* (revolver, pistol, rifle, assault rifle, muzzle) as the instance class (the output label of a binary classifier). The performance of the models is measured by the number of pictures the models have correctly or erroneously classified. Attempting a binary classifier is challenging because of the infinitely large number of images under the not-gun category. Using a multi-class classifier to solve this binary classifier dilemma, we had demonstrated that pre-trained classifiers can achieve the ideal 100% recall rate during evaluation.

The dataset for evaluation and testing used realistic data, simulated with balanced and unbalanced datasets. Balanced datasets have an equal number of guns and not-gun pictures; unbalanced, otherwise. In this study, there were three balanced datasets (WEAPON-DB3, WEAPON-DB5, and WEAPON-DF) while the rest of the eight datasets were unbalanced. Gun images were taken from *jpg/jpeg*, *gif*, *png* and *bmp* file formats downloaded from the Internet, some extracted in thumbnail form while some contained manufactured EXIF information as part of the testing dataset for the prototype.

The model selection was divided into two phases: model evaluation and model selection. The selected computer vision models from the Keras framework were: *InceptionV3*, *VGG16*, *ResNet*, and *Xception*. The model evaluation phase aimed to find the best model from the four selected models based on their performance on the WEAPON-DB3 dataset – the test dataset taken from Olmos *et al.*'s paper [15]. The evaluation criteria include the processing time per image and the class-based performance metrics generated from the confusion matrix, namely: sensitivity (recall), precision, false positive rate (FPR), and Matthew's correlation coefficient (MCC) – with recall as the priority. Both InceptionV3 and Xception scored the highest recall rate at 100% - thus, only these two models were further evaluated. Comparing processing time, precision, false positive rate and MCC metrics, InceptionV3 outperformed Xception, and was therefore chosen as the best model during the evaluation phase.

The model selection phase used the best model to estimate the performance of the model with unseen data by evaluating it against nine datasets. InceptionV3 performed reasonably well, with an estimated performance expressed as a five-number summary[1]: accuracy (0.71 ,0.86, 0.93, 0.97, 1.00) and mean of 0.92, sensitivity (1, 1, 1, 1, 1) with mean of 0.99, precision (0.00, 0.05, 0.67, 0.90, 1.00) with mean of 0.56, FPR (0.00, 0.03, 0.10, 0.15, 0.33) with mean of 0.10, MCC (0.00, 0.21, 0.81, 0.86, 1.00) with mean of 0.94, and processing time, seconds per picture (0.00, 0.87, 1.43, 1.56, 1.00) with mean of 2.03. InceptionV3 was the final model used as the prototype's core classifier.

InceptionV3 was tested on two unseen datasets (Android and Jane) with accuracy, precision, FPR and MCC within the expected range (i.e. $\pm$1.5IQR[2]). The sensitivity value in the Android dataset produced an outlier which might be explained by image filters distorting the colour and contrast of the images. Images with filters should be taken into consideration when planning to re-train models to improve accuracy.

The evaluation of the computer vision models used the Jupyter notebook while the prototype was developed using the Flask-Python 3 micro-framework with a TensorFlow abstraction architecture and Keras library support. Released in a test-and-measure approach, the users were given questions in the form of scenarios. The usability test using the System Usability Scale (SUS) approach was conducted at the end to measure usability (is the prototype usable in a forensic investigation?) and learnability (can the prototype be used with ease?).

The prototype had an overall SUS mean score of 69 (Average); the baseline was set at 68. Based on the overall mean score and item-level scores of the SUS computation, the findings showed an "Acceptable" level in both usability and learnability. Four out of five respondents feel that the prototype could be used in the field if the model and visualisation is improved. Thus, the results of the SUS demonstrated that a pre-trained classifier such as InceptionV3 can be used (usable and learnable) to aid forensic investigations.

[1] The numerical representation of the box-plot is a five-number summary: (Q1-1.5IQR, Q1, Median, Q3, Q3+1.5IQR) based on Tukey's method where IQR is Inter-Quartile Range
[2] Inter-Quartile Range

To measure the efficiency of the prototype, the time to complete the task using both manual method and the prototype is recorded. Using both paired t-test (normality holds) and Wilcoxon signed rank test (normality does not hold), and the speed of performance showed no significant difference between the manual method and the use of the prototype, that is, it is neither worse nor better.

Although a small sample size was deemed sufficient for an early-phase usability test, the underlying statistical analysis conducted, such as the test for efficiency or correlation between correct answers and duration, was affected by the low statistical power due to the insufficient number of respondents. A larger difference was expected to be able to conclude a significant difference between the two methods. Other than the small pool of testers, the number of test data examined was also considered as one of the contributing factors. The test cases barely covered half of the minimum estimated case load and only a tenth for organised crime.

Alternatives to InceptionV3 was also covered in this study: Magnet Axiom and the Xception computer vision model. Although Magnet Axiom, a forensic software tool that automatically categorises possible weapons, produced higher accuracy and precision because of the fewer false positives, the number of missed guns were higher. The performance for sensitivity in the Jane dataset was lowest at around 51%.

As an alternative to InceptionV3, we had further analysed the results from the Xception model (the model that had also achieved a 100% recall rate during model evaluation) using the nine other datasets used during model selection. The McNemar's test was used to evaluate the difference between the models. The findings showed that there is a statistical difference between InceptionV3 and Xception models. The test, however, did not prove which model is better, but only established that the two models do not come from the same distribution. The Area Under the Curve (AUC) metric was used to support the claim that Xception (AUC = 0.892) outperformed the InceptionV3 model (AUC = 0.856) by a significant difference. Although the recall rate for both models were the same (0.99), the overall means reported for other metrics were better: accuracy at 0.95, specificity at 0.93, precision at 0.58, FPR at 0.07, MCC at 0.67 and processing time at 1.65 seconds per picture.

The findings showed that the computer vision models we had tested: InceptionV3 and Xception, both pre-trained in the ImageNet dataset, outperformed Magnet Axiom in ensuring less missed classified guns, *i.e.* the models had a higher recall rate as this paper used sensitivity (recall) as the deciding factor in evaluating models.

Thus, we demonstrated that existing state-of-the-art classifiers (such as InceptionV3 and Xception models) trained in non-forensic data (ImageNet) produced acceptable results when presented to forensic examiners and potential forensic examiners as a tool for forensic investigation. This is indeed promising because both InceptionV3 and Xception models are powerful base classifiers and could be further improved by transfer learning techniques such as fine-tuning and feature extraction.

As the prototype is still in its infancy, future research could focus on photo categorization of real cases (with anonymised data), including more filtered images in training and test set, and testing by at least ten users. Future improvements on the prototype could also be expanded to searching for similar photos from other cases, evidence detection based on photos and videos, determining make and model of specific objects such as firearms and explosives, extracting and determining content from screenshots, triaging based on photos and video categorization. It is also encouraged to get respondents from different users and potential users – that is, from forensic students, technical examiners and senior consultants with a varying level of expertise. The constructive feedback of the professional forensic community is invaluable in encouraging the development and improvement of similar open-source tools.

The research study produced an open-source code, and thus the code could be easily validated by a human expert. The source code is available in GitHub at https://github.com/jrdelmar/cbis with the full installation guide for the console and the web application at https://cbis.jrdelmar.dev/help. Working demo is online at https://cbis.jrdelmar.dev/.

References

[1] NIST (National Institute of Standards and Technology), “Scientists Automate Key Step in Forensic Fingerprint Analysis,” 14 August 2017. [Online]. Available: https://www.nist.gov/news-events/news/2017/08/scientists-automate-key-step-forensic-fingerprint-analysis. [Accessed 15 October 2018].

[2] Chris Baraniuk, “The New Weapon in the Fight against Crime,” BBC, 4 3 2019. [Online]. Available: http://www.bbc.com/future/story/20190228-how-ai-is-helping-to-fight-crime. [Accessed 29 3 2019].

[3] X. H. ,. V. I. M. ,. L. S. D. Peng Zhou, “Learning Rich Features for Image Manipulation Detection,” *CVPR,* vol. abs/1805.04953, 2018.

[4] SanDisk, “Number of pictures that can be stored on a memory device,” SanDisk, [Online]. Available: https://kb.sandisk.com/app/answers/detail/a_id/69/~/number-of-pictures-that-can-be-stored-on-a-memory-device. [Accessed 20 9 2018].

[5] D. W. I. B. F. B. Vikram S. Harichandran, “CuFA: A more formal definition for digital forensic artifacts,” *Digital Investigation,* vol. 18, pp. S125-S137, 2016,.

[6] I. S. G. H. Alex Krizhevsky, “ImageNet Classification with Deep Convolutional Neural Networks,” *Communications of the ACM,* vol. 60, no. 6, p. 84–90, 24 5 2017.

[7] J. D. H. S. ,. J. K. ,. S. S. ,. S. M. ,. Z. H. ,. A. K. a. A. K. ,. M. S. B. ,. A. C. B. ,. L. F.-F. Olga Russakovsky, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, pp. 211-252, 2015.

[8] Peter Eckersley and Yomna Nasser, “Measuring the Progress of AI Research,” EFF AI Progress Measurement Project, 9 9 2017. [Online]. Available: https://www.eff.org/ai/metrics.

[9] S. Melendez, “How Pinterest Uses Machine Learning To Keep Its Users Pinned,” Fast Company, 21 November 2016. [Online]. Available: https://www.fastcompany.com/3065228/how-pinterest-uses-machine-learning-to-keep-its-users-pinned. [Accessed 24 November 2018].

[10] M. Lewontin, “How Google Photos uses machine learning to create customized albums,” 24 March 2016. [Online]. Available: https://www.cnet.com/news/google-photos-live-albums-automatically-adds-and-organizes-your-photos/. [Accessed 24 November 2018].

[11] A. Hern, “Facebook is using AI to tag your pictures to help blind people,” 5 April 2016. [Online]. Available: https://www.theguardian.com/technology/2016/apr/05/facebook-ai-tag-pictures-blind-people-machine-learning. [Accessed 24 November 2018].

[12] E. E. Kenneally, “Gatekeeping out of the Box: Open Source Software as a Mechanism to Assess Reliability for Digital Evidence,” *Virginia Journal of Law & Technology,* vol. 6, no. 3, pp. 1-38, 2001.

[13] B. Carrier, “Open Source Digital Forensics Tools : The Legal Argument,” 2002.

[14] David Robinson, “Stack Overflow Trends,” Stack Overflow , 2017. [Online]. Available: https://insights.stackoverflow.com/trends?tags=java%2Cc%2Cc%2B%2B%2Cpython%2Cc%23%2Cvb.net%2Cjavascript%2Cassembly%2Cphp%2Cperl%2Cruby%2Cvb%2Cswift%2Cr%2Cobjective-c&utm_source=so-owned&utm_medium=blog&utm_campaign=gen-blog&utm_content=blog-link&utm_term=.

[15] S. T. a. F. H. Roberto Olmos, “Automatic Handgun Detection Alarm in Videos,” *Neurocomputing,* vol. 275, pp. 66-72, 2018.

[16] J. R. d. Mar, “Content-based Image Search using pre-trained Keras model with Python3 and Flask,” [Online]. Available: https://github.com/jrdelmar/cbis.

[17] J. R. d. Mar, “Dataset for content-based image search,” [Online]. Available: https://github.com/jrdelmar/cbis-dataset.

[18] C. S. a. V. V. a. S. I. a. J. S. a. Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *CoRR,* vol. abs/1512.00567, 2015.

[19] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *CoRR,* vol. abs/1610.02357, 2016.

[20] X. Z. S. R. J. S. Kaiming He, “Deep Residual Learning for Image Recognition,” *CoRR,* vol. abs/1512.03385.

[21] A. Z. Karen Simonyan, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd International Conference on Learning Representations, {ICLR} 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings,* 2015.

[22] T. V. L. C. Hanan Hibshi, “Usability of Forensics Tools: A User Study,” in *2011 Sixth International Conference on IT Security Incident Management and IT Forensics*, 2011.

[23] D. J. B. a. P. Stephens, “A cognitive walkthrough of Autopsy Forensic Browser,” *Information Management & Computer Security,* vol. 17, no. 1, pp. 20-29, 2009.

[24] ISO 9241-11:2018(en), “ISO 9241-11:2018(en) | Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts,” ISO, 2018.

[25] J. Nielsen, Usability Engineering, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[26] M. a. A. M. a. C. C. Delgado, “Using open source for forensic purposes,” in *OSDOC '12: Proceedings of the Workshop on Open Source and Design of Communication*, 2012.

[27] “Tensorflow, an end-to-end open source machine learning platform,” Tensorflow, [Online]. Available: https://www.tensorflow.org/. [Accessed 31 3 2019].

[28] J. D. H. S. ,. J. K. ,. S. S. ,. S. M. ,. Z. H. ,. A. K. a. A. K. ,. M. S. B. ,. A. C. B. ,. L. F.-F. Olga Russakovsky, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, pp. 211-252, 2015.

[29] R. Girshick, “Deep Learning for Instance-level Object Understanding,” in *CVPR 2017 Tutorial on Deep Learning for Objects and Scenes*, 2017.

[30] V. Nigam, “Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning,” 11 9 2018. [Online]. Available: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90. [Accessed 1 4 2019].

[31] A. Karpathy, “CS231 Convolutional Neural Networks for Visual Recognition,” 2018. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed 2 11 2018].

[32] M. K. Thomas Gloe, “Can We Trust Digital Image Forensics?,” *ACM Multimedia,* 2007.

[33] W. T. ,. J.-L. D. Judith Redi, “Digital image forensics: a booklet for beginners,” *Multimedia Tools and Applications,* vol. 51, pp. 133-162, 2010.

[34] Adobe.com, “Spotting Image Tampering with AI,” 23 July 2018. [Online]. Available: https://research.adobe.com/spotting-image-tampering-with-ai/. [Accessed 22 September 2018].

[35] A. o. C. P. O. ACPO, “ACPO Good Practice Guide for Digital Evidence,” October 2011. [Online]. Available: https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf. [Accessed 12 October 2018].

[36] H. M. J. a. M. W. Bailie, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” Washington, DC, Office of Legal Education Executive Office for United States Attorney, 2009.

[37] J. L. S. D. W. H. Michael B. Mukasey, Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition, Washington, DC: National Institute of Justice , 2008.

[38] A. L. ,. G. M. ,. M. O. Antonio Grillo, “Fast user classifying to establish forensic analysis priorities,” *2009 Fifth International Conference on IT Security Incident Management and IT Forensics,* pp. 69-77, 2009.

[39] A. P.-W. D. H. a. J. M. Simson L. Garfinkel, “An Automated Solution to the Multiuser Carved Data Ascription Problem,” *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 5, no. 4, pp. 868-882, 2010.

[40] G. M. ,. R. B. ,. S. T. Fabio Marturana, “A Quantitative Approach to Triaging in Mobile Forensics,” *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications,* pp. 582-588, 2011.

[41] L. S. M. GÓMEZ, “Triage in-Lab: case backlog reduction with forensic digital profiling,” *Simposio Argentino de Informática y Derecho,* 2012.

[42] S. T. Fabio Marturana, “A Machine Learning-based Triage methodology for automated categorization of digital media,” *Digital Investigation,* vol. 10, no. 2, pp. 193-204, 2013.

[43] D. M. a. F. Marturana, “A Digital Forensics Triage methodology based on feature manipulation techniques,” *2014 IEEE International Conference on Communications Workshops (ICC),* pp. 676-681, 2014.

[44] S. Ł. R. S. Michał Grega, “Automated Recognition of Firearms in Surveillance Video,” *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA),* 2013.

[45] G. K. V. Rohit Kumar Tiwari, “A Computer Vision based Framework for Visual Gun Detection using SURF,” *International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO),* 2015.

[46] G. K. V. a. A. Dhillon, “A Handheld Gun Detection using Faster R-CNN Deep Learning,” *Proceedings of the 7th International Conference on Computer and Communication Technology,* pp. 84--88, 2017.

[47] A. Karpathy, “What I learned from competing against a ConvNet on ImageNet,” 4 September 2014. [Online]. Available: https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/. [Accessed 11 November 2018].

[48] Peter Eckersley, Yomna Nasser, Yann Bayle, Owain Evans, Gennie Gebhart and Dustin Schwenk., “Measuring the Progress of AI Research,” EFF AI Progress Measurement Project, 9 9 2017. [Online]. Available: https://eff.org/ai/metrics.

[49] S. L. Garfinkel, “Digital forensics research: The next 10 years,” in *10th annual conference on digital forensic research workshop*, 2010.

[50] S. Raghavan, “Digital forensic research: current state of the art,” *CSI Transactions on ICT,* vol. 1, pp. 91-114, 2012.

[51] B. B. ,. T. O. a. M. S. David Lillis, “CURRENT CHALLENGES AND FUTURE RESEARCH AREAS FOR DIGITAL FORENSIC INVESTIGATION,” *CoRR,* vol. abs/1604.03850, 2016.

[52] D. B. a. E. G. Vacius Jusas 1, “Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions,” *Symmetry,* vol. 9, p. 49, 2017;.

[53] T. T. ,. M. A. R. &. S. C. Ken Peffers, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems,* vol. 24, no. 3, pp. 45-77, 2007.

[54] E. Kutcher, “THUMBCACHE VIEWER,” 17 5 2018. [Online]. Available: https://thumbcacheviewer.github.io/.

[55] P. Harvey, “Read, Write and Edit Meta Information!,” 6 10 2007. [Online]. Available: https://web.mit.edu/Graphics/src/Image-ExifTool-6.99/html/.

[56] D. W. ,. T. H. Gareth James, An Introduction to Statistical Learning with Applications in R, Springer, 2017.

[57] D. Powers, “EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC,INFORMEDNESS, MARKEDNESS & CORRELATION,” *Journal of Machine Learning Technologies,* vol. 2, no. 1, pp. 37-63, 2011.

[58] J. Joseph, “The Best Metric to Measure Accuracy of Classification Models,” 28 November 2016. [Online]. Available: https://clevertap.com/blog/the-best-metric-to-measure-accuracy-of-classification-models/. [Accessed 9 November 2018].

[59] D. Chicco, “Ten quick tips for machine learning in computational biology,” *BioData Mining,* 2017.

[60] T. F. R. K. Foster Provost, “The Case Against Accuracy Estimation for Comparing Induction Algorithms,” *Proceeding of the 15th International Conference on Machine Learning,* p. 445–453, 1998.

[61] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters,* vol. 27, p. 861–874, 2006.

[62] A. J. R. R. Jorge M. Lobo, “AUC: a misleading measure of the performance of predictive distribution models,” *Global Ecology and Biogeography,* vol. 17, no. 2, pp. 145-151, 2008.

[63] R. C. H. Chris Drummond, “Explicitly Representing Expected Cost: An Alternative to ROC Representation,” *Proceeding of Knowledge Discovery and Datamining,* p. 198–207, 2000.

[64] M. G. Jesse Davis, “The Relationship Between Precision-Recall and ROC Curves,” *Proceedings of the 23rd International Conference on Machine Learning,* p. 233–240, 2006.

[65] P. T. K. &. J. T. M. Aaron Bangor, “The System Usability Scale (SUS): an Empirical evaluation,” *International Journal of Human-Computer Interaction,* vol. 24, no. 6, pp. 574--594, 2008 .

[66] P. J. S. James R. (Jim) Lewis, “Item Benchmarks for the System Usability Scale,” *Journal of Usability Studies,* vol. 14, no. 2, pp. 158-167, 2018.

[67] usability.gov, “System Usability Scale (SUS) | Usability.gov,” [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [Accessed 1 10 2018].

[68] A. a. J. J. A. Sears, “Usability Testing Basics,” in *Human-Computer Interaction: Development Process*, Boca Raton, FL, USA, CRC Press, Inc., 2009, pp. 232-236.

[69] J. Brooke, “SUS: A “quick and dirty” usability scale,” *Usability evaluation in industry (P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.)),* pp. 189-194, 1996.

[70] J. a. L. T. K. Nielsen, “A Mathematical Model of the Finding of Usability Problems,” in *ACM INTER CHI'93 Conference*, Amsterdam, The Netherlands, 1993.

[71] J. Sauro, “10 Things to Know About the System Usability Scale (SUS),” 18 6 2013. [Online]. Available: https://measuringu.com/10-things-sus/. [Accessed 8 4 2019].

[72] J. R. L. a. J. Sauro, “The Factor Structure of the System Usability Scale,” in *HCD 09 Proceedings of the 1st International Conference on Human Centered Design: Held as Part of HCI International 2009*, San Diego, CA, 2009.

[73] A. T. a. A. A. Efros, “Unbiased look at dataset bias,” *CVPR 2011,* pp. 1521-1528, 2011.

[74] N. M. R. a. Y. B. Wah, “Power comparisons of Shapiro-Wilk , Kolmogorov-Smirnov , Lilliefors and Anderson-Darling tests,” *Journal of Statistical Modeling and Analytics ,* vol. 2, no. 1, pp. 21-33, 2011.

[75] R. F. WOOLSON, “Wilcoxon Signed-Rank Test,” *Wiley Encyclopedia of Clinical Trials,* 2008.

[76] P. Jeff Sauro, “10 THINGS TO KNOW ABOUT TASK TIMES,” 9 8 2011. [Online]. Available: https://measuringu.com/task-times/.

[77] J. S. W. H. Weirsma W, Applied Statistics for the Behavioral Sciences, 5th edition, Boston: Houghton Mifflin, 2003.

[78] P. Jeff Sauro, “5 Ways to Interpret a SUS Score,” 19 9 2018. [Online]. Available: https://measuringu.com/interpret-sus-score/. [Accessed 11 4 2019].

[79] R. Kohavi, “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, Quebec, Canada, 1995.

[80] J. Brownlee, “How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python,” 31 8 2018. [Online]. Available: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.

[81] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[82] D. K. Gaurav, “OpenSource,” 8 September 2016. [Online]. Available: https://opensourceforu.com/2016/09/deep-learning-network-packet-forensics-using-tensorflow/. [Accessed 15 October 2018].

[83] T. a. K. M. a. W. A. a. B. R. Gloe, “Can We Trust Digital Image Forensics?,” in *15th ACM International Conference on Multimedia*, Augsburg, Germany, 2007.

[84] W. T. ,. J.-L. D. Judith Redi, “Digital image forensics: a booklet for beginners,” *Multimedia Tools and Applications,* vol. 51, pp. 133-162, 2010.

[85] G. G. ,. F. R. Davide Ariu, “Machine learning in computer forensics (and the lessons learned from machine learning in computer security),” in *AISec*, Chicago, 2011.

[86] J. Mena, “Machine Learning Forensics for Law Enforcement, Security and Intelligence,” Auerbach Publications, 2011. [Online]. Available: http://www.infosectoday.com/Articles/Machine_Learning_Forensics/Machine_Learning_Forensics.htm. [Accessed 15 October 2018].

[87] N. BILTON, “FAKE NEWS IS ABOUT TO GET EVEN SCARIER THAN YOU EVER DREAMED,” Vanity Fair, 26 January 2017. [Online]. Available: https://www.vanityfair.com/news/2017/01/fake-news-technology. [Accessed 11 November 2018].

[88] T. SRIVASTAVA, 19 February 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/. [Accessed 8 November 2018].

[89] W. D. R. S. L.-J. L. K. L. a. L. F.-F. Jia Deng, “ImageNet: A Large-Scale Hierarchical Image Database,” *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2009.

[90] S. T. M. J. P. S. R. Alan R. Hevner, “DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH1,” *MIS Quarterly,* vol. 28, no. 1, pp. 75-105, 2014.

[91] M. Rouaud, “Probability, Statistics and Estimation,” PDF, 2013.

[92] A. Karpathy, “Lessons learned from manually classifying CIFAR-10,” 27 April 2011. [Online]. Available: https://karpathy.github.io/2011/04/27/manually-classifying-cifar10/.

Appendix 1 – ImageNet Classes

ImageNet classes[1] used in Keras. The highlighted labels are classified as guns.

Afghan_hound African_chameleon African_crocodile African_elephant African_grey African_hunting_dog Airedale American_Staffordshire_terrier American_alligator American_black_bear American_chameleon American_coot American_egret American_lobster Angora Appenzeller Arabian_camel Arctic_fox Australian_terrier Band_Aid Bedlington_terrier Bernese_mountain_dog Blenheim_spaniel Border_collie Border_terrier Boston_bull Bouvier_des_Flandres Brabancon_griffon Brittany_spaniel CD_player Cardigan Chesapeake_Bay_retriever Chihuahua Christmas_stocking Crock_Pot Dandie_Dinmont Doberman Dungeness_crab Dutch_oven Egyptian_cat English_foxhound English_setter English_springer EntleBucher Eskimo_dog European_fire_salamander European_gallinule French_bulldog French_horn French_loaf German_shepherd German_short-haired_pointer Gila_monster Gordon_setter Granny_Smith Great_Dane Great_Pyrenees Greater_Swiss_Mountain_dog Ibizan_hound Indian_cobra Indian_elephant Irish_setter Irish_terrier Irish_water_spaniel Irish_wolfhound Italian_greyhound Japanese_spaniel Kerry_blue_terrier Komodo_dragon Labrador_retriever Lakeland_terrier Leonberg Lhasa Loafer Madagascar_cat Maltese_dog Mexican_hairless Model_T Newfoundland Norfolk_terrier Norwegian_elkhound Norwich_terrier Old_English_sheepdog Pekinese Pembroke Persian_cat Petri_dish Polaroid_camera Pomeranian Rhodesian_ridgeback Rottweiler Saint_Bernard Saluki Samoyed Scotch_terrier Scottish_deerhound Sealyham_terrier Shetland_sheepdog Shih-Tzu Siamese_cat Siberian_husky Staffordshire_bullterrier Sussex_spaniel Tibetan_mastiff Tibetan_terrier Walker_hound Weimaraner Welsh_springer_spaniel West_Highland_white_terrier Windsor_tie Yorkshire_terrier abacus abaya academic_gown accordion acorn acorn_squash acoustic_guitar admiral affenpinscher agama agaric aircraft_carrier airliner airship albatross alligator_lizard alp altar ambulance amphibian analog_clock anemone_fish ant apiary apron armadillo artichoke ashcan **assault_rifle** axolotl baboon backpack badger bagel bakery balance_beam bald_eagle balloon ballplayer ballpoint banana banded_gecko banjo bannister barbell barber_chair barbershop barn barn_spider barometer barracouta barrel barrow baseball basenji basketball basset bassinet bassoon bath_towel bathing_cap bathtub beach_wagon beacon beagle beaker bearskin beaver bee bee_eater beer_bottle beer_glass bell_cote bell_pepper bib bicycle-built-for-two bighorn bikini binder binoculars birdhouse bison bittern black-and-tan_coonhound black-footed_ferret black_and_gold_garden_spider black_grouse black_stork black_swan black_widow bloodhound bluetick boa_constrictor boathouse bobsled bolete bolo_tie bonnet book_jacket bookcase bookshop borzoi bottlecap bow bow_tie box_turtle boxer brain_coral brambling brass brassiere breakwater breastplate briard broccoli broom brown_bear bubble bucket buckeye buckle bulbul bull_mastiff bullet_train bulletproof_vest bullfrog burrito bustard butcher_shop butternut_squash cab cabbage_butterfly cairn caldron can_opener candle cannon canoe capuchin car_mirror car_wheel carbonara cardigan cardoon carousel carpenter's_kit carton cash_machine cassette cassette_player castle catamaran cauliflower cello cellular_telephone centipede chain chain_mail chain_saw chainlink_fence chambered_nautilus cheeseburger cheetah chest chickadee chiffonier chime chimpanzee china_cabinet chiton chocolate_sauce chow church cicada cinema cleaver cliff cliff_dwelling cloak clog clumber cock cocker_spaniel cockroach cocktail_shaker coffee_mug coffeepot coho coil collie colobus combination_lock comic_book common_iguana common_newt computer_keyboard conch confectionery consomme container_ship convertible coral_fungus coral_reef corkscrew corn cornet coucal cougar cowboy_boot cowboy_hat coyote cradle crane crane crash_helmet crate crayfish crib cricket croquet_ball crossword_puzzle crutch cucumber cuirass cup curly-coated_retriever custard_apple daisy dalmatian dam damselfly desk desktop_computer dhole dial_telephone diamondback diaper digital_clock digital_watch dingo dining_table dishrag dishwasher disk_brake dock dogsled dome doormat dough dowitcher dragonfly drake drilling_platform drum drumstick dugong dumbbell dung_beetle ear earthstar echidna eel eft eggnog electric_fan electric_guitar electric_locomotive electric_ray entertainment_center envelope espresso espresso_maker face_powder feather_boa fiddler_crab fig file fire_engine fire_screen fireboat flagpole flamingo flat-coated_retriever flatworm flute fly folding_chair football_helmet forklift fountain fountain_pen four-poster fox_squirrel freight_car frilled_lizard frying_pan fur_coat gar garbage_truck garden_spider garter_snake gas_pump gasmask gazelle geyser giant_panda giant_schnauzer gibbon go-kart goblet golden_retriever goldfinch goldfish golf_ball golfcart gondola gong goose gorilla gown grand_piano grasshopper great_grey_owl great_white_shark green_lizard green_mamba green_snake greenhouse grey_fox grey_whale grille grocery_store groenendael groom ground_beetle guacamole guenon guillotine guinea_pig gyromitra hair_slide hair_spray half_track hammer hammerhead hamper hamster hand-held_computer hand_blower handkerchief hard_disc hare harmonica harp hartebeest harvester harvestman hatchet hay head_cabbage hen hen-of-the-woods hermit_crab hip hippopotamus hog hognose_snake holster home_theater honeycomb hook hoopskirt horizontal_bar hornbill

[1] https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json

horned_viper horse_cart hot_pot hotdog hourglass house_finch howler_monkey hummingbird hyena iPod ibex ice_bear ice_cream ice_lolly impala indigo_bunting indri iron isopod jacamar jack-o'-lantern jackfruit jaguar jay jean jeep jellyfish jersey jigsaw_puzzle jinrikisha joystick junco keeshond kelpie killer_whale kimono king_crab king_penguin king_snake kit_fox kite knee_pad knot koala komondor kuvasz lab_coat lacewing ladle ladybug lakeside lampshade langur laptop lawn_mower leaf_beetle leafhopper leatherback_turtle lemon lens_cap leopard lesser_panda letter_opener library lifeboat lighter limousine limpkin liner lion lionfish lipstick little_blue_heron llama loggerhead long-horned_beetle lorikeet lotion loudspeaker loupe lumbermill lycaenid lynx macaque macaw magnetic_compass magpie mailbag mailbox maillot maillot malamute malinois manhole_cover mantis maraca marimba marmoset marmot mashed_potato mask matchstick maypole maze measuring_cup meat_loaf medicine_chest meerkat megalith menu microphone microwave military_uniform milk_can miniature_pinscher miniature_poodle miniature_schnauzer minibus miniskirt minivan mink missile mitten mixing_bowl mobile_home modem monarch monastery mongoose monitor moped mortar mortarboard mosque mosquito_net motor_scooter mountain_bike mountain_tent mouse mousetrap moving_van mud_turtle mushroom **muzzle** nail neck_brace necklace nematode night_snake nipple notebook obelisk oboe ocarina odometer oil_filter orange orangutan organ oscilloscope ostrich otter otterhound overskirt ox oxcart oxygen_mask oystercatcher packet paddle paddlewheel padlock paintbrush pajama palace panpipe paper_towel papillon parachute parallel_bars park_bench parking_meter partridge passenger_car patas patio pay-phone peacock pedestal pelican pencil_box pencil_sharpener perfume photocopier pick pickelhaube picket_fence pickup pier piggy_bank pill_bottle pillow pineapple ping-pong_ball pinwheel pirate pitcher pizza plane planetarium plastic_bag plate plate_rack platypus plow plunger pole polecat police_van pomegranate poncho pool_table pop_bottle porcupine pot potpie potter's_wheel power_drill prairie_chicken prayer_rug pretzel printer prison proboscis_monkey projectile projector promontory ptarmigan puck puffer pug punching_bag purse quail quill quilt racer racket radiator radio radio_telescope rain_barrel ram rapeseed recreational_vehicle red-backed_sandpiper red-breasted_merganser red_fox red_wine red_wolf redbone redshank reel reflex_camera refrigerator remote_control restaurant **revolver** rhinoceros_beetle **rifle** ringlet ringneck_snake robin rock_beauty rock_crab rock_python rocking_chair rotisserie rubber_eraser ruddy_turnstone ruffed_grouse rugby_ball rule running_shoe safe safety_pin saltshaker sandal sandbar sarong sax scabbard scale schipperke school_bus schooner scoreboard scorpion screen screw screwdriver scuba_diver sea_anemone sea_cucumber sea_lion sea_slug sea_snake sea_urchin seashore seat_belt sewing_machine shield shoe_shop shoji shopping_basket shopping_cart shovel shower_cap shower_curtain siamang sidewinder silky_terrier ski ski_mask skunk sleeping_bag slide_rule sliding_door slot sloth_bear slug snail snorkel snow_leopard snowmobile snowplow soap_dispenser soccer_ball sock soft-coated_wheaten_terrier solar_dish sombrero sorrel soup_bowl space_bar space_heater space_shuttle spaghetti_squash spatula speedboat spider_monkey spider_web spindle spiny_lobster spoonbill sports_car spotlight spotted_salamander squirrel_monkey stage standard_poodle standard_schnauzer starfish steam_locomotive steel_arch_bridge steel_drum stethoscope stingray stinkhorn stole stone_wall stopwatch stove strainer strawberry street_sign streetcar stretcher studio_couch stupa sturgeon submarine suit sulphur-crested_cockatoo sulphur_butterfly sundial sunglass sunglasses sunscreen suspension_bridge swab sweatshirt swimming_trunks swing switch syringe tabby table_lamp tailed_frog tank tape_player tarantula teapot teddy television tench tennis_ball terrapin thatch theater_curtain thimble three-toed_sloth thresher throne thunder_snake tick tiger tiger_beetle tiger_cat tiger_shark tile_roof timber_wolf titi toaster tobacco_shop toilet_seat toilet_tissue torch totem_pole toucan tow_truck toy_poodle toy_terrier toyshop tractor traffic_light trailer_truck tray tree_frog trench_coat triceratops tricycle trifle trilobite trimaran tripod triumphal_arch trolleybus trombone tub turnstile tusker typewriter_keyboard umbrella unicycle upright vacuum valley vase vault velvet vending_machine vestment viaduct vine_snake violin vizsla volcano volleyball vulture waffle_iron walking_stick wall_clock wallaby wallet wardrobe warplane warthog washbasin washer water_bottle water_buffalo water_jug water_ouzel water_snake water_tower weasel web_site weevil whippet whiptail whiskey_jug whistle white_stork white_wolf wig wild_boar window_screen window_shade wine_bottle wing wire-haired_fox_terrier wok wolf_spider wombat wood_rabbit wooden_spoon wool worm_fence wreck yawl yellow_lady's_slipper yurt zebra zucchini

Appendix 2 – Evaluation Questions and Answer Key

Questions	Answer
Record the time started	
1. In the "Evaluation folder", how many images with guns (revolver, rifle...) did you find?	4
2. What is the filename of the gun picture taken in Croatia?	picture.jpg
3. Were there any gun images potentially taken in Italy?	Yes
4. What is the model of the camera embedded in those gun pictures?	Apple iPhone 6s
5. The suspect uploaded a photo in Instagram of a restaurant in India on July 25, 2017 in the afternoon. Investigators speculated that he stayed the night in this area. Where do you think the suspect could have stayed?	FabHotel Blueberry Hauz Khas
6. Were there any gun images found in this folder?	Yes
7. Were there any gun images found in this folder?	Yes
Record the time completed	
List the tools you used (E.g. Windows Explorer, CLI EXIF Tool, Google maps)	
Record the time started	
Did you install the cbis tool yourself?	
1. Was the number of false-positives (detected as a gun but is not actually a gun) tolerable when performing a quick image search during a forensic task?	
2. What is the filename of the picture taken in Croatia?	picture.jpg
3. Were there any gun images potentially taken in Italy?	Yes
4. What is the model of the camera embedded in those gun pictures?	Apple iPhone 6s
5. The suspect uploaded a photo in Instagram of a restaurant in India on July 25, 2017 in the afternoon. Investigators speculated that he stayed the night in this area. Where do you think the suspect could have stayed?	FabHotel Blueberry Hauz Khas
6. Were there any gun images found in this folder?	Yes
7. Were there any gun images found in this folder?	Yes
Record the time completed	
-- Start of Usability Test ---	
1. I think that I would like to use this tool frequently if I have to deal with categorizing images and getting the exif information.	
2. I find the tool unnecessarily complex.	
3. I think that the tool is easy to use and easy to navigate.	
4. I think that I would need assistance (technical or non-technical) to be able to use this tool.	
5. I find that the various functions (console prediction, search, display exif information, report screen) in this tool are well-integrated. I was easily able to complete the tasks at hand.	
6. I think there is too much inconsistency in the tool. It would be difficult to deploy the tool in the real world.	
7. I think most forensic examiners would learn to use this tool very quickly.	
8. I find the tool inconvenient to use.	
9. I feel confident using this tool.	
10. The learning curve to use this tool is steep. I need to learn a lot of things before I could use the tool.	
-- Image Classification Questions ---	
11. In a forensic investigation, would you prefer to use the prototype than manually working your way through the images using other open-source applications?	
12. In your opinion, based on the Android test set, can the tool be used in forensic work despite the number and the quality of the undetected gun images?	
12.a If you have answered, "No, I prefer the tool to detect 'some' images", please enumerate the images that the tool should have detected.	

13. In your opinion, based on the Jane Doe test set, can the tool be used in forensic work despite the number and the quality of the undetected gun images?
14. Overall, do you think that the tool can be used to aid an actual forensic investigation?
15. Does the visualization (images below) showing overall labels detected in the images useful in a forensic investigation?
How long have you been conducting forensic analysis (school, work or as a hobby)?
I uphold freedom of speech. If you have something to say (that is related to the tool and this task), this space is for you. Shoot.

Appendix 3 – InceptionV3 Evaluation Results

Dataset	accuracy	sensitivity	specificity	precision
DB3-2	0.96	1.00	0.91	0.92
DF-2	0.93	1.00	0.85	0.87
DB5-2	0.91	0.93	0.90	0.90
DB6-2	0.86	0.95	0.86	0.05
DB7-2	0.83	1.00	0.83	0.04
DB8-2	0.83	1.00	0.83	0.04
DB9-2	0.98	1.00	0.98	0.92
DB10-2	0.98	1.00	0.98	0.67
DB11-2	0.98	1.00	0.98	0.67
Mean	0.92	0.99	0.90	0.56

Dataset	False positive rate	False discovery rate	False negative rate	MCC[1]
DB3-2	0.09	0.08	0.00	0.91
DF-2	0.15	0.13	0.00	0.86
DB5-2	0.10	0.10	0.07	0.83
DB6-2	0.14	0.95	0.05	0.21
DB7-2	0.17	0.96	0.00	0.17
DB8-2	0.17	0.96	0.00	0.17
DB9-2	0.02	0.08	0.00	0.95
DB10-2	0.02	0.33	0.00	0.81
DB11-2	0.02	0.33	0.00	0.81
Mean	0.10	0.44	0.01	0.94

Dataset	Processing Time (seconds/picture)	Log-Transformed
DB3-2	0.87	-0.14
DF-2	8.00	2.08
DB5-2	1.43	0.35
DB6-2	0.72	-0.32

[1] Matthews correlation coefficient

DB7-2	0.77	-0.26
DB8-2	1.52	0.42
DB9-2	2.17	0.77
DB10-2	1.45	0.37
DB11-2	21.40	3.06
Mean		0.70
Mean	2.02	
CI @ 95.0% Confidence	(0.84, 4.87)	

Dataset	Top-5 Accuracy
DB3-1	0.99
DF-1	0.97
DB5-1	0.95
DB6-1	0.98
DB7-1	0.97
DB8-1	0.98
DB9-1	0.99
DB10-1	0.99
DB11-1	0.99
Mean	0.98

Appendix 4 – McNemar's Computation

Let **B** = 385; **C** = 628. Computing for χ^2:

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C}$$

Plug-in the values:

$$\chi^2 = \frac{(|\,385 - 628| - 1)^2}{385 + 628} = 57.8124383$$

The chi-squared critical value, χ^2_{crit}, can be retrieved from MS Excel using α = 0.05 with 1 degree of freedom:

$$\chi^2_{crit} = = CHISQ.INV.RT(0.05,1) = 3.84$$

Appendix 5 – Rule of Thumb for Analysing Size of a Correlation Coefficient

Taken from [77]

Size of Correlation	Interpretation
.90 to 1.00 (-.90 to –1.00)	Very high positive (negative) correlation
.70 to .90 (-.70 to -.90)	High positive (negative) correlation
.50 to .70 (-.50 to -.70)	Moderate positive (negative) correlation
.30 to .50 (-.30 to -.50)	Low positive (negative) correlation
.00 to .30 (.00 to -.30)	Little if any correlation

Appendix 6 – Correlation Solution

Manual		**Tool**	
Answers	**Duration**	**Answers**	**Duration**
6	21	4	57
4	8	5	9
4	29	6	16
6	39	6	10
5	20	5	35

The values are normalised by proportion, $f(x) = x/\max\{(x_1, \ldots, x_5)\}$, to scale the values uniformly across different measures (answers and duration). The resulting values are now relative to the maximum number.

Manual		**Tool**	
Answers	**Duration**	**Answers**	**Duration**
1	0.538461538	0.666667	1
0.714286	0.205128205	0.833333	0.157894737
0.714286	0.743589744	1	0.280701754
1	1	1	0.175438596
0.857143	0.512820513	0.833333	0.614035088

Output of regression values using Microsoft Excel Data Analysis Pack:

Manual method:

Correlation Matrix	Correct answers	Duration(mins)
Correct answers	1	
Duration(mins)	0.499905509	1

Regression Statistics	
Multiple R	0.499906
R Square / Coefficient of Determination	0.249906
Adjusted R Square	-0.00013
Standard Error	0.142866
Observations	5

	Coefficients	Standard Error	t Stat	P-value
Intercept	0.711856	0.158748255	4.484183	0.020685192
Duration(mins)	0.242144	0.242205501	0.999748	0.391106414

Prototype:

Correlation Matrix	Correct answers	Duration(mins)
Correct answers	1	
Duration(mins)	-0.821058995	1

Regression Statistics	
Multiple R	0.821058995
R Square / Coefficient of Determination	0.674137873
Adjusted R Square	0.565517163
Standard Error	0.091914511
Observations	5

	Coefficients	Standard Error	t Stat	P-value
Intercept	1.008347179	0.070171106	14.36983449	0.000730458
Duration(mins)	-0.317944458	0.127624253	-2.491254213	0.088386106

Appendix 7 – Area Under the Curve Computation

Code inspired from MachineLearningMastery [80].

AUC

```
In [152]: result_file = "D:\\ANALYSIS\\forevaluation\\android_inception.csv"
          df_1 = pd.read_csv( result_file,
                           usecols=[1,2,3],
                           names=['Image', 'Actual', 'Predicted'],
                          header=0)
```

```
In [153]: result_file = "D:\\ANALYSIS\\forevaluation\\joe_inception.csv"
          df_2 = pd.read_csv( result_file,
                           usecols=[1,2,3],
                           names=['Image', 'Actual', 'Predicted'],
                          header=0)
```

```
In [160]: df_1.dropna().values
          df_2.dropna().values
          dfs = [df_1, df_2]
          result = pd.concat([df.squeeze() for df in dfs], ignore_index=True)
          result.head()
```

Out[160]:

	Image	Actual	Predicted
0	(28)(3)1394630374701.jpg	0	0
1	.thumbdata3--1967290299_embedded_2.jpg	0	0
2	10487715_1.jpg	0	0
3	12bdba79142e75f6_0_embedded_1.jpg	0	0
4	13160f67b7be7587_0_embedded_1.jpg	0	0

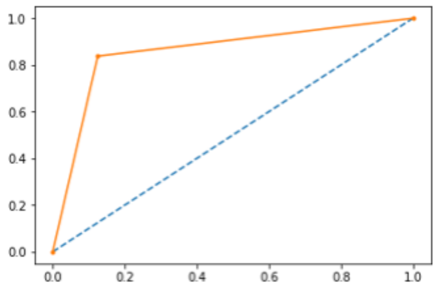
```
In [4]: # roc curve and auc
        from sklearn.datasets import make_classification
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import roc_curve
        from sklearn.metrics import roc_auc_score
        from matplotlib import pyplot
        import pandas as pd
```

```
In [163]: testy = result['Actual'].astype(int)
          probs = result['Predicted'].astype(int)
```

```
In [164]: # calculate AUC
          auc = roc_auc_score(testy, probs)
          print('AUC: %.3f' % auc)

          AUC: 0.856
```

```
In [165]: # calculate roc curve
          fpr, tpr, thresholds = roc_curve(testy, probs)
          # plot no skill
          pyplot.plot([0, 1], [0, 1], linestyle='--')
          # plot the roc curve for the model
          pyplot.plot(fpr, tpr, marker='.')
          # show the plot
          pyplot.show()
```

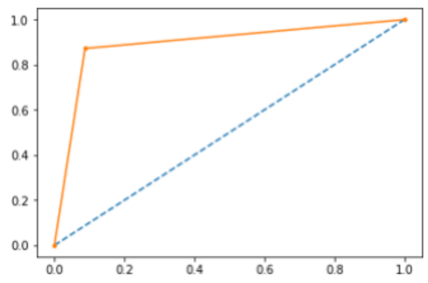


For Xception:

```
In [179]: # calculate AUC
          auc = roc_auc_score(testy, probs)
          print('AUC: %.3f' % auc)

          AUC: 0.892
```

```
In [180]: # calculate roc curve
          fpr, tpr, thresholds = roc_curve(testy, probs)
          # plot no skill
          pyplot.plot([0, 1], [0, 1], linestyle='--')
          # plot the roc curve for the model
          pyplot.plot(fpr, tpr, marker='.')
          # show the plot
          pyplot.show()
```

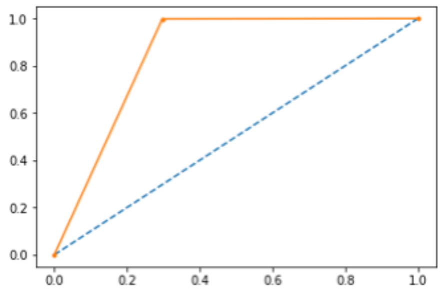


For Magnet Axiom:

```
In [9]: # calculate AUC
        auc = roc_auc_score(testy, probs)
        print('AUC: %.3f' % auc)

        AUC: 0.851
```

```
In [10]: # calculate roc curve
         fpr, tpr, thresholds = roc_curve(testy, probs)
         # plot no skill
         pyplot.plot([0, 1], [0, 1], linestyle='--')
         # plot the roc curve for the model
         pyplot.plot(fpr, tpr, marker='.')
         # show the plot
         pyplot.show()
```



Appendix 8 – Unstructured Interview – Raw Data

Snippets of the information taken from the internship weekly report – includes the discussions regarding the formulation of the research problem.

[Internship, Week 2, July 30 - Aug 3, 2018]
30.07.2018 Discussion with [redacted]:
Problem description: During a forensic analysis, huge amount of images are often extracted from a disk. Searching for a particular incriminating photo (a weapon for example) means having to manually search through tons of images manually.
Proposed solution: Develop a tool/system/CLI (anything) that can perform large scale object detection using machine learning with integrated visualization for digital forensics
Proposed steps:

- Code will scan through a directory of images (10,000 images or more)
- It will return objects detected (search on weapons, drugs, porn, bomb? - depends on the scope for thesis, depends on usability requirements) *Note from [redacted]: Persons, Weapons, Money, Vehicles, Pictures of Documents (papers or printscreen of a receipt, piece of paper)*
- Visualize the results
- User can extract the report (csv file with path/filename)

During the development of the tool, I need help on the following:

- Training, cross-validation and test images dataset
- Design and usability evaluation by actual forensic analysts and non-technically trained personnel (police officers?)

[Internship, Week 8, Sept 10-14, 2018]
13.09.2018 Discussion with [redacted]:
Object Detection:
Image classification, classifying weapons is more important than identifying where the weapon is in that particular image. Defining the location of that object within the image is not really important.
What they need: Reduce the manual way of reviewing images, save on analysis time
Ideas:

- Image classification, separate weapons category from the rest of those images non-pertinent to the case
- Filter by category, filter what kind of weapon
- Alert / Identify if weapons are found, tag for report, add notes
- Metadata extraction
  - Extract forensic information like date/time, serial number
  - Visualise date and time in a timeline
  - Extract GPS coordinates (visualise in a map)
    - Scenario: if the picture of a gun was taken in a warehouse, the GPS will be crucial in getting the exact location
  - Extract EXIF data like camera information
    - Scenario: if the picture of a gun was taken from a smartphone or camera, it will identify the phone for further analysis or perform search and seizure for that specific model (if it was not acquired), or find out if the camera has been attached to a computer
- Source image extraction
  - Determine whether the image was downloaded from the internet, shared in messenger groups (WhatsApp, Viber), sent/received from email, taken and saved by the user himself
  - Visualise graphically
- Weapon recognition
  - If there is a database of existing weapon repository, compare the detected image and identify what type of weapon it is (*Joanna: Machine learning can do this but training will be a pain*)
  - If there is a weapon picture, find out same instances of that weapon anywhere. Example, someone took a picture of a gun from a phone and showed it around. Task is to find that same gun (same picture or same instances of that gun (despite angle or illumination) from different forensic sources (desktop, email communication, etc)
  - Hand-gun detection in videos (*Joanna: Maybe I can integrate prior research about handgun detection in videos*)

- Data mining
    - aside from exported images, find out if the suspect has been viewing, googling for weapons over the internet/deep net/darknet
    - Linkages of images between different devices (computers, smartphones, usbs)

Evaluation Ideas:

- To measure time saved / create a benchmark measure
    - create a test or a forensic case with questions
    - compare speed of using it (1) manually, using a (2) proprietary tool and using my (3) tool
    - Note down human errors, how many misclassified images did the human make, how much time will it take to answer and find out EXIF specific questions

20.09.2018 Discussion with [redacted]:

Object Detection:

Image classification, classify pictures with persons and guns or just guns

Tool:

- There are existing python libraries that can search through browser apps (Chrome, FF, IE) or messenger apps (Skype, Whatsapp, Viber)
- You can also create a mapping with proven directories of where files are usually stored, example location of Temporary Internet files or Outlook files (to identify the source data of the image)
- Timeline can also be exported as a tool, there are python libraries that can export information in a text file or rtf file
- There are also existing python scripts that can search through keywords
- Do not making loading of the tool complicated, just point to a path maybe with the list of source information (FTK can export the file and give information on the source)

Appendix 9 –Raw Feedback Responses during Evaluation

Feedback from *p1*via email

- First setup was difficult, but it works if I imported OVA file to VirtualBox and I replaced old cbis folder with new cbis version. Graphical web interface works only in folder named “cbis” (other folder name printed errors on console)
- I think it’s problem that web interface can show only pictures in “cbis” folder, but I need analyze files on external device (mounted in “media” folder). I can analyze on terminal but can’t see in web interface.
- Third problem is keyword search. For example „rifle“ found „trifle“ – web interface show wrong results and think that trifle is also gun. Better solution is enumerate keywords – integer search is faster than keyword search and no matches in substrings – more accurate. You also will add to web interface keyword tree with checkboxes – user can choose interesting items.
- Probability filter can decrease false positives results – graphical interface show always 5 first (also near 0% probability), but I think that better limit is probability percent.
- First run offline map didn’t work (@console „urlopen.error“) because lab machine don’t have internet connection. I connected temporary to internet and I tried again – console output “downloading” texts . After download map worked (also offline). I’m not sure that map works correctly – I did’n see point on Italy, but one picture was taken in Italy.
- In future you will update your project with video files.
- It’s bad that I must use virtual machine in Windows to use CBIS because I can’t use full machine performance.
- I think that this functionality will more helpful in photo management system (for example Picasa). It will be nice in Accessdata FTK (“Overview” tab –files are categorized different groups) but I never have been need search general objects in image files. Maybe police needs. Sometimes I need search current image or face.

Feedback from *p3* via face-to-face discussion

- Search for filename
- Incomplete EXIF information displayed
- Map issues, Windows photo can find by location (sort of)
- Add timeline
- Enlarge the images
- Map isn't very helpful because it doesn't show properly
- Adjustment of threshold values like a slider

Feedback from *p5* via email

- Good work Joanna. The tool has a potential. It's fairly easy to use (after the prerequisites are met). I did spend much more than 30 minutes though ;). The tool took around 30+ minutes just to go through all three folders with images - I shouldn't have used the verbose switch I assume.
- **Does the visualization (images below) showing overall labels detected in the images useful in a forensic investigation?** It depends. Visualization is always a good thing to have at hand, because it helps to speed up the detection, classification, and overall investigation process. If we are searching only for guns - this particular bubble graph could be replaced with 2 bars: Bar1 - guns, Bar2 - not guns - which is a clickable option on the graph. Good work!

Appendix 10 – Raw Data for Metrics Computation

Computations for InceptionV3 Model: Performance Evaluation of Metrics

	Accuracy	**Sensitivity**	**Specificity**	**Precision**
WEAPON-DB3	0.955592	1	0.911184	0.918429
WEAPON-DF	0.925	1	0.85	0.869565
WEAPON-DB5	0.9125	0.926829	0.897436	0.904762
WEAPON-DB6	0.864565	0.95082	0.863857	0.054256
WEAPON-DB7	0.831562	1	0.830481	0.036474
WEAPON-DB8	0.831562	1	0.830481	0.036474
WEAPON-DB9	0.98374	1	0.979798	0.923077
WEAPON-DB10	0.977486	1	0.976424	0.666667
WEAPON-DB11	0.970817	1	0.97012	0.444444
	False positive rate	**False discovery rate**	**False negative rate**	**Matthews correlation coef**
WEAPON-DB3	0.088816	0.081571	0	0.914799
WEAPON-DF	0.15	0.130435	0	0.859727
WEAPON-DB5	0.102564	0.095238	0.073171	0.825039
WEAPON-DB6	0.136143	0.945744	0.04918	0.209334
WEAPON-DB7	0.169519	0.963526	0	0.174043
WEAPON-DB8	0.169519	0.963526	0	0.174043
WEAPON-DB9	0.020202	0.076923	0	0.951015
WEAPON-DB10	0.023576	0.333333	0	0.806814
WEAPON-DB11	0.02988	0.555556	0	0.656631

Appendix 11 – Speed: Hypothesis Testing

Parametric: t-Test

Participant	Manual Duration	Manual Duration(log)	Tool Duration	Tool Duration (log)
1	21	3.044522438	57	4.043051268
2	8	2.079441542	9	2.197224577
3	29	3.36729583	16	2.772588722
4	39	3.663561646	10	2.302585093
5	20	2.995732274	35	3.555348061
Geometric mean	20.69952485		19.57316596	

Duration is expressed in minutes

	Manual (log-trans)	Prototype (log-trans)
Mean	3.030110746	2.97416
Variance	0.355028834	0.643895
Observations	5	5
Hypothesized Mean Difference	0	
df	4	
t Stat	0.133474971	
P(T<=t) one-tail	0.450131797	$P(T>t_{0.05} \mid H_0)$, do not reject H_0
t Critical one-tail	2.131846786	

Non-parametric: Wilcoxon Signed Rank Test

Raw data of participants completing the tasks in minutes using manual (M) method and tool (T). The positive difference in time, M_d, shows that manual method takes more time (tool is better) and a negative difference shows that the manual method is faster (manual method is better).

Participant	Manual(M)	Tool (T)	$M_d = M - T$
1	21	57	-36
2	8	9	-1
3	29	16	13
4	39	10	29
5	20	35	-15

Setting up the hypothesis test:

As we want to test if the tool performs better than the manual method, the null and alternative hypothesis are stated as follows:

$$H_0: M_d = 0$$

$$H_1: M_d > 0$$

We will employ a one-tailed Wilcoxon signed test with a 5% level of significance (α = 0.05). Ignore the zeros (no differences between manual and tool) and rank accordingly.

M_d = M - T	**Participant**	**Sign**	**Rank**	**Signed Rank**
-36	1	-1	5	-5
-1	2	-1	1	-1
13	3	1	2	2
29	4	1	4	4
-15	5	-1	3	-3

Sum the rank of the positive differences:

$$W_+ = 2 + 4 = 6$$

Sum the rank of the negative differences:

$$W_- = 5 + 1 + 3 = 9$$

Interpreting the test statistics:

If the number of observations per pair is greater than 20, then the normal approximation using the z-table can be used.

$$\frac{n(n+1)}{2} = \frac{5(5+1)}{2} = 15 < 20$$

If the null hypothesis is true, there is no significant difference between the two methods, W_+ and W_- are relatively the same. There are two test statistics, in this case, W_+ and W_-, we take the minimum value W_+, suggesting that participants felt that the tool is better than the manual method.

$$W = \min\{W_+, W_-\} = \min(6,9) = 6$$

The critical value from the table is taken from the Wilcoxon table[1]:

Table 33 Wilcoxon Table for Critical Values

	Alpha values			
n	**0.2**	**0.1**	**0.05**	**0.02**
4	0			
5	2	**0**		
6	3	2	0	
7	5	3	2	0
8	8	5	3	1

The table shows the critical value at the level of significance (α) specified. Given that W_+ is a small value, what is the probability that this occurred by chance? For this case, we reject the null hypothesis at $W_+ \leq W_{critical}$. From the table, $W_{critical}$ is 0. Since, $W = 6$ > 0, then we do not reject the null hypothesis.

[1] https://www.real-statistics.com/statistics-tables/wilcoxon-signed-ranks-table/