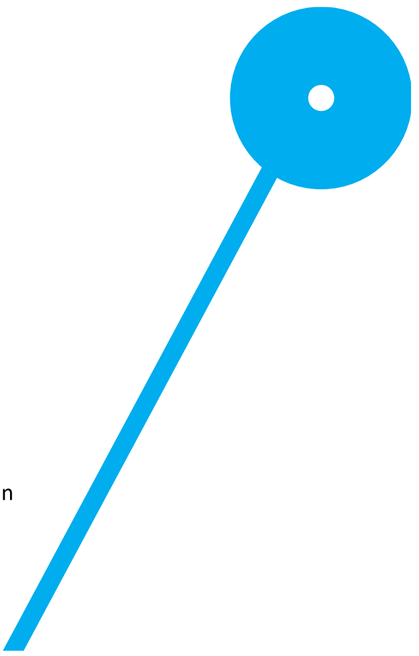
SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

P.PORTO

M

MASTER
COMPUTER ENGINEERING



Human-in-the-loop image classification
Bernardo Filipe Gonçalves Almeida

P.PORTO

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M

MASTER
COMPUTER ENGINEERING

Human-in-the-loop image classification
Bernardo Filipe Gonçalves Almeida
10/2022

P.PORTO

Acknowledgements

The participation and goodwill of those directly and indirectly involved in the project were essential to making this thesis possible. I wish to extend my heartfelt gratitude to everyone.

To ESTG, where all my academic path was made and transmitted the knowledge for being capable to complete a project like this.

To my academic supervisor, Professor Davide Carneiro, for all the knowledge and inputs transmitted during the dissertation, along with a two-way feedback with always an improvement state of mind. Thank you for all the patience and the improvements, we made it!

To my family, a pillar, for believing in me and always giving unconditional support along my academic. Your belief gave me the confidence I needed to go forward and complete all my objectives.

To my girlfriend, another pillar, your support was essential to keep me going pursuing my objectives. You were essential to always give my everything, keep a positive mind and finishing this project.

Resumo

Nos últimos anos, tem havido um crescimento na utilização de Machine Learning e uma necessidade crescente de aplicar modelos de Machine Learning a várias necessidades empresariais, desde a análise dos padrões de compra dos clientes até à tomada de uma decisão empresarial para fazer crescer esse mesmo negócio.

Num ambiente empresarial acelarado que nos encontramos atualmente, desenvolver e disponibilizar um bom modelo pode não ser um processo muito célere. O principal motivo são os dados necessários para obter o bom modelo, visto que para obtê-lo pode ser necessário uma grande quantidade de dados e isto pode afetar o tempo de treino do modelo, ou pode ser necessário um pré-processamento dos dados, levando ao aumento do tempo para obter o bom modelo. Com isto, este trabalho apresenta uma possível solução para este problema, onde, através do Active Learning, o humano aplica etiquetas a uma pequena quantidade dados, de seguida são criados vários modelos com parâmetros diferentes para serem treinados até que um intervalo de valores seja atingido. Por fim, algumas métricas serão extraídas e analisadas para concluir qual o melhor modelo. Por fim é apresentada a previsão do modelo em conjunto com uma explicação com o que o modelo considerou importante.

Palavras Chave: Active Learning; Explainable AI; Labelling; Early Stopping

Abstract

In recent years, there has been a growth in the use of Machine Learning and an increasing need to apply Machine Learning models to various business needs, from analysing customer buying patterns to making a business decision to grow that same business.

In the fast-paced business environment we currently find ourselves in, developing and delivering a good model may not be a very fast process. The main reason is the data required to obtain the good model, since to obtain it may require a large amount of data and this may affect the training time of the model, or a pre-processing of the data may be required, leading to increased time to obtain the good model.

With this, this work presents a possible solution to this problem, where, through Active Learning, the human applies labels to a small amount of data, then several models are created with different parameters to be trained until a range of values is reached. Finally, some metrics will be extracted and analysed to conclude which model is the best. Finally the prediction of the model is presented together with an explanation of what the model considered important.

Keywords: Active Learning; Explainable AI; Labelling; Early Stopping

Contents

1 Introduction **12**

1.1 Context . . . 12

1.2 Research Methodology . . . 13

1.3 Objectives . . . 14

2 Background **14**

2.1 Black-box models and Interpretable models . . . 14

2.2 Interpretability . . . 15

2.3 Explanations . . . 15

2.3.1 What makes an explanation, good? . . . 16

2.4 Explanations and models . . . 17

2.4.1 Model-agnostic and model-dependent explanations . . . 17

2.5 Labelling . . . 18

2.6 Active Learning . . . 18

2.7 Machine Learning Operations . . . 18

2.8 Classification Algorithms . . . 20

2.9 Artificial Neural Networks Anatomy . . . 20

2.10 Types of Artificial Neural Networks . . . 21

2.10.1 Perceptron . . . 21

2.10.2 Feed Forward Neural Networks . . . 22

2.10.3 Recurrent Neural Networks . . . 23

2.10.4 Convolutional Neural Networks . . . 24

3 State of Art **28**

3.1 Machine Learning Frameworks . . . 28

3.2 Datasets . . . 29

3.3 Methods and metrics for mask detection detection . . . 29

3.4 Transparency . . . 31

3.4.1 Recourse as Explainability . . . 32

3.4.2 Random Forests for Explainability . . . 36

3.4.3 Algorithm agnostic Explainability . . . 41

3.5 Learning process 50
3.5.1 Reinforcement Learning 51
3.6 Image Labelling 51

4 Material and Methods **53**
4.1 Technical approach to a Convolutional Neural Networks 53
4.2 Achieving a desired model 54
4.2.1 Metrics 55
4.2.2 Description of the runs 58

5 Results **64**
5.1 Dataset of 30 images per class 64
5.1.1 First Model 64
5.1.2 Second Model 65
5.1.3 Third Model 67
5.2 Dataset of 60 images per class 69
5.2.1 First Model 69
5.2.2 Second Model 70
5.2.3 Third Model 72
5.3 Dataset of 90 images per class 74
5.3.1 First Model 74
5.3.2 Second Model 75
5.3.3 Third Model 76
5.4 Models performance recap 78
5.5 Explaining an image 78

6 Conclusion **81**
6.1 Limitations 82
6.2 Future Work 82

List of Tables

1 Experiment metrics overview 31

2 Example of feature values extraction 39

3 F1 Score Values 47

4 Experiment Results 49

5 Classification terms 55

6 First Dataset Metrics results overview 69

7 Second Dataset Metrics results overview 74

8 Third Model's Metrics Summary 78

List of Figures

1 Machine Learning Operations Cycle . . . 19
2 Active Learning Cycle . . . 20
3 Artificial Neural Network Example . . . 21
4 Perceptron example . . . 22
5 Feed Forward Neural Network example . . . 23
6 Recurrent Neural Network example . . . 24
7 RGB color channels example . . . 25
8 Dot product operation example . . . 26
9 Image of a person with a mask . . . 27
10 Recourse list example . . . 33
11 Recourse models performance . . . 34
12 Second Experiment Results . . . 35
13 Cost of recourse disparities . . . 36
14 Random Forest Prediction Example . . . 37
15 Number of links threshold . . . 40
16 Rate of positive words . . . 40
17 Rate of negative words . . . 41
18 Intuition for LIME . . . 44
19 Image Classification Explanation Example . . . 44
20 Pick Step Example . . . 45
21 Recall on important features on the books dataset . . . 46
22 Recall on important features on the DVD's dataset . . . 47
23 % of correct choices on book dataset . . . 48
24 % of correct choices on dvd dataset . . . 48
25 Husky classified as "Wolf" . . . 49
26 Explanation . . . 49
27 SHAP Explanation Example . . . 50
28 LabelImg labelling process example . . . 52
29 Rectlabel labelling process example . . . 52
30 Area Under the Curve Example . . . 57

31 First Model Architecture 61
32 Second Model Architecture 62
33 Third Model Architecture 63
34 First Model Loss Values 64
35 First Model AUC Values 64
36 First Model Precision Values 65
37 First Model Recall Values 65
38 First Model F1 Score Values 65
39 Second Model Loss Values 66
40 Second Model AUC Values 66
41 Second Model Precision Values 66
42 Second Model Recall Values 66
43 Second Model F1 Score Values 67
44 Third Model Loss Values 67
45 Third Model AUC Values 67
46 Third Model Precision Values 68
47 Third Model Recall Values 68
48 Third Model F1 Score Values 68
49 First Model Precision Values 69
50 First Model Recall Values 69
51 First Model Precision Values 70
52 First Model Recall Values 70
53 First Model F1 Score Values 70
54 Second Model Loss Values 71
55 Second Model AUC Values 71
56 Second Model Precision Values 71
57 Second Model Recall Values 71
58 Second Model F1 Score Values 72
59 Third Model Loss Values 72
60 Third Model AUC Values 72
61 Third Model Precision Values 73
62 Third Model Recall Values 73

63 Third Model F1 Score Values 73
64 First Model Loss Values 74
65 First Model AUC Values 74
66 First Model Precision Values 75
67 First Model Recall Values 75
68 First Model F1 Score Values 75
69 Second Model Loss Values 75
70 Second Model AUC Values 75
71 Second Model Precision Values 76
72 Second Model Recall Values 76
73 First Model F1 Score Values 76
74 Third Model Loss Values 77
75 Third Model AUC Values 77
76 Third Model Precision Values 77
77 Third Model Recall Values 77
78 First Model F1 Score Values 77
79 Model explanation for no mask images 80
80 Model Explanation for no mask 80

Listings

1 Rescaling RGB image . . . 53
2 Convolution Block Example . . . 53
3 Tensorflow Convolution Block Example . . . 54
4 Output Bock Example . . . 54
5 Compile Model Method . . . 56
6 Early Stopping . . . 57
7 Fit Function . . . 58
8 Processing the XML File . . . 58
9 Generate training dataset . . . 59
10 Generate training dataset . . . 60
11 Generate training dataset . . . 60
12 First Model Architecture . . . 60
13 Second Model Last Convolution Block . . . 61
14 Third Model Last Convolutions Block . . . 62
15 Conversion to InceptionV3 . . . 79
16 Usage of LIME . . . 79
17 Generating explanation image . . . 79

Acronym List

API Application Programming Interface

ANN Artificial Neural Networks

AUC Area Under the Curve

CNN Convolution Neural Network

DARPA Defense Advanced Research Projects Agency

DL Deep Learning

LIME Local Interpretable Model-agnostic Explanations

ML Machine Learning

RGB Red-Green-Blue

SHAPE Shapley Additive explanations

XML Extensible Markup Language

1 Introduction

This chapter presents the investigation and the project developed during the master's degree in Computer Engineering, at the School of Technology and Management of the Polytechnic of Porto (ESTG). First, there is contextualisation and characterisation of the problem that was studied.

1.1 Context

In the last few years, there has been a growth in Machine Learning (ML) usage and an increasing necessity of applying ML models to several enterprise needs, from analysing customers purchase patterns to taking a business decision to grow that same business.

To develop and deploy these models to the real world, there are some frameworks that encapsulate all the boilerplate work and make the development stage easier and more efficient. Some of the most popular ML frameworks, such as Tensorflow and scikit-learn, were quickly adopted by developers due to the easy usage, a vast catalogue of types of ML algorithms and a huge community of developers and researchers that, continually, provide feedback and possible improvements to these frameworks. Also, for research purposes, PyTorch offers a robust ecosystem from prototyping, with distributed training and compatibility with most of the cloud platforms, up to the release of the models to the real world. As the term Machine Learning gained visibility and popularity, the fields where it is used also expand widely. For instance, in July of 2020, Microsoft conducted a research about mental healthcare where the aim was to examine different patterns of patient behaviour when engaging and interacting with an internet-based cognitive behavioural therapy from the point where the patients manifests symptoms of depression or anxiety [1], thus recurring to ML models to help in categorising the patients.

In another field, GitHub[1] is helping developers going open source with their project, where they are leveraging Machine Learning in order to help developers identifying issues in early stages of a future open-source project also recurring to deep learning.[2].

Lastly but not the least, Machine Learning is being used in e-commerce, like ebay[3] where they leveraged it in order to better categorise the products on the website by using simple text [2].

The pandemic of the SARS-CoV-2 that hit the world by storm in the beginning of 2020 forced people to stay at home and created a new paradigm of how people work, shop and travel. It was also the

[1] https://github.com/

[2] https://github.blog/2020-01-22-how-we-built-good-first-issues/

[3] https://www.ebay.com/

source of many interesting use cases and applications of ML. While nowadays people are trying to get back to normal, there are often rules that people must still follow in order to, for instance, travel or even enter an hospital or home care. Namely, it is still common across many countries that the wearing of a face mask is still mandatory.

However, in very crowded places such as airports or in places where there is no human surveillance, it can be humanly impossible to scan all the people to check if they are, or not, wearing a mask, which may pose a health risk. Moreover, this problem does not end with the end of the current pandemic, as others similar or worse may follow. To try to tackle this problem, in March of 2021, Microsoft made available on Azure Cognitive Services cloud platform a feature that can detect mask usage via images thourgh API endpoint calls[4].

Furthermore, Viso AI, a platform focused on computer vision developed a feature where it is possible to identify in real time the usage, or not, of masks with presenting to the developer useful metrics of the models performance[5]. In this context, this works presents a model that predicts whether a person is wearing a face mask and an explanation for what the model "saw" and interpreted as being a mask. Furthermore, and more technically, developing a model with good results/metrics takes time, more time if the dataset has a large number of images. Thus, by leveraging active learning with labelling and stopping models over-training with early stopping, we try to tackle the amount of time of the model's development, starting at its inception until it is deployed to the real world.

In summary, when developing several model with different parameters it must be taken into account a trade-off. To have a model more precise it might be needed a large number of images and a model that can be capable of identifying more key features in the images, however this model will take more time to train and we can take for granted the last epoch's metrics value, because it does not take into account the values along the validation phase.

1.2 Research Methodology

This dissertation will follow a approach named Design Science Research and can be separate by six main topics [3][4][5].

Starting with the identification of the problem, defining the research problem and justifying the value of the solution, followed by the definition of the objectives for the final solution with the desired results. Following up, in Section 2 is presented study of what was already developed and exists to tackle the

[4] `shorturl.at/sIR47`

[5] `https://viso.ai/application/mask-detection/`

several stages of the process, identified along Section 1. Next up, in Section 3, comes the main focus and that is the design and development where it is going to be explained what was developed and how. Moreover, to validate all the software that was developed comes the demonstration where this same software will be put to the test and validate if the problem was solved, described in Section 3. The final phase, in Section 4, is the model's evaluation where the results obtained from the previous phase are evaluated against a validation dataset within the training dataset and the defined objectives.

1.3 Objectives

With ML being introduced in several fields, and some fields with big decision making, for instance, banking, and medical, the predictions made by the models developed must be taken into account and not accepted blindly. Taking a step back, for a model to be effective in the real world it needs data, although this data may not be abundant, a model can still be trained without an enormous volume of data. Also just training the model is not enough, the model must be evaluated with some useful metrics to back it up has a good model.
As said before, a model's prediction can be taken blindly as the consequences are unpredictable, thus the human must know what the model saw or considered important to reach a certain prediction.
The following objectives try to answer the mentioned situations and the purpose that brings this work to life, they are:

- Implementation of an approach for optimisation of the learning process in active learning systems
- Achieve models with satisfactory metrics with the less volume of data possible
- Explain a model's prediction, mainly in the black-box models, because a model-based explanation of its prediction cannot be extracted

2 Background

2.1 Black-box models and Interpretable models

Although ML and the respective algorithms are being used in several fields and within that fields in a vast of different ways, it is not appropriate to just accept the prediction or the class that a model printed out to the developer or to the human with experience of the field that the models are being

used. A doctor cannot accept a prediction of the diagnosis of a patient when he doesn't know the internal process of the model, which is impossible in a black-box model and accepting a prediction from this model can bring unpredictable consequences. Thus, black-box models cannot be elected in the development phase. A black-box model is simply a system that does not reveal the internal process, hence, in the Machine Learning field, the term "black-box" refers to a model that cannot be understood by looking at the parameters of that same model [6].
The opposite of a black-box model is referred often to as a white model, but are also known as interpretable models and this type of model refers to the methods that make the internal behaviour and the predictions understandable to humans.

2.2 Interpretability

The most used definition of interpretability, in the ML field, refers to the degree to which a human can understand the cause of a decision [7]. Furthermore, it is the degree that a human can predict in advance what the model is going to predict. Thus, the goal nowadays is to deploy to the real world high interpretable models, since more and more humans with no knowledge of ML are adopting these models for daily tasks.
Technically, when developing a predictive model we end up at a crossroads, is the prediction enough or it is better that the prediction in combined with an explanation. For instance, does a doctor only want to know what medicine to administrate on a patient, or does he want to know why that prediction was made in the first place, thus dropping in performance but aiding the doctor making a better decision by combining his knowledge and the prediction. The process of integrating a machine learning algorithm in daily tasks requires interpretability to increase social acceptance [7]. Although interpretability may not be necessary in some use cases, on the other hand, it may be helpful to add interpretability to an use case that has no several impacts when deployed to the real world, for instance, a movie recommendation system.
Additionally, it is also preferred that the model also explains how it came to a certain prediction, because a prediction solves partially the original problem.

2.3 Explanations

As mentioned before an explanation should complement a prediction to give a perception of what the model understood with a prediction. Technically, an explanation relates the feature values of an instance to the model prediction in a way that a human can understand. Humans always get

curious when are in front of something that they don't know or something new, to learn from that experience, trying to find the answer to the why question [7], for instance:

- Why was my loan rejected?
- Why did not the a certain medicine worked on the patient?

Furthermore, an explanation can be used to handle social interactions, by creating a shared meaning about a certain topic. Although, to make solid social interactions the explanations returned by the model must be good.

2.3.1 What makes an explanation, good?

Explanations are contrastive, meaning, that very often, a human does not ask why a certain prediction was made, but why the model predicted X and not Y. Furthermore, a human would like to know if the prediction could had a different result if the inputs were different, for example, *"Would my request for a loan be accepted if I increased my income or even my graduation level?"*. Another example that expresses this point very clearly could be in the medical context, *"Why did medicine Z work on patient A and not on patient B?"*, this kind of contrastive explanation helps the human retrieve more information and make better future decisions. The best kind of explanation is the one that highlights the differences between instances [7].

Explanations are selected. Humans often don't want the full explanation when a certain event occurred, a reason or two is often good enough to get a basic understanding of the situation. Looking back at the medical example, the doctor, firstly, may only look at the data that differentiates the patients and secondly at the medical results with different patients. This type of situation that an event can have multiple explanations for is called Rashonom Effect. This effect is based on a Japanese movie that tells alternative stories about the death of a samurai. Thus, it is optimal that ML models make good predictions based on several features and usually, ensembles use this principle when put on a ML model because they use different features and perform well due to averaging over other predictions and that makes a prediction more robust. A prediction might contain from 1 to 3 reasons even when the problem has a considerable number of features and LIME [8] does apply this principle that will be explained in the State of Art chapter.

Explanations are social, if an objective of explanations is to handle social interaction, then an explanation must take into account the context it is inserted into. For instance, when showing the explanation of why the medicine had different results on different patients, it is logical to show the

information from both the patients and the medicine. Thus, it is important to take into account the environment the model is inserted and the target audience. This can be done during the development phase when one could call for help from an expert in the target environment.
Explanations focus on the abnormal, if for some reason, a feature has an abnormal value and this feature influenced the prediction, this same feature must be included in the explanation. For instance, if a model that predicts a price of a house, has a feature with an abnormal value that could indicate an expansive house with two balconies. Furthermore, with features such as recent renovation, and near public transport, the model does not change the price of the prediction, it means that the feature, of two balconies, has a greater influence on the prediction, thus, it should be included in the explanation [7].
Explanations are truthful, also known as fidelity, means that an explanation should predict the event as near to the real world as possible. An explanation that only shows one or two possible causes for the prediction does not cover all the possibilities of relevant causes. For us, humans, the fidelity of an explanation is not as important as selectivity, a human might only look to one or two causes for the prediction, but contrast and social aspect are the most important characteristics [7].

2.4 Explanations and models

When working with models that provide some kind of explanation, it is important to take into account the behaviour between the explanations and the model.

2.4.1 Model-agnostic and model-dependent explanations

Model-dependent explanations or model-specific explanations are a drawback in terms of applications in other use cases, because, as the name indicates, these types of models are bounded to the use case of the model.
Separating the explanations from the model can bring some advantages, the major advantage is the flexibility of model-agnostic explanations, meaning that they can be applied to several use cases from different environments. When developing ML models, there is a phase of evaluation, to extract the accuracy and other metrics, and when it comes to evaluating interpretability, it is easier to work with model-agnostic explanations, because they can be used in any type of model.
Ribeiro [9] proposed some desirable aspects for model-agnostic explanations:

- Model flexibility: The interpretation method must work in any ML model, from random forests to neural networks.

- Explanation flexibility: The explanations can take any form. In some cases can be in form of a linear formula and in other cases a graphic with feature importance.
- Representation flexibility: The explanation system should be able to use different representations, for instance, show important words in text classification or highlight important pixels on image classification

2.5 Labelling

Considering that the explanations involve a human in most of the cases, because the final target of the explanation is humans that are going to make a decision, it is useful to also think in labelling. Labelling is the process where a human interacts with the learning process. This human, or specialist has, at least, minimal knowledge of the use case of the model, since the context is in image classification, the labelling process has a focus on selected areas of an image that are inside a pre-determined class. The whole objective of labelling, as in image classification or in text classification, where the specialist selects words that are inside an emotion, for instance, is to generate an even better explanation for the target audience to make better and more conscious decisions and also to involve the human in the model learning process and to not just being the model outputting the predictions with a simple and opaque explanation.

2.6 Active Learning

The process of involving the human/specialist in the learning process is also denominated as Active Learning.

There are times when unlabelled data is plentiful yet manual labelling is prohibitively expensive.In such a case, learning algorithms can actively seek labels from the user/teacher.

Active learning refers to this sort of iterative supervised learning, because the learner selects the instances, the number of examples required to learn a concept is frequently substantially smaller than the number required in traditional supervised learning. There is a risk that the algorithm will be swamped by uninformative samples if this strategy is used.

2.7 Machine Learning Operations

All this process of developing models and putting them to train in several environments is a critical step towards a process of continuous integration/continuous delivery in the field of machine learning,

also known as, machine learning operations. In Figure 1 it is represented the full cycle of a ML model, from the creation to the deployment.

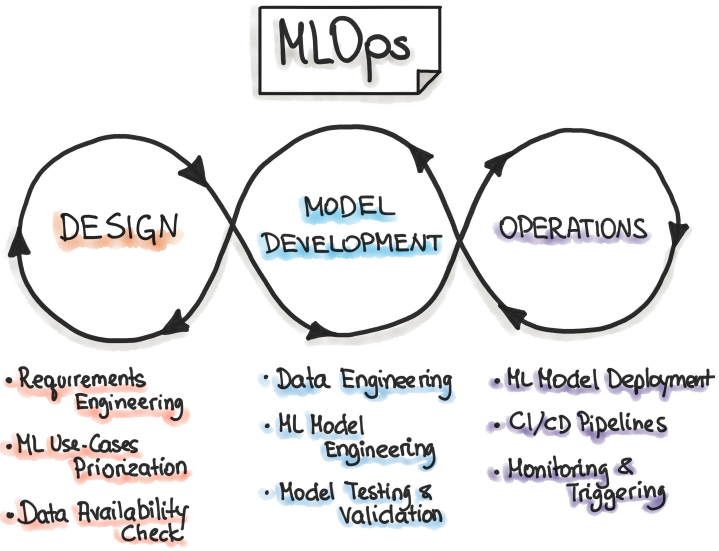


Figure 1: Machine Learning Operations Cycle [6]

Focusing on the middle step it is visible that this part of the process was what was done in this dissertation, but not all for it. In this step of the process, the model goes through a phase of fine-tuning where the hyperparameters are changed to see if the model has better metrics. Going back to Active Learning, referenced in Sub-section 2.6, ML operations integrates with active learning, represented in Figure 2, seamlessly in the way that the user keeps labelling data, thus, more models can be developed and consequently, the metrics keeps getting better and the predictions have more trust to deploy the models to the real world.

[6] `https://ml-ops.org/content/mlops-principles`

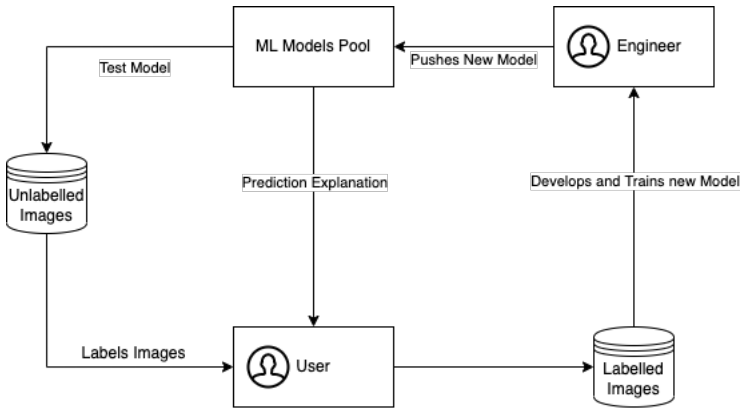


Figure 2: Active Learning Cycle

2.8 Classification Algorithms

Since the problem is placed on image recognition, we need to choose an algorithm to be in charge of the learning/training phase, these algorithms use several computing layers to extract high-level features from raw input. In this context, the first layers are responsible to identify edges on an image while the deeper layers are responsible to identify things that are relevant to humans and that can be human faces. These types of models are known as Artificial Neural Networks (ANN).

2.9 Artificial Neural Networks Anatomy

An ANN is inspired by biological neural networks and can relatively model a human brain. Beginning on the nodes of an ANN that are compared to the brain's neurons and all of these nodes are connected, as a brain's neurons are connected via synapses, these nodes transmit signals to other nodes. This "signal" is a number and the output of each neuron is computed by a non-linear function of the sum of the input values. Neurons are connected via edges and these two entities have a weight associated and this weight is adjusted along the learning phase.

As said above, these neural networks are organised in layers, more specifically in 3 types of layers, Figure 3 shows a standard artificial neural network where the first main layer is the input layer, which is the entry point for the initial data. Secondly comes the hidden layer where all the computation is done and this layer can have n sub-layers. Finally, the output layer returns the result for a given input.

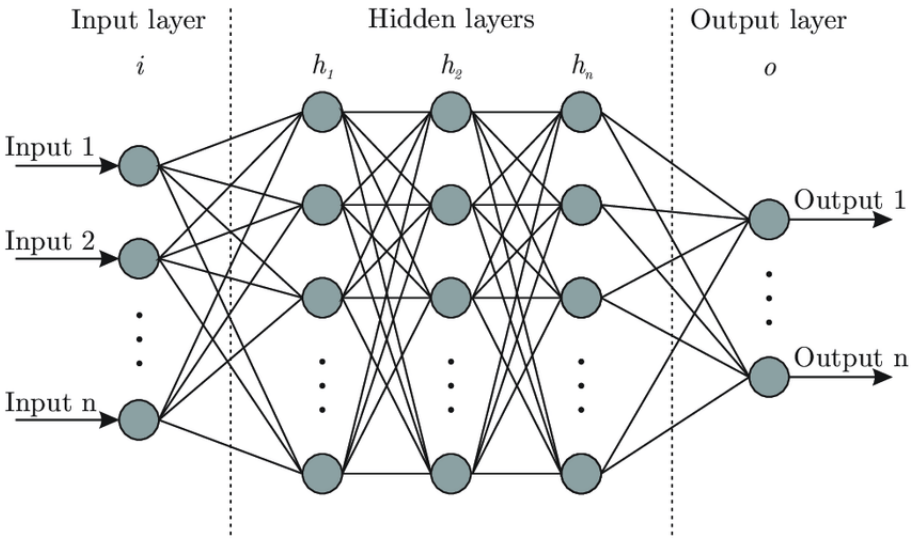


Figure 3: Artificial Neural Network Example [10]

Since an ANN is a generic algorithm, this algorithm also has some variations of it, some are to tackle specific problems.

2.10 Types of Artificial Neural Networks

2.10.1 Perceptron

The first approach to an ANN came in 1958 named Perceptron by Frank Rosenblatt [11], and its the simplest and small sample of an ANN used for binary classifiers. Figure 4 shows this type of ANN. Moreover, the perceptron separates the weighted input space and applies the activation function to obtain the final output that is one of the two categories. Nowadays it is also known as a threshold function that maps an input x to output via $f(x)$. Perceptron learns on linear problems, thus it does not apply to the case.

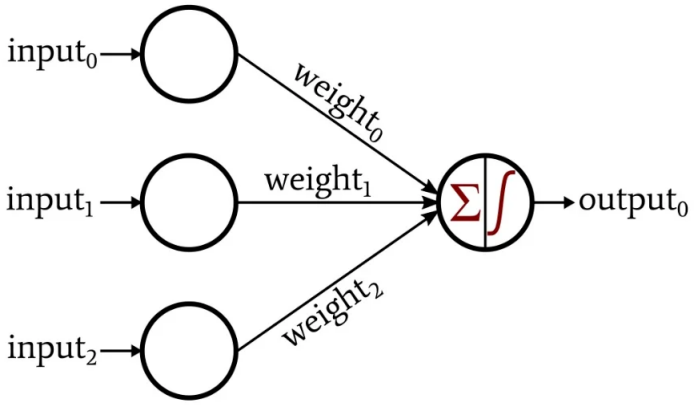


Figure 4: Perceptron example [7]

2.10.2 Feed Forward Neural Networks

Moving forward, another simple form of an ANN is a Feed Forward Neural Network (FFNN) and this type of ANN possesses already the 3 standard types of layers of an ANN. Furthermore, the data only travel in one direction passing through the nodes to the output nodes. The number of layers depends on the complexity of the function and it does possess backward propagation meaning that the weights have static values then they do not change along the process. Due to these static values, a FFNN is faster because the data only travels in one direction but is not the best ANN indicted to DL considering that this type of ANN lack dense/hidden layers. These types of ANN are typically used in computer vision use cases and simple face recognition problems. Figure 5 shows an example of a FFNN with two hidden layers.

[7] https://www.allaboutcircuits.com/technical-articles/how-to-perform-classification-using-a-neural-network-a-simple-perceptron-example/

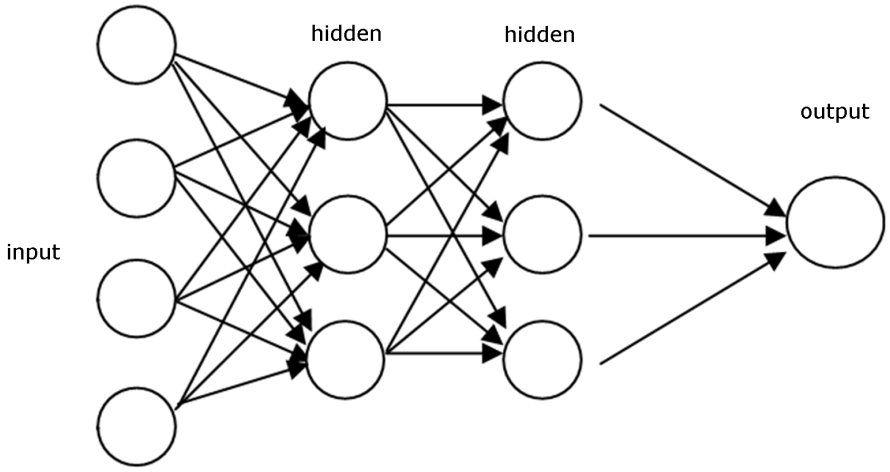


Figure 5: Feed Forward Neural Network example [8]

2.10.3 Recurrent Neural Networks

On the other hand, a Recurrent Neural Network (RNN) has the backpropagation algorithm, meaning that an output of a layer is used to update the weight values of a previous layer. If the prediction is wrong, the learning rate will converge towards making the right prediction during the backpropagation. It is right to say that this kind of ANN can be useful to make more trustworthy predictions however, training a RNN can be difficult due to the time that it takes learning and also it can be hard to maintain this types of ANN due to the recurrent part, this can be seen in Figure 6. RNN can be applied in several fields like robot control, human action recognition and also in sentiment analysis.

[8] https://www.oreilly.com/library/view/neural-networks-with/9781788397872/eb49931a-7122-4b97-a843-bb6e1817cc12.xhtml

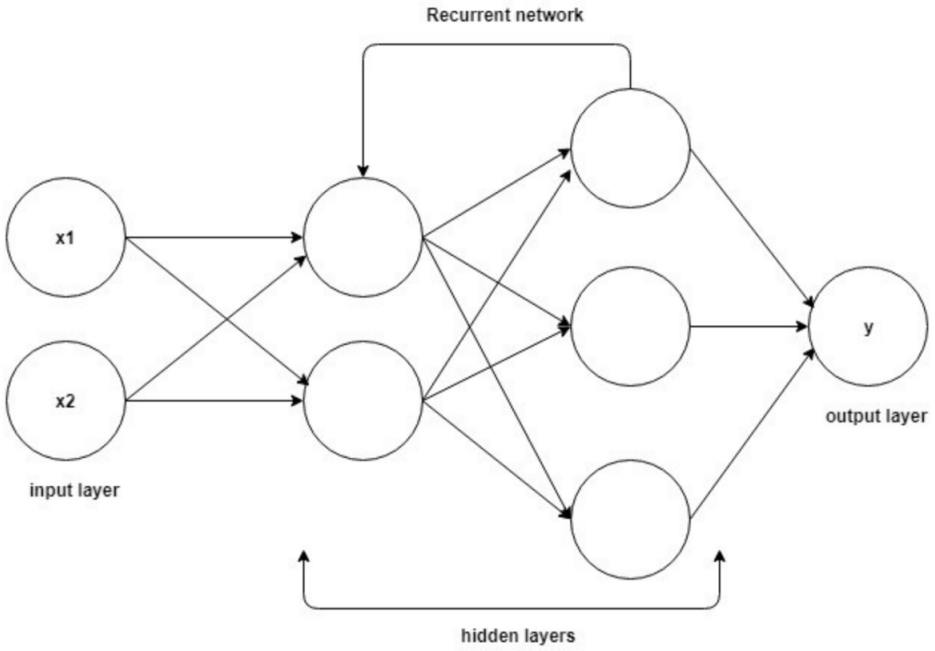


Figure 6: Recurrent Neural Network example [9]

2.10.4 Convolutional Neural Networks

Another common ANN used for face recognition, hence image processing, is a convolutional neural network [12]. A typical CNN consists of three, or more, layers. An input layer, n hidden layers and the output layer. However, in this type of ANN the hidden layer has, in itself, layers, a layer that performs convolutions, a pooling layer and a fully-connected layer and via convolutions, the input image is reduced into another image that is easier to process, without losing key features that are essential to getting a good prediction.

The image enters the networks, not in a normal format, but in its most fundamental level, a three dimensions (height, width and depth) matrix that corresponds to the Red-Green-Blue, RGB, of the image and Figure 7 shows a graphic version of this concept.

[9] https://abhijeet5793.medium.com/convolutional-recurrent-neural-networks-cc926e16fa89

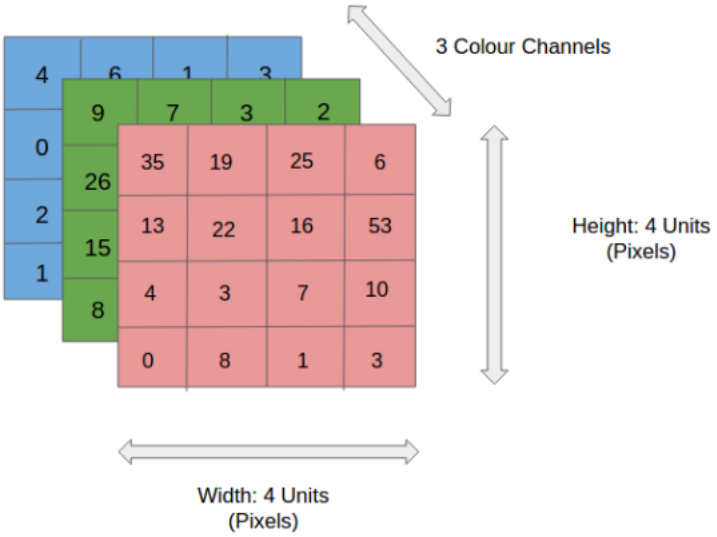


Figure 7: RGB color channels example [10]

Going through the CNN, a feature detector, also known as a filter or kernel is a three dimension matrix, since the image has three channels of colour, that slide across the image checking if a feature is present. This process is commonly known as convolution, thus giving the name to this type of ANN. Each level of the filter matrix will slide across the respective level for the colour channel's matrix. Along the strides the filter does across the image, at the time, it is calculated a dot product between the filter and the respective colour channel matrix, the operation consists of the sum of the multiplication between the value on the colour channel matrix and the filter matrix. Figure 8 shows a demonstrative example of this operation. The final product consists of a matrix with the respective values, this final output is also known as a feature map.

[10] https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306

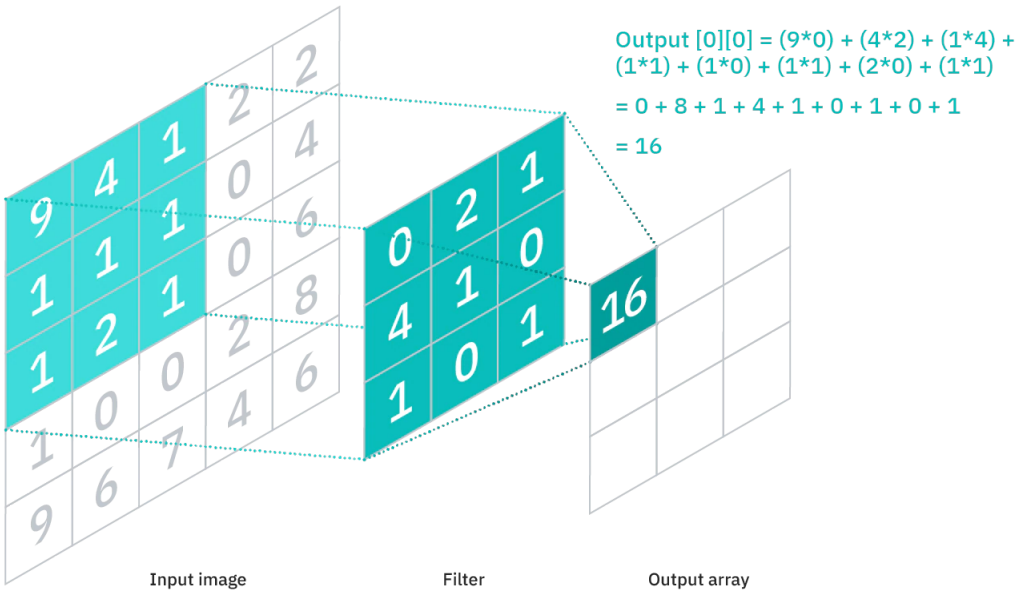


Figure 8: Dot product operation example [11]

During this operation the values that are in the filter do not change, although after this operation these same values can change due to backpropagation, meaning that these values are adjusted during the training phase. However, not all values are adjusted and this values are called hyperparameters, parameters that control the ANN. In CNN some of these hyperparameters are:

- Number filter - This parameter defines the dimension of the output space, for instance, if this number is 3 the final output would be three distinct feature maps, given this a depth of 3. It is a good practice that in early layers, closer to the original image, this parameter has a lower value, the most used value is 16.
- Stride - Represents the number of pixels that the filter should move over the input matrix, the best practice is to use a value less than two, although using a higher value will produce smaller output.
- Padding - This parameter is used when the filter does not fit in the image, if a column of the filter goes out of bounds of the image, the respective values produced in the dot function fall to zero.
  - Same padding - It ensures that the output layer has the same dimension has the input layer

[11] https://www.ibm.com/cloud/learn/convolutional-neural-networks

– Valid padding - When the filter goes out of bounds the respective convolution is dropped

After the convolution the node is going to be activated and produce its respective final output over the feature maps produced in the convolution, the activation function used is the Rectified Linear Units (ReLU), given by equation 1.

$$f(x) = x^{+} = max(0, x) \tag{1}$$

Simplifying the equation, it returns zero if x is negative and returns x if this is a positive value, thus implying no heavy processing of the values. Activation functions help the model in two primary ways. First, makes the model account for interaction effects, meaning that when a variable X affects the prediction differently according to the value of Y and second it helps the model account for non-linear effects, meaning that a variable does not stale a certain value but increases as long x also increases. As mentioned earlier a CNN can be composed of several hidden layers, thus having multiple convolution layers, but how can they work on the same image? Let's assume we want to classify Figure 9, we can break this image as a set of parts, for instance, the part of the mask itself or even the part where there is no mask, and this is what the convolution layers will do, they will traverse the image trying to find key features that meet a certain classification. Each part of the image makes a lower level pattern of the network and the combination of these parts makes the higher level of the same network, thus creating a feature hierarchy within the CNN.

Figure 9: Image of a person with a mask [12]

[12] https://unsplash.com/photos/sY7897-Sps0

After a convolution layer comes a pooling layer, also known as downsampling, it takes a two-by-two matrix that will slide over the input, the output from the convolutional layer, and takes the maximum value from the sliding matrix, thus, creating another matrix of the max values, thus reducing the computational complexity that is required to process huge volumes of data that is linked to a RGB image.

However, and rolling back to the input layer, the pixel values of the input image are not directly connected to the output layer via partially connected layers, but in a fully connected layer where each node in the output layer is connected to a node in the previous layer. This fully connected layer is responsible for classification tasks using the features that were extracted and the different filters, these layers also need an activation function and it is usual to use the ReLU function.

Finally, the output layer will produce the predicted class and this leverages the softmax activation function and this function calculates the probability, in a range $[0, 1]$ as the final output, in this case, if the person has or has not a face mask.

3 State of Art

In this section will be present the results of research on the topics that will be used in for this works. For instance Google has a division, Google AI [13] that offers several tools for tasks that are often associated with an ML project, i.e Image labelling, face detection and text recognition.

3.1 Machine Learning Frameworks

Furthermore, this last example converges into another reason for the attention that Machine Learning is getting, the growing number of open-source Machine Learning frameworks. Launched in 2015 Tensorflow is a big player in the field of ML frameworks, the popularity grew, even more, when Keras, a deep learning API that minimises the number of user actions, became part of the Tensorflow environment. Along with the Keras API, Tensorflow also offers Tensorflow Extended, which consists of a ML pipeline ready for the real-world environment. Moreover, in a world that is getting more connected via smartphone, Tensorflow also offers the Lite version of the library, to implement models in a mobile environment [14].

A different framework that has some popularity among Artificial intelligence researchers is Pytorch, this framework has python as the main language for development, whereas Tensorflow in addition to

[13] `https://ai.google/`

[14] `http://tensorflow.org/`

Python, also has a big community in the Javascript community. Pytorch offers a robust ecosystem for the models to flourish and be used in fields like computer vision, natural language processing and more. Additionally, Pytorch offers cloud integration with the major cloud providers, offering scalability and easy development. Other frameworks will be introduced later in this chapter.

3.2 Datasets

Another reason for this attraction to machine learning is the number of datasets that are public to the world. Consequently in the times, we are living in, these numbers are growing even more. Most parts of the mentioned datasets are hosted on Kaggle, a repository of datasets, that, to this date, contains over 50000 datasets. The most attractive feature is that anyone, person or even a company, can provide datasets to the public world. Google also entered the environment of public datasets and has a search engine that also finds datasets that the governments or even universities make available on their respective websites. In the image classification field that are some datasets that stand out:

- MNIST - Dataset of handwritten digits that contains a test set of 60000 samples and a test set of 10000 samples, resulting in a total of 70000 samples in 10 classes. After being trained, the model can be used in real-world problems with real-world data.
- MS-COCO - Dataset focused on image detection, segmentation and captioning problems, has 330000 images where 200000 are labelled, 80 object categories and over a million object instances [13].
- Open Image Dataset - Open Source dataset with more than 9 million images, where it is divided into a training set with over 9 million images and a validation set with over 40000 images.
- VisualQA - Dataset containing open-ended questions about image content. Has over 265000 images, three questions per image and three answers per question [14].

3.3 Methods and metrics for mask detection detection

Although this theme is attracting more interest among developers, there are many ways for one to achieve a prediction from a model where it is said if a person has or does not have a face mask. Within these ways of achieving the desired result, one key component that can change is the type of the model used for training the phase, the more standard approach is using a Convolutional Neural Network (CNN) , a sub-type of Artificial Neural Networks (ANN), the internal process will

be explained in the Material and Methods section, furthermore, in an experiment [15] with a similar context of this dissertation, the model used was, in fact, a CNN.

However, the one used is a bit different in a way that a convolution block is formed by two convolution layers and a pooling layer, followed by a flatten layers, three groups of dense and dropout layers and finally a layer with two neurons to output the prediction. In this experiment was used a dataset with a total of 1539 images where 20% of the same images were used for validation purposes. The model was trained for 100 epochs and obtained a value of 0.2 in the loss metric, 98.7% of accuracy on the validation data and 0.985 in Area Under the Curve (AUC) and this experiment took 100 epochs to achieve these results.

In another experiment [16], the approach followed was different when it comes to the model construction where a convolution block is composed of a convolution layer followed up by a pooling layer a flatten layer and the final layer responsible for giving the prediction. In this experiment, the dataset is composed of a total of 1376 images where 690 are images of people with masks and 686 of people without masks and 10% of the total number of images are going to be used for validation purposes and 90% to train the model for 20 epochs. This model achieved a value of 0.9637 for the model's accuracy and a value of 0.0855 in the loss metric.

Moreover, there was another experiment [17] where CNN was used, although, with a different architecture where the hidden layers were composed of two convolution blocks, with pooling, followed by two convolution layers without pooling. The dataset was composed of a total of 7740 images from both classes and 70% of these images were used for training purposes, 20% were for validation and the rest 10% for testing and the model ran this dataset for a total of 20 epochs.

In terms of model evaluation, the metrics extracted from the same model were precision, recall and F1-Score where the respective values were 0.99, 0.98 and 0.99. With all of these experiments, it is possible to make a simple overview of these models and the better model developed within this dissertation, which consists of 4 convolution blocks and a dataset of 60 images per class, this same overview can be seen in Table 1. It is safe to say that our better model provides more, not only on the number of metrics extracted but also what these same metrics mean in terms of performance, this meaning and metrics will be explained further in the Metrics section.

	Metrics					
Experiment	Epochs	Loss	Precision	Recall / Accuracy	F1 Score	AUC
Experiment 1 [15]	100	0.2	N.E	0.98	N.E	0.985
Experiment 2 [16]	20	0.085	N.E	0.98	N.E	N.E
Experiment 3 [17]	20	0.02	N.E	0.99	N.E	N.E
Developed Model	18	0.0401	1	1	1	1

Table 1: Experiment metrics overview

3.4 Transparency

Finally and notwithstanding, the reason that is getting more attention from the ML community and the focus of this thesis is transparency. With ML taking over real-life environments and problems, for instance, stock markets [18] and health care [19] along with the GDPR [20] the algorithms that are deployed to the real-world need to be transparent. A user of a certain model needs to get informed on why the model made a particular prediction because the user cannot blindly trust that prediction as the results of the action or actions based on the prediction cannot be measured. Figure **??** shows the accuracy of explanations over the type of explainability.

Nevertheless, let's examine how explainability can be applied to some machine learning models. The most simple model in ML is the linear model, where explainability is straightforward. The prediction is the combination of the feature values, weighted by the model coefficients. When a person gets diagnosed with a certain disease, the doctor, to confirm that prognostic can't access why the model gave a weight of 0.03 to the level of oxygen in the blood. The specialist in the field should be able to change the input variables, if not they would get the same output as long the model was deployed in the real world.

3.4.1 Recourse as Explainability

For these cases, recourse was introduced as "the ability of a user to obtain the desired outcome from a fixed prediction model" [21]. According to Ustun et al. [21], the lack of recourse in an algorithm that provides some form of decision making is often a sign that it requires transparency and explainability and that should be studied as a normal feature that a model should offer. Besides, a model should provide recourse when it affects a good that is essential for a large population, for instance, credit loans [22].

As it is showed in the article, even the simplest linear classification problem, can fail to provide recourse and a reason could be located in the initial modelling practices such as:

- Feature Choice - A classifier can use immutable features, for instance, genre, but also immutable feature, a feature that can change by the passing of time, an example of this type of feature can be *hasDegree*, a feature that indicates that an individual has a degree.
- Choice of Operating Point - In a request for a loan scenario, the same model may provide recourse in a threshold, for instance, recourse is given when the model predicts a denial of a loan with a risk greater than 50%, but denies recourse in the same prediction but with the risk being greater then 80%.
- Out-of-sample Deployment - A feature that can be changed to achieve an outcome is missing or immutable.

Consequently, the effects of these practices in the model development can vary significantly in the offer of recourse, and, ergo its target population. Therefore, when auditing a model before deploying it to the real world, it should "evaluate both feasibility and difficulty" for providing recourse for individuals that are the target population of a model.

To achieve recourse, Ustun et al. [21] present a tool that returns a list of actionable items that an individual can change to achieve the desired output.

Features to Change	Current Values		Required Values
n_credit_cards	5	⟶	3
current_debt	$3,250	⟶	$1,000
has_savings_account	FALSE	⟶	TRUE
has_retirement_account	FALSE	⟶	TRUE

Figure 10: Recourse list example [21]

After modelling the optimization problem, where several variables were accounted for, like the feasibility mentioned before, the cost of providing recourse and the audit cost came the part to train the model and access the results. Beginning in the setup, the dataset used was a processed version from UCI Repository [23]. This dataset contains 30000 individuals and 16 features related to data asked when a person requests a loan, for instance, education, credit history, payment pattern and education. For the experiment, payment patterns and education are actionable and the other features are immutable. All the classifiers were trained using scikit-learn [24] with a standard 10-fold cross-validation to tune free parameters and estimate predictive performance. The training phase consisted in training ℓ_1 models based on penalised logistic regression, a algorithm that penalises the logistic model for having too many variables [25] and in the experiment the penalties were $\{1, 2, 5, 10, 20, 50, 100, 500, 1000\}$. Afterwards, the training data was audited, but only in the cases where the prediction was that a user may fail a future credit card payment.

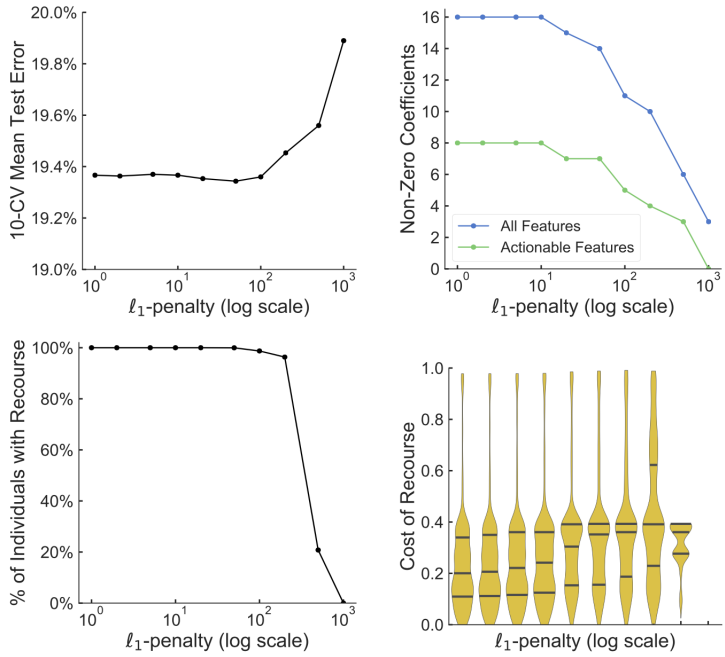


Figure 11: Recourse models performance [21]

As shown in Figure 11, increasing the penalty value has a minimal effect on the percentage of errors, but, in contrast, it increases the cost of providing recourse. It is also visible that models with low penalties provide recourse to all individuals in the population. On the other hand, increasing the penalty, decreases the number of actionable features. Looking more into the costs of recourse increasing the penalty almost double this cost and, according to Ustun et al. this increase goes from "0.29 to 0.39". The tools that the authors offer provide information for a practitioner to choose between classifiers that indicate the feasibility and the cost of actionable recourse in day-to-day development.

The next experiment was focused on an underrepresented population, more specifically, young adults, because they lack the credit history to apply for a loan and produce labelled data. For this experiment, the dataset used a processed version of the Give Me Some Credit [15] dataset where a class, $y_i = -1$, is defined if a person, i, will experience financial struggles over the next two years. The dataset contains over 150000 individuals and 10 features related to age, recent financial history and number of dependants. Moreover, all the features except for *age* and *numberOfDependents* are actionable. The auditing compares the cost of recourse for individuals in a population for two penalised logistic

[15] `https://www.kaggle.com/c/GiveMeSomeCredit/overview`

regression models:

- Baseline Classifier - Baseline models that are used for comparison. It uses all of 150000 individuals to train, which consists of the target population.
- Biased Classifier - The model that was trained on a population of 100000 individuals in the processed dataset, where individuals with an age inferior to 35 years would be under-sampled by 90% from the baseline dataset. Afterwards, this model was audited.

Note that the model was developed to approve half of the individuals in the population, for instance, the model will deny a loan even if the probability of an individual paying the same loan is higher than 90%.

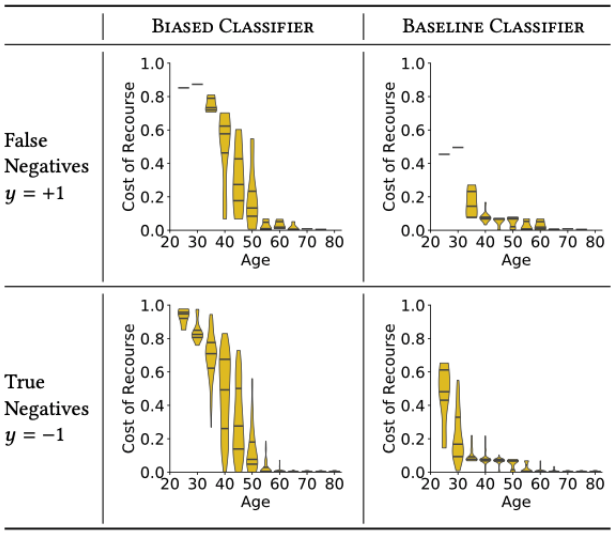


Figure 12: Second Experiment Results [21]

Analysing Figure 12 some conclusions can be retrieved. First, the cost of recourse for young adults, that have less than 35 years old is much higher for the biased classifier, independently of being a false negative or a true negative. This experiment has the main goal to show that out-of-sample deployment, using an underrepresented population, increases the cost of recourse.

The last experiment was focused on evaluating disparities in recourse, specifically, between females and males in a target population. The disparity occurs when two individuals from each gender have their loan request denied, thus, the goal is too compare who gets the easier changes between the features. The dataset used was a processed version of the *german* dataset on the UCI Repository [16] that

[16] `http://archive.ics.uci.edu/ml`

contains 1000 individuals and 26 features related to loan applications, financial status, demographics and to fully achieve the goal of this experiment it also contains the gender of the individuals. This experiment $y_i = -1$, shows if an individual, i, is a bad customer, meaning that they have not paid a loan. The classifier used penalised logistic regression, where the threshold approval was set to approve an individual with the predicted probability of 80%. The audit process took place on the individuals that have the loan request denied and, afterwards analyse the disparities with the individuals that got accepted on the request for a loan, $Pr(y_i = 1)$.

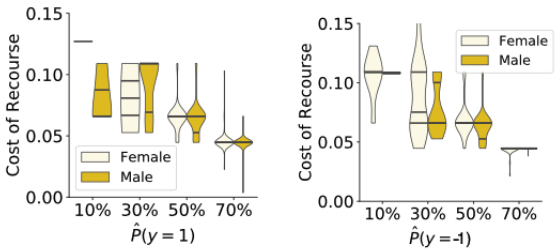


Figure 13: Cost of recourse disparities [21]

In Figure 13 it is visible that the cost of recourse can be volatile between males and females in the population. The plot on the left shows the cost of recourse for male and female individuals with the label $y_i = 1$, on the other side, it shows the cost of recourse for the individuals of the two genres but that got predicted with the label $y_i = -1$.
However recourse may present some limitations. Firstly, the list of actionable features could mislead a decision and this happens if a model uses a large number of features and then an individual who sees this list could reduce the score by changing some features that are not in the initial list. Lastly, there is always a chance of manipulation of these lists. Furthermore, it can lead to an individual manipulating the model to obtain a certain desired output according to [21]. this type of manipulation can be avoided by deploying the lists "with actions about features that are causally related to the predicated outcome". With all these limitations it is a very positive way of providing explainability to an individual or a group of individuals that wants to know how a certain model predicted a certain output.

3.4.2 Random Forests for Explainability

Moving to another ML algorithm, random forests, an algorithm that consists of several decision trees, uncorrelated between each other and each tree in the "forest" returns a predicted outcome and the

prediction that has more return from the whole "forest" is the final prediction [26]. In Figure 14 the final prediction will be 1 since six trees predicted 1 and only three predicted 0.

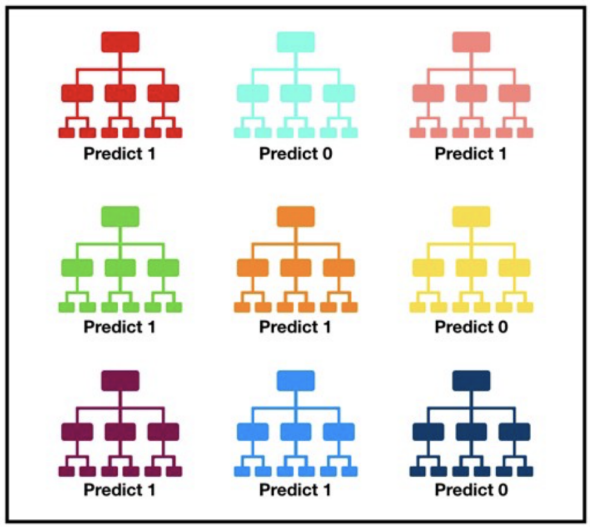


Figure 14: Random Forest Prediction Example [17]

Airbnb presented a way of explaining the outcome of a prediction. Cheng integrated the Trust and Safety team, thus, it was crucial providing accurate risk value when a user tried to rent a house. However, since the features need to be split to obtain a prediction, it is difficult to understand how these same features got split in the first place. So, they proposed and developed a method that aggregates and summarizes the split values by "generating weighted threshold distributions" [18].
Random forests, along with good values in terms of performance, could be also considered a black-box model. In the first place the node decision split is chosen from a random subset of the features set and, secondly, the model generates an ensemble, a model based on combinations of weaker models [27] from the training set, thus, making this type of model even more "obscure". Moreover, a feature might split into specific nodes with distinct values, and, this event might happen within other trees from the "forest".
With all these inconclusive numbers being returned by the model and during the training, it is still known their values for each node where they were splitted, and, thus, the threshold for these same values. The challenge was how to connect all this information to provide some type of explanation. A way to describe the result from the training phase of a random forest in terms of features is to rank

[18] `https://link.medium.com/1CnoTBU3ihb`

them by importance with the splitting efficiency. However, this method does not provide how the model made a decision based on the features. In this way, Shih-ho Cheng proposes a way to describe the decisions from a forest by mining each node from the same forest and presenting the weighted distribution, recurring to the split efficiencies and the sample size, of the threshold for each feature. To demonstrate this proposal, it was used a dataset containing online news popularity [19], with over 39000 observations and with a total of 58 features. An observation is declared "popular" if the number of shares was greater or equal to 1400. The features are all numeric and contain ranges, for instance, the number of words.
After the training phase, the forest was crawled to retrieve, from the non-ending nodes, the following information:

- Feature name
- Split threshold - The value from where a node splits
- Sample size - The number of observations that passed through a certain node
- Greater than the threshold?
- Gini change - Decrease in impurity after the split
- Tree Index - Identifier for the tree

This features can be collected into a table, see Table 2.

[19] `http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity`

feature_name	split_threshold	sample_size	greater_than_threshold	gini_change	tree_index
LDA_02	0.594373	18847	0	0.010756	0
kw_avg_avg	2915.460938	15949	1	0.011636	0
LDA_01	0.033449	7199	0	0.006488	0
data_channel_is_socmed	0.500000	3661	1	0.007763	0
self_reference_avg_shareness	1933.046631	3484	1	0.007761	0
weekday_is_sunday	0.500000	1635	1	0.007912	0
kw_avg_max	143320.39565	1534	0	0.011096	0
title_sentiment_polarity	0.341667	567	1	0.009926	0
weekday_is_thursday	0.500000	486	0	0.027183	0
is_weekend	0.500000	396	1	0.016704	0
...	...	...	...	...	...

Table 2: Example of feature values extraction

Unlike the table above, the same feature could appear multiple times, but in different trees across the forest. So, one could be tempted to extract all the thresholds to a specific feature, however, it does not take into account that optimal feature threshold tuples are found within several observations. Hence, for a splitting node, i, it defines the sample size N_i as the number of observations that reached the splitting $node_i$ during the training phase. Furthermore, a large change in the impurity, $gini_/change$ in Table 2, should be also considered and, thus, weighted, but, the near-zero changes are less weighted.

After the training phase, the model was crawled through the entire forest to obtain all the information needed to build a feasible explanation example. This information provides the thresholds that dominate the splitting.

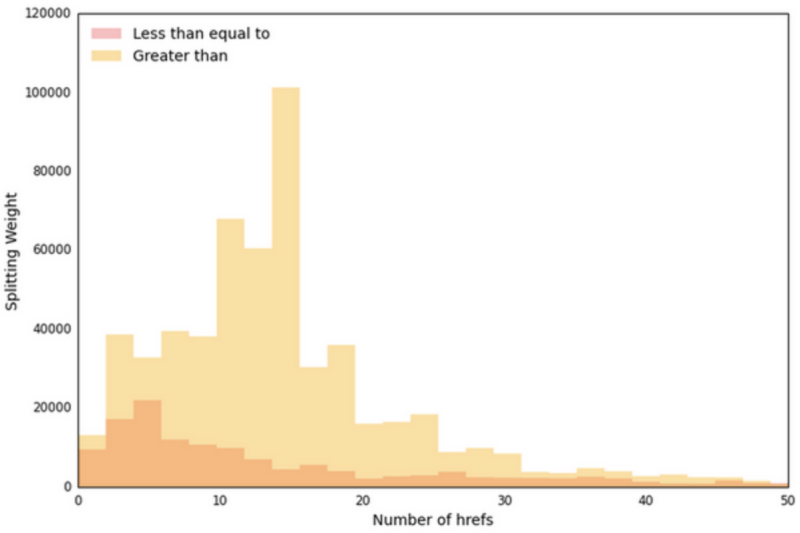


Figure 15: Number of links threshold[20]

Figure 15 shows an example for the thresholds, in this case, the number of links a certain article has. The orange distribution indicates that, in the situation where the feature num_refs is used to decide whether an article is popular or not, and has over 1400 shares, the dominant description is greater than, approximately, 15 links.

Another important feature that can be explained and, thus, analysed is the rate of positive and negative words. In Figure 16 it is visible the global rate of positive words, and, we can conclude that the rate that dominates popular articles is around 0.03.

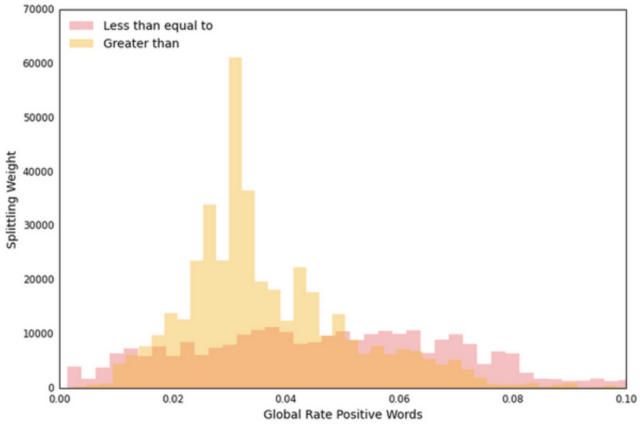


Figure 16: Rate of positive words

Passing through the global rate of negative words, as shown on Figure 17. In this image, it is

possible to say that a rate of negative of, approximately 0.01 adds a boost of popularity to the article, however, increasing the rate to 0.02 will make the model predict the lower popularity of that article.

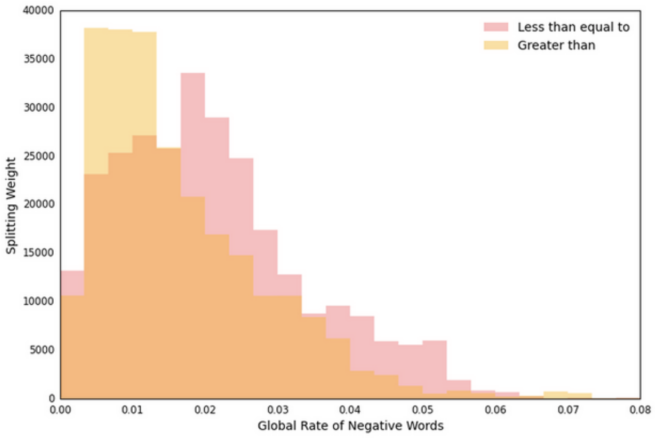


Figure 17: Rate of negative words[21]

For future work the Trust and Safety's AirBnB team will try to cluster observations and how the model reacts and how the features interact within the random forest.

3.4.3 Algorithm agnostic Explainability

As previous examples have shown, the solutions were too attached to the algorithm itself. However, a new generation of algorithms started to appear, such as neural networks, ensemblings, gradient boosting trees, etc... In this way, new techniques for explaining a classification problem using any algorithm started to emerge.

A first approach was developed by Ribeiro et al.[8] where they propose LIME, an open-source project that "provides explanations for individual predictions as a solution to the problem" [8] where a user cannot decide if he should trust a prediction. However, it was necessary to define trust in this context:

- Trusting a prediction - Whether to trust in an individual prediction that an action is taken based on the final prediction.
- Trusting a model - A user trust that the model will have a certain reasonable behaviour in the real world with real data

Furthermore, as these definitions of trust get more user, the bigger is the impact. Moreover, when the model is used in decision making, determining trust turns into a crucial task, and predictions without trust should not be acted on blind faith, because the consequences can not be measured. Notwithstanding, in the model development phase, these same models need to be evaluated, according to relatable metrics, as a whole before being released to the real world and having a good performance. At the moment, models are evaluated by getting accuracy metrics that are given by the validation dataset against the training dataset, yet, the real world can be significantly different from the dataset values used for validation. Likewise, it is important to evaluate individual predictions and respective explanations and it could be also helpful if the users got some aid by suggestions on which instances to evaluate.
Along with LIME, [8] proposed two more ideas that jointly provide a solution to the suggested instances to the user to evaluate the approach and the approach used to evaluate the solution:

- SP-LIME - Method that selects a set of representative instances with the respective explanations to tackle the trust a model issue.
- Both LIME and SP-LIME were tested with real people, that, in the experiments, users are unaware of LIME's explanations. They can select which classifier, from a pair, generalises better the real world, the one that is closer to the real world.

Along with the investigation, [8] came up with some desirable characteristics for explainers.

- A model must be interpretable, meaning, provide qualitative understanding between the input variables and the output. The explanations should be easy to understand by the user. Moreover, the notion of interpretability needs to be in the context of the target audience.
  - A possible interpretable representation for image classification could be a binary vector that indicates the presence or absence of a continuous patch of similar pixels.
- The models have to provide local fidelity. For an explanation to be valuable, it must be locally faithful, meaning, it must correspond to how the model behaves in the proximity of the instances that are being predicted.
  - The features that are globally important may not have the same level of importance in the local context and vice-versa.
  - Global fidelity implies local fidelity, however, identifying global faithful features may be a challenge of high complexity.

- An explainer should be model agnostic, thus, it should be able to explain any model. Moreover, it should treat the model as a black box.
- The model should provide a global perspective, meaning, standard metrics, such as accuracy, may not be suitable to evaluate the model behaviour, thus, the necessity of providing explanations for the predictions.

Initially, an explanation is defined as a model $g \in G$, where G is the potential class of an interpretable model. For instance, a model $g \in G$ can be presented to the user with visual artefacts, and, g acts over the presence/absence of the interpretable components.
However, not every model is simple enough to be considered interpretable. Thus, $\Omega(g)$ represents the complexity to explain a model. In decision trees the complexity may depend on how deep the tree is. Furthermore, the probability of an instance belonging to a certain class x is defined by $f(x)$. Moreover, to ensure local fidelity, $\pi_x(z)$ defines the proximity between an instance z to x, thus, defining the locality around x. Finally, $L(f, g, \pi_x)$ helps to measure how faithful g is in approximating to f in the locality defined by π_x. The final explanation that is produced by Lime is obtained by equation 2.

$$\varepsilon(x) = argminL(f, g, \pi_x) + \Omega(g) \tag{2}$$

The goal is to minimize the locality-aware loss, L, without getting assumptions from f for the reason that an explainer must be model agnostic. Furthermore, to learn how f behaves in the locality with the variance from the inputs, L must be approximated by drawing samples that were weighted by π_x, the proximity.

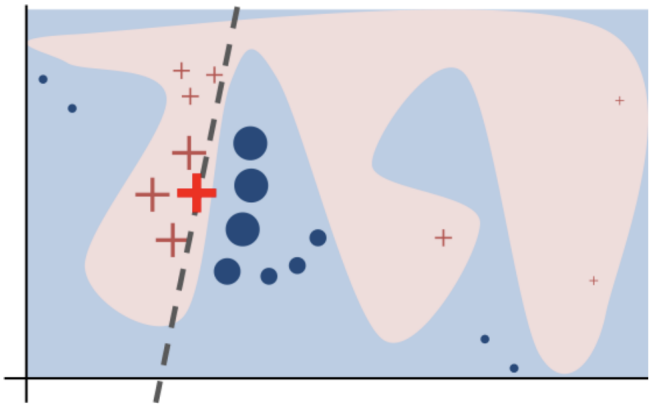


Figure 18: Intuition for LIME [8]

Figure 18 shows the first intuition that LIME does about an instance's prediction, the red bold cross. Moreover, to get this result, LIME samples instance, by getting the predictions using f and by calculating the weights along with the proximity to the instance that is being explained, the weight is proportional to the size of both the crosses and the circles.

The first experiment consisted of using random forests with 1000 trees and using $scikit - learn$ [24] with a total number of samples, N, of 5000. It took under 3 seconds to explain the result from the random forest without any parallelization, however, it took about 10 minutes to explain each prediction from the Inception Network [28] dataset for image classification. The result for this explanation can be seen in Figure 19 where a model was trained using a K of 10 and the objective was to show the super-pixels for the top 3 predicted classes.

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 19: Image Classification Explanation Example [8]

An interesting thing to note is that Figure 19b shows what the model considered as an electric guitar, although wrong, it still provides trust because it shows that the model is not acting so good

in certain edge cases.

Another key feature that [8] presents is a way to choose several instances, that gives a global understanding of the model, to show to the user. This number of instances to show to the user needs to be chosen wisely, since the user may not have the time or the interest to inspect a large number of explanations, so let B define the number of instances that the user is willing to evaluate.

A process that leads towards a possible value of B is the picking step and this process does not depend on the existence of explanations, the raw data is not enough to understand predictions and it should also take into account the explanations that may accompany each prediction. Figure 20 shows an example of the pick step process.

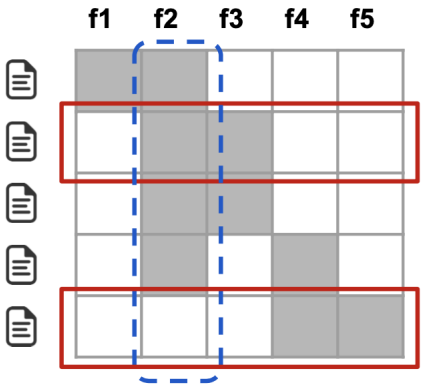


Figure 20: Pick Step Example [8]

Each column, j, in the matrix, W, has a I_j that represents the global importance of that component, words on a text or patches in an image, in the explanation space. The features to be selected must be the ones that have a higher value of I_j. In the process of picking the features that cover important components, the explanations must not be redundant when they are presented to the user. To avoid this kind of situation, a workaround could be avoiding selecting instances with similar explanations.

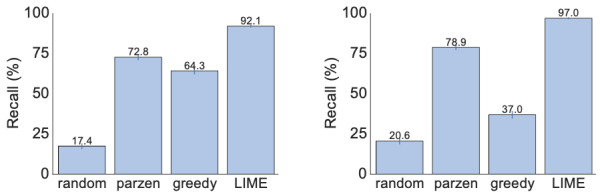
Afterwards, it was time to do some experiments. The first experiment was with simulated users, it had the objective to get an answer to three questions:

- Are the explanations faithful to the model?
- Can the explanations aid the user in trusting the predictions?
- Are the explanations useful for evaluating the model as a whole?

The data used for this experiment consisted of two sentiment datasets, books and DVDs with 2000 instances each and the main task was to classify product reviews as positive or negative. Each dataset was divided into a train set with 1600 instances and a test set of 400 instances. To compare individual explanations provided by LIME, [8] used three approaches:

- Parzen - A method that approximates the classifier globally recurring to Parzen windows, the K features with the highest absolute gradients are picked to provide explanations.
- Greedy - A procedure that removes features that contribure the most to the predicted class until the prediction changes or the maximum value of K is reached.
- Random - A procedure that randomly picks K features as an explanation

For the experiments, the value of K was 10. To pick the explanations to present to the simulated user, both the picking step and a random selection were used. Both datasets were trained using decision trees and sparse logistic regression, an algorithm that uses feature selection in the classification using the l_1-norm regularization [29]. Figure 21 and Figure 22 shows the recall averaged over all the test instancesand it is observable that the greedy approach has similar values to Parzen on logistic regression, although, as substantially different values on decision trees, because changing a single feature at a certain time, does not affect the prediction. However LIME, offers over 90% recall for both classifiers and both datasets, thus proving that the explanations provided by LIME are faithful to the model.



(a) Sparse LR (b) Decision Tree

Figure 21: Recall on important features on the books dataset [8]

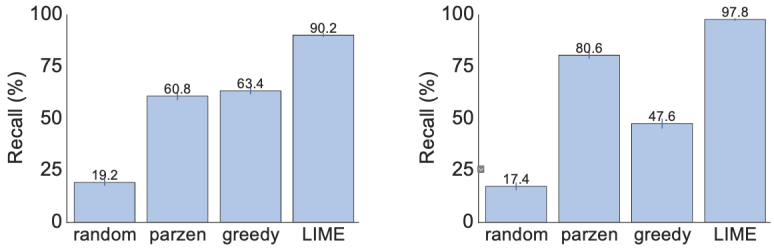


Figure 22: Recall on important features on the DVD's dataset [8]

To answer the trust question and if a user should trust a prediction, 25% of the features are labelled as untrustworthy, and it was assumed that the users could identify and would not trust these features. Afterwards, an oracle was developed to label the predictions from the test set as untrustworthy if the prediction changes when untrustworthy are removed from the instance, and trustworthy otherwise. To simulate the users, it was assumed that the user considers the explanations given by LIME and parzen as untrustworthy if the prediction's untrustworthy features are removed. For greedy and random processes, a prediction is mistrusted if any of the untrustworthy features are in the explanation.

To validate the experience, the metric F_1, a metric that shows an average value of the precision and recall, was evaluated on trustworthy predictions for each method.

	Books				DVD's			
	LR	NN	RF	SVM	LR	NN	RF	SVM
Random	14.6	14.8	14.7	14.7	14.2	14.3	14.5	14.4
Parzen	84.0	87.6	94.3	92.3	87.0	81.7	94.2	87.3
Greedy	53.7	47.4	45.0	53.3	52.4	58.1	46.6	55.1
LIME	96.6	94.5	96.2	96.7	96.6	91.8	96.1	95.6

Table 3: F1 Score Values

Table 3 shows the values for the F_1 metric and it's observable that LIME offers values consistently above 90, although the features were artificially labelled as untrustworthy it shows that LIME can assert trust in individual predictions.

To answer the final question, if the model can be trusted, [8] experimented to evaluate if the

explanations provided by the model could be used to select a model, the user could have to choose between two or more models that have the same accuracy metrics. Moreover, it is also to evaluate whether a user could identify the better classifier based on explanations of B instances from the validation dataset. First, it was added 10 new "noisy" and artificial features to the training and validation datasets, where each of these features appeared in 10% of the examples in one class and 20% in the other class, and another 10% appeared in the test instances. Afterwards, the simulated user labels the set of features as untrustworthy and then it is evaluated how many predictions in the validation dataset should be trusted.

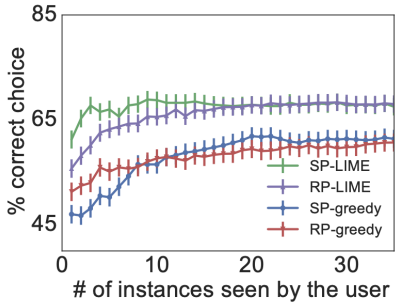


Figure 23: % of correct choices on book dataset [8]

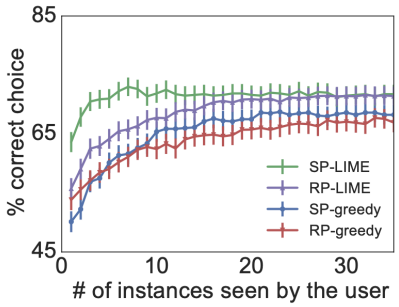


Figure 24: % of correct choices on dvd dataset [8]

Figure 23 and Figure 24 represent the accuracy of the correct choices and according to [8] both the SP-Parzen and RP-Parzen were omitted due to the fact that both didn't produced useful values. Moreover, it is visible that LIME offers more consistent values over the instances that the "user" selected.

However, the data that is used during the training phase can induce that the model can do some undesired correlations. To tackle and try to reproduce this behaviour, [8] experimented with real users that interacted with the model that was being developed to classify, based on images, wolves and Eskimo dogs (huskies). The 20 images used were hand-selected and all the wolves images had snow in the background and the huskies did not. For an early feature extraction, Google's pre-trained Inception neural network [14] was used with an addition of 60 images. The classifier predicted that the animal on an image was a "Wolf" if there were snow, as shown in Figure 26, or light/white background, and Husky otherwise where the goal was to evaluate if the users would detect this problem of bad classification.

Figure 25: Husky classified as "Wolf" [8]

Figure 26: Explanation [8]

Afterwards, the users had to answer three questions:

- Do they trust in the algorithm.
- Why?
- How do they think that the model distinguishes a wolf from a husky.

The subjects were asked these questions after interacting with the model and again after 3 independent evaluations explained to them how the model was built. Table 4 shows how the number of subjects had their trust in the model twisted after knowing the same was built to predict bad outcomes.

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a feature	12 out of 27	25 out of 27

Table 4: Experiment Results

This kind of experiment demonstrates how it is important to explain individual predictions to get some insights into whether the model is trustworthy or not.
Later, in May of 2017, another open-source framework appeared in the field of explanations, SHAP,

a framework that consists unifies LIME, regarding model agnostic explanations, and other several methods to return robust, consistent and human-readable results. Shapley Additive Explanations (SHAP), assigns an importance value to each feature, then, proceeds to identify new classes by adding measures of the importance of a feature and, finally presents some hypothetical results showing the class predicted along with the properties/explanation for that prediction [30]. Figure 27 shows how SHAP presents the explanation when it is a problem of image classification, pretty similar to LIME. SHAP is considered at this time, the state of art framework in the explanation field.

Figure 27: SHAP Explanation Example [30]

In general, statistical models provides the likelihood of an image compared other images given by the auditors. Furthermore, the method mentioned before has many approaches. One is that a model adapts to image patterns noticed during the learning process.

3.5 Learning process

To trigger the process, active learning, firstly, needs data, that can come from different external sources. The data has two distinct types of streams, the data that has the identifier of being a person with a mask, and another with the person without mask tag, but a problem appear when the requirement to start the learning process specifies that the data needs to come already labelled with one of the classes, at least in one of the total data sources connected to the process [31]. A practical example could be in fraud classifiers where an ideal implementation of this method could be by firstly, separating the data previous labelled by fraud or non-fraud in distinct datasets, or even by fraud type, by having a different dataset for each type of fraud we improve the learning process

since each fraud type has is deviance between transaction values [31].

3.5.1 Reinforcement Learning

Staying in the field of ML, another method is reinforcement learning. This technique distinguishes itself from active learning in the way that reinforcement learning does not need the first labelling process from the human. An agent is responsible for the model's learning process by interacting with an environment, thus changing the state of this one. Furthermore, the model does not know which actions to take but must discover a set of tasks that may bring more rewards and fewer penalties from trying the same set of tasks. There are particular cases in that tasks have certain dependencies between each other, thus impacting more than that process if a task fails, so, this turns into a trial-and-error process [32].

The problem with Active Learning is that the data requires to be previously labelled as one of the classes and doing this labelling work can take too much time and too many human resources. Moreover, to fully implement the streams of data that are responsible for pulling the data and feeding the adaptive classifier engine, more human resources need to be allocated and possible work closely with the auditors to get the best well-designed data system, this can take time of the human and the time to label the images, more specific labelling the data, gets smaller.

Since this proposal puts the agent in a crucial part of the process, the reinforcement learning approach would not cover the main use case, since the decision part comes from a developed agent. Furthermore, first, developing a prototype can take too much time and thus, human resources. Nevertheless, getting an optimised agent that returns results with a certain confidence level takes a lot of time and consequently computational resources. By this, the humans could never be a part of the classification process, because it lies with the agent discovering, among large datasets, the class of a certain image.

3.6 Image Labelling

Moving on to image labelling, this process is the core of the active learning cycle, where the human labels an image and adds a class to the zone that was drawn in an image. In this context, there are two open-source projects that stand out, one is the LabelImg [22] that, simplistically, allows the user two draw a rectangle in a certain area of an image and classify the image according to previously

[22] `https://github.com/tzutalin/labelImg`

created classes as Figure 28 demoanstrates.

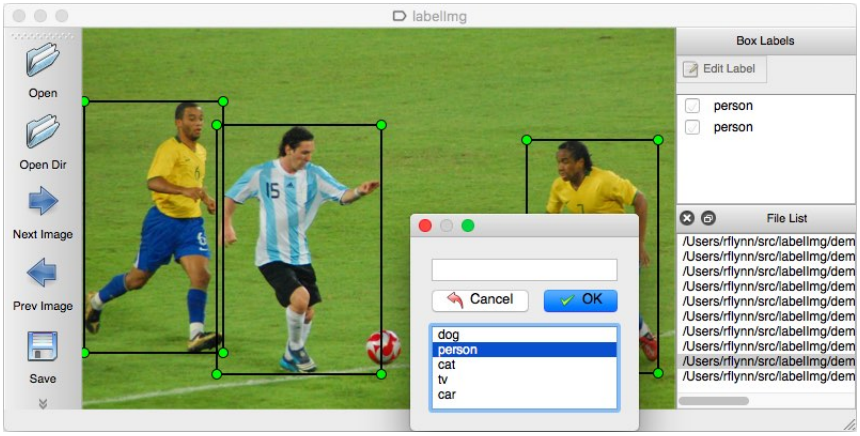


Figure 28: LabelImg labelling process example [23]

The second open-source labelling project is Rectlabel [24] and different from LabelImg, this framework allows users to detail specifically for the bounds of the part of an image they wish to select as seen in Figure 29. Along with this key feature it also allows users to select super-pixels of an image and manage image settings for objects with corresponding attributes and hotkeys for faster labelling.

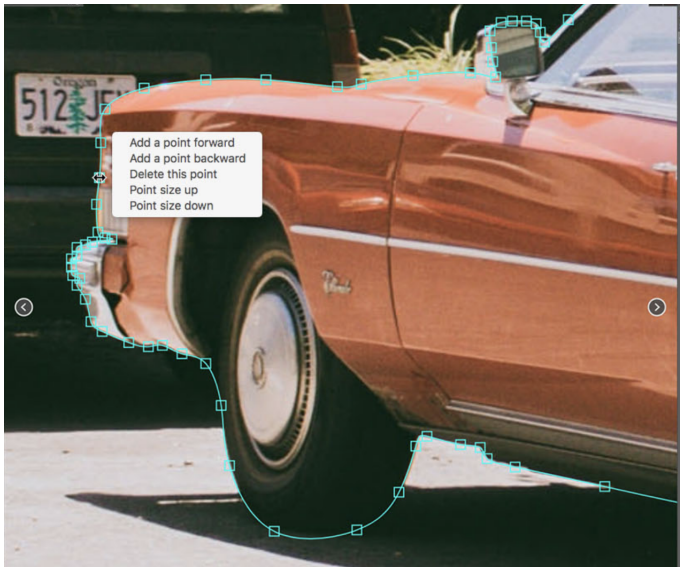


Figure 29: Rectlabel labelling process example

[24] https://github.com/ryouchinsa/Rectlabel-support

4 Material and Methods

4.1 Technical approach to a Convolutional Neural Networks

In this project, we used a CNN for image recognition, due to the advantage of identifying key elements in the labelled image along the training process, and translating the explanation above to code is quite simple, due to the Tensorflow library and its documentation.

Before passing the image to the input layer of the network it is usual to rescale the image and this process helps the model perform better since some images have multiple features rescaling transforms the pixels values that, initially, have values in the range [0, 255] to values in the range [0, 1] by dividing each pixel value by 255 and this operation is represented in Code Block 1.

The first parameter is the the the operation that will be done in each pixel, in this case, the value will be divided by 255 and the second parameter is the input shape, that is a tuple of order 3 that receives the image dimensions and the input channels since it is all around RGB images this value is 3.

```
layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(img_height,
    img_width, 3))
```

Code 1: Rescaling RGB image

After this first preprocessing the image will go on the CNN into the first convolution block, which consists of a convolution layer followed up by a max pooling layer. In the convolution layer the first parameter is the number of filters, and a best practice when it comes to defining this value is that the first block should have the value of 16 or 32 and the following blocks should have higher values and be a power of 2 and the second parameter is the filter size of the matrix that will slide across the image to detect important features and the second parameter is the hyperparameter padding that has the value *same*, this will apply the same padding technique explained in section 3.3.4 and the last hyperparameter is not explicitly in the layer but the stride has the default value of 1, meaning that the filter will slide 1 unit over the image. Finishing a convolution block comes the pooling layer and the default arguments are the same explained above where the matrix size is two by two and the default value for padding is *valid*. These concepts are succinctly implemented in Code Block 2.

```
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D()
```

Code 2: Convolution Block Example

Moreover, a CNN is composed by more than 1 convolution block and Tensorflow shows and example of a CNN with 3 convolution block as Code Block 3,

```
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(32, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D()
```

Code 3: Tensorflow Convolution Block Example

Lastly, comes the fully connected layer, represented in Code Block 4, that consists of some steps. A step that flattens the input and does not have an impact on the batch size produced from the last convolution block followed up by the fully connected layer itself that is represented by the Dense in line 2 that, in an initial phase, has 128 units and has the *relu* activation function. Finally comes the output layer that has the number of classes and the $softmax$ activation function to produce the prediction.

```
layers.Flatten(),
layers.Dense(128, activation='relu'),
layers.Dense(2, activation='softmax')
```

Code 4: Output Bock Example

Code Block 4 represents a standard approach to a CNN, and working with RGB can be heavy in terms of processing since the images and respective filters have to traverse in a width of 3 thus, the next section will tackle the process to overcome with a possible model with desirable metrics value.

4.2 Achieving a desired model

The process to achieve the desired model will consist of, firstly obtaining a group of metrics that will show how the model is progressing during the training phase. Following up several models will be built to study the metrics obtained above and these models will defer from each other in some aspects but not on an internal level this models will also be running against a predefined number of images to evaluate how the model behaves.

4.2.1 Metrics

In classification problems there are two metrics that are collected during the training phase, precision and recall. However, it is valuable that the terms like true positive and false negative are understood and these concepts can be easily clarified by observing Table 5 and these terms are expressed in quantities.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Table 5: Classification terms

Furthermore, precision is expressed by Formula 3 and by analyzing this formula it is visible that precision infers how accurate the model was, by quantifying the number of positive class predictions that belong to the positive class. Moreover, this metric is a good indicator of false negatives, since the higher his values, the lower the precision.

A precise model may not find all the positive cases, but the ones that the model class as positive are most likely to be correct, however, a not precise model may find a lot of positives but the selection method is not the best, thus, the same positives the model finds may not be positives and, consequently, leading to mistakes.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3}$$

In practice a false positive means that a person is not using a mask, but the model predicted in some way that the person is wearing a mask, and this situation is not ideal since that person may, or may not, be a carrier of a virus.

Moving forward, recall, given by Formula 4, calculates how many actual positives the model captured by labelling the prediction as positive, thus, making recall a good metric to understand the behaviour of the model in terms of correct classification and by addition, giving more trust to the model. However, a model with lower recall is not able to find a large part of the positive cases.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4}$$

Nowadays, besides precision and recall, the F1 Score is a standard metric that can be retrieved from the model and this metric is useful when it is needed to achieve a balance between Precision and Recall and is good in situations where there are imbalanced data. A model will have a high F1 score if both the precision and recall have high values, a low F1 score shows that both precision and recall are low and, what can be considered a medium value means that one of precision and recall has high values and the other has low values.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

Another metric that will be extracted and analysed from the model is the Area Under the Curve, AUC for short. AUC represents the measure of separability, it tells how much capacity the model can distinguish between classes. And to achieve it, this metric leverages the True Positive Rate, shown in Formula 6 and the False Positive Rate, demonstrated in Formula 7.

$$TruePositiveRate = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{6}$$

$$FalsePositiveRate = \frac{FalsePositives}{FalsePositives + TrueNegatives} \tag{7}$$

In the end, it will display a graph similar to the one on Figure 30, meaning that the higher the AUC the better the model can classify a prediction as mask if the person is wearing a face mask and no mask if the person is not wearing one.

These metrics must be provided to the model by adding them to the *metrics* parameters as Code Block 5 shows, but Tensorflow offers an out-of-the-box metric that is going to be used in a callback, this metric is called *loss* and represents a penalty for a bad prediction, moreover, the term "loss" refers to how inaccurate the model's forecast was in a particular case. The loss is 0 if the model's forecast is flawless, otherwise, the loss is greater.

```
1 model.compile(optimizer='adam',
2                 loss='binary_crossentropy',
3                 metrics=[metrics.Recall(), metrics.Precision(), metrics.AUC(),
   tfa.F1Score(num_classes=2, average='micro')])
```

Code 5: Compile Model Method

[25] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

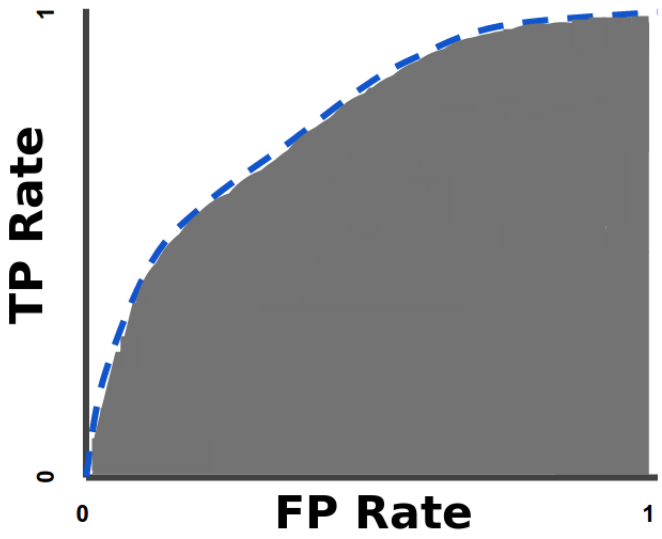


Figure 30: Area Under the Curve Example [25]

Furthermore, during the training phase, Tensorflow offers metrics that are related to the validation dataset, meaning that the metrics retrieved from that same dataset and these metrics are the same as that mentioned in Code Block 5 plus the loss. This validation metrics are useful for many cases, but in this case, the loss on the validation dataset is going to determine when the model should stop training and this process is commonly known as Early Stopping and this method is used as a callback on the fitting of the model and this process translates into Code Block 11.

To note that the metric F1 score, will receive another parameter and this parameter, called average, if not present will return the values for each class and each dataset, validation and training, however, the readability of these values can be compromised and some false conclusions could be taken thus, this metric will have the value *micro* meaning that all the variables needed to calculate this metric will be computed globally and a more detailed explanation will be provided on the first dataset performance analysis.

```
1 callback = EarlyStopping(monitor='val_loss', patience=3, mode='min',
    restore_best_weights=True)
```

Code 6: Early Stopping

Simplifying the code shown above, the model will monitor the validation loss value and will check if this value improves, if it does not improve in a 3 epoch interval with a minimum value, the model will stop and restore the best weights that were achieved during this process. This callback is then passed to the fit function of the model, the process responsible for the actual training, as shown in

Code Block 7.

```
model.fit(train_dataset, validation_data=val_dataset, epochs=epochs, callbacks=[
    callback])
```

Code 7: Fit Function

4.2.2 Description of the runs

As said before, to achieve a preformative model, first we want to try some combinations of layers and several images to verify the performance of the models. However, some boilerplate works need to be done. Firstly the number of images needs to be labelled, using the software LabelImg and since it generates a XML file with all the information related to the image and the label, this same file needs to be treated dynamically to process this information.

```
1  def crop_images_from_directory(directory, dest_dir):
2      for file in os.listdir(directory):
3          if file.endswith('.xml'):
4              full_path = str(directory + '/' + file)
5              image_path = get_attribute_from_xml(full_path, "path")
6              label = get_attribute_from_xml(full_path, "object/name").pop().text
7              image = image_path.pop().text
8              image_directory = get_bound_box_object(full_path)
9              filename = os.path.basename(str(directory + '/' + file))
10             with Image.open(image) as img:
11                 (xmin, ymin, xmax, ymax) = (image_directory['xmin'],
12                                             image_directory['ymin'],
13                                             image_directory['xmax'],
14                                             image_directory['ymax'])
15                 cropped_img = img.crop((int(xmin), int(ymin), int(xmax), int(ymax))
    )
16                 cropped_img.save(str(dest_dir + "/" + label + "/" + filename.split(
    '.')[0] + '.jpg'))
17         else:
18             print(str(file + " not compatible"))
```

Code 8: Processing the XML File

In Code Block 8 this process starts by extracting the information needed to send the image to the correct path since the model needs the images separated in folders with the class name as seen in lines 4 through 9.

Next up, the image is cropped, using the python Image module, by the values that are on the file that corresponds to the corners of the square or rectangle that was made to label the image and right after this image is saved in the correct folder that is named after the images classification by the labelling process and this process can be seen in line 10 through 16.

This process is customised to this use case, the rest of the process is the standard way suggested by Tensorflow. Code Block 9 shows how to create a dataset for training, where it uses the Tensorflow `image_dataset_from_directory` function to generate a dataset where 20% of the dataset is used for validation, line 4, and receives the image dimensions as parameters, line 8. On an important note, the parameter *batch_size*, line 9, will work as a hyperparameter. Not having a batch size defined means that the images will be fed to the training phase individually, however, this kind of process takes time and it will influence the final metrics since each image is different and key features identified will also be different.

Since not having a batch size has significant costs, and we know the number of images the model will be fed, it is possible to give this parameter values but it must not be a high value since this will lead to overfitting of the data and, usually, this value is a power of 2. However, since we know already the number of images this parameter will have the value of 10, this way the total number of batches for each dataset will be an integer value.

```
1 def generate_train_dataset(directory, img_height, img_width, batch_size):
2     return image_dataset_from_directory(
3         directory,
4         validation_split=0.2,
5         label_mode='categorical',
6         subset="training",
7         seed=123,
8         image_size=(img_height, img_width),
9         batch_size=batch_size)
```

Code 9: Generate training dataset

The same happens to the test dataset where the only change from Code Block 9 is the `subset` parameter where in the test dataset is set up to *training* and, also, 20% are used for validation. This datasets will be used in the fit function, the function that will train the model can be seen in Code Block 10 and one of the parameters is the validation data, this data is percentage of the training data will be used as validation data. The model will separate this portion of the training data, not train on it, and evaluate the loss and any model metrics on it at the end of each epoch.

```
model.fit(train_ds, validation_data=val_ds, epochs=epochs, callbacks=[callback])
```

Code 10: Generate training dataset

Another import parameters in the above Code Block is the callbacks, and is in this parameter that we will tell the model to stop training when a certain metric hits a minimum value within a range of epochs, and this callback can be seen in Code Block 11.

```
EarlyStopping(monitor='val_loss', patience=5, mode='min', restore_best_weights=True
    )
```

Code 11: Generate training dataset

Moving forwards, there are going to be three models that are going to be trained with three different numbers of images and after the metrics mentioned before will evaluate the performance of each model. Also, this models will be tested with a validation dataset, a dataset within the train dataset, to evaluate the loss and any model metrics at the end of each epoch, the model will not be trained on this data.

The first model will consist in three convolution blocks shown in Code Block 3 with an output layer shown in Code Block 4, as this is the standard model that Tensorflow suggests[26]. A full example can be seen in Code Block 12.

```
model = Sequential([
        layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(
    img_height, img_width, 3)),
        layers.Conv2D(16, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(32, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
```

Code 12: First Model Architecture

[26] https://www.tensorflow.org/tutorials/images/classification#compile_the_model

For a more visual concept, in Figure 31 shows the architecture of the fist model.

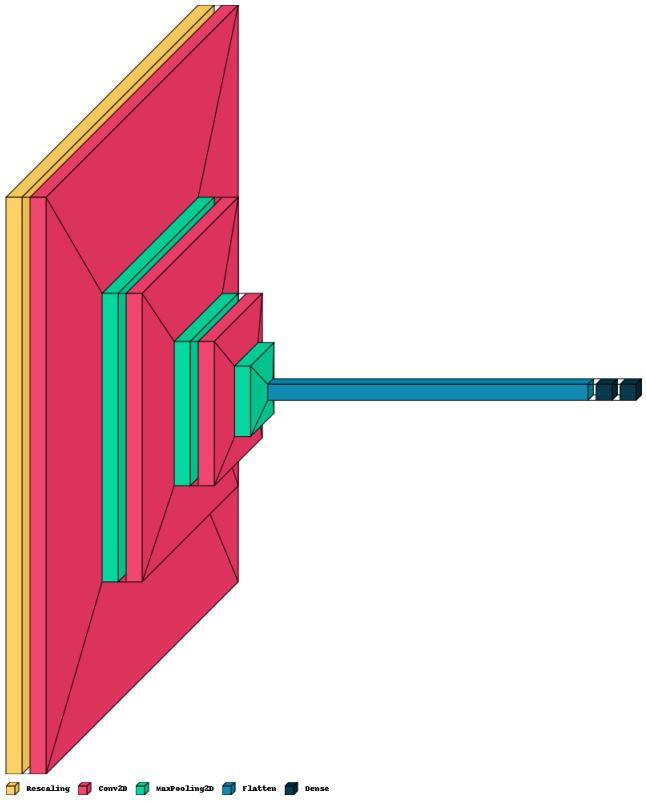


Figure 31: First Model Architecture

The second model will only add one more convolution block with 128 filters and, finally, Code Block 13 shows this last convolutional block. Moreover, in Figure 32 it is visible this same model architecture and the added convolution block.

```
layers.Conv2D(128, 3, padding='same', activation='relu'),
layers.MaxPooling2D()
```

Code 13: Second Model Last Convolution Block

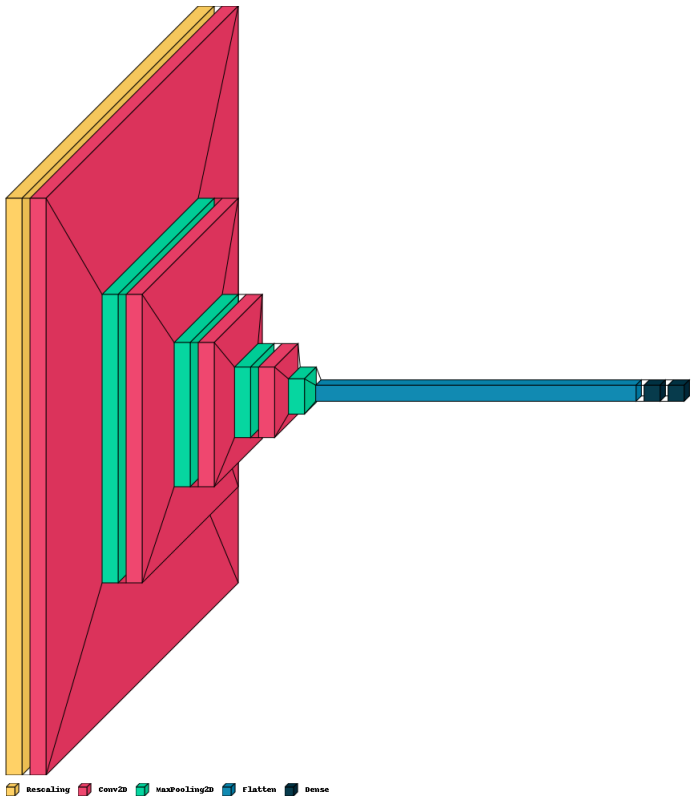


Figure 32: Second Model Architecture

The last model will have six total convolution blocks with the values 256 and 512 apart from the first model plus the convolution block with 128 filters and Code Block 14 shows the two last convolution blocks, since it is a sum of the two models. These models will run against three different environments and they differ only in the number of images, the first environment will have 30 images labelled as *mask* and 30 more labelled as *no mask*, and the second one will have 60 images for each label and the last one will have 90 images for each label.

```
layers.Conv2D(256, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(512, 3, padding='same', activation='relu'),
layers.MaxPooling2D()
```

Code 14: Third Model Last Convolutions Block

This third model is figuratively represented in Figure 33 and it is possible two see the convolution blocks referenced in Code Block 14.

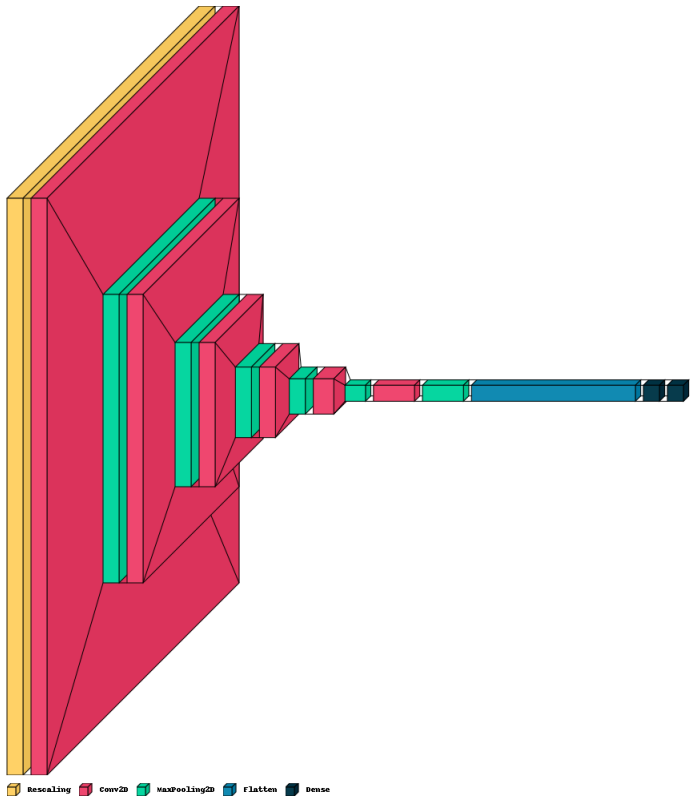


Figure 33: Third Model Architecture

Furthermore, this models will have three datasets to train in, the first dataset contains 30 images per class, meaning that the class *no_mask* has 30 images and the class *mask* has, also, 30 images, having a total of 60 images. The second dataset will add more 30 images in each class, making this dataset with the total size of 120 images and, finally, the third dataset will add more 30 images in each class, making each class have 90 images and a total of 180 images.

5 Results

5.1 Dataset of 30 images per class

5.1.1 First Model

As indicated previously, the first run was with 30 images for both classes, mask and no mask. Moreover, the metrics that are mentioned above are retrieved at this point of the development and the first couple of metrics can be visualised in Figure 34 and Figure 35 and the part of the Figures that require more attention are the orange lines because this indicates how the model will behave in the real world. Furthermore, looking at the loss, the desired result is to have the lowest values possible since loss indicates a penalty for a bad prediction and also, in this case, tells when the model should stop learning, thus the value that the validation loss demonstrates is acceptable.
Moving on to AUC, represented in Figure 35, it shows a very good performance against the validation dataset and by having very close values to the training dataset, shows that this model with the respective size can predict correctly for both classes.

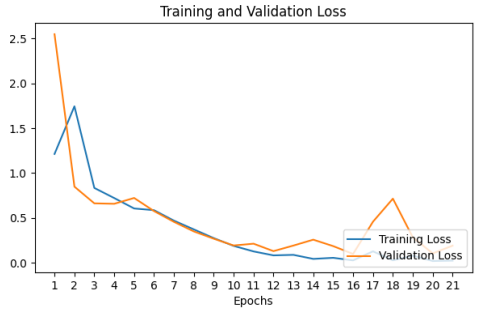


Figure 34: First Model Loss Values

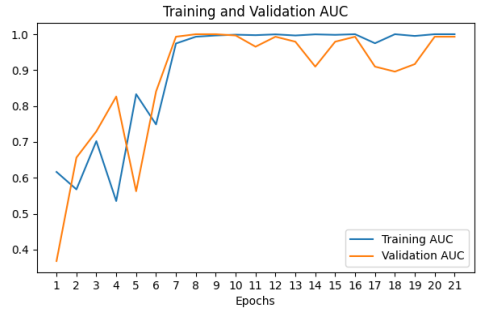


Figure 35: First Model AUC Values

Above was mentioned an out-of-the-box metric, loss and one that is not so much retrieved in machine learning problems. However, the metrics that are shown below are considered as standard in these types of problems having the names of precision, represented in Figure 36, and recall that is represented in Figure 37.
At a first glance, both Figures are equal, excluding the text differences. This may induce an error, however considering that the dataset has 30 images for each class, we can assume that the model classified an equal amount of images as false positives, as it classified false negatives and this conclusion can be achieved by saying that Formula 3 is equal to Formula 4, thus giving us the final answer of $FalsePositive = FalseNegative$. As said before, another reason is that the metrics provided by

Tensorflow count the total true positives, false negatives, and false positives to calculate metrics at a global level.

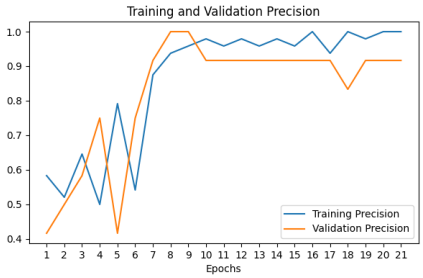


Figure 36: First Model Precision Values

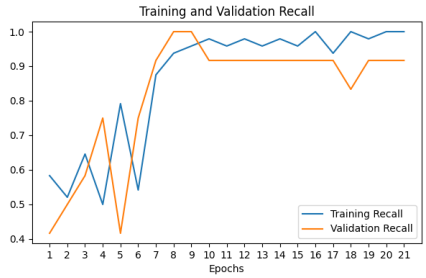


Figure 37: First Model Recall Values

Finishing the metrics for the first model, the F1 Score of the first model is represented in Figure 38 and it shows the same results of recall and precision because the F1 score gets its values from both precision and recall as observed in Figure 36 and Figure 37. Both metrics have high values, thus it only makes sense that the F1 score also, presents high values.

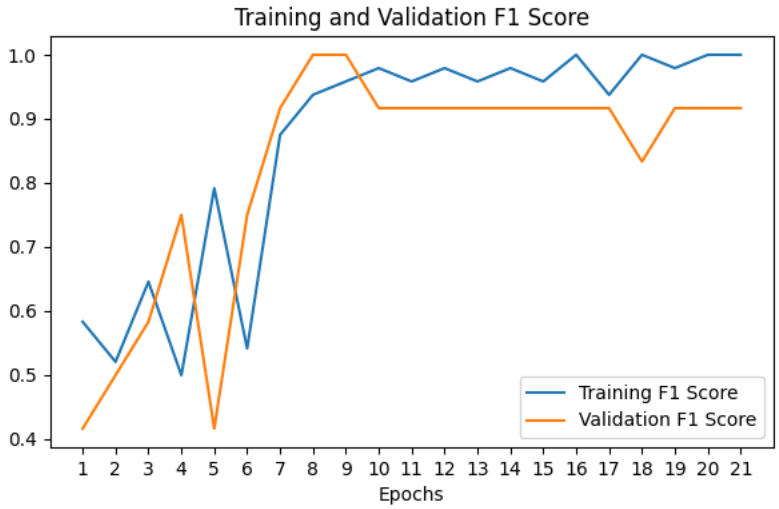


Figure 38: First Model F1 Score Values

5.1.2 Second Model

Moving along to the second model and remembering that this model is characterized by four convolution blocks, the first metrics to analyze are the loss and the AUC, represented in Figure 39 and Figure 40, respectively.

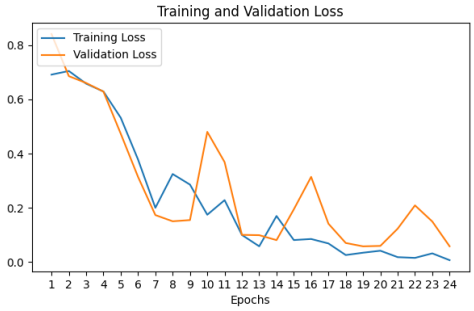


Figure 39: Second Model Loss Values

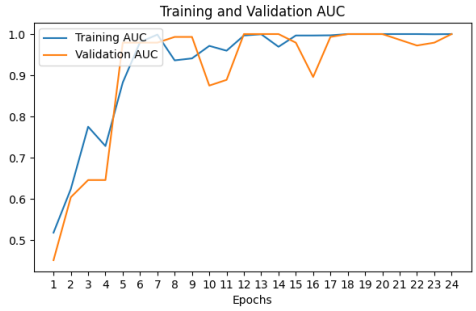


Figure 40: Second Model AUC Values

It may seem a bit odd to see these ups and downs on both of these images however, this fluctuation is visible in the rest of the metrics, and it will be explained after showing the rest of the metrics of the second model. Notwithstanding, these metrics show that the model can perform well in the real world, more precisely, the AUC metric that has the value of 1 in the last epoch. Furthermore, the same event demonstrated in Figure 37 and in Figure 36 also happens in this model, and for the second model the precision and recall are visible in Figure 41 and in Figure 42, respectively and presenting very good values, getting to the value of 1 in the last epoch.

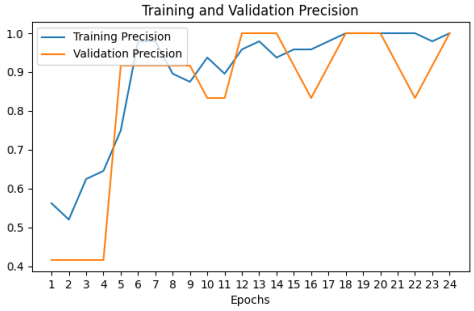


Figure 41: Second Model Precision Values

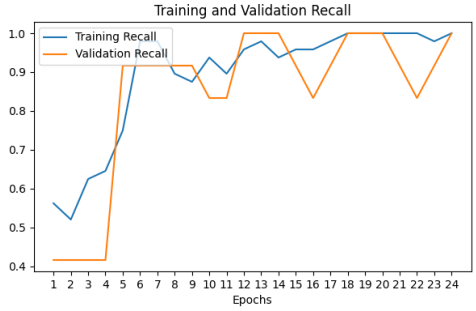


Figure 42: Second Model Recall Values

However, the trend of the metrics values on the validation dataset, this fluctuation may have some explanations and also some questions that may be answered in the larger dataset because a possible explanation for this event to happen is the small size of the dataset and the large size of the network, more precisely the number of filters with only 30 images per class, thus the model might end up just wandering around rather than locking down on a good value. Moreover, another reason that this might be happening could be be the usage of the *adam* optimiser, shown in Code Block 5, because in *adam*, the spikes are an inevitable result of Mini-Batch Gradient Descent and some mini-batches contain "by accident" unfortunate data for optimisation, causing the spikes visible in

Figure 41 and Figure 42.

F1 Score, represented in Figure 43 encapsulates what was said before about the parameter *micro* where all the values of this metric are calculated in a global level.

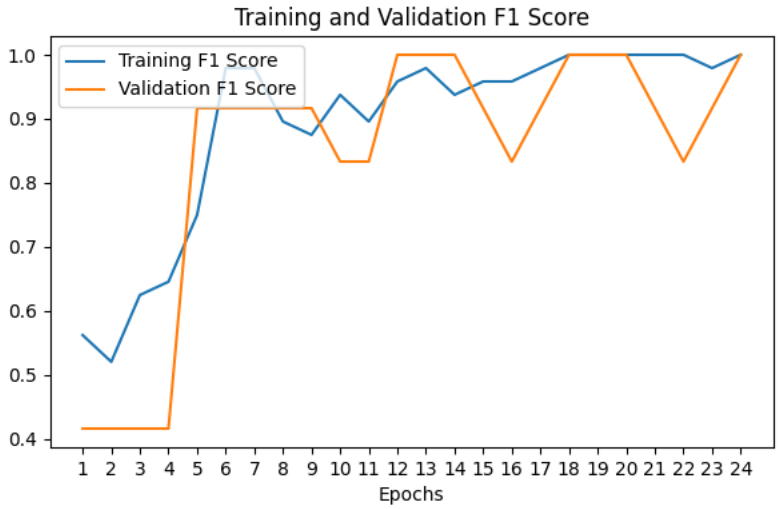


Figure 43: Second Model F1 Score Values

Also, due to this fluctuation, the callback has a hard time keeping track of the loss values, thus it takes more epochs to stop the model from training. Moreover, these items will be taken into account when analyzing the models using bigger datasets.

5.1.3 Third Model

The performance of the third model is not so far away as the performance of the second model. Firstly as shown in Figure 44 and Figure 45, the loss and the AUC have the expected trends, down and up, respectively, although the AUC does not reach the value of 1 like the second model and the validation loss has higher value in this model than from the second model.

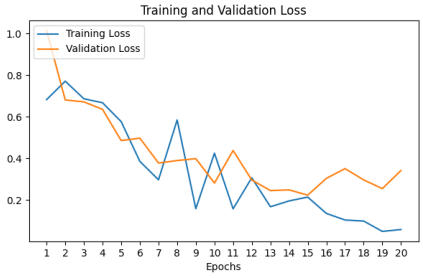


Figure 44: Third Model Loss Values

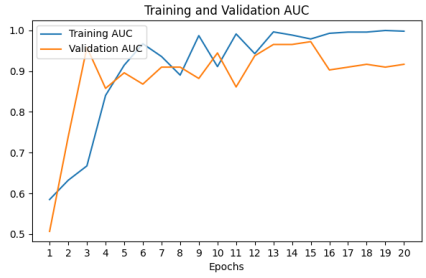


Figure 45: Third Model AUC Values

Moving forward to precision and recall, the same situation is visible here, with both the spikes and the same results from both the precision and recall.

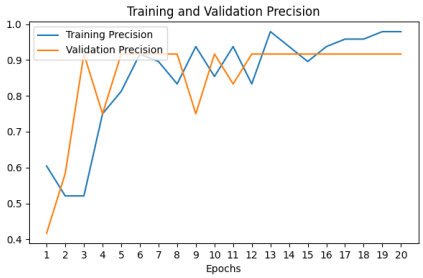


Figure 46: Third Model Precision Values

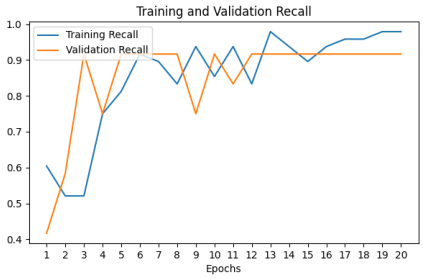


Figure 47: Third Model Recall Values

The same situation that happened in the second model also happened in this one, the F1 score, shown in Figure 48 has equal values for precision and recall.

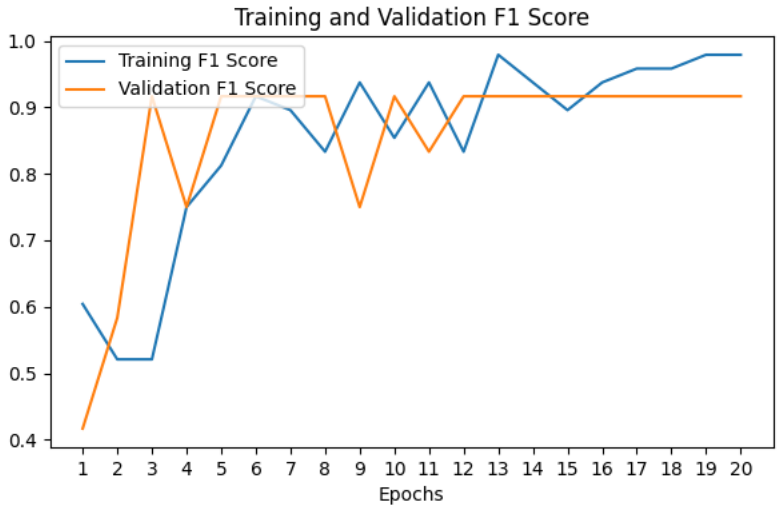


Figure 48: Third Model F1 Score Values

To finalise the analysis of the dataset with 30 images for each class, the model that achieves better results, as metrics, was the second model, starting with the loss that within the three models the second one has the lowest value and finishing in the AUC that in the second model this metrics is equal to one. An overview of the three models is represented in Table 6. Also, remember that this dataset only has 12 images for validation actions, thus, it may not be enough for a concrete analysis of the validation graph values.

	Metrics					
Model	Epochs	Loss	Precision	Recall / Accuracy	F1 Score	AUC
1	21	0.1913	0.9167	0.9167	0.9167	0.9931
2	24	0.0582	1	1	1	1
3	20	0.3417	0.9167	0.9167	0.9167	0.9167

Table 6: First Dataset Metrics results overview

5.2 Dataset of 60 images per class

5.2.1 First Model

Within the evaluation from the first model with a larger dataset, of 60 images per class, there are key topics that can be observed when looking at the loss, in Figure 51, and at AUC, in Figure 50, is that this model had smoother trends, the validation loss getting lower values than the homologous model with the first dataset and the validation AUC stayed at 1 for most of the testing phase.

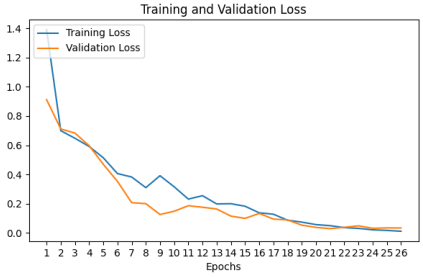


Figure 49: First Model Precision Values

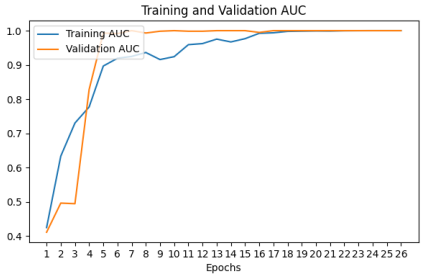


Figure 50: First Model Recall Values

Moving to the precision and recall, since this new dataset is composed of 60 images per class, the same situation from the first model, where they have the same value, also happened in this model. Moreover, in Figure 51 and in Figure 52, it is also visible that the validation metrics achieved the maximum value possible in the last epochs.

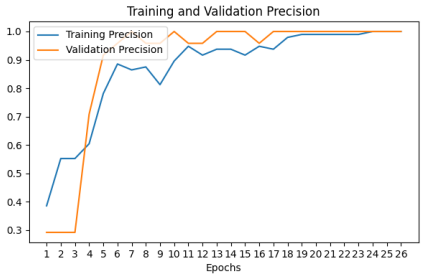


Figure 51: First Model Precision Values

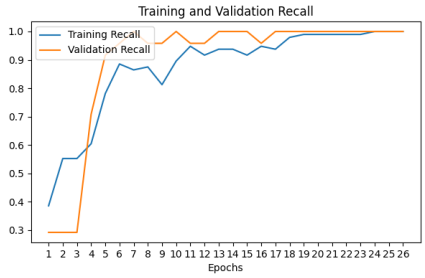


Figure 52: First Model Recall Values

Notwithstanding, the F1 score, shown in Figure 53, shows the same results from precision and recall. Furthermore, this first model also takes more time, since it consists of more images both for training and for validation.

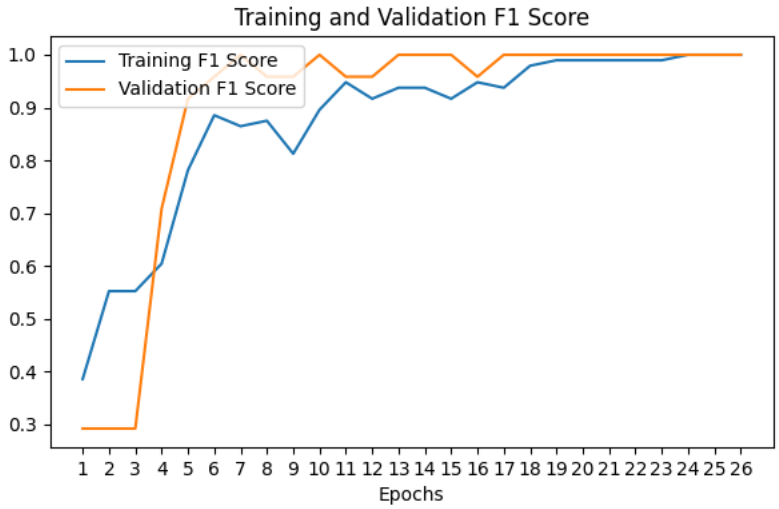


Figure 53: First Model F1 Score Values

5.2.2 Second Model

This model had better results than the first one, to remember that this model has 4 convolution blocks, and with more blocks the images are more processed to obtain a prediction, meaning that more key features are identified. Figure 54 and Figure 55 demonstrate that this model has similar behaviour to the first one, also this model took fewer epochs to achieve the lowest validation loss.

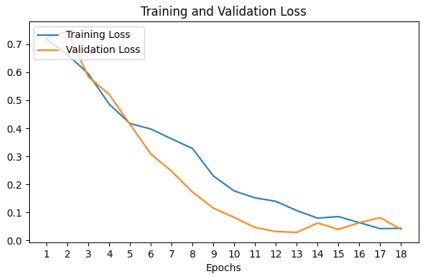


Figure 54: Second Model Loss Values

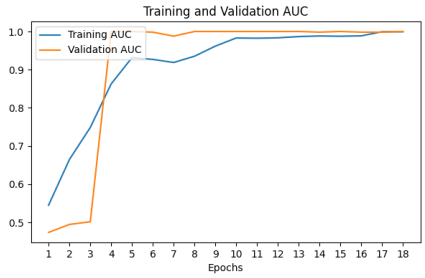


Figure 55: Second Model AUC Values

Moving now to the precision and recall, this model achieves impressive results on the validation dataset, maintaining for most of the training phase with values close to 1, this can be verified in Figure 56 and Figure 57.

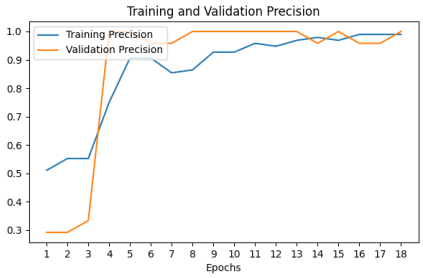


Figure 56: Second Model Precision Values

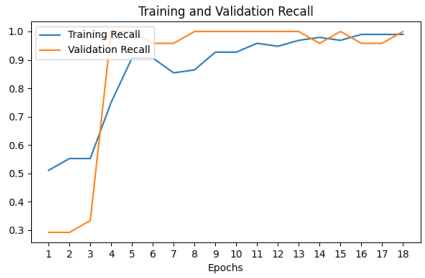


Figure 57: Second Model Recall Values

As expected, the situation where the F1 score is equal to both precision and recall still happened in this model, as seen in Figure 58.

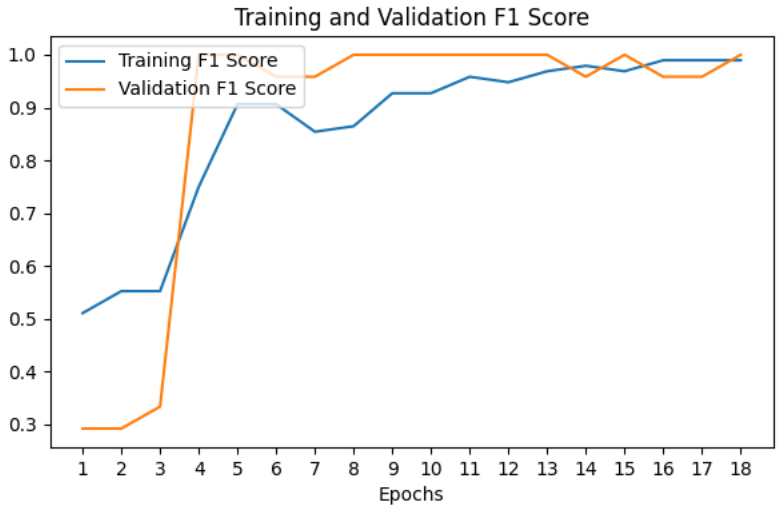


Figure 58: Second Model F1 Score Values

5.2.3 Third Model

Starting with the third model's loss, it is visible the downward trend, even with slights of fluctuation, as seen in Figure 59, also, it is visible that the validation phase got better results than the training phase. Moreover, the AUC, in Figure 60 also presents good values and an upwards trend, even reaching 1 in the validation dataset.

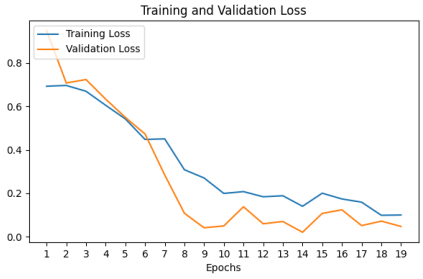


Figure 59: Third Model Loss Values

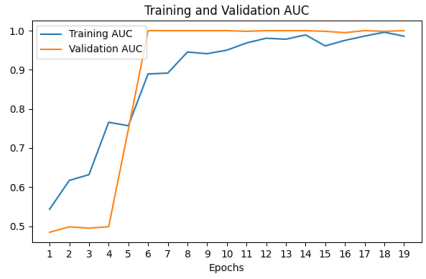


Figure 60: Third Model AUC Values

Moving on to precision and recall, it is observable that this model had good behaviour, because the values, even though they present the same value, the upwards trends start very early in training, considering the total number of epochs, the model achieved a stable value of both precision, represented in Figure 61, and recall in Figure 62.

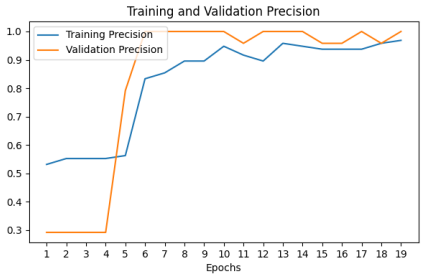


Figure 61: Third Model Precision Values

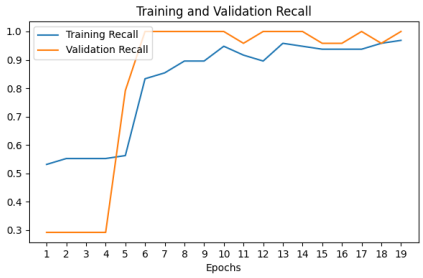


Figure 62: Third Model Recall Values

Finishing this model metrics with the F1 score, the values from this metrics are also equal to precision and recall, nonetheless, it also shows good values, represented in Figure 63.

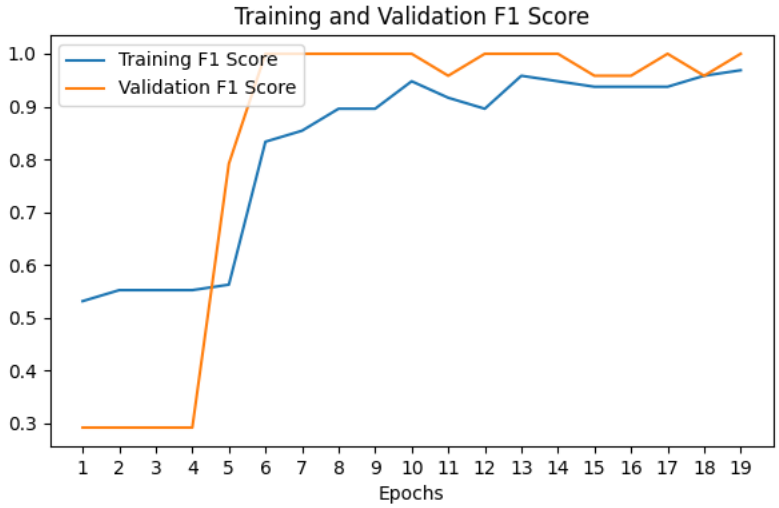


Figure 63: Third Model F1 Score Values

Concluding and looking at Table 7, this dataset achieved good results within a relatively small amount of epochs, although the first model presents the lowest number in terms of validation loss, the second model took fewer epochs, meaning less time, and the validation loss value is not so far away as in the value in the first model, making the second model the best one within the 60 images per class dataset.

	Metrics					
Model	Epochs	Loss	Precision	Recall / Accuracy	F1 Score	AUC
1	26	0.0330	1	1	1	1
2	18	0.0401	1	1	1	1
3	19	0.0477	1	1	1	1

Table 7: Second Dataset Metrics results overview

5.3 Dataset of 90 images per class

5.3.1 First Model

As expected, with this dataset the first model took a bit more time to train and run validation, because there are more batches of images. Furthermore, this model behaved well in terms of loss, represented in Figure 64, where the validation loss followed the training loss very closely, and this pattern is the desired one since it brings more confidence to the prediction. To make this argument strong the AUC values, represented in Figure 65, are pretty high since, roughly, the beginning of the training phase and staying at the maximum value for the rest of the training. This value indicates that this model with this dataset brings good confidence levels to deploy it to the real world, so let's analyze the rest of the metrics to confirm this argument.

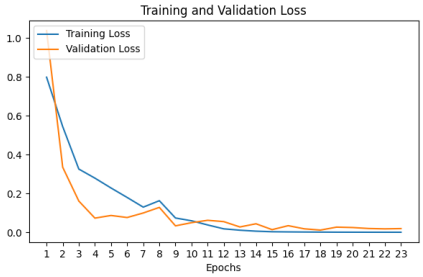


Figure 64: First Model Loss Values

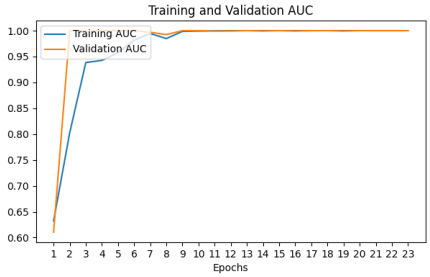


Figure 65: First Model AUC Values

Following up with precision and recall, these two metrics, as expected, got the same values. Notwithstanding, these values were also very good for a model with only 3 convolution blocks. As visible in both Figure 66 and Figure 67 shows the values that this model got and roughly from the start, got the maximum value. This is also applicable and visible in the F1 Score, represented in

Figure 68.

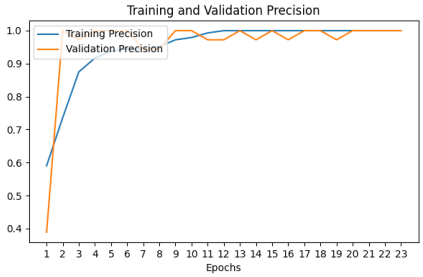


Figure 66: First Model Precision Values

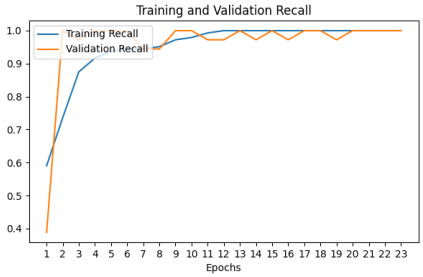


Figure 67: First Model Recall Values

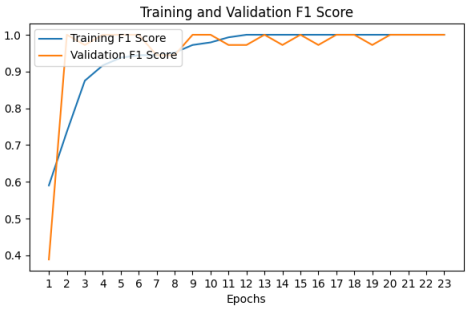


Figure 68: First Model F1 Score Values

5.3.2 Second Model

What differentiates this model from the first one is its time and this model took more epochs to achieve the callback, since it has more convolution blocks. Going back to the metrics, the loss, in Figure 69, more specifically, the validation loss followed the training loss with sometimes having lower values. Following up with AUC, in Figure 70, this metric also achieves outstanding values, maintaining for most of the training the maximum volume.

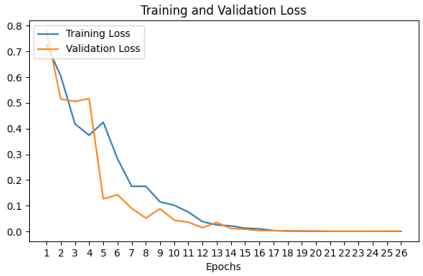


Figure 69: Second Model Loss Values

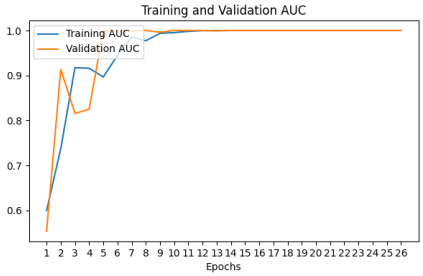


Figure 70: Second Model AUC Values

Next up comes the precision and recall, not ignoring the fact that this model also had the same values for precision and recall, this model showed very good results in these two metrics, represented in Figure 71 and Figure 72, by, again, the maximum value for, about, half of the training phase.

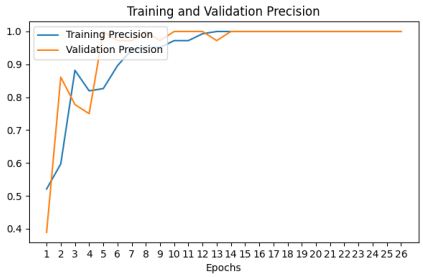


Figure 71: Second Model Precision Values

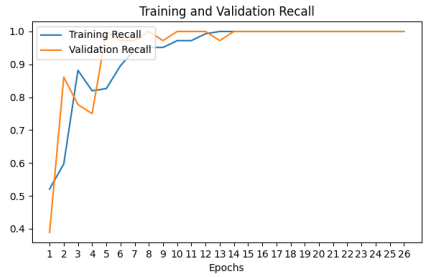


Figure 72: Second Model Recall Values

Notwithstanding what was said above about precision and recall, the F1 score, as expected returned the same values as the two metrics mentioned above.

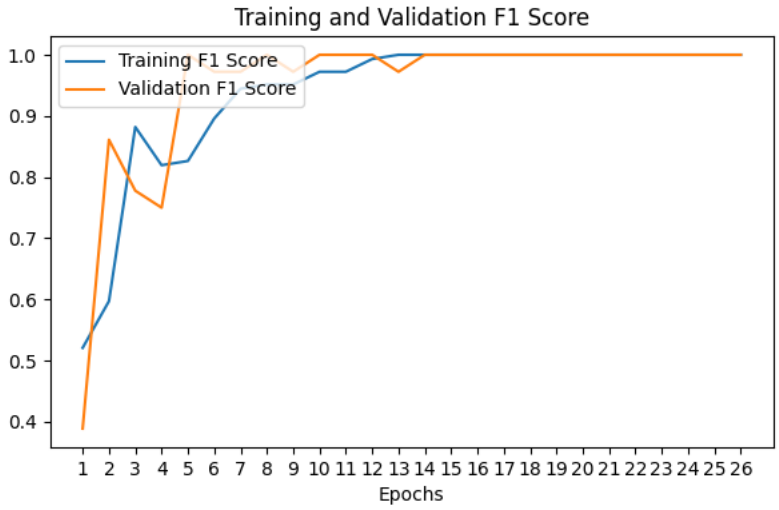


Figure 73: First Model F1 Score Values

5.3.3 Third Model

This model did not differ too much from the second one, just less than 2 epochs for training. The loss, in Figure 74 performed well by having lower values along the training phase and the AUC, in Figure 75, presented good values, making this model also trustworthy.

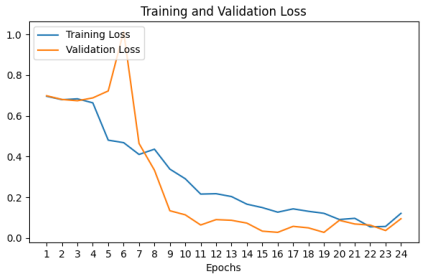


Figure 74: Third Model Loss Values

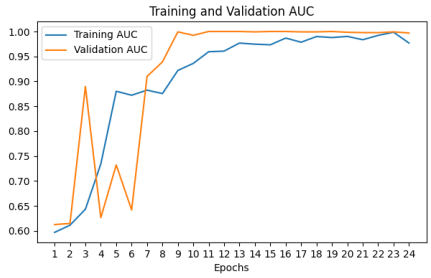


Figure 75: Third Model AUC Values

Moving on, both precision and recall, in Figure 76 and Figure 77 respectively, also presented good values apart from the situation where they have the same value. Also, the F1 score, represented in Figure 78, has the same value for both precision and recall, although it is still a really good value.

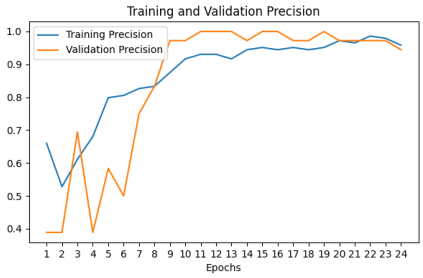


Figure 76: Third Model Precision Values

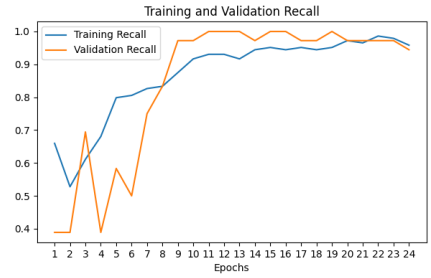


Figure 77: Third Model Recall Values

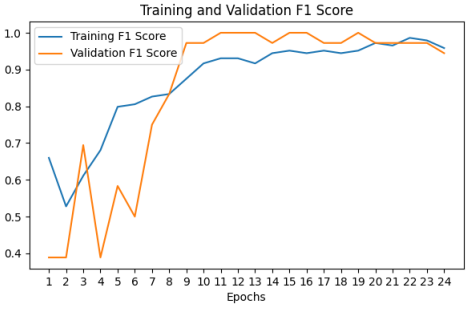


Figure 78: First Model F1 Score Values

Furthermore, in Table 8, it is possible to see a summary of the metrics values along with the number of epochs it took to reach the same values.

	Metrics					
Model	Epochs	Loss	Precision	Recall / Accuracy	F1 Score	AUC
1	23	0.0192	1	1	1	1
2	26	0.0011	1	1	1	1
3	24	0.0949	0.944	0.944	0.944	0.9969

Table 8: Third Model's Metrics Summary

5.4 Models performance recap

The first dataset had better results with the first model, and by saying better it is equal to saying that the model did not present an unexpected behaviour like the in the second model that demonstrated some fluctuation that can be seen in Figure 39, and the third model presented the highest loss value within the three models, it even couldn't get lower values than the train dataset.

Moreover, the second dataset achieved better results with the second model, because both precision and recall got the maximum value. However, the first model took more epochs and the difference in the loss metric is not substantial which makes this model the better one since the second one obtained proximal results in fewer epochs.

Finishing off, the third dataset worked well on all the models, and this can be due to the fact o the size of the dataset. In this situation a factor that can be important when choosing a model to deploy to the real world is the number of convolution blocks, meaning that with more convolution blocks the model can identify more key features that can help him make a better prediction, however as a trade-off, the model will take longer, where the models took at least 20 epochs to train.

5.5 Explaining an image

After training the model, LIME will enter in the scene, and by using the weights generated by the model it will convert into an image classification model accepted by LIME, in this case, InceptionV3 which is commonly used for this cases and, finally, proceed to draw the areas where the model considered a mask or not a mask.

Technically, after training the model the weights generated by the model are saved, in this case, saved locally, afterwards, these same weights are loaded to an InceptionV3 model, specifying the activation

function and the number of classes, that is two. This process can be seen in Code Block 15.

```
model2.save_weights("ckpt.h5")

v3 = tf.keras.applications.inception_v3.InceptionV3(include_top=True, weights=
    model2.load_weights("ckpt.h5"),input_tensor=None,
classifier_activation='softmax',lasses=2)
```

Code 15: Conversion to InceptionV3

Afterwards, it's just using the LIME's API to give us the mask over the image we desire to explain, although before this happens we need some boilerplate code and the main code is to transform the image to a format the model can understand, more specifically an array, then the process explain just above is represented in Code Block 16.

```
explainer = lime_image.LimeImageExplainer()
explanation = explainer.explain_instance(images[0].astype('double'), v3.predict,
    top_labels=2, hide_color=0, num_samples=1000)

temp, mask = explanation.get_image_and_mask(explanation.top_labels[0],
    positive_only=False, num_features=10, hide_rest=False)
```

Code 16: Usage of LIME

Finishing off, it just generates the image within the boundaries of what the model considered as a mask, to note that the dataset consists of images of masks on people from different angles so the model sometimes might consider just some angles of the mask.

```
plt.axis("off")
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask))
plt.title("Explanation for predicted class: " + str(class_predicted))
plt.show()
```

Code 17: Generating explanation image

In Code Block 17 it is represented the code necessary to generate the image, with attention to line 3 the function *mark_boundaries* is used and the integers inside are values to set the colour of the boundaries.

The final result can be seen in Figure 79 and Figure 80 for explanations of people without masks and people with masks, respectively. In Figure 79 it can be said that the model, to determine if the person is without a mask, "looks" for the upper part of the face, presented in the second and third

images within the Figure. On the other hand, to determine if the person is with a mask the model "looks" at the part where it is supposed to be a mask, although, an interesting insight is that the model's "vision" is towards the side of the mask, this is visible in all of the images in Figure 80 with more precision in the second image.

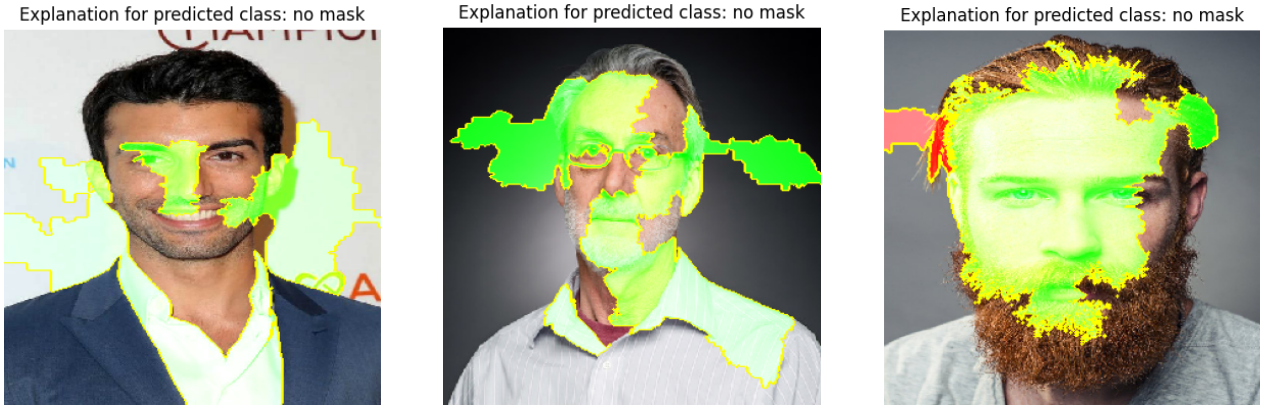


Figure 79: Model explanation for no mask images

Figure 80: Model Explanation for no mask

6 Conclusion

The real world is unpredictable, so are the conclusions one might take after a model's prediction is out if the reason for that prediction is not known. Providing the output with an explanation may reduce the persons' doubts and help them take better decision with less unknown impacts. Moreover, within the fast pace of the real world, Machine Learning models must be able to adapt to edge cases that can emerge from nowhere, thus compromising the model's accuracy and precision. To ensure that this does don't happen very often, a trade-off must be taken into account. That is, the dataset size versus the time a model takes to learn and show good metrics.

The work done here suggest a way to avoid the trade-off mentioned above. Firstly to avoid larger datasets it was proposed the active learning approach, where some images were labelled to build three datasets.

Secondly, to avoid the model's time to train three models with different hyperparameters and number of convolution blocks were created, and one of the parameters that was equal to all models was the early stopping, a parameter that indicates a model should stop training when a threshold of values is met.

Finally, a set of the model's metrics were extracted and put to evaluation and from all the models and datasets was possible to conclude that the model with the most convolution blocks with the bigger dataset achieved better results with an acceptable number of epochs.

Moreover, in the dataset, it is important to have a variety of cases that can happen in the real world. However, if we desire to have the most recent good model out in the world, it would be helpful that these cases are present in the dataset, images that may take time to acquire due to the unknown cases that might appear.

Furthermore, when evaluating a model's performance against the validation dataset, it is important to, on the one hand, analyse the relevant metrics and, on the other hand, the evolution of these values along the validation process. And, by graphing these values, it is possible to see the trend lines and inspect for fluctuation that can, eventually, output a bad prediction. Moreover this referred fluctuations where noted in the dataset with a smaller number of images, while in the bigger datasets the trend lines were smoother and with less to no fluctuations.

6.1 Limitations

As the labelling process takes time to do, the dataset size gets compromised since the focus of this project is to develop several models and dataset and evaluate the metrics. However, having bigger datasets would give us more insights about the models. Furthermore the models ran on a simple Macbook with default features. Better results could be obtained if the models ran against a server with parameters optimised for Machine Learning models.

6.2 Future Work

As the models need data to train there will be a focus to get more images for the human to label. Moreover, to get an easier usage of all the models developed and the models that will be developed, an API will be developed to manage all the models and respective metrics, along with the endpoint to get a prediction with an explanation. Furthermore, it will be developed a way to automate the way the human selects images to label and then store the images already cropped.

References

[1] I. Chien, A. Enrique, J. Palacios, T. Regan, D. Keegan, D. Carter, S. Tschiatschek, A. Nori, A. Thieme, D. Richards, G. Doherty, and D. Belgrave, "A Machine Learning Approach to Understanding Patterns of Engagement With Internet-Delivered Mental Health Interventions," *JAMA Network Open*, vol. 3, no. 7, pp. e2 010 791–e2 010 791, 07 2020. [Online]. Available: https://doi.org/10.1001/jamanetworkopen.2020.10791

[2] D. Shen, J.-D. Ruvini, and B. Sarwar, "Large-scale item categorization for e-commerce," 10 2012, pp. 595–604.

[3] L. V. Lapão, M. M. Da Silva, and J. Gregório, "Implementing an online pharmaceutical service using design science research," *BMC medical informatics and decision making*, vol. 17, no. 1, pp. 1–14, 2017.

[4] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[5] J. Grenha Teixeira, L. Patrício, K.-H. Huang, R. P. Fisk, L. Nóbrega, and L. Constantine, "The minds method: integrating management and interaction design perspectives for service design," *Journal of Service Research*, vol. 20, no. 3, pp. 240–258, 2017.

[6] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[7] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, """ why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[9] ——, "Model-agnostic interpretability of machine learning," *arXiv preprint arXiv:1606.05386*, 2016.

[10] F. Bre, J. Gimenez, and V. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, 11 2017.

[11] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[12] M. V. Valueva, N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[13] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[14] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud, and J.-H. Kim, "An automated system to limit covid-19 using facial mask detection in smart city network," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, 2020, pp. 1–5.

[16] M. S. Islam, E. H. Moon, M. A. Shaikat, and M. J. Alam, "A novel approach to detect face mask using cnn," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*. IEEE, 2020, pp. 800–806.

[17] H. Farman, T. Khan, Z. Khan, S. Habib, M. Islam, and A. Ammar, "Real-time face mask detection to ensure covid-19 precautionary measures in the developing countries," *Applied Sciences*, vol. 12, no. 8, p. 3879, 2022.

[18] S. Shen, H. Jiang, and T. Zhang, "Stock market forecasting using machine learning algorithms," *Department of Electrical Engineering, Stanford University, Stanford, CA*, pp. 1–5, 2012.

[19] A. L. Beam and I. S. Kohane, "Big data and machine learning in health care," *Jama*, vol. 319, no. 13, pp. 1317–1318, 2018.

[20] P. Voigt and A. v. d. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Springer Publishing Company, Incorporated, 2017.

[21] B. Ustun, A. Spangher, and Y. Liu, "Actionable recourse in linear classification," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019, pp. 10–19.

[22] N. Siddiqi, *Credit risk scorecards: developing and implementing intelligent credit scoring.* John Wiley & Sons, 2012, vol. 3.

[23] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

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

[25] A. Kassambara, *Machine learning essentials: Practical guide in R.* Sthda, 2018.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] D. Ramos, D. Carneiro, and P. Novais, "evorf: an evolutionary approach to random forests," in *International Symposium on Intelligent and Distributed Computing.* Springer, 2019, pp. 102–107.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[29] J. Liu, J. Chen, and J. Ye, "Large-scale sparse logistic regression," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 547–556.

[30] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[31] S. M. Zoldi, L. Peranich, J. Athwal, U. Mayer, and Sajama, "Adaptive fraud detection," May 2020.

[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.