DEEP LEARNING FOR THE DETECTION OF HOUSE MOULD

Abstract

In this paper, I researched, developed and deployed a machine learning solution able to detect the presence of mould in photographs. This will aid housing association, who in the last years have had more disrepair cases relating to damp and mould.

Dean Bonsu

[Email address]

Contents

1. INTRODUCTION

1.1. Introduction Chapter

Damp and mould is an issue affecting millions of properties worldwide. In the UK, in recent years, the risk of health problems caused by damp and mould has been brought to mainstream media attention following the death of a young child (BBC, 2022). As of the 2022 amount of private registered landlords and local authorities home in UK, amounts to 4.4m (Regulator of Social Housing, n.d.), with major proportion of residents being vulnerable or part of marginalised communities.

These organisations manage numerous properties, which are surveyed, serviced, and maintained at regular intervals. During these surveys and visits, photographic evidence is collected (Annila et al., 2018). These images are often stored and only reviewed when the needs arise. This can lead to defects and faults not being assessed and remediated in a timely manner.

This project is intended to develop a machine learning solution, that can assist these organizations in classifying the images according to the defect present.

1.2. Project Aims and Objectives

During the first meeting with my supervisor, the main aims and objectives of the project were discussed and agreed upon.

The main objectives:

- Propose a Convolutional Neural Network solution able to classify images that contain mould with at least 95% accuracy.
- Evaluate the performance of the model.

Optional Objectives:

- Demonstrate a realistic application integration with the model developed.

1.3. Client

The client is Housing Inc, a social housing organization based in the UK, who manages a stock of about 40,000 properties throughout the United Kingdom.

Currently photos taken during surveying and maintenance visits must be manually sorted and inspected. These will need to be reviewed to confirm the presence and location of any disrepair issues such as damp and mould.

They are looking at solutions to automate this process, to allow a faster proactive approach to resolving damp and mould issues.

1.4. Resources, Limitation and Ethical Issues

Resources

To complete this project, Python with PyCharm IDE were used. The PC used also has an NVIDIA graphical processing unit, which allowed for local running of model.

Limitations

Main limitations to the project are the availability of good quality photographs and project time.

Ethical Issues

The dataset used consist of publicly available images retrieved through google search engines.

Due to this no ethical concerns are identified for this project.

1.5. Methodology

Two different methodologies were considered for this project. The KDDM and CRISP-MD methods.

KDD

The KDD (Knowledge Discovery in Databases) method is a process used in data mining to extract useful knowledge from large data sets. It involves several stages including data selection, pre-processing, transformation, data mining, evaluation, and interpretation.

The data selection stage involves identifying the relevant data from the available data sources. The pre-processing stage involves cleaning the data and removing any irrelevant or redundant information. The transformation stage involves converting the data into a format suitable for analysis.

The data mining stage is the heart of the KDD process, where various techniques such as clustering, classification, association rule mining, and anomaly detection are applied to the data. The evaluation stage involves assessing the quality of the results obtained from the data mining process.

Finally, the interpretation stage involves interpreting the results obtained from the data mining process and drawing conclusions from them. The KDD method is widely used in various fields, including business, healthcare, finance, and social sciences, to extract useful insights from large data sets.

CRISP-MD

CRISP-DM (Cross-Industry Standard Process for Data Mining) is a well-known process model for data mining. It is a flexible and iterative model that consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment.

The Business Understanding phase involves understanding the business objectives and requirements for the data mining project. This includes identifying the problem to be solved, defining the scope of the project, and determining the success criteria.

The Data Understanding phase involves understanding the data that will be used in the project. This includes collecting and exploring the data, identifying any data quality issues, and establishing an initial data preparation plan.

The Data Preparation phase involves preparing the data for modelling. This includes selecting the relevant data, cleaning the data, transforming the data, and creating derived variables.

The Modelling phase involves building and testing the models. This includes selecting the appropriate modelling technique, creating the model, and assessing the model's performance.

The Evaluation phase involves evaluating the results of the model and determining if the model meets the business objectives and requirements.

Finally, the Deployment phase involves deploying the model into the production environment and monitoring its performance.

Overall, CRISP-DM provides a structured and systematic approach to data mining that can be applied across different industries and types of data.

For this purpose, the CRISP-DM methodology is applied to this research.

1.6. Structure of Dissertation

For the purposes of this research the CRISP-DM framework is slightly amended to fit the structure of this project.

- Literature Review
- Data Understanding & Preparation
- Methodology
- Modelling
- Evaluation
- Deployment
- Conclusion

2. Literature Review and background

Dampness and mould are a common problem in buildings around the world. The presence of dampness and mould can have a profound impact on both the health of the occupants and the structural integrity of the building.

In the recent years, this has become a topic of discussion in the UK. In 2020, mould in a property were ruled as the cause of death of a two-year-old child. (BBC, 2022).

The purpose of this literature review is to examine the effects of mould on human health, including its causes, symptoms, and potential preventive measures.

2.1. What is Mould?

Mould is a type of fungus that reproduces through the production of spores. These spores can be found in the air and can easily be inhaled, causing various health problems. Mould can grow in any environment where there is moisture, including bathrooms, kitchens, and basements. It can also grow on any surface, including walls, floors, and ceilings.

According to Peat et al., 1998, damp and mouldy homes can have negative health effects on children and post hoc repairs can be expensive and ineffective in the long term. It is suggested that properties should be built for primary prevention of respiratory problems in the first place. Mould and fungi are present everywhere in the environment; however, inhalation of sufficient spores can trigger symptoms of asthma, rhinitis, and bronchitis (Du et al., 2021).

Mould can be classified into three main categories based on their potential for causing health effects. Allergenic moulds are the most common type and can cause allergic reactions such as hay fever and asthma attacks. Pathogenic moulds can cause infections in people with weakened immune systems. Toxigenic moulds can produce harmful toxins that can cause serious health effects, including cancer (Silveira et al., 2019).

2.2. Effects of Mould on Health

The effects of mould on human health can vary depending on the individual and the extent of exposure. Damp and mould can disrupt the self-cleaning airway, and cause inflammation and damage to cells, such as red blood cells (Piecková, 2012). It can also trigger symptoms of asthma, rhinitis, and bronchitis, as well as worsen the psychological well-being of the building occupants, with a direct knock-on effect on factors such as productivity at work (Brambilla & Sangiorgio, 2020).

Studies have shown that exposure to dampness and mould in the home raises the risk for asthma development, ever diagnosed asthma, and current asthma by about 30–50% (Cox-Ganser, 2015)(Cox-Ganser, 2015). It is also associated with respiratory infections, eczema, and bronchitis. Indoor mould exposure is a risk factor for childhood allergies and airway diseases, especially for lifetime-ever asthma, wheeze, allergic rhinitis, and current wheeze (Cai et al., 2020; Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

The article by BBC (2022) reports on the findings of the Regulator of Social Housing (RSH) regarding the case of Awaab Ishak, a two-year-old boy who died from a respiratory condition caused by exposure to mould in his flat.

His family had repeatedly complained about the mould but the housing provider, Rochdale Boroughwide Housing (RBH), did not take any action until his death. The Regulator of Social Housing

(RSH) found that RBH had failed to treat the family with fairness and respect and had breached standards by waiting nearly two years to check for damp and mould in other homes on the estate.

The regulator also found that:

- RBH had failed to “treat Awaab Ishak’s family with fairness and respect”.
- RBH had failed to act quickly and protect more tenants from potential harm.
- RBH had waited nearly two years after Awaab’s death to check for damp and mould in other homes on the estate.
- RBH had given the regulator inadequate information shortly after Awaab Ishak’s death.
- RBH had weaknesses in its IT and internal communications, which led to vital information and risks being missed.

Sufficient evidence is found between the exposure to damp and respiratory health, more specifically epidemiologic studies found enough evidence to conclude that mould presence indoor has an association with symptoms such as cough, wheeze and hypersensitivity pneumonitis (Damp Indoor Spaces and Health, Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

The presence of mould also indicates the presence of mites and actinobacteria, with the latter preferring more humid environments compared to fungi. Additionally, mould can produce mycotoxins under the appropriate conditions. Sufficient evidence is found between the exposure to damp and respiratory health, more specifically, epidemiologic studies found enough evidence to conclude that mould presence indoor has an association with symptoms such as cough, wheeze, and hypersensitivity pneumonitis (Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

A range of studies conducted between 2004 and 2013 have also shown that mould is observed to increase or lead to the onset of asthma symptoms, respiratory infections, eczema, and bronchitis (Indoor dampness and mould health effects – ongoing questions on microbial exposures and allergic versus nonallergic mechanisms- PMC, 2015). WHO guidelines for indoor air quality indicate that indoor dampness is an issue in most countries, estimated to be between 10- 50%, with higher levels in deprived neighbourhoods, exceeding the national average (Heseltine et al., 2009). High humidity and water damage are factors that increase the incidence of damp and mould, therefore resulting in an increased exposure to mite and fungi. Water damage is harmful not only for the residents but also a risk to the building structure, such as rotting wood (Heseltine et al., 2009).

The best way to control moisture levels is through moisture control, which can be achieved by containing liquid water leaks, as well as ventilation and condensation levels. Ventilation can assist in mitigating health effects as it can provide enough airflow to remove indoor-generated pollutants and moisture (Indoor dampness and mould health effects – ongoing questions on microbial exposures and allergic versus nonallergic mechanisms- PMC, 2015)

Most of the resident most exposed to mould and damp, had a propensity to be identified as belonging in a lower socio- economic background. 24 % of unemployed people, resided in a property with damp and mould. Another study shows that household with the lowest incomes are the ones most affected by humidity and damp in their properties(Ginestet et al., 2020).

Mould can also have an impact on mental health. According to Brambilla and Sangiorgio (2020), exposure to mould can lead to cognitive impairment, depression, and anxiety. Furthermore, it can worsen the symptoms of pre-existing mental health conditions.

However, it is important to note that many of the studies cited here have limitations. For example, many of the studies rely on qualitative measures rather than specific exposure measures, making it difficult to draw specific conclusions about the relationship between mould and health. Additionally, publication bias may play a role, as studies that conclude that mould has negative effects may be more likely to be published.

2.3. Building Structures Dampness and Mould

Dampness and mould can also have a profound impact on the structural integrity of a building. Mould can produce mycotoxins under the appropriate conditions. Damp spaces can also facilitate the growth of bacteria that can have inflammatory effects (Damp Indoor Spaces and Health, Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

In a study in France visible mould is estimated to in up to 20% percent of dwellings, this due to high number of old buildings present.

Aside from harming the resident damp and mould also have a financial cost associated with them, in France the cost of repairing buildings damaged by mould is estimated to be around 5 billion euros in 2013, with an additional half a billion spent in the health care of mould caused health problems. (Ginestet et al., 2020)

Studies analysed different construction materials looking for patterns of contamination, formation, and growth of fungi, concluded that the surface properties, such as substrate composition, porosity, pollution on the surface, Ph, have a significant influence on the initial factors for mould fixation and growth rate. (Silveira et al., 2019). Clay blocks were more quickly covered by fungi, due to the greater presence of water on the surface of the clay block, which provided a higher amount of nutrients for fungal development. (Silveira et al., 2019)

It is recommended that indoor environments at risk should be assessed and monitored to ensure that mould is treated as soon as possible. Building should be designed, operated, and maintained to prevent water intrusion and excessive moisture accumulation when possible (Damp Indoor Spaces and Health, Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

The most effective way to manage microbial contaminants, such as mould, that are the result of damp indoor environments is to eliminate or limit the conditions that foster its establishment and growth (Damp Indoor Spaces and Health, Institute of Medicine (US) Committee on Damp Indoor Spaces and Health, 2004).

According to Ginestet et al. (2020), the main causes of mould are classified into two categories: high levels of humidity and presence of high levels of water in the building elements. Another factor exacerbating damp and mould in buildings is the issue of fuel poverty. Because of the increased fuel costs, some properties are occupied by more people than the property can manage, therefore making these properties more susceptible to mould.

New buildings are built with energy efficiency in mind; however, this unintendedly creates the right conditions due to the materials and ventilation achieved. New materials are often sustainable organic compounds, which provide the right nutrients for fungi to digest. No matter how much the building is developed, the most important factor to determine the moisture load in the environment is the residents. The presence of insulation in fewer parts of the building envelope (ceiling, underfloor and wall cavity) was associated with increased indoor dampness. (Taptiklis et al., 2022)

However, building operations and maintenance are critical aspects that contribute to modifying over time the operational conditions that may lead to mould growth. Thus, understanding and tracking the performance of the building over time and changing conditions is essential to better formulate design and maintenance strategies that could mitigate or prevent moisture-related damages (Mould growth in energy-efficient buildings: Causes, health implications and strategies to mitigate the risk, Brambilla & Sangiorgio, 2020).

2.4. Preventive Measures

Preventive measures can be taken to reduce the risks of mould growth in indoor spaces. The World Health Organization (WHO) recommends maintaining indoor humidity levels between 30-60% to prevent the growth of mould (Heseltine et al., 2009). This can be achieved through proper ventilation, dehumidifiers, and avoiding the use of carpets in damp areas.

Another important preventive measure is moisture control. This can involve fixing any leaks or water damage as soon as possible, ensuring proper drainage around the building, and fixing any plumbing or roof leaks. Regular maintenance of the heating, ventilation, and air conditioning (HVAC) system can also help to prevent mould growth.

In addition, it is important to address any signs of mould growth as soon as possible. This can include visible mould growth, musty odours, and water stains on walls or ceilings. Any mould growth should be cleaned up using proper safety measures, including wearing protective gear such as gloves and masks, and using proper cleaning solutions (Heseltine et al., 2009).

Following the indication of these organisations, various governments have implemented policies to maintain the safety and health of occupants.

The health and safety rating system were implemented by the Ministry of Housing in the UK. This gives social landlords a framework for the standards required for properties to be in maintained, for the health of the resident and the structural integrity of the building.

One of the 26 categories to track is Damp and Mould, which means it is a requirement for Housing organisations in the UK to collect this information for reporting to the Housing authorities. (Ministry of Housing, Communities & Local Government, 2006)

Machine Learning to Detect Mould and Defects

Organisations that manage multiple buildings and properties are constantly needing to collect intimate knowledge of the condition of all their properties for compliance and safeguarding reasons. (Perez et al., 2019)

These methods usually include regular survey checks by contractors or technicians who will report their findings to the organisations, allowing for proactive and essential maintenance repairs to take place. This process can be lengthy as the photographs, notes and drawings need to be compiled together. (Hoang, 2018; Koch et al., 2014)

Because photographs are being collected in this process, these can be used by automated systems to identify defect, however this can be challenging as with the current technology, the results and performance will be based on the quality of the images taken.

Lopez-Arce et al. (Lopez-Arce et al., 2020) state that identifying ways to avoid moisture problems and mould growth in buildings has been the subject of many research studies. Dampness and mould are commonly assessed by inspection and questionnaires, which can be highly subjective and not necessarily provide the real cause of the problem. The cause of mould growth may not be the result of just one parameter prompting the problem but a causal association between the different parameters.

Machine learning algorithms can be used to process, assess, and represent environmental data for root cause analysis of mould growth. Perez et al.(2019) built a CNN model capable to detect defects on building walls such as cracks, stains, mould and dampness, demonstrating that this avenue can be pursued to automate and efficiently improve these processes.

One limitation in this study, however, is the dataset used as this consisted of a total of 1890 photographs. Aside from studies using Machine learning to detect cracks on exterior buildings and structures, potholes on roads and spatial predictive modelling, there's a lack of research in applied machine learning algorithms specifically to the problem of mould.

Machine learning algorithms have been developed to aid in the process of building maintenance and to detect defects such as mould growth. However, these algorithms also have limitations. For example, the accuracy of these algorithms may be affected by factors such as the quality of the images taken, and the size of the dataset used for training.

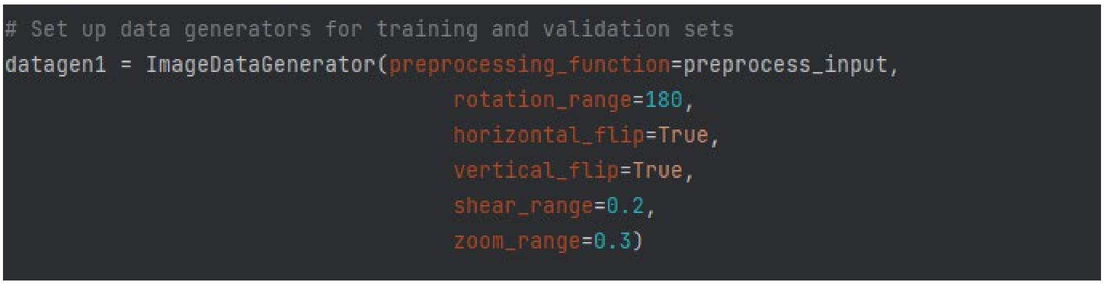
3. Data Understanding & Preparation

The dataset was compiled from publicly available photographs on the internet, these were mainly sourced from google image search.

A total of 7349 images were sourced; of which 3681 were photographs containing mould in some capacity.

My data directory is consequently split into a training and test set, with 70 % of the images used for training and the reaming 30 %, will be used as validation test, to ensure the model can be generalized to unseen images.

I also apply the same data augmentation techniques for the development of all the models.



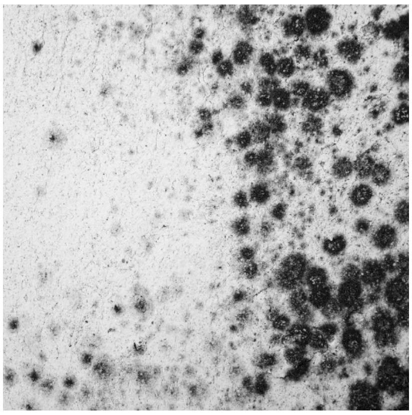
```
# Set up data generators for training and validation sets
datagen1 = ImageDataGenerator(preprocessing_function=preprocess_input,
                              rotation_range=180,
                              horizontal_flip=True,
                              vertical_flip=True,
                              shear_range=0.2,
                              zoom_range=0.3)
```

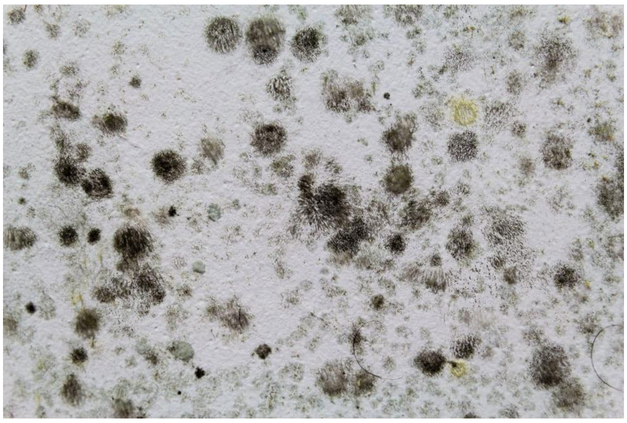
These data augmentation techniques will help the model generalise the prediction of the algorithm, aiding in avoiding overfitting (Michelucci, 2022). This is also useful the images taken during an inspection from a surveyor or contractor at a property, could not be of the best quality, due to the condition is which these are being carried out.

Leading to blurry, unclear, or just photographs, adding these data augmentation techniques will also help mimic those photos taken in worst conditions.

Sample of Photographs

Mould Photos





Clean Photos

4. Methodology

4.1. Convolutional Neural Networks

Deep Neural Networks (DNN) are a machine learning model used in Artificial Intelligence, in order to automate intellectual tasks and processes normally performed by humans (Chollet, 2018). DNN were applied for visual pattern recognition early in the 1980s (Fukushima, 1980, 1988).

Convolutional Neural Networks (CNN) are a class of Deep Learning Neural Network (DNN)architecture, mainly used for computer vision tasks like object detection in both video and photos or image classification of the content of represented in the data.(Aggarwal, 2018; Raschka & Mirjalili, 2019), enabling applications such as face recognition (Adhinata et al., 2021; Almabdy & Elrefaei, 2019; J. Wang & Li, 2018), self-driving vehicles (Dangskul et al., 2021; Duong et al., 2018) and self-service checkouts at supermarkets (Sridhar et al., 2022).

The design of CNNs is inspired by the human vision (Raschka & Mirjalili, 2019). It was observed that brain neurons respond differently. The primary layers of neurons were responsible for detecting straight lines and edges, while the deeper layers of neuron were responsible to the detection of more complex shapes and patterns (Hubel & Wiesel, 1959).

This principle was then adopted to design the first basic architecture known as the Neocognitron (Fukushima, 1980). This is a self-organising Neural Network, able to identify patterns in visual data (Ghosh et al., 2020). This architecture was further improved years later with the advent of LeNet-5, which was able to recognize handwritten digits from the MNIST dataset, without any prior feature engineering (LeCun et al., 1989).

However, these were not commonly used due to limitations in computational power required and data sets availability. The era of "big data" and the development of more powerful Graphical Processing Units (GPUs), now allows for more efficient processing and enough data to train these networks, therefore leading to the rising popularity of CNN (Perez et al).

4.2. CNN Architecture

The general architecture of a CNN is composed of multiple layers. An input layer, output layer and multiple inner layers, also known as hidden layers. The more hidden layers, the deeper the neural network (Chollet, 2018).

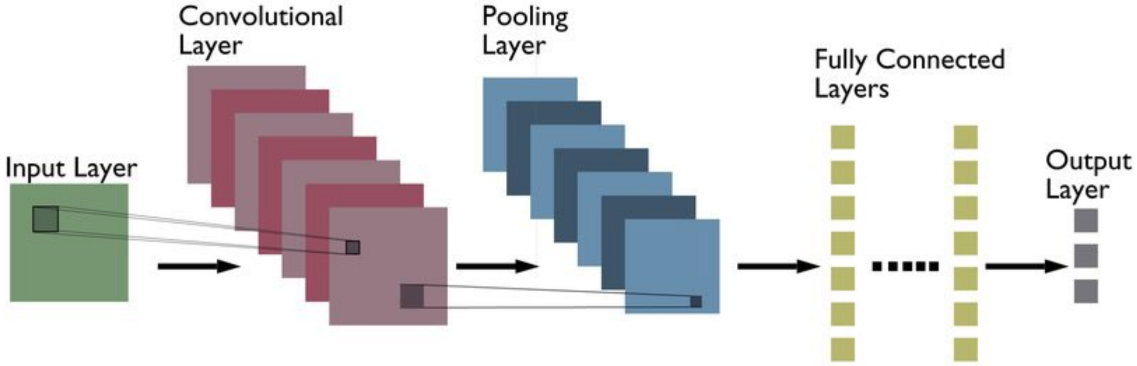


Figure 4.A: CNN Architecture (Yazdani Abyaneh et al., 2018)

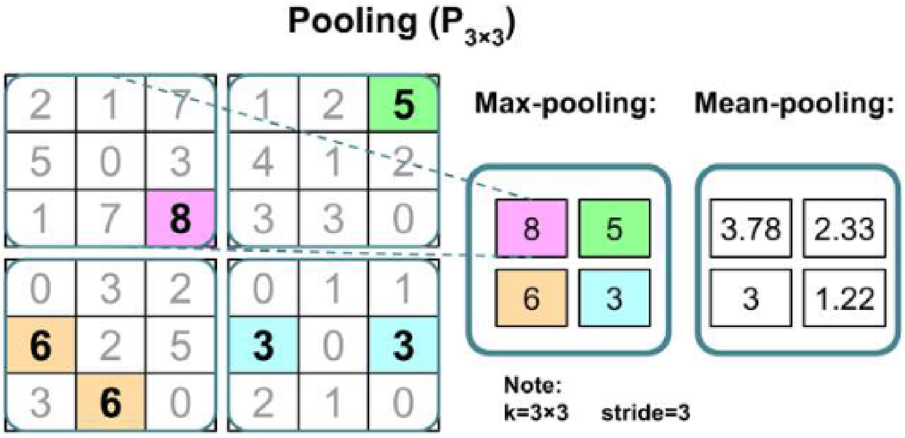
The input layer received data from the input images as raw pixel data, where each colour pixel is represented by a value between 0 and 255. Depending on the type of images being input the input layer can have a depth dimension of 1 if these are grayscale or 3 if these are RGB (Red, Green and Blue) images (Chollet, 2018).

In the convolutional layers a filter or kernel is then applied to each layer, this has smaller dimensions but the same depth as the input images. This filter it is used to slide over the entire image under a process called *convolution.* During this process the dot product between the filter and the input image, therefore producing a 2-dimensional activation map or output feature map, where every spatial location in the output feature map corresponds to the same location in the input feature map (Chollet, 2018; Ghosh et al., 2020). These feature maps will contain low level features or information such as edges, lines, and textures.

An activation function is then applied after each CNN layer, this is usually the Relu activation function. This function only retains the positive values predicted by the convolutional layers, while every negative value is changed to 0, and furthermore optimizing the efficiency of the model (Ganegedara, 2022; Michelucci, 2022)

Following a convolution, a *pooling* layer is then applied. This layer is used to reduce the size of the feature map created by the convolutional layer. Even though the size of the feature map is reduced, the most prominent features or information is retained during each pooling step (Chollet, 2018; Ghosh et al., 2020). Different types of pooling operations can be applied to in the various pooling layers.

A max – pooling layer, breaks down the convolutional layer in smaller regions, set by the pooling size, were the maximum value of each of the regions is retained, while the remaining are dropped. In an average – pooling (mean – pooling) layer, the average value of these regions is retained while dropping the rest (Raschka & Mirjalili, 2019)



2	1	7	1	2	5
5	0	3	4	1	2
1	7	8	3	3	0
0	3	2	0	1	1
6	2	5	3	0	3
3	6	0	2	1	0

8	5
6	3

3.78	2.33
3	1.22

Figure 4.B. Illustration of Pooling (Raschka & Mirjalili, 2019)

Max-pooling is the more frequently utilised technique, as it affords various advantages. With max pooling, small changes in within one of the small regions do not affect the result of pooling operation, leading to more robust features. This is known as local invariance (Raschka & Mirjalili, 2019).

Pooling further aids in decreasing the computational need of the model, parameters, and memory usage while increasing the computational efficiency. The convolutional layers extract local, hierarchical features from the input, while the pooling layers reduce spatial dimensions (Aggarwal, 2018).

Following convolutional and pooling layers, each of the features learned during the process are flattened then passed to fully connected layers. In these layers, all the neurons from the preceding layers are connected to a neuron inside the connected layer (Ghosh et al., 2020).

The last layer or output layer will be a classifier layer, with an activation function such as a SoftMax or sigmoid. This is to allows the model to use the local features learned to produce an output, such as in our case, predicting the probability of an image being part of a specific class.

Fully connected layers such as *dense* layers are more suited for learning global patterns, making them computational expensive.

CNN are the ideal network choice for image classification and recognition tasks as, when the network is trained to recognize a certain pattern in a specific location on an image, the network can recognize this anywhere else, this is known as *sparse connectivity*. This translates to data efficiency, as less training data is required for the network to learn and generalize these features (Chollet, 2018; Raschka & Mirjalili, 2019). Additionally in a CNN, the weights (parameters) are shared between all the layers in the network; this allows the network to learn one set of weight and optimise for those, instead of a new set of weight at every training layer and between each individual neurons (Ghosh et al., 2020).

CNN can learn spatial hierarchies of patterns; they are also known as *feature hierarchy*. The initial convolutional layers learn small patterns such as edges, then the successive layers will learn larger patterns made from the feature learned in the previous layers. This enables the model to learn increasingly complex features and visualisations, such as shapes of objects like microphones, smartphone or buildings (Chollet, 2018; Raschka & Mirjalili, 2019).

When it comes to images, we can assume that pixels and regions next to each other, are typically more relevant to each other, instead of regions further away. For this reason the architecture of a CNN is more suitable for extracting salient features from images (Raschka & Mirjalili, 2019).

4.3. Transfer Learning

When working with a relatively small dataset, an advantage of deep neural networks is the ability to use transfer learning. This technique consists of using a pre-trained model, as a starting point for a new model. Because of the ability of CNNs to extract low-level features, these same features can be reused to identify a different class of shapes and objects than the initial one the algorithm was trained on, effectively acting as generic model of the visual world (Chollet, 2018).

This allows algorithms to be trained on small datasets as, the convolutional layers which contained most of the weights, will not be retrained, therefore avoiding overfitting.

4.3.1. Feature Extraction

Feature Extraction is the process of using the features learned by a previous network to extract features on a new sample (Chollet, 2018). As previously detailed CNNs are composed of two areas; the convolutional base, made of a stack of convolutional layers and pooling layers, and a densely connected classifier on top.

Because convolutional layers extract general features of the input image, these same features can be used to classify different images for a different computer vision tasks (Kim et al., 2022), while the representation learned by the classifier are specialised to the set of classes the model was trained on. Furthermore, convolutional layers contain information regarding where the objects are located within an image, as the notion of space is embedded in the architecture of convolutional feature maps (Chollet, 2018). This is not present in fully connected layers, therefore making these largely ineffective for problems where the object location is necessary (Chollet, 2018).

During feature extraction these fully connected layers are replaced with a new classifier to suit the current task at hand as illustrated in figure 4.C.

The level of reusability of each model, will depend on the depth of the model. Initial convolutional layers will extract general features, which can be more easily reused for a different task. Later layers will extract more complex features more related to the training dataset. For this reason, re using a model with a dataset that widely differs from the initial training dataset, it might be advantageous to only keep the first few layers of the model (Chollet, 2018; Salehi et al., 2023)

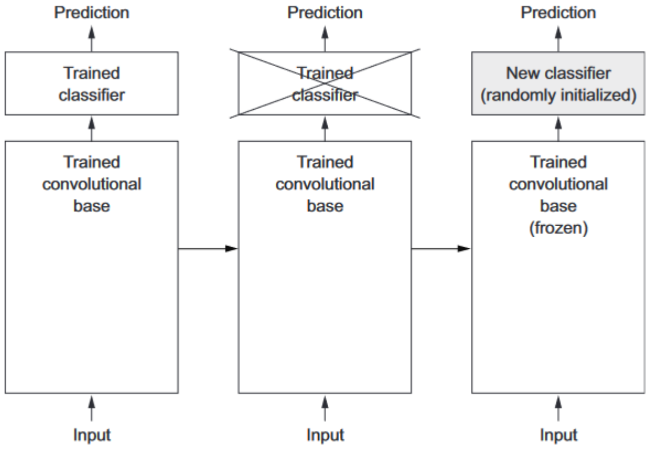

Prediction
Trained classifier
Trained convolutional base
Input
Prediction
Trained classifier
Trained convolutional base
Input
Prediction
New classifier (randomly initialized)
Trained convolutional base (frozen)
Input

Figure 4.C Transfer Learning Diagram (Chollet, 2018)

4.3.2.Fine – Tuning

Another transfer learning technique available, is f*ine-Tuning*. During fine-tuning, the top convolutional layers of a model, are jointly made available to train with the new fully connected classifier on top. This technique allows, to re-train the later convolutional layers, which will contain specialised features based on the original training data (Chollet, 2018; Salehi et al., 2023)

Re-training these layers will allow to repurpose these layers to be more specialised with the dataset at hand, therefore potentially improving the accuracy of the prediction.

4.4. Hyper parameters

There are various parameters used to build the network that can be changed to optimize the performance and generalisation ability of the network. These parameters will be tested, adapted, and tailored to the architecture and image recognition tasks at hand.

There are various of these moving parts in a CNN which can be tuned to better enhance the performance of model (Raschka & Mirjalili, 2019). Due to the variety and possible combinations of these hyper-parameters, not all of these could be considered and analysed; in the following sections of this chapter, a brief discussion will be presented regarding optimisers as these will be the variable that will be changed during the modelling process.

4.4.1.Optimiser

The optimizer are implemented to reduce the loss function of the model.

4.4.1.1. Adam

Adam is the better performing algorithm (Desai, 2020)

Introduced by Duchi et al. in 2011, Adagrad is an adaptive learning rate method designed to improve the efficiency and performance of stochastic optimization algorithms in machine learning.

Description: Adagrad modifies the general learning rate η at each time step t for every parameter based on the past gradients squared:

Rationale: The motivation behind Adagrad was to allow different learning rates for different parameters. Parameters associated with frequently occurring features would have their learning rates reduced, while infrequent features would have their rates increased.

Efficiency: While Adagrad can effectively handle sparse data and has the ability to adapt the learning rate, it has a shortcoming: its learning rate tends to decrease rapidly, which might make the algorithm converge prematurely.

Reference: Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul), 2121-2159.

2. Adadelta:

Following the introduction of Adagrad, Zeiler proposed Adadelta in 2012 to address some of Adagrad's limitations.

Description: Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size

�

w.

Rationale: The primary motivation behind Adadelta was to address the rapidly decreasing learning rates of Adagrad. By considering only a window of past gradients, Adadelta ensures a more dynamic adaptation of learning rates.

Efficiency: Adadelta often performs well in practice and requires no manual setting of a learning rate, making it more robust than Adagrad in certain scenarios.

Reference: Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

3. Adam (Adaptive Moment Estimation):

Introduced by Kingma and Ba in 2015, Adam combines the ideas of Adagrad and another optimization method called Momentum.

Description: Adam computes adaptive learning rates for each parameter by considering not only the first moment (the mean) but also the second moment (the uncentered variance) of the gradients. The method maintains running averages of both the gradients and their square values, using them to adaptively adjust the learning rate for each parameter.

Rationale: The innovation behind Adam was to provide the benefits of both Adagrad's adaptive gradient-based learning rates and Momentum's incorporation of past gradients to achieve smoother updates.

Efficiency: In many scenarios, Adam has been observed to outperform other adaptive learning rate optimization algorithms. Its balanced approach and consideration of momentums often lead to faster convergence.

Reference: Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

In conclusion, the quest for efficient optimization in deep learning has

4.5. Summary

Convolutional neural networks are the best type of machine learning technique to be used for computer vision tasks.

Explain which methods will be used and tested during the modelling process.

5. Modelling

During the modelling process, various hyper-parameters were tuned to make sure the model developed was optimal. For this reason, 3 different CNN architectures were considered:

- VGG-16
- EfficientNetV2
- MobileNetV2

For each architecture, various dropout layers were tested. This being no dropout, 1 dropout layer at 0.5 or 1 dropout layer at 0.8. The various additional layers were added if the base model was producing signs of overfitting.

The last component tested during the modelling phase are the different optimizer. An optimizer allows the model to minimize the loss function, by finding the combination of weight and parameters that minimize this curve.

Different optimizers have been developed throughout the years. The simplest optimizer developed is the Stochastic gradient descent (SDG).

For this research, 3 different optimizers were considered. Adam, AdaGrad and Adadelta to establish which of these:

1. Is more efficient in terms of speed of convergence and training.
2. Which one provides the most accurate model

All these models were also further fine-tuned by unfreezing the last convolutional layer and retraining the unfrozen layers. This practice should aid in increasing the accuracy of the model and assist the model to develop feature maps more specialised for my dataset.

During fine tuning a slower optimiser was used, with a slow learning rate. This is to avoid the network from drastically changing the previously learned features, and to allow it to slowly adapt the parameters.

The loss function used is the binary-cross entropy loss function. This is because the task at hand is a binary classification task, where we are just looking for the probability an image belonging to one of 2 classes (Chollet, 2018; Kapoor et al., 2022)

The training of the added additional custom layers will be carried out on 10 epochs, while the fine tuning is carried out for 25 epochs.

5.1. VGG-16

The VGG-16 model, is a deep convolutional neural network introduced in 2014 and the second place winner of the ImageNet Large Scale Visual Recognition Challenge (Aggarwal, 2018; Simonyan & Zisserman, 2015). VGG16 is a simple model to implement and understand making it one of the most popular CNN architectures.

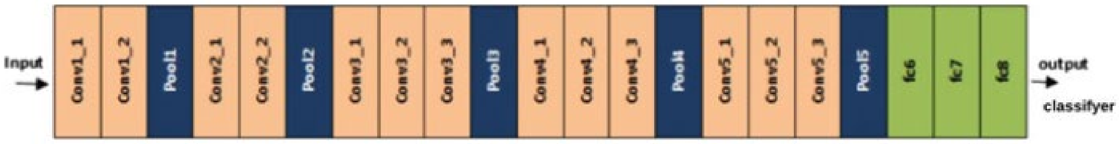


Figure 5.A VGG-16 Network Architecture (Ghosh et al., 2020)

It is comprised of 16 layers, 13 of which are convolutional layers followed by 3 fully connected layers (Shaha & Pawar, 2018). The size of the filters or kernels were reduced to the smallest dimension that can retain the idea of, up, down, left, and right, which is 3 x 3 kernel.

The use of small filters allows is beneficial because the number of parameters in each convolutional layer is produced by the square of the filter size (Aggarwal, 2018), this reduction in layer parameters, allows us to add more depth to the model, increasing non linearity due to the ReLU layers and more regularisation (Aggarwal, 2018).

The dimensionality of the convolutional layers is then reduced with a 2 x 2 pooling layers applied after every block of convolutional layers. These layers act as a funnel, allowing the features with the most information to make it down the pipeline and not get disperse in the noise (Simonyan & Zisserman, 2015).

VGG-16 has proved to be versatile and effective in image classification tasks achieving high accuracy with relatively low training times, especially when used as transfer learning (Adhinata et al., 2021; Albashish et al., 2021; Alippi et al., 2018; Mascarenhas & Agarwal, 2021).

There are various downside to the use of this model however, with 528 MB in size, this is one of the more "heavy" algorithms to use especially if this will be implemented in a web or mobile application (Simonyan & Zisserman, 2015). Furthermore because of the depth of the architecture, there are more than 138 million parameters available to train, which add computational cost to the development of any model based on this architecture. However, because this is used for as transfer learning most of the layers will be frozen to avoid retraining the entire model. This will aid in avoiding overfitting as the dataset I am working with is relatively small.

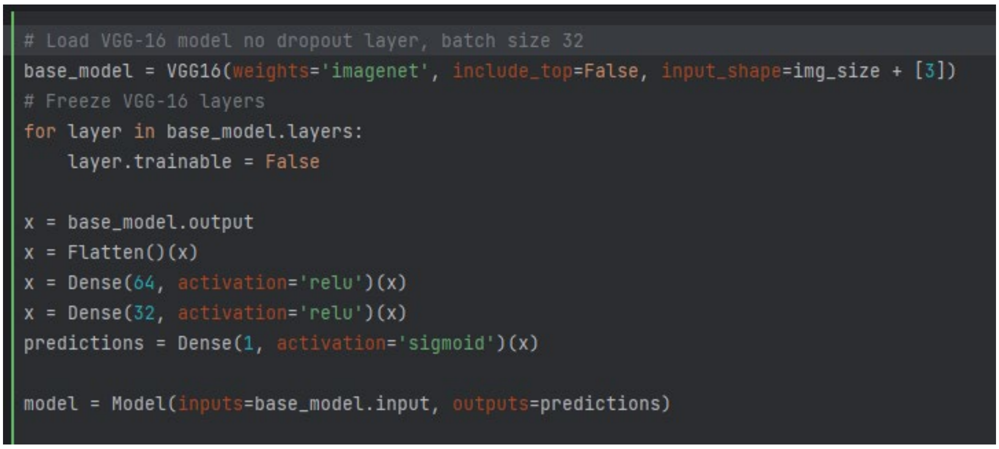
This model is being considered in this project due to its versatility, as well as the only other CNN paper research carried out on mould was carried out in 2016 using the VGG-16 architecture(Perez et al., 2019), in this study the model developed was able to classify images as containing mould with around 91 % accuracy.

As this model is used for transfer learning, the last 3 fully connected dense layers with the classifier will be replace with custom layers, for the problem at hand.

5.1.1.Model 1- Adam Optimizer

First, the model was initialised with the image net weights the VGG-16 was initially trained on, without the top classification layers.

For this model the image size is set at 224 x 224 as this is the size of used to train the network



```
# Load VGG-16 model no dropout layer, batch size 32
base_model = VGG16(weights='imagenet', include_top=False, input_shape=img_size + [3])
# Freeze VGG-16 layers
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dense(32, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

For the first version of the model, the convolutional base was flattened, followed by 2 dense fully connected layers of 64 and 32 neurons each, with a ReLU activation function.

Larger feature maps i.e., dense connected layers with many neurons are more beneficial for complex task and to extrapolate complex representations. Mould is a relatively easy shape to observe, and most of the photographs taken will have a clear contrast between the mould and the surface. For this reason, I believe a larger feature map than 128 neurons, would be impractical as it this adds computational complexity. This is the reason, a dense neural network of 64 neurons was chosen as a higher number of neurons would increase the number of parameters exponentially (Albashish et al., 2021)

Because the convolutional base has been frozen, the parameters that will be trained will only be the ones in these additional layers.

The last layer for all the models developed is a dense layer of size 1 with a sigmoid activation function. This is implemented because this is a binary classification problem.

The sigmoid activation function will convert the inputs into a probability classification between 0 and 1 (Chollet, 2018).

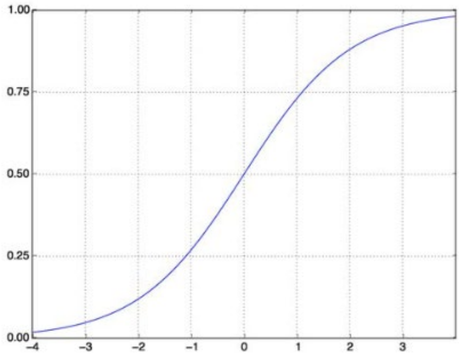


Figure 5.B Sigmoid Function (Chollet, 2018)

The batch size set for training was kept at 32 which is what was used for the initial training of the VGG-16 model.

As evident from the validation and accuracy loss graphs, although we achieved an accuracy of about 93%, it can be observed that the loss validation curve is does start to show sign of overfitting even though data augmentation has been applied.

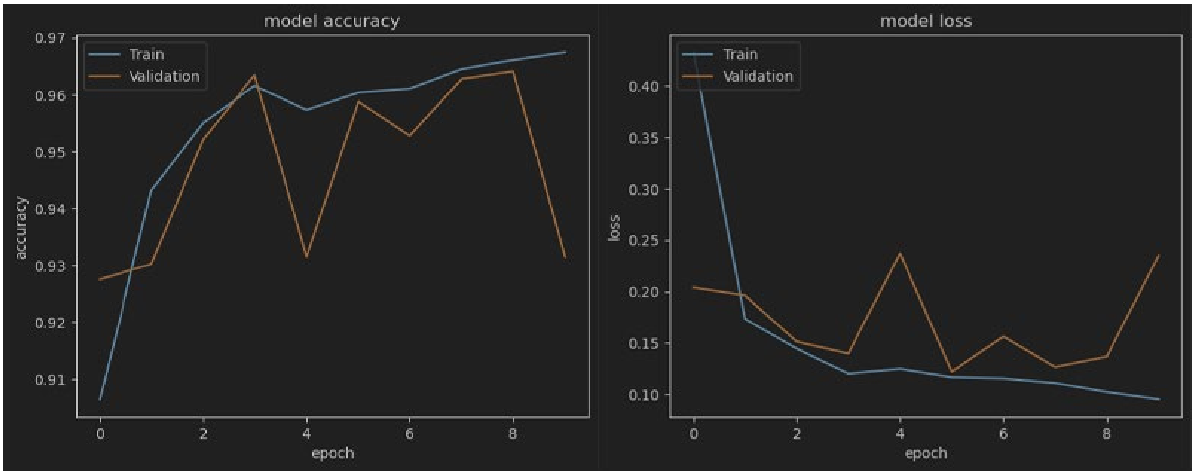


Figure 5.C VGG16 - Validation and Accuracy curves - No dropout and no fine-tuning

I proceed to apply fine tuning to this model, to see if a higher accuracy and lower loss can be achieved through this method.

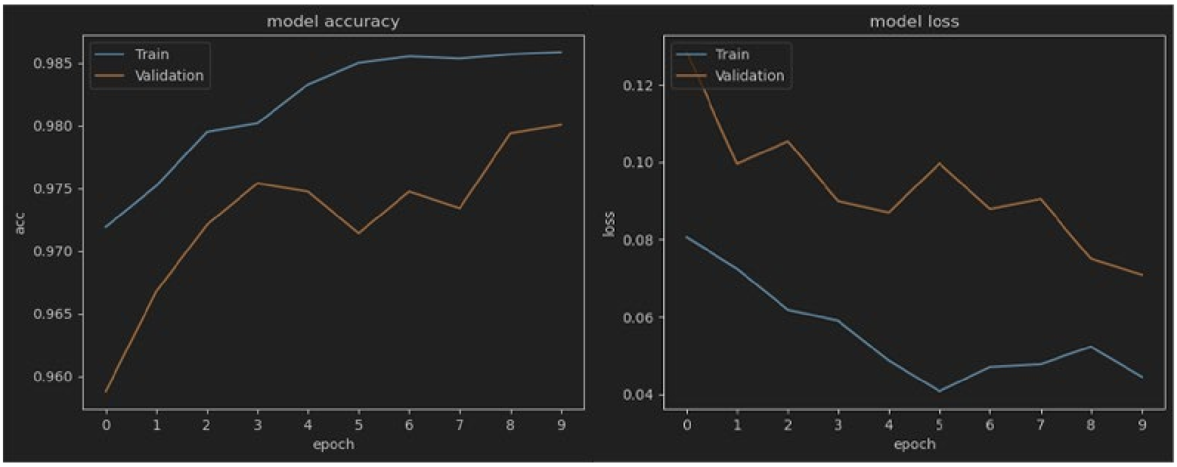


Figure 5.D VGG16 Fine-Tuned with Adam optimizer.

Fine tuning the model for 10 epochs, we can observe a higher accuracy already just after the first epochs with a validation accuracy of 95%. To always capture the best generalizing model, I employed a callback function to save the model with the lowest validation loss at the end of every epoch.

Following 10 epochs the validation accuracy is of 98%, however it is evident from the graph that we are observing some underfitting, it would be therefore beneficial to train the model for more epochs and observe the behaviour.

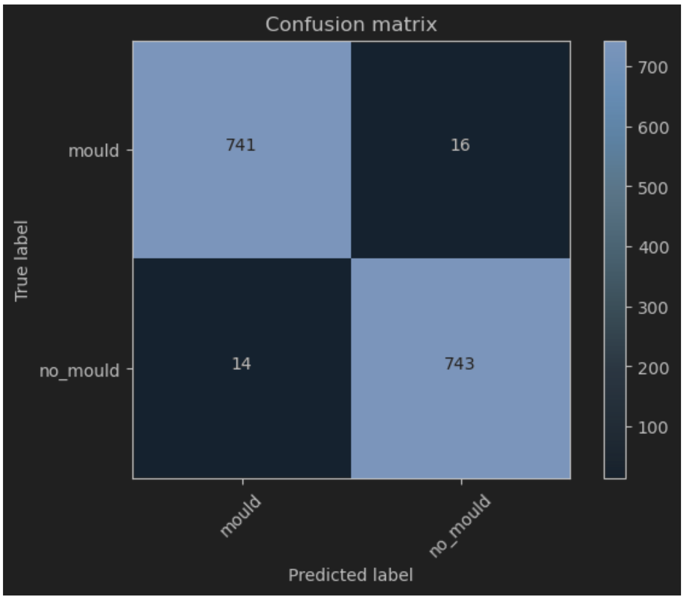
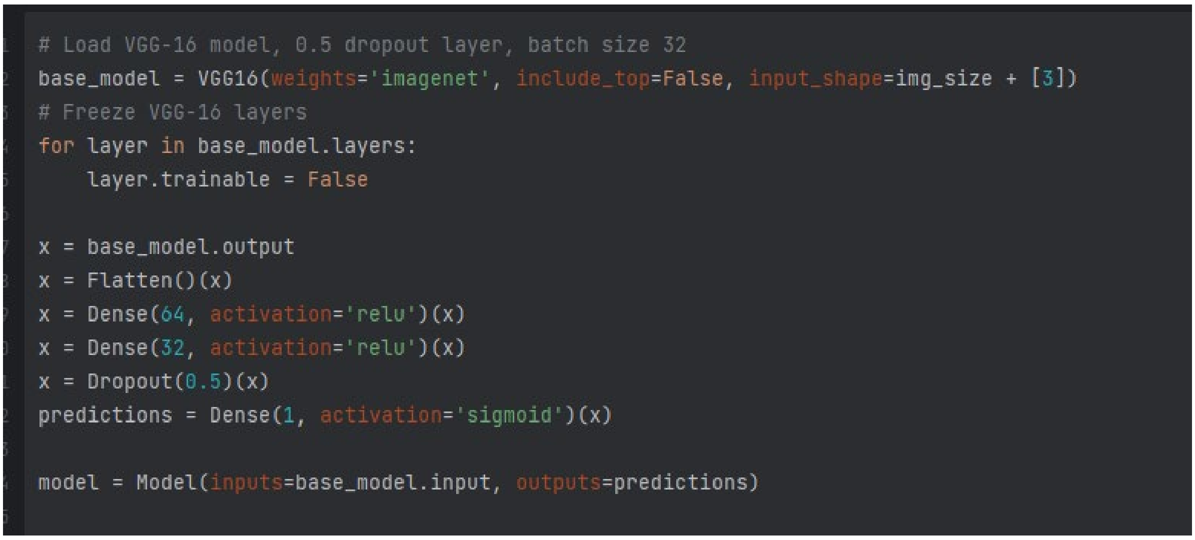


Figure 5.E VGG16 no dropout and fine-tuned, Adam Optimizer

A look at the confusion matrix of this first model we can observe a good recall and precision. The goal of the model is to be able to recognise photographs containing mould, although this model is performing excellently. It has a slight bias towards classifying a mouldy photograph, as being clean as we can see from the 16 false positive.

As this a health and safety concern, it would be more beneficial for the model to overestimate the presence of mould in a photograph, instead of underestimate them.

The next iteration of the model we add a drop out layer with a rate of 0.5, this means that half of the neurons of the dense layer before the classification layer won't be activated. This measure is used to combat the overfitting observed from the previous iteration.



```python
# Load VGG-16 model, 0.5 dropout layer, batch size 32
base_model = VGG16(weights='imagenet', include_top=False, input_shape=img_size + [3])
# Freeze VGG-16 layers
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

Figure 5.F VGG 16 - 0.5 dropout layer, Adam optimiser

The generalisation ability of this model looks to be already much better than the previous iteration. As observed from figure 5.G, the validation and training loss are closely trending downwards and converging.

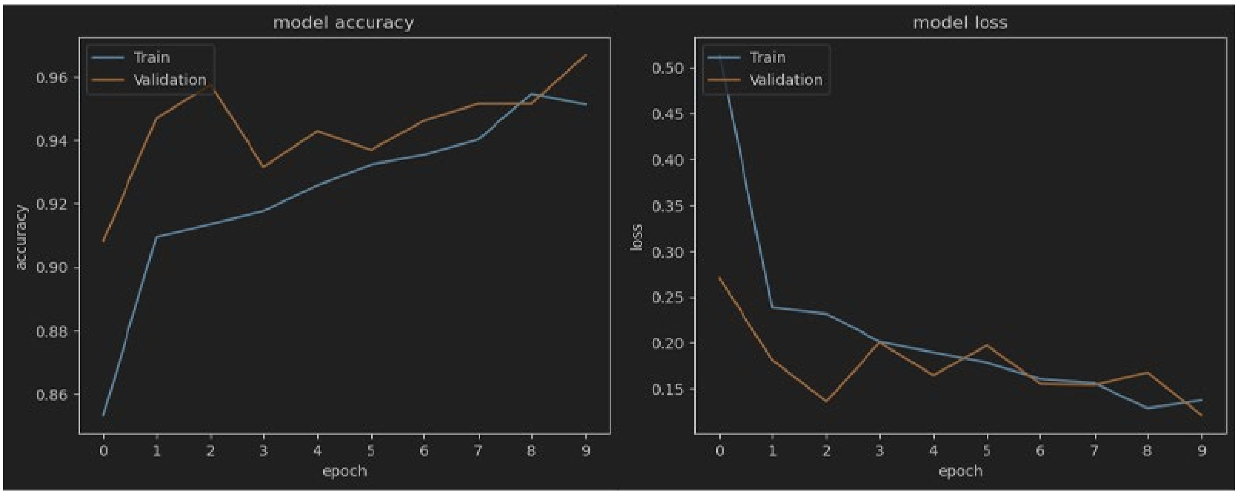
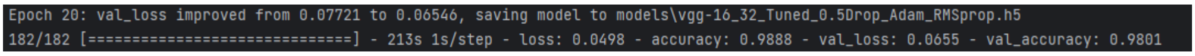


Figure 5.G VGG 16 Model 05 dropout layer, Adam optimiser

The highest accuracy achieved with this model however is about 96%. I proceeded to fine tune this model by unfreezing the last convolutional layer and retraining the model.



```
Epoch 20: val_loss improved from 0.07721 to 0.06546, saving model to models\vgg-16_32_Tuned_0.5Drop_Adam_RMSprop.h5
182/182 [==============================] - 213s 1s/step - loss: 0.0498 - accuracy: 0.9888 - val_loss: 0.0655 - val_accuracy: 0.9801
```

We can see on the last epoch the model was still improving increasing accuracy to 98%. The confusion matrix however does not paint a good picture however as this is still classifying 21 mould images as not containing mould.

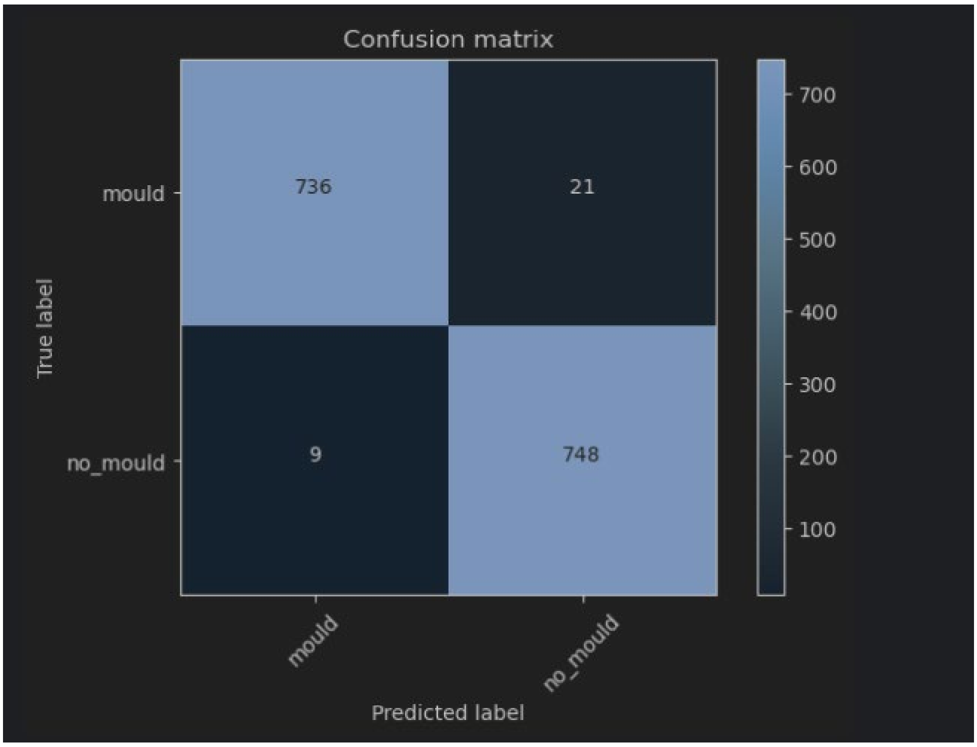


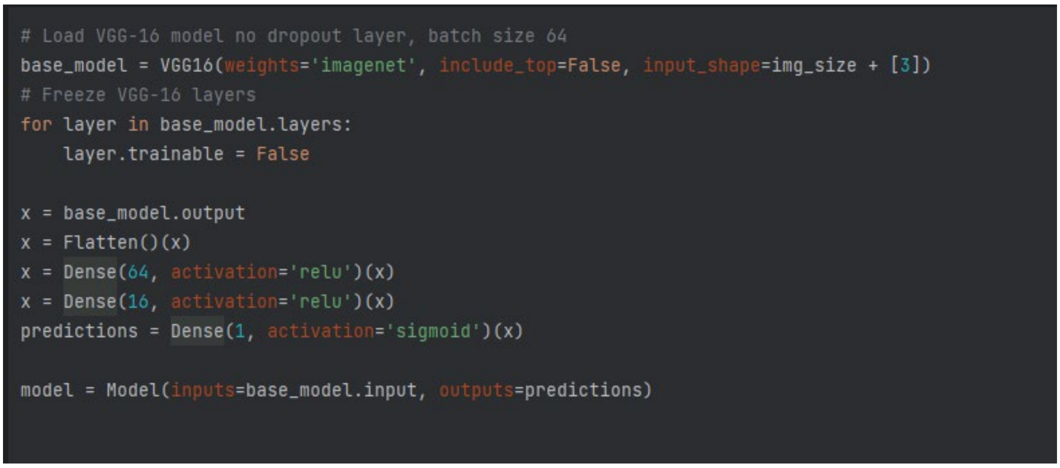
Figure 5.H VGG 16 Adam 0.5 dropout layer fine- tuned

A last model with an aggressive dropout layer of 0.8 was also tested. This model however was the worst performing model with high levels of underfitting.

5.1.2. Model 2 – AdaGrad Optimizer

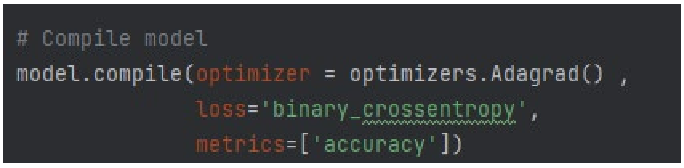
The second model developed as stated earlier, differs only in the optimizer used to minimise the loss function. The first implementation of the model is composed of the frozen base convolutional layer, with 3 fully connected layers on top. The first dense layer is composed of 64 neurons, while the following one has a smaller feature map. This decision was taken to aid the overfitting observed in the Adam model.

A small feature map has fewer parameters also help with faster training (Barman et al., 2019; Kapoor et al., 2022).



```
# Load VGG-16 model no dropout layer, batch size 64
base_model = VGG16(weights='imagenet', include_top=False, input_shape=img_size + [3])
# Freeze VGG-16 layers
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dense(16, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

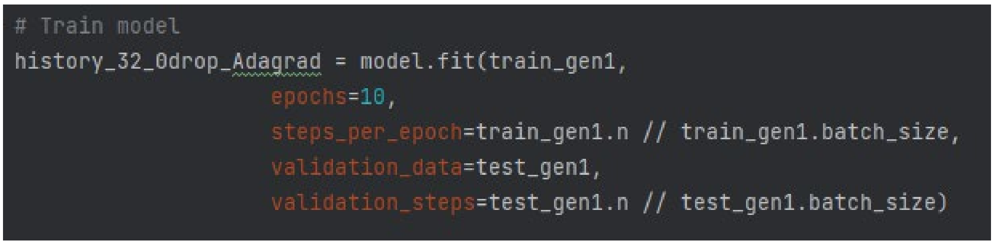
Figure 5.I VGG16 AdaGrad Optimizer model



```
# Compile model
model.compile(optimizer = optimizers.Adagrad() ,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Figure 5.J VGG16 - Adagrad Compile

The first iteration of the model was trained for 10 epochs. To ensure that during each epoch, the whole dataset is analysed, the following lines of codes were implemented during the fir of the model.



```
# Train model
history_32_0drop_Adagrad = model.fit(train_gen1,
                    epochs=10,
                    steps_per_epoch=train_gen1.n // train_gen1.batch_size,
                    validation_data=test_gen1,
                    validation_steps=test_gen1.n // test_gen1.batch_size)
```

As it can be observed from the above code. Instead of setting a fixed step per epoch, we are calculating how many complete batches will be crated by the generator during one epoch.

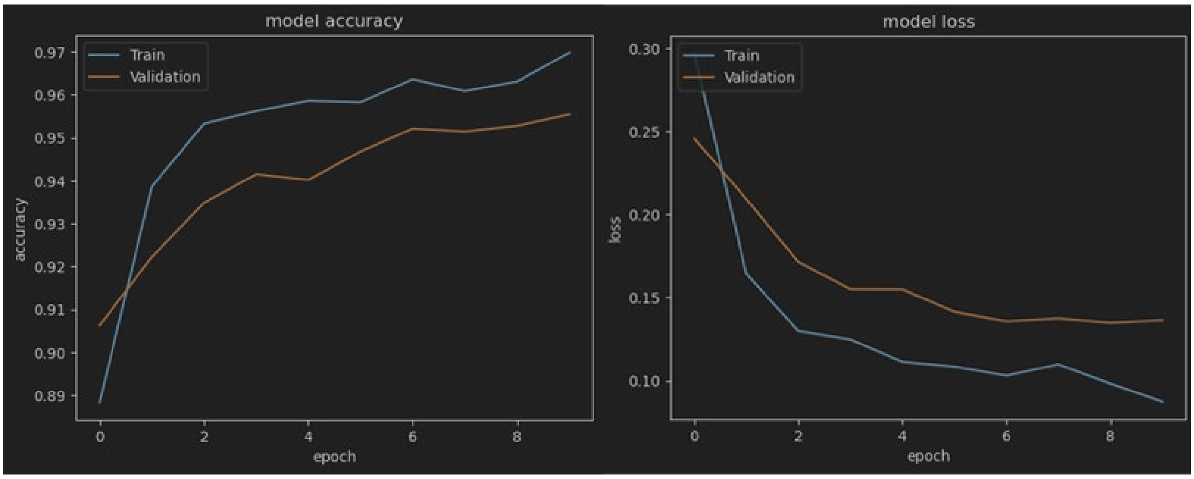


Figure 5.K VGG16 - Validation and Accuracy curves - No dropout and no fine-tuning Adagrad

Following the first 10 epoch of training we can observe that the model is starting to suffer from overfitting with the validation loss starting to plateau around epoch 7, while the training loss is still reducing.

Unsurprisingly the model following fine tuning, displays high levels of overfitting. This is to be expected as the dataset used is relatively small and with no dropout layer, the model is just getting better at learning features from the training data, without being able to generalise.

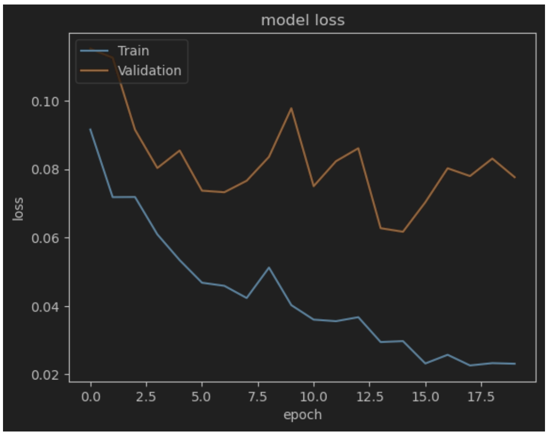
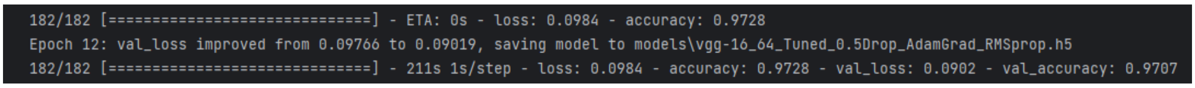


Figure 5.L VGG AdaGrad - Fine tuned loss

The next iteration of this model was to add a dropout layer as previously done for the Adam optimise model.

This model happened to be a really good model for generalisation as evident in figures 5.M and 5.N, there are little to no signs of either overfitting or underfitting, for the base model and the fined tuned model.

During fine tuning we observe a slight plateau of the loss curve around 12 epochs, using an early stopping method, I saved the model at this point in the training which happened to be the lowest loss in the whole model and an accuracy of 97.07%



```
182/182 [==============================] - ETA: 0s - loss: 0.0984 - accuracy: 0.9728
Epoch 12: val_loss improved from 0.09766 to 0.09019, saving model to models\vgg-16_64_Tuned_0.5Drop_AdamGrad_RMSprop.h5
182/182 [==============================] - 211s 1s/step - loss: 0.0984 - accuracy: 0.9728 - val_loss: 0.0902 - val_accuracy: 0.9707
```

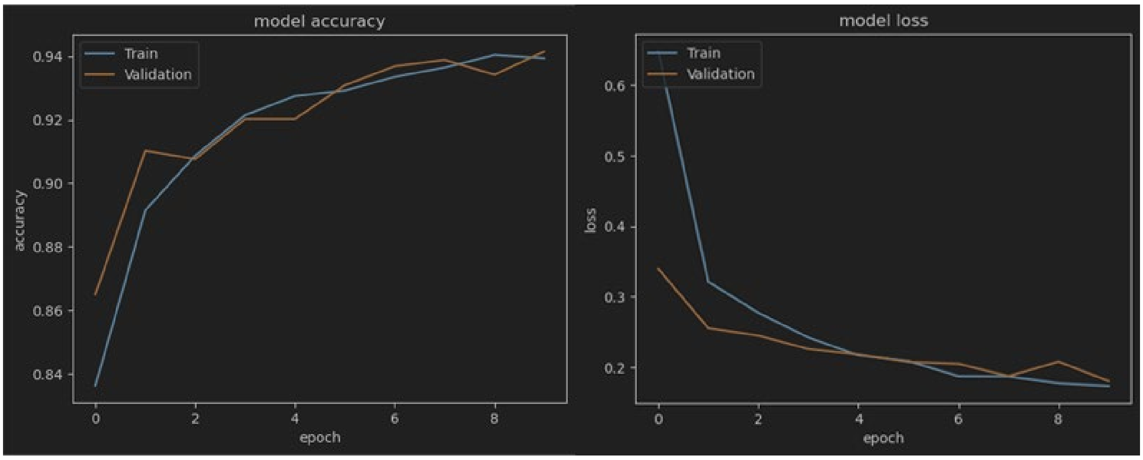


Figure 5.M VGG 16 Adagrad 0.5 Dropout

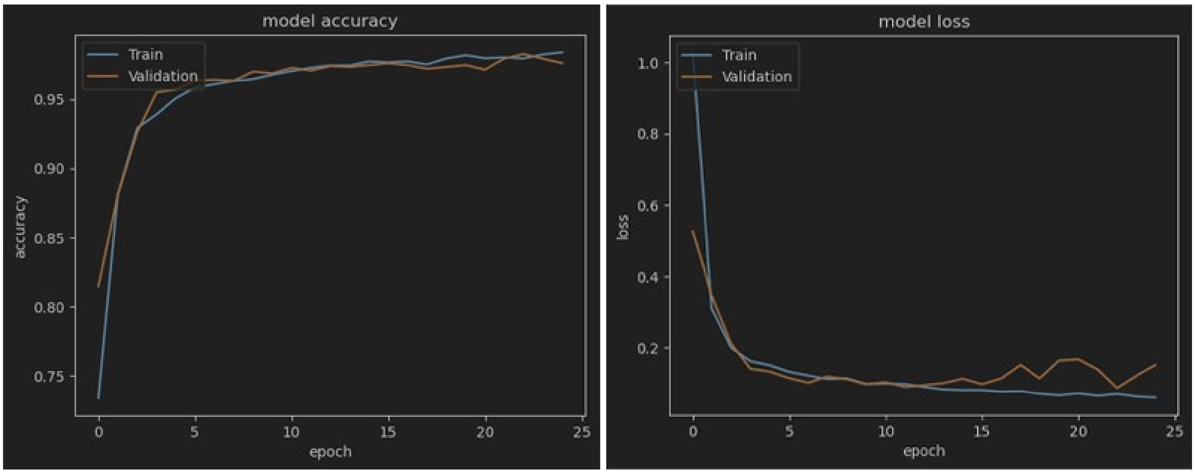


Figure 5.N- VGG 16 Adagrad 0.5 Dropout - Fine tuned

The confusion matrix for this model is also showing good signs of a good predictive model, however we are still observing the model being more capable in classifying clean images.

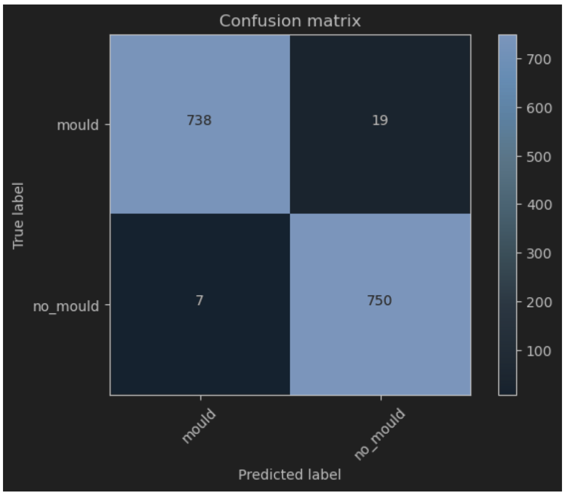
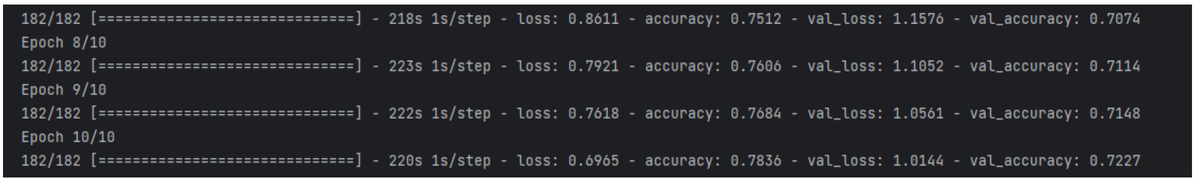


Figure 5.O VGG 16 - AdaGrad Fine-tuned confusion Matrix

5.1.3.Model 3 – Adadelta Optimizer

The 3rd iteration of the VGG 16 model, was initialised with the Adadelta optimiser and no dropout layers.



```
182/182 [==============================] - 218s 1s/step - loss: 0.8611 - accuracy: 0.7512 - val_loss: 1.1576 - val_accuracy: 0.7074
Epoch 8/10
182/182 [==============================] - 223s 1s/step - loss: 0.7921 - accuracy: 0.7606 - val_loss: 1.1052 - val_accuracy: 0.7114
Epoch 9/10
182/182 [==============================] - 222s 1s/step - loss: 0.7618 - accuracy: 0.7684 - val_loss: 1.0561 - val_accuracy: 0.7148
Epoch 10/10
182/182 [==============================] - 220s 1s/step - loss: 0.6965 - accuracy: 0.7836 - val_loss: 1.0144 - val_accuracy: 0.7227
```

This model after the first training session is the worst performing model, achieving only and accuracy of 72% following 10 epochs.

Plotting the loss curve also reveals that we are having overfitting in the model. However, by fine tuning the model, we observe much lower loss after 25 epochs and less overfitting.

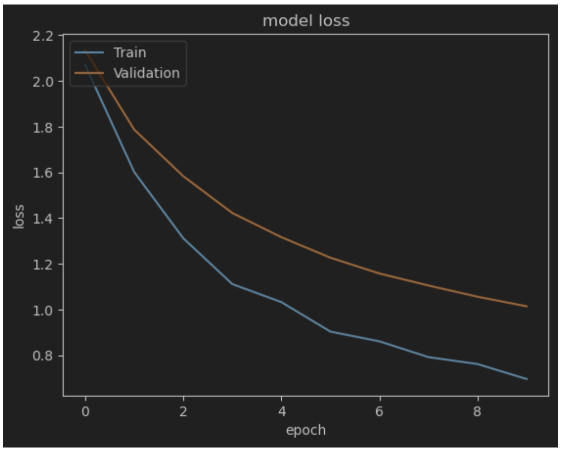


Figure 5.Q VGG 16 Adadelta no dropout – No Fine tuning

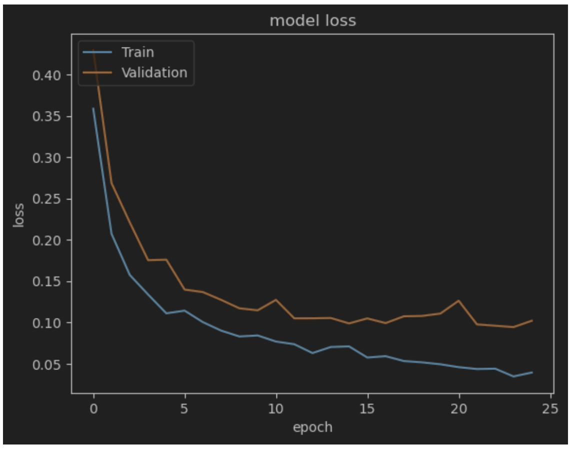


Figure 5.P VGG 16 Adadelta no dropout – Fine tuning

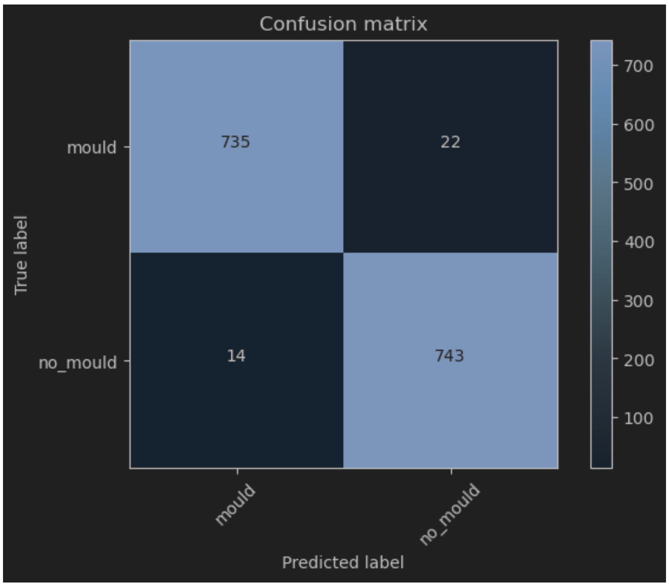


Figure 5.R VGG Adadelta no dropout - Confusion Matrix

To address the overfitting issues, just as before a drop out layer of 0.5 is added to reduce the complexity of the model and allow it to generalise to unseen data. Following training for 10 epochs the model derived was performing worse than the previous iteration without a drop out layer, because of the less information being passed to the classifier. This model only happens to achieve 65% validation accuracy. This could be due to the slow convergence of the Adadelta optimiser.

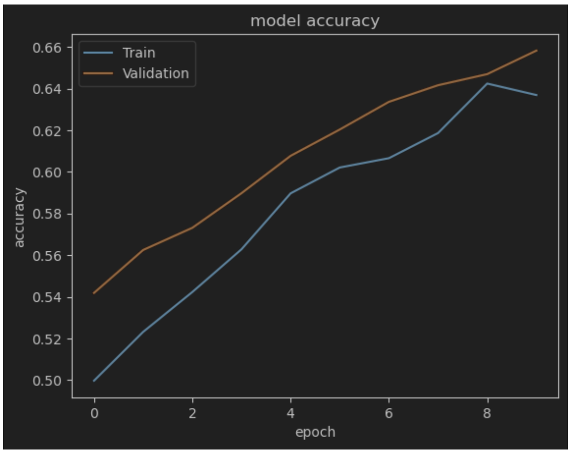


Figure 5.T VGG 16 Adadelta - 0.5 dropout

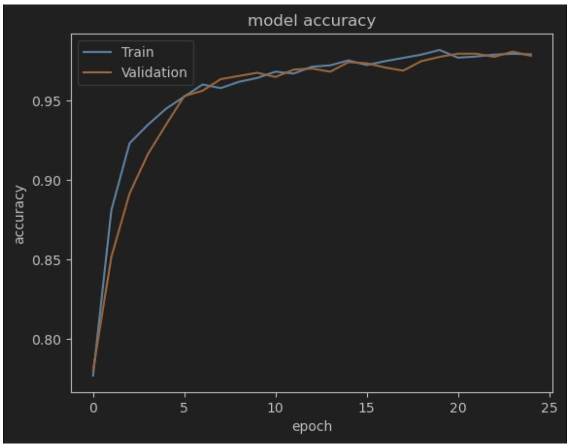


Figure 5.S VGG 16 Adadelta - 0.5 dropout fine- tuned

However, following fine tuning over 25 epochs with a slower learning rate, with RMSpropr optimiser, drastically increased both training and validation accuracy. With a high validation accuracy of 98.07%.

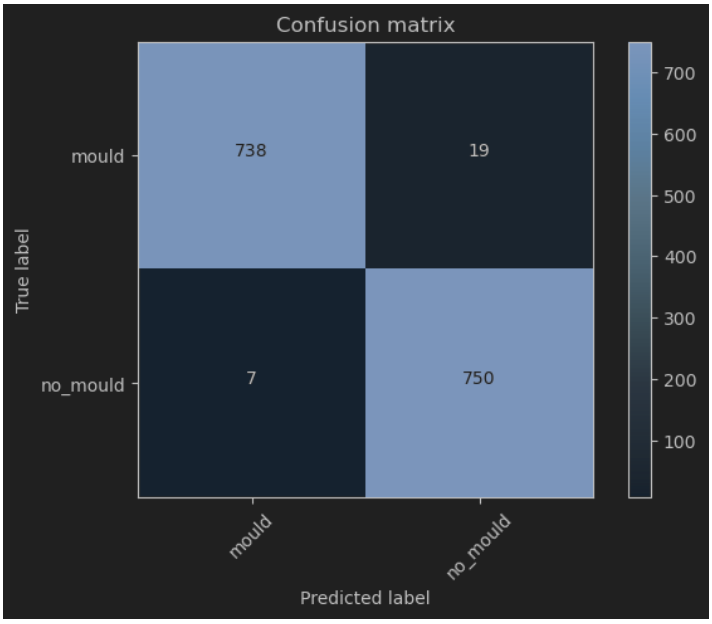


Figure 5.U Confusion Matrix - VGG 16 Adadelta - 0.5 dropout fine- tuned

5.2. EfficientNetV2s

EfficientNetV2 is a successor to the EfficientNet series, developed by researchers at Google Research. Introduced in 2021, this model builds on the foundational principles of the original EfficientNet models. The suffix "s" in EfficientNetV2s stands for "small", signifying that this is the smallest variant in the EfficientNetV2 series, which also includes medium and larger versions. (Tan & Le, 2021)

EfficientNetV2 employs compound scaling, a feature present in previous iteration of EfficientNet. This simultaneously scales the width, depth, and resolution of the model, ensuring a stable balance among these dimensions. This strategy of progressive learning allows the model to be trained with increasing resolutions in a sequential manner. This approach improves model accuracy and speeds up training. (Tan & Le, 2019)

This model is the smallest variant in the series, and has fewer parameters than its larger counterparts, making it more lightweight and faster. However, despite its smaller size, it offers a competitive performance, showcasing the efficiency of the architecture.

The network construction emphasizes efficiency in both model size and training speed. By leveraging fused-MBConv blocks and progressive learning, it achieves faster training times compared to its predecessor. Additionally, the architecture is carefully constructed using compound scaling to find an optimal balance between depth, width, and resolution (Tan & Le, 2021)

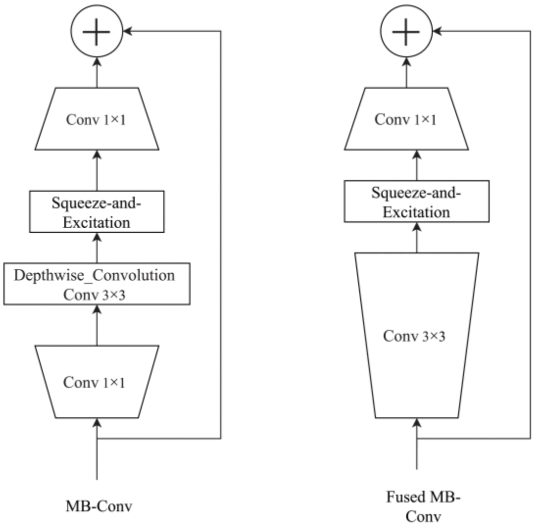


Figure 5.V Structure of MB-Conv and fused MB-Conv (Tan & Le, 2021)

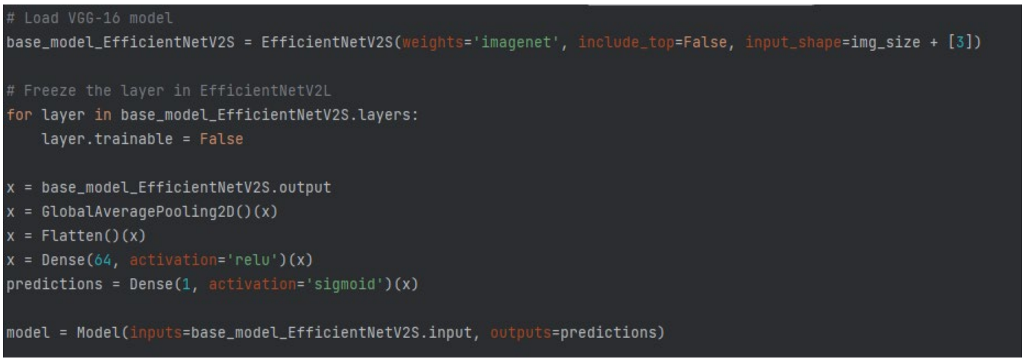
Since its introduction, EfficientNetV2 has gained significant attention in the image classification domain. Its ability to provide state-of-the-art performance with fewer parameters and faster training times makes it a preferred choice for many researchers and practitioners. The series, especially the smaller variants like EfficientNetV2s, is particularly popular in scenarios with resource constraints (C. K. et al., 2022)

This model was selected because of the requirement of the model to be used in web applications. Being much lighter than the VGG 16 architecture. This model will provide a good comparison level to the older VGG-16, therefore allowing us to test the new developments in computer vision.

As recommended by the technical documentation, we use an image size of 300 x 300 and a batch size of 32.

5.2.1. Model 1 – Adam optimiser

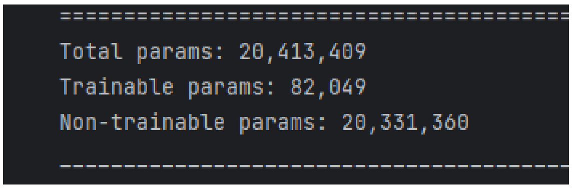
The first iteration of the EfficientNetV2 model, was initialised without the top classifier as usual. We followed this with a Global average pooling 2d layer, to reduce the dimensionality of the previous layers as EfficientNetV2, doesn’t use pooling layers (Tan & Le, 2021). This is then flattened before being passed to a dense layer of 64 neurons followed by a dense layer of 1 with a sigmoid function.



```
# Load VGG-16 model
base_model_EfficientNetV2S = EfficientNetV2S(weights='imagenet', include_top=False, input_shape=img_size + [3])

# Freeze the layer in EfficientNetV2L
for layer in base_model_EfficientNetV2S.layers:
    layer.trainable = False

x = base_model_EfficientNetV2S.output
x = GlobalAveragePooling2D()(x)
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model_EfficientNetV2S.input, outputs=predictions)
```

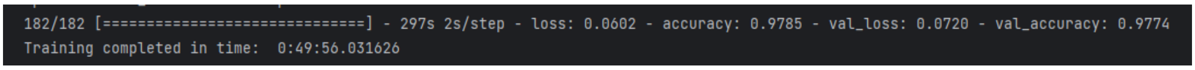
Figure 5.W- EfficientNetV2

These settings allow us to only train 82k parameters instead of the 20 million available.



```
==========================================
Total params: 20,413,409
Trainable params: 82,049
Non-trainable params: 20,331,360
__________________________________________
```

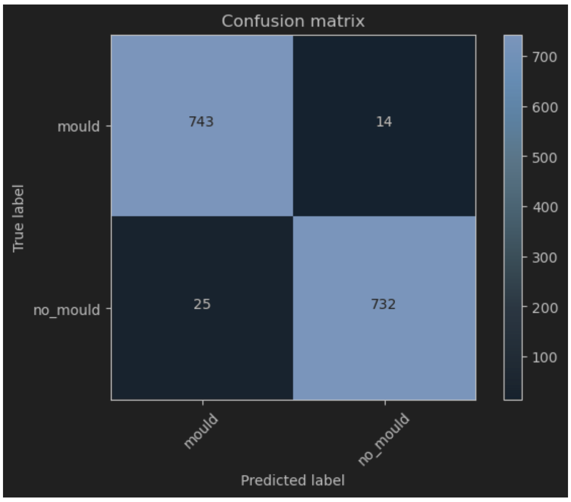
After 10 epochs of training, we are observing a drastic increase in train and validation accuracy.



```
182/182 [==============================] - 297s 2s/step - loss: 0.0602 - accuracy: 0.9785 - val_loss: 0.0720 - val_accuracy: 0.9774
Training completed in time:  0:49:56.031626
```

The performance of the model is optimal from a confusion matrix perspective, with a balanced prediction between true positives and predicted positives.

It also observed how the model now when the correct classification is not predicted, it is more likely to class this image as containing mould. In the context of a real life application, this is more beneficial for housing associations and tenants, as it is better to assume that mould is present in property, as to avoid cases where mould goes undetected for years.



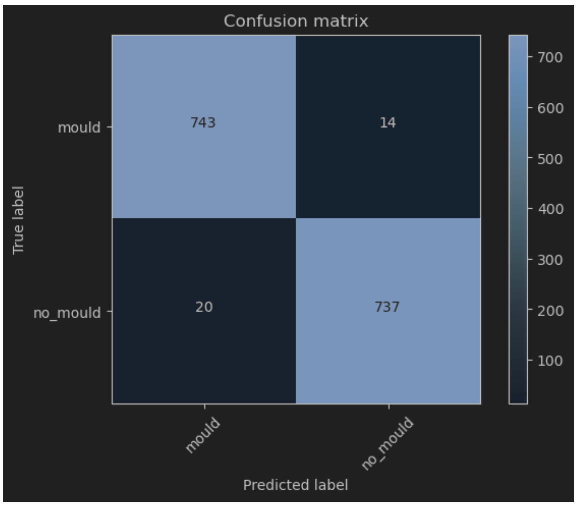


Figure 5.X EfficientNetV2 confusion matrix 0.5 dropout (Right)

Figure 5.Y EfficientNetV2 confusion matrix no dropout (Left)

Adding a layer of dropout however further improved the performance as the confusion matrixes above, however the accuracy of both model is at 97%.

5.3. MobileNetV2

MobileNetV2 is a continuation of the MobileNet series, pioneered by researchers at Google. Launched as a successor to the original MobileNet model, MobileNetV2 was introduced with the aim of providing both enhanced performance and efficiency. It's particularly tailored for mobile and edge devices, where computational resources are limited.

One of the defining features of MobileNetV2 is its use of inverted residuals. Traditional residual blocks expand the number of channels at the start and compress them towards the end. In contrast, the inverted residual blocks in MobileNetV2 expand the channels in the middle, allowing for more efficient processing. After depth wise convolutions, MobileNetV2 strategically avoids non-linear activations, this ensures that no unnecessary information is destroyed, enhancing the model's expressivity. (Sandler et al., 2019)

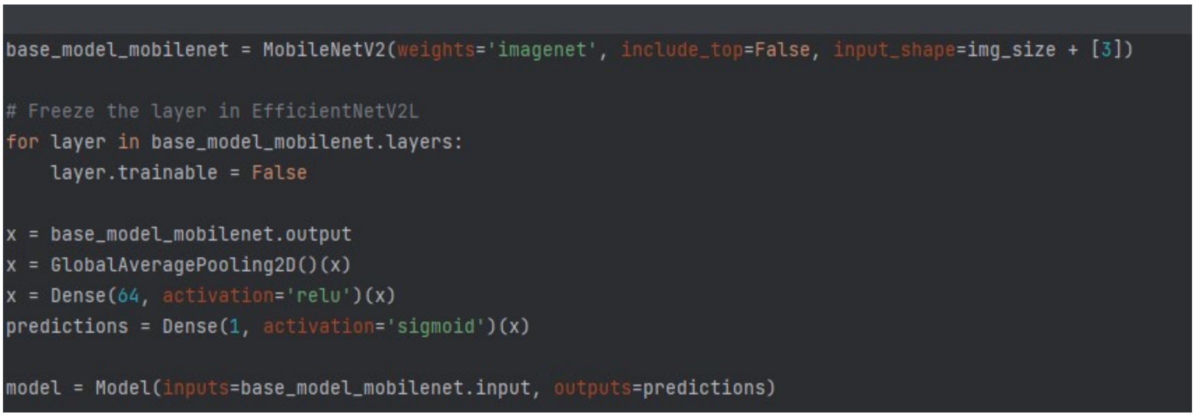
The model employs the ReLU6 activation function, which restricts activations to a range between 0 and 6. This ensures the network remains robust, especially in lower-precision environments.

MobileNetV2, with its optimized architecture, manages to achieve impressive performance with a substantially reduced number of parameters compared to many traditional CNNs. Its lightweight nature combined with its powerful capabilities makes it highly efficient, especially for on-device deployments (Kocacinar et al., 2022; N. Wang, 2022; Xiang et al., 2019)

Since its introduction, MobileNetV2 has been widely adopted in the deep learning community, especially in applications requiring real-time processing on resource-constrained devices, such as mobiles and IoT devices. Its versatility has cemented its position as a go-to model for lightweight tasks in image classification and beyond (Adhinata et al., 2021)

5.3.1. MobileNetV2 – Adam

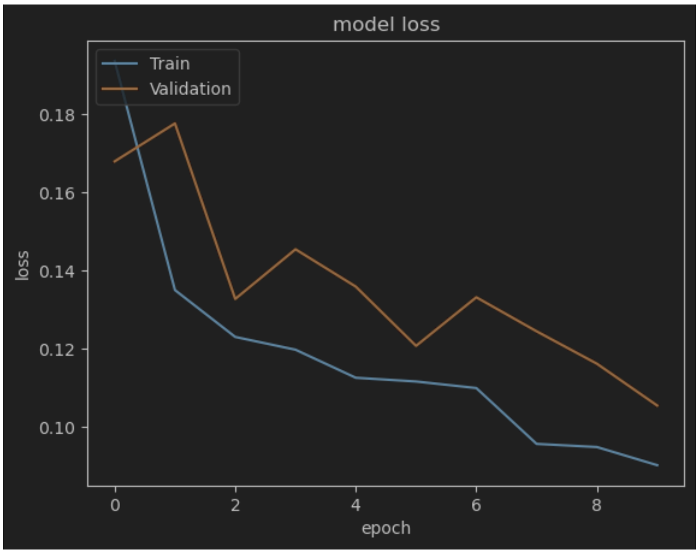
The first iteration of the Mobile NetV2 model, the model is loaded with the ImageNet training weights without the top classification layer. This is initialised with the same initial parameters as the EfficientNet models.



```
base_model_mobilenet = MobileNetV2(weights='imagenet', include_top=False, input_shape=img_size + [3])

# Freeze the layer in EfficientNetV2L
for layer in base_model_mobilenet.layers:
    layer.trainable = False

x = base_model_mobilenet.output
x = GlobalAveragePooling2D()(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model_mobilenet.input, outputs=predictions)
```

After 10 epochs we are already observing a validation accuracy of 96%, and low model overfitting.

Applying fine tuning to this model, did increase accuracy however this also increase the model generalisation issues, with the validation and training loss and accuracy widening and not converging even after 25 epochs.


model loss
Train
Validation
loss
0.18
0.16
0.14
0.12
0.10
epoch
0
2
4
6
8

6. Conclusion

In this paper, I explored various CNN architecture, with different combinations of optimisers, dropout level and fine-tuning, to create a model able to classify images, as either having the presence of mould or not. During the process a model of 98% accuracy was developed, with good generalisation abilities based on the VGG 16 architecture.

Although this provides the highest accuracy, it is important to consider that it is also the heaviest in term of space. For this reason I would recommend the MobileNet architecture, if this is to be implemented only on mobile phone, however if there's enough computational power and space, then using EfficientNetV2 might be beneficial, as this model achieved the highest accuracy.

References

Adhinata, F. D., Tanjung, N. A. F., Widayat, W., Pasfica, G. R., & Satura, F. R. (2021). Comparative Study of VGG16 and MobileNetV2 for Masked Face Recognition. *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika*, *7*(2), 230. https://doi.org/10.26555/jiteki.v7i2.20758

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing. https://doi.org/10.1007/978-3-319-94463-0

Albashish, D., Al-Sayyed, R., Abdullah, A., Ryalat, M. H., & Ahmad Almansour, N. (2021). Deep CNN Model based on VGG16 for Breast Cancer Classification. *2021 International Conference on Information Technology (ICIT)*, 805–810. https://doi.org/10.1109/ICIT52682.2021.9491631

Alippi, C., Disabato, S., & Roveri, M. (2018). Moving Convolutional Neural Networks to Embedded Systems: The AlexNet and VGG-16 Case. *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 212–223. https://doi.org/10.1109/IPSN.2018.00049

Almabdy, S., & Elrefaei, L. (2019). Deep Convolutional Neural Network-Based Approaches for Face Recognition. *Applied Sciences*, *9*(20), Article 20. https://doi.org/10.3390/app9204397

Annila, P. J., Lahdensivu, J., Suonketo, J., Pentti, M., & Vinha, J. (2018). Need to repair moisture- and mould damage in different structures in finnish public buildings. *Journal of Building Engineering*, *16*, 72–78. https://doi.org/10.1016/j.jobe.2017.12.010

Barman, R., Deshpande, S., Agarwal, S., Inamdar, U., Devare, M., & Patil, A. (2019). *Transfer Learning for Small Dataset*.

BBC. (2022, November 15). Mould in Rochdale flat caused boy's death, coroner rules. *BBC News*. https://www.bbc.com/news/uk-england-manchester-63635721

Brambilla, A., & Sangiorgio, A. (2020). Mould growth in energy efficient buildings: Causes, health implications and strategies to mitigate the risk. *Renewable and Sustainable Energy Reviews*, *132*, 110093. https://doi.org/10.1016/j.rser.2020.110093

C. K., S., C. D., J., & Patil, N. (2022). Cardamom Plant Disease Detection Approach Using EfficientNetV2. *IEEE Access*, *10*, 789–804. https://doi.org/10.1109/ACCESS.2021.3138920

Cai, J., Li, B., Yu, W., Wang, L., Yao, Y., & Wang, Y. (2020). Damp indicators in different areas of residence in different periods are strongly associated with childhood asthma and wheeze. *Building and Environment*, *182*, 107131. https://doi.org/10.1016/j.buildenv.2020.107131

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.

Cox-Ganser, J. M. (2015). Indoor dampness and mould health effects – ongoing questions on microbial exposures and allergic versus nonallergic mechanisms. *Clinical and Experimental Allergy : Journal of the British Society for Allergy and Clinical Immunology*, *45*(10), 1478–1482. https://doi.org/10.1111/cea.12601

Dangskul, W., Phattaravatin, K., Rattanaporn, K., & Kidjaidure, Y. (2021). Real-Time Control Using Convolution Neural Network for Self-Driving Cars. *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 125–128. https://doi.org/10.1109/ICEAST52143.2021.9426255

Desai, C. (2020). *Comparative Analysis of Optimizers in Deep Neural Networks*. *5*(10).

Du, C., Li, B., & Yu, W. (2021). Indoor mould exposure: Characteristics, influences and corresponding associations with built environment—A review. *Journal of Building Engineering*, *35*, 101983. https://doi.org/10.1016/j.jobe.2020.101983

Duong, M.-T., Do, T.-D., & Le, M.-H. (2018). Navigating Self-Driving Vehicles Using Convolutional Neural Network. *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, 607–610. https://doi.org/10.1109/GTSD.2018.8595533

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/BF00344251

Ganegedara, T. (2022). *Tensorflow in action*. Manning Publications.

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. In V. E. Balas, R. Kumar, & R. Srivastava (Eds.), *Recent Trends and Advances in Artificial Intelligence and Internet of Things* (pp. 519–567). Springer International Publishing. https://doi.org/10.1007/978-3-030-32644-9_36

Ginestet, S., Aschan-Leygonie, C., Bayeux, T., & Keirsbulck, M. (2020). Mould in indoor environments: The role of heating, ventilation and fuel poverty. A French perspective. *Building and Environment*, *169*, 106577. https://doi.org/10.1016/j.buildenv.2019.106577

Heseltine, E., Rosen, J., & World Health Organization (Eds.). (2009). *WHO guidelines for indoor air quality: Dampness and mould*. WHO.

Hoang, N.-D. (2018). Image Processing-Based Recognition of Wall Defects Using Machine Learning Approaches and Steerable Filters. *Computational Intelligence and Neuroscience*, *2018*, e7913952. https://doi.org/10.1155/2018/7913952

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, *148*(3), 574–591. https://doi.org/10.1113/jphysiol.1959.sp006308

Institute of Medicine (US) Committee on Damp Indoor Spaces and Health. (2004). *Damp Indoor Spaces and Health*. National Academies Press (US). http://www.ncbi.nlm.nih.gov/books/NBK215643/

Kapoor, A., Gullì, A., & Pal, S. (2022). *Deep learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models* (Third edition). Packt Publishing.

Kim, H. E., Cosa-Linan, A., Santhanam, N., Jannesari, M., Maros, M. E., & Ganslandt, T. (2022). Transfer learning for medical image classification: A literature review. *BMC Medical Imaging*, *22*(1), 69. https://doi.org/10.1186/s12880-022-00793-7

Kocacinar, B., Tas, B., Akbulut, F. P., Catal, C., & Mishra, D. (2022). A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System. *IEEE Access*, *10*, 63496–63507. https://doi.org/10.1109/ACCESS.2022.3182055

Koch, C., Paal, S. G., Rashidi, A., Zhu, Z., König, M., & Brilakis, I. (2014). Achievements and Challenges in Machine Vision-Based Inspection of Large Concrete Structures. *Advances in Structural Engineering*, *17*(3), 303–318. https://doi.org/10.1260/1369-4332.17.3.303

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, *2*. https://proceedings.neurips.cc/paper_files/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html

Lopez-Arce, P., Altamirano-Medina, H., Berry, J., Rovas, D., Sarce, F., & Hodgson, S. (2020). Building moisture diagnosis: Processing, assessing and representation of environmental data for root cause analysis of mould growth. *Building Simulation*, *13*(5), 999–1008. https://doi.org/10.1007/s12273-020-0680-8

Mascarenhas, S., & Agarwal, M. (2021). A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, *1*, 96–99. https://doi.org/10.1109/CENTCON52345.2021.9687944

Michelucci, U. (2022). *Applied Deep Learning with TensorFlow 2: Learn to Implement Advanced Deep Learning Techniques with Python*. Apress. https://doi.org/10.1007/978-1-4842-8020-1

Ministry of Housing, Communities & Local Government (Ed.). (2006). *Housing heath and safety rating system: Operating guidance: Housing Act 2004, guidance about inspections and assessment of hazards given under Section 9*. Office of the Deputy Prime Minister.

Peat, J. K., Dickerson, J., & Li, J. (1998). Effects of damp and mould in the home on respiratory health: A review of the literature. *Allergy*, *53*(2), 120–128. https://doi.org/10.1111/j.1398-9995.1998.tb03859.x

Perez, H., Tah, J. H. M., & Mosavi, A. (2019). Deep Learning for Detecting Building Defects Using Convolutional Neural Networks. *Sensors*, *19*(16), 3556. https://doi.org/10.3390/s19163556

Piecková, E. (2012). Adverse Health Effects of Indoor Moulds. *Archives of Industrial Hygiene and Toxicology*, *63*(4), Article 4. https://doi.org/10.2478/10004-1254-63-2012-2221

Raschka, S., & Mirjalili, V. (2019). *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2* (Third edition). Packt.

Regulator of Social Housing. (n.d.). *Regulator of Social Housing publishes initial findings on damp and mould in social housing*. GOV.UK. Retrieved 18 May 2023, from https://www.gov.uk/government/news/regulator-of-social-housing-publishes-initial-findings-on-damp-and-mould-in-social-housing

Salehi, A. W., Khan, S., Gupta, G., Alabduallah, B. I., Almjally, A., Alsolai, H., Siddiqui, T., & Mellit, A. (2023). A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope. *Sustainability*, *15*(7), Article 7. https://doi.org/10.3390/su15075930

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). *MobileNetV2: Inverted Residuals and Linear Bottlenecks* (arXiv:1801.04381). arXiv. https://doi.org/10.48550/arXiv.1801.04381

Shaha, M., & Pawar, M. (2018). Transfer Learning for Image Classification. *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 656–660. https://doi.org/10.1109/ICECA.2018.8474802

Silveira, V. D. C., Pinto, M. M., & Westphal, F. S. (2019). Influence of environmental factors favorable to the development and proliferation of mold in residential buildings in tropical climates. *Building and Environment*, *166*, 106421. https://doi.org/10.1016/j.buildenv.2019.106421

Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

Sridhar, B., Kiran, K. S. S., Likhitha, N., Vardhan, K. P. V. A., & Nikhil, K. V. (2022). Development of an Automatic Fruit Classification Using Hybrid Deep Learning Model for Super Markets. In V. V. S. S. S. Chakravarthy, W. Flores-Fuentes, V. Bhateja, & B. N. Biswal (Eds.), *Advances in Micro-Electronics, Embedded Systems and IoT* (pp. 181–190). Springer Nature. https://doi.org/10.1007/978-981-16-8550-7_18

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the 36th International Conference on Machine Learning*, 6105–6114. https://proceedings.mlr.press/v97/tan19a.html

Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training* (arXiv:2104.00298). arXiv. https://doi.org/10.48550/arXiv.2104.00298

Taptiklis, P., Phipps, R., Jones, M., & Douwes, J. (2022). Associations of house characteristics with indoor dampness and measured moisture: Results from three New Zealand House Condition Surveys in 2005, 2010 and 2015. *Building and Environment*, *208*, 108508. https://doi.org/10.1016/j.buildenv.2021.108508

Wang, J., & Li, Z. (2018). Research on Face Recognition Based on CNN. *IOP Conference Series: Earth and Environmental Science*, *170*, 032110. https://doi.org/10.1088/1755-1315/170/3/032110

Wang, N. (2022). Self-Driving Tour Travel Route Planning Model Construction Using Recurrent Neural Network. *Wireless Communications & Mobile Computing*, 1–7. https://doi.org/10.1155/2022/5162828

Xiang, Q., Wang, X., Li, R., Zhang, G., Lai, J., & Hu, Q. (2019). Fruit Image Classification Based on MobileNetV2 with Transfer Learning Technique. *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 1–7. https://doi.org/10.1145/3331453.3361658