A Generalisation Approach to Stock Detection Using Computer Vision

Submitted September 2024, in partial fulfilment of the conditions for the award of the degree MSc Computer Science.

Deniz Sagmanli

20539984

Supervised by Direnc Pekaslan
School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date 06/09/2024

Abstract

Generalisation has become a prevalent subject when discussing artificial intelligence models these days. Generalisation is a model's ability to undertake tasks it has barely any knowledge on. In this dissertation, the main problem that has been addressed is determining stock levels of products using images of the cabinets the products are held in. In literature, there are several possible approaches to this task. However, the common gap is that most methods either do not generalise well, or they rely on large amounts of manually annotated data. To that end, an approach has been proposed to generalise to new cabinet models or camera types, and be able to classify stock levels from these images with decent accuracy, while using as little data as possible to make the data collection and annotation process easier. The data used in this dissertation are real-world images of cabinets with ice cream in them, provided by Unilever PLC.

To overcome these obstacles, a custom workflow has been proposed, in which only two images per class (fullness level) are used to train on top of a custom neural network model that has been created. The custom model is a combination of DenseNet-201 and the Vision Transformer - B/8 models, which creates a custom dual input to be fed into both models at the same time. These two models were chosen specifically due to their superior classification capabilities in the preliminary tests that have been conducted. The DenseNet-201 model extracts the local and hierarchical features, while the Vision Transformer extracts global patterns. The features are then combined and passed through a series of custom layers to produce the predictions. Once again, this model is trained with a number of images of different fullness levels, then saved and loaded to train again with 2 images per class of the cabinet model/camera that we want to generalise to.

The novel custom approach has been tested on two scenarios: training on a new cabinet model with the same camera used to take the images as the original custom model's training dataset, and training on a new cabinet model with a different camera used to take the images than the original custom model's training dataset. Experimental results comparing the novel combination approach to several others mentioned in literature show that the custom methodology achieved higher accuracy. The limitations were that once the custom model has been further trained to generalise, its accuracy in classifying the stock levels in the original test set has decreased. Due to this decrease, for every generalisation to a new cabinet model/camera, a new AI model has to be trained using the original custom model and new data. This means that every new cabinet model would need its own AI model. Even though this is not an efficient solution, the actual goal of classifying the stock levels using cabinet images with high accuracy and small amount of annotated data has been achieved.

Keywords: DenseNet, Vision Transformer, Few-shot Learning, Feature Extractor, Generalisation

Acknowledgements

I would like to begin with extending a special thank you to my supervisor Direnç Pekaslan for his everlasting support in me; not just throughout my dissertation writing process, but also throughout my entire master's journey ever since the day we met at the beginning of the year. I would like to thank Unilever PLC for providing the data. I would like to especially thank Babak Rahi from the Unilever side for his support in my dissertation writing, giving me words of encouragement when it really counted and giving me a helping hand whenever I felt lost. I would also like to thank Isaac Triguero Velazquez for always inspiring me and showing me that I am capable of doing better, pushing me to persevere and bring out the best piece of research I can. Also, this dissertation was completed on time thanks to him giving me access to his server and letting me train my models there.

I would like to thank my parents and brother for making it possible for me to take on my master's. I would not be here, proudly authoring this dissertation if it wasn't for their moral and material support. They have always believed in me, even at times I didn't believe in myself. I would like to give my particular thanks to my beloved fiancée. Thank you for sticking with me through thick and thin, keeping me sane at times I felt stressed and discouraged, and giving me all your love, support and much more. Thank you for sacrificing your own time and energy to come visit me and boost my morale multiple times. I hope we get to grow old together and celebrate each other's many achievements to come. Finally, I would like to thank my fiancée's family members, for their emotional and intellectual endorsement in my master's journey. They have never withheld any guidance and always advocated me to strive and prosper since the day we met, for which I am forever grateful.

Table of Contents

List of Figures

List of Tables

1 - Introduction

1.1 - Motivation and Problem Statement

We are currently at the point of the Artificial Intelligence wave where we want our solutions and models to be able to generalise. Generalisation in AI is the ability of a model to be able to undertake tasks it does not have much knowledge on [1]. An example of this could be expecting a classifier model to identify images of horses, while it has mostly been trained on cats and dogs and has very little knowledge of what a horse is. In our case, we have images of cabinets that have ice cream in them, and want to classify the stock levels using these images. The generalisation task at hand is that sometimes, new cabinet models come out, or new cameras are rolled out to take images of the cabinets with. We want to be able to generalise to these circumstances. When a new cabinet is installed in a store, the store owner is asked to use cameras to take images of it and send these images. However, taking and annotating loads of images of the cabinets is a tedious task, and the owners do not like to have these cameras in their stores in the first place. Traditional image classifier models are trained with at least tens, even hundreds of thousands of labelled images to have decent accuracy, but usually struggle generalising to new tasks. The motivation here is to create a custom approach that will allow generalisation to and classification of the stock levels of new ice cream cabinet models and cameras with as much accuracy and as little manually annotated data as possible.

1.2 - Aim and Objectives

The aim of this dissertation is to be able to classify the stock levels of images of new cabinet models that are possibly taken with new and different cameras, using as little manually labelled data as possible to not be a burden to store owners and data annotators in real life scenarios. Stock levels will be classified as; “empty”, “1/4 full”, “1/2 full”, “3/4 full” and “full”. To achieve this, we will:

- Research and implement various different deep learning models and transformers to find the best ones for our classification task,
- Create a custom ensemble model using the best performing models in our experiments,
- Evaluate each model and their derivatives thoroughly and record our results,
- Formulate a custom workflow, in an event that a new cabinet model and/or a new camera comes, to classify the stock levels from images featuring these ,
- Compare our results to previous approaches in terms of accuracy.

The aim of this study is not efficiency, but rather being able to accurately detect stock levels. Deep research will need to be conducted into the state-of-the-art to be able to make an accurate comparison.

1.3 - Dissertation Overview

This dissertation will consist of 8 chapters. In chapter 1, we stated the problem and the motivation behind why it needs to be solved, along with how we plan on solving it. Chapter 2 will consist of a literature and technology review, analysing and critiquing the state-of-the-art and other approaches that have been attempted by others in solving similar tasks. In chapter 3, we will be giving some background information on the activation and loss functions, along with detailed explanations of the operating principles of models we based our final custom approach on. Additionally, we will include some information on data augmentation and how it relates to our case. Chapter 4 will constitute the methodology of how we achieved our custom approach, along with detailed explanations of our data engineering processes and the experiments we ran while aiming to find the best possible models for this task. In chapter 5, we will be showing the results of our experiments and the performance metrics of our generalisation approach, while also critically comparing our workflow to the state-of-the-art solutions. Here, we will be discussing the advantages and shortcomings of our final workflow. Chapter 6 will include the summary and concluding remarks of our study, wrapping up everything we have achieved, and how the problem has been solved. In chapter 7, we will transition into how our work can be improved upon with further research in the future. We will end on chapter 8 with the bibliography, which will include all our information and inspiration sources.

2 - Literature and Technology Review

2.1 - Computer Vision in Retail and Inventory Management:

Several approaches were proposed for using Computer Vision in retail and inventory management. One study used a novel approach to shelf management in grocery stores using Edge AI and computer vision. The approach includes using a YOLOv5 model along with Optical Character Recognition (OCR) and image classification to detect and manage stocks [2]. A system to be used on edge devices (devices with limited computational resources) has been proposed for accuracy and efficiency. The MobileNetV3 was implemented as the backbone to support YOLOv5. An annotated dataset of images of products, shelves and barcodes from the internet has been used. The main task to be carried out was looking at barcodes and detecting products to identify empty spaces where items were missing. The system classified the detected products, used OCR to read labels and alerted the staff about the empty shelves. The classification element of the system used a finetuned BEiT vision transformer which yielded very high accuracy on the validation set [2]. Although the system yielded a decent accuracy, it still had limitations. The limitations of the study are that the system included computational complexity and high memory demand, so even if the models used had high efficiency, the efficiency is still questionable to be used on edge devices. Also, the system depends on manually annotated data, which may affect its generalisability and performance. All in all, the research showed that using a vision transformer is a promising method for accuracy.

Another presented approach to supermarket inventory management was via computer vision and deep learning methods [3]. Once again, the goal was to detect missing and misplaced items and identify empty shelves. This system used a small YOLOv3 model that was optimised for real-time object detection for devices with limited computational resources. The system captured images of the shelves and compared these to a product layout that was pre-determined. It also used Contrast Limited Adaptive Histogram Equalisation (CLAHE), smoothing and sharpening to increase image quality during pre-processing. The dataset used was of images of grocery products in varying lighting and positions. Around 1,000 images per class were collected manually. The main task was to create a system that would detect these products in real time and count their numbers to know when a shelf is empty or when products are misplaced. The model proved very adequate in detecting nearly empty shelves and products but needs improvement in detecting misplaced products. The possible limitations of the paper are that the system again depends on a large amount of high-quality labelled data, which limits its generalisability. Also, the system is computationally demanding even though it was optimised for Tiny YOLOv3, so it is difficult to scale.

One study focused on detecting the defects on auto parts using an approach based on Few-Shot Learning, which is using very small amounts of data per class to increase generalisability [4].

The technology used in the study was a model based on metric learning, which is an advanced version of a Prototypical Network (ProtoNet). The model was improved using an Efficient Channel Attention (ECA) and a Transformer self-attention module. Using these techniques, the model aimed to capture channel-level information to better extract features. They used a ResNet-12 model trained on miniImageNet as the backbone to boost few-shot learning performance. The main task of the study was to detect defect in car parts using few-shot learning. They evaluated the performance using 1-shot, 3-shot and 5-shot learning, where the number designates the number of images per class. The model achieved an accuracy of 96.79% in the best-performing 5-shot setting. This result was slightly better compared to baseline models like ProtoNet and FEAT, which received 96.03% and 96.04% accuracy respectively. The limitations of this study seem to be that it relies heavily on the base dataset choice, which impacts its generalisation and may potentially limit its application to different tasks. Also, due to the attention modules, the solution seems to be extremely computationally demanding, which would affect its scalability. On the other hand, the fact that few-shot learning was used means that there is no need for vast amounts of data to train with, which coincides with our goal.

2.2 - Deep Learning Architectures and Image Classification:

A research paper explored using transformer models, commonly used for natural language processing applications on image classification tasks. The research introduced the Vision Transformer (ViT) that created patches from images and applied transformers to them to classify images, which is a novel alternative to Convolutional Neural Networks (CNNs) [5]. The approach split images into patches and turned these patches into tokens and processed these as how a natural language processing transformer would process words. The paper used 3 datasets to evaluate the performance, which were ImageNet, ImageNet-21k (a much larger version of ImageNet) and JFT-300M, which was an in-house dataset, which was almost 18 times the size of ImageNet-21k. They pre-trained the transformer on these datasets. Then, fine-tuned them further using the CIFAR-10, CIFAR-100, Oxford-IIT Pets and VTAB suite datasets to accommodate different specific tasks. The aim of the study was to demonstrate that a transformer would provide more accurate results with less computational demand than the commonly used CNNs. The model proved to use a lot less computational resources compared to CNN based models. The limitation of the research was that the model seems to need extremely large amounts of data to train with. Looking at the results, when used with smaller amounts of data, CNNs performed better, but as the training data increased, so did the performance of the ViT models. This would mean that the vision transformer on its own is not sufficient to classify images with very small amounts of data, but needs to be investigated in combination with other models, as was done in [2].

Another study compared CNNs and Residual Networks (ResNets) in terms of accuracy and robustness for satellite image classification tasks [6]. Vanishing gradient is a common problem in CNNs, which are also able to grab local features in images. However, the ResNets used in the

paper use residual blocks that have skip connections, which circumvent the vanishing gradient issue and grant the training of much deeper neural networks. Thanks to the skip connections, ResNets were able to grab long-range dependencies, which is essential in satellite imagery. The paper used a dataset of satellite images of different land cover categories. The primary task was to classify the different land cover types in each image using both CNN and ResNet and comparing the results in terms of accuracy and ability to grab local and long-range features in the images. The study proved that ResNets were superior in terms of accuracy with 98% while CNN yielded 95%. Due to the long-range dependencies the ResNet provided, it proved less disposed to overfitting. The limitation and weakness of the study would be the need for more computational resources due to the larger number of parameters being included in the model, which would limit scalability to applications requiring real-time processing. Once again, generalisation is also an issue due to the fact that the model depends on a single, large dataset. Nevertheless, the effectiveness of ResNets needs to be further studied using small amounts of data, which aligns with our goals.

Another piece of research leveraged the EfficientNet-B0 model to classify brain tumours using MRI images. The EfficientNet-B0 model was pre-trained on ImageNet and gets the name from its improved computational efficiency while providing high accuracy [7]. The model scaled the depth, width and resolution of the network with the help of a compound coefficient, allowing better generalisation to different image scales. The model's main advantage is its ability to maintain high accuracy using less computational resources. The researchers used a dataset of 3,279 MRI images, divided into classes of glioma tumours, meningioma tumours, pituitary tumours and no tumours. The main task was to classify images into one of the four classes and evaluate the accuracy. The model produced a high accuracy of 96.94%. The study was limited by the small amount of data used, as it may reduce generalisability. Also, the study did not explore how data augmentation techniques would impact the model's performance and lacked a comparison between other common models on the same task. EfficientNet also needs to be studied further for our use case as medical imagery is a very different domain than stock detection.

A novel deep learning framework to diagnose skin diseases with the aid of dermoscopy images was introduced [8]. The study introduced DP-ProtoNet, which is a prototype network that uses a dual-path to increase the accuracy of classifications. The model included ResNet-50 and DenseNet-121, which are two common neural network models. Combining these models to form their dual-path composition, the study adds a ProtoNet at the end to improve accuracy. The study used the HAM10000 dataset that includes 10,015 dermoscopy images, categorized into 7 classes. Looking at the results, the DP-ProtoNet approach showed superior results over other models it was compared to,. The possible shortcomings or limitations of the study include computational cost due to the increased complexity of the model, along with a struggle in handling imbalanced data, which is very common to see in medical image data. Using dual pathways is interesting, but

the two models used have alternatives that have more depth, which need to be explored to see if better results can be obtained.

A deep learning architecture was presented to increase the number of extracted features and circumvent the common vanishing gradient problem in neural networks that are very deep [9]. What is different from CNNs is that the DenseNet connects each layer to every other layer instead of connecting each layer to the subsequent layer. Doing this enabled each layer to have access to more feature maps and more information about the subject matter. Each layer receives input from both the previous layer and the layers before that. It includes Rectified Linear Unit (ReLU) activation and convolution. It also includes dense blocks and transition layers to include pooling (dimensionality reduction) of the data. The datasets used in the evaluation were CIFAR-10, which contains 60,000 images and 10 classes, CIFAR-100 which has 100 classes, SVHN which is a dataset of the street view house numbers, and the commonly used ImageNet. The main assignment at hand was classifying images to demonstrate the higher accuracy of the models compared to models like ResNet. On the ImageNet dataset, the DenseNet-201 model received a top-1 error of 22.58% and a top-5 error of 6.34%. Although the DenseNet model improved the flow of gradients by reducing the number of parameters, it is computationally very complex. It is not efficient at all in terms of memory, and it may not scale to three-dimensional or high-definition data. However, its use in other studies like [8] make it worthy to study further and possibly integrate into our approach.

2.3 - Few-Shot and Zero-Shot Learning Techniques:

A piece of research introduced a novel approach to address the issues that are part of zero-shot learning, generalised zero-shot learning and few-shot learning [10]. Zero-shot learning is when the model categorises the images without having any prior knowledge on any examples. Generalised zero-shot learning is when the model may or may not have prior knowledge on the class to be categorised, and the aim is better generalisation to unseen tasks compared to regular zero-shot learning. The approach discussed in the paper uses Class Adapting Principal Directions (CAPDs), that enable the access of multiple embeddings of features into the semantic space. In essence, the model generated a principal direction for every class that has been seen, and learned to bring these directions together to create the principal direction for classes that were unseen. This approach distinguishes the target class from other classes by making sure the test image's embedding is the same with the semantic embedding of the correct class. Four datasets have been used in the study, which are commonly used in zero-shot learning research. These were: aPascal & aYahoo, Animals with Attributes, SUN Attributes and Caltech-UCSD Birds datasets. The research carried out zero-shot, generalised zero-shot and few-shot learning tasks, and showed significant increase in robustness due to the CAPD component. The limitations of this paper include the fact that it is computationally very expensive, especially when scaling to very large datasets. Also, since the semantic embeddings are well-defined, if the embeddings are noisy, the model's generalisation ability would decrease immensely.

A new approach to few-shot learning which integrated self-supervised learning with an architecture of a transformer has been created [11]. The main aim was to increase the model's generalisation ability by using discriminability and diversity via self-supervised tasks. A technology called a Self-supervised Feature Fusion with Transformer (SFT) framework has been utilised. Firstly, two distinct models were pre-trained with individual self-supervised tasks. While one model was trained on a rotation prediction task, the other one was trained on a task to enhance discriminability of instances. The features generated from each model were then concatenated together. After that, the fused features were processed using a transformer to extract information specific to each individual task, thus helping the model generalise to unseen tasks. The study used miniImageNet (a smaller version of ImageNet), tieredImageNet (a smaller version of ImageNet larger than miniImageNet), and CIFAR-FS (a few-shot-specific version of CIFAR-100) datasets. These datasets are usually pretty common in few-shot learning applications when evaluating the performance of novel methods. The method was evaluated on 1-shot and 5-shot learning tasks with 5 classes to categorise images into. For 1-shot learning, the SFT model yielded an accuracy of 68.90% and for 5-shot learning, it yielded an accuracy of 83.81% on miniImageNet. The other datasets also showed similar results, and the results were observed as improvements over the state-of-the-art. It is important to note that the additional computational cost added from the SFT framework is a limitation of the study due to two models being trained at once. The model also relies on pre-training on large amounts of data, so it cannot generalise to scenarios where there is no data to be pre-trained with.

Another original method of few-shot learning that avoids traditional meta-learning associated with few-shot learning by using transfer learning with stochastic weight averaging (SWA) has been presented [12]. This approach enabled the handling of classification as well as regression assignments in few-shot learning. The main technique used is called Meta-Free Representation Learning (MFRL) and it stochastically averages weights to increase generalisation ability. SWA regularised the input to obtain low-level representations, which are the key to adequate few-shot tasks. Several datasets have been used to train and evaluate the model. Namely; miniImageNet, tieredImageNet, CIFAR-FS, FC100, Sine Waves and Head Pose Estimation datasets. The main task was to classify and regress using few-shot learning without depending on episodic meta-learning. The model proved superior to the state-of-the-art few-shot learning models on classification tasks in terms of accuracy. When it came to regression, it also provided lower mean squared error compared to the state-of-the-art. The room for improvement in this method is that the generalisability is dependent on the quality of the image input, so noise would effectively decrease accuracy when generalising. Nevertheless, as with [2] and [5], the use of transformers for image classification is an enticing subject that can be further examined with modifications.

Another study focused on demonstrating what kind of results few-shot learning techniques on image classification tasks would yield when done with different sampling sizes [13]. An

algorithm was proposed with a methodology that used coreset sampling and transfer learning using a CNN pre-trained on ImageNet. The study used ResNet as the backbone model. ResNet's ability to mitigate the vanishing gradient problem was combined with coreset sampling to select a subset of data (coreset) from the large ImageNet dataset to make the model more generalisable while reducing computational cost. To manage the data and embeddings, the method used a lightly framework. The dataset used included a total of 400 images of cats and dogs. The model achieved an accuracy of 95% when trained with only 32 samples, which was similar to the accuracy of training with 400 images. The study has shown that after reaching 16 samples, an increase in the number of samples did not improve accuracy greatly. It was important to see the effect of training with less data to cut out computational costs and improve generalisability. However, since the solution uses a ResNet model and is just training with less data, it is not very novel. Also, the method used is advertised as few-shot learning, however, it is actually just training a traditional model with less data.

An attempt has been made to address the issues of few-shot learning by using the relationship between similarity and difference in classification. A novel Similarity-Difference Relation Network (SDRN) has been introduced [14]. It improved upon few-shot learning by extracting and combining similarity and difference features between each sample. It used a Wide Residual Network (WRN-28-10) that extracted high-dimensional features. It also used a pseudo-Siamese network to grab and combine both similarity and difference metrics to categorise the data. Since this approach combines multiple models, it reduced the risk of overfitting, which is prevalent in few-shot learning due to the small amounts of data. miniImageNet and tieredImageNet datasets have been used. The research focused on 1-shot and 5-shot learning scenarios. While common methods focus only on similarity, this study aimed to improve accuracy by focusing both on similarity and difference. The SDRN model showed improvement over other few-shot learning techniques with an accuracy of 68.12% on 1-shot and 84.45% on 5-shot for the miniImageNet dataset. For the tieredImageNet dataset, it achieved 61.56% accuracy for 1-shot and 75.40% on 5-shot learning. Due to the relation extraction modules, the SDRN model invokes increased computational demand. Also, though the method reduces the possibility of overfitting, it is dependent on a simple WRN model, which most likely has trouble capturing more complex patterns and relationships in the images. The room for improvement here would be in using more complex architectures.

2.4 - Siamese Networks and Metric Learning:

One study experimented by adding a self-attention mechanism to a Siamese Neural Network, which is another common few-shot learning model effectively used in small sample sizes [15]. Using the self-attention mechanism, feature extraction was refined. The architecture was SiamRPN++, which is based on ResNet-50 and has a fusion module to integrate multi-layer feature maps. Using self-attention, the model imitated human visual attention, whereby improving accuracy via enhancing significant features. The dataset used was the Fashion MNIST

dataset that includes 60,000 images for training and 10,000 images for testing. The images are of clothing pieces. The goal of the study was to improve the Siamese Network's classification accuracy and assess whether adding the self-attention mechanism would improve performance on small datasets. Compared to the original SiamRPN++ model which had an accuracy of 94.67%, this model achieved an improved accuracy of 96.71%. Due to the addition of the self-attention mechanism, the training time increased, which poses problems in real-life situations where computational resources are limited. Also, since the study used a single dataset, the generalisability of the findings and hence, the model is limited. Testing on different datasets and adapting to generalise to new tasks could be named as areas for improvement.

Finally, a research paper presented a new method to improve the categorization of hyperspectral images (HSI) via a Siamese Convolutional Neural Network (S-CNN) [16]. An end-to-end version of the well-known S-CNN called ES-CNN has been introduced. The original S-CNN called for an extra classifier such as a Support Vector Machine (SVM) for the final step of classification. On the other hand, ES-CNN used fully connected layers and an output layer that is multi-class so that it functions as an end-to-end model. Pixel-pairs were used as input, which greatly increased the amount of training data due to the generation of many pixel combinations. Also, the model utilised a voting method that used the spatial information from adjoining pixels to improve the classification accuracy of the pixel in the centre. The dataset was the Pavia University dataset, which is a popular choice in hyperspectral imaging. The dataset includes 103 spectral bands and 9 classes. To train the model, 200 samples per class were used, adding up to over 1.6-million-pixel pairs. The ES-CNN model resulted in an accuracy of 95.96%, compared to the 88.53% accuracy of the S-CNN-SVM model. Since the model relies heavily on pixel-pairs, the model is not efficient at all but does have increased accuracy over the traditional method. Also, the voting strategy may not prove effective in more challenging classification problems.

In summary; combining multiple models in various ways, using few-shot and zero-shot learning techniques for generalisation, and trying to achieve high accuracy with limited computational resources are the common trends we see in literature. The common gaps in prior research are that the models either don't generalise with high accuracy, or they don't generalise at all. In addition to this, the approaches usually come with high computational complexity and lack a domain-specific workflow. Moreover, most studies came up with approaches that require large sums of data and lack a thorough experimentation process to decide on which models to use as the basis of combination approaches.

3 - Background Information

This section includes all the necessary information for the methodology and results to be understood. We will start off with the two models that made up our custom ensemble approach. Then we will move on to a brief review and explanation of the data augmentation steps that we did to artificially populate our dataset. Finally, we will give a some information on the activation functions we used in our custom layers, so that it is easier to understand why we used those specific ones.

3.1 - The DenseNet-201 Model

The idea behind the DenseNet-201 model is that it relies on dense connections between each layer. In a common convolutional neural network, each layer would have their individual feature maps and outputs. In DenseNet however, every layer of the network is connected to every other layer. For a given layer, the inputs are the feature maps of the layers that come before it. Since this type of behaviour requires a very close connection between layers, the model is called DenseNet [9].

DenseNet is made up of two major parts, which are dense blocks and transition layers. Dense blocks are various convolutional layers combined together in a way that each specific layer's inputs are the feature maps extracted from all of the layers that come before it. So, instead of receiving just the output of the previous layer like in a traditional convolutional neural network, a layer in the DenseNet receives both the individual outputs of the previous layers, and the initial input features concatenated together.

The mathematical expression for this is as follows:

$$X_l = H_l([X_0, X_1, X_2, ..., X_{l-1}]) \quad (1)$$

Where:

$[X_0, X_1, X_2, ..., X_{l-1}]$ are the feature maps that come from the prior layers.
$H_l(.)$ is a combination of normalisation, Rectified Linear Unit activation and a 3x3 convolution.

This helps improve the accuracy while making the model efficient in terms of accuracy. Following every dense block, we have transition layers that consist of convolutional layers for dimensionality reduction and a pooling layer to make the feature maps smaller in size [9].

DenseNet models are differentiated between each other by looking at their growth rate. This is the amount of feature maps produced by each layer and it is 32 for DenseNet-201. The block diagram of the architecture of the DenseNet-201 model can be found below in figure 1:

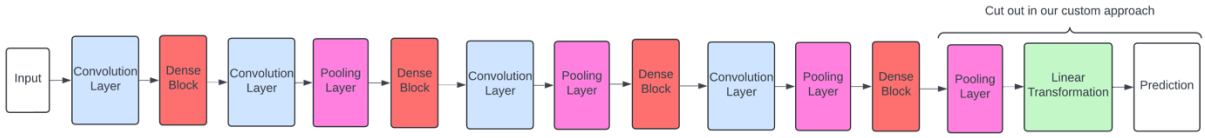

Figure 1: The architecture of the DenseNet-201 model.

It consists of 4 dense blocks. The convolution layers, pooling layers and linear transformation layers are all transition layers responsible for dimensionality reduction and feature extraction. After the fourth dense block, the pooling and linear transformation layers were cut out in our custom approach and were replaced by our custom layers.

3.2 - The Vision Transformer - B/8 Model

The Vision Transformer - B/8 splits the image into patches that do not overlap with each other, which are of 8x8 pixels in size, and then flattens the patches into a 1-dimensional vector. The model linearly projects every single patch into a vector that has a fixed length which we refer to as an embedding [5]. Since transformers are more commonly used in natural language processing, the vision transformer employs a similar logic. The embeddings are used equivalent to tokens in natural language processing applications. To keep the position information of each patch in the original image, every patch embedding is equipped with a position embedding. The main part of the vision transformer model is the transformer encoder. This part encapsulates a number of multi-head self-attention and feed-forward networks.

The multi-head self-attention (MSA) mechanism allows the model to capture the features on a global level by making every single image give attention to every other image patch.

For an embedding sequence X, the mechanism yields:

$$Attention(Q, K, V) = SoftMax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

Where:

Q, K and V are respectively the queries, keys and values that originated from X using the linear projections.
d_k is the dimensionality of keys.

The feed-forward networks use multi-layer perceptrons to feed the data to the next layer. The inputs of each layer are normalised. The entire transformer encoder mechanism allows the model to capture dependencies between image patches on a global (long-range) level. A classification

token is added to the start of each sequence. The model passes the token output through a fully connected layer and a SoftMax activation for classification [5]. The block diagram of the architecture of the ViT-B/8 model can be found below in figure 2:

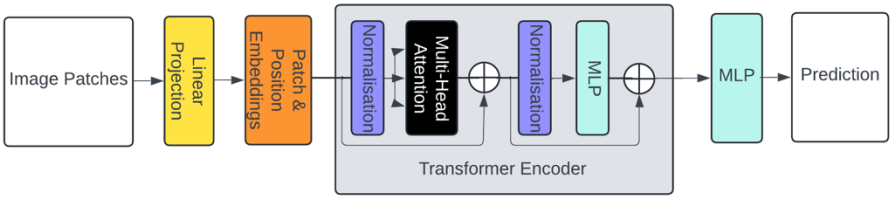

Figure 2: The architecture of the Vision Transformer - B/8 model.

The image is split into 8x8 patches and linearly embedded along with the position of each patch within the original image. Then, these pairs are passed into the transformer encoder where a token will be added to each sequence.

3.3 - Data Augmentation

Data augmentation is extremely important in deep learning approaches when images are the subject of interest [9]. Using data augmentation, it is possible to increase the diversity of the training images artificially by transforming the images in various ways and adding variety. This allows the model to generalise better and also decreases the possibility of overfitting [9]. Below are some important transformations that can be used to populate the dataset.

Rescale:
The pixel values in the images usually range between 0 and 255. Normalising the data is a common practice in image classification and allows a deep learning model to work better. Rescaling the data by 1./255 makes each pixel value between 0 and 1.

Rotation:
Sometimes, when images are taken by hand or fixed in place, images might have a small range of rotation due to human handling. A slight rotation range mimics this human error and helps with generalisation.

Width Shift:
Similarly to rotation, the placement of cameras while taking images is prone to human error. A small amount of width shift would accommodate a reasonable amount of human error that may occur across images while keeping the integrity of the main subject area of interest.

Height Shift:
Similarly to rotation and width shift, height shift is also a result of human error most of the time, which happens more often than not in a real-life scenarios. Once again, a small range height shift would cater to a reasonable amount of shifting without losing an important amount of the subject matter.

Shear:
Shear is a geometric effect that occurs when the images are taken from an angle and the object and/or objects appear skewed as a result of that. In our case, this would result in some subjects appearing to take more place than others. Adding a shear range would cater to the natural shift in these angles and provide variety without resulting in extreme distortion.

Zoom:
Zoom is something that can change with each camera, as well as the person taking the images. A small range of zoom would introduce a variation meaningful enough to improve generalisation but not so much that it distorts images.

Horizontal Flip:
Horizontal flip is when the image is mirrored on the y-axis. This has benefits for when the dataset has mirrored images, and also populates the data.

Fill Mode:
When transformations such as rotation, shifting, shear and zooming are applied to the image, the corners of the image may turn empty. In order to maintain the consistency of the images, we use fill_mode='nearest' to fill those empty pixels in with the closest pixel values.

3.4 - Activation and Loss Functions

3.4.1 - Rectified Linear Unit

The Rectified Linear Unit (ReLU) is a common activation function in deep learning approaches that allows neural networks to learn the complex associations and patterns in feature spaces. The Rectified Linear Unit is defined as follows:

$$ReLU(x) = \max(0, x) \tag{3}$$

This activation function returns zero if the input is zero or negative, and the input itself, if it is positive. This allows the output to never be negative, thus, introducing non-linearity [17]. It is especially important in image classification because getting a negative output would be unrealistic.

The mathematical representation of the Rectified Linear Unit is as follows:

$$ReLU(x) = \begin{Bmatrix} x\ if\ x \geq 0, \\ 0\ if\ x < 0 \end{Bmatrix} \quad (4)$$

Figure 3 below shows the graphical representation of the Rectified Linear Unit activation function:

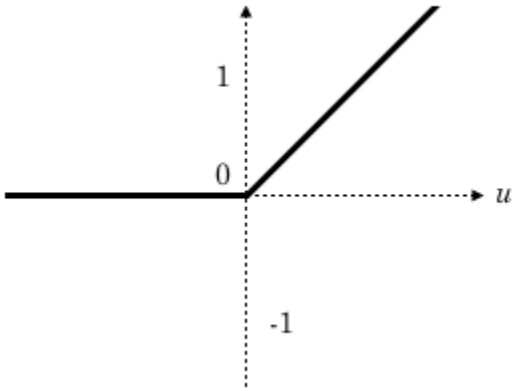

Figure 3: Graphical representation of Rectified Linear Unit activation function.

Since the Rectified Linear Unit only employs a simple comparison, it is a fairly simple function from a computational demand standpoint. Due to this, it is very efficient. Since the negative values are zeroed out, the Rectified Linear Unit only allows positive valued activations to propagate in the forward direction. This helps reduce overfitting and helps in generalisation. For positive inputs, the Rectified Linear Unit's gradient is always 1, and otherwise, it is 0. This allows the network to keep strong gradients while backpropagating. Mathematically speaking, since other activation functions like the sigmoid or hyperbolic tangent tend to get extremely small gradients, this would result in a very slow learning course [17].

3.4.2 - SoftMax Activation Function

The SoftMax activation function is another very popular function used in image classification tasks. SoftMax converts logits into probabilities where the sum of all probabilities equals 1. It has an argmax decision property. Basically, while predicting, the SoftMax activation function selects the class with the highest probability for each instance [18].

The SoftMax function's output is also differentiable. This property allows it to be used in backpropagation. The SoftMax function rewards high probabilities thanks to its exponent features. This means that it would not handle class imbalance very well as the probability of a prediction belonging to a smaller sized class would be low.

3.4.3 - Categorical Crossentropy Loss Function

The categorical crossentropy loss function has the ability to measure the distance between the probability distribution and the true labels [19]. The mathematical formula for the categorical crossentropy loss can be shown as follows:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log\left(\overline{y}_{ij}\right) \quad (5)$$

Where:

N is the amount of images per batch (32 in our case).
C is the amount of classes (5 in our case).
y_{ij} indicates if the image is correctly predicted by label j and can only take values 0 or 1.
$\overline{y}_{ij}$ is the prediction of the probability that an image i has a label j.

In this loss function, the logarithms of the output of the SoftMax function are summed up for the classes which are correct. Since the categorical crossentropy loss function is differentiable, it is useful for backpropagation.

4 - Methodology

4.1 - Data Preparation

We had access to over 300 thousand images of cabinets of various shapes and sizes. The common thing between the majority of them was that they were either vertical, or horizontal cabinets. Given this fact, we created a small dataset out of these images. We had five image classes, namely: empty, 1/4 full, 1/2 full, 3/4 full and full. For each of these classes, we gathered 40 images, so a total of 200 images. Since the goal is to generalise to a variety of cabinet models and lighting scenarios, we needed to have variety in our training dataset. Half of the images were of vertical cabinets, and the other half was of horizontal cabinets. For each orientation, we had 5 underexposed, 5 overexposed, and 10 regular images per class.

We then created a small script to determine which cabinet model we have not used in our training set using an excel file matching the serial numbers of the cabinets to their model numbers and compared them to the filenames in our dataset. Using this output, we created another dataset for a new unseen cabinet model that used the same camera as the images in the training dataset. This new dataset had 25 images, 5 for each class. The reason we had such a small amount of images was because we could only find so many images of a never-before-seen cabinet model that had enough samples for each class, so we were limited by our data.

We also created another dataset of an unseen cabinet model with a never-before-seen camera installed in it. This also had 5 images per class. The datasets of unseen cabinet models were to be used later when comparing the generalisability of the neural network models.

Since we could not find enough underexposed or overexposed images for some orientations and/or fullness levels, we used photoshop on the regular versions of those images and created overexposed/underexposed versions. Also, a total of 3 images had to be duplicated due to lack of images of vertical cabinets that were completely empty, but this would lose significance after data augmentation.

After running the images through CleanVision [20], which shows how different the images in the dataset are, we can see the variety below:

On the initial training dataset, CleanVision found 3 pairs of images that were exact duplicates of each other. These images were duplicates because of lack of quality data that were of the specific class. This was not an issue as data augmentation would reduce the significance of the duplicates immensely. It also found two images of a horizontal cabinet that were near duplicates of each other. This was due to the fact that both images had their exposure artificially reduced the same amount using photoshop. Finally, there was one image that CleanVision classified as “blurry”.

This is normal because some images were chosen on purpose to not be of the best quality so that the models would not overfit.

On the dataset of a new never-before-seen cabinet model with the same camera as the one used in the training dataset, CleanVision found a triplet and 2 pairs of images that were near duplicates of each other. This happened because the never-before-seen cabinet model had its images come from a single cabinet at a single store, so this was expected. Also, one image of an empty cabinet was found to be blurry.

On the dataset of a new never-before-seen cabinet model with a different camera than the one used in the training dataset, CleanVision only found a pair of images that were near duplicates of each other. Once again, this was expected as the images came from a single cabinet at a single store.

Before training any of the models, we did data augmentation on the training data. Looking at the initial raw images, the positions of the cameras sometimes changed due to human error. These would include slight rotations, shifts in height and width, zoom and sometimes the images were even horizontally flipped. To accommodate for these changes, we did data augmentation to generate more training data and make our models more generalisable. More details of the transformations we did can be found in section 3.

4.2 - Overview of Experiments

We had three datasets, which were explained in the designated section above. The first dataset was for training and testing a model. The second dataset had images of a never-before-seen cabinet model with the same camera as the training set. The last dataset had images of a never-before-seen cabinet with a new camera model. We took a funnel approach in eliminating methods to come to the final custom model. We applied transfer learning by chopping off the head of each model and training our own images on top of it to change the weights and biases. For easier understanding, figure 4 below shows the funnel approach we took in our experimentation until we decided on the final models to constitute our custom model.

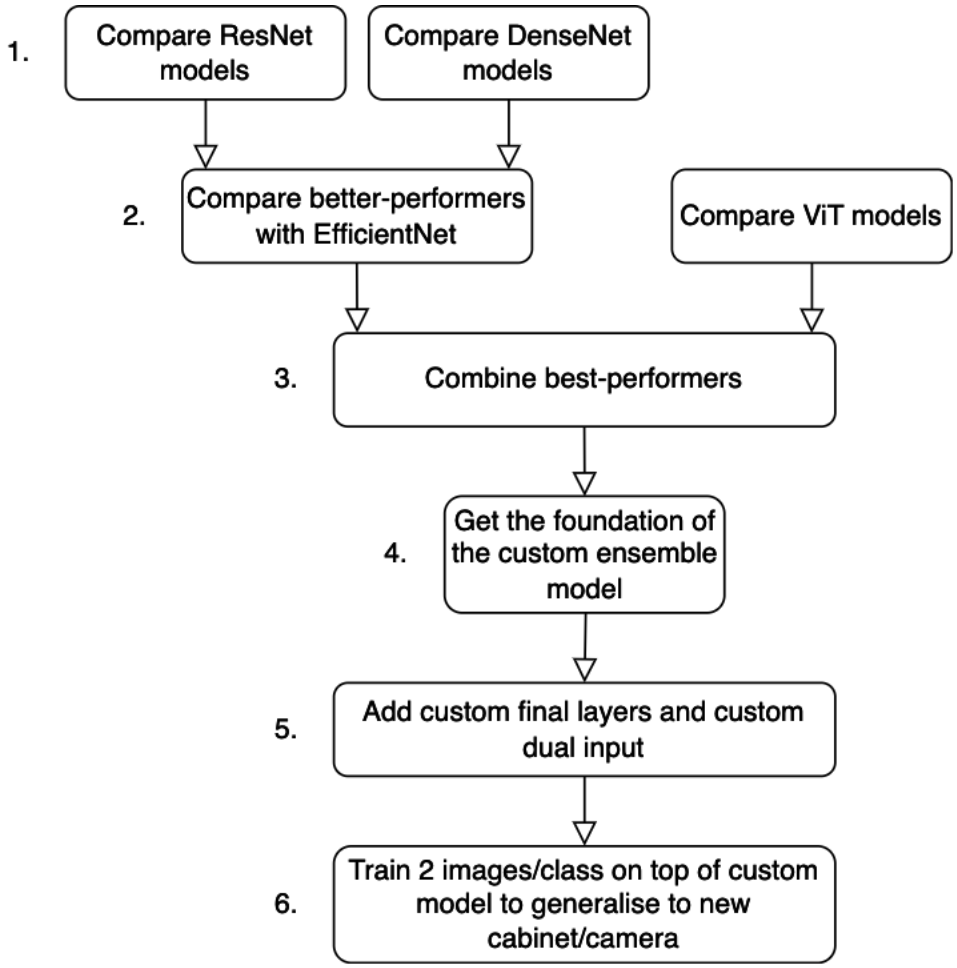

Figure 4: Representation of the funnel approach taken in the experimentation to decide which models would constitute our custom ensemble approach, along with the additional steps for generalisation.

Initially, we compared the performance of two residual network (ResNet) models within each other In terms of accuracy on the training set. We also compared two dense network (DenseNet) models within each other to find the best one. Then, we compared the best ResNet model with the best DenseNet model and an efficient network (EfficientNet) model. Due to its high accuracy, DenseNet became the winner out of the three. To see the effect of fine-tuning the DenseNet model, we tried different combinations of unfreezing different ranges of layers on the DenseNet model, and tested the results on both the training set, the unseen cabinet model with the same camera and the unseen cabinet model with the new camera to see how well it generalises, and proceeded with the best combinations.

Then, we compared two Vision Transformer (ViT) models, ViT-B8 and ViT-B16. Again, we tested on the training set, the unseen cabinet model with the same camera and the unseen cabinet model with the new camera. ViT-B8 Became the better performer due to having higher granularity on its image patches.

We then created a custom ensemble approach where we combined the features extracted from DenseNet and ViT together to increase accuracy. Since both DenseNet and ViT were feature extractor models, the expectation was to take the local features from the DenseNet model and the global features from the Vision Transformer to improve accuracy. We tried three combinations: DenseNet and ViT, DenseNet and ViT with early stopping callback and fine-tuned DenseNet then ViT with early stopping callback. We saved the best performing DenseNet + ViT custom model.

We compared our custom model with the previous best-performing DenseNet and ViT models to see how it would match up. Once again, we tested on the 3 datasets. Our custom approach was not the clear winner yet, but it was looking promising. To further-increase its generalisation ability, we trained it with 2 images per class from the unseen cabinet model with the same camera dataset and 2 images per class from the unseen cabinet model with the new camera dataset. We compared the testing results to the previous performance of the leading DenseNet and ViT models and showed that our accuracy became the best of all models. Since this approach was inspired by few-shot learning on the account that it was aiming to get the maximum accuracy and generalisation with as little data as possible, we compared it to common few-shot learning models as well. These were, Prototypical Networks (ProtoNet), Matching Networks, Siamese Networks and Relation Networks [4][8][14][15][16]. As such, our custom approach bested these models on its ability to classify the new cabinet model with the same camera and the new cabinet model with the new camera as well.

4.3 - Creating the Models

To try each model, we defined functions to create the model first. The following notions were all kept the same throughout each model's definition:

Input Shape:
We used an input shape of 224, 224, 3 across all models. This input shape is common to be used on models pre-trained on the ImageNet dataset. 224, 224 means a pixel size of 224x224. Since the models we used all have been pre-trained on images that were re-sized to 224x224 pixels, we re-sized our images in order to be certain that we made effective use of the pre-trained weights. The 3 corresponds to the amount of colour channels in the image. Since the images have pixels that are combinations of red, green and blue (RGB), we used the number 3.

Global Average Pooling 2D:
For the ResNet, DenseNet and EfficientNet models, we used an extra layer called GlobalAveragePooling2D after the output of the base model. The base models generate an output that is very high-dimensional. To contextualise, the DenseNet-201 pre-trained model's base model output is a 7x7x2048 feature map [9]. Using global average pooling, we took the average of each feature map, thus reducing the output to a single, size 1x1x2048 vector. In machine

learning and deep learning applications, the curse of dimensionality is a prevalent subject that causes many problems to the model. Having a high-dimensional output would require more computational resources to process all the information. Calculation of distances, training of data, hyperparameter searches and prediction times start to escalate in an exponential manner. Even error calculation becomes a tedious task that pushes the limits of hardware. In addition, the need for more data increases to accommodate the amount of dimensions, otherwise, the density of the data points fails to be maintained, and it becomes impossible to make predictions or classifications without immense amounts of data, which is the exact opposite of what we were trying to achieve. On top of all this, having an extremely high amount of dimensions would mean a higher chance of overfitting. Using global average pooling, we reduced the amount of parameters in comparison to the connected layers of the model. This straightforwardness aided us in reducing the chances of overfitting, principally, as we were working with small datasets due to the nature of our problem definition. As global average pooling took the average of each map instead of simply converting the information into a 1-dimensional vector, we retained our important importation for classification, while reducing the size of the output.

Rectified Linear Unit:
For the penultimate layer of each model, we used the Rectified Linear Unit as the activation function. Using the Rectified Linear Unit, we introduced non-linearity into our neural networks due to its mathematical nature. This allowed our neural networks to learn the complex associations and patterns in our feature spaces. It was especially important for our application because we did not want to get a negative output for our image classification.

Since the negative values were zeroed out, the Rectified Linear Unit only allowed positive valued activations to propagate in the forward direction. This was enticing in our image classification task as it reduced the risk of overfitting and increased the generalisation possibility. It also allowed our network to keep strong gradients while backpropagating. Mathematically speaking, since other activation functions like the sigmoid or hyperbolic tangent tend to get extremely small gradients, they would result in a very slow learning course. This would mean that the weights are not updated, and the model would stop learning. Thankfully, this did not happen to us, as our results suggested. The other extremity to this would be that since the Rectified Linear Unit has no bounds (i.e. it can go to infinity) for positive inputs, it could lead to unstable activations, which thankfully also did not happen to us.

SoftMax Activation Function:
For our output layer for the predictions, we went with the SoftMax activation function. We did this because SoftMax converted logits into probabilities where the sum of all probabilities equals 1. Since the problem at hand was a multi-class classification one, we used the argmax decision property of the SoftMax activation function. Basically, while predicting, the SoftMax activation function selected the class with the highest probability for each instance. The SoftMax function

rewards high probabilities thanks to its exponent features. This means that it would not handle class imbalance very well as the probability of a prediction belonging to a smaller sized class would be low. To circumvent this, the dataset we used was already balanced. Also, for a problem with a large number of classes, the computations would be expensive as the function is calculated for each class. In our problem, this was irrelevant as we only had five classes.

Adam Optimiser:
We chose the Adaptive Moment Estimation (Adam) optimiser as it is a prevalent and simple optimiser to be used in learning. We also chose a learning rate of 0.001 to make our training fast while still allowing the model to learn in each epoch. The Adam optimiser allows faster convergence by adjusting the learning rate individually for every parameter. It does this by looking at the magnitude of the gradients for each parameter [21].

Categorical Crossentropy Loss Function:
Since we had multiple classes to deal with, we used the categorical crossentropy loss function. We went with this specific loss function as it has the ability to measure the distance between the probability distribution and the true labels which were one-hot encoded. Since the categorical crossentropy loss function uses the probability distribution to measure the distance, we used it in conjunction with the SoftMax activation function mentioned above.

Due to the fact that the categorical crossentropy loss function is differentiable, it was useful for backpropagation. In the event that one class dominated the dataset and caused class imbalance, the model would be biased and would have predicted that class with higher probability, skewing the results of the loss function. Since we used a dataset that was already balanced, we circumvented this possibility.

4.4 - Training the Models

For each model's training, we split our 200 images into 160 and 40. The 40 would be left out as unseen test data to avoid data leakage. We did 5-fold cross validation on the remaining 160 images. We split the data into 5 folds of 32 images and trained on 4 folds while testing on the remaining one. After doing this for 5 iterations, we calculated the mean of the accuracies and root mean squared errors. The root mean squared error was just a precautionary measure in case two models had equal accuracies during comparison. We then tested on the unseen data. Each model was trained for 15 epochs, but the few-shot learning approaches were trained for 10 as they were already overfitting. The ViT models had an early stopping patience of 5, the custom approach had a patience of 8 and our custom workflow had a patience of 3. We decided on these numbers through numerous trials and picked the ones that yielded the best performance.
A summary of our entire comparison process can be seen in figure 5 below:

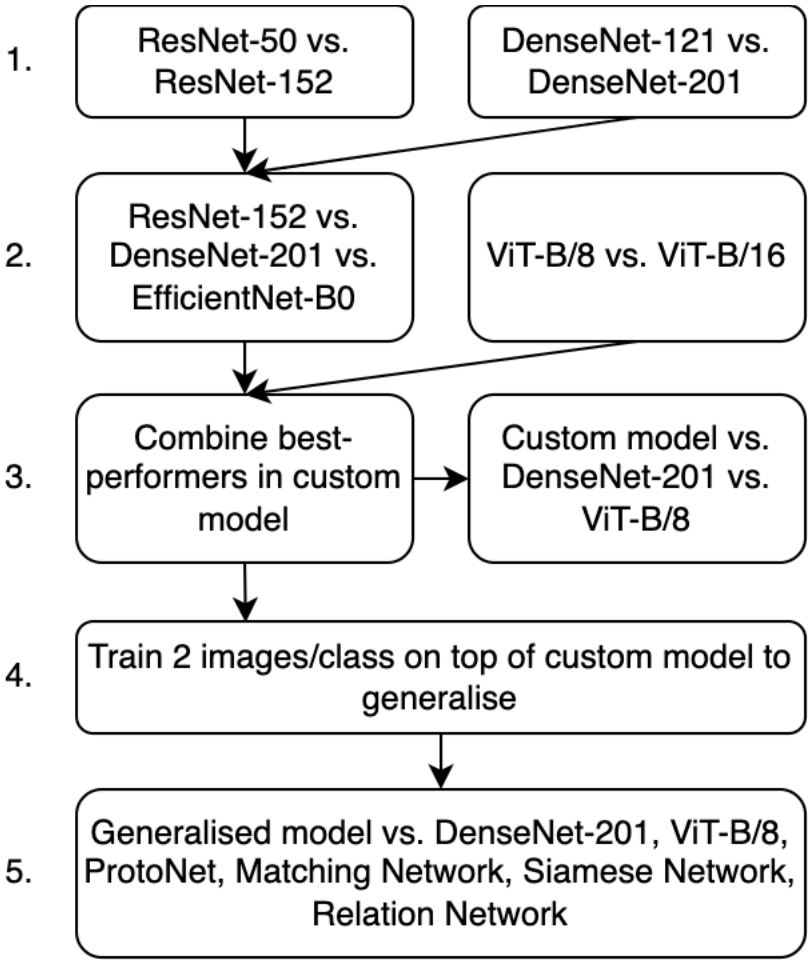

Figure 5: The summary of the entire model comparison process.

First, we compared two residual networks, which are ResNet-50 and ResNet-152, ResNet-152 performed better. The main difference between the two models is the amount of layers. ResNet-50 has 50 layers, while ResNet-152 has 152 layers. ResNet-50 is preferred for its small model size and fast implementation, while ResNet-152 provides higher accuracy using more computational power. Since this study focused on accuracy instead of efficiency, we went with ResNet-152.

Then, we compared DenseNet-121 and DenseNet-201, DenseNet-201 performed better. The difference between the two models is again the amount of layers they have. Also, DenseNet-201 has more than twice the amount of parameters that DenseNet-121 has, which makes it more computationally expensive, but more accurate [9].

To have some variety, we compared ResNet-152 with DenseNet-201 and EfficientNet-B0. EfficientNet, as the name suggests, uses less computational resources at the cost of accuracy [7]. DenseNet-201 performed best, so we proceeded with that.

Until this point, we only looked at accuracy and root mean squared error with our training dataset, which was split into training and testing sets to make the elimination process easier.

After that, we compared each model and combination on 3 tasks. The first one was, classifying the fullness level of a cabinet in the initial dataset (the 40 images we set aside earlier). The second one was classifying the fullness level of a never-before-seen cabinet model with the same camera as the one used in the initial dataset. The last one was classifying the fullness level of a never-before-seen cabinet model with a new camera.

Then, for DenseNet-201, we looked at different fine-tuning combinations to see unfreezing which layers yields the best performance on the training set, a never-before-seen cabinet model with the same camera, and a never-before-seen cabinet model with a new camera and proceeded with the best options.

Then, we compared two different Vision Transformer (ViT) models, B/16 and B/8. The difference between the two models is that ViT-B/16 uses 16x16 image patches while ViT-B/8 uses 8x8 image patches [5]. Due to B/8 using smaller patches, it creates more tokens, thus using more memory [5]. We compared ViT-B/8, ViT-B/16, ViT-B/8 with early stopping callback and ViT-B/16 with early stopping callback. We again proceeded with the best options.

After that, we created a custom ensemble approach where we connected the DenseNet-201 and ViT-B/8 models together. We tried three different combinations. DenseNet and ViT, DenseNet and ViT with early stopping callback and fine-tuned DenseNet and ViT with early stopping callback. Here, fine-tuning meant using the best combination of DenseNet-201 with certain layers unfrozen, as mentioned above. We tested these on the original training dataset, a never-before-seen cabinet model with the same camera and a never-before-seen cabinet model with a new camera. We proceeded with the best option out of the three and saved that model.

Then, we went on to create a workflow where we want to be able to correctly classify a new cabinet model's fullness. In a real-life scenario, the cabinet owners do not like cameras put into their cabinets. They also do not want to go through the trouble of sending images of their cabinets, and manually annotating large amounts of data is a cumbersome process. To bother them as little as possible, we want to be able to classify the fullness level of a cabinet with as little annotated data as possible. We ask a cabinet owner to send two images per class of their cabinet, which is a total of 10 images. Using those 10 images, we load our previously saved custom model and train on top of it. After training, the model is now able to correctly classify the new cabinet model even with a new camera. This approach was inspired by few-shot learning given that the aim is to train with as little data as possible, while being able to correctly adapt to a new task, which is classifying a new cabinet model possibly with a new camera in this case. We used early stopping so that the model would not overfit. Ideally, the workflow is that we

train the small number of images of a new cabinet on top of the saved custom model each time we want to generalise to a new cabinet model and/or a new camera model.

Finally, we compared our final results to the best combinations of DenseNet, ViT and the DenseNet + ViT for each task as well as common few-shot learning approaches, which are ProtoNet, Matching Network, Siamese Network and Relation Network.

4.5 - Custom Model Architecture

In literature, there have been instances of transformers being used in conjunction with other models to classify images [2][4][11]. There has also been a prior paper that combined the features from two separate models to improve accuracy [8]. However, the approach combining the features from two separate models used ResNet-50 and DenseNet-121, both of which have denser and more complex alternatives to provide higher accuracy. In addition, as per our research, there has never been an approach combining the features of a dense network model with a vision transformer. The opportunity for novelty here is that vision transformers focus on global features and dense models focus on local features, meaning they can complete each other [5][9]. To that end, the custom ensemble model we created is a combination of the standard DenseNet-201 model, the Vision Transformer ViT-B/8 and two custom dense layers on top. The detailed architectures and operating principles of the two models were discussed in section 3. The custom model architecture can be seen in figure 6 below:

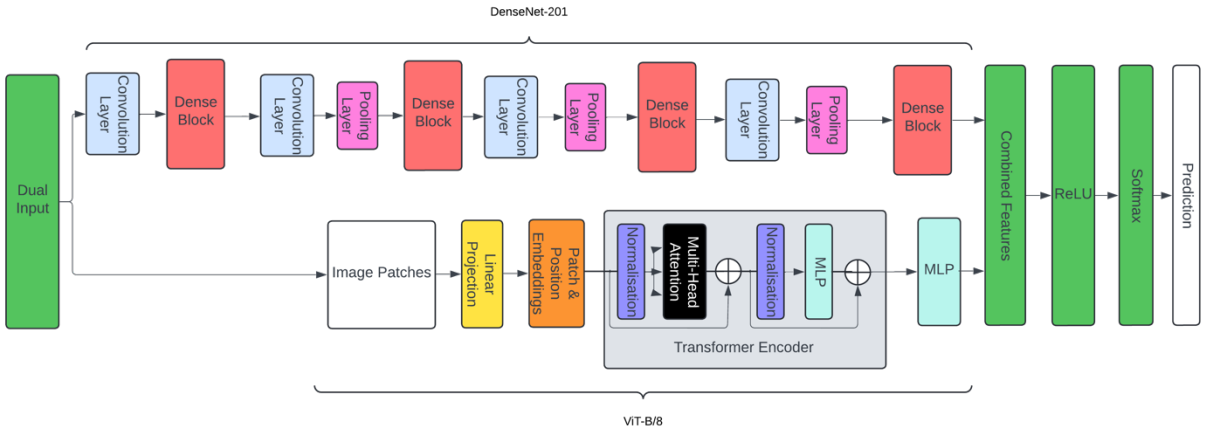

Figure 6: The architecture of the custom ensemble model. For reference, the green blocks are the custom steps we added.

A custom dual input generator feeds two copies of the same image to both the DenseNet-201 and the ViT-B/8 models. The final pooling, SoftMax and the probability conversion layers are removed from the DenseNet-201 model. No layers were removed from the Vision Transformer model as its output is already in the format we wanted to use in our custom layers. The extracted

features are combined and sent through a rectified linear unit and a SoftMax activation layer to get the final predictions.

The DenseNet and Vision Transformer models work simultaneously. We provided a custom function to generate dual inputs to be run through both models at the same time. Since both models were feature extractor models, we let them extract features from the input and combined those features.

To extract the features from the two models, we removed the global average pooling layer, the SoftMax layer and the final fully connected layers that convert the outcome into class probabilities from the DenseNet-201 model. We did not need to remove any layers from ViT-B/8 as its output is already in the format we wanted to use in our custom layers. After gathering the extracted features from both the DenseNet-201 model and the ViT-B/8 model, we concatenated them, and we ran the combined features through our dense layer that featured a Rectified Linear Unit activation feature. After that, we ran the activated output through our prediction (output) layer that featured a SoftMax activation function for probabilistic interpretation. This gave us the final predictions of our custom model. The advantage in combining the outputs of these two models was that we made it possible to combine the parts of the models that they specifically focus on and bring them together. The DenseNet model's feature extractor brings out hierarchical and local features using its convolutional layers [9]. On the other hand, the vision transformer's feature extractor focuses on global contextual features using its mechanisms of self-attention [5]. By bringing these features together, we were able to attend to both the local, hierarchical details and the global dependencies. We saved this model so that it could be used in our proposed generalisation workflow that we explained below.

4.6 - The Workflow of Generalisation

In an event where we want to be able to classify the stock level of a cabinet that just got released, or basically, any cabinet model that the custom model has not been trained on, we have proposed a workflow to generalise. This workflow also works when we want to classify an existing cabinet model that has been equipped with a new camera that is different from the ones we have used for the training of the custom model, or a new cabinet model with a new camera altogether. The workflow is as follows:

When the cabinet is installed in the store, we ask the store owner to send us just 10 images of the cabinet. This would be 2 images for each class (fullness level). This is advantageous because not only do store owners want to go through the trouble of organising and sending images, but they also usually do not even like to have the cameras in the first place. This means that we are achieving to be as small of a burden for them as possible. Then, we load the custom model we have saved earlier and train the two images per class on top of it. This would update the weights

and biases, allowing the stock level of the new cabinet model and/or new camera model to be able to correctly be classified. The workflow is shown on the flowchart in figure 7 below:

Figure 7: The workflow for when we want to generalise to a new cabinet model and/or a new camera model.

We first install the new cabinet or new camera into the store, then we get 2 images per class from the owner. Then, we load and train our images on top of the custom model. After that, we are generalised to the new cabinet and/or camera.

5 - Results and Discussion

5.1 - Initial Experimentation

Table 1: Comparison of ResNet-50 with ResNet-152, along with comparison of DenseNet-121 with DenseNet-201.

	ResNet-50	ResNet-152	DenseNet-121	DenseNet-201
Accuracy	0.25	0.25	0.78	0.60
RMSE	1.69	1.61	0.47	0.79

As we have mentioned in the 'Overview of Experiments' section, we used root mean squared error as a secondary measure in case the accuracies of two models were tied. Between the two ResNet models, we can see that the accuracies were equal at 0.25, so we looked at RMSE and went with ResNet-152 as it resulted in a smaller error of 1.61. Since none of the ResNet models looked like they would yield an acceptable result by the end of our project, we just picked the one with better accuracy and RMSE. On the other hand, we did something different while choosing the DenseNet model. This test was just done on the initial training set. Since the goal was generalisation, and as both DenseNet models yielded somewhat acceptable accuracies and RMSEs which looked like they could be improved to acceptable values by the end of the project, we deliberately picked the one with less accuracy to make it more generalisable to never-before-seen scenarios. Due to this reason, between the two DenseNet models, we went with DenseNet-201, which resulted in an accuracy of 0.60 and an RMSE of 0.79 instead of DenseNet-121, which resulted in an accuracy of 0.78 and an RMSE of 0.47. We also did this because we knew beforehand that the architecture of DenseNet-201 has more layers (hence, the number 201), so it would have a higher chance of capturing more complex relationships between features [9].

Table 2 below shows the comparison of the two picked options from the above experiment, compared with EfficientNet-B0 to add variety into the experiment. This experiment corresponds to step 2 of our experimentation process.

Table 2: Comparison of ResNet-152 with EfficientNet-B0 and DenseNet-121.

	ResNet-152	EfficientNet-B0	DenseNet-201
Accuracy	0.25	0.23	0.60
RMSE	1.61	1.38	0.79

The results of ResNet-152 and DenseNet-201 are the same as table 1. EfficientNet-B0 received an accuracy of 0.23 and an RMSE of 1.38. Since ResNet-152 and EfficientNet-B0 showed no promise of improving to an acceptable level, we decided to progress with DenseNet-201 for the further experiments and for our custom ensemble model.

5.2 - DenseNet-201 Comparisons

Table 3 below shows the results of the comparison of DenseNet-201 with different layers unfrozen to see which combination yields the best results on the original training set. This experiment, along with the results in tables 4 and 5 lie between step 2 and 3 of our experimentation process. In essence, we tried to find the best-performing combination of the DenseNet-201 model to be part of our custom approach.

Table 3: Results of DenseNet-201 with different layers unfrozen on the original training set.

	All Frozen	Last 50 Unfrozen	Last 100 Unfrozen	Last 150 Unfrozen	Last 200 Unfrozen	Last 150 to 200 Unfrozen
Accuracy	0.60	0.57	0.75	0.47	0.65	0.60
RMSE	0.79	0.92	0.63	0.77	0.71	0.69

When we kept all of the layers frozen, we received an accuracy of 0.60 and an RMSE of 0.79. With only the last 50 layers unfrozen, we received an accuracy of 0.57 and an RMSE of 0.92. With the last 100 layers unfrozen, we had an accuracy of 0.75 and an RMSE of 0.63. When we unfroze the last 150 layers of the model, we had an accuracy of 0.47 and an RMSE of 0.77. Unfreezing the last 200 layers brought about an accuracy of 0.65 and an RMSE of 0.71. Finally, we wanted to try and see what freezing the 50 layers between the last 150 and last 200 layers would yield, and we got an accuracy of 0.60 and an RMSE of 0.69. This experiment concluded that for the original training dataset, DenseNet-201 with the last 100 layers unfrozen brought the best results in terms of accuracy. From this point on, throughout our experiments, we used this version of the DenseNet-201 as our “fine-tuned” version. When we compared the different versions of our combination approach, the fine-tuned version included this DenseNet-201 model with the last 100 layers unfrozen.

Table 4 below shows the results of the comparison of the predictions of DenseNet-201 with different layers unfrozen to see which combination yields the best results on the dataset of a new cabinet model that had the same camera as the one on the original training set.

Table 4: Results of DenseNet-201 with different layers unfrozen on a new cabinet model with the same camera.

	All Frozen	Last 50 Unfrozen	Last 100 Unfrozen	Last 150 Unfrozen	Last 200 Unfrozen	Last 150 to 200 Unfrozen
Accuracy	0.68	0.56	0.52	0.60	0.68	0.56
RMSE	0.57	1.04	0.77	0.63	0.75	0.66

We have seen here that after training with the original training set, when we asked the model to predict on this new dataset of a never-before-seen cabinet model with the same camera, we got the best results with all the layers frozen. Specifically, an accuracy of 0.68 and an RMSE of 0.57. Last 200 layers unfrozen also received the same accuracy, but since it had a higher RMSE at 0.75, we decided that the better model was DenseNet-201 with all layers frozen on this task. This might be because adjusting layers would mean fine-tuning the model, and fine-tuning would mean focusing the model on a single task (the original dataset in our case), so no fine-tuning may have resulted in better generalisation to this dataset.

Table 5 below shows the results of the comparison of the predictions of DenseNet-201 with different layers unfrozen to see which combination yields the best results on the dataset of a new cabinet model that had a different camera from the one on the original training set.

Table 5: Results of DenseNet-201 with different layers unfrozen on a new cabinet model with a different camera.

	All Frozen	Last 50 Unfrozen	Last 100 Unfrozen	Last 150 Unfrozen	Last 200 Unfrozen	Last 150 to 200 Unfrozen
Accuracy	0.64	0.68	0.68	0.64	0.52	0.40
RMSE	0.85	0.57	0.57	0.69	0.69	0.77

The results of this experiment showed that DenseNet-201 with the last 50 layers and the last 100 layers unfrozen are both equally the best performing options. They both received accuracies of 0.68 and RMSEs of 0.57. However, since the version with the last 100 layers unfrozen was also the best performer on the original dataset, and since we decided to use that as the fine-tuned version of DenseNet-201 in the following experiments, we decided to keep that as our choice for this experiment. In the previous experiment, we suggested that the version with all the layers frozen performed best because no fine-tuning might have led to better generalisation. However, when predicting a new cabinet model with a new camera, we have seen that fine-tuning brought

about better results. This might have happened because the last 50-to-100 layers of the DenseNet-201 model might have affected weights and biases responsible for understanding the angles and colour coding of the new camera better.

5.3 - Vision Transformer Comparisons

Table 6 below shows the results of the comparison of ViT-B/8 with ViT-B/16 and both models with early stopping callbacks to see which combination yields the best results on the original training set. This experiment, along with the results in tables 7 and 8 correspond to step 2 in our experimentation procedure, where we tried to find the best combination of the Vision Transformer to be part of our custom approach.

Table 6: Results of ViT-B/8 vs. ViT-B/16 and both models with early stopping callbacks on the original training set.

	ViT-B/16	ViT-B/16 Early Stopping	ViT-B/8	ViT-B/8 Early Stopping
Accuracy	0.60	0.60	0.57	0.70
RMSE	1.08	0.88	0.89	0.80

When testing on the original dataset, we can see that ViT-B/8 with early stopping is the clear winner out of all the combinations with an accuracy of 0.70 and an RMSE of 0.80. Since a version of ViT-B/8 turned out to be the better version, we decided to use this model for the combination approach, and when we tried the combination approach with early stopping, we referred to ViT-B/8 with early stopping.

Table 7 below shows the results of the comparison of predictions of ViT-B/8 with ViT-B/16 and both models with early stopping callbacks to see which combination yields the best results on the dataset of the new cabinet model with the same camera. This experiment corresponds to step 2 of our experimentation process.

Table 7: Results of ViT-B/8 vs. ViT-B/16 and both models with early stopping callbacks a new cabinet model with the same camera.

	ViT-B/16	ViT-B/16 Early Stopping	ViT-B/8	ViT-B/8 Early Stopping
Accuracy	0.72	0.56	0.76	0.56
RMSE	0.53	0.66	0.69	0.66

When we tested on a new cabinet model with the same camera, we saw that ViT-B/8 is the clear winner with an accuracy of 0.76 and an RMSE of 0.69. The most plausible explanation to this performance would probably be similar to DenseNet-201's explanation for the new cabinet model with the same camera. Since applying early stopping would create a closer fit to the training data, and since a closer fit to the training data would mean less generalisation ability, this might be the reason why ViT-B/8 became the higher performer over its early stopping counterpart. When it comes to ViT-B/8 versus ViT-B/16, the size of the image patches being smaller in the B8, thus having more patches and more information than B16 probably gave B8 the edge.

Table 8 below shows the results of the comparison of predictions of ViT-B/8 with ViT-B/16 and both models with early stopping callbacks to see which combination yields the best results on the dataset of the new cabinet model with a new camera.

Table 8: Results of ViT-B/8 vs. ViT-B/16 and both models with early stopping callbacks a new cabinet model with a new camera.

	ViT-B/16	ViT-B/16 Early Stopping	ViT-B/8	ViT-B/8 Early Stopping
Accuracy	0.68	0.72	0.60	0.80
RMSE	0.66	0.52	0.63	0.45

Here, we saw that once again, ViT-B/8 with early stopping became the best performer, yielding an accuracy of 0.80 and an RMSE of 0.45. Using early stopping callbacks on the original data allowed better generalisation to the new cabinet model with new camera on not just ViT-B/8, but also on ViT-B/16. A plausible explanation to these results might be that due to the operating principles of the vision transformer explained in section 3.3, the model probably looked at each patch's fullness level as opposed to the whole image, leading to the model being less affected by the changing camera and generalising better. Since the early stopping worked better in the original training dataset, it would make sense for the results to transfer to the dataset with the new cabinet model and new camera.

5.4 - Custom Approach Comparisons

Table 9 below shows the results of different combinations of DenseNet-201 and ViT-B/8 combined on our custom ensemble approach, trained on the original dataset. Early stopping refers to ViT-B/8 with early stopping and fine-tuned refers to the last 100 layers of the DenseNet-201 model being unfrozen. From here on out, the results refer to steps 3 and 4 of our experimentation process, and further testing and comparisons.

Table 9: Results of the DenseNet-201 + ViT-B/8 combination approach on the original training dataset.

	DenseNet-201 + ViT-B/8	DenseNet-201 + ViT-B/8 Early Stopping	Fine-tuned DenseNet-201 + ViT-B/8 Early Stopping
Accuracy	0.66	0.70	0.70
RMSE	0.65	0.88	0.32

After this experiment, we saw that although it was a tie between DenseNet-201 + ViT-B/8 with early stopping and fine-tuned DenseNet-201 + ViT-B/8 with early stopping at 0.70 accuracy, the RMSE results broke the tie, and the latter became the better of the two with an RMSE of 0.32. This type of result is intuitive as we have seen that fine-tuning the DenseNet-201 model itself gave the best result in its own test and ViT-B/8 with early stopping was the best in its own test when testing on the original dataset. Hence, combining the two would logically bring about the best result on the original set as well.

Table 10 below shows the results of different combinations of DenseNet-201 and ViT-B/8 combined on our custom ensemble approach, trained on the dataset of a never-before-seen cabinet model with the same camera.

Table 10: Prediction results of the DenseNet-201 + ViT-B/8 combination approach on a new cabinet model with the same camera.

	DenseNet-201 + ViT-B/8	DenseNet-201 + ViT-B/8 Early Stopping	Fine-tuned DenseNet-201 + ViT-B/8 Early Stopping
Accuracy	0.88	0.76	0.80
RMSE	0.35	0.49	0.56

We see again that the non-fine-tuned version with no callbacks performed the best of all. We are starting to see a pattern that when we train the models on the original training dataset, adding callbacks and unfreezing specific layers gives the best results. However, when we test those models on the dataset of new cabinet model with the same camera, we see that simpler versions generalise the best. Here, simple refers to no callbacks or fine-tuning. Once again, we see that DenseNet-201 + ViT-B/8 gets the best result with an accuracy of 0.88 and an RMSE of 0.35.

Table 11 below shows the results of different combinations of DenseNet-201 and ViT-B/8 combined on our custom ensemble approach, trained on the dataset of a never-before-seen cabinet model with a new camera.

Table 11: Prediction results of the DenseNet-201 + ViT-B/8 combination approach on a new cabinet model with a new camera.

	DenseNet-201 + ViT-B/8	DenseNet-201 + ViT-B/8 Early Stopping	Fine-tuned DenseNet-201 + ViT-B/8 Early Stopping
Accuracy	0.76	0.64	0.64
RMSE	0.60	0.60	0.84

The results show that DenseNet-201 + ViT-B/8 generalised better to the new cabinet model with the new camera with an accuracy of 0.76 and an RMSE of 0.60. As opposed to using the two models separately, we are seeing that combining the models without any fine-tuning or early stopping yielded the best result on the new cabinet model and new camera. This is the type of result we would expect to see in ensemble approaches compared to single models, where simpler combinations result in higher accuracy.

Tables 12, 13 and 14 below compare our best results for the custom approach to the best results for DenseNet-201 and ViT-B/8 on their own in the three datasets consecutively, so that it can clearly be seen how the custom ensemble approach compares to using the models individually. The best results of each approach on the designated test were used for a fair comparison.

Table 12: Comparison of fine-tuned DenseNet-201 + ViT-B/8 early stopping (best result for custom approach) with DenseNet-201 last 100 layers unfrozen (best result for DenseNet-201) and ViT-B/8 Early stopping (best result for ViT) on the original dataset.

	DenseNet-201 Last 100 Unfrozen	ViT-B/8 Early Stopping	Fine-tuned DenseNet-201 + ViT-B/8 Early Stopping
Accuracy	0.75	0.70	0.70
RMSE	0.63	0.80	0.32

Table 13: Comparison of DenseNet-201 + ViT-B/8 (best result for custom approach) with DenseNet-201 all layers frozen (best result for DenseNet-201) and ViT-B/8 (best result for ViT) on the dataset of new cabinet model with the same camera.

	DenseNet-201 All Layers Frozen	ViT-B/8	DenseNet-201 + ViT-B/8
Accuracy	0.68	0.76	0.88
RMSE	0.57	0.69	0.35

Table 14: Comparison of DenseNet-201 + ViT-B/8 (best result for custom approach) with DenseNet-201 last 100 layers unfrozen (best result for DenseNet-201) and ViT-B/8 Early stopping (best result for ViT) on the dataset of new cabinet model with a new camera.

	DenseNet-201 Last 100 Unfrozen	ViT-B/8 Early Stopping	DenseNet-201 + ViT-B/8
Accuracy	0.68	0.80	0.76
RMSE	0.57	0.45	0.6

As can be seen from the results in tables 12, 13 and 14, our custom approach is only the best when it comes to predicting new cabinet models with the same camera as the training set. This means that it is not unanimously the best out of the three approaches. As per our workflow, we took DenseNet-201 + ViT-B/8 as it was the combination of the custom approach that showed the best performance for both of our generalisation predictions and saved it. After loading the custom model, we trained 2 images per class from both of our generalisation datasets individually, and recorded the test results below:
Training 2 images per class on top of combination approach:

- Testing on new cabinet model with the same camera: accuracy = 0.91, RMSE = 0.18
- Testing on new cabinet model with a new camera: accuracy = 0.89, RMSE = 0.17

This shows that in a real-life scenario, we can ask a cabinet owner to send 10 images of their cabinet (2 for each class), train with those on top of the combination approach and be able to generalise to that entire model/camera.

5.5 - Few-Shot Learning Comparisons

Since our workflow of training with 2 images per class on top of the custom model was inspired by the concept of few-shot learning, specifically, 2-shot learning, we ran tests on four common

few-shot learning approaches to see if they could match up to our approach. Tables 15, 16 and 17 show these approaches' results on the original training set, dataset of new cabinet model with the same camera and dataset of new cabinet model with the new camera, respectively.

Table 15: Results of common few-shot learning approaches, trained on two images per class from the original training dataset.

	ProtoNet	Matching Network	Siamese Network	Relation Network
Accuracy	0.32	0.27	0.36	0.60
RMSE	1.548	1.691	1.344	1.372

Table 16: Results of common few-shot learning approaches, trained on two images per class from the dataset of a new cabinet model with the same camera.

	ProtoNet	Matching Network	Siamese Network	Relation Network
Accuracy	0.44	0.24	0.32	0.32
RMSE	0.89	1.48	1.15	1.26

Table 17: Results of common few-shot learning approaches, trained on two images per class from the dataset of a new cabinet model with a new camera.

	ProtoNet	Matching Network	Siamese Network	Relation Network
Accuracy	0.32	0.12	0.12	0.24
RMSE	1.18	2.04	2.49	1.62

For the original dataset, we are seeing that a Relation Network became the best performer with an accuracy of 0.60 and an error of 1.372. For the dataset of a new cabinet model with the same camera as the original dataset, Prototypical Network (ProtoNet) became the best performer with an accuracy of 0.44 and an error of 0.89. For the dataset of a new cabinet model with a new camera, Prototypical Network (ProtoNet) again became the best performer with an accuracy of 0.32 and an error of 1.18.

Whether the camera was the same as the original dataset or not was not relevant here as we did not train the models on the original dataset and expected them to predict on the other sets. To be able to compare our custom workflow to these approaches, we had to create the experiments to fit the working principles of each approach. Hence, for the few-shot learning approaches each experiment required training the models on the new dataset, without giving it any prior knowledge of the original dataset. When answering the question of why these methods achieved

less accuracy compared to our workflow, it is important to note that our approach used 2 models trained on ImageNet, which is one of the largest image datasets available online. We just trained our small datasets on top of the models to adjust the weights and biases. On the other hand, the few-shot learning approaches had to actually train with 2 images per class. Nevertheless, we can see that our custom workflow of training 2 images per class from the desired set to generalise to that entire cabinet model/camera yielded higher accuracy in all the tests we have done.

The final tests to do were to test the models' performance on the initial 40 unseen images from the original dataset, after being trained on 2 images per class to see if they could still predict them as well as the original custom approach. The results can be found below:

After training on 2 images per class from the dataset with the new cabinet model and the same camera, and testing on the 40 unseen images from the original training dataset of the custom model, we received an accuracy of 0.60 and an RMSE of 0.95.

After training on 2 images per class from the dataset with the new cabinet model and a new camera, and testing on the 40 unseen images from the original training dataset of the custom model, we received an accuracy of 0.53 and an RMSE of 1.24.

In summary, our custom model did not do a decent job of predicting the initial 40 images after training on 2 images per class. This means that once generalised to a new cabinet model and/or camera, we cannot use the same model to predict the old models. This is efficient in neither computational resources, nor workflow. However, the goal of this study was not efficiency (as in most models discussed in this dissertation), it was creating an approach achieving the highest amount of accuracy when generalising to new cabinet models and cameras, with as little data as possible. When building the other models and approaches mentioned above, scientists and engineers also think about large-scale applications and the speed at which the approach can be deployed [2][3][7][13]. This requires them to sacrifice some of the potential accuracy they can get by making accuracy the only focus. We have seen this in the above test results. In the final analysis, we would expect the generalisation approach to be used such that the user would save the loaded custom model, train two images per class of the unseen cabinet and/or camera on top of the loaded model and use the newly trained model on images coming from that cabinet and/or camera only. This would mean training, saving and archiving a custom model for each cabinet model to come, which is a tedious task, but would allow for impressive accuracy in stock level detection.

6 - Conclusion

Since we were inspired by few-shot learning when coming up with the idea of training with 2 images per class, we compared our results against Prototypical Networks, Matching Networks, Siamese Networks and Relation Networks, along with the original DenseNet-201 and ViT-B/8 models. Testing on a new cabinet model with the same camera as the one used in the original training set of the custom model, we received an accuracy of 0.91 and an RMSE of 0.18. When testing on a new cabinet model with a new camera, we received an accuracy of 0.89 and an RMSE of 0.17. These results showed higher accuracy and lower RMSE than the aforementioned techniques.

A possible explanation for the results could be that since we are using very powerful models pre-trained on ImageNet which includes millions of images as the basis of our custom model, we started with a strong foundation. In addition, since the DenseNet-201 brings out local and hierarchical features, and the ViT-B/8 model brings out global patterns, combining them most likely created a comprehensive approach and brought out a holistic view of the images to be classified [5][9]. On top of all of this, we trained the custom model with two images of the new cabinets per class, which resulted in the weights and biases being fine-tuned to the new information, leading to an improved performance. On the other hand, due to their nature, common few-shot learning approaches had no prior information related to the cabinets, apart from being trained on two images per class, resulting in them being inferior to our approach.

The custom model workflow also has its drawbacks and limitations. We have seen that, after training the 2 images per class on top of our custom model, the model's ability to predict the previous images set apart from the original custom model's training set decreased. This would imply that to achieve the optimal accuracy while generalising to a new cabinet model or camera, we would need to train with new images on top of our custom model and save the resulting model, keeping it exclusive to that cabinet model/camera. This requires time and archiving. Since this research was focused solely on accuracy, we can say that we achieved our objective of generalisation in spite of this drawback.

In this research, our aim was to create an approach to identify the stock levels of ice cream cabinets using image classification. The main obstacle was that we needed an approach to generalise to new cabinet models and new cameras that take the images, as traditional deep learning approaches usually are not good at generalisation. Our objective was to maximise accuracy while using as little data as possible, as the images are collected from store owners and store owners do not like having to take and send lots of images all the time. Also, annotating vast amounts of data manually is a very time-consuming task.

In our approach, we created a custom workflow. To do so, we needed a custom deep learning model that was a combination of two distinct models. In order to decide on the models, we experimented on different ResNet, EfficientNet and DenseNet models by training our own dataset on top of the pre-trained models. Eventually, we decided on DenseNet-201 as the first part of our custom model due to the promising results it showed. Then, we experimented with different Vision Transformer models, again training our own images on top of the pre-trained models and ended up choosing ViT-B/8 due to its superior classification accuracy. We combined the two models together and fed a dual input to both models at the same time. When both models extracted their individual features from the input, we combined the features and ran them through our custom neural network layers to get the final classifications. We saved this custom model. Then, for the custom workflow, we proposed getting two images per class from the cabinet owners, which is easier for them, and training with those images on top of our saved custom model. This way, we effectively needed just 10 images to generalise to a new cabinet model and/or new camera.

7 - Further Research

Looking at our results and discussion section, we can say that we have achieved an acceptable amount of accuracy. However, we needed to create a cumbersome process for generalising to new cabinet models and/or cameras. Our goal was never efficiency, we were aiming for accuracy, while being as little of a strain to the store owners and data annotators as possible. That being said, further research can aim to improve upon our work by creating a workflow that would not need to be retrained each time we want to generalise to a new task. Also, our workflow requires us to have a specific custom model trained for each cabinet model. This can be improved upon by creating an approach where a single AI model can generalise to all cabinet/camera models.

Another thing that could be aimed for in later works could be creating an entirely new artificial neural network, or a transformer, instead of modifying and concatenating existing models. This is a long and involved process, so it would most likely take longer than a dissertation period to achieve.

Finally, if future work chooses to pursue a similar approach to ours, that is, having a separate model for each new cabinet type, they can look for ways to automatise the process of model training, and create a user interface to achieve this without having to code. In addition, they can add a model management system to organise trained models more easily.

8 - Bibliography

[1] I. Triguero, D. Molina, J. Poyatos, J. Del Ser, and F. Herrera, “General Purpose Artificial Intelligence Systems (GPAIS): Properties, definition, taxonomy, societal implications and responsible governance,” *Information Fusion*, vol. 103, p. 102135, Mar. 2024. doi:10.1016/j.inffus.2023.102135

[2] A. Savit and A. Damor, “Revolutionizing retail stores with computer vision and Edge Ai: A novel shelf management system,” *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, May 2023. doi:10.1109/icaaic56838.2023.10140947

[3] M. A. Majdi, B. Sena Bayu Dewantara, and M. M. Bachtiar, “Product Stock Management Using Computer Vision,” *2020 International Electronics Symposium (IES)*, Sep. 2020. doi:10.1109/ies50839.2020.9231673

[4] J. Xu and J. Ma, “Auto parts defect detection based on few-shot learning,” *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, May 2022. doi:10.1109/cvidliccea56201.2022.9823993

[5] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Proc. ICLR 2021, arXiv:2010.11929v2 [cs.CV], Oct. 2020.

[6] E. D and N. P. Bhavani, “An effective DNN based ResNet approach for satellite image classification,” *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, Sep. 2023. doi:10.1109/icosec58147.2023.10276330

[7] V. Goutham, A. Sameerunnisa, S. Babu, and T. B. Prakash, “Brain tumor classification using EfficientNet-B0 model,” *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Apr. 2022. doi:10.1109/icacite53722.2022.9823526

[8] L. Kong, L. Gong, G. Wang, and S. Liu, “DP-protonet: An interpretable dual path prototype network for medical image diagnosis,” *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Nov. 2023. doi:10.1109/trustcom60117.2023.00390

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected Convolutional Networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. doi:10.1109/cvpr.2017.243

[10] S. Rahman, S. Khan, and F. Porikli, “A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning,” *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, Nov. 2018. doi:10.1109/tip.2018.2861573

[11] J. Y. Lim, K. M. Lim, C. P. Lee, and Y. X. Tan, “SFT: Few-shot learning via self-supervised feature fusion with Transformer,” *IEEE Access*, vol. 12, pp. 86690–86703, 2024. doi:10.1109/access.2024.3416327

[12] K. Chen and C.-G. Lee, “Meta-free few-shot learning via representation learning with weight averaging,” *2022 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2022. doi:10.1109/ijcnn55064.2022.9892722

[13] K. Dedhia, M. Konkar, D. Shah, and P. Tawde, “A proposed algorithm to perform few shot learning with different sampling sizes,” *2022 IEEE Fourth International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, Jan. 2022. doi:10.1109/icaecc54045.2022.9716609

[14] C. Changhu and Y. Peng, “Similarity-Difference Relation Network for few-shot learning,” *2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID)*, May 2021. doi:10.1109/aiid51893.2021.9456570

[15] Y. Liu *et al.*, “Self-attention based Siamese neural network recognition model,” *2022 34th Chinese Control and Decision Conference (CCDC)*, Aug. 2022. doi:10.1109/ccdc55256.2022.10034228

[16] M. Rao, L. Tang, P. Tang, and Z. Zhang, “ES-CNN: An end-to-end Siamese convolutional neural network for hyperspectral image classification,” *2019 Joint Urban Remote Sensing Event (JURSE)*, May 2019. doi:10.1109/jurse.2019.8808991

[17] “ReLU activation function explained,” Built In, https://builtin.com/machine-learning/relu-activation-function#:~:text=The%20rectified%20linear%20unit%20(ReLU)%20or%20rectifier%20activation%20function%20introduces,activation%20functions%20in%20deep%20learning. (accessed Aug. 31, 2024).

[18] “Softmax function,” Engati, https://www.engati.com/glossary/softmax-function (accessed Aug. 31, 2024).

[19] “What is cross-entropy loss function?,” 365 Data Science, https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/ (accessed Aug. 31, 2024).

[20] Cleanlab, “Cleanlab/cleanvision: Automatically find issues in image datasets and practice data-centric computer vision.,” GitHub, https://github.com/cleanlab/cleanvision (accessed Aug. 31, 2024).

[21] D. Wei, “Demystifying the adam optimizer in machine learning,” Medium, https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e (accessed Sep. 5, 2024).