The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Applications Convolutional Neural Networks on Publicly Available Property Images

by

Nurfahimah binti Mohd Ghazali

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

July 2021

Supervised by
Dr Nicole Augustin and Dr Mike Allerhand

Executive Summary

Convolutional neural networks (CNNs) are very widely used in image classification. There are numerous available structures built to keep improving the performance of the models. The aim of this project is to determine whether these networks, whether built from scratch or utilizing weights from pre-trained models are able to classify publicly available property images. The final report mainly splits the findings into two parts: model comparison and final model evaluation. Baseline CNNs with varying hyperparameters and pre-trained models with ImageNet weights are fitted to the training set, then compared in their accuracies and speed. It was found that even the training accuracy for baseline CNNs were capped at roughly 50%. The experiments were continued on pre-trained models to utilize the knowledge from a different source task, but tailored to the dataset at hand when extracting higher level features.

In the second part of the report, MobileNet's architecture and final performance after tweaking the network has been presented. Despite being the model with the best performance in comparison to the models included in the experiments, it only achieved 55% accuracy. The findings in the misclassified images aligns with the issues with the data quality of the dataset when a sample of images and their corresponding labels were presented. It can be concluded that this poor data quality issue must be handled before moving forward with analysis.

Acknowledgments

I'd like extend my warmest appreciation to the supervisors and fellow colleagues who asked questions and given answeres that were very helpful to the writing of this report. To my dear friend, Zaty, thank you for always having my back and doing what you can to help me out, even on your holiday. Thank you to my housemates and Iqmal, for supporting me throughout the whole process of experimenting and writing.

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Nurfahimah binti Mohd Ghazali
Matriculation Number: S2464388
Title of work: Applications Convolutional Neural Networks on Publicly Available Property Images

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct).

Signature ....[illegible]........

Date ....7/7/2023........

Contents

List of Tables

List of Figures

1 Introduction

Premiums for landlord and house insurances takes into account the risk factors related to the property, such as the size of the property, number of rooms and bathrooms, age of constructions as well as whether the property is in an urban or rural area. The purpose of this project is to provide the starting point in estimating those risk factors, which is first determining the type of property from publicly available images obtained from Google maps. With these estimations, premiums calculation can be made much more accurate, where the premium ranges can be used in promoting insurance products in advertisements or pamphlets. Further, the outcome from the model chosen in this project will directly aid in possibly detecting fraud in insurance applications.

Therefore, extracting useful features from the images for the property classification. Particularly, convolutional neural networks (CNNs) are notable in literature when it comes to image processing, where classification is implemented using the addition of fully connected layers. There are two main ways to go about using CNNs, which are either building the layers and training the weights completely based off of the training set or utilising weights from models that have been pre-trained on very large datasets and fine-tuning some of the final layers to the dataset. The performances of the two methods on different model structures will be compared and contrasted to see whether the purpose of the project can be achieved, and a CNN model with high accuracy can be found.

2 Dataset

The dataset provided consists of the folder `street_view_juny12` of `.png` images, where the names include the property's ID. The classes for said images can be found in `properties_juny12.csv`. There are 5 unique classes, `Detached`, `Semi-detached`, `Flat`, `Terraced` and `Unknown` for 18,779 images. The classes are obtained from rightmove links that the properties were once listed from. This explains the presence of the `Unknown` class that takes up roughly 13% of the dataset. The class is excluded from the analysis since including it will not provide any useful insight on the classification accuracy. The properties are split into 70%, 15% and 15% for training, validation and testing respectively. New directories are created, `trainval` to contain the training and validation sets, and `test` to contain the test set. Each directory consists of 4 class-titled files.

Figure 1 shows a sample of the images and their corresponding labels displayed above them. This gives us an idea of the quality of the Google Maps snapshots when matched to their labels. There are two main issues with the images that should be noted. Some images do not seem to match their labels, which seems prevalent for the `Semi-Detached` class. For example, the second image in the first row looks more like a Terrace, whereas the first image in the third row looks like a Detached than Semi-Detached. Further, it is unclear what property some of the images show due to the surrounding. There is no telling what the first image in the first row or the last image of the fourth row is showing. Fortunately, there are also snapshots that clearly show properties and enough of the surrounding area to recognise the label, especially the second and third images in the fourth row which are Detached and Flat respectively.

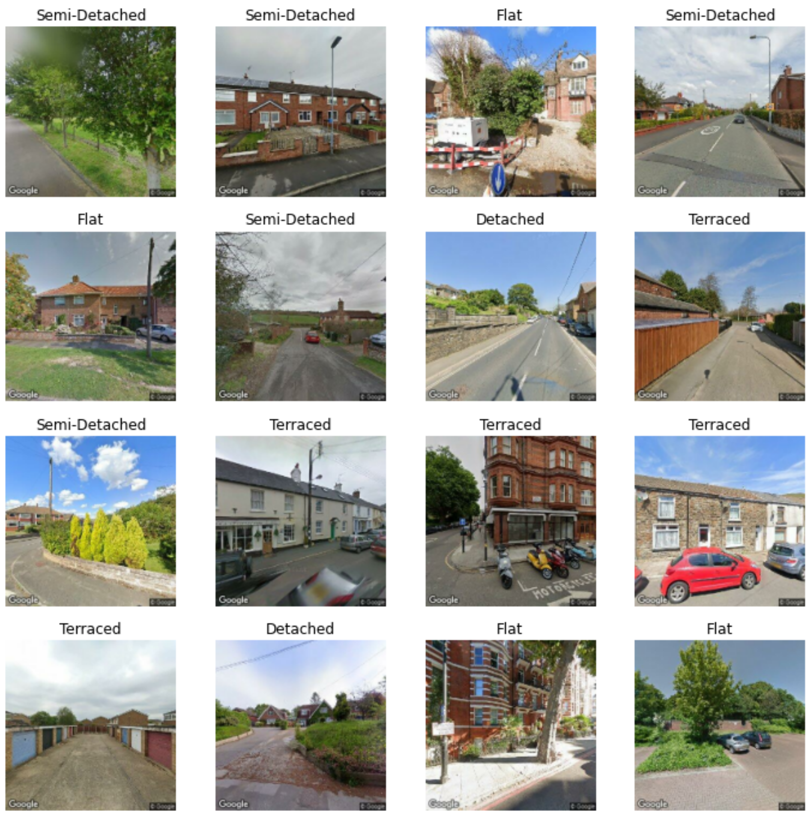


Figure 1: Sample images from dataset

2.1 Preprocessing

The data is loaded in batches of 32 images to increase the speed of training. This allows the model to only need to store errors for the 32 images and updates parameters after completing the whole batch, instead of updating for every image in the training set. To optimise the data performance, `.cache()`

is used in the pipeline which ensures the dataset does not become bottleneck during training. Then, only the train set is shuffled. At the end of the pipeline, `.prefetch()` is applied to fully utilise the GPU by overlapping data preprocessing for the next batch while the model is executed on a certain batch.

Although roughly 13,000 images are available for training, additional layers are added to the models to augment the images. The images are set to 180x180 when read in as tensors for the basic CNNs. The sizes vary for the pre-trained models depending on their requirements. The images are allowed to be randomly flipped horizontally and zoomed in or out by 10%. The random rotation factor is set to 10%, where the images will be randomly rotated up to 36° clockwise or counter-clockwise. This allows the network to predict different patterns to increase the accuracy. The settings are depicted in Figure 2.

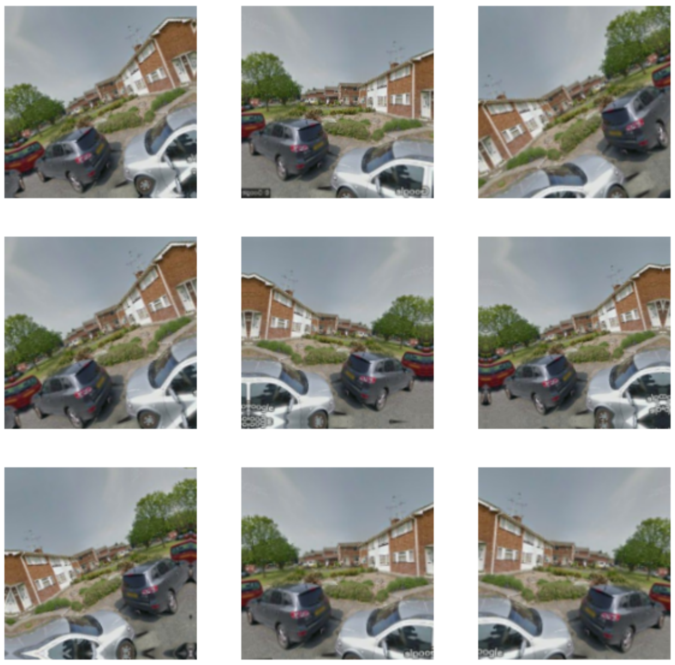


Figure 2: Data augmentation

3 Methodology

In order to compare the performance of different models, the optimizer, loss function and the activation in the final layer is fixed. With the multi-class classification in mind, the best overall optimizer is the 'Adam' optimizer [4]. This is due to it maintaining a per-parameter learning rate as the combination of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) which deals with issues of sparse gradients and non-stationary objects respectively [4]. To quantify the loss, categorical cross-entropy loss is generally used for multi-class problems [2]. For the output layer, the Softmax function is used as the activation so that the output are in the range of 0 to 1 and sums up to 1. These are the probabilities used to classify the images.

The metrics used to compare the networks are the validation accuracy and loss. Choosing the architecture that is suitable for a dataset depends on the tradeoff between accuracy and speed of model training.

3.1 Basic Convolutional Neural Networks

A typical CNN for image classification is made up of three main layers: convolution and pooling to extract features, and fully connected layers maps the features to their labels [6]. In the convolution layer, a 2D kernel is passed across the pixels of the images. Each element of the kernel is multiplied by the corresponding element of the input tensor and summed, resulting in an element of the output on the feature map. These outputs are then passed to the non-linear activation. To reduce the dimensionality of the feature maps, max pooling layers are added to downsample the its elements and retains only the maximum value in each patch. Finally, one or more fully connected layers, more widely known as dense layers map the features extracted to the probabilities for each class. Each dense layer is followed by an activation function, where the final one is set to have the same number of output as the number of classes. The predicted class is then class with the highest probability.

The two main hyperparameters used to compare the accuracies between the models are the number of convolution layers and dense layers. The experiments are done for multiple different networks, increasing the hyperparameters up to 3 layers each to increase the model complexity. This means the trainable parameters goes up to 17 million counts. Other hyperparameters, the number of filters and kernel size in the convolution layers, pooling size and number of neurons are varied to improve the performance of the model. Neural networks, however, are notorious for being data-hungry and tend to overfit the training set. Additional layers can be added such as a batch normalization layer between the convolution and pooling layers or dropout layers between the dense layers [6]. These are added only if there are signs of overfitting when the training and validation accuracies are compared. Epochs are set to 15 for all the models.

3.2 Transfer Learning

Transfer learning refers to the technique of adapting a model that has been trained on a classification task to improve the performance of a different classification task [5]. This allows us to utilise models that have been trained on extremely large datasets without the full computational costs in training all the layers since the pre-trained weights can be repurposed. The approach chosen in the experiments conducted in this project is the pre-trained model approach where a base model is chosen to develop the feature map, as a starting point [5]. Examples of this include MobileNet and EfficientNet. These models vary in their structure and returns distinct sets of features.

In image classification, ImageNet is one of the most prominent datasets that is used in research. The dataset spans 1000 classes with training and validation on over 1.2m and 50K images respectively, as well as 100K separated for testing [1]. The weighted layers trained on this dataset is leveraged to extract features from the google maps snapshots by freezing them during training. The extracted features are then fed into a shallow classifier. Three main ways this can be done are fully connected neural networks, global average pooling or support vector machines. Global average pooling differs from fully connected layers as the features are averaged and fed directly into the output layer with the softmax activation. This is the method adopted for the experiments. Unfreezing the last layers of the model allows fine-tuning of the pre-trained models to the dataset. This is set to the last 30 layers for all the models. Epochs are set to 15 with all the layers in the pre-trained models frozen, and an additional 15 for fine-tuning. Once a model has been chosen, the base learning rate and fine-tuned layers can be adjusted to further improve the accuracy.

4 Experiments and Results

4.1 Basic Convolutional Neural Networks

The hyperparameters are varied following [7] guidelines when comparing the time complexity of different structures of neural networks. This allows the sizes to range from around 500KB up to 64MB where the trainable parameters reach over 16 million counts. The training and validation performances for the different architectures are compiled into Table 1.

Model	*No. of Conv. Layers*	*No. of filters*	*Pooling size*	*Kernel size*	*No. of Dense Layers*	*Neurons in Dense Layers*	*No. of parameters*	*Training & validation accuracy and loss*
A	2	8, 16	all 2x2	all 3x3	1	4	130,996 (511KB, 2s val)	train: 45.67%, 1.182 val: 42.00%, 1.214
B	2	16, 32	2x2, 1x1	7x7, 5x5	1	4	274,404 (1.05 MB, 6s)	train: 46.77%, 1.179 val: 44.81%, 1.206
C	2	32, 64	2x2, 1x1	5x5, 3x3	1	4	539,332 (2.06MB, 6s val)	train: 48.01%, 1.170 val: 44.56%, 1.210
D	2	8, 16	all 2x2	all 3x3	2	64, 4	2,075,316 (7.92MB, 2s)	train: 47.29%, 1.170 val: 43.16%, 1.234
E	2	32,64	2x2, 1x1	all 3x3	2	64, 4	8,314,116 (31.72MB, 5s val)	train: 48.34%, 1.165 val: 45.29%, 1.255
F	2	32,64	all 2x2	all 5x5	3	128, 64, 4	16,651,140 (63.52MB, 9s val)	train: 47.87%, 1.156 val: 44.81%, 1.207
G	3	16, 32, 64	2x2, 2x2, 1x1	all 3x3	2	128, 4	3,989,156 (15.22MB, 3s val)	train: 48.43%, 1.141 val: 44.89%, 1.201
H	3	16, 32, 64	2x2, 2x2, 1x1	all 3x3	3	128, 64, 4	3,997,156 (15.25MB, 3s val)	train: 50.41%, 1.115 val: 45.57%, 1.197

Table 1: Comparison of Basic CNNs

The highest training accuracy obtained from the basic CNNs is at 50.41% which is achieved by model H, with 3 convolutional layers and 3 dense layers. Model H is also the structure achieving the highest validation accuracy, slightly above 45%. Dropout or batch normalization layers have not been added to any of the models as there are no signs of overfitting. Overall, the training and validation accuracy seems to be capped at 50.5% and 46% respectively, where the models barely accurately classifying the images by chance. The low performance on the training dataset may be a sign of underfitting. This gives motivation to adopt bigger, deeper pre-trained networks with weights that generally work well in extracting features.

4.2 Pretrained-Models

Table 2 compiles pre-trained models chosen for the experiments. These range widely by the sizes of the base models, from 29 MB to 215MB. They are chosen such that the sizes can be reasonably compared to their performances in terms of prediction speed and accuracy.

With the base model weights frozen, the best performing model is EfficientNetB0 where the training and validation accuracy surpasses making predictions by chance. Two other models with training accuracy passing 50% is InceptionV3 and MobileNet, with validation accuracies between 48% and 49%. ResNet50, a model of comparable size to InceptionV3, is the worst performing model in Table 2. Its validation accuracy does not even reach 40% but is higher than its training counterpart. It is interesting to note how much faster InceptionV3 trains compared to ResNet50 despite having a much larger trainable parameter count.

Pre-trained model	*No. of layers*	*Total Parameter Count*	*Trainable Parameter Count*	*Training and validation accuracy, loss*	*Trainable Parameter Count for fine tuning*	*Fine-tuned training & validation acc, loss*
EfficientNetB0	230 (29 MB)	4,054,688	5,124 (113s/ep) (15s val)	train: 51.82%, 1.110 val: 50.94%, 1.127	1,498,980 (113s/ep) (17s val)	train: 56.18%, 1.029 val: 53.43%, 1.098
InceptionV3	311 (92MB)	22,065,572	262,788 (53s/ep) (5s val)	train: 50.21%, 1.128 val: 48.26%, 1.183	5,418,372 (59s/ep) (5s val)	train: 56.48%, 1.002 val: 48.46%, 1.183
InceptionResnetV2	780 (215 MB)	54,342,884 (58s val)	6,148 (353/ep)	train: 49.97%, 1.132 val: 48.90%, 1.145 (60s val)	6,742,884 (365s/ep)	train: 60.81%, 0.930 val: 53.03%, 1.078
MobileNet	86 (16 MB)	3,232,964	4,100 (87s/ep) (13s val)	train: 50.52%, 1.118 val: 48.01%, 1.142	2,398,724 (100s/ep) (11s val)	train: 60.44%, 0.946 val: 53.55%, 1.078
ResNet50	175 (98MB)	23,595,908	8,196 (157s/ep) (24s val)	train: 34.89%, 1.327 val: 36.46%, 1.323	14,458,372 (167s/ep) (24s val)	train: 41.46%, 1.251 val: 39.99%, 1.258

Table 2: Comparison of Pre-trained Models with ImageNet weights

When fine-tuned at 30 layers, the time it takes to train and validate per epoch increased slightly for all the models. InceptionResnetV2 and MobileNet has the largest improvement at around 10% and 4% to 5% for training and validation accuracy respectively. Although InceptionV3 seemed to be one of the more promising models before fine-tuned to the dataset, the validation accuracy had not improved despite the training improving by about 6%. The best validation accuracies surpasses 53%, which is achieved for the following models: EfficientNet50, InceptionResnetV2 and MobileNet. Since choosing the model to make predictions also depends on the prediction speed, the following section visualises the performances of selected models in Table 2.

4.2.1 Comparison

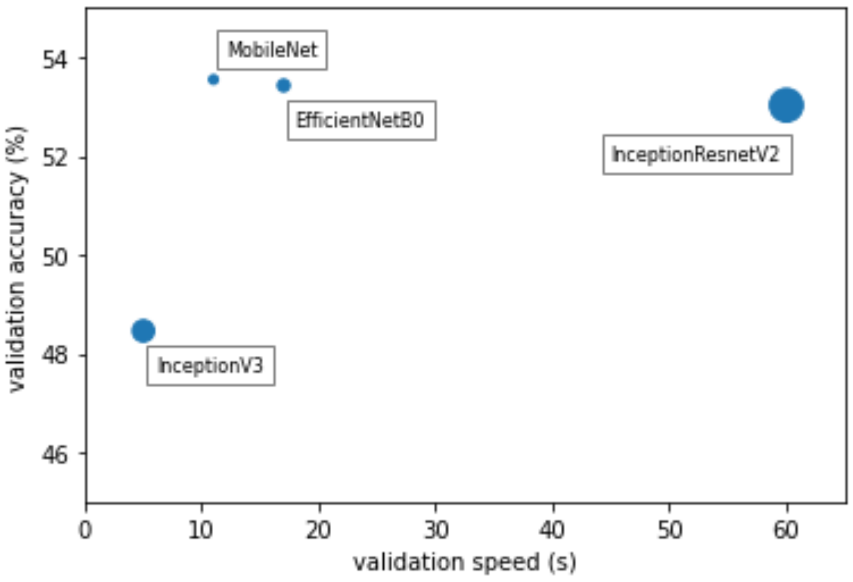


Figure 3: Fine-tuned models validation accuracy with respect to validation speed

The bubble plot in Figure 3 compares fine-tuned models that achieve at least 50% training accuracy and 48% for validation, in terms of the trade-off between accuracy and speed of the model. The size of the bubbles represent the sizes of the pre-trained models as listed in Table 2. InceptionResnetV2 takes the longest for validation, at 60 seconds, which is in line with the model being the largest in size. Despite being the second largest base model, InceptionV3 performs the fastest among the models but only achieved an accuracy slightly above 48%. MobileNet only takes 11 seconds to make predictions and is faster than EfficientNetB0 due to its much smaller size. It is safe to say that MobileNet is the best performing model in the pool of selected models for the experiments.

5 Model and Evaluation

5.1 MobileNet

5.1.1 Structure

MobileNet is primarily built off of depthwise separable convolution layers, which consists of depthwise convolution and a pointwise convolution [3]. For the standard RGB channels, a standard convolutional layer convolves the input with a filter for each channel and *stacks* the output together, resulting in a 3D output. On the other hand, a depthwise separable convolution requires an additional step of *combining* the 3D output to form a single channel output tensor, allowing the model to be much lighter since less multiplications are required to get the same output. MobileNet starts with a full convolutional layer, followed by Depthwise Separable Convolutional layers. ReLU activations and batch normalization layers are included between all the layers.

Additionally, MobileNet differs from the other pretrained models through the introduction of the following two hyperparameters used to shrink the model: width and resolution multiplier [3]. Denoted as $\alpha \in (0,1]$, the width multiplier plays a role in thinning the network uniformly at each layer. With this hyperparameter, the computational costs is reduced since the number of parameters is quadratically reduced by about α^2. The second hyperparameter, resolution multiplier, $\rho \in (0,1]$, which is set such that the input resolution is one of the following: 224, 192, 160 or 128. With a given ρ, the computational costs can be reduced by ρ^2. Section 4 of the original paper presents experiments when the hyperparameters are adjusted. It was presented that the smaller the values were, the smaller the accuracy. Due to this, these two hyperparameters are kept at $\alpha = 1$ and resolution 224 to fully maximize the accuracy of the model.

5.1.2 Training and validation

The main changes made to improve the accuracy presented for MobileNet in Table 2 is by increasing the epoch and the number of layers in fine-tuning. The number of epochs are set to 50 for the fully frozen base model, and a further 75 epochs when the last few layers are set to trainable. To avoid overfitting when the epochs is drastically increased, early stopping is adopted. If the validation accuracy had not improved by 0.001 from in 8 epochs, the training is stopped. Early stopping is applied to both the fully frozen layers and fine-tuning section. In terms of the number of layers to unfreeze, the fine-tuned layers are varied by increasing or decreasing the count by 5 incrementally from the initial 30 set for section 4.2.

Finally, it is found that unfreezing the final 35 layers of the chosen pre-trained model allows the fine-tuned model to achieve a final highest validation accuracy. The progress of training is presented in Figure 4. With early stopping, the training is stopped at 41 epochs for the fully frozen portion, reaching validation accuracy of 50.66%. The final 35 layers are then set to trainable and the model is further trained, stopping at 71 epochs and ending at a validation accuracy of 55.03%. This is an increase of 2% compared to the accuracy presented in Table 2. The green vertical bar shows the point that the model is fine-tuned, where there is some evidence of improvement when the higher level features extracted is tailored to the train set. Past this bar, we can see that the training accuracy starts improving quite significantly after reaching convergence and is stopped at 67.36%. However, the validation accuracy only slightly improved within a range of 5%. That being said, this is the best performing model we have seen so far with a good balance in speed, size and accuracy, taking a total 10s to predict on 78 batches of the validation set.

5.2 Evaluation

With a test dataset set aside in the beginning of the analysis, the model can be fully evaluated in its ability to make predictions. The test accuracy obtained by the trained weights of the model is 54.31%, which is on par with the validation accuracy. The fine-tuned MobileNet model can barely make predictions exceeding chance. Clearly, this makes the model insufficient in achieving the purposes

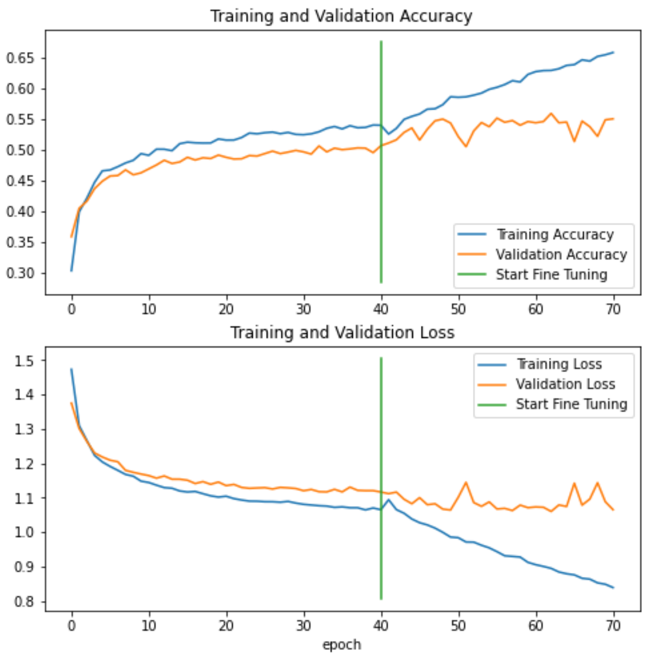
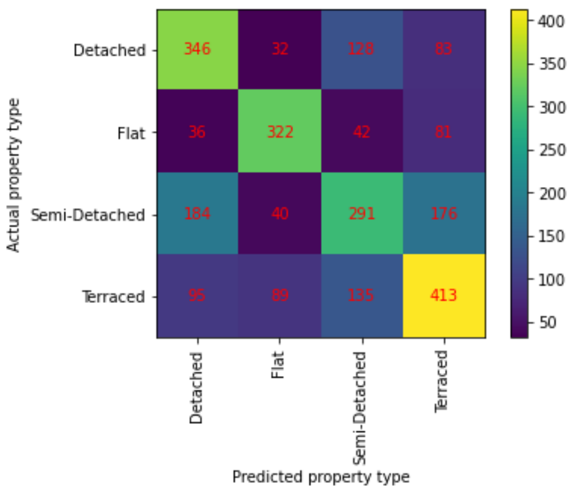
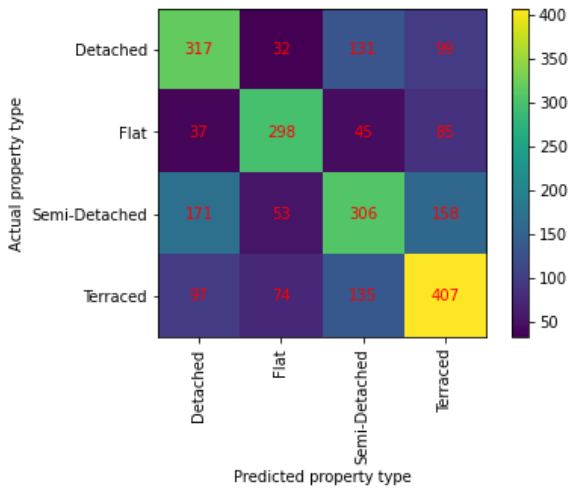


Figure 4: Training progress

laid out in Section 1. Plotting a confusion matrix helps to reveal which class has been predicted well and which classes were too similar for the model to recognize the distinction.



(a) Validation



(b) Test

Figure 5: Confusion matrix

For a well performing model, the confusion matrix heatmap is expected to be the brightest along the diagonal entries and much darker for the non-diagonal elements. Based on the similar shades of the heatmaps obtained from validation and test sets, it is clear that the validation set is a good representation of the test set. Out of the diagonal elements, the true positives are largest for Terraced. However, this is only due to the class having the largest support. The class is mostly mistaken for semi-detached property. About 25% and 23% of semi-detached properties are misclassified as detached and terraced respectively. This confirms the issues highlighted for the image quality for semi-detached properties in Figure 1. 18% of the Flats are misclassified as Terraced while 22% of the Detached properties are misclassified as Semi-detached.

Table 3 summarises the precision, recall and F1-score for each class, as well as macro and weighted

Class	*Precision*	*Recall*	*F1-score*	*Support*
Detached	0.547	0.51	0.528	579
Flat	0.641	0.652	0.646	465
Semi-Detached	0.445	0.496	0.469	688
Terraced	0.571	0.543	0.557	713
Macro average	0.551	0.55	0.550	2445
Weighted average	0.543	0.543	0.542	2445

Table 3: Model evaluation metrics

averages. The precision metric shows the fraction of the predicted class is actually true, whereas recall is the proportion of actual samples that were correctly predicted. The F1-score is a combination of both recall and precision. Across all 3 metrics, the class with the highest score is Flat. This tells us that the features for this property is distinct in comparison to the other properties. The lowest performing class is Semi-Detached, with metrics barely reaching 50%. Overall, the metrics for the model is not promising since it is only able to make predictions slightly above chance.

6 Limitations and Conclusions

6.1 Limitations and further research

The biggest limitation faced in the project is the quality of the dataset itself. As explained in the EDA section of the project, the photos are not necessarily distinct netween classes. Most of the photos are unclear of the property is or hidden by surrounding objects. A suggestion on tackling this issue would be to choose a relatively small portion of the photos and manually label whether the photo can easily be recognised by the human eye, labelling either clear or unclear to the photos. A neural network can then be trained and validated on this dataset, and predictions can be made on the rest of the samples. Although this method is likely to misclassify some of the images, it is worth trying to improve the quality of the training set.

There could be signs of under fitting due to the low number of epochs, which is how many times the data is exposed to the network. However, due to the time constraint of the project, the experiments are deemed sufficient. This can be directly improved by setting larger epoch values and adopting early stopping for the comparison section of the project. Furthermore, a larger selection of pre-trained models are freely available to be experimented on, while this project has been limited to the available architectures in keras. Similarities between the dataset and pre-trained models should be exploited to extract distinct features between the classes, by choosing pre-trained models that are more closely related to building images.

6.2 Conclusions

This project has explored fitting a wide variety of convolutional networks to determine whether it is possible to make predictions based on Google snapshots of properties. This includes comparing accuracies from training done on baseline CNNs and adopting weights of pre-trained models from the ImageNet dataset. On baseline CNNs, the hyperparameters are varied to obtain different sizes and extracting different features, where it was found that even the training accuracy was capped at roughly 50%. The experiments were continued on pre-trained models to exploit the model's ability to extract the lower level features by keeping the earlier layers frozen during the fine-tuning stages.

The final selected model, MobileNet, only managed to predict new images at 55% accuracy. Although there is some improvement from baseline, the performance is insufficient to be considered a well-performing model. The poor quality of the dataset makes the task of fitting models with more desireable metrics very difficult. This is the main aspect of the exploration that must be improved before any other models are fitted to the dataset.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[2] Y. Ho and S. Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813, 2019.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[6] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[7] B. Shah and H. Bhavsar. Time complexity in deep learning models. *Procedia Computer Science*, 215:202–210, 2022.

Appendix

The full code that was run for the project is in this github link: simply business .All the models have separate scripts dedicated to them, so further tweaking can be made to a specific model. The trained models have been saved in the models folder to obtain the training and validation accuracies reported in this project. The training progress for each model, and other figures in this report is saved into the figures folder.