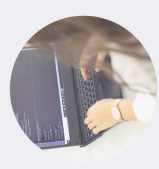
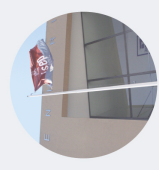




Um Classificador de Divisões de Interiores de Casas para Suportar Agentes Imobiliários

BRUNO TIAGO SILVA VEIGA

Julho de 2023



An Indoor Room Classifier to Support Real Estate Agents

BRUNO TIAGO SILVA VEIGA
Julho de 2023



An Indoor Room Classifier to Support Real Estate Agents

Bruno Tiago Silva Veiga

Student No.: 1180712

**A dissertation submitted in partial fulfillment of
the requirements for the degree of Master of Science,
Specialisation Area of Artificial Intelligence Engineering**

**Supervisor: Doctor Tiago Manuel Campelos Ferreira Pinto, Assistant Profes-
sor, University of Trás-os-Montes and Alto Douro**

**Co-Supervisor: Doctor Carlos Fernando da Silva Ramos, Main Coordinator
Professor, Polytechnic of Porto - School of Engineering**

External Supervisor: Engineer David Mota, DevScope

Evaluation Committee:

President:

Doctor Luiz Felipe Rocha de Faria, Coordinator Professor, Polytechnic of Porto - School of Engineering

Members:

Doctor Paula Maria Marques Moura Gomes Viana, Coordinator Professor, Polytechnic of Porto - School of Engineering

Doctor Tiago Manuel Campelos Ferreira Pinto, Assistant Professor, University of Trás-os-Montes and Alto Douro

Abstract

Everyday, Real Estate Agents retrieve pictures from houses that are then manually labelled and filtered to pick the best ones to present on their websites and better describe the homes they are selling. This is a time consuming task that delays the whole process of advertising homes and receiving propositions and could be solved with Computer Vision solutions. Machine Learning, more in particular Deep Learning, is a field that has been enjoying much success in the resolution of these types of problems, so it is a very promising solution to be used in this context. For that reason, this work proposes a neural network classifier for indoor images, that will save countless hours spent in these tasks. Moreover, to help achieve better results and provide additional contributions to the field, a study of the main architectures down the years, going from Convolutional Neural Networks to Transformers, and their performance for several datasets retrieved from online sources is also presented, settling all the information in one place and helping future researchers for similar tasks. The selected architectures for the case studies were the most relevant ones according to their popularity and performance, originating from different years, in order to provide an exploratory analysis of the evolution of the architectural aspects and the degree of impact these aspects have on the results obtained. These results confirmed that Transformers are indeed the ideal architecture for image classification tasks, especially in the real estate field, although Convolutional Neural Networks still play an important role, particularly in hybrid networks combining Transformers and Convolutional Neural Networks, where they are capable of achieving similar performance to purely Transformer networks. The best model achieved was able to predict images by training on a research dataset of 79500 images with an accuracy of 90%, which is quite good. This model was adapted and made available through an API. Some promising results were also obtained using a smaller and unbalanced private dataset of 10496 real estate images, where F1-scores of 87% and 86% were achieved for the multi-class and multi-label problems, respectively.

Keywords: Indoor Classification, Real Estate, Deep Learning, Computer Vision, Convolutional Neural Network, Vision Transformer

Resumo

Todos os dias, agentes Imobiliários recolhem imagens de casas, as quais, posteriormente, têm de filtrar e agrupar por categorias para apresentarem nos seus websites, com o intuito de descrever com mais facilidade e transparência as propriedades que pretendem vender. Esta é uma tarefa que consome muito tempo e que atrasa todo o processo de publicitação, enquanto retém um trabalhador e que pode ser automatizada a partir de soluções de Visão por Computador. Aprendizagem Automática, mais concretamente Aprendizagem Profunda, é uma área que tem vindo a ter muito sucesso na resolução deste tipo de problemas e apresenta-se como uma solução muito promissora para este contexto. Por essa razão, este trabalho propõe um classificador de divisões de casas baseado em redes neuronais, que irá poupar uma elevada quantidade de tempo despendido nestas tarefas. Mais do que isso, para alcançar resultados superiores e fornecer uma contribuição a esta área, um estudo das principais arquiteturas de redes neuronais, aplicadas ao longo dos anos com resultados promissores nesta área, é apresentada, incluindo modelos baseados em Redes Neurais Convolucionais e a sua evolução até aos Transformadores Visuais. Juntamente, o desempenho destas arquiteturas é exposto para vários conjuntos de dados obtidos de fontes online e datasets privados, reunindo todas as informações em um só lugar, com o intuito de ajudar futuros investigadores em tarefas semelhantes. As arquiteturas selecionadas para os casos de estudo foram as mais revelantes de acordo com a sua popularidade e performance e têm a sua origem em diferentes anos, de forma a oferecer uma análise exploratória da evolução dos aspetos arquiteturais e o grau de impacto que esses aspetos têm nos resultados obtidos. Os resultados obtidos comprovaram que, de facto, os Transformadores são agora a arquitetura ideal para tarefas de classificação de imagem, sobretudo na área de imobiliário, ainda que as Redes Neuronais Convolucionais ainda tenham um papel importante, principalmente nas redes mistas de Transformadores e Redes Neuronais Convolucionais, que são capazes de apresentar performances semelhantes às redes puramente de Transformadores. O melhor modelo obtido foi capaz de prever imagens treinando com um dataset de investigação de 79500 imagens com acurácia de 90%, um valor bastante bom, tendo sido este modelo adaptado e disponibilizado a partir de uma API. Alguns resultados promissores foram também obtidos com o uso de um dataset privado, não tão grande ou balanceado, com 10496 imagens de imobiliário onde *F1-scores* de 87% e 86% foram obtidos, tanto para o problema de *multi-class* como de *multi-label*, respetivamente.

Palavras-chave: Classificação de Divisões de Habitação, Imobiliária, Aprendizagem Profunda, Visão por Computador, Rede Neuronal Convolucional, Transformador Visual

Acknowledgement

I would like to thank all of those who played a part in the success of this work.

To my supervisors, Prof. Tiago Pinto and Prof. Carlos Ramos for their orientation and help throughout the development of this thesis, as well as for all the availability and teachings provided to me in the last years.

To my supervisors and co-workers at DevScope, in special, Eng. David Mota, Eng. Rúben Teixeira and Eng. Luís Maia, for the opportunity of developing my thesis in a work environment and for providing me with all the resources and help required for a quick and successful adaptation.

Finally, to my friends and family, for being there for me when I most needed them and giving me support and encouragement throughout all the years of my life.

Contents

List of Acronyms	xvii
1 Introduction	1
1.1 Context	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Contributions	3
1.5 Document Structure	4
2 State of the art	5
2.1 Image Classification	5
2.2 Neural Networks	6
2.2.1 Convolutional Neural Network	9
2.2.2 Transformers	10
2.3 Models Architectures	13
2.3.1 AlexNet (2012)	13
2.3.2 Zeiler Fergus Network (2013)	14
2.3.3 Inception Network (2014)	15
2.3.4 Residual Neural Network (2015)	17
2.3.5 ResNeXt (2016)	18
2.3.6 Squeeze-and-Excitation Network (2017)	18
2.3.7 PNAS-5 and AmoebaNet-A (2018)	19
2.3.8 EfficientNet (2019)	20
2.3.9 Vision Transformer (2020)	21
2.3.10 CoAtNet (2021)	22
2.3.11 ConvNext (2022)	23
2.3.12 Contrastive Captioner (2022)	24
2.4 Indoor Scenes Classification	25
2.5 Summary	31
3 Data, Methodology and Tools	33
3.1 Frameworks	33
3.1.1 Scikit-learn	33
3.1.2 Tensorflow	33
3.1.3 Keras	34
3.1.4 PyTorch	34
3.1.5 Hugging face	34
3.1.6 JAX	34
3.1.7 Fast.ai	34
3.1.8 Caffe	35
3.1.9 Microsoft Cognitive Toolkit	35

3.1.10	Framework Comparison	35
3.2	Datasets	36
3.2.1	Datasets Creation	36
3.2.2	Data Privacy and Protection	40
3.3	Work Methodology	40
3.4	Deployment	41
3.5	Security Analysis	42
3.6	Summary	44
4	Experimentation and Discussion	45
4.1	Case Studies Overview	45
4.2	ResNet	46
4.3	EfficientNet	50
4.4	ConvNeXt	53
4.5	ViT	56
4.6	CoCa	59
4.7	Application of the Lion optimizer to CoCa	63
4.8	Multi-label Classification Experimentation	63
4.9	Summary	64
5	Conclusion	67
5.1	Summary of Results	67
5.2	Objectives Overview	67
5.3	Ethical Issues	68
5.4	Limitations and Future Work	68
5.5	Final Remarks	69
	Bibliography	71
A	Confusion Matrix Analysis of ResNet Models	81
B	Confusion Matrix Analysis of EfficientNet Models	85
C	Confusion Matrix Analysis of ConvNeXt Models	87
D	Confusion Matrix Analysis of ViT Models	89
E	Confusion Matrix Analysis of CoCa Models	91

List of Figures

1.1	Indoor classification examples (from [14])	2
1.2	Graphical representation of the contributions	4
2.1	Simple Neural Network architecture (from [31])	6
2.2	Confusion matrix	8
2.3	CNN convolution layer(from [44])	10
2.4	Transformer architecture(from [53])	11
2.5	Attention and multi-head attention mechanism(from [53])	12
2.6	AlexNet's architecture representation (from [59])	14
2.7	ZFNet architecture representation (from [62])	15
2.8	Inception module, naive version architecture representation (from [63]) . .	16
2.9	Inception module, dimension reduction architecture representation (from [63])	16
2.10	GoogLeNet architecture representation (from [63])	17
2.11	ResNet architecture representation (from [66])	17
2.12	ResNeXt architecture representation (from [67])	18
2.13	Comparison between the ResNet and the SE-ResNet modules (from [69])	19
2.14	EfficientNet architecture representation (from [73])	21
2.15	Vision Transformer architecture representation (based on [78])	22
2.16	CoAtNet architecture representation (from [79])	23
2.17	CoCa architecture representation (based on [82])	25
2.18	RE-Tagger architecture representation (from [90])	27
2.19	Overall architecture representation (from [3])	29
2.20	Overall LeNet architecture representation (from [93])	30
2.21	Overall ResNet 50 architecture representation (from [93])	30
3.1	Dataset pre-processing	37
3.2	Full training methodology	41
4.1	Scope of trainable weights for each methodology	45
4.2	Results of the model based on ResNet 152v2 with Methodology 1 - 24 batch size - 224x224 resolution	47
4.3	Results of the model based on ResNet 152v2 with Methodology 2 (Feature extraction step) - 24 batch size - 224x224 resolution	47
4.4	Results of the model based on ResNet 152v2 with Methodology 2 (Fine tuning step)- 24 batch size - 224x224 resolution	48
4.5	Results of the model based on ResNet 152v2 with Methodology 3 - 64 batch size - 224x224 resolution	49
4.6	Examples of photos with multiple rooms (from [100])	49
4.7	Results of the model based on EfficientNet V2L with Methodology 1 - 8 batch size - 224x224 resolution	51

4.8	Results of the model based on EfficientNet V2L with Methodology 2 (Feature extraction step) - 32 batch size - 224x224 resolution	51
4.9	Results of the model based on EfficientNet V2L with Methodology 2 (Fine tuning step) - 32 batch size - 224x224 resolution	52
4.10	Results of the model based on EfficientNet V2L with Methodology 3 - 32 batch size - 224x224 resolution	52
4.11	Results of the model based on ConvNeXt Large with Methodology 1 - 8 batch size - 224x224 resolution	54
4.12	Results of the model based on ConvNeXt Large with Methodology 2 (Feature extraction step) - 32 batch size - 224x224 resolution	54
4.13	Results of the model based on ConvNeXt Large with Methodology 2 (Fine tuning step) - 32 batch size - 224x224 resolution	55
4.14	Results of the model based on ConvNeXt Large with Methodology 3 - 32 batch size - 224x224 resolution	55
4.15	Results of the model based on ViT Base Patch 16 with Methodology 1 - 32 batch size - 224x224 resolution	57
4.16	Results of the model based on ViT Base Patch 16 with Methodology 2 - 32 batch size - 224x224 resolution	57
4.17	Results of the model based on ViT Base Patch 16 with Methodology 3 - 32 batch size - 224x224 resolution	58
4.18	Results of the model based on CoCa Large with Methodology 1 - 4 batch size - 224x224 resolution	59
4.19	Results of the model based on CoCa Large with Methodology 2 - 32 batch size - 224x224 resolution	60
4.20	Results of the model based on CoCa Large with Methodology 3 - 32 batch size - 224x224 resolution	60
4.21	Living room with a chaise longue sofa	62
A.1	Confusion matrix of the model based on ResNet 152v2 with Methodology 1 - 24 batch size - 224x224 resolution	81
A.2	Confusion matrix of the model based on ResNet 152v2 with Methodology 2 - 24 batch size - 224x224 resolution	82
A.3	Confusion matrix of the model based on ResNet 152v2 with Methodology 3 - 64 batch size - 224x224 resolution	82
B.1	Confusion matrix of the model based on EfficientNet V2L with methodology 1 - 8 batch size - 224x224 resolution	85
B.2	Confusion matrix of the model based on EfficientNet V2L with methodology 2 - 32 batch size - 224x224 resolution	86
B.3	Confusion matrix of the model based on EfficientNet V2L with methodology 3 - 32 batch size - 224x224 resolution	86
C.1	Confusion matrix of the model based on ConvNeXt Large with methodology 1 - 8 batch size - 224x224 resolution	87
C.2	Confusion matrix of the model based on ConvNeXt Large with methodology 2 - 32 batch size - 224x224 resolution	88
C.3	Confusion matrix of the model based on ConvNeXt Large with methodology 3 - 32 batch size - 224x224 resolution	88

D.1	Confusion matrix of the model based on ViT Base Patch 16 with methodology 1 - 32 batch size - 224x224 resolution	89
D.2	Confusion matrix of the model based on ViT Base Patch 16 with methodology 2 - 32 batch size - 224x224 resolution	90
D.3	Confusion matrix of the model based on ViT Base Patch 16 with methodology 3 - 32 batch size - 224x224 resolution	90
E.1	Confusion matrix of the model based on CoCa Large with methodology 1 - 4 batch size - 224x224 resolution	91
E.2	Confusion matrix of the model based on CoCa Large with methodology 2 - 32 batch size - 224x224 resolution	92
E.3	Confusion matrix of the model based on CoCa Large with methodology 3 - 32 batch size - 224x224 resolution	92

List of Tables

2.1	Accuracies of the main neural networks architectures during the years . . .	13
2.2	Research query	25
2.3	Number of research papers retrieved per data source	26
2.4	Summary of datasets and techniques present in state of art	26
3.1	Comparison between machine learning frameworks used for training	35
3.2	Description of the datasets used	36
3.3	Description of the research dataset	39
3.4	Loading and inference times of the CoCa models in the API for 5 images .	42
3.5	Loading and inference times of the CoCa models in the API for 567 images	42
4.1	F1-score for ResNet models	50
4.2	F1-score for EfficientNet models	53
4.3	F1-score for ConvNeXt models	56
4.4	F1-score for ViT models	58
4.5	F1-score for CoCa Large models trained with the research dataset and Adam optimizer	61
4.6	F1-score for the CoCa model applied to the private real estate dataset . .	62
4.7	F1-score for CoCa Large models trained with the Lion optimizer	63
4.8	F1-score for the tested multi-label models	64
4.9	F1-score for the best models	65

List of Acronyms

AE	Auto Encoder.
AI	Artificial Intelligence.
BM	Boltzmann Machine.
CNN	Convolutional Neural Network.
CoCa	Contrastive Captioner.
DBN	Deep Belief Network.
DL	Deep Learning.
FFN	Feed Forward Network.
GAN	Generative Adversarial Network.
GELU	Gaussian Error Linear Unit.
GRU	Gated Recurrent Unit.
HN	Hopfield Network.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
KNN	K-nearest neighbors.
LSTM	Long/Short Term Memory Network.
MC	Markov Chain.
MHSA	Multi-Headed Self-Attention.
MLP	Multilayer Perceptron.
NLP	Natural Language Processing.
NN	Neural Network.
ONNX	Open Neural Network Exchange.
PNAS	Progressive Neural Architecture Search.
ReLU	Rectified Linear Unit.
ResNet	Residual Neural Network.
RF	Random Forest.
RNN	Recurrent Neural Network.

SENet	Squeeze-and-Excitation Network.
SVM	Support Vector Machine.
Tanh	Hyperbolic Tangent.
ViT	Vision Transformer.
ZFNet	Zeiler Furgus Network.

Chapter 1

Introduction

This chapter presents a succinct explanation of the context and problem behind the development of the thesis. It starts by exploring the domain of the problem and then moves on to presenting how those problems can be handled, showing which objectives can help solve them and which contributions are given to the fields addressed.

1.1 Context

Real estate [1] is the field that englobes all aspects of owning lands and buildings, including the economic aspects inherent to it, such as buying and selling them. Real estate companies are companies that buy, sell, develop and invest in real estate, by employing agents who usually negotiate deals and are somewhat capable of influencing property prices according to some studies [2].

These agents typically capture several images of homes that they intend to advertise on their websites or other platforms. This is an important task, since the images selected for the promotion of a house may be the difference between selling it quickly or not [3]. Another important point is that these photos may also help buyers accept the properties selling prices if they comprise to the prices expected of these when they share similar rooms' characteristics [4]. This is a time consuming task, since, usually, a big volume of photos needs to be sorted every day and human resources have to be allocated. This problem could be solved by Artificial Intelligence (AI) solutions [5], particularly with Deep Learning (DL) [6]–[8] and Computer Vision [9].

These fields have been developing fast in recent years in tasks of classification [10], segmentation [11] and object detection [12]. For the objective of this project, classification is the main field to be employed, even though, segmentation and object detection may sometimes help in classification tasks due to properties and objects being common in some scenes and easily helping identify the spaces present in images [13]. Despite this, research conducted on the application of Computer Vision in indoor classification is still somewhat lacking in recent years, with existing solutions still relying on classic state of the art models, like Inception Networks and Residual Neural Networks (ResNet), and no new approaches showing how the new rising architectures, such as Transformers, perform in the field.

That is the challenge proposed by DevScope¹ which gave the author of this document the opportunity to explore the problem of image classification applied to the real estate field. At the same time, a commercial solution is also produced, since it is a needed product for a current DevScope customer in the real estate industry.

¹DevScope - <https://devscope.net/>

1.2 Problem Statement

This thesis addresses the problem of indoor image classification, where the objective is to provide the right indoor room tag for images as shown in figure 1.1, since the allocation of human resources for this type of task starts to become unsustainable as the volume of images to be classified starts to increase. For this reason, this work develops and proposes a system that contributes to surpassing this classification problem and help real estate agents.

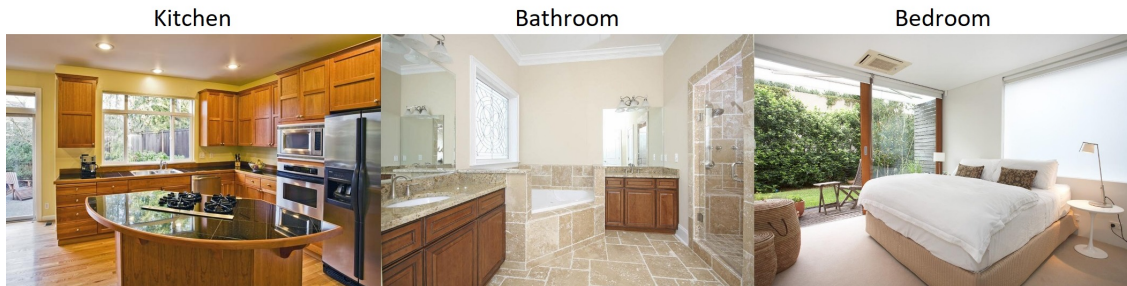


Figure 1.1: Indoor classification examples (from [14])

For this purpose, the study of several distinct approaches and architectures of Neural Networks (NN), and a comparison of their performances is carried out, going from Convolutional Neural Networks (CNN) to Vision Transformers (ViT), which at the date are the main and best types of neural networks for the task of image classification [15], as discussed on detail in chapter 2 of this document. So, an exploration of the more particular architectures is presented in detail, along with a critical analysis of the results obtained, focusing on the promising and widely used models, namely ResNet, EfficientNet and ConvNext for CNNs, and ViT and Contrastive Captioner (CoCa) for Transformers.

An iterative process is implemented, between the adaptation of models and datasets, with these being retrieved from several online sources, from state of the art datasets and web scraping. Since these have different origins, special attention is required for their balancing and treatment, due to their different characteristics, image quality and lack of labels when these have their origin in web scraping.

This study is very helpful in the identification of the best combination of image characteristics, hyperparameters and models' architectures for the creation of the final solution for DevScope, which will then utilize a different private dataset for the effect. Ultimately, the solution will save time spent by workers in this classification task, while helping improve the quality of predictions over time when new data is retrieved.

1.3 Objectives

The main objective of this thesis is to develop an indoor scene classifier. In order to achieve this main objective, several more specific objectives have been defined to help reach the main one and conduct the research and implementation. These are:

- **O1:** Study the state of the art of the image classification problem and how it has been evolving throughout the years.
- **O2:** Analyse different datasets and how some properties changes in them influence the results.

- **O3:** Study how the different neural network architectures influence the performance of the models in the indoor classification task.
- **O4:** Perform a study of how the best model performs on real photos taken from other contexts outside online sources, like photos taken of the author's home.
- **O5:** Investigate the best deployment method for the best found classification model.
- **O6:** Analyse the achieved results critically and draw the most significant conclusions.

These objectives are explored throughout the document and in the final section 5, some regards are given on how these objectives were explored and fulfilled.

1.4 Contributions

This thesis offers several contributions:

- **C1:** A summary of the main state of the art classification architectures and their key aspects in recent years.
- **C2:** A review of existing solutions for the problem of indoor classification and what methodologies were used.
- **C3:** The comparison of several neural network architectures for a built dataset and how the characteristics of both influence the performance of the solutions.
- **C4:** A dataset of images retrieved outside online sources.
- **C5:** A deployed indoor classification model to be used by DevScope.
- **C6:** A critical analysis of the results and conclusions obtained.
- **C7:** A comparison between different approaches in the training/test process.
- **C8:** The submission of a conference paper already accepted: B. Veiga, T. Pinto, R. Teixeira, and C. Ramos, "Vision Transformers applied to Indoor Room Classification", EPIA2023 (22nd EPIA Conference on Artificial Intelligence), submitted by 30-04-2023
- **C9:** The submission of a journal paper still under review: B. Veiga, T. Pinto, R. Teixeira, and C. Ramos, "Indoor room classification: comparing Convolutional Neural Networks and Vision Transformers based on a Contrastive Captioner approach", IEEE Access, submitted by 19-05-2023.

This document delves into the field of indoor image classification, following a serialized set of steps to understand and develop a system capable of auxiliating real estate agents in the process of classifying indoor images. For that purpose, the document gathers key information where further exploration of similar cases and problems are shown along with the solutions chosen to resolve them. It also thoroughly presents the main architectures of image classification throughout the years, showing the new aspects and contributions in the performance of those architectures. Both these studies are essential for this work, by providing enough knowledge to guide the remaining research and ultimately, the creation of the final classification model. In particular, this research promotes a practical comparison of many of the studied architectures when applied to research and real estate datasets, along with different methodologies, where the freezing of layer weights is tuned, along with the hyperparameters in each one of the methodologies. The results of this comparison are then

critically analyzed and allow drawing conclusions and experiments that are also presented in two papers, a conference and a journal article. A graphical representation of the previous description of the contributions of this thesis is present in figure 1.2.

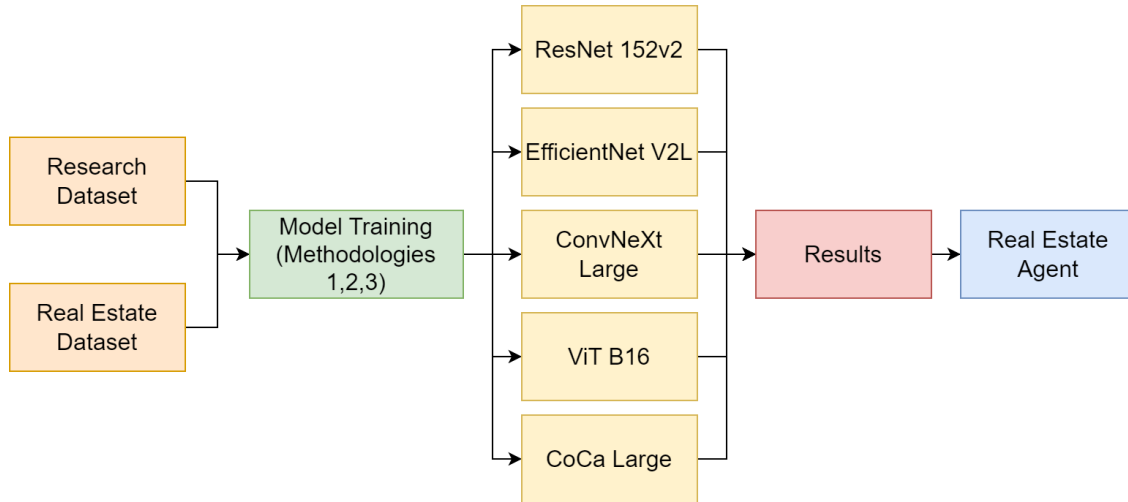


Figure 1.2: Graphical representation of the contributions

Here, both the research and real estate dataset are used for training 5 different model architectures based on 3 different methodologies for each architecture. Results are then obtained on the datasets, helping establish the best ones, which are then used by real estate agents and facilitate in the task of filtering house images.

1.5 Document Structure

The document is organized in different chapters with the following structure:

- Chapter 1 presents an overview of the context and problems that motivate the elaboration of this thesis, as well as its objectives and contributions.
- Chapter 2 presents a state of the art of the theme of the thesis, focusing on a general overview of computer vision solutions, neural networks and the types of networks used for image classification, a study of how the specific architectures have evolved throughout the years and finally, existing solutions on the field of indoor classification.
- Chapter 3 presents the data, methodology and tools used, focusing on existing frameworks and their comparison, the dataset building and pre-processing procedure, an analysis of data protection, security analysis and of course the work methodology and deployment of the final solution.
- Chapter 4 demonstrates case studies of how different neural network architectures perform based on the dataset that was used and compares them.
- Chapter 5 presents the final conclusions of the project, where the main results are presented, along with an analysis of the fulfilled objectives, ethical issues, limitations, future work and the final remarks.

Chapter 2

State of the art

In this chapter, an analysis of the state of the art for the indoor classification problem is presented, going through a general explanation of the image classification problem and Neural Network solutions, state of the art architectures used throughout the years and how they contributed to this problem and finally, a study of the similar work in this field.

2.1 Image Classification

Image Classification is one of the problems treated by Computer Vision [16] and refers to the ability of a machine, when presented with an image, successfully indicating its category [17]. One of the first works in image classification dates back to 1967, with the creation of a face recognition system by Woodrow W. Bledsoe, which utilized a graphical computer input device named RAND tablet [18].

After that, several studies appeared basing image processing and classification on low-level image analysis, but this technique presented difficulties in semantically describing scenes [19], [20]. The next step was the usage of K-nearest neighbors and this was a very predominant technique for this task [20], [21], with some of the papers even pairing it with Bayes Networks [22]. Other classic machine learning algorithms have also been used throughout the years, such as: Random Forest (RF) [23], Support Vector Machine (SVM) [24] or even Decision Tree [25].

More recently, [17] presents an approach that utilizes a 5 step technique: image segmentation, object identification, image feature extraction, image feature selection and classification. For classification, the authors use 4 classifiers: BayesNet, Naive Bayes Updatable, Random Tree and IBk. This approach presented good results for the Caltech-101 dataset [26] when using the BayesNet classifier, with accuracies above 95%. Despite this, an analysis of the source [26] suggests a recent shift to the usage of CNNs when training with this dataset, although not many examples are present. This lack of results for the given dataset can also be explained due to the preference of other bigger datasets, that better reflect the quality of these types of models, such as ImageNet [27].

Furthermore, the results presented for the ImageNet dataset clearly show that Deep Learning is now the predominant form for classification tasks [15], exhibiting since 2012, an increased use of CNNs and then more recently, a shift towards ViTs. Neural Networks are precisely what will be explored in the following sections of the document, focusing the these two best types of DL techniques for image classification and their particular architectures.

2.2 Neural Networks

Neural networks are a part of machine learning methods and the most important computational model for Deep Learning. They have their origin in 1943, when McCulloch and Walter Pitts wrote the first paper proposing an artificial neuron [28]. These systems are very powerful tools that help in decision making tasks that require processing of many factors and details and utilize algebra calculations internally to perform them. They are usually data greedy, meaning that they perform extremely well with a lot of information [29].

Usually, the neural networks, in their simplest state, follow an architecture present in figure 2.1, with one input layer with a number of units equal to the number of the features of the input data, one or more hidden layers where more complex logic is computed and finally, the output layer with a number of units equal to the number of classes that have to be predicted [30].

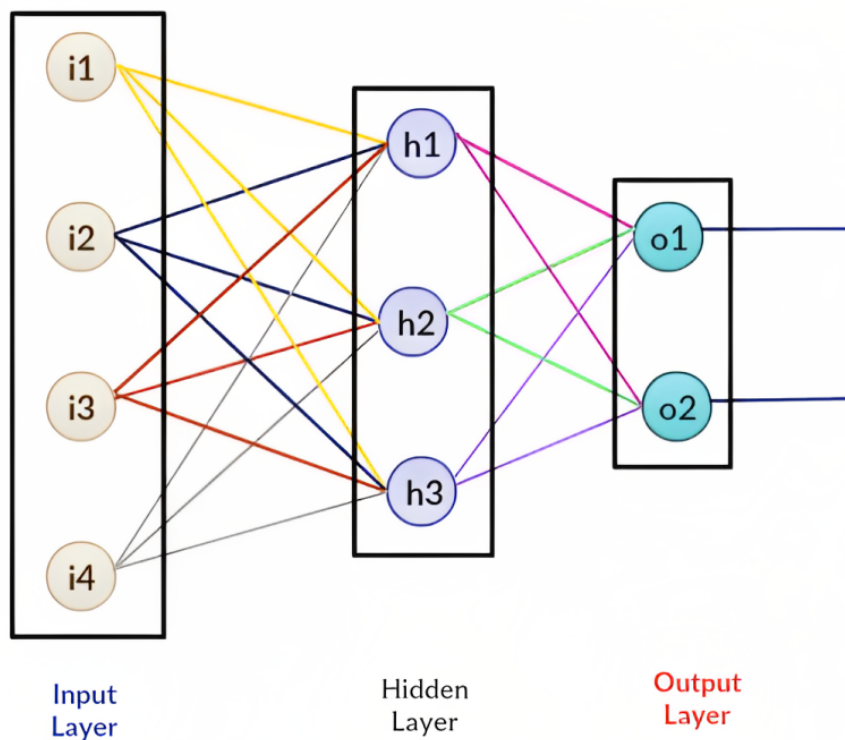


Figure 2.1: Simple Neural Network architecture (from [31])

For that, the networks retrieve data, train on it and then predict the outputs for other never seen data [31]. They are represented by a set of units connected with associated weights and biases. The training is composed of two steps, where first, the network utilizes the associated weights and bias and flows through the network layers to obtain a prediction. And second, these associated weights are adjusted according to the error of prediction by a backtracking algorithm, such as Gradient Descent [32], [33], Follow the Regularized Leader (FTRL) [34], Adam Optimizer [35] or the recent Lion Optimizer [36], that it is used in the training of the current best Image Classification model, the Basic-L [37]. The Lion optimizer, due to being quite recent is only used in section 4.7, with the best model architecture, to serve as a comparison between the previous best optimizer Adam that was used throughout

the remaining results. The Basic-L model, despite showing its best result in 2023, it is not shown in more detail in section 2.3 because it is based on the CoAtNet design that is already explained in the given section.

The connection between layers is done by intermediary activation functions that pick up the input signal and transform them into output signals that will posteriorly be fed to the next layer of the neural network. There are several activation functions [38], [39]:

- Binary Step Function, which is the simplest activation function and it is defined as:

$$f(x) = 1, x \geq 0 \quad (2.1)$$

$$f(x) = 0, x < 0 \quad (2.2)$$

- Linear Function, which is directly proportional to the input and it is defined as:

$$f(x) = \alpha x \quad (2.3)$$

- Sigmoid, which compresses values in between 0 and 1. It is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.4)$$

- Hyperbolic Tangent (Tanh), a variation of the Sigmoid function which is symmetric around the origin. Its definition is:

$$f(x) = 2\text{sigmoid}(2x) - 1 \quad (2.5)$$

- Rectified Linear Unit (ReLU), which is capable of deactivating some neurons when the output of the linear transformation is zero. This is defined as:

$$f(x) = \max(0, x) \quad (2.6)$$

- Leaky ReLU, a variation of the ReLU that defines the previous 0 outputs as an extremely small linear component of x and is defined as:

$$f(x) = 0.01x, x < 0 \quad (2.7)$$

$$f(x) = x, x \geq 0 \quad (2.8)$$

- Parametrized ReLU, which is an improved version of ReLU that solves the zero gradient, when x is negative. This function is defined as:

$$f(x) = \alpha x, x < 0 \quad (2.9)$$

$$f(x) = x, x \geq 0 \quad (2.10)$$

- Gaussian Error Linear Unit (GELU), which aims to combine the properties of ReLU with dropout regularization and it is mostly used in Transformers. This is defined as:

$$f(x) = 0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3))) \quad (2.11)$$

- Exponential Linear Unit (ELU), which uses a log curve for the negative values. Its definition is:

$$f(x) = x, x > 0 \quad (2.12)$$

$$f(x) = \alpha(e^x - 1), x \leq 0 \quad (2.13)$$

- Swish, a better performing method when compared to ReLU, that has the particularity of decreasing its value when x increases. It can be defined as:

$$f(x) = x * \text{sigmoid}(x) \quad (2.14)$$

$$f(x) = \frac{x}{1 + e^{-x}} \quad (2.15)$$

- SoftMax, which is a combination of several sigmoid functions, usually, utilized to obtain the probability of a registry belonging to a class. It is defined as:

$$\omega(x)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K \quad (2.16)$$

		prediction outcome		
		p	n	total
actual value	p'	True positive	False negative	P'
	n'	False positive	True negative	N'
total		P	N	

Figure 2.2: Confusion matrix

After the training of a neural network is complete, there are several ways to evaluate the network for tasks of classification, such as accuracy, precision, recall, F1-score with the help of a confusion matrix [40]–[42]. A confusion matrix is a matrix of size 2x2 in its simplest version that separates the results of the predictions obtained in true positives, false positives, true negatives and false negatives and this is shown in figure 2.2. These can be defined as:

- True Positives (TP) are the number of correct predictions for the positive class.
- False Positives (FP) are the number of predictions predicted as positives but are actually negative.
- False Negatives (FN) are the number of predictions predicted as negative but are actually positive.
- True Negatives (TN) are the number of correct predictions a model predicted as negative.

The accuracy gives the fraction of the correct predictions and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.17)$$

The precision gives the fraction of positive predictions predicted well and can be defined as:

$$Precision = \frac{TP}{TP + FP} \quad (2.18)$$

The recall gives the fraction of actual positives predicted well and can be defined as:

$$Recall = \frac{TP}{TP + FN} \quad (2.19)$$

The F1-score gives the harmonic mean between precision and recall, being defined as:

$$F_1 = 2 \frac{Precision * Recall}{Precision + Recall} \quad (2.20)$$

There are also different types of neural networks that can be built depending on the purposes and tasks to be done [43]. Some of the most used are: Feed Forward Network (FFN) [44], Recurrent Neural Network (RNN) [32], Long/Short Term Memory Network (LSTM) [45], Gated Recurrent Unit (GRU) [46], Auto Encoder (AE) [47], Markov Chain (MC) [48], Hopfield Network (HN) [49], Boltzmann Machine (BM) [50], Deep Belief Network (DBN) [51], Convolutional Neural Network (CNN) [52] and Transformers [53]. In this section, Convolutional Neural Networks and Transformers are explored, since these are the state of the art architectures used for models in computer vision and help achieve the objectives of the project.

2.2.1 Convolutional Neural Network

Convolutional Neural Networks are Deep Learning networks inspired by the visual cortex of a cat [54] that can be applied to many tasks, namely Computer Vision [55], Natural Language Processing (NLP) [56] and Speech Processing [57]. According to [58], they have three major benefits: parameter sharing, sparse interactions, and equivalent representations. Local connections and shared weights in the network are used instead of standard fully linked networks to fully leverage the two dimensional structure of an input data, as well as decrease the number of parameters and consequently the training times of the networks. This does simulate the process that occurs in the visual cortex cells, because they also focus on specific parts of the scenes rather than the full images.

Usually, these CNNs are composed of several convolutional layers, represented by three steps: two convolutions and a pooling step as represented in figure 2.3. The two first convolutions substitute the usual matrix multiplication with a convolution [44], and differ from one another due to the activation function used. While the first step uses a linear activation, the second utilizes a non-linear one in order to capture both linear and non-linear features. These convolutions are represented by a matrix multiplication between parts of the input and a smaller kernel/filter where these filters of three dimensions for width, height and channels are then applied through the tensor to generate important features based on local analysis.

The pooling layers reduce the dimensionality of the representations, which decreases the required necessary computation power and allows a more condensed analysis of the input for future layers. After these convolutional layers, normally, fully connected layers (layers where all the neurons are connected to the neurons of the previous and next layer) are used to analyse the produced features and finally obtain the result of the network.

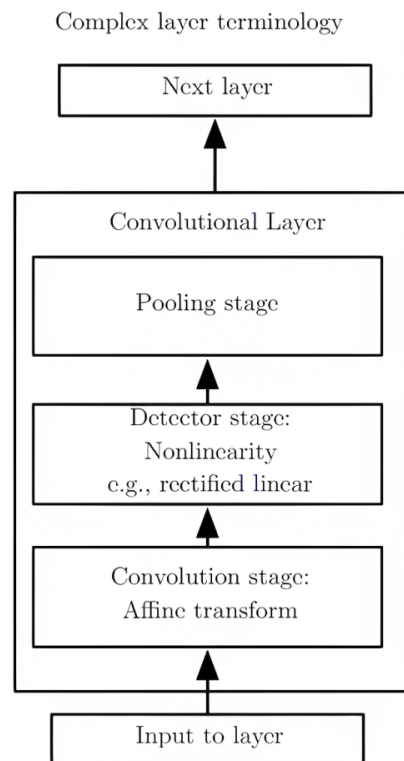


Figure 2.3: CNN convolution layer(from [44])

2.2.2 Transformers

Transformers are a neural network architecture that utilize attention to improve Deep Learning models and were first introduced in the paper [53]. They were initially used only for NLP tasks but are expanding now to other areas like computer vision, beating even the most famous CNNs that held the monopoly of classification. Figure 2.4 presents this overall architecture.

The architecture is divided in two different blocks, the encoder and decoder. For both, after embedding the input and output, a positional embedding is added. The encoder block is composed of 6 stacked layers with 2 sub-layers each. In the first sub-layer, the main transformer characteristic is used, the multi-head self-attention block followed by a normalization layer. The second sub-layer is composed of a simple Feed Forward Network and a normalization layer. The decoder block is similar to the first, it has 6 stacked layers but instead of two sub-layers per layer, it has three, two with a multi-head self-attention and one with a FFN. The extra layer is used to conduct multi-head attention on the output of the encoder. Another important change in the decoder block is that the first multi-head

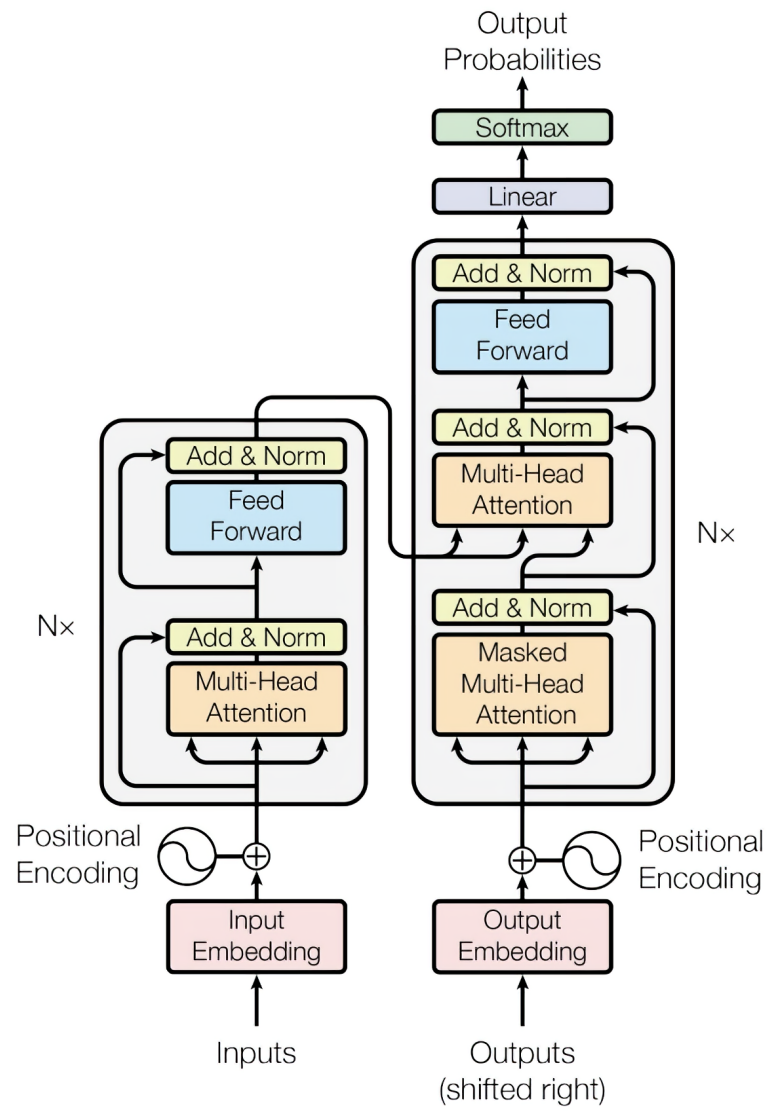
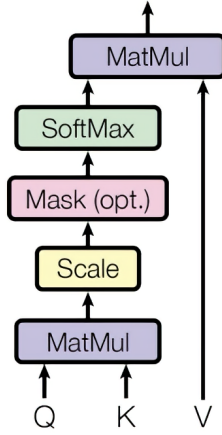


Figure 2.4: Transformer architecture(from [53])

attention is masked and the output embedding suffers from a 1 position offset to avoid position knowledge based on subsequent positions. This architecture also employs residual connections between sub-layers, which is explained further in this document, in subsection 2.3.3.

Scaled Dot-Product Attention



Multi-Head Attention

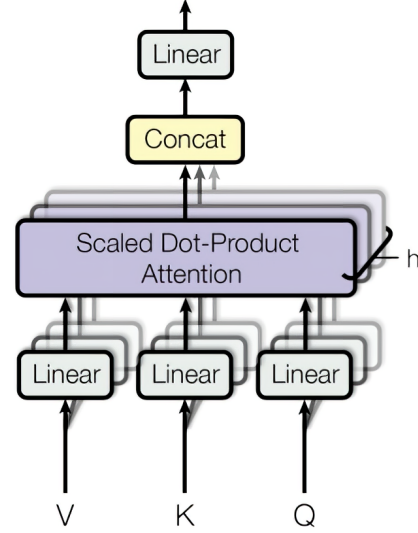


Figure 2.5: Attention and multi-head attention mechanism(from [53])

To understand multi-head self-attention, attention must first be explained. Attention is a mechanism that relates positions and indicates to one position what to focus on, around it, to help understand its meaning. For example, in a sentence, for a word, the attention mechanism will tell which other words in the sentence are closely related to it. For this, three vectors are necessary: the Query (Q), Key (K) and Value (V), which are obtained by multiplying the embedding by three trainable weight matrices. Then, the scaled dot-product attention occurs, present in figure 2.5. The dot product between Queries and Keys is calculated, followed by a division of each by the square root of the number of the Key values and an application of a softmax function to get the Values weights that are then multiplied by the Values. In reality, all these operations are done in matrices to increase the speed of processing and the formula can be represented as the equation 2.21 according to [53]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2.21)$$

where Q , K and V are the Query, Key and Value matrices and d_k is the Key dimension. The multi-headed attention then, improves the self-attention mechanism by allowing the model to focus on different positions and subspaces with multiple sets of Queries, Keys and Values depending on the number of attention heads, so that the number of sets is equal to the number of attention heads for encoder and decoder. Finally, in order to adapt the output of the multi-head attention layer to the feed forward layer, a condensation is performed by

multiplying the concatenated matrices by another weight trained matrix. This process is represented in equation 2.22.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (2.22)$$

where $head_i$ represents the output of each one of the attention mechanisms that occurs for different subspaces and W^O represents the weight trained matrix that will be used to adapt multi-head attention output to the next layer.

2.3 Models Architectures

In this section, a compilation of models architectures is presented, ordered by time, and focusing especially on the results achieved by the architectures in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for classifying images, that existed between 2012 and 2017. After that, the best models from each year according to [15] are described, if there are present some new architectural methods, or if the architectures are present in some of today's machine learning frameworks and other libraries. Here, table 2.1 also presents the top 1 and top 5 accuracies of the model architectures that are going to be explored with more detail. Most of the accuracies are from the best model variation of the year they surged with the exception of SENet, because such information was not available and that is also the reason why this model has a bigger accuracy when compared with its successor of the next year.

Table 2.1: Accuracies of the main neural networks architectures during the years

	Top 1 accuracy	Top 5 accuracy
AlexNet	63.3%	84.5%
ZFNet	64.0%	85.3%
Inception Network	69.8%	89.5%
ResNet	78.3%	94.0%
ResNeXt	80.9%	96.6%
SENet	83.8%	96.6%
PNAS-5	82.9%	96.2%
AmoebaNet-A	83.9%	96.6%
EfficientNet	86.9%	98.1%
ViT	88.6%	
CoAtNet	90.9%	
ConvNext	87.8%	
CoCa	91.0%	

2.3.1 AlexNet (2012)

2012 was marked by the appearance of AlexNet [59], a CNN that won the ILSVRC in that year by creating the biggest CNN ever seen with results superior to the state of the art until then and only limited by the GPU capacity. This was also very important for the evolution of neural network architectures because it showed how much better CNNs were when compared

to standard Feed Forward Networks used until then for image classification. Not only were they better but they also had much fewer parameters and connections, which decreased their training times. This network had 60 million parameters and 650,000 neurons and was constituted by 8 layers: 5 convolution layers along with their max pooling layers and 3 fully connected final layers accompanied by a dropout method, a new technique that appeared then to help reduce the overfitting of the model. Figure 2.6 shows the full architecture of AlexNet.

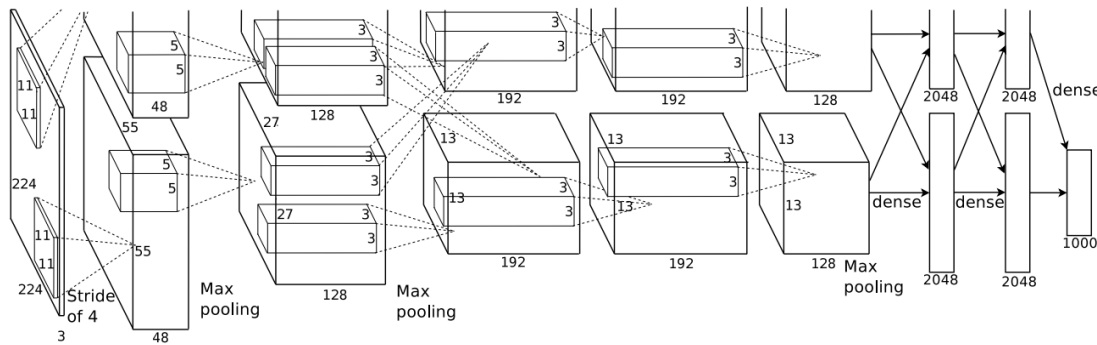


Figure 2.6: AlexNet's architecture representation (from [59])

Looking at figure 2.6, it is observable that the architecture requires images of resolution 224x224x3 pixels and applies convolutions with 11x11x3 filters with a stride of 4 pixels for the first layer, further reducing the size of these filters as the network gets smaller after the pooling is done to filters of size 5x5 and 3x3. These smaller filters present in the middle of the network, despite getting smaller in height and width, have increased depth to help deal with the continuous kernel growth, result of several filters being applied. In the end, three fully connected layers with 4096 neurons are used to deal with the features created by the convolutional layers and obtain the right classification out of 1000 classes.

Despite the results, the network suffered from overfitting, even with the use of data augmentation and dropout techniques, and the size of the ImageNet dataset [27], that was significantly bigger than the standard datasets that usually were used, like CIFAR [60] and NORB [61]. According to [15], the model reached a top 1 accuracy of 63.3% and a top 5 accuracy of 84.5%, very good for that time.

2.3.2 Zeiler Fergus Network (2013)

The ZFNet [62] is another architectural model, that won the ILSVRC in 2013 and that tried to solve the problem of not understanding why CNNs were able to produce such good results, that is, its main motivation was to find ways to try understand the decisions that are taken in the training process of the network so that future networks can be created without a pure tentative, failure process. This architecture utilized previous CNNs as a starting point, especially the AlexNet, already described, and continued to evolve its architecture by applying new methodologies and changing some aspects.

In order to solve the problem of understanding the inner workings of neural networks, [62] created the Deconvnet technique, that works as a convolution layer with filtering and pooling but in an opposite way, that is, this technique picks features created by convolution layers and reverses them back to pixels so that it is possible to see the images in each step

of the network training. This allowed them to see the evolution of the network. In the example given in the paper, it was noticeable that future layers focused on more complex features and complete objects present in the images, while initial layers looked for corners and edges/colours conjunctions. With this knowledge, the authors of the paper were able to realize some of the problems existing in the AlexNet architecture. A discrepancy of different frequency information was noticeable in the first layers, with little medium frequency information, as well as aliasing artifacts due to a big stride value used. Because of this, a new architecture was created and it is presented in Figure 2.7.

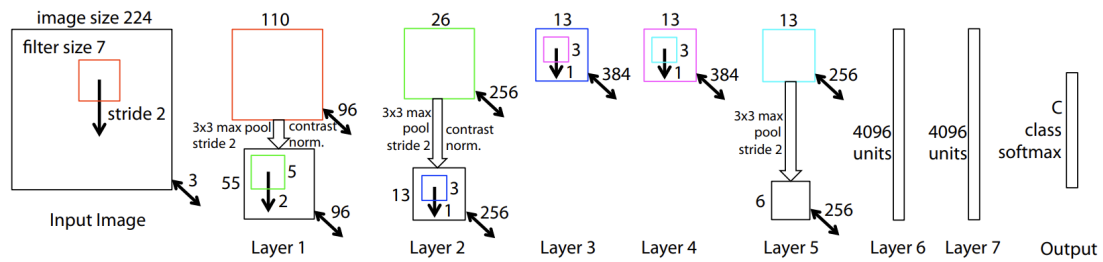


Figure 2.7: ZFNet architecture representation (from [62])

In image 2.7, it is visible what the authors did to resolve the problems explained and increase the quality of the architecture:

- A reduction of the stride of the network from 4 to 2.
- A reduction of the filter dimension from 11x11 to 7x7.
- An increase in the number of neurons of the first two dense layers to 4096 units.
- An increase in the kernel size of both layers and filters in the middle convolutional layers.

Finally, according to [15], this model obtained a top 1 accuracy of 64.0% and a top 5 accuracy of 85.3%.

2.3.3 Inception Network (2014)

The Inception [63] architecture was the big winner of the ILSVRC 2014 and did it so, while reducing the number of parameters when compared to the AlexNet model, in 12 times. This architecture started from two ideas:

- Bigger networks, both in depth and width, will be able to obtain better results.
- Bigger networks tend to cause overfitting and increase the computational resources required.

Because of this, the authors proposed then an inception module, that was capable of creating networks with bigger depth without the problems described. These inception modules would basically apply different filters with different lengths at the same level (1x1, 3x3 and 5x5 filters), with this constituting their naive approach. Their better approach consists in an inception module with dimensionality reduction that applies, before these expensive 3x3 and 5x5 filters, a 1x1 filter to reduce the number of channels, with the exception of the pooling layers where the 1x1 filter is applied after. Figures 2.8 and 2.9 present the two variations of the inception module.

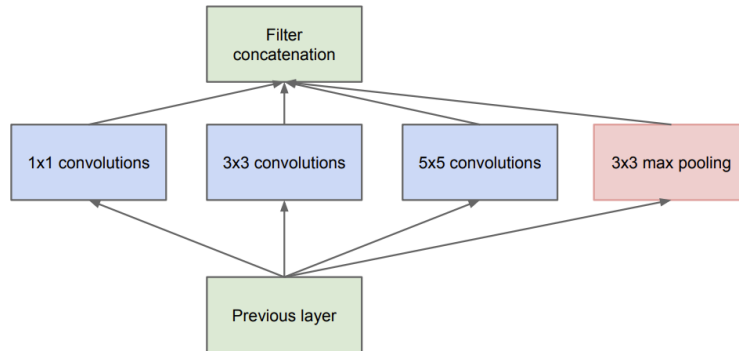


Figure 2.8: Inception module, naive version architecture representation (from [63])

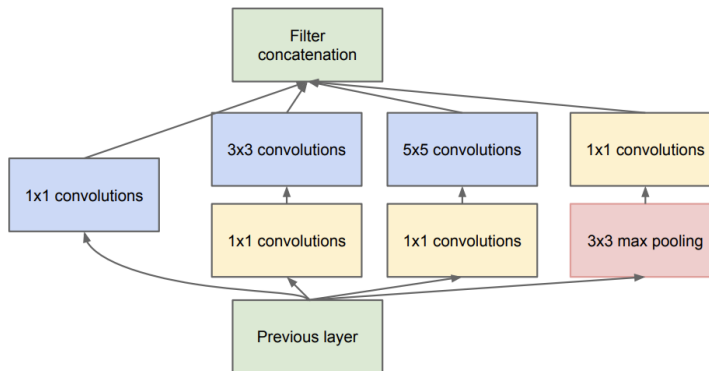


Figure 2.9: Inception module, dimension reduction architecture representation (from [63])

One application of this architecture and the true winner of the image net contest is the GoogLeNet architecture that utilizes dimensionality reduction inception modules and its architecture is shown in figure 2.10.

Here, an architecture of 27 layers was created, that expects an input of $224 \times 224 \times 3$ images and has 9 inception modules stacked. It is important to understand also that the columns *#3x3 reduce* and *#5x5 reduce* indicate the number of 1×1 convolutions done before and the columns with only the filter name represent the number of 3×3 and 5×5 convolutions applied in inception layers. The authors also tend to prefer pooling layers to dense layers in the end due to improvements in accuracy. Finally, because of the risk of gradient loss in such a big network architecture, the paper presents a methodology based on auxiliary loss functions in intermediate steps of the network, that will be weighted sum in the final loss function. According to [64], this network has a top 1 accuracy of 69.8% and a top 5 accuracy of 89.5 %.

2.3. Models Architectures

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 2.10: GoogLeNet architecture representation (from [63])

2.3.4 Residual Neural Network (2015)

The winner of the ILSVRC of 2015 was the Residual Neural Network [65] with the better variation of this topology in 2015, the ResNet-101 having a top 1 accuracy of 78.3% and a top 5 accuracy of 94.0%, according to [15]. This architecture emerged from the inherent problem of the existing architectures until that moment, the plain networks, that suffer from gradient vanishing and for that reason, make it impossible to increase their depth to get better results. The new approach presented by the authors, the skip connections, allow identity mappings to be performed between several weight layers. So, in this case, even if the degradation problem occurs, the identity mapping will help diminish its effects because the gradient will still reach earlier layers with adequate values. In figure 2.11, the architecture of a ResNet 34 is shown. The number after ResNet indicates how many layers the ResNet network has and it has been increasing in recent years.

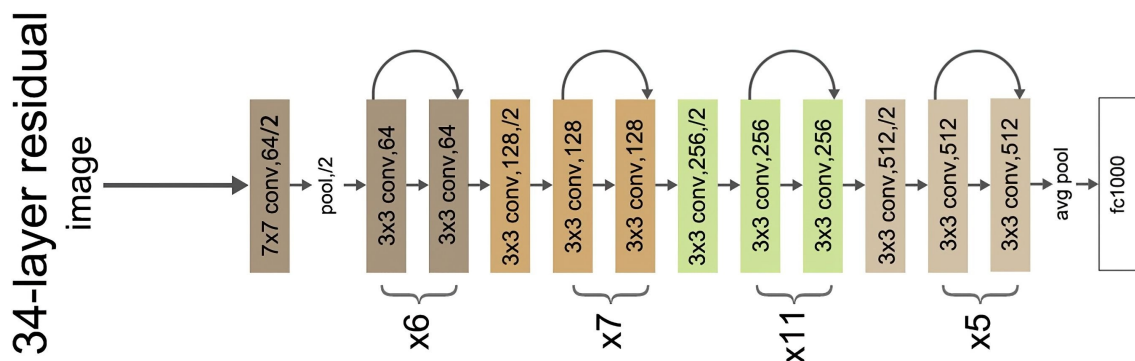


Figure 2.11: ResNet architecture representation (from [66])

Looking at the image, it is possible to see the residual blocks already described that will help reduce the vanishing gradient problem and that englobe two weight layers each. Once again,

the network receives images of 224x224 and uses mostly 3x3 filters with the exception of the first layer that uses a 7x7 filter and finalizes with a pooling and dense layer to predict one of the 1000 ImageNet classes.

2.3.5 ResNeXt (2016)

Although ResNeXt [67] is not the winner of the 2016 ImageNet classification competition, it finished in second and presents the next evolution for both the ResNet model and the models of that year, since the true winner of competition did not use a new architecture to win but a combination of previous architectures [68]. This architecture was built on top of what ResNet proposed and a new dimensionality per layer was added, the cardinality. This, basically defended the use of grouped convolutions instead of just one convolution per layer level, so that after all convolutions on one level have been computed, they can be joined together. Figure 2.12 presents a comparison/evolution of the ResNeXt architecture to the original ResNet Architecture.

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128, C=32 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256, C=32 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512, C=32 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 1024 \\ 3 \times 3, 1024, C=32 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		25.5 ×10 ⁶	25.0 ×10 ⁶
FLOPs		4.1 ×10 ⁹	4.2 ×10 ⁹

Figure 2.12: ResNeXt architecture representation (from [67])

Here, it is possible to see that despite this increase in cardinality, the number of parameters are similar and the network will increase the accuracy of the previous methods without increasing its complexity. [15] presents a top 1 accuracy of 80.9% and a top 5 accuracy of 95.6% for this architecture.

2.3.6 Squeeze-and-Excitation Network (2017)

Squeeze-and-Excitation Networks (SENet) [69] are the winners of the ImageNet contest of 2017 and introduce a new type of block that helps the network gain accuracy without

compromising in complexity. The idea is that the standard approach to channels' weights, where all the channels have the same importance when creating the feature map, can be improved. This is done by simply using a weight vector with the size of the channels where each position of that vector represents the importance weight of its corresponding channel. This process starts by first compressing (squeezing) the feature maps of each channel to a single value and then sending this vector through a two layer network that will produce the weight vector. Figure 2.13 presents the described process for the particular case of application of the SENet module to a ResNet.

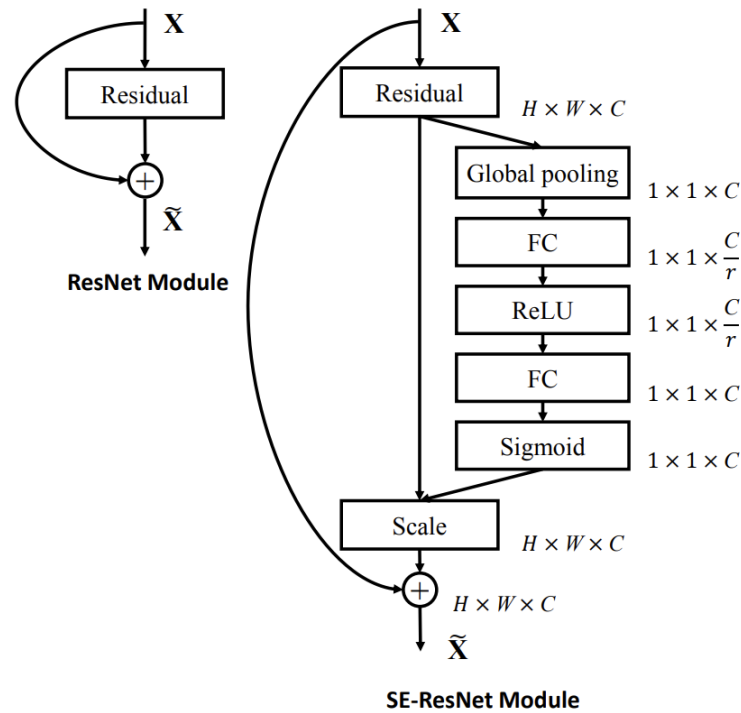


Figure 2.13: Comparison between the ResNet and the SE-ResNet modules (from [69])

This figure is very interesting because it shows how such a simple change can affect the performance of the model. The authors show that, with this module, a ResNet with 50 layers will almost achieve the performance of a ResNet with 101 layers, which means that, with half the layers and fewer computational resources, a simpler network can achieve much better results. In [15], it is shown that the best version of the application of this architecture (SeNet-350) from 2021 achieves a top 1 accuracy of 83.8% and top 5 accuracy of 96.6%.

2.3.7 PNAS-5 and AmoebaNet-A (2018)

In 2018, the search for better architectures paradigm changed. Besides some new ResNet models, there was a bigger focus on automatic machine learning architectures methods to make the process of creating network architectures better and automatic. The idea was creating methods, that given certain different blocks of layers would then pick them up, create a network by stacking those types of blocks, train and test the network for the created architecture and then tune it based on the results obtained [70]. Even though this process takes time, the datasets used for these tests are much smaller than usual training

datasets for real scenarios, because the authors defend that, if an architecture performs better than other for small datasets, the same performance relation will be maintained in bigger datasets.

Progressive Neural Architecture Search 5 (PNAS-5) [71] continues to build on this concept but stands aside from reinforcement learning and genetic algorithms to select the blocks and create the architectures, and supports itself on the use of a sequential model-based optimisation (SMBO) strategy, a smarter way of picking these blocks by searching for structures in order of increasing complexity, while concurrently building a learning model to direct the search across the structure space. This allowed them to achieve state of the art accuracies, with a top 1 accuracy of 82.9% and a top 5 accuracy of 96.2% [15], on the ImageNet dataset.

AmoebaNet-A [72] is another method with the objective of finding the architectures automatically, that modifies the tournament selection of evolutionary algorithms with the introduction of an age property, favouring the younger genotypes and applying the simplest possible set of mutations that permit an evolution in the NASNet search. This method performs better than the previously described one and has a top 1 accuracy of 83.9% and a top 5 accuracy of 96.6% [15].

2.3.8 EfficientNet (2019)

EfficientNet [73] picks up on the problem of scaling width, depth and resolution, by starting from a neural architecture search to obtain a new baseline model and then scaling it to obtain better accuracy. This process of scaling consists in obtaining the best values for several parameters such as width, depth and resolution of the input images. While previously, scaling has already been taken into account for the creation of several models, this process is not easy and usually results in the manual work of changing these values and trying to obtain better results. For this reason, the authors of the paper present a methodology called compound scaling method where they defend that width (w), depth (d) and resolution (r) can be scaled uniformly according to a compound coefficient ϕ , and influences d, r and w in the way represented in equations 2.23 to 2.25:

$$\text{depth} : d = \alpha^\phi \quad (2.23)$$

$$\text{width} : w = \beta^\phi \quad (2.24)$$

$$\text{resolution} : r = \gamma^\phi \quad (2.25)$$

ϕ is a coefficient specified by the user that defines how many resources are available and α, β and γ are constants that can be determined by a grid search method. The paper also presents how these values influence computation performance (FLOPS). While the depth influences it in a linear proportion, the width and resolution influence it in a quadratic proportion and in order to stop the models from achieving disproportional FLOPS, a constraint is defined as presented in equation 2.26:

$$\alpha \times \beta^2 \times \gamma^2 \approx 2 \quad (2.26)$$

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 2.14: EfficientNet architecture representation (from [73])

Figure 2.14 presents the architecture of version B0 of existing B0 to B7 models of EfficientNet, where each row presents a stage for the resolution and number of channels and this is the model that was obtained from performing a neural architecture search. With this model created, the authors then proceeded to the process of scaling it. First, they defined a value of ϕ constant equal to 1 and performed a grid search to find the best α , β and γ that comply with the constraint 2.26. Then, these values are fixated and the ϕ value is adjusted, forming the variation B1 to B7 of the EfficientNets. According to [15], the best version of this model in 2019 was the B7 that had top 1 accuracy of 86.9% and top 5 accuracy of 98.1%. More recently, new variations start to appear, EfficientNet-L2 for example, with top 1 accuracy of 90.2% and top 5 accuracy of 98.8%. For further reading, these variations are described in [74]–[77].

2.3.9 Vision Transformer (2020)

In 2020, some work related to the application of Transformers for computer vision tasks started to appear. The article [78] is one of these examples and it presents a Vision Transformer architecture based on the original transformer architecture presented in [53]. Although, Transformers are the best architectures for natural language processing, capable of processing over 10B parameters, first tests of this architecture in computer vision did not produce such results. This was due to the fact that Transformers thrive with bigger datasets than the ones that were used at first and when the authors trained these models on datasets with more than 14M images, they were able to obtain state of the art top 1 accuracy of 88.6% for the ImageNet dataset. Figure 2.15 presents the architecture used by Vision Transformer.

This figure shows that the process starts by dividing the images into several patches of the same size for height and width, each one with several channels. These patches are then flattened and sent to a linear projection where embeddings for each patch will be created, as well as a general embedding representative of the whole image. Position embeddings are also added to help the model understand their original positions.

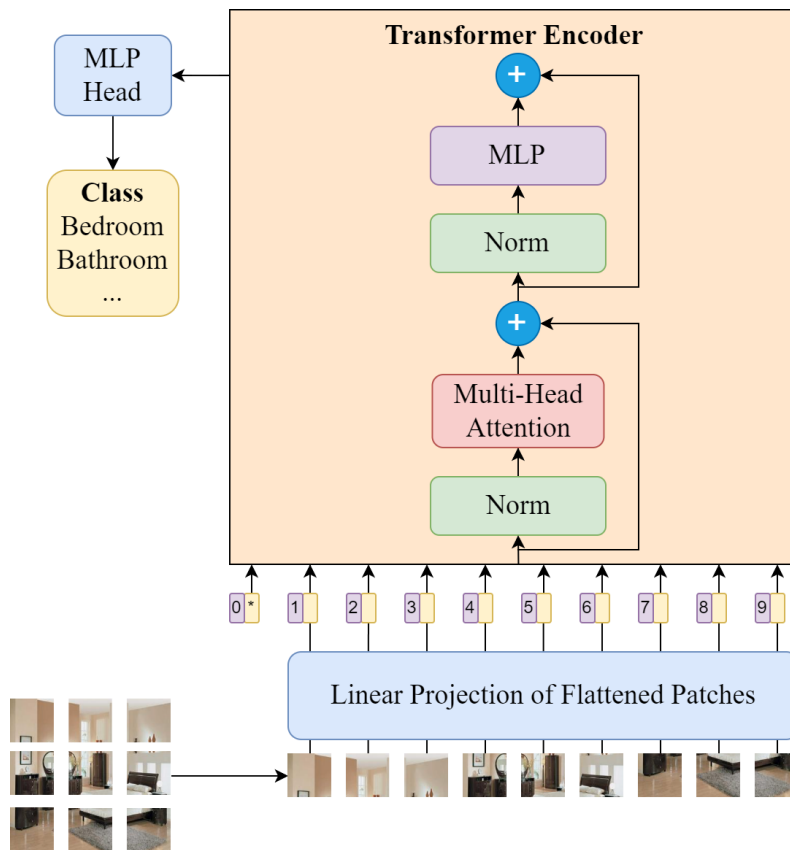


Figure 2.15: Vision Transformer architecture representation (based on [78])

After that, these patches will be used by the transformer encoder, which consists of several alternating layers of Multi-Headed Self-Attention (MHSA) and Multilayer Perceptron (MLP) blocks with a normalization applied before each one of them, as well as a residual connection after each block. The MHSA follows the architecture already described in the previous subsection 2.2.2, while the MLP consists in two linear layers with a GELU function for nonlinearity. Finally, a MLP head is added which consists of a simple fully connected layer to classify the outputs of the transformer block in the respective class.

2.3.10 CoAtNet (2021)

So far, it has been shown that CNNs have been the state of the art models for computer vision for many years, but more recently some work involving Transformers has also proved to be quite effective to the point of dethroning CNNs. CoAtNet [79] represents an approach that utilizes both architectures and unites them in one to create more accurate models.

This architecture utilizes an inverted bottleneck design, that expands the channel size 4 times and applies a depthwise convolution, which is just a convolution applied to each channel, bringing later the number of channels back to its original, which allows the use of residual blocks, and the self-attention mechanism already explained in subsection 2.2.2. By exploring the strengths of both methods, the authors reached the conclusion that convolutions are much better at understanding translation equivariance while self-attention understands input-adaptive weighting and global receptive field.

Translation equivariance refers to the capacity of the model understanding well the relations between samples rather their specific position, which self-attention has a hard time because of the usage of positional embeddings. The input-adaptive weighting refers to the capacity of identifying difficult complicated relationships interactions between spatial positions and it is where self-attention thrives. At last, global receptive field represents the ability of the models capturing contextual information and that was one of the main motivations to apply self-attention in these tasks. After this, the authors focused on how to join these two paradigms and the architecture present in figure 2.16 is the result.

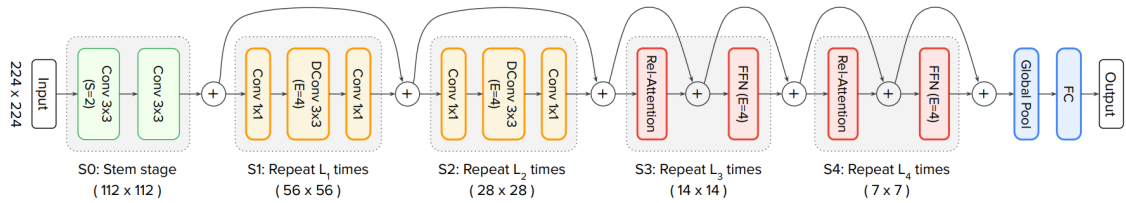


Figure 2.16: CoAtNet architecture representation (from [79])

Here, a 5 stage layout is presented where input images of size 224x224 are expected. The first stage, composed by two layers, a normal convolutional layer and a MBConv are used to reduce the dimensionality until it is enough for the output to be processed. In the next two stages, MBConv blocks are used, where the process described as inverted bottleneck design is used. The final two stages use a relative self-attention followed by a FFN that does the same expansion and compression described in the MBConv blocks and the final result is then passed on to a global pooling and a fully connected layer to obtain the final classification. This architecture has a top 1 accuracy of 90.9%.

2.3.11 ConvNext (2022)

In 2022, ConvNext [80] surges with the objective of trying to understand if CNNs are still able to compete with the new Vision Transformers that started to appear in 2020. It is a fact that Transformers have better accuracies on the major datasets used for benchmark, such as ImageNet [27], although the authors defend that CNNs already satisfy many of the desired properties for a computer vision model but tend to lose to Transformers due to MHSA modules.

To try to improve CNNs and help them make a comeback, the authors decided to start from a ResNet 50 and modernize it based on methods and characteristics from Vision Transformers, that they divide into two groups, macro and micro designs. The macro designs are:

- Changing stage compute ratio to be more similar with Swin-t [81], going from 3,4,6,3 blocks for each stage to 3,3,9,3 blocks.
- Changing stem to "Patchify", in order to follow the Swin-t transformer that uses non-overlapping convolutions and smaller patch sizes. So this changes the stem to a 4x4 convolution layer with a stride of 4.

The micro designs are:

- Replacing ReLU with GELU that is used more in Transformers.

- Fewer activation functions to try to replicate the behaviour in Transformers where there is only an activation function in the MLP block.
- Fewer normalization layers where two BatchNorm layers are removed.
- Replacing BatchNorm Layers with Layer normalization present in Transformers architectures.
- Separate downsampling layers, following what Transformers already do of adding a downsampling layer between stages.

Besides these improvements, the authors also presented more changes that they didn't really group in macro or micro design choices. These are:

- The inverted bottleneck module that has been presented in CoAtNet architecture.
- Increasing the kernel size and moving up depthwise convolution layers.

Training techniques that have been working well on these Transformers were also explored, like an increase to 300 epochs from 90 and the new AdamW optimization function that is used by Vision Transformers. These changes by themselves increase the accuracy of the model in around 2%.

All of these changes helped CNNs make a comeback, surpassing the accuracies of some Transformers and reaching values of top 1 accuracy of 87.8%, according to [15], despite still losing to many new Vision Transformers implementations.

2.3.12 Contrastive Captioner (2022)

Contrastive Captioner or CoCa [82] was until recently the best image classification architecture to ever be applied on ImageNet obtaining a top 1 accuracy of 91.0% and it is based on the Transformer architecture. Contrary to what previous models do, CoCa does not rely on image annotations and labelled vectors, but instead, prefers to utilize the knowledge of natural language when describing an image. The authors also explored previous works involving Transformers that use single-encoder, dual-encoder and encoder-decoder architectures and in this paper, they pretend to increase the efficiency of these strategies by combining the three into a single adapted encoder-decoder architecture. Figure 2.17 shows this approach.

Here, three elements are present: the image encoder, the unimodal text decoder and the multimodal text decoder. The image encoder corresponds to a Vision Transformer already described in this document while the last two elements are created by dividing the text decoder in two. The unimodal text decoder loses its cross-attention module in order to encode the text-only representation and prepare it for the contrastive loss calculation, while the multimodal text decoder suffers no alterations after this division. The process of training an image starts by dividing it into patches (the paper uses 18x18 patches for a 228x228 image) as well as tokenizing the text and feeding both to the image encoder and unimodal text decoder, respectively. The output of the encoder passes by an attention pooler that will help understand the image better, and it is then that the contrastive loss function is applied using the image and text representations as a whole, usually called [CLS] tokens. The remaining representations of the image and the text are then fed to the multimodal text decoder, which will learn image-text representations and calculate the captioning loss based on the output of this element.

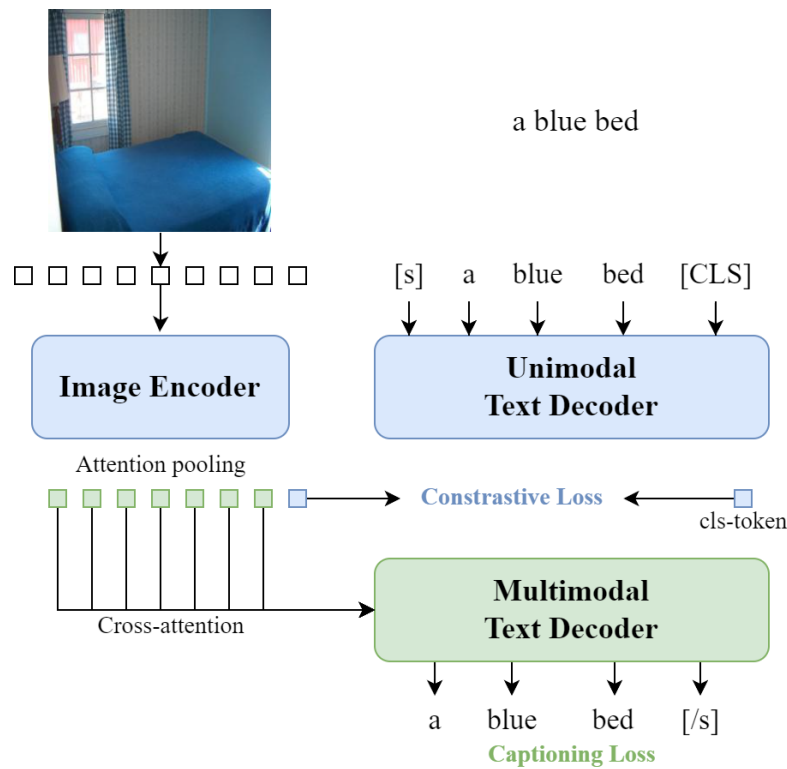


Figure 2.17: CoCa architecture representation (based on [82])

2.4 Indoor Scenes Classification

In this section, similar work applied to the classification of house divisions is going to be explored in order to identify the techniques and proceedings that have been developed with similar objectives of the thesis.

Table 2.2: Research query

Domain	Keywords
Indoor classification	"indoor scenes classification" OR "indoor scenes labelling" OR "indoor scenes tagging" OR "real-estate labelling" OR "real-estate tagging" OR "real-estate classification" OR "room tagging" OR "room labelling" OR "room classification" OR "indoor labelling" OR "indoor tagging" OR "indoor classification"
Computer vision	"neural network" OR "deep learning" OR "computer vision"

This research was conducted based on the PRISMA methodology [83]. A query, that is presented in table 2.2 is defined and searched with, on several data sources, namely ACM Digital Library [84], IEEE Explore [85] and EBSCO Information Services [86]. Table 2.3 shows the number of papers that resulted from the query search and how many of those were relevant and are presented in the document. Since for the searches conducted, the number of resulting papers was substantially low with no duplicates, no further filters were utilized and all papers were analysed fully, until it was clear if they were relevant or not for the

problem in this document. Extra research was also performed on Google Scholar [87] and Medium [88], but without the use of the query. Then, in table 2.4, a summarised version of the papers presented is shown, focusing more on the datasets and techniques used in each one of them.

Table 2.3: Number of research papers retrieved per data source

Libraries	Articles	Relevant
ACM Digital Library	16	1
IEEE Explore	6	3
EBSCO Information Services	3	1

Table 2.4: Summary of datasets and techniques present in state of art

Paper	Dataset	Technique
[89]	10M tagged images from tagged images from hoteliers and a third party company	GoogLeNet and Inception-v3
[90]	Manual extraction of images present in the Housing.com website with 6 categories	Inception v-3
[91]	15620 images and 67 classes	Prototype based model that utilizes local and global information
[13]	Selected images retrieved from Label Me, Google Images and Caltech 101	Generative probabilistic hierarchical model
[92]	Scene15, MIT67 and SUN397	Sparse coding setting that utilizes global and local features in a CNN
[3]	REI: collection of images from real estate listings and other sources with 6 classes	LSTM accompanied by a fully connected network
[93]	REID: collection of images with 4 general classes and 8 specific ones	LeNet, ResNet and SVM
[94]	5000 images of floor plans crawled from the web	Classification model based on a C4.5 decision tree and YOLOv3
[95]	Places365	GoogLeNet
[96]	MIT-67, Scene-15, Sports-8, Graz-02 and NYU	CNN using a new feature descriptor
[97]	1500 photos taken under very different environment and lighting conditions for 10 different classes	Linear Discriminant Analysis (LDA) and Probability Density Response Maps (PDRM)
[98]	Graph dataset from 224 apartment layouts with 9 room types	Graph Neural networks (GNN) It tries to classify 9 room types and

In the end, the research focused on the 12 papers present in table 2.4, where there is a predominance of neural network techniques, such as the Inception and ResNet architectures.

These techniques have been experimented on several datasets with different characteristics and properties, and some of these will even be utilized for experimentation as described in subsection 3.2.1, like the MIT-67 [91], the SUN397 [14], [99] and the Places 365 [100] datasets.

One of the first examples is the system utilized by Booking.com [101] to attribute tags to images of different rooms and contexts of their hotels and provide their customers with the right context [89]. Two approaches were studied: using models that have been pre-trained and training the models from scratch. The data scientists decided to use two model architectures for the effect: GoogLeNet, based on Inception-v1 and Inception-v3. For the dataset, they used data from a third party company and tagged images from hoteliers, uniting both and creating a dataset with over 10M tagged images. For dataset treatment, several techniques were applied, more particularly:

- Data augmentation, which increased the labelled data by 10 times.
- Hierarchy definition to help classify those images that create doubts, like river view or sea view, but are part of a greater group, like water view.
- Application of several labels to each image.
- Data imbalance resolution by over-sampling

With all this in mind, the networks were trained using Tensorflow and the results were evaluated in precision and recall, where they tried to improve one without compromising much the other. Although concrete results are not provided, Inception-v3 produces slightly better results than GoogLeNet and the final results should be considerably good, since the system was implemented on their website.

RE-Tagger [90] is another real-estate image classifier example created with the objective of increasing user experience and reducing the time consumed manually tagging images, which they estimate that delays the process of uploading them by 40 hours. Once again, Inception-V3 is used for this task, even though this paper is much more recent than the Booking publication. Figure 2.18 presents the full architecture used.

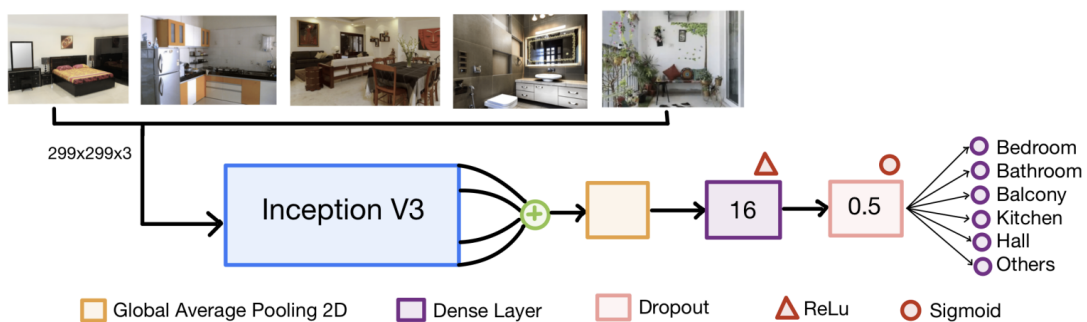


Figure 2.18: RE-Tagger architecture representation (from [90])

Here, a two stage transfer learning approach is used. First, the classic Inception-V3 network pre-trained with ImageNet weights is used, followed by an average pooling layer, a fully connected layer, a dropout layer and a softmax layer that will classify the images in one of six classes: bedroom, bathroom, balcony, kitchen, hall and others. The dataset was manually

extracted from images present in the Housing.com website and good results were obtained with the F1-scores for all classes above 0.88. Finally, the authors also make another remark related to the choice of Inception-V3, since they tested other models (ResNet, VGG and Xception) and ended up obtaining better results with this one.

The paper [102] presents a literature review of the application of computer vision tasks applied to real estate, such as the identification of the architectural style, the elements present, its values, other similar houses and of course, indoor tasks. For indoors, the paper presents, besides classification, reconstruction of 3D images of spaces and evaluation of the market value based on those indoors. For the problem in hand, the review presents 5 different papers that address it, although the last two only introduce a new dataset, the SUN 397 [14], [99] with 108,753 images of 397 categories.

The first one, [91] combines a set of global and local features to help the model. This is defended, because for some indoor spaces, global features will be sufficient to understand the space which is being dealt with, like corridors, while others, like bookshelves, make the classification task easier by identifying the existence of books. The dataset is constituted of 15620 images distributed by 67 classes and in order to apply the technique described, the authors utilized a human annotator to segment a set of images with their regions of interest (ROI). These scenes are not labelled, only identified, but the final objective is that these local properties help the model understand what type of room or division is present in an image, since similar objects will likely be in similar divisions. The results are quite heterogeneous for all the classes but that is not totally unexpected since the training was performed with a high number of classes.

The second paper, [13] presents a generative probabilistic hierarchical model, that utilizes the identification of objects present in rooms as intermediate representations to help classify them. Once again, the reason for this approach is that particular scenes will have particular objects that help models predict with bigger conviction levels. Here, a dataset with selected images from Label Me [103], Google Images and Caltech 101 [26] was used. The results were promising, with the authors identifying better performances when compared with Oliva and Torralba Gist approaches [104].

Finally, the third paper, [92] presents a sparse coding setting that utilizes global and local features in a CNN, in order to solve the problem of losing important local details due to the way weights are adjusted. It shows good results by being able to compete with previous approaches for some datasets, like the MIT67 [91], Scenes15 [105] and SUN 397 [14], [99].

Paper [3] presents an approach to facilitate the task of manually asserting real estate images to their right label by utilizing a LSTM accompanied by a fully connected network. For this task, the authors created their own dataset that goes by the name of REI and that has been collected from a series of real estate listing websites as well as other sources, namely Google. With the main objective in mind, the paper proposes three types of classification:

- Place classification, where rooms are classified in some category. These categories are: bathroom, bedroom, kitchen, living room, front-yard and backyard with more than 1000 images for each category.
- Counter-top classification that identifies materials from which objects are made.
- Floor classification in different classes, such as tile, hardwood, laminate and carpet.

For pre-processing, the authors utilized a technique called Contrast-Limited Adaptive Histogram Equalization (CLAHE) in order to improve low resolution images. Finally, for classification, two LSTMs are used, divided into h-LSTM and v-LSTM which learn the correlation between neighbouring pixels in horizontal and vertical directions, respectively. They consider 128x128, 256x256 and 64x64 images, as well a three aspects ratio: 1:2, 2:1 and 1:1. Both have 128 cells and use Adam optimizer, sending the output to a fully connected and a softmax layer that will allocate images to the corresponding class. Figure 2.19 shows the overall architecture that was described. When compared with CNNs, it shows competitive results with similar performances.

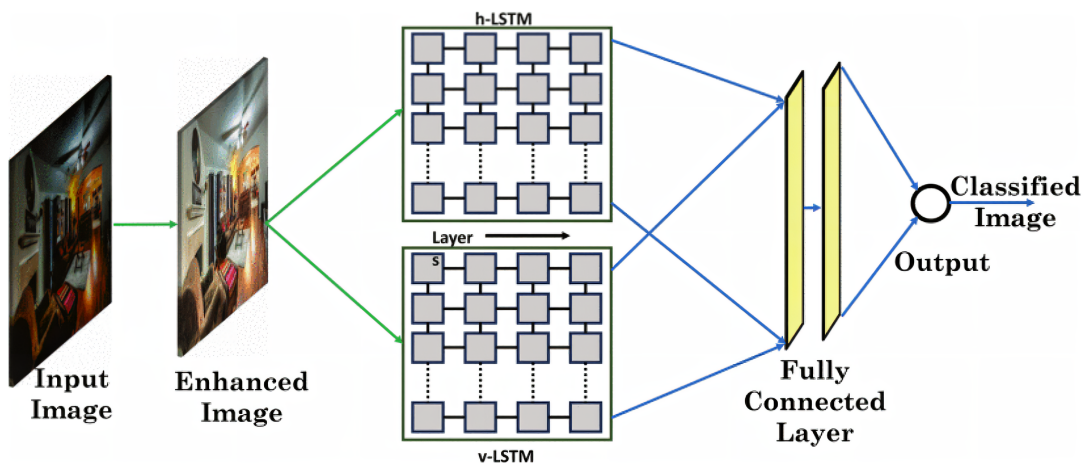


Figure 2.19: Overall architecture representation (from [3])

Paper [93] proposes two another approaches: using a LeNet architecture from scratch and transfer learning based on the fusion of ResNet and Support Vector Machine (SVM). For this task, the authors propose a new dataset, the REID created with 4 general classes and 8 specific ones (living room, dining room, bathroom, kitchen, bedroom, exterior, documents and other). The LeNet model has 4 convolution layers, with max pooling in between, finalized by two fully connected layers with 4096 units and a softmax. This architecture is present in figure 2.20. The pre-trained model freezes the last layer of the ResNet 50 model and replaces it with a SVM classifier as shown in figure 2.21. Both methods showed decent results with the LeNet model having a validation accuracy of 78 %, while the ResNet model has an accuracy of 87 %.

The paper [94] presents an approach that aims to facilitate the task of 3D room construction and decoration based on floor plans, by providing a classifier that provides textual information on rooms that are not described, which is required by the 3D models. For that, a door and window detection model is also developed using YOLOv3 to help their classification model based on a C4.5 decision tree perform better. The results were very promising with the classification algorithm having an accuracy of 78.71%.

The article [95] utilizes a GoogLeNet CNN to help classify rooms along with the Places365 dataset [100], which has 20 custom indoor room categories and uses its output to select binaural reverberation of the rooms. This approach obtained proportional errors between 11% and 16% for frequencies between 125 and 8000Hz, which the authors concluded to be better when compared to a blunt alternative system performance.

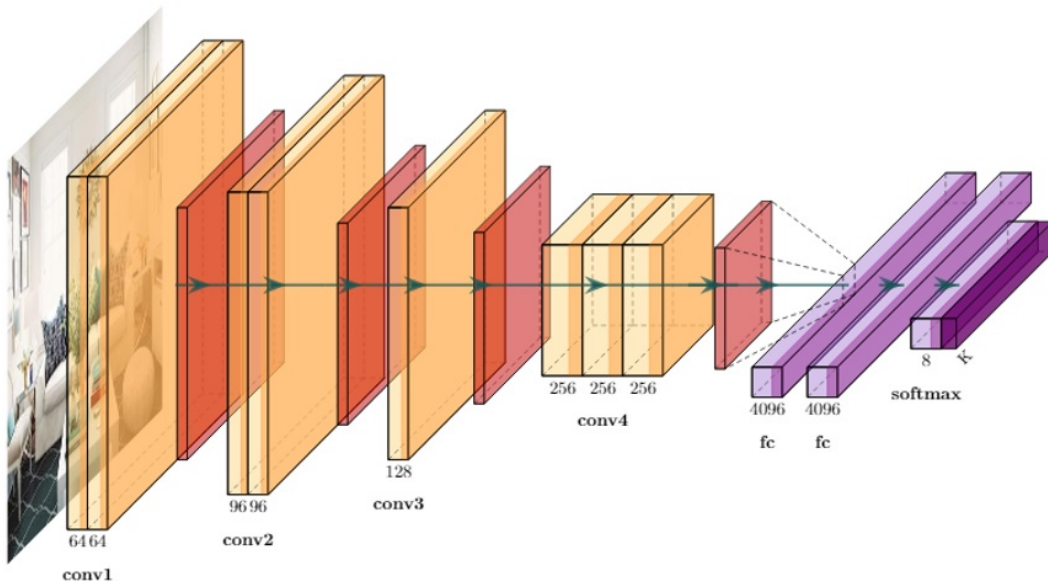


Figure 2.20: Overall LeNet architecture representation (from [93])

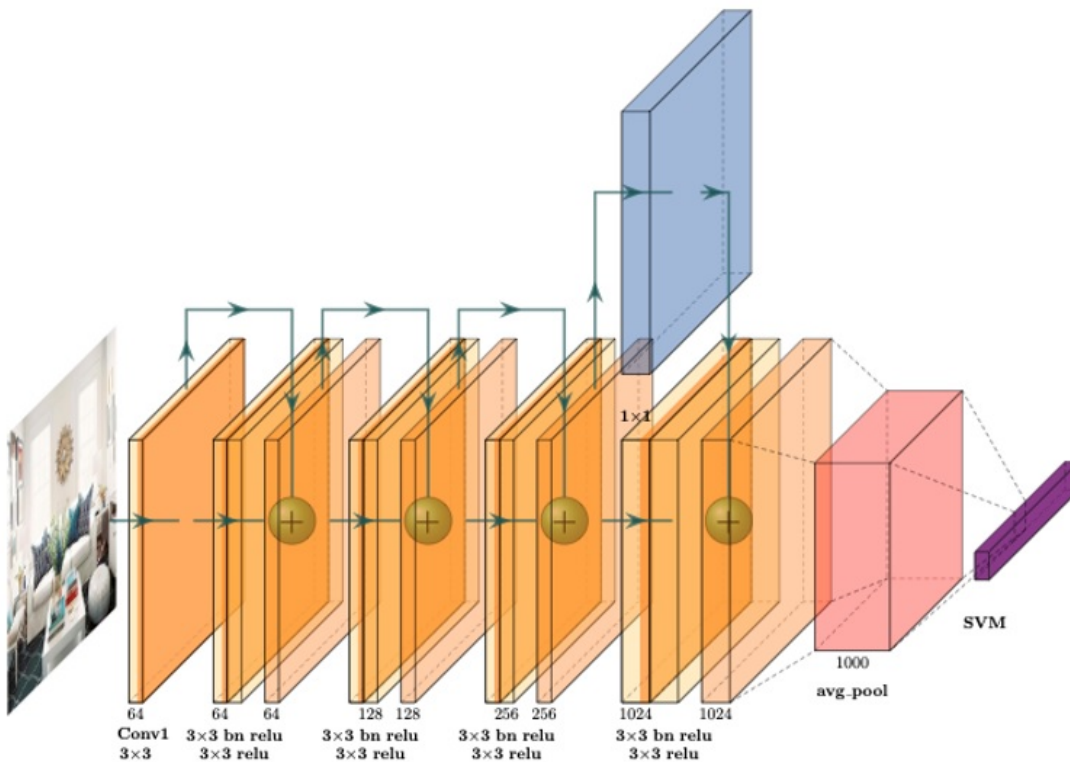


Figure 2.21: Overall ResNet 50 architecture representation (from [93])

The article [96] proposes a CNN that uses a new feature descriptor that deals with the challenge of large-scale spatial layout deformations and scale variations thanks to a new “spatially unstructured” layer that is able to implement robustness against spatial layout deformations and obtains better accuracies than the baseline models, like VGG and GoogLeNet for MIT-67, Scene-15, Sports-8, Graz-02 and NYU datasets.

The paper [97] shows an approach to classify both binary indoor-outdoor images and multi-class indoor rooms, that uses probability density response maps (PDRM) that indicate how likely each region is to be created by a class, extracting features from it and using them to train a Linear Discriminant Analysis (LDA) for classifying 10 different classes, with a dataset of 1500 photos taken in different conditions, obtaining accuracies of 90.1%.

Paper [98] proposes a new algorithm, SAGE-E that utilizes Graph Neural networks (GNN) to semantically enrich Building Information Modeling (BIM). It uses a graph dataset of 224 different apartment layouts and tries to classify 9 room types and achieves accuracies of 79%. It shows better performances than other algorithms, like decision trees while shortening the training and validation process.

2.5 Summary

The image classification problem is a field that has been addressed for many years now and has seen recently a shift from the classic CNN architectures to Transformers [78]. Although Transformers seem to be getting better results than CNNs [15], there are still not a lot of studies that focus on Transformers applied to the task of indoor classification, with Inception and ResNet being the main models chosen by the authors of some papers to address this problem, even in more recent years [89], [90], [93].

This attachment to CNNs is also seen in recent attempts for the continued use of CNNs, by either trying to increase their performance by analysing what they are missing when compared to Transformers [80] or combining them with Transformers for the tasks they perform better [79]. This thesis fills the gap by further analysing the indoor classification problem, and examining how several architectures of both CNNs and Transformers perform for several circumstances. For this, the results of the specific architectures of ResNet, EfficientNet and ConvNext are studied for CNNs, ViT and CoCa for Transformers. These are selected not only because they are state of the art, but also because these are the ones that still perform better and, inclusive, are used for the creation of better and improved variations every day [15]. This way, this study provides clear explanations of which architectures are better for the problem of indoor classification, concluding if Transformers are really taking over CNNs or if the authors of the similar work analysed are right in keeping using classic CNN architectures.

Chapter 3

Data, Methodology and Tools

This section presents data, methodology and tools that are used for the project construction. It starts by providing a summary of the existing frameworks and their comparison, moving then to how the considered dataset was created, an analysis of data privacy and protection, and ending with the work methodology, deployment and security analysis.

3.1 Frameworks

Since the project utilizes neural networks, existing frameworks created for these types of tasks have been explored from several sources [106]–[109]. These frameworks usually make the task of creating, training and testing neural networks a lot easier and already contain pre-trained models which facilitate the task of applying certain architectures to other datasets with fewer records.

3.1.1 Scikit-learn

Starting with scikit-learn [110], this is one of the most popular frameworks for machine learning because it focuses on a variety of tasks like classification, regression, clustering, dimensionality reduction, model selection and pre-processing, and disponibilizes several models other than neural networks, such as SVM, KNN, RF, among others. Its flexibility makes it good not only for training tasks, but also for integrating with other frameworks for pre-processing or evaluation duties. The only downside is that scikit-learn is not the most advisable library for Deep Learning tasks and there are many other frameworks which are more flexible.

3.1.2 Tensorflow

Tensorflow [111] is an open-source framework developed by Google perfect for Deep Learning and Computer vision. Due to being so popular, it makes the task of understanding and finding examples very easy. Like Scikit-learn, it also disponibilizes tools for creating models, deploying models on cloud and implementing MLOps. Besides this, it also contains several pre-trained models on the ImageNet dataset from several sources and articles, where some have been created by them, while others have been developed by the community. Another valuable asset is that Tensorflow is prepared to be trained in the GPU, and consequently, reduce training times, when used in a partnership with the CUDA toolkit [112].

3.1.3 Keras

Keras [113] is a high level wrapper around Tensorflow or GNTK that facilitates the creation of neural networks at a higher level than usually other libraries allow. It is very simple and quick to use and similarly to Tensorflow, it disponibilizes several pre-trained models, some of which are common to Tensorflow due to their high interconnection. According to them, Keras is also the most downloaded framework for machine learning in 2021, even though the matter of popularity is always questionable, due to other sources stating divergent results and it seems this relies more on preference. The interconnection with Tensorflow makes it so it inherits many of the advantages of Tensorflow like the distributed GPU training and the easy deployment of its solutions.

3.1.4 PyTorch

PyTorch [114] is another open-source machine learning framework developed by Facebook. Like the previous two frameworks, it is perfect for Deep Learning and computer vision and according to the sources cited at the beginning of this section, it is simpler and has better performance than Tensorflow. It is also lighter and more flexible, while still supporting multi-GPU training. In terms of pre-trained models, it also has many available, as well as some datasets to help users quickly start with the framework.

3.1.5 Hugging face

Hugging face [115] is a library specialized in Transformers implementations for NLP and Computer vision. It provides a lot of implementations for these two areas, as well as datasets, making it one of the best choices for tasks where Transformers are required. Another great advantage is that it can easily be integrated with the best Deep Learning libraries like Tensorflow, PyTorch and JAX, so it is very flexible and has been getting more popular recently with the increase in popularity and performance of Transformers.

3.1.6 JAX

Another framework, JAX [116] is a solution created and utilized by Google that makes use of XLA and autograd, just like Tensorflow to make high-performance research in machine learning. It was created with the objective of increasing the speed of training models by providing the functions to automatically distribute the model training through several GPUs in a much more efficient way, as well as facilitating the training in GPUs and TPUs by allowing the same code to work in both. Right now, JAX is still growing and even Google warns users that it is still just a research project and uses it mainly in research to speed up new discoveries when testing things out, so documentation and support are more limited than the other frameworks already described.

3.1.7 Fast.ai

Fast.ai [117] is a high-level library built on top of PyTorch that is very good in the process of accelerating Deep Learning training. It tries to make the process of training models easier, even for people outside of the machine learning context. It is very flexible, especially if used with PyTorch and the authors even describe how to easily convert code directly from PyTorch to Fast.ai. Some of the main advantages presented in the library page include a GPU-optimized computer vision library which can be extended in pure Python and a novel

2-way callback system that can access and change any part of the data, model, or optimizer at any moment during training.

3.1.8 Caffe

Caffe [118] is another library merged with PyTorch that is used especially for image classification and segmentation, but it is more directed to mobile platforms. It also presents its users with a set of pre-trained neural networks although it creates some difficulties when working with RNNs and sequence modeling.

3.1.9 Microsoft Cognitive Toolkit

The Microsoft Cognitive Toolkit [119] is a large scale production deployment framework that allows users to easily create and combine different model types but that has been deprecated and will not be further updated. This is a library more indicated for bigger projects that aim to utilize several servers at once to increase the learning process performance.

3.1.10 Framework Comparison

All the frameworks presented have their advantages and disadvantages but the main factors that were taken into consideration when picking one were the amount of documentation, support, and performance, as well as what DevScope utilizes and the pre-trained models available in order to increase training speed. From the analysis of these parameters, several libraries will be used, namely Keras, Tensorflow, PyTorch, Scikit-learn and Hugging Face, since these are the most used and fulfil pretty well all the characteristics that were described as necessary. Some of the other libraries have also been discontinued in the meantime and others have already been integrated into these, so utilizing them directly is not really a necessity because they will end up not offering much more. Keras, Tensorflow and PyTorch will all be used, mainly because they have different pre-trained models and this way, it is possible to analyse both the performance of the tools as well as the different architectures. Scikit-learn in this instance will have the objective of facilitating the evaluation of the trained models due to its very good evaluation features. Finally, hugging face will be used for the training of Transformers in junction with Tensorflow and PyTorch to help test this new rising architecture when compared to the classic CNNs.

Table 3.1: Comparison between machine learning frameworks used for training

	Tensorflow	PyTorch	Keras
Architecture	Complex	Complex	Simple
API level	High and low	Low	High
Speed	Fast	Fastest	Slow
Pre-trained models	Yes	Yes	Yes
Big Datasets	Fast	Fast	Slow

Table 3.1 also presents some comparisons between the training libraries according to [120], [121], at a general level. From it, it is possible to conclude that all of them are good choices and the differences come from the type of datasets used, or the development and

training speed of the networks. So, it is reasonable to conclude that for tasks that require fast training of methods, PyTorch will be better, but if the development of the network is somewhat simple and does not require low level definition of layers, Keras may be a good solution despite being slower at training. Tensorflow is now very integrated with Keras, although it can be used solo, and seems to be somewhat in the middle between what Keras and PyTorch can offer. Hugging face is not present in table 3.1 because it is used more as an auxiliary library for the creation of Vision Transformers and it is usually used along with the libraries presented.

3.2 Datasets

Another important section for neural network tasks is the dataset. A good, balanced dataset with plenty of samples of each class to predict can transform the good results of a model architecture into a great one and for that reason, it is important to create the most robust and complete dataset.

3.2.1 Datasets Creation

Two datasets were created for the purpose of studying the architectures of the models: a research and a private dataset. The first step for the creation of the research dataset was the search of several datasets adequate for the task of indoor classification, that were retrieved from several sources to obtain a significant amount of data. Even though this research produced a considerable number of datasets, further extracting was conducted through web scraping of more images to increase the diversity of the collected images. These datasets, due to non-commercial licenses, will only be used for the purpose of this dissertation experimentation. The final solution developed for DevScope will then utilize a private DataSet, property of DevScope for the effect. Table 3.2 presents the datasets retrieved, as well as information related to the number of classes and images of each one.

Table 3.2: Description of the datasets used

Dataset	Number of categories	Number of images
Places 365	20	99967
SUN 397	19	12453
Mit Indoor Scenes	67	15620
House Rooms & Streets Image Dataset	18	24907
Houses Dataset	4	2140
Web Scraping	7	369

The first dataset, [100] is a dataset for several goals such as scene context, object recognition, action and event prediction. Originally, the downloaded dataset had 1.8 million images for 365 categories, but it was filtered to only include relevant classes for indoor classification and some of the images were also removed due to the existence of duplicates between train, test and validation sets.

The second, [14], [99] is a dataset that contains 108,753 images of 397 categories, where each category has at least 100 images. It was created with the purpose of providing a better

dataset that was capable of capturing the largest set of scene categories. Once again, this dataset was filtered to remove the classes that had no interest in indoor classification and some duplicates that were present.

The third, [91] is one of the most famous datasets for this task and has 15620 images distributed by 67 classes, with no class with less than 100 images. It was created with the objective of helping with the problem of bad classification of the models that, despite understanding well outside images, had a hard time distinguishing some house rooms.

The fourth, [122] is a dataset with 25,000 images of rooms and streets, although only the rooms images and some of the street images that represent the outside of a home are going to be used, which is roughly about 24907 images dispersed in 18 categories.

The fifth, [123] was created for the purpose of the paper [123] for estimating houses prices with visual and textual information, and contains 2140 images, 4 from each home for each one of 4 categories of rooms.

And finally, the last one which was created from web scraping of Google images, consists of a few images that were fetched with the purpose of increasing some of the sets of images of the classes that were filtered due to duplicates and had their numbers shortened. Then, for creating the research dataset, the pre-processing steps present in figure 3.1 were taken.

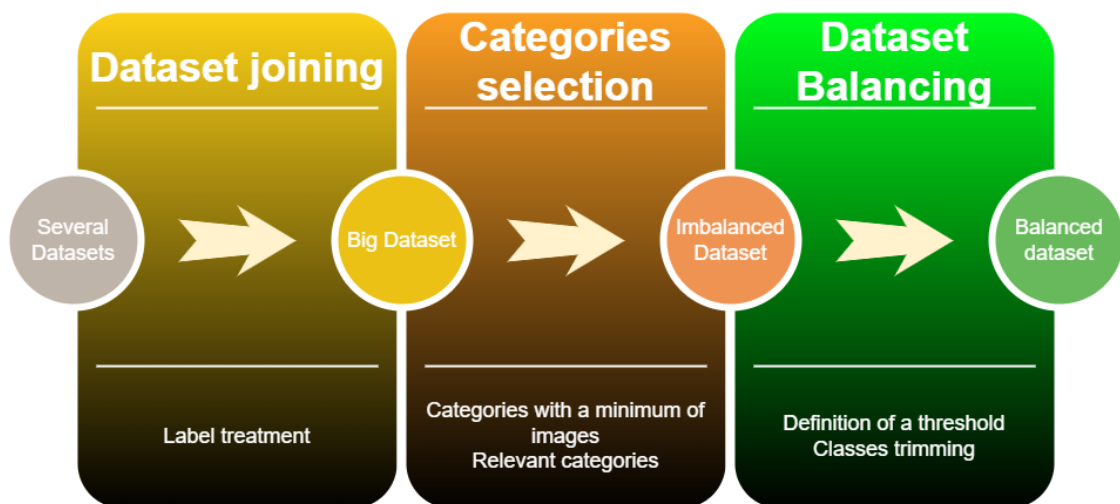


Figure 3.1: Dataset pre-processing

First, all the images retrieved were joined into one gigantic dataset, with the labels of the images passing through some treatment. This consisted in using a dictionary definition of synonyms for the various classes present and changing the different variations into the main synonym chosen since different datasets would have different tags for the same category and sometimes, even different labels would have the same or very similar images.

After that, an analysis of the obtained dataset was conducted to verify the number of images per class and the classes that have a significant amount of images were chosen in order for the model to have a minimum of registries and produce good results. This is performed because, although all categories could be included, utilizing those that have few images would result in the model being confused and probably not classifying well for these classes, besides some of them referring to other domains that have no interest for this project. From this analysis,

the following 15 categories were chosen: balcony, bathroom, bedroom, closet, corridor, diningroom, garage, kitchen, livingroom, office, outsideview, patio, staircase, utilityroom, yard. Figure 3.3 presents the number of images for each one of these classes for train and validation.

At this stage, despite the number of registries being quite good for all the classes, there is a problem, the balance of the dataset. Here, the number of images per class is not the same for all the classes, which may influence the model, when training, to better understand some classes than others. Another problem is that the datasets only contained training and validation sets, but no testing set, and it is ideal to have a test set to verify the efficiency of the model after training. To solve this problem, the training classes, which were the ones with more images, were randomly trimmed to increase the validation and test splits. In this case, the aim was to try to have a split of 10% for both the validation and test splits and after this process, 60000 images were available for training, 10500 images were free for testing and 9000 images were used for validation, which results in a split of roughly 76/13/11 % for training, testing and validation, respectively.

To reach these numbers and split the train images accordingly, firstly thresholds were defined based on the number of pretended images for each class. Here, 4000 was the accepted threshold for each training class, 700 for each testing class and 600 for each validation class, since this number allowed the aimed distribution split already explained. To achieve this, train images were transferred from its set to the validation and testing split until the threshold defined was obtained, while still leaving plenty of records for the training set. After this, it was important to make verifications regarding similar images in different splits, since usually, when using the validation and testing sets, it is ideal to ensure that the predictions that are occurring are based on never seen images, and for that reason, a tool for finding duplicates or similar images [124] was used to remove them. Although some similar photos were found, the dataset remained very balanced, so no more balancing was performed.

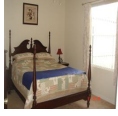
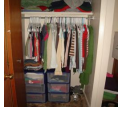
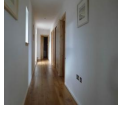
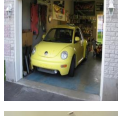
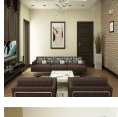
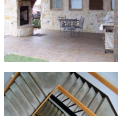
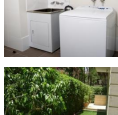
Finally, before the dataset was ready to be used for training, an adaptation of the image types was performed to a type supported by the libraries referred to in section 3.1, as well as a study of the resolution of the images present in the dataset and the ones used by pre-trained models to choose the best ones for training.

The private dataset of DevScope, on the other hand, consisted of images from a real estate company. This dataset had 10496 unlabeled images, so the Azure machine learning tool was used to speed up the process of manual labelling. This tool was capable of, given a certain number of labelled images, train a model based on those images and perform inference on the unlabeled, pre-labelling them. These pre-labelled images were then checked and corrected, if necessary, and this whole process started again until all the images were labelled. Contrary to the research dataset, this one is multi-labelled, which means that an image may have several labels. This is done because, in many photos, several different rooms are there represented.

With the labelled dataset, a simple holdout was performed on the dataset with a split of 80/10/10 % for training, testing and validation, respectively. This resulted in a balanced dataset between each split of each class, but unbalanced between classes, which means that despite each class having an overall number of images disproportional, inside of each one of them, the train, test and validation split maintained a close relation to the 80/10/10 % division. Despite all the limitations of this private dataset, it is important for testing more context applied images.

3.2. Datasets

Table 3.3: Description of the research dataset

Dataset	Image number (Train)	Image number (Validation)	Example image
Balcony	5014	302	
Bathroom	6141	1068	
Bedroom	6808	2204	
Closet	4995	416	
Corridor	5098	453	
Dining Room	6160	1298	
Garage	7197	318	
Kitchen	6588	1867	
Living Room	6293	2481	
Office	5015	328	
Outside View	24329	1999	
Patio	5000	305	
Staircase	4998	780	
Utility Room	5001	326	
Yard	5003	301	

3.2.2 Data Privacy and Protection

The creation and quality of the dataset is not the only important matter regarding data. As years go by, questions regarding data privacy and protection start to arise more and more, and regulations such as the GDPR [125] that gives users control over their personal data and how it is handled, start to appear. This regulation is applied by providing people with the rights to explicitly consent their personal data, delete it when they no longer want to give access to it [126] and define how it can be transferred. It also adds a security layer of protection over it, guaranteeing that only people with access can see it. This causes an impact on new software projects and solutions that are now forced to follow these regulations accordingly.

In this project, only data available for research has been used, retrieved from public online datasets and web scraping, as well as some free image providers, so data privacy problems are non existing for these. The same can not be said for the solution developed for DevScope, since the used dataset, code and models, are all property of DevScope, and for that reason are private. After the creation of the final model, the indoor images that will be retrieved from homes will require that the owners give their explicit consent for such, usually through established contracts where these points are well established.

3.3 Work Methodology

For the training of all the models present in chapter 4, the execution methodology present in figure 3.2 is adopted. First, the process explained in subsection 3.2.1 occurs, in order to produce the three datasets that will be used in this stage: train, validation and test datasets. Next, the two main model notebooks (for Keras and PyTorch) are used for all architectural tests. These utilize, besides the train and validation datasets, two other modules:

- A JSON file where the detailed information relative to the training of the model and the input is presented. This encompasses the specific name of the model present in the Keras or PyTorch libraries, the batch size and epochs for training, and the image resolution for resizing.
- A Python file where the model definition is present. Usually, the model definition consists of a pre-processing layer to adjust the inputs to the used architecture and the pre-trained model itself. After that, a global averaging pool layer, a dropout layer for regularization and a dense layer for classification are added, to substitute the previous existing classification layer that was removed from the pre-trained model. The last layer will have the same number of output neurons as the different training classes, in this case 15 and the final result will be a probability of each processed image belonging to each class. To test the usage of augmentation, this technique is applied sometimes before the pre-processing layer, with an augmentation layer that will apply random rotations, zooms, and flips. Another important aspect in the model definition is that several techniques of freezing and unfreezing layers are tested, as will be shown in section 4.1.

From here, the main notebooks start by retrieving a set of properties from the JSON file. After that, the training and test dataset are both loaded and the model is defined from the Python file. With the model and the dataset ready, the training is performed, utilizing a callback to save the best weights avoiding some cases where the last model produced is not the best due to overfitting, for example. The trained model is then used to evaluate

the test dataset and compute the results and evaluation metrics, that have already been described in section 2.2 for the classification problem. These consist in graphics of the evolution of the training of the model in terms of training and validation accuracy and loss, confusion matrices, accuracy, recall, F1-scores and the final test accuracy. Finally, the model is serialized and saved to be used later without the need for retraining.

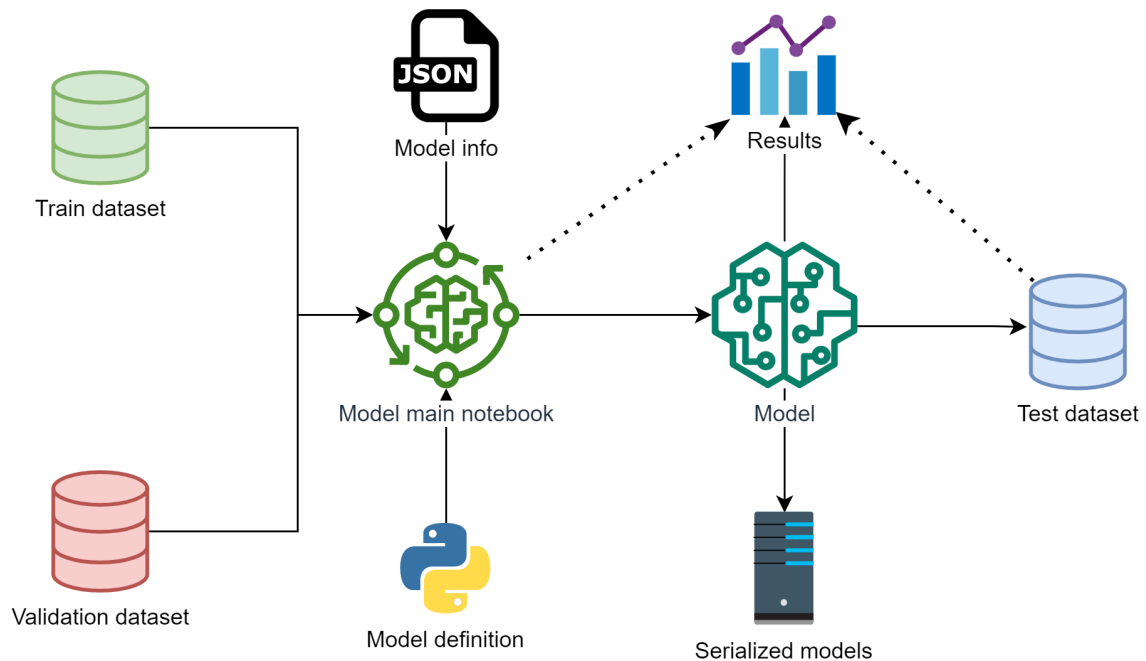


Figure 3.2: Full training methodology

3.4 Deployment

For deploying the models to make both their usage easier and quicker, several Python frameworks were analysed. FastAPI [127] is one of the main Python web frameworks, very suitable for deploying deep learning models. It is very fast, easy-to-use and production-ready with many important features like support for asynchronous programming. In comparison, Django [128] and Flask [129] despite, probably, being the most known and used Python frameworks are not as fast and were not used, since the inference with Deep Learning models is a heavy task and speed is a key factor. For this reason, FastAPI was the selected tool for the API creation, along with the conversion of the models to the Open Neural Network Exchange (ONNX) [130] format, very suitable and recommended for the deployment of heavy models in production due to its standardization and inference performance when paired with the ONNX Runtime [131].

Initial versions of the API contained three different model types: ConvNeXt Large, CoCa Base and CoCa Large. This version was used to make a few tests regarding the classification of the photos taken of the home of the author of the document to verify how well the system would do in a real situation. The most recent version contains only the versions of the CoCa architecture, since these are significantly better, as shown in chapter 4. Having both, the Base and the Large version available, gives the users an option of selecting a better performing or faster model, knowing that in most cases, the Base version will present similar results and that the difference between them will be mostly seen in more difficult images

and classes. In fact, the difference in performance between the best variations is only of 1% with the test dataset, which is very low.

The API has, in its latest state, two available requests. The most basic is a GET request to obtain the models available and select them in the POST request that receives either one or several images and classifies them according to the room division they represent. To increase performance in more prolonged tasks, the selected model is loaded the first time a POST request is performed and it is then maintained in memory for future requests, increasing significantly the speed of inference by around 6 or 7 seconds, as shown in the comparative tables 3.4 and 3.5. In these tables is also noticeable the difference in time performance between the CoCa Large and Base that tends to get bigger with a raise of the number of images to classify.

The models only stop being loaded in memory when the API is reinitialized or a request for a different model version available is sent, and in that case, the new version is loaded. To avoid conflicts when making requests since these are async, which means that several may be being sent at the same time to the API, a system of locks is implemented to avoid requests for one model, being executed by another.

Table 3.4: Loading and inference times of the CoCa models in the API for 5 images

Model	Loading and Inference (s)	Inference (s)
CoCa Large	8.16	1.19
CoCa Base	7.32	1.1

Table 3.5: Loading and inference times of the CoCa models in the API for 567 images

Model	Loading and Inference (s)	Inference (s)
CoCa Large	16.48	9.48
CoCa Base	7.97	2.03

3.5 Security Analysis

In recent years, there has been an increasing concern with the security of created digital solutions due to a rise in the number of cyberattacks and the costs it causes to companies [132], with some sources [133] stating a cost of \$10.5 Trillion Annually by 2025.

Obviously, Neural Networks also require a security assessment, since as with many other software solutions, there are breaches and weaknesses that can be exploited. Source [134] presents 5 types of attacks on neural networks, that can happen during the different stages of creating a model and these are:

- **Training set poisoning:** As the name indicates, this technique consists in introducing to the training dataset, poison information that will, consequently, create a defective model. There are several ways of performing these attacks, with the paper [134]

presenting the usage of a GAN to create poisoned data and another method that utilizes back-gradient propagation and it is capable of generalizing well to different models.

- **Backdoor attack:** This technique allows the attacker to create a backdoor hidden in the model, so that the model will work in perfect conditions until an input with the specified backdoor is sent, and in that case, predict the backdoor input wrong, according to what the attacker pretends.
- **Adversarial example attack:** These attacks try to trick the neural network model by altering the input in the inference phase in some way that makes the model predict a different class of what is expected. These attacks are most commonly tested and used with images, where the attackers try to change some of the pixels of an image resulting in a wrong prediction, even though, sometimes it is hard to detect these changes when looking at the altered image.
- **Model extraction attack:** This strategy is a rising threat, because it consists of attackers creating their own model, which will be similar to the one they are trying to extract by providing different inputs, observing the outputs and then using these results to train their own model.
- **Recovery of sensitive data:** This attack pretends to extract sensible information from the training dataset, either by inferring some information or verifying if some information was present for training.

Even though the model will only be available internally to a set of known and required people and most of these attacks will be very unlikely to happen, if there is a security breach in the network, the models can be compromised and for that reason, for each attack, a set of defensive measures is presented [134]:

- **Training set poisoning:** When the model is updated, the loss of the model should be verified and if a value above a certain threshold is obtained, there is a chance of data poisoning and the model should be discarded.
- **Backdoor attack:** Several approaches can be used here, with Neural Cleanse, fine-pruning and re-training the model to forget a Trojan, being a few examples that should be tested.
- **Adversarial example attack:** This attack can be avoided by training a robust model, where random transformations can be applied to the input, working a bit like the augmentation of the data.
- **Model extraction attack:** Here, an analysis of the distribution of the requests and a detection algorithm for verifying anomalous continuous queries should increase the defence.
- **Recovery of sensitive data:** Since the dataset used is constituted by public images, this attack does not present a big threat.

Besides this, there is another security problem, but not related to the neural network. These indoor photos contain information not only related to the contents inside the house (privacy issues already addressed in subsection 3.2.2) but they may also increase the risks for intrusions or robberies by providing details regarding access points and weak spots. Unfortunately,

for this kind of problem, there is not a lot that can be done, with the exception of trying to ensure that all security measures have been taken in the house to avoid these situations.

3.6 Summary

This chapter started by presenting several tools available for the neural network classification task. While certainly, all of them have their merits and advantages, four were selected for the task of actually training the models due to their selection of good pre-trained models and high popularity and performance: TensorFlow, PyTorch, Keras and Hugging Face.

Regarding the data, several datasets were retrieved from online sources and some web scrapping was performed, to retrieve more images to complement the and test dataset, with many different examples that will allow a good generalization of the model. The final dataset had a number of 60000 training images, 10500 testing images and 9000 validation images.

Related to the aspects of data protection, due to the nature of the project, there is little few issues related to it due to the research purpose nature of the images used for this task, with protection concerns only related to the algorithms and images retrieved from real house owners.

The work methodology adopted makes use of the created dataset and pre-trained models available in Keras and PyTorch and creates models for different architectures that are then used for evaluation and results analysis.

Security, on the other hand, requires special attention mainly due to the classic problems and attacks that can be conducted in the field of machine learning, with adversarial example attacks being the most common [134].

Finally, the best created models were deployed to an API by using a good performing framework, such as FastAPI and by converting the models to the ONNX format, that together with ONNX Runtime are capable of speeding up the inference time of the best models.

Chapter 4

Experimentation and Discussion

In this chapter, a series of experimentations are performed for a diverse set of architectures from Convolutional Neural Networks to Vision Transformers. A model analysis is also presented, in a discussion that focuses not only on which general architecture is better, but also on how particular architectural styles influence the results.

4.1 Case Studies Overview

In this section, a set of case studies regarding the application of different neural network architectures are presented, where several Convolutional Neural Networks (ResNet, EfficientNet and ConvNeXt) and Vision Transformers (ViT and Coca) are shown. These architectures will be analysed in terms of how good results they produce for the classification of the indoor dataset with 15 classes. For analysing the trained models three methodologies, which are present in figure 4.1 are used:

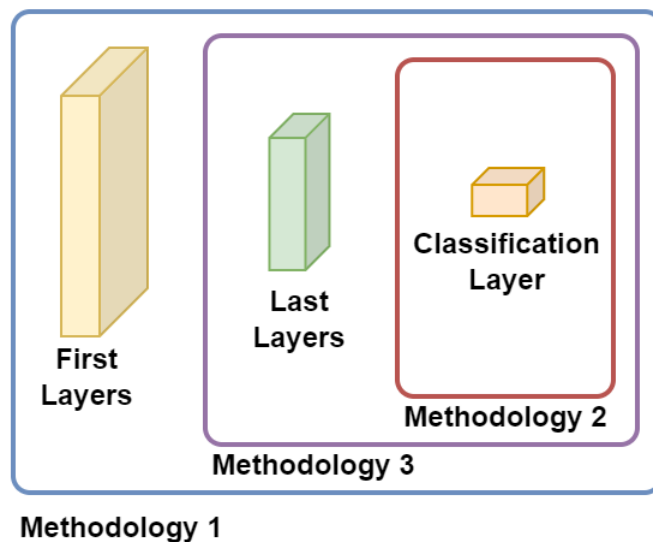


Figure 4.1: Scope of trainable weights for each methodology

- **Methodology 1:** Training all the weights available by unfreezing all the pre-trained model layers. This methodology requires low learning rate to avoid overfitting the model, since the first, more general layers are being trained.
- **Methodology 2:** Leaving all the layers of the pre-trained model frozen, acting like a feature extractor in a first instance, and unfreezing all or some of the last blocks of the

network when the results are minimally good, retraining, with a low learning rate, the unfrozen layers. This methodology is based on the fact pre-trained weights are usually trained on millions of images and present good performances without retraining.

- **Methodology 3:** Unfreezing only the last blocks of the pre-trained network and train these unfrozen layers from the start. This methodology follows the idea of the previous one, but builds up on the fact that the last layers usually produce more specific features related to the images used in training.

To make comparisons between models mostly focused on the methodology and architecture, the resolution, train, test and validation datasets were maintained throughout the experimentation, while the hyperparameters, such as batch size and learning rate, were tuned to produce the best model performance for each methodology. In this section, to only keep the results of the most important metrics, only the evolution of the models' accuracies, the confusion matrix all vs all and the F1-scores are shown, because these are good enough to give an evaluation of the models globally and by class, and because most of the other metrics that were described to be computed in section 3.3 and formalized on section 2.2 can be inferred indirectly through these. The tables of this section include different colors that highlight the quality of the achieved results, as follows. Red: below 80%, yellow: 80% - 84%, light green: 85% - 89%, dark green: 90% or above.

For this analysis, a GeForce GTX 1080 Ti with 11 GB of RAM was utilized and in some cases due to the size of the created models some down tuning of hyperparameters, such as the batch size was performed, as well as some extra freezing of layers of the pre-trained models to reduce the number of updatable weights, in order to balance the model produced and the capability of the given hardware. The cross entropy loss is used because it is the one indicated for multi-class problems, as well as the Adam optimization function since it was the one that originated better results after fine tuning. For the best model present in this document, the recent Lion optimization function used in the current best image classification model applied to ImageNet is also tested, in order to help understand how it affects the performance of the model. In the end, after the detailed analysis of each architecture and methodology has been concluded, a study is made on how multi-label classification can help some of the problems detected throughout the analysis of the results and a general comparison is presented between all the models.

4.2 ResNet

ResNet was one of the first tested models, and the Keras implementation was utilized, because it already contained pre-trained weights on the ImageNet dataset, helping the model perform better in its task. Of the available versions of ResNet, the ResNet 152v2 was picked due to being the one with better accuracy available. The 224x224 resolution was chosen due to most original pre-trained models using this resolution. This way, the versions of the models that use the 224x224 resolution were selected, which made comparisons easier to perform and allowed a maximum batch size of 64. As described before, the architecture used here will include a pre-processing layer, the pre-trained ResNet model layer, the average pooling layer, the dropout layer and the final classification dense layer.

The models of this architecture produced somewhat good results with the evolution of the training and validation accuracy during the training of the models, for each methodology, being present in figures 4.2, 4.3, 4.4 and 4.5.

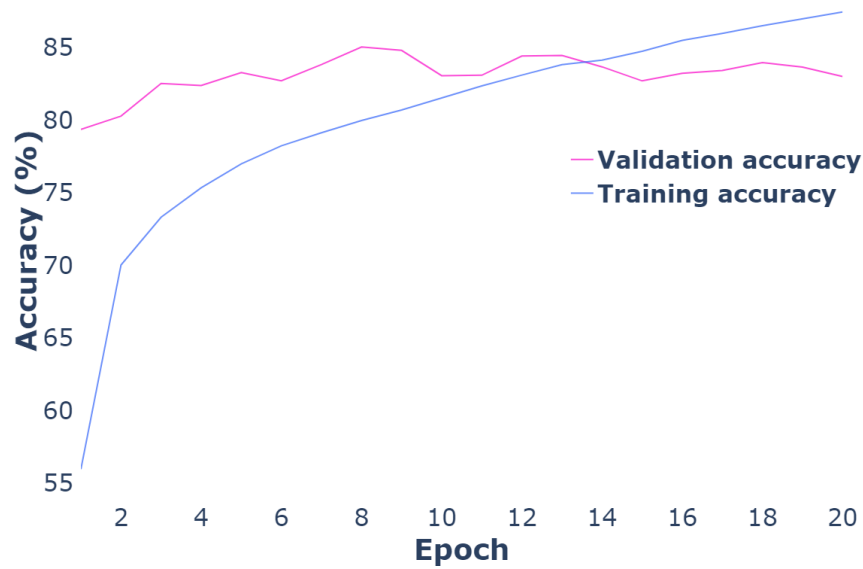


Figure 4.2: Results of the model based on ResNet 152v2 with Methodology 1 - 24 batch size - 224x224 resolution

Starting with the first methodology, in figure 4.2, it obtained a top validation accuracy of 85% in epoch 8, with a training accuracy of 80% and a test accuracy of 82%. Here, a batch size of 24 was used since it was the one that produced better results and could be allocated in memory. It is visible that the validation accuracy does not have a big variance during training and that once the training accuracy reaches similar values, the validation accuracy starts to stagnate and even diminish in some epochs, like in the 10th and 15th epochs. These are not bad results for this type of network, even though they still have many flaws.

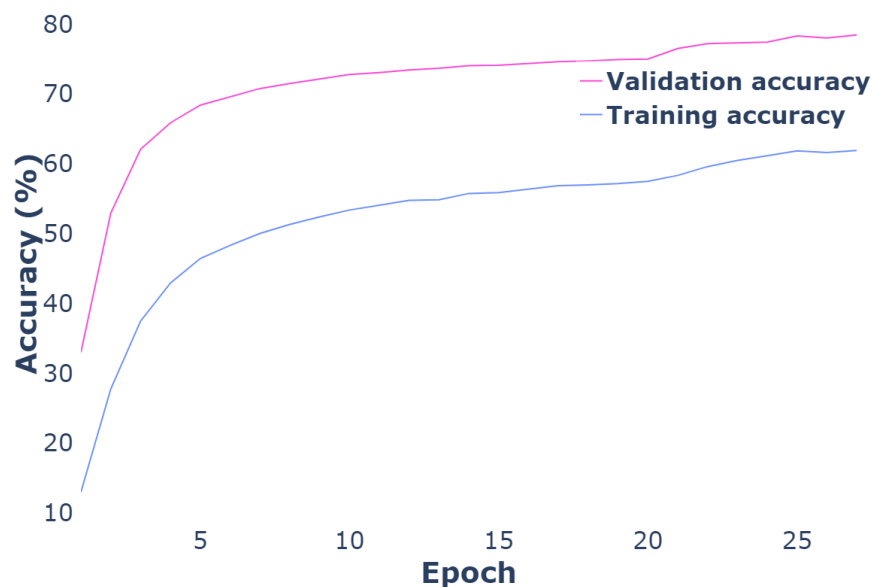


Figure 4.3: Results of the model based on ResNet 152v2 with Methodology 2 (Feature extraction step) - 24 batch size - 224x224 resolution

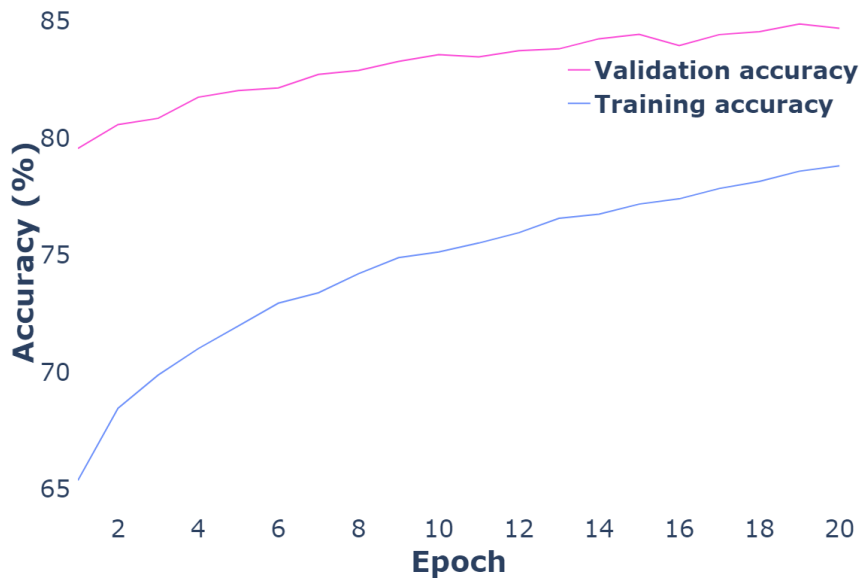


Figure 4.4: Results of the model based on ResNet 152v2 with Methodology 2 (Fine tuning step)- 24 batch size - 224x224 resolution

Looking at the results of the second methodology, in figures 4.3 and 4.4, the variation of the validation accuracy is a lot more noticeable due to the fact that the pre-trained ResNet has its weights frozen which means that the adaptation of the model will be slower, because there are less trainable weights to adjust. Despite this, it reaches a validation accuracy of around 78% at the end of the feature extraction process and when all the other layers were unfrozen, it was actually capable of reaching a top validation accuracy of 85%, the same as the previous methodology, although the training and testing accuracy were a bit lower, with values of 79% and 81%, respectively. This model could still obtain better accuracies if it was left training for longer, but since the improvement rate of the validation and training at the final stages was extremely low (less than 1%) and there was a risk of overfitting, the training was cut short.

Finally, for the third methodology, in figure 4.5, the model was left training for 51 epochs, since its training convergence was proving to be quite slow due to part of the pre-trained model being frozen, which means that only some layers were having its weights updated, taking more time to train the model, despite the validation accuracy staying more or less stable throughout the training. Here, it is visible that the best obtained result for the validation accuracy occurred in epoch 33 with a validation accuracy of 86%, which resulted in a training accuracy of 85% and a test accuracy of 82%.

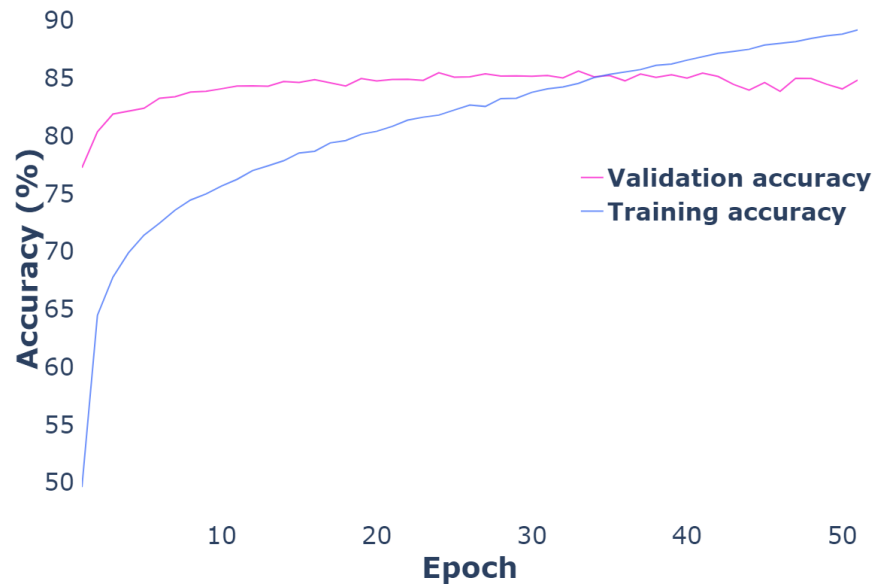


Figure 4.5: Results of the model based on ResNet 152v2 with Methodology 3 - 64 batch size - 224x224 resolution

Looking now at the F1-scores for these methodologies, table 4.1 presents a summarized version for each class of their F1-scores for the different methodologies. As was already pretty perceptible before, the models have very similar results, and this table helps in demonstrating the problems related to the living room, kitchen, dining room, patio and yard classifications. The low F1-scores for these classes come from the fact that these spaces coexist usually without any physical barrier between them, which means that photos will have several classes in one photo, as shown in image 4.6.



Figure 4.6: Examples of photos with multiple rooms (from [100])

The first image can be both living room and kitchen, while the second can be an outside view or a yard.

From this fact, two problems originate: Either the model predicts one of the classes present in the picture but not the one used as a label, so, even though the accuracy gets lower, the model is actually understanding what is present in the images. Or, the model predicts

a class that is not present in the picture due to some confusion with similar elements that appear in more than one label, like the tables of the kitchen or the tables of the dining rooms, for example. All this means that the F1-scores present in this and future tables for other architectures would probably be slightly higher for the cases where the model predicts other room present in the image that is not the identified as the correct label. In Appendix A, a further analysis of the confusion matrices of the ResNet models is provided.

Table 4.1: F1-score for ResNet models

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	82	83	82
bathroom	87	87	87
bedroom	86	86	87
closet	87	86	86
corridor	88	88	87
diningroom	77	77	76
garage	85	83	85
kitchen	80	76	78
livingroom	75	73	74
office	81	81	80
outsideview	82	82	83
patio	71	72	73
staircase	84	85	84
utilityroom	81	78	81
yard	79	81	80
overall	82	81	82

4.3 EfficientNet

As pointed out before, EfficientNet was another of the tested model architectures. In this case, the EfficientNet V2L version from Keras was utilized, since it was the one with the best top-1 accuracy, according to [135]. As usual, the resolution 224x224 was used to keep comparing the architectures in the same conditions or similar when hardware or other conditions do not allow it. Unfortunately, here, the greatest batch size was 32, because it was not possible to allocate bigger batch sizes into memory, due to an increased size and complexity of the models when compared to the ResNet.

Once more, the evolution of the models is presented in figures 4.8, 4.9 and 4.10.

Methodology 1 results are shown in figure 4.7. Here, it is visible that the model rapidly reaches its top validation accuracy of 84 % in epoch 2, starting then to decline until the end of the training, due to overfitting of the model. The fact that in this Methodology, all the weights start as trainable and start to be adjusted from the beginning may be the cause of why the model quickly reaches its top value in the first epochs and then enters in overfit, even with a low learning rate.

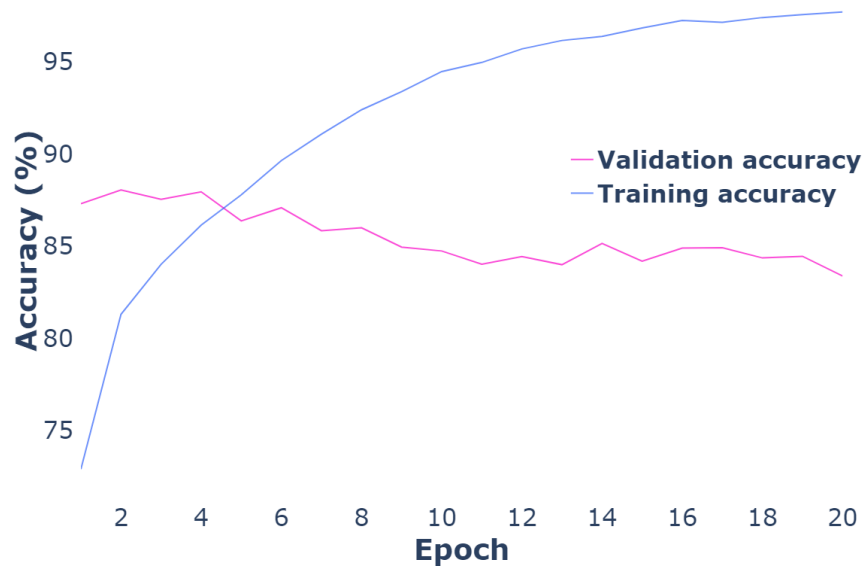


Figure 4.7: Results of the model based on EfficientNet V2L with Methodology 1 - 8 batch size - 224x224 resolution

For the second methodology, present in figures 4.8 and 4.9, the results obtained are the best so far, with a training accuracy of 91%, a validation accuracy of 88% and test accuracy of 86% in epoch 20. The evolution of the validation accuracy follows the trend of the training accuracy, with the exception in figure 4.9 after epoch 14, where it was noticeable that an increase in the learning rate of the model could be beneficial, and so the model converged quicker and the validation accuracy stagnated a bit after that.

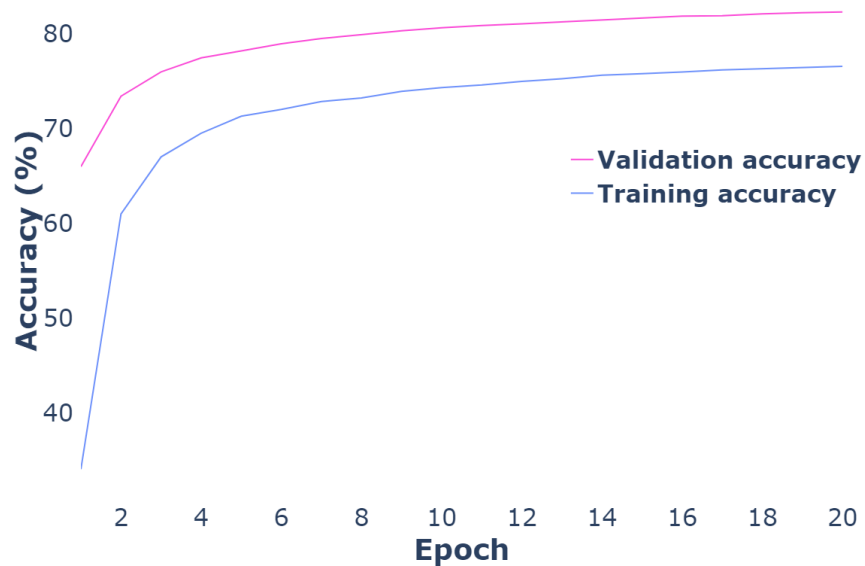


Figure 4.8: Results of the model based on EfficientNet V2L with Methodology 2 (Feature extraction step) - 32 batch size - 224x224 resolution

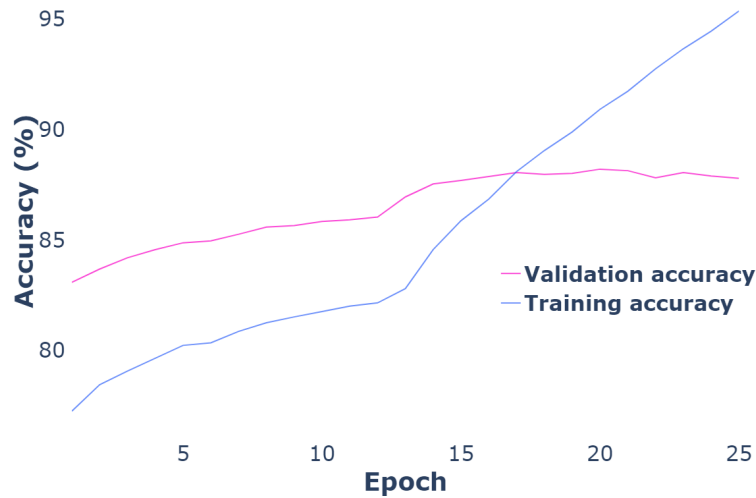


Figure 4.9: Results of the model based on EfficientNet V2L with Methodology 2 (Fine tuning step) - 32 batch size - 224x224 resolution

The third methodology trained did not need as many epochs to converge and after 12, it was already above 90% in training accuracy. Both training and validation accuracy improved parallelly, even though in later epochs, the increase rate of the validation accuracy started to slow down, reaching a top validation accuracy of 88%, along with a training and test accuracy of 90%.

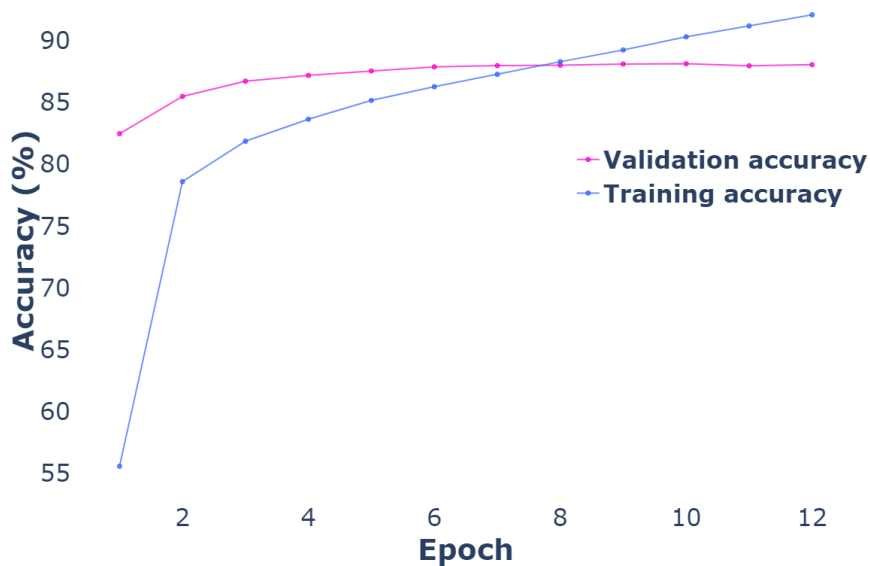


Figure 4.10: Results of the model based on EfficientNet V2L with Methodology 3 - 32 batch size - 224x224 resolution

Moving on to the analysis of the F1-scores in table 4.2, a general improvement is visible when compared to the ResNet results, especially in Methodologies 2 and 3, where there is only one class below 80%, the patio, while many others reached the 90% mark, such as the bathroom, bedroom and closet classes. The problems related to certain classes

4.4. ConvNeXt

with no physical separation between them, shown previously, are still existing but are much less noticeable, with the exception of patio that has still very low values, even in the best methodologies. A further examination of these models is present in Appendix B, regarding the confusion matrices of each methodology.

Table 4.2: F1-score for EfficientNet models

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	82	84	84
bathroom	91	92	92
bedroom	93	93	93
closet	88	91	90
corridor	91	89	89
diningroom	81	82	83
garage	85	86	86
kitchen	82	82	83
livingroom	77	80	81
office	80	87	86
outsideview	82	84	85
patio	75	78	77
staircase	88	89	90
utilityroom	80	82	83
yard	79	83	84
overall	84	86	86

4.4 ConvNeXt

The ConvNeXt architecture was another one of the architectures selected, since it is one of the most recent CNN designs created to try to stop the Transformers supremacy. In specific, the ConvNeXt Large version of PyTorch was preferred because it was the most balanced in terms of performance and memory. A better version was available, the extra large, but a few tests were enough to conclude that memory issues were going to occur. Once again, the images were resized to a 224x224 resolution since the original network utilized this resolution and the batch size was maintained at 32 for some of the methodologies where this value resulted in a good performance, in order to carry out comparisons based mainly on the architecture of the models.

As usual, figures 4.11, 4.12, 4.13, and 4.14 present the evolution of accuracies of training and validation for all the methodologies.

The first one, shown in figure 4.11, resulted in the best model of the ConvNeXts and the previous models shown, obtaining a test accuracy of 87%, one more than the best EfficientNet model, along with a training accuracy of 89% and a validation accuracy of 90%, in epoch 11. Since the last epoch was the best one, this model could be trained for a few extra epochs, but the truth is that the validation accuracy had already been pretty much established and further training would very likely only overfit this model while taking a very long time to complete it.

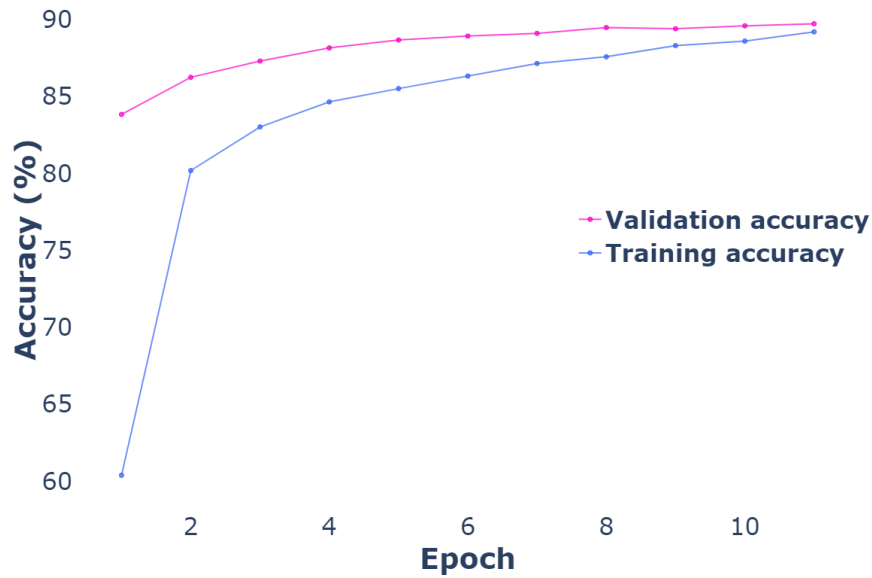


Figure 4.11: Results of the model based on ConvNeXt Large with Methodology 1 - 8 batch size - 224x224 resolution

For the second methodology, present in figures 4.12 and 4.13, the results were also very good, in pair with Methodologies 2 and 3 of the EfficientNet V2L, during the second step of fine tuning, where, in epoch 15 a training and validation accuracy of 89% was obtained and that consequently resulted in a test accuracy of 86%. The feature extraction step did not produce such bad results either ending with a test accuracy of 85%, before going through the fine tuning step to reach the optimal result.

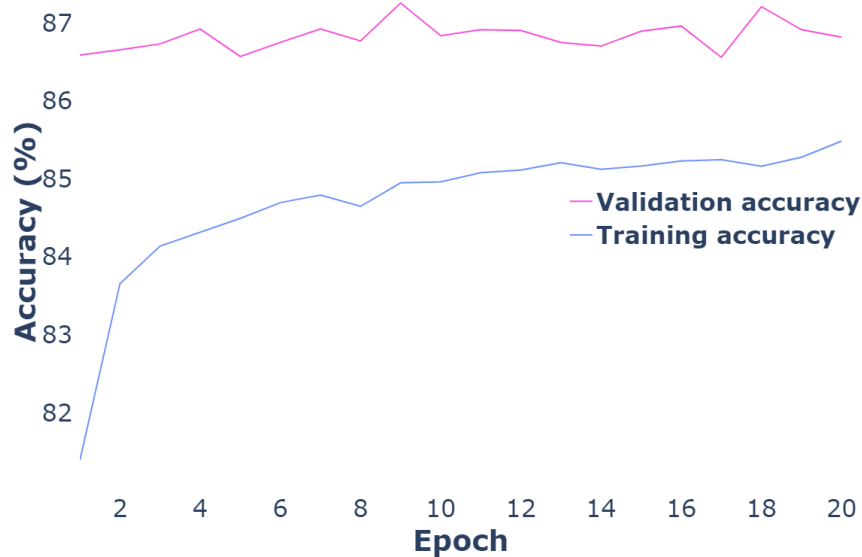


Figure 4.12: Results of the model based on ConvNeXt Large with Methodology 2 (Feature extraction step) - 32 batch size - 224x224 resolution

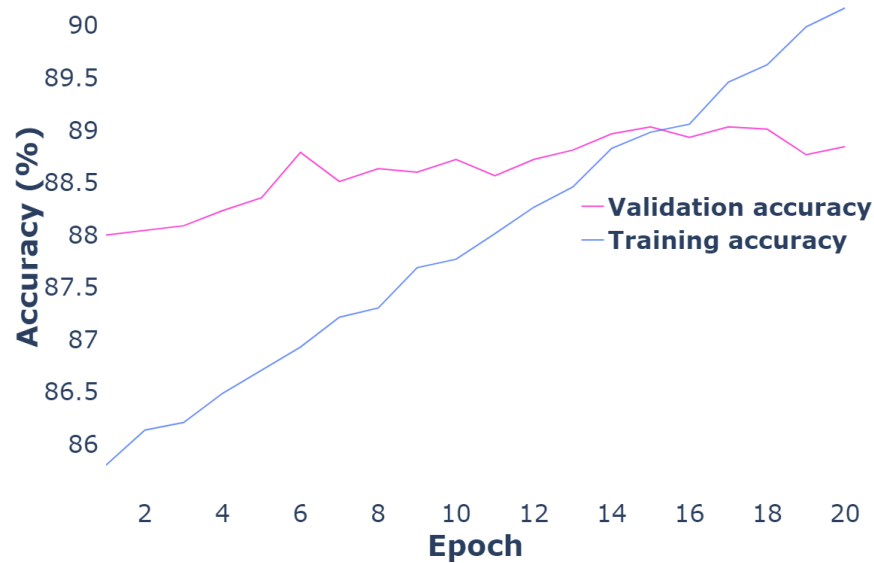


Figure 4.13: Results of the model based on ConvNeXt Large with Methodology 2 (Fine tuning step) - 32 batch size - 224x224 resolution

Finally, the third methodology, present in figure 4.14, had a bit of a strange evolution throughout the epochs in terms of validation accuracy. It first started by increasing at similar rates as the training accuracy until epoch 3, entering then in periods where it would go down and up intermittently. This Methodology reached top values in epoch 7, with a train and validation accuracy of 90% and 89%, respectively, which resulted in a value of 86% for the test accuracy, equal to Methodology 2. It is probably the worst of the three, due to the irregular way in which the best version of it was obtained.

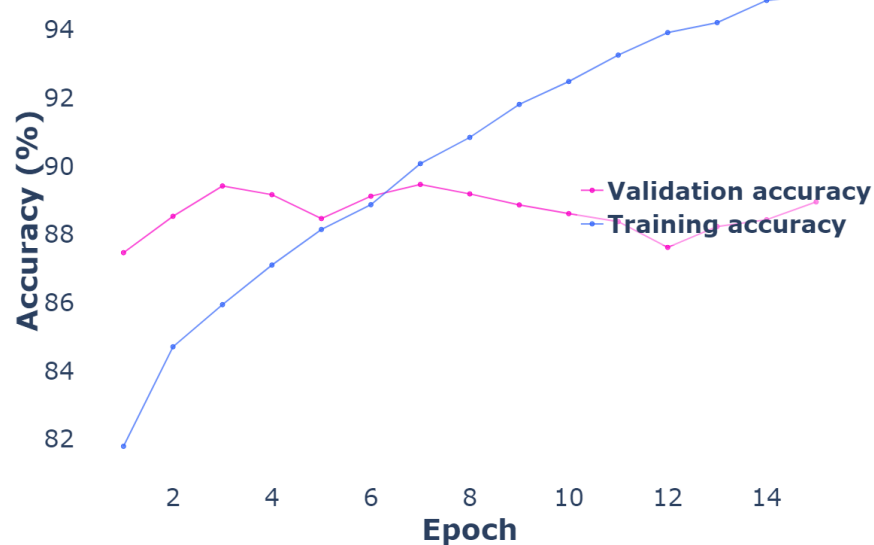


Figure 4.14: Results of the model based on ConvNeXt Large with Methodology 3 - 32 batch size - 224x224 resolution

Finally, an analysis of the F1-scores, present in table 4.3, shows that Methodology 1 is the best model so far, even though the results are all very close when compared to each other or even to the ones of Methodologies 2 and 3 of the EfficientNet V2L. The classes patio and yard, once again, seem to be the main classes where the following models should improve. As usual, further comments and examination related to the confusion matrices of these models are present in Appendix C.

Table 4.3: F1-score for ConvNeXt models

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	86	84	84
bathroom	94	93	92
bedroom	94	92	93
closet	92	91	91
corridor	91	89	90
diningroom	85	84	84
garage	88	86	87
kitchen	84	84	84
livingroom	84	82	82
office	91	89	90
outsideview	85	84	84
patio	79	77	78
staircase	90	88	90
utilityroom	84	83	84
yard	83	81	82
overall	87	86	86

4.5 ViT

The Vision Transformer model was the first Transformer tested for this task of image classification and one of the most basic versions of this architecture was used, the Base Patch 16 from Hugging Face, which was trained on resolutions of 224x224, perfect for comparing with the remaining architectures. This allowed the checking of how one of the simplest Vision Transformers performs against the best CNN architectures. It was also possible to train all the models of the three usual methodologies with a batch size of 32.

Figures 4.15, 4.16 and 4.17 present the evolution of the training of these models. Contrary to what was previously done, there are no results related to the fine tuning of the model in Methodology 2, since after a few tests, the improvement was not significant enough to be included.

So, starting with the first methodology, in figure 4.15, this was one of the best tested, in pair with Methodology 3, obtaining a top train, validation and test accuracy of 94%, 90% and 87%, respectively, from the state of the model in epoch 17. Both the train and validation values evolved well parallelly, even though the validation line, as has been habitual, grows slower and tends to stagnate quicker. Overall, this model already shows that a simple version of Vision Transformers is able to easily rival previously seen Large CNN architectures, obtaining the same general values of accuracy.

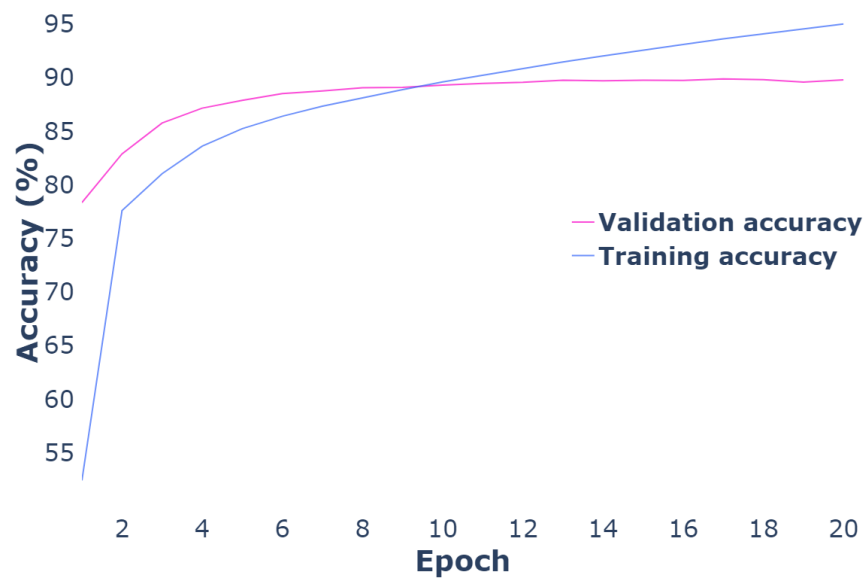


Figure 4.15: Results of the model based on ViT Base Patch 16 with Methodology 1 - 32 batch size - 224x224 resolution

Moving now to figure 4.16, where the second methodology is represented, this was the worst of the versions tested, despite final results having their accuracy lower by only 1 %. Here, the validation accuracy, which throughout the whole training did not vary that much, quickly achieved its best value of 89% in epoch 7, even though the training values increase significantly in the first stages.

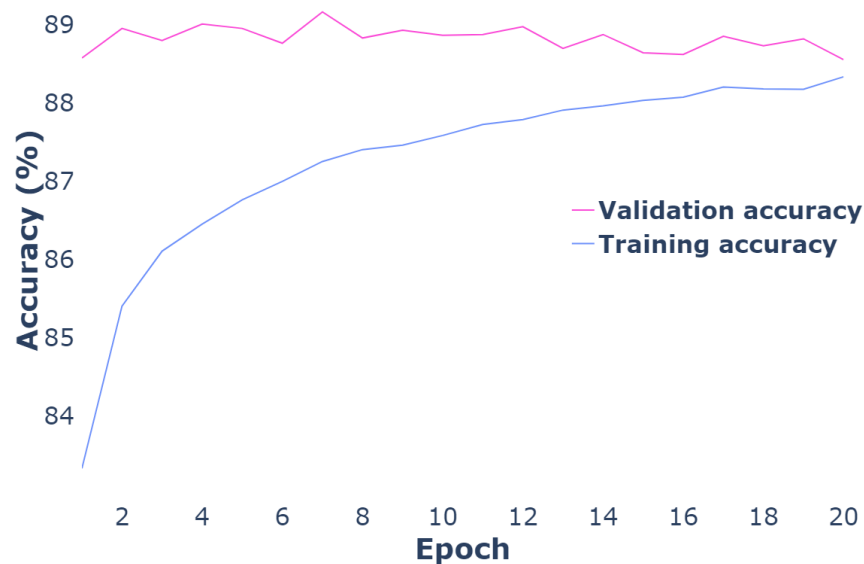


Figure 4.16: Results of the model based on ViT Base Patch 16 with Methodology 2 - 32 batch size - 224x224 resolution

Finally, the last methodology, shown in figure 4.17, is very similar to the first one, so much that it is even hard to notice the difference between them, but this one has worse top training

and validation accuracies of 92% and 89%, respectively.

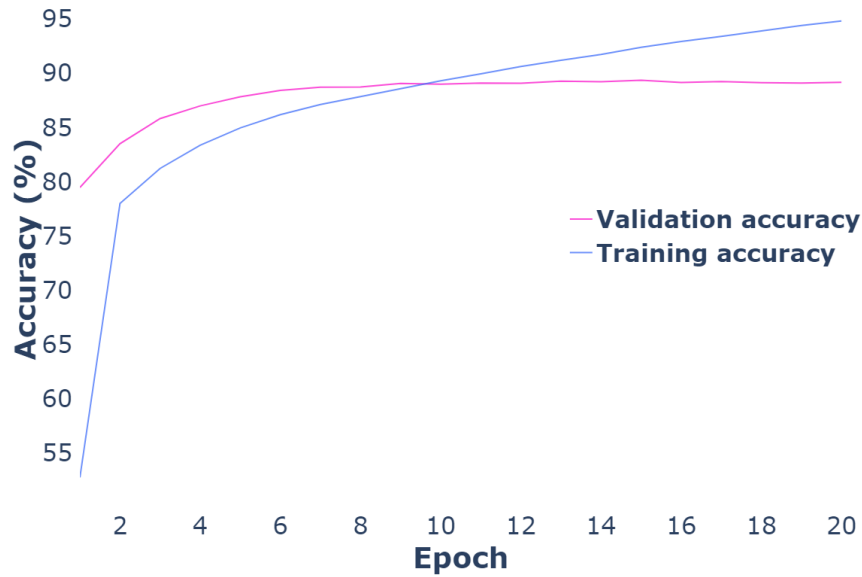


Figure 4.17: Results of the model based on ViT Base Patch 16 with Methodology 3 - 32 batch size - 224x224 resolution

Table 4.4: F1-score for ViT models

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	86	84	86
bathroom	92	93	92
bedroom	94	94	94
closet	91	91	92
corridor	92	89	92
diningroom	83	85	84
garage	89	84	87
kitchen	82	85	83
livingroom	81	82	82
office	88	90	87
outsideview	86	83	85
patio	77	77	78
staircase	91	88	91
utilityroom	83	83	83
yard	84	81	83
overall	87	86	87

At last, by analysing the F1-scores of these models present in table 4.4, these do not change much between each other and when compared to the ConvNeXt large, they always vary between 1% and 2% for all classes, which on one hand, suggests that this simple Vision Transformer version is already competitive enough with the best CNNs and on another

hand, means that a better model is still necessary to obtain better results for classes like patio or yard, for example. The confusion matrices analysis is also available in Appendix D.

4.6 CoCa

The last tested model was the Contrastive Captioner and this was up until recently the best architecture for image classification on the ImageNet dataset. In this case, the large version of this model was used because it was the best and 224x224 resolution images were utilized since, once again, the pre-trained model used this resolution. The common 32 batch size used in most tests conducted was also possible to be used for Methodologies 2 and 3, while the Methodology 1 model could only be allocated with a batch size of 4.

Due to the nature of this architecture, there are two possible ways of classifying images:

- A zero-shot approach where the encoder and decoder of the pre-trained model are both used to compute the cosine similarity between each one of the prompts of each class and the images. This approach resulted in a test accuracy of 77%, which is not bad considering that in this case the model was not even trained on a specific downstream task and was applied directly to the test images.
- A trained way where only the encoder is used to compute features for the images and then, a linear layer for classification is added on top of this. Figures 4.18, 4.19 and 4.20 present the evolution of the three methodologies applied to this approach.

The first methodology 4.18 was without a doubt the one that produced worse results among the three, likely to the fact that unfreezing all the layers right from the beginning made it so the model quickly adapted to the train values, hence leading to the overfitting of the model. For this reason, this Methodology resulted in a top train, validation and test accuracy of 88%, 90% and 87%, respectively, in epoch 2.



Figure 4.18: Results of the model based on CoCa Large with Methodology 1 - 4 batch size - 224x224 resolution

The next two methodologies produced very similar results. In Methodology 2, once again, there was no need for the fine tuning step that was so familiar in the CNNs that were tested, because the model reached its optimal values without even the need of unfreezing the remaining weights of the pre-trained layers. Figures 4.19 and 4.20 show very similar graphics, that end the training with at least 90% accuracy for training, validation and test, the best results obtained so far, a lot superior when compared with CNNs and even the basic Vision Transformers shown before.

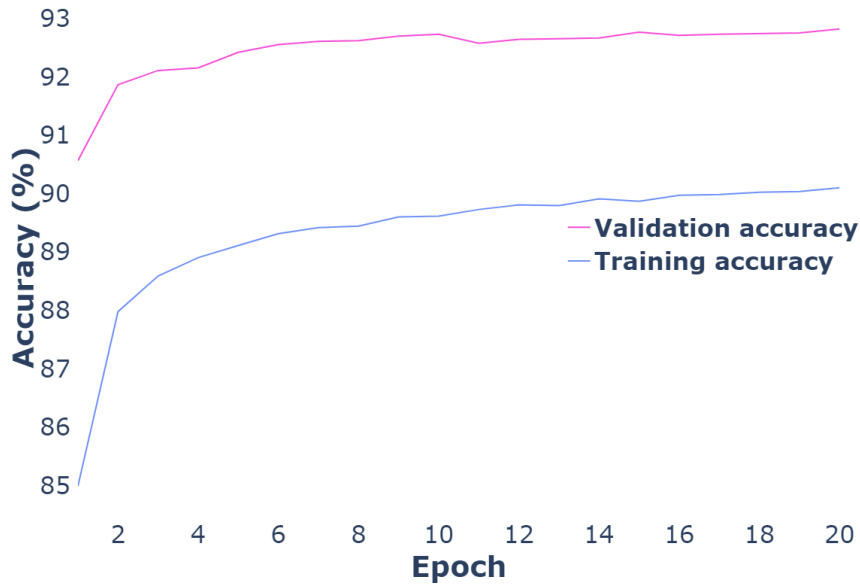


Figure 4.19: Results of the model based on CoCa Large with Methodology 2 - 32 batch size - 224x224 resolution

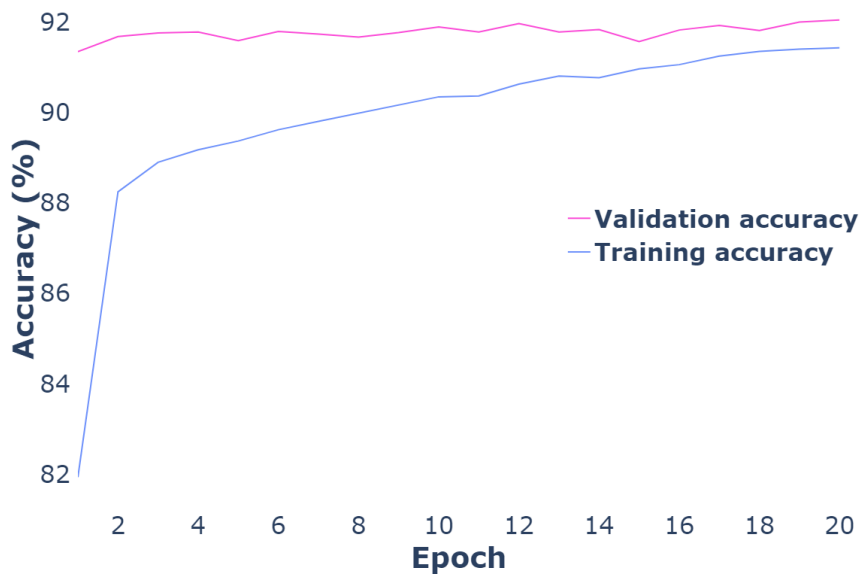


Figure 4.20: Results of the model based on CoCa Large with Methodology 3 - 32 batch size - 224x224 resolution

4.6. CoCa

Ending with the F1-scores of the Contrastive Captioner models, in figure 4.5, these are incredibly good, especially in Methodology 3. Here, with the exception of the patio label, all classes have F1-scores above 86% and 7 out of the 15 present surpass the 90 % threshold, the best obtained so far. These values are so good that the original problematic classes with no physical barriers between them can now be disregarded as such. The remaining analysis of the confusion matrices are also available in Appendix E.

Table 4.5: F1-score for CoCa Large models trained with the research dataset and Adam optimizer

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	86	86	87
bathroom	92	96	96
bedroom	94	95	96
closet	91	93	94
corridor	92	93	94
diningroom	83	88	88
garage	88	88	90
kitchen	85	88	87
livingroom	83	88	87
office	89	94	95
outsideview	87	87	89
patio	79	82	84
staircase	91	93	94
utilityroom	84	87	86
yard	85	86	86
overall	87	90	90

To verify the performance of the trained CoCa models on a more realistic dataset, these were also applied to the private dataset, described in subsection 3.2.1, with a few adaptations for using it as a multi-class problem. First, the previously trained CoCa models were used on top of this multi-labeled dataset, where if the models predicted one of the labels of an image well, they were given full accuracy for the given image. Second, the images where the models did not identify any of the labels were counted as misclassified examples towards the most obvious category present in the image, which was selected during the manual labelling process. Table 4.6 presents the results obtained on the private dataset.

Despite very heterogeneous results, mainly because of the balance of the data, Methodology 2 presents the closest performances to the ones obtained previously on the research dataset, achieving an overall F1-score of 87%, 3% away from the best previous one. This is still a good result and the performance of the model can obviously be increased with continuous training on newly obtained real estate images, especially from misclassified examples. The other two methodologies were not as good, and it is probable that some overfit may have occurred here, because more layers are trainable and the dataset is smaller.

Table 4.6: F1-score for the CoCa model applied to the private real estate dataset

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	68	74	75
bathroom	94	96	97
bedroom	89	92	93
corridor	77	83	84
diningroom	84	91	91
garage	67	85	75
kitchen	86	95	90
livingroom	80	90	87
office	65	78	76
outsideview	85	89	79
patio	59	66	65
staircase	77	74	83
utilityroom	28	51	33
yard	46	48	48
overall	80	87	82

Both the Large and Base versions of this architecture were included in the latest developed API for easily using the best models for inference. With this API, some tests were conducted on images taken from the home of the author to simulate a real scenario and check the performance of the model in another context. Both the CoCa models available were able to predict correctly all the photos taken of the home with good confidence levels. The same did not happen when the same images were tested with the best ConvNeXt model that was used in the initial stages of the creation API for the same effect, where the model got confused between the living room and bedroom due to the existence of a chaise longue sofa. Figure 4.21 shows the image responsible for the occurrence of this problem.



Figure 4.21: Living room with a chaise longue sofa

4.7 Application of the Lion optimizer to CoCa

With the emergence of a new optimization function named Lion, which was used to train the new current best image classification model, an analysis of the implication of this function on the performance of the studied architectures becomes relevant. In this case, the function was applied to the best architecture tested in this document, the CoCa, and the results are presented in table 4.7.

The results are overall very good, especially in Methodology 2 where the F1-scores were very similar to the ones obtained on the best CoCa model trained with the Adam Optimizer. The remaining models of Methodologies 2 and 3 were not that bad either, although not as good as the results obtained for the same methodologies with the Adam optimizer. Overall, the results are in line with what was expected, since even in the benchmark results obtained with the ImageNet dataset, the difference between the use of Adam and Lion optimizers was of only 0.1%, even though, in this case, the tested models were distinct.

Table 4.7: F1-score for CoCa Large models trained with the Lion optimizer

Classes	Methodology 1 (%)	Methodology 2 (%)	Methodology 3 (%)
balcony	85	86	84
bathroom	90	96	94
bedroom	93	96	95
closet	90	93	91
corridor	91	93	90
diningroom	82	89	87
garage	89	87	88
kitchen	83	88	86
livingroom	81	88	84
office	89	93	92
outsideview	87	87	85
patio	78	83	80
staircase	89	94	90
utilityroom	85	87	84
yard	84	85	83
Overall	86	90	88

4.8 Multi-label Classification Experimentation

To try to resolve the problem that comes from using a multi-class classification where an image may have several classes and the model may predict one of them well but not the expected label, multi-label classification tests were conducted on the private dataset of DevScope, described in subsection 3.2.1.

In this case, the training process suffered a few changes from what was done previously, due to the multi-labeled characteristic of this dataset. These are:

- The softmax that was usually applied to the logits obtained in the end of the networks was substituted by a sigmoid to interpret all probabilities of each class in an independent

way, where a threshold of 0.5 was defined to indicate if an image belongs to a class or not. This means that results with probabilities above 50% meant that an image was considered part of a class and below 50%, it was not.

- The replacement of the Cross Entropy Loss function, ideal for multi-class classification problems, to the Binary Cross Entropy Loss function, more indicated for these multi-label problems. Here, since the number of images of classes was highly unbalanced, a weight tensor is passed to transform this function into a Weighted Loss function and overcome the issues brought by the class unbalance in the dataset.

Table 4.8: F1-score for the tested multi-label models

Classes	F1-score ResNet 152v2 (%)	F1-score CoCa Large (%)
bathroom	93	95
bedroom	87	96
garage	57	86
corridor	80	69
staircase	77	80
kitchen	88	96
diningroom	76	89
livingroom	78	90
office	52	48
other	81	89
outsideview	74	83
balcony	67	78
house	88	88
patio	62	68
yard	72	76
utilityroom	40	75
overall	80	86

This methodology was applied to the ResNet 152v2 and CoCa Large, and contrary to previous results, some rare classes of the ResNet model had better results than the CoCa Large, like corridor or office. Table 4.8 presents the F1-scores of both models. Here, one can see that the results are pretty heterogeneous, probably due to the big unbalance of the dataset. Despite this, the results obtained were pretty positive for the small dataset used, with an overall F1-score of 80 % and 86 % for the ResNet 152v2 and CoCa Large, respectively. All of this is a good indication that with a better and bigger dataset, these results could easily be much better reaching acceptable levels for production. Future work should go through this topic, in order to resolve the problems associated with the presence of multiple classes in one image that affects the performance of multi-class classification models.

4.9 Summary

This chapter presented several results of the state of the art image classification neural network models, when presented with an indoor classification dataset of 15 classes. So far,

4.9. Summary

it is clear that in the field of CNNs, the most recent architectures tend to produce better results, like the ConvNeXt, even though the EfficientNet does not stay that far behind. The ResNets, despite being the classic state of the art network, seem to start falling behind when compared to these new models. However, from this study, it is also clear that the new Vision Transformers are now definitely surpassing the performance of the best CNNs, with the most basic ones already performing similarly, and the most recent ones, like CoCa, showing clearly better results. Table 4.9 shows the best variations of each model and their overall result, where it is visible that the Contrastive Captioner Large for Methodology 2 has the best validation, test accuracy, and F1-score, while the ViT B 16 despite having lower performances, achieved a better training accuracy.

Overall, the obtained results are good. However, considering that each image has only one label but sometimes images may include several different indoor rooms, it is expectable that the accuracy values could be better if further studies were performed to understand, for the badly classified images, if the predicted class exists in the image but it is not the one predicted as the label. Another solution to this problem is the usage of multi-label classification by adapting these same models for the effect. A few tests conducted show that they can be effective in tackling it, but bigger datasets should be used to obtain more stable and reliable results.

Table 4.9: F1-score for the best models

Best model variation	Train accuracy (%)	Validation accuracy (%)	Test accuracy (%)	F1-score (%)
ResNet 152v2 (Methodology 3)	85	86	82	82
EfficientNet V2L (Methodology 2)	91	88	86	86
ConvNeXt Large (Methodology 3)	89	90	87	87
ViT B 16 (Methodology 1)	94	90	87	87
CoCa Large (Methodology 2)	90	93	90	90

Chapter 5

Conclusion

This chapter presents the main conclusions of the developed work, presenting a summary of the achieved results and of how these enable meeting the defined objectives, an overview of ethical issues related to this work, and the identification of limitations and promising paths for future research.

5.1 Summary of Results

The Image Classification problem has been a field with a lot of research performed throughout the years. From the initial CNNs to the current Vision Transformers, there have been many architectures created and tested, and contests like the ImageNet Large Scale Visual Recognition Challenge is one of many that incited researchers to find new solutions and reach the levels that exist today. This work tried to make an extensive analysis of several different state of the art architectures that have arisen over the years and how these performed for a created dataset for the objective of classifying indoor rooms and helping real estate agents in the task of filtering through a big volume of images.

With this purpose, several comparisons have been made between versions of ResNet, EfficientNet, ConvNeXt, Vision Transformers and Contrastive Captioners and it is now clear that Vision Transformers, in particular the Contrastive Captioners, are indeed the state of the art in image classification, at least for the problem of indoor classification. If a few years ago, there was no doubt about CNNs and ResNets being the ideal architectures for image classification, recently, it has been seen that Vision Transformers have taken that spot and are only getting better. Even the most recent CNN architectures, such as ConvNeXt are only able to hold performances similar to the more initial Vision Transformers, while the more recent design, the Contrastive Captioner presents much better results.

This dissertation also tried to fill a gap by introducing results of the most recent image classification neural network architectures applied to the indoor classification problem, since the papers explored in section 2.4 tended to utilize older architectures, like Inception and ResNet, and there were no works showing Vision Transformers applied to this field.

5.2 Objectives Overview

All the objectives presented at the beginning of this document have been fully or partially tackled throughout the document, as follows.

O1 (Study the state of the art of the image classification problem and how it has been evolving throughout the years) has been addressed in chapter 2, where the main state of

the art architectures for image classification have been explored and their innovative aspects have been presented.

O2 (Analyse different datasets and how some properties changes in them influence the results) has been discussed in subsection 3.2.1 where the different datasets used are explored and presented and in chapter 4 where some properties and classes of the images used are analysed to help explain the obtained results.

O3 (Study how the different neural network architectures influence the performance of the models in the indoor classification task) is resolved in chapter 4 where several models based on state of the art architectures are trained using the created dataset and the results and their performance is studied.

O4 (Perform a study of how the best model performs on real photos taken from other contexts outside online sources, like photos taken of the author's home) is analysed at the end of section 4.6, where the overall classification results obtained with these images are explained, and in section 4.8, where a small case study conducted with real world images from real estate is presented.

O5 (Investigate the best deployment method for the best found classification model) is tackled in section 3.4 where the framework utilized is described, along with steps taken to create the API and the available requests.

Finally, O6 (Analyse the achieved results critically and draw the most significant conclusions) is tackled in the end of chapter 4 and 5 where the final conclusions related to the best models for the problem in hand are presented.

5.3 Ethical Issues

These days, AI exists in so many technologies and digital solutions, that concerns related to the ethical issues it raises start to appear. In AI, especially machine learning, the main ethical problems come from the datasets and the explainability of the model. On one hand, datasets may deal with sensitive data from users and must have their consent, or these may be imbalanced and generate a model that discriminates some groups of society. On the other hand, the lack of explainability of these models may cause problems to users when the decisions taken affect them but there is no explanation provided [136].

Since this project only utilizes images available for research and tries to group them by indoor classes, there is no significant ethical issues to address, since even if an image is badly classified, the only issue here is that the user experience will suffer a bit.

5.4 Limitations and Future Work

Despite all the objectives of this thesis being achieved, there are still some key aspects that should be taken into consideration while interpreting the results and that should be improved for future work. These are:

- The private dataset used in sections 4.6 and 4.8 has a lot fewer images than the research dataset and it is extremely unbalanced, compromising a lot of the performance result of some of the categories with fewer images. The fact that this is a private dataset also means that further details could not be revealed.

- The research dataset was composed of many single labeled images, which means that the experimentation conducted in section 4.8 could not be performed because every image had only one label.
- Some of the models, especially the ones trained on Methodology 1, had to be trained on smaller batch sizes due to the hardware used, which resulted in slightly worse performances in some cases.

For these reasons and to improve upon this work, which already shows good performances, the following topics are appointed as future work:

- Automatization of the process of selecting images that characterize a home for selling, so that, after the filtering of the taken photos in categories, the best ones from each class can be selected automatically.
- Elaboration of tests with bigger balanced real estate datasets to improve on the models created.
- Expansion of the multi-label experimentation, since it has shown promising results and it is capable of resolving the issues where the images have more than just one category represented in them.
- The study of how hibrid architectures of Convolutional Neural Networks and Vision Transformers perform in this domain.

5.5 Final Remarks

This thesis demonstrated that the Image Classification problem has been evolving at an exciting rate in recent years and every year new and improved technologies rise to the point of reaching human like levels of performance. The work performed provides a detailed analysis of how this evolution has been conducted throughout the years and how the state of the art models perform in datasets based on indoor classification, filling a gap in the research community related to the performance of more recent models to this problem, in particular, Vision Transformers.

Bibliography

- [1] E. Glickman, *An introduction to real estate finance*. Academic Press, 2013.
- [2] P. M. E. Garboden, "Review of Upsold: Real Estate Agents, Prices, and Neighborhood Inequality," *Social Forces*, vol. 100, no. 4, e13–e13, Dec. 2021, issn: 0037-7732. doi: 10.1093/sf/soab160. eprint: <https://academic.oup.com/sf/article-pdf/100/4/e13/43510338/soab160.pdf>. [Online]. Available: <https://doi.org/10.1093/sf/soab160>.
- [3] J. H. Bappy, J. R. Barr, N. Srinivasan, and A. K. Roy-Chowdhury, "Real estate image classification," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 373–381. doi: 10.1109/WACV.2017.48.
- [4] A. Nouriani and L. Lemke, "Vision-based housing price estimation using interior, exterior & satellite images," *Intelligent Systems with Applications*, vol. 14, p. 200 081, 2022, issn: 2667-3053. doi: <https://doi.org/10.1016/j.iswa.2022.200081>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2667305322000217>.
- [5] B. Veiga, G. Santos, T. Pinto, R. Faia, C. Ramos, and Z. Vale, "Simulation tools for electricity markets considering power flow analysis," *Energy*, vol. 275, p. 127 494, Apr. 2023. doi: 10.1016/j.energy.2023.127494.
- [6] H. Ko and C. Ramos, "A survey of context classification for intelligent systems research for ambient intelligence," in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 2010, pp. 746–751. doi: 10.1109/CISIS.2010.68.
- [7] C. Ramos, J. C. Augusto, and D. Shapiro, "Ambient intelligence—the next step for artificial intelligence," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 15–18, 2008. doi: 10.1109/MIS.2008.19.
- [8] B. Mota, T. Pinto, Z. Vale, and C. Ramos, "Deep learning in intelligent power and energy systems," in *Intelligent Data Mining and Analysis in Power and Energy Systems: Models and Applications for Smarter Efficient Power Systems*. 2023, pp. 45–67. doi: 10.1002/9781119834052.ch3.
- [9] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309–320, 2019, issn: 0167-9260. doi: <https://doi.org/10.1016/j.vlsi.2019.07.005>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167926019301762>.
- [10] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: a survey," *Optical Engineering*, vol. 58, no. 4, p. 040 901, 2019. doi: 10.1117/1.OE.58.4.040901. [Online]. Available: <https://doi.org/10.1117/1.OE.58.4.040901>.
- [11] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, 2022. doi: 10.1109/TPAMI.2021.3059968.

- [12] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020, issn: 0924-2716. doi: <https://doi.org/10.1016/j.isprsjprs.2019.11.023>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0924271619302825>.
- [13] P. Espinace, T. Kollar, A. Soto, and N. Roy, "Indoor scene recognition through object detection," in *2010 IEEE International conference on robotics and automation*, IEEE, 2010, pp. 1406–1413.
- [14] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "Sun database: Exploring a large collection of scene categories," *International Journal of Computer Vision*, vol. 119, no. 1, pp. 3–22, 2016.
- [15] *Image classification on imagenet*, <https://paperswithcode.com/sota/image-classification-on-imagenet>, Accessed: 2023-04-14, 2023.
- [16] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.
- [17] M. Mokhairi, S. A. Engku Fadzli Hasan, and K. Nur Shazwani, "Comparison of image classification techniques using caltech 101 dataset," *journal of theoretical and applied information technology*, vol. 71, no. 1, pp. 79–86, 2015.
- [18] E. Sullivan, "FACIAL RECOGNITION TECHNOLOGY," 2021. [Online]. Available: <https://leg.mt.gov/content/Committees/Interim/2021-2022/Economic%5C%20Affairs/Meetings/September%5C%202021/facial-recognition-technology.pdf>.
- [19] S. M. Smith and J. M. Brady, "Susan—a new approach to low level image processing," *International journal of computer vision*, vol. 23, no. 1, pp. 45–78, 1997.
- [20] M. Szummer and R. W. Picard, "Indoor-outdoor image classification," in *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, IEEE, 1998, pp. 42–51.
- [21] A. VAILAYA, A. JAIN, and H. J. ZHANG, "On image classification: City images vs. landscapes," *Pattern Recognition*, vol. 31, no. 12, pp. 1921–1935, 1998, issn: 0031-3203. doi: [https://doi.org/10.1016/S0031-3203\(98\)00079-X](https://doi.org/10.1016/S0031-3203(98)00079-X). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S003132039800079X>.
- [22] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8. doi: 10.1109/CVPR.2008.4587598.
- [23] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4409066.
- [24] J. Li, S. Huang, R. He, and K. Qian, "Image classification based on fuzzy support vector machine," in *2008 International Symposium on Computational Intelligence and Design*, vol. 1, 2008, pp. 68–71. doi: 10.1109/ISCID.2008.51.
- [25] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Biomedical image classification with random subwindows and decision trees," in *International Workshop on Computer Vision for Biomedical Image Applications*, Springer, 2005, pp. 220–229.
- [26] *Caltech-101*, <https://paperswithcode.com/dataset/caltech-101>, Accessed: 2022-12-21, 2022.
- [27] 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*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

- [28] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [29] S. Dikaleh, D. Xiao, C. Felix, D. Mistry, and M. Andrea, "Introduction to neural networks," in *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '17, Markham, Ontario, Canada: IBM Corp., 2017, p. 299.
- [30] A. Pinto, T. Pinto, I. Praça, and Z. Vale, "Classification of local energy trading negotiation profiles using artificial neural networks," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*, 2019, pp. 1–5. doi: 10.1109/PESGM40551.2019.8973600.
- [31] P. Shukla and R. Iriondo, *Neural networks from scratch with python code and math in detail*, <https://pub.towardsai.net/building-neural-networks-from-scratch-with-python-code-and-math-in-detail-i-536fae5d7bbf>, Accessed: 2022-11-22, 2020.
- [32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [33] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. arXiv: 1609.04747. [Online]. Available: <http://arxiv.org/abs/1609.04747>.
- [34] H. B. McMahan, G. Holt, D. Sculley, et al., "Ad click prediction: A view from the trenches," ser. KDD '13, Chicago, Illinois, USA: Association for Computing Machinery, 2013, pp. 1222–1230, isbn: 9781450321747. doi: 10.1145/2487575.2488200. [Online]. Available: <https://doi.org/10.1145/2487575.2488200>.
- [35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [36] X. Chen, C. Liang, D. Huang, et al., *Symbolic discovery of optimization algorithms*, 2023. arXiv: 2302.06675 [cs.LG].
- [37] H. Pham, Z. Dai, G. Ghiasi, et al., "Combined scaling for zero-shot transfer learning," *arXiv preprint arXiv:2111.10050*, 2021.
- [38] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *towards data science*, vol. 6, no. 12, pp. 310–316, 2017.
- [39] A. Nguyen, K. Pham, D. Ngo, T. Ngo, and L. Pham, "An analysis of state-of-the-art activation functions for supervised deep neural network," in *2021 International Conference on System Science and Engineering (ICSSE)*, 2021, pp. 215–220. doi: 10.1109/ICSSE52999.2021.9538437.
- [40] R. Sucky, *An overview of performance evaluation metrics of machine learning (classification) algorithms in python*, <https://towardsdatascience.com/an-overview-of-performance-evaluation-metrics-of-machine-learning-classification-algorithms-7a95783a762f>, Accessed: 2022-12-02, 2021.
- [41] C. Jung, *Popular machine learning performance metrics*, <https://towardsdatascience.com/popular-machine-learning-performance-metrics-a2c33408f29>, Accessed: 2022-12-02, 2021.
- [42] J. Junior, T. Pinto, and H. Morais, "Hybrid classification-regression metric for the prediction of constraint violations in distribution networks," *Electric Power Systems Research*, vol. 221, p. 109401, 2023, issn: 0378-7796. doi: <https://doi.org/10.1016/j.epsr.2023.109401>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0378779623002900>.

-
- [43] P. Shukla and R. Iriondo, *Main types of neural networks and their applications — tutorial*, <https://pub.towardsai.net/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e>, Accessed: 2022-12-02, 2020.
 - [44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
 - [45] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
 - [46] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
 - [47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
 - [48] P. A. Gagniuć, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.
 - [49] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.
 - [50] G. E. Hinton, T. J. Sejnowski, *et al.*, “Learning and relearning in boltzmann machines,” *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, no. 282-317, p. 2, 1986.
 - [51] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
 - [52] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
 - [53] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: <http://arxiv.org/abs/1706.03762>.
 - [54] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.
 - [55] D. Bhatt, C. Patel, H. Talsania, *et al.*, “Cnn variants for computer vision: History, architecture, application, challenges and future scope,” *Electronics*, vol. 10, no. 20, p. 2470, 2021.
 - [56] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of cnn and rnn for natural language processing,” *arXiv preprint arXiv:1702.01923*, 2017.
 - [57] D. Palaz, R. Collobert, *et al.*, “Analysis of cnn-based speech recognition system using raw speech as input,” *Idiap*, Tech. Rep., 2015.
 - [58] S. Pouyanfar, S. Sadiq, Y. Yan, *et al.*, “A survey on deep learning: Algorithms, techniques, and applications,” *ACM Comput. Surv.*, vol. 51, no. 5, Sep. 2018, issn: 0360-0300. doi: 10.1145/3234150. [Online]. Available: <https://doi.org/10.1145/3234150>.
 - [59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

- [60] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," 2009.
- [61] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE, vol. 2, 2004, pp. II–104.
- [62] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.
- [63] C. Szegedy, W. Liu, Y. Jia, et al., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [64] *Pytorch - model and pre-trained weights*, <https://pytorch.org/vision/0.12/models.html>, Accessed: 2022-11-22.
- [65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: <http://arxiv.org/abs/1512.03385>.
- [66] H. Zhang, J. Mo, H. Jiang, et al., "Deep learning model for the automated detection and histopathological prediction of meningioma," *Neuroinformatics*, vol. 19, Jul. 2021. doi: 10.1007/s12021-020-09492-6.
- [67] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016. arXiv: 1611.05431. [Online]. Available: <http://arxiv.org/abs/1611.05431>.
- [68] jie Shao, X. Zhang, Z. Ding, et al., *Trimps-soushen@ilsvrc2016*, <https://www.youtube.com/watch?v=NaoV00hVC3w>, Accessed: 2022-11-23, 2016.
- [69] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *CoRR*, vol. abs/1709.01507, 2017. arXiv: 1709.01507. [Online]. Available: <http://arxiv.org/abs/1709.01507>.
- [70] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- [71] C. Liu, B. Zoph, J. Shlens, et al., "Progressive neural architecture search," *CoRR*, vol. abs/1712.00559, 2017. arXiv: 1712.00559. [Online]. Available: <http://arxiv.org/abs/1712.00559>.
- [72] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *CoRR*, vol. abs/1802.01548, 2018. arXiv: 1802.01548. [Online]. Available: <http://arxiv.org/abs/1802.01548>.
- [73] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. arXiv: 1905.11946. [Online]. Available: <http://arxiv.org/abs/1905.11946>.
- [74] H. Pham, Q. Xie, Z. Dai, and Q. V. Le, "Meta pseudo labels," *CoRR*, vol. abs/2003.10580, 2020. arXiv: 2003.10580. [Online]. Available: <https://arxiv.org/abs/2003.10580>.
- [75] C. Jia, Y. Yang, Y. Xia, et al., "Scaling up visual and vision-language representation learning with noisy text supervision," *CoRR*, vol. abs/2102.05918, 2021. arXiv: 2102.05918. [Online]. Available: <https://arxiv.org/abs/2102.05918>.
- [76] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," *CoRR*, vol. abs/2010.01412, 2020. arXiv: 2010.01412. [Online]. Available: <https://arxiv.org/abs/2010.01412>.

- [77] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," *CoRR*, vol. abs/2104.00298, 2021. arXiv: 2104.00298. [Online]. Available: <https://arxiv.org/abs/2104.00298>.
- [78] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. arXiv: 2010.11929. [Online]. Available: <https://arxiv.org/abs/2010.11929>.
- [79] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "Coatnet: Marrying convolution and attention for all data sizes," *CoRR*, vol. abs/2106.04803, 2021. arXiv: 2106.04803. [Online]. Available: <https://arxiv.org/abs/2106.04803>.
- [80] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *CoRR*, vol. abs/2201.03545, 2022. arXiv: 2201.03545. [Online]. Available: <https://arxiv.org/abs/2201.03545>.
- [81] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021. arXiv: 2103.14030. [Online]. Available: <https://arxiv.org/abs/2103.14030>.
- [82] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.
- [83] M. J. Page, J. E. McKenzie, P. M. Bossuyt, *et al.*, "The prisma 2020 statement: An updated guideline for reporting systematic reviews," *Systematic reviews*, vol. 10, no. 1, pp. 1–11, 2021.
- [84] *Acml digital library*, <https://dl.acm.org/>, Accessed: 2022-12-02, 2022.
- [85] *ieee xplore*, <https://ieeexplore.ieee.org/Xplore/home.jsp>, Accessed: 2022-12-02, 2022.
- [86] *Ebsco information service*, <https://www.ebsco.com/pt>, Accessed: 2022-12-02, 2022.
- [87] *Google scholar*, <https://scholar.google.com/>, Accessed: 2022-11-22, 2022.
- [88] *Medium*, <https://medium.com/>, Accessed: 2022-11-22, 2022.
- [89] E. Tasli, *Automated image tagging at booking.com*, <https://booking.ai/automated-image-tagging-at-booking-com-7704f27dcc8b>, Accessed: 2022-11-25, 2017.
- [90] P. Chhikara, A. Goyal, and C. Sharma, "Re-tagger: A light-weight real-estate image classifier," *arXiv preprint arXiv:2207.05696*, 2022.
- [91] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 413–420. doi: 10.1109/CVPR.2009.5206537.
- [92] G. Nascimento, C. Laranjeira, V. Braz, A. Lacerda, and E. R. Nascimento, "A robust indoor scene recognition method based on sparse representation," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, M. Mendoza and S. Velastin, Eds., Cham: Springer International Publishing, 2018, pp. 408–415.
- [93] V. H. Vo, N. N. Khoa, and P. H. Nhat, "Real estate image classification for e-commerce website," *Science and Technology Development Journal*, vol. 25, no. 1, pp. 2224–2238, 2022.
- [94] W. Wang, S. Dong, K. Zou, and W.-s. Li, "Room classification in floor plan recognition," in *2020 4th International Conference on Advances in Image Processing*, ser. ICAIP 2020, Chengdu, China: Association for Computing Machinery, 2021, pp. 48–54, isbn: 9781450388368. doi: 10.1145/3441250.3441265. [Online]. Available: <https://doi.org/10.1145/3441250.3441265>.

- [95] D. Dziwis, S. Zimmermann, T. Lübeck, J. M. Arend, D. Bau, and C. Pörschmann, "Machine learning-based room classification for selecting binaural room impulse responses in augmented reality applications," in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, 2021, pp. 1–8. doi: 10.1109/I3DA48870.2021.9610915.
- [96] M. Hayat, S. H. Khan, M. Bennamoun, and S. An, "A spatial layout and scale invariant feature representation for indoor scene classification," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4829–4841, 2016. doi: 10.1109/TIP.2016.2599292.
- [97] L. Lu, K. Toyama, and G. Hager, "A two level approach for scene recognition," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 688–695 vol. 1. doi: 10.1109/CVPR.2005.51.
- [98] Z. Wang, R. Sacks, and T. Yeung, "Exploring graph neural networks for semantic enrichment: Room type classification," *Automation in Construction*, vol. 134, p. 104 039, 2022.
- [99] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 2010, pp. 3485–3492.
- [100] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [101] *Booking.com*, <https://www.booking.com/>, Accessed: 2022-11-25, 2022.
- [102] D. Koch, M. Despotovic, S. Leiber, M. Sakeena, M. Doeller, and M. Zeppelzauer, "Real estate image analysis: A literature review," *Journal of Real Estate Literature*, vol. 27, pp. 269–300, Dec. 2019. doi: 10.22300/0927-7544.27.2.269.
- [103] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1, pp. 157–173, 2008.
- [104] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.
- [105] N. Ali and B. Zafar, "15-Scene Image Dataset," Aug. 2018. doi: 10.6084/m9.figshare.7007177.v1. [Online]. Available: https://figshare.com/articles/dataset/15-Scene_Image_Dataset/7007177.
- [106] B. Neo, *Top machine learning frameworks used by data scientists*, <https://medium.com/bitgrit-data-science-publication/top-machine-learning-frameworks-used-by-data-scientists-47ab9a273c95>, Accessed: 2022-11-28, 2021.
- [107] *State of machine learning and data science 2021*, <https://storage.googleapis.com/kaggle-media/surveys/Kaggle's%20State%20of%20Machine%20Learning%20and%20Data%20Science%202021.pdf>, Accessed: 2022-11-28, 2021.
- [108] R. Desai, *State of machine learning and data science 2021*, <https://towardsdatascience.com/top-8-ai-and-machine-learning-frameworks-for-beginners-4e3996fc27f>, Accessed: 2022-11-28, 2021.
- [109] O. .-. O. D. Science, *Top 7 machine learning frameworks for 2020*, <https://odsc.medium.com/top-7-machine-learning-frameworks-for-2020-7e45164914e1>, Accessed: 2022-11-28, 2020.
- [110] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

- [111] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: <https://www.tensorflow.org/>.
- [112] *Cuda toolkit*, <https://developer.nvidia.com/cuda-toolkit>, 2022.
- [113] F. Chollet *et al.*, *Keras*, <https://keras.io>, 2015.
- [114] A. Paszke, S. Gross, F. Massa, *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [115] T. Wolf, L. Debut, V. Sanh, *et al.*, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- [116] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: <http://github.com/google/jax>.
- [117] J. Howard *et al.*, *Fastai*, <https://github.com/fastai/fastai>, 2018.
- [118] Y. Jia, E. Shelhamer, J. Donahue, *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [119] <https://learn.microsoft.com/en-us/cognitive-toolkit/>, <https://learn.microsoft.com/en-us/cognitive-toolkit/>, Accessed: 2022-11-28, 2022.
- [120] *Tensorflow vs keras vs pytorch*, <https://www.kaggle.com/general/177187>, Accessed: 2022-11-28, 2020.
- [121] *Keras vs tensorflow vs pytorch: Key differences among the deep learning framework*, <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>, Accessed: 2022-11-28, 2022.
- [122] *House rooms & streets image dataset*, <https://www.kaggle.com/datasets/mikhailma/house-rooms-streets-image-dataset>, Accessed: 2022-12-12, 2022.
- [123] E. Ahmed and M. Moustafa, “House price estimation from visual and textual features,” *arXiv preprint arXiv:1609.08399*, 2016.
- [124] *Czkawka*, <https://github.com/qarmin/czkawka>, Accessed: 2022-01-11, 2022.
- [125] P. Voigt and A. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Jan. 2017, isbn: 978-3-319-57958-0. doi: 10.1007/978-3-319-57959-7.
- [126] T. T. Ke and K. Sudhir, “Privacy rights and data security: Gdpr and personal data driven markets,” *Available at SSRN 3643979*, 2020.
- [127] *Fastapi*, <https://fastapi.tiangolo.com/>, Accessed: 2023-04-10, 2023.
- [128] *Django*, <https://www.djangoproject.com/>, Accessed: 2023-04-10, 2023.
- [129] *Flask*, <https://flask.palletsprojects.com/en/2.2.x/>, Accessed: 2023-04-10, 2023.
- [130] *Onnx*, <https://flask.palletsprojects.com/en/2.2.x/>, Accessed: 2023-04-10, 2023.
- [131] *Onnx runtime*, <https://onnxruntime.ai/index.html#getStartedTable>, Accessed: 2023-04-10, 2023.
- [132] C. Griffiths, *The latest 2022 cyber crime statistics (updated december 2022)*, <https://aag-it.com/the-latest-2022-cyber-crime-statistics/#:~:text=The%20latest%202022%20cyber%20crime%20statistics%20are%20as%20follows%3A>

- 20UK%20has%20the%20highest , a%2013%25%20decrease%20over%202020. , Accessed: 2022-12-09, 2022.
- [133] S. Morgan, *Cybercrime to cost the world \$10.5 trillion annually by 2025*, <https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/>, Accessed: 2022-12-09, 2020.
- [134] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine learning security: Threats, countermeasures, and evaluations," *IEEE Access*, vol. 8, pp. 74 720–74 742, 2020. doi: 10.1109/ACCESS.2020.2987435.
- [135] *Keras applications*, <https://keras.io/api/applications/>, Accessed: 2022-01-13, 2022.
- [136] M. Coeckelbergh, "Artificial intelligence: Some ethical issues and regulatory challenges," *Technology and Regulation*, vol. 2019, pp. 31–34, May 2019. doi: 10.26116/techreg.2019.003. [Online]. Available: <https://techreg.org/article/view/10999>.

Appendix A

Confusion Matrix Analysis of ResNet Models

In this first appendix, an analysis how each ResNet model performs for each class is provided. Here it is clear that, even though some indoor classes are easy to classify, like the bedroom, others may create some confusion due the type of objects present in them being also present in other divisions and also, because some photos contain partially parts of other rooms that are adjacent to the one that serve as label. Figures A.1,A.2 and A.3 show the confusion matrices of these models in a all vs all fashion.

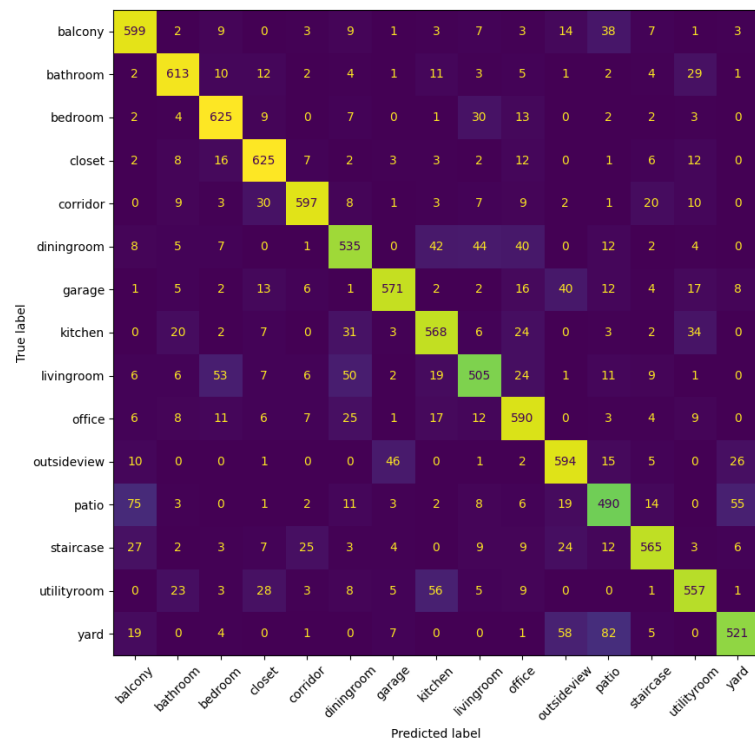


Figure A.1: Confusion matrix of the model based on ResNet 152v2 with Methodology 1 - 24 batch size - 224x224 resolution

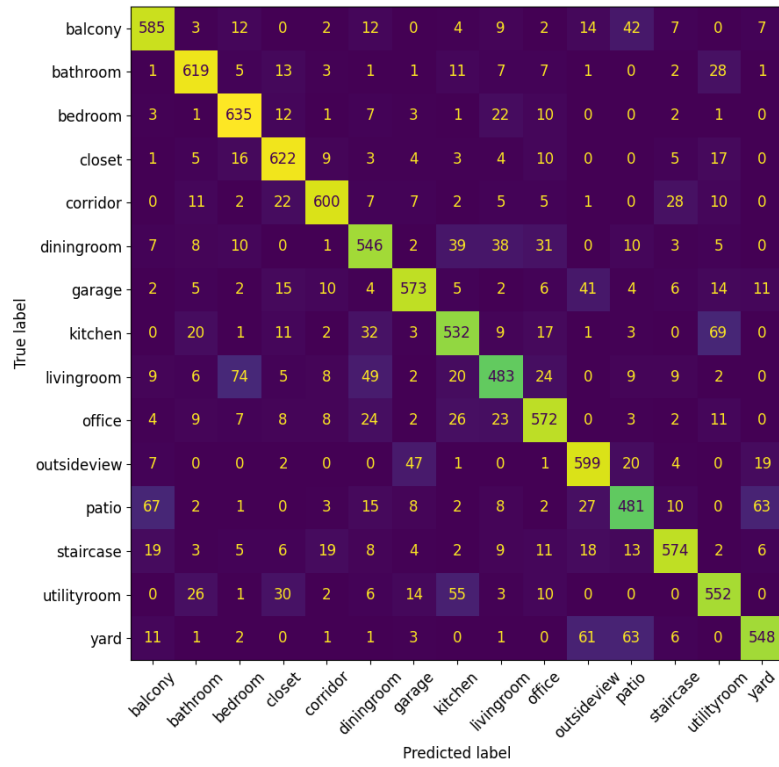


Figure A.2: Confusion matrix of the model based on ResNet 152v2 with Methodology 2 - 24 batch size - 224x224 resolution

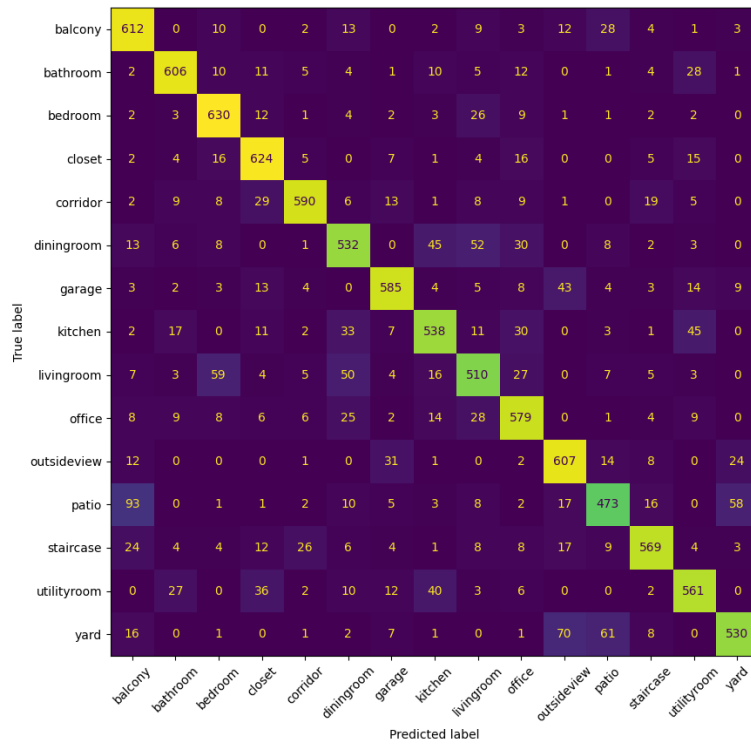


Figure A.3: Confusion matrix of the model based on ResNet 152v2 with Methodology 3 - 64 batch size - 224x224 resolution

Looking at them, it is visible that the easiest classes to classify are the balcony, bathroom, bedroom, closet and outside view with more than 600 correct predictions. Others, like dining room, kitchen, living room, patio and yard seem to be harder to get right and when the results are observed more closely, it is noticeable that the model confuses the first three classes quite a bit. This may be due to the fact these divisions usually coexist without a wall between them and of course, in some houses, dining rooms don't even exist and the dining tables tend to be placed in the kitchens and living rooms. In the case of the patio and the yard, the confusion may come from both being zones outside of the house usually close to each other, which increases the chances of photos having both, and of course, the outside view images also tend to capture a bit of these spaces.

Appendix B

Confusion Matrix Analysis of EfficientNet Models

In this appendix, the analysis of the confusion matrices of the EfficientNet models are presented. Starting with the analysis of the confusion matrix of the first methodology, shown in figure B.1, the problems shown before are still present and perhaps even more accentuated, despite the slight increase in accuracy. This happens because the model was able to better learn the classes that already had good results, but misclassify even more images of the classes with problems like living room, patio and yard. Looking at the confusion matrices of the second and third methodology, in figures B.2 and B.3, it is noticeable that despite all the problems presented before related to the dining room, living room, kitchen, patio, yard and outside view still being present, it seems that these have been greatly improved with the only significant downgrades being the number of true positives of the balcony class, which decreased a bit.

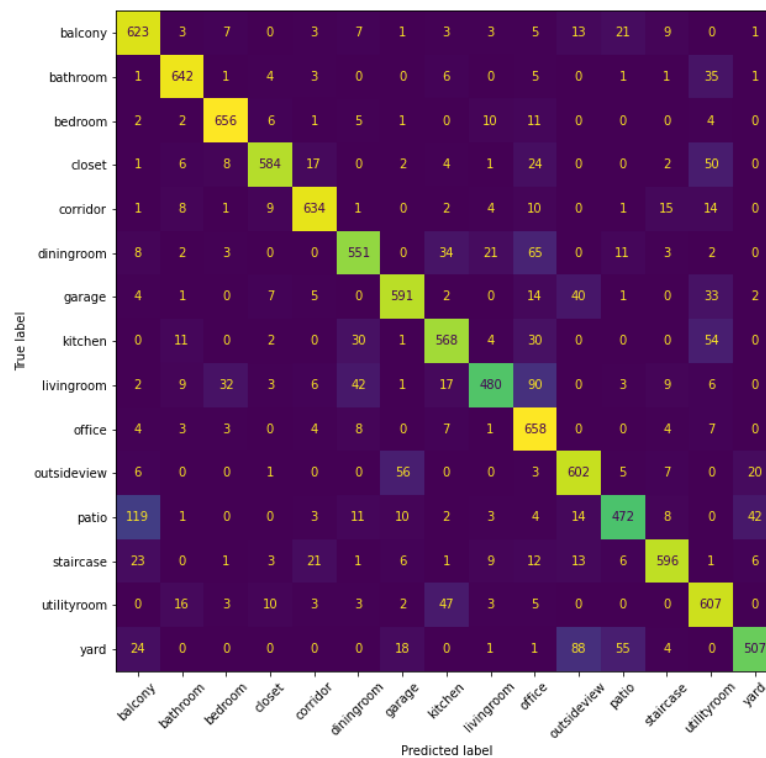


Figure B.1: Confusion matrix of the model based on EfficientNet V2L with methodology 1 - 8 batch size - 224x224 resolution

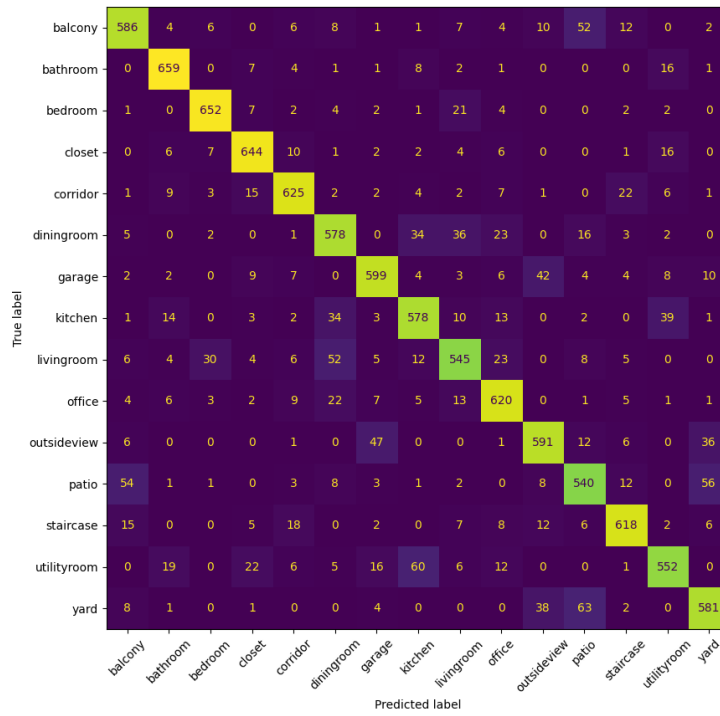


Figure B.2: Confusion matrix of the model based on EfficientNet V2L with methodology 2 - 32 batch size - 224x224 resolution

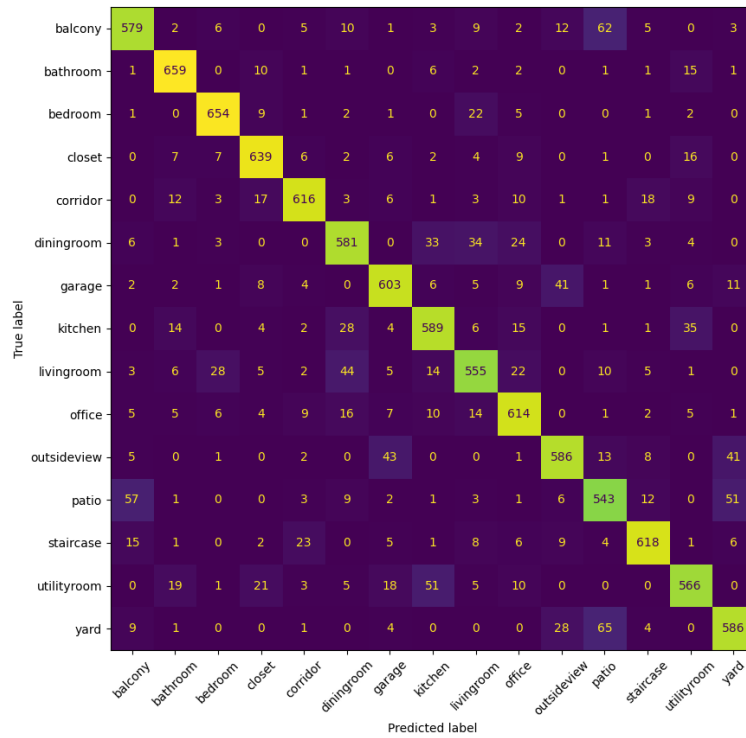


Figure B.3: Confusion matrix of the model based on EfficientNet V2L with methodology 3 - 32 batch size - 224x224 resolution

Appendix C

Confusion Matrix Analysis of ConvNeXt Models

This appendix presents the analysis of the confusion matrices of the ConvNeXt models. Commencing with the confusion matrix of the first methodology, shown in figure C.1, it exhibits, as expected, equal or better values than those present in the best models of the EfficientNet architecture for all classes with a few exceptions of the labels outside view, patio and balcony by a very little margin. This means that the problems related to the outside of the house labels are still very much present, despite the remaining previous issues being less existing, in the case of the kitchen, dining and living room classifications.

In the remaining two methodologies, in figures C.2 and C.3, there is not much to be pointed out, since the overall result was very similar to the best ones of the EfficientNet networks and these did not result in much of a progress.

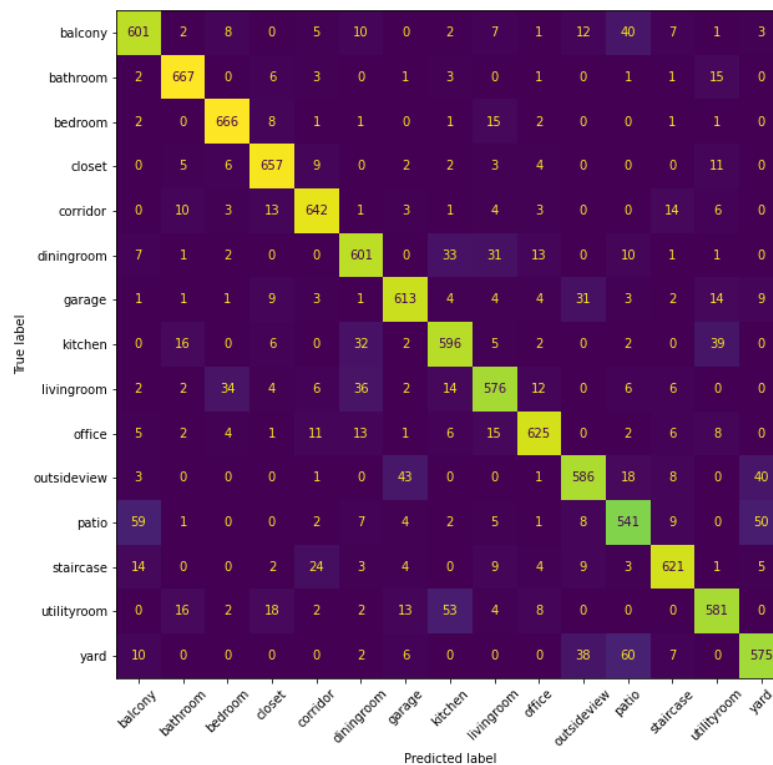


Figure C.1: Confusion matrix of the model based on ConvNeXt Large with methodology 1 - 8 batch size - 224x224 resolution

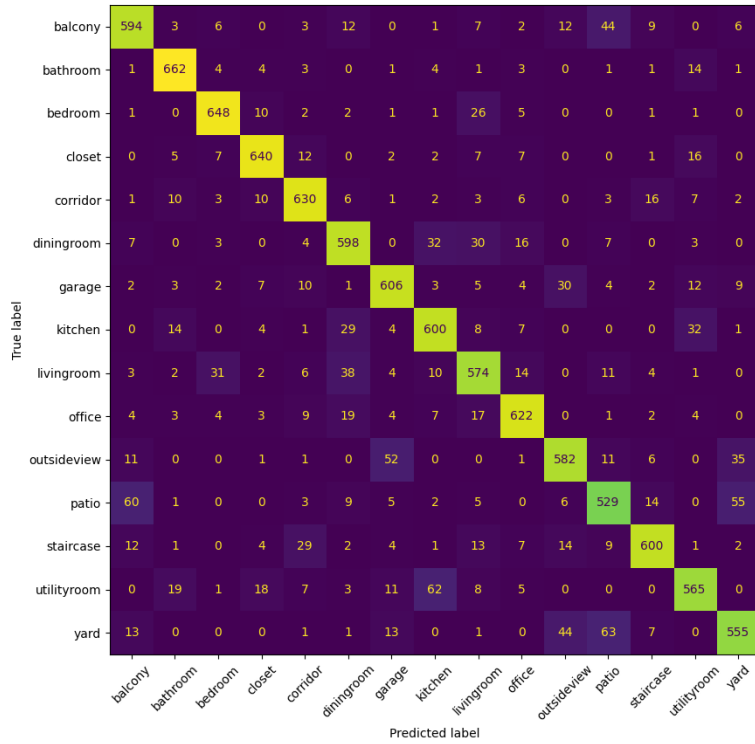


Figure C.2: Confusion matrix of the model based on ConvNeXt Large with methodology 2 - 32 batch size - 224x224 resolution

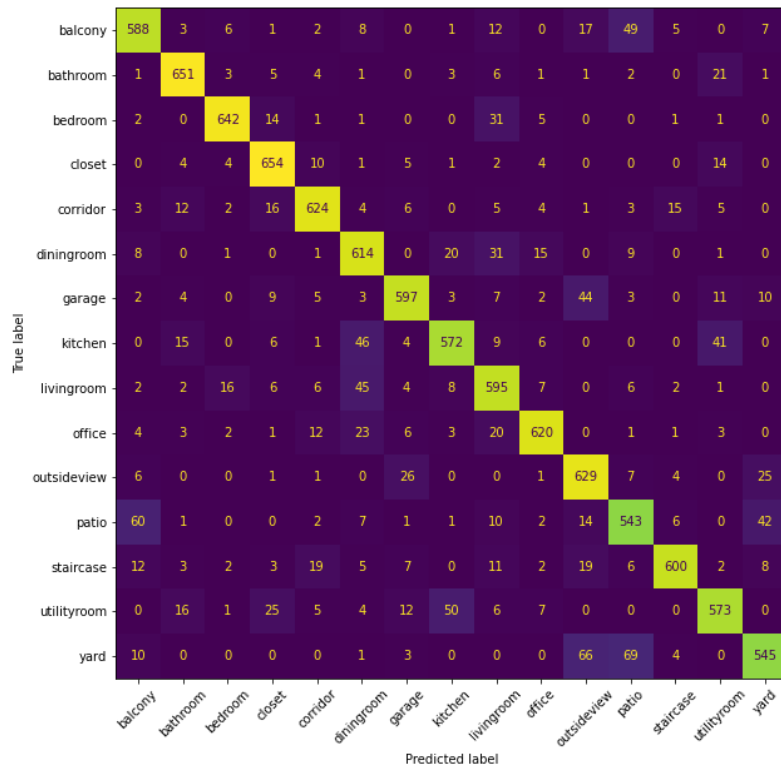


Figure C.3: Confusion matrix of the model based on ConvNeXt Large with methodology 3 - 32 batch size - 224x224 resolution

Appendix D

Confusion Matrix Analysis of ViT Models

In this appendix the confusion matrices of the ViT models are shown, in figures D.1, D.2 and D.3, where there is not a lot to point out, since the overall more specific results of each class don't change much from the ones seen before, which means that previously shown problems are still present.

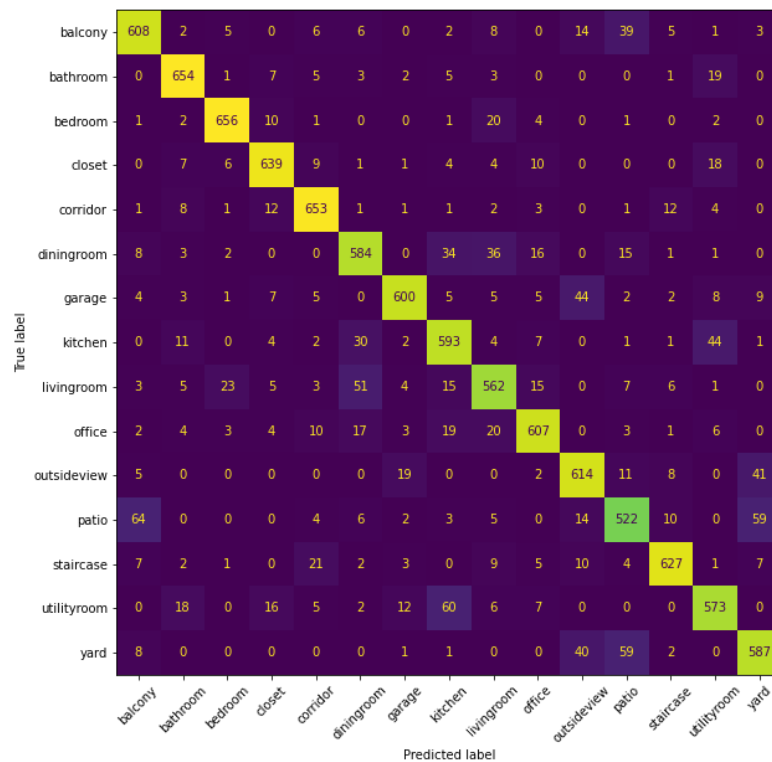


Figure D.1: Confusion matrix of the model based on ViT Base Patch 16 with methodology 1 - 32 batch size - 224x224 resolution

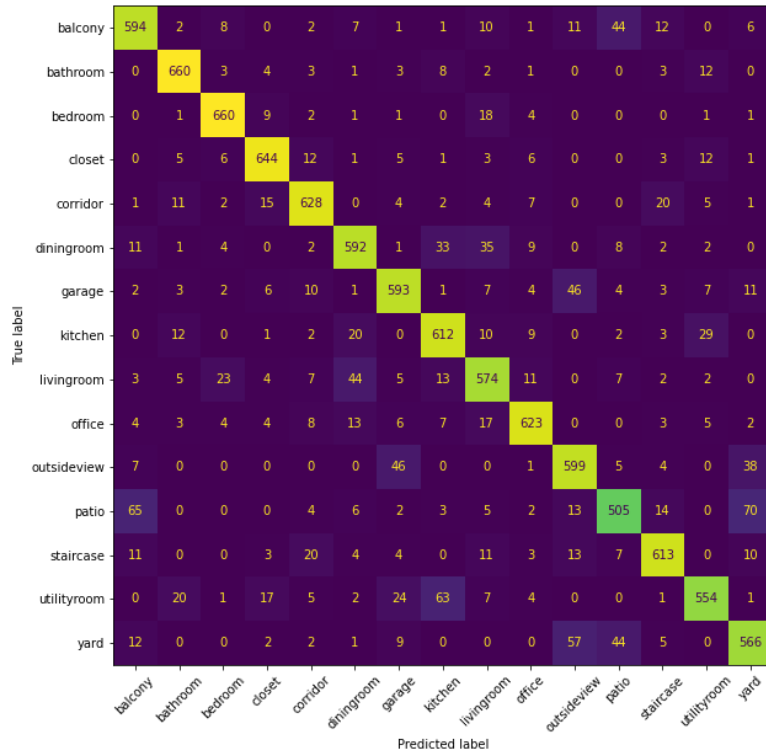


Figure D.2: Confusion matrix of the model based on ViT Base Patch 16 with methodology 2 - 32 batch size - 224x224 resolution

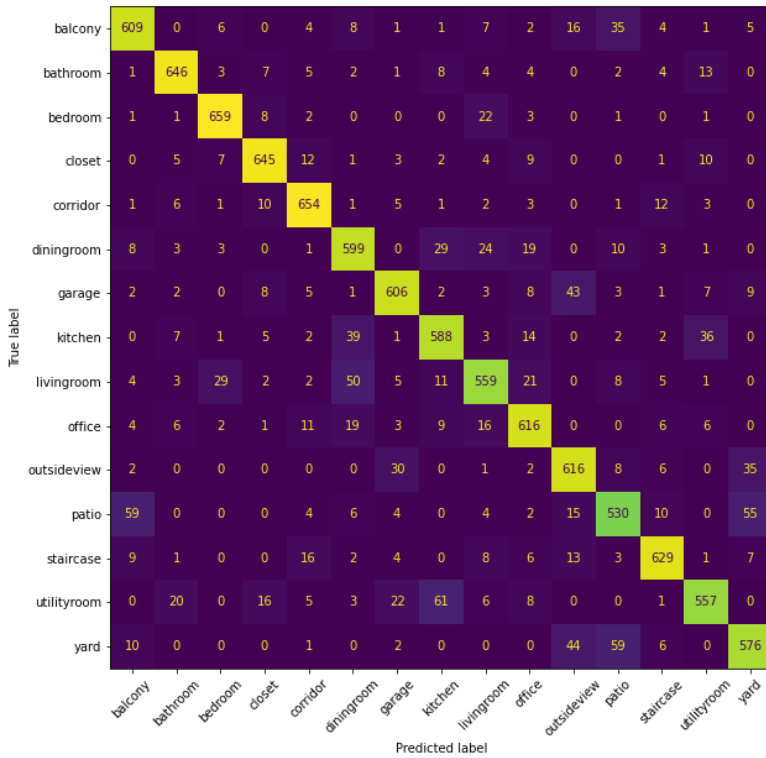


Figure D.3: Confusion matrix of the model based on ViT Base Patch 16 with methodology 3 - 32 batch size - 224x224 resolution

Appendix E

Confusion Matrix Analysis of CoCa Models

The final appendix presents the analysis of the confusion matrices of Methodologies 1,2 and 3 of the CoCa models which are present in figures E.1, E.2 and E.3, respectively.

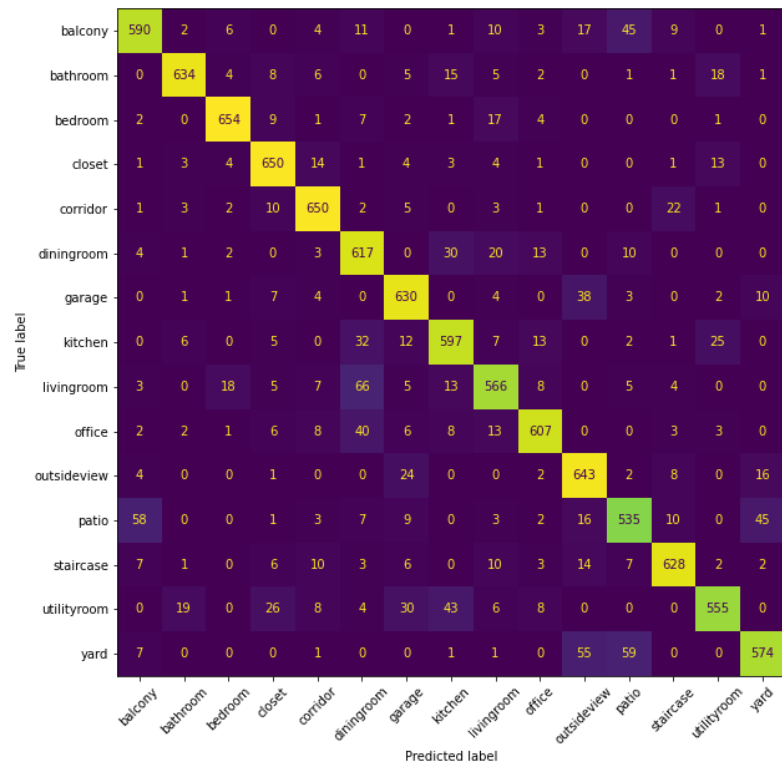


Figure E.1: Confusion matrix of the model based on CoCa Large with methodology 1 - 4 batch size - 224x224 resolution

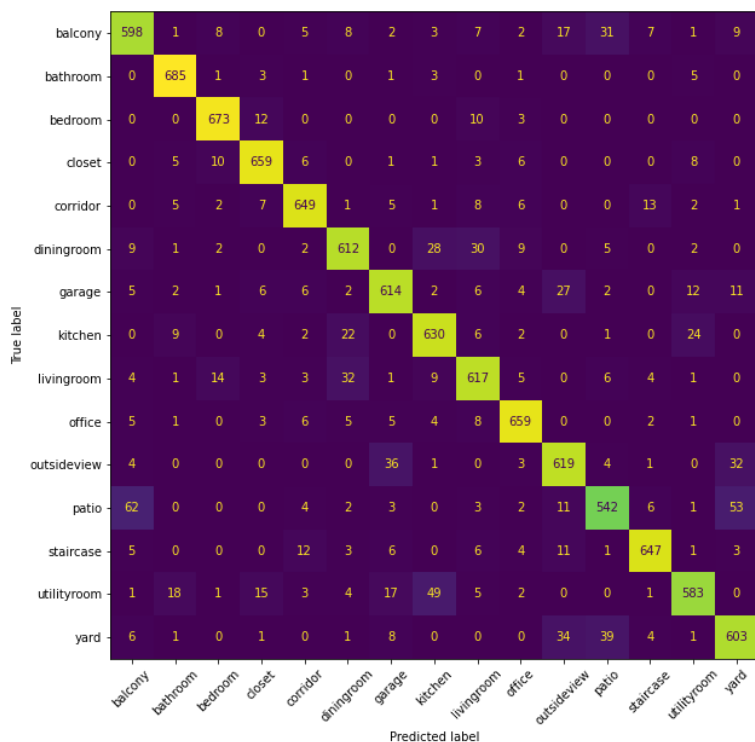


Figure E.2: Confusion matrix of the model based on CoCa Large with methodology 2 - 32 batch size - 224x224 resolution

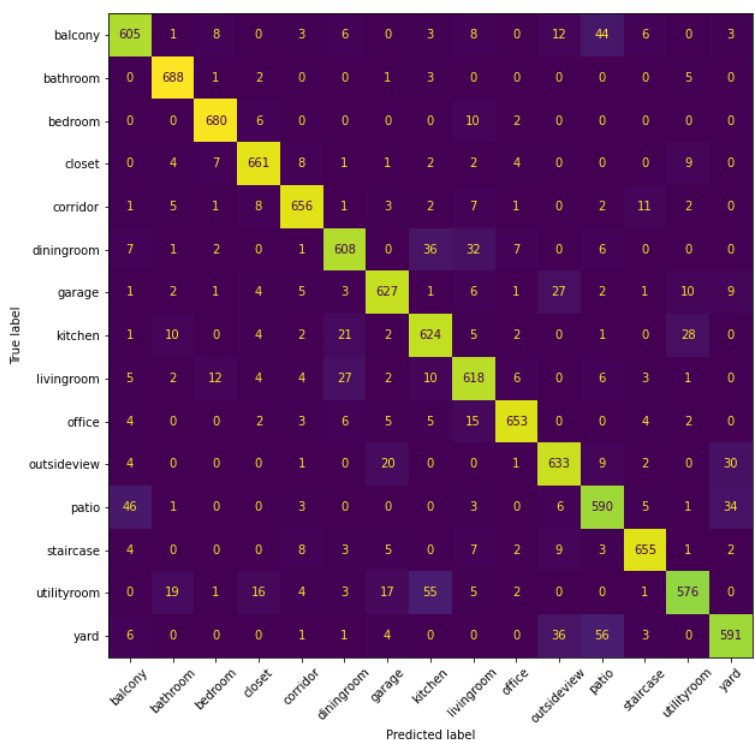


Figure E.3: Confusion matrix of the model based on CoCa Large with methodology 3 - 32 batch size - 224x224 resolution

While the first methodology certainly does not improve much from previously seen results, the last two show a big improvement in problematic classes inside and outside of the house, with patio being the only one that improved less in both. In fact, in both images, only three classes have true positives below 600 and not by much, with the worst case in Methodology 2 being the patio class, which is missing 58 to reach the 600 threshold, and utility room in Methodology 3, which is missing 24. This means that both methodologies present very good results and considering that many images may contain several classes due to non existing physical boundaries in some cases, like patio and yard, means that the model may predict well what it is in the image even though it does not correspond the the expected label contributing to a lower accuracy. All of this indicates that these accuracy values are likely superior if further analysis is conducted in the supposed wrongly classified images.