

University Fernando Pessoa



Master Thesis

**Deep Learning for Building Stock
Classification for Seismic Risk Analysis**

Author

Jorge Miguel Soares Lopes

*Dissertation presented to University Fernando Pessoa as part of the requirements
for the degree of Master in Computer Engineering, Mobile Computing*

Supervisor

Prof. Feliz Ribeiro Gouveia

October 2023

RESUMO

Nas últimas décadas, a maioria dos esforços para catalogar e caracterizar o ambiente construído para a avaliação de riscos múltiplos têm-se concentrado na exploração de dados censitários habitacionais, conjuntos de dados cadastrais e pesquisas locais. A primeira abordagem é atualizada apenas a cada 10 anos e não fornece informações sobre as localizações dos edifícios. O segundo tipo de dados está disponível apenas para algumas áreas urbanas, e a terceira abordagem requer levantamentos realizados por profissionais com formação em engenharia, o que é proibitivo em termos de custo para estudos de risco em larga escala. Portanto, é evidente que os métodos para caracterizar o ambiente construído para a análise de riscos em larga escala, estão atualmente ausentes, o que dificulta a avaliação do impacto de fenómenos naturais para fins de gestão de riscos. Alguns esforços recentes têm demonstrado como algoritmos de aprendizagem-máquina podem ser treinados para reconhecer características arquitetônicas e estruturais específicas dos edifícios a partir de imagens das suas fachadas e propor, de forma probabilística, uma ou várias classes de edifícios. Neste estudo, demonstrou-se como tais algoritmos podem ser combinados com dados do OpenStreetMap e imagens do Google Street View para desenvolver modelos de exposição para a análise de riscos múltiplos. Um conjunto de dados foi construído com aproximadamente 5000 imagens de edifícios da freguesia de Alvalade, no distrito de Lisboa (Portugal). Esse conjunto foi utilizado para testar diferentes algoritmos, resultando em níveis de desempenho e exatidão distintos. O melhor resultado foi obtido com o Xception, com uma exatidão de cerca de 86%, seguido do DenseNet201, do InceptionResNetV2 e do InceptionV3, todos com exatidões superiores a 83%. Estes resultados servirão de suporte a futuros desenvolvimentos na avaliação de modelos de exposição para análise de risco sísmico. A novidade deste trabalho consiste no número de características de edifícios presentes no conjunto de dados, no número de modelos de aprendizagem profunda treinados e no número de classes que podem ser utilizadas para construir modelos de exposição.

Palavras-chave: Aprendizagem Profunda; Redes Neurais Convolucionais; Visão Computacional; Modelos de Exposição de Edifícios; Análise de Risco Sísmico.

ABSTRACT

In the last decades, most efforts to catalog and characterize the built environment for multi-hazard risk assessment have focused on the exploration of housing census data, cadastral datasets, and local surveys. The first approach is only updated every 10 years and does not provide information on building locations. The second type of data is only available for some urban areas, and the third approach requires surveys carried out by professionals with an engineering background, which is cost-prohibitive for large-scale risk studies. It is thus clear that methods to characterize the built environment for large-scale risk analysis at the asset level are currently missing, which hampers the assessment of the impact of natural hazards for the purposes of risk management. Some recent efforts have demonstrated how machine learning algorithms can be trained to recognize specific architectural and structural features of buildings based on their facades, and probabilistically propose one or multiple building classes. This study demonstrates how such algorithms can be combined with data from OpenStreetMap and imagery from Google Street View to develop exposure models for multi-hazard risk analysis. A dataset was built with approximately 5000 images of buildings from the parish of Alvalade, within the Lisbon district (Portugal). This dataset was used to test different algorithms, which led to distinct performance and accuracy levels. The best result was obtained with Xception, with an accuracy of approximately 86%, followed by DenseNet201, InceptionResNetV2 and InceptionV3, all with accuracies above 83%. These results will support future developments for assessing exposure models for seismic risk analysis. The novelty of this work consists in the number of building characteristics present in the dataset, the number of deep learning models trained and the number of classes that can be used for building exposure models.

Keywords: Deep Learning; Convolutional Neural Networks; Computer Vision; Building Exposure Models; Seismic Risk Analysis;

ACKNOWLEDGEMENTS

To my research and master's thesis supervisor, Professor Feliz Ribeiro Gouveia, for his dedication, support, and guidance during the development of this work. Also, a mention to my research scholarship co-supervisor, Professor Vitor Silva, who during this period transmitted me a lot of knowledge related to the work developed.

This work was inspired by the project AI4DRR (Artificial Intelligence for Disaster Risk Reduction), funded by Fundação para a Ciência e Tecnologia, I.P., through research grant EXPL/ECI-EGC/1555/202, along with other academic projects previously undertaken in the field of the dissertation, i.e., “Activity Recognition in Intelligent Spaces”.

To Paulo Yannick, Rui Gomes, Vander Escovalo, and Romain Sousa who played a key role in this work in the field image collection and further manual annotation. Without them, this work would not have been possible.

To my parents and sister for always believing in my choices, supporting me unconditionally. I know I couldn't have gotten this far without them! Every achievement I make is proof of the positive impact they have had on my life.

To my dear and true friends who, present or not, through words, gestures, support or affection, encouraged me to fulfill another chapter of my life.

To each and all of you who contributed at their best to this final result - Thank you!

INDEX

LIST OF FIGURES	VII
LIST OF TABLES	VIII
ACRONYMS AND DEFINITIONS	IX
1. INTRODUCTION	1
1.1 Problem Statement	1
1.2 Proposed Approach	2
1.3 Contributions	4
2. LITERATURE REVIEW	5
2.1 Building Classification	5
2.2 Exposure Models	7
3. TOOLS	10
3.1 OpenStreetMap	10
3.2 QGIS	11
3.3 Google Street View Static API	12
4. DATASET CONSTRUCTION	13
4.1 Project Pipeline	13
4.2 Data Gathering	15
4.3 Feature Collection	17
4.4 Analysis and Processing of Collected Data	19
4.4.1 Outliers Removal Algorithm	19
4.4.2 Black Edge Trimming	20
4.4.3 Data Annotation (Web Platform)	22
4.5 Creation of the Dataset	25
4.5.1 Cleaning the Dataset	25

4.5.2	Classes Configuration.....	25
4.5.2.1	Configuration A.....	26
4.5.2.2	Configuration B.....	27
4.5.2.3	Configuration C.....	28
4.6	Splitting.....	29
4.7	Image Augmentation.....	30
4.8	Class Weights	31
5.	MACHINE LEARNING ARCHITECTURES	32
5.1	Selected Models.....	32
5.1.1	DenseNet201	33
5.1.2	InceptionResNetV2	34
5.1.3	InceptionV3	35
5.1.4	NasNetLarge.....	36
5.1.5	ResNet50V2	37
5.1.6	Xception	38
6.	TRAINING	39
6.1	Transfer Learning	39
6.1.1	Feature Extraction.....	40
6.1.2	Fine-Tuning	41
6.2	Hyperparameter Tuning.....	42
6.2.1	Regularization Methods.....	42
6.2.2	Evaluation Metrics.....	43
6.2.2.1	Accuracy.....	43
6.2.2.2	Precision.....	44
6.2.2.3	Recall.....	44
6.2.2.4	F1 Score.....	44
6.2.3	Optimization Algorithms.....	44

6.3	Training History.....	45
7.	RESULTS	47
8.	DISCUSSION AND CONCLUSION	54
9.	BIBLIOGRAPHY	57

LIST OF FIGURES

Figure 3.1 - Buildings in Alvalade as shown on OpenStreetMap.	10
Figure 3.2 - Division of Alvalade buildings by zones and subzones.	11
Figure 4.1 - Overall project pipeline.	14
Figure 4.2 - A.1 subzone excels file example.....	17
Figure 4.3 - Before and after removing black borders.	20
Figure 4.4 - OpenCV developed code to remove black borders from images.	21
Figure 4.5 - Interface for viewing the contents of the database.	23
Figure 4.6 - Image and classification interface for a building.....	24
Figure 4.7 - Configuration A classes frequency.	26
Figure 4.8 - Configuration B classes frequency.	27
Figure 4.9 - Configuration C classes frequency.	28
Figure 4.10 - Dataset splitting for configuration A.	29
Figure 6.1 - Transfer Learning Feature Extraction VS Fine-Tuning procedures.	40
Figure 6.2 - Model layers order and organization.	42
Figure 6.3 - Model fitting callbacks.	43
Figure 6.4 - Xception model training history.	45
Figure 6.5 - InceptionV3 model training history.....	45
Figure 7.1 - Confusion matrix for configuration A and Xception model trained with fine-tuning.....	49
Figure 7.2 - ROC Curves for configuration A and Xception model trained with fine-tuning.....	51
Figure 7.3 - Three randomly picked activation heatmaps from the test subset.	52
Figure 7.4 - Three activation heatmaps from the test subset with mispredictions.	53

LIST OF TABLES

Table 8.1 - GSV Static API response image and related information.....	15
Table 8.2 - Images captured on the field where the building facade isn't clearly visible.	19
Table 8.3 - Option for expert assessment.	22
Table 8.4 - Configuration A classes details.....	26
Table 8.5 - Configuration B classes details.	27
Table 8.6 - Configuration C classes details.	28
Table 9.1 - Input sizes of pre-trained models.	32
Table 10.1 - Training history results for each model with fine-tuning (configuration A).	46
Table 11.1 - Accuracy results comparison between each model-configuration with fine- tuning (training w/70%).	48
Table 11.2 - Accuracy results comparison between each model-configuration with feature extraction (training w/80%).	48
Table 11.3 - Accuracy results comparison between each model-configuration with fine- tuning (training w/80%).	48
Table 11.4 - Classification report for configuration A and Xception model trained with fine-tuning.	49

ACRONYMS AND DEFINITIONS

AI4DRR	Artificial Intelligence for Disaster Risk Reduction
API	Application Programming Interface
AUC	Area Under the Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FoV	Field of View
DCNN	Deep Convolutional Neural Network
GEM	Global Earthquake Model
GIS	Geographic Information Systems
GPS	Global Positioning System
GPU	Graphics Processing Unit
Grad-CAM	Gradient-Weighted Class Activation Mapping
GSV	Google Street View
LLRS	Lateral Load Resisting System
ML	Machine Learning
OSM	OpenStreetMap
OvA	One-VS-All
QGIS	Quantum Geographic Information System
RAM	Random Access Memory
ROC	Receiver Operating Characteristic
SBST	Seismic Building Structural Type

1. INTRODUCTION

1.1 Problem Statement

Droughts, floods, storms and earthquakes have been the main cause of casualties and economic losses ever since (Ritchie & Roser, 2014). Consequently, the disaster risk management caused by these natural disasters is a permanent worldwide concern.

Exposure models play a crucial role in assessing a building's vulnerability to seismic events. These models use building information, such as construction material, year or period of construction, number of storeys, number of doors and windows and other structural information to make them as useful as possible. However, ensuring access to up-to-date, complete, and reliable building-level information is a challenge, especially in specific regions and cities. Traditional methods for collecting building data typically involve housing census and cadastral data.

The housing census has the advantage of providing nationwide coverage, but the spatial resolution makes the data insufficient for assessing the impact of localized catastrophes such as floods or landslides or even earthquakes. Cadastral data, on the other hand, is frequently collected at the building level, but it is only available for a restricted number of places and ignores vulnerability-related factors such as the primary construction method or the presence of structural problems. Due to a lack of accurate information on the built environment, it is impossible to determine the potential for damage, economic losses, and fatalities, which might aid in the development of disaster risk management methods. Similarly, a lack of data makes it difficult to estimate the impact of natural or anthropogenic risks as soon as they occur. In the case of Portugal, the assessment of the impact of natural disasters is based on observations and damage data that are normally published weeks or months after the event, influencing the strategic allocation of human resources or the early release of financial help to promote recovery. But why choose the parish of Alvalade as the study area? This region of Portugal has been the focus of several prior studies that provide actual earthquake data specific to this location. The goal of these studies was to develop an extensive understanding of the type of construction we should

anticipate in this specific region. Additionally, the level of seismic hazards and risk in this area is quite high, making it particularly important for assessing the risk of earthquakes.

According to (Silva et al., 2014), the number of people affected as well as the economic losses for a given earthquake are determined by the frequency and size of earthquakes in the study region (seismic hazard), the inventory of people and infrastructure (exposure), and the capacity of buildings to handle earthquake loading (vulnerability).

In order to establish effective emergency plans, it is crucial to possess and maintain trustworthy and up-to-date exposure models. These provide a comprehensive overview of all assets in the region under study. However, developing such models is a complex task, especially in regions where data is not consistently gathered by government entities, such as emerging nations. In some cases, data from local surveys, housing census records, or even cadastral databases may be used to acquire information on building characteristics when analyzing specific regions (if available). Nevertheless, as the region grows larger, it becomes more expensive and time-consuming to accomplish these activities, presenting a significant challenge for researchers as well as politicians who attempt to quickly and affordably build large-scale exposure models.

1.2 Proposed Approach

This dissertation addresses this kind of problems by looking at innovative solutions, in particular, how machine learning algorithms may be trained to identify specific building architectural and structural characteristics from images of their facades. Additionally, these algorithms can probabilistically suggest one or more building classes. This methodology contributes to advancing earthquake risk assessment and management by offering a scalable and effective solution to the data constraints that often limit the development of exposure models.

As a result, it is evident that using traditional approaches for characterization of the built environment for risk assessment on a national or regional scale is unsustainable. To achieve such a milestone, it is critical to rely on recent advancements in crowdsourcing, satellite imagery, big data, and artificial intelligence. The OpenStreetMap (OSM) project, an editable map of the world with information given by volunteers, is one example of a

growing supply of building information. OSM is typically limited to the building footprint and main use, but it provides adequate coverage for most urban areas. Satellite imagery released in the previous decade has also reached a degree of spatial resolution that permits determining exposure factors such as the height of buildings and large structures (Cao & Huang, 2021). Finally, the well-known Google Street View service allows users to explore and visualize building imagery even in remote locations. While none of these sources of information are sufficient in and of themselves to produce an accurate and comprehensive exposure model, their major characteristics can be merged using an artificial intelligence framework to create digital models of entire regions for disaster risk reduction.

This work exploits Google Street View images and OpenStreetMap building footprints to construct a comprehensive dataset for training deep learning models. The dataset was meticulously manually annotated on-site, simplifying the process and increasing the collection of building facade images. Additionally, a selection of deep learning models was made, based on their past performance in similar tasks. The study not only evaluates and contrasts the performance of these chosen models but also recommends additional dataset structuring for a more nuanced analysis of the building stock.

Following the construction of the dataset, algorithm training, and comparisons of inference results, the primary objective is to employ the obtained data to create an exposure model that will be used to estimate the impact of an earthquake scenario. Subsequently, these estimated results can be compared with the impact calculated using ground truth data, providing a comprehensive evaluation of the model's predictive accuracy and reliability.

1.3 Contributions

Due to the significance and results of the work undertaken in this project, scientific publications have already been made, with others currently under submission. It is important to highlight not only the importance of these contributions to the scientific community but also the direct impact these results have on the real world.

- Feliz Gouveia, Vítor Silva, Jorge Lopes, Rui Moreira, José Torres, Maria Guerreiro, “*Automated Identification of Building Features with Deep Learning for Risk Analysis*”, **SN Applied Sciences**, Springer Nature, ISSN 2523-3971. (Submitted)
- Vitor Silva, Romain Sousa, Feliz Gouveia, Jorge Lopes, Maria João Guerreiro, “*A Building Imagery Database for the Calibration of Machine Learning Algorithms*”, **Earthquake Spectra**.
- Vitor Silva, Jorge Lopes, Feliz Gouveia, Romain Sousa, “*Exposure Modeling through Machine Learning for Multi-Hazard Risk Assessment*”, **14th International Conference on Applications of Statistics and Probability in Civil Engineering**, ICASP14, July 9-13, 2023, Dublin, Ireland.
- Jorge Lopes, Feliz Gouveia, Vítor Silva, Rui Moreira, José Torres, Maria Guerreiro, Luís Paulo Reis, “*Using Deep Learning for Building Stock Classification in Seismic Risk Analysis*”, **Encontro Português de Inteligência Artificial**, EPIA 2023, September 5-8, 2023, Ilha do Faial, Azores, Portugal.

2. LITERATURE REVIEW

In the past years there has been a growing interest in employing deep learning techniques to accomplish and simplify human tasks. For example, various studies have previously been conducted in which the usage of Google Street View (GSV) images proved to be effective in detecting building facades with graffiti artwork (Novack et al., 2020), urban frontage classification (Law et al., 2020), visual screening of soft-story buildings (Yu et al., 2020), classification of building's utility classes (Laupheimer et al., 2018), seismic damage prediction (Bhatta & Dang, 2023) and estimating building age (Li et al., 2018). There has also been some work on the prediction of building features, such as construction material and age, which is directly relevant to this work and is presented in the following sections.

2.1 Building Classification

Kang et al. (2018) proposed to use deep learning to classify building facades from street view images. The authors retrieved, for several cities in Canada and the United States, the building footprints and their geographic locations from online geographic information systems (GIS), such as OpenStreetMap¹.

Since each building had an associated GPS coordinate (latitude, longitude), the authors downloaded each image using the Google Street View Static API² with the corresponding metadata³, i.e., the image size and pitch value defined as 512 x 512 pixels and 10 degrees, respectively, which show facade structures of individual buildings.

Due to the uncontrolled quality of street view images, many of them could not be directly utilized for the building classification. For example, some of the images were taken from

¹ <https://www.openstreetmap.org/>

² <https://developers.google.com/maps/documentation/streetview/overview>

³ <https://developers.google.com/maps/documentation/streetview/metadata>

the inside the building and others had the facade hidden by vehicles and trees. The authors removed those outliers using the VGG16 model (Simonyan & Zisserman, 2015) trained on the Places2 database (Zhou et al., 2018).

This process resulted in a dataset with 19658 street view images from eight classes, i.e., *apartment, church, garage, house, industrial, office building, retail and roof*, each with approximately 2500 images, in other words, a balanced dataset.

The authors chose to fine-tune all the convolutional layers of pre-trained state-of-the-art CNNs (Convolutional Neural Networks) on a large dataset, such as ImageNet (Russakovsky et al., 2015), however now applied to a lower sized dataset. For example, for the training of the CNNs used, i.e., AlexNet (Krizhevsky et al., 2017), VGG16, ResNet18 and ResNet34, they experienced overfitting behaviors in both ResNet architectures (He et al., 2016a). Overall, the architecture with worst classification performance was AlexNet and the best was VGG16 with an F1 Score of 0.53 and 0.58, respectively.

2.2 Exposure Models

In Santiago (Chile's capital), a city prone to earthquakes, Aravena Pelizari et al. (2021) created a reference dataset based on which they assess the potential of DCNNs (Deep Convolutional Neural Networks) to predict a risk-oriented SBST (Seismic Building Structural Type) and individually estimate the LLRS (lateral load resisting system) material and the height of buildings. In order to categorize building exposure in a standardized way, by using the GEM or the GED4ALL taxonomy, an adaptation for multi-hazard risk analysis (Silva et al., 2018), the authors seek to automatize the classification of structural features of buildings for large-area seismic risk assessments.

Unlike the work mentioned previously, these authors obtained the buildings information and geo-location through property cadastral data from the Chilean Internal Revenue Service and the Ministry of Housing and Urbanism as well.

Aravena Pelizari et al. (2021) used the exact same process of Kang et al. (2018) for downloading the building images and also to remove the outliers; both with minor changes in parameters definition.

They downloaded three street-level images for every building using the Google Street View Static API with a size of 640 x 640 pixels, FOV of 100 degrees and pitch of 15 degrees; the first image downloaded corresponded to the GPS location of the building and the second and third images corresponded to a 90 degrees deviation, i.e., right (+90°) and left (-90°) side views. For the images that remained with non-facade building being visible, the authors also used VGG16 DCNN trained on the Places365⁴ dataset, the latest subset of Places2 database, to retrieve the top 4 predictions filtering the intersection of these with a group of 24 predefined outdoor classes, such as: *apartment building, beach house, building facade, chalet, church, cottage, courthouse, embassy, fire station, hangar, hospital, hotel, house, hunting lodge, mansion, manufactured home, motel, office building, palace, school-house, shed, skyscraper, synagogue, tower*; if in the top 4 model predictions, 2 or more belonged to the 24 classes of the group, a building facade was

⁴ <https://github.com/CSAILVision/places365>

considered as found, otherwise, the image was discarded (meaning the facade was not visible).

Their dataset had a total of 204,030 filtered building facade images and was used to train from scratch, and through transfer learning, several pre-trained architectures, such as InceptionResNetV2 (Szegedy et al., 2017), Xception (Chollet, 2017) and NasNet-A (Zoph et al., 2018) obtaining very promising results. The transfer-learned NasNet-A model has performed the best overall with accuracy bigger than 0.80 and 0.85 for SBST and for LLRS material and height prediction, respectively.

Unlike the works previously mentioned, Gonzalez et al. (2020), manually annotated and filtered (without using any possible outlier's removal algorithm) a dataset containing approximately 10,000 GSV facade images of buildings within the Medellín (Colombia) urban area. The authors did not however specify how they obtained the geo-location of each building (coordinates or addresses) to feed into the GSV Static API.

Their objective was the same as the Aravena Pelizari et al. (2021) work, that is, to predict the building material and the lateral load-resisting system type. For that, they selected five state-of-the-art CNNs that also have shown great results in ImageNet: VGG16, VGG19, InceptionV3 (Szegedy et al., 2016), ResNet50 (He et al., 2016a), and Xception.

Among the five network architectures trained in their study, ResNet50 showed the best performance because it classified fewer non-ductile buildings as ductile. However, overall, the results were not as good as expected, falling short of the results of the previously mentioned works.

In Oslo, Norway, Ghione et al. (2022) have also performed a study similar to the one in this document and other works already mentioned. The main goal remains the same: to develop a cost-effective building exposure model. To this end, the general information of the buildings, such as their total number with the corresponding coordinates, number of stories, number of housing units, usable and total area, was obtained from the public cadastre of Norway, which compares and manages detailed public geographic information of the country. As well as the other works, the street view facade images were retrieved from GSV Static API using the building location. In addition to those, the authors also photographed the facades of the buildings on location. Then, with the help

of earthquake engineering experts, all the images were labeled and filtered using a topology classification.

They did not manually check the quality of the images (whether the facades were occluded by trees, passing vehicles or scaffoldings), instead, during the labeling process, they labeled those images as “other”, expecting that it would result in a considerable source of uncertainty for training the models.

Their dataset had a total of 5,074 manually labeled images from fieldwork and from GSV divided into test (20%) and train (80%). At training, looking at the lower number of images compared to other works, the authors increased the size of the dataset artificially, i.e., by using image augmentation, by applying randomly zooming in by up to 20%, randomly rotating by up to 25° and randomly mirroring along the vertical axis. Still in the training process, by choice, the authors did not apply class-specific weights, although the dataset was unbalanced, i.e., all images were given equal priority. According to the authors, this should increase the final overall accuracy.

They used state-of-the-art CNNs pre-trained on the ImageNet database, such as Xception, VGG16, VGG19, ResNet50V2 (He et al., 2016b), InceptionV3, InceptionResNetV2, DenseNet201 (Huang et al., 2017).

The fine-tuned DenseNet201, classified typology in previously unseen images with 82.5% accuracy, using only data sources available online: the public cadastre and Google Street View. Without fine-tuning, this model had an accuracy of only 76.3%, showing that fine-tuning is greatly beneficial to performance.

3. TOOLS

3.1 OpenStreetMap

OpenStreetMap (OSM) is a collaborative tool that provides an editable geographic database completely free of charge. Through it, by layering satellite images, it is possible to view, add or edit polygons (building footprints) in a given area, in this case the Alvalade parish, and also manipulate specific information linked with the polygon, such as the type, name, address, or height of the building. Because any user can edit the map, it is vital to maintain a critical and cautious eye during this process, as a single polygon may sometimes delimit two or more buildings, which is incorrect because a polygon should delimit one and only one building.

In the case of Alvalade, it is worth mentioning that, whether correctly or incorrectly delimited, almost 75% of the parish already had the buildings bounded. Therefore, the remaining 25% had to be done manually.



Figure 3.1 - Buildings in Alvalade as shown on OpenStreetMap.

3.2 QGIS

QGIS (Quantum Geographic Information System) is a free and open-source geographic information system (GIS) application that supports the visualization, editing, and analysis of geospatial data. Using one of the available plugins, “QuickOSM”, queries were executed on the extracted OSM layer to acquire all the objects marked with the tag “building”, that is, all the previously delimited buildings. This allowed to manipulate Alvalade data, and based on the area of each building, the center point of each polygon (also called the centroid) and its corresponding location (latitude and longitude) was geometrically calculated. As mentioned before, besides accessing and manipulating the characteristics that derive from the OSM, such as the *osm_id* (unique building footprint identifier), it is also possible to visualize all buildings, assign them an identifier (id), and divide them by zones and, consequently, sub-zones.

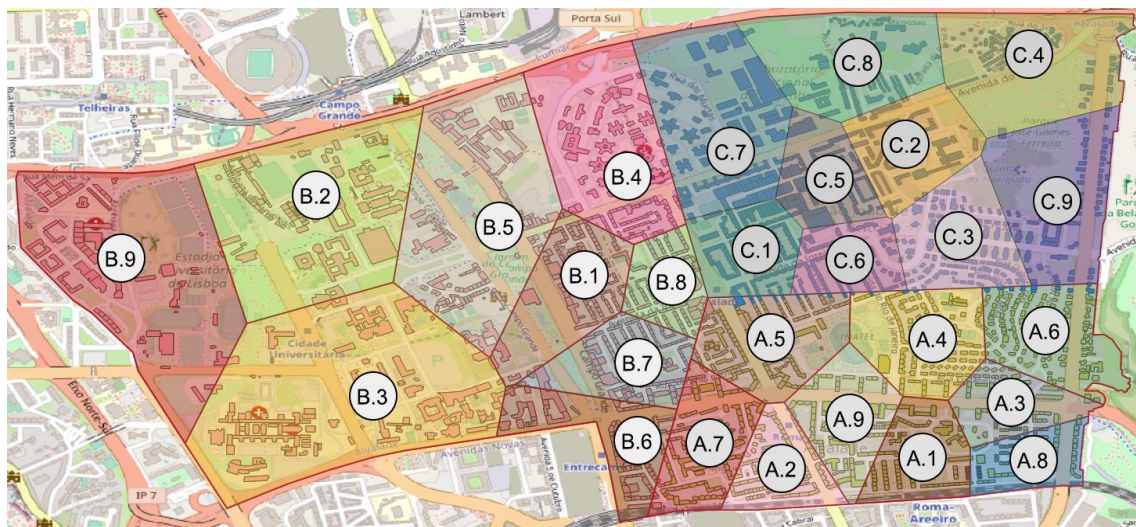


Figure 3.2 - Division of Alvalade buildings by zones and subzones.

3.3 Google Street View Static API

Google Street View (GSV) is a Google Maps and Google Earth feature that provides panoramic views of numerous areas around the world, allowing users to virtually explore streets and landmarks as if they were physically present. Using the Street View Static API, for a given *osm_id* and the associated building location, i.e., the geographic coordinate pair of its center point, an image is returned that, most of the time, corresponds to its facade (if it exists, of course).

For this work, and for each location, the camera parameters were previously defined for all the requests, where *pitch* (camera orientation angle, e.g., 90° : up and -90° : down) was set to 20° , *source* set to outdoor (limiting searches to outdoor collections) and *radius* set to 150 meters (maximum distance to search for a panorama). These values were not randomly chosen; for example, the pitch was defined to avoid seeing cars, people, sidewalks, or the road itself; the source was selected because we only want the facades of the buildings; and the radius was set because some buildings had a larger area, causing the distance from the central point to the street to be greater than 50 meters (the default radius), causing the API to return no image. All the remaining parameters were not specified, being automatically initialized by the API defaults.

The main purpose of using this tool is to automate the process of collecting images of the facades and compare the results obtained between these images and those that were later collected in the field. In case of a failure, i.e., the API cannot find images for the specified building, no image will be stored in the database. However, such buildings (without image) are kept since later it will be necessary to collect data regarding their characteristics (construction material, number of floors, among others) as well as to capture images of them on the field.

4. DATASET CONSTRUCTION

4.1 Project Pipeline

Natural disasters, such as earthquakes, can create a wide range of issues, including loss of life, injury, damage to buildings and infrastructure, and displacement of people. For example, when attempting to reduce disaster risk, the construction material, height, and construction period are some of the most important building features to consider.

The most common ways to collect these types of data are through cadastral data or surveys, such as housing census, crowdsourcing, such as the OpenStreetMap platform, and Geographic Information Systems, such as Quantum Geographic Information System (QGIS) or even Google Street View (in part).

Recently, as described in the previous sections, it is possible to automate this process of collecting data, such as the properties of the building.

As with any project involving deep learning, first of all, it is necessary to have an annotated dataset for a given problem context. In this case, based on data retrieved using OpenStreetMap and using the Google Street View Static API, street-level images of the facade of each building in the Alvalade parish were collected. In addition, in order to increase the amount of data in the dataset, groundwork was also performed where, for each building, up to three images of the building's facade were taken from different angles. With the help of civil engineers, a quality control of the collected data was also performed during the labeling process in order to obtain the best results for the future.

Gathering data is a crucial phase in any machine learning project since it supplies the raw material needed to train and refine models. Machine learning algorithms cannot be trained effectively without sufficient and high-quality data, as the resulting models will be inaccurate and unreliable. Additionally, both human skills and technical tools are required in the data collection process, as they assist in identifying and collecting important data, preprocessing and cleaning it, and assuring its accuracy and completeness. As a result,

devoting time and resources to this phase is critical to the success of any machine learning project.

The models to train were selected from the best performers in the ImageNet challenge, and were also used in the related works previously described: ResNet50V2, InceptionResNetV2, NASNetLarge, Xception, InceptionV3 and DenseNet201.

To achieve the best-performing CNN for predicting the building material, the number of floors and the construction period of buildings (for example), the transfer-learning feature extraction and fine-tuning procedures were used.

With all that data gathered and results, the final goal is to develop an exposure model for multi-hazard risk analysis for the parish of Alvalade.

The Figure 4.1 illustrates the images acquisition and training pipeline developed during this work.

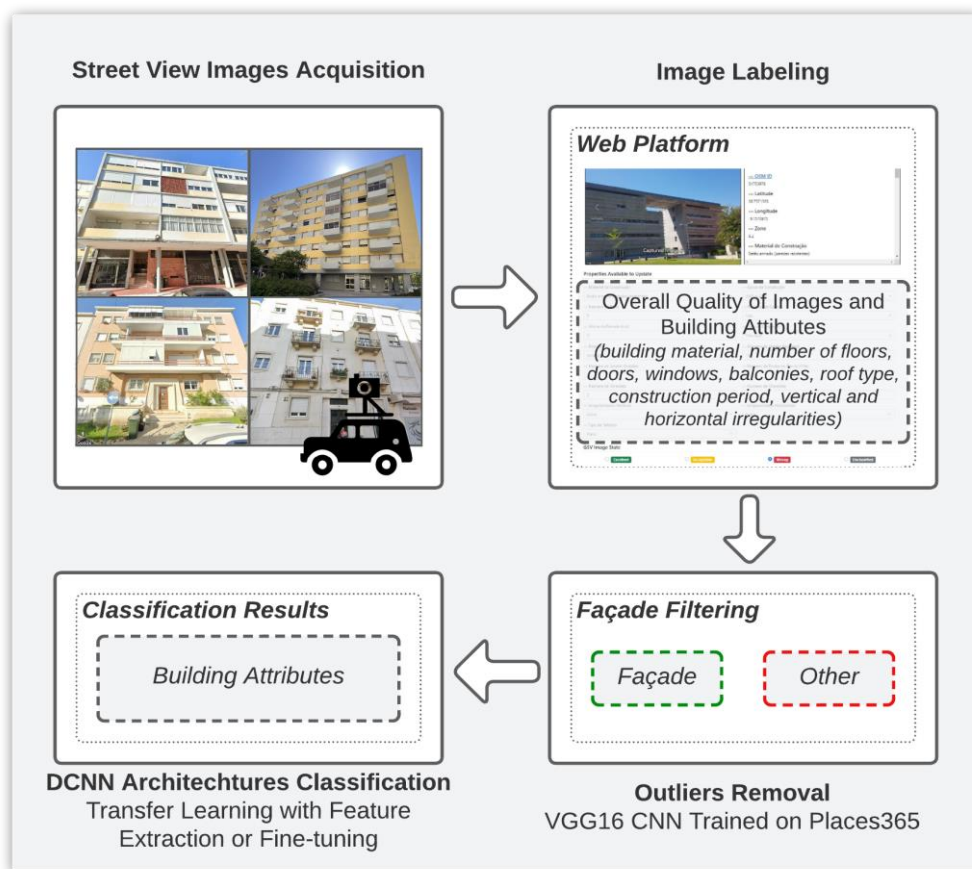


Figure 4.1 - Overall project pipeline.


To perform the complex computational tasks efficiently, the hardware setup used was an Intel Core i9-10900K 10-Core processor, running at a base clock of 3.7GHz with a turbo boost capability up to 5.3GHz. This CPU provided the necessary processing power for data analysis and model training. A Gigabyte RTX 3090 Gaming OC 24GB graphics card was used, which delivered the necessary performance for GPU-intensive machine learning tasks, ensuring rapid model training and inference. To support these components, the system was equipped with 64GB (2x32GB) of Kingston DDR4 3200MHz HyperX Fury Black RAM, offering enough memory for data manipulation and model storage.

4.2 Data Gathering

Initially, using OpenStreetMap and its satellite images, the building footprints were delimited. Then, all the building information was extracted into QGIS, where the available data was used in order to calculate the geographic coordinates of each building to feed into the Google Street View Static API.

The following table gives an example of the result of one of the database entries, that is, a building, its image and other information. Afterwards, all these images went through a manual quality filtering, which will be described in the following sections.

Table 4.1 - GSV Static API response image and related information.

osm_id	97680569	
id	79	
y_Lat	38.7498558	
x_Long	-9.1425503	
ZONE_LABEL	A.5	

In total, 2844 buildings were delimited in the parish of Alvalade, 2670 street views were acquired from the GSV API (one per building) and subsequently 4085 images were captured on the field (up to three images per building). The difference between the total number of buildings and the total number of street views reflects a constrain that is the limited coverage and/or access restrictions. In other words, Google Street View may not have coverage for all geographic locations. Some areas, especially in remote regions, may not have Street View imagery available. In addition, access to certain locations may not exist because these buildings may be on private property.

4.3 Feature Collection

Given the number of features that were selected to be identified, for practical reasons and to make this task easier, the parish of Alvalade was divided into 3 major zones (A, B, and C) with around 900 buildings each. Each zone was then divided into 9 smaller subzones, each with roughly 100 buildings (A.1, A.2, and so on). This division was made to make the field work easier to organize.

To aid this task, an excel file per subzone was automatically prepared, in order to allow the manual addition of the visible attributes, including up to three photographs of the particular building's facade.

id	y_Lat	x_Long	ZONE_LABEL	Building Material	Construction Period	Number of Floors	Presence of Basements	Entry Height (cm)	...
21	38,7475644	-9,1350445	A.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
22	38,747934	-9,135595	A.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
23	38,7465131	-9,1368064	A.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
...						

Figure 4.2 - A.1 subzone excels file example.

With the help of three civil engineering students the following features were identified:

- Building Material;
 - Masonry, Adobe, Reinforced Concrete (granties), Reinforced Concrete (load-bearing walls), Reinforced Concrete (prefabricated), Wood or Metallic.
- Construction Period;
 - Newer than 2010, 2000 to 2010, 1985 to 2000, 1960 to 1985 or Older than 1960.
- Number of Floors;
- Presence of Basements;
 - Yes, No or Not sure.
- Entry Height (cm);

- Occupancy Type;
 - Residential, Residential + Commercial, Commercial, Industrial, Public, Education, Health or Other.
- Position;
 - With adjacent building on one side, With adjacent building on both sides or Isolated.
- Number of Small Windows;
- Number of Large Windows;
- Number of Doors on Ground Floor;
- Number of Balconies;
- Number of Chimneys;
- Vertical Irregularities;
 - Soft-storey, Vertical alteration, Pounding or Other.
- Horizontal Irregularities;
 - Potential for torsion or Other.
- Roof Type;
 - Flat, Inclined with ceramic tile, Inclined with sandwich panel or Other.
- Facade Images;

A total of 15 features were selected for this study. For instance, when compared to state-of-the-art works such as Gonzalez et al. (2020), who used only two features (number of storeys and the material of the LLRS), Ghione et al. (2022), who focused on a single feature (building typologies), and Aravena Pelizari et al. (2021), who used three features (number of storeys, material of the LLRS, and SBST), the approach developed in this work offers a comprehensive and detailed dataset for analysis.

4.4 Analysis and Processing of Collected Data

As expected, several issues emerged concerning the quality and excess of information contained in the images. The building's facade sometimes was not clearly visible in both the street views Google returned and the images that were taken on the field, leading to images with black borders, images in which the building's facade or the building itself is not visible, and images that had obstructions (such as trees, vehicles, traffic signs, or even people) in the way.

Table 4.2 - Images captured on the field where the building facade isn't clearly visible.

		
419369328_1024_0	209356615_319_2	708520232_1735_2

4.4.1 Outliers Removal Algorithm

To address problems related to non-visible facades or the presence of obstructions the approach of Aravena Pelizari et al. (2021) was used. The VGG16 DCNN (Simonyan & Zisserman, 2015), trained on the Places365 dataset (the latest subset of Places2 database with more than 1.8 million photos from 365 scene classes), was used to determine whether any of the images were actually of a building facade. To do this, a group S of 24 predefined building classes was defined: [*'apartment_building/outdoor'*, *'beach_house'*, *'building_facade'*, *'chalet'*, *'church/outdoor'*, *'cottage'*, *'courthouse'*, *'embassy'*, *'fire_station'*, *'hangar/outdoor'*, *'hospital'*, *'hotel/outdoor'*, *'house'*, *'hunting_lodge/outdoor'*, *'mansion'*, *'manufactured_home'*, *'motel'*, *'office_building'*, *'palace'*, *'schoolhouse'*, *'shed'*, *'skyscraper'*, *'synagogue/outdoor'*, *'tower'*].

Then, the set of street view images was fed into the DCNN where the following rule was applied on the output (Aravena Pelizari et al., 2021):

$$L_i = 'Facade' \text{ if } |C_i \cap S| \geq 2, \text{ else 'Other'},$$

with L_i representing the label of image i , and C_i the 4 classes assigned as most likely.

If the intersection of the top four model predictions with the set S is greater than or equal to two, the image is considered to contain a building (that is, there's a facade). For example, if two of the top four predictions are "hospital" and "house", this image is automatically labeled as a "Facade" because these two classes are in set S . If an image is classified as "Other", it is discarded.

4.4.2 Black Edge Trimming

Regarding the black borders, since different smartphones were used in the field to capture the photographs of each building's facade, the existence of images with black borders has proven to be a relevant factor and worthy of being fixed. This was done by using the Python library OpenCV.

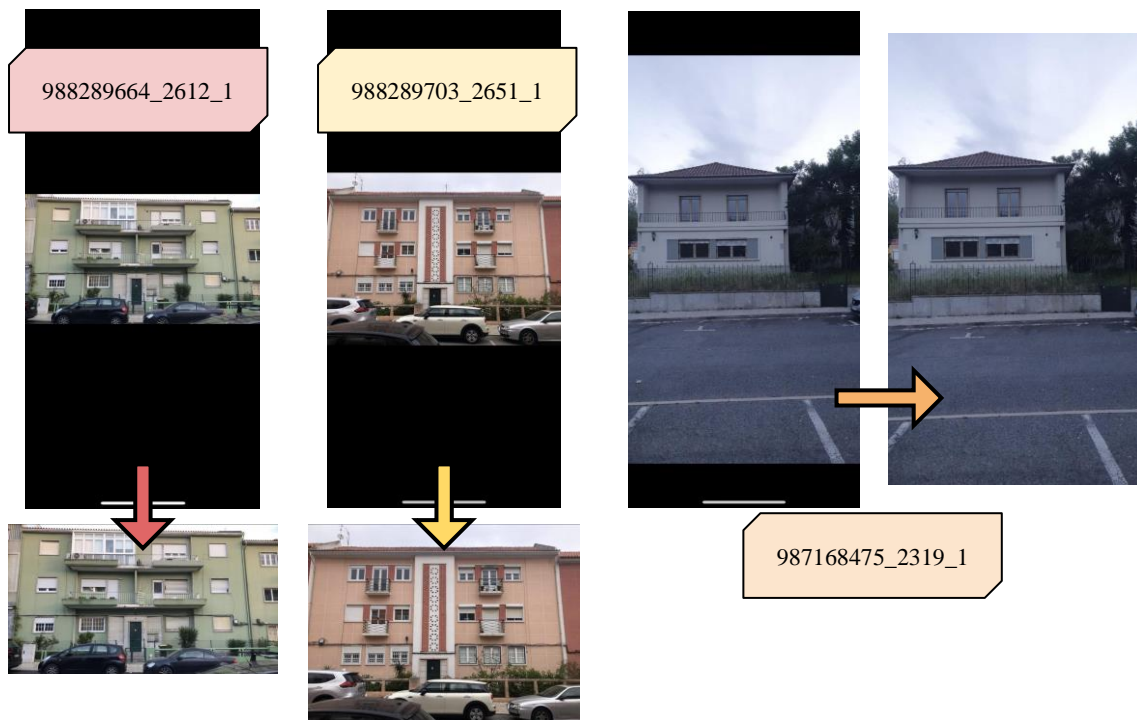


Figure 4.3 - Before and after removing black borders.

First, the image is scanned and converted to grayscale before being thresholded to create a binary image that distinguishes between the object of interest and the background. It is then morphologically opened, which removes small objects while smoothing the boundaries of the larger object. The biggest exterior contour in the processed image is then detected using OpenCV's *findContours* function. Its bounding box coordinates will then be acquired using *boundingRect* function. At last, the original image is cropped to isolate the largest object using these resulting bounding box coordinates, and the resulting image is saved back to the original file. If the bounding box coordinates begin at the top left corner of the image, it has already been cropped, and the algorithm skips it.

This process wasn't necessary on GSV images because there were no black edges, in other words, either there is an image for a given coordinate pair or not at all.

```
1 for img_path in imgs:
2     img = cv2.imread(img_path)
3
4     # load image as grayscale
5     img_gray = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
6     thresh = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY)[1]
7     kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (15, 15))
8     morph = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
9
10    # get bounding box coordinates from largest external contour
11    contours = cv2.findContours(morph, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
12    contours = contours[0] if len(contours) == 2 else contours[1]
13    big_contour = max(contours, key=cv2.contourArea)
14    x, y, w, h = cv2.boundingRect(big_contour)
15    if x == 0 and y == 0:
16        print("NOT CROPPING")
17        continue
18    else:
19        print("CROPPING")
20
21    # crop the image at the bounds adding back the two blackened rows at the bottom
22    crop = img[y:y + h - 1, x:x + w]
23
24    # save result
25    cv2.imwrite(img_path, crop)
26
```

Figure 4.4 - OpenCV developed code to remove black borders from images.

4.4.3 Data Annotation (Web Platform)

For both the street views returned by Google and those collected in the field, the facade of the building may not be clearly visible since it may be obstructed by trees, vehicles, traffic signs, or even people, i.e., likely to contain anomalies.

In order to obtain data in the best possible quality, a web platform (backend and frontend) hosted on AWS (Amazon Web Services) was also developed using Python combined with the Flask framework. Using the platform an expert could assess the quality of the data collected for both the images and the properties of each building using the following scale: “Excellent”, “Acceptable”, “Wrong”, and “Unclassified”. As shown in the following figures (Figure 4.5 and Figure 4.6), it was also possible to add or correct the properties that were previously collected in the field and also add comments when necessary (for example, “building is not visible”).

Table 4.3 - Option for expert assessment.

	Images - State	Properties - State
Excellent	Images where the building’s facade is clearly visible.	Properties are correct.
Acceptable	Images where the building’s facade is visible but obstructed by something.	Properties are correct but there may be some missing.
Wrong	Images without visible facade or building.	Properties do not match at all what is seen in the image.
Unclassified	Images yet to be reviewed (default).	Properties to be reviewed (default).

The platform allowed visualization and editing of the database contents, searching with filters, statistics to monitor the development and status of the analysis, and a page dedicated to the classification of the properties and images of each building. The main goal of this manual process was to increase the quality of the data to be used for the training. The experts were specialists in Civil Engineering and had the appropriate training to use the platform.

The following images show some of the platform's interfaces.

AI4DRR Zones (Home) Users Stats Logout adminUFPAI4DRR

Zones

All Only Excellent Only Classifiable

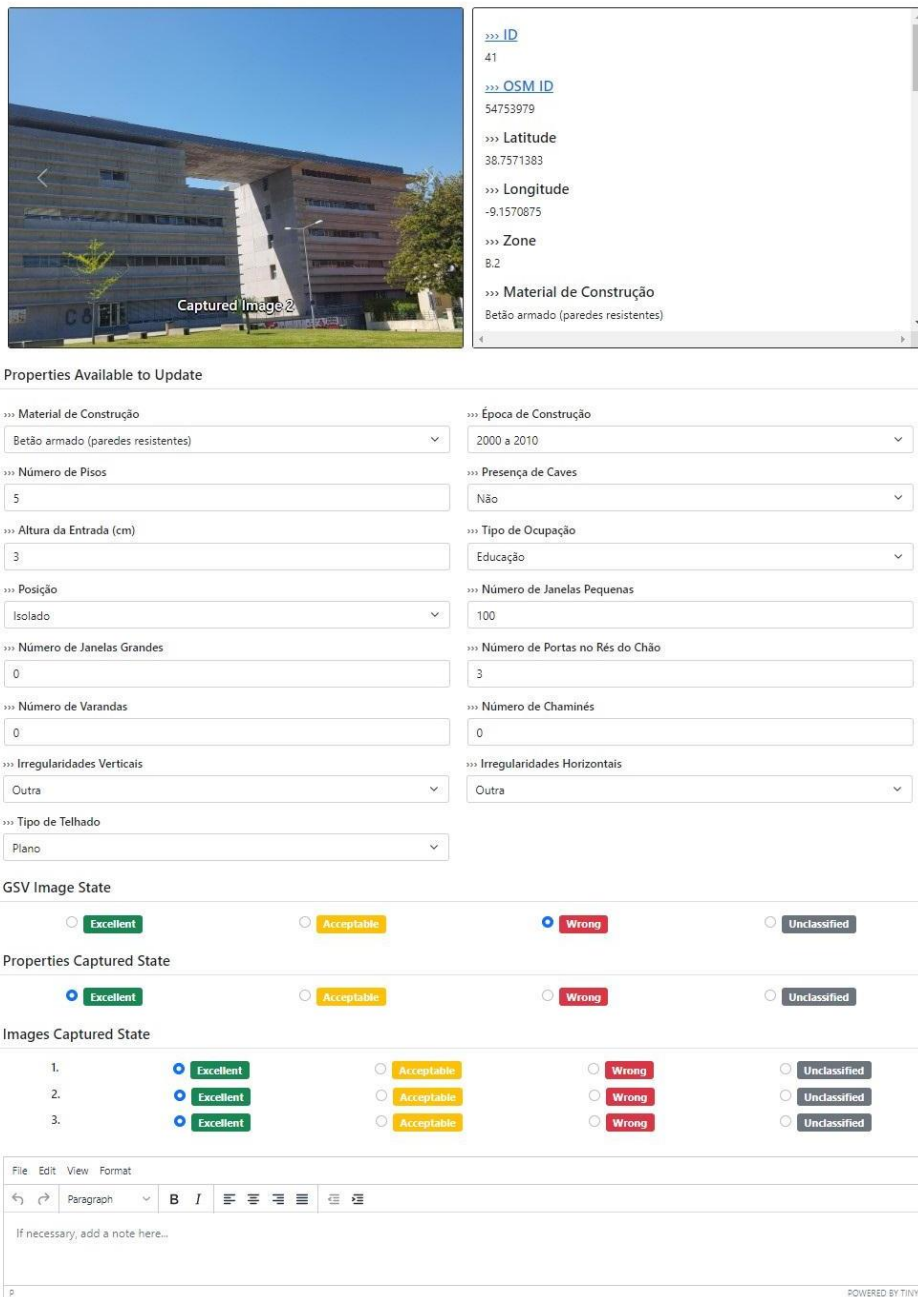
Show entries Search:

ID ↑	OSM ID ↑	Building ID ↑	Zone ↑	Image Path	GSV Img State	Properties State	Captured Img. 1 State	Captured Img. 2 State	Captured Img. 3 State	Command
1384	54753979	41	B.2		Wrong	Excellent	Excellent	Excellent	Excellent	Classify
2681	54753984	42	B.2		Wrong	Wrong	Acceptable	Acceptable	Acceptable	Classify
1385	54753985	43	B.2		Wrong	Wrong	Wrong	Wrong	Wrong	Classify
1386	54753986	44	B.2		Acceptable	Wrong	Wrong	Wrong	Wrong	Classify
2682	54754719	45	B.3		Wrong	Excellent	Excellent	Wrong	Wrong	Classify
1387	54754720	46	B.2		Wrong	Unclassified				Classify
2195	97678457	47	A.7		Wrong	Excellent	Acceptable	Acceptable	Acceptable	Classify
2196	97678465	48	A.7		Acceptable	Acceptable	Acceptable	Acceptable	Acceptable	Classify
2197	97678471	49	B.6		Acceptable	Acceptable	Excellent	Excellent	Wrong	Classify
2198	97678476	50	A.7		Wrong	Acceptable	Acceptable	Acceptable	Acceptable	Classify

Showing 41 to 50 of 2,844 entries
[Previous](#)
[1](#)
[...](#)
[4](#)
[5](#)
[6](#)
[...](#)
[285](#)
[Next](#)

Figure 4.5 - Interface for viewing the contents of the database.

StreetView [1384]



Captured Image

Properties Available to Update

Material de Construção Betão armado (paredes resistentes)	Época de Construção 2000 a 2010
Número de Pisos 5	Presença de Caves Não
Altura da Entrada (cm) 3	Tipo de Ocupação Educação
Posição Isolado	Número de Janelas Pequenas 100
Número de Janelas Grandes 0	Número de Portas no Rés do Chão 3
Número de Varandas 0	Número de Chaminés 0
Irregularidades Verticais Outra	Irregularidades Horizontais Outra
Tipo de Telhado Plano	

GSV Image State

Excellent Acceptable Wrong Unclassified

Properties Captured State

Excellent Acceptable Wrong Unclassified

Images Captured State

1.	<input checked="" type="radio"/> Excellent	<input type="radio"/> Acceptable	<input type="radio"/> Wrong	<input type="radio"/> Unclassified
2.	<input checked="" type="radio"/> Excellent	<input type="radio"/> Acceptable	<input type="radio"/> Wrong	<input type="radio"/> Unclassified
3.	<input checked="" type="radio"/> Excellent	<input type="radio"/> Acceptable	<input type="radio"/> Wrong	<input type="radio"/> Unclassified

File Edit View Format

Paragraph B I [List Icons]

If necessary, add a note here...

POWERED BY TINY

Figure 4.6 - Image and classification interface for a building.

4.5 Creation of the Dataset

4.5.1 Cleaning the Dataset

Considering the classifications after all the preprocessing tasks described previously and the manual annotation using the developed web platform, each image resulted in an entry in the dataset, i.e., a building with N associated images will have N entries.

However, if any of the following occurs, the entries will be ignored:

- The GSV image is “Unclassified” or “Wrong”;
- The captured image is “Unclassified” or “Wrong”;
- There are comments/notes;
- The features to be used for training (e.g., construction material and number of floors) are not filled.
- Finally, the outlier’s removal process (VGG16 + Places365) was also used.

4.5.2 Classes Configuration

As not all attributes could be used to define the classes, and to identify what would be the best combination of attributes for the future creation of the exposure model the following class configurations were created.

A configuration reflects the classes resulting from the aggregation of two or more features previously chosen, as can be seen in the following sections, which shows the letter assigned to the respective configuration and its set of features.

4.5.2.1 Configuration A

Buildings were grouped by **building material** and **number of floors** (in ranges). For this and other configurations where it was chosen to keep the building material it was necessary to remove buildings of type ‘Metallic’, ‘Adobe’ and ‘Wood’ since they had very few images associated with (9, 1 and 0, respectively) compared to the other options (‘Concrete’ and ‘Masonry’).

Table 4.4 - Configuration A classes details.

Classes	Attributes				
<ul style="list-style-type: none"> • Masonry 1-3 • Masonry 4+ • Concrete 1-3 • Concrete 4-6 • Concrete 7+ 	<div style="display: flex; align-items: center; justify-content: center;"> <div style="border: 1px solid blue; border-radius: 15px; padding: 5px; background-color: #4a86e8; color: white; text-align: center; width: 150px;"> Building Material No "Metallic", "Adobe" and "Wood" </div> <div style="margin: 0 10px; font-size: 2em;">+</div> <div style="border: 1px solid blue; border-radius: 15px; padding: 5px; background-color: #4a86e8; color: white; text-align: center; width: 150px;"> Num. Floors Bigger Ranges </div> </div>				
Total Dataset Entries	After “Cleaning” (VGG16 + Places365)				
5252	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: left; padding: 5px;">Num. “Facade”s</td> <td style="text-align: right; padding: 5px;">4239 (80.71%)</td> </tr> <tr> <td style="text-align: left; padding: 5px;">Num. “Other”s</td> <td style="text-align: right; padding: 5px;">1013 (19.29%)</td> </tr> </table>	Num. “Facade”s	4239 (80.71%)	Num. “Other”s	1013 (19.29%)
Num. “Facade”s	4239 (80.71%)				
Num. “Other”s	1013 (19.29%)				

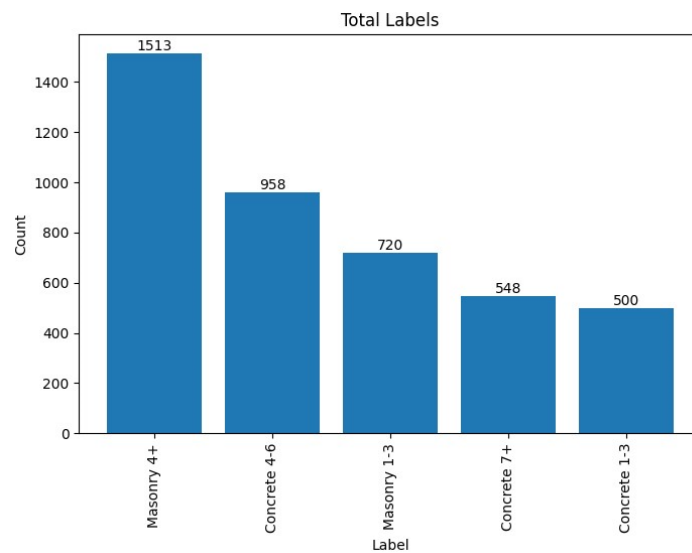



Figure 4.7 - Configuration A classes frequency.

4.5.2.2 Configuration B

Buildings were grouped by **building material** and **number of floors**, using none or smaller intervals compared to configuration A.

Table 4.5 - Configuration B classes details.

Classes	Attributes				
<ul style="list-style-type: none"> ● Concrete 1 ● Concrete 2 ● Concrete 3 ● Concrete 4 ● Concrete 5 ● Concrete 6 ● Concrete 7+ ● Masonry 1 ● Masonry 2 ● Masonry 3 ● Masonry 4 ● Masonry 5+ 					
Total Dataset Entries	After “Cleaning” (VGG16 + Places365)				
5252	<table border="0" style="width: 100%;"> <tr> <td style="text-align: center;">Num. “Facade”s</td> <td style="text-align: right;">4239 (80.71%)</td> </tr> <tr> <td style="text-align: center;">Num. “Other”s</td> <td style="text-align: right;">1013 (19.29%)</td> </tr> </table>	Num. “Facade”s	4239 (80.71%)	Num. “Other”s	1013 (19.29%)
Num. “Facade”s	4239 (80.71%)				
Num. “Other”s	1013 (19.29%)				

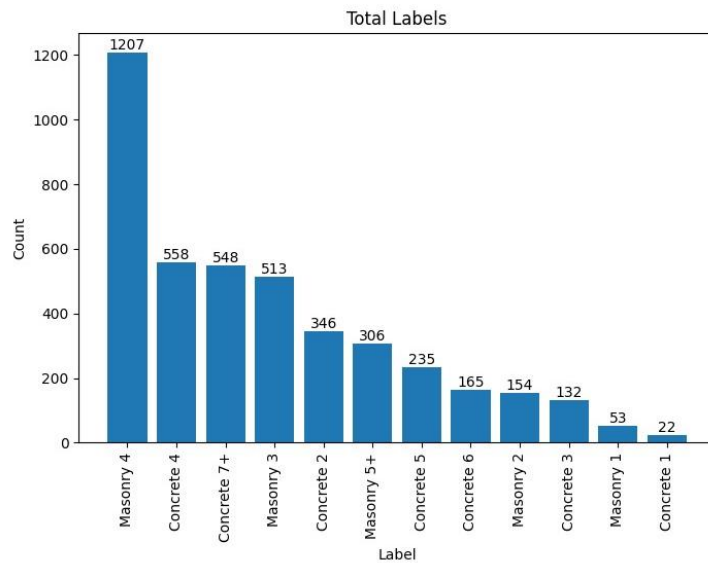


Figure 4.8 - Configuration B classes frequency.

4.5.2.3 Configuration C

Buildings were grouped according to their **building material** and **construction period**. Due to the removal of buildings with incomplete construction period data, for example ids 1831 (2 entries), 1840 (2 entries), and 2224 (3 entries), the dataset’s overall entries were reduced to 5247 from earlier configurations. However, because both construction material and period characteristics were now available, buildings with ids 1120 and 2072, which previously lacked the number of floors, were included in this arrangement. Additionally, one entry containing information on a facade was also removed during the cleaning process. In particular, the building with the id 2072 was first added to the dataset but then removed following an anomaly removal procedure.

Table 4.6 - Configuration C classes details.

Classes	Attributes				
<ul style="list-style-type: none"> ● Concrete 1960>=<1985 ● Concrete <1960 ● Concrete >1985 ● Masonry 1960>=<1985 ● Masonry <1960 ● Masonry >1985 	<div style="display: flex; align-items: center; justify-content: center;"> <div style="border: 1px solid blue; border-radius: 15px; padding: 5px; background-color: #4a86e8; color: white; text-align: center; width: 150px;"> Building Material No "Metallic", "Adobe" and "Wood" </div> <div style="margin: 0 10px; font-size: 2em;">+</div> <div style="border: 1px solid blue; border-radius: 15px; padding: 5px; background-color: #4a86e8; color: white; text-align: center; width: 150px;"> Construction Period </div> </div>				
Total Dataset Entries	After “Cleaning” (VGG16 + Places365)				
5247	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 50%;">Num. “Facade”s</td> <td style="text-align: right;">4233 (80.67%)</td> </tr> <tr> <td style="text-align: center;">Num. “Other”s</td> <td style="text-align: right;">1014 (19.33%)</td> </tr> </table>	Num. “Facade”s	4233 (80.67%)	Num. “Other”s	1014 (19.33%)
Num. “Facade”s	4233 (80.67%)				
Num. “Other”s	1014 (19.33%)				

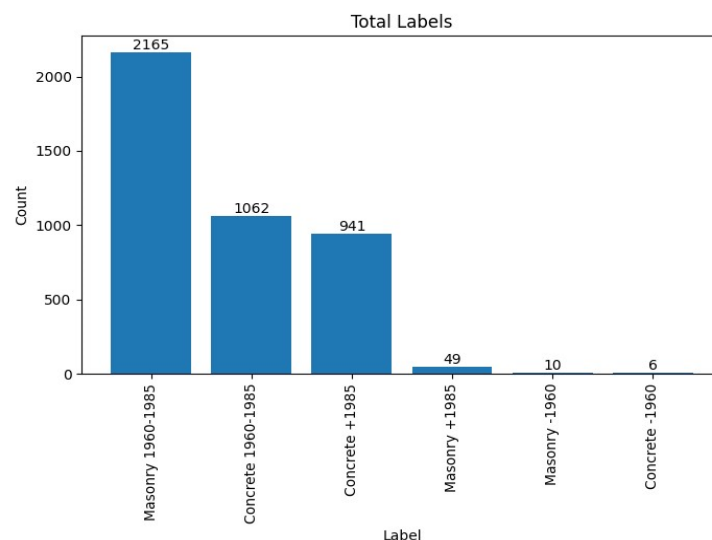


Figure 4.9 - Configuration C classes frequency.

4.6 Splitting

Regardless of the class configuration chosen, for the results in the following sections, the entire procedure, including the removal of the black borders, the web platform states, and the outlier removal step, was always used. The dataset was then divided into three subsets: training, test and validation. Randomly, for the training set, 80% of the dataset was assigned, with the remaining 20% assigned to the test set. For the validation set, 20% of the data from the training set was assigned (nested-splitting), resulting in 16% of the total dataset. In addition, the results were also tested for a training set with 70% of the dataset and the remaining 30% assigned to the test set. Again, for the validation set, 20% of the data from the training set was assigned.

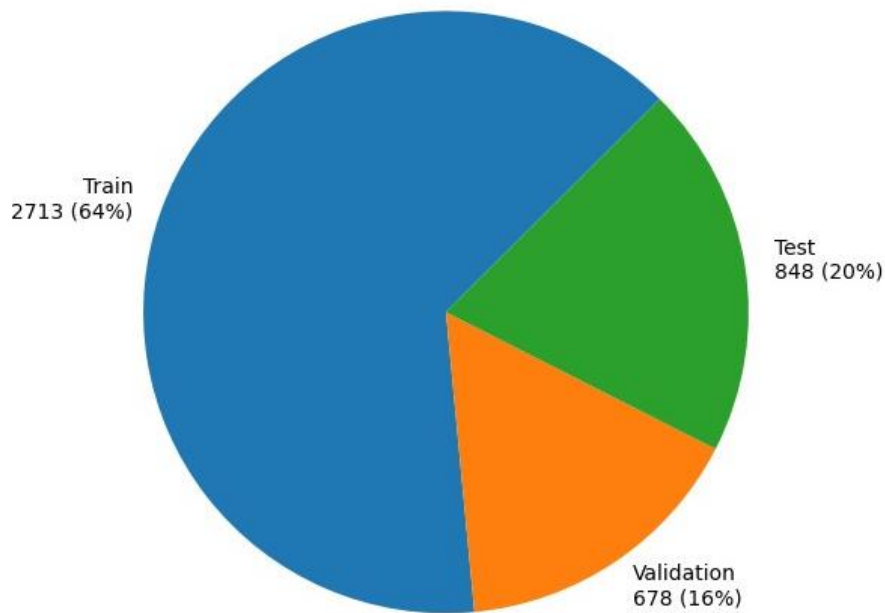


Figure 4.10 - Dataset splitting for configuration A.

4.7 Image Augmentation

Image augmentation is a well-established machine learning strategy for improving the accuracy of models that rely on image input by producing new versions of existing data to artificially expand the size of a dataset. A *data_augmentation* model was created through TensorFlow Sequential, where the output of each layer is fed as input to the next layer in sequence. In this case, each layer represents an image augmentation technique that modifies the original input. The following augmentation layers were used on the training subset:

- **RandomFlip:** This layer randomly flips the image horizontally. By setting the seed parameter, it ensures that the same transformation is applied to all images in a given batch during training, which helps with reproducibility.
- **RandomRotation:** This layer randomly rotates the image by a degree between -0.1 and 0.1, with *fill_mode* set to 'nearest'. The fill mode parameter specifies how empty pixels should be filled after rotation. The 'nearest' option uses the pixel values that are closest to the empty space.
- **RandomZoom:** With fill mode set to 'nearest', this layer randomly zooms in or out of the image by a factor between -0.1 and 0.1 in both the horizontal and vertical axes. This can aid in simulating photographs shot from various distances or with various camera settings.
- **RandomTranslation:** With fill mode set to 'nearest', this layer randomly translates the image horizontally and vertically by a factor between -0.1 and 0.1. This can also aid in simulating photographs captured from various perspectives or positions.
- **RandomBrightness:** This layer randomly adjusts the image's brightness by a factor ranging from -0.1 to 0.1. This can help in simulating different lighting conditions.
- **RandomContrast:** This layer randomly modifies the image's contrast by a factor ranging from -0.1 to 0.1. This can aid in simulating images captured with various camera settings or under diverse lighting situations.

By applying these image augmentation techniques to the training data, the model can be made to be more robust to variations in the input data, which can lead to better performance on new, unseen data. Therefore, the size of the dataset was efficiently expanded which helped to prevent overfitting and improve the performance of the machine learning models.

4.8 Class Weights

Overfitting occurs when a model becomes overly complicated and begins memorizing the training data rather than learning general patterns. Because it has more samples to learn from some classes when dealing with unbalanced datasets, such as this one, the method may be more prone to overfitting on the majority class. In general, applying class weights is a practical method for handling unbalanced datasets and can improve the performance of machine learning algorithms (Fernández et al., 2018). The algorithm can develop a more balanced representation of the data by giving each class the proper weights, which can improve generalization and prevent overfitting. In other words, when one class has much more samples than another in a dataset, the algorithm may be biased towards the majority class and perform poorly on the minority class. This method can provide more attention to the minority class during training by assigning different weights to each class, resulting in a more balanced and accurate model.

5. MACHINE LEARNING ARCHITECTURES

5.1 Selected Models

Based on the articles referenced in the literature review section, six Convolutional Neural Networks (CNNs) that have been shown to be most successful on ImageNet, a large-scale classification, were selected: InceptionResNetV2, ResNet50V2, NasNetLarge, InceptionV3, Xception and DenseNet201.

These architectures and their respective weights pre-trained on ImageNet allowed to use a technique called Transfer Learning with Fine Tuning, in which the earlier layers of the network are preserved, maintaining the low visual level of features they learned on the ImageNet dataset, while the later layers are re-trained to learn features specific to a particular domain, i.e., our problem. Fine tuning techniques has been shown to improve generalization performance in many classification tasks.

Table 5.1 - Input sizes of pre-trained models.

Models	Image Sizes (Input)
DenseNet201	224x224
InceptionResNetV2	299x299
InceptionV3	299x299
NasNetLarge	331x331
ResNet50V2	224x224
Xception	299x299

Apart from the characteristics and differences of each architecture, it should be noted that each one receives the data in its own way and that each image has to be resized both in the training and classification processes.

5.1.1 DenseNet201

Traditional CNNs sequentially connect one layer to the next, which can result in information loss due to the vanishing gradient problem. It occurs during the training process when the gradients of the loss function with respect to the network's parameters become extremely small as they are backpropagated through the layers of the network. When gradients become increasingly small, it becomes challenging to update the weights of the earlier layers effectively, and these layers may not learn meaningful features or contribute much to the overall learning process. DenseNet201 (Huang et al., 2017), on the other hand, is a DCNN architecture that overcomes this issue by adding dense connections between layers, where the number (201) in the model's name denotes the number of neural network layers.

The DenseNet architecture is made up of several dense blocks, each with a set of convolutional layers. Each dense block's output is connected as input to the next dense block, resulting in a dense connection among the layers. This enables for more efficient exchange of data between layers and minimizes the number of training parameters.

The basic idea behind dense connections is to concatenate the feature maps of all previous layers to the current layer rather than adding them as in residual networks (such as ResNet). This gives each layer access to the feature maps of all preceding layers, which improves feature reuse and, as a result, overall network performance.

This design is made up of four dense blocks, each with a different number of convolutional layers. Each dense block is made up of a series of convolutional layers that have batch normalization and ReLU (Rectified Linear Unit) activation function, followed by a transition layer. The transition layer reduces the feature maps spatial dimensions while preserving the number of feature maps. This helps reduce the network's computing cost.

DenseNet201's last layer is a global average pooling layer, followed by a fully connected layer with softmax activation. For each input image, the global average pooling layer computes the average of the feature mappings across spatial dimensions, resulting in a fixed-length feature vector that is mapped to a probability distribution over the classes by the fully connected layer.

DenseNet201 outperforms other state of the art architectures on benchmark datasets like CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009), and ImageNet while utilizing fewer parameters. Its dense connections allow for faster information flow, which leads to better feature reuse and, as a result, higher accuracy.

5.1.2 InceptionResNetV2

The InceptionResNetV2 architecture (Szegedy et al., 2017) combines the Inception architecture with residual connections (ResNet), which are shortcut connections that allow data to bypass neural network layers.

It is built on the concept of “Inception modules”, which are layers that perform convolution, pooling, and other operations in parallel, allowing the network to learn information at different scales. Additionally, the architecture also incorporates “ResNet modules” that are based on the residual connection concept. These modules are used to connect the Inception modules and allow the network to learn more sophisticated and abstract characteristics in order to increase gradient flow and help prevent the vanishing gradient problem.

The InceptionResNetV2 also features a number of other performance-enhancing techniques, including batch normalization, dropout, and weight decay, in which the model is trained using the cross-entropy loss function and optimized using stochastic gradient descent with momentum.

This architecture is an enhancement to a previous version that includes a greater number of Inception-ResNet modules. It incorporates broader layers in some areas of the network, increasing the network’s ability to learn more complex features and representations. Furthermore, the Inception blocks utilized in InceptionResNetV2 include batch normalization before the activation function, which can increase the training process’s stability.

InceptionResNetV2 architecture, as a whole, is a powerful CNN that can achieve cutting-edge performance on a range of computer vision tasks, including picture classification and object recognition. For the network to acquire features at various dimensions and levels of abstraction, which is necessary to achieve high accuracy on challenging visual tasks, both Inception and ResNet modules must be used.

5.1.3 InceptionV3

InceptionV3 (Szegedy et al., 2016) is a CNN architecture built for image classification and object recognition tasks. It is based on the concept of “Inception modules”, which are designed to collect spatial characteristics of images at various sizes efficiently.

This architecture includes multiple layers of convolutional and pooling operations, as well as inception modules with multiple parallel convolutional layers of varying filter sizes (1x1, 3x3, and 5x5, for example), as well as 1x1 convolutions to reduce the number of channels and prevent overfitting.

Furthermore, “auxiliary classifiers” at intermediary layers allow the network to learn more robust features and assist prevent the vanishing gradient problem during training.

Overall, the InceptionV3 architecture is one of the best among the competition on various large-scale picture classification benchmarks (such as ImageNet), while being computationally efficient and scalable. It is commonly used in computer vision applications such as image recognition, object detection, and segmentation.

5.1.4 NasNetLarge

NASNet, or Neural Architecture Search Network (Zoph et al., 2018), is a deep learning architecture designed for image recognition applications, with the main goal of automating the creation of neural network architectures. Creating a neural network architecture is typically a time-consuming and complex procedure that requires a great deal of trial and error. NASNet, on the other hand, uses a process known as reinforcement learning to automatically seek for the best architecture for a given task.

Reinforcement learning is a type of machine learning in which an agent learns to make decisions by interacting with its surroundings. The neural network architecture is the agent in the case of NASNet, while the environment is the image recognition problem. The agent learns from the environment by getting feedback in the form of a reward signal, which indicates how well the architecture is performing on the job.

In the NASNet architectural search process, a large number of candidate architectures are trained on a smaller amount of data. After that, the best architectures are chosen and trained on the entire dataset. This technique is repeated until the required level of performance has been achieved.

NASNet has demonstrated the usefulness of the automated architectural search strategy by achieving state-of-the-art performance on a variety of image recognition benchmarks. Furthermore, NASNet has been proved to be transferable, which means that designs discovered for one work can be employed for other similar activities, decreasing the requirement for manual architecture design even further.

Overall, it is a big step forward in the field of deep learning, giving an automated and scalable technique to creating neural network designs for image identification tasks.

But why use “Large” architecture rather than “Mobile”? In summary, NASNetLarge is a larger and more complex architecture intended for high-accuracy image recognition applications, whereas NASNetMobile is a smaller and more efficient architecture intended for mobile and embedded devices with limited processing resources. Both designs were created using automated architectural search and have performed well on a range of workloads.

5.1.5 ResNet50V2

ResNet50V2 (He et al., 2016b) is a deep learning architecture designed to improve the performance of CNNs for use in computer vision applications such as image recognition and object detection. It is a 50-layer ResNet based design, which pioneered the concept of residual blocks in deep neural networks to tackle the disappearing gradient problem.

The introduction of residual connections, which enables the training of very deep neural networks with many layers, was ResNet50V2's key advance. The remaining connections allow data to move from one layer to the next without passing through multiple intermediate layers. By addressing the vanishing gradient problem, this strategy makes it easier to train deeper networks.

The residual connections are introduced between convolutional layer blocks, each of which has numerous convolutional layers with a growing number of filters being the last layer of the network, a fully connected layer that outputs a probability distribution across the various classifications. It was trained on large datasets such as ImageNet, which contains millions of annotated pictures, and delivers world-class performance on a range of benchmark datasets.

Overall, ResNet50V2 is a sophisticated machine learning architecture that has proven to be cutting-edge in a range of computer vision applications.

5.1.6 Xception

Xception (Chollet, 2017) is a deep learning architecture based on the concept of depth wise separable convolutions, a sort of convolutional operation that can be utilized to minimize the computational complexity of CNNs while maintaining performance.

Each convolutional layer in a typical CNN applies a set of filters to the full input volume, specifically, the entire three-dimensional input data, including all the pixels and channels, in order to extract features from it. This can be computationally costly, particularly if the input volume is very large. Depthwise separable convolutions address this issue by splitting the typical convolution into two operations: a depthwise convolution and a pointwise convolution.

Depthwise convolution applies a single filter to each input channel individually, resulting in a set of output feature maps with the same number of channels as the input. The pointwise convolution then applies a series of 1x1 filters to the depthwise convolution output, resulting in a set of output feature maps with differing channel counts than the input.

Depthwise separable convolutions have the advantage of requiring fewer parameters and computations than classic convolutions while still capturing crucial spatial and channel-wise correlations in the data. As a result, they are ideal for mobile and embedded systems with low processing resources.

Xception uses the concept of depth wise separable convolutions to construct a DCNN architecture capable of performing state-of-the-art computer vision applications. It replaces classic CNNs regular convolutional layers with a set of depthwise separable convolutions interleaved with shortcut connections to optimize information flow and gradient propagation.

6. TRAINING

Initially, for the training of each model architecture, a set of metrics and parameters were set, such as a threshold of **50 epochs** and a **batch size** of **32**.

6.1 Transfer Learning

Transfer Learning (Yang et al., 2020) is a machine learning technique that applies knowledge learned from a pre-trained model on bigger dataset to another similar task or dataset. The underlying notion behind it is that knowledge gained from solving one problem can be applied to a different but related problem. Transfer learning, by employing pre-trained models as a starting point, can speed up the training process, need less labeled data for the target task, and frequently result in greater generalization and performance. There are two general techniques to transfer learning: Feature Extraction and Fine-Tuning.

The following image shows the developed code snippet with the flexibility to implement these techniques on a model based on the specified “*FINE_TUNE*” flag and the chosen pre-trained model architecture. It begins by defining an input layer with a specific shape, representing the size of input images with three color channels: red, green and blue (RGB). The script then calls a function called “*get_pretrained_model*” to obtain the desired pre-trained neural network model. Next, if the flag “*FINE_TUNE*” is set to “True”, the script proceeds to fine-tune the model. The choice of layers to unfreeze from is determined by a dictionary, which maps the pre-trained model with the corresponding layer. If the flag is set to “False”, then all the layers from the pre-trained model are frozen – default transfer learning with feature extraction. Lastly, regardless the flag state, the code defines additional layers for classification tasks on top of the model. These layers include flattening the output, adding a dense layer with ReLU activation, applying dropout for regularization, and finally, a dense output layer with a softmax activation function for classification.

```

1 inputs = tf.keras.Input(shape=self.img_size + (3,), name='input')
2 pretrained_model, _ = self.get_pretrained_model(input_tensor=inputs)
3
4 if self.FINE_TUNE: # Transfer-Learning + Fine-Tuning
5     unfreeze_from = {
6         'resnet_v2_50': 'conv5_block1_preact_bn',
7         'inception_resnet_v2': 'conv2d_160',
8         'nasnet_large': 'normal_conv_1_17',
9         'xception': 'block10_sepconv1_act',
10        'inception_v3': 'conv2d_80',
11        'densenet_201': 'conv5_block1_0_bn',
12    }
13
14    pretrained_model.trainable = True
15
16    fine_tune_layer_name = unfreeze_from.get(self.model_name, None)
17    try:
18        fine_tune_layer = pretrained_model.get_layer(fine_tune_layer_name)
19        fine_tune_at = pretrained_model.layers.index(fine_tune_layer)
20        # Freeze All the Layers Before the `fine_tune_at` Layer
21        for layer in pretrained_model.layers[:fine_tune_at]:
22            layer.trainable = False
23
24    except ValueError:
25        print(f"\n[ERROR]\n-> Tuning Layer Not Found for Model '{self.model_name}'")
26        pretrained_model.trainable = False
27
28    for layer in pretrained_model.layers:
29        if isinstance(layer, tf.keras.layers.BatchNormalization):
30            layer.trainable = False # Fix BatchNormalization Layers
31
32 else: # Transfer-Learning + Feature Extraction
33     pretrained_model.trainable = False
34
35 new_model_name = f"MY_{self.model_name}_FT" if self.FINE_TUNE else f"MY_{self.model_name}"
36 # Add Remaining Classification Layers
37 x = Flatten(name='flatten')(pretrained_model.output)
38 x = Dense(1024, activation='relu')(x)
39 x = Dropout(0.2)(x)
40 outputs = Dense(self.num_classes, activation='softmax', name='output')(x)
41 my_model = Model(inputs=inputs, outputs=outputs, name=new_model_name)
42

```

Figure 6.1 - Transfer Learning Feature Extraction VS Fine-Tuning procedures.

6.1.1 Feature Extraction

In this approach, the pre-trained model is employed as a feature extractor. Its weights are frozen, and only the weights of the newly added layers, also known as “classifier layers”, are trained on the target task. The pre-trained model extracts significant features from the input data, which are then fed into the newly added layers to perform the final classification task.

6.1.2 Fine-Tuning

Fine-tuning goes beyond feature extraction by allowing the pre-trained model's weights to be adjusted while training on the target task. During this process, a few of the top layers of a frozen base model are unfrozen, allowing for joint training of the newly added classifier layers as well as the base model's final layers. Using this method, we can “fine-tune” the higher-order feature representations in the base model, making them more relevant to the job at hand. The model can adapt and specialize its acquired features to the target job by updating these unfrozen layers, exploiting the pre-trained knowledge to improve performance and generalization.

When using the Transfer-Learning with Fine-Tuning technique to train the models, a “*unfreeze_from*” dictionary was used to specify which layer to start fine-tuning from for a specific pre-trained model; in other words, it maps model names to layer names, indicating the last layer that will remain frozen during training. The pre-trained model is then made trainable, and the dictionary of unfreezing layers is searched to see if a fine-tuning layer exists for the particular model architecture. If the layer is found, all layers before it are frozen, and only the layers following the fine-tuning layer are trained. If it cannot be found, an error is displayed, and all layers are frozen (working as a default Transfer-Learning with Feature Extraction). Finally, following TensorFlow's documentation⁵, a loop is executed over all the layers in the pre-trained model, and if a layer is an instance of “BatchNormalization”, it is frozen (setting trainable as “False”) to avoid training issues:

→ “(...) When you unfreeze a model that contains BatchNormalization layers in order to do fine-tuning, you should keep the BatchNormalization layers in inference mode by passing *training = False* when calling the base model. Otherwise, the updates applied to the non-trainable weights will destroy what the model has learned.”

5

https://www.tensorflow.org/tutorials/images/transfer_learning#important_note_about_batchnormalization_layers

6.2 Hyperparameter Tuning

As can be seen in the Figure 6.2, using TensorFlow’s Keras API, the neural network model started with a “Input” layer defined with the shape of the input data specified as a tuple of the size of the image (pre-trained model input dimensions) and three-color channels (RGB).

```
1 inputs = tf.keras.Input(shape=self.img_size + (3,), name='input')
2 pretrained_model, _ = self.get_pretrained_model(input_tensor=inputs)
3
4 (...)
5
6 new_model_name = f"MY_{self.model_name}_FT" if self.FINE_TUNE else f"MY_{self.model_name}"
7 x = Flatten(name='flatten')(pretrained_model.output)
8 x = Dense(1024, activation='relu')(x)
9 x = Dropout(0.2)(x)
10 outputs = Dense(self.num_classes, activation='softmax', name='output')(x)
11 my_model = Model(inputs=inputs, outputs=outputs, name=new_model_name)
12
13 my_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
14                 loss=tf.keras.losses.CategoricalCrossentropy(),
15                 metrics=['accuracy', tf.keras.metrics.Precision(),
16                        tf.keras.metrics.Recall()])
```

Figure 6.2 - Model layers order and organization.

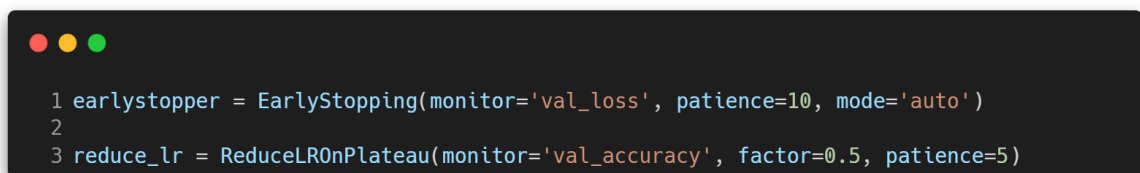
The code invokes the “*get_pretrained_model*” auxiliary method to retrieve a pre-trained model to be used as a feature extractor. This model’s output is then flattened with the “Flatten” layer before being fed through a fully connected (“Dense”) layer with 1024 nodes and a ReLU activation function. To prevent the model from overfitting, a dropout layer is introduced with a rate of 0.2. Lastly, the output is a fully connected layer with a softmax activation function that predicts the probability of each class. The built model is saved in the variable “*my_model*” with a custom name that reflects the pre-trained model and the used technique (Transfer-Learning with Feature Extraction or Fine-Tuning).

6.2.1 Regularization Methods

As explained before, overfitting occurs when a model begins memorizing the training data rather than learning general patterns. In order to prevent this from happening, a Dropout layer was used with a rate of 0.2 as a regularization technique to avoid overfitting and to increase the robustness of the neural network in order to improve the generalization capabilities of the model. In other words, the Dropout layer works by randomly “dropping

out” a specific fraction of the input units (neurons) throughout each training stage. The “0.2” means that during training, 20% of the input units will be set to 0 at each update, while the remaining 80% will be rescaled by $1 / (1 - dropout_rate)$ to retain the overall magnitude of the input.

Beyond that, when fitting the model, “EarlyStopping” and “ReduceLRonPlateau” callbacks were added to help prevent overfitting and enhance model performance. The first callback checks the validation loss throughout training and stops it if it does not improve after a set number of epochs (specified by patience with 10 epochs). The second callback is used to dynamically adjust the learning rate during training by monitoring validation accuracy and reducing the learning rate by a factor of 0.5 if the metric does not improve after a given number of epochs (defined by patience with 5 epochs), in other words, taking smaller steps towards the optimal solution.



```
1 earlystopper = EarlyStopping(monitor='val_loss', patience=10, mode='auto')
2
3 reduce_lr = ReduceLRonPlateau(monitor='val_accuracy', factor=0.5, patience=5)
```

Figure 6.3 - Model fitting callbacks.

6.2.2 Evaluation Metrics

When analyzing the efficiency of machine learning models, it is critical to take into account a variety of metrics that provide insight into various elements of the model’s performance.

6.2.2.1 Accuracy

Accuracy computes the ratio of correctly predicted instances to the total number of instances in the dataset as follows:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

6.2.2.2 Precision

Precision measures the proportion of accurately predicted positive cases (TP - true positives) out of all instances predicted as positive ($TP + FP$, with FP representing false positives). It calculates the model's capacity to minimize false positives as:

$$Precision = \frac{TP}{TP + FP}$$

6.2.2.3 Recall

The proportion of properly predicted positive cases (true positives) out of all actual positive instances ($TP + FN$, with FN representing false negatives) is measured by recall, also known as sensitivity or true positive (TP) rate. It assesses the model's ability to minimize false negatives as follows:

$$Recall = \frac{TP}{TP + FN}$$

6.2.2.4 F1 Score

The F1 Score is derived from the harmonic mean of precision and recall. It gives a balanced assessment of a model's performance by taking both recall and precision into account at the same time, and is calculated as follows:

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

6.2.3 Optimization Algorithms

Optimizers are algorithms that adjust or tweak the properties of a neural network, such as layer weights or learning rate, to reduce loss and so enhance the model. Adam (Adaptive Moment Estimation) (Kingma & Ba, 2015) is a deep neural network training adaptive optimization technique that can be thought of as a combination of the properties of AdaGrad (Adaptive Gradient Algorithm) (Duchi et al., 2011) and RMSprop (Root Mean Square Propagation Algorithm) (Tieleman & Hinton, 2012). When compared to others, this algorithm was chosen for the task with a learning rate of 0.0001 because of its efficiency and speed, low memory demand, wide applicability, and robustness to sparse gradients.

6.3 Training History

For configuration A, Xception and InceptionV3 were the best performing models. The following figures show the training history of these models with Transfer Learning and Fine-Tuning.

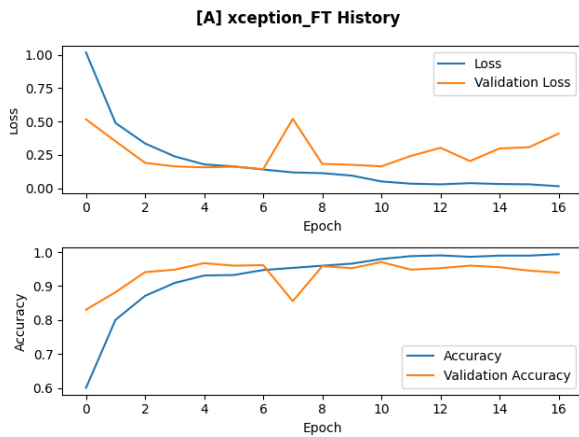


Figure 6.4 - Xception model training history.

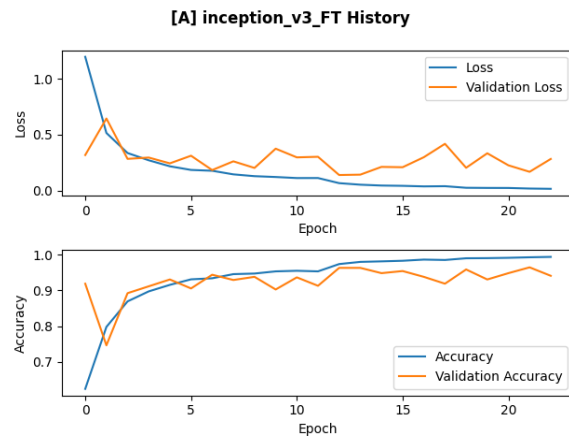


Figure 6.5 - InceptionV3 model training history.

When inspecting the Xception model training history, Figure 6.4, an oscillation or unstable pattern can be seen around epoch number 7. In other words, “sharp spikes” or “dips” in the validation loss or accuracy may suggest that the model is struggling to generalize and is extremely sensitive to slight differences in the training data, a condition known as overfitting. The loss/accuracy discrepancy is another major symptom of this issue. After a certain number of epochs (ten for the first, twelve for the second), the training loss (blue line) continues to decrease while the validation loss plateaus or begins to increase, indicating that the model is memorizing the training data rather than learning generalizable patterns. Similar to the loss values, a significant difference between training and validation accuracy, also indicates potential overfitting. As can be seen, training accuracy improves while validation accuracy stagnates or decreases, indicating that the model is becoming too specialized in the training data and may not perform well on new examples.

In summary, overfitting is a complex phenomenon driven by factors such as dataset quantity, quality and diversity of samples (which are unchangeable at the time of writing), model complexity, and hyperparameter settings. It is critical to analyze these elements collectively rather than relying exclusively on individual outcomes from training graphs. As a result, regularization techniques such as dropout, early stopping, or class weighting were applied to add constraints to the model’s learning process; otherwise, the outcomes would have been substantially worse.

Table 6.1 - Training history results for each model with fine-tuning (configuration A).

Transfer Learning + Fine-Tuning						
Training History Results (Configuration A)	Train Loss	Val. Loss	Train Accuracy	Val. Accuracy	Elapsed Time (mm:ss)	Num. Epochs
DenseNet201	0.02258	0.22636	0.99337	0.96165	13:38	23
InceptionResNetV2	0.01509	0.31952	0.99466	0.94100	20:57	26
InceptionV3	0.01585	0.28321	0.99392	0.94100	12:03	23
NasNetLarge	0.03079	0.34904	0.98876	0.93068	28:52	20
ResNet50V2	0.03487	0.20026	0.98821	0.95870	08:00	19
Xception	0.01463	0.40957	0.99392	0.93953	11:55	17
<i>Average (\bar{x})</i>	<i>0.02230</i>	<i>0.29799</i>	<i>0.99214</i>	<i>0.94543</i>	<i>15:54</i>	<i>21</i>

The average training accuracy of 0.99214 and validation accuracy of 0.94543 show that the models perform well on both sets. Similarly, the low average loss values for training of 0.02230 and validation of 0.29799 indicate that the models effectively minimize errors throughout training and evaluation.

Aside from the models clearly exhibiting overfitting, as evidenced by the significant disparity between training and validation performance, the high training accuracy indicates that the models have learned the training data well, whereas the lower validation accuracy indicates difficulty generalizing to new data.

Overall, the results suggest the models are performing well and reaching high accuracy with low losses, implying that learning and generalization were successful.

7. RESULTS

The six different trained CNN architectures were evaluated using transfer learning with two different approaches: feature extraction and fine-tuning. As expected, while all models showed similar results, the best performance was consistently achieved when using transfer learning with fine-tuning.

The results achieved match the expected benefits of using this technique, including task adaptation, increased model capacity, and knowledge transfer. Overall, the results show how this technique can improve model performance and highlight how pre-trained models can be used to take on new tasks. Besides, when compared with some related works referenced earlier, the results are at least similar, and in some cases are better.

Table 11.1 demonstrates that, compared to the results that were obtained for models trained with a training set with a larger number of data (80%), a smaller training set (70% of the total data) and a correspondingly larger test set (30%) result in a worse accuracy regardless of the selected configuration and model. This occurs because the model struggles to recognize the patterns in the data since it has less data to learn from during training. As a result, it performs worst when tested against new, unseen data.

Table 7.1 - Accuracy results comparison between each model-configuration with fine-tuning (training w/70%).

Transfer Learning + Fine-Tuning (Training Set w/70%)			
Accuracy Results (Test Subset)	CONFIGURATIONS		
	A	B	C
DenseNet201	0.788	0.733	0.750
InceptionResNetV2	0.810	0.736	0.761
InceptionV3	0.812	0.752	0.747
NasNetLarge	0.770	0.704	0.728
ResNet50V2	0.773	0.741	0.741
Xception	0.821	0.763	0.759
<i>Average (\bar{x})</i>	<i>~0.796</i>	<i>~0.738</i>	<i>~0.748</i>

Nevertheless, as shown by the results in Table 11.3, it is advised to choose a ratio of 80% to 20% for training and testing data since it achieves a fair balance between having enough data for the model to be trained efficiently and enough test data for accurate evaluation. Compared to smaller training set sizes, such as the one used for Table 11.1, it enhances model generalization, which results in improved overall performance.

Table 7.2 - Accuracy results comparison between each model-configuration with feature extraction (training w/80%).

Transfer Learning + Feature Extraction (Training Set w/80%)			
Accuracy Results (Test Subset)	CONFIGURATIONS		
	A	B	C
DenseNet201	0.798	0.750	0.754
InceptionResNetV2	0.783	0.684	0.739
InceptionV3	0.750	0.702	0.724
NasNetLarge	0.762	0.673	0.730
ResNet50V2	0.776	0.716	0.719
Xception	0.776	0.700	0.744
<i>Average (\bar{x})</i>	<i>~0.774</i>	<i>~0.704</i>	<i>~0.735</i>

Table 7.3 - Accuracy results comparison between each model-configuration with fine-tuning (training w/80%).

Transfer Learning + Fine-Tuning (Training Set w/80%)			
---	--	--	--

Accuracy Results (Test Subset)	CONFIGURATIONS		
	A	B	C
DenseNet201	0.833	0.781	0.793
InceptionResNetV2	0.835	0.787	0.744
InceptionV3	0.836	0.764	0.754
NasNetLarge	0.800	0.749	0.757
ResNet50V2	0.829	0.769	0.763
Xception	0.858	0.791	0.779
<i>Average (\bar{x})</i>	<i>~0.832</i>	<i>~0.774</i>	<i>~0.765</i>

For instance, the accuracy results for each model configuration in the test subset (images that the models have never seen) are shown in the tables above. As can be seen, the accuracy is generally better when the models were trained using the fine-tuning technique, with an average accuracy of roughly 80%. In configuration A, Xception reached the best result with 85.8% accuracy (Figure 7.1 and Table 7.3).

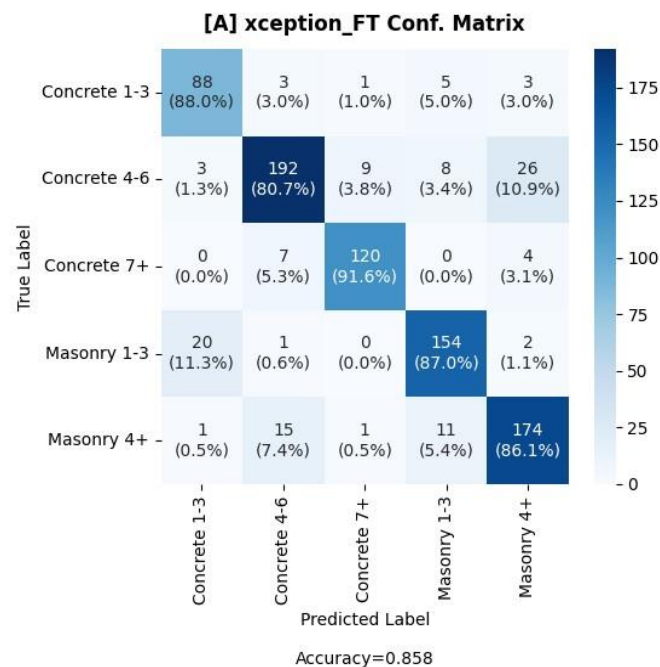


Figure 7.1 - Confusion matrix for configuration A and Xception model trained with fine-tuning.

Table 7.4 - Classification report for configuration A and Xception model trained with fine-tuning.

	Precision	Recall	F1 Score	Support
Concrete 1-3	0,7857143	0,8800000	0,8301887	100

Concrete 4-6	0,8807339	0,8067227	0,8421053	238
Concrete 7+	0,9160305	0,9160305	0,9160305	131
Masonry 1-3	0,8651685	0,8700565	0,8676056	177
Masonry 4+	0,8325359	0,8613861	0,8467153	202
Macro AVG	0,8560366	0,8668392	0,8605291	848
Weighted AVG	0,8602514	0,8584906	0,8585408	848

The “Support” column in the classification report (Table 7.4) provides the number of instances in the test subset for each class. Additionally, the “Weighted AVG” row considers each class’s performance weighted by its proportion in the data, which accounts for the class imbalance. As a result, the Precision, Recall, and F1 Score weighted averages are, respectively, 0.8602512, 0.8584906, and 0.8685389.

It’s important to point out that “Concrete 1-3” had the worst accuracy and, as a result, the worst F1 Score. This is likely because, as can be seen in Figure 4.7, it has the fewest data among all dataset subsets. That is, there weren’t enough samples in the training subset to allow the model to generalize and get a better result on the test subset.

Despite the differences in these measurements between classes, they show overall fairly good performance in identifying each of the categories.

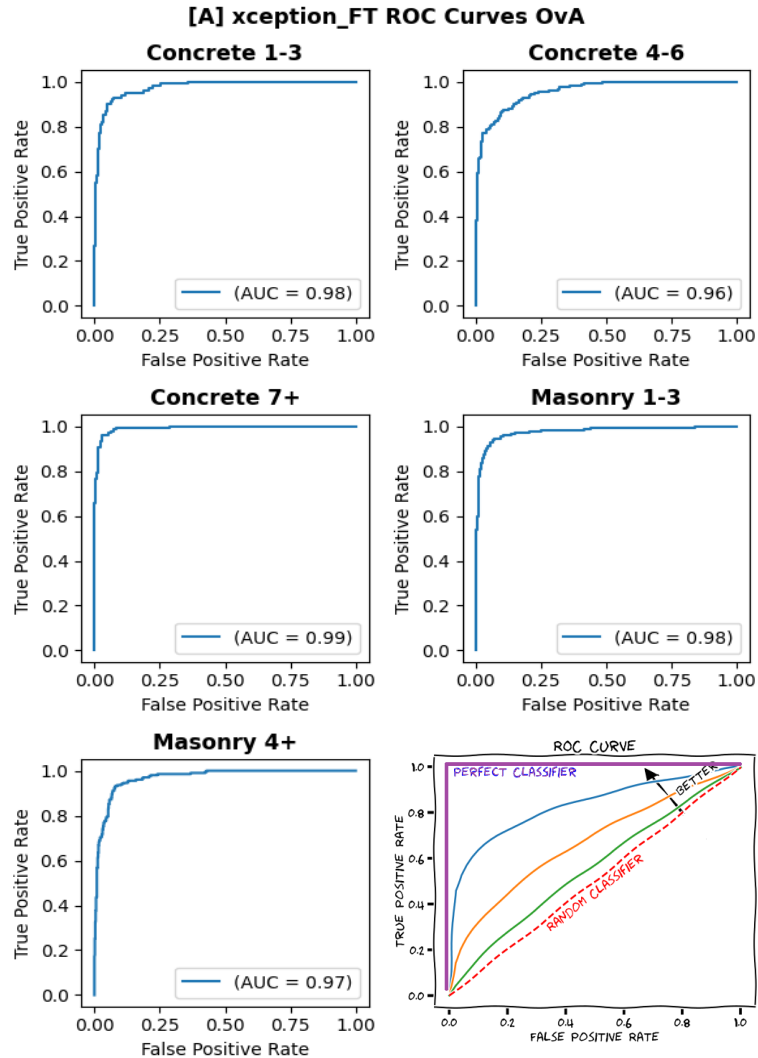


Figure 7.2 - ROC Curves for configuration A and Xception model trained with fine-tuning.

Using the “One-VS-All” (OvA) approach, the ROC (Receiver Operating Characteristic) Curves can be expanded to assess the performance of each class separately, which helped to look into this model more effectively. Its performance in distinguishing between classes improves with increasing Area Under the Curve (AUC) size. By looking at the theoretical graph in the lower right corner of Figure 7.2, the model performs very well overall, approaching excellence. However, the question of how the algorithm could anticipate these features simply by looking at an image arose throughout development. Activation heatmaps were created to try to explain this behavior.



Figure 7.3 - Three randomly picked activation heatmaps from the test subset.

Grad-CAM (Gradient-Weighted Class Activation Mapping) is a machine learning technique that identifies the region of the image that has the most influence on the prediction that was made (the region that is most activated). This enables us to find potential errors or areas for improvement as well as comprehend the logic behind the model's predictions.

Figure 7.3 displays three randomly selected pictures from the test subset that the model correctly predicted. The first image (on the left) was returned by Google's API, while the other two were taken in place.

A tree can be seen in the first image, obtained from the GSV, hiding a small region of the building's facade. However, it was neither significant enough to be detected during the anomaly removal procedure nor to mislead the model prediction.

Figure 7.4Figure 7.3 illustrates three images from the test subset, all of which were incorrectly predicted by the model. The first and third images (on the left and right, respectively) were captured in the research area, while the middle image was obtained through Google's API.

These mispredictions highlight the challenges associated with classifying building characteristics, including construction material and the number of floors. Notably, even for qualified professionals, identifying materials such as concrete and masonry solely through visual inspections can be difficult.

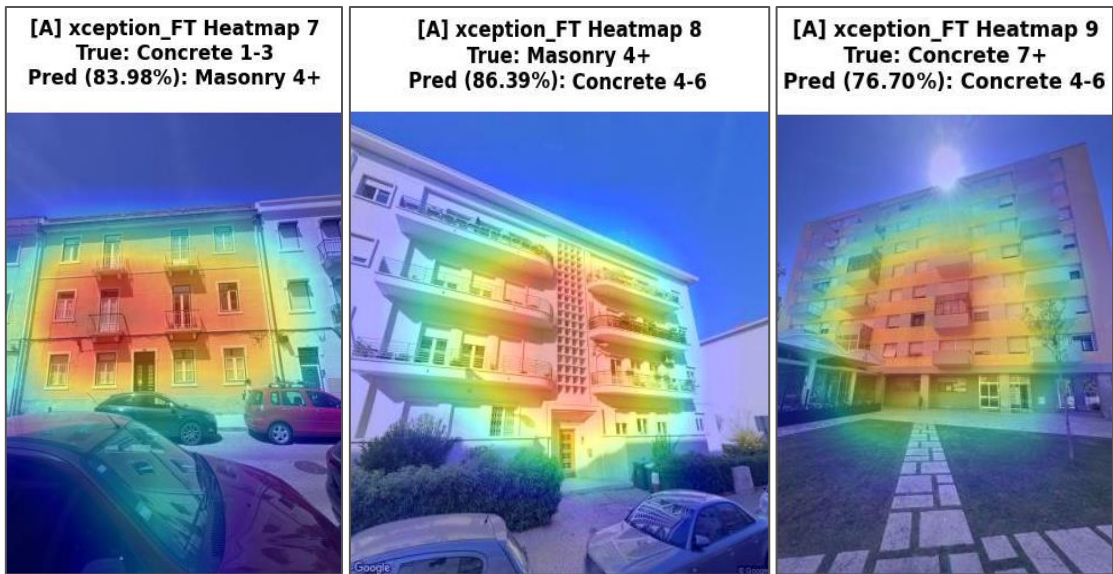


Figure 7.4 - Three activation heatmaps from the test subset with mispredictions.

While the model accurately predicts the number of floors in the central image, other instances of prediction errors may result from the inherent difficulty in visually distinguishing between these specific building materials. Furthermore, upon closer examination of the third (right) image, it becomes clear that the activated region is not properly concentrated on the building's facade, i.e. it ignores the upper floors, which is in line with the incorrect prediction of the number of floors. This observation suggests a potential limitation in the model's ability to capture details in higher buildings or, for the opposite situation, that is, when it fails the prediction because it ignores the first floor of the building (ground floor).

Overall, the model performs as expected given its training. In other words, it sharply concentrates on the building's facade independently of other elements in the scene, such as nearby cars and trees.

8. DISCUSSION AND CONCLUSION

The contributions of this work, as compared to the state of the art, are the wider variety of CNN architectures utilized, the increased number of building characteristics used, and the careful handling of the data - being manually and automatically verified.

When directly comparing with the results obtained from the related state of the art, besides the context and limitations of each study as well as possible dataset biases and other external factors, the Transfer Learning approach with Fine-Tuning seems to yield the better results. However, different datasets and building attributes in different locations can definitely impact the model's performance. Factors such as the number of classes, data quantity and quality, and distribution may vary, leading to different results (as demonstrated by the results obtained from different class configurations and subset sizes - training subset with 70% or 80% of the total data, for example).

In particular, the developed Xception model has shown promising performance, outperforming the best model reported in the Oslo paper in terms of accuracy and precision, although its recall is slightly lower compared to the top-performing models in other works. Overall, the performance metrics (accuracy, precision, and recall) of the developed models are generally in the same range as those reported in the papers, indicating that the model (this one in particular) is a strong competitor against the state-of-the-art models in the field.

For example, in the study conducted by Aravena Pelizari et al. (2021), different accuracy results were obtained based on three distinct building features. Notably, the transfer-learned NasNet-A consistently performed the best across all scenarios. In the first scenario, which involved six classes representing only the height of the buildings (storeys), the highest accuracy achieved was 85.9%. In the second scenario, with seven classes representing the material of the LLRS, the top accuracy reached 87.1%. Lastly, in the third scenario, with 14 classes representing the SBST (an aggregation of classes from the two previously described scenarios), the best accuracy obtained was 82.7%. On the other hand, in the Oslo Study (Ghione et al., 2022), with 6 classes representing only the building typologies, the fine-tuned DenseNet201 was the best model with 82.5% of accuracy.

Although the results so far are encouraging, the performance of the final model can still be improved. In comparison to other similar works, the dataset used in this one can be considered as being relatively small. The chosen approach also has inherent limitations. We have identified 2844 buildings in total for the Alvalade parish, a number that doesn't change in a short period of time. If we were to rely only on the automated method, we would have only 2844 images in our dataset, which would be considerably insufficient given the difficulty of our target problem. The effectiveness of the model is strongly dependent on the size of the dataset, which relies on the region being studied.

Future research should be centered on the developed CNNs capacity to be used in various regions or even countries, both with and without retraining the entire neural network, that is, without going through the full procedures again. This means that it's crucial to know how much of what has already been done may be applied to other cities and still produce acceptable results. This is expected to be possible, at least in other locations or cities in Portugal, where the building types are fairly similar and the models can generalize.

In order to decide the best predicting characteristics, future work will test the use of additional classes that combine other pertinent building-related attributes. Also, the trained models will be applied to other Lisbon boroughs to assess the generalizability of the overall approach. The outcomes will also be compared to ground truth seismic event data. Additionally, at the time of this document's submission, optimization of several of the steps of this work has already begun. It will be possible to determine not only the precise location of each building but also the route between the person's current location and the building thanks to the development of a mobile application and the resulting new website. In addition, it will be possible to take pictures of its facade and annotate its features directly on the application. With this new improvement, independent of the application's speed, usability, and responsiveness, a rise in productivity related to this step is anticipated as a number of tasks are brought together in one place at the same time.

In conclusion, this study clearly demonstrates the possibility for automating the classification of building typologies in order to simplify wide seismic risk assessment using machine learning and existing street-level images. A state-of-the-art CNN, fine-tuned on the ImageNet database, such as Xception, was used to construct a model that achieved 86% accuracy on unseen data by utilizing information from Google Street View and images captured during fieldwork. The

workflow that has been produced points out a quick and highly automated method for assessing seismic risk while minimizing costs and time demands.

Furthermore, this research emphasizes the efficiency of CNN models for attribute classification, offering insightful information on the effectiveness of different architectures and techniques in diverse geographic locations. However, as with any comparison research, careful interpretation of the results is important: considering the particular attributes, classes, and unique characteristics of each study. It is also very important to keep in mind that real-world applications could bring up more complications and differences requiring for more research. Nevertheless, the results of this study highlight the potential influence of machine learning-driven techniques in seismic risk assessment and present a promising direction for further development in the area.

9. BIBLIOGRAPHY

- Aravena Pelizari, P., Geiß, C., Aguirre, P., Santa María, H., Merino Peña, Y., & Taubenböck, H. (2021). Automated building characterization for seismic risk assessment using street-level imagery and deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 180. <https://doi.org/10.1016/j.isprsjprs.2021.07.004>
- Bhatta, S., & Dang, J. (2023). Seismic damage prediction of RC buildings using machine learning. *Earthquake Engineering and Structural Dynamics*, 52(11). <https://doi.org/10.1002/eqe.3907>
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January*. <https://doi.org/10.1109/CVPR.2017.195>
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12.
- Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. In *Learning from Imbalanced Data Sets*. <https://doi.org/10.1007/978-3-319-98074-4>
- Ghione, F., Mæland, S., Meslem, A., & Oye, V. (2022). Building Stock Classification Using Machine Learning: A Case Study for Oslo, Norway. *Frontiers in Earth Science*, 10. <https://doi.org/10.3389/feart.2022.886145>
- Gonzalez, D., Rueda-Plata, D., Acevedo, A. B., Duque, J. C., Ramos-Pollán, R., Betancourt, A., & García, S. (2020). Automatic detection of building typology using deep learning methods on street level images. *Building and Environment*, 177. <https://doi.org/10.1016/j.buildenv.2020.106805>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December*. <https://doi.org/10.1109/CVPR.2016.90>

- He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9908 LNCS. https://doi.org/10.1007/978-3-319-46493-0_38
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January*. <https://doi.org/10.1109/CVPR.2017.243>
- Kang, J., Körner, M., Wang, Y., Taubenböck, H., & Zhu, X. X. (2018). Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145. <https://doi.org/10.1016/j.isprsjprs.2018.02.006>
- Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.
- Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Science Department, University of Toronto, Tech*.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6). <https://doi.org/10.1145/3065386>
- Laupheimer, D., Tutzauer, P., Haala, N., & Spicker, M. (2018). NEURAL NETWORKS for the CLASSIFICATION of BUILDING USE from STREET-VIEW IMAGERY. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(2). <https://doi.org/10.5194/isprs-annals-IV-2-177-2018>
- Law, S., Seresinhe, C. I., Shen, Y., & Gutierrez-Roig, M. (2020). Street-Frontage-Net: urban image classification using deep convolutional neural networks. *International Journal of Geographical Information Science*, 34(4). <https://doi.org/10.1080/13658816.2018.1555832>

- Li, Y., Chen, Y., Rajabifard, A., Khoshelham, K., & Aleksandrov, M. (2018). Estimating building age from google street view images using deep learning. *Leibniz International Proceedings in Informatics, LIPIcs*, 114. <https://doi.org/10.4230/LIPIcs.GIScience.2018.40>
- Novack, T., Vorbeck, L., Lorei, H., & Zipf, A. (2020). Towards detecting building facades with graffiti artwork based on street view images. *ISPRS International Journal of Geo-Information*, 9(2). <https://doi.org/10.3390/ijgi9020098>
- Ritchie, H., & Roser, M. (2014). *Natural Disasters: Our World in Data*. University of Oxford. <https://ourworldindata.org/natural-disasters>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3). <https://doi.org/10.1007/s11263-015-0816-y>
- Silva, V., Crowley, H., Pagani, M., Monelli, D., & Pinho, R. (2014). Development of the OpenQuake engine, the Global Earthquake Model's open-source software for seismic risk assessment. *Natural Hazards*, 72(3). <https://doi.org/10.1007/s11069-013-0618-x>
- Silva, V., Yepes-Estrada, C., Dabbeek, J., Martins, L., & Brzev, S. (2018). *GED4ALL - Global Exposure Database for Multi-Hazard Risk Analysis*.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*. <https://doi.org/10.1609/aaai.v31i1.11231>
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December*. <https://doi.org/10.1109/CVPR.2016.308>

- Tieleman, T., & Hinton, G. (2012). Divide the gradient by a running average of its recent magnitude. *Human and Machine Hearing*, 4(2).
- Yang, Q., Zhang, Y., Dai, W., & Pan, S. J. (2020). Transfer Learning. In *Transfer Learning*. <https://doi.org/10.1017/9781139061773>
- Yu, Q., Wang, C., McKenna, F., Yu, S. X., Taciroglu, E., Cetiner, B., & Law, K. H. (2020). Rapid visual screening of soft-story buildings from street view images using deep learning classification. *Earthquake Engineering and Engineering Vibration*, 19(4). <https://doi.org/10.1007/s11803-020-0598-2>
- Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6). <https://doi.org/10.1109/TPAMI.2017.2723009>
- Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR.2018.00907>