KSCHOOL

MASTER OF DATA SCIENCE

KSCHOOL

February 2023

Master Final Project

Room Classifier

Author : Daniel Collado Bertomeu

Contents

Chapter 1

Introduction

The room image classifier is a machine learning system that is used to identify and categorize images of different rooms. This system is built using TensorFlow, a popular open-source software library for machine learning. The primary goal of this classifier is to accurately recognize different types of rooms based on their visual characteristics, such as furniture, flooring, and wall colors.

The development of this room classifier is relevant because it can help improve the user experience and interface of web pages. The system automates the process of categorizing images of rooms, reducing the manual effort required for tagging and increasing the accuracy of the results. This results in a better user experience for those uploading images, as they no longer need to manually tag the class of the room image. The system can be used in a variety of applications, such as interior design, real estate, and home improvement.

The state of the art in room image classification involves the use of deep learning techniques, particularly convolutional neural networks (CNNs), which have demonstrated excellent performance in image recognition tasks. Previous related work in this field includes the use of transfer learning, where a pre-trained model is fine-tuned for a specific task, and the use of data augmentation techniques to increase the size of the training dataset.

In this project, we will use transfer learning to fine-tune a pre-trained model on the room image dataset. This approach allows us to take advantage of the pre-training to quickly learn relevant features for the room classification task, while also allowing us to further adapt the model to our specific task by fine-tuning its parameters on the room image dataset. Through this process, we hope to achieve high accuracy in classifying images of rooms.

Chapter 2

Raw Data Description

In this project, the raw data that is used to classify room images is comprised of over 6000 images belonging to five different room types: bedrooms, house plans, kitchens, living rooms, and bathrooms. These images have been stored in JPEG, JPG, and PNG formats, which are the image formats supported by TensorFlow and have varying resolutions.

The labeling of each room type is done based on the directory name, making it easier for TensorFlow to read and process the data. The room type is represented by a numerical value, with 1 representing bedrooms, 2 representing house plans, 3 representing kitchens, 4 representing living rooms, and 5 representing bathrooms. This numerical representation simplifies the analysis and processing of the data by TensorFlow.

The ***GetImagesGoogle.py*** code was created to obtain images of rooms for the classifier. The code uses the BeautifulSoup and urllib libraries to scrape the image URLs from Google images and download the images. The code also uses the Selenium webdriver to automate the scrolling process on the Google images page to gather more images. The code defines the search term and creates a directory with the same name to store the downloaded images. The images are then parsed and the URLs are obtained using the img tag and class. The code downloads the images using the URL and saves them in the created directory.

The data was split into 80% training and 20% validation, ensuring that the classifier is tested on unseen data. This split is crucial in ensuring that the classifier is able to generalize well to new images, rather than simply memorizing the training data.

This dataset is carefully curated to ensure a balanced representation of the different types of rooms, with approximately 900 images per class. This balance is crucial in ensuring that the classifier is not biased towards any particular type of room and can accurately classify all classes.

It is crucial to acknowledge the observation in image 2.1, which demonstrates an imbalanced distribution of images between the five room classes. Specifically, the class of Bedroom, Kitchen, Living Room, and Bathroom had a higher number of images available compared to the class of House Map. This design choice was deliberate and aimed to improve the performance of the Convolutional Neural Network (CNN) used in this project.

By having a greater quantity of images for the more challenging classes, the CNN was compelled to learn and distinguish between more complex and nuanced features. This led to better overall performance, rather than relying solely on the easier House Map class. Furthermore, the increased number of images for the difficult-to-classify classes helps to reduce the likelihood of overfitting and enhances the robustness of the classifier.

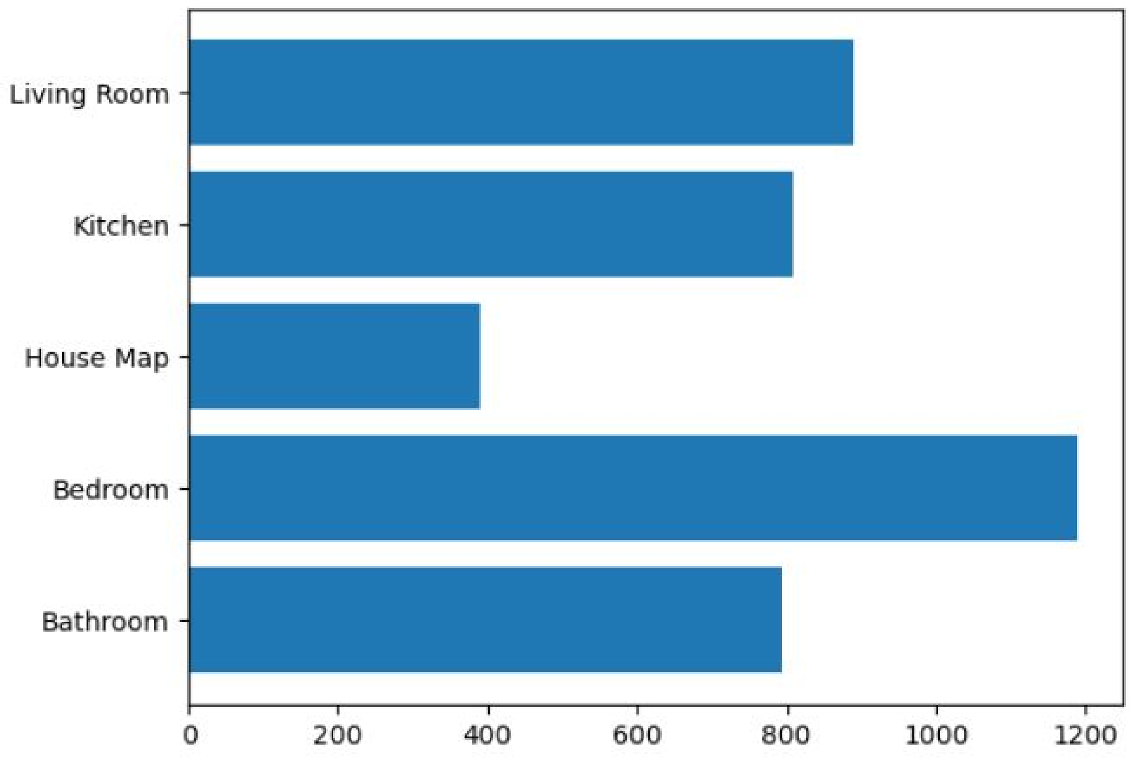

Figure 2.1: Representation of the number of images per room class.

In order to ensure the accuracy and quality of the raw data used for this room image classifier project, each image was thoroughly reviewed and verified. Any image that did not align with the corresponding room class was discarded to prevent any negative impact on the training of the model. This process is illustrated in figure 2.2. The following images were specifically discarded:

1. Images generated through computer software.
2. Images that contained a combination of multiple images.
3. Images of rooms that were a combination of multiple room classes, such as a living room-kitchen.
4. Images that contained text.
5. Images that contained watermarks.
6. Images that contained any individuals or animals.

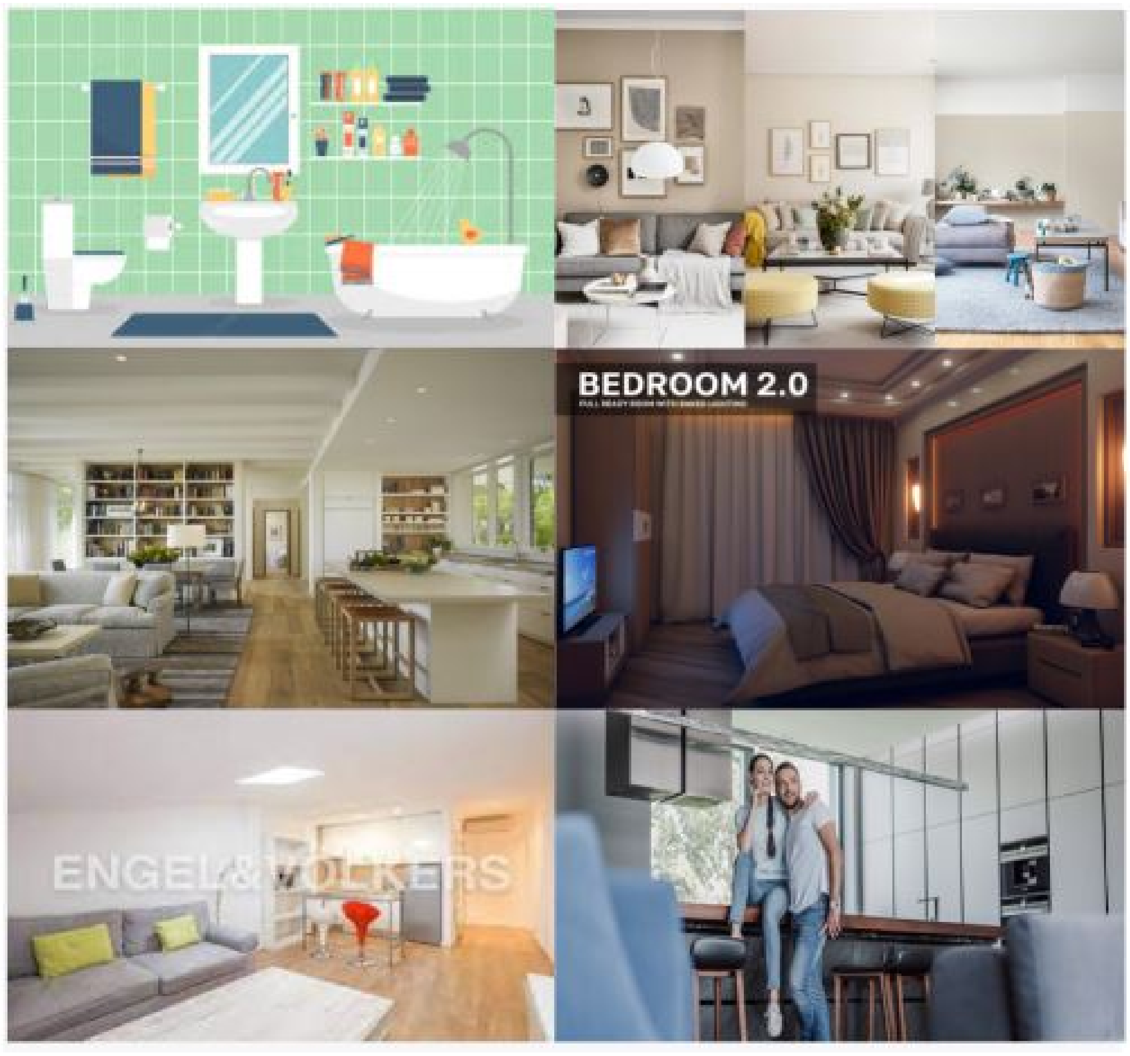

Figure 2.2: Representation of examples of images deleted from the dataset.

Chapter 3

Methodology

The goal of this project was to create a room image classifier using deep learning. The data collection and normalization phase of this deep learning project aimed to build a room image classifier.

The collected dataset consisted of diverse room images, which were carefully curated to guarantee the model's accuracy by incorporating images from various perspectives and illumination conditions. The pre-processing phase involved standardizing the image size, normalizing the pixel values, and transforming the images into a format suitable for the deep learning model. These steps were crucial for the subsequent development of the room image classifier model.

The approach used in this room image classifier project consisted of several stages: Model Design, ImageDataGenerator and Transfer Learning and Fine-Tuning.

3.1 Model Design

In the initial stage, several Convolutional Neural Network (CNN) models were designed and implemented from scratch. These models consisted of multiple layers of convolution, activation, pooling, and dense layers. The architecture of the models was optimized to achieve the best results in terms of accuracy.

The model is designed using a sequential architecture, where the input is processed through several layers in a linear manner.

The first layer is a Conv2D layer with 32 filters and a kernel size of (3,3). The input shape of the images is also defined in this layer. The activation function used is ReLU, which is a non-linear activation function that outputs the input if it is positive, and 0 if it is negative. This layer is then followed by a MaxPooling2D layer which reduces the spatial dimension of the data by half.

The next two layers are similar to the first one, but with the same number of filters and kernel size. These additional Conv2D layers increase the depth of the network and allow the model to learn complex features in the images.

After the Conv2D layers, the data is flattened and passed through a dense layer with 1024 neurons. This layer is used to process the high-level features learned by the Conv2D layers and generate a prediction. The activation function used in this layer is ReLU. A Dropout layer is added to prevent overfitting by randomly dropping out 50% of the neurons during training.

Finally, the output layer is a dense layer with the same number of neurons as the number of classes. The activation function used in this layer is Softmax, which is a probability distribution function that normalizes the outputs of the model to sum up to 1. This allows the model to output a probability for each class, indicating the likelihood that a given image belongs to each class.

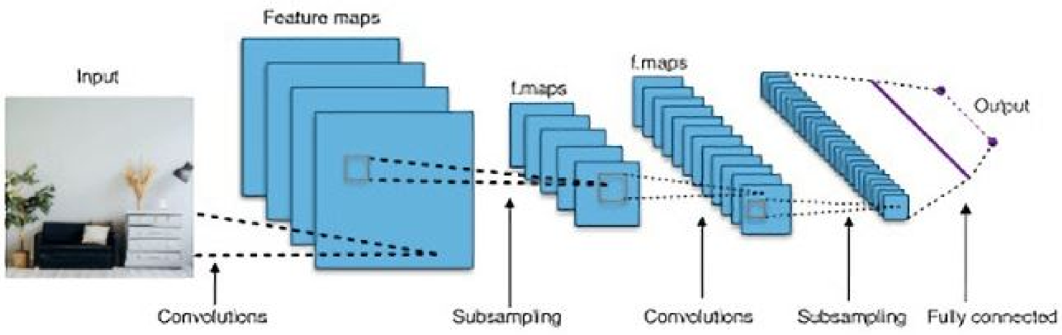

Figure 3.1: Image of an example of phases of a CNN.

In the context of a CNN designed to classify room images, the Sparse Categorical Crossentropy loss is a suitable choice because it measures the error between the predicted class probabilities and the true class probabilities, which aligns well with the goal of accurately classifying the room images into different categories. Additionally, the use of Adam optimizer makes the training process computationally efficient, allowing for fast convergence towards a good model solution.

Accuracy is a common metric used to evaluate the performance of a machine learning model, including CNNs. It measures the proportion of correct predictions made by the model, which is a direct reflection of the model's ability to correctly classify the room images.

For a CNN designed to classify room images, using accuracy as a performance metric is appropriate because it provides a simple and straightforward way to evaluate the model's ability to correctly identify the room images and assign them to the correct class. This can provide insight into the model's performance and help to identify areas for improvement. Additionally, accuracy is a widely used metric in the field of computer vision and is well understood by the research community, making it a good choice for evaluating the performance of the CNN model.

During validation, the model achieved around 60% accuracy. However, the model overfit the training data, meaning that it performed well on the training data but not as well on new, unseen data.

3.2 ImageDataGenerator

The ImageDataGenerator from Keras was utilized to apply data augmentation techniques to the training data.

The ImageDataGenerator class is used to augment the training data, which can help the model generalize better to unseen images. The augmentation parameters specified in the code include rescaling the pixel values to be in the range of 0 to 1, applying random zoom, shear, width shift, and height shift transformations, randomly flipping images horizontally, and setting aside a portion of the data for validation. By randomly transforming the training data, the model is exposed to a wider range of variations in the images as we can see in 3.2, which can help it generalize better to new images. This is important in the context of a room image classifier as the model needs to be able to correctly classify rooms even if they have different perspectives or lighting conditions than the images seen during training. By using the ImageDataGenerator class, the model is able to train on a larger and more diverse set of images, which can improve its performance on unseen data.

The utilization of this data augmentation technique along with the previously designed CNN model resulted in an accuracy of approximately 73 % on the validation set. Furthermore, the model did not show evidence of overfitting, as its performance on the validation set was consistent with its performance on the training set.

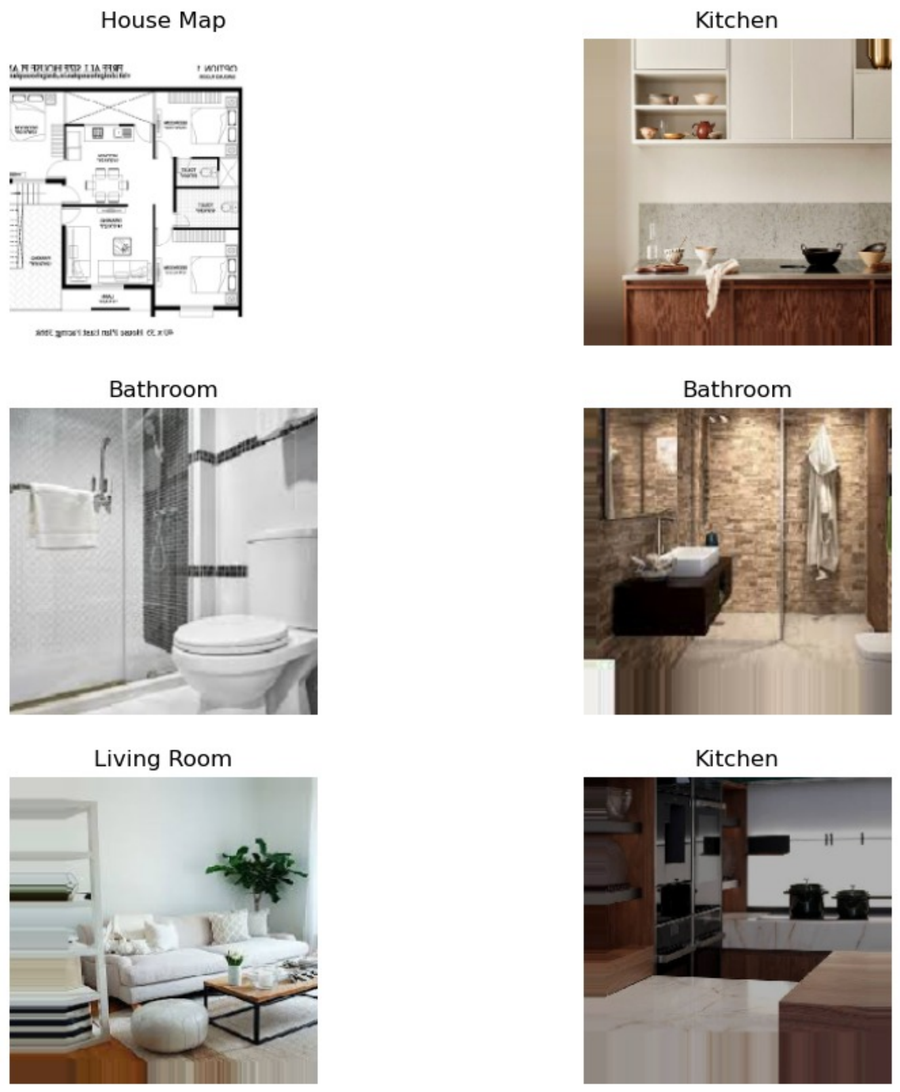

Figure 3.2: Example of images after applying ImageDataGenerator.

3.3 Transfer Learning and Fine-Tuning

Transfer learning and fine-tuning techniques were utilized to improve the performance of the CNN models. The models were fine-tuned using pre-trained models such as VGG16, InceptionV3, and MobilenetV2. This approach leverages the features learned from large datasets by the pre-trained model and fine-tunes them for the specific task at hand.

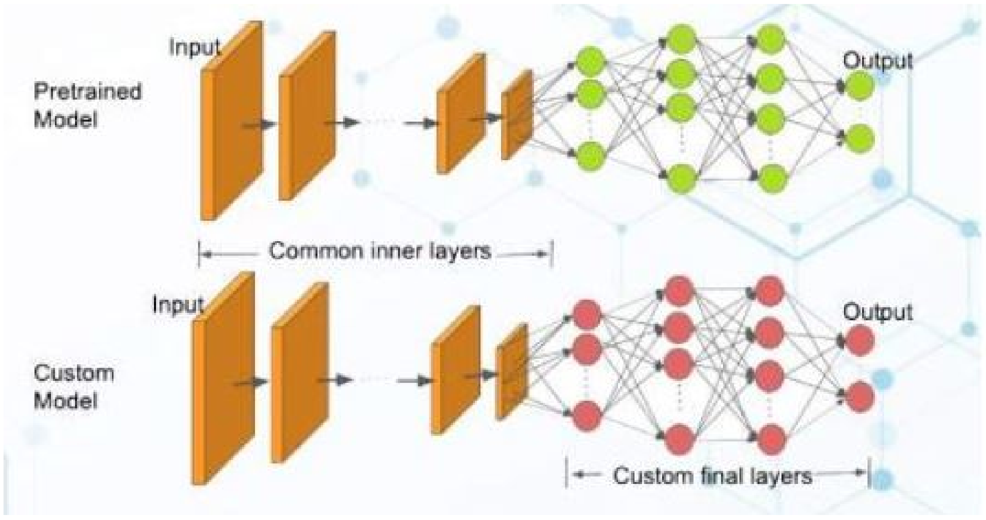

Figure 3.3: Representation of transfer learning technique.

The function ***train_transfer_learning_model*** used in the notebook trains a transfer learning model using the trained model architecture as the base model. First, the function freezes all the layers in the base model to prevent the model from changing the weights of the pre-trained layers during training. Then, a new set of dense layers is added on top of the base model to allow the model to learn more complex functions for image classification.

The new layers consist of a global average pooling layer, followed by three dense layers with ReLU activation functions, and a dropout layer to prevent overfitting. The final layer uses a softmax activation function to generate class probabilities for the image classification problem.

For the purpose of faster and more efficient hyperparameter tuning and additions to the base model, we created a program called ***fine_tuning.py***. This program was used to carry out all the tests to obtain the best performing model.

The model in this project was compiled using Stochastic Gradient Descent (SGD) as the optimizer, instead of the popular Adam optimizer that was used previously. This choice was made in order to have more control over the learning rate. The learning rate of 0.001 and the momentum of 0.9 are hyperparameters that control the speed and stability of the model's weight updates during training.

After the first training phase, the model is unfrozen, and all layers are made trainable. This allows the model to learn new features that are specific to the task of room image classification. In the second training phase, the model is recompiled using SGD with a lower learning rate of 0.0001 and the same momentum of 0.9. This ensures that the model's weights are updated more gradually, which can prevent overfitting and lead to more stable and accurate predictions.

The CNN models were trained on the training data, and the accuracy was measured on the validation data. The best performing model was selected based on the highest accuracy achieved on the test data. In the next table 3.1 we can see the results of each model.

Table 3.1: Table with the accuracy obtained in each transfer learning model.

Model	VGG16	InceptionV3	MobilenetV2
Validation	93.1%	91.87 %	92.36 %
Test	88.42 %	86.11 %	86.4 %

In conclusion, the approach used in this project involved designing and implementing multiple Convolutional Neural Network models, utilizing transfer learning and fine-tuning techniques with pre-trained models, and utilizing the ImageDataGenerator to apply data augmentation techniques. The models were trained and evaluated on the room image classification task, and the best performing model was selected based on its accuracy on the validation data. This approach resulted in an accurate and effective room image classifier with a validation accuracy of 93 %.

Chapter 4

Summary of Main Results

The room image classifier developed in this project achieved a validation accuracy of 93% and a test accuracy of 88%. The accuracy achieved demonstrates that the model is capable of accurately classifying images of rooms into one of the five classes: Bedroom, House Map, Kitchen, Living Room, and Bathroom.

This model can provide a significant advantage for a real estate platform that needs to automatically label the images of rooms in properties for sale. The accurate labeling of the images can improve the user experience by allowing customers to quickly and easily navigate the platform and find the properties they are interested in. The automatic labeling of the images also saves time and resources for the real estate platform, as manual labeling is a time-consuming and repetitive task.

The results in table 4.1 represent the performance metrics of a room image classifier with the test data. The classifier has been evaluated on a test dataset containing 691 images of different rooms.

The table shows that the individual performance of each class varies, with some classes performing better than others. The classifier has performed particularly well for the 'House Map' class, with a precision, recall and f1-score of 1.00, indicating 100% accuracy in classifying images in this class. On the other hand, the classifier has performed less well for the 'Bedroom' and 'Living Room' classes, with a lower precision, recall and f1-score compared to other classes. This suggests that the classifier may have a harder time distinguishing these two classes from the others and may result in misclassifications.

In conclusion, the overall accuracy of the classifier is 88 %, which is a good result. The macro average of the precision, recall and f1-score is over 90%, which suggests that the classifier is performing well across all classes. The weighted average of the precision, recall, and f1-score is 89%, which further confirms the overall good performance of the classifier.

Table 4.1: Table of the classification report of the transfer learning model using test data.

Class	Precision	Recall	F1-Score	Support
Bathroom	0.92	0.90	0.91	164
Bedroom	0.77	0.94	0.84	94
House Map	1.00	0.99	0.99	72
Kitchen	0.88	0.87	0.87	194
Living Room	0.82	0.85	0.83	167
Accuracy	-	-	0.89	691
Macro avg	0.89	0.90	0.89	691
Weighted avg	0.88	0.87	0.88	691

From the confusion matrix represented in 4.1, we can draw the following conclusions:

1. The classifier has a relatively high accuracy in classifying the rooms, as most of the samples are classified correctly.

2. The classifier exhibits a higher error rate for classifying bedrooms and living rooms, as compared to the other classes. This could be attributed to the classifier's difficulty in discerning the distinctive features that separate these classes from the rest.

3. Conversely, the classifier has a relatively low error rate in classifying kitchens and bathrooms, indicating that the features that distinguish these classes are well represented and captured by the classifier.

4. The overall accuracy of the classifier is enhanced by its proficiency in correctly classifying a high number of bathrooms.

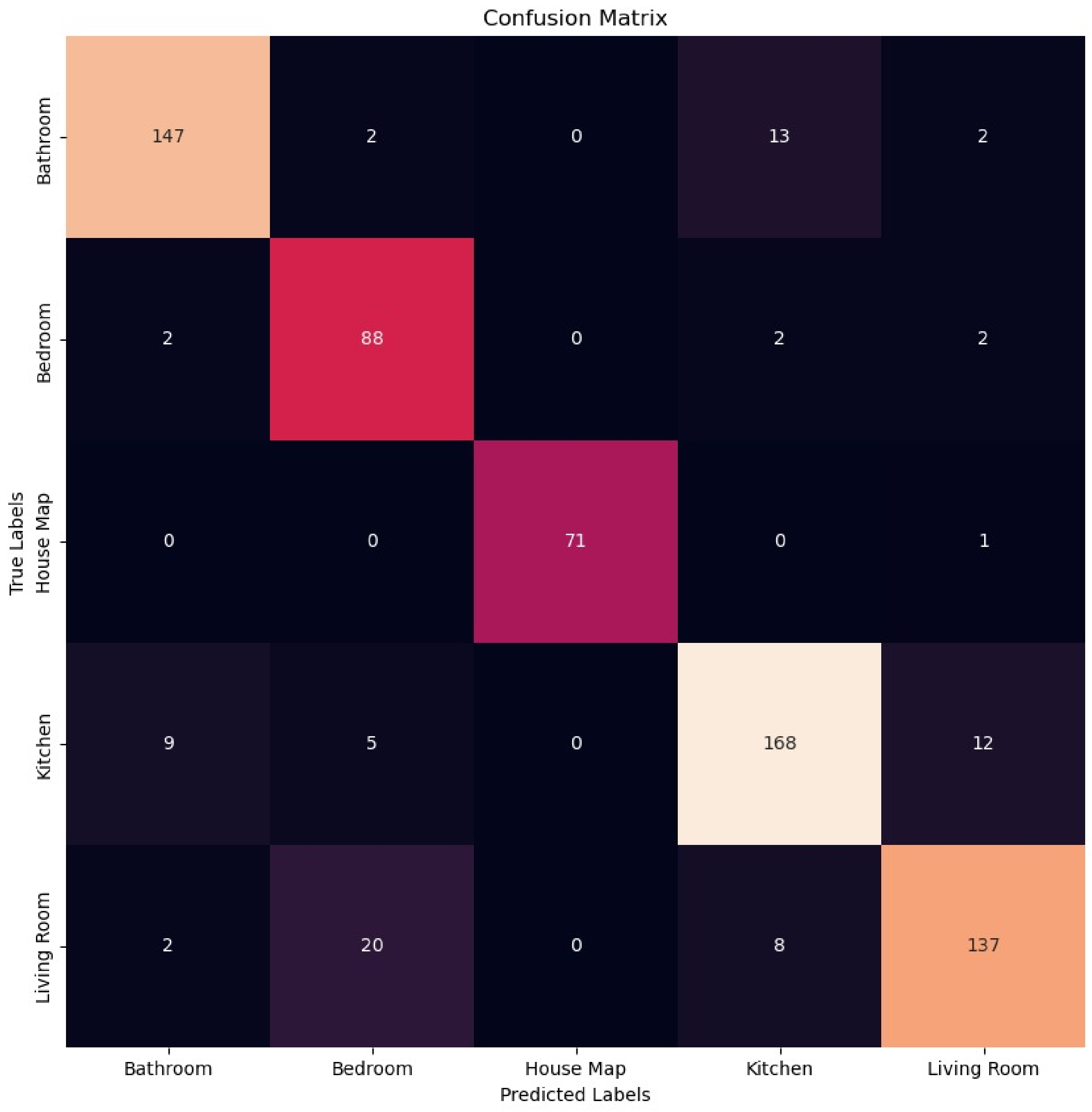

Figure 4.1: Confusion matrix of the transfer learning model using test data.

Chapter 5

Conclusions

The room image classifier developed in this project demonstrates the power of deep learning techniques in the field of computer vision. The model achieved a validation accuracy of 93% and a test accuracy of 88%, which demonstrates its ability to accurately classify images of rooms into one of the five classes: Bedroom, House Map, Kitchen, Living Room and Bathroom.

This project highlights the importance of transfer learning and fine-tuning techniques in improving the performance of deep learning models. By leveraging the features learned from large datasets by pre-trained models, the fine-tuning process was able to significantly improve the accuracy of the model. Additionally, the use of the ImageDataGenerator in applying data augmentation techniques helped to prevent overfitting and increase the diversity of the training set.

The results of this project have important implications for a real estate platform. The accurate labeling of images of rooms in properties for sale can greatly improve the user experience by allowing customers to quickly and easily navigate the platform and find the properties they are interested in. Additionally, the automatic labeling of the images saves time and resources for the platform, as manual labeling is a time-consuming and repetitive task.

Aside from its application in real estate platforms, the room image classifier developed in this project can have a wide range of applications in various fields. The room image classifier can also be used in virtual and augmented reality applications to provide an immersive experience for users. These are just a few examples of the potential applications of this model, and the list is certainly not exhaustive. The results achieved in this project demonstrate the versatility and potential of deep learning techniques in solving real-world problems and improving our daily lives.

In future work, it would be beneficial to expand the dataset to include more diverse images of rooms, as this would give the model more information to learn from and could lead to improved accuracy. Additionally, incorporating images of new room types, such as balconies or corridors, would give the model more features to work with and potentially increase its performance even further. Moreover, incorporating data from different regions and cultures would help the model generalize better and make it applicable to a wider range of real estate platforms. The potential for further improvement highlights the importance of continuously growing and refining the dataset in deep learning projects, particularly in computer vision.

In conclusion, the room image classifier developed in this project demonstrates the power of deep learning in the field of computer vision and has important implications for real estate platforms. The approach used in this project can serve as a blueprint for future projects in the field, and the results achieved demonstrate the effectiveness of transfer learning and fine-tuning techniques in improving the performance of deep learning models.

Chapter 6

User manual for the frontend

6.1 Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

6.1.1 Prerequisites

Make sure you have Python 3.9 and either **conda** or **virtualenv** installed on your system. You can check your Python version by running the following command in your terminal:

```
python3 --version
```

If you do not have Python 3.9 installed, you can download it from the official Python website **here**.

6.1.2 Setting up a Virtual Environment

You can choose to create a virtual environment using either conda or virtualenv.

Using conda

1. Open your terminal or command prompt.
2. Run the following command to create a virtual environment with Python 3.9:

```
conda create --name myenv python=3.9
```

Replace `myenv` with the name you want to give to your virtual environment.

3. Activate the virtual environment by running the following command:

```
conda activate myenv
```

4. Verify that the virtual environment is activated by checking the command prompt. It should display the name of the virtual environment in brackets at the beginning of the line.

Using virtualenv

1. Open your terminal or command prompt.
2. Navigate to your project's directory using the cd command.
3. Run the following command to create a virtual environment:

```
python3 -m venv myenv
```

Replace `myenv` with the name you want to give to your virtual environment.

4. Activate the virtual environment by running the following command:

```
source myenv/bin/activate
```

Note: If you're on Windows, the activation command is slightly different:

```
myenv\Scripts\activate
```

5. Verify that the virtual environment is activated by checking the command prompt. It should display the name of the virtual environment in brackets at the beginning of the line.

6.2 Installing the Project

Once you have activated your virtual environment, you can install the project and its dependencies by following these steps:

1. Clone the project repository:

```
git clone https://github.com/Panatore/RoomsClassifier
```

2. Navigate to the project directory:

```
cd RoomsClassifier
```

3. Install the dependencies:

```
pip install -r requirements.txt
```

6.3 Download Model

Using these link you can download a zip with the **Model** you need to make the predictions.

Once you have downloaded the zip folder you must unzip it in the same directory where the repository has been cloned. Using the next code.

In case you are using Bash:

```
unzip Models.zip
```

In case you are using PowerShell:

```
Expand-Archive .\Models.zip
```

6.4 Execute the webapp

In case you want to deploy the Streamlit app, you will need to execute the next command while you are in the same directory where the file webapp.py is:

```
streamlit run webapp.py
```

This webapp shows us how the user experience will improve using the room image classifier. The room image classifier makes the real estate advertisement creation process even more efficient and user-friendly. The classifier allows users to easily label each room in the property, providing a clear and organized way for potential buyers to view the layout of the property. The labeling process is quick and simple, and the classifier accurately identifies the different rooms in the property, reducing the need for manual input from the user. This feature enhances the visual appeal of the advertisement, making it easier for potential buyers to understand the property layout. The improved organization and visual appeal provided by the room image classifier increase the chances of attracting more potential buyers and ultimately lead to a successful sale. The addition of this feature further demonstrates the webapp's commitment to providing an enhanced user experience and making the process of advertising real estate properties easier and more efficient.

The architecture of the webapp comprises a two-page interface, where the first page is dedicated to the creation of a real estate advertisement. The user is prompted to provide detailed information regarding the property, including its operational status, price, location, street address, and a comprehensive description. The user is also able to upload relevant images to accompany the advertisement, enhancing its visual appeal and providing potential buyers with a comprehensive understanding of the property.

The second page of the webapp serves as the display platform for the created advertisement. Once the advertisement has been created, it is promptly displayed on this page, showcasing all the information and images provided by the user in an organized and visually appealing manner. The display page serves as the main platform for potential buyers to interact with the advertisement and gain a comprehensive understanding of the property being offered for sale.