UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Utilizarea Învăţării prin Transfer pentru îmbunătăţirea performanţei sarcinilor de Viziune Computerizată

Coordonator ştiinţific
Prof. dr. Czibula Gabriela

Student
Ciubotariu George

2021

BABEŞ-BOLYAI UNIVERSITY FACULTY OF MATHEMATICS AND COMPUTER SCIENCE COMPUTER SCIENCE SPECIALIZATION

DIPLOMA THESIS

Using Transfer Learning for enhancing the performance of Computer Vision tasks

Scientific supervisor
Prof. PhD. Czibula Gabriela

Student
Ciubotariu George

2021

Abstract

The main objective of the thesis is to highlight the relevance and importance of applying *unsupervised learning* methods as a preliminary step before developing predictive models. This thesis approaches two problems, one from *Computer vision* and one from *Educational data mining*: from Computer Vision, the problem is that of classifying indoors and outdoor images using *machine learning* and *deep learning* models, while from Educational Data Mining the problem is that of students' performance prediction.

Concerning the Computer Vision domain, we are going to perform an unsupervised learning based analysis with the aim of determining the relevance of depth maps in the context of classification. For further tests to decide on the granularity of information extraction means, features are aggregated from sub-images of different sizes from DIODE data set to compare multiple scales of region attention. Four feature sets will be proposed and comparatively analysed in the context of indoor-outdoor image classification. To empirically confirm the advantage of using features automatically learned from depth maps, the features are fed into a supervised classification model. The automatically learned features are extracted from the pretrained encoder of a deep learning model so that the effectiveness of transfer learning is underlined. The performance of the classification using the proposed feature sets are then compared with the results of existing related work, highlighting the clear advantage of using features encoding depth information.

Furthermore, *Educational Data Mining* is an attractive interdisciplinary domain in which the main goal is to apply data mining techniques in educational environments in order to offer better insights into the educational related tasks. To analyse the relevance of three *unsupervised learning* models, *self-organizing maps*, *principal component analysis* and *relational association rule mining* in the context of students' performance prediction, we will perform experiments on a real academic data set in order to highlight the potential of *unsupervised learning* models for uncovering meaningful patterns within educational data, patterns which will be relevant for predicting the students' academic performance.

This thesis is structured in 4 chapters. We introduce the general domains and present some theoretical aspects in the first chapter. Then we continue to discuss the research carried out on the academic data set collected from Babeş-Bolyai University. The next chapter introduces our original contribution in computer vision regarding depth-maps relevance in the context of image indoor-outdoor classification. Afterwards, an application for showcasing the importance of deep learning models in computer vision is described. In the end, we outline a short overview of the whole thesis' results and provide future research directions.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Ciubotariu George

Contents

List of Figures

List of Tables

Introduction

The main objective of the thesis is to highlight the relevance and importance of applying *unsupervised learning* methods as a preliminary step before developing predictive models. Two application domains are selected for achieving our goal: *Computer Vision* (CV) and *Educational Data mining* (EDM). In both CV and EDM domains we used methods of extracting lower- and higher-level features from our data. However, CV tasks are well known for their densely packed information, therefore we have also studied supervised DL models' performance in our experiments. Nonetheless, we can see that the common denominator of the tasks performed in both domains is the relevance of unsupervised learning based methods when studying classification problems using clustering algorithms in order to expose separation and relevant patterns in data.

One of the main areas of interest in *Computer Vision* is *Semantic Segmentation* (SS) which is commonly achieved using end-to-end supervised *Deep Learning* (DL) models. The process outputs another image, the mask, which identifies certain objects by having different colours for each class. Furthermore, *Depth Estimation* (DE) uses *DL* models that output a mask. The two aforementioned tasks use similar architecture and use images as input and output. Therefore, the models' inner algorithm is task-independent, in what concerns *semantic segmentation* and *depth estimation*. In this thesis we will discuss an approach of highlighting the relevance of depth-maps features in the context of indoor-outdoor classification with the help of handcrafted and automatically extracted features using unsupervised models and transfer learning with deep learning models. The experiments performed on DIODE data set using *Unsupervised Learning* (UL) and *Supervised Learning* (SL) algorithms assesses the usefulness of depth information in computer vision tasks by displaying consistent improvements in the area of the problem. DIODE is a complex indoors and outdoors data set that would challenge networks to learn visual cues in the different environments and reveal the best performing task-independent model. The experimental setup consists of a supervised deep learning model having an encoder-decoder architecture that has been pretrained for performing both semantic segmentation and depth estimation. We extract the features from its encoder and via transfer learning we perform the classification task using a lightweight multilayer perceptron model to emphasize the benefits of transfer learning. The importance of depth-maps does not simply limit to indoor-outdoor classification, as the potential of enhancing the performance of other visual tasks is clearly visible and we are going to put forward the advantages of using such feature augmentation. SS learnt features significantly benefit from depth information and also lower-level features that include depth show an increase in performance in both UL and SL compared to their raw counterparts.

The main purpose of the EDM domain is to provide additional insights into the students' learning process and thus to offer a better comprehension of the educational related phenomena. This thesis investigates the usefulness of *unsupervised machine learning* methods, particularly *principal component analysis* and *relational association rule mining* in analyzing students' academic performance data, with the broader goal of developing supervised learning models for students' performance prediction. Experiments performed on a real academic data set highlight the potential of *unsupervised learning* models for uncovering meaningful patterns within educational data, patterns which will be relevant for predicting the students' academic performance.

The main contributions of the thesis are summarized in the following.

- We investigate the usefulness of *unsupervised machine learning* methods, particularly *principal component analysis* and *relational association rule mining* in analyzing students' academic performance data [CCCD20]. The relevance of *self-organizing maps* and *relational association rule mining* in the context of students' performance prediction was analysed in [CC19]. Experiments performed on a real academic data set highlight the potential of *unsupervised learning* models for uncovering meaningful patterns within educational data, patterns which will be relevant for predicting the students' academic performance.

- In our latest work [CTC21] we have studied the importance of including depth-maps features in visual tasks whether they are unsupervisedly or supervisedly performed. The results of our experiments show a significant increase in performance when considering both lower-level handcrafted features and higher-level features extracted by DL models with the help of TL from DIODE data set.

The rest of the thesis is structured as follows. Chapter 1 introduces our means and methods that have been used in the performed experiments and displays several related articles that we used as reference for our research direction. Chapter 2 presents our original contribution in developing unsupervised learning models for students' academic performance prediction. Chapter 3 presents another original contribution in enhancing computer vision tasks by using features from depth-maps. A mobile application for showcasing deep learning models performance for semantic segmentation and depth estimation is described in Chapter 4. The conclusions of our thesis and directions to further continue and improve our research are given in Chapter 4.3.3.

List of publications

[CC19] **George Ciubotariu**, Liana Maria Crivei (2019). *Analysing the academic performance of students using unsupervised data mining*. Studia Universitatis Babes-Bolyai, Informatica 64, pp. 34–48. (**indexed Mathematical Reviews**)

[CCCD20] Liana Maria Crivei, Gabriela Czibula, **George Ciubotariu**, Mariana Dindelegan. *Unsupervised learning based mining of academic data sets for students' performance analysis* (2020). IEEE 13th International Symposium on Applied Computational Intelligence and Informatics, SACI 2020, Timisoara, Romania, pp. 457-462. (**indexed Web Of Science**)

[CTC21] **George Ciubotariu**, Vlad-Ioan Tomescu, Gabriela Czibula. *Enhancing the performance of image classification through features automatically learned from depth-maps* (2021). 13th International Conference on Computer Vision Systems, ICVS 2021, Virtual Conference, pp. 1-12, submitted. **indexed Web Of Science**

Chapter 1

Background

Computer Vision (CV) aims to cover a wide range of visual tasks in order to automate the processes of decision making in domains such as *autonomous driving*, *robotic process automation*, *product quality control*, or creating virtual environments for VR/AR. Recently, all CV tasks are performed using *Deep Learning* (DL) models that consist of multiple types of layers for processing input images at different resolutions. The most popular architecture in image processing is the one of *Deep Convolutional Neural Networks* (DCNN) which has the *convolution* as core concept. The aim of such networks is to encode pictures' spatial information with the help of convolutions while increasing or decreasing the resolution of images. Such convolutions are intensely researched in order to maximise the amount of information extracted and minimise the computational cost of network training.

We have extensively studied multiple state-of-the-art architectures in order to compare their strengths and choose the most suitable one for the experiments we performed.

A recent article has proposed a different approach to convolutions. Instead of using standard kernels, it uses pyramidal convolution kernels which are able to process the input at different scales for capturing multiple levels of detail in the scene, without increasing computational cost and parameters of a standard convolution [DLZS20]. Another approach tackles *Semantic Segmentation* (SS) by focusing on the distribution of the object classes in urban scenes. Instead of changing the way a DCNN works, it uses a backbone with their own adds-on that processes data according to the vertical position of each pixel [CKC20].

Nonetheless, designing models is of great interest and other works such as [YYC$^+$20, ZWZ$^+$20] have come up with new processing blocks. The first approach redesigns tightly coupled transformations of a block by making them independent while backpropagating. This results in the *Disentangled Non-Local* (DNL) block which achieves great potential in visual tasks learning. The latter approach defines a *Split-Attention* block that improves cross-feature interactions capturing while also being more efficient and less computationally intensive than other networks.

On the one hand, research focuses mostly on DCNNs as they still present interest and there are lots of improvements to be made when talking about multi-layer hierarchical feature extractors. On the other hand, there are implicit deep networks and deep equilibrium networks that can simulate infinitely deep models. An example of such a model class is *Multiscale Deep Equilibrium Models* (MDEQs) that do not provide explicit computation graphs, therefore forward and backward propagation have independent paths that drive the model to equilibrium. Moreover, this kind of model consists of only one layer that can simultaneously learn multiple tasks at once while having constant memory footprint during training [BKK20].

The five models discussed in this thesis [BKK20, CKC20, DLZS20, YYC+20, ZWZ+20] have been chosen to represent different approaches in SS while achieving state-of-the-Art results on different data sets. The contributions of all the models tackle diverse problems and have been brought to different stages in the computation pipeline of a building a deep model. Thus, a consistent set of models was comparatively analysed together with the other mentioned approaches and we concluded which is the best choice for our problem.

1.1 Machine learning models used

1.1.1 Supervised Learning Models

DPT

One of the used deep learning models is called *vision transformers* (DPT). The approach introduced by Ranftl et al. [RBK21] leverages visual transformers instead of convolutions. The main advantage of this change is that compared to convolutions which have a limited spatial understanding due to their size and need downsampling to capture such information, *DPT* has included global receptive field information in each layer. Therefore, this leads to finer, sharper and more coherent predictions, establishing a new state-of-the-art in DE and *semantic segmentation* (SS) [RBK21]. Virtually all recent research articles on TL towards DE use pretrained ImageNet backbones. Having that said, the model represents a good platform for transfer learning thanks to it better feature extracting characteristics that would ensure a consistent comparison between the results of the experiments we performed.

MLP

The second supervised classification model we used is a simple, lightweight neural network model, the *Multilayer Perceptron* (MLP) [AC20]. The MLP classifier was used for indoor-outdoor image classification. The goal of the supervised learning analysis is to strengthen the interpretation of the unsupervised learning one through the evaluation metrics used for assessing the performance of the MLP classifier. It has been used in all the performed experiments in order to ensure a fair comparison between the proposed sets of features and also underline the concentrated yet discriminative nature of the features enhanced with depth information.

1.1.2 Unsupervised learning models

Unsupervised learning models are known in the ML literature as *descriptive* models, due to their ability to detect how data are organized. Unsupervised learning algorithms receive only unlabeled examples and learn to detect hidden patterns from the input data based on their features. UL methods are useful for discovering the underlying structure of the data.

Relational association rule mining

Ordinal association rules (OARs) [CcTM06] represent a a particular class of *association rules* [CLCH16] able to express ordinal relationships between the features characterizing data instances. *Relational association rules* (RARs) [CcM06, SCC06] extend OARs by expressing various types of relationships between data features (attributes).

The RAR notion is defined in the following paragraphs. We consider $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ a set of *instances* or *records* and $\Omega = (a_1, \ldots, a_m)$ a sequence of m attributes characterizing each instance from the data set $\mathcal{D}$. The attributes a_i take values from a domain Δ_i. By $\Psi(d_j, a_i)$ we denote the value of a_i for the instance d_j and by $\mathcal{T}$ the set of all binary relationships which can be defined between the domains Δ_i and Δ_j.

Definition 1 *A* relational association rule *[SCC06] R represents an expression $(a_{j_1}, a_{j_2}, a_{j_3}, \ldots, a_{j_h}) \Rightarrow (a_{j_1}\tau_1 a_{j_2}\tau_2 a_{j_3} \ldots \tau_{h-1} a_{j_h})$, where $\{a_{j_1}, a_{j_2}, a_{j_3}, \ldots, a_{j_h}\} \subseteq \Omega$, $a_{j_k} \neq a_{j_p}$, $1 \leq k, p \leq h$, $k \neq p$ and $\tau_k \in \mathcal{T}$ is a relation over $\Delta_{j_k} \times \Delta_{j_{k+1}}$.*

a) If $a_{j_1}, a_{j_2}, \ldots, a_{j_h}$ are non-missing in q records from $\mathcal{D}$ then $s = \frac{q}{n}$ is called the support *of R.*

b) If $\mathcal{D}' \subseteq \mathcal{D}$ represents the set of records where the attributes $a_{j_1}, a_{j_2}, a_{j_3}, \ldots, a_{j_h}$ are non-missing and all the relationships $\Psi(d, a_{j_1})\tau_1\Psi(d, a_{j_2})$, $\Psi(d, a_{j_2})\tau_2\Psi(d, a_{j_3})$, $\ldots$, $\Psi(d, a_{j_{h-1}})\tau_{h-1}\Psi(d, a_{j_h})$ hold for each $d \in \mathcal{D}'$ then $c = \frac{|\mathcal{D}'|}{n}$ is called the confidence *of R.*

The RARs whose *support* and *confidence* are greater than or equal to given thresholds are called *interesting* [SCC06]. For uncovering all interesting RARs of any length within a data set the $DRAR$ algorithm (*Discovery of Relational Association Rules*) was proposed [CBC12]. $DRAR$ is an Apriori-like algorithm consisting of a RAR generation process that starts from the 2-length rules which are filtered such that to preserve only the interesting rules. The iterative process continues with 3-length rules, then with 4-length rules and so on. At a certain iteration, the RARs of length n are generated by joining [SCC06] interesting RARs of length n-1. The obtained set is then filtered for preserving only the interesting n-length rules. When no new interesting RARs are identified at a certain iteration, the generation process stops.

Principal component analysis

Principal component analysis (PCA)[GBC16] is an unsupervised learning algorithm used for reducing the dimensionality of a set of input data points, by learning a lower dimensional representation of the data while preserving as much information about the the original data distribution as possible in the reduced data set. The PCA algorithm uses an orthogonal transformation for mapping the high dimensional input data into a lower dimensional output space represented by linearly uncorrelated variables known as *principal components* [GBC16]. The aim of the transformation is to capture in the output space as much variance as possible in the original data set. The goal of the PCA algorithm is to map a set of points $Y = \{y_1, y_2, \ldots, y_p\}$ ($y_i \in \mathbb{R}^k$) into a set $Y' = \{y'_1, y'_2, \ldots, y'_p\}$ ($y'_i \in \mathbb{R}^m$, where $m < k$. Thus, the algorithm aims at optimizing an encoding function $f : \mathbb{R}^k \to \mathbb{R}^m$ and a decoding function $g : \mathbb{R}^m \to \mathbb{R}^k$, such that $f(y) = y'$ and $y \approx g(f(y))$.

Self-organizing maps

Self-organizing maps (SOMs), also known in the literature as *Kohonen maps*, were introduced by Teuvo Kohonen and are *unsupervised learning* models widely used as tools for visualizing high-dimensional data. SOMs are connected to the *artificial neural networks* (ANNs) in literature and to *competitive learning*. In *competitive learning*, the output neurons compete themselves to be activated. A *self-organizing map* [SK99] is trained using an unsupervised learning algorithm (Kohonen's algorithm) to map, using a non-linear mapping, the continuous input space of high-dimensional instances into a discrete (usually two-dimensional) output space called a *map* [EDB08]. The *topology preserving mapping* is the main characteristic of the mapping provided by a SOM. This means that the input samples which are close to each other in the input space will be also close to each other on the map (output space). Thus, a SOM is able to provide clusters of similar data instances [LO92].

t-Distributed Stochastic Neighbor Embedding

In our approach we are applying 3D t-SNE (*t-Distributed Stochastic Neighbor Embedding*) [vdMH08] for *non-linear* dimensionality reduction of the m-dimensional representation of the images from the data set, computed according to the representations that will be presented in 3.1. t-SNE is used in exploratory data analysis for uncovering patterns in data useful for clustering. The model uses *Student t-distribution* [Mar12] to better disperse the clusters. This clustering method is better than PCA due to its ability to extract non-linear features, therefore exposing more useful characteristics in the data. In the domain of computer vision, a thorough analysis of the images we used is necessary to unveil meaningful details, hence we used t-SNE.

1.2 Related Work

1.2.1 Indoor-outdoor image classification

An approach to indoor-outdoor classification that relies on edge analysis is [PS05]. The assumption made by the authors relies on the nature of the artificial and natural shapes of the objects, as they focused on the straightness of region edges in order to classify the images. Each point is assigned a straightness score based on the edge features. The method achieves 87.7% accuracy.
Szummer et al. [SP98] introduced a model independently processes both color and texture features. The two separate classifications are then put back together for the final classifier to reach 85% accuracy. A more recent approach uses HSV (H-Hue, S-Saturation, V-Value) instead of RGB color encoding, as they state that it is invariant to scale and illumination.
The method proposed by Raja et al. [RRDR13] uses color, texture and entropy features that a KNN classifier uses, achieving a performance of 81.55% accuracy.
Furthermore, Cvetkovic et al. [CNI14] uses a combination of multiple color and texture descriptors and a SVM classfier to obtain accuracies of 93.71% and 92.36% on two public data sets, while Tahir et al. [TMR15] computes the GIST descriptor of the scene as a feature vector. A custom neural network classifier with one hidden layer then achieves 90.8% accuracy.
A review on indoor-outdoor scene classification, feature extraction methods, classifiers and data sets is done by Tong et al. [TSYW06]. The features analysed include color, texture, edge, as well as others, while the classification was done using machine learning methods, both classical approaches and deep learning, or by employing a method called bag of word (BoW), specific to computer vision. Multiple data sets were mentioned, but not DIODE, mainly due to the recent nature of the data set. The paper concludes that there is no ideal set of features for the indoor-outdoor classification task, but that deep learning classifiers have an edge over other models.

1.2.2 Semantic segmentation

DNL

Unfortunately, non-local blocks do not learn visual clues well because the unary term representing the saliency of every pixel and the whitened pairwise term accounting for the relationship between two pixels are tightly coupled which hinders their learning as seen in the back-propagation of loss which can train only one of the two terms' understanding of visual clues due to vanishing gradient issues.
Therefore the Disentangled Non-local Neural Networks were introduced which effectively detach the learning of pairwise and unary terms.
In order to disconnect the two terms, multiplication was changed to addition when computing similarity of pixels so the gradients of the two terms would be now independent to the other, they changed the shared key transformation in the unary term to be an independent linear transformation so that the two terms are further decoupled. Therefore, DNL has significantly larger overlap with the ground-truth compared to the standard NL network, which indicates better learning of the two visual clues by the DNL block in comparison to the standard NL block.
This method proved to be effective as DNL is superior to NL and other random-attention networks thanks to the combined learning of visual clues that result in clearer within-region meaning and more focused regions for semantic segmentation tasks.
The experiments were performed on three benchmark data sets: Cityscapes, ADE20K and PASCAL-Context. The architecture used consists of a ResNet-101 backbone with the DNL block inserted before the final classifier. Hence, DNL achieves on Cityscapes test 82.0% mIoU using ResNet-101 and 83.0% mIoU using HRNetV2-W48, on ADE20K test 56.23% mIoU using ResNet-101 and 55.98% mIoU using HRNetV2-W48, obtaining state-of-the-art results and on PASCAL-Context test 54.8% mIoU and 55.3% mIoU respectively.

HANet

Urban-scenes are structured on a specific own geometry, having multiple patterns that occur in different places according to the height of the image. Consequently, their segmentation is highly dependant on y-axis positional distribution of entities, hence a potential solution to enhance a models' output is to split the image into three attention zones according to the occurrence probability of the classes of objects in each row, as class distribution is exceptionally unbalanced. Therefore, the HANet lightweight add-on module was proposed in order to improve performance with negligible computational and memory overhead, by exploring the inter-channel relationship of features and determining the importance of each channel, according to which their activation is decided. Therefore, HANet extracts height-wise contextual information and computes the likelihood of different classes of objects to appear in certain image rows.
As [fig x] presents, the input feature maps are segmented using the main networks into higher-level feature maps, ”Xh” and also processed through their proposed method. The input undergoes width-wise pooling, interpolation for coarse attention, 1x1 convolutions while the positional encoding is introduced to the feature maps and finally upsampling which results in the height-driven channel-wise attention map, ”A”, being obtained. This final attention map is used to tell which channels are the most important for each row. As there can be more than one feature associated with every row, a sigmoid function computes the

attention map instead of a softmax. In the end, ”A” and ”Xh” are multiplied element-wise which results in ”X̃h”, the new transformed feature map.
The positional encoding incorporated into the attention map computation while applying 1x1 convolutions refers to human-like prior knowledge of the scene layout, especially on the vertical distribution of each class of objects. These encodings are represented as sinusoidal values dependent on the vertical position, the attention map index and the channel dimension, so that when introduced, positional information is summed element-wise with an intermediate feature map.
In the experiments, HANet is added at five layers, after the backbones outputed their high-level information encoding, as these contain more vertical positioning clues so that HANet optimisation is more effective. Best backbone proved to be DeepLabv3+ ResNet with PyTorch SyncBatch-Norm and a replaced 7x7 convolution by three 3x3 in its first layer. The results confirm the significant efficiency increase and negligible cost of adding HANet and achieves state-of-the-Art performance with 82.05 mIoU(%) on Cityscapes validation set and 65.60 mIoU(%) on BDD100K validation set.

pyconvsegnet

Most Computer Vision models have CNNs integrated into their architecture that use 1x1 or 3x3 convolution kernels. However, these relatively small kernels cannot always encode the complex context of a picture and downsample operations have to be applied to capture more of the receptive field. Unfortunately, downsampling is a lossy operation through which useful context information can be lost.
Indoors and outdoors images are both complicated due to their uneven distribution of entities and crowded or overlapping spaces of objects of varying shapes and scales.
The paper mentions the difficulty of CNNs to understand the information because using a single type of convolution for all classes of objects would be ineffective for capturing such complex data.
Therefore, to address these problems PyConv is introduced, a neural network that is composed of multiple kernels of varying size and depth in order to be suited for every details that appear in images. PyConv enlarges the receptive field and still maintains the same complexity as standard convolutions while being flexible and extendable.
PyConv uses networks from the ResNet family as baselines. It uses ResNeXt-like grouped convolution but in a different architecture.
Compared to PPM or ASPP architectures, PyConv represents a novel head for parsing the feature maps provided by the ResNet backbone. In order to parse the input, it uses a local multi-scale context aggregation module and a global multi-scale context aggregation block, therefore having a lighter computational complexity than other architectures, thus the advantage of multi-scale processing. Due to the pyramidal convolutions, the network gradually focuses its attention on objects of any size, gathering both local and global visual clues which combined help boost the learning process. The shape of the kernels can vary depending on the goal of the model, but the constraint is that the number of groups for each pyramid level and the number of input and output feature maps are a power of 2. Nonetheless, they help with complementary information collection which makes PyConv so flexible and efficient.
For semantic segmentation, the PyConv Parsing Head which they introduced is able to encode both local and global visual clues thanks to the aforementioned qualities of the architecture. PyConvPH is composed of three parts: Local PyConv block which is best used for

gathering fine details, Global PyConv block which better understands the scene as a whole and its contextual information due to the large 9x9 convolutions, and Merge Local-Global PyConv block which is responsible with processing the information of the previous two components in order to prepare it for classification.
The performed experiments prove the network's efficiency and accuracy compared to PPM and ASPP models and also highlights its potential to be used in other visual recognition tasks by achieving state-of-the-Art results on scene parsing, with PyConvSegNet-152 scoring 45.99% mIoU on ADE20K validation set.

ResNeSt

ResNeSt consists of a simple architecture based on channel-wise attention and multipath layout. The main interest is the Split-Attention Block whose transformation apprehend cross-channel feature correlations. The network performs better than other previous work in what regards quality and latency as shown in multiple experiments, including transfer learning. Moreover, ResNeSt has served as backbones for the winning entries of 2020 COCO-LVIS.
In a cardinality-major implementation, the Split-Attention block consists of sections called featuremap groups, also referred as cardinal groups and split attention operations. The cardinals contain information organized in splits which send their outputs to the split-attention unit where they are aggregated to obtain the representation of each group, then the values are put together to get the whole result.
However, in the radix-major implementation that ResNeSt uses, the procedures are optimised using CNN operators. The main differences are that instead of the input being separated into cardinals ”k” having ”r” splits within, it is divided into ”r” splits having ”k” cardinals. Therefore, concatenated results undergo another step of aggregation and processing. This makes it possible for the Split-Attention block to be modularised and optimised.
In what regards transfer learning, especially for segmentation where it is essential to protect positional clues, a 3x3 average pooling layer is used instead of a strided convolution one. In addition to this, the 7x7 convolution is replaced by three 3x3 in its first layer and a 2x2 average pooling layer is added prior to the 1x1 convolutional layer for the stride-2 downsampling blocks, as implemented in ResNet-D.
While performing the experiments, the split-attention is employed using a radix index of 2, cardinal index of 1 and a width of 64 to ensure scalability for deeper networks. When radix index is 1, the Split-Attention block applies squeeze-and-attention to each cardinal comparable to SE-Net. When radix index is 2, the attention block applies transforms similar to SK-Net. From the experiments it emerges that the use of more Split-Attention modules results in a better performance. ResNeSt is used as backbone for the implementation of DeeplabV3, yielding a score of 80.42% mIoU on Cityscapes validation set using ResNeSt-101, 82.70% using ResNeSt-200, 46.91% mIoU on ADE20K validation set using ResNeSt-101 and 48.36% using ResNeSt-200. Overall, the network outperforms all prior models having ResNet backbones. Consequently, ResNeSt achieves state-of-the-Art results in semantic segmentation as proved in the experiments on the two data sets.

1.2.3 Depth estimation

Depth estimation (DE) [LHKS19] or *depth extraction* aims towards gathering information about the structure of a scene using mathematical or learning means, hence every pixel of an

image is supposed to be assigned a depth value. Stereo vision methods are highly dependent on illumination intensity and angle, therefore monocular depth estimation is more convenient for calibration or identification than stereo vision means. *Monocular depth estimation* (MDE) is a difficult task as even humans misinterpret *Depth of Field* when looking with one eye open.
We are going to focus mainly on *Monocular Depth Estimation* (MDE). The difficulty of determining the depth in a single image is an ill-posed one as stated in the literature [Bho19], as we have to rely on features such as shapes, textures and occlusions in order to gain information about the scene geometry and different semantic meaningful regions that would help us compute the depth value of every pixel in an image. The great difficulty in this situation is that one 2D photo is not enough to reconstruct a 3D environment as there are infinite 3D layouts that can be projected into a 2D space to reproduce an image. As optical illusions have proven us, humans make assumptions about the shapes and sizes of the objects, ultimately leading them to the wrong conclusions. Still, that assumptions are also to be made by learning DL models when dealing with so little spatial information. Nonetheless, if depth information would be available, we could therefore use the advantages of combining RGB features with depth cues to enhance visual tasks, by offering the models extra guidance. All SOTA papers use CNNs and treat the monocular depth estimation problem as a regression problem.

How is DE performed

Geometry-based DE mainly acquires its depth information from sequences of frames or dual cameras. Therefore 3D structures are recovered from images with the help of *Structure from Motion* (SfM) or stereo vision matching, each of them having their own advantages. However, those methods cannot be used for single image MDE. As a result, *Machine Learning* end-to-end solutions were used to solve this ill-posed problem by exploiting the geometric relationships between particular regions of the scene. Almost all recent studies utilise the DCNN encoder-decoder architecture in order to extract higher level features maps from the images and down- and up-sample them in order to gain more spatial and contextual information.

Supervised Learning DE

The novel DPT model [RBK21] that we have used in our experiments leverages *vision transformers* as the core concept of processing instead of convolutions. The main advantage of this change is that compared to convolutions which have a limited spatial understanding due to their size and need downsampling to capture such information, *DPT* has included global receptive field information in each layer. Therefore, this leads to finer, sharper and more coherent predictions, establishing a new SOTA in DE and SS.

Bhat et al. [BAW20] pose the question of how well global information processing improves the results. Therefore, a transformer based architecture block is proposed. Its purpose is to divide the depth values into bins. The bins' center value is dynamically estimated according to each input image. The new building block is named AdaBins.

It is stated that MDE is an ill-posed problem and difficult to compensate for the loss of depth perception from stereo vision. The motivation of employing a novel local planar guidance layer lays in the observation that humans not only use local cues but also global context for depth estimation. These layers are used at each decoding stage in the network that encode information which is later combined to predict depth in full resolution [LHKS19].

A recent approach states that some other networks probably fail to exploit all the encoded features and therefore it proposes a scheme that uses the Laplacian pyramid for improving the decoding step. Outputs from the decomposition of the pyramid are gradually combined to obtain finer results as depth boundary and global layout estimation are more effectively computed [SLK21].

Another usage of DE was highlighted in [NMYL19], as 3D Ken Burns effects are popular nowadays for creating DoF resemblance in 2D images. As creating such effects require complex image editing aptitudes and lots of time, a semantic-aware neural network is used in the DE pipeline. The approach renders the *Point Cloud* from multiple camera positions for creating such 3D effects. Moreover, for missing information context aware color- and depth-inpainting and refinement of object boundaries is necessary.

The importance of higher-order 3D geometric constraints in the scene space is studied in [YLSY19], by designing a new loss function that computes the direction of *virtual normals* by using three non-colinear points in the generated scene. As a result, normals can be extracted directly from depth instead of having to train a separate model for that task.

Self-Supervised Learning DE

Recent work states that rigid photometric constraints have limited usability in DE, therefore [BK20] regresses disparity in an exponential domain, given their novel *Mirrored Exponential Disparity* probability volumes which improve stereo depth estimation edge cases handling. This method facilitates obtaining more consistent occlusion maps via the multi-view *Mirrored Occlusion Module* which fine-tunes the network for DE in the second stage of the view synthesis and *Single Image Depth Estimation* (SIDE) training process.

A recent approach features an appearance matching loss that robustly handles occlusions, a sampling method that considerably improves results consistency and an auto-masking loss that disregards pixels that do not respect motion assumptions. All the components have been tested in isolation to prove their effectiveness and when put together they set a new SOTA in monocular and stereo self-supervised DE [GAFB19].

Another recent approach states that self-supervised based DE models that use photometric consistency have problems concerning scale consistency as this type of supervision is fragile in a real-world environment. The problem is solved by imposing geometric constraints on adjacent frames so that the generated depth maps' forward-backward relative pose are consistent in order to achieve scale consistency [XZZ+20].

The problem of high-resolution image depth estimation improvement has been tackled in [LLW+20]. By reducing the bilinear interpolation errors via redesigning skip-connections in order to obtain finer images and using a module to better fuse the features, the proposed model outperforms others methods for both low- and high-resolution DE.

In replacement to photometric loss, a feature-metric loss was proposed in [SYDY20], because the first one poses difficulties in optimising less discriminative pixels comprehension. The novel approach sets a new SOTA for depth estimation.

Unsupervised Learning DE

Unsupervised tasks have also a widespread use on multiple domains. Generally, unsupervised learning is used for uncovering meaningful patterns in *Data Mining* with the broader goal of eventually developing supervised models for the same task. The great advantage of unsupervision is that it can use unlabeled data and instead of using a comparison to a ground truth labeled prediction, it can just learn by being consistent in making assumptions, as there are many ways someone can fail, but there is only one way to be correct. Consequently unsupervised CV models benefit from learning from any available data on the internet, which leads to better overall performance.

In our previous work [CC19, CCCD20] we have highlighted the potential of unsupervised learning in student academic performance prediction and better comprehension of the students' learning mechanism. Furthermore, in our latest paper [CTC21] we have used the t-SNE unsupervised clustering algorithm for emphasizing the relevance of features extracted from depth-maps in the context of indoor-outdoor image classification.

Instead of building dense and complex networks that require lots of memory, [PATM18] introduced an architecture trained unsupervisedly that treats DE as an reconstruction problem. The model computes depth on the CPU, using a pyramid of extracted features, which is able to perform well even on embedded systems.

A recent unsupervised approach proposes a *feature pyramid matching loss* that evaluates the differences between sequences of photos with better results than photometric error and an *occlusion-aware mask network* which improves occlusion estimation and camera pose by observing the changes of the mask [GCZW21].

1.2.4 Transfer Learning

Transfer Learning is a method of having a model's useful information achieved from learning a specific task used for training the model on a goal that is related to the initial task but different.

All the work so far focuses on using pretrained models on ImageNet data set [DDS+09] because it is so large and it has a wide variety of objects types, allowing backbones trained on it to enhance the performance visual tasks such as SS. The benefit brought by information encoded from OD consists of having pretrained weights that gather semantic meaning of the objects' features, but no spatial information.

However, there are not any backbones used in TL trained on more complicated tasks such as SS. Consequently, we chose to tackle not the problem of Transfer Learning, but rather retraining from scratch whole models for performing other tasks than they were designed to, in order to emphasize the potential of different networks and analyse their learning abilities on related tasks. We decided to consider this study one of Transfer Learning, as its focus consists of evaluating how different models' architectures grant task independence and what influences their particularities not to be compatible with other related visual cues.

Recent research show that using deeper models increase the convergence difficulty and that DE methods for indoor do not perform great in outdoor environments. Therefore, an approach tries solving these problems by describing a training strategy that uses TL and *Ordinal Regression* to improve convergence speed. The motivation consists of the observation that in human perception, we tend to selectively focus our attention on only a part of the scene rather than the whole. The contribution consists of designing a spatial-channel dual attention module that emphasises meaningful features and treating the DE problem as a regression and classification one depending on the input data [YHLC20].

A rather lightweight CNN approach for DE was introduced in [HQC+20]. TL leaded by geometric guidance proved to employ better feature extraction when computing the depth maps. The coarse results are then fused in a two-stream encoder–decoder architecture to jointly estimate normals. Afterwards, the depth maps are postprocessed by using surface normal guidance.

In what regards self-supervision, [WCG+20] proposes a uncalibrated video-based learning that uses geometric-based TL that result in more accurate and sharper depth maps. This method collects structural information from a large number of videos by using a conditional autoencoder-decoder architecture.

A different approach introduces a novel multi-task framework that uses *Multispectral Transfer Learning* to estimate depth from a single thermal image. The model uses geometric priors and chromaticity clues to unsupervisedly learn DE. Moreover, their proposed multi-task module *Interleaver* successfully uses chromaticity and skip-connections details for DE, while keeping layer independence [KCHK18].

An article that has an approach similar to ours regarding TL is [CMM20], in which a novel *Serial U-Net* (NU-Net) is introduced that could potentially use *N U-Nets* one after another. They hypothesize that the proposed architecture could support *N-1* pretrained *U-Nets* for different visual tasks and the *N-th U-Net* to be the single one that learns a specific tasks with the help of previous ones' accumulated knowledge. Moreover, their experiments present the use of *W-Nets*, which are *2U-Nets* with the first *U-Net* pretrained on ImageNet data set [DDS+09] and the second able to learn DE.

Another presented experiment focuses on the use of *W-Nets Connected* which benefit from an added connection between the input RGB image and the segmented image generated by the first *U-Net* whose weights are pretrained on OC and frozen. The *W-Nets Connected* performs the best out of the *U-Net*, *W-Net* and *W-Net Connected* models.

However, as they state, they are not sure if the improvement in performance is due to using the first *U-Net* in the sequence for SS or it's because of the deeper architecture. Furthermore, the networks perform pretty accurate segmentations but can misinterpret depth. Hence, their work is not conclusive on whether SS does improve the results of DE, since only a pretrained SS model on ImageNet would probably not be sufficient to improve DE, as the task itself introduces new errors in the computational pipeline and a more direct approach could outperform the *Serial U-Nets*.

Another article similar to ours in what regards the direction of SS correlation with DE uses a synergy network for simultaneously learning SS and DE by information sharing between two computational heads guided by an attention module. They prove that the weight sharing between the sub-networks is of great advantage, which further supports our point that *Semantic Segmentation* can be used to improve *Depth Estimation*. The main difference between our approaches is that [JCSL18] splits the encoded features into two sub-networks that perform SS and DE in collaboration and we are studying the relationship between models designed exclusively for segmentation and their depth estimation potential.

1.2.5 Educational data mining

Extracting relevant patterns from the educational processes could be effective for understanding students and their learning methods, as well as improving the educational outcomes (e.g. learning outcomes). EDM is of major interest for the research community since mining knowledge from educational related data is of particular interest for academic institutions as it may be useful for improving the teaching methodologies and learning processes [MT13]. Various applications using data mining techniques have been developed, so far, in the EDM domain. *Machine learning* methods are widely investigated, both from a *supervised* and *unsupervised* perspective, as data mining techniques for developing course planning systems, predicting the students' performance for courses, detecting what type of learners are the students, grouping students according to their similarities, assisting instructors in the educational process [JRHR15].

We briefly review, in the following, several unsupervised machine learning approaches which have been developed for analyzing data related to the performance of students in educational environments. Various applications using data mining techniques have been proposed, so far, in the EDM field. From a supervised learning perspective, various learning models have been investigated: decision trees (DT), Naive Bayes, artificial neural networks [JRHR15], radial basis function networks [OTOK15], linear regression and support vector machines [TDD$^+$17]. Despite the numerous existing approaches, predicting the students' performance is a difficult task. The best performance achieved so far in the EDM literature is an F-score of 0.8 using DTs [PP13] and an accuracy of 0.85 using J48 [BR18]. Ayers et al. applied in [AND09] clustering methods such as hierarchical agglomerative and partitional clustering for grouping students according to their skill sets. A survey on using *unsupervised learning* methods for various EDM tasks is presented by Dutt et al. in [DAIM15]. Parack et al. used in [PZM12] the $K - means$ clustering algorithm to group students that have similar learning patterns. These groups are further used for identifying related cognitive styles for each group. Kurdthongmee [Kur08] used SOMs as a tool to partition student data into clusters according to their study results.

Chapter 2

New unsupervised learning approaches in Educational Data Mining

This chapter introduces our original unsupervised learning approaches [CCCD20], [CC19] developed for *educational data mining* (EDM).

In [CC19] we analysed the relevance of two *unsupervised learning* models, *self-organizing maps* and *relational association rule mining* in the context of students' performance prediction. The experimental results obtained by applying the aforementioned unsupervised learning models on a real data set collected from Babeş-Bolyai University emphasize their effectiveness in mining relevant relationships and rules from educational data which may be useful for predicting the academic performance of students. Our original paper [CCCD20] investigated the usefulness of *principal component analysis* and *relational association rule mining* in analyzing students' academic performance data, with the broader goal of developing supervised learning models for students' performance prediction. Experiments performed on a real academic data set highlight the potential of *unsupervised learning* models for uncovering meaningful patterns within educational data, patterns which will be relevant for predicting the students' academic performance.

Educational data mining (EDM) represents an interesting research domain in which the major goal is that of uncovering meaningful patterns from data that comes from various educational environments. One purpose of EDM is to offer additional comprehension of the students' learning mechanism and thus to offer a better understanding of the educational processes. Applying *machine learning* (ML) techniques in education [BCR18] is continuously attracting researchers from the EDM domain. *Unsupervised learning* (UL) techniques are extensively applied nowadays in various domains including software engineering, medicine, bioinformatics, financial field, to discover meaningful patterns in data, particularly due to their potential of uncovering hidden patterns.

We start by describing in Section 2.1 with the data set used in our experiments. Then we introduce in [CCCD20] and [CC19] our original approaches developed in the EDM field.

2.1 Data set

The data set used in our evaluation is a real data set, denoted by D, containing the grades received by students at a Computer Science undergraduate course offered at Babeş-Bolyai University in a time frame of six academic years (2011-2017), both at the regular and retake examination session. The complete data set is available at [dat18]. There are a total of 905 instances characterized by 5 attributes, denoted by a_1, a_2, a_3, a_4, a_5. The first four attributes represent scores obtained by students during the academic semester: seminar score (a_1), project score (a_2), project status score (a_3) and written test score (a_4). Attribute a_5 represents the *final examination grade* for a student received after the final examination, at the end of the semester. The data used in our experiments does not include the written examination grade (obtained in the exams session), since we want to test if there is a certain correlation between the grades obtained by a student during the semester and the final examination grade. Thus, in our experiments, attribute a_5 will be used only for evaluation purposes, without being used for building the unsupervised learning models (PCA, RAR and SOM).

Figure 2.1 depicts a histogram of grades (4-10) built on the data set D.

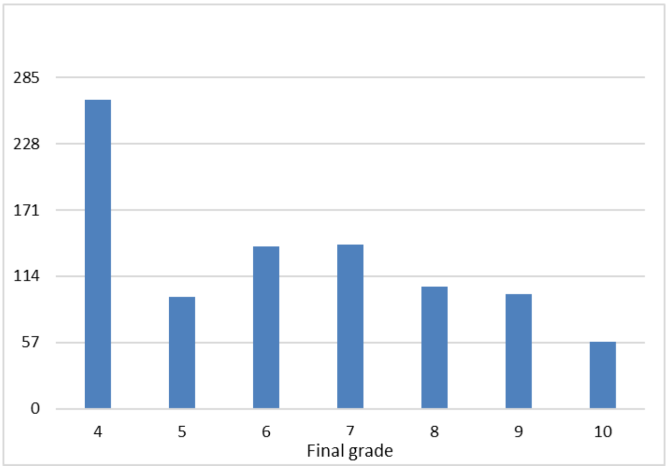


Figure 2.1: Histogram of grades from the data set D.

The histogram from Figure 2.1 reveals a distribution of passing grades (grades without 4) close to the normal one.

For analyzing the relevance of attributes, the Pearson correlation coefficients between features a_1, a_2, a_3, a_4 and the final examination grade (attribute a_5) were computed. A Pearson correlation coefficient of 1 or -1 expresses a linear monotonic relationship between the two variables being compared, while a value of 0 means that there is no linear relationships between the variables. We observed that the features a_1, a_2, a_4 are well enough correlated with the final examination grade (0.66, 0.685 and 0.634), the *project score* (attribute a_2) having the highest correlation with the finale examination grade. The smallest correlation of 0.479 was noticed for attribute a_3.

2.2 Unsupervised learning based mining of academic data sets for students' performance analysis

In this section we are conducting a study towards emphasizing the relevance and importance of using *unsupervised learning* techniques for analyzing the students' academic performance [CCCD20]. Two unsupervised machine learning methods will be further investigated: *principal component analysis* (PCA) and *relational association rule* (RAR) mining. A study similar to ours has not been performed in the EDM literature, so far..

The rest of the section is structured as follows. Section 2.2.1 introduces our experimental methodology, while the results and their analysis are provided in Section 2.2.2. Section 2.2.3 presents the conclusions of our study and indicates several directions for future work.

2.2.1 Methodology

As previously defined, the major goal of our study is to investigate the usefulness of *unsupervised learning* models for analyzing students' academic performance data, with the broader goal of developing supervised learning models for academic students' performance prediction.

Let us consider the following theoretical model. We denote by $\mathcal{S} = \{st_1, st_2, \ldots, st_n\}$ a data set in which an instance s_i represents the performance of a student at a certain academic course $\mathcal{C}$, during an academic semester. The instances are characterized by a set of features (attributes) $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ which were identified as relevant for measuring the performance of the students for the given course, such as the students' grades received during the semester evaluations. Accordingly, each st_i is represented as an k-dimensional vector $st_i = (st_{i1}, st_{i2}, \ldots, st_{ik})$, where st_{ij} expresses the value of attribute a_j for student st_i. The unsupervised analysis does not include the grades of the students' at the written exam (obtained in the examination session), which are also part of their final examination grade. Our aim is to analyse if only the grades received by the students during the semester are enough to discriminate their written exam grade and, accordingly, the students' final examination grade.

The current study aims to investigate if two unsupervised learning models, PCA and RAR mining, are able to identify some patterns which would be useful for determining the *final examination grade* for the students, based on their grades obtained during the academic semester.

Experiments and setup

Two experiments will be conducted on the data set described in Section 2.1. The first experiment consists on conducting a PCA analysis on the data set, for obtaining a better understanding of its complexity (i.e how difficult it is to discriminate between the students having the same final grade). The PCA is used to transform the initial attribute space in a two dimensional space where the two independent principal components are expressing the maximum variance in data. Two PCA visualizations will be provided: one for the entire data set (characterized by all attributes a_1, a_2, a_3, a_4) and the second for the data set without attribute a_3 (characterized only by the attributes a_1, a_2, a_4). We aim to test the hypothesis that by removing the attribute which has the smallest correlation with the final examination grade a better separation between the instances is obtained.

For the PCA analysis applied for the first experiment, the implementation from [Ped11] is used.

The second experiment consists of detecting, using the $DRAR$ mining algorithm, two sets of interesting RARs, namely RAR_- mined from the subset D_- of students with the final examination grade less than 5 (which have not passed the exam) and RAR_+ mined from the subset D_+ of students with the final examination grade greater than 5 (which have successfully passed). We note that $D_+ \bigcup D_- = D$. For the RAR mining experiment, we added 4 additional attributes to the data sets D_+ and D_- ($a_5 = 5$, $a_6 = 6$, $a_7 = 7$ and $a_8 = 8$) and we used the following parameters for the mining process: a minimum support threshold of 1, a minimum confidence threshold of 0.55 and two binary relationships between the data attributes ($\geq$ and $<$). After RAR_- and RAR_+ were determined, the sets were filtered such that $RAR_+ \bigcap RAR_- = \emptyset$. Attributes a_5, a_6, a_7, a_8 were added for extending the set of possible RARs which may be discovered in the data set. The RAR mining experiment is applied both on the entire data set and on the data set after removing attribute a_3.

Evaluation measures

For evaluating the results of the experiments, we are introducing in the following subsection several quality measures.

First experiment. For measuring how well a 2D PCA separates the students according to their final examination grade, we are introducing an evaluation measure $Prec$ that expresses the precision of the PCA mapping. $Prec$ is a measure ranging from 0 to 1, evaluating how "close" are the students with the same output (final grade) on the 2D space provided by the PCA transformation. Higher values for $Prec$ will indicate a better quality of the PCA mapping. Let us denote by y a student in the input space ($\mathbb{R}^4$ or $\mathbb{R}^3$) and by $f(y)$ its final examination grade. By N_k, where $k > 1$, we express the set $\{x_1, \ldots, x_k\}$ of k nearest neighbors of y in the 2D PCA output space. For an instance y, we compute the precision of mapping y in the 2D PCA space, with respect to its k nearest neighbors and a certain threshold τ, $prec(y,k,\tau) = \frac{\sum\limits_{x \in N_k} \delta(f(x), f(y), \tau)}{k + \sum\limits_{x \in N_k} w(f(x), f(y))}$, where $\delta(a,b,\tau) = \begin{cases} 1 & if\ |a-b| \leq \tau \\ 0 & otherwise \end{cases}$ and $w(a,b) = \frac{|b-a|}{10}$. The overall precision $Prec(D,k,\tau)$ is now defined as averaging the precision values for all the instances from D, as $Prec(D,k,\tau) = \frac{\sum\limits_{y \in D} prec(y,k,\tau)}{|D|}$.

We note that the threshold τ is used in computing the distance δ between the final grades (a and b) of two neighboring data points (x and y) and is a natural number used for controlling the accuracy of computing the $Prec$ measure. A value of 0 for τ means that x and y are considered correctly mapped only if $a = b$ and this leads to a maximum accuracy of the computations. Larger values for τ (e.g. 1) weaken the constraint in computing the $Prec$ value, i.e. x and y are considered correctly mapped if $|a - b| \leq \tau$. This leads to a lower accuracy in measuring the quality of the PCA mapping, but provides higher values for $Prec$.

Second experiment. For evaluating how well the uncovered sets of RARs (RAR_+ and RAR_-) characterize the set of students which passed or failed (i.e. D_+ and D_-, respectively) we introduce two evaluation measures $AccM$ and $ErrM$ (*accuracy* and *error* of mining the interesting RARs from the data set D). Let us denote by $RAR = RAR_+ \bigcup RAR_-$.

If R denotes a set of RARs mined from the data set D, by $Acc(R,D)$ we aim to express the degree to which the rules from R cover the instances from D. For a certain relational association rule $rule$, we denote by $acc(rule, D)$ the *accuracy* of the $rule$ in the data set D, computed as the average confidence of all the subrules of $rule$ (including the rule itself), i.e. $acc(rule, D) = \frac{\sum\limits_{r\ subrule\ of\ rule} c(r)}{m}$ (where $c(r)$ denotes the confidence of rule r and m represents the number of subrules of $rule$, including the rule itself). The accuracy $Acc(R,D)$ of a set of rules R with respect to the data set D is now defined as averaging the accuracies $acc(rule, D)$ for all the rules from R: $Acc(R,D) = \frac{\sum\limits_{rule \in R} acc(rule, D)}{|R|}$. The evaluation measure $Acc(R,D)$ ranges in [0,1], higher values expressing that R characterizes better the data set D.

The overall *accuracy of the mining* in the data set D, $AccM(RAR, D)$, will be expressed as the mean accuracy of the sets RAR_+ and RAR_-, i.e. $AccM(RAR, D) =$

$\frac{Acc(RAR_+,D_+)+Acc(RAR_-,D_-)}{2}$. We note that $AccM(RAR, D)$ takes values in the interval [0,1], higher values for $AccM(RAR, D)$ expressing that the interesting RARs mined from D characterize better the data set D.

Besides the accuracy of mining, we also express the $ErrM(RAR, D)$ (*error of mining*) as $ErrM(RAR, D) = \frac{Err(RAR_+)+Err(RAR_-)}{2}$, where $Err(RAR_+) = Acc(RAR_+, D_-)$ and $Err(RAR_-) = Acc(RAR_-, D_+)$. The intuition behind defining the error is that we aim to obtain rules which are able to distinguish very well between classes ("+" and "-") and thus we do not want the rules from RAR_+ to characterize the instances from D_- and viceversa. Obviously, $ErrM(RAR, D)$ ranges in [0,1], lower values indicating that the set R of rules characterized better the data set D.

2.2.2 Results and discussion

This section presents the experimental results obtained by applying the methodology described in Section 2.3.1 and an analysis of the obtained results.

First experiment

Figures 2.2 illustrate the 2D PCA visualization of the data set from Section 2.1 using all attributes a_1, a_2, a_3, a_4, while Figure 2.3 depicts the two dimensional visualization of the data set using only the attributes a_1, a_2, a_4.

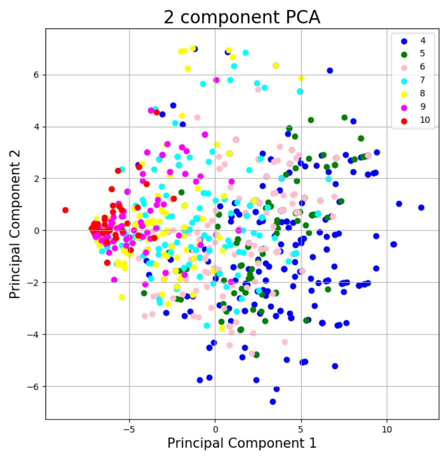


Figure 2.2: 2D PCA visualization of the entire data set D.

On both images, the data points (students) are coloured according to their final grade. The students with the same final grade (depicted on the map) are marked with the same colour. Even if there is no clear separation between the grades, we observe a good enough mapping. In the left side of the image we found mostly students with grades ranging from 7 to 10, while in the right side of the image most of the students have grades from 4 to 7. It can be observed that the most of the students which failed to pass the exam (their final grade is 4) are well separated from the students with high grades (8, 9, 10). Generally, we observe a slow transition between the instances: students with the final grade 4 are neighbors with those having the final grade 5, and so on.

We also observe some anomalies in the data set, such as neighboring instances which are labeled with very dissimilar grades (e.g. 4 and 10). The presence of such anomalies in

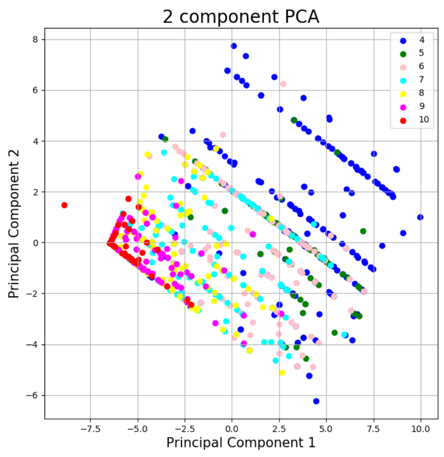


Figure 2.3: 2D PCA visualization of data set D considering the attributes (a_1, a_2, a_4).

data is explainable from at least two perspectives. First, our analyzed data set includes the results of the students in both the regular and the retake session. It is very likely that there are students with high grades during the semester, but which fail to pass the exam from the regular session (due to some unpredictable events). It is very probable that in the retake session these students will pass the written exam and accordingly, will pass the course. Such cases (a student which failed in the regular session and passed in the retake session) are visualized in the PCA plot as anomalies, as two neighboring instances (characterized by the same grades during the semester) may have two very different labels (a small one and higher one). The data set may also include noisy instances and this is also expectable in academic environments. An analysis of the data set revealed several outlier instances for which the grades received during the semester evaluations and uncorrelated with the final examination grade. Such situations might occur due to: (1) a biased evaluation of the activities during the semester (seminar, written test, project); (2) the difference between the passing criterion of various instructors; or (3) unpredictable incidents which may appear during the students' learning process.

The PCA plot from Figure 2.3 obtained without considering the attribute a_3 (the project status score) indicate a slightly better mapping and separation between the students grades than the visualization from Figure 2.2. The slight transition between the grades is better observable in Figure 2.3. Moreover, the number of anomalous and noisy instances observable in Figure 2.2 seems to be reduced when discarding attribute a_3. A possible explanation might be that the project status score (attribute a_3), an attribute with a smaller correlation with the final examination grade, may be a possible cause for the anomalous instances from the PCA visualization. The PCA analysis previously performed denotes the difficulty of the task for predicting the final examination grade for the students, based on the grades they received during the semester. Despite the difficulty of estimating the exact final grade, it seems that the task of predicting if a student will pass or fail the course is easier, as a more clear separation between the two classes is observable mainly in Figure 2.3. We can conclude that the grades obtained by the students during the semester may be relevant in predicting their final examination grade. However, the number of attributes considered is too small. For obtaining a more accurate mapping, the attribute set characterizing the students has to be extended with other relevant characteristics.

For evaluating the quality of the PCA plots from Figures 2.2 and 2.3 we are illustrating in Table 2.1 the values for $Prec$ computed for the entire data set D , as well as for the data set without attribute a_3, for various number k of neighbors and for two possible values for the threshold τ (0 and 1).

τ	k	$Prec$ - **entire data set**	$Prec$ - **data set without** a_3
0	1	0.443	0.445
	3	0.350	0.359
	5	0.316	0.336
1	1	0.706	0.722
	3	0.639	0.660
	5	0.608	0.625

Table 2.1: The performance of the PCA mapping.

We note that the results from Table 2.1 are correlated with the previous interpretation of the PCA plots, namely that the PCA built on the data set without the attribute a_3 (Figure 2.3) is more accurate than the one built on the entire data set D (Figure 2.2). In addition, as expected, the precision of the PCA decreases as the number k of nearest neighbors considered gets larger.

Analyzing the PCA plots from Figures 2.2 and 2.3 we also observe, as expectable, that one of the noisiest class of data points is those of students with the final grade 4 (blue colored). For testing this hypothesis, we depict in Figure 2.4 a PCA visualization only for the subset of students with grades higher than or equal to 5 (denoted by D_5). The precision $Prec(D_5, 1, 1)$ computed for this PCA mapping using $\tau = 1$ is 0.786, higher than the precision value 0.722 from the fourth row of Table 2.1. A value of 1 was selected for the threshold τ since we observed that the computation of $Prec$ for $\tau = 0$ is highly dependent on the structure of the data set. It is also noticeable that the PCA shape significantly changes when ignoring the students with the final grade 4, supporting our previous assumption.

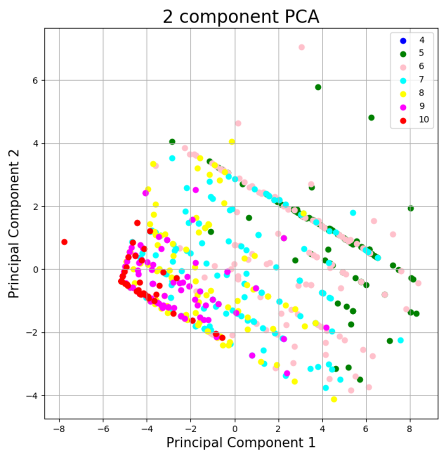


Figure 2.4: 2D PCA visualization of students having the final examination grade higher than or equal to 5. Attributes (a_1, a_2, a_3) are used for characterizing the instances.

The difficulty of discriminating between the instances with the final grade less than 5 and those which passed is explainable, due to various uncertain and unpredictible events such as:

students' plagiarism, the passing criterion applied by the instructors for the students that are close to passing, etc.

Second experiment

As previously show in Section 2.3.1, our second unsupervised learning based experiment is aimed to investigate how relevant is the relational association rules mining process in discriminating between the class of students that *passed* (the *positive* class) or *failed* to pass (the *negative* class). Table 2.2 depicts the sets RAR_- and RAR_+ mined from the data set D described in Section 2.1 using the experimental methodology from Section 2.2.1.

Set	Length	Rule	Confidence
RAR_-	2	$a_1 < 7$	0.84
	2	$a_2 < 6$	0.703
	2	$a_3 < 8$	0.748
	3	$a_1 \geq a_4 < 8$	0.695
RAR_+	2	$a_1 \geq a_4$	0.654
	2	$a_1 \geq 6$	0.635
	2	$a_2 \geq a_3$	0.601
	2	$a_2 \geq 7$	0.676
	2	$a_4 \geq 5$	0.706
	3	$a_1 < a_2 \geq 5$	0.615

Table 2.2: Interesting RARs mined from D (with all attributes). A minimum confidence threshold of 0.6 is used.

From Table 2.2 we observe two sets of RARs which characterize the set of students with the final grade less than 5 and the set of students with the final grade greater than 5, respectively. For example, the fourth RAR from Table 2.2 indicate that for 69.5$\%$ of the students which have not passed the exam, the seminar score is greater than or equal to the written test score, which is less than 8. Looking to the sixth rule in RAR_+ from Table 2.2 we observe that for 61.5$\%$ of the students which have passed the exam, the seminar score is less than the project score, which is greater than or equal to 5.

For evaluating the quality of the mined sets of RARs (RAR_+ and RAR_-), i.e. how well they discriminate between the *positive* and *negative* class, the evaluation measures introduced in Section 2.2.1 are computed for the entire data set D, as well as for the data set without attribute a_3. The obtained results are given in the second and third columns of Table 2.3.

	Evaluation measure	Entire data set	Data set without a_3
Accuracy	$Acc(RAR_+, D_+)$	0.669	0.682
	$Acc(RAR_-, D_-)$	0.760	0.764
	$AccM(RAR, D)$	0.714	0.723
Error	$Err(RAR_+)$	0.412	0.406
	$Err(RAR_-)$	0.432	0.435
	$ErrM(RAR, D)$	0.422	0.421

Table 2.3: The performance of the RAR mining process.

The results from Table 2.3 indicate a good enough performance of the mined set of RARs: an overall accuracy of **0.714** and an overall error of **0.422**. This is an indication of the fact that the interesting set of RARs mined from the set of students which passed the exam and which failed to pass may discriminate between the two classes. However, looking to the errors $Err(RAR_+)$ and $Err(RAR_-)$, we observe a certain overlapping between the rules characterizing the positive and negative classes, i.e there are RARs mined from RAR_+ which are verified by the students from D_- and viceversa. These may be due, as shown in Section 2.2.2, by possibly noisy or anomalous instances. Further pre-processing of the data is required before applying the RAR mining for reducing the overlapping between the grades. We also note that the RARs without the attribute a_3 (third column from Table 2.3) provide a higher accuracy (**0.723**) and a smaller error (**0.421**) than the RARs which include attribute a_3. Thus, the assumption from Section 2.2.2 that the attribute a_3 is not that relevant in discriminating between the students' final examination grade is confirmed.

The results previously presented highlight the potential of the sets RAR_+ and RAR_- to differentiate the final examination grade of students. However, we have to extend and carefully select the set of relations used in the mining process, as well as the attributes to be used in the mining process. Further analysis will be performed in this direction. Both experiments presented in this section highlighted the potential of unsupervised learning models in expressing interesting patterns in students' academic performance data and are useful for providing a better insight into the problem of students' academic performance prediction.

2.2.3 Conclusions and future work

This thesis investigated the effectiveness of applying two unsupervised learning models, *principal component analysis* and *relational association rule mining*, as intelligent tools for academic data sets analysis. The study conducted in this thesis highlighted the potential of unsupervised learning models in detecting patterns in analyzing students' academic performance. However, we observed that the performance of the unsupervised classification is influenced by possible anomalies from the academic data set and by the small number of attributes used for characterizing the instances.

Future work will be performed for extending the experimental evaluation on other academic data sets and to interpret the interesting RARs mined. The RAR mining experiment will be extended such that to mine relevant rules for each class of grades (ranging from 4 to 10), instead of using only two classes (*pass* and *fail*). For improving the performance of the mining process, we will also investigate methods for detecting anomalies and outliers from the academic data sets such that to reduce the impact of the noise on the learning process. As a natural subsequent step of our research, supervised classification models for academic students' performance prediction are envisaged.

2.3 Analysing the academic performance of students using unsupervised data mining

The study performed in this section is aimed to highlight the potential of applying two UL techniques (*self-organizing maps* and *relational association rule mining*) in analysing students' academic performance [CC19].

The main research question we are investigating in this thesis is regarding the ability of unsupervised learning models (SOMs and RARs) to detect hidden relationships between the grades received by the students during the semester and their final examination grade category at a certain academic discipline. In addition, we aim to analyse if the unsupervised learning models may reveal some information regarding the quality of the educational processes.

The rest of the section is organized as follows. Section 2.3.1 introduces our experimental methodology, while Section 2.3.2 discusses about the experimental results. The conclusions of our study together with several directions for future research are summarized in Section 2.3.3.

2.3.1 Methodology

As previously stated, our study [CC19] aims at investigating the relevance of unsupervised SOM and RAR models in analysing the academic performance of students.

We are introducing the following theoretical model. We denote by $\mathcal{S}tud = \{stud_1, stud_2, \ldots, stud_n\}$ a data set in which the instances $stud_i$ describe the performance of students during an academic semester, at a given academic discipline $\mathcal{D}$. Each instance $stud_i$ is characterized by a set of *grades* received during the semester $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$ representing attributes for measuring the performance of the student for the given discipline. Thus, each $stud_i$ is represented as an m-dimensional vector $stud_i = (stud_{i1}, stud_{i2}, \ldots, stud_{im})$, $stud_{ij}$ representing the value of attribute g_j for student $stud_i$.

The goal of the current study is to investigate if two unsupervised data mining models, *self-organizing maps* and *relational association rule mining*, are able to discover some rules and relationships which would be useful for predicting the *final performance* for the students, based on their grades obtained during the academic semester. Since predicting the exact final examination grade for a student is a difficult task, considering the uncertainty in the learning and evaluation processes, we are considering in this thesis four categories of final grades: (1) *excellent* (denoted by E and representing the final grades 9 and 10); (2) *good* (denoted by G and representing the final grades 7 and 8); (3) *satisfactory* (denoted by S and representing the final grades 5 and 6); and (4) *fail* (denoted by F and representing the final grade 4). Let us denote by $\mathcal{C} = \{E, G, S, F\}$.

We note that our unsupervised analysis does not include the grades of the students' at the written exam (obtained in the examination session), which are also part of their final examination grade. Thus, we aim to analyse if only the grades received by the students during the semester are enough to discriminate their written exam grade and, accordingly, the students' final examination grade category.

The problem investigated in this thesis, from an *unsupervised learning* perspective, is that of assigning to each student (characterized by its grades received during an academic semester) the category corresponding to its final grade. This assignment may be formalized by a mapping $f : \mathbb{R}^m \rightarrow \mathcal{C}$.

Experiments

The experiments described in this section are aimed to test the ability of SOMs and RARs, as unsupervised learning models, to detect relevant relationships in the students' grades (received during the semester) which are well correlated with their final grade category. For a certain grade category (class) $c \in \{E, G, S, F\}$ we denote by $D_c \subset D$ the subset of students from D whose final grade category is c. We note that $\bigcup\limits_{c \in \mathcal{C}} D_c = D$.

The data set described in Section 2.1 is used in our experiments. For analysing how correlated are the attributes a_1, a_2, a_3, a_4 with the target output (category corresponding to the final examination grade), the Pearson correlation coefficients are computed. The correlation values are shown in Table 2.4.

a_1	a_2	a_3	a_4
0.631	0.676	0.461	0.607

Table 2.4: The Pearson correlation coefficient between attributes a_1, a_2, a_3, a_4 and the target output.

From Table 2.4 we observe that there is a good enough correlation between the attributes a_1, a_2, a_4 and the category corresponding to the final examination grade. The *project score* (attribute a_2) shows the maximum correlation with the final category. The smallest correlation is observed for attribute a_3.

The *first experiment* is conducted for obtaining, using a SOM, a two dimensional representation of the data set. Two SOM visualizations will be provided: one for the entire data set (characterized by all attributes a_1, a_2, a_3, a_4) and the second for the data set without attribute a_3 (i.e the data set characterized only by the attributes a_1, a_2, a_4). After the SOM was unsupervisedly built, the U-Matrix method [KK96] will be used for visualization. For the SOM, a torus topology is used, with the following parameters: 200000 training epochs and a learning rate of 0.1. The *Euclidian distance* is used as a distance metric between the input instances.

The goal of our *second experiment* is to uncover in each subset D_c, using the $DRAR$ algorithm, a set RAR_c of interesting RARs. We aim to verify the hypothesis that the sets RAR_c are able to discriminate between the classes of students having different final grades.

For the RAR mining experiment, five additional attributes were added to the data sets D_c ($a_i = i, \forall i, 5 \leq i \leq 9$) and we used the following parameters for the mining process: 1 for the minimum support threshold, 0.6 for the minimum confidence threshold and two possible binary relations between the attributes ($<$ and $\geq$). Attributes a_5, a_6, a_7, a_8, a_9 represent some thresholds for the grades (i.e. $5, 6, 7, 8, 9$) and were added with the goal of enlarging the set of uncovered RARs, allowing the discovery of binary RARs such as $a_i \leq 5$.

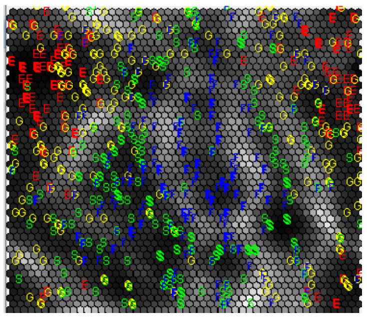
For evaluating how well the set RAR_c characterizes the set of students from the set D_c we use the average confidence of all the subrules from the rules from RAR_c, denoted by $Prec_c = \frac{\sum\limits_{r \in RAR_c} \sum\limits_{sr \in \mathcal{S}_r} conf(sr)}{|RAR_c|}$. By $conf(r)$ we denote the confidence of the rule r in the data set D_c and $\mathcal{S}_r$ represents the set of subrules of r (including itself). We note that $Prec_c$ ranges from 0 to 1, higher values for $Prec_c$ indicating that the set RAR_c better characterizes the data set D_c.

2.3.2 Results and discussion

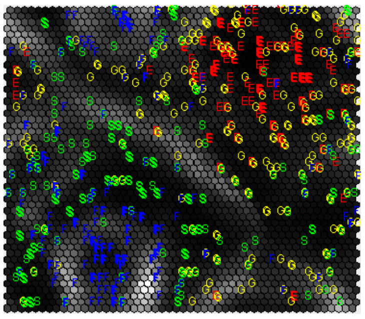
This section presents the experimental results obtained following the experimental methodology introduced in Section 2.2.1 and discusses about the patterns unsupervisedly discovered using the SOM and RAR models.

Experiments using SOMs

The left hand side image from Figure 2.5 illustrate the SOM obtained on the data set from Section 2.1 using attributes a_1, a_2, a_3, a_4, while the right hand side image from Figure 2.5 depicts the SOM trained on the instances characterized only by attributes a_1, a_2, a_4. On both images, the students with the same final class (final grade category which is depicted on the map) are marked with the same colour: red for the E labels, yellow for G, green for S and blue for F. As expected, Figure 2.5a depicts a good enough mapping, but still there is no clear separation between the grades. It seems that a slightly better mapping and separation between the grades is provided by the map from Figure 2.5b when attribute a_3 (the project status score) has not been considered.



(a) SOM visualization considering attributes a_1, a_2, a_3, a_4.

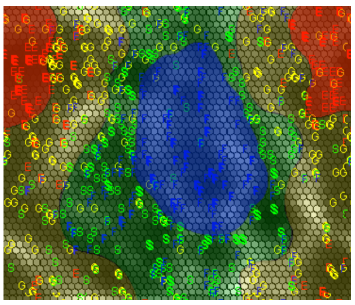


(b) SOM visualization considering attributes a_1, a_2, a_4.

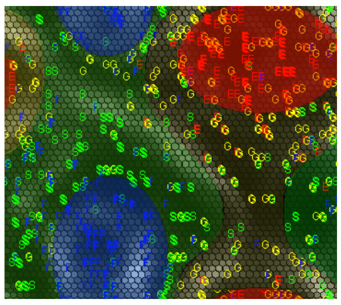
Figure 2.5: U-Matrix visualization of the SOM built on the data set D using attributes a_1, a_2, a_3, a_4 (left) and a_1, a_2, a_4 (right).

The SOMs from Figure 2.5 reveal the difficulty of the task for predicting the final examination grade category for the students, based on the grades they received during the semester. However, the unsupervisedly built SOMs are able to uncover some patterns regarding the students' final grade category. We observe two main areas on both maps, a cluster of students with the final categories F and S, which is well enough delimited and one containing the categories G and E. Inside the first cluster, we observe a well distinguishable subclass containing students with the final category F.

Figure 2.6 illustrate a detailed visualisation of the SOMs from Figure 2.5, considering the torus topology used for building the SOMs. The left side image from Figure 2.6 corresponds to the SOM from Figure 2.5a, while Figure 2.6a corresponds to the visualization from Figure 2.5b. On both SOMs, the distinguishable classes of students are highlighted and coloured according to their class label (E - red, G - yellow, S - green, F - blue).



(a) Detailed visualisation of the SOM from Figure 2.5a.



(b) Detailed visualisation of the SOM from Figure 2.5b.

Figure 2.6: Detailed visualisation of the SOMs from Figure 2.5.

A comparative analysis of the two images from Figure 2.6 and the highlighted areas reveal the following. In Figure 2.6a we observe that the F labeled cluster is well disgtinguishable. However, there are a few outliers that go beyond the cluster's border, entering in the S class zone. Moreover, the E labeled cluster also interferes with the outer regions and creates noisy zones on the map. The flaw with the image from Figure 2.6a is that, even though the centres of the E and F labeled groups are compact and solid, the margins of each one tend to be more fuzzy, exchanging different grades with neighbouring regions. As we can see, there are overlapping grades, especially regarding the F labels, that incline towards a defiance of the boundaries, which suggests that the unsupervised classification model is not capable of clearly discriminating all the students and, consequently, misinterprets some of the patterns of their performance. Regardless these misclassifications that may be due to the small number of attributes characterizing the instances and the presence of outliers, the SOM model is confident enough to make correct prediction most of the time. In Figure 2.6b there are two contrasting, well separated, areas of high/low grades with a sharp gap between them. The regions corresponding to the average grades surround tightly these two clusters. Few exceptions still occur when separating the grades.

On the other hand, the SOM from Figure 2.6b seems to provide a better classification of the data. The two opposed areas (of students with high/low grades) now are more compact and clearly separated. Their margins tend to be smoother, especially for those labeled F that are now a lot more compact than in Figure 2.6a, with less perturbations from the other grades, as a virtual median barrier keeps them apart. However, the class of average grades is not so well distinguishable, and, there has been a trade-off between size and accuracy, since the higher grades are now more compact, as well, but less separated from each other. Nonetheless, if we would combine the two previous interpretations, we may analyse and classify the data better by using both of their strengths, as in each model the data is more or less scattered across the map, which would be of use in cases when we desire a greater confidence on a particular class of grades. While the model from Figure 2.6a may offer us a better understanding of the students with passing grades (as they belong to rather concentric groups), the SOM from Figure 2.6b may show us an antithetical approach of the highest grades and the lowest ones. Moreover, the SOM model built without using attribute a_3 is particularly good at classifying lower grades with greater accuracy, and, even if there still is noise in the data categorised as E or G, the model can confidently predict the performance of a good student. What makes it so difficult to classify all the students is the fact that there is a discrepancy mainly among the F labeled class, as there is an inconsistent progress for each one, that would result in a more scattered pattern that interferes with better classified data.

Analysing both maps from Figure 2.5 we also note that most of the students belonging to category F (i.e. having the final grade 4) are, on both maps, well enough delimited from the students from other categories (S, G, E). Overall, we observe as a main pattern that neighboring students belong to near categories (F-S, G-E). But several outliers may be observed on the map: neighboring students having very different categories (e.g. E and F). A possible explanation for such incorrect mappings may be that the data set includes the examination results not only for the normal session, but also for the retake session. Thus, it is very likely to have the same instance but with different final labels (i.e. the categories from the normal and retake session) which may be very different (e.g. F and S). Besides the previously mentioned cause for outliers, another possible one is given by the intrinsic uncertainty of the educational processes. The data set includes instances for which there is a visible uncorrelation between the grades received during the semester and the final examination grade. Such discordances may appear due to a bias evaluation or some unexpected events in the students learning process. To avoid introducing noise in the data set which will affect the performance of the learning, we will further investigate preprocessing techniques for detecting such outliers.

Experiments using RARs

The results of the RAR mining experiment are further presented, with the aim of highlighting the relevance of the relational association rules mining process in distinguishing between classes of students having different final grade category. For each category $c \in \mathcal{C}$ we present in Figure 2.7 the set RAR_c of maximal RARs mined from D_c using the experimental methodology from Section 2.2.1. In addition, we indicate the value of $Prec_c$ which evaluates the quality of the set RAR_c.

c	**Length**	**Rule**	**Confidence**	**Prec**
E	2	$a_1 > 8$	0.898	0.716
	2	$a_2 > 9$	0.713	
	2	$a_4 > 8$	0.771	
	3	$a_1 \le a_2 > 8$	0.707	
	3	$a_1 \le a_3 > 8$	0.650	
	3	$a_1 \le a_4 > 7$	0.643	
	3	$a_2 \le a_3 > 9$	0.618	
G	2	$a_1 > 5$	0.87	0.689
	2	$a_1 \le 8$	0.675	
	2	$a_2 > 7$	0.728	
	2	$a_2 \le 9$	0.679	
	2	$a_3 > 7$	0.683	
	2	$a_3 \le 9$	0.630	
	2	$a_4 > 6$	0.630	
	2	$a_4 \le 8$	0.626	
	3	$a_1 \le a_2 > 6$	0.614	
	3	$a_1 \le a_3 > 6$	0.606	

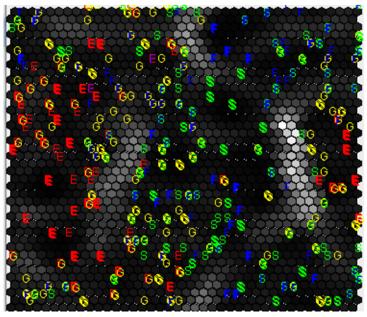
c	**Length**	**Rule**	**Confidence**	**Prec**
S	2	$a_1 \le a_3$	0.686	0.69
	2	$a_1 \le 6$	0.737	
	2	$a_2 \le 6$	0.661	
	2	$a_3 > 5$	0.623	
	2	$a_3 \le 7$	0.619	
	3	$a_1 \le a_2 \le 9$	0.640	
	3	$a_1 \le a_3 > a_4$	0.636	
	3	$a_1 > a_4 \le 5$	0.653	
	3	$a_2 > a_4 \le 5$	0.640	
	3	$a_3 > a_4 \le 7$	0.631	
	4	$a_1 \le a_2 > a_4 \le 7$	0.619	
F	2	$a_1 \le a_3$	0.643	0.695
	2	$a_1 \le 5$	0.624	
	2	$a_2 \le a_3$	0.654	
	2	$a_2 \le 5$	0.680	
	2	$a_3 \le 6$	0.661	
	3	$a_1 > a_4 \le 5$	0.628	
	3	$a_3 > a_4 \le 8$	0.617	

Figure 2.7: The sets of maximal interesting RARs mined for each category of grades: E, G, S, F.

From Figure 2.7 we observe that the sets of RARs characterizing the classes of students with different final grade category are disjoint, in general. For example, the fourth RAR of length three from the left side table indicate that for 61.8% of the students who have received a final examination grade of 9 or 10 (category E), the project score is less than or equal to project status score, which is greater than 9. We note that this RAR does not characterize the other categories of students, thus it is very likely to be useful for discriminating students according to their final category.

For facilitating the interpretation of the RARs, we decided to build a SOM for having a visual representation of the rules and highlighting how well they characterize the classes of students. Let us denote by $Rules$ the sequence of all distinct mined RARs given in Figure 2.7, including all their subrules. If a RAR of a certain length appears in more than one category, it will appear in $Rules$ only once (e.g. the RAR $a_1 \le a_3$ of length 2 appear as an interesting rule for both categories S and F). Thus, an additional data set D_{RAR} is created by characterizing each student $stud_i$ from the original data set D by a 48-length binary vector $V_i = (v_1^i, v_2^i, \ldots, v_{35}^i)$, where 48 is the size of the sequence $Rules$, i.e the number of distinct RARs mined. An element v_j^i from V_i is set to 1 if the j-th RAR from $Rules$ is verified in $stud_i$ and 0 otherwise. Figure 2.8 depicts the SOMs built on the data set D_{RAR} using all attributes a_1, a_2, a_3, a_4 (left) and using only attributes a_1, a_2, a_4 (right). On each SOMs, the instances are labeled with their final grade category (E, G, S, F).

From the interesting RARs depicted in Figure 2.7 and visualized in Figure 2.8a we also


(a) SOM visualization considering attributes a_1, a_2, a_3, a_4.

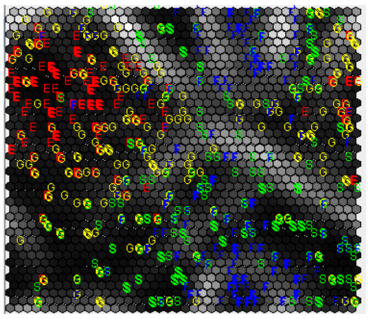

(b) SOM visualization considering attributes a_1, a_2, a_4.

Figure 2.8: U-Matrix visualization of the SOM built on the data set D_{RAR} using attributes a_1, a_2, a_3, a_4 (left) and a_1, a_2, a_4 (right).

observe that there is an overlapping, in general, between the set of RARs characterizing near categories (F/S, G/E). This is expectable, as previously shown in Section 2.3.2 where the SOM mapping highlighted that there are instances, mainly from near categories, that are hard to discriminate. For instance, the rule $a_1 \leq a_3$ appears for both S and F categories with highly similar confidences (0.686 for S and 0.643 for F). Another example is the rule $a_2 > 7$ from G and $a_2 > 9$ from E which is also explainable due to some instances that are on the border between the two categories. Certainly, such overlapping rules are not useful for discriminating between classes. A post-processing step would be useful for detecting and removing such rules from the mining process and will be further investigated. However, we observe interesting RARs, such as the 4-length rule from the S category, which characterizes only this category of students. On the other hand, the RARs expressing the E category have the higher precision (0.716) and this is also observable on the SOM, as this category is easily distinguishable from other classes, sustaining the conclusions from Section 2.3.2.

Regarding the usefulness of attribute a_3 in mining relevant RARs, the following were observed by analysing the RARs depicted in Figure 2.7 and visualized in Figure 2.8b. On the one hand, a_3 creates some misleading relationships between the features, such as the rule $a_3 > a_4$, creating overlapping values with other grades (i.e. it is interesting for both F and S categories). That would cause some misclassifying when interpreting border cases, students that are in between two classes. When removing the a_3 feature, the RAR model is improved, as there are less overlapping areas, even though there are not as many relationships between the features (but the number of RARs may be increased by reducing the minimum confidence threshold). This can also be seen in the experiment from Section 2.3.2, when comparing the two SOMs (built with and without attribute a_3). On the SOM from Figure 2.8b built without considering a_3 feature, there is a tendency of the higher and lower grades to create distinct clusters that have little or no noise. The class of F labeled instances is more clearly distinguishable in Figure 2.8b than in Figure 2.8a.

The results previously presented highlight the potential of the sets RAR_c to differentiate the students according to their final examination grade category, based on their grades received during the academic semester. As previously shown, RARs are able to express interesting patterns in academic data sets and are useful for providing a better insight into the problem of students' academic performance prediction. However, we have a small number of attributes in our case study. By increasing the number of relevant attributes, it is very likely that more informative and meaningful RARs would be mined.

It is worth mentioning that the results obtained using both SOM and RAR models conducted to similar conclusions, which were detailed in Section 2.3.2 and 2.3.2. For obtaining a more accurate representation of the input instances (students) using both unsupervised learning models investigated in this thesis (SOMs and RARs), the attribute set characterizing the students must be enlarged with other relevant characteristics. It would be useful to have multiple attributes in the mining process and to extend the set of relations used in the mining process in order to obtain much more informative and relevant RARs as well as a better separation using the SOM model.

A more in depth analysis of the outlier instances provided by the SOM and RAR models may provide valuable information regarding the improvement of the educational processes. For instance, the results of the unsupervised learning processes may reveal the following: (1) the examination grades for some of the evaluations received during the academic semester may be incorrect due to the variations within the instructors evaluation criteria or standards, as well as possible cheating methods used by a few students; (2) some of the partial examinations may be redundant; (3) a change of the computation method for the partial grades may be required; (4) it could be necessary to increase the number of the examinations performed during the academic semester.

2.3.3 Conclusions and future work

This thesis examined two unsupervised learning models, *self-organizing maps* and *relational association rule mining*, in the context of analysing data sets related to students' academic performance. Experiments performed on a real data set collected from Babeş-Bolyai University, Romania highlighted the potential of unsupervised learning based data mining tools to detect meaningful patterns regarding the academic performance of students.

We may conclude that the grades received by the students during the semester may be relevant in predicting their final performance. However, several outliers were observed in the data set. Such anomalous instances may be due to: (1) a small number of students' evaluations during the semester (attributes); (2) the students' learning process which is not continuous during the academic semester; (3) the difference between the evaluation standards of the instructors from the laboratory and seminar activities. As a consequence, an increased number of evaluations during the academic semester would be useful, for stimulating students to study during the semester and not only for the final examination.

Future work will be performed in order to extend the experiments and the analysis of the obtained results. For increasing the performance of the unsupervised learning process, methods for detecting anomalies and outliers in data will be further investigated. In addition, a post-processing phase for filtering the set of mined RARs will be analysed for removing rules which overlap with multiple classes.

Chapter 3

Enhancing the performance of image classification through features automatically learned from depth-maps

This chapter introduces our original unsupervised learning approach [CTC21], developed for computer vision (CV). In [CTC21] we analysed the relevance of using depth-maps extracted features in the context of indoor-outdoor image classification. We used t-SNE unsupervised learning clustering algorithm for proving the usefulness of depth-maps and also the MLP and DPT supervised learning models for supporting the aforementioned method. The experimental results obtained by applying the aforementioned supervised learning models on DIODE dataset emphasize depth-maps features' effectiveness in classifying indoor and outdoor scenes.

Computer Vision (CV) aims to cover a wide range of visual tasks in order to automate the processes of decision making in domains such as *autonomous driving*, *robotic process automation*, *product quality control*, or creating virtual environments for Virtual Reality/Augmented Reality. Recently, all CV tasks are performed using *Deep Learning* (DL) models that consist of multiple types of layers for processing input images at different resolutions. The most popular architecture in image processing is the one of *Deep Convolutional Neural Networks* (DCNN) which has the *convolution* as core concept. The aim of such networks is to encode pictures' spatial information with the help of convolutions while decreasing the resolution of images in order to grasp more of the scene context. Such convolutions are intensely researched in order to maximise the amount of information extracted and minimise the computational cost of network training.

Depth estimation (DE) [LHKS19] or *depth extraction* aims towards gathering information about the structure of a scene using mathematical or learning means, hence every pixel of an image is supposed to be assigned a depth value. Stereo vision methods are highly dependent on illumination intensity and angle, therefore monocular depth estimation is more convenient for calibration or identification than stereo vision means. *Monocular depth estimation* (MDE) is a difficult task as even humans misinterpret *Depth of Field* when looking with one eye open.

The problem of determining the depth in a single image is an ill-posed one as stated in the literature [Bho19], as we have to rely on features such as shapes, textures and occlusions in order to gain information about the scene geometry and different semantic meaningful regions that would help us compute the depth value of every pixel in an image. The great difficulty in this situation is that one 2D photo is not enough to reconstruct a 3D environment as there are infinite 3D layouts that can be projected into a 2D space to reproduce an image. As optical illusions have proven us, humans make assumptions about the shapes and sizes of the objects, ultimately leading them to the wrong conclusions. Still, that assumptions are also to be made by learning DL models when dealing with so little spatial information. Nonetheless, if depth information would be available, we could therefore use the advantages of combining RGB features with depth cues to enhance visual tasks, by offering the models extra guidance.

The goal of the research conducted in this thesis is to study the usefulness of *depth* information applied to both *Machine Learning* (ML) and DL models in indoors and outdoors contexts that have totally different layouts and detail density.

The contribution of the thesis is twofold. Firstly, we are proposing four feature sets for characterizing the images with the aim of comparatively studying their relevance for the visual task of indoor-outdoor classification. One feature set is manually engineered, based on using RGB information and features extracted from depth-maps, while other three features sets are automatically learned from images using a deep learning model. The relevance of the proposed feature sets will be investigated through an unsupervised learning classifier. Then, in order to confirm the hypothesis that the depth related features are useful in discriminating between indoor-outdoor images, a supervised learning classifier will be used to evaluate the performance of the image classification task on DIODE (Dense Indoor and Outdoor DEpth) data set [VKZ+19]. To the best of our knowledge, a study similar to ours has not been performed in the literature yet.

To sum up, the current work is based on the following research questions:

RQ1 How relevant are depth maps in the context of indoor-outdoor image classification and to what extent does aggregating visual features from more granular sub-images increase the performance of classification?

RQ2 Which of the engineered and automatically learned features from depth-maps are able to better discriminate between indoor-outdoor images?

RQ3 How correlated are the results of the unsupervised based analysis and the performance of supervised models applied for indoor-outdoor image classification?

The rest of the thesis is structured as follows. The methodology proposed for answering the previously stated research questions is introduced in Section 3.1, whilst Section 3.2 presents the results of our unsupervised and supervised learning-based analysis. Section 3.3 concludes the thesis and indicates directions for further improvements and future work.

3.1 Methodology

This section describes the methodology we are proposing with the goal of answering the research questions stated in the beginning of the chapter and highlighting that depth-maps features improve the performance of indoor-outdoor image classification.

The classification problem we are focusing on is a binary classification one, that of learning to assign to an image I a class $c \in \{I = indoor, O = outdoor\}$. To achieve the learning task, we are comparatively analysing the relevance of several image features in distinguishing the target class (I/O) for a specific image. The relevance of the proposed features is first assessed in an unsupervised learning scenario, by testing the ability of an unsupervised learning model to partition the input images in two clusters (corresponding to the two output classes). Then, in order to strengthen the unsupervised learning based analysis, the performance of a supervised learning classification model is evaluated considering various feature vectors characterizing the images.

We start by describing in Section 3.1.1 the data set used in our experiments, then we continue with introducing the previously summarised stages of our study : (1) **feature extraction**; (2) **unsupervised learning-based analysis**; (3) **supervised learning-based analysis and performance evaluation**.

3.1.1 Data set

The image database used in our experiments is DIODE data set [VKZ+19] consisting of 27858 RGB-D (R-Red, G-Green, B-Blue, D-Depth) *indoor* and *outdoor* images collected with a **FARO Focus S350** laser scanner. The data set is diverse, the photos have been taken both at daytime and night, over several seasons (summer, fall, winter). The full resolution of the images from the data set is 1024 × 768. Apart from RGB-D images, DIODE data set also provides us with normal maps that could further enhance the learning of depth and vice-versa. Normal maps are RGB images in which its components correspond to the X, Y, and Z coordinates, respectively, of the surface normal, which would help better determine regions with similar characteristics.

Figures 3.1 and 3.2 depict the frequencies of depth values represented as percentages of the total number of pixels. There are almost twice as many outdoor images (18206) than indoors (9652) in order to capture the variety of different environments. Also, the data set is partitioned by default in train (25458) and validation (771) splits with a suggested test split of 1629 images.

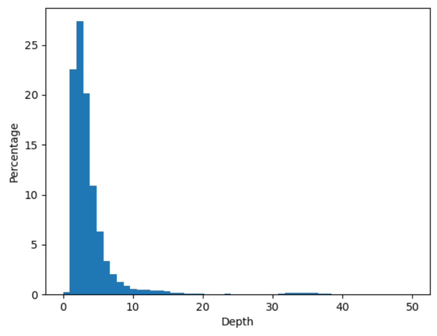


Figure 3.1: indoor depth frequency

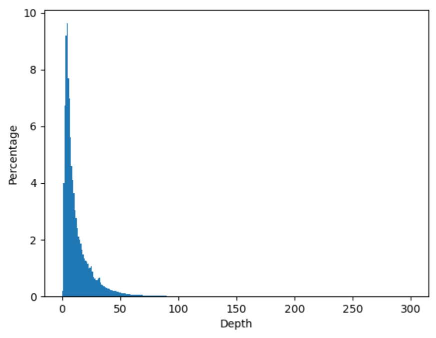


Figure 3.2: outdoor depth frequency

In the experiments we chose to use a subset of every hundredth image from the train split, resulting in a 255 images sample from both indoors and outdoors settings. The resolution for the RGB/RGB-D experiments is the images' original one and for the DPT experiments a resolution of 384 × 384 was used.

3.1.2 Feature extraction

The *feature extraction* stage refers to extracting a set of features relevant for representing the images in a vector space model. Formally, an image is encoded by a set $F = (F_1, F_2, \dots F_m)$ of relevant features. Thus, an image I will be represented as a high-dimensional numerical vector $I = (i_1, i_2, \ldots, i_m)$, where i_j represents the value of feature F_j obtained for image I. An image will be, subsequently, visualized as a data point in $\mathbb{R}^m$.

We are proposing four feature sets for characterizing the images with the aim of comparatively studying their relevance for the visual task of indoor-outdoor classification. The first feature sets is manually engineered, based on using RGB information and features extracted from depth-maps, while the next two feature sets are automatically learned from raw images and the latter follows the same procedure but for depth-augmented images.

Manually engineered features.

In our approach we considered *RGB* of the images and the *Depth* of the depth-maps. Instead of resizing the images (which would have led to loss of information), we have chosen to aggregate the information of some regions of the image. We started from the intuition that smaller sub-images would generate more specific features.

The first vectorial representation proposed for the images is based on aggregating features (RGB information and features extracted from depth-maps) from sub-images of different sizes to compare multiple scales of region attention. Before the feature extraction step, the image is divided, recursively, in 4^n sub-images of equal sizes: the initial image is divided in 4, then each of the four sub-images is again divided in 4, and so on. An example of such a split is given in Figure 3.3, for $n = 2$, resulting in 16 equally sized sub-images.

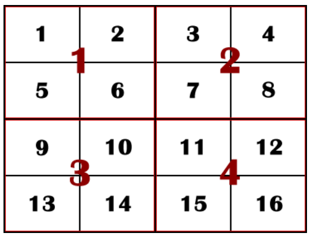


Figure 3.3: Structure of 16 (4^2) image splits.

1. **RGBD features** (RGBD) are obtained by aggregating RGBD values from sub-images. Thus, for an image splitted in 4^n sub-images (as previously described), a vector of size $m = 4 \times 4^n$ is built by concatenating, in the order top-to-bottom and left-to-right (see Figure 3.3), the average R, G, B, and D values for each sub-image.

Automatically learned features.

The three sets of features we propose are automatically learned from raw images using a deep learning model, called *vision transformers* (DPT). The approach introduced by Ranftl et al. [RBK21] leverages visual transformers instead of convolutions. The main advantage of this change is that compared to convolutions which have a limited spatial understanding due to their size and need downsampling to capture such information, *DPT* has included global receptive field information in each layer. Therefore, this leads to finer, sharper and more coherent predictions, establishing a new state-of-the-art in DE and *semantic segmentation* (SS) [RBK21].

2. **DPT encoder DE features** (DPT-DE) are the features automatically learned by a pre-trained DPT on MIX 6 DE data set [RBK21].

3. **DPT encoder SS features** (DPT-SS) are the features automatically learned by a pretrained DPT on ADE20K SS data set [ZZP+17].

For representations 3 and 4, the features are extracted after the encoder of the DPT model. We mention that an image will be represented as an m-dimensional numerical vector, where m is the dimensionality of the encoding layer of the DPT model.

The last set of features we propose is automatically learned from depth-augmented images using the DPT model. We consider the RGBA (RGB, A-Alpha which encodes pixel transparency) opaque images (that means the A value is 255 for each pixel) and we use $A_{ij} = 255 - clip(D_{ij})$ for every pixel in the image to create a progressively denser fog that simulates the depth of field. We clip the depth values D to a $[0, 255]$ meters interval so that they fit in a byte. Then, we blend the RGBA image with variable transparency with a white background and encode it to a RGB representation that the DPT model uses.

4. **DPT encoder SS depth-augmented features** (DPT-SS+D) are the features automatically learned by a pretrained DPT on ADE20K SS data set from the DIODE depth-augmented images [ZZP+17, VKZ+19].

3.1.3 Unsupervised learning-based analysis

In our approach we are applying 3D t-SNE (*t-Distributed Stochastic Neighbor Embedding*) [vdMH08] for *non-linear* dimensionality reduction of the m-dimensional representation of the images from the data set, computed according to the representations 1.-4. t-SNE is used in exploratory data analysis for uncovering patterns in data useful for clustering. The model uses *Student t-distribution* [Mar12] to better disperse the clusters.

The unsupervised learning-based analysis using t-SNE is conducted with the goal of providing useful insight about data organization and features' importance. Before the t-SNE visualization, for the RGBD data (representation 1) *data normalization* with the **inverse hyperbolic sine (asinh)** was applied in order to increase sensitivity to particularly small and large values. For automatic feature extraction (representations 2, 3, and 4), we did not use any preprocessing step, as DPT models are pretrained on raw RGB data.

3.1.4 Supervised learning based analysis

To empirically analyze the relevance of the features sets proposed in Section 3.1.2, the features are fed into a supervised classification model whose performance will point out the importance of the used features. As a supervised classification model we decided to use a simple, lightweight neural network model, the *Multilayer Perceptron* (MLP) [AC20]. The MLP classifier will be used for the indoor-outdoor image classification task. The goal of the supervised learning analysis is to strengthen the interpretation of the unsupervised learning one through the evaluation metrics used for assessing the performance of the MLP classifier. We expect the results of the unsupervised learning analysis (Section 3.1.3) to be correlated with the results of the MLP classifier, i.e. higher performance for MLP to be obtained for the feature set providing the best clustering using t-SNE.

Four evaluation metrics, usually used in the supervised learning literature, will be used for assessing the performance of the MLP classifier [GZC09]: *accuracy* (Acc), *recall* or *sensitivity* (Sens), *specificity* (Spec) and *Area under the ROC curve* (AUC). All these measures range from 0 to 1, higher values for the metrics corresponding to better classifiers. The literature reveals that AUC is one of the best metrics used for measuring the performance of supervised classifiers, particularly for imbalanced data sets.

3.2 Results and discussion

This section presents and analyzes the experimental results obtained by applying the methodology introduced in Section 3.1 on the DIODE data set described in Section 3.1.1. The experiments are conducted with the goal of answering research questions RQ1-RQ3, namely investigating the relevance of features extracted from depth maps in the context of indoor-outdoor image classification.

3.2.1 Experimental setup

Regarding the DPT architecture used for automatic feature extraction, a resolution of 384 × 384 was used for the image and $m = 49152$ features were extracted after the encoder.

For t-SNE the following parameters setting was used: a *perplexity* of 20, a learning rate of 3 (for obtaining a finer learning curve at the expense of slower convergence) and 1000 iterations.

For the supervised classifier, we used the MLP model with its default configuration. For a thorough training and evaluation, we chose to implement the k-fold validation with $k = 10$. For each of the aforementioned performance metrics (Section 3.1.4) we reported the average value over the k folds, together with the 95% confidence interval (CI) of the mean, computed as follows: $1.96 \cdot \sigma/\sqrt{k}$ (σ denotes the standard deviation of the values obtained during the k folds).

For both t-SNE and MLP we used the implementation from scikit-learn [Sci21]. The version of the library we worked with is 0.22.2.post1.

3.2.2 Results of the unsupervised learning based analysis

Figures 3.4 - 3.7 depict the 3D t-SNE visualization for the images represented with the four feature sets proposed in Section 3.1.2. For extracting the RGBD features (representation 1), four splits of the original image were used (i.e. $n = 1$). The t-SNE visualization using RGBD features (Figure 3.4) depicts a good enough boundary region between the indoor and outdoor classes.

As we can see in Figure 3.6, the DPT model trained for SS outputs a set of features with high relevance for indoor-outdoor classification as there are many object classes that are uniquely found only in indoor or outdoor settings. Concerning DE, the DPT model's features extracted from the encoder are less connected to the indoor-outdoor classification as there are fuzzier boundaries between the values of depth specific for each scene. Overall, Figure 3.7 presents the best separation, having more distant and compact clusters with a better boundary region, benefiting of both semantic cues and depth augmentation.

To conclude, the unsupervised learning based analysis previously conducted reveals that the best features are those learned by the encoder of DPT trained on SS depth-augmented features.

3.2.3 Results of the supervised learning based analysis

According to the methodology introduced in Section 3.1.4, the MLP classifier is applied on the DIODE data set with the aim of reinforcing the results of the t-SNE analysis conducted in Section 3.2.2. Table 3.1 presents the performance of the classifier for the feature sets introduced in Section 3.1.2 in terms of the evaluation metrics described in Section 3.1.4.

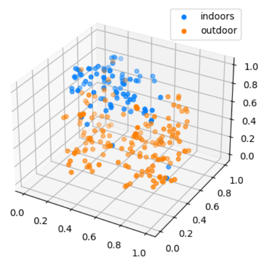


Figure 3.4: t-SNE for RGBD representation with 4 splits.

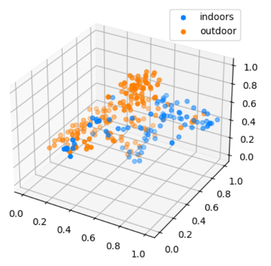


Figure 3.5: t-SNE of DPT DE learnt features.

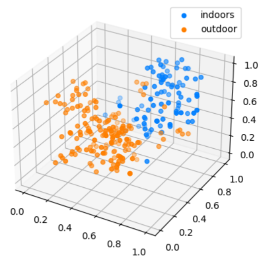


Figure 3.6: t-SNE of DPT SS learnt features.

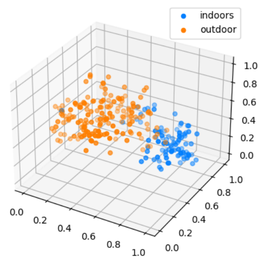


Figure 3.7: t-SNE for DPT encoder SS depth-augmented features.

In addition to the four feature sets introduced in Section 3.1.2, we also considered the RGB representation of an image, which is similar to the RGBD one but ignoring the Depth feature. Thus, for an image splitted in 4^n sub-images (as described in Section 3.1.2), a vector of size 3×4^n is built by concatenating, the average R, G, and B values for each sub-image. The best values are highlighted in Table 3.1, for each performance measure.

Features	**# Splits (n)**	**Acc**	**AUC**	**Spec**	**Sens**
RGB	0	0.692±0.077	0.525±0.056	0.980±0.028	0.070±0.121
	1	0.688±0.064	0.517±0.022	**0.989**±0.014	0.046±0.049
	2	0.669±0.049	0.545±0.048	0.912±0.068	0.163±0.136
RGBD	0	**0.880**±0.039	0.858±0.041	0.898±0.058	0.817±0.081
	1	0.876±0.043	**0.862**±0.044	0.894±0.046	0.829±0.063
	2	0.838±0.044	0.826±0.053	0.848±0.060	0.804±0.099
DPT-DE	0	0.823±0.131	0.831±0.076	0.812±0.185	0.850±0.069
DPT-SS	0	**0.950**±0.027	**0.942**±0.029	0.969±0.034	**0.915**±0.053
DPT-SS+D	0	**0.961**±0.015	**0.956**±0.021	**0.970**±0.019	**0.941**±0.041

Table 3.1: Results of indoor-outdoor supervised classification on DIODE data set. 95% CIs are used for the results. The DPT encoder features are used in all the experiments.

As seen in the experiments we performed, the results of the supervised classification using the MLP model are highly correlated with the unsupervised learning findings.

Table 3.1 reveals that the best performance of the MLP classifier, in terms of AUC, as highlighted by the t-SNE analysis, is obtained using features automatically extracted from depth-augmented images by the DPT model trained on SS (DPT-SS+D feature set). By using this representation, an AUC of 0.956 and an accuracy of 0.961 have been reached. One may also observe that the second best performance in terms of AUC and accuracy is obtained using the features learned by DPT trained for SS. As expected, in case of RGB features, the performance is poor, it did not surpass 70% in accuracy and the AUC metric is a little better than random guessing. However, by adding the Depth feature (RGBD), the performance is significantly increased (88% accuracy) while being consistent and reliable as the AUC metric suggests.

It has to be noted that neural networks learn very different features when comparing DE to SS. That happens because SS is a classification problem and it does not need learning the camera pose or its localisation in the scene context. The features learnt by SS models consist of different patterns for predicting and localising the regions relevant objects are situated in. However, it does not necessarily need to know about depth information, as the most important features lay within the semantic regions. The results reveal that the solely learnt depth cues do not help the indoor-outdoor classification as much as the semantic information about the present regions in the images.

What is somehow surprising is that the third best AUC and accuracy (highlighted in red) are achieved by using RGBD features, surpassing the performance obtained using features learned by DPT trained on DE. The performance for RGBD representation is with only 8% - 9% lower than the best performance obtained using the features learned by DPT trained on SS. Thus, a good enough performance is achievable using RGBD features, but with a significantly reduced computational cost.

Even though a direct comparison to the literature is somewhat difficult, due to the recent nature of the DIODE data set, our top accuracy obtained (96%) surpasses others reported on the indoor-outdoor classification task (up to 93.71%), obtained from similar principles of selecting features and feeding them into a classifier, albeit on different data sets.

Moreover, the accuracy of MLP with RGBD features (88%) outperforms classical machine learning classifiers already applied in the literature (accuracy ranging from 81.55% to 87.70%). Additionally, it is less computationally expensive overall compared to other work, as it aggregates features in a single step using basic mathematical operators from the images then inputs them into a lightweight MLP model. Using the RGBD features, we tried maximising the trade-off between cost and performance.

To summarise, there is a clear improvement obtained using depth-maps features. In addition, a main benefit of the features extracted from depth-maps (i.e. the RGBD representation) is that a good performance is obtained using a lightweight model (MLP) but using less features and parameters compared to other models, such as DPT. Additionally, the learning model requires low memory and computational cost compared to other deep learning methods, being a significant increase in performance when enhancing RGB information with depth cues.

3.3 Conclusions and future work

We have conducted in this thesis a study towards determining the relevance of depth maps in the context of indoor-outdoor image classification. Four feature sets for characterizing the images were introduced and their relevance for the visual task of indoor-outdoor classification was analyzed. Among the proposed feature sets, one was manually engineered and defined based on RGB information and features extracted from depth-maps, while the other features sets were automatically learned from raw images using a deep learning model. The advantage of depth-maps in the context of indoor-outdoor image classification has been empirically proven through experiments performed on DIODE data set, using both ML and DL models.

Moreover, we have shown the benefits transfer learning brings in the context of our task of choice, as both pretrained DPT models perform well, especially the SS one when using depth augmented features. DPT proved itself to be a good platform for future research thanks to its ability to learn more contextual information.

The research questions addressed in the beginning of the chapter were answered. The four feature sets proposed in this thesis serve as a good proof of concept in what regards the addition of depth-maps information to the RGB images. Our handcrafted RGBD set of features brought a significant improvement compared to the RGB one for indoor-outdoor image classification. As we have experimentally proven, a more granular feature extraction method is better but up to a point, as we faced incomplete data that made the learning process harder.

Concerning the unsupervised based learning methods, we can confidently say that the t-SNE proved itself as a trustworthy means of anticipating the supervised results as we have seen in the comparative study on the proposed feature sets. Each method presented has its strengths, whether it's the computational efficiency or the overall performance and consistency. Nevertheless, the accuracy reached using so few features on the handcrafted RGBD feature set is remarkable, competing even with DL models. Eventually, the automatically learned features performed better because there were in a larger number, therefore the greater possibility of more patterns to be found and they had the flexibility of following their own learnt rules instead of handcrafted ones. Moreover, we have proven the *transfer learning* usefulness of SS learnt features in the context of our problem of choice.

Future work may refer to better a understanding of the learning patterns of SS and DE deep learning models in the context of transfer learning so that we could identify other domains of interest and adapt the studied problem of indoor-outdoor classification to a more general one. We are further aiming to extend our study to multiple data sets, using a wider variety of scenes, for a better validation of the current findings.

Chapter 4

Depth on Field (DoF) - a software solution for visual task performance using handheld devices

4.1 Problem definition and specification

Monocular Depth Estimation is a complicated task that implies gathering geometric cues from a single image. Compared to *Stereo Depth Estimation*, MDE does not benefit from a second calibrated camera which would help better understand the scene, therefore all we have to work with is one image. Nonetheless, MDE-based networks can also compute occlusion masks and determine local and global pose when inputting a sequence of frames from on-board videos like the ones KITTI dataset [USS+17] has. However, our work focuses on estimating depth from one single image which is a truly challenging task. Nonetheless, *Semantic Segmentation* is also a complex problem that requires lots of data to learn from so that the models are robust to errors and the predictions are consistent. As we have previously mentioned, DPT proved itself to be a good platform for transfer learning, therefore we are going to use it in our implementations. The application we created for performing SS tasks and solving the ill-posed problem of DE could potentially help its users to measure indoors and outdoors distances with the help of SOTA DL models, provided that they have internet connection. Moreover, we have implemented the choice of the task and model to be performed by so that we can comparatively observe the SS and DE tasks and highlight the DPT model for being such a good transfer learning platform. The front-end application is designed for Android 11 and it could be used on portable devices such as phones and tablets which would make it well suited for on-field work in domains such as land cadastre, firearm training (i.e. target ranging) and any other tasks that requires ranging. The back-end exposes an API that could be further used by any device that supports Firebase Notifications. For the ML module, a strong processor and GPU are needed for meeting the networks' requirements, hence we are running the servers on a laptop with an Intel Core i7-6700HQ CPU @2.6 GHz, 24 GB DDR4 RAM, Samsung SSD 860 EVO M.2 2TB, NVIDIA GeForce GTX 1060 and a Wi-Fi connection of maximum 50 Mbps. The servers are reachable at the address http://depthonfield.ml.

4.1.1 Features

The back-end API exposes two POST methods: one for sending the RGB image to the server and the other one for receiving the processed RGB-D image. Due to hardware limitations, the images are resized at 500x500 pixels at most.

The front-end application allows the user to upload photos taken from the camera or from their gallery, visualize the images, send them to the server for processing, compare the results and save them on the device.

Take Photo

This functionality consists of pressing the ”Take Photo” button on the application screen which opens the device’s camera, allowing the user to take a photo that will then be displayed on the screen.

Upload Photo

This functionality consists of pressing the ”Upload Photo” button on the application screen which opens a dialog for the user to choose the application they want to use to choose a photo from the device (whether it’s default gallery, Google Photos or other applications), allowing the user to choose an image (only JPEG / JPG / PNG / BMP are allowed) that will then be displayed on the screen.

View Photo

This functionality is a passive one, meaning that the user does not directly interact with it, but it is used by other functionalities to better understand the result of previous actions. The photos to be send to the server will be shown on the screen and after the result is received, it will replace the initial photo.

Estimate Depth in Photo

This functionality consists of pressing the ”Submit” button then one of the buttons corresponding to each ML back-end architecture. The buttons also briefly present the strength of each model to inform the user about their next choice. When one of the buttons is clicked, then the application will send the image to the server, that will later send back the processed image and display it on the screen.

Compare Result to Initial Photo

This functionality consists of clicking on the image in order to swap between the initial RGB photo and the RGB-D result. When the user touches the image, the initial photo is shown and when they touch it again, the result is displayed as before. This was inspired by other photo editor applications that also have such features for better visualization of the differences.

Save Result

This functionality consists of pressing the ”Save Result” button that is visible only after the result arrived from the server. After the button is pressed, the application automatically saves the image to a ”/Pictures” folder as JPG files.

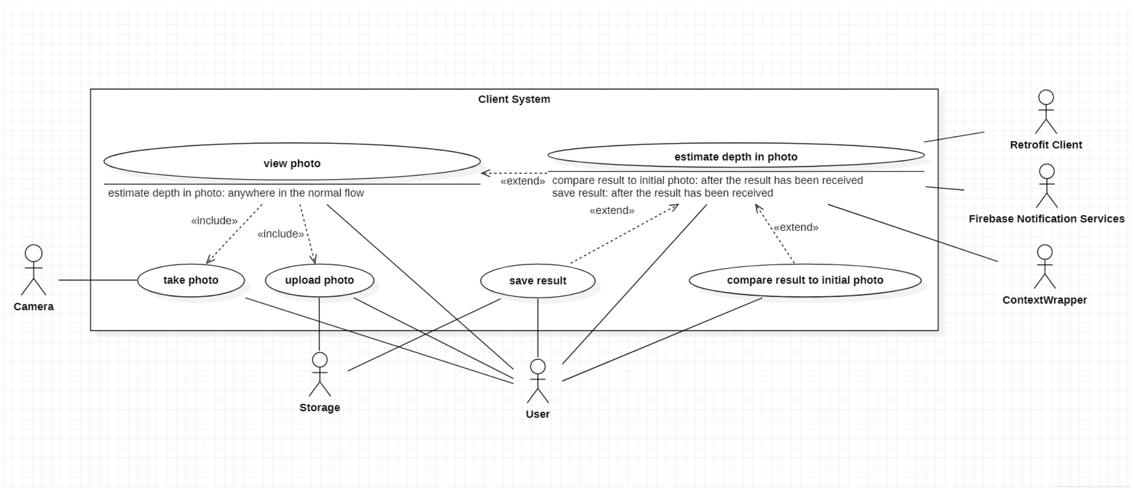

Client System
view photo
estimate depth in photo: anywhere in the normal flow
estimate depth in photo
compare result to initial photo: after the result has been received
save result: after the result has been received
«extend»
«include»
«include»
«extend»
«extend»
take photo
upload photo
save result
compare result to initial photo
Camera
Storage
User
Retrofit Client
Firebase Notification Services
ContextWrapper

Figure 4.1: Use Case Diagram

As seen in the diagram 4.1, The user has initially access to the two features for uploading an image that will be displayed on the screen. Then, if they want to, they can choose another image or submit the uploaded one for *Depth Estimation*. For sending the image, the application uses a Retrofit Client for managing the API calls, a ContextWrapper class to internally subscribe to messages sent on a certain channel between the main activity and messaging services and Firebase Notification Services to receive notifications on a certain topic that the application has subscribed to. After the result has been received, the user can visualize the differences between the initial photo and the result and save the result on their device.

4.2 Analysis and Design

4.2.1 Domain

The center concepts involved in estimating depth from one image consist of using a *Bitmap*, which is a matrix of *Pixels* that encode the three color channels, namely *Red*, *Green* and *Blue*. We do not consider *Alpha*, because the images we use have no transparent pixels. These bitmaps are then processed through a *Machine Learning* algorithm that infers the depth value for each pixel of the bitmap. That algorithm has been trained on DIODE dataset [VKZ+19] and fine-tuned by adjusting the values of its hyperparameters. Both the client-side and the server-side applications have in common the concept of bitmap, which is slightly different represented as we need to adapt the pixels format according to the stage of processing. While on the server-side the pixels are represented as tuples of red, green and blue values from 0 to 255, on the client-side, the list of pixels is flattened to a simple array of integers, or the Android graphics bitmap is used. All the previously mentioned functionalities work with different kinds of bitmaps and pixels, according to the specific context in which they are used.

4.2.2 Deployment Architecture

The Android application that runs on a portable device is composed of the main thread that is the UI thread and other background processes for listening to internal broadcasts or external notifications, or performing API calls. Over the network, the Communicator module

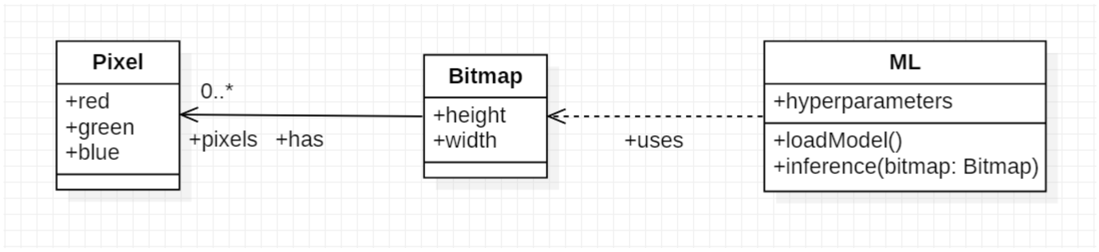


Figure 4.2: Analysis Diagram

receives API requests from the client and sends the information to the Processor module or retrieves data from the disk. The Spring framework implements the necessary concurrency for receiving and serving multiple requests at once. Nonetheless, the RabbitMQ Listener by default has concurrency implemented and is capable of handling multiple messages at once. Furthermore, in what concerns the Processor module, the application uses a pool of several threads and the only limitation is the GPU memory that would be easily consumed by multiple requests. However, if the application would have been hosted on a server with sufficient graphic cards, then the distributed GPU processing of the images would have dramatically decreased the requests' latency. According to the optimal configurations used by the ML models, we are aiming for at least four GPU-s with more than 25 GB of GPU memory.

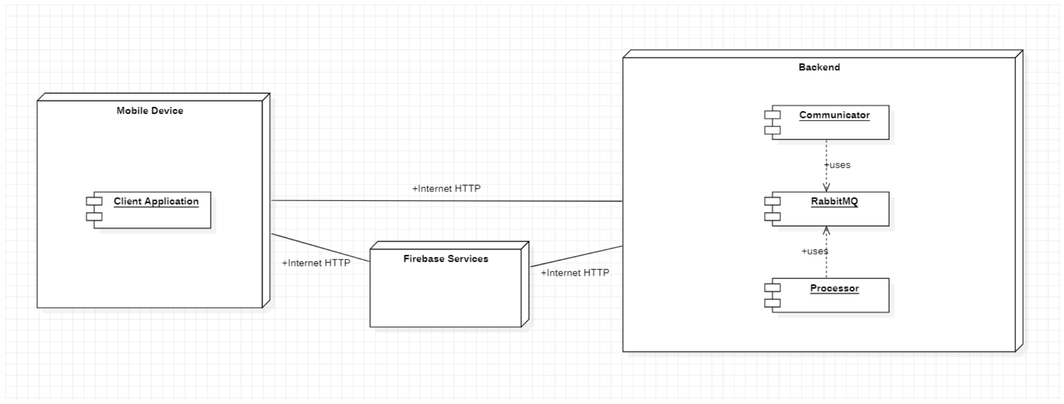


Figure 4.3: Deployment Diagram

As seen in figure 4.3, the applications communicate through the HTTP protocol over external networks for API calls and notifications. Moreover, in what regards the communication between the two server-side modules, the RabbitMQ message queues also use HTTP over the local area network. Two message queues are used for implementing a two-way communication so that the Communicator module represents a façade of for the entire processing done by the ML models.

4.2.3 Logical Architecture

On the one hand, the front-end application is structured in four layers - the model, the service, the util and the user interface packages. The MainActivity class lies in the root of the project and it can be considered the View in the MVC design pattern. On the other hand, the server-side applications are composed as follows.

As seen in figure 4.4, the MainActivity class is situated in the base package of the project and it uses all the other drawn packages. The model contains all the necessary classes for

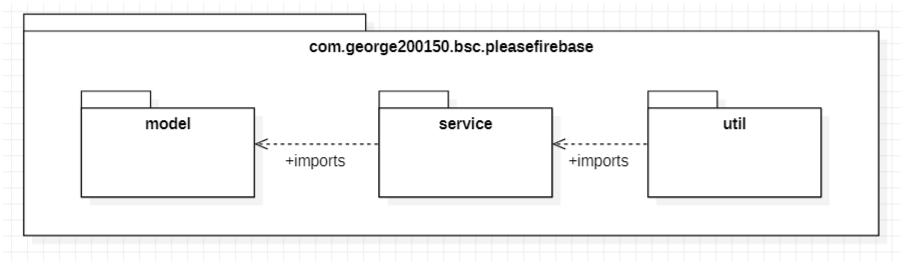


Figure 4.4: Android application Package Diagram

the bitmaps to be wrapped and exchanged through the network. The service contains the Firebase messaging service that always runs and listens for incoming notification via WebSockets and the Retrofit API interface used for communication with the Java server. The util package contains the factory method for creating a singleton Retrofit client.

The Communicator application written in Java has configuration, model, exception, persistence, service, util and controller packages. In the configuration package, the Spring Beans are instantiated and other classes are initialized for the RabbitMQ listener. The model and exception packages expose the main entities that we are working with in the application. The exceptions are separated from the model because they have a specific purpose that is more connected to the controller layer and handling errors in an explicit way so that the user receives useful feedback about their operation's status. The persistence layer consists of a Data Access Object that manages the bitmaps stored on the disk. The controller receives REST API requests and forwards them to the service, which is responsible with adapting the bitmaps according to the next stage of processing, posting messages onto the queue (which are done with the help of util classes), retrieving images stored locally and sending API requests to the Firebase notifications services.

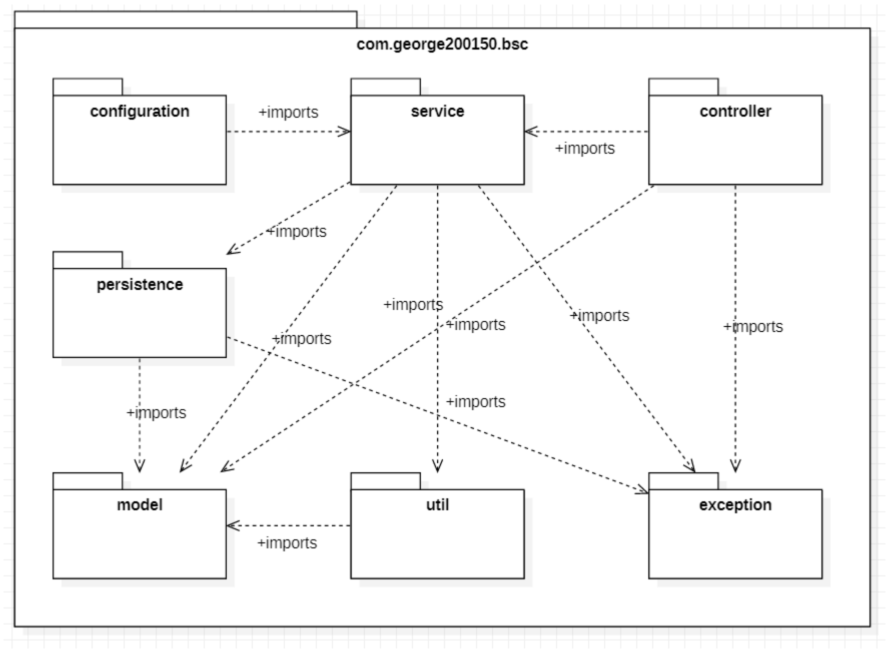


Figure 4.5: Java application Package Diagram

As figure 4.5 presents, the previous description matches the package diagram. What is to be underlined about the Java application is that it is the most complex component of the whole project in what concerns the architecture. Its design is inspired by *Service-Oriented Architecture* and it implements patterns such as adapter, DAO, oberserver and singleton.

The Processor application written in Python is the main focus of the research component and uses different *Semantic Segmentation* and *Monocular Depth Estimation* models. Each model lies in a separate package that bears their name. In the root of the project there is the script that contains all the classes responsible with handling the RabbitMQ messages, adapting bitmaps to the correct format and loading the desired ML model and inferring the results. Python is a multi-paradigm language, hence it would be as good to create a parsing script for the messages and use classes only for the ML models. Moreover, the style of the code would be cleaner when presenting the processing steps in a functional manner, by using a pipeline rather than encapsulating functions in classes when it is not necessary.

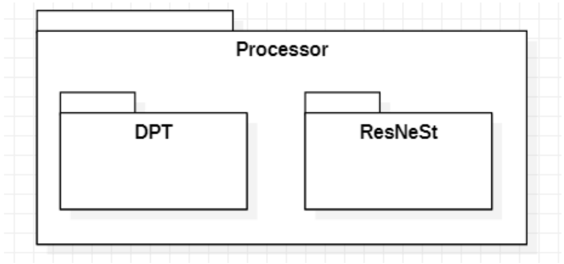


Figure 4.6: Python application Package Diagram

Figure 4.6 shows that the packages that contain the *Machine Learning* models are decoupled from the rest of the project as they should be. Each model is used only in the main script by calling their *load* and *inference* methods. Apart from those exposed methods, the models are considered black-boxes to the rest of the program because it would be very difficult to abstractly manage all the hyperparameters and other configuration presets of the neural networks. It is best to consider only the optimal configurations for the physical machine the algorithms run on and keep them encapsulated into their modules. Future improvements could refer to having a configuration script that allocates resources according to the available machine and manages the optimal settings in order to maximise the quality of the algorithms' results.

4.3 Detailed Design

In this section we will put forward the structure of the projects and interaction between the classes. The three applications follow a layered architecture, as mentioned before and implement various design patterns.

4.3.1 Client Application

The front-end application has less packages than the Java one because most of the work is done in the MainActivity class that, as observed in figure 4.7 has a lot of methods. However, some of those methods are overridden base methods that now have the behaviour we desired, others are extracted lambda functions for better sequence diagrams representation, or are functions that manage the user interface elements. Most of the use cases utilize only the MainActivity because all the features except for EstimateDepth are implemented only on the client side. Most of the interactions are facile and straightforward as they use camera, storage, or UI elements to be completed. Nevertheless, the image processing request is done by performing a two-step API request and further computations on the server-side.

Required Permissions

First of all, the application requires permissions to read and write storage and access network at runtine that the user must grant in order to be able to upload images.

Communication Channels

Then, because there is an internal broadcast system for sending the notification content from the Firebase notification service to the MainActivity, when initializing the application, a broadcast receiver is created and in the *prepareSendPhoto* method is registered in the context of the application in order to establish the communication channe between the service and the activity, and accepting only the messages sent on Intents with the *SubscriptionMessages.NOTIFICATION_ARRIVED* user-defined action.

Memory Management

Therefore, to prevent memory leaks, the manifest specifies that the application should not be destroyed when rotating the device and the *onStop* must be overridden to explicitly try to unregister the receiver in case the operating system decides to free memory occupied by the application.

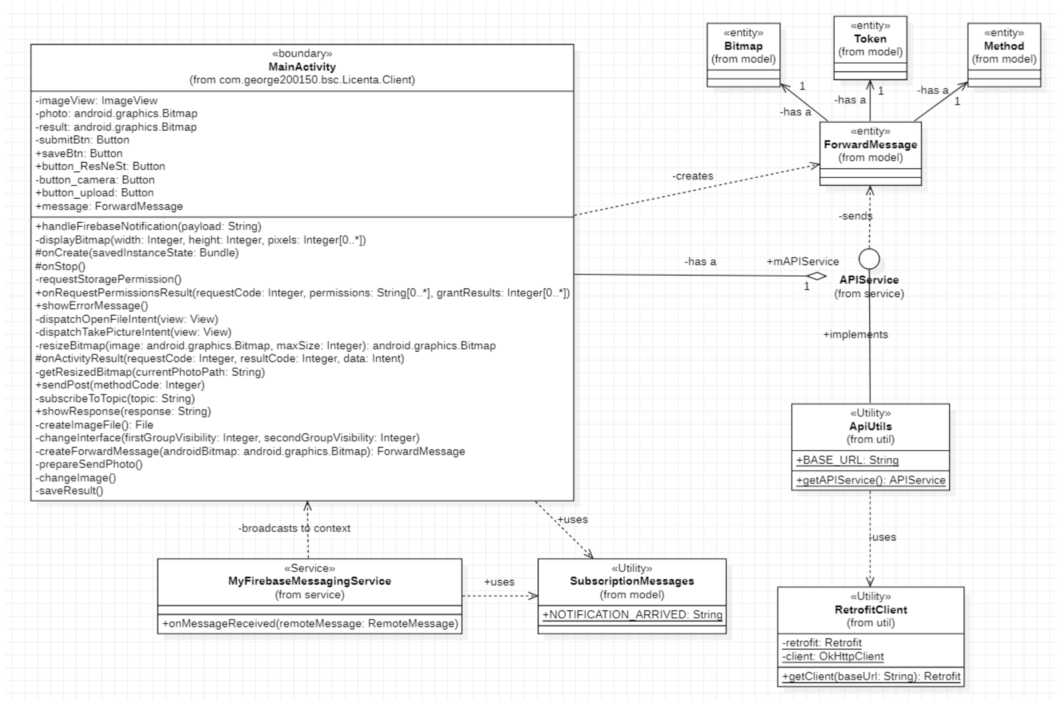

«boundary»
MainActivity
(from com.george200150.bsc.Licenta.Client)
-imageView: ImageView
-photo: android.graphics.Bitmap
-result: android.graphics.Bitmap
-submitBtn: Button
+saveBtn: Button
+button_ResNeSt: Button
-button_camera: Button
+button_upload: Button
+message: ForwardMessage
+handleFirebaseNotification(payload: String)
-displayBitmap(width: Integer, height: Integer, pixels: Integer[0..*])
#onCreate(savedInstanceState: Bundle)
#onStop()
-requestStoragePermission()
+onRequestPermissionsResult(requestCode: Integer, permissions: String[0..*], grantResults: Integer[0..*])
+showErrorMessage()
-dispatchOpenFileIntent(view: View)
-dispatchTakePictureIntent(view: View)
-resizeBitmap(image: android.graphics.Bitmap, maxSize: Integer): android.graphics.Bitmap
#onActivityResult(requestCode: Integer, resultCode: Integer, data: Intent)
-getResizedBitmap(currentPhotoPath: String)
+sendPost(methodCode: Integer)
-subscribeToTopic(topic: String)
+showResponse(response: String)
-createImageFile(): File
-changeInterface(firstGroupVisibility: Integer, secondGroupVisibility: Integer)
-createForwardMessage(androidBitmap: android.graphics.Bitmap): ForwardMessage
-prepareSendPhoto()
-changeImage()
-saveResult()
«entity»
Bitmap
(from model)
«entity»
Token
(from model)
«entity»
Method
(from model)
1
-has a
-has a
1
-has a
1
«entity»
ForwardMessage
(from model)
-creates
-sends
-has a
+mAPIService
1
APIService
(from service)
+implements
«Utility»
ApiUtils
(from util)
+BASE_URL: String
+getAPIService(): APIService
+uses
-broadcasts to context
«Service»
MyFirebaseMessagingService
(from service)
+onMessageReceived(remoteMessage: RemoteMessage)
+uses
«Utility»
SubscriptionMessages
(from model)
+NOTIFICATION_ARRIVED: String
+uses
«Utility»
RetrofitClient
(from util)
-retrofit: Retrofit
-client: OkHttpClient
+getClient(baseUrl: String): Retrofit

Figure 4.7: Class Diagram for Android Application

4.3.2 Communicator Server

The Java server applications represents a façade for the more complex processing that the images undergo. The main focus of the application lies on the REST API controller, the service for managing all the classes, the RabbitMQ message queue, the disk storage and the Firebase notifications services.

Application Flow

The requests for image processing are received at first as the *handlePostBitmap* method that retrieves the ForwardMessage and posts it onto the queue. Then, the rabbit listener handles the incoming result from the ML algorithm, saves it locally and sends the notification to the client that the result has been received. Then, the client sends the second request to get the Bitmap from the disk.

Design Patterns

While the front-end application implements *MVC* for the GUI, a *Singleton* pattern for lazy instantiating the Retrofit client, and uses built-in *Observers*, the Java server uses design patterns such as *Adapter* for handling different message formats, *DAO* for working with files on disk and the *Proxy* for managing messages posted onto the RabbitMQ message queue.

Error Handling

Because of the complexity of the application, there are many custom exception classes for handling potential errors. Some of the errors may be caused by JSON incorrect format, RabbitMQ faulty message posting, Firebase incompletely sent notifications, or disk access. Any thrown errors will be caught in the *ExceptionAdviceController* and sent to the users as HTTP Entities that will provide useful feedback on the result of the actions. Any other uncaught error refers to severe server errors or client errors (e.g. 400 bad request, 404 not found, 502 bad gateway). Since now, RabbitMQ and Firebase have not posed any problems when handling incoming requests, because their contents are managed by the application and any bad-intended content sent by the user would have been filtered by the time it reaches the Processor.

Security

First of all, the API accepts only ForwardMessage payloads which means that the message must respect that format and once posted on the queue, the Processor expects only integers that are not evaluated in any manner as a code or executed as script which further grants the robustness and consistency of the application security. Secondly, the second API request that contains a file path for the bitmap to be retrieved can contain malicious injected paths designed to get other content from the computer than the application intended that must be made inaccessible for the regular users. Hence, the string is checked not to contain any ".." or ":" in order to prevent uncovering forbidden content (i.e. "../../../Windows/System32", "C:/Windows/System32"). In order to uniquely identify a user's request, a special Token is created when sending the first API request, that contains the timestamp and the hash of the image. The Token is also used to subscribe to a Firebase Topic that the notifications will be sent to.

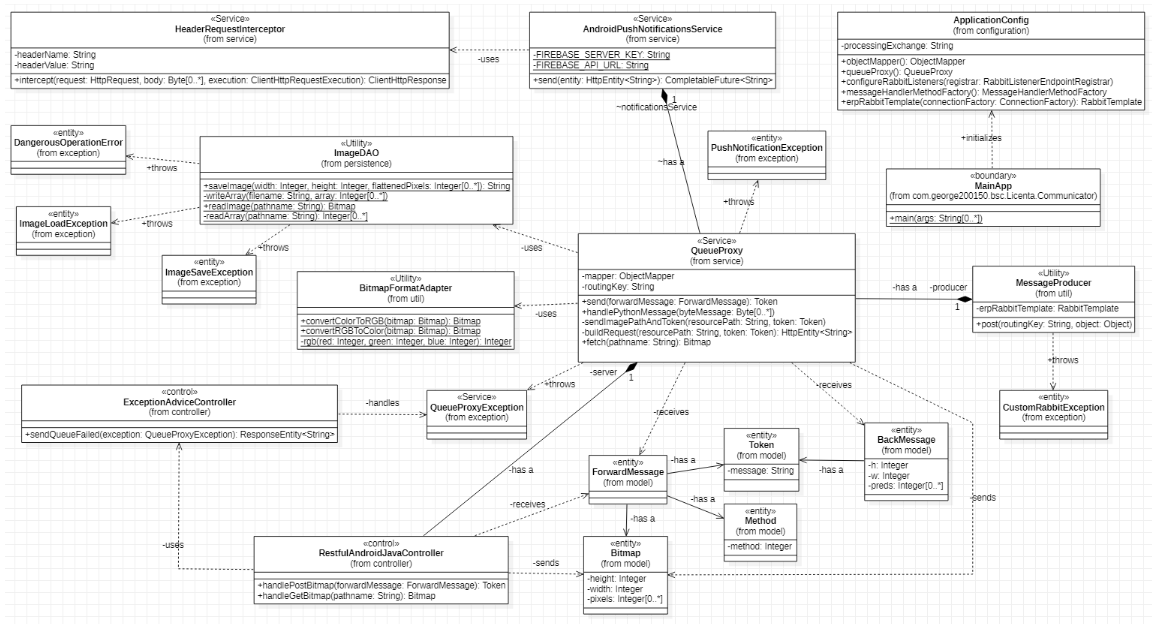

«Service»
HeaderRequestInterceptor
(from service)
-headerName: String
-headerValue: String
+intercept(request: HttpRequest, body: Byte[0..*], execution: ClientHttpRequestExecution): ClientHttpResponse
-uses
«Service»
AndroidPushNotificationsService
(from service)
-FIREBASE_SERVER_KEY: String
-FIREBASE_API_URL: String
+send(entity: HttpEntity<String>): CompletableFuture<String>
1
~notificationsService
~has a
ApplicationConfig
(from configuration)
-processingExchange: String
+objectMapper(): ObjectMapper
+queueProxy(): QueueProxy
+configureRabbitListeners(registrar: RabbitListenerEndpointRegistrar)
+messageHandlerMethodFactory(): MessageHandlerMethodFactory
+erpRabbitTemplate(connectionFactory: ConnectionFactory): RabbitTemplate
+initializes
«boundary»
MainApp
(from com.george200150.bsc.Licenta.Communicator)
+main(args: String[0..*])
«entity»
DangerousOperationError
(from exception)
+throws
«Utility»
ImageDAO
(from persistence)
+saveImage(width: Integer, height: Integer, flattenedPixels: Integer[0..*]): String
-writeArray(filename: String, array: Integer[0..*])
+readImage(pathname: String): Bitmap
-readArray(pathname: String): Integer[0..*]
«entity»
ImageLoadException
(from exception)
+throws
«entity»
ImageSaveException
(from exception)
+throws
-uses
«entity»
PushNotificationException
(from exception)
+throws
«Utility»
BitmapFormatAdapter
(from util)
+convertColorToRGB(bitmap: Bitmap): Bitmap
+convertRGBToColor(bitmap: Bitmap): Bitmap
-rgb(red: Integer, green: Integer, blue: Integer): Integer
-uses
«Service»
QueueProxy
(from service)
-mapper: ObjectMapper
-routingKey: String
+send(forwardMessage: ForwardMessage): Token
+handlePythonMessage(byteMessage: Byte[0..*])
-sendImagePathAndToken(resourcePath: String, token: Token)
-buildRequest(resourcePath: String, token: Token): HttpEntity<String>
+fetch(pathname: String): Bitmap
-has a
-producer
1
«Utility»
MessageProducer
(from util)
-erpRabbitTemplate: RabbitTemplate
+post(routingKey: String, object: Object)
+throws
«entity»
CustomRabbitException
(from exception)
«control»
ExceptionAdviceController
(from controller)
+sendQueueFailed(exception: QueueProxyException): ResponseEntity<String>
-handles
«Service»
QueueProxyException
(from exception)
+throws
-server
1
-receives
-receives
-has a
«entity»
ForwardMessage
(from model)
-has a
«entity»
Token
(from model)
-message: String
-has a
«entity»
BackMessage
(from model)
-h: Integer
-w: Integer
-preds: Integer[0..*]
-has a
«entity»
Method
(from model)
-method: Integer
-has a
-receives
-sends
-uses
«control»
RestfulAndroidJavaController
(from controller)
+handlePostBitmap(forwardMessage: ForwardMessage): Token
+handleGetBitmap(pathname: String): Bitmap
-sends
«entity»
Bitmap
(from model)
-height: Integer
-width: Integer
-pixels: Integer[0..*]

Figure 4.8: Class Diagram for Java Application

4.3.3 Processor Server

The Python application does not implement any security measures because the Java server acts like a firewall for it. It receives a binary serialized message via the RabbitMQ queue that is decoded into a JSON string then into a dictionary. The bitmap data is retrieved from the dictionary and processed by one of the *Depth Estimation* or *Semantic Segmentation* algorithms. All the algorithms share the same interface that allows the main python script to load the models and infer the desired depth or segmentation masks from the RGB images. As figure 4.9 depicts, the flow is rather simple and the main focus of this application is the ML black-boxes that cannot be represented in the diagrams due to their complexity.

GPU Management and Limitations

Each model has their requirements for GPU memory usage, so we decided to use images no bigger than 500x500 pixels in order to avoid *OutOfMemoryException* error, as the server runs on a machine with only 6 GB of GPU memory which can easily be consumed by any larger photo. Moreover, we consider that it is good to maintain a consistent scale across the models in order to better compare their baseline performances instead of adapting the bitmap size to fill the entire GPU which would be a rather tedious task and unpredictable, as the GPU is also used by other applications which would make it impossible to decide upon the maximum supported sizes for a certain neural network. Furthermore, we consider a lazy initialisation approach in order to adapt to the hardware limitations, so we load the models only when a request is needed so that the GPU memory is effectively managed.

Communication Methods

While on the Android and Java applications, the main components of the model were shared between them (i.e. Bitmap, Method, Token, ForwardMessage), Python and Java implicitly share the Forward- and BackMessage concepts. The Python application does not map the JSON to classes anymore because that would be an unnecessary overhead to the object. The goal of the RabbitMQ message handler is to transform the encoded bitmap into a *PIL image* that the ML models can eventually use. The resulted depth or segmentation maps are then encapsulated into a JSON dictionary and posted onto the RabbitMQ queue and mapped to *BackMessages* when intercepted by the Java *handlePythonMessage* method.

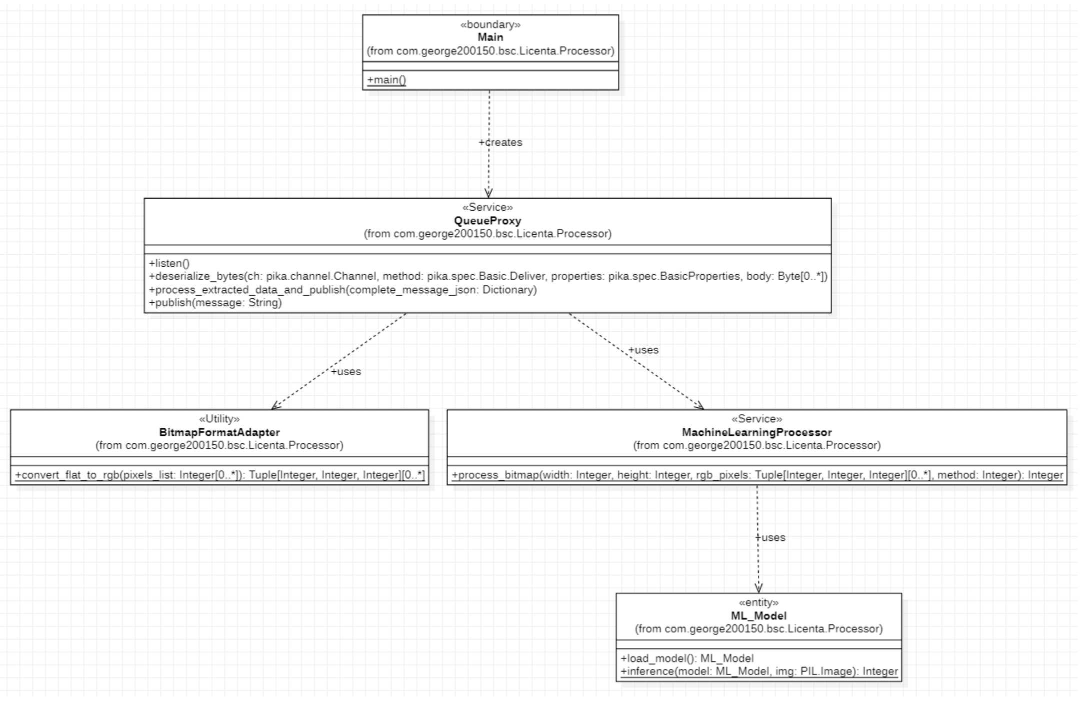

«boundary»
Main
(from com.george200150.bsc.Licenta.Processor)
+main()
+creates
«Service»
QueueProxy
(from com.george200150.bsc.Licenta.Processor)
+listen()
+deserialize_bytes(ch: pika.channel.Channel, method: pika.spec.Basic.Deliver, properties: pika.spec.BasicProperties, body: Byte[0..*])
+process_extracted_data_and_publish(complete_message_json: Dictionary)
+publish(message: String)
+uses
+uses
«Utility»
BitmapFormatAdapter
(from com.george200150.bsc.Licenta.Processor)
+convert_flat_to_rgb(pixels_list: Integer[0..*]): Tuple[Integer, Integer, Integer][0..*]
«Service»
MachineLearningProcessor
(from com.george200150.bsc.Licenta.Processor)
+process_bitmap(width: Integer, height: Integer, rgb_pixels: Tuple[Integer, Integer, Integer][0..*], method: Integer): Integer
+uses
«entity»
ML_Model
(from com.george200150.bsc.Licenta.Processor)
+load_model(): ML_Model
+inference(model: ML_Model, img: PIL.Image): Integer

Figure 4.9: Class Diagram for Python Application

Take Photo

This feature uses only the MainActivity class, as it is triggered by a click on the *Take Photo* button as seen in the sequence diagram 4.10. The *onClick* event is handled in the MainActivity by dispatching a camera intent whose result is handled in *onActivityResult* method where the temporarily stored bitmap is used to initialize the *photo* field and then display it on screen in the *imageView*.

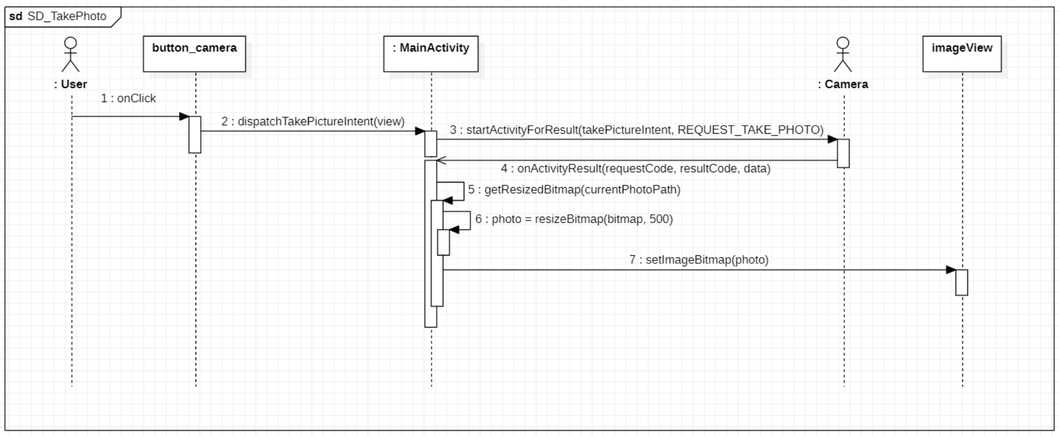

sd SD_TakePhoto
: User
button_camera
: MainActivity
: Camera
imageView
1 : onClick
2 : dispatchTakePictureIntent(view)
3 : startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO)
4 : onActivityResult(requestCode, resultCode, data)
5 : getResizedBitmap(currentPhotoPath)
6 : photo = resizeBitmap(bitmap, 500)
7 : setImageBitmap(photo)

Figure 4.10: Sequence Diagram for TakePhoto Use Case

Upload Photo

Uploading a photo is silimar to taking one. The difference is that instead of calling a *dispatchTakePictureIntent*, the *onClick* event triggered by clicking the *Upload Photo* button calls a *dispatchOpenFileIntent* that allows the user to choose the application to manage the photos and choose one which will then be used to initialize the *photo* field and be displayed on the screen. An important notice is that in MainActivity, the overridden *onActivityResult* checks the type of the image to be valid (i.e. .jpeg, .png and .bmp only). As observed in figures 4.10 and 4.11, the UploadPhoto use case resembles TakePhoto.

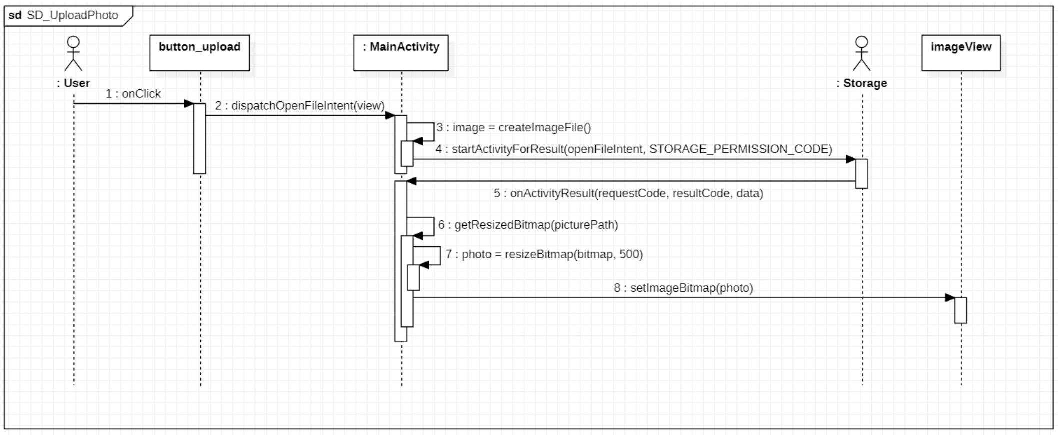

sd SD_UploadPhoto
: User
button_upload
: MainActivity
: Storage
imageView
1 : onClick
2 : dispatchOpenFileIntent(view)
3 : image = createImageFile()
4 : startActivityForResult(openFileIntent, STORAGE_PERMISSION_CODE)
5 : onActivityResult(requestCode, resultCode, data)
6 : getResizedBitmap(picturePath)
7 : photo = resizeBitmap(bitmap, 500)
8 : setImageBitmap(photo)

Figure 4.11: Sequence Diagram for UploadPhoto Use Case

View Photo

The use case of viewing a photo is included in the other flows, being always initialized by other events. MainActivity contains the field *imageView* that is the subject of this use case. Every *photo* to be displayed on the screen will call *imageView.setImageBitmap(photo)*.

Estimate Depth in Photo

For estimating the depth in a photo, the client application must communicate with the servers that the ML algorithms run on. It would be almost impossible to run such DL models on a mobile device because of the complexity of the networks and intensive memory usage that would drain the battery. For sending a photo, first the user must upload one in the application by whether taking a photo or choosing one from the gallery. Then, when the user clicks on the *Submit* button and the chosen image is valid, the triggered event will call *prepareSendPhoto* that registers the broadcast receiver to the action type defined in *SubscriptionMessages*, creates a *ForwardMessage* that wraps the *Bitmap* and shows the buttons for every ML model. The sequence diagram 4.12 further considers that the user pressed *button_ResNeSt*. As a result, the buttons will disappear and the previous interface will be displayed in place with the *Submit* button disabled. Instead of the *imageView*, a spinner will be displayed while the application waits for the results to come. Then, the *ForwardMessage* that now has set the *Method* to 0 since ResNest has been chosen, will be sent to the server via the *mAPIService.sendBitmapPOST(message)* API request available from *APIService*, using the Retrofit client initialized in *ApiUtils*. The back-end discussion will be held on the 4.13 and 4.14 diagrams. After the first call finishes, the unique token is received as a response to ensure the completion of the request. Then, the application waits to receive a Firebase notification containing the resounce path on the server disk. When received, *MyFirebaseMessagingService* sends a broadcast to the *MainActivity* that bears the bitmap path. Once the broadcast received, *handleFirebaseNotification* is called that unregisters the receiver from other internal broadcasts and performs the second API call *mAPIService.sendFetchPOST(payload)*, now requesting the *Bitmap* by its resource name. The progress spinner can now be eventually hidden and the processed photo received will be displayed in the *imageView* and the *Save Result* button will be displayed.

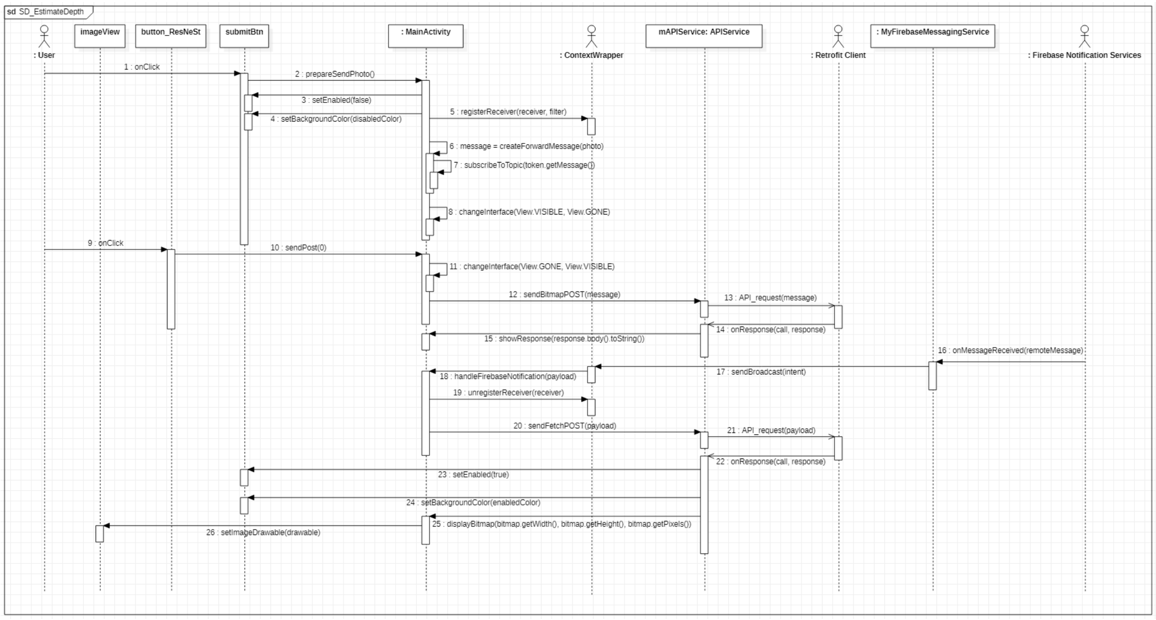

sd SD_EstimateDepth
: User
imageView
button_ResNeSt
submitBtn
: MainActivity
: ContextWrapper
mAPIService: APIService
: Retrofit Client
: MyFirebaseMessagingService
: Firebase Notification Services
1 : onClick
2 : prepareSendPhoto()
3 : setEnabled(false)
4 : setBackgroundColor(disabledColor)
5 : registerReceiver(receiver, filter)
6 : message = createForwardMessage(photo)
7 : subscribeToTopic(token.getMessage())
8 : changeInterface(View.VISIBLE, View.GONE)
9 : onClick
10 : sendPost(0)
11 : changeInterface(View.GONE, View.VISIBLE)
12 : sendBitmapPOST(message)
13 : API_request(message)
14 : onResponse(call, response)
15 : showResponse(response.body().toString())
16 : onMessageReceived(remoteMessage)
17 : sendBroadcast(intent)
18 : handleFirebaseNotification(payload)
19 : unregisterReceiver(receiver)
20 : sendFetchPOST(payload)
21 : API_request(payload)
22 : onResponse(call, response)
23 : setEnabled(true)
24 : setBackgroundColor(enabledColor)
25 : displayBitmap(bitmap.getWidth(), bitmap.getHeight(), bitmap.getPixels())
26 : setImageDrawable(drawable)

Figure 4.12: Sequence Diagram for Android EstimateDepth Use Case

Having considered the client-side flow, the back-end follows the following actions. The *handlePostBitmap* method intercepts the first API request, forwards the request body to *QueueProxy* which formats the *Bitmap* by using *BitmapFormatAdapter.convertColorToRGB*, then maps the *ForwardMessage* to a JSON string which the *MessageProducer* serializes and posts onto the RabbitMQ *Licenta.PythonQueue*. Then the processing takes place which will later be explained in diagram 4.14. Once the processing is complete, a message will be received from the RabbitMQ *Licenta.JavaQueue* and handled by *handlePythonMessage* method, that deserializes the message, maps it to a *BackMessage*, saves the retrieved processed bitmap by calling *ImageDAO.saveImage* with the image data and sends the API request to the Firebase notification services in order to acknowledge the client that the processed resource is now available. This is done by calling *sendImagePathAndToken* which further builds the HTTP request body and uses *AndroidPushNotificationsService* to send it. Nonetheless, the *notificationsService* must also mention the authorization and content-type headers before posting the request, so the *RestTemplate* class is informed of them by having added the *HeaderRequestInterceptor* interceptors with the corresponding data. Then, after the first request finished the client will send their second request that triggers *handleGetBitmap* which forwards the *pathname* to *QueueProxy* which reads the image data from the disk after having inspected the resource path not to contain any restricted symbols. Once the *Bitmap* is obtained, *BitmapFormatAdapter* will format the image to be compatible with *Android.graphics.Bitmap* so that the photo can be displayed on the screen. This is the last step that marks the end of the series of request if nothing threw any errors. An important notice is that the client must always have a good connection to the internet to perform all the previously mentioned actions. The application is not designed to have any interruptions in network connection because it may miss important notifications or request responses. Moreover, the user must keep the application running in the foreground, as notifications cannot be received when it is stopped.

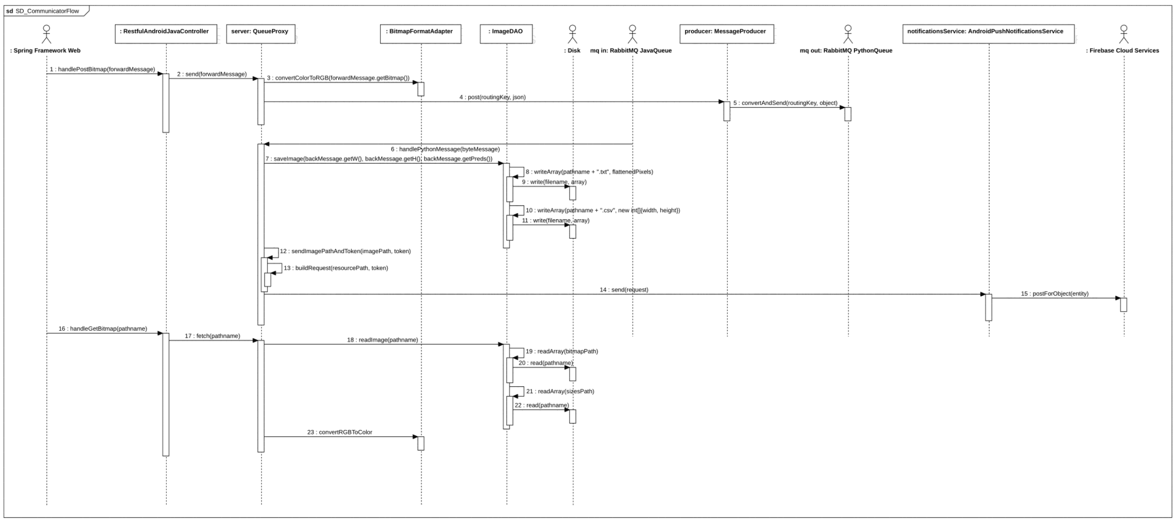

sd SD_CommunicatorFlow
: Spring Framework Web
: RestfulAndroidJavaController
server: QueueProxy
: BitmapFormatAdapter
: ImageDAO
: Disk
mq in: RabbitMQ JavaQueue
producer: MessageProducer
mq out: RabbitMQ PythonQueue
notificationsService: AndroidPushNotificationsService
: Firebase Cloud Services
1 : handlePostBitmap(forwardMessage)
2 : send(forwardMessage)
3 : convertColorToRGB(forwardMessage.getBitmap())
4 : post(routingKey, json)
5 : convertAndSend(routingKey, object)
6 : handlePythonMessage(byteMessage)
7 : saveImage(backMessage.getW(), backMessage.getH(), backMessage.getPreds())
8 : writeArray(pathname + ".txt", flattenedPixels)
9 : write(filename, array)
10 : writeArray(pathname + ".csv", new int[]{width, height})
11 : write(filename, array)
12 : sendImagePathAndToken(imagePath, token)
13 : buildRequest(resourcePath, token)
14 : send(request)
15 : postForObject(entity)
16 : handleGetBitmap(pathname)
17 : fetch(pathname)
18 : readImage(pathname)
19 : readArray(bitmapPath)
20 : read(pathname)
21 : readArray(sizesPath)
22 : read(pathname)
23 : convertRGBToColor

Figure 4.13: Sequence Diagram for Java EstimateDepth Use Case

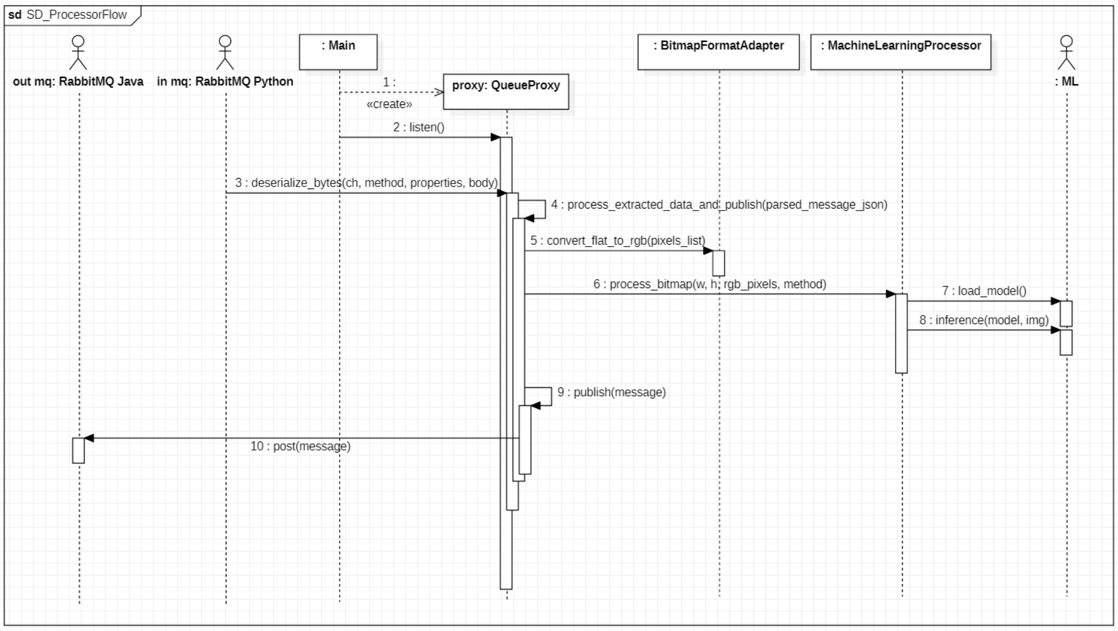

sd SD_ProcessorFlow
out mq: RabbitMQ Java
in mq: RabbitMQ Python
: Main
proxy: QueueProxy
: BitmapFormatAdapter
: MachineLearningProcessor
: ML
1 :
«create»
2 : listen()
3 : deserialize_bytes(ch, method, properties, body)
4 : process_extracted_data_and_publish(parsed_message_json)
5 : convert_flat_to_rgb(pixels_list)
6 : process_bitmap(w, h, rgb_pixels, method)
7 : load_model()
8 : inference(model, img)
9 : publish(message)
10 : post(message)

Figure 4.14: Sequence Diagram for Python EstimateDepth Use Case

In the sequence diagram 4.14 the flow is observed since the beginning of the application start-up. The *Main* class creates a new object, *proxy* that is responsible with handling the RabbitMQ messages. The *proxy* will be continuously listening to the RabbitMQ *Licenta.PythonQueue* and post processed bitmaps onto the RabbitMQ *Licenta.JavaQueue*. Firstly, when a message arrives, it is deserialized and mapped to a JSON dictionary. The *bitmap* contains all the pixels that *BitmapFormatAdapter* converts from a flat array of integers to a list of tuples of three integers, representing RGB pixels. That format is required by *PIL.Image* which is used by all the ML models. Now, *MachineLearningProcessor* can start processing the bitmap according to the selected method. It loads the model and calls the *inference* method to obtain the processed depth map. Then, once all processing is done, *proxy* receives the data and posts it onto the message queue.

Compare Result to Initial Photo

Comparing two photos is a rather straightforward task as it requires just a couple of presses on the screen to be completed. The MainActivity's *imageView* has a *onClickListener* that calls *changeImage* every click it records. That method checks if there are two photos initialized on the client (i.e. the processed result has come from the server) and swaps the bitmaps stored in *result* and *photo* fields. Diagram 4.15 traces the two *onClick* events triggered by pressing on the area allocated by the *imageView*.

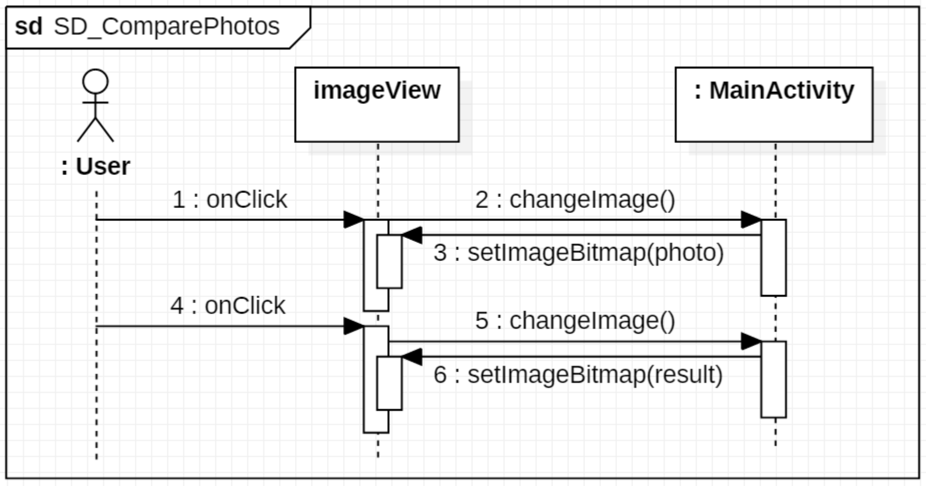
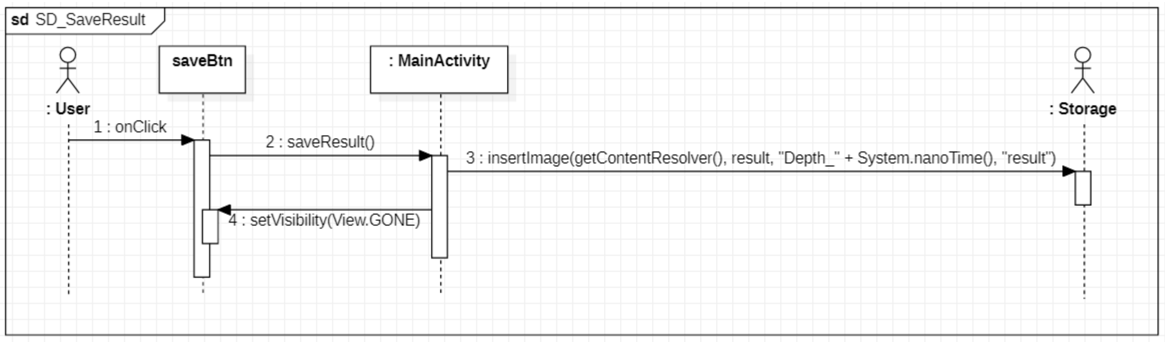


Figure 4.15: Sequence Diagram for ComparePhotos Use Case



sd SD_SaveResult
: User
saveBtn
: MainActivity
: Storage
1 : onClick
2 : saveResult()
3 : insertImage(getContentResolver(), result, "Depth_" + System.nanoTime(), "result")
4 : setVisibility(View.GONE)

Figure 4.16: Sequence Diagram for SaveResult Use Case

Save Result

In order to save the result, the user must have already sent an image for processing and the result must have been received. When the processed bitmap is received after the API request ended successfully, the *Save Result* button appears on the screen and when pressed it triggers the *saveResult* function that uses the Android API 29 *MediaStore.Images.Media* to save the bitmap via *insertImage* call in the device gallery. After the image has been successfully saved, the button will disappear because it would not make sense to just save the same photo twice. Diagram 4.16 displays the rather simple action of saving the image, as most of the work is handled by the internal storage.

Conclusions and future work

This thesis investigated two problems from computer vision and educational data mining.

First, it investigated three unsupervised learning models, *self-organizing maps*, *principal component analysis* and *relational association rule mining*, in the context of analysing data sets related to students' academic performance. Experiments performed on a real data set collected from Babeş-Bolyai University, Romania highlighted the potential of unsupervised learning based data mining tools to detect meaningful patterns regarding the academic performance of students.

Future work will be performed in order to extend the experiments and the analysis of the obtained results. For increasing the performance of the unsupervised learning process, methods for detecting anomalies and outliers in data will be further investigated. In addition, a post-processing phase for filtering the set of mined RARs will be analysed for removing rules which overlap with multiple classes.

Lastly, it analysed four feature sets (one of handcrafted lower-level features and three of automatically extracted features from a supervised deep learning model's encoder, the latter set being augmented with depth information) in both UL and SL manners that highlight the benefit of using depth-maps information in the context of indoor-outdoor image classification. Moreover, we have shown the semantic segmentation transfer learning potential towards classification tasks.

Further investigations will be performed using more diverse and larger data sets to guarantee the models do not overfit. Furthermore, considering the highlighted potential of DL models in the context of transfer learning through the experiments using DPT, we could additionally adapt and retrain models from scratch in order to perform other tasks than they were designed for, through a process we name *architecture transfer*. These extensive experiments would examine the feature extractors of multiple DL architectures so that the most general ones having a wider application range would be then employed in retraining the models for other tasks. The results would help create universal models that could serve as baselines across multiple domains and simplify comparisons between them.

References

[AC20] S. Abirami and P. Chitra. Chapter fourteen - energy-efficient edge based real-time healthcare support system. In Pethuru Raj and Preetha Evangeline, editors, *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in Computers*, pages 339–368. Elsevier, 2020.

[AND09] Elizabeth Ayers, Rebecca Nugent, and Nema Dean. A comparison of student skill knowledge estimates. In *Educational Data Mining - EDM 2009, Cordoba, Spain, July 1-3, 2009. Proceedings of the 2nd International Conference on Educational Data Mining.*, pages 1–10, 2009.

[BAW20] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. *CoRR*, abs/2011.14141, 2020.

[BCR18] Alejandro Bogarín, Rebeca Cerezo, and Cristóbal Romero. A survey on educational process mining. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 8(1), 2018.

[Bho19] Amlaan Bhoi. Monocular depth estimation: A survey. *CoRR*, abs/1901.09402, 2019.

[BK20] Juan Luis Gonzalez Bello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with MED probability volumes. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[BKK20] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale deep equilibrium models. In *Proceedings of 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 1–17, Vancouver, Canada, 2020.

[BR18] Cédric Beaulac and Jeffrey S. Rosenthal. Predicting university students' academic success and major using random forests. *arXiv:1802.03418*, pages 1 – 23, 2018.

[CBC12] Gabriela Czibula, Maria-Iuliana Bocicor, and Istvan Gergely Czibula. Promoter sequences prediction using relational association rule mining. *Evolutionary Bioinformatics*, 8:181–196, 04 2012.

[CC19] George Ciubotariu and Liana Maria Crivei. Analysing the academic performance of students using unsupervised data mining. *Studia Universitatis Babes-Bolyai Series Informatica*, 64(2):34–48, 2019.

[CCCD20] Liana Maria Crivei, Gabriela Czibula, George Ciubotariu, and Mariana Dindelegan. Unsupervised learning based mining of academic data sets for students' performance analysis. In *14th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2020, Timisoara, Romania, May 21-23, 2020*, pages 11–16. IEEE, 2020.

[CcM06] Alina Câmpan, Gabriela Şerban, and Andrian Marcus. Relational association rules and error detection. *Studia Universitatis Babes-Bolyai Informatica*, LI(1):31–36, 2006.

[CcTM06] Alina Campan, Gabriela Şerban, Traian Marius Truta, and Andrian Marcus. An algorithm for the discovery of arbitrary length ordinal association rules. In *DMIN*, pages 107–113, 2006.

[CKC20] Sungha Choi, Joanne Taery Kim, and Jaegul Choo. Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9370–9380. IEEE, 2020.

[CLCH16] H. Y. Chang, J. C. Lin, M. L. Cheng, and S. C. Huang. A novel incremental data mining algorithm based on fp-growth for big data. In *2016 International Conference on Networking and Network Applications (NaNA)*, pages 375–378, July 2016.

[CMM20] Kyle J. Cantrell, Craig D. Miller, and Carlos Morato. Practical depth estimation with image segmentation and serial u-nets. In Karsten Berns, Markus Helfert, and Oleg Gusikhin, editors, *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2020, Prague, Czech Republic, May 2-4, 2020*, pages 406–414. SCITEPRESS, 2020.

[CNI14] Stevica Cvetkovic, Sasa Nikolic, and Slobodan Ilic. Effective combining of color and texture descriptors for indoor-outdoor image classification. *Facta universitatis - series: Electronics and Energetics*, 27:399–410, 01 2014.

[CTC21] George Ciubotariu, Vlad-Ioan Tomescu, and Gabriela Czibula. Enhancing the performance of image classification through features automatically learned from depth-maps. In *13th International Conference on Computer Vision Systems, ICVS 2021, Virtual Conference, September 22-24, 2021*, pages 1–12. IEEE, 06 2021.

[DAIM15] Ashish Dutt, Saeed Aghabozrgi, Maizatul Akmal Binti Ismail, and Hamidreza Mahroeian. Clustering algorithms applied in educational data mining. *Intern. J. of Information and Electronics Engineering*, 5(2):112–116, May 2015.

[dat18] Academic data set, 2018. http://www.cs.ubbcluj.ro/∼liana.crivei/ Academic-DataSets/ThirdDataSet.txt.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009),* *20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.

[DLZS20] Ionut Cosmin Duta, Li Liu, Fan Zhu, and Ling Shao. Pyramidal convolution: Rethinking convolutional neural networks for visual recognition. *CoRR*, abs/2006.11538:1–16, 2020.

[EDB08] N. Elfelly, J.-Y. Dieulot, and P. Borne. A neural approach of multimodel representation of complex processes. *International Journal of Computers, Communications & Control*, III(2):149–160, 2008.

[GAFB19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3827–3837. IEEE, 2019.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GCZW21] Ente Guo, Zhifeng Chen, Yanlin Zhou, and Dapeng Oliver Wu. Unsupervised learning of depth and camera pose with feature map warping. *Sensors*, 21(3):923, 2021.

[GZC09] Qiong Gu, Li Zhu, and Zhihua Cai. Evaluation measures of the classification performance of imbalanced data sets. In *Computational Intelligence and Intelligent Systems*, pages 461–471. Springer Berlin Heidelberg, 2009.

[HQC$^+$20] Kang Huang, Xingtian Qu, Shouqian Chen, Zhen Chen, Wang Zhang, Haogang Qi, and Fengshang Zhao. Superb monocular depth estimation based on transfer learning and surface normal guidance. *Sensors*, 20(17):4856, 2020.

[JCSL18] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson W. H. Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 55–71. Springer, 2018.

[JRHR15] Syed Tanveer Jishan, Raisul Islam Rashu, Naheena Haque, and Rashedur M. Rahman. Improving accuracy of students' final grade prediction model using optimal equal width binning and synthetic minority over-sampling technique. *Decision Analytics*, 2(1):1, Mar 2015.

[KCHK18] Namil Kim, Yukyung Choi, Soonmin Hwang, and In So Kweon. Multispectral transfer network: Unsupervised depth estimation for all-day vision. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6983–6991. AAAI Press, 2018.

[KK96] S. Kaski and T. Kohonen. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets*, pages 498–507. World Scientific, 1996.

[Kur08] Wattanapong Kurdthongmee. Utilization of a self organizing map as a tool to study and predict the success of engineering students at walailak university. *Walailak Journal of Science and Technology*, 5(1):111–123, 2008.

[LHKS19] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *CoRR*, abs/1907.10326, 2019.

[LLW$^+$20] Xiaoyang Lyu, Liang Liu, Mengmeng Wang, Xin Kong, Lina Liu, Yong Liu, Xinxin Chen, and Yi Yuan. Hr-depth: High resolution self-supervised monocular depth estimation. *CoRR*, abs/2012.07356, 2020.

[LO92] J. Lampinen and E. Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(3):261–272, 1992.

[Mar12] B.R. Martin. Chapter 6 - sampling distributions associated with the normal distribution. In B.R. Martin, editor, *Statistics for Physical Science*, pages 105–122. Academic Press, Boston, 2012.

[MT13] Siti Khadijah Mohamad and Zaidatun Tasir. Educational data mining: A review. *Procedia - Social and Behavioral Sciences*, 97:320 – 324, 2013.

[NMYL19] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Trans. Graph.*, 38(6):184:1–184:15, 2019.

[OTOK15] Oyebade K. Oyedotun, Samuel Tackie, Ebenezer Olaniyi, and Adnan Khashman. Data mining of students' performance: Turkish students as a case study. *International Journal of Intelligent Systems Technologies and Applications*, 7(9):20–27, 2015.

[PATM18] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on CPU. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 5848–5854. IEEE, 2018.

[Ped11] F. et al. Pedregosa. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PP13] Ajay Kumar Pal and Saurabh Pal. Analysis and mining of educational data for predicting the performance of students. *International Journal of ElectronicsCommunication and Computer Engineering*, 4(5):2278–4209, 2013.

[PS05] Andrew Payne and Sameer Singh. Indoor vs. outdoor scene classification in digital photographs. *Pattern Recognit.*, 38(10):1533–1545, 2005.

[PZM12] Suhem Parack, Zain Zahid, and Fatima Merchant. Application of data mining in educational databases for predicting academic trends and patterns. In *22012 IEEE International Conference on Technology Enhanced Education (ICTEE)*, pages 1–4, 2012.

[RBK21] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, pages 1–15, 2021.

[RRDR13] R. Raja, S. Md. Mansoor Roomi, D. Dharmalakshmi, and S. Rohini. Classification of indoor/outdoor scene. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–4, 2013.

[SCC06] Gabriela Serban, Alina Câmpan, and Istvan Gergely Czibula. A programming interface for finding relational association rules. *Intern. Journal of Computers, Communications & Control*, I(S.):439–444, June 2006.

[Sci21] Scikit-learn. Machine learning in Python, 2021. http://scikit-learn.org/stable/.

[SK99] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10:151–159, 1999.

[SLK21] M. Song, S. Lim, and W. Kim. Monocular depth estimation using laplacian pyramid-based depth residuals. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2021.

[SP98] Martin Szummer and Rosalind W. Picard. Indoor-outdoor image classification. In *1998 International Workshop on Content-Based Access of Image and Video Databases, CAIVD 1998, Bombay, India, January 3, 1998*, pages 42–51. IEEE Computer Society, 1998.

[SYDY20] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, volume 12364 of *Lecture Notes in Computer Science*, pages 572–588. Springer, 2020.

[TDD$^+$17] Thi-Oanh Tran, Hai-Trieu Dang, Viet-Thuong Dinh, Thi-Minh-Ngoc Truong, Thi-Phuong-Thao Vuong, and Xuan-Hieu Phan. Performance prediction for students: A multi-strategy approach. *CYBERNETICS AND INFORMATION TECHNOLOGIES*, 17(2):164 – 182, 2017.

[TMR15] Waleed Tahir, Aamir Majeed, and T. Rehman. Indoor/outdoor image classification using gist image features and neural network classifiers. *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, pages 1–5, 2015.

[TSYW06] Zhehang Tong, Dianxi Shi, Bingzheng Yan, and Jing Wei. A review of indoor-outdoor scene classification. In *Proceedings of the 2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017)*, pages 469–474. Atlantis Press, 2017/06.

[USS$^+$17] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 11–20. IEEE Computer Society, 2017.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[VKZ$^+$19] Igor Vasiljevic, Nicholas I. Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A dense indoor and outdoor depth dataset. *CoRR*, abs/1908.00463:1–8, 2019.

[WCG$^+$20] Kaixuan Wang, Yao Chen, Hengkai Guo, Linfu Wen, and Shaojie Shen. Geometric pretraining for monocular depth estimation. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 4782–4788. IEEE, 2020.

[XZZ$^+$20] Mingkang Xiong, Zhenghong Zhang, Weilin Zhong, Jinsheng Ji, Jiyuan Liu, and Huilin Xiong. Self-supervised monocular depth and visual odometry learning with scale-consistent geometric constraints. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 963–969. ijcai.org, 2020.

[YHLC20] Chia-Hung Yeh, Yao-Pao Huang, Chih-Yang Lin, and Chuan-Yu Chang. Transfer2depth: Dual attention network with transfer learning for monocular depth estimation. *IEEE Access*, 8:86081–86090, 2020.

[YLSY19] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 5683–5692. IEEE, 2019.

[YYC$^+$20] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV*, volume 12360 of *Lecture Notes in Computer Science*, pages 191–207. Springer, 2020.

[ZWZ$^+$20] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander J. Smola. Resnest: Split-attention networks. *CoRR*, abs/2004.08955:1–12, 2020.

[ZZP$^+$17] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5122–5130. IEEE Computer Society, 2017.