university of groningen / faculty of mathematics and natural sciences / artificial intelligence

MORE: Simultaneous Multi-View 3D Object Recognition and Pose Estimation

Final Year Project
(Computational Intelligence and Robotics)

Tommaso Parisotto (s3866033)

March 17, 2021

Internal Supervisors:
S.H. Mohades Kasaei, PhD (Artificial Intelligence, University of Groningen)
M.A. Wiering, PhD (Artificial Intelligence, University of Groningen)

Artificial Intelligence
University of Groningen, The Netherlands

Abstract

In computer vision the problem of 3D object recognition has been tackled by many successful algorithms. In particular, in recent times, convolutional neural networks based on multi-view approaches figure among the best performing classifiers on 3D object datasets such as Princeton ModelNet, constituting de facto state-of-the-art. Although the research on the architectures for such networks has been widely explored, there has been little investigation on the benefits of employing best-views estimation algorithms to select the views for prediction. Many objects have regular 3D shapes which carry symmetrical features and sets of projections of such objects carry redundancy of information which can be reduced selecting only very informative views. To demonstrate the employability of best-view selection algorithms, we propose an entropy estimation model to retrieve best-views for a multi-view convolutional neural network to perform simultaneous recognition and pose estimation on 3D objects. We demonstrated that our model learns features to generate entropy maps that approximate closely the entropy evaluation of depth-images projections of a 3D object. With such evaluation we select a small number of highly-informative camera poses to observe the object in space. We demonstrated that the views obtained from such positions are descriptive enough to achieve accuracy scores comparable to other state-of-the-art approaches on the ModelNet10 dataset.

Summary

List of Figures

1 Introduction

Recent object recognition and pose estimation approaches are designed based on Convolutional Neural Networks. These approaches can be categorized into three main categories, volume-based, point-based, and view-based approaches. Volume-based approaches use volumetric representation of data, in particular they employ voxels to obtain a discrete representation of objects and define a data structure. These networks learn features from the occupancy grids which represent the objects in space. Point-based approaches are popular with data retrieved with 2.5D depth sensors. These sensors capture a dense set of depth samples from the scene, representing the surface of the objects as a collection of points in the Euclidean space named 'Point Clouds'. PCs can be easily processed into 2D matrices, therefore they are one of the most efficient representations for deep learning. Point-based neural networks learn features about the positional relations between points on the surfaces of objects. View-based approaches use one or more images representations of the objects, usually captured with a camera from a specific viewpoint. CNNs trained on such representations learn features from the visible attributes of the objects. Among these approaches, it has been shown that view-based methods outperformed other methods and achieved better performance. In most view-based approaches, the number of virtual cameras is predefined in advance, and therefore, they are not optimized for real-time applications. In this research we propose a deep learning approach to handle 3D object recognition and pose estimation simultaneously based on multiple views of the object. Similar to other multi-view methods, our approach uses multiple projections of an object to jointly recognize its class and estimate its orientation (Figure 1). Our approach uses a supplementary model to infer which orientations provide the most informative views of an object. In particular, the model learns to predict the most descriptive views of the object mainly by learning an entropy estimation function based on depth images. This allows the system to detect the most valuable information to perform multi-view classification while removing the redundancy of views that carry less information about the object.

1.1 Object Recognition

When we speak of object recognition we refer to the ability to visually classify an object as an instance of a determined kind, or 'class', that identifies different

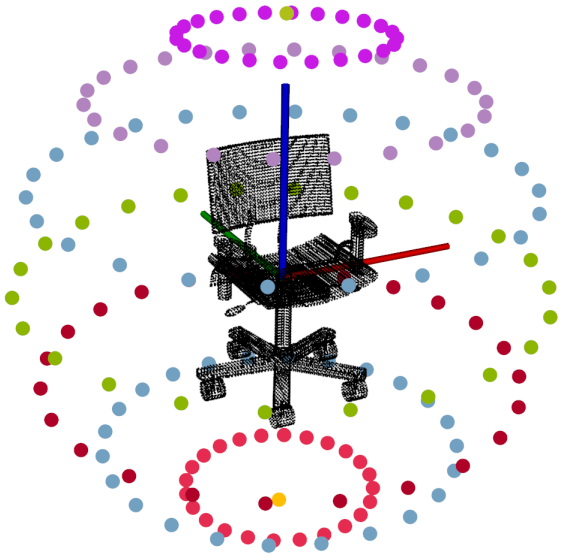


Figure 1: Sphere distribution of camera viewpoint around a 3D model.

objects having specific features in common. For example; we commonly use the word 'table' to define an object having four legs supporting a rectangular flat top, however, some tables have a different number of legs or a different shaped top while still being referred as tables, and other objects with four legs and a flat top might be classified as different objects, such as stools or a very minimalist night stand. Therefore, detecting the features to determine the membership of an object to a class is non-trivial, hence, deciding on the most informative way to digitally represent an object and retrieve these features is up for discussion.

In the field of machine learning the type of data already partially defines the methods to use to build the model. The two main domains of inference are either classification or regression, when respectively the data comes in the discrete or in the continuous domain. Trying to detect the breed of a dog would be a classification task, since the number of breed is limited to a list of possibilities, while predicting the temperature of tomorrow would be a regression task since the temperature can vary in a pseudo-continuous domain (the values are still limited by the precision of the measures and the digitalization of the numerical value). The case of visually recognizing an object falls in the former category, as the target of the prediction is a discrete label among a limited collection that we assign to the object.

1.2 Pose Estimation

Within an unchanging environment, such as an automated chain of production, a classifier would only be expected of making a prediction on the object's class, for example detecting anomalies in the product, while the object positioning would be irrelevant due to the mechanical repetitiveness on the processes. However, a more complex robot would require the perception tools necessary to recognize the object and its pose, to then carry out manipulation tasks. Let's think about a household helper robot: to implement the capability to put back standing a spilled glass it would need the ability to recognize the object and to understand it was spilled. Pose estimation is the ability to identify the position and the rotation of a stationary target, therefore it is a critical task for modern applications of robotics. The pose is indicated by coordinates in 6 dimensions: x, y and z define the position of the object in space while roll, pitch and yaw define its rotation.

1.3 Object Data Representations

The input data domain depends on the context of the task: classification is performed on representations of features of an object, which, as mentioned before, can have multiple formats. We are discussing the most common ones.

1.3.1 Images

The most common type of representation of an object is an image, as in the projection of the object's appearance on a plane, 'rasterized' to a matrix of pixels (or many in case of different color channels). In this format, each pixel represents the value in color intensity of a specific area, and depending on the point of view of the source taking the picture, defined as 'camera' or 'eye', and the environment where the object is located, these values may vary substantially. Images depend on the source in terms of resolution and color quality, and then may be compressed by the file type encoding. We are not going to discuss the impact of these factors on the classification, however, we must keep in mind that to ensure the quality of the results the images need to be processed in the same way, otherwise the classifier might detect variations in the images due to the different sources and learn them as features, while they are not relevant for the decision. It is also worth to mention that higher resolution images potentially give more information about the object, while

increasing the number of parameters to train, hence increasing the computational cost and the likeliness of learning unrelated features.

1.3.2 Feature-based Representations

Another way of representing an object is to summarize its appearance in a number of key features. There are several methods to extract features from data through processing: in the image domain, for example, it is possible to detect edges with the Canny algorithm [3] or to use a feature descriptor such as SIFT [17] or SURF [2]. Representing an object via extracted features instead of the original data may prove to be more efficient, since the amount of data is usually lower and it provides the classifier only with the relevant information. On the other hand, these methods need to be calibrated manually and they might not generalize well on a dataset where the data varies widely.

1.3.3 Polygon Meshes

The most common representation of objects in three dimensions is a polygon mesh. A mesh is a collection of vertices, edges and faces that represent an object as a polyhedral approximation of its shape. The faces are usually simple convex polygons, such as triangles or quadrilaterals, combined to represent complex surfaces. Meshes can be a very precise method to represent an object in 3D, in fact they are the most popular data structure in computer graphics and 3D design however, in comparison to other structures, they are more complex and require more data. Hence, in machine learning it is a common practice to process this type of data to more efficient representations.

1.3.4 Point Clouds

When it comes to representations in space the information about the objects becomes more complex but also more complete. Since in three dimensions the data becomes observable from multiple points-of-view, all of the visible features of the objects become potentially available within a single data structure. However, 3D data is exponentially more complex than - for example - images, hence its visualization and manipulation are computationally heavy tasks. Point clouds are collections of points, in the 3D space, sampled from the observed objects. Only requiring the

coordinates of a number of positions, point clouds are one of the most efficient data structures for 3D representation, and depending on the complexity of the surface they can retain most of the object information even at lower sample rates. Point clouds are susceptible to noise, so if the surface of the objects presents many irregularities, a low sample rate might return some representations that do not resemble the original object or makes it difficult to classify.

1.3.5 Voxel Grids

Voxel grids are volumetric occupancy grids, they consist of a multi-dimensional matrix of 3D pixels, the 'voxels', where each position represents a unit cube in space. In the most simple implementation they only consist of a three dimensional binary matrix defining the shape of the object. An algorithm is needed to perform the 'voxelization' of a point cloud or a polygon mesh, hence the quality of a voxel grid is determined by the previous data and mostly by the resolution given by the size of the unit cube. While being less complex structures than polygon meshes, the voxel grids require more computational resources to be visualized and manipulated in comparison to point clouds, although they are less sensitive to noise: the voxelization algorithm can act as a smooth filter for the objects, simplifying irregular surfaces into less complex shapes.

1.4 Classification

There is a wide array of techniques to make statistical modeling on different formats of data. Some of the more complex of these techniques are suitable to perform high-level tasks such as object recognition. Methods for object recognition generally fall into either machine learning or deep learning-based approaches. In machine learning it is necessary to first define features of the objects, extract them and train classifiers such as the K-Nearest Neighbours algorithm (k-NN) or the Support Vector Machine (SVM) to make the classification on the resulting data. Deep learning based techniques usually perform end-to-end object recognition on data without the need of explicitly defining specific features: a neural network is capable of extrapolating and learning features from data within itself in a black-box fashion, with no direct manipulation of the parameters in its hidden layers. While it is not the objective of this thesis to give a comprehensive study on these practices, we are discussing the main concepts behind the methods applicable to the visual classification

of objects.

1.4.1 Support Vector Machines

As we mentioned, it is common practice in machine learning to synthesize complex data, such as images and 3D shapes, into specific features (for example, sharp angles, roundness, color, etc.). The SVM is a supervised learning algorithm that analyzes data for classification: given a set of labeled training data, it produces a model that is capable of assigning labels to previously unseen data from the same domain. In practice it uses known data to subdivide the domain into regions defined by non-linear functions, implicitly mapping the inputs into high-dimensional feature spaces, then it assigns to all of the new data falling within a region its respective label. SVMs proved to be quite effective, even with data where the number of features is greater than the number of samples, however, the main drawback of using an SVM is its nature as a binary classifier. The most common way to perform multi-class classification with SVMs is to reduce the problem to multiple binary classifications, either in a one-versus-all binary labeling for each class or in a one-versus-one classification between each pair of labels. These approaches increase largely the computational cost of the classification.

1.4.2 Artificial Neural Networks

A modern approach to solve classification tasks is using deep learning algorithms. The core concept of these algorithms is to build a Neural Network, a complex network of parameters and functions, capable of learning high-level features from labeled data to then make predictions on unseen new data. The single unit of a neural network is called a ‘neuron’ and it consists of a numerical parameter combined with its activation function. Neurons are grouped in vectors or matrices called ‘hidden layers’, which are stacked on top of each other by linking each neuron with neurons of the adjacent layers. If each neuron is linked to every neuron of the adjacent layers it is called a ‘fully connected’ or ‘dense’ layer. When we refer to the most common acyclic structure of a network we refer to it as a ‘feed-forward’ network. Neural networks are supervised algorithms that ‘learn’ by comparing their output to the correct results given by the training data. They evaluate the error with a ‘loss’ function, whose derivatives indicate how to modify the parameters within the hidden layers via a cascading procedure called ‘backpropagation’.

1.4.3 Convolutional Neural Networks

While neural networks composed of fully connected layers proved to be a very consistent method of supervised learning when it comes to learning specific patterns from data with a strong correlation between neighboring features (eg. images, speech, etc.) some more complex architectures achieve more accurate results. In the last few years, state-of-the-art accuracy scores have been achieved by networks implementing many different types of layers, one of them being the convolutional layer which gave the name to the most popular archetype of networks for object recognition: the convolutional neural network (CNN). A convolutional layer differs from a fully connected layer in the relationship between a layer and its subsequent: every convolutional layer implements a set of filters (or kernels), lower dimensions matrices of parameters which are multiplied by convolution across the width and height of the input volume. The operation of stacking convolution layers to perform recognition on visual data was inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex, hence CNNs generally achieve better generalization on vision problems.

1.5 3D Object Recognition Archetypes

In this section, we discuss the main concepts behind popular methods that have been employed in recent years to achieve state-of-the-art performances for object recognition tasks on three-dimensional data.

1.5.1 The Multi-View paradigm

The availability of three-dimensional data is generally scarcer than two-dimensional data. The reason for this unbalance is due to the difficulty of obtaining sets of digital 3D objects, since they are required to be either inferred from the interpolation of 2D images, retrieved with a depth-camera or manually designed with specific CAD software. On the other hand images datasets are easily produced from data already available on the internet or by taking pictures with common equipment such as cameras or even smartphones. Image data is also easier to visualize and manipulate: about any computing device with a sufficient resolution and color space capability can display a picture and perform basic alterations such as rotations and color shift-

ing, while 3D data is computationally more expensive to visualize and to modify and needs sufficiently powerful resources. Given the difference in availability of image datasets and the demand for applications on image data, the development of object recognition algorithms has resulted in substantially more powerful and efficient classification models for 2D images. Well integrated in deep learning frameworks, there are instances of pre-designed architectures for image recognition such as VGG [24], Mobilenet [8] or ResNet [7] where parameters have been pre-trained on the popular dataset Imagenet [4]. Instead, the 3D object recognition remains a field open to improvements where there are not well defined standard architectures yet. For the aforementioned reasons, a recent approach to 3D object recognition has been to transform the problem into an image recognition task. MV-CNN [25] and OrthographicNet [10] employ this approach, essentially making use of efficient architectures to classify a 3D object from a collection of pictures synthesized from the original data.

1.5.2 The Voxel-Based paradigm

Deep learning frameworks require data to be processed into regular structures as matrices and vectors, and often times the data provided for 3D object recognition is in the form of polygon meshes or point clouds. In both cases the structure of the data is variable and requires to be processed to a standard format before being fed to the learning algorithm. Point clouds are relatively simple to reformat, as they can be subsampled to a constant number, given this number is less than the number of points in the smallest cloud. Polygon meshes are more complex to deal with since removing points would require a complete re-factor of the edges and faces of the polygons. As we mentioned in Section 1.3 both these data structures can be transformed into voxel grids via a 'voxelization' algorithm. The concept of voxelization is quite simple: a limited portion of space is subdivided into a grid, for each cube in the grid the algorithm searches for at least a vertex, edge, face or point located within the boundaries of the cube and marks the 'presence' of the object in that location, therefore producing a 3D mask of the object, with the resolution given by the fineness of the grid. Voxel grids represent a convenient way to represent a 3D object as matrices of constant shapes, compatible with deep learning algorithms. Recent competitive approaches are based on VoxNet, a 3D convolutional neural network that uses these occupancy grids to deal with large amounts of point cloud data [18].

2 Related Work

In this section we discuss the theoretical framework of our project, reporting approaches for object recognition and pose estimation of 3D objects visioned while studying the problem and designing our approach.

2.1 CNNs for 3D Object Recognition

There are substantially two main approaches for CNN-based object recognition: voxel-based and 2D image-based approaches. Qi et al. [21] provides a comprehensive study on voxel-based CNNs and multi-view CNNs for three-dimensional object classification, stating that empirical results from these two types of CNNs exhibit a large gap, indicating that existing volumetric CNN architectures and approaches are unable to fully exploit the power of 3D representations.

2.1.1 Voxel-based CNNs

Among voxel-based systems, the earliest work would be 3D ShapeNets [29] which developed a Convolutional Deep Belief Network to learn probability distributions of binary occupancy grids. The principle of using Convolutional Neural Networks with voxel grids as inputs is conceptually similar to the feature mapping performed by convolution layers of image recognition CNNs: A 3D CNN can be used to progressively extract higher-level representation of the shapes of the objects as shown in Figure 2.

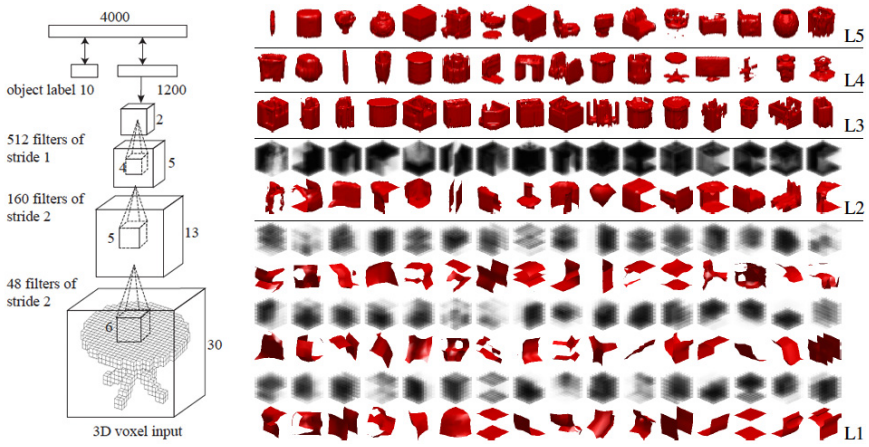


Figure 2: 3D ShapeNets Architecture and feature understanding [29].

The latest researches on similar approaches achieved improvements in performance by enhancing the architecture of the network. Among these, we cite a few notable examples: VoxNet [18] is a CNN architecture designed to tackle object recognition by integrating the voxel representation to deal with large amounts of point cloud data. FPNN [16] employs field probing filters to efficiently extract features from voxel data. 3D-GAN [28] implements Generative Adversarial Networks (GAN) to generate 3D objects from a probabilistic space and obtain a 3D object descriptor from an adversarial discriminator.

2.1.2 Multi-View CNNs

The availability of 3D data usually induces to directly apply recognition algorithms on three-dimensional representation of that data. However, as resulted from the work of MVCNN [25], it is possible to achieve significant improvements in classification accuracy by using collections of rendered views of 3D objects. They obtained view images by retrieving 2D projections of the objects with a set of virtual cameras positioned in a regular setup, as shown in Figure 3.

The authors opted for a fixed number of virtual camera points, positioning of the cameras on a regular structure around the objects. However, they mention that in many cases a single view already achieves satisfying classification accuracy.

They demonstrated that a convolutional neural network, trained on a fixed set of rendered views of a 3D shape, could outperform most architectures trained on three-dimensional structured data.

RotationNet [9] to this date achieved the best accuracy score on the ModelNet

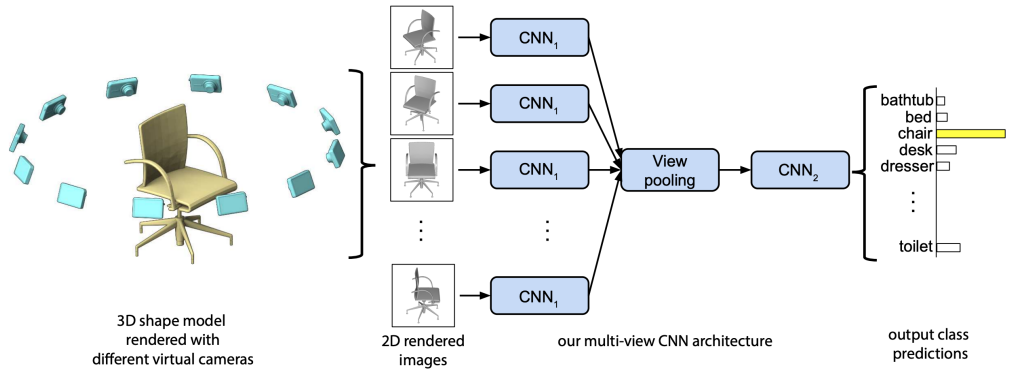


Figure 3: Architecture of Multi-View Convolutional Neural Network (MVCNN). [25].

datasets [29], resulting in the current best approach for 3D object recognition. They proposed a CNN-based model which takes multi-view images as input and jointly estimates its pose and object category. Viewpoint labels are learned in an unsupervised manner during the training and the architecture is designed to use only a partial set of multi-view images for inference (Figure 4). Unlike MVCNN, their method is able to classify an object using a partial set of images that may be sequentially observed by a moving camera. The system infers the probability of a retrieved view to match the camera position it has been taken from, subsequently determining the orientation of the object.

2.2 Simultaneous Object Classification and Pose Estimation

Joint learning of object classification and pose estimation has already been unraveled [23, 13, 30, 1], however, very few of them address inter-class features learning for pose alignment. It has been proved beneficial to share appearance information across classes to simultaneously solve for object classification and pose estimation [12]. Elhoseiny et al. [6] studied CNNs for joint object classification and pose estimation based on multi-view images, discussing architectures of the following archetypes: Parallel Model (PM) consisting of two base networks running in parallel; Cross-Product Model (CPM) explores a way to combine categorization and pose estimation by building the last layer capable of capturing both; Late Branching Model (LBM) splits the network into two last layers, each designed to be specific to the two tasks; Early Branching Model (EBM) similar to LBM, however, the branch-

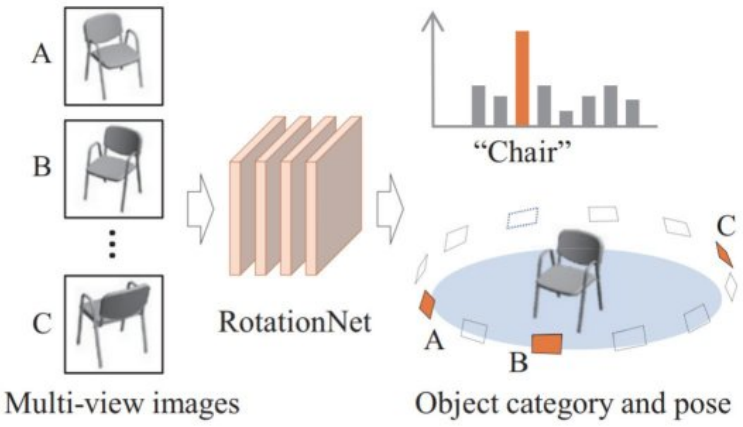


Figure 4: RotationNet Multi-View CNN [9]

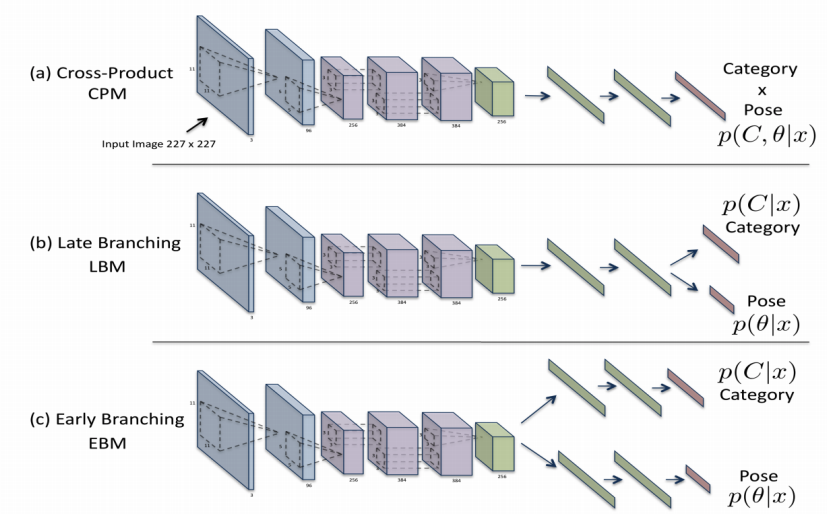


Figure 5: Models showing the joint loss layer in CPM, late branching in LBM and early branching in EBM. Blue layers correspond to layers with convolution, pooling and normalization. Violet colored layers correspond to layers with just convolution. Green layers correspond to fully-connected layers. [6]

ing is moved to an earlier layer in the network. Examples of these archetypes are shown in Figure 5. While their method takes a single image as input for its prediction, later works focused on how to aggregate predictions from multiple images captured from different viewpoints: RotationNet [9] used a model that makes use of images with view-point labels that are shared across classes to achieve state-of-the-art accuracy on both tasks.

2.3 Best-View Selection

The best-view selection corresponds to the automated task of selecting the most representative view of a 3D model. Dutagaci et al. [5] provides a benchmark for the evaluation of best-view algorithms and a survey on popular methods of best-view selection. The algorithms discussed by Dutagaci et al. differ with respect to the descriptor they use to assess the goodness of a view, which are assumed to measure the geometric complexity of the visible surface of an object. The descriptors are listed as follows:

- **View Area**: The area of the surface of the object seen from a specific point of view.

- **Ratio of visible area**: the ratio of the visible surface area to the total surface area of the object [20].
- **Surface area entropy**: in this method, the ratio of the projected area of a triangle to the total projected area of the object is assigned to be the "probability" of the triangle with respect to a particular viewpoint. The entropy over this probability distribution is the surface area entropy-based view descriptor [27].
- **Silhouette length**: Length of the outer contour of the silhouette of the object as seen from a particular viewpoint [20].
- **Silhouette entropy**: Entropy over the curvature distribution of the outer contour of the silhouette [19].
- **Curvature entropy**: Entropy of the curvature distribution over the visible surface of the object. By maximizing this quantity, the view with the most diverse curvature values is selected as the best view. We used the mean curvature at the visible vertices to calculate the curvature entropy [19].
- **Mesh saliency**: Mesh saliency is also based on the local curvature over the surface. The mean curvature at each vertex is weighted by two Gaussian filters one with scale twice the other. The absolute difference between the weighted curvatures at two scales corresponds to the mesh saliency at that scale pair. Then, the total mesh saliency at a vertex is calculated as the sum of mesh saliency values at successive scale pairs. The best view is selected as the one which maximizes the sum of saliency values at the visible vertices [14].

2.4 Research Objectives

The objectives of this research project is to investigate on the following concepts:

- With regards to the 3D Object Classification and Pose Estimation problem, we want to determine the viability and performance of using a CNN-based model to predict the best-views for classification of an object. Such model should take a triangular mesh or a point-cloud representing the object as an input and return a set of camera points.
- We want to validate the hypothesis that the images retrieved from the camera points detected by the model represent sufficient information about the object to allow for satisfactory recognition accuracy.
- We want to develop a system for Object Recognition and Pose Estimation that employs the best-views prediction to select the inputs for a Multi-View Convolutional Neural Network. We will then test its performance and compare it to the current state-of-the-art.

3 Methods

The intention of this project is to answer the points declared in section 2.4. To achieve our goals, we design a Convolutional Neural Network to infer the best-views of a 3D model and uses them to make the classification. We subdivided the problem in two main tasks, the first being the best-view prediction and the second being the multi-view based object recognition.

3.1 Best-Views Prediction Model

We first discuss the theoretical approach to the problem, then we report on the design choices during the realization.

3.1.1 Approach

The first objective is to design a model that predicts which point of views are most informative. At first, we need to define how we measure quantitatively the goodness of a view. We opted for designing our own descriptor: we evaluate the quantity of information for each view by obtaining a grayscale depth image from the same point of view and calculating the entropy of it with the definition from Shannon's information theory:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log P(x_i) \tag{1}$$

Most measures require a less extensive evaluation of the 3D model, while this method expects to have several projected views of the object which are computationally expensive to obtain. However, our intention is to make use of a 3D-CNN to directly infer the best-views from the 3D model, in fact providing an evaluation of the information about its surfaces with no processing required. At first our approach had been of utilizing the network to infer directly a number of viewpoints and coordinates deemed to be the most informative by a threshold on the entropy values. This approach carried a few issues which we report in the next section. We later moved to a more effective method: instead of a multi-label classification problem where we classified each viewpoint as informative or not, we defined the problem as a regression to infer the entropy values of every viewpoint, generating in fact a spherical entropy map of the object. The entropy map $H(\phi, \theta)$ is learned in

the form of a 2D function that maps two spherical coordinates, ϕ and θ necessary to identify the viewpoints on a sphere around the object to the inferred entropy values.

$$H : \phi, \theta \rightarrow h \tag{2}$$

Evaluating the peaks of the entropy map we obtain a set of coordinates of the most informative views.

$$\{(\phi_v, \theta_v)\} = arg_{\phi,\theta}(\frac{d^2}{d\phi d\theta} H = 0) \tag{3}$$

3.1.2 Design Choices

The first step is to generate a dataset taking depth images from a number of views of a collection of objects, in particular, we took images from 60 positions, regularly distributed on a sphere as shown in Figure 6. The virtual cameras are positioned on 12 points on a section ring of the sphere, each one at an angle of 30 degrees from the next one. The sphere is circled by 5 rings, parallels to the horizontal axis of the object, which are looking at the center of the sphere from each at an angle of 30 degrees from the next one, cutting the sphere at 30, 60, 90, 120 and 150 degrees from the vertical axis of the object. The structure of the camera positions ensures we obtain a complete overview of any object while having a limited number of fixed positions.

Once we have the positions for the cameras, we take a grayscale depth image of 224×224 pixels of the object for each of the 60 views. We can then evaluate the

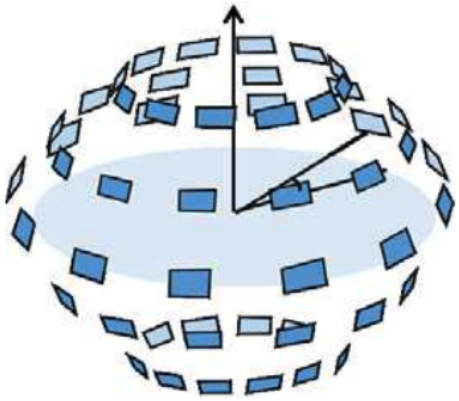


Figure 6: Distribution pattern for the viewpoints.

quantity of information in each picture from calculating their entropy. We build the dataset by matching each 3D model to its 60 ordered entropy values and extracting the best-views.

A variable number of outputs is non-trivial to obtain when designing a CNN architecture: we managed to circumvent this problem by encoding the outputs in n-hot vectors. Each output is defined by a binary vector carrying 1s on the positions of the encoded view coordinates which have entropy values higher than a set threshold. This method tended to provide a list of best-views based on the most common perspectives for high entropy values instead of learning the relationship between silhouettes and entropy. This issue was due to the unbalance in the training data: the models were presented always aligned to the x-y axes, hence frontal and lateral point of views were often the most informative ones. The distribution would be at a 100:1 ratio for many labels, reaching 1000:1 for the most represented label over the least represented. This issue was partially solved by adding a randomized rotation to the 3D models and by operating some balancing techniques of subsampling and oversampling over the dataset.

Anyway, at a later moment, we opted for a more simple and efficient solution: we built the dataset by matching each 3D object to its entropy values and trained the 3D-CNN to infer by regression the values from any 3D model. With this solution, we observed the network generalizes better on new data and it allows for a more precise evaluation of the best-views (Figure 7).

3.1.3 Data Processing

To build the dataset for our entropy model we started from the 3D polygon meshes provided by the Princeton ModelNet10 dataset [29]. ModelNet is a dataset for computer vision and robotics of 3D CAD models. The version we employed is ModelNet10 which consists of a collection of meshes from 10 popular object categories. The CAD models are in Object File Format (OFF) and represent triangle meshes. As mentioned in Section 1.3, 3D object data is among the most complex and computationally expensive type of data to manage in object recognition. Polygon meshes are not an efficient way of representing 3D data for deep leaning applications, hence, while we captured the depth images from the original data, when used as input for the CNN we applied a series of transformations to obtain their occupancy grids. To achieve a consistent input we scaled each model to fit in a unit cube centered in the origin, we then subdivided the unit cube in a $50 \times 50 \times 50$

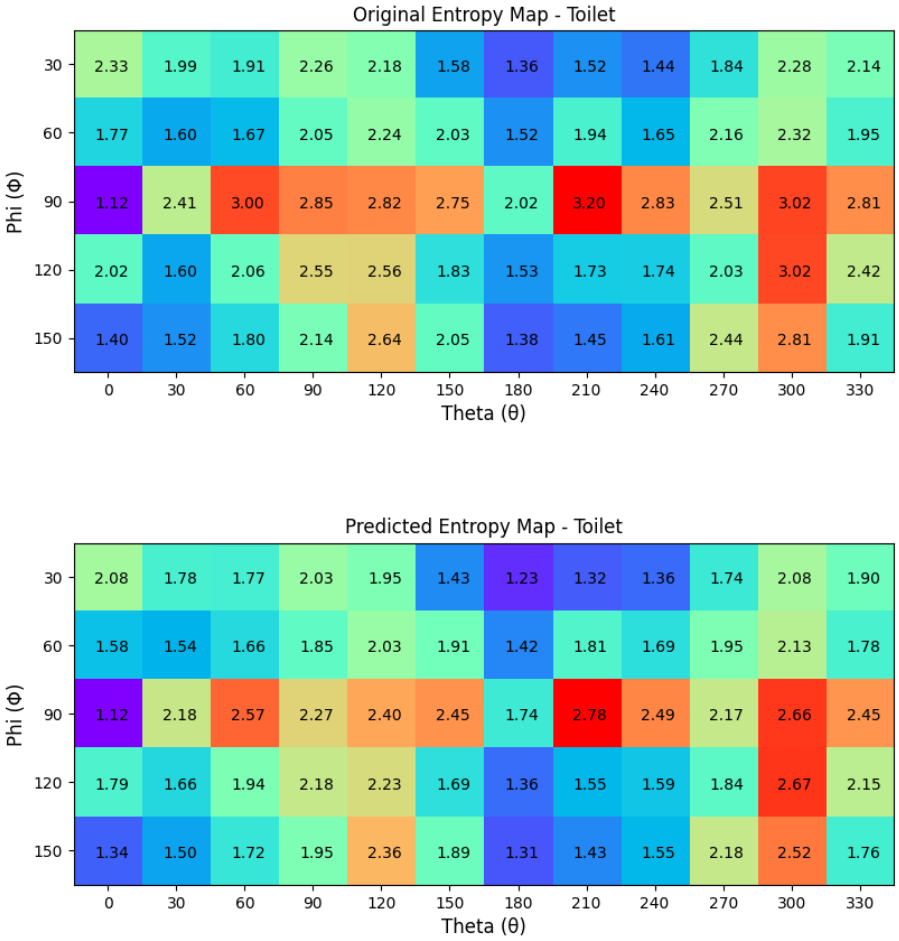


Figure 7: Comparison between predicted and ground truth for entropy evaluation on an object of class Toilet.

matrix of cubes and run an algorithm of voxelization which marks values at 1 in the position corresponding the coordinates in the voxel grid that are occupied by at least a component of the model, while setting the rest to zeros. The binary matrix obtained represent the 3D model silhouette at a resolution of $50 \times 50 \times 50$ voxels. We experimented with different grid sizes, while an higher number would have increased the resolution of the object, it would have increased exponentially the size of the data. We settled for a grid size of 50^3 which offered an acceptable trade off between size and resolution. The smoothing effect due to the little details in the object being cut of by the voxelization resolution happened to have a regularization effect in the learning, since the convolution layers of the network would not try to learn such details as high-level features. To supply to the closeness of the models

to the sides of the cube due to the scaling and bounding process, we added a zero-padding of 3 voxels per side resulting in the occupancy grids to be of $56 \times 56 \times 56$ voxels.

3.1.4 Development and 3D-CNN Design

As the framework of choice we decided to use the Keras API with Tensorflow 2.3.1 backend for Python. The manipulation of the 3D data was performed with the support of the Open3D library [31]. All of the computation heavy tasks were performed on the HPC Peregrine of the University of Groningen. Due to the headless nature of the cluster and the incompatibility for a direct installation of the Open3D library, the employment of the library was enabled by the use of a container for an Ubuntu 18.04 image with Singularity 3.6.3.

The dataset was generated by a script that stored 60 views per object of the ModelNet10 dataset, it calculated the entropy values via the Scikit-Learn method for Shannon's entropy and stored them in a local Pandas DataFrame. The best-views were selected from the obtained entropy maps with a peak detection algorithm for images from the Scikit-Image library. The algorithm would find the peak entropy values in the matrix. An example of the extraction of the views from an entropy map is shown in Figures 8 and 9.

The design of our CNN for estimating the entropy maps employs a splitting branches approach. We decided to employ two convolutional branches with kernels of different sizes separating the flow of the graph from the input layers. Supposedly the different kernel size help the network identifying high level features of different scales. The output of the convolutional branches both receive average pooling, batch normalization ad dropout before being transformed into flat vectors. The outputs of the branches are then concatenated in a single vector and sent as input to a fully connected hidden layer. After a new batch normalization and a higher factor dropout the last layer is a fully connected output layer with linear activation that outputs 60 entropy values (Figure 10).

The optimal number of filters for the convolutional layers and the number of units in the hidden layer was estimated empirically with a hyper-parameters tuner provided with the library Keras-Tuner for Python. The library provided the search algorithm Hyperband [15], a bandit-based approach to hyper-parameters optimization that speeds up random search through adaptive resource allocation and early-stopping. It evaluates architectures by training a set of configurations for a limited

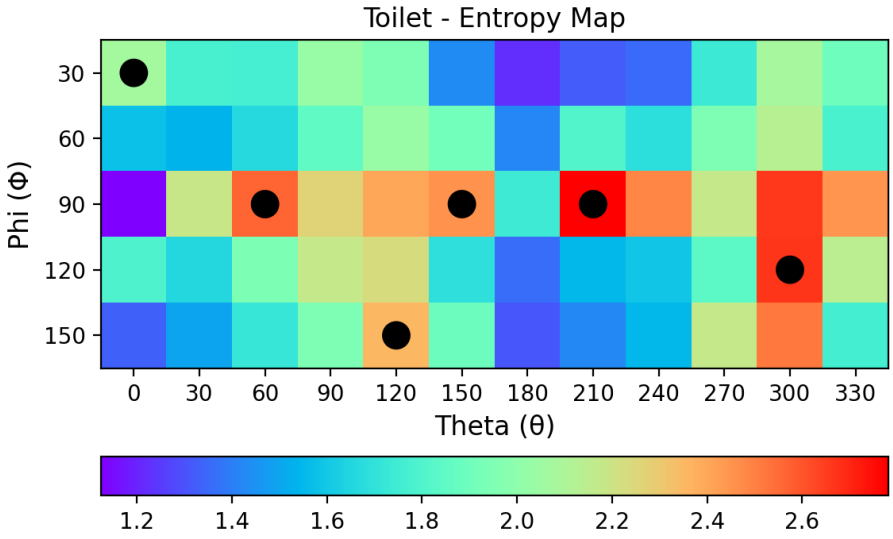


Figure 8: Entropy map calculated from the model of a Toilet object from the ModelNet10 dataset. The color coding shows the highest entropy values in orange-red. The black dots represent the best-views selected by the peak detection algorithm.

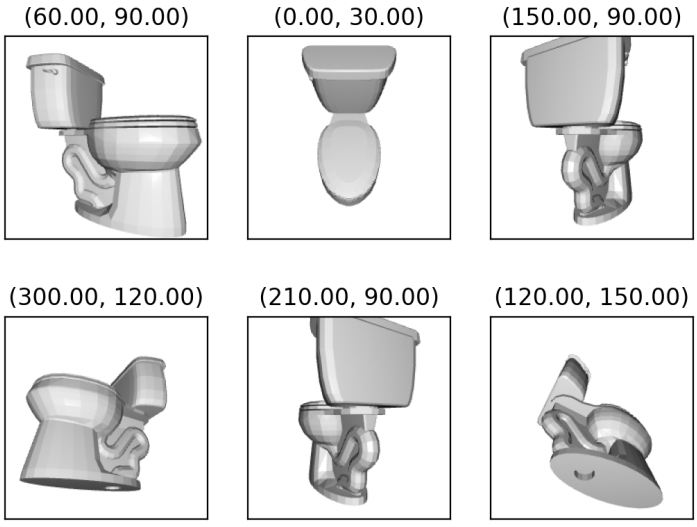


Figure 9: The best-views selected by the peak detection algorithm from the map in Figure 8.

number of epochs and carrying the evaluation only for the most promising half until it reaches the best set of parameters.

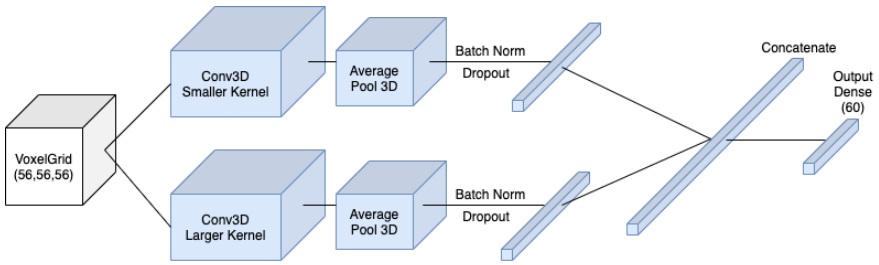


Figure 10: Design for the 3D-CNN employed for the entropy estimation.

3.2 Multi-View Classifier

In this section, we discuss the design and realization of the second module of the project, which consists of a Multi-View based Convolutional Neural Network for object classification and estimation of poses.

3.2.1 Architecture Design

Our approach is to train a Convolutional Neural Network to learn shared high-level features for the joint classification of class and viewpoints. Following the notation from Elhoseiny et al. [6], our design falls in the category of Late Branching Models (LBM). The network we designed uses a core module that employs an instance of a popular architecture for image recognition, splitting the last layer into fully connected layers with softmax activation of size 10 for the object classification and 60 for the pose estimation. Our model employs MobileNetV2 [22] as its core architecture, pre-trained with weights from Imagenet [4]. However we also experimented with other well-known architecture such as VGG-16 [24] and EfficientNet [26] achieving similar results. As regularization techniques, in addition to the structural methods already employed in the configurations of the core architectures (e.g. dropout layers, etc.), we used a dynamic learning rate. This allows the learning process to switch to a progressively finer tuning in the later stages of the training. The network was trained as a single-view classifier on the whole dataset of views, it takes a single-view as an input and it predicts the class of the originating object and the estimated viewpoint. The multi-view consists of the aggregation of M single-view classifiers where M is the number of views provided for the prediction. This method allows the network to accept a variable number of view images, to then return as outputs the classes represented by the majority votes. While the object labels are quite straightforward to aggregate, different views result in differ-

ent viewpoints. The predicted viewpoints are matched to the angle the image views they were taken, the offset between these two values is the predicted rotation of the object from a standing front-facing position. With this values we can evaluate a majority vote for the pose estimation with precision up to half the distance between each of the 60 originating viewpoints in the dataset (15 degrees on rotation around the z-axis and 15 degrees on rotation around the y-axis). This precision can be utterly improved generating a dataset with a more dense configuration of viewpoints and reshaping the network to classify a larger number of positions. This approach would however increase the complexity of the network and the number of parameters, and potentially reduce the accuracy of the classifier.

3.2.2 Data Management

We generate the dataset with the support of the Open3D library by rendering the 3D objects in an empty space and positioning the camera on viewpoints on a sphere around them, as described in previous sections. The view images are taken by a virtual depth camera, which stores the information as grayscale images. The core architectures we employed are pre-trained via transfer learning from Imagenet, which is a 3-channel color image dataset. While our depth image dataset is a single-channel image set, with the simple trick of 'faking' the second and third channel we can still make use of the pre-trained weights and achieve satisfactory results with fine tuning.

The dataset is composed of about 240.000 views, each of the size of 224×224 pixels multiplied by 3 channels during the pre-process, hence the process of loading the full extent as Numpy matrices in a Python script requires an amount of memory that often exceeded our possibilities. We solved this issue with a data generator function, to load batches of images at runtime, which resulted in a more efficient use of the memory space.

4 Results

In this section, we present and discuss the implementation of our system and we report the performance and the results we obtained with tests we performed on of our system.

4.1 Entropy Distribution

The first data we report is a fundamental component in the building of the model we designed to infer the entropy values, which is the dataset we generated to train the model. As mentioned in previous sections we produced a set of 60 depth images from every object in the ModelNet10 dataset. We evaluated the quantity of information of every image with Shannon's Entropy definition (1). In Figure 11 we show the average entropy distribution for the class Bed. Given the shape of the average bed, a low rectangular cuboid, the larger entropy values are found in correspondence to viewpoints closest to the four angles, since such viewpoints frame more faces of the cuboid. It can be seen from the orange-red colours being predominant in the columns relative to the rotation around the z-axis at the values of 30, 120, 210 and 300 degrees. The lowest values instead are found on the row relative to the 90 degrees rotation on the y-axis, meaning the object is being observed frontally. This is the worst angle to observe a cuboid since the upper and lower faces are not visible, hence resulting in the blue-violet row at 90 degrees.

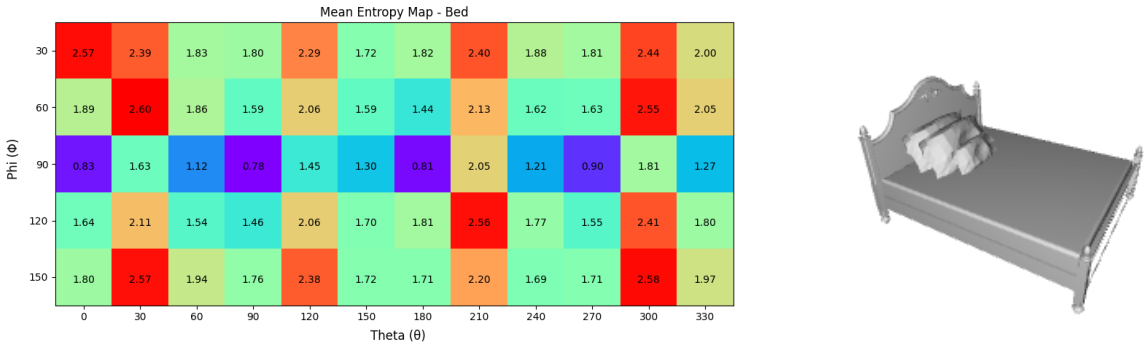


Figure 11: Average distribution of the 60 views entropy for the class Bed. The coordinates (x, y) of the graph indicate the rotation (θ, ϕ). θ represents the yaw angle, while ϕ represents the pitch angle.

4.2 Entropy Model

The Hyperband algorithm resulted in a simpler architecture than we expected. With respect to Figure 12, for input Voxel Grids of size $56 \times 56 \times 56$ we designed two branching 3D convolutional layers with kernels sizes $3 \times 3 \times 3$ and $5 \times 5 \times 5$. The Hyperband algorithm [15] evaluated 8 as the best number of kernels for both the convolutional layers and 512 units for the fully connected layer before the output layer. The architecture of the network is relatively simple, however the dimensionality of the problem makes the number of parameters quite large (179,864,364 trainable parameters). To accelerate and stabilize the learning process we applied batch normalization in-between layers. We also applied regulation techniques such as a progressively smaller learning rate (starting at $5 \cdot 10^{-5}$ until reaching $3 \cdot 10^{-7}$) and dropout factors of 0.25 on the convolution layers output and of 0.5 on the fully connected layer.

As we mentioned in Section 3.1.2 the problem is in form of a multi-output regression with 60 values, hence the output layer uses a linear activation function and the resulting values are to be intended as entropy values and not probabilities. To evaluate the quality of the learning we used two measures: Mean Absolute Error (MAE) and Mean Squared Error (MSE), the former being employed as the loss function. The resulting learning process is shown in Figures 13 and 14. The network seems to learn a function to approximate the entropy of 60 viewpoints. We show in Figure 15 an example comparison between the original entropy map extracted evaluating the view-images of a Chair object and the map built with the prediction from our model. The distribution of the predicted values resembles closely the distribution of the true values. To extract the best-views from the entropy map we use a peak detection algorithm that returns the coordinates of the local maxima in the matrices. In both cases the peak detection algorithm returns the coordinates: $\{$(30,30), (30, 120), (210, 60), (210, 150), (300, 90), (330, 30$\}$ which results in the

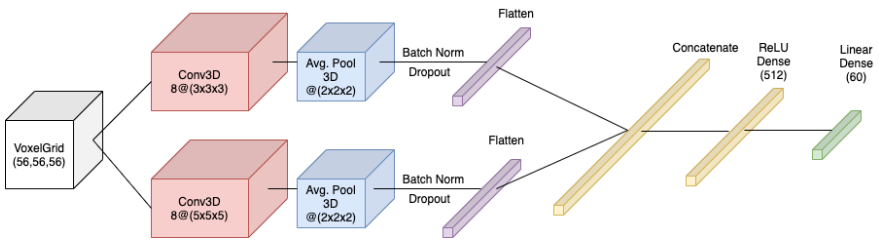


Figure 12: Architecture of the 3D-CNN employed for entropy modelling.

views shown in Figure 16.

When the input is an object of an unseen class the model still uses the high-level features it learned to predict the best views for the new object. As shown in Figure 17 the distribution of the predicted entropy values is very close to the ground truth.

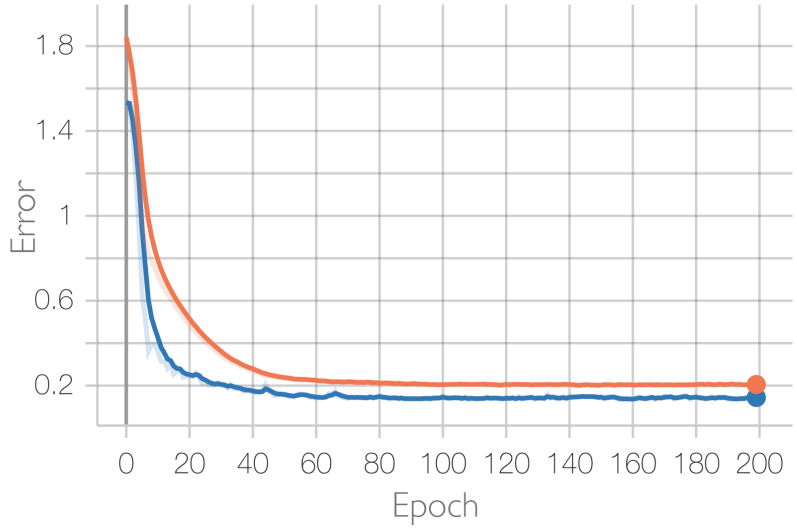


Figure 13: Mean Absolute Error (MAE) on 100 epochs training of the Entropy Model

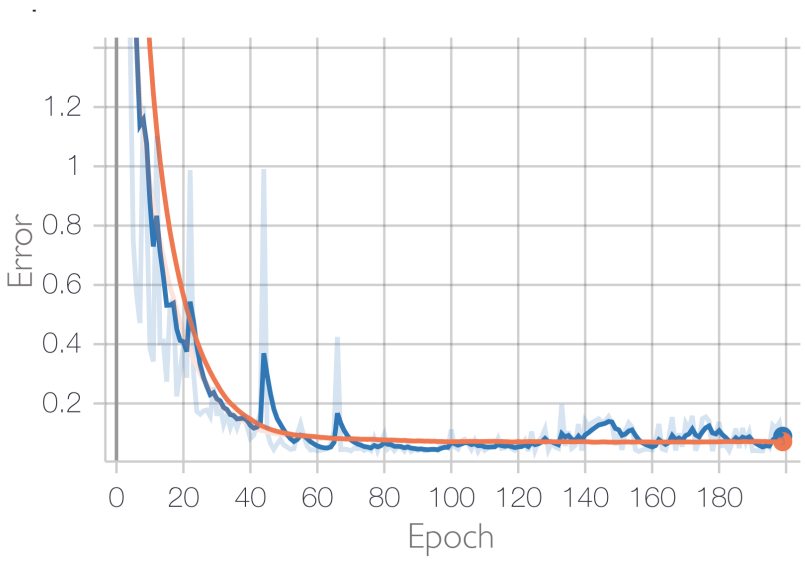


Figure 14: Mean Squared Error (MSE) on 100 epochs training of the Entropy Model

4.3 Multi-View System

The purpose of this project is to explore the possibility of an entropy-based approach to the pre-evaluation of the best-views, in depth image format, for object recognition. To establish this approach as viable we developed a second model to classify the images-views retrieved from the first model output.

4.3.1 Single-View CNN

We experimented with two core architectures, VGG-16 [24] and MobileNetV2 [22], which are reliable CNN architectures for image recognition, pre-trained using the ImageNet dataset [4], a large dataset consisting of 1.4M images and 1000 classes. The architecture is instantiated without the top layers, to adapt to the branching structure for object classification and pose estimation. The branching is performed at the last layer, following the Late Branching Model (LBM) example

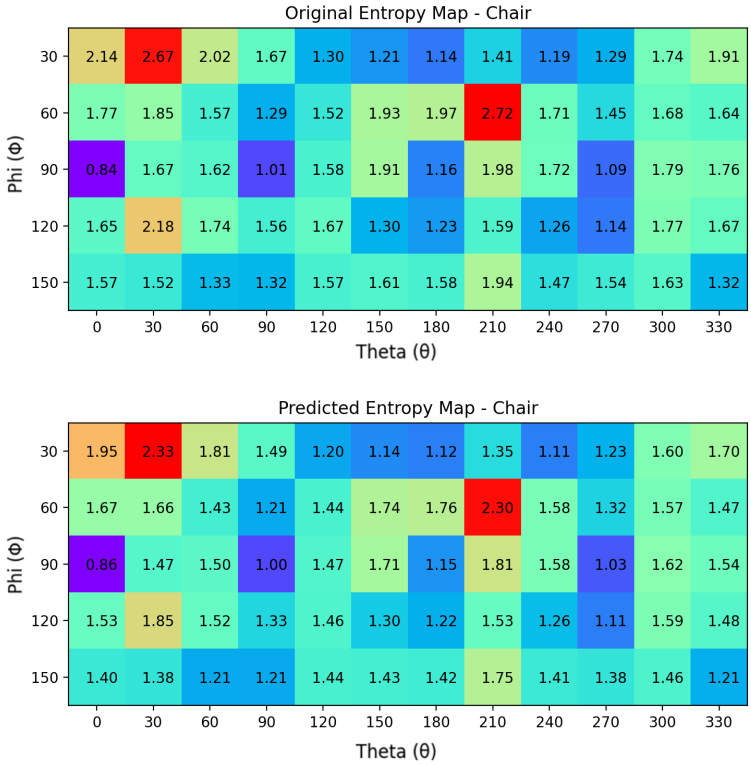


Figure 15: Comparison of the entropy map extracted from the depth-views with the entropy map predicted from our model.

by the notation from Elhoseiny et Al. [6]. We experimented with different branching archetypes, however the number of parameters increased dramatically without a corresponding improvement in accuracy. The last layer of the core architecture is split into two fully connected layers of size 10 and 60 which represent the number of possible classes and poses. We used the Adam [11] optimizer with a starting learning rate of 0.0001 which is dynamically reduced on plateauing validation loss. As loss functions we employed categorical cross-entropy for both class ad pose. We trained the architectures on a dataset of 293,940 RGB images of size $224 \times 224 \times 3$, composed of projections from 60 viewpoints of 4,899 3D models from The ModelNet10 dataset. The results and the learning process are shown in Table 1 and in the Appendix in Figures 30-41.

4.3.2 Multi-View CNN

The complete system bases its final classification on majority vote. Each prediction from the instances of the single-view CNN is pooled and contributes to the decision of the system (Figure 18).

We tested the full system with the best performing models with architectures based on VGG and MobileNetV2. The tests were conducted on the unmodified ModelNet10 dataset, transforming the objects in $50 \times 50 \times 50$ voxel grids at execu-

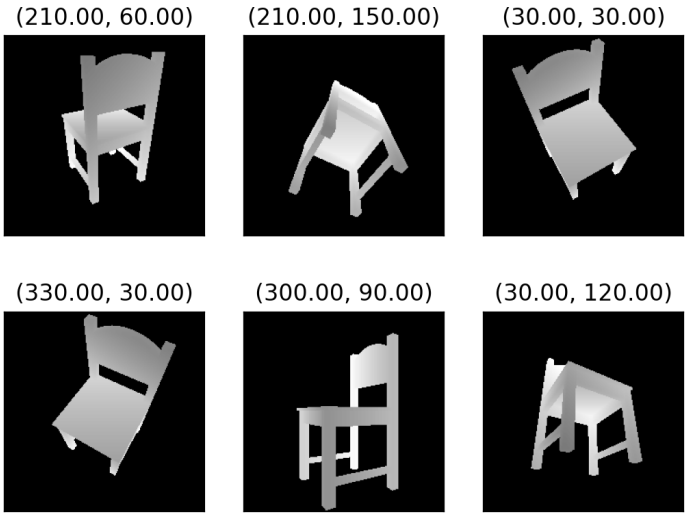


Figure 16: Depth views detected by the peak detection algorithm from the entropy maps in Figure 15.

tion time. For a fair comparison we trained the single-view CNN and then tested the full system using the training and test split of the original dataset as in [29, 9]. The accuracy scores are presented in Table 2. MobileNetv2 proved to perform bet-

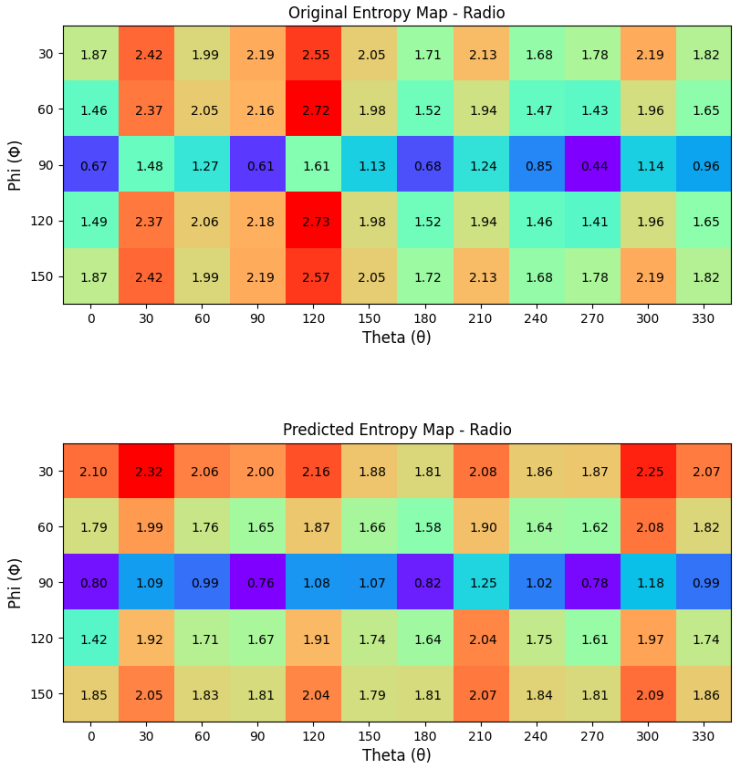


Figure 17: Comparison of ground truth and predicted entropy maps for an object of an unseen class.

Table 1: Accuracy on single-view recognition based on VGG and MobileNetV2 architectures. The dataset has been randomly sampled with the indicated ratio from the original dataset at every epoch, speeding up the training process substantially of a corresponding factor.

Architecture	Dataset Subsamp.	Class Acc.	View Acc.
VGG	1:1	0.8339	0.7890
VGG	1:3	0.8257	0.7972
VGG	1:20	0.7807	0.7239
MobileNetV2	1:1	0.8459	0.8110
MobileNetV2	1:3	0.8312	0.8092
MobileNetV2	1:20	0.7985	0.7473

Table 2: Class and Pose accuracy scores for the Multi-View system based on the VGG and MobileNetV2 architectures. The data used for the evaluation is the proposed test split from the ModelNet10 dataset to allow for a fair comparison with other approaches.

Architecture	Class Accuracy	Pose Accuracy
VGG	0.9020	**0.9394**
MobileNetV2	**0.9130**	0.9372

ter than VGG for the classification of the objects, while the latter showed a slightly improved performance in the estimation of the pose.

To better understand the features learned by the two architectures we can observe the confusion matrices in Figure 19. From these graphics we notice how both architectures achieve more than 90% accuracy over the classes bed, chair, dresser, monitor, sofa and toilet. The differences between learned features are highlighted by the different scores on the bathtub, desk and night-stand.

The VGG architectures mistakes bathtub objects for sofas more times than MobileNetV2. It also misclassifies more frquently a desk for other objects. While the overall accuracy on the desk class is better for MNV2, VGG is more stable when separating the classes table and desk which are arguably the most difficult to distinguish. Another difference lies in the classification of the night-stand class where VGG performs significantly better. Overall both architectures seem to have difficulties in separating objects with very similar shape features, this might be solved with a system that also considers the sizes of the objects.

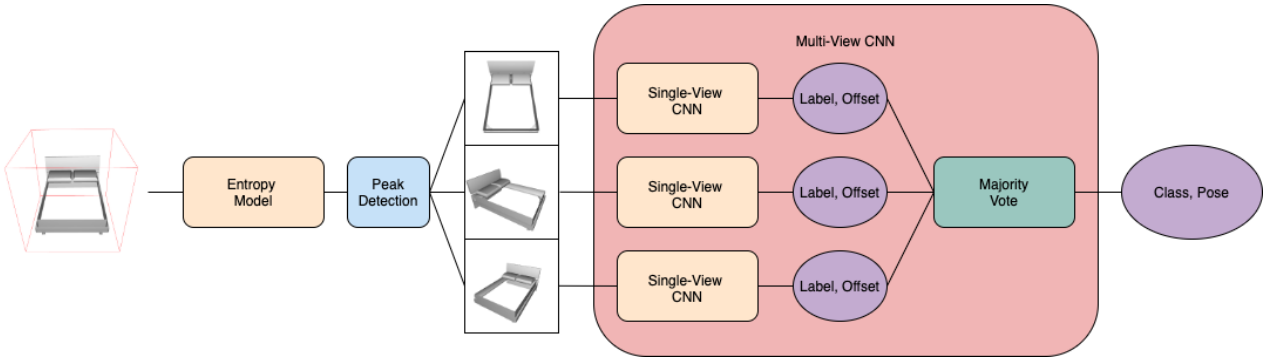


Figure 18: Multi-View Architecture: example with 3 views estimated by the entropy model.

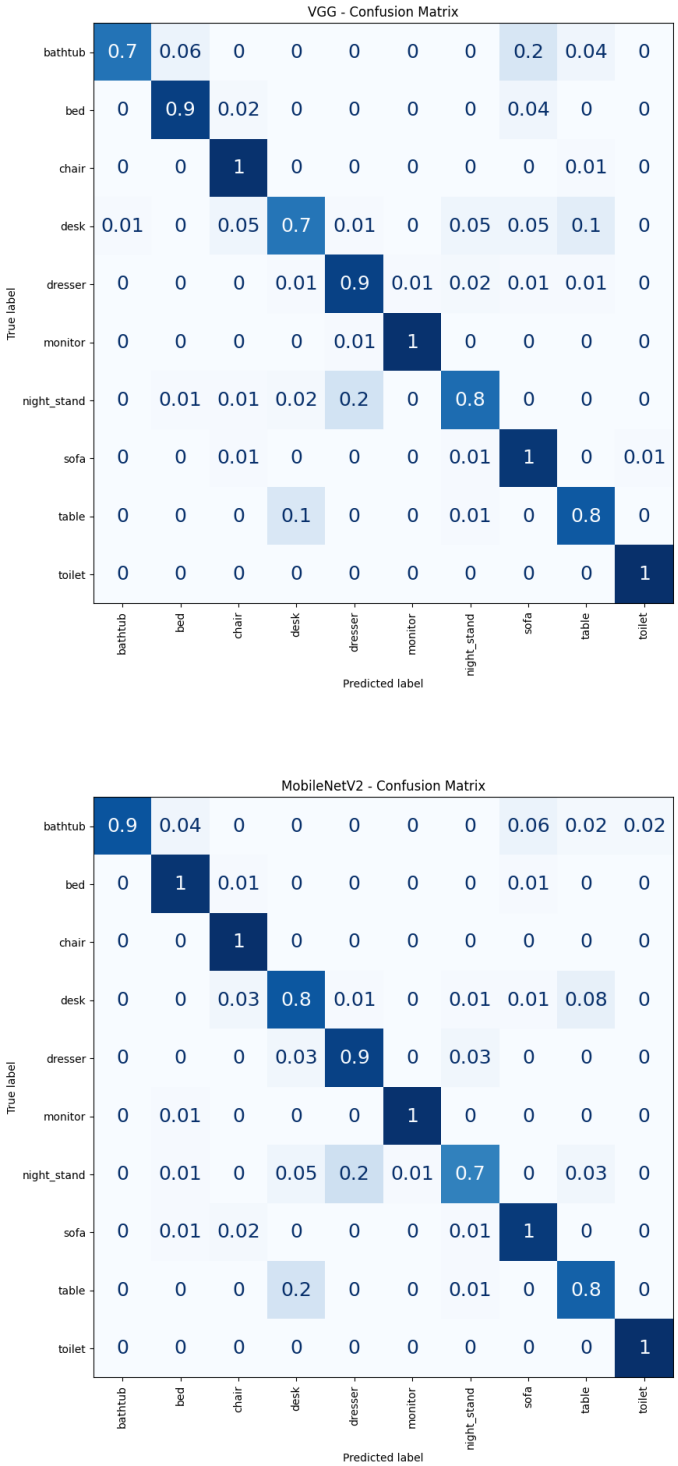


Figure 19: Confusion matrix for classification with VGG architecture.

5 Conclusions

In this thesis, we proposed a system that employs an entropy-based model to retrieve the best-views for a multi-view convolutional neural network to tackle the simultaneous recognition and pose estimation of 3D objects. We demonstrated that a convolutional neural network can learn features to approximate closely the evaluation of the entropy of the depth-images of projections of a 3D object. With such evaluation we can detect the most informative camera poses to observe the object in space. We also demonstrated that the image-views obtained from such positions are descriptive enough to achieve accuracy scores on the ModelNet dataset comparable to other state-of-the-art approaches. Similarly to other methods, our system uses a multi-view approach to classify the object, however, our system uses a model to detect the most informative views to base the multi-view classification.

5.1 Future Work

Our work investigated the viability of an entropy-based estimation a priori of best-views, however the objective of the project was not to compete with state-of-the-art approaches when it comes to accuracy scores. We believe further experiments with network architectures and other entropy-based descriptors might benefit our system's performance. It could also be beneficial to test the architectures on the wider ModelNet40 dataset and other 3D object classification datasets to obtain a clearer comparison with other methods. At last, the discrete nature of the pose estimation of this system leaves the sensitivity of the classifier depending on the density of the dataset, hence it would be possible to achieve more precise estimations by sampling a dataset with a larger number of viewpoints.

Appendix

Table 3: Class accuracy scores on the ModelNet10 dataset for comparison with other approaches.

Algorithm	Class Accuracy
RotationNet	98.46
VoxNet	92
MORE	91.3
3D-GAN	91
DeepPano	85.45
3DShapeNets	83.50
PointNet	77.6

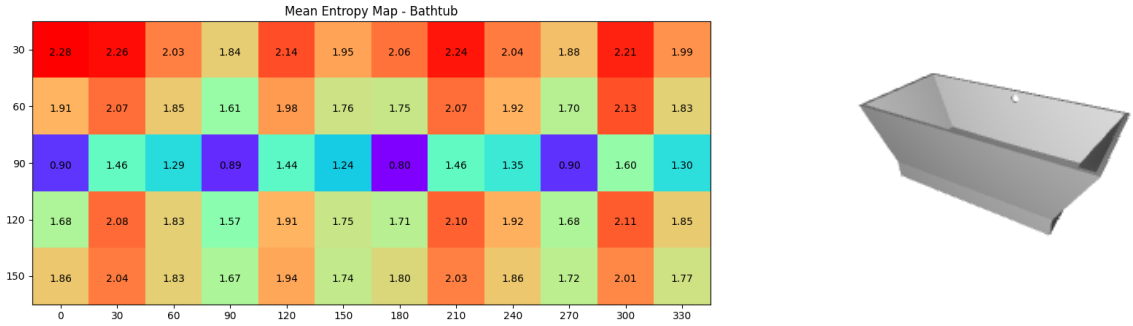


Figure 20: Average distribution of the 60 views entropy for the class Bathtub.

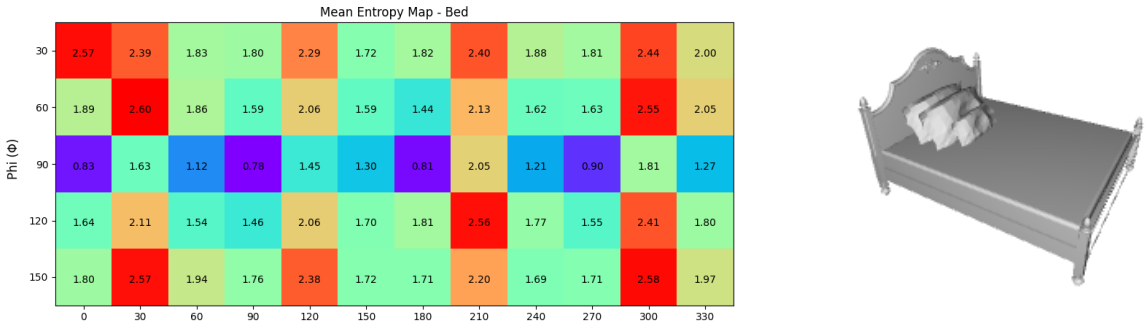


Figure 21: Average distribution of the 60 views entropy for the class Bed.

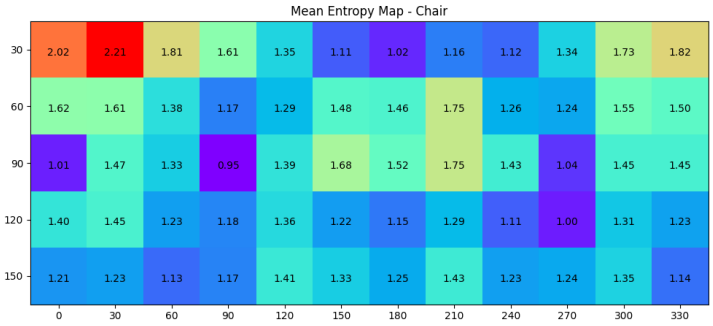


Figure 22: Average distribution of the 60 views entropy for the class Chair.

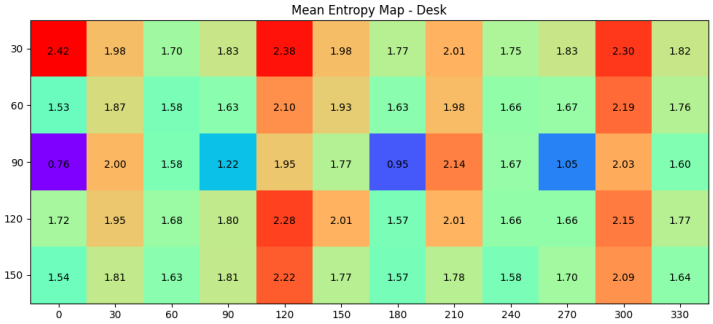


Figure 23: Average distribution of the 60 views entropy for the class Desk.

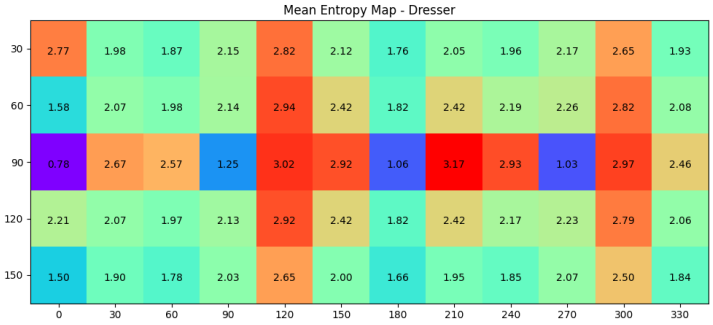


Figure 24: Average distribution of the 60 views entropy for the class Dresser.

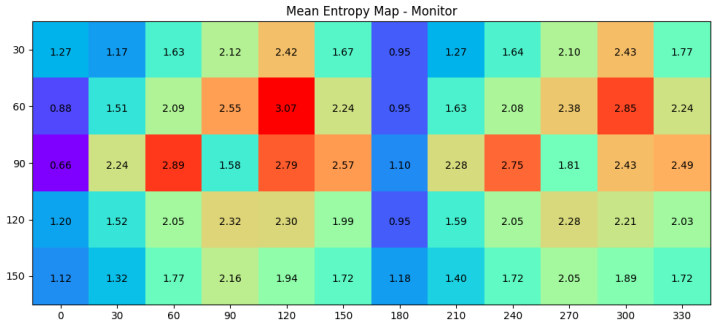


Figure 25: Average distribution of the 60 views entropy for the class Monitor.

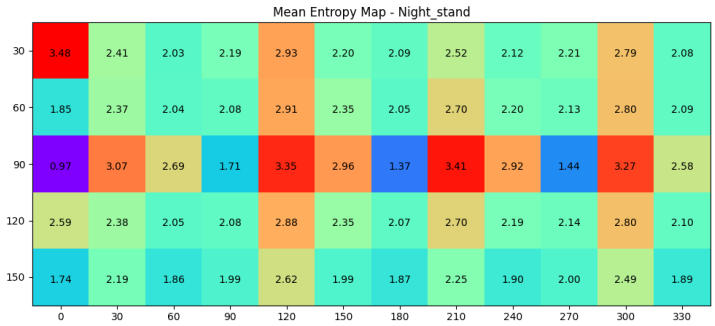
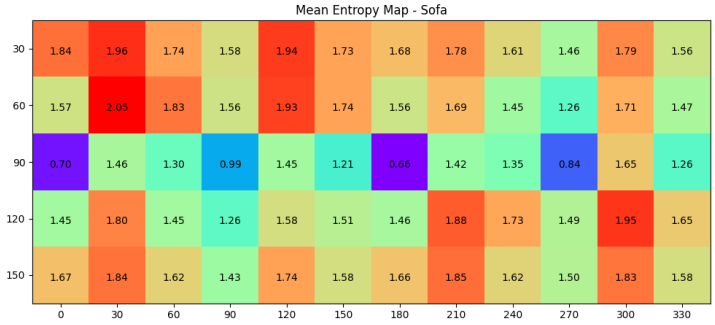


Figure 26: Average distribution of the 60 views entropy for the class Night-Stand.



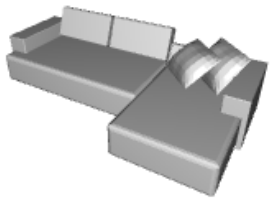


Figure 27: Average distribution of the 60 views entropy for the class Sofa.

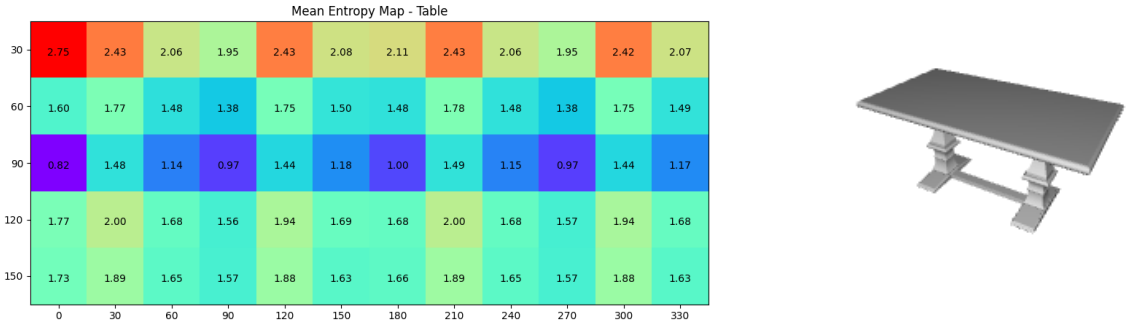


Figure 28: Average distribution of the 60 views entropy for the class Table.

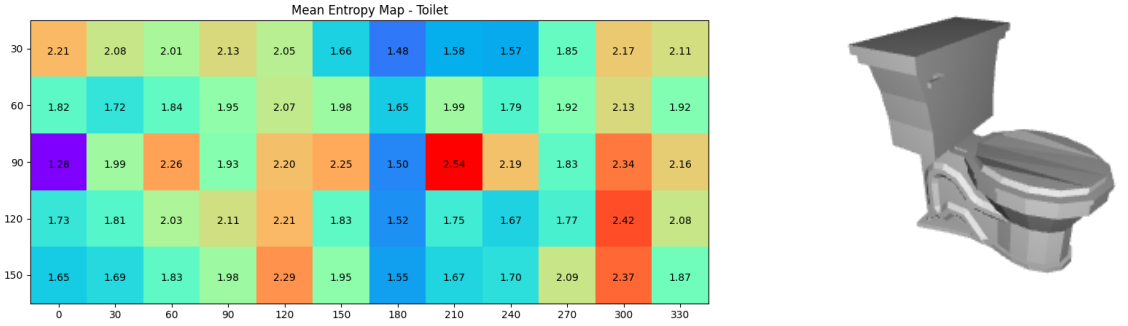


Figure 29: Average distribution of the 60 views entropy for the class Toilet.

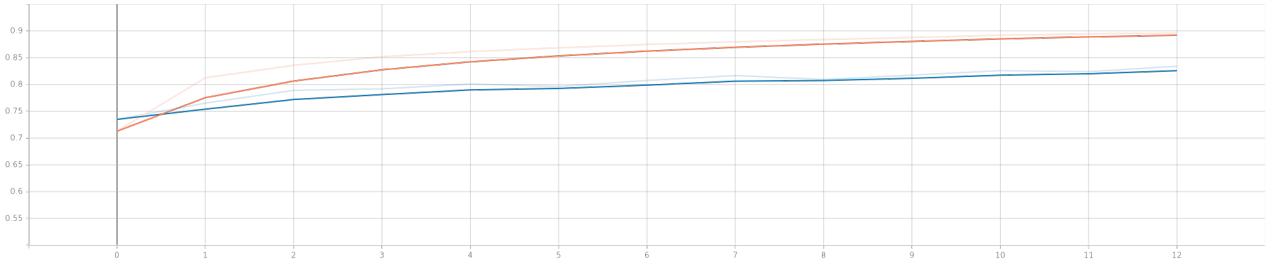


Figure 30: VGG architecture: class accuracy on train (orange) and validation (blue) data per epoch on 1:1 dataset.

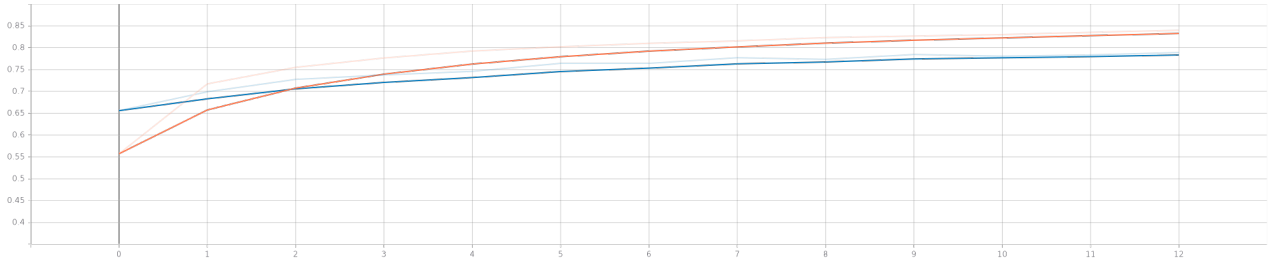


Figure 31: VGG architecture: view accuracy on train (orange) and validation (blue) data per epoch on 1:1 dataset.

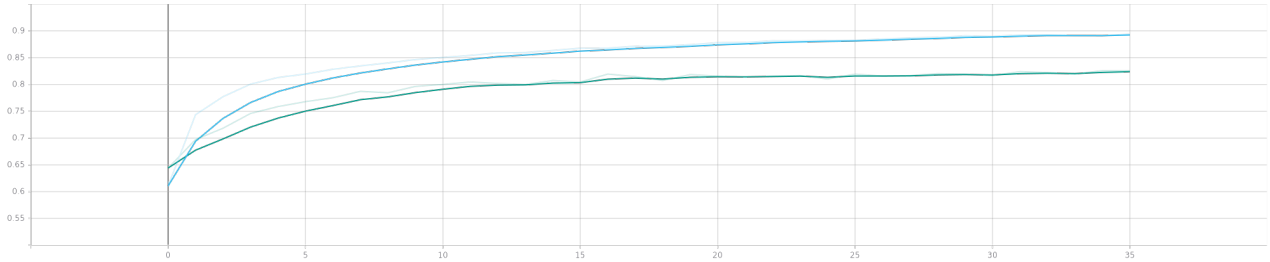


Figure 32: VGG architecture: class accuracy on train (blue) and validation (green) data per epoch on 1:3 dataset.

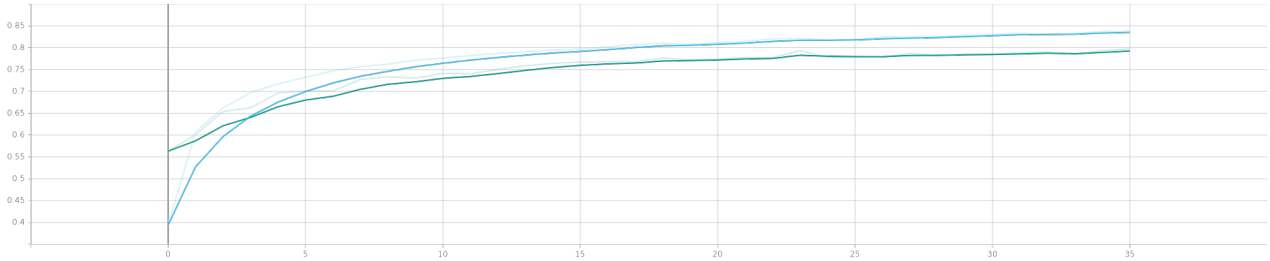


Figure 33: VGG architecture: view accuracy on train (blue) and validation (green) data per epoch on 1:3 dataset.

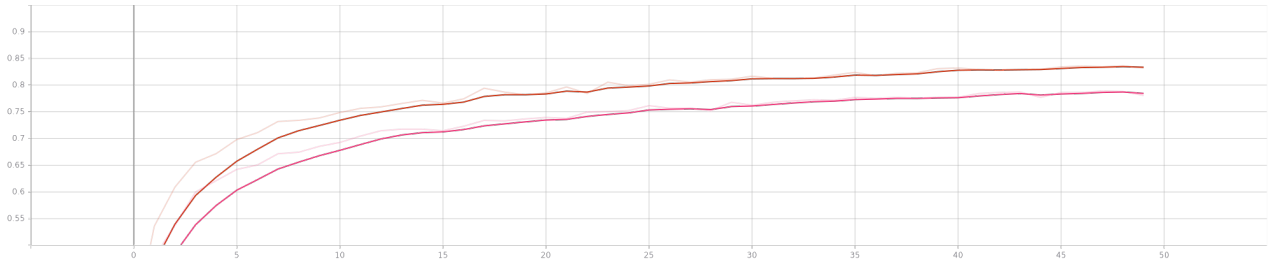


Figure 34: VGG architecture: class accuracy on train (orange) and validation (pink) data per epoch on 1:20 dataset.

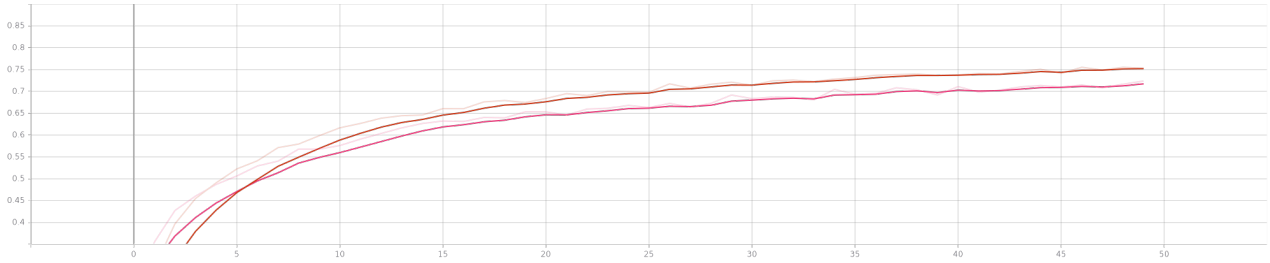


Figure 35: VGG architecture: view accuracy on train (orange) and validation (pink) data per epoch on 1:20 dataset.

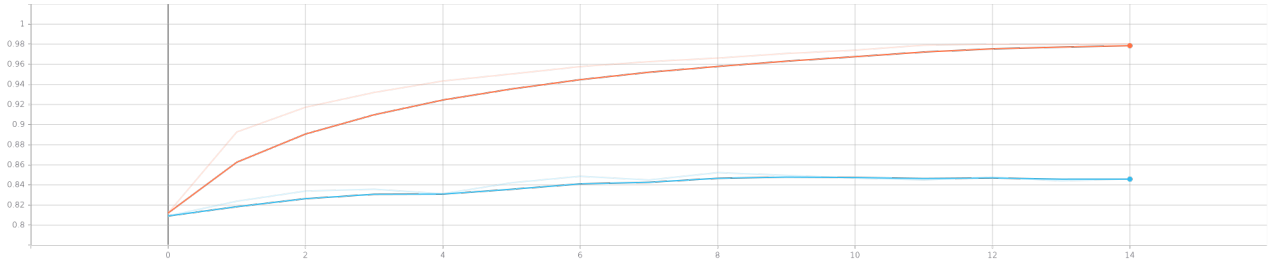


Figure 36: MobileNetV2 architecture: class accuracy on train (orange) and validation (blue) data per epoch on 1:1 dataset.

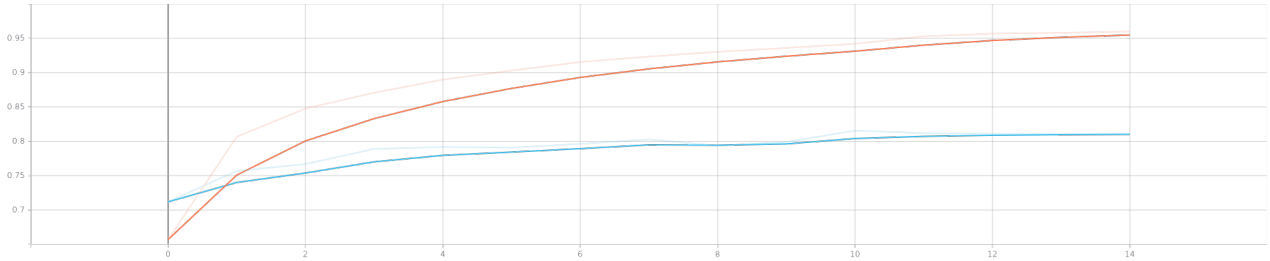


Figure 37: MobileNetV2 architecture: view accuracy on train (orange) and validation (blue) data per epoch on 1:1 dataset.

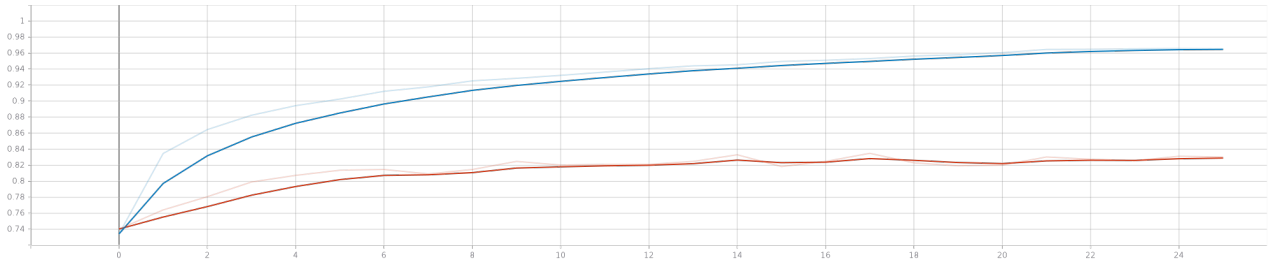


Figure 38: MobileNetV2 architecture: class accuracy on train (blue) and validation (orange) data per epoch on 1:3 dataset.

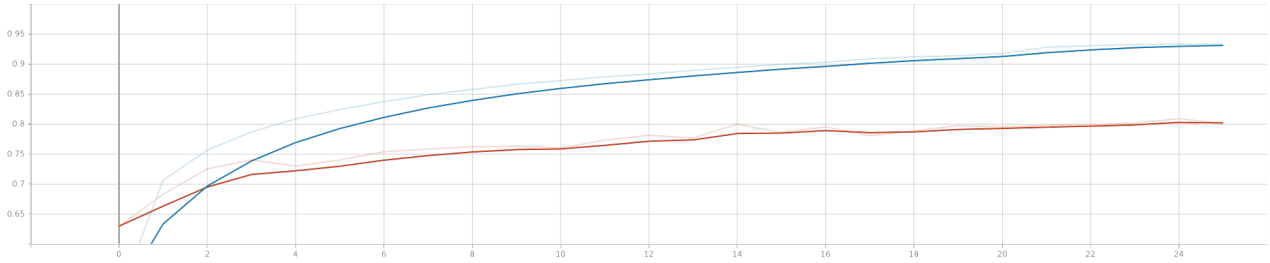


Figure 39: MobileNetV2 architecture: view accuracy on train (blue) and validation (orannge) data per epoch on 1:3 dataset.

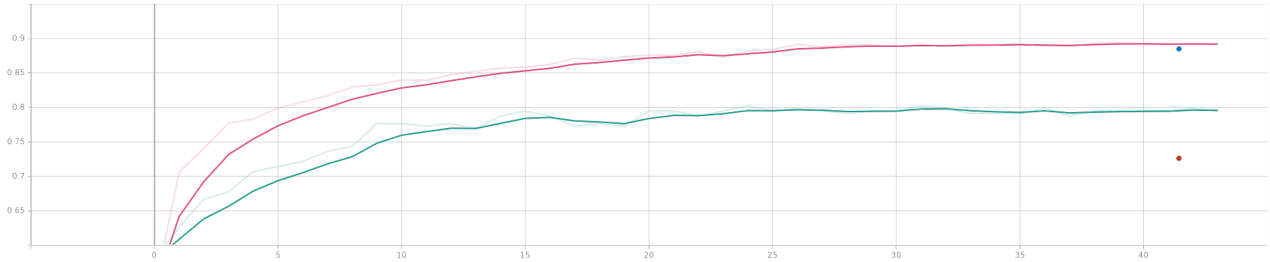


Figure 40: MobileNetV2 architecture: class accuracy on train (pink) and validation (green) data per epoch on 1:20 dataset.

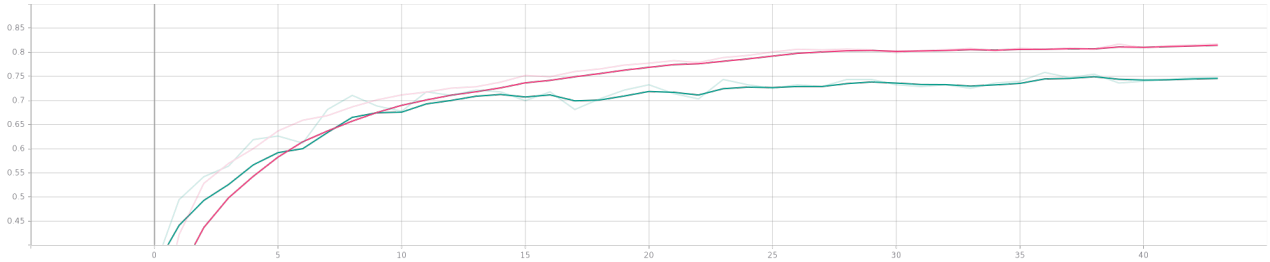


Figure 41: MobileNetV2 architecture: view accuracy on train (pink) and validation (green) data per epoch on 1:20 dataset.

References

[1] A. Bakry and A. Elgammal. Untangling object-view manifold for multiview recognition and pose estimation. In *European conference on computer vision*, pages 434–449. Springer, 2014.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] H. Dutagaci, C. P. Cheung, and A. Godil. A benchmark for best view selection of 3d objects. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 45–50, 2010.

[6] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. Elgammal. A comparative analysis and study of multiview cnn models for joint object categorization and pose estimation. In *International Conference on Machine learning*, pages 888–897. PMLR, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[9] A. Kanezaki, Y. Matsushita, and Y. Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] H. Kasaei. Orthographicnet: A deep learning approach for 3d object recognition in open-ended domains. *arXiv preprint arXiv:1902.03057*, 2019.

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

[12] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal. Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[13] K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

[14] C. H. Lee, A. Varshney, and D. W. Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, pages 659–666. 2005.

[15] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[16] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. *arXiv preprint arXiv:1605.06240*, 2016.

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[18] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE.

[19] D. L. Page, A. F. Koschan, S. R. Sukumar, B. Roui-Abidi, and M. A. Abidi. Shape analysis algorithm based on information theory. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, volume 1, pages I–229. IEEE, 2003.

[20] O. Polonsky, G. Patané, S. Biasotti, C. Gotsman, and M. Spagnuolo. What's in an image? *The Visual Computer*, 21(8):840–847, 2005.

[21] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[23] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[26] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[27] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *VMV*, volume 1, pages 273–280. Citeseer, 2001.

[28] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *arXiv preprint arXiv:1610.07584*, 2016.

[29] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[30] H. Zhang, T. El-Gaaly, A. Elgammal, and Z. Jiang. Joint object and pose recognition using homeomorphic manifold analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 2013.

[31] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.