Comparative Analysis of 3D Object Classification Methods

Alessandro Adami[†], Maddalena Boscaro[†], Giorgio Povegliano[†]

Abstract—**This paper explores the application of neural networks for 3D object recognition using the ModelNet and ModelNet-2D datasets. We introduce two neural network architectures, ORION and Extended ORION, which incorporate orientation information to enhance object classification. Additionally, we compare these architectures with ResNet-34, a widely used 2D image classification model. Our experiments demonstrate the impact of pose prediction on classification accuracy, highlighting the trade-offs between model complexity, computational efficiency, and dataset size.**

Index Terms—**Neural Networks, Convolutional Neural Networks, ORION, ResNet-34, 3D object classification, ModelNet.**

I. Introduction

The field of 3D object recognition has witnessed significant advancements with the advent of neural networks. In this paper, we delve into the utilization of the ModelNet and ModelNet-2D datasets to train and test neural network architectures for 3D object recognition tasks. Our primary focus lies in investigating the influence of orientation information on object classification accuracy. Secondly, we direct our attention to a comparative analysis between a 3D classification model and a 2D model, both aimed at solving the same 3D object classification task.

Section IV introduces the datasets utilized in our experiments, with a particular emphasis on ModelNet and its variations. ModelNet is a comprehensive dataset containing 3D CAD models across various object categories, serving as a benchmark for evaluating algorithms in object classification and shape retrieval within 3D space. Then, we elaborate on the data preprocessing steps undertaken, emphasizing the transformation of input data and the crucial role played by data augmentation. The alignment of ModelNet objects and the introduction of auto-aligned versions are detailed, setting the stage for subsequent experiments.

Section V provides insights into the neural network architectures employed in our study. We introduce ORION, an orientation-boosted voxel net, and its extended version. Additionally, we incorporate ResNet-34, a prominent 2D image classification model, for comparative analysis. The network structures, activation functions, and optimization techniques are discussed, laying the foundation for the subsequent experimental section.

[†]Department of Information Engineering, University of Padova, email:
`alessandro.adami.4@unipd.it`
`maddalena.boscaro.2@unipd.it`
`giorgio.povegliano@unipd.it`

In Section VI, we present the results obtained from our experiments, focusing on the performance of ORION, Extended ORION, and ResNet-34. The evaluation encompasses class accuracy, orientation accuracy, and a comparative analysis of different activation functions and optimizers. These findings contribute to a deeper understanding of the trade-offs between model complexity, dataset size, and computational efficiency. Additionally, we offer a comparative analysis of the various approaches along with the implications of incorporating orientation information into neural network architectures.

Lastly, in Section VII, we draw conclusions based on the observed results and explore potential future developments that may emerge from the results of this study.

II. Related Work

Early methods for 3D object classification heavily relied on handcrafted feature descriptors. However, these approaches may be ineffective as they are not able to express complex patterns. Additionally, they are not scalable and are highly sensitive to variations. To address these issues, there has been a shift toward machine learning-based methods, particularly deep learning. The initial applications of 3D Convolutional Neural Networks (CNNs) for object recognition were primarily introduced in the domain of RGB-D images, and subsequently extended to the field of video analysis. In the current scenario, 3D CNNs have become crucial to the recognition of 3D objects. Going beyond their initial uses, these networks showcase their versatility as they are able to process many different input types, with particular emphasis on representations like point clouds (the one we are considering in this project). In the context of 3D object classification, the studies conducted by Wu et al. [6] and Maturana & Sherer [7] closely align with our work. Both present an approach that leverages advanced techniques for the classification of three-dimensional objects. Wu et al. [6] introduced 3D ShapeNets, utilizing a Deep Belief Network to represent geometric 3D shapes via a probability distribution of binary variables on a 3D voxel grid. It is important to note that the contribution of this work is closely tied to the introduction of ModelNet datasets, which we are exploiting for our project. The approach proposed by Maturana & Sherer [7] is called VoxNet and it's a simple CNN architecture that operates on voxel grids. VoxNet, similar to 3D ShapeNets, incorporates data augmentation during training by rotating objects, aiming to introduce a level of rotation invariance. Both of these methodologies focus solely on predicting the class label, representing a key distinction from the networks under consideration in our work, that aims also at predicting the object orientation. In a major breakthrough, Sedaghat et al. [5] introduced the ORION framework, which is an extension of VoxNet. The main goal is to predict the object's class, but as an extra task, the network also tries to figure out the object's pose. As a result, the loss function used during the neural network's training includes factors related to both the object's class and its pose estimation. As demonstrated, this deliberate adjustment noticeably improves the accuracy of object classification.

An alternative approach is presented by Jinglin Xu et al. [3] where a novel multi-view framework is introduced. This framework employs multiple 2D-CNNs collaboratively to capture discriminative information and relationships. The methodology incorporates multiple 2D views of a 3D object as input, integrating intra-view and inter-view information through view-specific 2D-CNNs and various modules, including outer product, view pair pooling, 1D convolution, and fully connected transformation. This learning structure, widely employed in image classification, benefits from easier optimization tasks due to numerous known results in the literature. Our work takes inspiration from their methodology, however we focus on the unique challenge of recognizing a 3D object from a single 2D image. This reinterpreted approach utilizes a ResNet-34 architecture, emulating the spirit of multi-view learning in a simplified yet effective manner. This study will comprehensively evaluate and compare the overall performances of all presented approaches.

III. Processing Pipeline

In this section, we present an exhaustive examination of our experimental framework, encompassing dataset characteristics, preprocessing methodologies, and the detailed architectures of neural networks employed. The focal point of this exploration is the ModelNet dataset, comprising 3D CAD models subjected to voxel grid transformations which are feeded into our network.

This is followed by a meticulous discussion of ModelNet-2D, which reveals its derivation from 3D models through a rendering process, specifically employing the Phong reflection model. 2D datasets are obtained using three camera setups, each contributing to a different set of rendered views.

Subsequently, a comprehensive overview of the learning framework unfolds, delineating the architecture of the ORION network, its extended counterpart and ResNet-34. Rigorous experimentation is conducted, varying activation functions and optimizers. The empirical results, meticulously documented, shed light on the impact of rotations, activation functions, and optimizer choices within the ORION networks and ResNet-34.

ORION, with its augmented complexity version, is thoroughly analyzed, paving the way for a comparative discussion on computational efficiency and accuracy.

The same is done for ResNet-34: its architectural features are explained, emphasising its use through transfer learning. The results obtained from experiments with ResNet-34 are then examined and compared with those obtained with ORION networks.

IV. Signals and Features

In this section, we describe the dataset used to train and test the neural networks and the procedures performed in order to transform the input data. As well known, the data pre-processing highly influence the results obtained during experiments, in particular data-augmentation helps to prevent overfit.

A. ModelNet for ORION

ModelNet is a large-scale dataset used for 3D object recognition. It contains 3D CAD models from various object categories, like bathtubs, chairs and sofas. Each object category has a collection of 3D models, and the dataset is often used as a benchmark to train and evaluate algorithms for tasks such as object classification and shape retrieval in 3D space. The objects' meshes in this dataset are converted into a voxel grid of size 28x28x28, which is then padded to 32x32x32. Two versions are available, ModelNet10 and ModelNet40, which contains respectively 10 and 40 different classes. Each object is initially presented with a predefined rotation, and additional rotations are manually introduced for the purpose of data augmentation. Both the aligned sets are given and then made freely available by N. Sedaghat et al. [6].

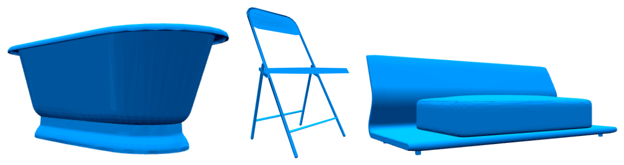
Fig. 1. Examples of CAD models of ModelNet10 and ModelNet40. Respectively of class bathtub, chair and sofa .

Data from ModelNet are pre-processed before feeding them to the network. As first thing the CAD models are converted into voxel grid elements. During this operation also data augmentation have been performed. For each 3D model differently oriented copies are generated following a pose plan. Each element is then labeled with an index associated to the class, and the same for the pose. However, the rotational indexes are not shared between classes, as we do not aim to extract information from the absolute pose of the objects. In essence, there is no unique orientation class for all objects, as our objective is to learn orientation as an auxiliary task to enhance object classification. Therefore we cast the orientation estimation into a classification problem. The number of rotations is fixed at 12 and 24 in two different types of pose planes for the objects. Without any loss of generality, the rotations are only performed around the *z-axis* (azimuthal axis). The 'pose' of an object will always refers to rotations around this axis in this work. However, for some objects, the number of rotations is reduced due to symmetries. This decision is made to prevent the network from distinguishing between an object and its symmetrical, rotated counterpart. In this way we only dedicate

one node where no meaningful labels can be assigned. Being ModelNet10 the smaller dataset, this reduction is carried out by hand, while for ModelNet40 it can be done during auto-alignement, as discussed in [5].
Both the sets, with 10 or 40 classes, were downloaded from [6] with the orientation already done. They come with a division in train and test set, without a validation one. In order to introduce this last set, the total number of samples for each class is manually divided into a 70% for train set, 20% for validation set and 10% for test set. In some cases those values are slightly different in order to maintain a balance as close as possible to the original one between train and test set.

	train elements	test elements	total elements
ModelNet10	4403	496	4899
ModelNet40	11079	1232	12311

TABLE I
DIVISION IN TRAIN (TRAIN AND VALIDATION) AND TEST ELEMENTS FOR THE WHOLE DATASET.

The dataset are transformed using the submitted Matlab code (which is a modified version of the one provided by authors of [5]). So that the objects are transformed from the CAD format to bin files with different poses. Finally the hdf5 files are created to preserve all the useful information needed for processing each sample (bin file with associated class and pose label).
In the case in which the network is trained and tested only taking into account the class of the objects and not the orientation (in order to evaluate the performances with the addition of the pose variable), a pose plan with only one rotation wasn't created. In this case the pose plan of ModelNet10 with 12 rotations is used to feed the network with different perspectives of the objects.

B. *ModelNet-2D for ResNet-34*

Multi-view 2D images can be derived from ModelNet datasets as done by Su et al. in [1]. To generate rendered views of polygon meshes, they used the Phong reflection model [2]. The mesh polygons are rendered under a perspective projection and the pixel color is determined by interpolating the reflected intensity of the polygon vertices. We used the same data provided by the authors of [1], in particular those generating according to the following camera setup:

- 1^{st} setup: 12 rendered views by placing 12 virtual cameras around the mesh every 30 degrees. The cameras are elevated 30 degrees from the ground plane, pointing towards the centroid of the mesh. The centroid is calculated as the weighted average of the mesh face centers, where the weights are the face areas.
- 2^{nd} setup: The renderings are generated by placing 20 virtual cameras at the 20 vertices of a dodecahedron enclosing the shape. All cameras point towards the centroid of the mesh. With this method 20 views are obtained.
- 3^{rd} setup: The last one is generated from the previous case setup by capturing 4 rendered views from each camera, using 0, 90, 180, 270 degrees rotation along the axis passing through the camera and the object centroid, yielding total 80 views.

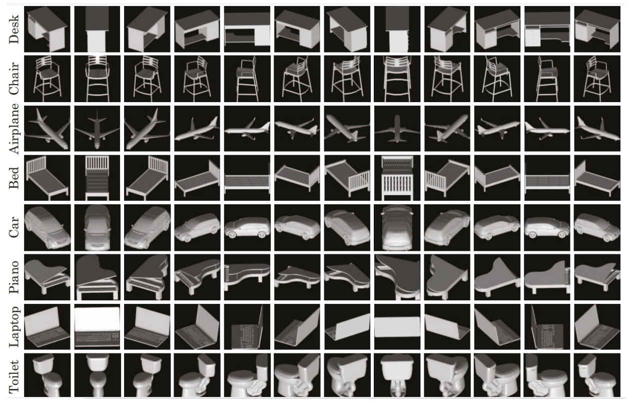

Fig. 2. Example views generated by some categories of 3D objects in ModelNet40.

Given the volume of data, we decided to use only the 2^{nd} setup of the dataset. Since the original dataset was divided only in train and test set, data were subsequently regathered into training, validation, and test sets, with an approximate partition of 70%, 20%, and 10%, respectively.

	train	validation	test	total elements
ModelNet10-2D	64040	19920	14020	97980
ModelNet40-2D	159800	49440	39680	248920

TABLE II
DIVISION IN TRAIN, VALIDATION AND TEST ELEMENTS FOR THE WHOLE DATASET.

We utilised a pre-trained ResNet-34 model designed to take 224x224 pixel-sized images as input, which aligns with the dimensions of the ModelNet-2D data. Consequently, no resizing or padding was necessary; instead, we applied random horizontal flips to the images. This approach demonstrated improved model generalization by further extending the diversity of the dataset.

V. LEARNING FRAMEWORK

In this section the architecture of the used neural networks are presented. The implementation codes for them is provided with this document. Please refer to the accompanying code files for a detailed view of the implementation.

A. *ORION*

ORIentation-boosted vOxel Net (ORION) is a neural network obtained concatenating two 3D-Convolutional layers, a Max. Pooling operation and a Fully Connected layer, as shown in Fig. 3 and presented in paper [5]. The pooling layer follows the last convolutional one, before the fully connected layer. The output, given by two more parallel fully connected layers, is divided into object class and orientation class. In our analysis only the object class is taken into account as result, but orientation class helps to improve the performances of the network as shown as result of this paper. So that, both the outputs are fundamental in our architecture, in particular during the training and model selection phase. In fact the total loss of the net is computed as the weighted sum of the class and orientation loss:

$$\mathcal{L} = (1-\gamma)\mathcal{L}_C + \gamma\mathcal{L}_O \tag{1}$$

where $\mathcal{L}_C$ and $\mathcal{L}_O$ respectively indicates the two class and orientation losses. In our experiments γ is always equal to 0.5 (except when only dealing with classes without objects pose). The chosen loss function for our experiments is the Cross-Entropy one in any case. The obtained total loss $\mathcal{L}$ is taken into account for the model selection during the validation process. The model is updated when a lower value is obtained with respect to the previous ones. Moreover an early stopping criteria is implemented to terminate the training process if the model no longer improves for a certain amount of epochs. On the other hand the accuracy criteria (in particular the class accuracy) is used in the comparison of the performances, between different implementation of the neural networks in this paper.
In order to show the importance of the pose in the classification of the class of an object, the net is also trained and tested without considering the pose part. In this case $\gamma = 0$ and then only $\mathcal{L}_C$ is taken into account computing the total loss.

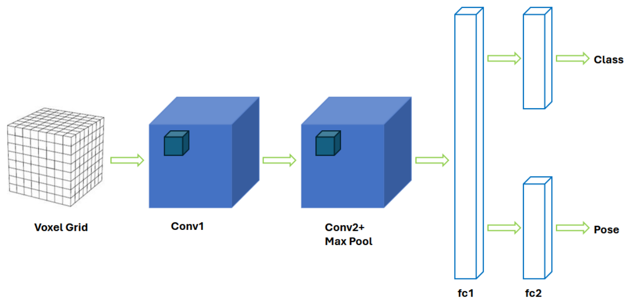

Fig. 3. ORION network architecture. It is possible to notice the presence of two convolutional layers followed by two fully connected ones. In the very last the labels are computed making use of two parallel layers.

Network functioning: A train set, made of 3D samples of different classes as explained in section IV-A, is fed to the first convolutional layer and through network 3 the outputs of the predicted class labels and pose labels are computed. Those predicted labels are computed by the very last layer, which is made by two fully connected ones working in parallel.
During validation phase we feed multiple rotations of the objects to the previously trained network and obtain the final consensus on the class label based on the votes we obtain from each inference pass, as follows:

$$c_{final} = \arg\max_k \sum_r S_k(x_r) \tag{2}$$

where S_k is the score assigned to the k^{th} node of the main output layer by the network and x_r is the validation input with the rotation index r. This last passage is fundamental in order to perform model selection after training.
Different types of activation functions are tested to make comparison between them, in particular: ReLU, LeakyReLU (as did in [5]) and ELU. All the other parameters used in order to implement the network are the same provided by the author of [5] and them are showed in table III.

	Conv1	Conv2	Pool2
# of filters	32	64	-
kernel size	5x5x5	3x3x3	2x2x2
stride	2	1	2
padding	0	0	-
drop out ratio	0.2	0.3	-
batch normalization	✓	✓	-

	fc1	fc2:class	fc2:orientation
# of outputs	128	10 or 40	variable
drop out ratio	0.4	-	-
batch normalization	x	x	x

TABLE III
DETAILS OF THE ORIONS ARCHITECTURE. NOTICE THAT THE NUMBER OF ROTATIONAL OUTPUTS IS VARIABLE FOR DIFFERENT EXPERIMENT, WHILE THE CLASS OUTPUT IS ALWAYS 10 FOR MODELNET10 DATASET WHILE IT IS 40 FOR MODELNET40.

B. Extended ORION

This net shows the main behaviour of the previous network ORION, with the introduction of two more convolutional layers as shown in Fig. 4 and [5], which makes it slightly deeper. The complexity of the structure may lead to overfit with the smallest dataset ModelNet10. As for ORION the experiments are made with different activation functions and all the architectural details are reported in table IV.
This will not trained and tested considering only the class without the orientation.

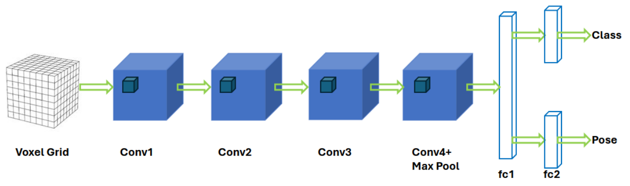

Fig. 4. Extended ORION network architecture. It is possible to notice the fact that the number of convolutional layer is increased, making the structure slightly deeper.

	Conv1	Conv2	Conv3	Conv4	Pool4
# of filters	32	64	128	256	-
kernel size	3x3x3	3x3x3	3x3x3	3x3x3	2x2x2
stride	2	1	1	1	2
padding	0	0	0	0	-
drop out ratio	0.2	0.3	0.4	0.6	-
batch normalization	✓	✓	✓	✓	-

	fc1	fc2:class	fc2:orientation
# of outputs	128	10 or 40	variable
drop out ratio	0.4	-	-
batch normalization	x	x	x

TABLE IV
DETAILS OF THE EXTENDED-ORIONS ARCHITECTURE. NOTICE THAT THE NUMBER OF ROTATIONAL OUTPUTS IS VARIABLE FOR DIFFERENT EXPERIMENT, WHILE THE CLASS OUTPUT IS ALWAYS 10 FOR MODELNET10 DATASET WHILE IT IS 40 FOR MODELNET40.

C. ResNet-34

To assess the performance of the 3D CNN ORION, we opted to compare it with a widely used 2D image classification model: Residual Net (ResNet). ResNet is a type of deep neural network architecture introduced by He et al. in the paper 'Deep Residual Learning for Image Recognition' [4].

The key innovation of ResNet is the use of residual blocks which facilitate the training of extremely deep networks. Residual blocks introduce shortcut connections that add the input of a certain layer directly to its output, bypassing one or more intermediate layers. The primary motivation for introducing shortcut connections is to address the vanishing gradient problem that can occur in very deep networks during backpropagation. Indeed, when the gradients of the loss function become very small and they are backpropagated through many layers, it becomes very difficult for the network to learn effectively. In our tests we focused on Resnet-34 consisting of 34 parameter layers. They also exist in different sizes with an increasing number of layers (18, 34, 50, 101). However, the choice to use the 34-layer model represents a compromise between good generalisation and moderate resource consumption.

ResNet-34 (Fig. 5) consists on one convolution and pooling step (first block in white) followed by 4 layers of similar behavior. Each of the layers follow the same pattern. They perform 3x3 convolution with a fixed feature map dimension [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions. Furthermore, the width and height dimensions remain constant during the entire layer.

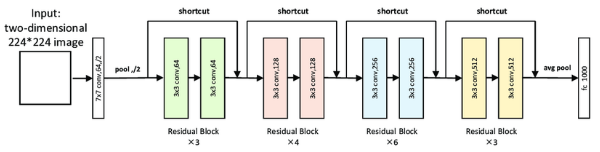

Fig. 5. ResNet34 architecture.

As the classification task for 2D images is a well-studied and widespread problem in the literature, we resorted to the so-called Transfer Learning. This deep learning technique consists on using a model trained on one task as starting point for a model on a second related task. Transfer learning can save time and resources compared to training a model from scratch and can also improve the performance of the new model. In our case we used a pretrained model of ResNet-34. Originally, as we can see from the architecture in Figure 5, the model is trained on a 1000 classes classification task (using ImageNet dataset) and indeed it has a 1000-D fully connected layer.

In our problem instead we have 10 or 40 classes depending on the dataset, so we replaced the old fully connected layer with a new one suitable for ModelNet10 and ModelNet40 respectively. Remember that unlike ORION, the output doesn't necessitate orientation information, it only requires the assignment to its respective class.

We can see the changes made to the ResNet-34 scheme in the last row of Table V. The rest of the architecture instead, is kept the same as the original one.

Layer name	Output Size	
Conv1	112x112	7x7, 64, stride 2
Conv2	56x56	3x3, max pool, stride 2
		$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}$ x 3
Conv3	28x28	$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix}$ x 4
Conv4	14x14	$\begin{bmatrix} 3\times3, & 256 \\ 3\times3, & 256 \end{bmatrix}$ x 6
Conv5	7x7	$\begin{bmatrix} 3\times3, & 512 \\ 3\times3, & 512 \end{bmatrix}$ x 3
	1x1	Avg Pool, 10-D or 40-D FC, Softmax

TABLE V
RESNET-34 ARCHITECTURE. OBSERVE THAT THE DIMENSIONS OF THE FULLY CONNECTED LAYER DEPEND ON THE NUMBER OF CLASSES PRESENT IN THE TWO DATASETS, NAMELY MODELNET10 AND MODELNET40.

VI. RESULTS

In this section, we present the results obtained from testing the previously introduced neural networks, carefully analyzing and commenting each outcome.

All experiments are conducted using two architectures of the ORION network, as presented in V-A and V-B and the ResNet-34, as introduced in V-C. Concerning the optimizers, we implemented both SGD with momentum and Adam for each network. Through experimentation, we discovered that the optimal results could be achieved with ORION (and his extended version) by setting the learning rate to 10^{-3} and momentum to 0.9 for SGD, while for Adam the optimal learning rate was found to be 10^{-5}. Any information about the used optimizer is given in [5].

The equation for the loss function, as presented in (1), serves also for the early stopping criterion (V-A), with the parameter γ selected accordingly in the various experiments. The training process is stopped if there is no improvement in the loss for 20 consecutive epochs, and in any case, the training is stopped after 200 epochs. Additionally, all the ORION networks are initialized with random weights for their parameters.

Each reported result is an average derived from three independent runs. The decision to conduct multiple runs is motivated by the manageable computational burden of the training process, ensuring a more robust evaluation of the neural network's performance.
All experimental results presented in this paper were obtained using the Google Colab environment for code execution and analysis. This environment offers robust resources and proves effective for training neural networks.

A. Only Class evaluation ORION

We begin by evaluating the performance of the network that exclusively trains and predicts the class feature. To achieve this, the information related to orientation is intentionally omitted. In practical terms, in equation (1), the parameter γ

is set to 0, so that the total loss is equal to the class loss $\mathcal{L}_C$. Additionally, the output is narrowed down to only the predicted class.

These experiments concentrate on utilizing the ReLU activation function and, as previously stated, two different types of optimizers for the training process: SGD with momentum and Adam.

Observing the results showed in table VI it can be noticed that a deeper network (i.e. Extended ORION) performs better when it comes to predicting the class label. This finding, while not trivial, can be attributed to the large number of training elements available. It is important to keep this in mind in different scenarios, particularly those with limited training data. However deeper networks may also encounter overfitting issues, emphasizing the importance of considering the dataset characteristics in designing the network architecture.

Comparing the two optimizers we implemented, SGD tends to converge, on average, twice as fast than Adam both in ModelNet10 and ModelNet40 dataset. This highlights the effectiveness of SGD in terms of overall optimization performance, suggesting that, in this specific scenario, a higher learning rate of SGD contributes to a better overall optimization outcome. All the results are reported in table VI: the accuracy refers to the test set and it's computed on the best model obtained during one training session.

Network	Act. Function	Optimizer	Avg. Class accuracy
ORION	ReLU	SDG	**90.94%**
ORION	ReLU	Adam	89.74%
Ext. ORION	ReLU	SDG	**90.41%**
Ext. ORION	ReLU	Adam	89.28%

TABLE VI
RESULTS OBTAINED ON TEST SET CONSIDERING ONLY THE CLASSES IN THE TRAINING PROCESS. ALL THE RESULTS ARE OBTAINED USING MODELNET10 WITH A MAXIMUM NUMBER OF 12 ROTATIONS AS DATASET.

B. ORION

We proceed introducing the results obtained with a NN that learns also the orientation of the object, along with its class. In this case the parameter γ in (1) is set to 0.5, as specified in V-A. The experiments are carried out employing both SGD with momentum and Adam optimizers, like in VI-A. However, to gain a more comprehensive understanding of the results and facilitate the analysis, three different types of activation functions are utilized in this scenario: ReLU, LeakyReLU and ELU. This variation aims to explore the network's behavior while using different non-linear activation functions and to analyze the impact on performance.

The results of the tests on the datasets with 12 and 24 rotations are showed in the rows associated to the network ORION in table VII, for both ModelNet10 and ModelNet40. It can be observed that the performance of the network with different activation functions is comparable in both ModelNet10 and ModelNet40. Overall, in each run, ELU tends to perform slightly better than the other two, improving the class accuracy by 0.2% on average. Let us now analyze the choice of the optimizers: it can be seen that overall SDG outperforms Adam in terms of class accuracy (as observed also in VI-A), while Adam yields better results in predicting the orientation of the object. Since our final scope is object recognition, we may assert that overall the first mentioned optimizer is more efficient for this specific task. In addition, as previously stated in VI-A, SGD tends to often converge faster than Adam, making it more suitable in scenarios where datasets are large (like in the case of ModelNet40) or computational resources are limited. All these observations may be confirmed by Fig. 6.

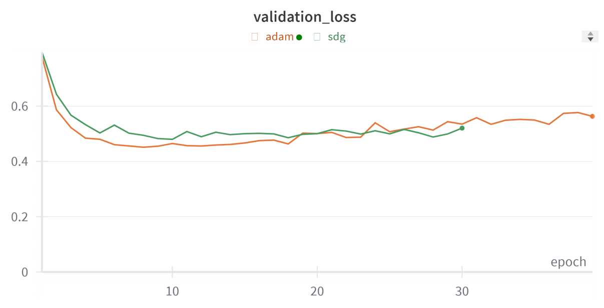

Fig. 6. Validation loss of two ORION networks using SGD and Adam optimizers, trained on the ModelNet10 dataset with a maximum number of rotations set to 12.

Regarding data augmentation, it is observed that increasing the number of rotations of the object does not enhance the performance of the network. The results for both 12 and 24 rotations show comparable class accuracy, while pose accuracy is slightly higher for 12 rotations, though not significantly. The decrease in accuracy in pose prediction with an increased number of rotations is expected. This is because a higher number of rotations introduces more variability and complexity into the dataset, leading to a proportional increase in prediction errors. In conclusion, the results we obtained underscore the fact that the number of rotations of the objects tested (12 and 24) does not significantly impact the network's ability to predict the object's class.

C. Extended ORION

We present the performance of the Extended ORION, as introduced in V-B. This neural network is slightly more complex than its predecessor, implying a potentially longer training time. However, the increased complexity suggests an expectation of overall improved accuracy on the test set. This hypothesis aligns with the results presented in table VII.

Overall, the Extended ORION exhibits a similar behavior to the shallower architecture. All the considerations made in VI-B remain relevant and applicable in the analysis of this extended network. This includes the observations related to activation functions, data augmentation, and optimizer choices.

Dataset	max. # of rotations	network	activation function	Optimizer	Avg Class acc.	Avg Pose acc.
ModelNet10	12	ORION	ReLU	SDG	93.55%	84.09%
-	-	-	LeakyReLU	SDG	93.11%	83.81%
-	-	-	ELU	SDG	**93.75%**	83.72%
-	-		ReLU	Adam	92.34%	84.52%
-	-	-	LeakyReLU	Adam	92.54%	84.28%
-	-	-	ELU	Adam	92.67%	84.32%
ModelNet10	24	ORION	ReLU	SDG	93.55%	82.90%
-	-	-	LeakyReLU	SDG	93.24%	83.16%
-	-	-	ELU	SDG	**93.67%**	82.63%
-	-	-	ReLU	Adam	93.15%	82.97%
-	-	-	LeakyReLU	Adam	92.85%	83.25%
-	-	-	ELU	Adam	93.12%	83.17%
ModelNet10	12	Extended ORION	ReLU	SDG	94.56%	84.14%
-	-	-	LeakyReLU	SDG	94.27%	83.92%
-	-	-	ELU	SDG	**94.62%**	83.89%
-	-		ReLU	Adam	93.82%	84.85%
-	-	-	LeakyReLU	Adam	93.86%	84.36%
-	-	-	ELU	Adam	94.25%	83.97%
ModelNet10	24	Extended ORION	ReLU	SDG	94.15%	83.27%
-	-	-	LeakyReLU	SDG	94.25%	83.15%
-	-	-	ELU	SDG	**94.56%**	82.98%
-	-	-	ReLU	Adam	94.12%	83.35%
-	-	-	LeakyReLU	Adam	94.11%	83.52%
-	-	-	ELU	Adam	93.92%	83.24%
ModelNet40	12	ORION	ReLU	SDG	87.56%	76.00%
-	-	-	LeakyReLU	SDG	87.49%	75.96%
-	-	-	ELU	SDG	**87.60%**	76.12%
-	-	-	ReLU	Adam	87.28%	76.25%
-	-	-	LeakyReLU	Adam	87.12%	76.31%
-	-	-	ELU	Adam	87.25%	76.28%
ModelNet40	24	ORION	ReLU	SDG	88.31%	76.95%
-	-	-	LeakyReLU	SDG	87.93%	77.32%
-	-	-	ELU	SDG	**88.74%**	76.12%
-	-	-	ReLU	Adam	87.82%	76.48%
-	-	-	LeakyReLU	Adam	86.92%	77.15%
-	-	-	ELU	Adam	87.43%	76.93%
ModelNet40	12	Extended ORION	ReLU	SDG	87.76%	76.61%
-	-	-	LeakyReLU	SDG	87.47%	76.11%
-	-	-	ELU	SDG	87.82%	76.58%
-	-	-	ReLU	Adam	**87.97%**	77.20%
-	-	-	LeakyReLU	Adam	87.36%	76.95%
-	-	-	ELU	Adam	87.57%	77.13%
ModelNet40	24	Extended ORION	ReLU	SDG	**87.91%**	75.12%
-	-	-	LeakyReLU	SDG	87.79%	76.11%
-	-	-	ELU	SDG	87.83%	75.92%
-	-	-	ReLU	Adam	86.54%	76.21%
-	-	-	LeakyReLU	Adam	85.43%	75.98%
-	-	-	ELU	Adam	87.76%	77.12%

TABLE VII
RESULTS OBTAINED TRAINING THE NETWORKS USING MODELNET10 AND MODELNET40 DATASETS, FOR THE MAXIMUM NUMBER OF ROTATIONS OF 12 AND 24

D. ResNet-34

Regarding the second type of approach, which involves using the ResNet-34 model for 2D image classification, we implemented a training loop comprising 100 epochs. Additionally, early stopping was applied, terminating training if there were no improvements in validation accuracy over 10 consecutive epochs. We considered two type of optimizer: SGD with momentum and Adam. In addiction, we maintained ReLU as activation function and Cross-Entropy as loss function for all ResNet-34 experiments. In Table VIII the final accuracy results over the test set can be seen for both the datasets.

Dataset	Optimizer	Avg. Class Accuracy
ModelNet10-2D	SDG	95.79%
ModelNet10-2D	Adam	**95.89%**
ModelNet40-2D	SDG	90.56%
ModelNet40-2D	Adam	**91.87%**

TABLE VIII
RESULTS OBTAINED ON TEST SET USING RESNET-34.

In the case of ModelNet10-2D, we found that the optimal parameters using SGD were a learning rate of 10^{-4}, a batch size of 20, momentum set to 0.9, and a weight decay of 10^{-4} for regularization. Alternatively, when using Adam, the best parameter choices remained the same for learning rate and weight decay, but the optimal batch size was 15.

For ModelNet40-2D, the optimal parameter set remained the same as before, except for the batch size, which was adjusted to 80 for both optimizers. This modification was influenced by the substantial increase in the dataset size (approximately

2.5 times that of ModelNet10-2D) and, consequently, the computation time. While a smaller batch size could potentially yield better results, the available computational resources were insufficient for such a task.

Comparing the two optimizers, they exhibit similar convergence times. However, in terms of accuracy, Adam consistently outperforms SGD in all the conducted experiments, as observed in Table VIII.

E. Comparative Analysis

After examining each individual approach, it is pertinent to carry out a comparative analysis. The most relevant outcome is a notable improvement in class accuracy, when comparing the results in table VII and VI-A. As hypothesized, this highlights an advancement in the network's performance when integrating pose prediction into the object classification task. The incorporation of pose information significantly enriches the understanding of the objects, resulting in enhanced accuracy and overall efficiency in the classification process. This statement is further substantiated in the following Fig. 7. As highlighted, the total validation loss of the network, which also learns the pose, is higher. This is because learning the pose alone is more challenging and contributes to a higher loss. Specifically, the class validation loss of the network with pose information is smaller than that of the network learning only the class.

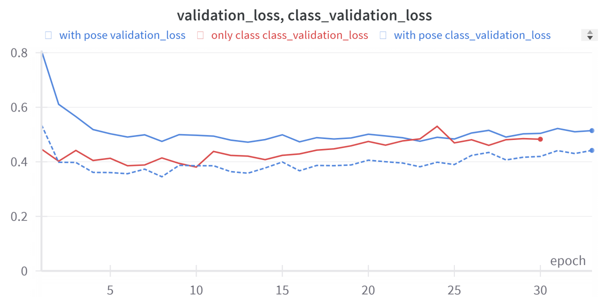

Fig. 7. Class validation and total loss of two ORION networks—one learning both class and pose, the other only the class. Both networks use the SGD optimizer and are trained on the ModelNet10 dataset

When dealing with large datasets, such as the ones utilized in our study, shallower networks tend to perform worse, as it is less likely to overfit the data. This phenomenon is evident when comparing the results obtained using ORION and Extended ORION with the latter demonstrating greater efficiency (as proved in Fig. 8), even though the performance of the former are still satisfactory.

Another example of this principle is the comparison involving ResNet-34, which outperforms the other two methodologies. The complexity of ResNet-34 significantly surpasses that of Extended ORION, comprising 34 parameter layers. Moreover, the ResNet-34 approach incorporates the technique of transfer learning, initiating the training process with a pretrained network model designed for object classification.

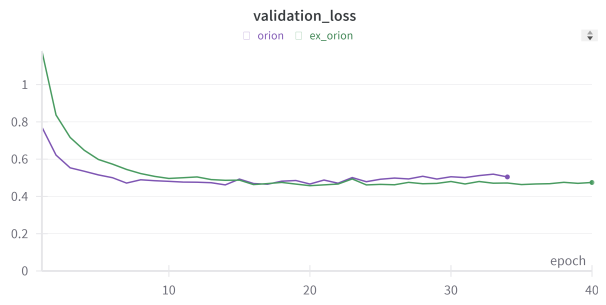

Fig. 8. Class validation loss of the ORION and Extended ORION. Both networks use the SGD optimizer and are trained on the ModelNet10 dataset with a maximum number of rotations set to 12.

Despite maintaining a bidimensional structure, the ResNet-34 model benefits from a considerably larger dataset during the training phase compared to ORION, as evident from the comparison between Table II and Table I. All these factors combined provide a logical explanation for the substantial accuracy increase obtained by the ResNet-34 network. The improvements are not negligible for both ModelNet10 and ModelNet40 datasets, gaining more than 2% accuracy in both cases.

However, it is important to note that this approach comes with the downside of increased computational time. For instance, one epoch in ResNet-34 requires approximately 15 minutes, while in ORION it takes around 2 minutes, considering our computational resources. This tradeoff between accuracy and time efficiency is crucial and needs to be taken into account, especially in scenarios where computational resources are limited.

VII. Concluding Remarks

In this study we introduced three different methodology to solve the 3D classification problem, ORION, Extended ORION and ResNet-34, highlighting their structure and comparing the performances both in terms of accuracy and computational efficiency.

Upon comparing the results obtained from the two ORION networks, we discovered that Extended ORION outperform its simplified counterpart. Furthermore, we proved the advantages of introducing pose information in the classification task. Additionally, our investigation revealed that, while the use of ResNet leads to superior accuracy results, it necessitates not only a higher computational load but also the creation of a multi-view dataset derived from CAD models. Consequently, depending on the application, one model may be preferred over the other, and vice versa.

In the context of ORION, potential avenues for further exploration going into detail into datasets with increased complexity, such as those which comprises additional perspectives (like rotations around other axes) or an augmented number of elements. Additionally, the investigation of more intricate neural network architectures, for instance, another advanced versions of ORION with increased depth or complexity, could offer insights into enhanced performance.

Regarding ResNet-34, this research can be extended by analyzing the third setup dataset, which contains much more data. Simultaneously, exploring deeper ResNet architectures such as ResNet-50 or ResNet-101 could be beneficial. These steps would likely lead to an improvement in model accuracy, even if at the cost of increased computational time. Thus, there is a need to find methods to accelerate the training phase, such as considering alternative cloud-based platforms to Google Colab or experimenting with distributed training across multiple devices.

VIII. Contributions

We would like to acknowledge the individual contributions of each team member to this project.
The work was evenly distributed among the team members as much as possible. Alessandro took care of the data preprocessing related to ModelNet and the coding of ORION in all its versions, with the contribution of Giorgio. Then, Giorgio extensively tested the architectures and collected experimental data. Maddalena, in turn, led the entire work related to ResNet-34, from data preprocessing to the experimental phase.

Despite the specificity of individual roles, collaboration was a constant thread woven throughout every phase of the project. This collaborative spirit aimed not only to address challenges that arose but also to foster a deeper understanding of the project's global dynamics. Regular communication and shared problem-solving sessions created an environment where insights and expertise were seamlessly exchanged, enhancing the overall quality of our work.

Through this research, each component of the group achieved a comprehensive understanding of various critical aspects in the realm of 3D object classification using neural networks. In particular, we developed skills in the experimental design field, enabling us to select fair comparisons, choose appropriate evaluation metrics, and fully understand the significance of the results.

The main difficulties encountered concerned data preparation: from understanding the structure of the ModelNet datasets to the actual data preprocessing. Another challenge was coping with limited computational resources, which slowed down the experimentation process for the ResNet part. In particular, the training phase had to be interrupt multiple times due to a shortage of Google Colab resources.

References

[1] Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller; ”Multi-View Convolutional Neural Networks for 3D Shape Recognition”, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 945-953.

[2] B. T. Phong. ”Illumination for computer generated pictures.”, Commun. ACM, 18(6), 1975.

[3] Xu, Jinglin and Zhang, Xiangsen and Li, Wenbin and Liu, Xinwang and Han, Junwei, ”Joint Multi-view 2D Convolutional Neural Networks for 3D Object Classification”, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020, pp. 3202-3208.

[4] K. He, X. Zhang, S. Ren and J. Sun, ”Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[5] N. Sedaghat, M. Zolfaghari, E. Amiri, T. Brox”Orientation-boosted Voxel Nets for 3D Object Recognition” in British Machine Vision Conference (BMVC), 2017.

[6] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao ”3D ShapeNets: A Deep Representation for Volumetric Shapes”. Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)

[7] D. Maturana and S. Scherer. ”VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition.” IROS2015.