Optimizing 3D Object Classification with Lightweight CNN Architectures

Marco Cali'[†]

***Abstract*—3D object recognition is a crucial task in robotics and computer vision, enabling machines to better understand and interact with their environment. While traditional methods rely on 2D images, incorporating 3D data enhances object recognition by capturing depth and spatial structure. In this work, we develop an efficient 3D convolutional neural network (3D CNN) for voxel-based object classification. By leveraging the ModelNet10 dataset, we demonstrate that small and computationally efficient 3D CNN architectures can achieve accurate object classification without sacrificing performance. Our approach focuses on optimizing network design to ensure fast inference times while handling medium-resolution voxel grids, making it well-suited for real-time applications in robotics and autonomous systems. Experimental results show that our model strikes a balance between computational cost and classification accuracy, offering a promising solution for 3D object recognition tasks.**

***Index Terms*—Supervised Learning, ModelNet10, Convolutional Neural Networks, 3D Classification, Robotics Perception.**

I. Introduction

The integration of computer vision and machine learning has greatly advanced robotics, enabling machines to perceive and interact with their environment more effectively. A key aspect of robotic perception is vision, which helps robots interpret and engage with their surroundings. While 2D images have been the traditional input for visual systems, incorporating 3D data, such as depth and spatial structure, provides a richer understanding of the environment. Deep learning models can leverage this 3D information to classify and identify objects based on their geometry, offering significant advantages for tasks like autonomous navigation, grasping, and spatial reasoning.

3D data, often acquired through LiDAR sensors, is commonly represented as point clouds, unordered sets of points defining positions in 3D space. An alternative representation is provided by volumetric pixels, or *voxel* grids, which discretize 3D space into uniform cells, providing a structured, volumetric representation similar to pixels in 2D images. This structure allows for the application of Convolutional Neural Networks (CNNs) in a manner analogous to 2D image processing, making voxel grids an intuitive and powerful representation for object recognition in 3D space.

In this work, we focus on developing efficient and fast 3D convolutional neural networks (3D CNNs) for voxel-based object classification. Leveraging the ModelNet10 dataset [1], we demonstrate how compact 3D CNN architectures can achieve accurate object recognition while processing medium-resolution voxel grids. Our goal is to design networks that balance high performance with computational efficiency, ensuring fast inference times without compromising accuracy.

This paper is structured as follows: Section II covers related work, Section III provides an overview of our approach, Section IV details the data structure and preprocessing steps, and Section V explains the model and algorithm. In Section VI, we present our results, and finally, in Section VII, we conclude with a discussion of our findings and outline possible directions for future work.

II. Related Work

In recent years, 3D object recognition has garnered significant attention, with numerous studies exploring deep learning techniques for processing 3D data. The ModelNet10 dataset, introduced by Wu et al. [1], serves as a benchmark for 3D object classification using CAD models. This dataset has been extensively adopted for training and evaluating deep learning models that operate on voxel grids and point clouds, underscoring its pivotal role in advancing 3D perception.

Maturana and Scherer [2] introduced VoxNet (2015), a pioneering 3D Convolutional Neural Network (3D CNN) designed specifically for voxel grid-based classification. Their work demonstrated how volumetric representations like voxel grids could be effectively used with 3D CNNs to improve object recognition accuracy. The architecture presented in VoxNet is notable for its simplicity and computational efficiency, making it a key reference for subsequent research in this area.

Building on the foundation laid by VoxNet, Sedaghat et al. [3] introduced the ORION algorithm, which posits that objects induce distinct features in the network under rotation. They formulated a multi-task learning approach, enabling the simultaneous classification of objects and prediction of their orientations.

Another significant contribution is FusionNet, proposed by Hegde and Zadeh [4]. FusionNet integrates multiple data representations by fusing both voxel grids and depth images. This multi-view approach takes advantage of the strengths of each representation, resulting in higher classification accuracy. The model leverages a much larger architecture with a total of 118 million parameters, achieving state-of-the-art results at the time. However, the significant computational complexity and memory requirements make it less suitable for real-time or resource-constrained applications.

[†]Department of Information Engineering, University of Padova, email: marco.cali.2@studenti.unipd.it

In parallel to voxel-based approaches, Qi et al. introduced PointNet [5], a groundbreaking architecture designed for processing point clouds. PointNet departed from the voxel grid representation by working directly with point cloud data, significantly reducing memory and computational costs while still achieving competitive performance.

These foundational works illustrate the potential of 3D CNNs in addressing complex geometric data for voxel-based object classification. Our research builds upon these contributions by focusing on the design of efficient, smaller 3D CNN architectures that achieve a balance between accuracy and computational efficiency, particularly in processing medium-resolution voxel grids.

III. Processing Pipeline

This section outlines the key steps of our approach. First, we convert the ModelNet10 dataset from its original mesh format (.off files) to a voxel grid representation, which is compatible with CNNs. The resolution for voxelization and preprocessing options offer flexibility, and these are discussed in greater detail in the next section. Once voxelized, we augment the dataset by generating rotated versions of the objects to increase variety. The dataset is then split into training and validation sets, with the test set loaded from specific folders in the ModelNet10 dataset. The validation set plays a crucial role in performing early stopping and tuning hyperparameters.

We utilize the Simple3DCNN architecture as the training model, employing the Adam optimizer [6] over a specified number of epochs. To prevent overfitting, early stopping is implemented. After training, the model is evaluated on the test dataset, with accuracy serving as the primary performance metric. However, recognizing its limitations, we also analyze confusion matrices for a more detailed evaluation.

IV. Signals and Features

The dataset consists of .off files, which describe the polygons that compose each geometric object.

Fig. 1: Visualization of a chair CAD .off file.

The first step involves converting these .off files into voxel grids. For this, we need to define the grid size. A higher grid resolution provides more accurate voxelization, but since the memory requirements scale cubically with grid size, it's essential to find a balance between data quality and computational overhead. In this work, prioritizing efficiency and speed, we find that a 32x32x32 grid offers a good tradeoff.

To fit an object within this grid, we set the object's longest dimension to 28 units, uniformly scaling the rest of the object accordingly. This process determines the grid's pitch (voxel size), which defines the resolution of the voxel grid. After scaling, the object is centered within the grid by padding it uniformly. To enhance the dataset, we augment the voxel grids by applying three 90-degree rotations. Finally, all voxel grids are transformed from a binary range of $\{0, 1\}$ to $\{-1, 1\}$ for improved numerical stability during training.

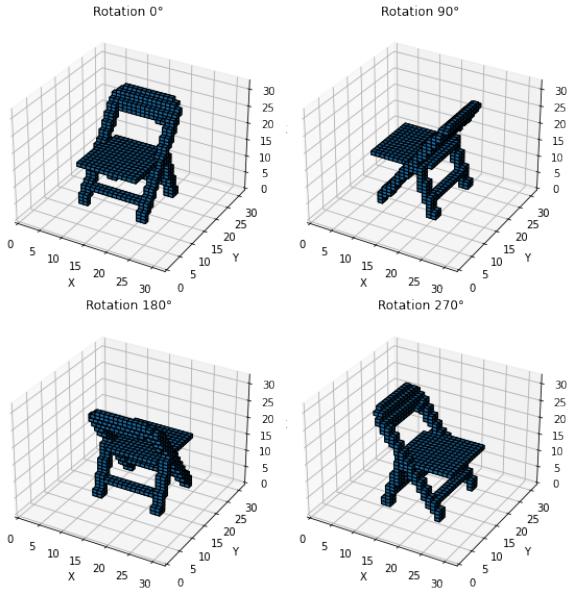

Fig. 2: Rotated voxel grids

Following the data processing steps, the dataset is split into training and validation sets with an 85/15 ratio. The training set is taken from the dataset's designated training folders, while the test set is kept separate and exclusively drawn from the test folders. This ensures a fair evaluation of the model's performance during testing.

Fig. 3: Class distributions of training and validation datasets.

From Figure 3, we observe that training and validation datasets have very similar class distributions. This consistency ensures that the model is trained and validated on comparable data, leading to more reliable performance evaluations.

V. Learning Framework

This section presents the architecture of our chosen model, Simple3DCNN. Starting with the input, the model takes in a voxel grid represented by a single channel, as voxel grids do not have color channels like RGB images. This single-channel input is passed through a series of three 3D convolutional layers, each progressively extracting more complex features from the input data.

The first convolutional layer uses 16 filters, each with a kernel size of 3x3x3, and it applies a stride of 1 with padding to ensure the spatial dimensions of the voxel grid are preserved after this operation. After the convolution, a ReLU activation function is applied to introduce non-linearity, followed by a 3D max pooling layer. This pooling layer, with a kernel size of 2x2x2, effectively downsamples the voxel grid, reducing its dimensionality and extracting the most significant features.

Next, the second convolutional layer increases the number of filters to 32, again using a 3x3x3 kernel, and follows the same process of ReLU activation and max pooling. By increasing the number of filters, the model captures more detailed and abstract features as the data progresses through the network.

The third convolutional layer further increases the filter count to 64, continuing the pattern of convolution, activation, and pooling. By this stage, the voxel grid has been significantly downsampled, and the features extracted represent high-level characteristics of the 3D object.

Following the final convolutional layer, the data is flattened into a 1D vector, which serves as input to the fully connected layers. The first fully connected layer has 64 units, with a ReLU activation function applied to introduce non-linearity. To prevent overfitting, dropout regularization is applied at this stage with a dropout rate of 20%, randomly setting some activations to zero during training to enhance generalization.

Finally, the second fully connected layer maps the output to the desired number of classes, corresponding to the categories in the ModelNet10 dataset. This final layer outputs the class scores, from which the predicted class is determined.

This architecture effectively balances complexity and computational efficiency, making it well-suited for classifying 3D objects represented by medium-resolution voxel grids, despite having only *333k* parameters. By progressively extracting more abstract features and incorporating dropout for regularization, the Simple3DCNN model aims to achieve both high accuracy and fast inference times.

The model is trained using the Adam optimizer with a learning rate of 0.001. The training process consists of 50 epochs, with a patience parameter set to 5 for early stopping based on validation loss. The loss function used is categorical cross-entropy, suitable for multi-class classification tasks.

Layer	**Output Shape**	**Parameters**
Input	(1, 32, 32, 32)	-
Conv3D + ReLU	(16, 32, 32, 32)	16 filters, 3x3x3
MaxPool3D	(16, 16, 16, 16)	2x2x2
Conv3D + ReLU	(32, 16, 16, 16)	32 filters, 3x3x3
MaxPool3D	(32, 8, 8, 8)	2x2x2
Conv3D + ReLU	(64, 8, 8, 8)	64 filters, 3x3x3
MaxPool3D	(64, 4, 4, 4)	2x2x2
Flatten	(4096)	-
FC + ReLU	(64)	Fully Connected
Dropout (20%)	(64)	Dropout
FC (Output)	(Num Classes)	Fully Connected

TABLE 1: Simple3DCNN Architecture

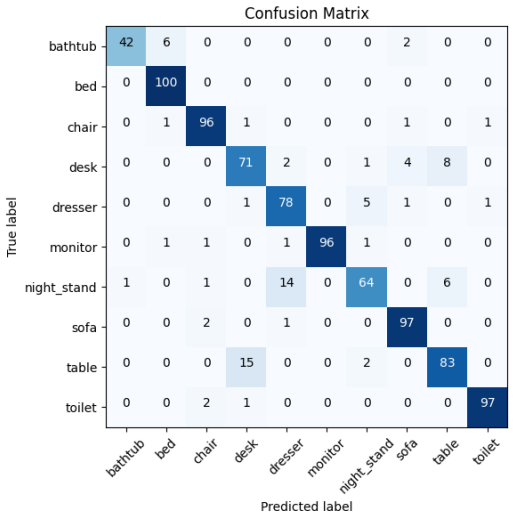

Fig. 4: Confusion matrix

VI. Results

The results obtained from the experiments show that the Simple3DCNN model performs well on the ModelNet10 dataset, achieving a classification accuracy of 90.7%. Additionally, a top-2 accuracy score of 96.6% suggests that, even when the model does not predict the correct class as the top choice, it frequently ranks it among the top two predictions, indicating robustness in the model's feature extraction and classification abilities.

The confusion matrix in Figure 4 further highlights the performance of the model across different classes. The classification report provides detailed performance metrics, including precision, recall, and F1-score, for each class.

Notably, certain categories like "monitor," "bed," and "toilet" achieve near-perfect scores, with high precision and recall, showing that the model can confidently classify these objects with high accuracy. However, other categories like "night stand" and "desk" show slightly lower performance, likely due to greater variation in object shapes or similarities with other classes. These challenges are reflected in the slightly lower recall values for these categories. Moreover, the unbalanced

class distribution of the dataset favors the most represented classes.

	precision	**recall**	**f1-score**	**support**
chair	0.94	0.96	0.95	100
sofa	0.92	0.97	0.95	100
toilet	0.98	0.97	0.97	100
night stand	0.88	0.74	0.81	86
bathtub	0.98	0.84	0.90	50
dresser	0.81	0.91	0.86	86
bed	0.93	1.00	0.96	100
desk	0.80	0.83	0.81	86
monitor	1.00	0.96	0.98	100
table	0.86	0.83	0.84	100
accuracy			0.91	908
macro avg	0.91	0.90	0.90	908
weighted avg	0.91	0.91	0.91	908

TABLE 2: Metrics

The macro average of the F1-score (0.90) and the weighted average (0.91) are both strong indicators of overall model performance, demonstrating that the model generalizes well across all classes. Finally, some example predictions from the test dataset are presented in Figure 5, providing a qualitative view of the model's predictions in real-world cases. These visualizations help validate the numerical results, illustrating how the model accurately captures the 3D structure of the objects in most cases.

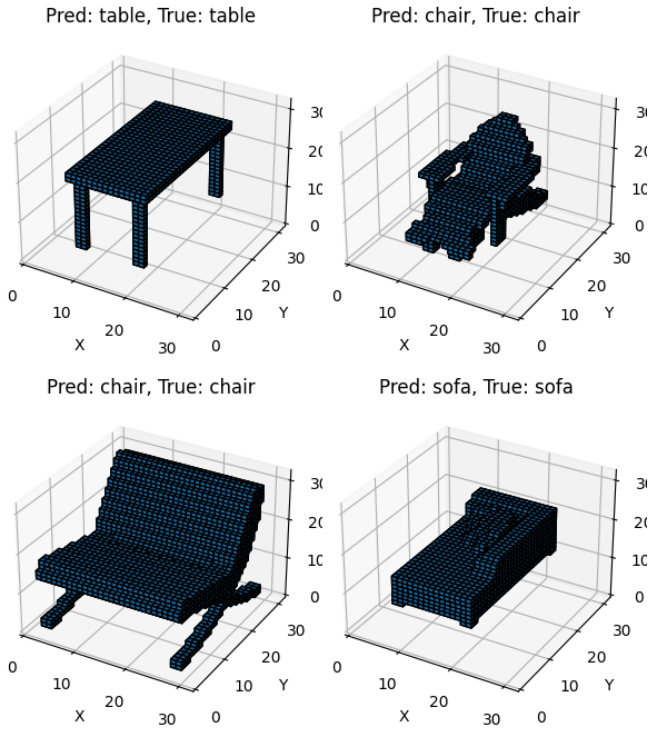

Fig. 5: Prediction examples

To better contextualize the performance of our proposed architecture, we compare it with several established models in 3D object classification. As shown in Table 3, although models like FusionNet and ORION achieve higher accuracy scores, they come at the cost of significantly larger parameter sizes. FusionNet, for example, requires over 118 million parameters, making it much less computationally efficient than our model, which uses only 333k parameters. While our model achieves an accuracy of 90.7%, slightly below that of VoxNet, it offers a much more compact and efficient solution, with just a third of the parameters. This demonstrates the effectiveness of our approach in balancing accuracy and computational efficiency, particularly in scenarios where limited resources are a concern.

Model	**Accuracy (%)**	**Number of Parameters**
3D ShapeNet [1]	83.5	12.4 M
VoxNet [2]	92	0.92 M
ORION [3]	93.9	4 M
PointNet [5]	89.2	3.5 M
FusionNet [4]	93.1	118 M
Our	90.7	0.33 M

TABLE 3: 3D Classification Models on ModelNet10

Importantly, the entire test dataset takes only 1.21 seconds to infer, showcasing the efficiency of our model in real-time applications. The complete code and model are publicly available at https://github.com/MarcoCali0/3dclassifier

VII. Concluding Remarks

In this work, we presented a focused approach towards 3D object classification using voxel grids, emphasizing the design of a compact and computationally efficient neural network architecture. The Simple3DCNN model we developed balances the need for high accuracy while maintaining a significantly lower number of parameters compared to more complex architectures. By leveraging three convolutional layers followed by fully connected layers with dropout regularization, the model efficiently captures high-level features from medium-resolution voxel grids, leading to an accuracy of 90.7% on the ModelNet10 dataset. Notably, the model achieves this performance with only 333k parameters, showcasing its potential for deployment in resource-constrained environments.

Our results demonstrate that the Simple3DCNN is competitive with more parameter-heavy models while requiring significantly fewer computational resources. Despite its simplicity, our model provides a viable solution for 3D object classification, especially in scenarios where speed and memory efficiency are crucial.

Looking forward, there are several avenues for future work. One potential direction involves exploring hybrid approaches that fuse voxel grid representations with point cloud data, similar to FusionNet, but with a focus on maintaining computational efficiency. Another area of interest could be to incorporate techniques like data augmentation or transfer learning

to further improve accuracy, especially for underrepresented classes in the dataset.

Throughout this project, we encountered the real-world challenges of working with neural networks, particularly the complexities that arise when dealing with unbalanced datasets and how outcomes can sometimes deviate from theoretical expectations. One of the key lessons was the critical importance of data processing, as it significantly influences the model's performance. We also recognized the delicate balance between computational speed and memory consumption, and the limitations of commonly used metrics like accuracy, which do not always tell the full story. This project underscored the importance of thoughtful trade-offs in model design and evaluation, which are essential for developing efficient, real-world neural network solutions.

In summary, our approach demonstrates that it is possible to strike a balance between model complexity and performance, setting the stage for future developments in lightweight 3D object recognition architectures.

REFERENCES

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, June 2015.

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

[3] N. Sedaghat, M. Zolfaghari, and T. Brox, "Orientation-boosted voxel nets for 3d object recognition," *CoRR*, vol. abs/1604.03351, 2016.

[4] V. Hegde and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations," *CoRR*, vol. abs/1607.05695, 2016.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016.

[6] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, (San Diega, CA, USA), 2015.