Residual Neural Networks for 3D Object recognition

Enrico D'Alberton, Pietro Girotto, Yi Jian Qiu

***Abstract*—In the rapidly evolving domain of 3D object classification, traditional Convolutional Neural Networks (CNNs) have dominated the landscape, primarily leveraging spatial feature extraction techniques. This research introduces a novel approach by employing Residual Neural Networks (ResNets) to address 3D shape recognition, challenging the conventional paradigm of static 3D CNN architectures. By exploiting the sequential processing capabilities of ResNets, we demonstrate a sophisticated method for capturing intricate geometric and topological features of 3D objects that goes beyond traditional spatial convolution techniques. Our proposed ResNet-based classification framework was rigorously evaluated on the ModelNet40 benchmark dataset, a standard and comprehensive repository for 3D shape classification research. The experimental results reveal significant improvements over existing methodologies, achieving a remarkable 87.56% accuracy. The key change lies in the ResNet's ability to process 3D point cloud data as a sequential representation, enabling dynamic feature learning that captures both local and global structural dependencies. By treating 3D object point clouds as voxel grids, our model can learn hierarchical representations that traditional spatial convolution methods often struggle to extract. This work provides a foundation for exploring the potential of Residual Neural Networks in 3D object classification, with opportunities for refinement and adaptation in future research.**

I. Introduction

The field of 3D object recognition has emerged as a crucial component in computer vision applications, ranging from autonomous navigation to robotics and augmented reality. However, accurate classification of 3D objects presents several fundamental challenges that distinguish it from traditional 2D image recognition tasks. Objects viewed from different angles can appear drastically different, making it difficult to maintain consistent recognition across varying viewpoints. Additionally, 3D data often undergoes various geometric transformations, including rotation, scaling, and translation, which must be properly handled by recognition systems. Deep learning approaches, particularly Convolutional Neural Networks (CNNs), have revolutionized how we address these challenges in 3D object recognition. CNNs excel at automatically learning hierarchical features from raw input data, eliminating the need for hand-crafted feature engineering. The convolutional operations in these networks can be adapted to process 3D input formats, whether represented as voxel grids, point clouds, or multi-view projections. Furthermore, the spatial invariance properties of CNNs make them particularly suitable for handling the geometric variations inherent in 3D data. Despite the success of CNNs, training deep neural networks for 3D object recognition remains challenging due to the vanishing gradient problem, where the gradients become exponentially small as they propagate through multiple layers. Residual Networks (ResNets) address this limitation through their skip connection architecture, which allows gradients to flow directly through the network. These skip connections enable the training of significantly deeper networks, which is crucial for learning the complex hierarchical features necessary for robust 3D object recognition. ResNets have demonstrated remarkable success in 2D vision tasks, and their principles can be effectively adapted to the 3D domain. This paper presents some approaches for 3D object recognition using ResNets by training the network on ModelNet40. We introduce modifications to the traditional CNN VoxNet architecture [1] specifically designed to handle 3D input data, by including a serie of residual blocks after having handled the input. Moreover, we propose data augmentation techniques that enhance the network's ability to handle sparse 3D data and geometric transformations. These contributions might allow new insights to advance the state-of-the-art in 3D object recognition and provide valuable guidelines for future research in this domain. This report is structured as follows: in Section II we present related works in which ResNets have been used for 3d object classification and also different state-of-the-art approaches. The methodologies used for training the model and the relative results are respectively presented in Sections III and IV. Finally, we present the conclusions and the future works for this project.

II. Related Work

The field of 3D object recognition has advanced significantly, transitioning from handcrafted features like Spin Images, Point Feature Histograms, and 3D Shape Context to deep learning approaches that automate feature extraction. Early works such as VoxNet [1] employed 3D convolutional networks to process voxelized 3D data, achieving promising results by capturing spatial relationships within 3D shapes. ShapeNets [2] extended this by representing 3D shapes as probabilistic voxel grids using Deep Belief Networks, laying the groundwork for tasks like shape completion and recognition.

Approaches such as Multi-View CNNs (MVCNNs) [3] introduced the idea of projecting 3D models into multiple 2D views and leveraging 2D CNNs to learn from these projections. Kd-Networks [4], on the other hand, directly process point clouds without requiring voxelization, avoiding potential scaling issues. Moreover, also orientation-aware methods like Orientation-Boosted VoxelNet (ORION) [5] are integrated as auxiliary tasks adding pose estimation into

classification frameworks. This multi-task learning approach not only improved classification accuracy but also enhanced rotational invariance, demonstrating its utility on datasets like ModelNet40.

Despite these advancements, many existing approaches face challenges related to computational cost and the ability to fully exploit geometric properties in 3D data. Ensemble models, while achieving high accuracy, require extensive computational resources, making them impractical in many applications.

This paper builds on these foundations by introducing a novel ResNet-based framework that adapts residual connections for 3D data processing using ModelNet40 as dataset. Our method effectively addresses sparsity and geometric transformations through occupancy grids and interpolation.

III. Methodology

A. Dataset preparation

This section describes the preprocessing pipeline and feature extraction methods applied to the ModelNet40 dataset to prepare the data for classification using deep learning techniques. The ModelNet40 dataset contains 3D CAD models of objects represented in .off (Object File Format) files. Each file specifies the geometry of an object using vertices and faces, describing the mesh structure of the 3D model. For processing, these files are loaded into memory, extracting the vertices and faces that define the object. The models are then transformed into a voxelized representation for compatibility with the neural network input requirements. The voxel grid is formatted as a binary 3D array, with each voxel indicating the presence or absence of object material. These have been the followed steps:

- **Voxelization:** The 3D mesh data is voxelized by projecting the geometry onto a uniform grid of a specified resolution (default: 64x64x64). This step marks the voxels that intersect the surface of the object, capturing the geometric structure.
- **Filling Gaps:** To handle sparsity and ensure continuity, the binary voxel grid is further processed to fill internal gaps using morphological operations, resulting in a solid representation of the object.
- **Rescaling:** The filled voxel grid is resized to a standardized resolution (default: 32x32x32) using a scaling factor. This ensures uniformity across the dataset and reduces computational overhead during training.
- **Data Augmentation:** Rotational augmentation is applied to the models to improve the network's robustness to orientation variations. Multiple orientations on every axis of each object are generated and included in the dataset.

The voxelized representation serves as the input feature format for the classification model. Each voxel grid is treated as a fixed-size 3D array, where the binary values encode spatial information about the object. No explicit feature engineering is performed, as the deep learning model is designed to learn hierarchical features directly from the voxel data.

B. Network Architecture

The proposed neural network architecture, referred to as ResVoxNet, is designed for 3D voxel-based classification tasks. The architecture leverages both convolutional layers and residual connections to effectively capture spatial dependencies within the input data while ensuring efficient gradient propagation during training. Below, we provide a detailed description of the network showed in Figure 1.

The input to the network consists of 3D voxel grids with dimensions $64\times64\times64$ and a single channel, which is processed through a sequence of three convolutional layers. These layers progressively increase the feature representation depth while reducing spatial resolution. The initial convolution applies a large kernel to capture broad patterns, followed by two additional convolutions that refine the extracted features, each incorporating batch normalization and ReLU activation to enhance stability and non-linearity.

The core of the architecture is a series of ResVoxBlocks, which introduce skip connections to improve gradient flow and prevent degradation in deeper networks. These blocks consist of pairs of 3D convolutional layers, each paired with batch normalization and ReLU activation. The ResVoxBlocks are organized into three stages, with the first stage processing features at 30 channels, the second at 60 channels, and the third at 120 channels. This hierarchical arrangement ensures effective learning across varying scales of abstraction.

After feature extraction through convolutional and residual layers, spatial dimensions are reduced via 3D average pooling, producing a compact representation. The resulting tensor is flattened and passed through two fully connected layers: a dense layer with 512 hidden units that applies dropout for regularization, followed by an output layer that generates logits for classification.

This architecture strikes a balance between depth and computational efficiency. By combining hierarchical feature extraction, residual learning, and effective regularization, the network is adept at processing complex 3D data while maintaining robust generalization capabilities.

Full code repository of the experiment can be found at https://github.com/gp-1108/VoxNet.

C. Learning Framework

To optimize the performance of the ResVoxNet architecture, we explored several variations in the learning framework and experimental setup. Specifically:

- **Input Resolution:** We experimented with input voxel grids of sizes $64\times64\times64$ and $32\times32\times32$ to evaluate the trade-off between computational efficiency and accuracy.
- **K-Fold Cross Validation:** A k-fold cross-validation strategy was employed to ensure robustness and generalization across different splits of the dataset. We stopped each fold training by exploiting a patience factor that allows us to determine if validation and training loss were not improving anymore.

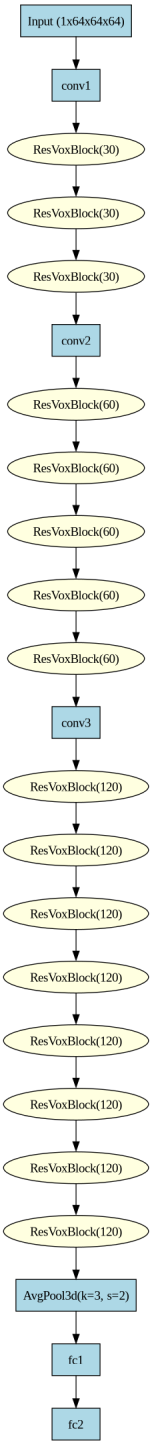

Fig. 1: ResVoxNet architecture.

- **Batch Normalization:** The network extensively utilizes batch normalization layers to stabilize training and accelerate convergence.
- **Loss Function:** In addition to standard cross-entropy loss, we experimented with focal loss to address class imbalance of ModelNet40 and improve the network's focus on challenging samples.

These variations allowed us to systematically evaluate the impact of architectural and optimization choices, leading to insights into the most effective configurations for 3D voxel-based classification tasks.

IV. Results

Our extensive experimentation with various model configurations yielded promising results on the ModelNet40 benchmark dataset. The optimal model configuration achieved an accuracy of 87.56% on the test set. This performance was obtained using a ResNet-based architecture, which incorporated several key design elements. The architecture processes input dimensions of 64×64×64 voxels and employs batch normalization throughout the network. We implemented 3-fold cross-validation to ensure robust evaluation while using a batch size of 256 across 200 training epochs per fold with a patience factor of 5. The model was optimized using Adam with a learning rate of 1e-3, and training was guided by a cross-entropy loss function. To enhance generalization, each sample of the training set was represented in all possible 90-degree rotations throughout the axis as our primary data augmentation strategy.

To systematically evaluate the impact of individual hyperparameters on model performance, we conducted a study where we tested the impact of removing or changing specific components while keeping all other parameters constant. This approach allows us to isolate the contribution of each hyperparameter to the model's overall performance and identify critical design choices in our architecture. In this section, we present a detailed analysis of how variations in key hyperparameters affected the model's performance, providing insights into the robustness and sensitivity of our approach. Please note that many attempts with various combinations of parameters were tried, but here we highlight the key findings.

We first investigated whether the input resolution could impact the learning capabilities of the network. We tested an input grid size of 64×64×64 while keeping all other hyperparameters constant.

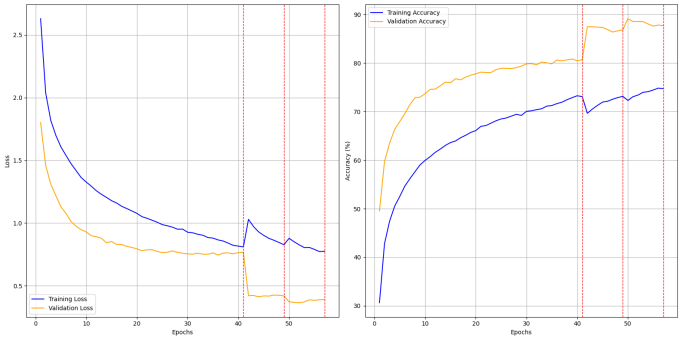

Fig. 2: ResVoxNet with input size 32x32x32. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 77.80%.

Figure 2 shows the learning curve and the accuracy progression over time for ResVoxNet with an input grid of size

$32\times32\times32$. This change reduced the model's test accuracy by almost 10%. Initial performance was poor, and upon reviewing the dataset, we realized that distinguishing the 3D object representations was particularly challenging, even manually. Therefore, we implemented the larger voxel grid size, which improved performance substantially.

Next, we tested different values for the number of folds in K-Fold Cross Validation. Through several attempts, we observed that the number of folds didn't impacted the final results too much.

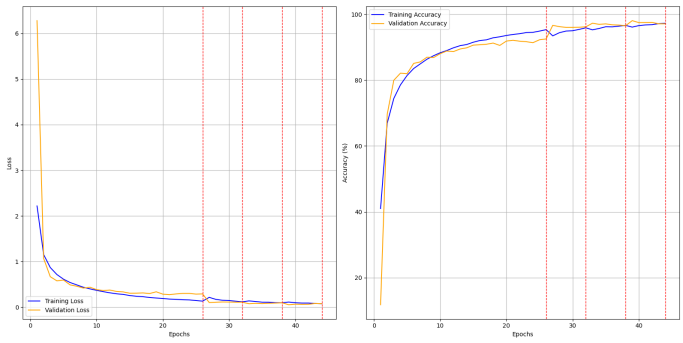

Fig. 3: ResVoxNet with $k = 4$. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 85.53%.

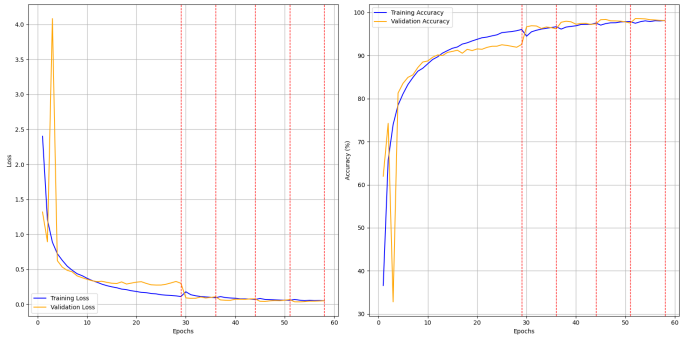

Fig. 4: ResVoxNet with $k = 5$. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 87.03%.

As seen in Figures 3 and 4, the number of folds did not substantially affect the final performance of the model on the test set. Additionally, no specific pattern or correlation between performance and the number of folds was observed, even when testing with other models.

Another key modification was the introduction of batch normalization, which significantly boosted the model's performance, increasing the accuracy by nearly 10% compared to the previous ResVoxNet model without batch normalization. Figure 5 illustrates the learning curve and accuracy progression of ResVoxNet without batch normalization.

Finally, we observed that the class distribution in the ModelNet40 dataset was imbalanced, as shown in Figure 6.

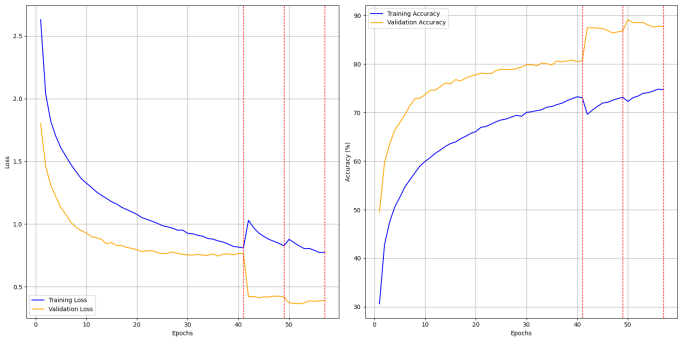

Fig. 5: ResVoxNet without Batch Normalization. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 77.80%.

To address this, we experimented with Focal Loss, a loss function designed to give more weight to less-represented classes. Although we were optimistic about this approach, the results led us to reconsider its effectiveness, as demonstrated in Figure 7.

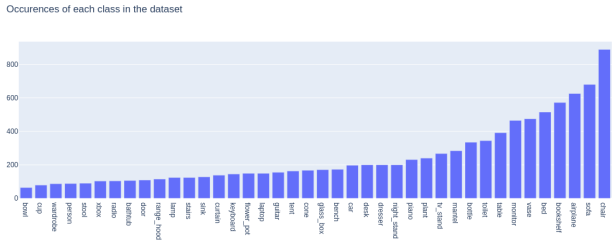

Fig. 6: Class distribution in ModelNet40 dataset.

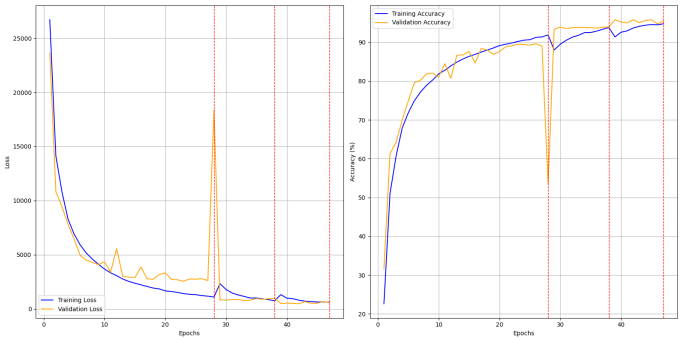

Fig. 7: ResVoxNet with Focal Loss. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 85.29%.

As seen in Figure 7, the performance with Focal Loss was close to that of the baseline ResVoxNet, and we expected it to surpass the standard cross-entropy loss. However, this was not the case.

The final performance of ResVoxNet is shown in Figure 8, where we present the learning curve and accuracy progression for the model with the optimal configuration.

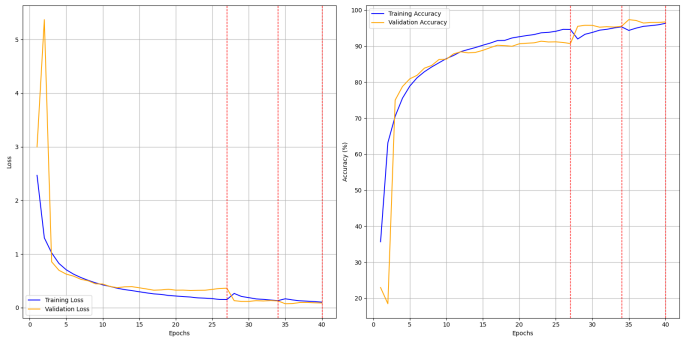

Fig. 8: ResVoxNet. On the left, the learning curve; on the right, the accuracy progression. Vertical dashed lines represent the change of fold. Test accuracy: 87.56%.

For comparison, we include a table of recent state-of-the-art models and their performance on the ModelNet40 dataset.

Model	**Test Accuracy (%)**	**Reference**
ResVoxNet	87.56	our
Pointwise-CNN	86.1	[6]
Deep Sets	87.1	[7]
ECC	87.4	[8]
PointNet	89.2	[9]
SCN	90.0	[10]
Kd-Net(depth=10):	90.0	[4]

TABLE 1: Comparison of Your Model's Test Accuracy with Recent Models on ModelNet40

V. Conclusions and Future Work

In this paper, we presented ResVoxNet, a ResNet-based architecture for 3D object classification, which achieved a test accuracy of 87.56% on the ModelNet40 dataset. This model outperforms several existing approaches, demonstrating the effectiveness of key design choices such as batch normalization and the use of a $64\times64\times64$ voxel grid input. Our experiments also included a focused study to assess the impact of various hyperparameters, revealing that batch normalization significantly improved performance and also a greater voxel grid size. Despite this, we found that handling class imbalance through Focal Loss only marginally improved results, suggesting that further exploration into more robust strategies for balancing classes is warranted. Additionally, although our model performs competitively, future work could focus on enhancing its efficiency, possibly through the use of lighter architectures and exploring more advanced data augmentation strategies, as well as alternative loss functions like triplet or contrastive loss. Moreover, evaluating the model on other datasets like ShapeNet or ScanNet would help assess its generalizability. Overall, while the model shows promising results, there are numerous avenues for further improvement and expansion, which could contribute to the ongoing development of 3D object classification techniques.

References

[1] 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)*, pp. 922–928, 2015.

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," 2015.

[3] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," 2015.

[4] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," 2017.

[5] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3d object recognition," 2017.

[6] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," 2018.

[7] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," 2018.

[8] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," 2017.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017.

[10] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional shapecontextnet for point cloud recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4606–4615, 2018.