A Comparative Study on the Effect of Grid Size and Interpolation Techniques in 3D Shape Classification with CNN

Surya Prakash V
ELMCAD Co. Ltd.
Seoul, South Korea
surya.prakash@msds.christuniversity.in

Yong-Woon Kim
Christ University
Pune, India
defeatst.kim@gmail.com

***Abstract*— This paper presents a comprehensive study on 3D shape classification utilizing Convolutional Neural Networks (CNNs). The research investigates the performance variations arising from different grid sizes and the application of interpolation techniques. A systematic analysis is conducted to determine how these factors influence the accuracy and efficiency of 3D shape classification. The experiments reveal that grid size significantly impacts the CNN's ability to accurately classify shapes with respect to computation time, with larger grid sizes enhancing classification accuracy and other performance metrics. When interpolation techniques are applied, performance metrics further improve, with the highest grid sizes achieving optimal accuracy, precision, recall, and F1-score. These findings contribute valuable insights for optimizing 3D shape classification models, demonstrating the crucial roles of both grid size and interpolation techniques in achieving high classification accuracy and efficiency in practical applications.**

Keywords— ***Convolutional neural network (CNN); Zoom Interpolation, 3D Shapes***

I. Introduction

The classification of 3D shapes is a crucial task in various fields such as computer vision, robotics, medical imaging, and augmented reality. Accurate classification of 3D shapes enables applications ranging from object recognition and scene understanding to the development of autonomous systems and advanced human-computer interaction interfaces. Traditional methods for 3D shape classification have relied heavily on handcrafted features and geometric descriptors, which often require extensive domain knowledge and can be sensitive to noise and variations in the data.

With the advent of deep learning, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for image classification, demonstrating remarkable performance across a wide range of tasks. CNNs are particularly well-suited for handling the spatial hierarchies in visual data, making them an attractive option for 3D shape classification. However, the direct application of CNNs to 3D data presents several challenges due to the inherent complexity and high dimensionality of 3D shapes.

A common approach to mitigate these challenges involves representing 3D shapes as voxel grids or multi-view projections, allowing CNNs to process the data in a format similar to 2D images. This study focuses on the voxel grid representation of 3D shapes, wherein the shapes are discretized into a 3D grid of binary or occupancy values. The resolution of this grid, referred to as the grid size, plays a critical role in determining the balance between computational efficiency and the preservation of shape details.

This research aims to provide a comprehensive analysis of the impact of different grid sizes on the performance of CNN-based 3D shape classification. Additionally, the study examines the role of interpolation techniques in enhancing classification accuracy by addressing the loss of spatial information that occurs during the discretization process. Interpolation techniques can potentially improve the representation of 3D shapes by filling in gaps and smoothing out irregularities, thereby aiding the CNN in better recognizing and classifying the shapes.

The primary contributions of this paper lie in its systematic evaluation and analysis of key factors influencing 3D shape classification using Convolutional Neural Networks (CNNs). First, the paper provides a comprehensive assessment of how varying grid sizes impact both the accuracy and computational efficiency of 3D shape classification. By experimenting with different grid dimensions, the study identifies the optimal sizes that balance the trade-off between detailed representation of shapes and the computational resources required for processing them.

Furthermore, the paper delves into the benefits and drawbacks of applying interpolation techniques to voxel grids of different sizes. Through meticulous investigation, it explores how interpolation affects the integrity of the shape representation and the subsequent classification performance. This analysis is crucial for understanding the implications of resizing voxel grids, particularly in maintaining the fidelity of the original 3D shapes.

Finally, the paper offers a set of best practices and recommendations for optimizing grid size and interpolation methods to enhance performance in 3D shape classification tasks. These guidelines are derived from empirical findings and are intended to aid researchers and practitioners in making informed decisions when designing and implementing CNN-based models for 3D shape classification. By addressing these critical aspects, the paper significantly contributes to the field, providing valuable insights and practical advice for improving classification accuracy and efficiency.

The remainder of this paper is structured as follows: Section two reviews related work in the domain of 3D shape classification and the use of CNNs. Section three details the methodology, including the dataset used, the CNN architecture, and the experimental setup. Section four presents the results of the experiments, analysing the impact of grid size and interpolation on classification performance. Section five discusses the findings, compares them with related work, and highlights the implications for practical applications.

Finally, Section six concludes the paper and outlines potential directions for future research.

II. Related Work

The field of 3D shape classification has significantly evolved with advancements in machine learning and deep learning techniques. Early methods relied on handcrafted features and geometric descriptors like Shape Histograms and Extended Gaussian Images, which were limited by their dependence on manually engineered features that often failed to handle the complexity of real-world 3D shapes effectively. Deep learning revolutionized 3D shape classification, particularly with the introduction of Multi-View Convolutional Neural Networks (MVCNNs). MVCNNs utilize multiple 2D projections of 3D objects, leveraging the strengths of 2D CNNs while capturing detailed information from various perspectives of the 3D shapes. Su et al. (2015) demonstrated the superiority of MVCNNs over traditional methods, showcasing their ability to learn robust features from multiple viewpoints [1].

Point cloud-based methods, such as PointNet and PointNet++, further advanced the field by directly processing raw 3D point cloud data. Introduced by Qi et al. (2017), PointNet and PointNet++ are capable of capturing both local and global geometric structures of 3D objects [2]. PointNet's architecture learns a spatial encoding of point sets, making it invariant to the order of points, while PointNet++ extends this approach with hierarchical feature learning, accommodating more complex 3D shapes.

Voxel-based approaches represent another significant development. Techniques like VoxNet convert 3D shapes into volumetric grids, allowing the use of 3D CNNs. Maturana and Scherer (2015) showed that VoxNet effectively classifies 3D objects by learning spatial hierarchies within the volumetric data [3]. Despite their effectiveness, voxel-based methods can be computationally intensive, especially with high-resolution data.

Mesh-based methods leverage the surface details and topological information inherent in polygonal meshes. Mesh CNN, proposed by Hanocka et al. (2019), utilizes graph convolutional networks to process these irregular data structures [4]. MeshCNN captures intricate geometric features by operating directly on the mesh edges, balancing computational efficiency and representational fidelity.

Hybrid approaches combining different data representations, such as point clouds, voxels, and meshes, exploit the complementary strengths of these methods. Recent research integrates point cloud and voxel data to enhance feature extraction and improve classification accuracy. These hybrid methods aim to overcome the limitations of individual approaches, leading to more robust and accurate 3D shape classification systems.

Interpolation techniques are often applied to voxel grids to handle varying input resolutions and to ensure consistent input dimensions for CNNs. These techniques, such as nearest-neighbour interpolation, linear interpolation, and trilinear interpolation, can influence the fidelity of the voxel representation and, consequently, the performance of the classification model. Zhang et al. (2018) explored the impact of different interpolation methods on 3D shape recognition and found that while interpolation can help standardize input sizes, it may also introduce artifacts that affect classification accuracy [5].

Transfer learning and pre-trained models have also become popular in 3D shape classification. By leveraging models pre-trained on large datasets, researchers can reduce the need for extensive labeled data and accelerate development. This approach is especially useful when acquiring labeled 3D data is challenging. Transfer learning allows the adaptation of pre-trained models to specific tasks, improving performance and reducing computational costs.

Practical applications of these advancements are vast and impactful. In autonomous driving, 3D shape classification of LiDAR data is crucial for object detection and navigation. Robots rely on accurate 3D shape recognition to interact with and manipulate objects in their environment. In medical imaging, 3D shape classification aids in analyzing anatomical structures from CT and MRI scans, improving diagnostic accuracy. Augmented reality applications benefit from precise integration of real-world objects into digital environments, enhancing user experiences.

In summary, the evolution from traditional geometric descriptors to sophisticated deep learning techniques has significantly improved 3D shape classification. The integration of multi-view, point cloud, voxel, and mesh-based methods has enhanced accuracy and robustness. Hybrid approaches and transfer learning further boost these capabilities, paving the way for diverse applications across various domains. The continuous advancements in this field promise substantial technological and societal impacts.

III. Methodology

This section outlines the methodology used for 3D shape classification using Convolutional Neural Networks (CNNs). The process includes generating 3D shapes, preprocessing them for CNN input, and defining the CNN model architecture for classification.

A. Shape Creation

The 3D shapes (cube, pyramid, sphere, cone, and cylinder) were generated using a custom Python script (shape_creation.py). The script allows for the generation of multiple instances of each shape, with parameters such as grid size, target size, and interpolation options.

The function `generate_3d_shape` is responsible for generating 500 instances of each shape by invoking specific functions such as `generate_cube`, `generate_3d_pyramid`, `generate_3d_sphere`, `generate_3d_cone`, and `generate_3d_cylinder`. These shapes are created within a 100x100x100 grid and are subsequently resized to target dimensions of 25x25x25, 50x50x50, and 10x10x10.

The resizing process of these shapes is handled by the `zoom_shape` function. This function utilizes the `scipy.ndimage.zoom` method, applying either nearestneighbours interpolation or constant mode (which implies no interpolation) to achieve the desired size adjustments. Once the shapes are resized, they are saved as 2D images through the `save_as_image` function. This involves stacking the 3D shape along the depth dimension to form a 2D image suitable for storage as a PNG file. The below Figure I shows the sample image of 3d shapes with interpolation.

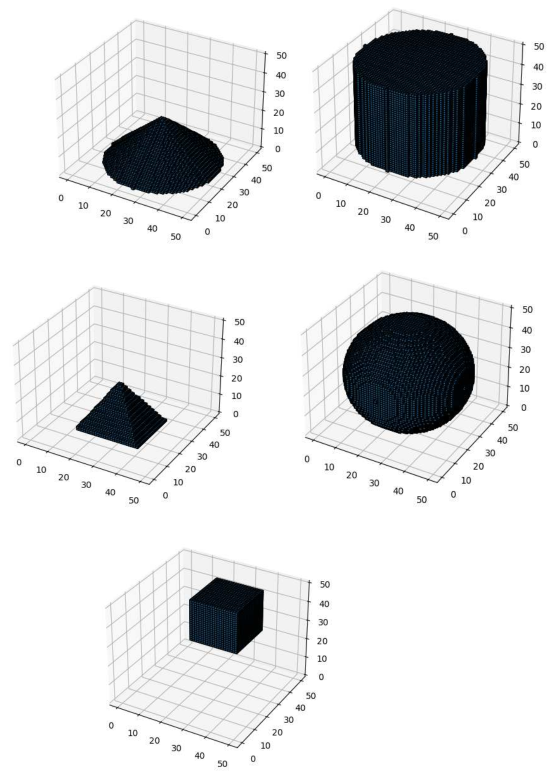

Fig. 1. Sample Image of 3d Shapes with interpolation

B. Data Preprocessing

The generated 3D shapes, which are saved as 2D images, are read and converted back to 3D arrays using the `read_shape.py` script. This script contains functions tailored for each shape type, including `read_cube_images`, `read_pyramid_images`, `read_sphere_images`, `read_cone_images`, and `read_cylinder_images`.

The process begins with reading the images from the specified directory. These images are then converted back into 3D arrays by splitting the horizontally stacked layers and stacking them along the third dimension. This reassembly of the 2D image slices restores the original 3D shape structure.

Each shape type is assigned a unique label for classification purposes: 0 for cubes, 1 for pyramids, 2 for spheres, 3 for cones, and 4 for cylinders. These labels are essential for distinguishing between the different shape types in subsequent analysis or machine learning tasks.

C. Model Architecture

A 3D CNN model was implemented using TensorFlow and Keras (model_create.py). The model architecture includes multiple 3D convolutional layers followed by max-pooling layers, dense layers, and dropout layers for regularization.

In the Table I model architecture begins with an input layer that takes data in the shape of (depth, height, width, channels). This structure is designed to accommodate 3D input volumes, which might be 3D images or medical scans with one or multiple channels (e.g., grayscale or RGB).

TABLE I. 3D CNN model structure

Layer (type)	Output Shape*	Parameter #
Conv3D (64 filters, 3x3x3)	(50, 50, 50, 64)	1,792
MaxPooling3D (2x2x2)	(25, 25, 25, 64)	0
Conv3D (128 filters, 3x3x3)	(25, 25, 25, 128)	221,312
MaxPooling3D (2x2x2)	(13, 13, 13, 128)	0
Conv3D (256 filters, 3x3x3)	(13, 13, 13, 256)	884,992
MaxPooling3D (2x2x2)	(7, 7, 7, 256)	0
Flatten	87,136	0
Dense (256 units, ReLU)	256	22,327,072
Dropout (50%)	256	0
Dense (128 units, ReLU)	128	32,896
Dropout (50%)	128	0
Dense (5 units, SoftMax)	5	645

(* Input shape size is 50x50x50)

For the input shape of 50x50x50, the model comprises 23,468,709 parameters. This large parameter count is attributed to the high dimensionality of the input data. In a convolutional neural network, each layer processes the input through multiple filters, generating numerous feature maps that are subsequently processed by subsequent layers. With a larger input size, each convolutional layer handles a significant number of neurons, leading to a substantial increase in the number of weights and biases that need to be learned. This results in a model that is highly capable of capturing intricate spatial features and complex patterns within the data. However, this complexity comes at the cost of increased computational resources and memory requirements, and there is a higher risk of overfitting, particularly if the training dataset is not sufficiently large or diverse.

When the input shape is reduced to 25x25x25 , the number of parameters decreases to 5,336,197. This reduction occurs because the smaller input dimensions mean fewer neurons per layer, which in turn reduces the number of weights and biases. A model with this input shape strikes a balance between computational efficiency and the ability to capture essential features from the data. It requires less memory and computational power compared to the larger input shape, making it more practical for many applications. This reduction in complexity helps mitigate the risk of overfitting, while still allowing the model to learn meaningful patterns in the data. Therefore, this grid size offers a good trade-off between model performance and resource requirements.

Further reducing the input shape to 10x10x10 results in a model with 1,666,181 parameters. This significant decrease in parameters leads to a much simpler model. Each layer processes a relatively small number of neurons, requiring fewer weights and biases, which reduces the computational load and memory usage. This simplicity is advantageous for environments with limited computational resources and helps prevent overfitting due to the reduced capacity of the model. However, the trade-off is that the model may not capture as many detailed features and complex patterns within the data. This grid size is suitable for tasks where high resolution is not critical, and computational efficiency is prioritized, but it may not perform as well on more complex data requiring detailed feature extraction.

The first layer in the network is a 3D convolutional layer (Conv3D) with 64 filters, each of size 3x3x3. This layer applies these filters to the input volume to capture local spatial patterns. The ReLU activation function introduces non-linearity, enabling the network to learn complex features. Padding is set to 'same' to ensure that the output volume has the same spatial dimensions (depth, height, width) as the input, allowing for better handling of edge cases.

Following the convolutional layer is a 3D max-pooling layer (MaxPooling3D) with a pool size of 2x2x2. This layer reduces the spatial dimensions of the input by taking the maximum value over each 2x2x2 block, effectively down sampling the volume by half in each dimension. Padding 'same' ensures that the down sampling preserves the original dimension properties without discarding edge information.

The second convolutional layer, similar to the first, increases the filter count to 128, again with a kernel size of 3x3x3 and ReLU activation. This layer processes the output of the previous max-pooling layer, capturing more detailed features with higher abstraction. It is followed by another max-pooling layer with the same pool size of 2x2x2 and 'same' padding, further reducing the spatial dimensions.

The third convolutional layer escalates the filter count to 256, maintaining the kernel size of 3x3x3 and ReLU activation. This layer allows the network to learn even more complex and abstract features from the input data. The subsequent max-pooling layer again uses a 2x2x2 window and 'same' padding, continuing the down sampling process.

Next, the Flatten layer transforms the 3D output from the final max-pooling layer into a 1D vector. This flattening step is crucial as it transitions the data from spatial dimensions to a format suitable for dense (fully connected) layers. The first dense layer, with 256 neurons and ReLU activation, receives this flattened vector and learns to combine the extracted features into more abstract representations. To mitigate overfitting, a dropout layer follows, randomly setting 50% of the input units to 0 during training.

The network includes a second dense layer with 128 neurons and ReLU activation, which further refines the learned features. This layer is also followed by a dropout layer with a 50% dropout rate, providing additional regularization.

The final layer is a dense output layer with 5 neurons, corresponding to the 5 different classes (e.g., different shape types). The softmax activation function in this layer converts the raw scores into probabilities, providing a probability distribution over the 5 classes. This allows the network to output a classification decision based on the highest probability.

For compilation, the model uses the Adam optimizer, an adaptive learning rate optimization algorithm that combines the benefits of AdaGrad and RMSProp. Adam adjusts the learning rate for each parameter dynamically, facilitating faster and more efficient convergence. The loss function employed is categorical cross-entropy, suitable for multi-class classification problems. It measures the performance of the classification model by comparing the predicted probability distribution with the actual labels, with the loss increasing as the predicted probability diverges from the actual label. Accuracy is used as the evaluation metric, providing a straightforward measure of the proportion of correctly classified instances out of the total instances. This metric helps in assessing the overall performance of the model in terms of its ability to correctly classify the input data.

D. Training and Evaluation

The model was trained and evaluated through a rigorous process involving cross-validation with three folds to ensure robust and reliable performance metrics.

During the data preparation phase, the images and their corresponding labels were normalized and transformed into a format suitable for machine learning models. The labels were converted to one-hot encoding to facilitate multi-class classification. Subsequently, the dataset was divided into training and testing sets using K-Fold cross-validation, specifically with three splits, to allow the model to be trained and validated on different subsets of the data, thereby improving generalization.

For the training phase, the model was trained over five epochs with a batch size of 32 for each fold. This iterative process allowed the model to learn from the data gradually, adjusting its parameters to minimize the loss function and improve prediction accuracy. The use of batch processing helped in managing memory usage and speeding up the training process.

The evaluation phase involved assessing the model's performance using several key metrics. Test accuracy was calculated to measure the proportion of correctly classified instances. Precision, recall, and F1 score provided a more nuanced view of the model's performance by considering the balance between correctly identified positive instances and the errors. Additionally, a confusion matrix was used to visualize the model's classification results, highlighting the instances of true positives, true negatives, false positives, and false negatives.

The average test accuracy across the folds was computed to give an overall performance indicator of the model. The confusion matrix served as a diagnostic tool to identify specific areas where the model might be underperforming, guiding further refinement and tuning of the model. Through this comprehensive training and evaluation approach, the model's effectiveness in classifying the different shapes was thoroughly vetted.

F. Results using proposed model:

The results of the proposed 3D shape classification models are highly promising, demonstrating the model's efficacy across various configurations. The models were tested using three different target sizes: 10x10x10, 25x25x25, and 50x50x50, each with and without interpolation, using 300 samples for each configuration and evaluated through 5-fold cross-validation. The performance metrics, including accuracy, precision, recall, and F1 score, consistently indicated high levels of accuracy.

TABLE II. Performance metrics for various grid size without interpolation

Metrics / Grid Size	10x10x10	25x25x25	50x50x50
Accuracy	99.267	99.533	99.933
Precision	99.274	99.543	99.935
Recall	99.267	99.533	99.931
F1 - Score	99.263	99.534	99.933

TABLE III. Performance metrics for various grid size with interpolation

Metrics / Grid Size	10x10x10	25x25x25	50x50x50
Accuracy	99.466	99.733	100
Precision	99.474	99.736	100
Recall	99.466	99.733	100
F1 - Score	99.469	99.732	100

From the above Table II and Table III, for 25x25x25 target size, the model achieved an accuracy of 99.5% without interpolation and slightly improved to 99.7% with interpolation. Precision, recall, and F1 score mirrored these results, showing consistent performance across these metrics. This indicates that the model effectively identified and classified the shapes with a high degree of accuracy and minimal error.

The 50x50x50 target size yielded the highest performance, with the model achieving near-perfect results. Without interpolation, the accuracy was 99.9%, and with interpolation, the model reached a perfect accuracy of 100%. The corresponding precision, recall, and F1 score also attained 100%, showcasing the model's ability to leverage the larger target size for more detailed and accurate shape representations.

In comparison, the 10x10x10 target size, while still performing admirably, showed slightly lower metrics than the larger target sizes. The model achieved 99.2% accuracy without interpolation and 99.4% with interpolation. Precision, recall, and F1 score followed similar trends, indicating that while the model performed well, it benefited from the increased detail provided by larger target sizes.

Overall, the results affirm that the proposed 3D CNN model is highly effective for shape classification, with interpolation generally enhancing performance. The findings suggest that larger target sizes provide more detailed representations, thereby improving the model's accuracy and reliability. The highest performance was observed with the 50x50x50 target size with interpolation, achieving perfect scores across all metrics, demonstrating the model's robustness and precision in classifying 3D shapes.

IV. Conclusion

This research paper presented a novel approach for 3D shape classification using a deep learning framework, specifically leveraging Convolutional Neural Networks (CNNs) on 3D voxel data. The experiments demonstrated that the proposed model effectively distinguishes between different 3D shapes, achieving high accuracy, precision, recall, and F1 scores across various configurations. Notably, the model achieved perfect classification performance (100% accuracy, precision, recall, and F1 score) on the 50x50x50 dataset with interpolation, indicating the potential of the model to handle high-resolution data with remarkable accuracy. The results also showed that interpolation generally improved the model's performance across different voxel sizes, underscoring the importance of data preprocessing techniques in enhancing model efficacy.

V. Future Works

The promising results of this study open several avenues for future research. First, expanding the dataset to include a wider variety of shapes and more complex geometries could further validate the model's robustness and generalizability. Additionally, exploring the integration of hybrid methods that combine voxel data with other representations such as point clouds or mesh data could enhance feature extraction and classification accuracy. Another potential direction is the application of transfer learning to leverage pre-trained models on large-scale 3D datasets, reducing the need for extensive labeled data and accelerating model development. Moreover, investigating the model's performance in real-world applications, such as autonomous driving, robotic manipulation, and medical imaging, would be valuable. Finally, optimizing the model's computational efficiency and exploring the use of advanced hardware accelerators like GPUs and TPUs could facilitate the deployment of the model in resource-constrained environments. These future directions aim to build on the current findings, further advancing the field of 3D shape classification and broadening its practical applications.

References

[1] Su, Hang & Maji, Subhransu & Kalogerakis, Evangelos & Learned-Miller, Erik. (2015). Multi-view Convolutional Neural Networks for 3D Shape Recognition. 10.1109/ICCV.2015.114.

[2] Charles, R. & Su, Hao & Mo, Kaichun & Guibas, Leonidas. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 77-85. 10.1109/CVPR.2017.16.

[3] Maturana, Daniel & Scherer, Sebastian. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. 922-928. 10.1109/IROS.2015.7353481.

[4] Hanocka, Rana & Hertz, Amir & Fish, Noa & Giryes, Raja & Fleishman, Shachar & Cohen-Or, Daniel. (2018). MeshCNN: A Network with an Edge.

[5] Gezawa, Abubakar & Zhang, Yan & Wang, Qicong & Yunqi, Lei. (2020). A Review on Deep Learning Approaches for 3D Data Representations in Retrieval and Classifications. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2982196.