Implementation and comparison of PointNets and orientation-boosted Voxel Nets for 3d object recognition

Max Henning Höth
2055977
maxhenning.hoeth@studenti.unipd.it

***Abstract*—This paper presents an implementation and comparative analysis of two state-of-the-art deep learning models, namely PointNets and orientation-boosted Voxel Nets, for the task of 3D object recognition. With the proliferation of 3D data in various domains such as autonomous driving, robotics, and augmented reality, efficient and accurate 3D object recognition has become a critical research area.**

***Index Terms*—Object Classification, Optimisation, Neural Networks, Convolutional Neural Networks, Voxel Nets.**

I. Introduction

3D object recognition has gained significant attention in recent years due to its potential applications in various fields. The task involves recognising objects in 3D scenes based on their shape, size, and orientation. Over the years, deep learning models have emerged as powerful tools for 3D object recognition, offering improved performance and generalisation capabilities compared to traditional handcrafted feature-based approaches.

Among the various deep learning architectures developed for 3D object recognition, PointNets and orientation-boosted Voxel Nets have garnered significant attention due to their promising results. PointNets, proposed by Qi et al. in 2017, revolutionised the field by directly processing raw point cloud data without the need for voxelization or other intermediate representations. PointNets capture local and global geometric information efficiently, enabling robust recognition of 3D objects. On the other hand, orientation-boosted Voxel Nets uses volumetric representations and orientation information to enhance the classification results.

The objective of this paper is to implement and compare PointNets and orientation-boosted Voxel Nets for 3D object recognition. We aim to comprehensively evaluate their performance and analyse their strengths and weaknesses on the Modelnet10 data set.

II. Related Work

In this section, we provide an overview of the existing literature on 3D object recognition, with a focus on deep learning approaches.

Early deep learning approaches for 3D object recognition employed volumetric representations, where 3D shapes were voxelized into regular grids and fed as inputs to 3D CNNs. Wu et al. proposed Volumetric CNNs (VoxNet), which achieved competitive results on the ModelNet dataset. However, these methods suffered from high computational complexity due to the large memory requirements of volumetric representations. To address the limitations of volumetric representations, Qi et al. introduced PointNets, an architecture that directly operates on the point cloud data. PointNets effectively capture local and global geometric information by employing shared multi-layer perceptrons (henceforth referred to as MLP) and max-pooling operations over individual point features. PointNets achieved state-of-the-art performance on various 3D object recognition benchmarks, surpassing volumetric-based methods in terms of both accuracy and efficiency. While PointNets demonstrated remarkable success, they struggled to handle objects with complex geometric structures and were sensitive to variations in object orientation. To address these challenges, Maturana and Scherer proposed orientation-boosted Voxel Nets. This approach combined volumetric representations with orientation-aware features, encoding both shape and orientation information. By exploiting orientation orientation-boosted Voxel Nets achieved improved classification power, especially for objects with more complicated shapes.

III. Methodology

A. Base VoxNet

The architecture is 3d convolutional neural network designed for the 3d object recognition using volumetric representation of objects.

The 3d input is voxelized, which involves dividing the object into 3d grid, where each cell represents a voxel. Either the voxel value is 0 means no part of the object is inside the cell or the voxel value is 1 means a part of the object is in the cell, even if it is very small part.

The voxelized shapes are then fed into 3d convolutional layers. These layers apply a series of learnable filters to the grid, scanning through the shape. They try to identify and capture spatial patterns and features from the voxel grid. After each convolutional layer a batch normalisation layer is applied for the regularisation, robustness and accelerated training. Then we always apply a non-linear activation function, in this case ReLU, to model the complex behaviour of between input and

features. To reduce spatial dimension and extract the important features, max pooling is applied after the non-linear activation function. It down samples the feature maps and maintains the primary features. Afterwards we apply a dropout layer for regularisation and to prevent over fitting.
The output of these layers is then flattend and fed into a linear layer with 128 neurons. Following that another linear layer is added leading to the number of classes of the input data set. Since it is multi-class classification we add a softmax activation to the output in order to receive an ideal result.

B. Orientation-boosted VoxNet

To extend the Base VoxNet network to an Orientation-booosted VoxNet network the spatial orientation of the network must be encoded inside the network. In order to do so the network receives an additional linear layer with an auxiliary parallel task. This layer encodes the spatial orientation. It uses the output of the 128 neuron linear layer as input and counts 105 neurons. Based on the classes and each classes unique number orientation labels, e.g. the class bed receives 12 orientation labels so 12 neurons just for the class bed. Since a table and it's 180° rotated version are the same we don not want every class to have 12 rotation labels so the same rotation labels from Sedaghat *et al.* [1] were used. Also here we apply a softmax activation for ideal result of the multi-class classification.
To take this task into account the loss of of both tasks need to be added together. This happens with the given formula:

$$L = (1 - \gamma) * L_L + \gamma * L_O$$

With $\gamma = 0.5$.
In Sedaghat *et al.* [1] found that the result does not depend on the exact value of γ. The initialisation of the Orientation-boosted VoxNet uses pre-trained weights from the Base VoxNet. The Base VoxNet will be trained with the basic voxelized ModelNet10 dataset till convergence. The weights will be transferred to the Orientation-boosted model and the layer for the parallel task of orientation estimation will be initialised using on the Xavier method. The training will be continued then but with a much lower learning rate and small amounts of rotated samples (probability of rotated samples henceforth referred to as p) within the dataset. In order to find the best learning rate and amount of rotated samples for the network we did a grid search for these parameters and will evaluate their performance later on.

C. PointNet

PointNet is a deep learning architecture designed to use raw sample points as inputs without need of voxelization of the data beforehand. This design is very efficient since the voxelization is memory intensive operation. The architecture is bases on a multi-layer perceptron (henceforth referred to as mlp) with non-linear activations. In this case the MLP consits of convolutional layers, batch normalisation layers and linear layers. In order to extract features and and relations in the point cloud. Afterwards max-pooling is applied to gather local features and create a global features vector representing the information of the input. As usual for classification a feed-forward neural network with dropout layers is used ending with a softmax activation for multi-class labels. The data used for training consits of point clouds. An fixed amount of Points will be randomly chosen and used as input for the network.

D. Implementation Details

For the implementation of both networks the deep learning framework PyTorch is used. The implementation of the Orientation-boosted VoxNet is inspired by the paper from Sedaghat *et al.* [1] and the PointNet implementation is based on [2] which is the PyTorch implementation of Qi *et al.* [3]. For training and evaluation of the networks the ModelNet10 data set is employed. During training of the networks cross-entropy loss is used for the loss of multi-class classification as well as Adam for the optimisation of the networks. In order to evaluate the networks we use numerous metrics to compare their efficiency and performance.

IV. Experimental Setup

A. Dataset

The popular ModelNet dataset is utilised for training and evaluation of the networks. Since it was run on a small, local machine the smaller version ModelNet10 was used. This data set still uses 10 categories with 4899 samples in total and is split into 80:20 for training and validation. In order to voxelize the samples a self made function was used but did not work as expected. So the binvox program by [4] [5] was used to voxelize the object files. It took an extensive amount of time in order to voxelize the whole data set. Furthermore another copy of the ModelNet10 data set was created but this time with randomly rotated samples based on the rotation labels of their class. In order to do the object file was rotated and subsequently voxelized with binvox [4] [5]. Considering the simplification we just applied a rotation around the z-axis.

B. Evaluation

For ideal comparison of efficiency and performance of the networks we will analyse their accuracy per label as well as their ROC curves, average precision, recall and F1 score. Since the time for voxelization was quite extensive the computing and training time will also be analysed.

V. Experimental Results and Discussion

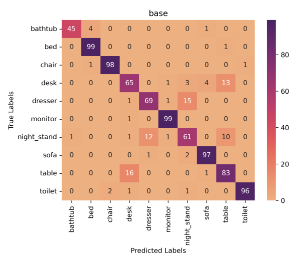
A. Accuracy

First we begin with accuracy of Base VoxNet, Orientation-boosted VoxNet and PointNet. In Table 1 are the best accuracies of each network presented.

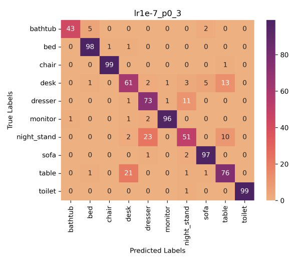
	Accuracy
Base VoxNet	91.63%
Orientatio-boosted VN	91.08%
PointNet	88.88 %

TABLE 1: Timings of Architectures all number are in unit seconds.

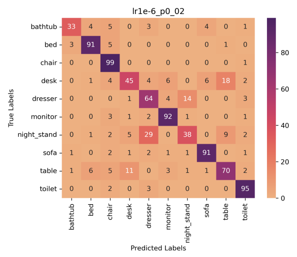
As seen above the accuracy of the Base VoxNet has the best accuracy overall with 91.63 % but close behind is the Orientation-bossted VoxNet with 91.08 %. That means the parameters used were not the optimal for Orientation-boosting but also did not harm the Base network too much. The PointNet on the other hand only has and accuracy if 88.88 % and is thus far behind the VoxNet versions.
The best parameters found for the Orientation-boosted VoxNet were $p = 0.3$ and $lr = 1e - 7$. This can be seen in Figure3 also for two parameter sets we trained them for 600 epochs and received similar results.



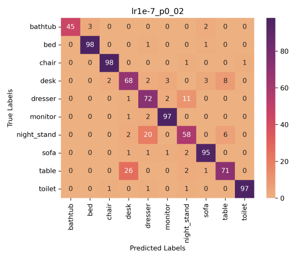
(a) Base VoxNet



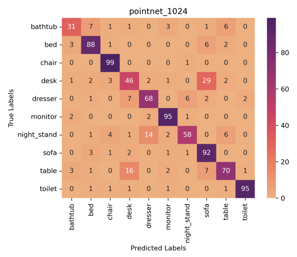
(b) Orion $lr = 1e - 7$, $p = 0.3$



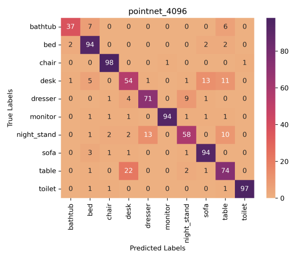
(c) Orion $lr = 1e - 6$, $p = 0.02$



(d) Orion $lr = 1e - 7$, $p = 0.02$



(e) PointNet 1024 points



(f) PointNet 4096 points

Fig. 1: Heatmap of all Networks

However the heat map shows that both networks struggle to identify the same objects and both networks achieve outstanding performance on the same objects. Most of the objects like the desk, dresser or nightstand have similar shape and just differ slightly. The objects with the highest accuracy are the toilet chair or monitor which have a quite unique shape compared to the rest of the objects. The same result we can see in Figure6 where desk, dresser and nightstand perform the worst. The difference PointNet models behave very similar and the number of randomly chosen points just effects the convergence time as seen in Figure 5. Also worth to mention and very significant that the PointNet architecture has a very low false positive rate compared to the VoxNet model in Figure6. Same behaviour is observed in Figure 8 where the avg. precision of the PointNet is very high but recall is below the average voxnet recall. So the general F1 score also falls below the average VoxNet F1 score. This can be caused by the ability of the PointNet to better in decide between details. VoxNets are limited through voxelization which generalises the point cloud and removes fine details which can still be perceived by the PointNet.

B. *Computational Efficiency*

Next we evaluate the computational efficiency and computing time of each network especially the inference time and time per epochs:

	Avg. Inference t	Std Inference t	Avg t per Epoch
Base VoxNet	0.9672	0.04120	3.8707
Orientatio-boosted VN	0.9664	0.0449	4.6615
PointNet	3.9035	0.07940	90.5904

TABLE 2: Timings of Architectures all number are in unit seconds.

As seen above the average inference time of the PointNet is way longer due to the more complex architecture of the network. Also the the time to train an epoch is much higher than the one from the VoxNet which makes is more time consuming. On the other hand is that the PointNet does not need to voxelize the input before which saves quite a big amount of time.

VI. Conclusion and Future Work

The experimental results demonstrate the strengths and limitations of PointNets and Orientation-boosted Voxel Nets for 3D object recognition. Orientation-boosted Voxel Nets offer a competitive alternative for 3D object recognition. They leverage the benefits of voxelized representations and utilise local orientations to enhance recognition accuracy. Their usage of convolutions and 3D convolutions makes them well-suited for processing volumetric data. In conclusion, the choice between PointNets and orientation-boosted Voxel Nets depends on the specific requirements of the application. PointNets are preferable when fine-grained geometric details. Orientation-boosted Voxel Nets can be a viable option when efficiency in volumetric data processing and competitive accuracy are desired. Future research directions could focus on hybrid approaches that combine the strengths of both methods to achieve even higher performance in 3D object recognition tasks.

References

[1] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3d object recognition," in *British Machine Vision Conference (BMVC)*, 2017.

[2] fxia22, “pointnet.pytorch,” 2019.

[3] 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.

[4] P. Min, “binvox.” `http://www.patrickmin.com/binvox` or `https://www.google.com/search?q=binvox`, 2004 - 2019. Accessed: yyyy-mm-dd.

[5] F. S. Nooruddin and G. Turk, “Simplification and repair of polygonal models using volumetric techniques,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, 2003.

APPENDIX

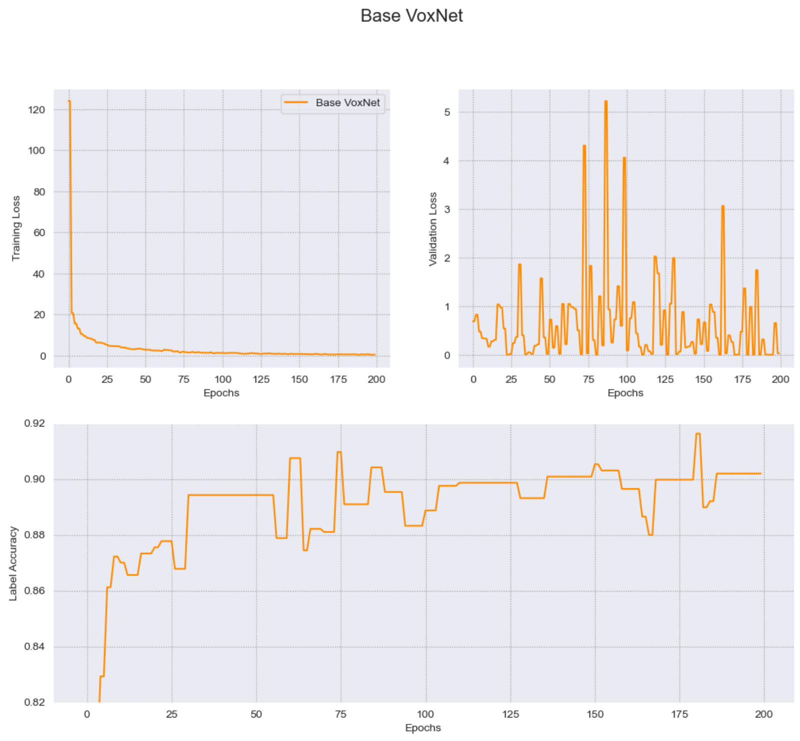


Fig. 2: Base VoxNet

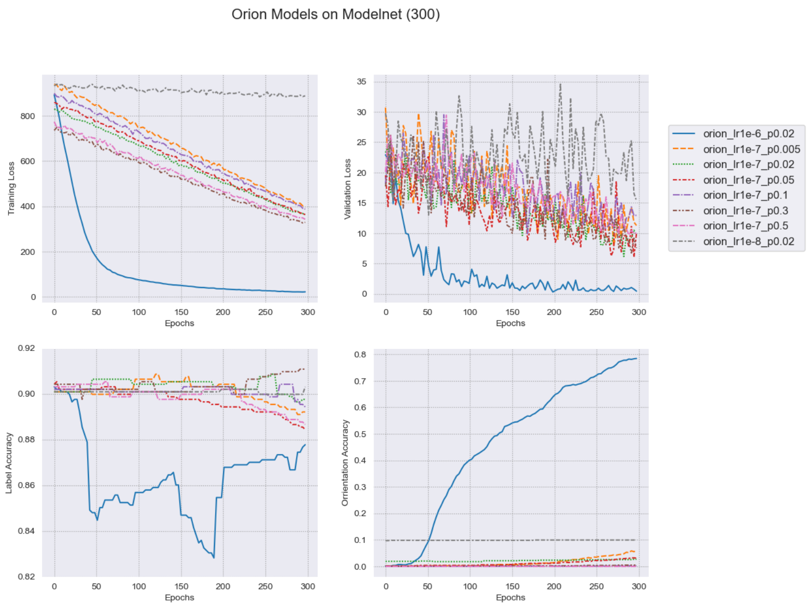


Fig. 3: Orientation-boosted VoxNet with all parameter combinations used in the grid search for 300 Epochs

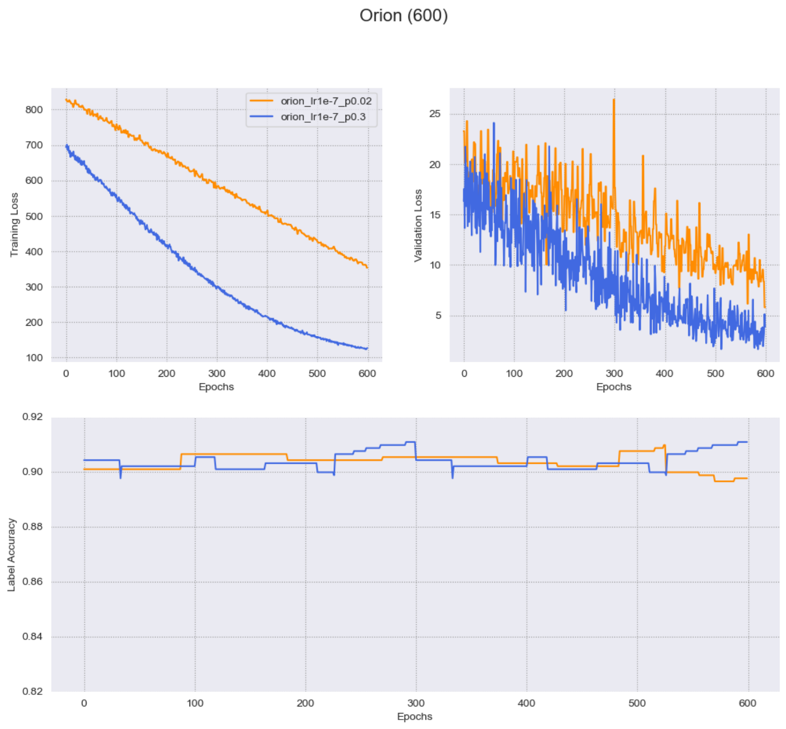


Fig. 4: 2 parameter combinations for 600 epochs

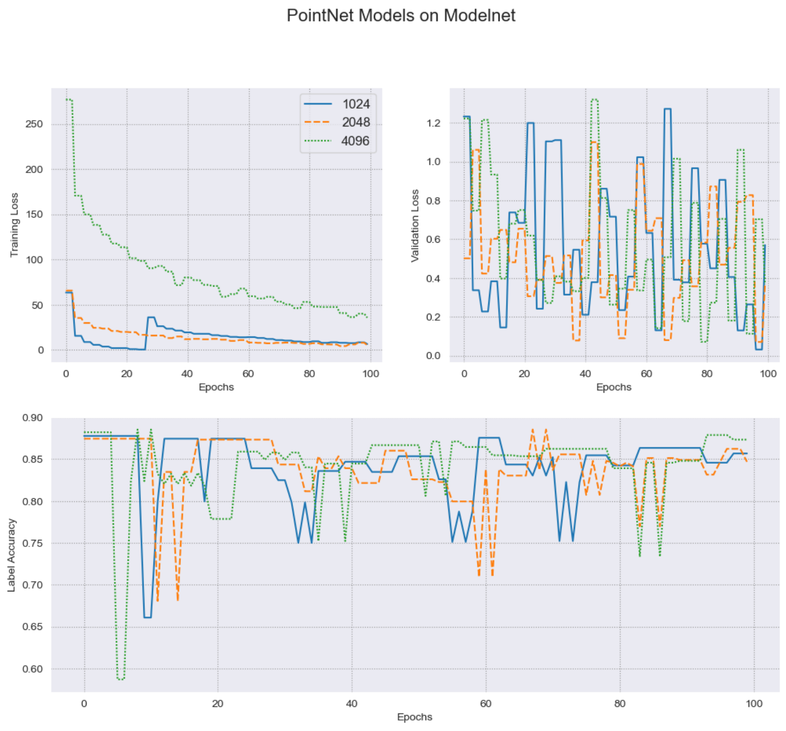
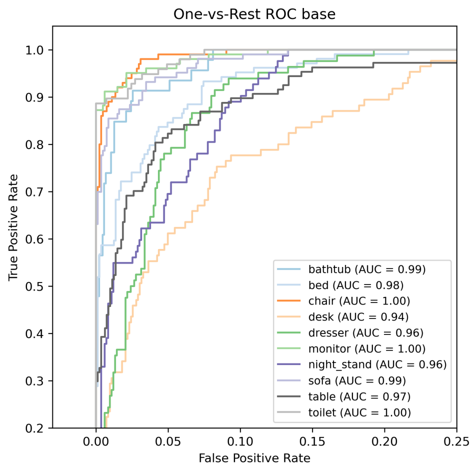
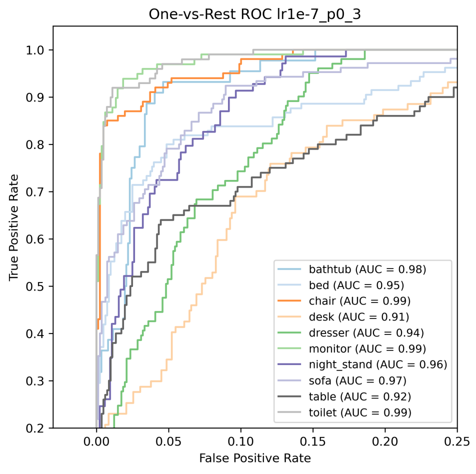


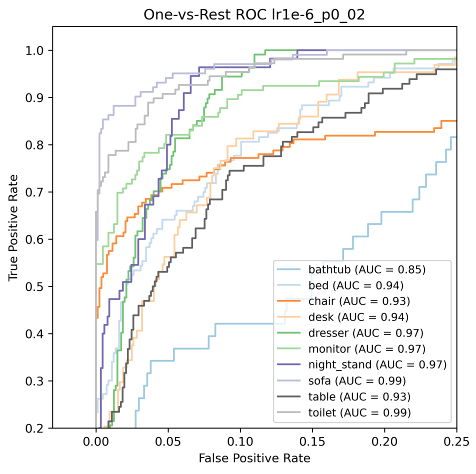
Fig. 5: PointNet



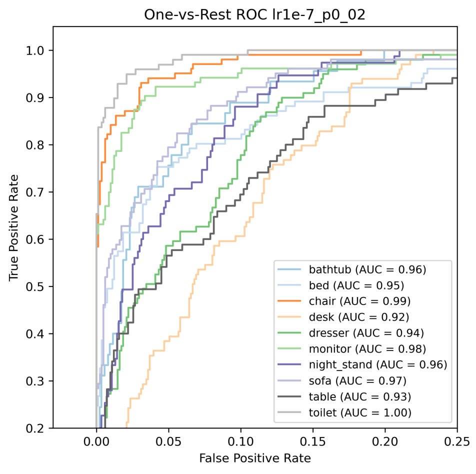
(a) Base VoxNet



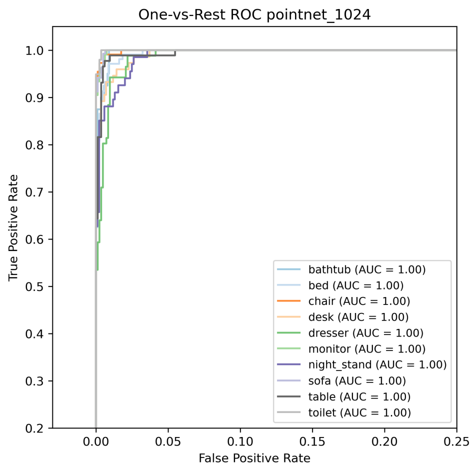
(b) Orion $lr = 1e-7$, $p = 0.3$



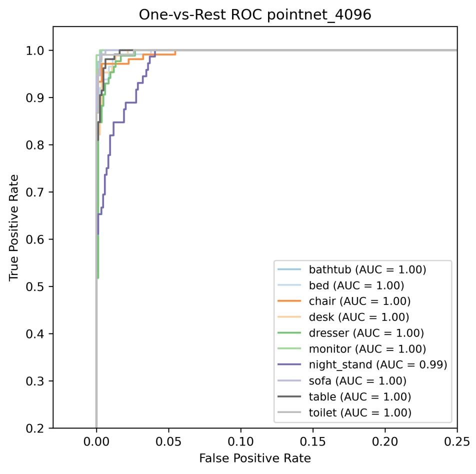
(c) Orion $lr = 1e-6$, $p = 0.02$



(d) Orion $lr = 1e-7$, $p = 0.02$

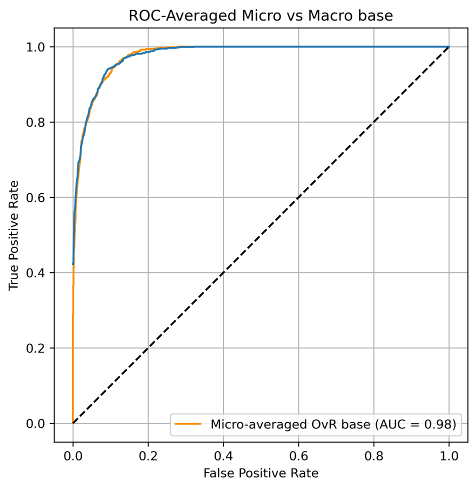


(e) PointNet 1024 points

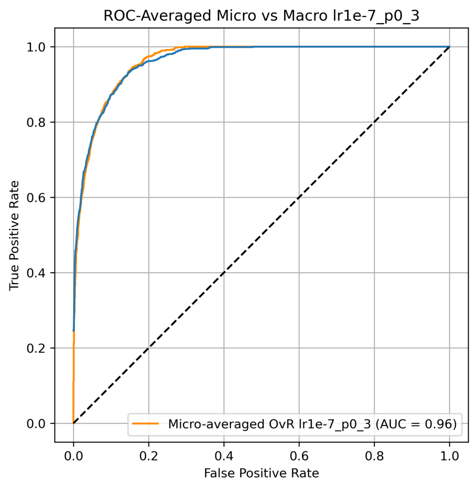


(f) PointNet 4096 points

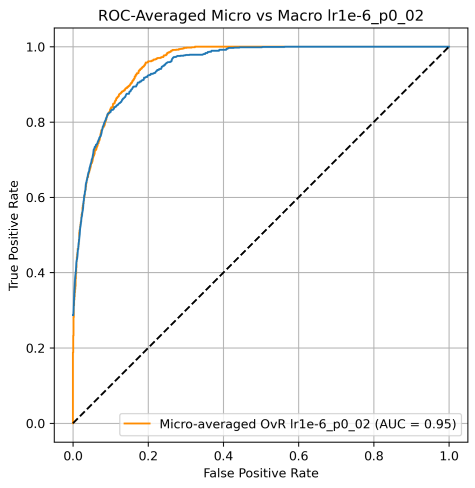
Fig. 6: One-vs-Rest ROC-curve



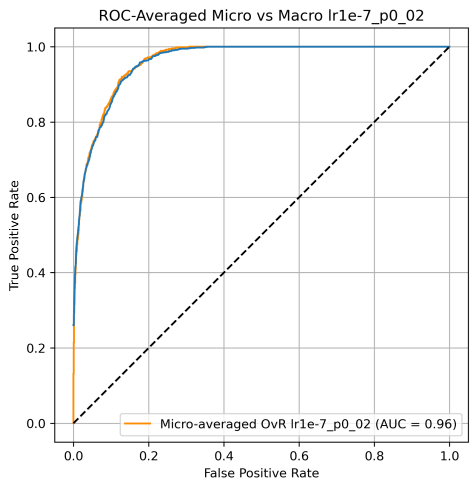
(a) Base VoxNet



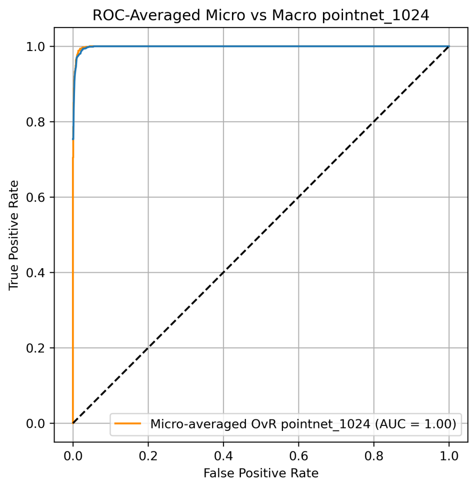
(b) Orion $lr = 1e-7$, $p = 0.3$



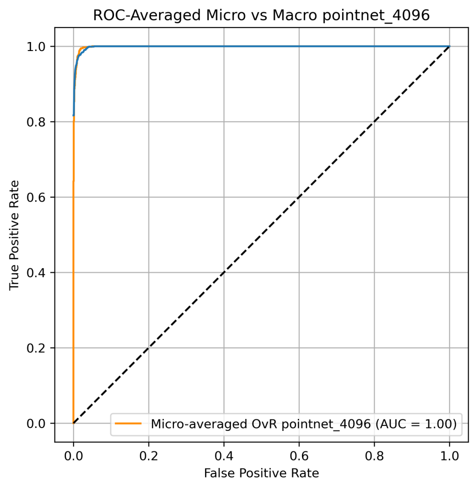
(c) Orion $lr = 1e-6$, $p = 0.02$



(d) Orion $lr = 1e-7$, $p = 0.02$

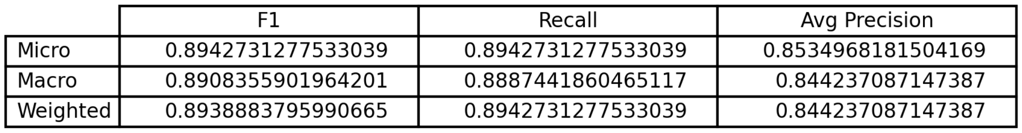


(e) PointNet 1024 points



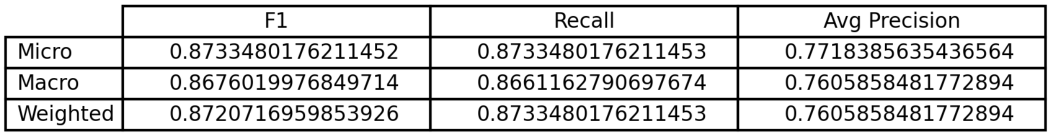
(f) PointNet 4096 points

Fig. 7: ROC-curve Micro vs Macro averaged



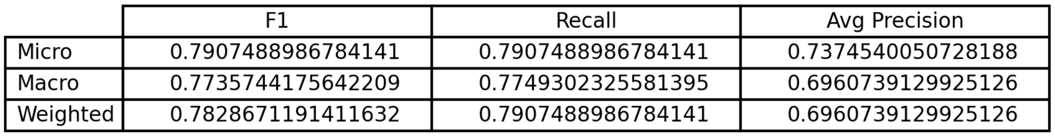
	F1	Recall	Avg Precision
Micro	0.8942731277533039	0.8942731277533039	0.8534968181504169
Macro	0.8908355901964201	0.8887441860465117	0.844237087147387
Weighted	0.8938883795990665	0.8942731277533039	0.844237087147387

(a) Base VoxNet



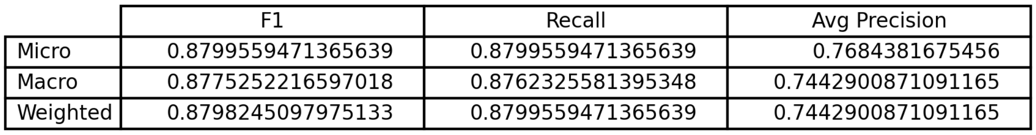
	F1	Recall	Avg Precision
Micro	0.8733480176211452	0.8733480176211453	0.7718385635436564
Macro	0.8676019976849714	0.8661162790697674	0.7605858481772894
Weighted	0.8720716959853926	0.8733480176211453	0.7605858481772894

(b) Orion $lr = 1e-7$, $p = 0.3$



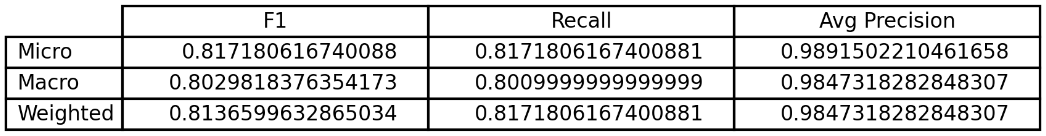
	F1	Recall	Avg Precision
Micro	0.7907488986784141	0.7907488986784141	0.7374540050728188
Macro	0.7735744175642209	0.7749302325581395	0.6960739129925126
Weighted	0.7828671191411632	0.7907488986784141	0.6960739129925126

(c) Orion $lr = 1e-6$, $p = 0.02$



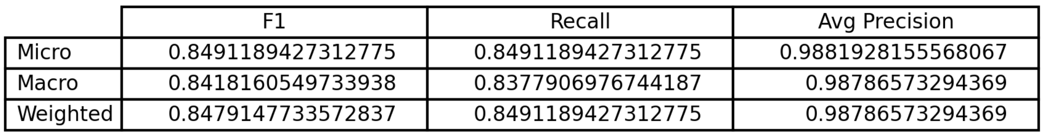
	F1	Recall	Avg Precision
Micro	0.8799559471365639	0.8799559471365639	0.7684381675456
Macro	0.8775252216597018	0.8762325581395348	0.7442900871091165
Weighted	0.8798245097975133	0.8799559471365639	0.7442900871091165

(d) Orion $lr = 1e-7$, $p = 0.02$



	F1	Recall	Avg Precision
Micro	0.817180616740088	0.8171806167400881	0.9891502210461658
Macro	0.8029818376354173	0.8009999999999999	0.9847318282848307
Weighted	0.8136599632865034	0.8171806167400881	0.9847318282848307

(e) PointNet 1024 points



	F1	Recall	Avg Precision
Micro	0.8491189427312775	0.8491189427312775	0.9881928155568067
Macro	0.8418160549733938	0.8377906976744187	0.98786573294369
Weighted	0.8479147733572837	0.8491189427312775	0.98786573294369

(f) PointNet 4096 points

Fig. 8: F1, Recall and Avg Precision