Data-Efficient Transformer-Based 3D Object Detection

Aidana Nurakhmetova[a], Jean Lahoud[b] and Hisham Cholakkal[c]
Department of Computer Vision, Mohamed bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, U.A.E.

Keywords: 3D Point Clouds, Data-Efficient Transformer, 3D Object Detection.

Abstract: Recent 3D detection models rely on Transformer architecture due to its natural ability to abstract global context features. One is the 3DETR network - a pure transformer-based model designed to generate 3D boxes on indoor dataset scans. It is generally known that transformers are data-hungry. However, data collection and annotation in 3D are more challenging than in 2D. Thus, our goal is to study the data-hungriness of the 3DETR-m model and propose a solution for its data efficiency. Our methodology is based on the observation that PointNet++ provides more locally aggregated features that can be useful to support 3DETR-m prediction on small dataset problem. We suggest three methods of backbone fusion that are based on addition (Fusion I), concatenation (Fusion II), and replacement (Fusion III). We utilize pre-trained weights from the Group-free model trained on the SUN RGB-D dataset. The proposed 3DETR-m outperforms the original model in all data proportions (10%, 25%, 50%, 75%, and 100%). We improve 3DETR-m paper results by 1.46% and 2.46% in mAP@25 and mAP@50 on the full dataset. Hence, we believe our research efforts can provide new insights into the data-hungriness issue of 3D transformer detectors and inspire the usage of pre-trained models in 3D as one way towards data efficiency.

1 INTRODUCTION

It has not been long since 3D point cloud analysis became one of the crucial areas of research. 3D object detection on point clouds is concerned with generating bounding boxes tightly surrounding the objects in a 3D scene, thus, it recognizes and localizes objects simultaneously. 3D object detection has many applications in real-world such as autonomous driving, robotics, healthcare, and augmented reality. 3D point cloud object detection is a challenging computer vision task, since 3D point cloud data is *sparse, orderless* and *irregular*. Due to such properties, various methods of treating the 3D point cloud data were born in the research community. Hence, there are three key reasons for the motivation behind the deeper study of 3D point clouds among computer vision researchers:

- Emergence of readily available 3D scanning sensors
- Presence of richly labeled datasets
- Development of 3D deep learning methods

Various large-scale 3D datasets were accumulated and annotated due to the emergence of the latest point processing technologies such as LiDAR and RGB-D sensors that can effectively scan the 3D environment.

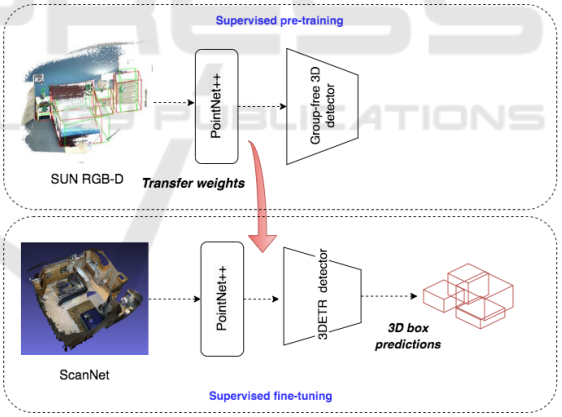


Figure 1: Supervised pre-training and fine-tuning of 3DETR-m model.

Many works in 3D vision utilize deep learning for 3D shape classification (Su et al., 2015) (Qi et al., 2016), 3D object detection (Qi et al., 2017a) (Chen et al., 2019) and tracking (Giancola et al., 2019) (Qi et al., 2020), 3D point cloud segmentation (Landrieu and Simonovsky, 2017) (Graham et al., 2017).

Previous deep learning methods in 3D relied on convolutional neural networks (ConvNet) and multi-layer perceptrons (MLPs). Inspired by the success

[a] https://orcid.org/0000-0001-6308-9397
[b] https://orcid.org/0000-0003-0315-6484
[c] https://orcid.org/0000-0002-8230-9065

Nurakhmetova, A., Lahoud, J. and Cholakkal, H.
Data-Efficient Transformer-Based 3D Object Detection.
DOI: 10.5220/0011673200003417
In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 4: VISAPP*, pages 615-623
ISBN: 978-989-758-634-7; ISSN: 2184-4321

of ConvNet, which is due to the ability to capture finer features at local levels, PointNet++ (Qi et al., 2017b) was designed to hierarchically process raw point cloud input to extract features at multiple scales. Many object detection models are built upon this backbone which has proven to be effective in learning 3D representations of the point cloud input.

Recently, Transformers has made a breakthrough in 3D scene understanding. Transformer architecture (Vaswani et al., 2017) initially was designed for Natural Language Processing (NLP) tasks, which inherently suits sparse and orderless input. Most importantly, it can capture long-range dependencies vital for perceiving contextual patterns in the scenes. Due to such amenities, Transformers and the self-attention operation offered more accurate detections on point clouds. Nevertheless, unlike non-transformer models, pure transformer-based ones require more data samples to train the network.

This paper aims to study the robustness of 3D detection models towards limited dataset condition. Depending on various reasons, data might not always be available, and the robust model trained on fewer data can save resources such as time and computational overhead on the network. In this work, we compare four different models with four different architectures with the purpose of preliminary study and analysis, which are: 1) Group-free 3D (Liu et al., 2021) (hybrid model with transformer decoder and Pointnet++ backbone); 2) MLCVNet (Xie et al., 2020a) (point-based model with a combination of MLPs and attention); 3) 3DETR (Misra et al., 2021) (pure transformer with the encoder-decoder); 4) 3DETR-m (Misra et al., 2021) is another version of 3DETR which uses 2 set abstraction layers from PointNet++, where a masked encoder is used for local feature aggregation. All models were trained on the ScanNetv2 benchmark dataset under four settings: 10%, 25%, 50%, and 75% of the original dataset.

The contributions from our research efforts are outlined below:

- We design a novel architecture by integrating the PointNet++ backbone into 3DETR-m via simple fusion methods such as addition, concatenation, and replacement.
- We use SUN RBG-D pre-trained model of Group-free 3D network to initialize backbone with more meaningful weights, thus, yielding higher performance in all dataset proportions (10%, 25%, 50%, and 75%). By improving over the baseline 3DETR-m models, we make purely transformer-based architecture more data efficient. We improve 3DETR-m paper results by **1.46%** and **2.46%** in mAP@25 and mAP@50 on the full dataset.
- We are the first to explore the data-hungriness of a transformer-based 3D object detector to the best of our knowledge.

2 RELATED WORKS

Point-Based Approaches. PointRCNN (Shi et al., 2018) is a two-stage 3D object detector similar to the FasterRCNN method in 2D object detection, which exploits the PointNet backbone to extract useful features and generate 3D object proposals. KP-Conv (Thomas et al., 2019) is a point convolution-based approach where any number of kernels can be used, making it a more flexible technique than grid convolutions. MLCVNet (Xie et al., 2020a) is a multi-level context VoteNet (Qi et al., 2019) extracting features at multiple levels such as at patch, object and global levels.

Projection/Voxel-Based Approaches. Previous works focused on the traditional convolution approach after the irregular shape of 3D input data was transformed into 2D/3D grids. These methods PIXOR (Yang et al., 2019), AVOD (Ku et al., 2017) project point cloud to 2D planes/bird's eye view (BEV) and convert them into 2D grids to apply 2D ConvNets, which can learn features and output 3D bounding boxes. A number of methods VoxNet (Maturana and Scherer, 2015), MV3D (Chen et al., 2016), FCAF3D (Rukhovich et al., 2021) depend on voxelization, which first maps point cloud input to a volumetric 3D grid (voxels) so that 3D ConvNets can be utilized to compute features and predict 3D bounding boxes.

Transformer Based Approaches. Authors of Point Cloud Transformer (Guo et al., 2021) design a solely transformer-based network for learning global context through a self-attention mechanism. Pointformer (Pan et al., 2020) paper is a pure transformer U-net like architecture that relies on the ability of Transformers to capture long-range dependencies. It is built with three different transformers: Local Transformer, Local-Global Transformer, and a Global Transformer. The authors of (Zhao et al., 2020) propose Point Transformer layer to effectively process raw point cloud igniting a permutation-invariant property.

2.1 Training with Limited Data

In 3D deep learning, there are few works that have used pre-training for 3D point cloud data. PointContrast (Xie et al., 2020b) framework enables unsupervised pre-training for the first time in high-level 3D

scene understanding that provided promising results in segmentation and detection tasks with six benchmark indoor and outdoor datasets. *Yamada et al.* (Yamada et al., 2022) suggest using pre-training to initialize weights from a model trained on a synthetic dataset and fine-tune on a smaller dataset with real scans. The authors developed a new fractal geometric database called PC-FractalDB, on which source networks were pre-trained. In the 2D domain, a recent work (Wang et al., 2022) focused on pushing the performance of DETR (Carion et al., 2020) and Deformable-DETR (Zhu et al., 2020) and could beat original models with much less data on subsampled COCO 2017 and small-sized CityScapes datasets. For COCO 2017, smaller training data was created by sampling 0.1, 0.05, 0.02, and 0.01 from the full training examples, whereas the evaluation set is kept the same.

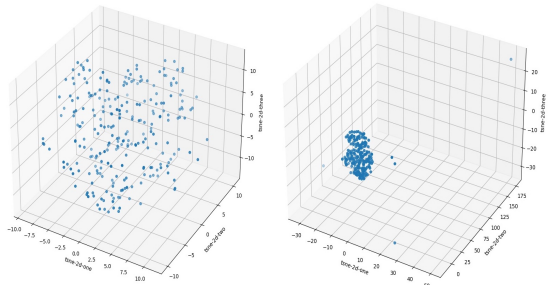
Summary. Current 3D object detection transformers demonstrate competitive performance and simple design compared to point-based or grid-based methods. Nevertheless, it is not clear whether transformer models can keep high accuracy with smaller training data. The recent study in 2D (Wang et al., 2022) shows that with smaller training samples, detection transformers suffer from performance drop, which proves that transformers are data-hungry networks. Hence, motivated by an existing gap, we aim to study the data-hungriness of 3D detection transformers by evaluating the 3DETR-m model on smaller sets of ScanNet dataset and present a novel 3DETR design that achieves better performance than the baseline scores. As object detection on point clouds is still in its infancy, the existing 3D detectors do not utilize any transfer learning. Hence, we are the first to use an off-the-shelf pre-trained model on the detection transformer network.

3 APPROACH

3.1 Leveraging 3DETR-m Model with PointNet++ Backbone

First we will provide an overview of 3DETR (Misra et al., 2021) model architecture focusing on the *Masked Encoder* and *Decoder* of the transformer.

Masked Encoder. The masked encoder of 3DETR-m consists of three layers of self-attention and MLP. The attention matrix of size $(N' \times' N)$ in each layer is multiplied by a binary mask matrix M of the same shape. Any M_{i_j} entry of M is set to 1 if i and j points are located within a radius r distance with each other. 3DETR-m uses 2 Set Abstraction layers from PointNet++. The first set abstraction layer is used to downsample initial scene points to 2048 points, and later to 1024 points by the second set abstraction layer with a radius of 0.4. The second stage of subsampling and set aggregation is named interim downsampling that is used for the masking stage and is applied to the encoder self-attention output.



(a) PointNet++ (b) 3DETR-m encoder

Figure 2: t-SNE visualization of feature distribution.

Decoder. The transformer decoder receives an input of $(N' \times d)$ from the encoder and $(B \times 256)$ query embeddings, where $B = 256$ query points sampled by fps from N' points. Fourier positional embedding is applied to include positional information, as the decoder does not have knowledge about initial positions. The decoder is used to produce final $(B \times d)$ boxes with $d = 256$ box features. There are eight decoder layers and four heads in each MHA. Cross-attention in the decoder computes the relation between query embeddings and encoder features, while self-attention in the decoder computes the relation between query embeddings.

3.2 Analysing Feature Distribution

In order to understand the feature scales, a dimensionality reduction technique *t-distributed stochastic neighbor embedding* (t-SNE) was used. The 256-dimensional feature vector of the PointNet++ features and 3DETR-m encoder features were projected to dimension 50 using truncatedSVD for sparse matrices. Using t-SNE plots to visualize feature dimensions helped better understand the feature scales and data distribution. Differences in data distribution mean some normalization techniques need to be used prior to fusion. The plots in Figure 2 showcase the features reduced to 3D space for one of the batch data samples.

Fusion I: Addition. To mitigate the data-hungriness of 3DETR, we used backbone features from PointNet++. We added the backbone features with the encoder features, where encoder features were normalized between the [0, 1] range. A detailed scheme of the modification can be seen in Figure 3. For the nor-

malization, MinMaxScaler was applied according to equations below:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

$$x_{scaled} = x_{std} * (max - min) + min \quad (2)$$

where x_{min} is a minimum value and x_{max} is a maximum value in feature space, while x is the feature of current point in a cluster. The range to which normalization has been applied is given by *min*, *max* variables. In the below equation, the fusion method is shown with a simple addition operation, where $F(x,y)$ is the new fused features, x_{scaled} is the normalized encoder features, and y is the backbone features.

$$F(x,y) = x_{scaled} + y \quad (3)$$

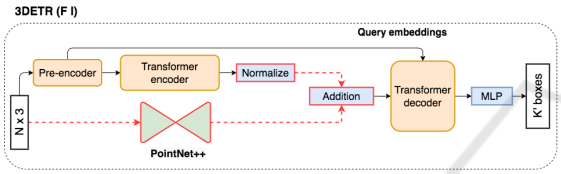


Figure 3: Architecture of proposed model 3DETR-m-F1 with backbone fusion on 3DETR-m (Fusion I). The red lines represent our modifications on the original 3DETR.

Fusion II: Concatenation. We utilized concatenation method to preserve original features from the backbone. The architecture sketch can be seen in Figure 4.

$$F(x,y) = ReLU(\gamma(Concat(x_{scaled}, y))) \quad (4)$$

where γ is MLP network with hidden dimension of 128. $Concat(x_{scaled}, y)$ is performed across feature axis, giving in total 512 output features, which is projected by the MLP back to 256 dimension. ReLU nonlinearity is used as an activation function. The encoder features were normalized between [0, 1] based on Equation 2.

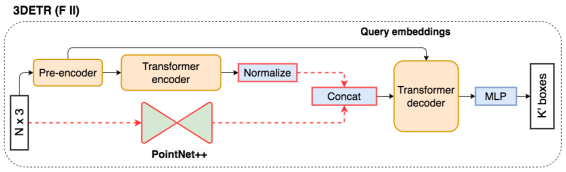


Figure 4: Architecture of proposed model 3DETR-m-F2 with backbone fusion on 3DETR-m (Fusion II). The red lines represent our modifications on the original 3DETR.

Fusion III: Replacement. In this fusion technique, the pre-encoder is fully replaced by the backbone or, in other words, extended by two more set abstraction layers followed by two feature propagation layers as in PointNet++ (see Figure 5).

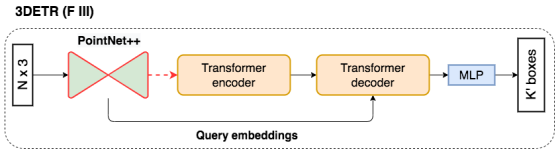


Figure 5: Architecture of proposed model 3DETR-m-F3 with backbone fusion on 3DETR (Fusion III). The red lines represent our modifications on the original 3DETR.

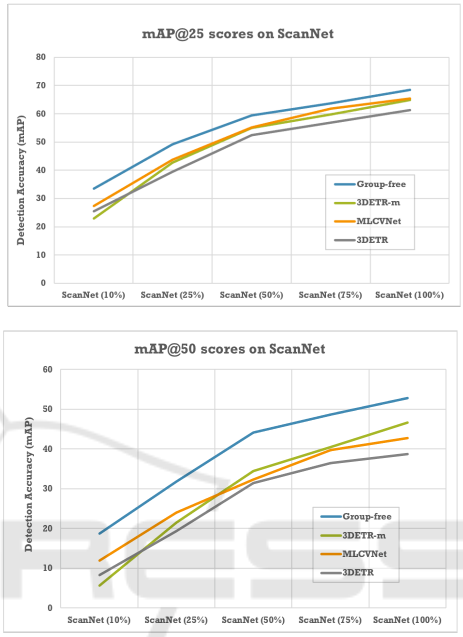


Figure 6: Comparison of Group-free, 3DETR-m, MLCVNet and 3DETR vanilla models.

4 DATA-EFFICIENT 3DETR-m WITH SUPERVISED PRE-TRAINING

In this section we explain how we improved 3DETR-m by utilizing the transfer learning technique. Group-free model (Liu et al., 2021) was selected as a source pre-training model as it is partially transformer-based architecture and it has competitive performance. Hence, in order to support the fused backbone, we initialized its weights with SUN RBG-D pre-trained Group-free model weights from the same backbone while eliminating the rest of the architecture layers. As we are interested in increasing the performance of 3DETR-m in all data proportions, we strove to take advantage of the pre-trained backbone from the Group-free model (GF3D). For the baseline models in all data proportions, we use 10 decoder layers instead of original 8 layers and refer to it as 3DETR-m-10L. Since the features from a source

dataset are different from the target dataset, thus, it may need additional layers to be refined.

As depicted in Figure 1, the weights of a pre-trained GF3D model are transferred to 3DETR-m to initialize the backbone with meaningful values rather than just random numbers. Therefore, only the PointNet++ feature weights from GF3D are implanted into the same backbone in 3DETR-m. GF3D is an end-to-end network that optimizes backbone features based on downstream detection task, which perfectly suits our goal - to make maximum use of the PointNet++ backbone. In addition, SUN RBG-D is also an indoor dataset that can, at the same time, supply diverse features together with features extracted from ScanNet.

SUN RBG-D pre-trained GF3D model with 6 layers, 256 object candidates, and width 1 of the backbone was downloaded from an official GitHub repository. We created an instance of the GF3D model in 3DETR code to have access to a state dictionary and model parameters. After the pre-trained model weight was loaded, a new state dictionary was created, providing all source model parameters.

5 EXPERIMENTS

Dataset and Metrics. ScanNetv2 (Dai et al., 2017) is an indoor dataset built from richly annotated 3D reconstructions comprising 1513 indoor scenes and 18 object categories. 3D bounding box annotations, as well as per-point instance semantic labels, are provided. The standard mean Average Precision (mAP) protocol is followed under IoU thresholds of 0.25 and 0.5, eliminating the oriented bounding boxes. The dataset was split into a training set consisting of 1201 scenes and 312 test set for validation.

Source Dataset. SUN RBG-D (Song et al., 2015) is a single-view RBG-D dataset of indoor environments with 3D bounding box annotations. It contains 10,335 RGB-D image frames labeled with amodal and oriented bounding boxes with 37 class categories. Abiding by a standard evaluation protocol, only 10 classes are used during training and validation. The training set has 5285 frames, and the test set has 5050 frames.

Training Details. The training configurations for the experimental models were kept as provided in the original papers and GitHub repositories. The baseline models were all trained on cluster GPUs provided by the university. MLCVNet and 3DETR used a single GPU, while GF3D utilized 4 GPU resources. The code is written in the PyTorch framework. Both 3DETR and GF3D use AdamW optimizer, while MLCVNet uses Adam. Initial learning rates for 3DETR, GF3D and MLCVNet are 0.0005, 0.006 and 0.01, respectively. All three models are trained end-to-end with a batch size of 8. We also trained the 3DETR-m version of vanilla 3DETR, which uses 2 set abstraction layers and a masked encoder with interim downsampling. 3DETR-m has the same settings as 3DETR except for a few differences, such as an encoder dropout of 0.3 is used and masking is enabled. The models trained under limited data condition used the same configurations as the baseline ones with 100% of the dataset.

We run the experiments on 10%, 25%, 50%, and 75% subsets of ScanNet dataset. All the pre-trained 3DETR-m were trained for 1080 epochs as the original model. It is important to note that GF3D trained on SUN RBG-D uses a width of 1 for the backbone. Therefore, we modified the PointNet++ width 2 in 3DETR-m models to width 1 for consistency. Base learning rate of 0.0005 is used in all experiments. The models were trained using a batch size of 8 on a single NVIDIA GPU.

6 RESULTS AND DISCUSSION

In order to better understand behavior of 3D models with limited data, we selected four detection models which are Group-free (Liu et al., 2021), MLCVNet (Xie et al., 2020a), vanilla 3DETR (Misra et al., 2021) and 3DETR-m (Misra et al., 2021). The reason for choosing Group-free and MLCVNet along with 3DETR models is because both Group-free and MLCVNet rely on PointNet++ backbone and are not full transformer networks as 3DETR. Our study explores the data-hungriness of transformer-based model 3DETR; therefore, we consider it crucial to compare against non-transformer architectures. We evaluated our models on ScanNetV2 3D object detection benchmark dataset.

The overall trend for baseline models is reasonable (see Figure 6): with more data, the models perform better, and the mAP range between the models is preserved for all settings, except 10% and 25% ScanNet training on 3DETR-m. Compared with 3DETR vanilla, which has lower mAP scores on full dataset and other proportions than 3DETR-m, at 10% and 25% ScanNet 3DETR-m fails to reach the same performance trend. This may imply that 3DETR-m is more data-hungry than 3DETR. Noticing this behavior, we wondered whether it is possible to improve over 3DETR-m (at 10% and 25%). Hence, as a first step, it was logical to try utilizing PointNet++ backbone features along with the encoder features since both Group-free and MLCVNet, which possess higher scores than 3DETR, rely on the backbone

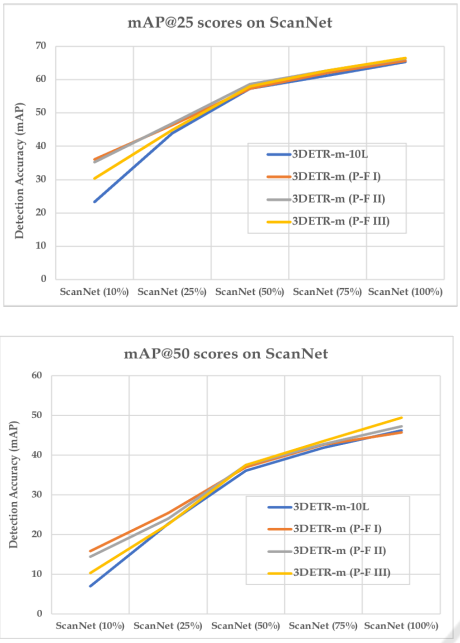


Figure 7: 3DETR-m (P-Fusion I, II, III) performance comparison on ScanNet.

for feature generation. In contrast, 3DETR is a pure transformer-based network only using a single set abstraction layer from the PointNet++ backbone for the purpose of down-scaling.

Models with all fusion techniques 3DETR-m (P-Fusion I, II and III) have increased detection accuracy results compared to 3DETR-m-10L baseline in all data proportions (10%, 25%, 50%, and 75%), as depicted in Figure 7 and Table 1. Note that the 3DETR-m-10L baseline is the exact 3DETR-m model with 10 decoder layers. On 10% ScanNetV2, 3DETR-m (P-Fusion I) with SUN RGB-D pre-trained weights significantly improves on top of our 3DETR-m-10L baseline by 12.74% accuracy gain in mAP@25 and by 8.88% gain in mAP@50. While 3DETR-m (P-Fusion II) outperform the baseline scores by 11.93% mAP@25 and 6.97% mAP@50. It is worth mentioning that mAP@25 scores of 3DETR-m (P-Fusion I) and 3DETR-m (P-Fusion II) are higher than the Group-free and MLCVNet baseline scores on 10% ScanNetV2 (see Figure 8 and 9). On the other hand, 3DETR-m (P-Fusion III) has higher mAP@25 scores than the MLCVNet. This proves that our proposed models can perform on par with the non-transformer models on the small-scale dataset.

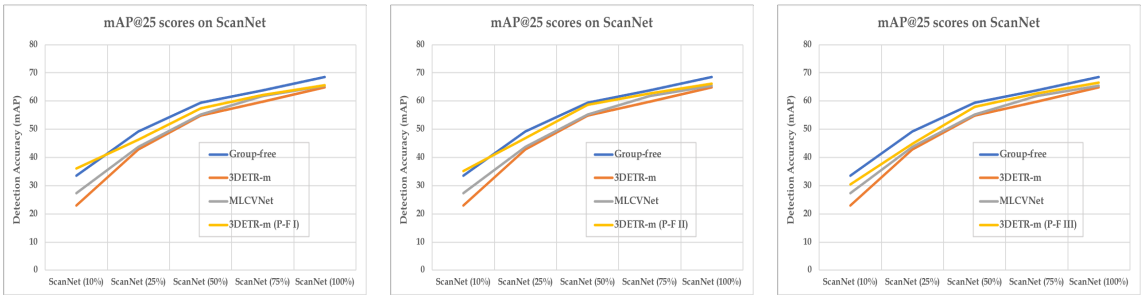
While for the 100% ScanNetV2, Fusion II (Concatenation) and Fusion III demonstrated the best mAP outcomes. As reported in Table 1, 3DETR-m (P-Fusion II) performance on the full dataset yielded high results in both mAPs reaching 66.24% and 47.25%, while 3DETR paper reports 65.0% and 47.0%. Table 3 illustrates per-class outputs in mAP@25: 3DETR-m (P-Fusion II) increases the accuracy of 13 object categories, except the chair, bookshelf, toilet, sink, and bath. Interestingly, the first two and last three classes are all related objects, meaning the model understands the relationship between the objects. The model has significant gains in the detection of some objects. For example, for the challenging picture object, it provides 0.85% gain, for the window 7.45%, and the counter 7.53%. 3DETR-m (P-Fusion III) has the highest mAP@25 and mAP@50 results reaching 66.46% and 49.46% which provides with 1.46% and 2.46% gains as compared to original paper 3DETR-m scores. Although 3DETR-m (P-Fusion III) has the best overall score, class-wise it improves the performance of only 9 object categories (see Table 3). Similarly, 3DETR-m (P-Fusion I) improves the performance over 10 classes and overall it has the lowest mAP scores. Table 2 compares our modified models with the rest of the state-of-the-art model performances.

Regarding the qualitative outputs (Figure 10), 3DETR-m-10L baseline misses the window and door objects, while our 3DETR-m (P-Fusion I) successfully detected the door and 3DETR-m (P-Fusion II) correctly recognized the window. This proves that our proposed models are more efficient at capturing the global context. Among three fusion techniques (P-F I, P-F II and P-F III), 3DETR-m (P-Fusion II) does not capture redundant object, whereas 3DETR-m (P-Fusion I) and 3DETR-m (P-Fusion III) have false positives for another window object. In addition, 3DETR-m (P-Fusion III) has misses too. Hence, 3DETR-m (P-Fusion II) is the best model.

7 CONCLUSIONS

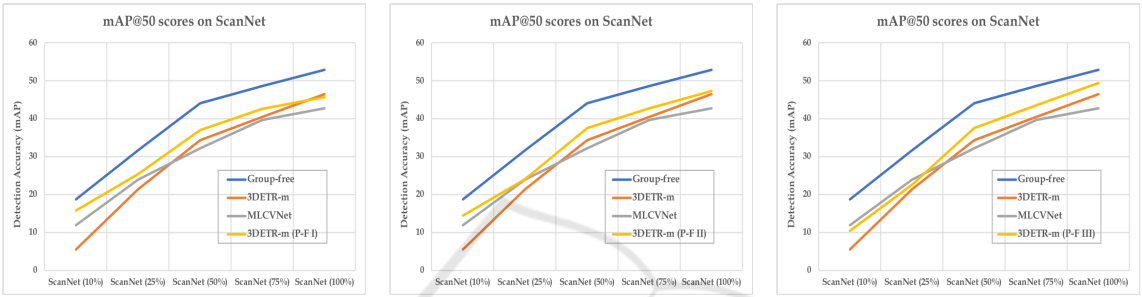
To summarize, this work focused on exploring the robustness of 3D detector models toward small dataset problem. Seeing the gap between PointNet++ based models (Group-free, MLCVNet) and pure transformer models (3DETR, 3DETR-m), we aimed to push the potential of 3DETR further by leveraging the architecture with the backbone features. Although transformer self-attention is good at capturing long-range dependencies in the scene, it may lack in forming a local shape geometry from grouped cluster points. Hence, we showed that the performance could be increased by fusing locally aggregated features from PointNet++ to 3DETR.

Future Research. As our target was 3D object detection, our research efforts can be extended to other



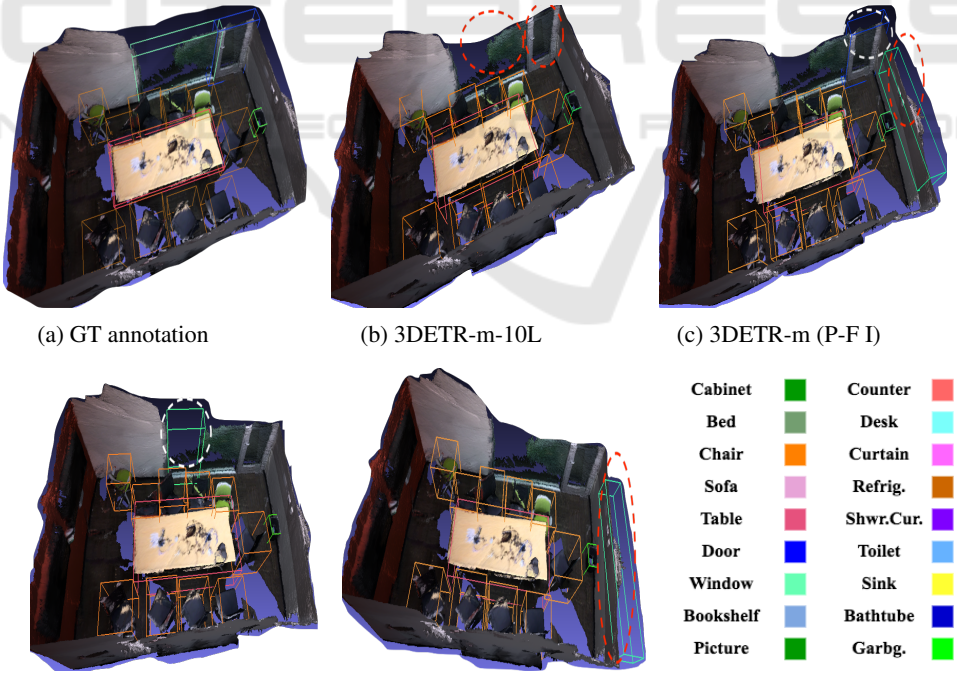
(a) 3DETR-m (P-Fusion I) (b) 3DETR-m (P-Fusion II) (c) 3DETR-m (P-Fusion III)

Figure 8: Performance comparison with state-of-the-art on ScanNet at mAP@25.



(a) 3DETR-m (P-Fusion I) (b) 3DETR-m (P-Fusion II) (c) 3DETR-m (P-Fusion III)

Figure 9: Performance comparison with state-of-the-art on ScanNet at mAP50.



(a) GT annotation (b) 3DETR-m-10L (c) 3DETR-m (P-F I)

(d) 3DETR-m (P-F II) (e) 3DETR-m (P-F III) (f) Color reference

Figure 10: Qualitative results comparison between (b) 3DETR-m-10L baseline and our proposed models (c), (d), and (e). P-F I, P-F II, and P-F III stand for P-Fusion I, II, and III, which are the pre-trained models with the Fusion (I, II, or III) technique. The objects detected correctly are labeled with a white dashed circle, whereas wrong or missed object detections are shown with a red dashed circle. The bounding box colors used in visualization images are given in (f). Ground truth (GT) annotation is depicted in (a). 3DETR-m (PF-1) detects a door object and 3DETR-m (PF-2) detects a window object, while the baseline fails to detect either of these.

Table 1: Mean Average Precision (mAP) results on ScanNetv2 validation set with pre-trained weights. Numbers outside brackets represent the best epoch results, while those inside are the last epoch results. P-Fusion implies a pre-trained model with the Fusion (I, II, or III) technique. Our proposed models have improvements (highlighted in bold) in all percentage sets (10%, 25%, 50%, 75%, and 100%) as compared to 3DETR-m baselines.

Architecture	Set	Backbone	mAP@0.25	mAP@0.5
3DETR-m (baseline)	10%	-	23.0 (22.99)	5.58 (6.24)
3DETR-m-10L (baseline)	10%	-	23.31 (22.21)	6.99 (5.66)
3DETR-m (P-Fusion I)	10%	PointNet++	**36.05 (35.03)**	**15.87 (14.6)**
3DETR-m (P-Fusion II)	10%	PointNet++	**35.24 (34.41)**	**13.96 (14.43)**
3DETR-m (P-Fusion III)	10%	PointNet++	**30.4 (28.98)**	**10.41 (10.13)**
3DETR-m (baseline)	25%	-	43.3 (42.76)	21.56 (21.45)
3DETR-m-10L (baseline)	25%	-	43.86 (43.51)	22.84 (22.66)
3DETR-m (P-Fusion I)	25%	PointNet++	**46.36 (45.49)**	**25.36 (25.46)**
3DETR-m (P-Fusion II)	25%	PointNet++	**46.84 (45.82)**	**22.76 (24.10)**
3DETR-m (P-Fusion III)	25%	PointNet++	**44.83** (43.18)	20.35 (22.65)
3DETR-m (baseline)	50%	-	56.37 (56.05)	35.22 (35.18)
3DETR-m-10L (baseline)	50%	-	57.34 (55.84)	34.5 (36.07)
3DETR-m (P-Fusion I)	50%	PointNet++	57.32 **(56.08)**	**37.0** (35.44)
3DETR-m (P-Fusion II)	50%	PointNet++	**58.66 (57.27)**	**35.87 (37.52)**
3DETR-m (P-Fusion III)	50%	PointNet++	**58.0** (57.17)	33.7 **(37.59)**
3DETR-m (baseline)	75%	-	60.41 (59.76)	41.2 (40.49)
3DETR-m-10L (baseline)	75%	-	61.29 (60.48)	41.15 (41.84)
3DETR-m (P-Fusion I)	75%	PointNet++	**62.1** (59.94)	39.89 **(42.64)**
3DETR-m (P-Fusion II)	75%	PointNet++	**62.69 (61.06)**	**42.78 (41.35)**
3DETR-m (P-Fusion III)	75%	PointNet++	**62.68 (61.20)**	**42.24 (43.56)**
3DETR-m (paper)	100%	-	65.0	47.0
3DETR-m (baseline)	100%	-	63.93 (62.55)	45.43 (45.52)
3DETR-m-10L (baseline)	100%	-	65.34 (64.16)	44.72 (46.22)
3DETR-m (P-Fusion I)	100%	PointNet++	**65.68 (64.73)**	**45.69 (45.6)**
3DETR-m (P-Fusion II)	100%	PointNet++	**66.24 (65.23)**	**47.25 (47.0)**
3DETR-m (P-Fusion III)	100%	PointNet++	**66.46 (65.5)**	**45.17 (49.46)**

Table 2: Mean Average Precision (mAP) performance comparison with the state-of-the-art models trained on ScanNetv2 dataset. PointNet++w2 means the backbone has width of 2 in MLP networks. P-Fusion implies a pre-trained model with the Fusion (I, II or III) technique. Our proposed model 3DETR-m (P-Fusion III) demonstrates competitive scores as with Group-free (L6, O256) result.

Architecture	Backbone	mAP@25	mAP@50
VoteNet	Pointnet++	62.9	39.9
MLCVNet	Pointnet++	64.5	41.4
Group-Free (L6, O256)	Pointnet++w2 ×	67.3 (66.2)	48.9 (48.4)
Group-Free (L12, O512)	Pointnet++w2 ×	69.1 (68.6)	52.8 (51.8)
3DETR	-	62.7	37.5
3DETR-m	-	65.0	47.0
3DETR-m (P-Fusion I) (ours)	PointNet++	**65.68**	45.69
3DETR-m (P-Fusion II) (ours)	PointNet++	**66.24**	**47.25**
3DETR-m (P-Fusion III) (ours)	PointNet++	**66.46**	**49.46**

Table 3: Per-class evaluation of 3DETR-m (SUN RBG-D pre-trained) on validation-set on 100% ScanNetV2 at AP@25 IoU. Numbers in bold show increases with our modification in 3DETR-m (P-Fusion II) as compared to 3DETR-m (paper) results.

Method	cab	bed	chair	sofa	tabl	door	windw	bkshf	pic	contr	desk	curtn	frdge	showr	toilt	sink	bath	grbin	mAP
3DETR-m (paper)	49.4	83.6	90.9	89.8	67.6	52.4	39.6	56.4	15.2	55.9	79.2	58.3	57.6	67.6	97.2	70.6	92.2	53.0	65.0
3DETR-m (baseline)	52.71	80.71	89.75	89.92	66.04	54.91	38.14	51.48	13.52	58.76	75.08	47.09	56.6	62.86	94.34	70.17	85.96	51.42	64.88
3DETR-m-10L (baseline)	52.39	82.81	89.25	91.37	67.73	54.32	42.57	46.67	15.56	57.19	74.85	63.76	56.63	73.77	99.53	67.63	89.54	50.61	65.34
3DETR-m (P-Fusion I)	48.04	83.08	**91.88**	88.89	**70.04**	**56.39**	**40.44**	47.86	14.31	**57.39**	75.52	**62.15**	56.19	**69.49**	**99.52**	**73.05**	**95.26**	52.76	**65.68**
3DETR-m (P-Fusion II)	**51.17**	**85.48**	90.71	**90.78**	**71.21**	**55.07**	**47.05**	47.49	**16.05**	**63.43**	**80.59**	**59**	**58.52**	**69.65**	95.99	68.36	88.43	**53.42**	**66.24**
3DETR-m (P-Fusion III)	47.1	83.35	**91.56**	89.2	**73.24**	52.3	**46.28**	55.54	**15.43**	**56.48**	77.69	**62.26**	**58.59**	**71.72**	96.32	**75.63**	92.11	51.4	**66.46**

computer vision tasks such as classification and segmentation.

REFERENCES

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2016). Multi-view 3d object detection network for autonomous driving.

Chen, Y., Liu, S., Shen, X., and Jia, J. (2019). Fast point r-cnn.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Giancola, S., Zarzar, J., and Ghanem, B. (2019). Leveraging shape completion for 3d siamese tracking.

Graham, B., Engelcke, M., and van der Maaten, L. (2017). 3d semantic segmentation with submanifold sparse convolutional networks.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. (2021). PCT: Point cloud transformer. *Computational Visual Media*, 7(2):187–199.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. (2017). Joint 3d proposal generation and object detection from view aggregation.

Landrieu, L. and Simonovsky, M. (2017). Large-scale point cloud semantic segmentation with superpoint graphs.

Liu, Z., Zhang, Z., Cao, Y., Hu, H., and Tong, X. (2021). Group-free 3d object detection via transformers.

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

Misra, I., Girdhar, R., and Joulin, A. (2021). An end-to-end transformer model for 3d object detection.

Pan, X., Xia, Z., Song, S., Li, L. E., and Huang, G. (2020). 3d object detection with pointformer.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2017a). Frustum pointnets for 3d object detection from rgb-d data.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

Qi, H., Feng, C., Cao, Z., Zhao, F., and Xiao, Y. (2020). P2b: Point-to-box network for 3d object tracking in point clouds.

Rukhovich, D., Vorontsova, A., and Konushin, A. (2021). Fcaf3d: Fully convolutional anchor-free 3d object detection.

Shi, S., Wang, X., and Li, H. (2018). Pointrcnn: 3d object proposal generation and detection from point cloud.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576.

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

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wang, W., Zhang, J., Cao, Y., Shen, Y., and Tao, D. (2022). Towards data-efficient detection transformers.

Xie, Q., Lai, Y.-K., Wu, J., Wang, Z., Zhang, Y., Xu, K., and Wang, J. (2020a). Mlcvnet: Multi-level context votenet for 3d object detection.

Xie, S., Gu, J., Guo, D., Qi, C. R., Guibas, L. J., and Litany, O. (2020b). Pointcontrast: Unsupervised pre-training for 3d point cloud understanding.

Yamada, R., Kataoka, H., Chiba, N., Domae, Y., and Ogata, T. (2022). Point cloud pre-training with natural 3d structures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21283–21293.

Yang, B., Luo, W., and Urtasun, R. (2019). Pixor: Real-time 3d object detection from point clouds.

Zhao, H., Jiang, L., Jia, J., Torr, P., and Koltun, V. (2020). Point transformer.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection.