Dense Captioning for 3D Scenes with Transformers

Kağan Küçükaytekin Antonio Oroz
{kagan.kuecuekaytekin, antonio.oroz}@TUM.de
Github Repositories: S2C-MMT, 3DETR-S2C

Abstract

We tackle the task of 3D Dense Captioning by changing the Scan2Cap architecture to incorporate Transformers. We propose two models: 3DETR-S2C, which replaces the VoteNet style proposal module of the Scan2Cap with a 3DETR style transformer and S2C-MMT, which replaces captioning module with a transformer inspired from MMT. We show that both models achieve higher scores on all benchmarks than the baseline model, proving the reliability of transformers in the task of 3D Dense Captioning.

1. Introduction

3D Dense Captioning is a task, in which object detection and captioning are done jointly. Scan2Cap tackles this problem with modules exploiting GNN's and Attention. Inspired by Vaswani et al. [13], we believe that transformers offer a promising upgrade to the Scan2Cap architecture. We consider two different approaches. The first approach will focus on changing the Detection module, while the second approach replaces the Context-Aware Captioning Module of Scan2Cap [2].

2. Related Work

Dense Captioning on 3D Point Clouds. Scan2Cap, the initial model proposed by Chen et al. [2], consists of three main features: The PointNet++ Backbone [10] with a Voting Module from Qi et al. [9] for extracting the object features, the Relational Graph Module to discover inter-object relationships and enhance object features, and finally the Context-Aware Attention Module for generating descriptive tokens for each object.

Object Detection on 3D Point Clouds with Transformers. Misra et al. [8] and Liu et al. [6] proved that transformers can improve the performance in the task of detecting objects and generating bounding boxes in 3D Space.

Image Captioning with Meshed-Memory Transformers. The Meshed-Memory Transformer proposed by Cornia et al. [3] can be used to generate captions for images. This architecture can be split into an Encoder and Decoder. It is able to encode multi-level object relationships and generate captions based on those encodings.

3. Method

3.1. 3DETR-S2C

Since 3D object detection from point clouds is an order-independent task, transformers qualify as suitable candidates for this task. In common benchmarks ScanNet [4] and SUNRGB [12], recent works from Misra et al. [8] and Liu et al. [6] prove that transformers can indeed perform better than VoteNet [9]. However, their performance as a building block of a larger framework such as Scan2Cap is not fully explored yet. In our project, we show that transformers are in fact capable of working in combination with graph and captioning modules to provide better results than Scan2Cap.

3.1.1 Comparison of Different Proposal Modules

In the following we explain advantages of 3DETR over VoteNet and why we have chosen 3DETR over GF3D, the model proposed by Liu et al. [6].
Both Pointnet++ backbone of VoteNet and the transformer making up the 3DETR consist of an encoder-decoder type architecture. In its encoder part, Pointnet++ groups the point clouds with hand-crafted components which make use of the local geometry. The number of centroids, the grouping radius and the grouping strategy are important hyperparameters for the performance of this backbone. On the contrary, a transformer encoder can represent point-to-point relationships by only relying on the self-attention mechanism, which reduces the model complexity significantly. But computing self-attention for tens of thousands of points in a 3D scene is highly costly. Therefore, a single set abstraction layer is used in 3DETR before the encoder layer to subsample arbitrary number of 3D points into 2048 points.
Just like VoteNet, GF3D uses the Pointnet++ backbone instead of an attention based encoder. It also relies on learnable positional embeddings and point sampling modules. All of these factors increase the model complexity. Addi-

tionally, in contrast to the shared MLP in 3DETR, GF3D uses separate MLP heads for each decoder output to generate bounding boxes. Although this leads to having a highly reliable ensemble of decoders, it increases the model size. Even though it does not have transformer encoder layers and a smaller decoder stack, it ends up having 14.5 million parameters, which is almost double the amount of parameters in 3DETR. On a different account, even though it has less model dependency, 3DETR has 7.3M parameters, which is more than 7 times of the number of parameters in VoteNet, which rely on shared vote-generation and vote-aggregation layers to reduce the number of parameters substantially. Because transformers usually perform better by stacking multiple encoders or decoders, model size becomes a remarkable weakness of transformer architectures in contrast to VoteNet.

Model size was an important design consideration for us. Due to the relatively small dataset size of ScanRefer, overfitting has been a present issue throughout the course of our project. For this reason, instead of end-to-end training the 3DETR from scratch on ScanRefer dataset, we prefer to transfer a model trained on the larger ScanNet dataset for better generalization.

3.1.2 Implementation Details

In our initial attempt, we kept the encoder and decoder modules of the 3DETR and used its final decoder output as input and trained the model using the proposal module and losses from VoteNet. In this version we were not able to achieve comparable results to our baseline. This result was in accordance with the observation made by Misra et. al. [8] when they have obtained lower performance than VoteNet with VoteNet's loss functions. In the end we replaced the whole detection module including the bounding box generation step with the complete architecture of 3DETR including its loss functions. We augmented an additional shared MLP head to the decoder output to generate 128-dimensional object features used for caption generation.

One remarkable difference between 3DETR and VoteNet is the semantic class called "background" in 3DETR. In VoteNet, the "objectness" probability of a bounding box is determined by comparing two classes representing "object" and "not-an-object", while in 3DETR the "background" class is added to the available semantic classes. The "background" class easily translates into "not-an-object" class in Vote-Net, but there is not a direct equivalent of "object" class in 3DETR. The existence of the "object" class is of crucial importance for calculating bounding-box masks used for masking out empty boxes during caption generation. In our first attempt, we masked out all the bounding boxes having more than 50% probability of belonging to the background-class, but this condition turned out to be a harder one. In the more relaxed final version, we compare the logits of the background class with the maximum logit from all the semantic objects and mask out the bounding box if the background logit is greater.

3.2. S2C-MMT

In this approach we set out to replace the Relational Graph Module together with the Context-Aware Captioning Module in the original Scan2Cap implementation [2] by the Meshed-Memory Transformer introduced by Cornia et al. [3].

Our idea was initially motivated by the promising results of the Transformer Architecture in Natural Language Processing tasks, as shown by Vaswani et al. [13]. One key advantage in using Transformers is that we're able to generate our tokens for a caption in parallel during training, relying on the previous ground truth words to predict the current word.

The idea of using Transformers also for captioning tasks was explored by the Meshed-Memory Transformer [3]. We were inspired by the idea to encode our object proposals with the Meshed-Memory Encoder and to use the Meshed Decoder to generate the captions for each object proposal in a given scene.

During our experiments we realized that it might be useful the reintegrate the Relational Graph Module into our architecture as it also provides a stabilizing loss term and only focuses on the local neighborhood instead of the whole scene. In this case we would use the output of the Relational Graph Module as the input into the Meshed-Memory Encoder.

To simplify our model we decided to remove the Meshed-Memory Encoder as both the Relational Graph Module and the Meshed-Memory Encoder now were solving the same task of capturing inter-object relationships. With this approach we would still retain advantages of Transformers like the parallelizable training and being able to attend to the to target object and it's neighbors. At the same time we are able to reduce our number of parameters significantly. A comparison of the models is done in section 4.3.4.

3.2.1 Caption Generation

To generate the captions we use the decoder as used in Cornia et al. [3]. As input to the decoder we use the output of the Relational Graph Module, which provides enhanced object features V and relation features E. To transform the features from dimension d_{f} into the needed internal decoder dimension d_{dec}, we use a single fully-connected layer.

$$\hat{V}_i = \text{LayerNorm}(\text{ReLU}([V_i W_d, 1])) \tag{1}$$

$$\hat{E}_{i,j} = \text{LayerNorm}(\text{ReLU}([E_{i,j} W_d, 0])) \tag{2}$$

Where W_d is a matrix with shape $d_{\text{f}} \times (d_{\text{dec}} - 1)$. $[\cdot, \cdot]$ is concatenation.

$$\hat{X}_{i,1} = \hat{V}_i, \quad \hat{X}_{i,j+1} = \hat{E}_{i,j} \tag{3}$$

For $i = 1, ..., n$ and $j = 1, ..., k$. $\hat{X}$ has dimension $n \times (k + 1) \times d_{\text{dec}}$. n being the batch size and k being the amount of neighborhood objects used in the Relational Graph Module.

The concatenation serves the purpose of indicating the target object for which the caption will be generated. Different to the original Meshed-Memory Transformer implementation, we use $\hat{X}$ as the only input layer into the decoder. Identically to [3], we also use the previously generated words and the positional encoding as input.

3.3. Training and Inference

S2C-MMT We train S2C-MMT with AdamW [7] optimizer for 50 epochs with the learning rate set to 1e-3 and weight decay set to 1e-4. We then train the best model further for 5 epochs with the learning rate set to 1e-4. During evaluation we utilize Beam Search, which we adapted from Cornia et al. [3] with a size of 2.

3DETR-S2C We train 3DETR-S2C using ADAM [5] optimizer. Initially we do 10 epochs using a frozen 3DETR-m module which is trained 1080 epochs on ScanNet dataset. We use a learning rate of 1e-4 and weight decay of 1e-5 at this step. We then allow end to end training for 30 epochs with the same parameters and finally fine tune the best model for 10 additional epochs with a 1e-5 learning rate and a weight decay of 1e-4.

4. Experiments

4.1. Quantitative Analysis

We compare our model using the benchmarks provided by Chen et al. [2] on the official val split of ScanRefer [1]. As seen in 1, both transformer models outperform Scan2Cap in all benchmarks. At 0.5 IoU, S2C-MMT improves the CIDEr score by a margin of 13%, while 3DETR-S2C yields a 14% increase in the BLEU-4 score.

4.2. Qualitative Analysis

A qualitative analysis is shown in figure 1. We can see that the models produce comparable captions. 3DETR-S2C and S2C both failed to correctly predict the object type for the desk.

4.3. Ablations

In the following we will present ablations on the S2C-MMT method. 3DETR-S2C was not considered for ablation studies due to computational and time constraints.

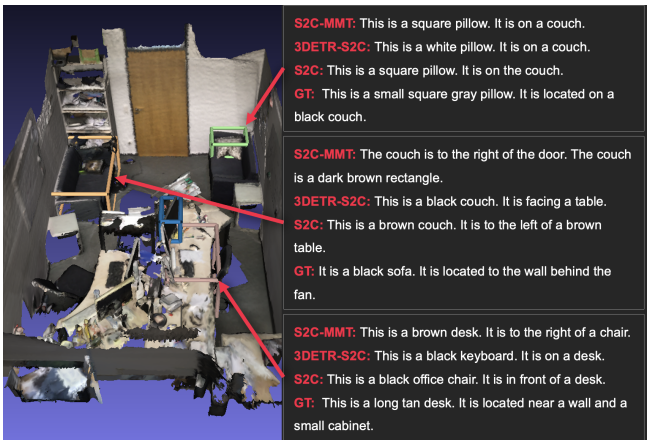

Figure 1. Qualitative Analysis. Both of the transformer based models are able to generate comparable captions to Scan2Cap. For the last caption, only S2C-MMT is able to generate a caption which represents the object.

4.3.1 Does the Number of Decoder Layers Matter?

We trained our S2C-MMT model with different numbers of layers. Our comparison in table 2 shows, that for a large number of decoder layers ($L_d = 5$), it becomes impossible to train the model in 50 epochs. Our experiment also shows, that multiple decoder layers are still preferable to a single decoder layer.

4.3.2 Does Beam Search Help?

For this experiment we want to compare the Iterative Search method against an adapted Beam Search implementation from Cornia et al. [3] with different beam sizes. Our experiments show that Beam Search can improve the results for smaller beam sizes. Although the increase in performance degrades for our model when considering beam sizes greater than 2. The comparison is shown in table 3.

4.3.3 Can Reinforcement Learning Improve the Model further?

We also considered using Reinforcement Learning to train our model further. For this we first trained our model for 50 epochs on the Cross-Entropy Loss as done for the original Scan2Cap architecture [2]. Afterwards the model performing best on the evaluation set is picked and further trained with Self-Critical Sequence Training as proposed by Rennie et al. [11]. We also adapted the change to use the reward averaged over 5 sentences generated with Beam Search as described in Cornia et al. [3]. After a few hundred steps we observed a significant drop in performance as shown in figure 2.

A closer look at the generated sentences showed that our

	C@0.25IoU	B-4@0.25IoU	M@0.25IoU	R@0.25IoU	C@0.5IoU	B-4@0.5IoU	M@0.5IoU	R@0.5IoU	mAP@0.5IoU
Scan2Cap	56.82	34.18	26.29	55.27	39.08	23.32	21.97	44.78	32.21
3DETR-S2C	57.52	35.64	**26.92**	**57.21**	41.53	**26.63**	**23.07**	**47.60**	38.61
S2C-MMT	**64.35**	**36.45**	26.90	56.28	**44.17**	24.34	22.30	45.36	**38.72**

Table 1. Comparison of the proposed models with Scan2Cap. Our models are able to outperform the baseline.

	C@0.5IoU	B-4@0.5IoU	M@0.5IoU	R@0.5IoU
$L_d = 1$	38.39	21.81	21.43	43.89
$L_d = 3$	**40.90**	**23.85**	**21.80**	**45.13**
$L_d = 5$	0.02	0.00	9.89	20.32

Table 2. Comparison of different numbers L_d of decoder layers. Our results show that multiple layers offer an improvement in performance. Too many layers (e.g., $L_d = 5$) become too difficult to train, resulting in significantly lower scores.

	C@0.5IoU	B-4@0.5IoU	M@0.5IoU	R@0.5IoU
Iterative/$BS = 1$	42.29	23.99	22.18	45.07
$BS = 2$	**44.17**	**24.34**	**22.30**	**45.36**
$BS = 3$	43.31	23.57	22.13	45.17
$BS = 5$	41.80	22.79	21.93	45.13
$BS = 10$	40.35	21.77	21.72	44.86
$BS = 20$	39.09	20.91	21.48	44.48

Table 3. Comparison of different Beam sizes (BS) for Beam Search. Iterative Search is equivalent to Beam Search with $BS = 1$. Our comparison shows, that Beam Search can improve captioning performance. For bigger Beam sizes the scores start to deteriorate.

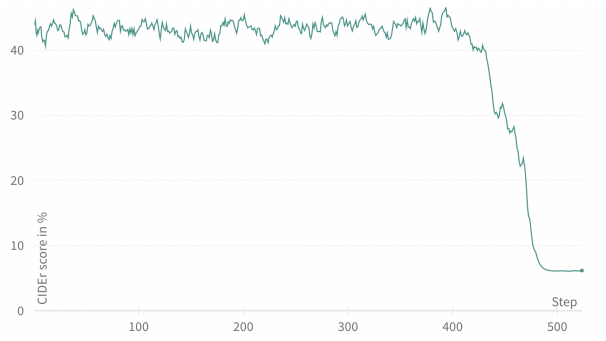

Figure 2. Plot of CIDEr scores in percent on the current batch during training with Reinforcement Learning. The CIDEr score falls off after a few 100 steps.

model started to predict only a few words repeatedly. We tried different methods including lowering the learning rate significantly, reverting to a reward baseline by greedy caption generation and also limiting the loss function to only positive or negative rewards. We were not able to stabilize the training with Reinforcement Learning.

	C@0.5IoU	B-4@0.5IoU	M@0.5IoU	R@0.5IoU
E+D	32.99	21.92	20.96	44.40
G+E+D	39.61	22.59	21.63	44.46
G+D	**44.17**	**24.34**	**22.30**	**45.36**

Table 4. Comparison of three different compositions of modules used in S2C-MMT. G is the Relational Graph Module from Scan2Cap [2], E is the Meshed-Memory Encoder and D is the Decoder from the Meshed-Memory Transformer [3]. It is important to note that the E+D model was trained in the initial part of our project. For this model we were not able to generate results under equal conditions due to time constraints. The E+D model has a bigger internal transformer feature size, for evaluation Iterative Search instead of Beam Search was used and unlike the other two models it was not trained for 5 additional epochs on a lower learning rate and different weight decay. Our results show that the G+D model performs the best in this comparison.

4.3.4 Can the Meshed Encoder Replace the Graph Module

As explained in section 3.2 we tried three different compositions of modules to test the usefulness of the Meshed-Memory Encoder and the Relational Graph Module.

As shown in the table 4 the model only using the Relational Graph Module together with the Decoder (G+D) performs the best across all metrics regarding caption evaluation.

5. Conclusion

We show that transformers are reliable system components on the large task of 3D Dense Captioning. Both models we propose, 3DETR-S2C and S2C-MMT, obtain higher scores in all benchmarks than the baseline method Scan2Cap. We provide our code and the results of our experiments.

6. Future Work

After confirming the reliable performance of transformers for all the modules in Scan2Cap, we wanted to take one more step further than the goal of our project and combine 3DETR and MMT into a single model. Due to hardware limitations we had to choose between using our resources on maturing this new model or finishing our ablation studies for S2C-MMT. A model consisting of 3DETR and MMT could be a promising future endeavour.

References

[1] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. *16th European Conference on Computer Vision (ECCV)*, 2020. 3

[2] Dave Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X. Chang. Scan2cap: Context-aware dense captioning in RGB-D scans. *CoRR*, abs/2012.02206, 2020. 1, 2, 3, 4

[3] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. M^2: Meshed-memory transformer for image captioning. *CoRR*, abs/1912.08226, 2019. 1, 2, 3, 4

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017. 1

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[6] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *CoRR*, abs/2104.00678, 2021. 1

[7] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. 3

[8] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. *CoRR*, abs/2109.08141, 2021. 1, 2

[9] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. *CoRR*, abs/1904.09664, 2019. 1

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 1

[11] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016. 3

[12] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576. IEEE Computer Society, 2015. 1

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 1, 2