DD2412 Deep Learning Advanced

Mini-Project on Graph Networks and Architectures

Federico Baldassarre

December 2020

1 Introduction

When dealing with unordered point clouds, two viable deep learning techniques are transformers and graph networks. Transformers are a form of attention-based set-to-set transformation, where each point can attend to any other point in the input. Graph networks are usually constrained to operate on local neighborhoods of each point, e.g. using a k-nearest connectivity pattern.

In this project, we consider the ShapeNet dataset [Yi et al., 2016] and evaluate two different network architectures, based respectively on the Performer architecture [Choromanski et al., 2020] and on the Graph Attention Network architecture [Veličković et al., 2018]. The two architectures are compared on a classification task comprising the 16 shape categories in ShapeNet. To further analyze the differences between the two architecture, we apply GradCAM to the model predictions and highlight the most relevant points of the input[1].

2 Method

2.1 Performers

The self-attention mechanism used in the Transformer architecture was formalized in "Attention is all you need" [Vaswani et al., 2017]. Given a set of input vectors with no connectivity information, it computes a set-to-set output as

[1] Code, data, and trained models at: github.com/baldassarreFe/performer-gat-shapenet.

such:

$$
\begin{aligned}
\boldsymbol{x}_i &\in \mathbb{R}^d \\
\boldsymbol{q}_i &= f_{\text{query}}(\boldsymbol{x}_i) \in \mathbb{R}^d \\
\boldsymbol{k}_j &= f_{\text{key}}(\boldsymbol{x}_j) \in \mathbb{R}^d \\
\boldsymbol{v}_j &= f_{\text{value}}(\boldsymbol{x}_j) \in \mathbb{R}^d \\
\alpha_{i,j} &= \frac{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_j)}{\sum_\ell \exp(\boldsymbol{q}_i^T \boldsymbol{k}_\ell)} \\
\boldsymbol{y}_i &= \sum_{j=1}^{N} \alpha_{i,j} \boldsymbol{v}_j
\end{aligned}
\tag{1}
$$

Each vector $\boldsymbol{x}_i$ in Equation 1, representing a point in the input point cloud is projected three times into resp. a query vector, a key vector, and a value vector. The dot product between queries and keys measures how much a point i is "interested" in the value of point j. The pairwise dot products are normalized using a softmax operation which yields weighting coefficients for the final weighted sum.

In the multi-head attention formulation, the projection, dot product and weighted sum operations are repeated a number of times on the same input with different projection parameters. The size of the projected queries, keys, and values is divided by the number of heads used. Fianlly, the output values of each "head" are then concatenated in the output.

A transformer layer consists in the above multi-head self-attention operation, denoted as Att, combined with residual connections, layer normalization and dropout:

$$
\begin{aligned}
&\boldsymbol{x}_i \in \mathbb{R}^d \\
&\boldsymbol{u}_i = \textsc{LayerNorm}(\boldsymbol{x}_i + \textsc{Dropout}(\textsc{Att}(\boldsymbol{x}_i))) \\
&\boldsymbol{v}_i = \textsc{Dropout}(\boldsymbol{W}_2 \textsc{Dropout}(\textsc{ReLu}(\boldsymbol{W}_1 \boldsymbol{u}_i))) \\
&\boldsymbol{z}_i = \textsc{LayerNorm}(\boldsymbol{u} + \boldsymbol{v}) \\
&\quad \text{with } \boldsymbol{W}_1 \in \mathbb{R}^{2d \times d}, \boldsymbol{W}_2 \in \mathbb{R}^{d \times 2d}
\end{aligned}
\tag{2}
$$

The main issue with the attention mechanism is the quadratic dependence on the number of inputs. In fact, for N inputs vectors, one has to compute and maintain a $N \times N$ matrix of dot-product weights α. This prevents transformer-based architectures from being applied to large inputs, such as long text sequences or dense point clouds. In the article "Rethinking attention with performers", Choromanski et al. [2020] introduce an alternative formulation that removes the explicit computation of the dot-product attention matrix and instead approximates it using a similar procedure as kernel methods. The resulting attention approach, named Fast Attention Via positive Orthogonal Random features approach (FAVOR+), scales linearly with the number of inputs allowing to process much longer inputs.

2.2 Graph Attention Networks

Graph Attention Networks, introduced by Veličković et al. [2018], make explicit use of a graph-based connectivity between the points in the input. Each point, i.e. a node in the graph, aggregates information from its neighbors by means of a weighed sum, whose weights are determined as dot product of between a fixed vector and the concatenated features of the node and its neighbor.

$$
\begin{aligned}
\boldsymbol{x}_i &\in \mathbb{R}^d \\
\boldsymbol{u}_{i,j} &= [\boldsymbol{W}\boldsymbol{x}_i \,\|\, \boldsymbol{W}\boldsymbol{x}_j] \in \mathbb{R}^{2d} \\
\alpha_{i,j} &= \frac{\exp\left(\textsc{ReLU}\left(\mathbf{a}^T \boldsymbol{u}_{i,j}\right)\right)}{\sum_{\ell \in \mathcal{N}(i)\cup\{i\}} \exp\left(\textsc{ReLU}\left(\mathbf{a}^T \boldsymbol{u}_{i,\ell}\right)\right)} \\
\boldsymbol{y}_i &= \sum_{j=1}^{N} \alpha_{i,j} \boldsymbol{W}\boldsymbol{x}_j \\
&\text{with } \boldsymbol{W} \in \mathbb{R}^{d\times d}
\end{aligned}
\tag{3}
$$

In this project, we employ the graph attention in Equation 3 in its multi-headed version. For parity with the transformer layer in Equation 2, we embed graph attention in a layer that includes the same residual connections, layer normalization and dropout. Since the point clouds in ShapeNet do not have explicit edges between them, we connect each point to its 6 nearest neighbors in the original 3D space.

2.3 Network architecture

The following is the architecture of the classifier for the 16 shape categories in ShapeNet. The input features represent the 3-dimensional position and normal vectors of each point in the input point cloud $\{\boldsymbol{x}_i\}_1^N$.

$$
\begin{aligned}
\boldsymbol{x}_i &\in \mathbb{R}^6 \\
\boldsymbol{h}_i^0 &= \textsc{LayerNorm}(\boldsymbol{W}_2 \textsc{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}_i)) \\
\boldsymbol{h}_i^\ell &= \textsc{TransformerLayer}(\boldsymbol{h}_i^{\ell-1}) \qquad \ell = 1 \dots L \\
\bar{\boldsymbol{h}} &= 1/N \sum_{i=1}^{N} \boldsymbol{h}_i^L \\
\boldsymbol{y} &= \boldsymbol{W}_{\text{class}} \bar{\boldsymbol{h}} \\
\hat{c} &= \operatorname*{argmax}_c y_c
\end{aligned}
\tag{4}
$$

In the equation above, the input points are first projected into a higher-dimensional space by a 2-layer point-wise network and then processed through a stack of L transformer layers (either based on performer attention or of graph attention). Finally, all vectors are globally averaged to obtain a single representation for the entire point cloud, which is processed through a final linear

projection to obtain the 16-dim logits for classification. The `argmax` operation simply selects the category with the highest score as the classification output of the trained model.

2.4 Training

The network is trained by minimizing the cross-entropy loss between the predicted category and the true label of each sample in the training set. The Adam optimizer [Kingma and Ba, 2015] is used for mini-batch stochastic optimization. During training, the 12137 samples in the training set are iterated for a maxiumum of 15 epochs, unless the loss relative to the 1870 samples in the validation set stops decreasing. During training, each sample in the ShapeNet dataset is subsampled to 1000 points.

3 Evaluation

3.1 Hyperparameters

First, a hyperparameter search is performed over a few architectural parameters. In particular, we consider all combinations of the following hyperparameters:

Table 1: Hyperparameters for ShapeNet

Hyperparameter	Options
Architecture	Performer, Graph Attention
Number of layers	2, 4
Number of heads	2, 4
Per-head features	8, 16

For both architectures, the best choice of parameters appears to be: 2 layers, 4 attention heads, 16 features per head. These parameters are the ones used later for testing and explanation. The performer-based models seem to obtain slightly higher validation accuracy, but they remain comparable to the graph attention-based ones. In Figure 1, we can observe that the training dynamics of both architectures greatly overlap.

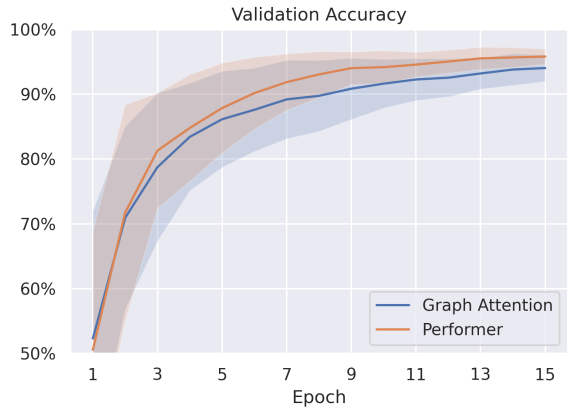

Figure 1: Mean and standard deviation of the validation accuracies obtained during training by the models in Table 1 grouped by architecture.

3.2 Testing

When tested on the 2874 samples of the test set, which are never used during training, the best perfomer-based and graph attention-based models obtain once again similar performance, with the former begin slightly better. Since during training the number of points was kept fixed to 1000, it is interesting to see how robust the two architectures are w.r.t. the number of input points. The plots in Figure 2 show the testing metrics of the two architecture types evaluated using sparser or denser representations of the point clouds. Both method show similar dynamics, with high performance when at least ~ 400 points are shown and degraded performance for sparser inputs. Interestingly, graph attention performs slightly better at 750 points than the 1000 used during training.

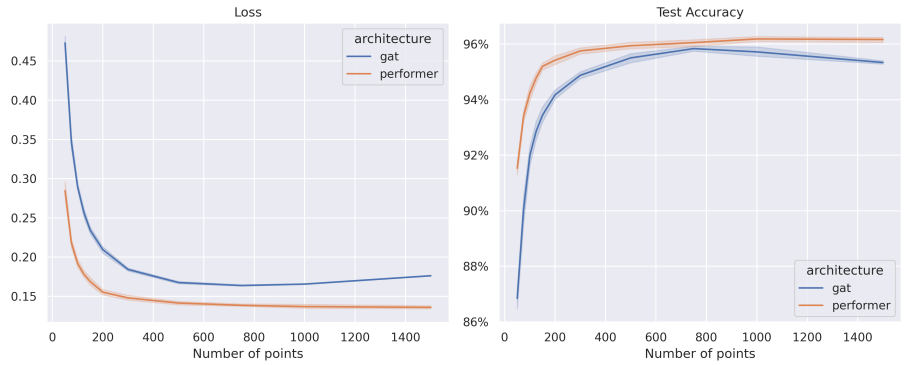

Figure 2: Loss and accuracy evaluated for several degrees of density of the point clouds in the test set (mean and standard deviation of 5 evaluations at 50, 75, 100, 125, 150, 200, 300, 500, 750, 1000, and 1500 points per sample).

Example predictions for a single sample are shown in Figure 3. Note how at the lower resolution it is actually hard to identify the object category. More examples are found in Appendix B.

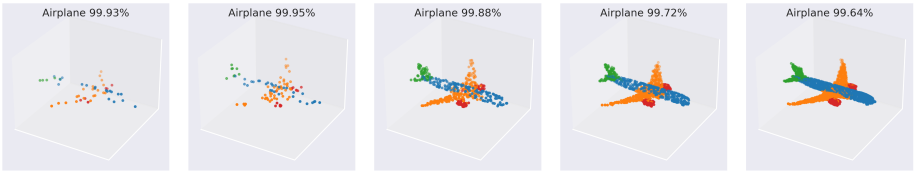

Figure 3: Graph attention model predicting the class "airplane" for a point cloud at different levels of density (50, 100, 500, 1000, 2000 points). The color of each point represents different parts of the object as annotated in the datasets. However, part information is not used in this project.

3.3 Explanation

We apply GradCAM [Selvaraju et al., 2017] to the predictions to highlight the parts of the input that more strongly contributed to the classification output. Specifically, with reference to Equation 4, the GradCAM relevance R of each input node i for a predicted class c is obtained as follows:

$$\begin{aligned} \alpha_k^c &= 1/N \sum_{i=1}^{N} \frac{\partial y^c}{\partial h_{i,k}^L} \\ R_i^c &= \text{ReLU}\left(\sum_{k=1}^{d} \boldsymbol{\alpha}^T \boldsymbol{h}_i^L\right) \\ \text{with } \boldsymbol{\alpha}^T &= \begin{bmatrix} \alpha_1 & \dots & \alpha_k & \dots & \alpha_d \end{bmatrix} \end{aligned} \tag{5}$$

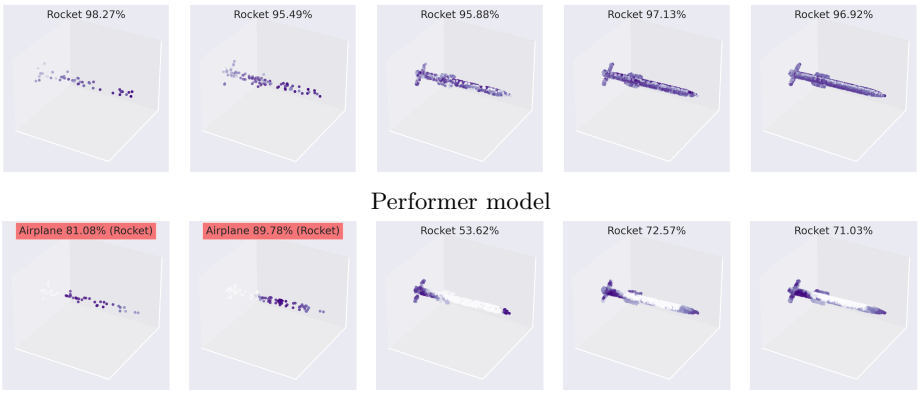

Figure 4: GradCAM visualization for the class "rocket" for the graph attention model and the performer model. A strong purple color indicates higher relevance, a white color indicates low relevance.

Figure 4 shows two visualization of GradCAM relevances computed for the graph attention model and the performer model. It appears that the performer model focuses on the object as a whole, attributing similar relevance to all points in the input. On the contrary, graph attention seems to focus on salient regions of the input. More GradCAM heatmaps can be found in Appendix C.

4 Conclusion

In this project, we considered two architectures for point cloud classification. Performers are a linear-complexity approximation of transformer architectures that allows to scale full self-attention to large number of inputs. Graph attention networks rely on graph connectivity to compute a local attention-based update for each node in the graph. Both architectures demonstrated great classification performance on the ShapeNet dataset. The different behavior of the architectures is revealed when using an explanation method such as GradCAM to visualize the most important points for a prediction.

References

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2015.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

A ShapeNet dataset

The following plots contain some statistics about the training subset of ShapeNet.

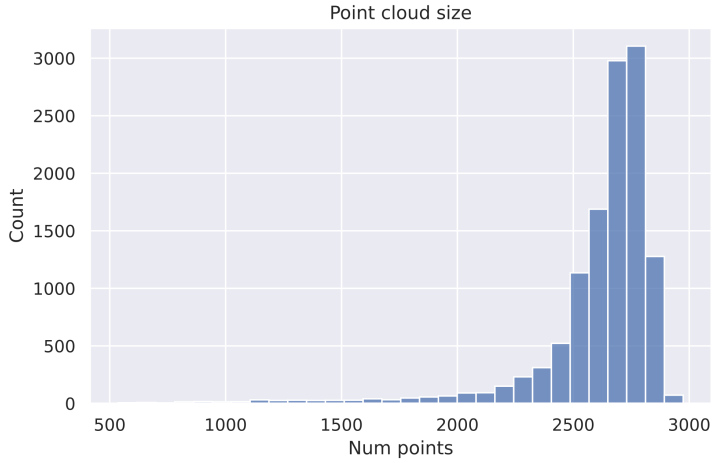

Figure 5: Point cloud size distribution in the ShapeNet dataset.

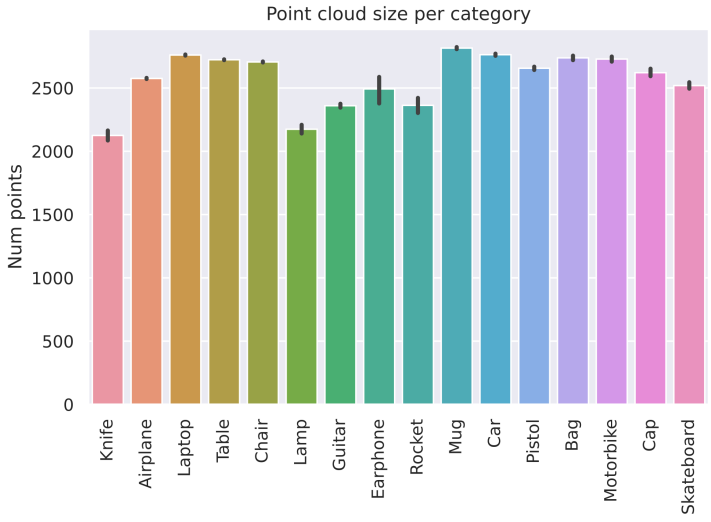

Figure 6: Mean and standard deviation of point cloud sizes per category.

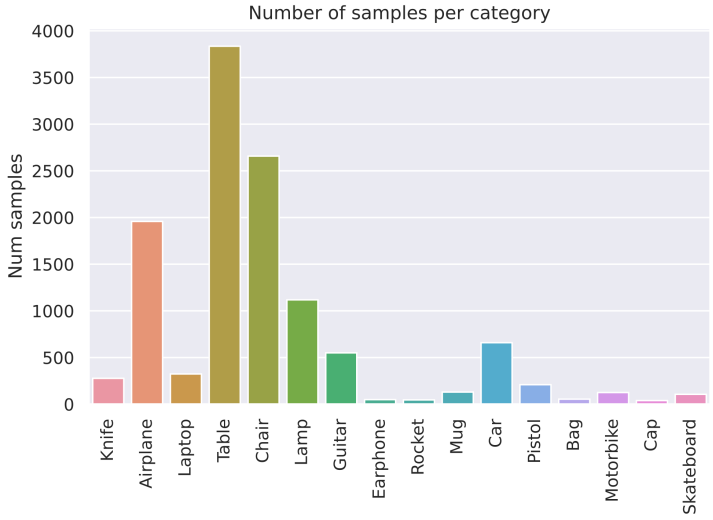

Figure 7: Number of sample in each category in the training set.

B Prediction visualization

The following pages contain a visualization of the predictions of the performer model and the graph attention model at different levels of point cloud density (50, 100, 500, 1000, 2000 points). The color of each point represents different parts of the object as annotated in the datasets. However, part information is only used for visualization and not as input to the classifiers.

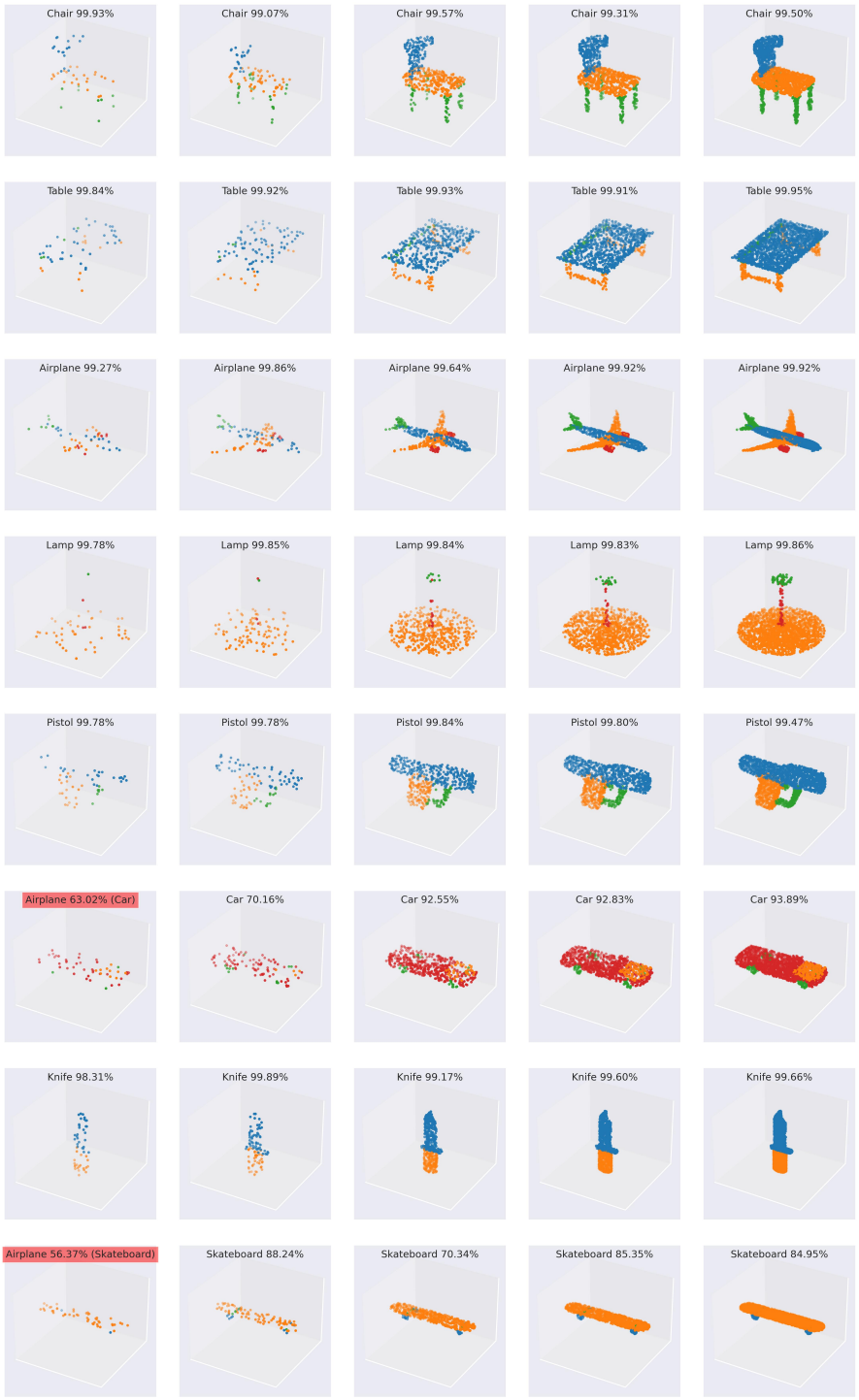

Figure 8: Performer model predictions (1/2).

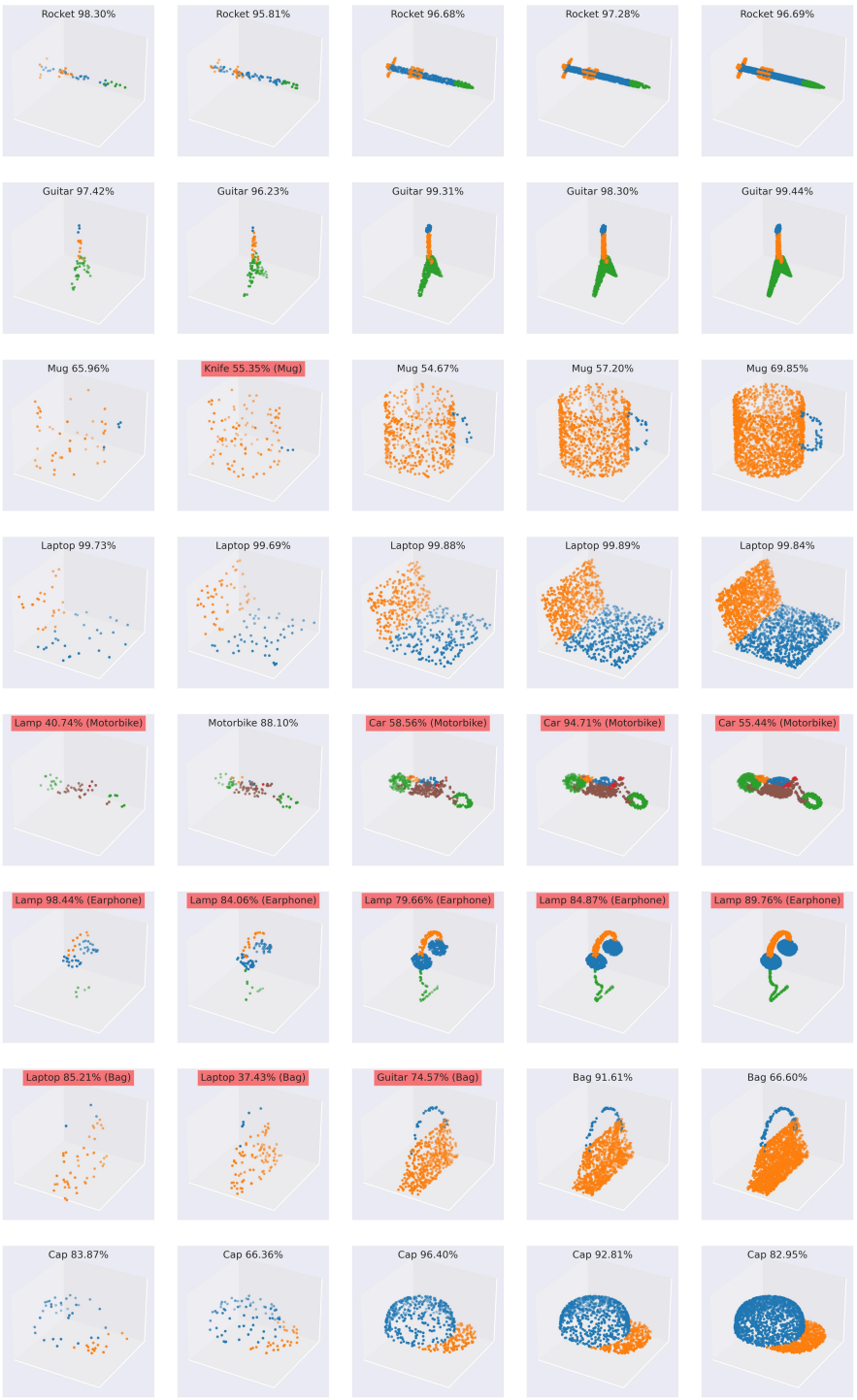

Figure 9: Performer model predictions (2/2).

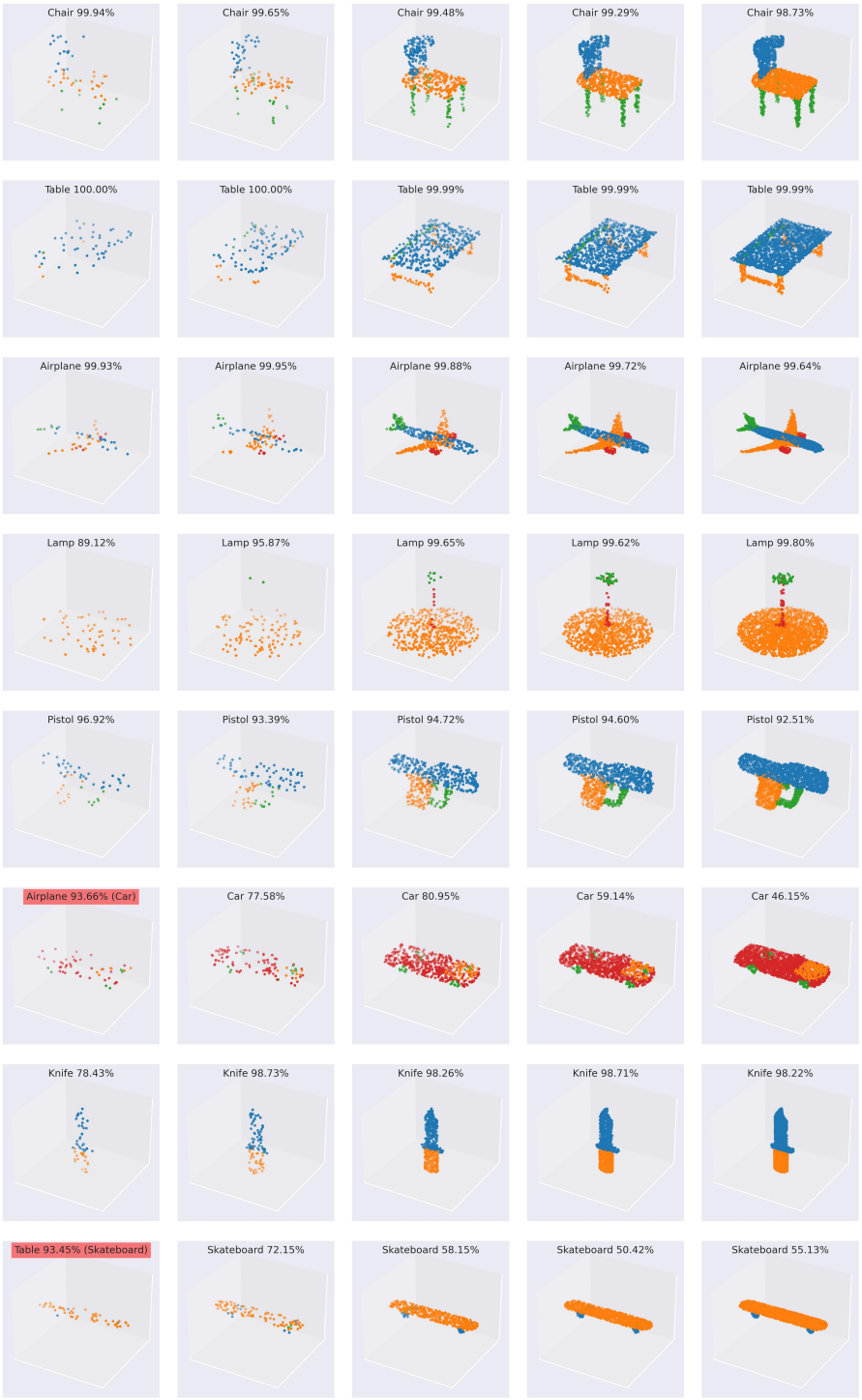

Figure 10: Graph attention model predictions (1/2).

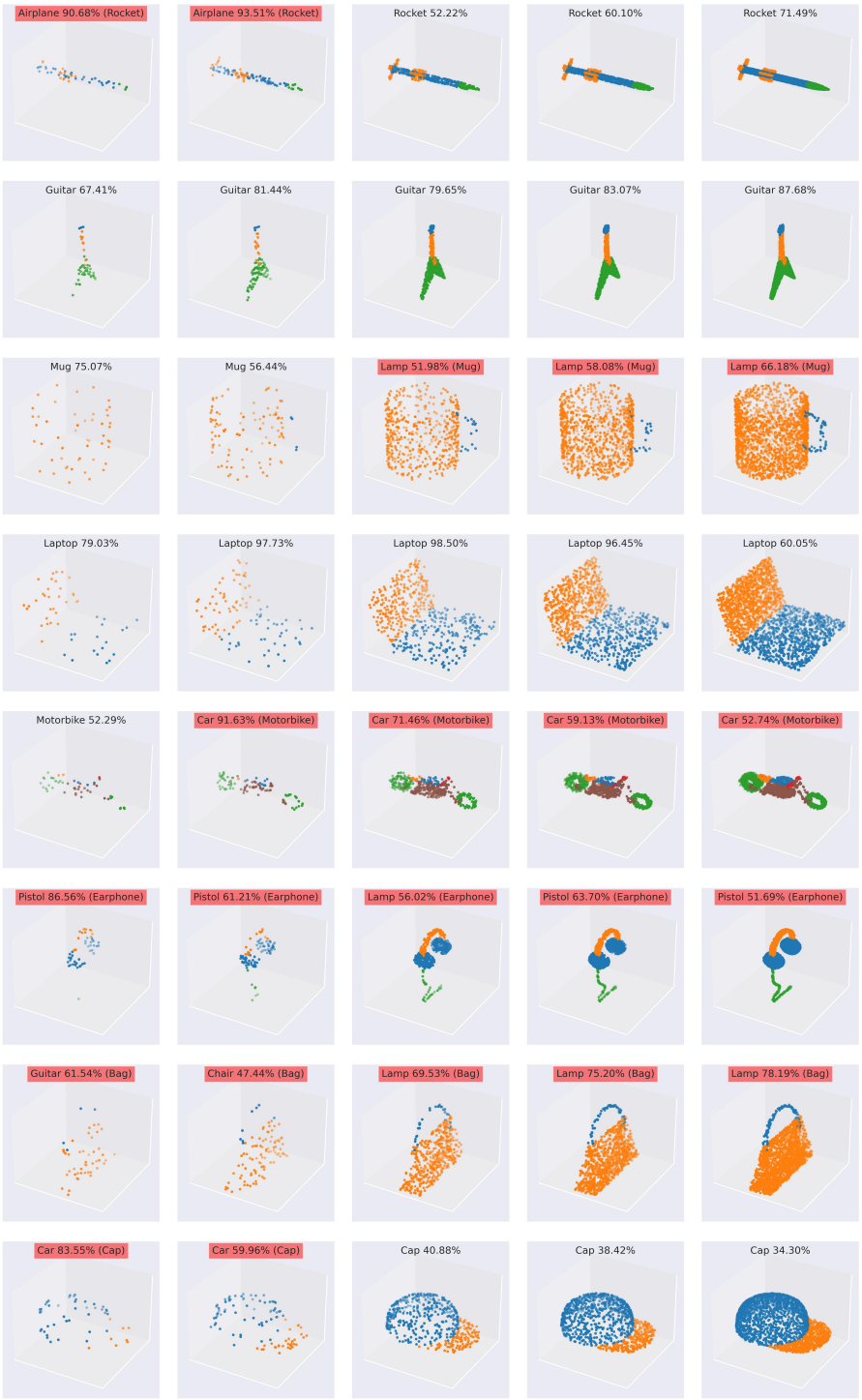

Figure 11: Graph attention model predictions (2/2).

C GradCAM visualization

The following pages contain a visualization of the GradCAM relevances assigned by each model the points in the input point clouds. From left to right, the resolution of each point cloud is 50, 100, 500, 1000, 2000 points.

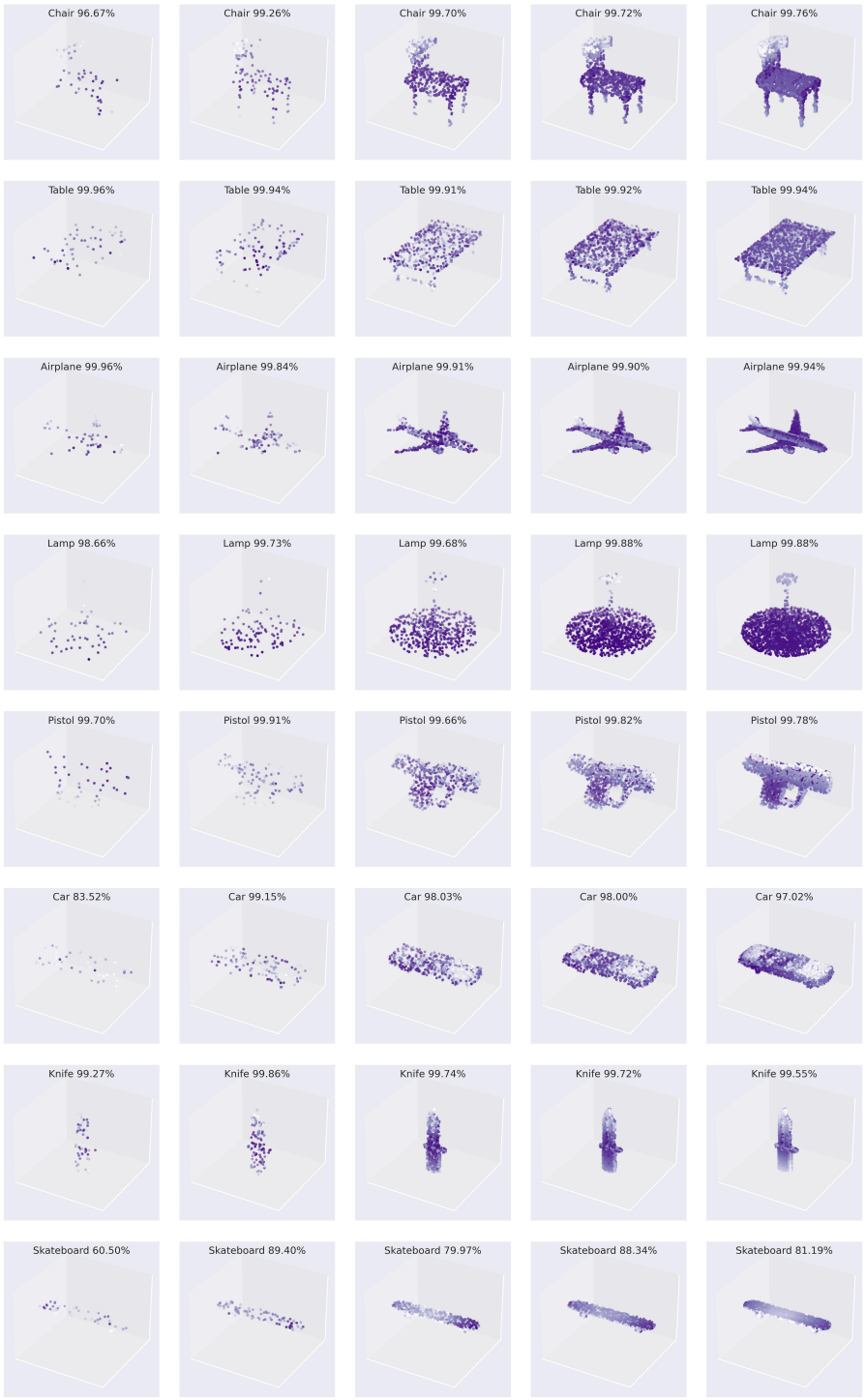

Figure 12: Performer model GradCAM visualization (1/2).

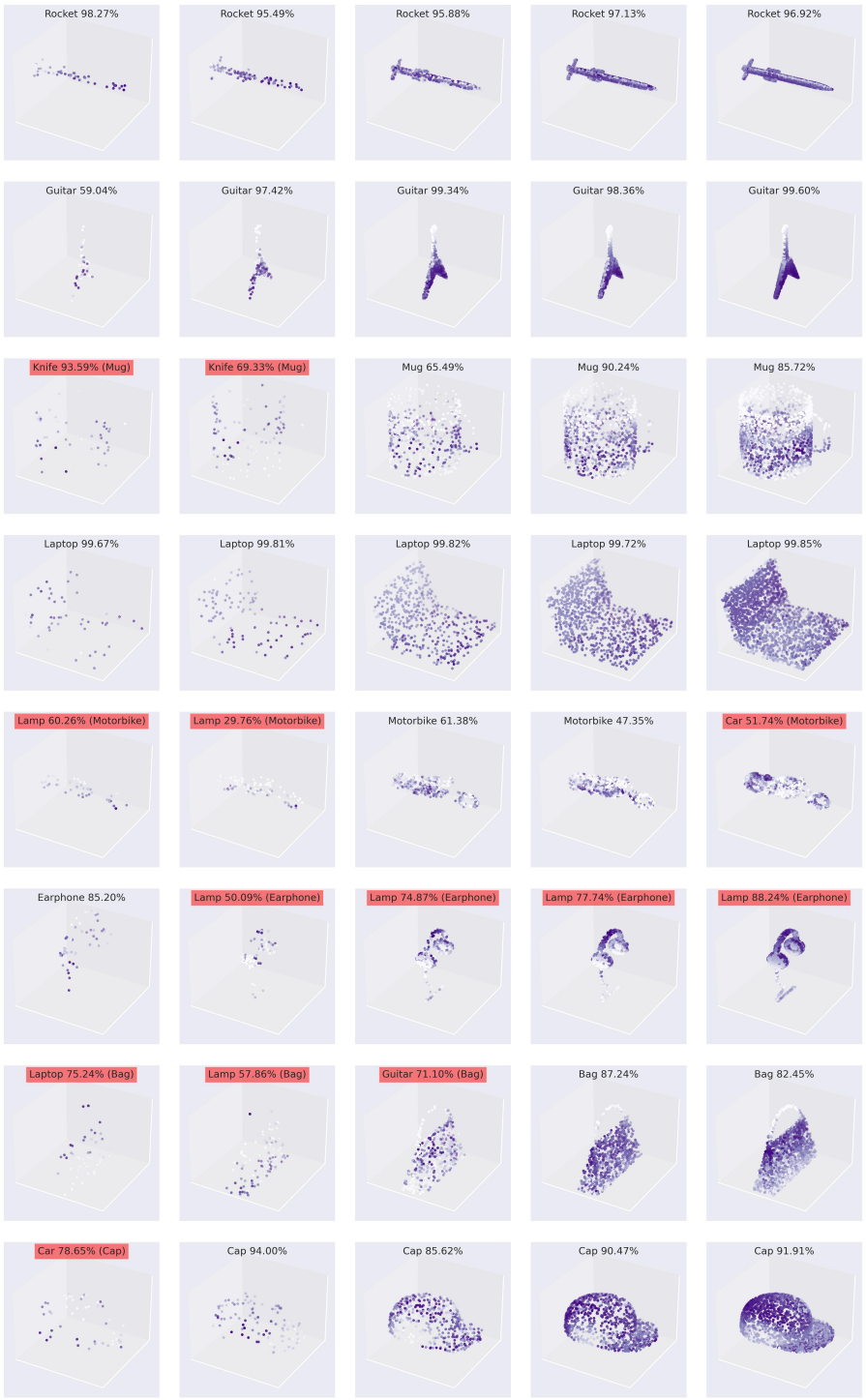

Figure 13: Performer model GradCAM visualization (2/2).

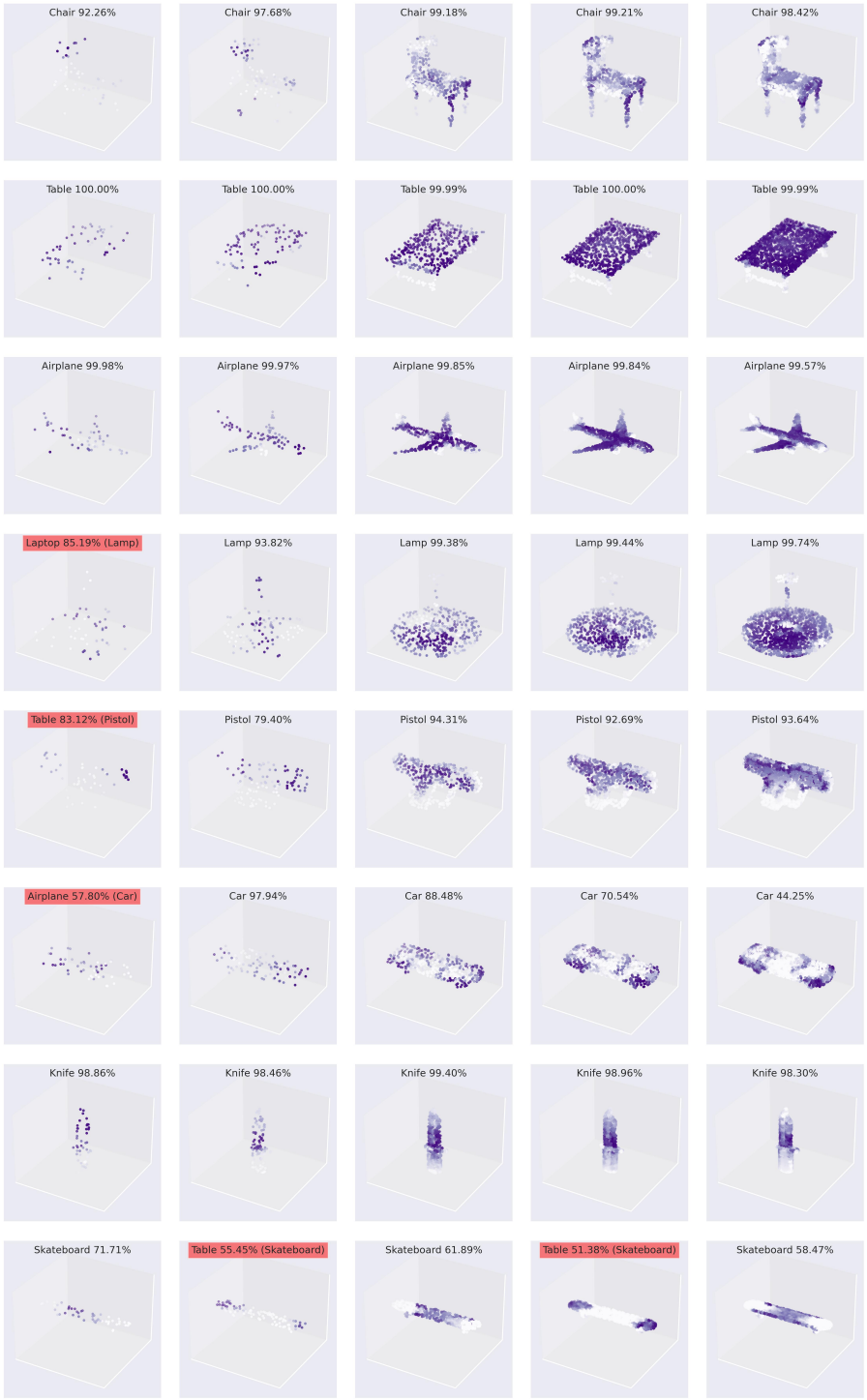

Figure 14: Graph attention model GradCAM visualization (1/2).

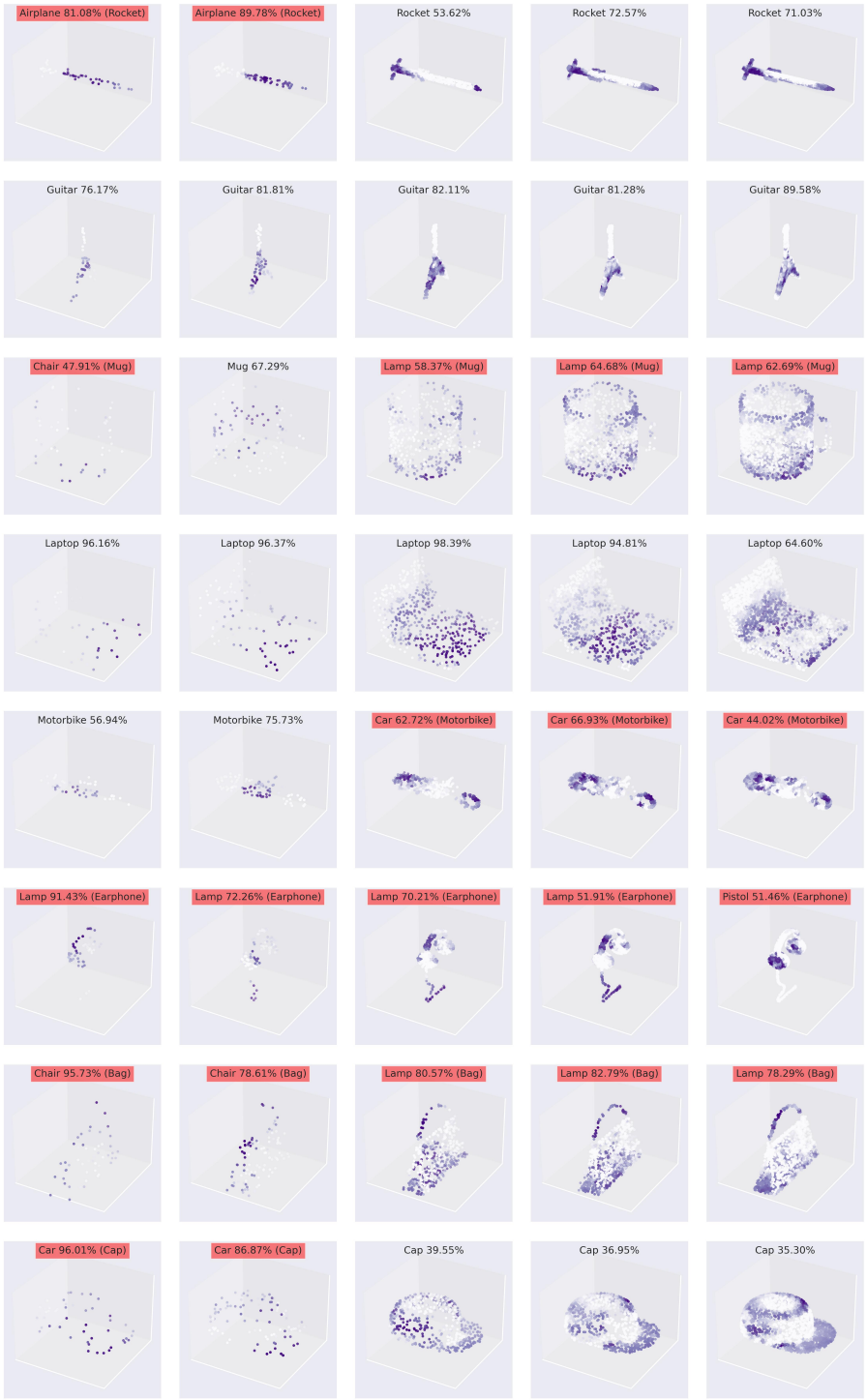

Figure 15: Graph attention model GradCAM visualization (2/2).