# 3D Point Cloud Feature Explanations Using Gradient-Based Methods

Ananya Gupta, Simon Watson, Hujun Yin
Department of Electrical and Electronic Engineering
The University of Manchester
Manchester, UK
{ananya.gupta, simon.watson, hujun.yin} @manchester.ac.uk

***Abstract*—Explainability is an important factor to drive user trust in the use of neural networks for tasks with material impact. However, most of the work done in this area focuses on image analysis and does not take into account 3D data. We extend the saliency methods that have been shown to work on image data to deal with 3D data. We analyse the features in point clouds and voxel spaces and show that edges and corners in 3D data are deemed as important features while planar surfaces are deemed less important. The approach is model-agnostic and can provide useful information about learnt features. Driven by the insight that 3D data is inherently sparse, we visualise the features learnt by a voxel-based classification network and show that these features are also sparse and can be pruned relatively easily, leading to more efficient neural networks. Our results show that the Voxception-ResNet model can be pruned down to 5% of its parameters with negligible loss in accuracy.**

## I. Introduction

Deep neural network (DNN) models are increasingly being used in a number of fields from medical diagnosis [1] to autonomous driving [2] due to their ability to learn meaningful abstractions from data and their successes in many vision tasks. Such models were initially treated as black box operators, but as their popularity has increased, so has the need to make these models interpretable and explainable [3]–[5].

Explainability is important to gain user trust in areas such as medical diagnosis where machine learning is being used for applications such as cancer prediction [3]. Interpretations are also important for identifying biases in models [4] and can be used for extracting insights and debugging models [5]. Driven by these reasons, there has been a lot of work done on the interpretability and explainability of DNNs for image based tasks, and to a lesser extent, language models. We refer readers to [6] for a more detailed review on methods for interpretability.

*Interpretability* can be defined as the degree to which a human can understand the cause of a decision. It is the mapping of an abstract concept such as a model's parameters into a domain that can be understood by humans [6]. An example of this would be feature optimisation where given an output neuron, the input image is optimised such that the activation of said neuron would be maximised [7].

A. Gupta is funded by the President's Doctoral Scholarship from the University of Manchester and the ACM SIGHPC/Intel Computational and Data Science Fellowship.

*Explainability* is a closely related topic to interpretability. Whereas interpretability focuses on abstract concepts, explainability is the identification of relevant features in the interpretable domain that are useful for attaining a specific decision such as identifying the input pixels that are important for the decision of a classification algorithm. A large number of explainability approaches are gradient-based and produce sensitivity maps or saliency maps [7]–[9]. These two terms are used interchangeably in literature, but for the purposes of this work, we will assume the definition given here.

*Saliency maps* in computer vision are used to represent the most noticeable pixels in an image [10]. In the context of model explainability, saliency maps denote the pixels that are deemed important for the decision of the model under consideration [7].

Features learnt from 2D data can be visualised and intuited as images [11]. However, 3D data is not necessarily as intuitively understood. In this work, we explore features learnt by 3D networks as a means of explainability for such networks. More specifically, our contributions are as follows:

- Methods developed for obtaining saliency maps from image data are extended to deal with 3D point cloud and voxel data.
- This is the first work that analyses input features that are deemed important to 3D classification networks.
- The filters learnt by a 3D voxel-based network are visualised and it is shown that they are inherently sparse and can be pruned efficiently with minimal loss in accuracy, leading to a smaller, more efficient network.

### A. Models and Data Types

3D data can be represented in a number of formats such as point clouds, wireframes, surface models and solids. For the purposes of this study, we limit our focus and experiments to point cloud data[1] and voxel data.

Point clouds obtained from LiDAR scanners are unordered point sets with non-uniform density. The point density depends on the sensor scanning pattern and the distance of the surface being scanned from the sensor head. These point clouds can be converted into a uniform voxel format. Voxels are 3D equivalents of pixels, where the space under consideration is

[1]The kind of data obtained from laser scanners.

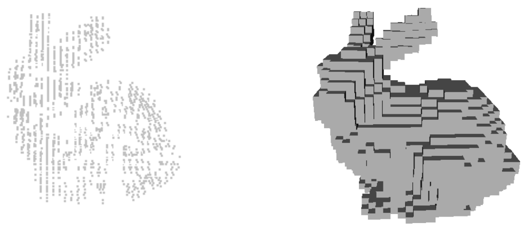


Fig. 1: Stanford Bunny [14]. *Left*: Point cloud representation. *Right*: Voxel representation

divided into a 3D grid and each volumetric element of the grid is known as a voxel. Voxels can be seen as a special case of point clouds with uniform density and quantised dimensions. An example of these two representations are shown in Fig. 1.

We choose popular classification models designed for these data types for further investigation: Voxception-ResNet(VRN) [12] for voxel data and PointNet++ [13] for point cloud data.

## II. Related Work

### A. Explainability Methods

Explainability is a fast expanding area of research with a number of different sub-areas. Popular approaches to explainability of DNN models include creating a saliency map to identify and highlight the important areas in the input space [15] and creating a proxy model which has similar behaviour to the original model but is easier to explain [16].

Perturbation methods such as LIME [17], IME [18] and EXPLAIN [19] are often used to create proxy models [20]. These methods are model-agnostic and usually perturb the neighbourhood of an input space to observe the effect of the perturbations on the output. EXPLAIN and IME are based on the premise that ”hiding” some feature or a set of features in the input space can be used to identify the contribution of the aforementioned features to the decision process. EXPLAIN computes the contribution of each feature individually, which has the disadvantage of missing connections between input variables. IME deals with this issue by computing the importance of all subsets of the feature space. However, this leads to the issue of exponential time complexity.

LIME explains the prediction of a classifier by approximating it with a locally interpretable model around the prediction. It presents the interpretation as an optimisation problem and hence avoids the exponential time complexity issue. An occlusion-based approach was also popularised by Zeiler and Fergus [11] where parts of the input were masked and the output decision was computed on a number of such inputs to obtain the importance of a specific input feature. However, similar to the methods described previously, this method was very slow especially as the input space grew large.

Saliency mapping methods are often used for attribution analysis [15]. They are typically gradient-based and are relatively straightforward to compute using backpropagation. They are faster than perturbation-based methods, which typically require a single forward and backward pass through the network. The gradient of the output class score with respect to the input pixels can be visualised as a heatmap where the highest gradient gives the most important pixel since the least change in that pixel would cause the largest change in the output value [7]. A number of different techniques such as Guided Backpropagation [8] and Integrated Gradients [9] build on this premise and have some differences in how they propagate gradients which are detailed further in Section III. These methods have been used for further analysis of neural networks for 3D data since, as pointed out in [9], they are immediately applicable to existing models and provide intuitive explanations.

There are a number of other backpropagation methods. Layerwise Relevance Propagation [21] was shown to be equivalent within a scaling factor to the element-wise product of the gradient and input [22]. DeepLift [23] assigns an attribution to each input feature based on the relative activation of a reference input. Deep Taylor Decomposition [24] produces sparse explanations but assumes no negative evidence, only showing positive attributions which is not necessarily a valid assumption [15].

### B. 3D Feature Analysis

There has been limited related work on analysing 3D features. Some previous work on voxel classification visualised the average surfaces learnt by certain neurons of their model and showed that the initial layers of their model activated mostly on simple surfaces and corners while later layers had high responses for more complex shapes [25]. The authors of PointNet++ visualised point cloud patterns learnt by the initial neurons in their network by searching for points in a unit sphere that activated the neurons the most [13].

FoldingNet [26] was designed as an interpretable model for unsupervised learning where a 2D grid was folded onto a 3D object surface for reconstruction. The authors expressed this as an intrepretable model since the folding could be seen as a granular warping.

## III. Attribution Maps

The formulation for vanilla gradients is given by Equation 1. These gradients can be visualised as a heatmap or a saliency map [7] and are similar to the output from deconvolutional networks [11].

$$Grad_i = \frac{\partial F(x)}{\partial x_i} \tag{1}$$

The input is given by $x$ and each element of the input is indexed by subscript $i$. $Grad_i$ is the gradient attribution of element $x_i$ and $F$ is the function of the neural network.

**Saliency maps** zero out gradients during the backward pass if the inputs coming into the rectified linear units (ReLU) during the forward pass are negative. On the other hand, **deconvolutional networks** zero out gradients from the ReLU during the backward pass only if those incoming gradients during the backward are negative.

**Guided Backpropagation** [8] combines the approaches from saliency maps and deconvolutional networks. In this method, the gradient is backpropagated through a ReLU only if the ReLU is switched on (input is non-negative) and the gradient during backward propagation is also non-negative.

Such saliency maps have a lot of noise and a number of methods have been proposed to refine them. A straightforward method to improve the sharpness of the attribution map is to use the element-wise product of the gradient and the input [22].

**Integrated Gradients** [9] computes the average of all the gradients along the straight line path between a baseline, $x'$, and the input, $x$, given by Equation 2. In the case of an image, the baseline can be a zero image. This method has the desirable property of *completeness* [9], which implies that the attributions add up to the difference between the target and the baseline outputs.

$$IntGrad_i = (x_i - x'_i) \cdot \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} \partial\alpha \quad (2)$$

## IV. 3D Features and Network Pruning

Learnt voxel features can be visualised as 3D filter maps. Since 3D spaces are inherently sparse, we hypothesise that discriminative features for voxel-based networks should also be sparse. However, some 3D CNNs are dense extensions of 2D networks for 3D structures and do not take into account the sparse nature of 3D data. Hence, we took inspiration from pruning methods to test the sparse nature of dense 3D networks.

Pruning methods are broadly divided into fine-grained and coarse-grained pruning [27]. The former is based on pruning individual weights to make the DNNs sparse, while the latter is based on pruning entire kernels or channels. We have extended a popular fine-grained pruning method called Dynamic Network Surgery (DNS) [28] to work with 3D filters to test our hypothesis. The formulation of this pruning method is given below.

The weight tensor representing the weights in layer $k$ is given by $W_k$. An additional tensor $T_k$ is defined which has the same dimensionality as $W_k$ and is a binary mask matrix to indicate if the corresponding weights in $W_k$ have been pruned or not.

The optimization problem is summarised as :

$$\min_{W_k,\, T_k} \mathrm{L}(W_k \circ T_k) \; s.t. \, T_k = \, \mathbf{h}_k(W_k), \quad (3)$$

where L is the loss function, $\circ$ represents the Hadamard product. The function $\mathbf{h}_k$ is used to determine the importance of the weights. In our experiments, following the work in [28], $\mathbf{h}_k$ is the absolute value of the weights. Hence, the smaller the absolute value, the less important the weight parameter.

Hence, Equation 3 looks to minimize the loss by optimising the values of $W_k$ and $T_k$ and is an N.P. hard problem. In this case, these values are optimised iteratively where the weight updates are given by a slight modification of the standard gradient descent algorithm during backpropagation in order to incorporate the weight mask as follows:

$$W_k \leftarrow W_k - \, \beta \frac{\partial}{\partial(W_k T_k)} \mathrm{L}(W_k \circ T_k), \quad (4)$$

where $\beta$ represents the learning rate.

This update carries through for all weights, including the ones where the corresponding value in the weight mask is zero, allowing the weight mask to be updated by removing certain values and restoring others during the next forward pass operation as follows:

$$\mathbf{h}_k(W_k) = \begin{cases} 0 & if \; t_k > |W_k| \\ 1 & if \; t_k < |W_k| \end{cases} \quad (5)$$

where the threshold $t_k$ is defined using the mean and variance of the absolute values of the weights in layer $k$.

## V. Experimental Details

The Pointnet++ and VRN models were trained according to the details given by the original authors of the respective papers. The VRN model was reimplemented in Pytorch where the original implementation of Pointnet++ was used for all experiments.

Following the implementation in the original papers, the Modelnet40 models were voxelised to a resolution of 32x32x32 for VRN and 1024 points were sampled on the surface of each model for Pointnet++.

The baseline was assumed to be an empty voxel space for integrated gradients, with 50 steps between the baseline and the input.

## VI. Results

### A. Attribution Maps

Examples of attribution maps for Pointnet++ obtained using vanilla gradients, guided backpropagation and integrated gradients as outlined in Section III are shown in Figure 2. As can be seen from the figure, vanilla gradients attribute more importance to edges and corners than they do to flat surfaces, though the relevance of points along surfaces is not uniform, leading to the assumption that these attribution maps are fairly noisy.

The maps obtained using guided backpropagation are somewhat more uniform, with higher saliency attributions given to highly discriminative features, such as the stand in the case of a television and the tap in the case of a bathtub. The clearest results are achieved with the integrated gradients, which identify corners and edges and do not give much importance to flat surfaces.

The attribution maps for VRN are shown in Figure 3. As can be seen, the vanilla gradient maps are a lot noisier in this case as compared to those for Pointnet++. This is due to the fact that the voxel inputs encode free space along with occupied space

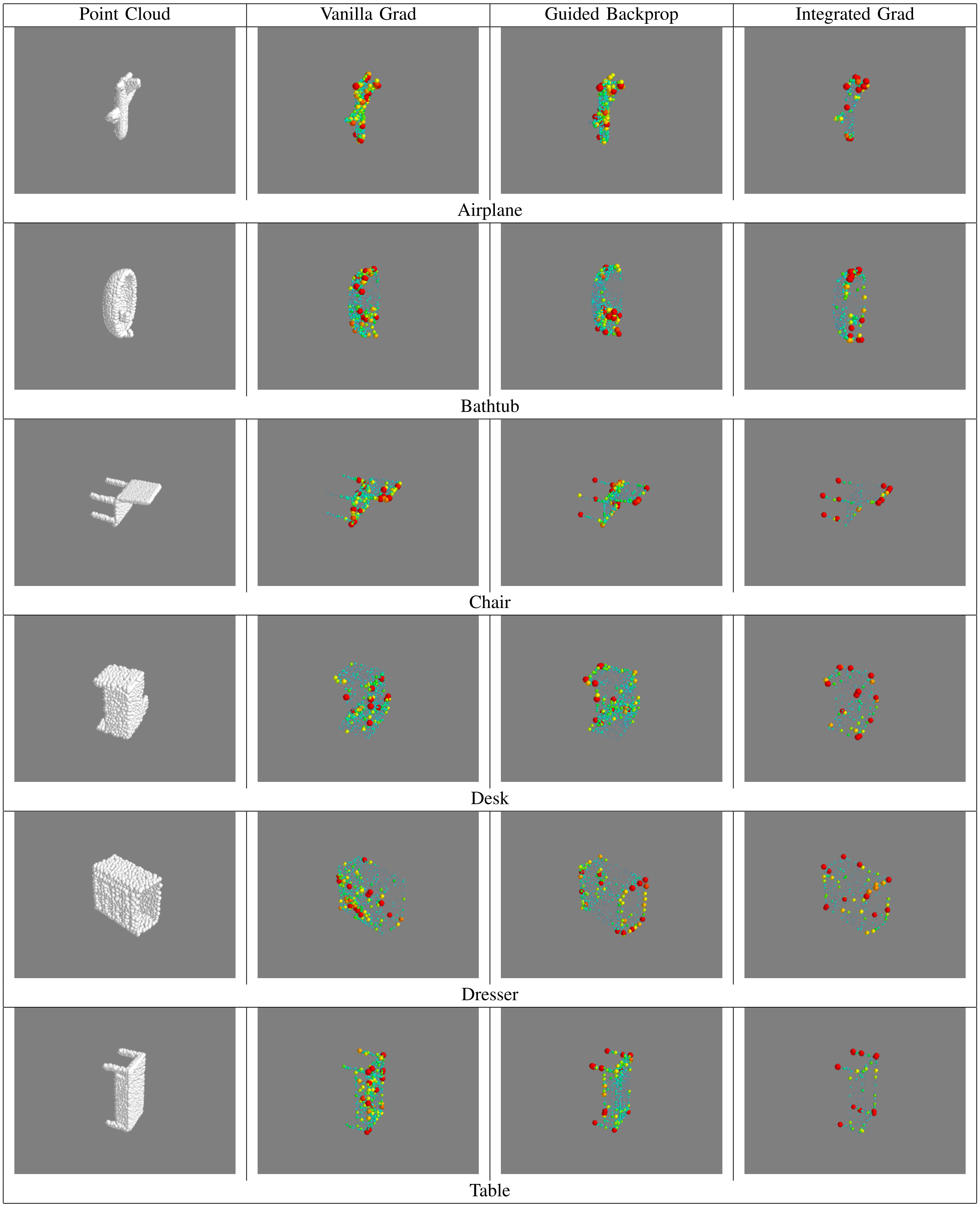


Fig. 2: Visualisation of attribution maps for Pointnet++. The attributions are given as a heatmap by Red (large) to Blue (small).

TABLE I: 3D Weight Pruning Results for VRN

| | # Parameters | Params Left (%) | Accuracy (%) |
|---|---|---|---|
| Original Model | 13,829,792 | 100 | 87.77 |
| Prune, no finetune | 728,092 | 5.26 | 62.39 |
| Prune, 1 epoch tuning | 700,720 | 5 | 87.18 |

while the point clouds only encode occupied space. Hence, the vanilla gradients in the voxel space are also, in a way, affected by the empty voxels. In order to make these maps less noisy, we show the element-wise product of the gradients and input as the 'Masked Vanilla' output in Figure 3.

These masked maps can show the salient features in the input space more clearly. For example, in the case of the cup, the handle and the shape of the cup are important for the classification. It is interesting to compare the masked gradients with the results of the integrated gradients, where the most important voxels seem to overlap. The latter does deem some voxels in the unoccupied space as being important. However, in contrast to vanilla gradients, these unoccupied voxels are given almost negligible importance.

### B. Pointnet++ Error Analysis

The PointNet++ model achieved 90.2% accuracy on the ModelNet40 test dataset when trained according to the parameters given by the original authors. The confusion matrix for the test set is shown in Figure 4.

The confusion matrix shows that the major errors are between classes that have a fair amount of semantic overlap, such as plants being recognised as flower pots and tables being labelled as desks. Some of these misidentified objects are shown in Figure 5 along with their saliency maps based on the Integrated Gradients. From these images, it can be seen that the mistakes made by the model could have been also made by humans since these classes are fairly similar.

### C. Features Learnt by Voxel Networks

Some of the features learnt by VRN have been visualised in Figure 6 where the size of each element denotes the relative absolute value of the weight. The figure also shows the same features after pruning and finetuning. The difference between the pruned features with and without finetuning is minimal and has also been shown.

From the results in Table I, it can be seen that pruning the network down to almost 5% of its parameters decreased the accuracy by 25% but finetuning for only 1 epoch brings the accuracy back up to the original results even with the pruned model. This is contrary to the process with image based models which require finetuning in the order of over 10k iterations [28]. This seems to support the hypothesis that the 3D features learnt are fairly sparse and removal of small weights does not overly affect the performance. The visualisations in Figure 6 also verify this as it can be seen that the difference between the original model and the pruned and finetuned model is minimal.

## VII. Conclusions

This work is an initial study on explainability of neural network models for 3D data. To this end, popular attribution methods currently used with image data have been extended to deal with point cloud and voxel data. It has also been shown that the features learnt by voxel based networks are sparse and can be pruned easily with little finetuning required.

Our results show that edges and corners are considered as important features by gradient-based methods, while planar surfaces do not contribute as much to the classification decision. Vanilla gradients are fairly noisy but the use of integrated gradients makes the attribution maps more uniform. In the case of voxel-based inputs, vanilla gradients attribute a lot of importance to empty space. These attributions become a lot more sensible when masked gradients are used, or with the use of integrated gradients.

We have visualised the learnt features of the voxel classification network and showed the sparsity of these learnt filters. The network can be pruned down to 5% of its original number of parameters with minimal loss in accuracy and only one epoch of finetuning; as compared to image based networks which require over 10k iterations of iterative pruning and finetuning. We believe this is due to the fact that 3D data is inherently sparse and hence the features learnt for this kind of data are also sparse.

This work can be extended in a number of directions. A natural extension of this work would be to use the insights gained from the gradient-based models to prune DNNs during training time rather than as a post-processing step. Some other relatively straightforward extensions include testing 3D models using some perturbation-based methods such as the ones described in Section II-A. Another important area of research is the systematic quantification of the extracted explanations. We refer readers to [15] for ideas on the same.

## References

[1] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, jan 2019.

[2] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother, "Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2017, pp. 1025–1032.

[3] Y. Xiao, J. Wu, Z. Lin, and X. Zhao, "A deep learning-based multi-model ensemble method for cancer prediction," *Computer Methods and Programs in Biomedicine*, vol. 153, pp. 1–9, jan 2018.

[4] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & Explorable Approximations of Black Box Models," *arXiv preprint*, jul 2017.

[5] G. Cadamuro, R. Gilad-Bachrach, and X. Zhu, "Debugging machine learning models," in *ICML Workshop on Reliable Machine Learning in the Wild*, 2016.

[6] G. Montavon, W. Samek, and K. R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing: A Review Journal*, vol. 73, pp. 1–15, feb 2018.

[7] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv preprint*, dec 2013.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," in *ICLR (workshop track)*, dec 2014.

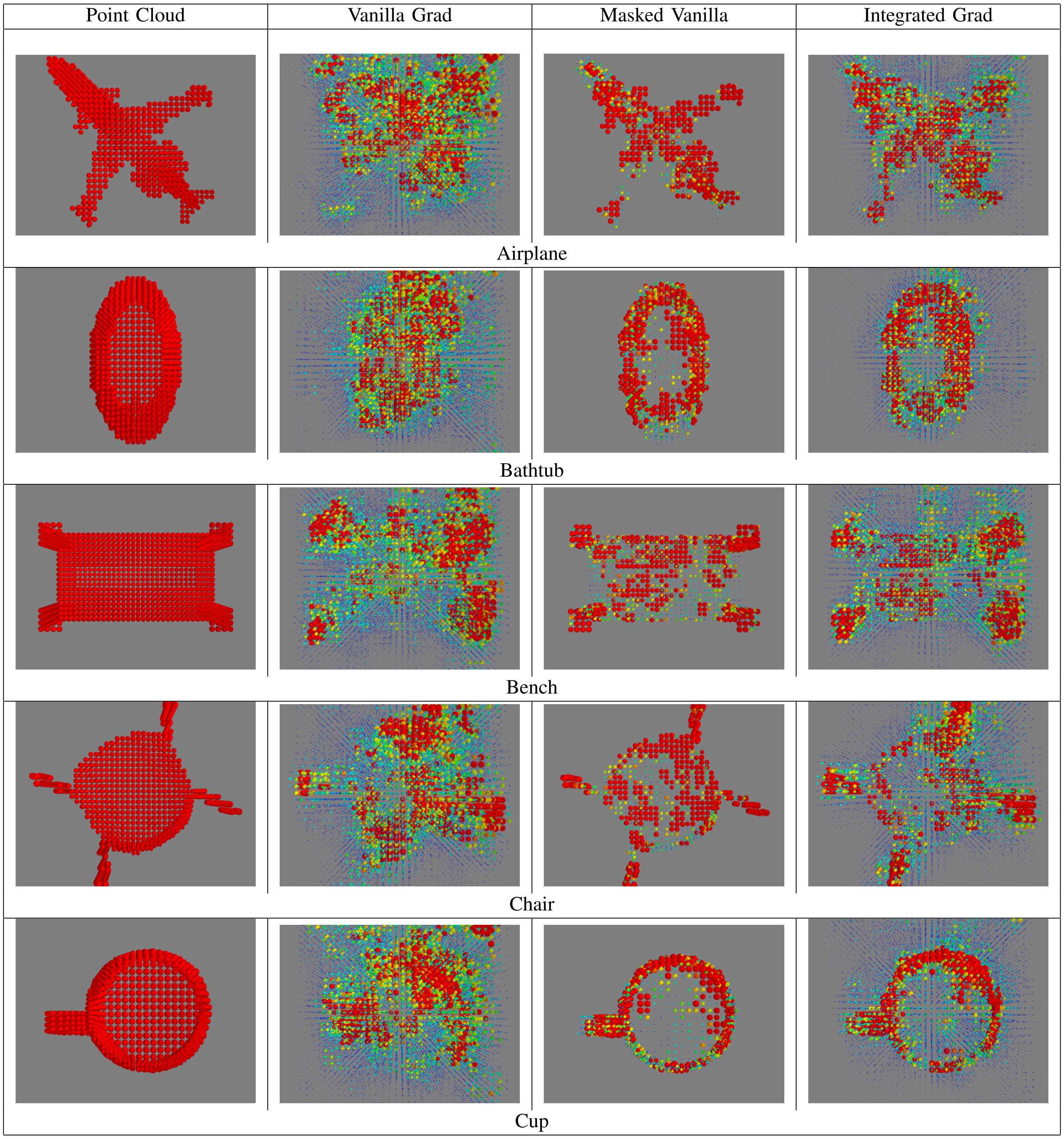


Fig. 3: Visualisation of the attribution maps for VRN.

[9] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *Proceedings of the 34th International Conference on Machine Learning*, mar 2017.

[10] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[11] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *European Conference on Computer Vision*, vol. 8689, nov 2014, pp. 818–833.

[12] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and Discriminative Voxel Modeling with Convolutional Neural Networks," in *3D Deep Learning Workshop, NIPS*, aug 2016, p. 9.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Neural Information*

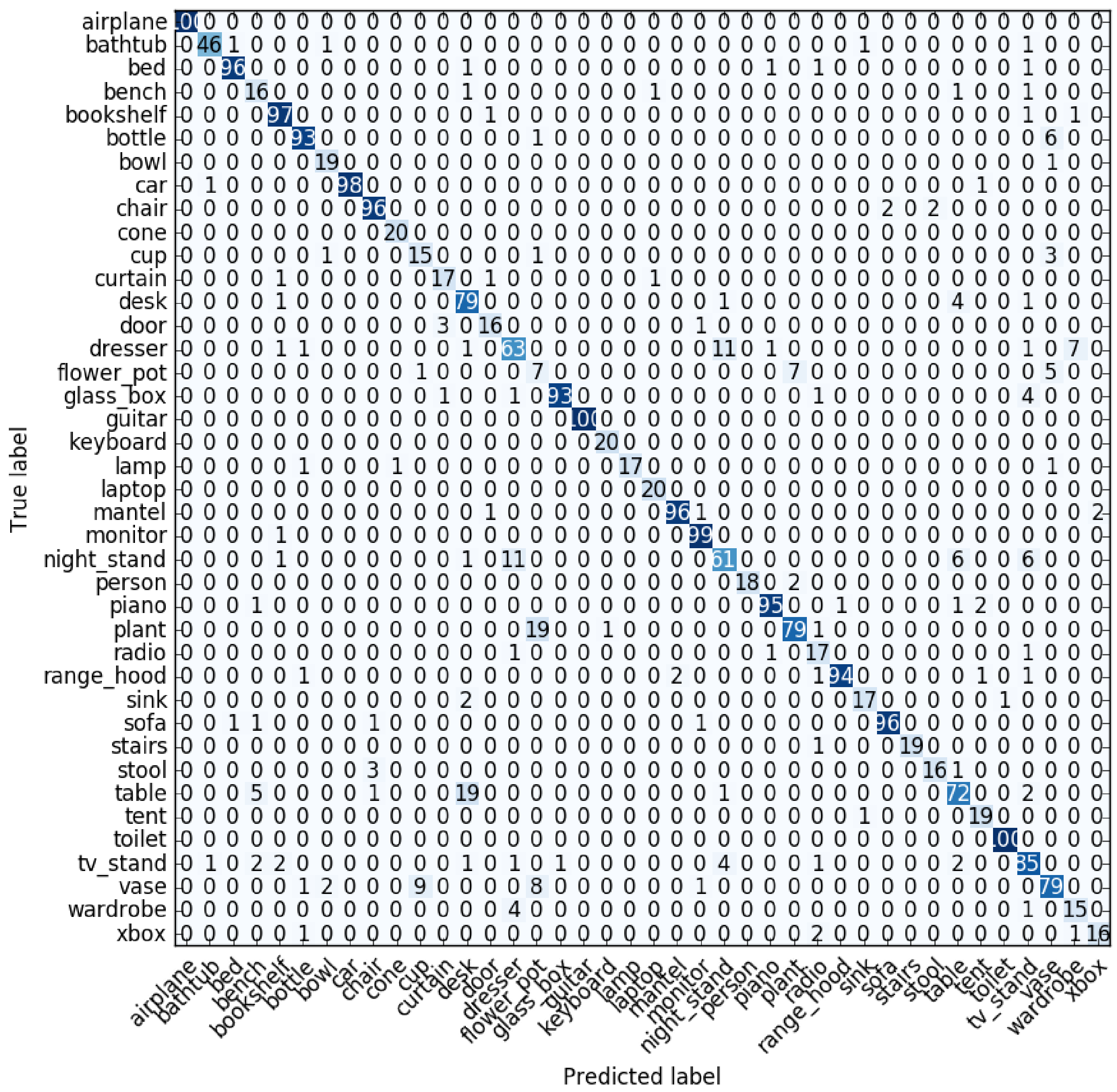


Fig. 4: Confusion matrix of Pointnet++ results on ModelNet40.

*Processing Systems*, pp. 601–610, jun 2017.

[14] S. U. C. G. Laboratory, "Stanford Bunny," 1993.

[15] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for Deep Neural Networks," *arXiv preprint*, nov 2017.

[16] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, oct 2018, pp. 80–89.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, 2016, pp. 1135–1144.

[18] E. Štrumbelj, I. Kononenko, and M. Robnik Šikonja, "Explaining instance classifications with interactions of subsets of feature values," *Data and Knowledge Engineering*, vol. 68, no. 10, pp. 886–904, oct 2009.

[19] M. Robnik-Šikonja and I. Kononenko, "Explaining classifications for individual instances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 5, pp. 589–600, 2008.

[20] M. Robnik-Šikonja and M. Bohanec, "Perturbation-Based Explanations of Prediction Models," in *Human and machine learning*, 2018, pp. 159–175.

[21] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-Wise Relevance Propagation: An Overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 193–209.

[22] A. Shrikumar, P. Greenside, and A. Kundaje, "Not Just a Black Box Learning Important Features Through Propagating Activation Differences," *arXiv preprint*, apr 2017.

[23] ——, "Learning important features through propagating activation differences," in *34th International Conference on Machine Learning, ICML 2017*, vol. 7, 2017, pp. 4844–4866.

[24] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, may 2017.

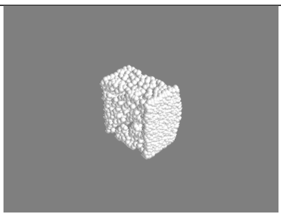
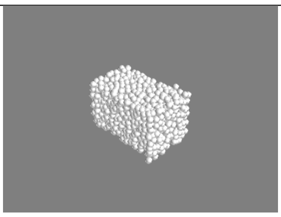
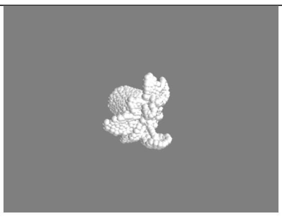
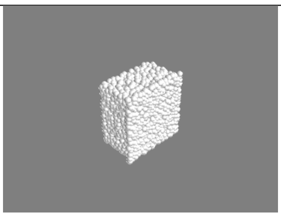
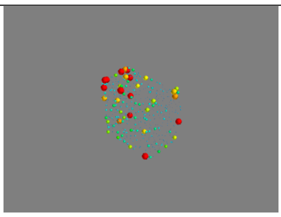
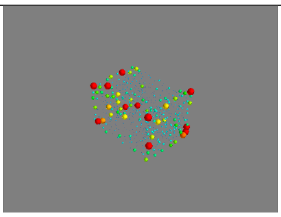
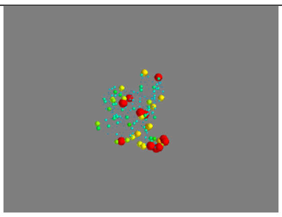
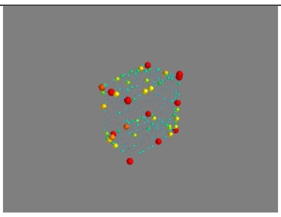
| | | | | |
|---|---|---|---|---|
| Point Cloud |  |  |  |  |
| Integrated Grad |  |  |  |  |
| Original Class | Dresser | Night Stand | Plant | Table |
| Classification | Night Stand | Dresser | Flowerpot | Desk |

Fig. 5: Visualisation of the incorrectly classified point clouds.

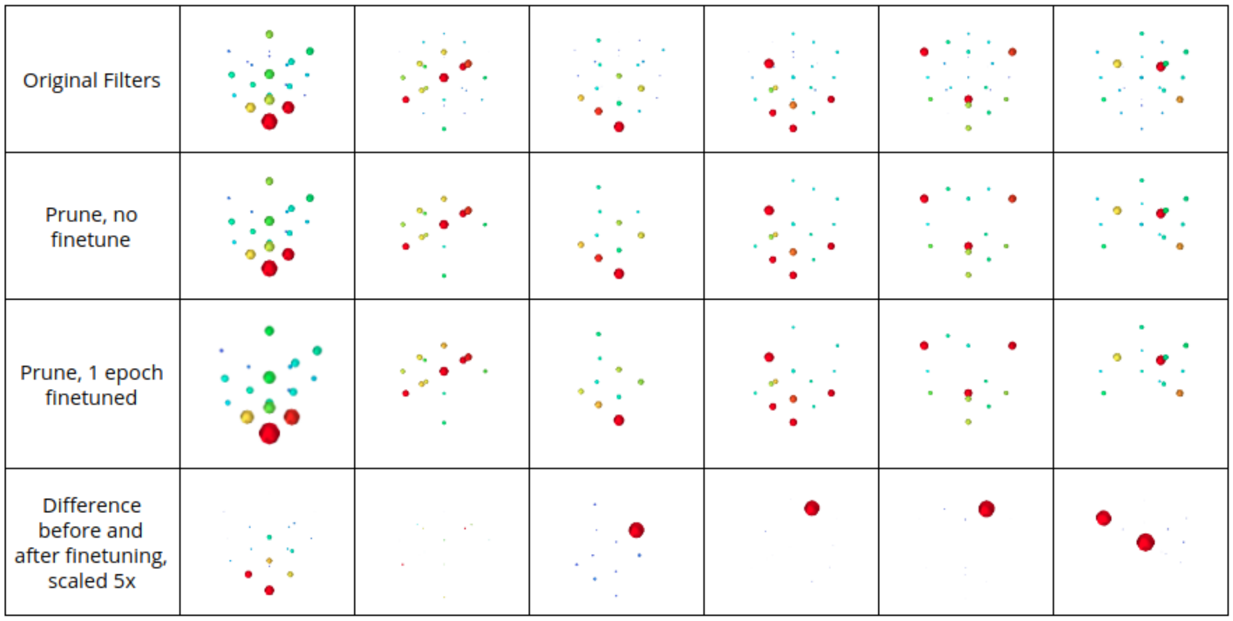


Fig. 6: VRN features where each column denotes one feature from the first layer of the network. From top to bottom, the features are as follows: *top*: original features, *second row*: pruned, *third row*: finetuned, *bottom*: 5x scaled difference between the pruned and finetuned version. It can be seen that there is very little difference between the weights of the pruned network and original network because the learnt features are inherently sparse.

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1912–1920, 2015.
[26] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, dec 2018, pp. 206–215.
[27] J. Cheng, P.-S. Wang, G. Li, Q.-H. Hu, and H.-Q. Lu, "Recent advances in efficient computation of deep convolutional neural networks," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 64–77, jan 2018.
[28] Y. Guo, A. Yao, and Y. Chen, "Dynamic Network Surgery for Efficient DNNs," in *Neural Information Processing Systems*, 2016.