UNIVERSITY OF AMSTERDAM

MSc ARTIFICIAL INTELLIGENCE

TRACK: MACHINE LEARNING & COMPUTER VISION

MASTER THESIS

# Towards unsupervised object Classification and Prediction in 3D through Semantic Sampling


by

RAY FEATHER HOLT ORMESHER

10967680


August 22, 2017

36 EC
02/2017 - 08/2017

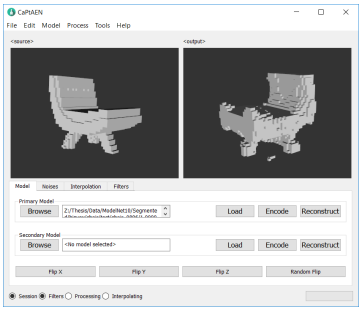

*Supervisors:*
Dr. S. Karaoglu
Prof. Dr. T. Gevers

*Assessor:*
Dr. M.W. van Someren

UNIVERSITEIT VAN AMSTERDAM & 3DUNIVERSUM

## Contents

# Towards unsupervised object Classification and Prediction in 3D through Semantic Sampling




**Ray Ormesher**[1,2]

**Abstract**

In this thesis I will investigate the potential use of pre-existing Volumetric Variational Auto-Encoder architectures for object in-filling and de-noising. From the experiments presetned here it can be seen that even with relatively simple architectures, complex and varied noises can be repaired by learning generative latent spaces from training with data augmentation. For further improving the VAE's predictive abilities, I propose two novel redefinition of the Variational Bayes Auto-Encoder architecture for management of partial, semantically scaled input samples. The Located-VAE (LVAE) and Prior-VAE (PVAE) are extensions of variational reconstruction networks that attempt to connect real-world sliding window object segments to a latent space of known 3D objects for classification and prediction. Their predictive abilities are shown visually through use of the Classification and Prediction through Auto-Encoder Network (CaPtAEN) application for basic reconstruction tasks, as well as reconstruction with varying noise qualities at input. The classification abilities are demonstrated empirically through comparison of latent space representations of segments taken from the same object. Finally, we argue that although voxel models are visually interesting to work with, the computational complexity and massive sparsity are prohibitive for working with high-resolution models and prevent learning of structured high-level 3D filters. The lack of filter descriptiveness is visually explained using the application presented in this work.

**Keywords**

Unsupervised Learning, Classification and Prediction, Variational Auto-Encoder, Learnt Priors, Semantic Sampling

## 1 Introduction

### *1.1 Motivation*

Variational Auto-Encoders (VAE's) are a form of reconstructing Neural Network architecture that learns an approximate probability distribution using the Stochastic Gradient Variational Bayes (SGVB) estimator. In practice this comprises three main parts: the Encoder, a low-dimensional Sampling distribution and the Decoder. The Encoder maps an input to the parameters of a Multivariate Gaussian distribution on the latent space. The sampling is achieved by a random perturbation of the mean of the sampling distribution with respect to its predicted variance. The Decoder then attempts to reconstruct the original input from the estimated latent embedding. It is this reconstruction that classes VAE's, as well as Auto-Encoders in general, as Generative networks, networks that generate known or original but valid data.

Recent advances with Generative and Volumetric networks have shown the many qualities these approaches can learn with respect to 3D Computer Vision. These include predicting 3D from 2D[12, 17], learning latent distributions of Voxelized shapes[3, 32] and Generative De-Noising[30]. In this work, baseline VAE's are analyzed alongside some altered reconstruction architectures, with respect to bridging the gap between noisy, real-world use cases for unsupervised generative models and abstract reconstruction tasks.

Generative methods have shown they can produce authentic, unique instances derived from learnt latent spaces[57]. These latent spaces can even be used to perform complex operations over the distribution of objects recognized to create unseen, unique and authentic looking objects. They have also shown impressive de-noising and in-filling capabilities, as in [30, 22] and figure 1, which show a lot of promise for development of real-world application that can work outside of noiseless conditions, such object classification by voxel modelling[3].

Volumetric networks are quickly achieving high object classification scores[3, 26, 46, 19] with architectures mimicked from 2D networks. However, for both 2D and

[1] 3DUniversum, The Netherlands
[2] Universiteit van Amsterdam

**Corresponding author:**
Ray Ormesher, 3DUniversum Amsterdam Science Park, Matrix 3...
Email: ray.ormesher@student.uva.nl

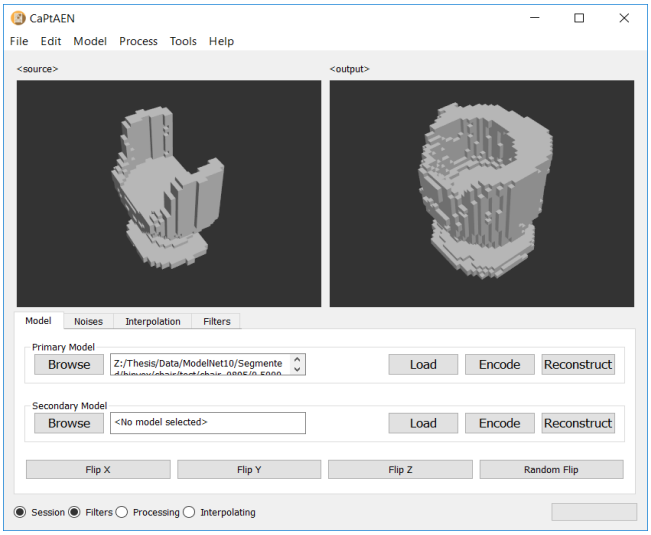



**Figure 1.** Generative capabilities of conventional VAE

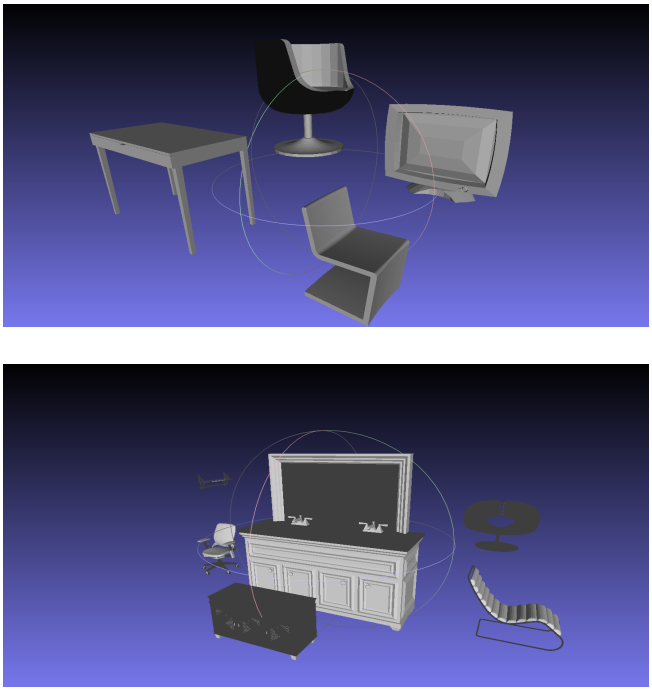


**Figure 2.** Top: a group of objects scaled uniformly, Bottom: a group of objects scaled semantically

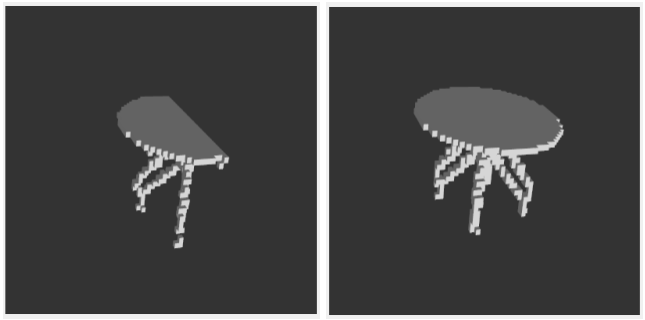


**Figure 3.** An example input/output pair for training

3D Deep Learning to be applied in real-world situations the matter of scale must be addressed. So far, in 2D, this issue has been largely handled by the use of max-pooling and strided convolutions[15, 50, 49] to process the inputs at multiple scales. For volumetric data this task becomes even more challenging as the number of symmetries and translations increases exponentially. This requires that implementations are made to be computational efficient[52, 41], they maximize resources to minimize training time, but also have theoretical efficiency, such as equivariant convolutions and filters[7]. Although these approaches still struggle to detect very large or very small objects due to changing resolutions. The relevance of an objects scale can be seen in figure 2, it is clear the scale is key for recognition and classification of objects for in the wild 3D problems.

To fuse the de-noising and recognition tasks involved in 3D computer vision requires careful handling of the data. The large variances must be retained to improve recognition, or separability in the latent space, without heavy reliance on expensive manual preprocessing. So it is reasonable to ask how we can use architectures with rigid input and apply them to widely varying data while incorporating as much semantic information as possible. To this end, the VAE's ability to cope with wide varieties of noise will be explored and then the input will be fixed to represent a real-world, semantic, volumetric window. The latter experiments will be to understand if a link can be made between sliding window scanning and the objects to whom contents of the window belong. An example of a possible training pair is given in figure 3, the left segment is a $1m^3$ section of the table on the right

It is reasonable to ask why this segment based analysis should work. Intuitively the task is tractable as most people, when shown a section of an object in real-life such as the corner of a sofa, can differentiate what object it might belong to even if the whole object is not known. Unfortunately, given that the surface texture and shading are not included with voxel models the task becomes much more difficult. These details are key factors for our understanding and recognition of a given object.

Moreover, the segment to whole training is hindered by the networks scale conversion task as well. This is the requirement of converting the input from a fixed scale input to a variable scale output. However, to encode greater information about the whole objects distribution of parts, two novel architectures are tested. The Prior-VAE (PVAE) attempts to learn a mixture of gaussians prior in parallel to the decoder. The Located-VAE (LVAE) attempts to predict a secondary sampling distribution to represent the distribution of segments from the whole.In PVAE's, the prior is predicted from the latent space representation and is used to evenly distribute more information about the whole object. In LVAE's a secondary latent distribution is estimated for representing the distribution of segments from the object.

Brock et al.[3], is worth note as their volumetric VAE will be used as the baseline model for comparison in the following experiments. The experiments presented here will first validate on ModelNet10 with the reconstruction VAE

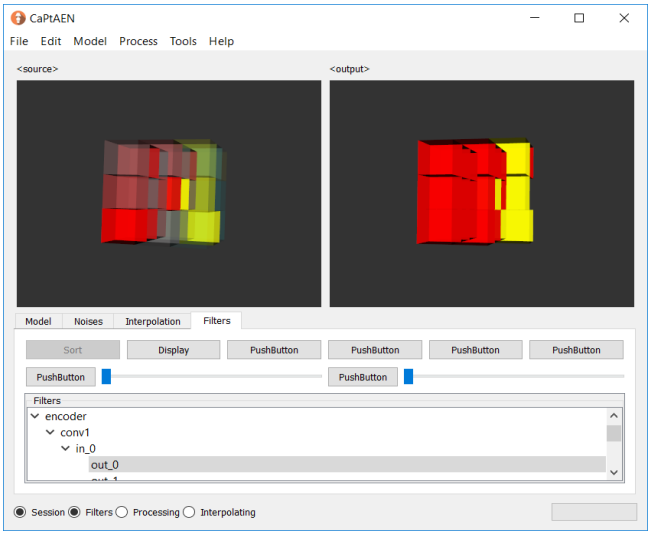



**Figure 4.** A learnt volumetric edge filter displayed in CaPtAEN

and compare to Brock et al. This will serve as a ground-truth for conventional Volumetric VAE's before testing these architectures with further tasks.

Next the task of 3D scanning by sliding window is introduced, a novel semantically linked method for analysis and development of volumetric approaches. This will be in the form of a prediction by segment task on ShapeNet to evaluate its tractability. Followed by a discussion about the possible benefits to object detection by sliding window and options for further research.

Finally, the CaPtAEN application for reconstruction analysis is introduced with some short detail of its capabilities. These include visualization, manipulation and interpolation of Voxel models for use with Auto-Encoders.

## 1.2 Related Work

### 1.2.1 Generative Methods

Variational Auto-encoders[25] have gained increasing popularity in recent works, from de-noising[22] to feature learning[21] and their ability to model probabilistic, generative latent manifolds for 3D data has been discussed[3]. Training VAE's requires using Stochastic Gradient Variational Bayes(SGVB) fully explained in [25]; a variant of SGD, the reconstruction loss is balanced by a Küllbach-Leibler divergence which can be seen as a regularizer for the parameters of the latent space, encouraging the approximate posterior to be close to the prior (a unit multi-variate normal distribution). This has been shown to learn a smooth, interpretable manifold for not only digit reconstruction with MNIST, but also on more complex image distributions such as faces[21]. Due to their discriminative abilities so far, it is reasonable to study how far existing architectures can be pushed with respect to inherent understanding of the subject matter.

In their study of VAE's with respect to voxelized 3D objects, Brock et al.[3] show the completeness and generative properties of the learnt latent space representations. Re-implementing known networks for 3D voxel modelling has given rise to learnt geometric manifolds capable of generating new, unique data points through sampling and interpolation[57, 12]. This shows that the network is not just learning the dataset but learns a complete distribution of objects representations. As it is possible to learn generative manifolds of voxelized objects it is potentionally possible to approximate this manifold through training on batches of semantically linked inputs.

Generative architectures have also gained traction due to their capacity for relating partial/noisy data, in both 2D and 3D, to smooth, linear manifolds for predicting noiseless representations, such as in figure 5. Dai et al.[8] show how well networks handle infilling heavily noised datapoints with the inclusion of a shape prior. Sharma et al[47] showed how Auto-Encoded volumetric data could be accurately reconstructed, even with up to 50% noise. In this project, these de-noising characteristics are expanded on and tested with challenging real-world de-noising tasks.

Most generative methods rely on regularized inputs and outputs. In general, this is handled by means of voxelized meshes for use in processing. Sinha et al.[48] attempt to generate surface from regularized point clouds. This is an interesting approach as it generates higher resolution outputs than voxel models while maintaining smoothness in the latent space. It also allows for re-training an Encoder from new labels, similar to transfer learning and an interesting approach for further research with segment to whole prediction.

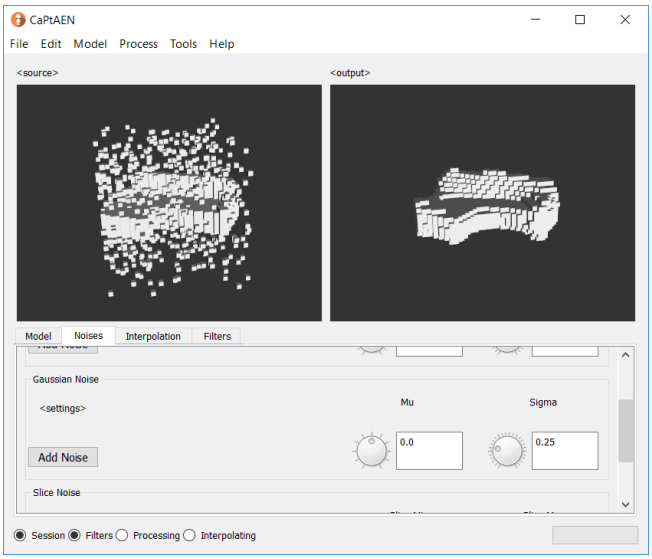



**Figure 5.** VAE voxel de-noising

### 1.2.2 Classification and Prediction

Volumetric classification is a relatively new but already well-studied field. Maturana et al.[33] were among the first to use voxel models as regularized network inputs for classification tasks. More recently, Zreik et al.[63] used volumetric classification networks for segmenting regions in 3D scans. It shows that although voxel grids are sparse,

there is quantitatively enough information for recognition and classical computer vision tasks.

Use of rotation is a common theme in Classification, whether in multi-view 2D[40] or in volumetric[3, 46] architectures, in improving accuracy. Intuitively these approaches make sense as the positioning and viewpoint from which an object is seen, whether it is represented in 2D or 3D, heavily influence the information contained in the data. Recent work by Cohen et al.[7] has shown that equivariant filters can be learnt to avoid multiple instances of symmetrically relatable conventional filters which may help reduce error from rotational variances. In this project multiple orientations are used to enrich the learnt latent space and robustify it to symmetries. It is also hoped that this will improve similarity between encoded segments, even under rotation.

Shape priors are used to boost accuracy by implying likely distributions to inform the latent space of encoder-decoder architectures, such as Dai et al.[8]. This also makes sense because a good understanding of an object can be gained from their general shape, the prior, and the recognition from fine detail can then be handled separately, by the encoder. This provides motivation for the learnt shape priors used in the PVAE architecture.

Grant et a.[16] and Girdhar et al.[12] both attempt to predict 3D volumes from 2D images via Auto-Encoder with modulated latent spaces. These approaches make sense as a 2D image is a projection from 3D space to 2D, so the 2D images can be used to inform the latent space at train time. This can be thought of as trying to infer the shape of the object from a segment, however the segment is only the points visible from a specific orientation. This is similar to the task of predicting a full object only given a segment with real-world dimensions.

In the field of unsupervised learning for classification and prediction, one common approach is the use of U-nets[42, 6, 45, 43] for learning abstract manifolds for comparison of latent representations for segmentation predictions. Cicek et al.[6] find that full, accurate segmentation prediction can be made with only 3 labelled samples, enough data augmentation to express all possible variances and sparse annotation. Similarly, in this project augmentation is used to cover possible alterations in voxel representation.

### *1.2.3 Semantics*

In general, most results reported are based around the use of controlled data such as MNIST[28] or lab condition faces such as the Frey Faces[54][61][55], that abstract away some of the semantically correlated information between data points. This has lead to nice test results however these methods suffer when applied to "in the wild" problems. Data augmentation[10][9][29][60][5][56] is frequently used as a method to combat over fitting but this cannot address issues such as the scale of an object in a scene, which is often indicative of its type, shape and boundaries. This means that object detectors, in 2D and 3D, will fail when testing in the wild due to lack of learning with respect to real-world scale. In 2D, this is in some part managed by processing the image at mutliple resolutions. In 3D processing at different resolutions becomes very expensive, although Moeskops et al.[35] attempt this for 3D MR rain segmentation.

In this work, to combat the varying sizes and scales of objects the input of the network is fixed to represent a $1m^3$ "sliding window" for scanning a a 3D scene. With such an approach it is guaranteed that all sampled inputs will be semantically related by their scale. This will be costly in that many windows will have to be tested for a single indoor scene, but this can be calculated beforehand as the window has a fixed size. This is hopefully one of the further application of this work as a labelling tool by segment analysis. However, if the network learns a purely 1-input, 1-output matching then result would be less applicable in real-world situations. Tested segments may never be similar enough to the training input such that the output was meaningful.

Another big challenge when considering unsupervised learning for in the wild use is that trained CNN's have fixed size input. This can make direct work with point cloud very difficult as there is no fixed number of points or size. PointNet[39] and PointNet++[38] apply networks directly to point clouds by scaling them to a uniform subspace first. This has the implicit effect of setting each model in a voxel grid but is then processed in a point-by-point manner instead of considering the point cloud as a whole. While this approach is very effective, it does not take advantage of the local information considered by convolutions.

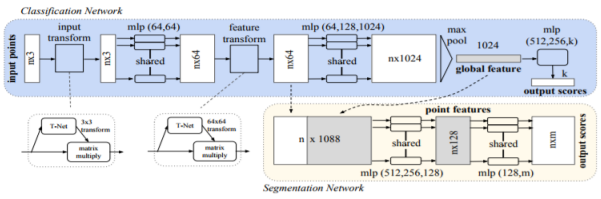



**Figure 6.** The convolution-less PointNet architecture

Despite not using convolutions in their architecture, as seen in figure 6, PointNet results provide point-wise labelling and very accurate results in part- and scene-segmentation. This is good indication that even though they do not use convolutions, the spacial scaling incorporates local details in the position. With that in mind, it may be possible to use a similar architecture for further work with segment based analysis.

Similarly, Lawin et al.[27] explore separation of point clouds into synthetic images for processing as 2D inputs before recombining for final point-wise labelling of segmentation. This is similar to PointNet as they attempt to avoid too much regularization of the network input by use of projection. However, the projection and management of all the images adds its own expensive preprocessing steps. Although, this method of semantic segmentation from

collections of 2D images has good chances to succeed for in the wild situations.

Finally in the field of 3D semantics, Tulsiani et al.[53] learn volumetric primitives for 3D objects. They assume a distribution of parts for every object and attempt to predict the locations and sizes of each part, as in figure 7. While this task is similar in that they resemble the prior used in the PVAE, they are non-gaussian shape priors and not supported by a parallel prediction network for improving the fine-detail of the output. Intuitively, shape prior learning is very similar to PVAE's but from the opposite direction. While learning the prior from the full model is an attempt to learn a distribution over parts for any given object, the PVAE aims to learn a distribution over objects given a distribution over parts. This task is harder in that to provide labelled segmentation data is almost impossible for large scale learning on 3D models and is likely to always be very noisy.

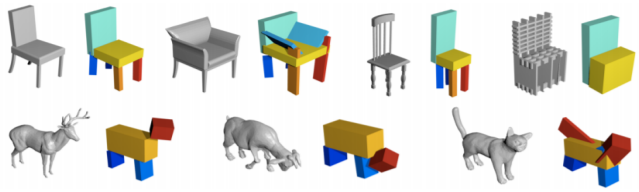


**Figure 7.** Volumetric primitives as assembled by Tulsiani et al.

### 1.3 Research Questions

In this work we attempt to answer two main questions; Can a Variational Auto-Encoder learn a predictive latent space over objects given sets of segments of objects? If so, can the distribution of segments on the manifold be leveraged for application as a point cloud labelling tool?

For testing, each collection of parts for each object will be encoded and the similarity between all parts per model measured. However this promotes a third question, are similarities in the latent space a strong enough identifier for classification by segments?

## 2 Data

Throughout this project the data, its size and sheer sparsity have been one of the greatest challenges. With that in mind it is worth considering a few details. For both datasets used, ModelNet10[59] and ShapeNet[4], a lot of preprocessing and management is required. The consequences of which have effects running right though the work, the two main factors are explained further.

The first issue is the availability of scale information in the given datasets. In ModelNet10, no such information is available. This makes it impossible to train the network on this dataset. Next, in order to produce fined grained results with nice definition it is required to use a high number of dimensions in the voxel grid; however, increasing the voxel grid size comes with exponentially increasing computational complexity.

Recent works have explored the use of octrees to combat low efficiency for high dimensional data, Tatarchenko et al.[52] created a decoder style network for generating realistic models from learnt embedding representations and Riegler et al.[41] train a full auto-encoder on octree representations and learn a predictive latent space for their network. Of the two octree approaches the first would be most suited for inclusion in this project. If the Octree Generating Network could be switched in for a standard deconvolutional NN decoder architecture, higher definition results would be achievable for minimal overhead. Possibly the regularity and translational variances of the octree representations would help when generating objects in a uniform space. However, the OctNet encoder would suffer from the translational variances in the input segments and so would be more difficult to use.

### 2.1 ModelNet10

The first dataset to be considered is the often used and much referenced ModelNet10, as supplied by Princeton, Stanford and TTIC[1][59]. This dataset has become a standard for 3D Computer Vision and Machine Learning tasks. For state-of-the-art 3D classification and retrieval accuracies see there website.

ModelNet10 is comprised of 10 categories, a subset of the larger ModelNet40, which are bathtub, bed, chair, desk, dresser, monitor, night stand, sofa and toilet. Each model is scaled to the same size and orientations aligned. Models are chosen to represent as much per-class variability without over or under representing each class.

### 2.2 ShapeNet

ShapeNet, also supplied by Princeton, Stanford and TTIC[2][4], is the second dataset to be used; which in turn is comprised of two main subsets, ShapeNetCore and ShapeNetSem. Included with ShapeNet is a taxonomy viewer for the whole dataset[3]. The viewer, as in figure 8, can be used for analyzing individual models, semantic groupings and the full dataset.

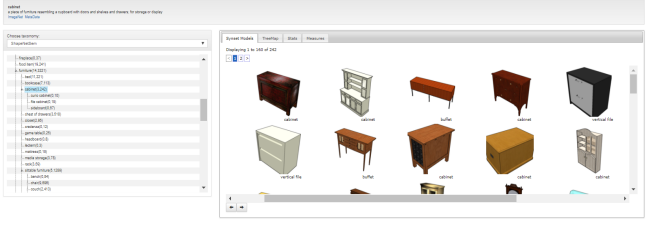


**Figure 8.** The ShapeNet Taxonomy viewer

ShapeNetSem is the smaller, more densely annotated subset consisting of 12,000 models spread over a broader set

of 270 categories. In addition to manually verified category labels and consistent alignments, these models are annotated with real-world dimensions, estimates of their material composition at the category level and estimates of their total volume and weight. As ShapeNetSem is the only subset containing real-world dimensional annotation it will be the only one considered from ShapeNet.

ShapeNetSem does not fully cover the classes consider in ModelNet10 so some re-ordering is required to improved comparability. Classes such as monitor, bathtub and toilet are not extensively, or at all, supported in ShapeNet. To remedy this, 10 classes have been constructed of similarly sized and balanced subsets from ShapeNetSem based only on the furniture classification. To make up the 10 classes, the Furniture subset of ShapeNetSem is further divided into bed, bookcase, chair, chest, cupboard, desk, sofa, stool, table with any remaining models put into a "misc" class to increase the inherent noise and hopefully improve generalizability of trained networks.

Of the 3211 selected models from ShapeNetSem, 80% are taken for training and 20% for testing. A further validation set is taken from 10% of the training set. These sets are split before any preprocessing or segmentation. This is to avoid having the same target model appearing in multiple splits with different input segments. Not only could that make training to convergence difficult but also would open up the possibility of overfitting specific examples. The splitting is done as to ensure equal percentages of models per class respective to ShapeNetSem - Furniture as a whole, ie $80/20$ overall $<=>$ $80/20$ per class.

### 2.3 Preprocessing

Although ShapeNetSem comes with pre-voxelized models, to ensure comparability across datasets a single pipeline for processing sparse point clouds through to voxelized whole models or sub-sampled segments is used. The pipeline is built on top of PCL[44] and the voxelization uses Binvox[4][37, 34]. The preprocessing is briefly explained in algorithm 1.

For these experiments $1m^3$ is chosen as a fair balance between not enough and too much. By increasing the segment size it would reduce smaller objects to pure reconstruction task and large objects would be closer to de-noising sliced models. By decreasing the segment size the task would become overly noisy and a lot more challenging. Finally, considering $1m^3$ is a well understood volumetric size it seems like a fair starting point for understandability by people and machine learning alike.

Following algorithm 1 ends up with roughly 220,000 'part'/'full' training pairs for the ShapeNetSem - Furniture subset, as seen in figure 3. The training pairs vary in difficulty depending on the objects size, the largest provide the most but least recognizable segments while the smallest objects produce the least with most recognizable segments.

**Algorithm 1** Data Preprocessing

```
For a given split of the dataset
for all Object φ in split do

    Scale object φ to real-world size
    Poisson-Disk re-sampling of φ
    12 θ rotations in x − z plane per object
    for all θ do

        Object φ at angle θ
        Let δ = max(φ, θ) be φ's greatest span
        With central point φ_c = (φ_cx, φ_cy, φ_cz)
        Set whole bounding box limits as φ_c ± δ/2
        return Voxelized φ given whole bounding box
        if ShapeNetSem then

            Given step size s
            Given segment size μ
            for all μ segments in φ at s separation do

                Let φ_s ∈ φ be the central point after given steps

                Set segment bounding box φ_s ± μ/2
                return Voxelized segment given bounding box
            end for
        end if
    end for
end for
```

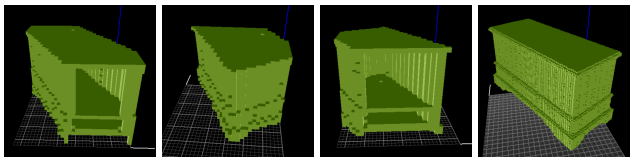


**Figure 9.** Example of segmentation process

## 3 Experimental Design

In this project three network architectures will be assessed. The first, to improve comparability, will be the VAE architecture from Brock et al.[3]. This will be trained on ModelNet10 and assessed under varying noise conditions to show it standard capacity. It will then be trained with segment prediction task to see how well it copes. The next two networks will be the Prior-VAE (PVAE) and Located-VAE (LVAE) for segment to whole prediction. The PVAE and LVAE architectures are largely inspired by Brock et al.'s volumetric VAE with the addition of specialized modules.

### 3.1 VAE

#### 3.1.1 Architecture

As set out by Brock et al., implemented in Tensorflow, the VAE consists of three main section; the encoder, consisting of 4 convolutional layers and a fully connected layer; the latents, a linear projection from the encoder output to the

latent distribution parameters; the decoder, which has an identical but inverted architecture with weights disconnected to the encoders.

Each convolutional layer has a bank of 3x3x3 filters, starting with 8 filters in the layer furthest from the latents and doubling at each subsequent layer. Strided de/convolutions are used in place of pooling to decrease loss of information due to the sparsity of input models. Figure 10 shows the graph architecture as displayed by TensorBoard. The full architecture overview and module construction can be found in Appendix 8.4.1.

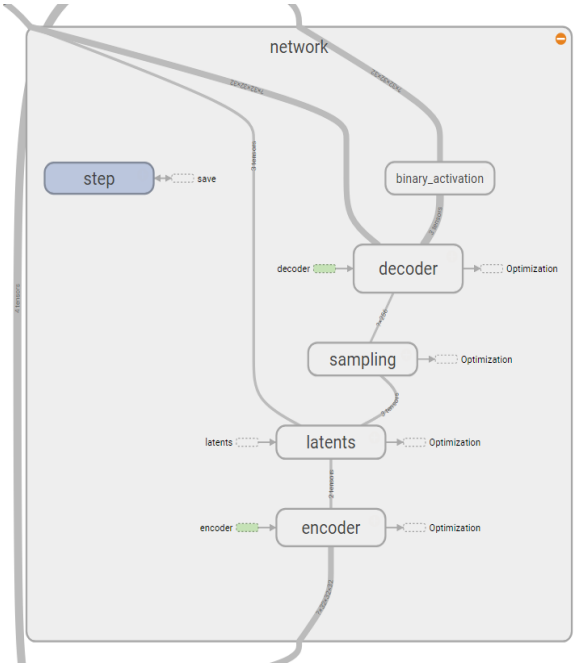



**Figure 10.** VAE graph architecture

Glorot initialization[14] is used throughout with Exponential Linear Units for layer activations as Brock et al. However, no Batch Normalization[23] was used as empirically it lead to numerical instability.

### 3.1.2 Training

By examining the capabilities of the vanilla VAE with respect to varying noise levels we can analyse its predictive abilities, for this reason the network is trained with increasing levels of noise. As an extension to volumetric reconstruction, it is interesting consider the sliding window segmentation as noise. If the VAE can handle widely varying levels of noise, then the siding window segment to whole prediction task will be possible.

The VAE is trained with 4 levels of increasing noise where each level of noise includes the previous. First training without noise, second training with gaussian noise, next training with the addition of translational noise and finally all previous noises plus slicing. These are chosen for their increasing difficulty and probability of occurence in the wild, an example of each type of noise and how they are handled by the trained VAE's can be found in table 3. The VAE is trained for 20 epochs per noise level, this is in constrast to Brock et al.'s 80 epochs as high precision results are not required for determining general ability to cope with de-noising. When training with noise, batches of $\frac{1}{3}$ genuine samples and $\frac{2}{3}$ samples with noise are used throughout.

Stochastic Gradient Descent with Nesterov momentum[2] is used for all training, as Brock et al. Gradient clipping is also applied to avoid too many adverse exploding gradient effects. Adam[24] was briefly tested but the addition of extra hyper-parameters required extensive training for certainty of the results. Exponential decay is used throughout as the primary method for altering the learning rate, starting at $1e-5$ and halving. All training was completed on an NVIDIA Titan X for all experiments.

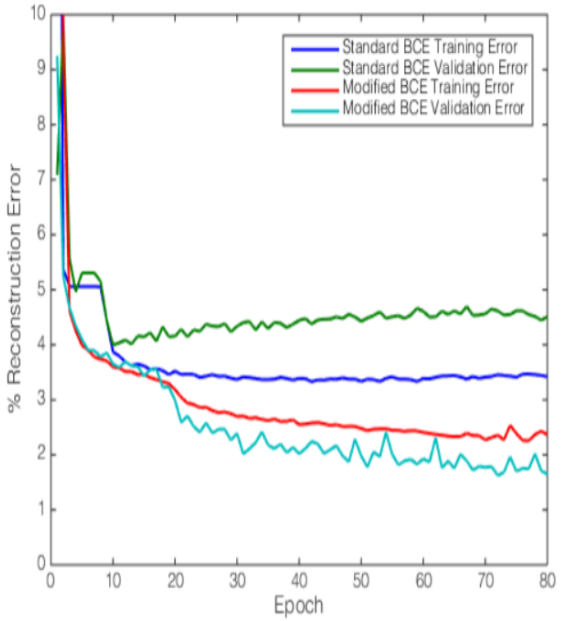



**Figure 11.** Comparison of training/validation losses with and without the modified cross-entropy from Brock et al.

### 3.1.3 Loss

For back-propagation, the Küllbach-Leibler divergence is used for maximization of the variational lower bound and a modified cross-entropy loss is used for reducing the reconstruction error. The modified loss

$$\mathcal{L}(x, y) = -\gamma x \log(y) - (1-\gamma)(1-x)\log(1-y)$$
$$\gamma \in (0, 1)$$

as shown empirically by Brock et al, improves the training stability of very sparse source/target pairs. Here $x$ is the prediction, $y$ is the target and $\gamma$ is the weighting parameter.

It is worth noting, without addition of this heuristic, the sheer sparsity of the models will actively promote convergence to zero. Figure 11 shows the empirical results from Brock et al. when first implementing the gamma loss

alteration.

## 3.2 PVAE

### 3.2.1 Architecture

The PVAE architecture is similar to the VAE with some alterations with respect to the method of manifold learning. The encoder, latent distribution modules and the decoder are the same as above. However, the Decoder, now a Predictor, is altered by a mixture of gaussians prior for generalization of shape. Figure 12 shows the graph architecture as displayed by TensorBoard. The full architecture overview and module construction can be found in Appendix 8.4.2.

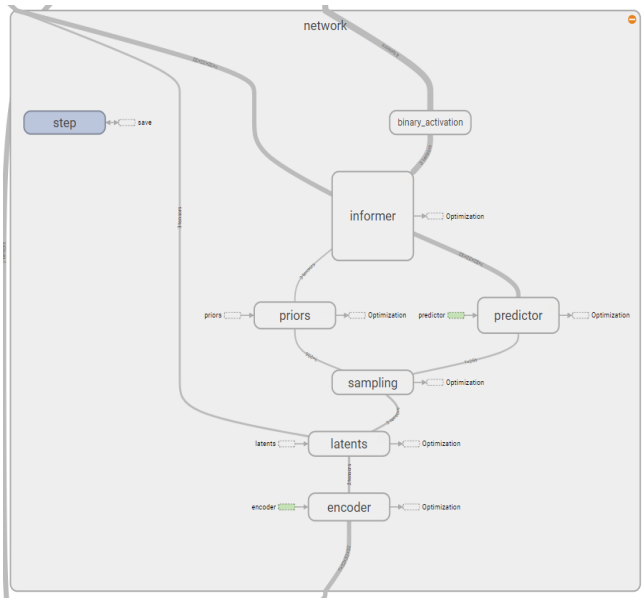



**Figure 12.** PVAE graph architecture

The Informed shape prior is composed of a mixture of 8 multivariate Gaussian distributions estimated from the latent distribution sample. For training stability, the output of the prior distribution estimator has a sigmoid activate followed by a $(0, 32)$ scaling, such that the means and variances of all predicted gaussians are within the limits of the target voxel grid. The prior is pushed through another sigmoid activation before multiplying with the prediction. The final sigmoid is intended as a way of relating the prior to a probability mass "switch", however it is also necessary for numerical stability.

The final network output is the binarized product of the prediction and the prior. During training the binary threshold is randomly selected from the range $(0.6, 0.8)$ to stochastically increase the back-propagated gradients of boundary cases.

### 3.2.2 Training

In general, the noisiness of the segments could be steep enough that the network would not overfit. However, in practice some small amounts of gaussian noise were necessary to prevent exploding gradients. Similarly, the source/target pairs were randomly flipped in the same dimensions to prevent reliance on the orientation of the model being tested. On top of these a regularizing term is used to prevent exploding parameters.

Although Dropout[20] and Batch-Normalization are often used, here they were found to lead to further instabilities. The added cost of running a parameter search on the dropout was prohibitive.

Similar to the VAE, vanilla SGD with Nesterov momentum and gradient clipping is used. Starting with an initial rate of $1e-5$ and decaying exponentially as preiviously.

Due to the time complexity involved while using voxel models certain measure had to be taken to reduce any excess overhead. Multi-threading is used whenever possible for data preparation and validation scores are only subsampled to reduce time spent validating. These have adverse effects though, such as only approximation of train or validation scores could lead to non-smooth loss. However, They have been necessary requirements for manageable training times.

### 3.2.3 Loss

The same loss heuristic as before is used with $\gamma = 0.97$. An L2-regularizer is used with $\epsilon = 0.01$.

## 3.3 LVAE

### 3.3.1 Architecture

The LVAE architecture is again similar to the VAE with some alterations. This time the latent space representation is split into two distributions, the full distribution and a secondary distribution representing the location of the segment in the whole. The location distribution is the same as before in that its divergence from a unit gaussian is reduced via the Küllbach-Leibler divergence. However, the full distribution is unconstrained and sampled at the latent space embedding location given by the location distribution sample. Figure 13 shows the graph architecture as displayed by TensorBoard. The full architecture overview and module construction can be found in Appendix 8.4.3.

This choice of alteration is motivated by the implicit sampling of segments from the whole object. It is clear from training the VAE with the segment task that the uniqueness of latent representations on the manifold is inhibiting reconstruction of the same object from multiple relevant parts. To that end, the two distributions can learn unique but separate representations for both the object as a whole and segments from the object.

The final network output is the binarized decoding. Again, during training the binary threshold is randomly selected from the range $(0.6, 0.8)$ to stochastically increase the back-propagated gradients of boundary cases.

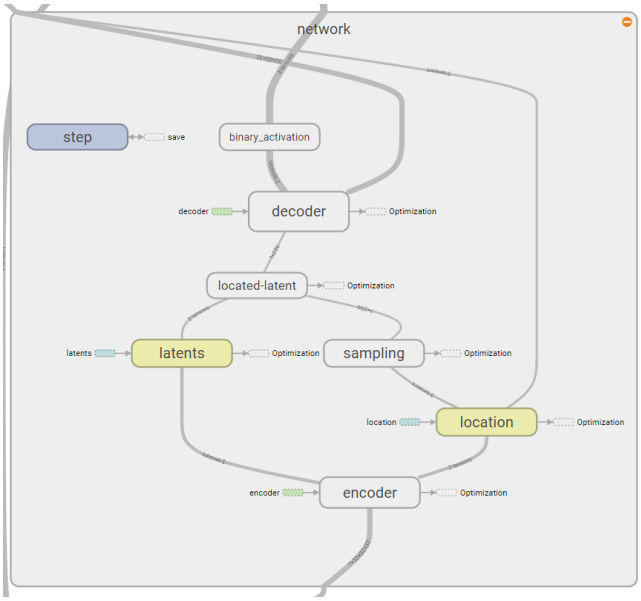



**Figure 13.** LVAE graph architecture

#### 3.3.2 Training and Loss

The same training regime and loss heuristic as before were used with $\gamma = 0.97$. An L2-regularizer is used with $\epsilon = 0.01$.

## 4 VAE Results

### 4.1 Training on ModelNet10

#### 4.1.1 Comparison of VAE reconstruction results

Table 1 shows the reconstruction results achieved by Brock et al after training for 80 epochs using batches of 1/2 translational noise.

| | Predicted: Positive | Predicted: Negative |
|---|---|---|
| Actual: Positive | 99.39% | 0.61% |
| Actual: Negative | 7.64% | 92.36% |

**Table 1.** Reconstruction accuracies from Brock et al.

These values will be used as control for validating the efficacy of the baseline VAE's training. Table 2 shows the comparison of each level of noised training against the control as measured on a held out test set.

| Model | Precision | Recall | Specificity |
|---|---|---|---|
| Brock et al. | 92.86 | 99.39 | 92.36 |
| Ours w/o Noise | 46.12 | 98.86 | 89.43 |
| Ours w/ Gaussian | 49.36 | 98.04 | 91.09 |
| Ours w/ Translation | 35.76 | 97.14 | 85.42 |
| Ours w/ Slicing | 37.62 | 98.73 | 85.07 |

**Table 2.** Table of Reconstruction metrics for the VAE examined in this work given varied augmentation training for 20 epochs compared to Brock et al. for 80

Table 2 shows that although the VAE is only trained for one quarter the number of epochs as Brock et al it still acheives high recall and specificity. The precision however is heavily affected. This is most likely due to the high sparsity of the voxel models. Even though the number of mis-predicted voxels may be low, as can be seen from the high recall and specificity, the high sparsity diminishes the precision. This will largely affect fine detail only and overall reconstruction structure will not be affected.

#### 4.1.2 Augmentation

Figure 14 shows the validation scores by training epoch. The visible, foreground lines represent smoothed averages over samples tested at validation time. The faded, background lines represent the unsmoothed values and per epoch statistical ranges.

It is clear that the larger augmentation noises affect the overall reconstruction accuracy. Most effected is the recall but this is expected as the networks trained with greater noise levels are working harder to predict missing or misplaced voxels. This will cause increased mis-prediction, especially in early stages of training.

Table 3 (Appendix) shows how each VAE handles varied levels of noise given its training. We can see that the fine detail is lost with progressive augmentation training but the capacity to cope with noised or missing data increases

### 4.2 Training on ShapeNetSem segmented

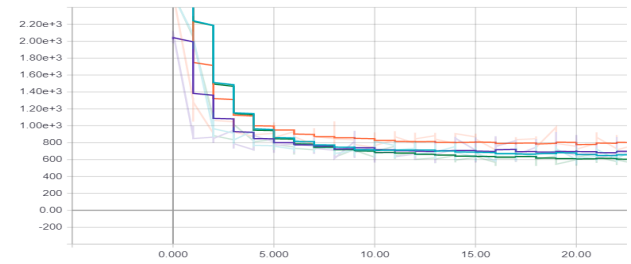
#### 4.2.1 difficulties

There are several major difficulties involved in making predictions of full objects given segments. The uniqueness of latent space representations will inhibit multiple parts predicting the same object. The challenge of varying scale in the output space, as each object is predicted in a uniform voxel grid given semantically comparable sizes in the input space. Furthermore, the high similarity of parts in the input sample window is a large source of noise, a surface without full context may be an object like a table or something closer to a chest of drawers.

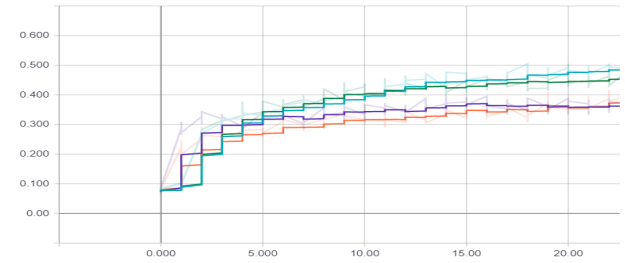
Figure 15 shows some training results for a vanilla VAE architecture trained on segments. It is clear how much the VAE struggles to learn a predictive manifold when given largely incomplete training samples. The manifold uniqueness rapidly starts overfitting on previously seen samples, seen in how the (orange) training results continue to improve while the (turquoise) validation results diverge quickly. They are compared to the (green) noiseless validation and (blue) noiseless training results for standard reconstruction.

## 5 PVAE and LVAE

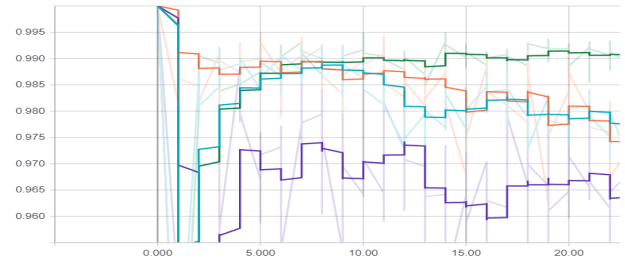
### 5.1 Training on ShapeNet segments



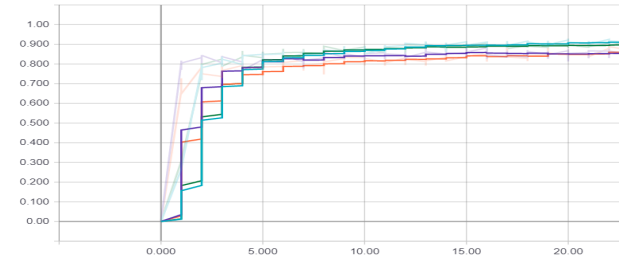
**(a)** Cross-entropy loss



**(b)** Precision



**(c)** Recall



**(d)** Specificity


**Figure 14.** Comparison of validation scores by epoch over different noise levels with fixed hyperparameters as displayed by TensorBoard

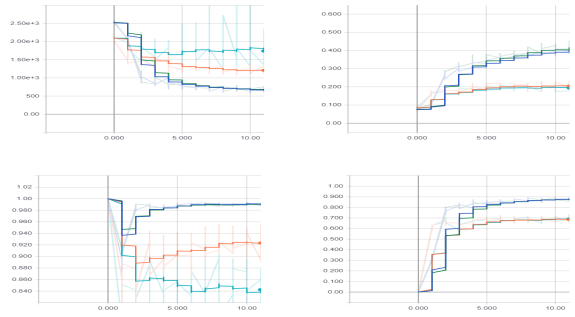


**Figure 15.** Table of VAE training on segments (train: orange, validation: turquoise) compared to vanilla noiseless (training: blue, validation: green) reconstruction (clockwise from the top left: loss, precision, specificity, recall)

Shown in figure 16 shows the training statistics for each of the architectures for the segment to whole prediction task. It can be seen that although some learning is being done, the task is more complex than first thought. To achieve visually pleasing reconstruction results requires $loss < 700$ at least. Although the LVAE gets down to $\approx 1000$ it is still not enough for visually pleasing reconstructions. However this is not to say that the Segments lack similarity in the latent space.

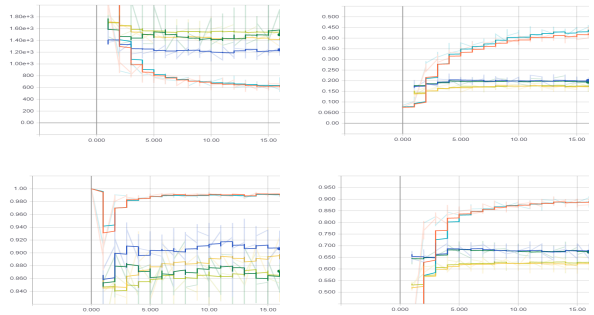


**Figure 16.** Average training of PVAE (yellow: train, lime:validation), LVAE (blue: train, green:validation) with respect to a Noiselessly trained VAE (orange: train, turquoise: validation) (clockwise from the top left: loss, precision, specificity, recall)

## *5.2 Classification by Embedding*

Figure 17 shows a PCA clustering of the latent space representations of all whole object in ShapeNet given a VAE trained without noise on ModelNet10. Here colour indicates matching object class taken from the constructed classes of ShapeNet. Despite being trained on ModelNet10, the clustering for ShapeNet is still well defined. This shows that, similar to training on large 2D image datasets, there is generalizability across similar datasets. This implies that learnt manifolds are not necessarily specific to the data used. But exist, in part, as general distributions of points in voxel space.

In figure 18 we see results from the same VAE as above but with embeddings of object segments. Although there is some structure visible, the clusterings are a lot less tidy and separable. This shows that although reconstruction of the segments is possible, their representation in the latent space is not related to the object class from which they are taken.

In figures 19 and 20 show the segment embeddings of the PVAE and LVAE respectively. Interestingly the network with better final training loss, the LVAE, has learnt an almost entirely linear latent space. Some small groupings can be seen, such as the collection of green near the top, but in general there is no clear separation. The PVAE clustering is a lot more visually understanderable, separate object classes can be discerned from the groupings despite each embedding representing an encoded segment. Although the results are not as clearly defined as with the VAE it is definite proof of

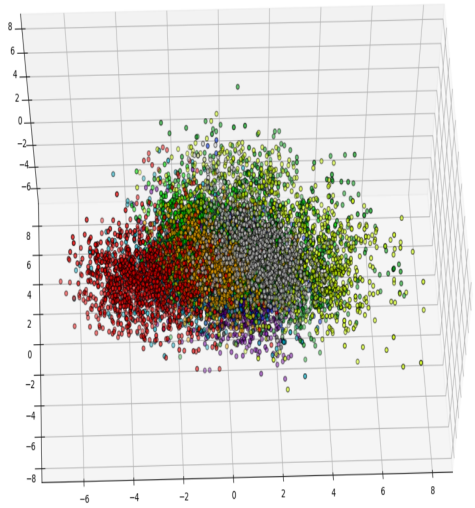


**Figure 17.** PCA embedding of whole objects as latent space embeddings given noiseless VAE training

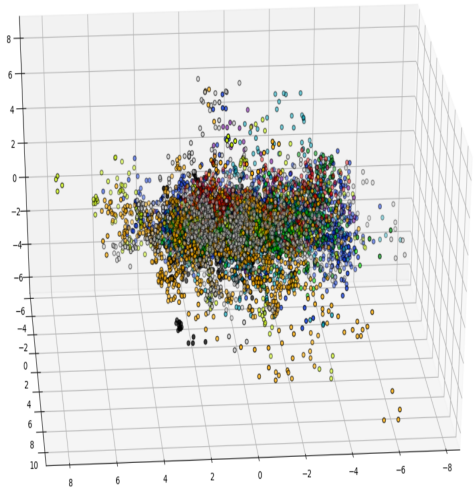


**Figure 18.** PCA embedding of segmented objects as latent space embeddings given noiseless VAE training

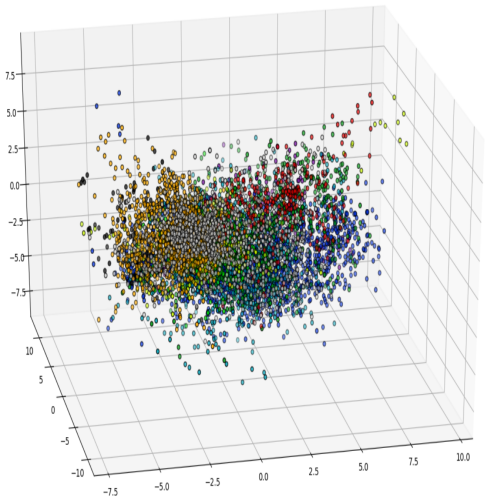


**Figure 19.** PCA embedding of segmented objects as latent space embeddings given PVAE architecture

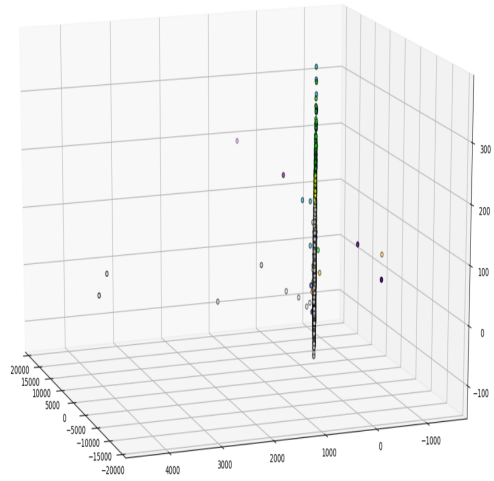


**Figure 20.** PCA embedding of segmented objects as latent space embeddings given LVAE architecture

similarity between object segments.

Interestingly, the PCA structure seen in figure 20 is indicative of the underlying process of the LVAE. The main column shows the whole distribution of objects however by zooming in we see that the column represents a sampling region. Figure 21 shows the local structure of the sampling region learnt by the LVAE, here it can be seen that there are object clusters of segments in the main column. This is significant as it shows that although the whole sample distribution is large and varied, the local regions contain structured subsection relating to the embeddings overall.

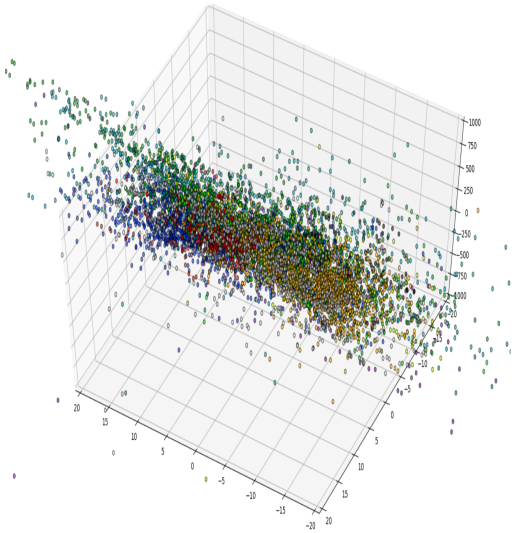


**Figure 21.** Magnified LVAE PCA embedding

## 6 CaPtAEN

### *6.1 Visualization*

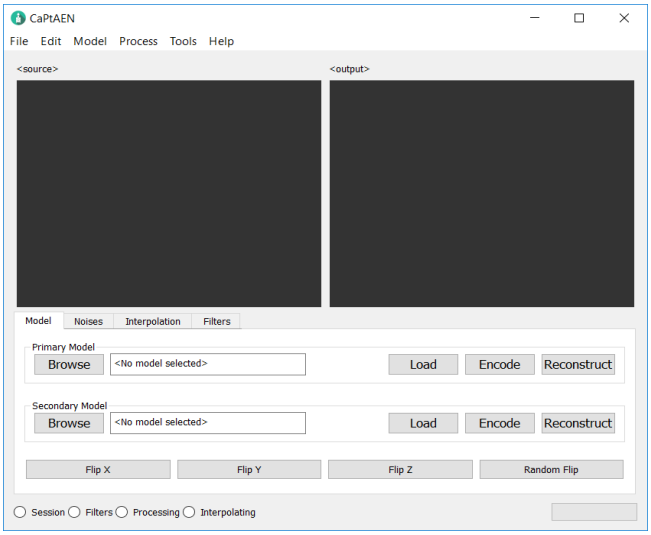



**Figure 22.** Model selection

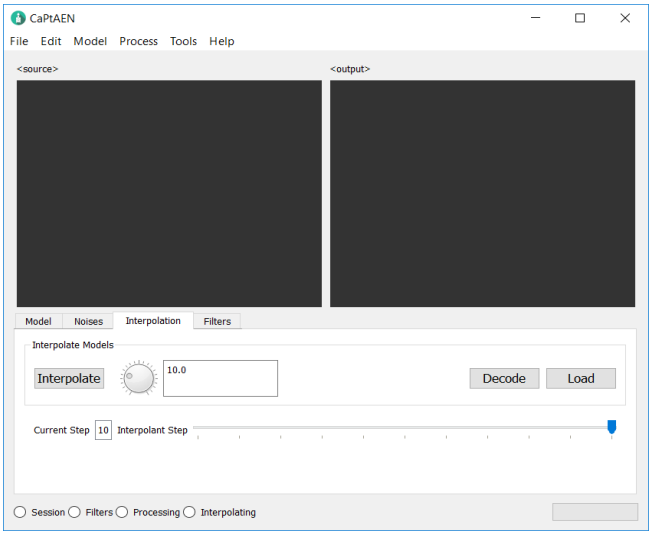



**Figure 23.** Latent interpolation

For ease of assessing and testing trained networks I also include an application and framework for loading voxel model reconstructors to be tested. The input and output are shown with conventional 3D viewers, shown in figure 22 above multiple options for noising and interpolating between objects. From this interface it is also possible to encode secondary objects and/or flip objects in each axis.

Figure 61 (Appendix) highlights the various objects taken from the ModelNet10 dataset. Figure 62 (Appendix) shows various segments of objects sampled from ShapeNetSem.

## 6.2 Noising

Figure 59 shows the noise selection and alteration interface. These tools can be used for binary threshold alterations, addition of gaussian noise and slicing.

In table 3 (Appendix) it can be seen how VAE's with different levels of augmentation training are effected by differing type of noise. The first row shows results after application without noise, the second row demonstrates gaussian noise andhe final row shows results after slicing the input.

It is plain to see that the additional noises during training are incorporated into the networks predictive capabilities and not much further. This could imply that each is learning a unique or sub-manifold of a more complex distribution over voxel space.

## 6.3 Interpolation

A common test for generative methods is interpolation through the latent space. This is helpful to show how objects are related on the manifold and also to show the smoothness of transitions between objects. It provides similar intuition as gained from visualization such as figure 24.

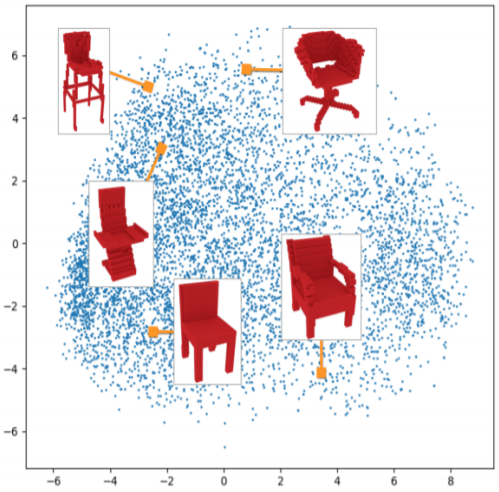



**Figure 24.** 2D PCA embedding of voxelized chairs by Liu et al

The CaPtAEN application allows for easy interpolation and quick analysis of latent space smoothness. Interpolation is as simple as loading and encoding a primary and secondary model using the model interface. Once two latent representations are loaded, the interpolation interface allows from smoothly transitioning through the latent space given an arbitrary amount of linear steps. Steps are taken between two embeddings along the line $L = l_1 * t + l_2 * (1 - t) \forall t \in (0, 1)$, where $l_1, l_2$ are the encodings of the primary and secondary model respectively while $t$ is the percentage travelled between them.

Table 4 (Appendix) shows various results from interpolation between models in the latent space. Visualizing in 3D allows for manipulation and detailed examination of the generated interpolants. This is helpful for clear understanding of how the interpolations change.

### 6.4 Filters

The filter interface, figure 60, allows for loading, selection and viewing of learnt network filters. This is particularly useful for confirming the network is learning structured filters for geometric voxel analysis.

The left viewer shows the normalized filters. Filters are normalized such that for a given filter bank the maximum is scaled to 1 (if greater than 0, 0 otherwise), the minimum is scaled to -1 (if less than 0, 0 otherwise) and 0 remains unchanged. The magnitude of each filter element is used for opacity scaling. Colouring is a divergent colour scale from -1 (yellow) to 1 (red).

The right viewer shows a ”binarized” form of the normalized filter. Each element is set to -1, 0 or 1 and visualized the same as before. This is useful for showing the overall structure of the filter.

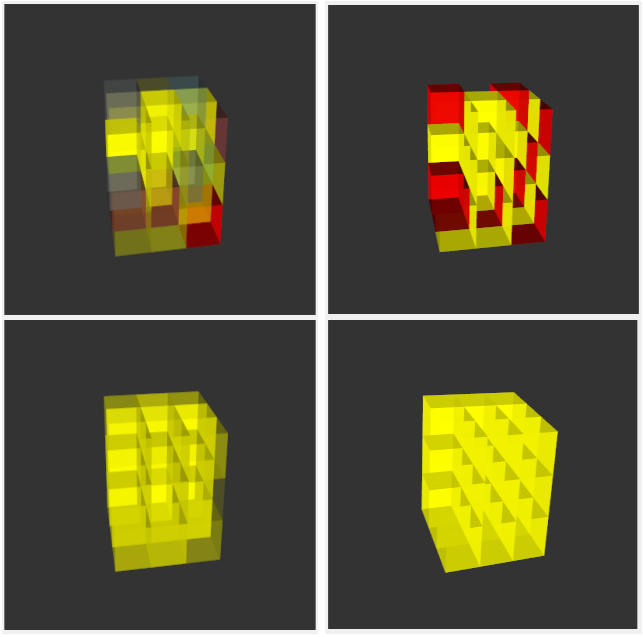


**Figure 25.** Example low level filters visualized using the CaPtAEN viewers (left: normalized, right: binarized)

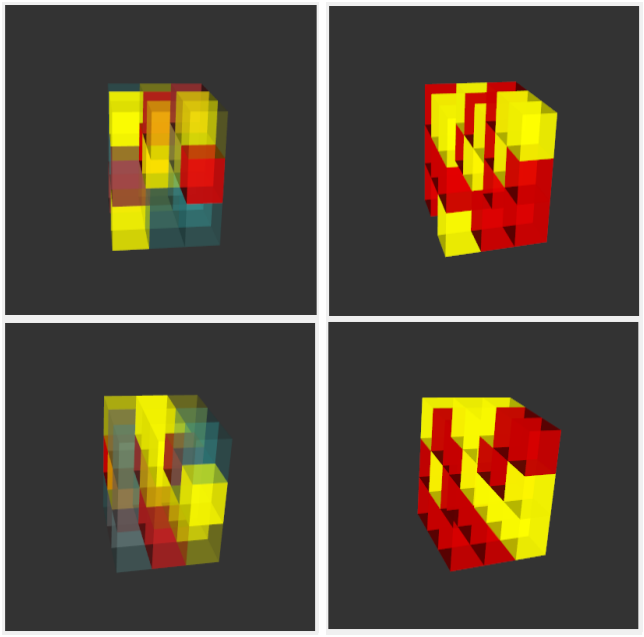


**Figure 26.** Example high level filters visualized using the CaPtAEN viewers (left: normalized, right: binarized)

## 7 Conclusion

### 7.1 Discussion

Although VAE’s have a lot of potential for de-noising and in-filling tasks, the task of predicting whole object models from given segments is a lot more complex. In this work it has been shown possible, particularly with modified architectures such as the PVAE and LVAE. So in this respect, the answer to ”is it possible to link semantically scaled input segments to a latent space of objects” is yes. However, the full task of accurate, fine-detail prediction is yet to be seen.

Throughout this project, the sparsity of information in voxel grids has been the largest difficulty. The computational complexity and size in memory of each object can be challenging enough. The sparsity of each model, reducing useful information for back-propagation, is also very challenging, especially at higher resolution models.

Recent work using octrees[52, 41] and point clouds directly[39, 38] show promise for better, more efficient architectures of CNN’s for volumetric computer vision. It may be possible with these to increase the resolution and fine-detail predictions of generative methods. If that were possible then the sliding volumetric window approach would be more viable for 3D scene scanning.

With respect to which architecture is better for segment to whole prediction, the LVAE achieves a lower loss and generally better metrics but the PVAE shows more promise for separability in the latent space. These two results imply that the possibility of training an accurate prediction system from segments to whole is possible, however more work refining the architecture is necessary. Possibly a combination of the LVAE and PVAE would improve both results.

In consideration as to whether the learnt manifold is accurate enough to provide high similarity between parts. In some experimentation with standard architectures it can be seen that the VAE’s tendency to ”unique-ify” each inputs latent representation causes severe overfitting when given the segment to whole task. Of the two architectures, the PVAE shows a lot better separability in the latent space, shown in the PCA embeddings.

In some short experiments attempting to train the PVAE with just the Informer, it lead to far too much instability. This is most likely due to the high levels of false predictions made without amendment for the fine detail. In testing the standard

VAE task though it rapidly became over-fit, this is most likely due to the KL-divergences tendency to ”uniquify” latent space representations. For these reasons it seems plausible that the product of both Informer and Predictor had the effect of ”snapping” the latent representations to a more general manifold, as in figure 27 from Liu et al.[31]. This would be a likely explanation for the closer similarity between the PCA clusterings for the PVAE and VAE, however the lack of visually pleasing generated results shows there is more work to do.

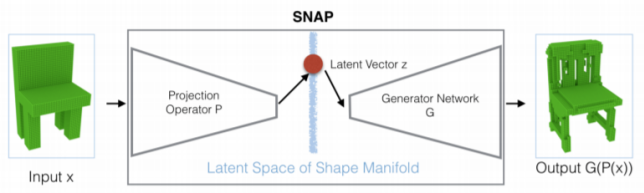



**Figure 27.** ”SNAP”ing to manifold by Liu et al.

Due to the many challenges involved in developing this project, the sparsity and the data science in particular, measures had to be taken in order to explore multiple avenues of research while still ensuring reproducable results. However even with the optimization already in place, to collect results for 20 epochs required $\approx$ 2 days of training. It is possible that with longer, slower training the results would improve. In the scope of this project, the time constraints made extensive training challenging. Managing the time constraint is an issue that would require further development in order to follow further volumetric work in this field.

The cost of this computational efficiency is no more obvious than in the filters learnt from volumetric learning. Low level filters, such as those in figure 4, are as we would expect, akin to edge and corner filters[13, 51, 62], but higher level filters become more and more abstract and random seeming, as in figure 26. In 2D networks these issues are resolved by increasing filter size to allow for greater information captured by the filters. However, increasing the filter size for 3D becomes unfeasible for two reasons. From a resource point of view, larger 3D convolutional filters increase the computational complexity and time costs; from a relevance point of view, the resolution of the voxelized models is already low so max-pooling and strides, as is usual in 2D CNN's, only decrease the information present.

## 7.2 Further Work

For future research in this direction it would be interesting to explore the use of octree representations[52, 41], pointnet archtiectures[39, 38] and/or hierarchical surface prediction[18]. Although new variances brought by using these architectures, such as tranlational variance for octrees, may cause new troubles.

As it was relatively easy to learn a generative manifold for whole objects, it would also be interesting to use a pre-trained generator for a transfer learning approach. This would reduce the segment to whole prediction to learning only the connection between segments and the pre-trained manifold.

The use of G-Convolution[7] could also lead to better, more generalized filters. This could help for reducing filter quantities and also improving the information capacity of each filter.

As mentioned in Mu et al.[58], it is often better to use Annealed Importance Sampling (AIS)[36] when validating generative methods. This was not included but may have helped with better intuition of validation scores.

### Notes

1 http://modelnet.cs.princeton.edu/

2 https://www.shapenet.org/

3 https://www.shapenet.org/taxonomy-viewer

4 http://www.patrickmin.com/binvox/

### Acknowledgements

The majority of my thanks go to my family, who have supported and motivated me throughout my time on the Msc AI; I would like to thank all the fantastic people at 3DUniversum, who are always willing to have in-depth and thought provoking discussions (no matter the relevance of the idea); finally, I would like to thank Universiteit van Amsterdam for two great years of sleepless nights, mind-boggling problems and last minute deadlines.

## References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *CoRR* abs/1511.00561 (2015). URL: http://arxiv.org/abs/1511.00561.

[2] Aleksandar Botev, Guy Lever, and David Barber. “Nesterov’s Accelerated Gradient and Momentum as approximations to Regularised Update Descent”. In: *CoRR* abs/1607.01981 (2016). URL: http://arxiv.org/abs/1607.01981.

[3] André Brock et al. “Generative and Discriminative Voxel Modeling with Convolutional Neural Networks”. In: *CoRR* abs/1608.04236 (2016). URL: http://arxiv.org/abs/1608.04236.

[4] Angel X. Chang et al. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015). URL: http://arxiv.org/abs/1512.03012.

[5] Christoforos C. Charalambous and Anil A. Bharath. “A data augmentation methodology for training machine/deep learning gait recognition algorithms”. In: *CoRR* abs/1610.07570 (2016). URL: http://arxiv.org/abs/1610.07570.

[6] Özgün Çiçek et al. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: *CoRR* abs/1606.06650 (2016). URL: http://arxiv.org/abs/1606.06650.

[7] Taco S. Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *CoRR* abs/1602.07576 (2016). URL: http://arxiv.org/abs/1602.07576.

[8] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. “Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis”. In: *CoRR* abs/1612.00101 (2016). URL: http://arxiv.org/abs/1612.00101.

[9] Antonio D’Innocente et al. “Bridging between Computer and Robot Vision through Data Augmentation: a Case Study on Object Recognition”. In: *CoRR* abs/1705.02139 (2017). URL: http://arxiv.org/abs/1705.02139.

[10] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. “Unsupervised feature learning by augmenting single images”. In: *CoRR* abs/1312.5242 (2013). URL: http://arxiv.org/abs/1312.5242.

[11] P.D. Ellis. *The Essential Guide to Effect Sizes: Statistical Power, Meta-Analysis, and the Interpretation of Research Results*. Cambridge University Press, 2010. ISBN: 9780521142465. URL: https://books.google.nl/books?id=5obZnfK5pbsC.

[12] Rohit Girdhar et al. “Learning a Predictable and Generative Vector Representation for Objects”. In: *CoRR* abs/1603.08637 (2016). URL: http://arxiv.org/abs/1603.08637.

[13] Ross B. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). URL: http://arxiv.org/abs/1311.2524.

[14] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. 2010, pp. 249–256. URL: http://www.jmlr.org/proceedings/papers/v9/glorot10a.html.

[15] Benjamin Graham. “Fractional Max-Pooling”. In: *CoRR* abs/1412.6071 (2014). URL: http://arxiv.org/abs/1412.6071.

[16] Edward Grant, Pushmeet Kohli, and Marcel van Gerven. “Deep disentangled representations for volumetric reconstruction”. In: *CoRR* abs/1610.03777 (2016). URL: http://arxiv.org/abs/1610.03777.

[17] JunYoung Gwak et al. “Weakly Supervised Generative Adversarial Networks for 3D Reconstruction”. In: *CoRR* abs/1705.10904 (2017). URL: http://arxiv.org/abs/1705.10904.

[18] Christian Häne, Shubham Tulsiani, and Jitendra Malik. “Hierarchical Surface Prediction for 3D Object Reconstruction”. In: *CoRR* abs/1704.00710 (2017). URL: http://arxiv.org/abs/1704.00710.

[19] Vishakh Hegde and Reza Zadeh. “FusionNet: 3D Object Classification Using Multiple Data Representations”. In: *CoRR* abs/1607.05695 (2016). URL: http://arxiv.org/abs/1607.05695.

[20] Geoffrey E. Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *CoRR* abs/1207.0580 (2012). URL: http://arxiv.org/abs/1207.0580.

[21] Xianxu Hou et al. “Deep Feature Consistent Variational Autoencoder”. In: *CoRR* abs/1610.00291 (2016). URL: http://arxiv.org/abs/1610.00291.

[22] Daniel Jiwoong Im et al. “Denoising Criterion for Variational Auto-Encoding Framework”. In: *CoRR* abs/1511.06406 (2015). URL: http://arxiv.org/abs/1511.06406.

[23] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[24] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[25] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *CoRR* abs/1312.6114 (2013). URL: http://arxiv.org/abs/1312.6114.

[26] Roman Klokov and Victor S. Lempitsky. “Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models”. In: *CoRR* abs/1704.01222 (2017). URL: http://arxiv.org/abs/1704.01222.

[27] Felix Järemo Lawin et al. “Deep Projective 3D Semantic Segmentation”. In: *CoRR* abs/1705.03428 (2017). URL: http://arxiv.org/abs/1705.03428.

[28] Yann LeCun and Corinna Cortes. “MNIST handwritten digit database”. In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[29] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. “Smart Augmentation - Learning an Optimal Data Augmentation Strategy”. In: *CoRR* abs/1703.08383 (2017). URL: http://arxiv.org/abs/1703.08383.

[30] Yijun Li et al. “Generative Face Completion”. In: *CoRR* abs/1704.05838 (2017). URL: http://arxiv.org/abs/1704.05838.

[31] Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. “Interactive 3D Modeling with a Generative Adversarial Network”. In: *CoRR* abs/1706.05170 (2017). URL: http://arxiv.org/abs/1706.05170.

[32] Shikun Liu, Alexander G. Ororbia II, and C. Lee Giles. “Learning a Hierarchical Latent-Variable Model of Voxelized 3D Shapes”. In: *CoRR* abs/1705.05994 (2017). URL: http://arxiv.org/abs/1705.05994.

[33] Daniel Maturana and Sebastian Scherer. “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: (Sept. 2015).

[34] Patrick Min. *Binvox*. 2004 - 2017. URL: http://www.patrickmin.com/binvox.

[35] Pim Moeskops et al. “Automatic segmentation of MR brain images with a convolutional neural network”. In: *CoRR* abs/1704.03295 (2017). URL: http://arxiv.org/abs/1704.03295.

[36] Radford M. Neal. “Annealed Importance Sampling”. In: *Statistics and Computing* 11.2 (Apr. 2001), pp. 125–139. ISSN: 0960-3174. DOI: 10.1023/A:1008923215028. URL: http://dx.doi.org/10.1023/A:1008923215028.

[37] Fakir S. Nooruddin and Greg Turk. “Simplification and Repair of Polygonal Models Using Volumetric Techniques”. In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 191–205.

[38] Charles Ruizhongtai Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *CoRR* abs/1706.02413 (2017). URL: http://arxiv.org/abs/1706.02413.

[39] Charles Ruizhongtai Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CoRR* abs/1612.00593 (2016). URL: http://arxiv.org/abs/1612.00593.

[40] Charles Ruizhongtai Qi et al. “Volumetric and Multi-View CNNs for Object Classification on 3D Data”. In: *CoRR* abs/1604.03265 (2016). URL: http://arxiv.org/abs/1604.03265.

[41] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *CoRR* abs/1611.05009 (2016). URL: http://arxiv.org/abs/1611.05009.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597 (2015). URL: http://arxiv.org/abs/1505.04597.

[43] Holger R. Roth et al. “Hierarchical 3D fully convolutional networks for multi-organ segmentation”. In: *CoRR* abs/1704.06382 (2017). URL: http://arxiv.org/abs/1704.06382.

[44] Radu Bogdan Rusu and Steve Cousins. “3D is here: Point Cloud Library (PCL)”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011.

[45] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. “Tversky loss function for image segmentation using 3D fully convolutional deep networks”. In: *CoRR* abs/1706.05721 (2017). URL: http://arxiv.org/abs/1706.05721.

[46] Nima Sedaghat, Mohammadreza Zolfaghari, and Thomas Brox. “Orientation-boosted Voxel Nets for 3D Object Recognition”. In: *CoRR* abs/1604.03351 (2016). URL: http://arxiv.org/abs/1604.03351.

[47] Abhishek Sharma, Oliver Grau, and Mario Fritz. “VConv-DAE: Deep Volumetric Shape Learning Without Object Labels”. In: *CoRR* abs/1604.03755 (2016). URL: http://arxiv.org/abs/1604.03755.

[48] Ayan Sinha et al. “SurfNet: Generating 3D shape surfaces using deep residual networks”. In: *CoRR* abs/1703.04079 (2017). URL: http://arxiv.org/abs/1703.04079.

[49] Jost Tobias Springenberg et al. “Striving for Simplicity: The All Convolutional Net”. In: *CoRR* abs/1412.6806 (2014). URL: http://arxiv.org/abs/1412.6806.

[50] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). URL: http://arxiv.org/abs/1409.4842.

[51] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *CoRR* abs/1312.6199 (2013). URL: http://arxiv.org/abs/1312.6199.

[52] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *CoRR* abs/1703.09438 (2017). URL: http://arxiv.org/abs/1703.09438.

[53] Shubham Tulsiani et al. “Learning Shape Abstractions by Assembling Volumetric Primitives”. In: *CoRR* abs/1612.00404 (2016). URL: http://arxiv.org/abs/1612.00404.

[54] G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. “Principal Component Analysis of Image Gradient Orientations for Face Recognition”. In: *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (FG’11)*. Santa Barbara, CA, USA, Mar. 2011, pp. 553–558.

[55] G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. “Subspace Learning from Image Gradient Orientations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.12 (2012), pp. 2454–2466.

[56] Sebastien C. Wong et al. “Understanding data augmentation for classification: when to warp?” In: *CoRR* abs/1609.08764 (2016). URL: http://arxiv.org/abs/1609.08764.

[57] Jiajun Wu et al. “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling”. In: *CoRR* abs/1610.07584 (2016). URL: http://arxiv.org/abs/1610.07584.

[58] Yuhuai Wu et al. “On the Quantitative Analysis of Decoder-Based Generative Models”. In: *CoRR* abs/1611.04273 (2016). URL: http://arxiv.org/abs/1611.04273.

[59] Zhirong Wu et al. “3D ShapeNets for 2.5D Object Recognition and Next-Best-View Prediction”. In: *CoRR* abs/1406.5670 (2014). URL: http://arxiv.org/abs/1406.5670.

[60] Xun Xu, Timothy M. Hospedales, and Shaogang Gong. “Multi-Task Zero-Shot Action Recognition with Prioritised Data Augmentation”. In: *CoRR* abs/1611.08663 (2016). URL: http://arxiv.org/abs/1611.08663.

[61] S. Zafeiriou et al. “Regularized Kernel Discriminant Analysis With a Robust Kernel for Face Recognition and Verification”. In: *IEEE Transactions on Neural Networks and Learning Systems (accepted for publication)* (2011).

[62] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). URL: http://arxiv.org/abs/1311.2901.

[63] Majd Zreik et al. “Automatic Segmentation of the Left Ventricle in Cardiac CT Angiography Using Convolutional Neural Network”. In: *CoRR* abs/1704.05698 (2017). URL: http://arxiv.org/abs/1704.05698.

# 8 Appendix

## 8.1 Denoising given Augmentation during training

Table 3 shows how the baseline VAE responds to noise in the input given varying amounts of augmentation during training. It is clear that augmentation during training increases the networks capabilities for de-noising and prediction but at the cost of fine-grained detail. This is most obvious by comparing the sliced desk reconstructed by the VAE w/ slice and the noiseless sofa reconstructed by the VAE w/o noise. The lack of noise during training allowedd the VAE to rapidly learn fine-grained detail such that its reconstruction of the sofa is the tidiest of all. The VAE trained with slice noise though is the only one capable of predicting that there are legs missing from the desk, even though its reconstructions with less noise are not so detailed. This interplay of de-noising and reconstruction is a likely sign that a further, more complex architecture will be capable of both with a higher accuracy.

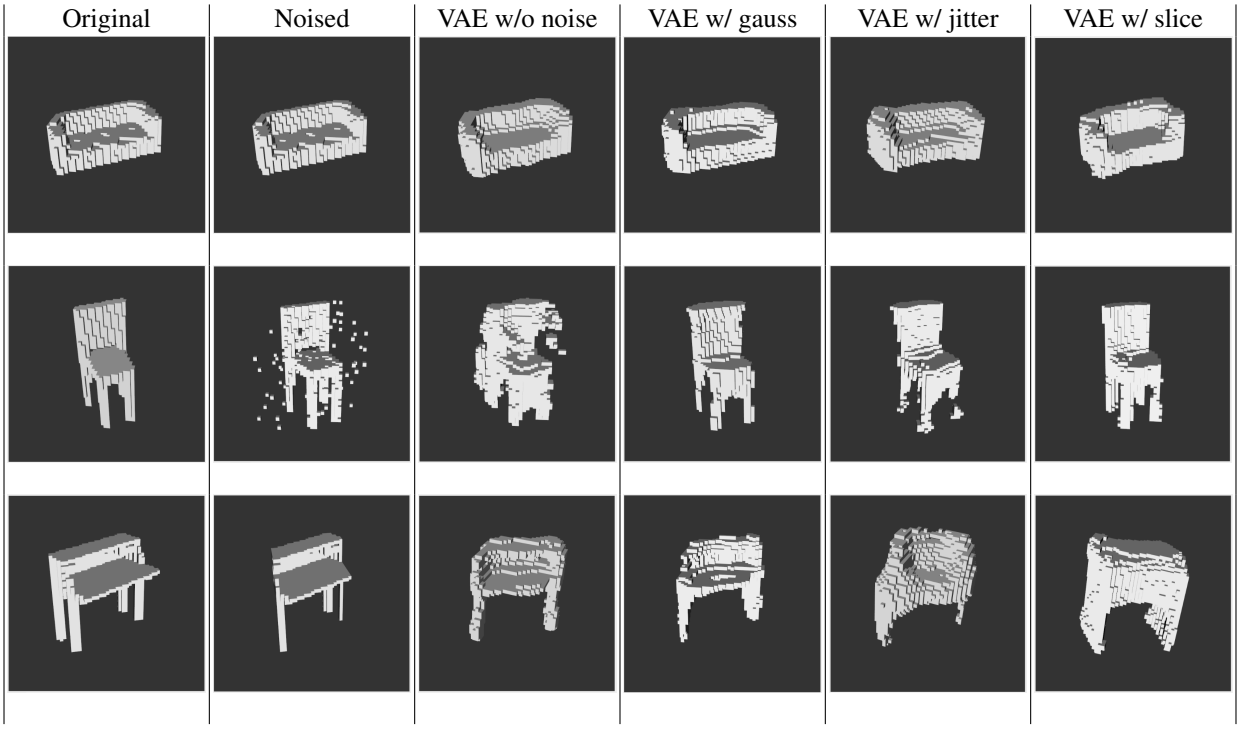



**Table 3.** VAE handling of varying noises given varied level of augmentation training.

## 8.2 Interpolation results

Table 4 shows the same interpolation series with respect to varying levels of noise during training. The start and end models are the original bed and sofa, taken from the test set, used for generating the latent space encodings. The models in between the originals are generated from linearly traversing the latent space between the encodings. The first row represents the latent space of a VAE without any augmentation training, the second row is with Gaussian noise, the third is with Translational noise and the fourth is with slicing noise as well. This figure is useful for expressing the similarity of the latent space representations for unknown data points, even when the trained with different noises. This implies that the learnt manifold is more general than the specific data used or how it is processed beforehand.

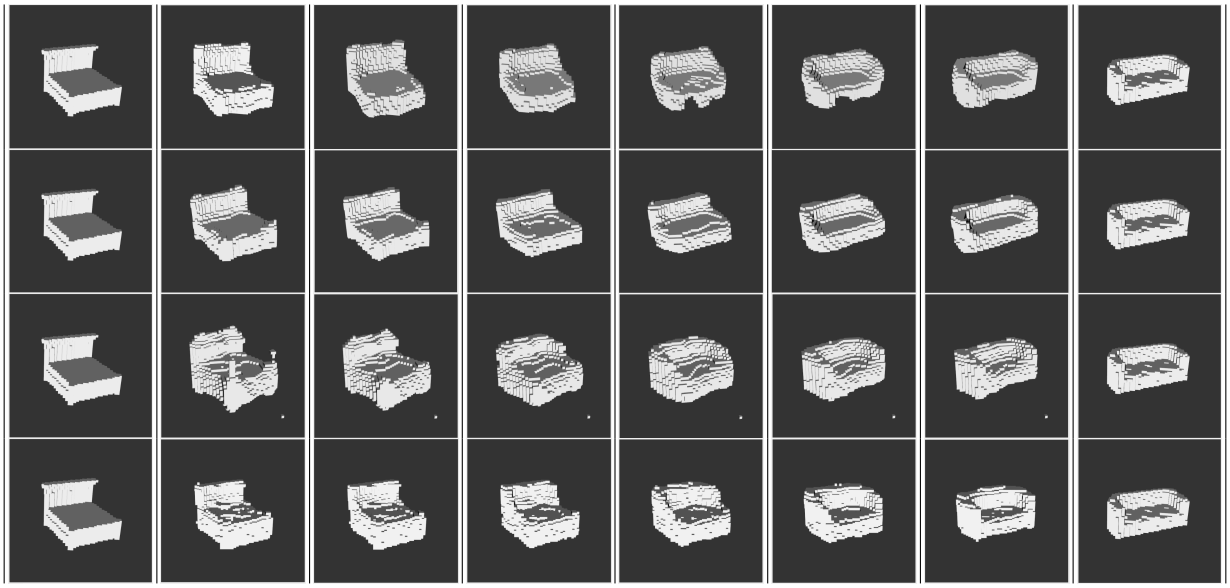


**Table 4.** VAE interpolation of ModelNet10 objects given different levels of augmentation at training, visualized using CaPtAEN

## 8.3 Loss Gamma

From studying voxel models at multiple resolutions, empirically it seems that a general solution for the modified loss gamma is given by $avg\_sparse(X) < \gamma < 1$, for the dataset $X$. Where $avg\_sparse(X)$ is the average, decimal, percentage of zeros throughout the dataset, eg 95% zeros == $\gamma = 0.95$. By varying $\gamma$, more focus can be placed on positive or negative prediction importance in the loss.

## 8.4 Architectures

### 8.4.1 VAE

Figure 28 shows the overview graph of the VAE network and training nodes as displayed by TensorBoard. Figure 29 shows the network architecture, figures 30 and 31 show the optimization and loss node respectively and figure 52

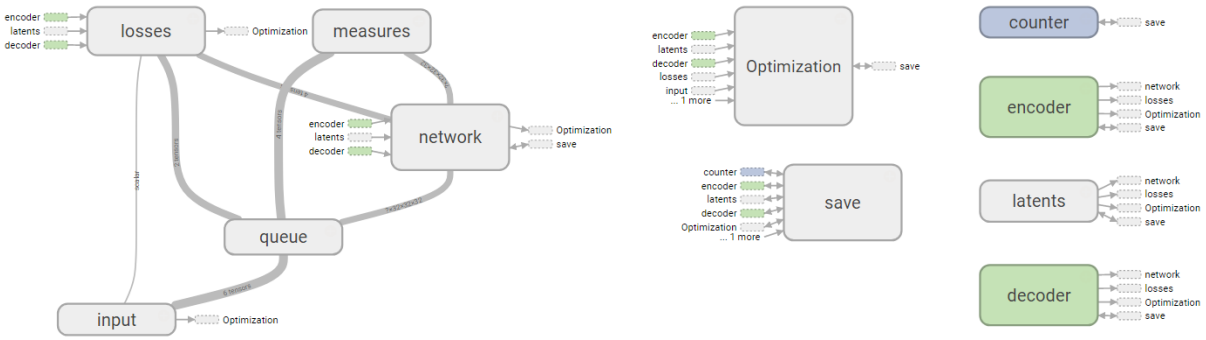



**Figure 28.** VAE overview

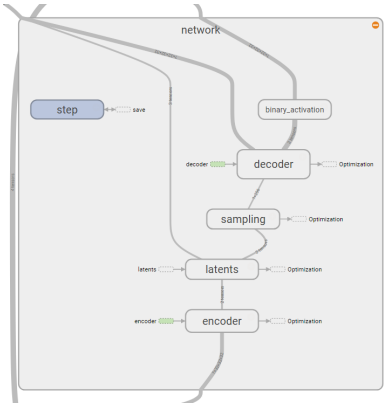



**Figure 29.** VAE network

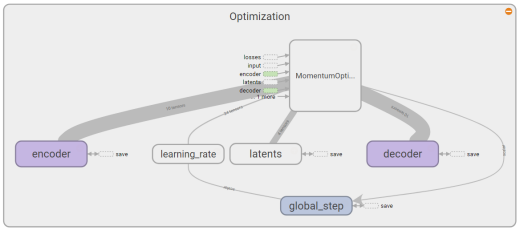



**Figure 30.** VAE optimization node

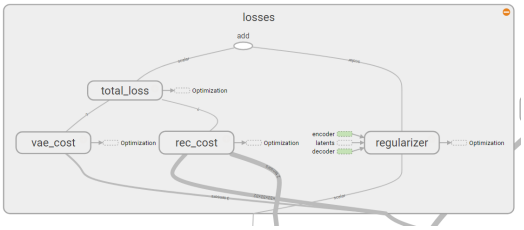



**Figure 31.** VAE loss node

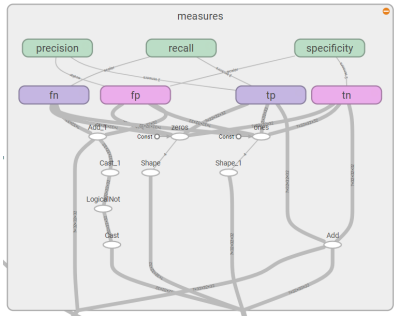



**Figure 32.** Training Metrics

Figures 33, 34, 35 and 36 show the Encoder, latent estimation, latent Sampling and Decoder respectively.

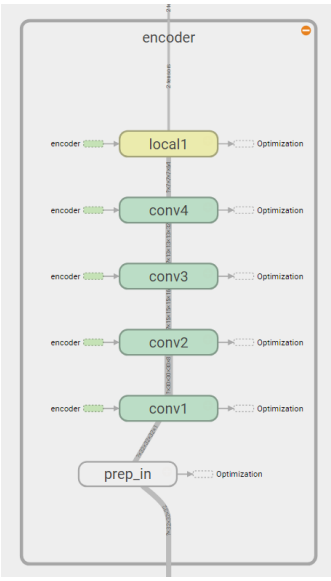



**Figure 33.** VAE Encoder architecture

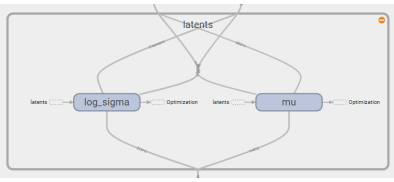



**Figure 34.** VAE latent estimation

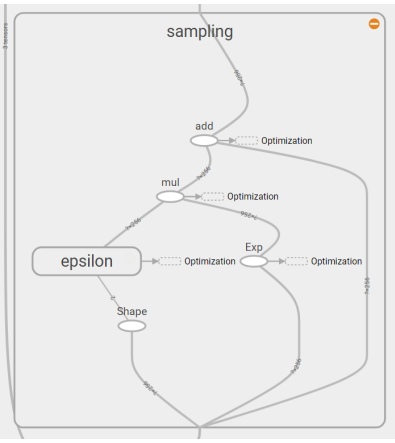



**Figure 35.** VAE latent Sampling

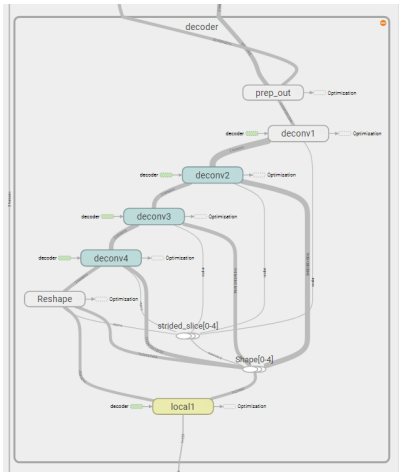



**Figure 36.** VAE Decoder architecture

### *8.4.2 PVAE*

Figure 37 shows the overview graph of the Prior-VAE network and training nodes as displayed by TensorBoard. Figure 38 shows the network architecture, figures 39 and 40 show the optimization and loss node respectively and figure 52

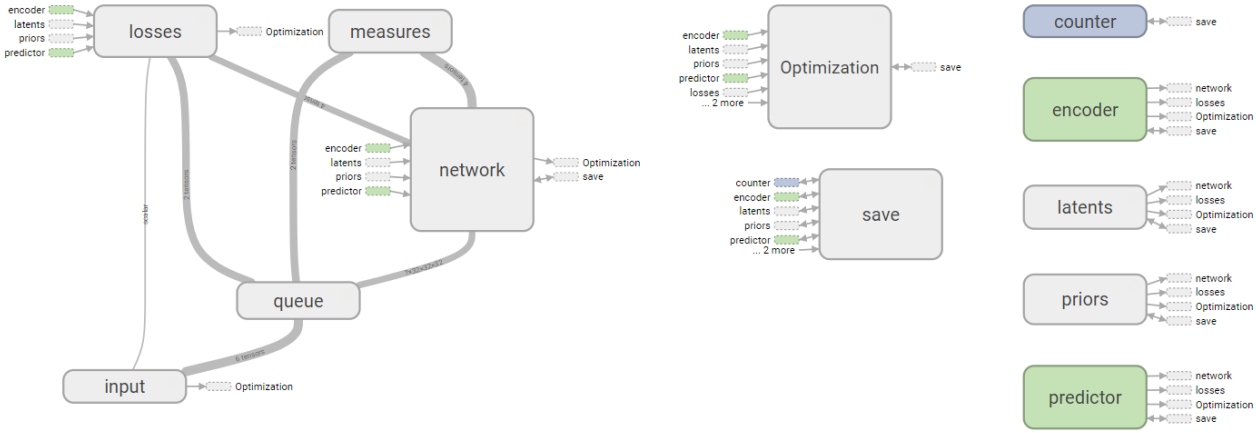



**Figure 37.** PVAE overview

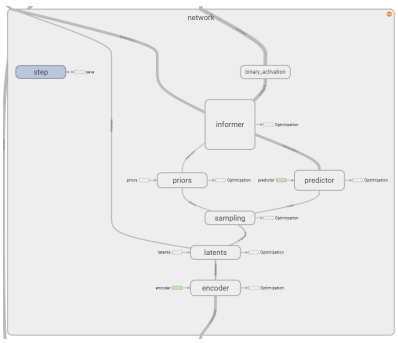



**Figure 38.** PVAE network

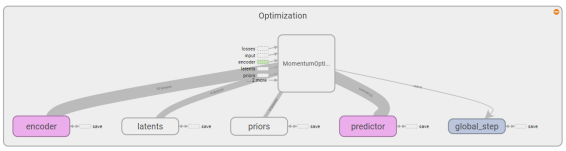



**Figure 39.** PVAE optimization node

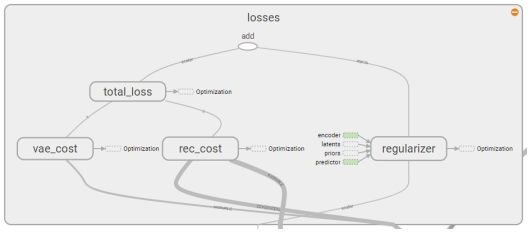



**Figure 40.** PVAE loss node

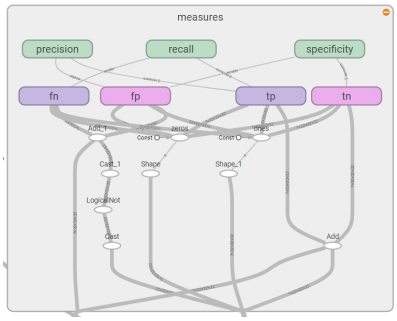



**Figure 41.** Training Metrics

Figures 42, 43, 44 and 45 show the Encoder, latent estimation, latent Sampling and Decoder respectively.

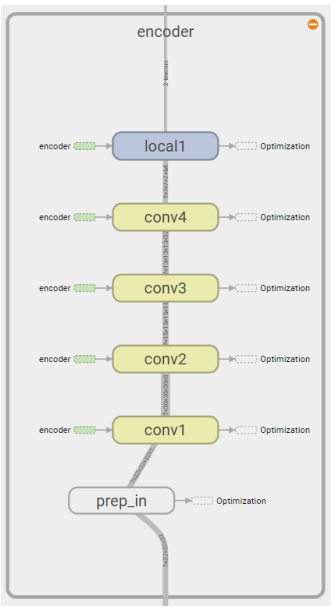



**Figure 42.** PVAE Encoder architecture

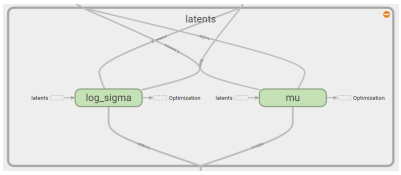



**Figure 43.** PVAE latent estimation

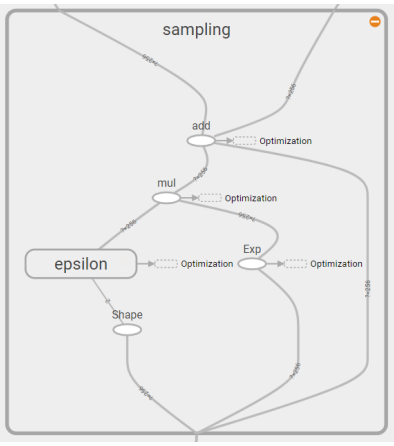



**Figure 44.** PVAE latent Sampling

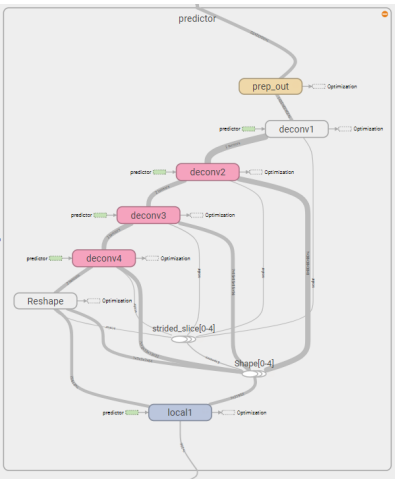



**Figure 45.** PVAE Decoder architecture

Figures 46 and 47 show the prior parameter estimate and multivariate gaussian informing nodes.

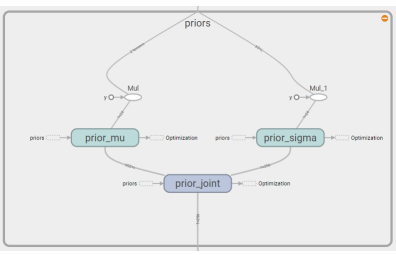



**Figure 46.** PVAE Prior estimation

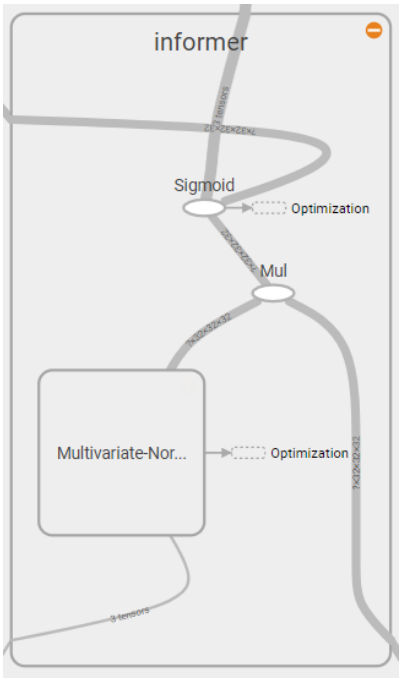



**Figure 47.** PVAE Informer architecture

### *8.4.3 LVAE*

Figure 48 shows the overview graph of the Located-VAE network and training nodes as displayed by TensorBoard. Figure 49 shows the network architecture, figures 50 and 51 show the optimization and loss node respectively and figure 52

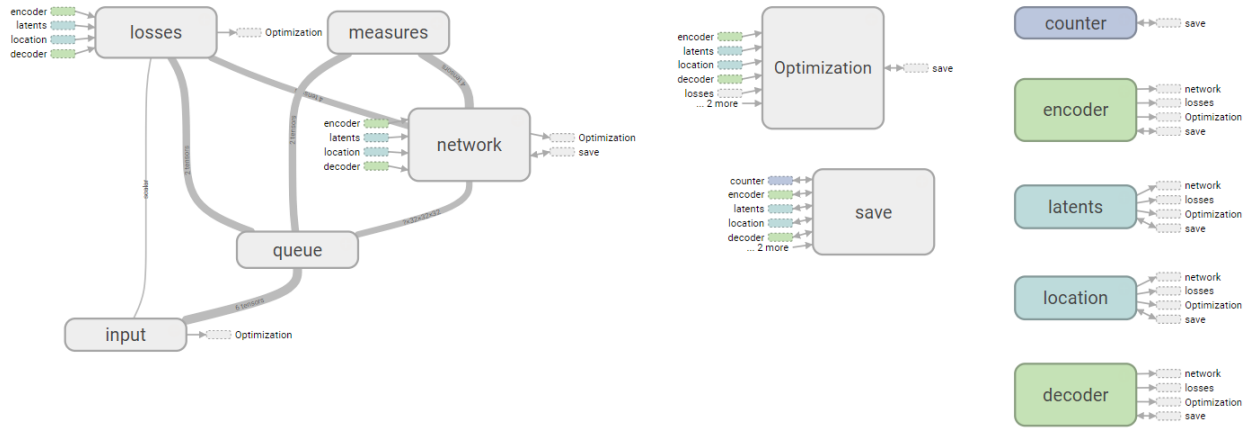



**Figure 48.** LVAE overview

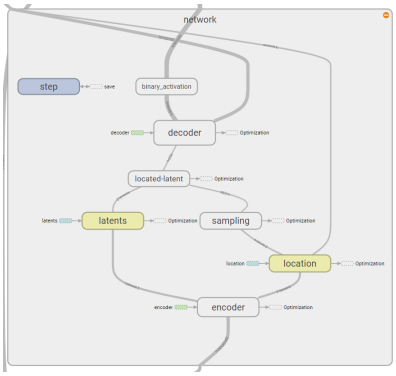



**Figure 49.** LVAE network

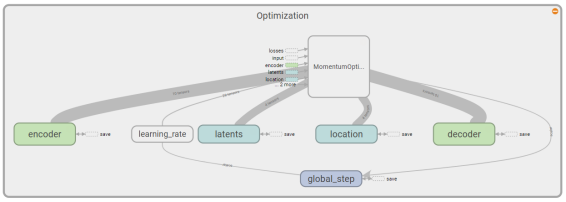



**Figure 50.** LVAE optimization node

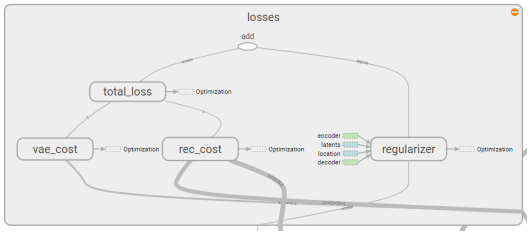



**Figure 51.** LVAE loss node

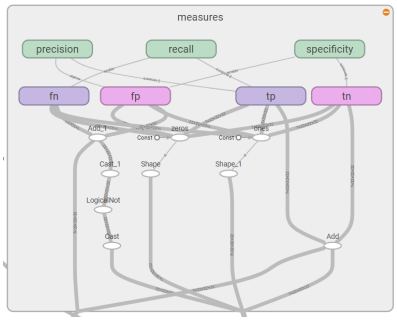



**Figure 52.** Training Metrics

Figures 53, 54, 55 and 56 show the Encoder, latent estimation, latent Sampling and Decoder respectively.

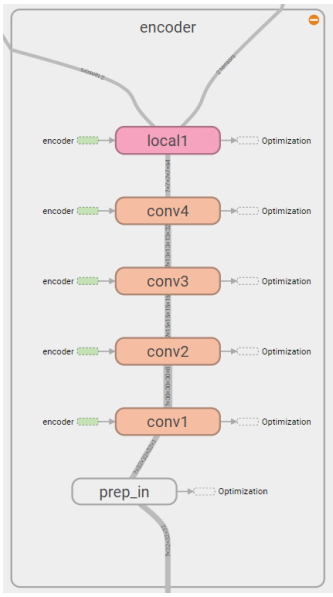



**Figure 53.** LVAE Encoder architecture

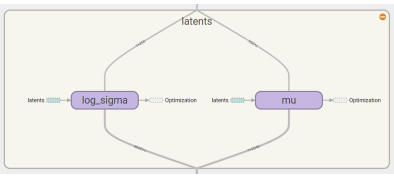



**Figure 54.** LVAE latent estimation

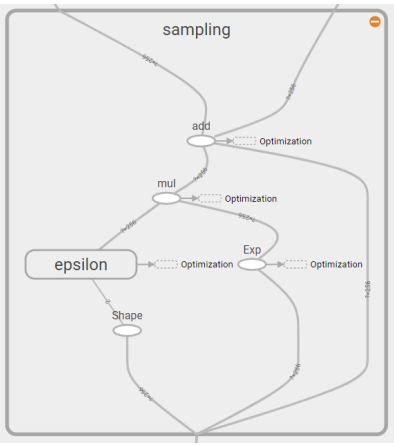



**Figure 55.** LVAE latent Sampling

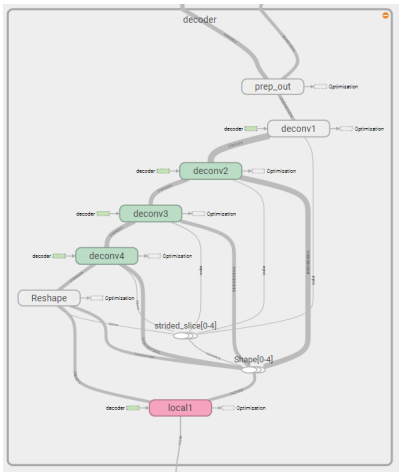



**Figure 56.** LVAE Decoder architecture

Figures 57 and 58 show the prior parameter estimate and multivariate gaussian informing nodes.

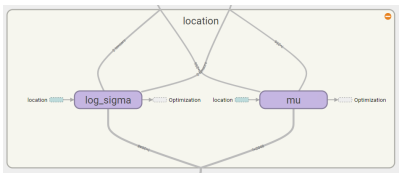



**Figure 57.** LVAE Prior estimation

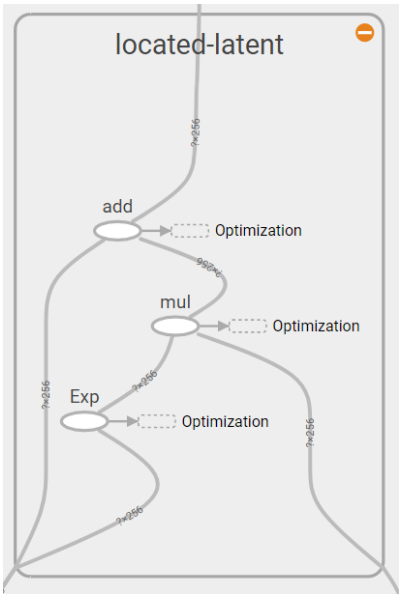



**Figure 58.** LVAE Informer architecture

## *8.5 CaPtAEN examples*

*8.5.1 Interfaces* Figure 59 and 60 show the noise selection and filter selection interfaces respectively.

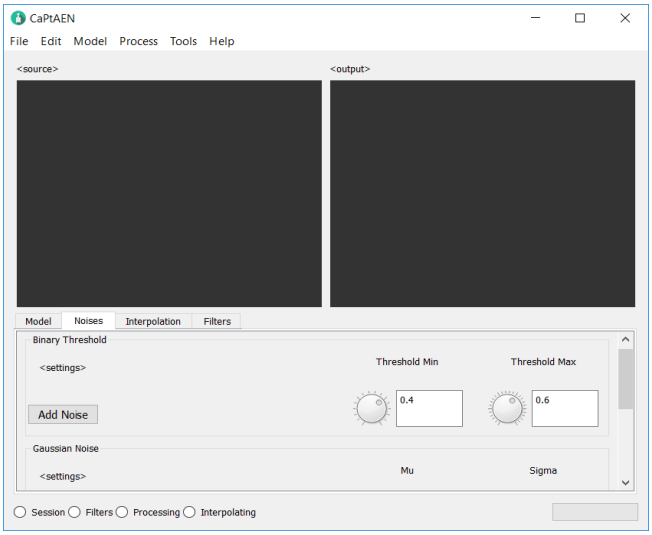



**Figure 59.** Noise selection

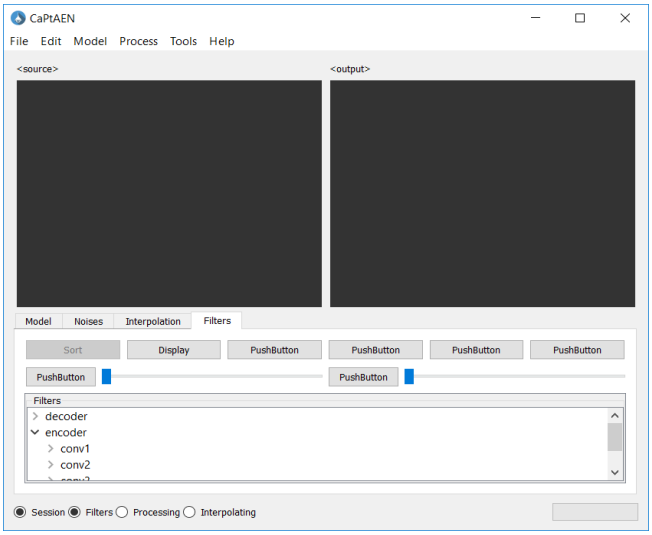



**Figure 60.** Display filters

*8.5.2 Object models* Figure 61 shows some examples taken from ModelNet10 viewed with CaPtAEN and figure 62 shows examples of segments from ShapeNetSem - Furniture.

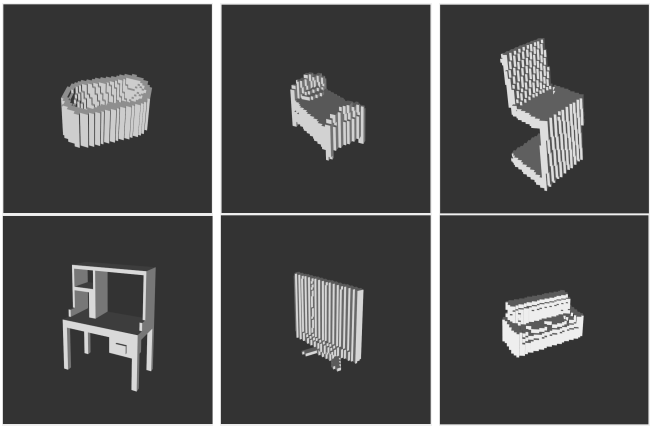


**Figure 61.** A selection of ModelNet10 objects as displayed by CaPtAEN

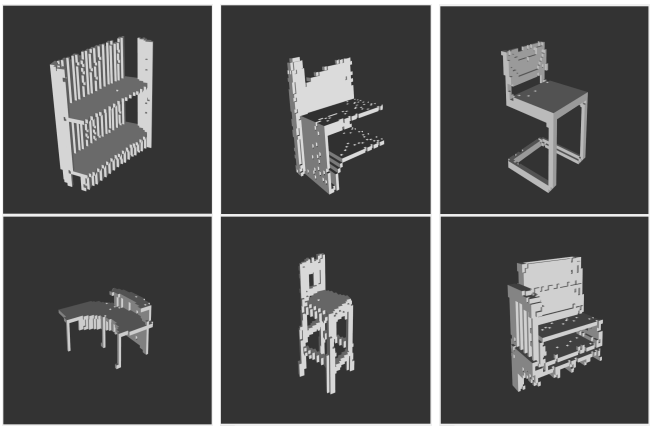


**Figure 62.** A selection of ShapeNetSem segments as displayed by CaPtAEN