Generative and Discriminative Voxel Modeling with Convolutional Neural Networks

Andrew Brock, Theodore Lim, J.M. Ritchie
School of Engineering and Physical Sciences
Heriot-Watt University
Edinburgh, UK
{ajb5, t.lim, j.m.ritchie}@hw.ac.uk

Nick Weston
Renishaw plc
Research Ave, North
Edinburgh, UK
Nick.Weston@renishaw.com

Abstract

When working with three-dimensional data, choice of representation is key. We explore voxel-based models, and present evidence for the viability of voxellated representations in applications including shape modeling and object classification. Our key contributions are methods for training voxel-based variational autoencoders, a user interface for exploring the latent space learned by the autoencoder, and a deep convolutional neural network architecture for object classification. We address challenges unique to voxel-based representations, and empirically evaluate our models on the ModelNet benchmark, where we demonstrate a 51.5% relative improvement in the state of the art for object classification.

1 Introduction

3D data offer computer vision systems a rich view of the world, but also pose a unique set of challenges, particularly in applications where understanding the surrounding environment is critical. In particular, data such as a point cloud extracted from an RGB-D image or a polygonal mesh are not guaranteed to be arranged in a regular grid, making them unsuitable for use with high-performance machine learning algorithms such as Convolutional Neural Networks (ConvNets). Deep ConvNets are currently used in state-of-the-art systems for a number of tasks in computer vision, and to date, the three most recent systems to achieve state-of-the-art performance in 3D object recognition on the ModelNet40 [1] benchmark have made use of 2D ConvNets pre-trained on ImageNet [2][3] [4] and evaluated using multiple rendered object views.

Voxel models, wherein object shape is represented as a binary occupancy grid, provide a representation suitable for use with ConvNets, but present a number of difficulties. The addition of a third spatial dimension in the regular grid comes with a corresponding computational cost, and the curse of dimensionality is a central issue, limiting the available resolution of the voxel grid. Low resolution grids make it difficult to differentiate between similar shapes,and toss some of the texture information available in 2D renderings of equivalent dimesnionality. Shallow 3D ConvNets have been evaluated on the ModelNet benchmark [5] [6], but are generally outperformed by multi-view 2D ConvNets.

Despite these challenges, we posit that deep ConvNets are viable for use in modeling voxel-based 3D objects for both generative and discriminative tasks. In this work, we present deep ConvNet architectures for both generative and discriminative voxel modeling, and explore issues specific to voxel-based representations. Our generative methods display high fidelity shape interpolation, and our discriminative methods outperform the current state of the art by a relative 51.5% and 53.2% on the ModelNet40 and ModelNet10 benchmarks.

3D Deep Learning Workshop at NIPS 2016, Barcelona, Spain.

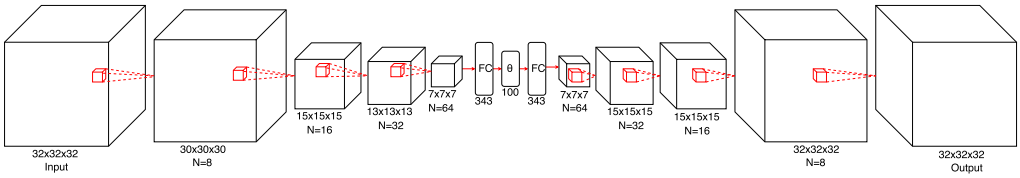


Figure 1: VAE Architecture.

2 Voxel-Based Variational Autoencoders

Interpolating between binary grids permits no obvious mathematical interpretation; if we wish to learn how voxellated objects relate to one another, we require a model capable of reasoning in an abstract feature space that captures the salient factors of variation. For this work, we select the Variational Autoencoder (VAE) [7], a probabilistic framework that learns both an inference network to map from an input space to a set of descriptive latent variables, and a generative network that maps from the latent space back to the input space. As is typical, we model the latents as an isotropic Gaussian, such that the encoder network learns to output μ and σ, and the input to the decoder network is taken as $\mu + \sigma * \mathcal{N}(0, I)$ during training and μ during testing. This prior is enforced by augmenting the objective with a KL divergence term between the output values of μ and σ and the desired values of $\mu = 0, \sigma = 1$.

By training a network to infer the latent variables which describe the underlying factors of variation between objects, we gain the ability to smoothly transition between objects by interpolating between each object's latent description and reconstructing using the decoder network.

2.1 Model Architecture

Our model, implemented in Theano[8] with Lasagne,[1] comprises an encoder network, the latent layer, and a decoder network, as displayed in Figure 1. The encoder network consists of 4 convolutional layers and a fully connected layer, followed by a linear projection from the fully connected layer to the latent layer. The decoder network has an identical, but inverted, architecture, and its weights are not tied to the encoder's. 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.

All layers use the Exponential Linear Unit (ELU) [9] nonlinearity, with the exception of the final layer, which uses a sigmoid nonlinearity. The output of each element of the final layer can be interpreted as the predicted probability that a voxel is present at a given location. Downsampling in the encoder network is accomplished via strided convolutions (as opposed to pooling) in every second layer. Upsampling in the decoder network is accomplished via fractionally strided convolutions, implemented as the gradient of an equivalent strided convolution[10], in every second layer.

The network is initialized with Glorot Initialization [11], and all but the output layer are Batch Normalized[12]. The variance and mean parameters of the latent layer are individually Batch Normalized, such that the output of the latent layer during training is still stochastic under the VAE parameterization trick.

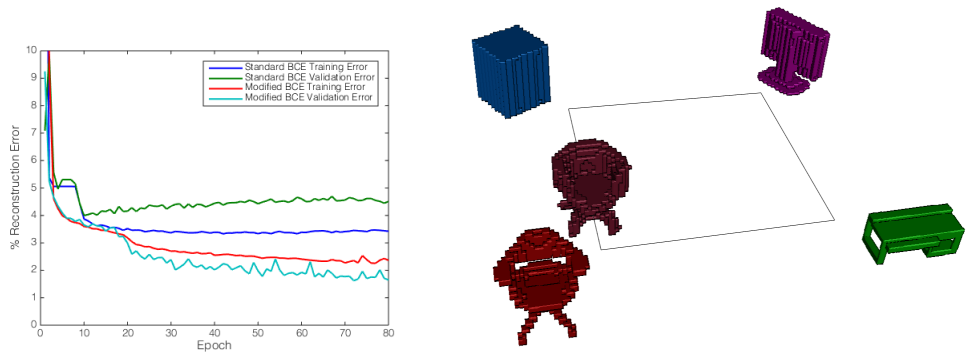
2.2 Loss Function

The loss function consists of the KL divergence prior on the latents, L2 weight regularization, and the reconstruction error, for which we use a specialized form of Binary Cross-Entropy (BCE). The standard BCE loss is:

$$\mathcal{L} = -t\,log(o) \; - \; (1-t)\,log(1-o)$$

Where t is the target value in $\{0,1\}$ and o is the output of the network in (0,1) at each output element. The derivative of the BCE with respect to o severely diminishes as o approaches t, which can result in vanishing gradients during training. Additionally, the standard BCE weights false positives and false

[1] https://github.com/Lasagne/Lasagne



(a) (b)

Figure 2: Comparison of training regimes (a), user interface (b).

negatives equally; because over 95% of the voxel grid in the training data is empty, the network can confidently plunge into a local optimum of the standard BCE by outputting all negatives.

We make two key modifications to the BCE to improve training. First, we change the range of the target and output to {-1,2} and [0.1,1),respectively. This change increases the magnitude of the loss gradient throughout the domain of o, reducing the probability of vanishing gradients. Second, we add a hyperparameter γ which weights the relative importance of false positives against false negatives:

$$\mathcal{L} = -\gamma\, t\, log(o) \; - \; (1-\gamma)\,(1-t)\, log(1-o)$$

During training, we set γ to 0.97, strongly penalizing false negatives while reducing the penalty for false positives. Setting γ too high results in noisy reconstructions, while setting γ too low results in reconstructions which neglect salient object details and structure.

2.3 Training

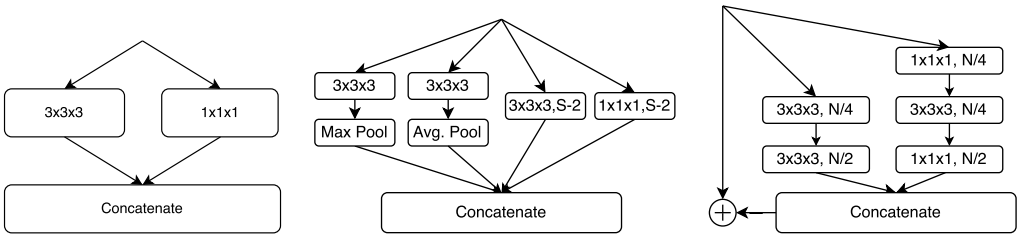
The model is trained using stochastic gradient descent with Nesterov momentum [13] for 100 epochs, or until the reconstruction error on a held-out validation set bottoms out. The learning rate is set to 0.0001 for the first epoch, then increased to 0.001. The data is augmented by adding random translations and horizontal flips to each training example, as in [5], then training on one noisy and one uncorrupted copy of each instance, randomly shuffled. By training the network to reconstruct both corrupted and uncorrupted data, we force it to learn invariance to small structural variations.

We first validate our modification to the BCE by comparing the validation errors of two identically initialized networks, one trained with the standard BCE, and one trained with our modification. The reconstruction error is plotted against training epochs in Figure 2(a). Interestingly, we note that the validation error is lower than the training error for this particular training run. We experiment with a number of different model architectures and training regimes before converging on the final method detailed above. In particular, we experiment with augmenting the training objective by adding a 10-unit fully connected softmax layer for classification in parallel with latent estimation, as well as a denoising objective, neither of which result in any observable performance improvement.

2.4 User Interface

We present a graphical user interface modeled after [14]. The interface, implemented using VTK[15], allows the user to drag a center object that interpolates between up to four different objects, and supports class-unconditional random shape generation. The interpolant endpoint models are randomly selected from the ModelNet10 test set at runtime, and both inference and reconstruction run in real time on a laptop with a GT730m graphics card. Figure 2(b) shows a screenshot of the interface, and a video of the interface in action is available online.[2]

[2]https://www.youtube.com/watch?v=LtpU1yBStlU



Voxception | Voxception-Downsample | Voxception-Resnet

Figure 3: Voxception and Voxception-Resnet Blocks.

3 Voxel-Based Convnets for Classification

Voxel-based ConvNets were first applied to 3D object recognition in Voxnet[5], a shallow volumetric ConvNet architecture, and were also used in ORION[6], wherein the classification task was augmented with an orientation estimation task. A recent extension, FusionNets[4], combines ConvNets trained on voxellated models and pre-trained ConvNets fine-tuned on rendered object views.

3.1 Architecture

Our model is designed in line with approaches used for high performance 2D ConvNets for object classification. Key to our approach is the use of Inception-style modules[16], Batch Normalization[12], Residual connections with pre-activation [17][18] and stochastic network depth [19]. In contrast to previous 3D ConvNet approaches which used shallow networks, we train networks with up to 45 layers to take advantage of the increased expressivity that comes with model depth. Compared to FusionNets[4], our model requires significantly fewer parameters (18M as opposed to 118M for a full FusionNet) and fewer object views (12 or 24, compared to 60). Code to train and test our models is publicly available.[3]

3.1.1 Voxception

After initial tests with vanilla ConvNets, we adopted a simple Inception-style architecture. The intuition behind the design was to maximize the number of possible "pathways" for information to propagate through the network, while still maintaining simplicity and efficiency.

For non-downsampling layers (Figure 3, left) , we concatenate equal numbers of 1x1x1 and 3x3x3 filters, allowing the network to choose between taking a weighted average of the featuremaps in the previous layer (i.e. by heavily weighting the 1x1x1 convolutions) or focusing on spatial relationships (i.e. by heavily weighting the 3x3x3 filters). For downsampling layers (Figure 3, center), we stack 3x3x3 convolutions with strided pooling operations, using both max and average pooling, and concatenate those features with strided 3x3x3 and 1x1x1 convolutions. Our intent is for the downsampling layers to let the network learn the best relative weighting of the various downsampling methods, maximizing propagation of information while still producing a more compact representation. Our final model is nine layers deep, with four Voxception blocks and three Voxception Downsample blocks, followed by two fully connected layers and a softmax nonlinearity.

3.1.2 Voxception-ResNet

The Voxception-ResNet (VRN) architecture is based on the ResNet architecture[17], but concatenates both the ResNet Bottleneck Block and the standard ResNet block into a single Inception[16]-style block (Figure 3, right). To improve parameter efficiency, the early layers in each path of the block have half as many filters as the final layer. Downsampling is accomplished through Voxception-Downsample blocks, which we do not change in our ResNet model. We change the order of application of rectifying nonlinearities and Batch Normalization to obtain pre-activation blocks[18]. Finally, we stochastically drop the non-residual paths of blocks[19], where the keep probability is linearly decreased from 1.0 in the first layer to 0.25 in the final VRN layer, and used as a weighting value instead of a drop probability at test time.

[3] https://github.com/ajbrock/Generative-and-Discriminative-Voxel-Modeling

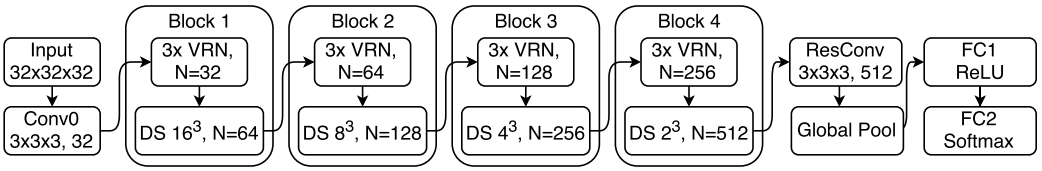


Figure 4: Voxception-ResNet 45 Layer Architecture. DS are Voxception-Downsample blocks.

Our best-performing architecture is shown in Figure 4, and consists of an initial convolutional layer, four main units, each containing three stacked VRN blocks and a Voxception-Downsample block, a final convolution with a residual connection and keep probability of 0.5, then a global pooling layer and two fully-connected layers. The number of filters begins at 32, and is doubled at each downsampling block. The deepest path through the network is 45 layers (going through the 3-layer section of the VRN block), and the shallowest path (assuming all droppable non-residual paths are dropped) is 8 layers deep.

3.2 Data Augmentation and Training

Unless otherwise specified, all models were initialized using Orthogonal Initialization[20], Batch Normalized[12], and trained using Nesterov Momentum[13] with a momentum value of 0.9. Other than the final softmax nonlinearity, Exponential Linear Units[9] are used as activations throughout the network. We initially experimented with standard ReLU but found that ELUs tended to have higher average performance for models with 15 or more layers, perhaps because of the improved gradient flow provided by ELUs. We change the binary voxel range from {0,1} to {-1,5} to encourage the network to pay more attention to positive entries.

We experiment with multiple learning rate decay schemes and find that dividing the learning rate by a factor of 2 every time the validation loss bottoms out to be more effective than annealing at a constant rate after a set number of minibatches or epochs. Initial hyperparameter studies were validated using a held-out, class-balanced tenth of the training set, but for final evaluation the annealing schedule was fixed and the entire training set was used.

We voxellate the data (which comes as a set of vertices and edges) using tools provided by [1] to center and scale each instance before discretization. During each epoch, we train on two copies of each example, where one of the copies is randomly flipped about a horizontal axis and/or translated, as was done in VoxNet[5]. We use a fixed random seed scheme to ensure that different training runs make use of identically augmented datasets. We use two versions of the training set, one with 12 rotations of each instance, and one with 24 rotations of each instance. For our best performing models, we warm up the network by training for twelve epochs on the 12-rotation training set, then anneal the learning rate and fine-tune on the 24-rotation training set. During testing, we measured predictions on a single view and averaged predictions across 24 rotated copies of each instance. We found this data augmentation to be essential for training deeper networks, especially Voxception-Resnet. We produce a simple ensemble by summing predictions from five VRN models and one Voxception model. A single training epoch, using a batch size of 50 and the full 24-rotation augmented dataset, takes around 6 hours on a single Titan X, and most models require around 6 days of training to converge.

We also experiment with treating the different rotations as separate channels for a single instance (analagous to RGB channels in a 2D image) but found that training a model to look at a single orientation and averaging predictions across rotated versions of an instance yielded better performance. We suspect that this is because, without otherwise changing the model architecture, the rotation-channel network must still pass information through equivalent representational bottlenecks, as opposed to being able to separately evaluate each rotation as individual instances. Additionally, we experimented with a variety of values for the range of the binary voxel grid, including an adaptive method wherein the numerical value of positive entries is equal to the 100 times the percentage of the grid occupied by the object, but did not find these methods to significantly alter performance.

We experimented with stacking 3x1x1,1x3x1,1x1x3 blocks in place of 3x3x3 convolutions, but found that this did not noticeably affect performance or training time. We tried to use the Adam and Adamax[21] optimizers but found that doing so reduced generalization performance.

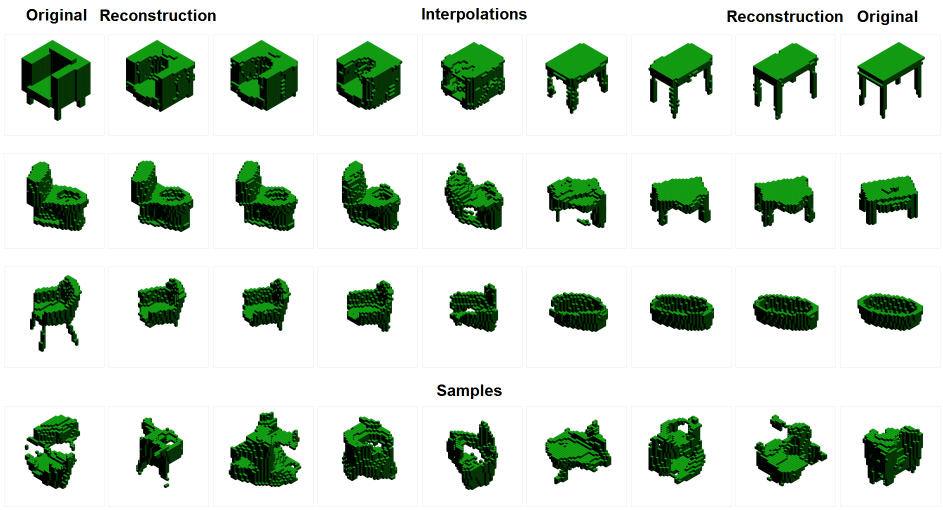


Figure 5: Voxel-Based VAE Reconstructions, Interpolations, and Samples.

We also found that aggressively downsampling early in the network by replacing the initial convolution with a strided convolution enabled us to train significantly deeper models, but that such networks underperformed their 45 layer counterparts. We theorize that placing an early representational bottleneck causes the network to toss features that it might otherwise learn to keep or propagate to deeper layers, placing an upper bound on performance, or that we simply did not provide sufficient data to train networks of such depth.

4 Results

4.1 Voxel-based VAE

The reconstruction accuracy of our fully-trained VAE, evaluated on the ModelNet10 test set, is displayed in Table 1. The model attains a 99.39% true positive and 92.36% true negative reconstruction accuracy on the ModelNet10 test set, indicating that it learns to reconstruct with high fidelity, but tends to slightly overestimate the probability of a voxel being present. Reconstruction and interpolation examples are displayed in Figure 5 alongside class-unconditional random samples.

4.2 Object Classification

The accuracy of our discriminative models is evaluated and compared against competing approaches in Table 2. Our best single VRN model obtains 91.33% accuracy on the ModelNet40 test set, and 93.61% accuracy on the ModelNet10 test set, which are respectively better than any previous published results and competitive with the current state of the art. Our best ensemble of VRN models obtains state-of-the art 95.54% and 97.14% accuracy on both ModelNet subsets, improving the state of the art by a relative 51.5% and 53.2%, respectively. Our best VRN model obtains 88.98% accuracy when tested using a single view of the object. Our best ensemble of models trained solely on ModelNet10 obtains 94.71% accuracy, which is also better than any previously published result; we report our ModelNet10 accuracy for models trained on ModelNet40 for consistency with previous results[2][3][4].

5 Discussion

5.1 Voxel-Based VAE

The network achieves passable reconstruction accuracy, and learns to smoothly interpolate between arbitrary, previously unseen shapes. The network is additionally capable of generating random shapes

Table 1: Reconstruction Results.

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

Table 2: Classification Results.

Model	ModelNet40	ModelNet10
Voxnet [5]	83.00%	92.00%
MVCNN [2]	90.10%	-
Pairwise [3]	90.70%	92.80%
FusionNets [4]	90.80%	93.11%
ORION [6]	-	93.80%
MVCNN-Multires [22]	91.40 %	-
Voxception	90.56%	93.28%
VRN (One-View)	88.98%	-
VRN	91.33%	93.61%
VRN Ensemble	**95.54%**	**97.14%**

with consistent structure, indicating that the learned latent space is successful in disentangling the factors of structural variation, though these new shapes.

The network performs well for dense objects, particularly thick dense objects such as sofas and toilets, but occasionally struggles to reconstruct objects with long, thin members, such as tables or chairs. We suspect these features are too small to activate in the receptive field of the appropriate latents, and are lost in favor of denser features which weigh more heavily in the loss function, and suggest imposing an additional "local" reconstruction function that measures reconstruction accuracy in subsets of the voxel grid, such that the network learns to reconstruct features regardless of how small they are relative to the entire object.

The network also struggles to reproduce crisp edges, preferring to output smooth, rounded edges. We posit that this is analogous to the way in which a vanilla 2D VAE will tend to output images with blurry edges rather than crisp edges to avoid overconfidently making incorrect predictions.

5.1.1 Interpolation

The system is capable of smoothly interpolating between reconstructions, indicating that it learns a representation which captures the underlying factors of structural variation. For example, when interpolating between two objects of the same class but slightly different orientation, the model will make only minimal changes in the output during interpolation, rather than completely deconstructing and reconstructing the output (i.e. passing through the origin of the latent space).

When interpolating between drastically different objects, the interface exhibits a “flowing water” effect, wherein preexisting voxels will appear to smoothly shift between shapes, rather than appearing at random, as can be seen in Figure 5.

5.1.2 Sampling

Samples generated by our model are shown in the last row of Figure 5. Our samples consistently bear a semblance of structure, with few to no free-floating voxels, suggesting that the decoder network has learned to maintain output voxel connectivity regardless of the latent configuration. The major limitation of the VAE is that its generated samples do not, however, resemble real objects. We hypothesize that training a deeper, more expressive model on the ModelNet40 dataset, and augmenting the latent vector with a class-conditional vector, would enable the generation of objects which clearly belong to a particular class, but leave such an investigation to future work.

5.2 Object Classification

Our VRN model takes advantage of the increased expressivity associated with its substantially increased depth to achieve significant improvements on the ModelNet classification benchmark.

We found that averaging predictions across rotations was critical to achieving top performance, but that even taking predictions from a single view resulted in passable 88.98% ModelNet40 accuracy, and even an ensemble of 2 models predicting on a single view achieves over 91% accuracy. We also found that ensembling with a mix of predictions averaged over 12 and 24 rotations resulted in even

higher test performance on ModelNet40 (95.78%), but we suspect that this result is not general, and do not claim it with our main results.

Our best single model is better than all previously published single models, and is competitive with the best previously-published ensemble method of [22], wherein an SVM is trained on the final fc layer of three Multi-View 2D CNNs operating on sphere-rendered instances of various resolution. Our model is competitive with, but does not outperform the previous state of the art on ModelNet10, ORION[6], which augments the classification task with a rotation estimation task. We suspect that our deeper models do not perform quite as well when trained on the smaller subset due to a lack of data, with only 3991 training instances in ModelNet10 compared to 9843 instances in ModelNet40. Additionally, ORION[6] incorporates class-specific priors to determine precisely which rotations to train on, which we eschew in favor of generality.

Our (by no means novel) hypothesis that the upper bound on model depth is dependent on the amount of available data is consistent with our observations that significant data augmentation was required to train our 45-layer model. We also note that our highest performance on ModelNet10 came from models trained on ModelNet40, despite ModelNet40 not containing any additional ModelNet10 instances. This is interesting from a transfer learning perspective: by learning to distinguish between a wider variety of classes, the model learns to better discriminate between a given subset of classes.

5.2.1 Suggestions for Future Work

We believe that there still remains plenty of low-hanging fruit to be gained by investigating deep ConvNets for 3D object classification, and provide several suggestions for improvement:

- Refine the voxel grid beyond 32x32x32 to increase spatial resolution and improve the available detail.
- Change the voxel grid to be real-valued based on the percentage of space occupied by the instance at each grid element. This could also be used with the VAE to allow it to represent more complex shapes by fitting a corner-connected polyhedron with volume equal to the predicted occupancy percentage.
- Experiment with more data augmentation, such as rotating about random axes (or just adding more rotations about the central axis), random crops, or random rescaling.
- Try out different Voxception architectures, downsampling methods, and activation functions. Our experiments focused primarily on high-level network architecture, and there are likely more effective ways to compose a Voxception-ResNet block.

6 Conclusion

We presented a voxel-based Variational Autoencoder and a graphical user interface for exploring the latent space of 3D generative models, along with a voxel-based deep convolutional neural network for classification. Our methods take into account challenges specific to voxel representations, and demonstrate the viability of voxel representations in discriminative tasks by improving the state of the art on the ModelNet classification task by large margins.

Acknowledgments

This research was made possible by grants and support from Renishaw plc and the Edinburgh Centre For Robotics. The work presented herein is also partially funded under the European H2020 Programme BEACONING project, Grant Agreement nr. 687676.

References

[1] Z. Wu, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR 2015*.

[2] H. Su, S. Maji, E. Kalogerakis, and E.Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV 2015*.

[3] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR 2016*.

[4] V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. arXiv Preprint arXiv: 1607.05695, 2016.

[5] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS 2015*.

[6] N. Sedaghat, M. Zolfaghari, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. arXiv Preprint arXiv: 1604.03351, 2016.

[7] D.P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR 2014*.

[8] The Theano Development Team. Theano: A python framework for fast computation of mathematical expressions. arXiv Preprint arXiv: 1605.02688, 2016.

[9] D-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR 2016*.

[10] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. arXiv Preprint arXiv: 1603.07285, 2016.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS 2010*.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML 2015*.

[13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML 2013*.

[14] M. Yumer, P. Asente, R. Mech, and L. Kara. Procedural modeling using autoencoder networks. In *UIST 2015*.

[15] W. Schroeder, K. Martin, and B. Lorenson. *The Visualization Toolkit,*. Kitware, 4 edition, 2006.

[16] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv Preprint arXiv: 1602.07261, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR 2016*.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. arXiv Preprint arXiv: 1603.05027, 2016.

[19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. arXiv Preprint arXiv: 1603.09382, 2016.

[20] A.M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR 2014*.

[21] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv Preprint arXiv: 1412.6980, 2014.

[22] C.R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR 2016*.