LatEd: A Geometric Latent Vector Editor

Alexander Komar
a.komar@cgv.tugraz.at

Michael Kammerer
michael.kammerer@student.tugraz.at

Saeedeh Barzegar Khalilsaraei
s.barzegar@cgv.tugraz.at

Ursula Augsdörfer
u.augsdorfer@cgv.tugraz.at

Institute of Computer Graphics and Knowledge Visualisation, Graz University of Technology, Austria

ABSTRACT

Using a neural network approach, a shape may be compressed to a one-dimensional vector, the so-called latent dimension or latent vector. This latent shape dimension is examined in this paper.
This latent vector of a shape is used to identify the corresponding shape in a database. Two types of networks are evaluated in terms of lookup accuracy and reconstruction quality using a database of Lego pieces. Even with small training set a reasonable robustness to rotation and translation of the shapes was achieved.
While a human can interpret uncompressed data just fine, the compressed values of the network might be cryptic and thus offer no insight regarding the uncompressed input. Therefore, we introduce a latent dimension editor which allows the user to examine the geometry content of the latent vector and its influence on the decoded shape. The latent vector editor enables the visual exploration of the latent vector, by making changes to the latent vector visible in real-time via a 3D visualization of the reconstructed object.

Keywords

Neural Networks, GAN, Signed Distance Fields

1 INTRODUCTION

Extracting the defining features of a geometry allows us to represent a 3D object more efficiently, e.g. one could describe a Lego piece by precisely defining its geometrical shape, even though the same information can be condensed into its structural attributes, such as size and stud count.

In this paper we use neural networks to extract condensed information to a concise descriptor with a lower dimension than the original representation of the provided object.

Autoencoders [1], a subclass of neural networks (NN), are used to learn an efficient encoding of their input data in terms of latent dimensions in compact form. The autoencoder first encodes the input into a lower dimensional latent representation. The outcome of the compression is then decoded to reconstruct the original input.

The latent representation is a central part of this paper. The latent shape descriptor is useful in many applications, e.g. for storing large amounts of data efficiently due to its dimension and size.

To assess the applicability of using the latent vector in context for 3D object retrieval (3DOR) we derive the database lookup accuracy for a range of Lego pieces. Using a Lego database [12] as an example, we demonstrate that a fast and accurate retrieval of an arbitrary 3D shape can be achieved.

To visually examine the latent vector and its geometry encoding, we present a real-time latent vector editor. By visualizing the reconstructed 3D objects such that alteration to the shape latent dimension becomes visible in real-time, the user gets immediate feedback on how any changes applied to the latent vector affect the reconstruction and, thus, the encoding of the geometry. This feature opens up a world of exciting experiments in which the user gains insight into how the latent vector relates to the uncompressed input geometry. While in this paper we focus on two shape representations, namely a voxel or signed distance field (SDF) representation the approach may be extended to other shape representations.

2 RELATED WORK

Neural networks in combination with 3D voxel grids and SDF have been used frequently in context of machine learning geometry.

Voxel grids have been employed e.g. in surface reconstruction by Brock et al. [3]. Their generative convolu-

tional autoencoders achieved a good reconstruction for 3D voxel grids, and discriminative convolutional neural networks achieved a robust classification of the latent dimension values. Wang et al. [16] proposed a network which uses voxel and SDF data of an object as its inputs. It processes the given data using a convolutional neural network, autoencoders, and an extreme learning machine. The primary purpose of said configuration is for 3D shape classification tasks. Maturana and Scherer [13] proposed a classification network, that uses a voxel occupancy grid as an input. They also showed some robustness to rotation. Wu et al. [18] proposed a method to recognize different shapes as voxel grids from a 2.5D input image. They applied their method to real world data.

Wang et al. [15] propose a new way of storing information in a Oct-tree and using it as an input for a Neural Network. They show their method on object classification, shape retrieval and shape segmentation.

Wohlhart et al. [17], trains a convolutional neural network (CNN) to perform object recognition and 3D pose estimation by computing the similarity of different descriptors through their Euclidean distance and employing nearest neighbor search to handle large amounts of objects.

The method proposed by Girdhar et al. [8] works by combining two encoder networks to produce one latent vector and one decoder network reconstructing a voxelized representations of the shape. In the testing phase the shape is reconstructed from the image alone. They further analyzed the latent vector for meaningful feature values.

DeepSDF [14] focuses on an auto-decoder neural network learning a continuous signed distance function representation of a class of shapes for shape representation, interpolation, and completion, even from partial and noisy 3D input data. It is a generative model that produces a continuous signed distance field given, for example, a depth map.

Chen et al. [5] proposed their implicit field decoder, called IM-NET to auto-encode 3D meshes by training it to a certain category of meshes. They also proposed a method to generate and interpolate between meshes using their Network setup. Further, the researchers proposed using 2D images as an input in order to generate and interpolate 2D shapes and reconstruct 3D shapes from a single image (Single-view 3D reconstruction).

The compressed latent vector has frequently been proposed as a tool to edit the geometry and often used in generative models such as Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs), see e.g. [4, 6, 10].

In this paper, we examine two autoencoder networks and explore their latent dimension of geometry for use in several tasks. The latent vectors are used for database queries and shape reconstruction. The look-up accuracy achieved demonstrates that the latent vector indeed is a well compressed representation of the shape. To visually explore the compressed latent shape representation further we introduce a latent editor, which enables the user to modify the latent dimension and visualize its effect on the decompressed geometry real-time, thus enabling the user to visually inspect the latent vector wrt geometry changes.

3 METHOD

We use deep neural networks (DNNs) [19] to encode and decode 3D geometry. Such networks consist of multiple layers of nodes (neurons), each having one or numerous in- and outputs. Each neuron generates an output by summing up its weighted inputs and bias and then evaluating an activation function. The activation function describes when the neuron "fires" and can be applied on a layer-per-layer basis.

We combine multiple such layers, each with its individual assigned activation functions, to build a network that can be split into two halves: The first half encodes a given input to a latent dimension. By forcing the network to trade information for a lower dimension, it must extract features representing the input well enough for the compression to be viable. The second half takes this representation and expands it into a reconstruction of the input. The error (loss) between input and output is used to adjust the weights and biases of the network such that the reconstruction quality improves with each epoch.

An implicit representation can depict arbitrary geometry at a fixed size, thus fulfilling the fixed input-output constraints of any network. In this paper, we use an implicit representation where each voxel of the input and output data contains a floating point value that indicates whether the voxel is outside ($+1.0$), inside (-1.0), or directly on the surface ($+0.0$) of the volume.

This representation may be expanded to contain the actual distance of the cell center to the closest object's surface, essentially becoming a signed distance field (SDF). In this paper we assess whether this additional information increases the accuracy of the networks prediction. For the networks, the aim was to obtain a 1D latent vector which could be further employed in a latent modelling interface.

We compare the results of two types of autoencoder networks, a densely connected NN and a large CNN:

The first network we trained and tested features various densely connected LeakyReLU and batch normalization layers and its structure is shown in Figure 1. LeakyReLU was chosen due to its fast computation time [9] and to enable negative float values to pass

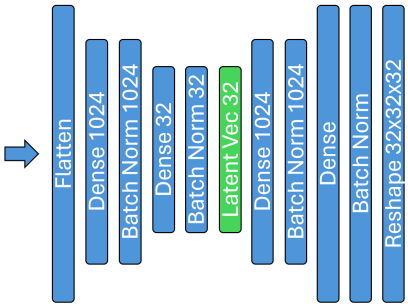

Figure 1: Structure of the first network.

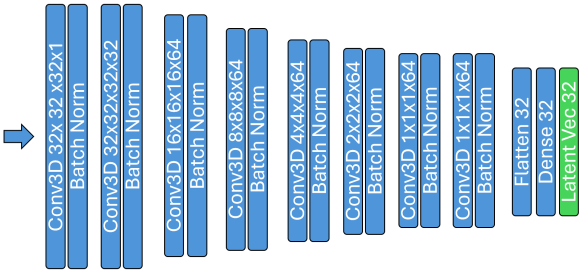

Figure 2: Structure of the second network. Decoder part is mirrowed from the encoder part and hidden due to limited space.

through the network. Batch normalization layers are employed to avoid vanishing gradients [7]. The encoder half of this network consists of four layers. First a densely connected layer reducing the size from 32^3 to 32^2, followed by a batch normalization layer, then a densely connected layer reducing the size to 32, followed by a batch normalization layer. The decoder half has the same structure as the encoder part, just mirrored.

The encoder of the second network is considerably deeper and consists of eight 3D convolution layers followed each by a batch normalization layer. The structure is visualized in Figure 2 The decoder consists of seven 3D convolution layers, again each followed by a batch normalization layer.

In our work, the input volumes are compressed into a 1D latent dimension vector containing 32 values. This value was found to be the lowest, where meaningful results could be achieved.

The network's input and output have the same dimensionality, size, and data type. We compute the reconstruction quality of the autoencoder network using a loss function based on the mean squared error between the implicit shape of the network's input and output.

Both networks are trained for 250 epochs using an 80/20 split for training and test data and use the Adam [11] as their optimizer. The networks are trained and tested with cuDNN-enabled Python using TensorFlow and the Keras API. cppflow was utilized to port the trained networks to C++ to visualize the output.

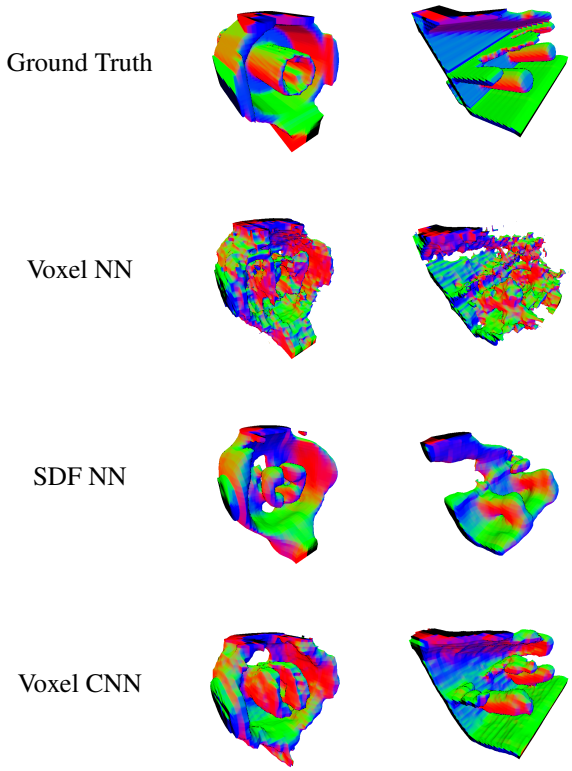

Figure 3: Examples of 3D shape reconstruction using two different shape representations (voxel or SDF) and two different NNs (a dense NN or a CNN).

To train and test the networks we gathered data from an online library of Lego meshes called LDraw [12]. The meshes of the Lego pieces are converted the CAD representation of the Lego pieces into their voxel and SDF representation. To gather sufficient data takes time, we therefore randomly rotate (max. $\pm 180°$), translate (max. 10%), and scale (max. 10%) the converted shapes to expand the data set from 40 to 4000 shapes, each consisting of 32^3 cells. Finally, the dataset is normalized, such that all values lie in the interval $[-1, 1]$.

4 RESULTS

We explored the results of the two autoencoder networks, by comparing the quality of their 3D reconstruction. We also test the applicability of the latent code in various tasks, like 3D object retrieval (3DOR), real-time latent editing, and morphing shapes.

4.1 3D Reconstruction

The decoder half of the network and the latent dimension vectors are loaded into the visualization framework, which calculates and displays the resulting volumes.

The reconstruction results in Figure 3 depict the reference and reconstructed volumes. The colors represent

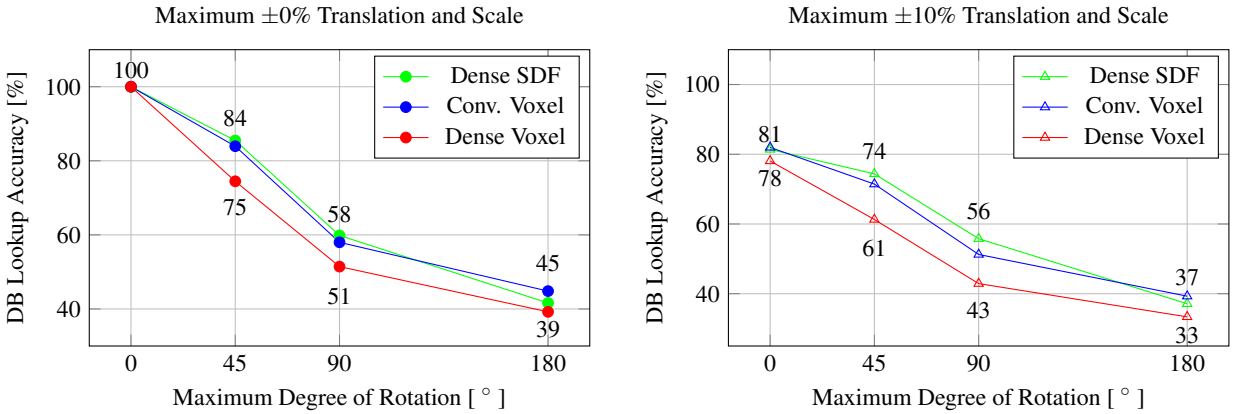

Figure 4: Database lookup accuracy of the dense and convolutional network with voxel and SDF input. The lookup accuracy decreases as the volumes are scaled, rotated, and translated.

the normal vectors of the object. The mean squared error of both depicted samples is below 0.1 for the SDF input. Since we can set negative values to -1.0 and positive values above zero to $+1.0$ as a post-processing step of the voxel output, we can simplify our measure of reconstruction error by simply counting the cells which have an incorrect sign. The percentage of cells with an incorrect sign of the two depicted samples equals 17% (left) and 14% (right) for the dense network with voxel input and 15% and 12% for SDF input.

In general the convolutional network produces smoother surfaces with less noise. It also achieves higher reconstruction accuracy. Using voxels the CNN achieves an incorrect cell sign percentage of 14% and 8% for both samples, respectively. The improvement is even more significant when using SDF as shape representation.

4.2 Latent based Object Retrieval

We prepared the data base by compressing all shapes to their 1D latent vector. After also deriving the latent shape for the query shape, the corresponding shape was retrieved from a database of shapes by finding the closest match between the query latent vector and those in the data base. The shape was retrieved with high accuracy from the shape data base.

The database lookup accuracy is given by the percent of correctly identified input samples. The database lookup accuracy depicted in Figure 4 shows that the retrieval accuracy is reasonably high for quiery shapes which are rotated up to 45° rotation from the target shape but drops significantly afterward.

The dense autoencoder network, combined with SDF input data, yields the best results, with the convolutional network with voxel input data performing similarly. The dense network with voxel data performs significantly worse. Increasing the maximum possible translation and scale to ±10% reduces the accuracy by twenty percent on average when no rotation is applied and reduces with increasing rotation.

4.3 Real-time Latent Editor

To examine the latent dimension further we implemented an interactive latent modelling editor which enables direct modification of the latent vector by the user. The user can alter the latent dimension of an input geometry and see the affected shape changes in the decoded result in real-time. The user can choose from a collection of 3D volumes, choose from two input representations, analyze the reconstruction error, rotate and scale the 3D volumes, and alter or set individual latent dimension values.

Some interesting results obtained by modifying the latent dimension values can be seen in Figure 5. In this example, the latent dimension vector of a 2x1 Lego piece represented as voxels is edited and decoded by the trained CNN decoder network which was loaded into the visualization framework. The editor is flexible such that it allows to upload any trained en- or decoder in order to experiment with the latent shape code.

Using more expressive latent vectors with this method, shapes that do not exist in the database could be created. This has a lot of potential in a creative workflow and designers could benefit from this ability. Another potential use case could be creating a design process based on this. A basic model can be encoded in a meaningful latent vector and then edited to add more features or quickly change the look of the created object.

4.4 Latent Editor: Morphing Shapes

We extended the Latent Editor to enable the user to select a source and target volume. The morphing tool of the editor then defines new shapes in between both by slowly linearly interpolating between the latent dimension vector of the source volume and the latent dimension vector of the target volume.

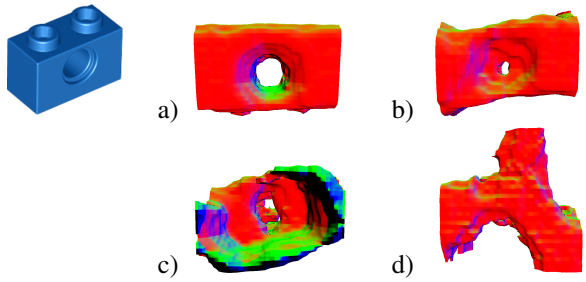

Figure 5: Examples of increasingly altering the latent dimension values of a Lego piece sample. Left: original Lego piece [2].

The result of this operation is the source volume slowly morphing into the shape of the target volume, as visible in Figure 6. This may be interesting in an entertainment context. But geometry which emerges during the morphing may also be interesting in a process, for example it could be used as inspiration for new shape designs.

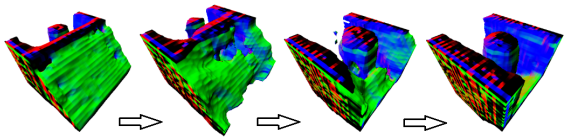

Figure 6: Morphing a Lego piece into another one via linear interpolation of the latent dimension values.

5 SUMMARY AND FUTURE WORK

In this paper we evaluate different shape representations and networks and their corresponding latent vectors in various scenarios. We evaluated the networks based on the quality of their 3D reconstruction and how accurate their corresponding latent vector succeed to retrieve 3D objects from a data base. We also evaluated two implicit shape representations, namely a simple voxel occupancy grid and a SDF. SDFs have performed significantly better than simple voxels when reconstructing shapes, as more information is contained in the input from which the network can learn. We explored the latent vector for two different types of autoencoder networks, namely a dense NN and a CNN.

While the latent vector derived using a CNN leads to a more accurate and smoother reconstruction of the original shape, a simple NN set may already suffice in using the latent code to retrieve a corresponding shape from a shape data base.

The shown results indicate that executing a database lookup favors a specialized network trained to do classification instead of using an autoencoder, as it is, in essence, a classification task. Future research will include developing and testing two separate networks, one for reconstruction and one for classification, and comparing the obtained results to the results described in this paper.

We also present an interactive tool that enables the user to interactively alter the latent vector and immediately explore the changes in the reconstruction. This tool can be used to explore the geometric meaning captured in a latent vector. It also enables the user to look into a potential meaning for the latent vector values and analyze each component on its own in real time. With interpolation between the database entries of different shapes a user can examine what different shapes lie between values and travel along the latent space.

Future research questions will focus on extending the capability of the editor and analyzing the latent dimensions with the proposed tool.

REFERENCES

[1] Bank, D., Koenigstein, N., Giryes, R., 2020. Autoencoders. URL: https://arxiv.org/abs/2003.05991, doi:10.48550/ARXIV.2003.05991.

[2] BrickLink, accessed 2024-01-02. https://www.bricklink.com.

[3] Brock, A., Lim, T., Ritchie, J.M., Weston, N., 2016. Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236 .

[4] Chen, M., Xie, J., Laina, I., 2023. Shap-editor: Instruction-guided latent 3D editing in seconds. arXiv preprint arXiv:2312.10825 .

[5] Chen, Z., Zhang, H., 2019. Learning implicit fields for generative shape modeling, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 5939–5948.

[6] Elsner, T., Ibing, M., Czech, V., Nehring-Wirxel, J., Kobbelt, L., 2022. Intuitive shape editing in latent space. arXiv preprint arXiv:2111.12488 .

[7] Foster, D., 2019. Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play. First ed., O'Reilly Media, Inc.

[8] Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A., 2016. Learning a predictable and generative vector representation for objects, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14, Springer. pp. 484–499.

[9] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

[10] Hu, V.T., Zhang, D.W., Mettes, P., Tang, M., Zhao, D., Snoek, C.G.M., 2023. Latent space editing in transformer-based flow matching. arXiv preprint arXiv:2312.10825 .

[11] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980, doi:10.48550/ARXIV.1412.6980.

[12] LDraw, accessed 2024-01-02. https://www.ldraw.org/.

[13] Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE. pp. 922–928.

[14] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S., 2019. Deepsdf: Learning continuous signed distance functions for shape representation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[15] Wang, P., Liu, Y., Guo, Y., Sun, C., O-cnn, X.T., . Octree-based convolutional neural networks for 3d shape analysis., 2017, 36. DOI: https://doi.org/10.1145/3072959.3073608 , 1–11.

[16] Wang, Y., Xie, Z., Xu, K., Dou, Y., Lei, Y., 2016. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. Neurocomputing 174, 988–998. URL: https://www.sciencedirect.com/science/article/pii/S0925231215014940, doi:https://doi.org/10.1016/j.neucom.2015.10.035.

[17] Wohlhart, P., Lepetit, V., 2015. Learning descriptors for object recognition and 3d pose estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[18] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920.

[19] Yun, K., Huyen, A., Lu, T., 2018. Deep neural networks for pattern recognition. URL: https://arxiv.org/abs/1809.09645, doi:10.48550/ARXIV.1809.09645.