Generative Enhancement of 3D Image Classifiers

Michal Varga (✉ michal.varga@tuke.sk)

Technická univerzita v Kosiciach <https://orcid.org/0000-0002-9952-3914>

Ján Jadlovský

Technická univerzita v Kosiciach

Slávka Jadlovská

Technická univerzita v Kosiciach

Research

Keywords: Generative modeling; Image classification; Convolutional neural network; Deep learning; 3D imaging

Posted Date: March 12th, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-17005/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at Applied Sciences on October 22nd, 2020.
See the published version at <https://doi.org/10.3390/app10217433>.

RESEARCH

Generative Enhancement of 3D Image Classifiers

Michal Varga^{*}, Ján Jadlovský and Slávka Jadlovská**Abstract**

In this paper, we propose a methodology for generative enhancement of existing 3D image classifiers. This methodology is based on combining the strengths of both non-generative classifiers and generative modeling. Its purpose is to streamline the creation of new classifiers by embedding existing compatible classifiers in a generative network architecture. The demonstration of this process and evaluation of its effects is performed using a 3D convolutional classifier and its generative equivalent - a conditional generative adversarial network classifier. The results show that the generative model achieves greater classification performance, gaining a relative classification accuracy improvement of 7.43%. Improvement of accuracy is also present when compared to a plain convolutional classifier trained on a dataset augmented with examples produced by a trained generator. This suggests there is a desirable knowledge sharing within the hybrid discriminator-classifier network.

Keywords: Generative modeling; Image classification; Convolutional neural network; Deep learning; 3D imaging

*Correspondence:

michal.varga@tuke.sk

Dept. of Cybernetics and Artificial
Intelligence, FEI TUKE, Košice,

Slovak Republic

Full list of author information is
available at the end of the article

1 Introduction

Generative modeling has been successfully applied to solve a wide range of tasks including realistic data generation (numeric, text, image, audio, video, etc.), extrapolation, image super-resolution, inpainting, and many others. Data classification is also one of those, even though it is usually not the main goal of designing a generative network.

Deep generative models have been proven to learn a salient representation of the data they had been trained on, which is ideal for the classification task. This latent representation is perceptually significant on the generated output which has been shown by various latent vector arithmetic experiments. [1, 2]

Lately, one of the most popular varieties of generative models is the Generative Adversarial Network (GAN) [3]. From the classification perspective, a GAN network's ability to generate highly realistic data indicates a potential for high performance in classification tasks, especially where large training datasets are not available, which is not uncommon in the domain of 3D imaging. While it was not originally designed

¹for classification, simple modifications exist to enable this functionality. One of those¹
²is to use the discriminator itself as a classifier. This can be achieved by extending²
³the existing discriminator output with additional units, one per class, and training³
⁴the model (semi)supervisedly. [4] While mostly used with 2D image data, a simple⁴
⁵modification of the network (using 3D (de)convolution instead of 2D) enables the⁵
⁶usage of 3D data in the form of voxel grids or point clouds. ⁶

⁷ State of the art generative and non-generative classifiers compete very closely in ⁷
⁸terms of classification accuracy [5]. While models utilizing the two approaches differ⁸
⁹significantly in structure, we see the potential for their combination by enhancing⁹
¹⁰the non-generative classifiers with elements of generative modeling. Aside from¹⁰
¹¹improving the classification accuracy, this could likely help with the 3D domain-¹¹
¹²specific problems, such as small datasets, too generic and synthetic training examples,¹²
¹³etc. ¹³

¹⁴ This idea is lightly supported in [6], where authors observe an occurrence of bene-¹⁴
¹⁵ficial interaction within the model when combining supervised and unsupervised loss¹⁵
¹⁶when training a feature matching GAN classifier. This hints towards a potential for¹⁶
¹⁷such interactions within a generative classifier which allows for further improvement¹⁷
¹⁸of non-generative state-of-the-art models by embedding them within a generative¹⁸
¹⁹network architecture. ¹⁹

²⁰ For these reasons, we propose a general methodology of generative enhancement,²⁰
²¹which aims to enable any compatible classifier to utilize a generative modeling²¹
²²approach to improve its classification accuracy. We evaluate the methodology by²²
²³applying it to a simple 3D voxel convolutional classifier, although it is not theoretically²³
²⁴limited to this type of data. We investigate the reproducibility and measurability²⁴
²⁵of the benefits of this transformation by comparing the classification performance²⁵
²⁶of the resulting conditional GAN network to the original 3D Convolutional Neural²⁶
²⁷Network (CNN) classifier, identical to the GAN discriminator. Experiments are also²⁷
²⁸performed to investigate possible explanations such as the introduction of noise or²⁸
²⁹data augmentation by the generator. ²⁹

³¹2 Related Work ³¹

³² As the training of a GAN network is very difficult in general, a lot of research focuses ³²
³³on the training process itself and examines various tweaks and hacks that improve ³³

¹the convergence of the model during the training. Many of these were utilized in our¹

²research. [6, 7]²

³ Generative networks have been used very successfully as a tool for data augmenta-³
⁴tion, improving existing (vanilla) classifiers' performance. The benefit of GAN-based⁴
⁵data augmentation was shown to even exceed that of standard augmentation meth-⁵
⁶ods. This augmentation has proven helpful especially with small and/or unbalanced⁶
⁷datasets. [8, 9, 10]⁷

⁸ Conditional generation combined with a classifying discriminator (auxiliary classi-⁸
⁹fier) has been shown to improve the quality of generated samples when trained on a⁹
¹⁰labeled dataset using a model called Auxiliary Classifier GAN (ACGAN) [11]. This¹⁰
¹¹network includes separate discrimination and classification terms in the objective¹¹
¹²function. Another approach, also used in our models, is to combine the terms in¹²
¹³a way, where a sample cannot be classified as both fake and of a particular class.¹³
¹⁴This is achieved by creating a 'fake' class and discarding the fake-real discrimination¹⁴
¹⁵function. Such an approach is also used in BAGAN [12], a generative network¹⁵
¹⁶designed to restore the balance of an unbalanced dataset by generating samples of¹⁶
¹⁷the minority classes. Additionally, this network tackles the mode collapse issue by¹⁷
¹⁸initializing the parameters using an autoencoder.¹⁸

¹⁹ Data classification is also a domain, where generative models have found their¹⁹
²⁰application. Many deep generative models, using both supervised and semi-supervised²⁰
²¹training, have been used for various learning and classification tasks. Unsupervised²¹
²²training of generative models is also viable and various techniques exist to perform²²
²³this task, such as Output Distribution Matching cost function [13] or CatGAN [14]²³

²⁴ A combination of GAN architecture and variational autoencoders (VAE) have²⁴
²⁵found their use in classification and retrieval of 3D models, also being able to²⁵
²⁶convert 2D images to 3D models. [2] Similarly, these models are also useful for tasks²⁶
²⁷concerning other types of data, such as spoken language identification systems. [15]²⁷

²⁸ In our research, we focus mostly on the classification of 3D images. ModelNet²⁸
²⁹(a labeled dataset of 3D objects) website [5] publishes a benchmark leaderboard²⁹
³⁰comparing the results of different 3D object classifiers trained on the dataset, which³⁰
³¹shows that generative models can achieve state-of-the-art performance and sometimes³¹
³²also exceed it. VRN Ensemble [16], a generative model based on voxel variational³²
³³

¹autoencoders and convolutional architecture, achieves one of the best performances¹
²in the benchmark.²

³ The standard deconvolutional layer is not the only building block which the³
⁴generator can be constructed with. A novel factorized generative model in [17]⁴
⁵generates the 3D models by first building it with generic primitives and then refining⁵
⁶the resulting shape. The resulting representation was evaluated using a classifier⁶
⁷which achieved state-of-the-art results on ModelNet10 among the unsupervised⁷
⁸models.⁸

⁹ **3 Dataset**¹⁰

¹¹The models presented in this paper have all been trained on and evaluated on a 3D¹¹
¹²model dataset called ModelNet [5]. Two subsets of the dataset exist, ModelNet10,¹²
¹³and ModelNet40, containing 10 and 40 classes of objects respectively. The orientation¹³
¹⁴of the examples is normalized so that every model's vector of gravity is aligned to¹⁴
¹⁵the common vertical axis. Therefore, no manual adjustments of the mesh objects¹⁵
¹⁶were necessary.¹⁶

¹⁷ 3.1 Preprocessing and Transformation¹⁷

¹⁸The ModelNet dataset contains examples represented as 3D CAD meshes. To be¹⁸
¹⁹usable as an input for 3D convolution-based networks it has to be converted to a¹⁹
²⁰voxel grid format. For this purpose our data transformation pipeline has been used²⁰
²¹[18].²¹

²² This pipeline is utilizing a voxelization tool called `binvox` by Patrick Min [19, 20].²²
²³The pipeline then fills the hollow voxel model and scales it down to reduce the²³
²⁴amount of data due to performance considerations.²⁴

²⁵ Filling the model is done by labeling the connected regions in the voxel grid and²⁵
²⁶joining everything except the largest connected region, which is usually the free²⁶
²⁷space around the model. This is not fault-proof, as some models are either not²⁷
²⁸watertight or not aligned perfectly to one face of the volume. While faulty results²⁸
²⁹are rare, this is one of the possible areas of improvement in the processing pipeline.²⁹

³⁰ The scaling is performed by a combination of max pooling and interpolated³⁰
³¹subsampling. This process removes small artifacts and holes introduced by the³¹
³²voxelization step while preserving most of the information by using floating-point³²
³³values in the subsampling pass. The resulting size of one sample is $32 \times 32 \times 32$ ³³

¹floating-point value voxels. The value of a single voxel after pre-processing is the¹

²approximation of the fraction of the voxel volume occupied by the 3D object. The²

³voxel values, originally from $[0, 1]$ range, are finally normalized in $[-1, 1]$ range.³

⁴There is no color information available in the dataset. 4

⁵ For debugging and visualization of the dataset and generated examples, the voxel⁵

⁶grids are drawn using our custom rendering pipeline [21]. The main parts of the⁶

⁷pipeline are surface normal estimation using Sobel operators and texture mapping⁷

⁸of an equirectangular lightmap to the voxels. The resulting RGBA-valued voxel grid⁸

⁹is rendered using the PyQtGraph library [22]. 9

10 10

¹¹**4 Generative enhancement methodology** 11

¹²We propose a methodology for constructing generatively enhanced classifiers by em-¹²

¹³bedding existing well-performing neural network classifiers in a compatible generative¹³

¹⁴network architecture. By that, the classification performance of the original classifier¹⁴

¹⁵is improved along with its tolerance for small training datasets. The methodology is¹⁵

¹⁶designed to formalize this transformation process and its creation is in line with one¹⁶

¹⁷of the goals established in [23]. 17

¹⁸ The process of modification consists of several steps. First, a conditional generator¹⁸

¹⁹network is designed with output identical to the classified data type (same number¹⁹

²⁰of dimensions, size, number of channels, etc.) and from the desired class. Then, the²⁰

²¹classifier is extended to perform the discrimination task (for example, by adding a²¹

²²'fake' class to the output value set). Finally, the training algorithm is modified to²²

²³include adversarial optimization of the generator. 23

²⁴ Therefore, this can be conceptualized as a neural network transformation, which²⁴

²⁵turns a traditional convolutional neural network classifier into a conditional GAN²⁵

²⁶classifier. By applying this transformation to different classifiers, entirely new archi-²⁶

²⁷tectures can be produced with potentially improved properties. 27

²⁸ The main constraint for this methodology is the classifier must be fully differentiable²⁸

²⁹from one end to the other so that the gradients can be calculated for all layers,²⁹

³⁰including the input layer. This is required to optimize the generator, which is³⁰

³¹attached to the classifier via the input layer itself. 31

32 32

³³**5 Models and Methods** 33

Figure 1 High-level architecture of the tree models designed for the proposed experiment. The first model (a) is a plain 3D CNN trained directly on ModelNet10 training examples. The other CNN (b) is trained on an augmented dataset created by combining ModelNet10 examples and the conditional generator (G) output. The generator output is labeled according to its input condition vector. The last model (c) is a 3D conditional GAN network where the discriminator (D) is used as the classifier during the testing phase.

To verify the viability of the proposed methodology, we evaluate its simplest application case – the transformation of a convolutional classifier into a conditional GAN network. For this purpose, three separate models have been designed. The first is a control model represented by a plain 3D convolutional neural network (plain CNN) trained only on the training dataset examples. The second model is another classifier, which shares the same structure as the control, however, it is also trained on the output produced by the generator from a GAN network (augmented CNN). The third model is the aforementioned 3D conditional GAN network. A high-level diagram of the designed models is shown in Figure 1.

5.1 3D Convolutional Neural Network Classifiers

The first two models are identical deep 3D convolutional networks consisting of 3 convolutional layers. Stride has been used instead of pooling layers for downsampling. Batch normalization and Leaky ReLU activation function have been used following each convolutional layer. The output is produced by a single fully-connected layer with a unit count of C , equal to the number of classes. A softmax function is used to convert the output logits to class probabilities.

The loss function used to train these classifiers is a categorical cross-entropy shown in (1).

$$L_C = - \sum_{i=1}^M \sum_{j=1}^C \mathbf{y}_j^{(i)} \log(\hat{\mathbf{y}}_j^{(i)}) \quad (1)$$

¹ In the equation, M stands for the number of examples (in a minibatch or total), C ¹
²is the number of classes in the dataset, $\mathbf{y}$ is a vector of one-hot encoded training data²
³labels and $\hat{\mathbf{y}}$ is the classifier prediction of the class vector. The $\mathbf{y}_j^{(i)}$ notation refers³
⁴to the j^{th} element of the vector belonging to the i^{th} sample within the minibatch.⁴
⁵The optimizer used to train these classifiers was RMSprop.⁵

6

6

⁷5.2 3D Conditional Generative Adversarial Network⁷

⁸The final model is a 3D conditional generative adversarial network (3D CGAN),⁸
⁹consisting of a generator-discriminator pair connected in sequence and trained⁹
¹⁰in opposition to each other. The role of the generator is to transform a latent¹⁰
¹¹representation (vector) of an object to its visual representation (voxel grid), while¹¹
¹²the discriminator tries to determine whether the sample is real (from the training¹²
¹³dataset) or fake (produced by the generator). The goal of the adversarial training is¹³
¹⁴to achieve the equilibrium, where the generator produces samples that are realistic¹⁴
¹⁵enough to confuse the discriminator [3]. A conditional GAN introduces a condition¹⁵
¹⁶for the generator, specifying the desired output by providing a one-hot encoded class¹⁶
¹⁷vector or another form of specification (eg. keywords or description text) [24]. The¹⁷
¹⁸starting point for this architecture has been the deep convolutional 3D GAN model¹⁸
¹⁹published in [2].¹⁹

²⁰ The main difference from the standard GAN architecture in our model is the²⁰
²¹discriminator, which is mostly identical to the plain and augmented CNN models²¹
²²mentioned above. The only difference between the discriminator and the two previ-²²
²³ously described convolutional classifiers is in the output layer, which, in addition²³
²⁴to the class label, contains one extra output indicating whether the input is con-²⁴
²⁵sidered fake or not. Essentially, it distinguishes one additional "fake" class. The²⁵
²⁶layers of discriminator use the Leaky ReLU activation function, along with batch²⁶
²⁷normalization.²⁷

²⁸ The generative part of the CGAN network (generator), shown in Figure 2, consists²⁸
²⁹of 3 deconvolutional (or transposed convolutional) layers with a descending amount²⁹
³⁰of volumetric filters. The upsampling of the activation volumes is achieved by kernel³⁰
³¹stride, which is equal to 2 in every layer. ReLU activation function is used after³¹
³²each layer, followed by a batch normalization function. The output layer uses $\tanh$ ³²
³³activation function instead to fit the output into the dataset value range of $[-1, 1]$.³³

The generator is conditioned by appending the desired one-hot encoded class vector to the latent feature vector, containing 200 feature values sampled from a Gaussian distribution $\mathcal{N}(0, 1)$ forming an input vector z . The input is first fed through a fully connected layer expanding it to the shape required by the first deconvolutional layer. The dense layer is also followed by ReLU and batch normalization functions.

The loss function chosen to train the discriminator, shown in (2), is a standard categorical cross-entropy.

$$L_D = - \sum_{i=1}^M \sum_{j=1}^{C+1} \mathbf{y}_j^{*(i)} \log(\hat{\mathbf{y}}_j^{*(i)}) \quad (2)$$

To include both fake-real discrimination error and classification error, an extended target vector $\mathbf{y}^*$ is constructed, as shown in (3).

$$\mathbf{y}^{*(i)} = \begin{cases} [0, \mathbf{y}^{(i)}] & \text{real} \\ [1, 0, 0, \dots, 0] & \text{fake} \end{cases} \quad (3)$$

For real samples, a single zero is prepended to the class label from the dataset. In the case of generated samples, the target is a $C + 1$ long vector, with the first element (the "fake" class) set to one and the rest to zero.

For the generator, the loss function is chosen to minimize the "fake" output of the discriminator as well as achieve the class output according to the conditioning vector

provided to generate the sample. The generator loss function formula is essentially¹
 identical to the discriminator loss with a real example. The only difference is the²
 true class vector, which is created randomly, instead of sampling it from the dataset,³
 and used both as generator input (concatenated with noise vector) and as the target⁴
 vector compared with the discriminator output.⁵

An alternative loss function we experimented with is a modified Wasserstein loss⁶
 function [25] shown in (4) and (5).⁷

$$L_D^w = \frac{1}{M} \sum_{i=1}^M D(\mathbf{x}^{(i)}) - \frac{1}{M} \sum_{i=1}^M D(G(\mathbf{z}^{(i)})) - \gamma \sum_{i=1}^M \sum_{j=1}^C \mathbf{y}_j^{(i)} \log(\hat{\mathbf{y}}_j^{(i)}) \quad (4)$$

$$L_G^w = \frac{1}{M} \sum_{i=1}^M D(G(\mathbf{z}^{(i)})) - \gamma \sum_{i=1}^M \sum_{j=1}^C \mathbf{y}_j^{(i)} \log(\hat{\mathbf{y}}_j^{(i)}) \quad (5)$$

In these formulas, M stands for the number of samples (in a minibatch or total),
 C is the number of classes in the dataset, $\mathbf{x}$ is a vector of training data samples,
 $\mathbf{y}$ is a vector of one-hot encoded training data labels, $\hat{\mathbf{y}}$ is the classifier prediction
 and γ is a hyperparameter that is used to tune the weight of the classification loss
 term during training. D and G are the discriminator and the generator functions
 respectively. The output of the discriminator function referred to as D is the logit
 value of the "fake" class output of the discriminator. This formula further decouples
 the discrimination and classification objectives and allows for more competition
 which can be tweaked by the γ parameter. ADAM optimizer [26] has been used to
 train both generator and discriminator.

5.3 Training

The training is performed for all three models concurrently. In each training epoch,
 the training data from the ModelNet10 dataset containing all 10 classes is shuffled
 and split into minibatches of 16 samples. Each minibatch is then fed into the plain
 and augmented CNN classifiers, as well as the discriminator. The discriminator is
 trained in two stages, first with the training data and then with generated 'fake'
 data. This fake-real training separation of gradient calculation has been shown to

¹improve training stability [7]. Experiments using both approaches (separating or¹
²mixing real and fake data) have been conducted, however, no significant difference²
³has been observed. One-sided label smoothing is also utilized to improve overfitting³
⁴resistance. 4

⁶6 Results and Discussion 6

⁷The main evaluation criterion of the designed models is the classification accuracy 7
⁸on the ModelNet10 dataset. We also examine the generative performance of the 8
⁹GAN model to verify whether the performance benefit is not caused by some other 9
¹⁰effect unrelated to the learned latent representation. 10

¹²6.1 Classification 12

¹³All three models have been evaluated using the test data from the ModelNet10 dataset. 13
¹⁴Classification accuracy, as well as mean average precision, have been calculated. 14
¹⁵The results are shown in Table 1. Several other modern models utilizing generative 15
¹⁶modeling with published ModelNet10 results are also included in the table. The 16
¹⁷results show that generative enhancement of a very simple convolutional classifier 17
¹⁸(plain CNN) brings its average precision close to the state-of-the-art classifiers. 18
¹⁹19

²⁰**Table 1** Comparison of classification accuracy and mean average precision score of generative 20
²¹modelling-based classifiers evaluated on the ModelNet10 dataset. [5] 21

	Accuracy [%]	Mean AP [%]
Primitive-GAN [17]	92.20	
VIPGAN [27]	94.05	90.69
Achlioptas et al. [28]	95.40	
VRN Ensemble [16]	97.14	
3D-GAN [2]	91.00	
Plain CNN (ours)	83.04	83.75
Augmented CNN (ours)	84.58	86.31
GAN Classifier (ours)	89.21	89.18

²⁸Another comparison of classification performance of our three models is shown in 28
²⁹the ROC plot in Figure 3 and some additional insight into classification performance 29
³⁰is provided by the confusion matrix in Figure 4. 30

³¹While the accuracy of our models does not exceed the state of the art, the 31
³²results show that training an identical classifier inside a GAN model instead of 32
³³conventionally, using just the training dataset, can produce a significant classification 33

Figure 3 ROC curves for each of the three designed models. These curves are calculated by micro-averaging the curves for each class.

accuracy improvement. In our case, the relative improvement from plain CNN to the GAN-trained classifier was 7.43%.

Comparing the augmented CNN to the GAN classifier, the difference is smaller (5.47% relative improvement), however still significant. This suggests, that the improvement is caused not only by introducing noise to the dataset. It also means the generator is not just a data augmentation mechanism, because in that case, the augmented CNN and GAN classifier should achieve similar performance. The results suggest there exists some form of interaction between the generative and classifying part of a conditional GAN network that improves the classification accuracy of the model.

The interactions likely improve the generative performance as well, as other researchers recommend using class labels, if available when training a GAN network in general. [7]

6.2 Object Generation

We also examined the output of the generator to verify the model is working as expected. The quality of the generated examples was not our primary goal, however,

Figure 4 Confusion matrix of the GAN classifier evaluated on the ModelNet10 testing set. The values are normalized in $[0,1]$ range per each true label. It shows the most misclassified classes are desks, tables, and nightstands. This can most likely be attributed to low resolution and non-identical scale of the voxel models caused by the voxelization tool scaling the mesh models to fit the voxel grid volume.

it can be an indicator of errors within the architecture design or implementation. Several generated examples are shown in Figure 5. Mode collapse, a common pitfall for generative models, has been a major problem during our experimentation. Several solutions have been tested, such as using the Wasserstein loss function, however, the most significant improvement of both mode collapse and general convergence of the network was caused by initializing the ADAM optimizer with a very low learning rate $\alpha \approx 10^{-4}$. As can be seen, there is a visible intra-class variance in the output of the generator. Some noise and artifacts are also produced, however, apparently, it does not cause any adverse effects during classification aside from the lower visual quality of the output.

6.3 Performance and Timing

We also investigated the computational performance of the proposed models. As training has been performed concurrently, we can compare the accuracy evolution

over training steps or epochs instead of time. This reveals the data efficiency of the models instead of computing efficiency. On our hardware (Intel i7 6700K CPU, 16GB RAM, Nvidia GeForce GTX 1080), the plain CNN classifier achieved its peak performance around step 15k. The augmented CNN classifier took 40k steps to reach its peak. The GAN classifier reached its maximum around step 88k. This would not translate proportionately to training time if the models were trained separately, however it shows the complexity gained by generative enhancement incurs a significant hit to the training speed.

Inference speed has been measured separately and the results confirm the expected outcome: the generative enhancement does not significantly modify the classifier and its complexity does not increase enough to measurably reduce the inferential performance. The testing set containing 908 samples has been classified in

¹830ms (plain CNN), 826ms (augmented CNN) and 786ms (GAN). This translates¹
²to 1092, 1098 and 1153 samples per second respectively.²

³

⁴**7 Conclusion and Future Work**⁴

⁵In this paper, we proposed a methodology for generative enhancement of 3D image⁵
⁶classifiers and experimentally verified its viability and evaluated its performance in⁶
⁷a basic application scenario.⁷

⁸The results of the experiment support the initial idea, that a supervisedly trained⁸
⁹conditional GAN network (i.e. generatively enhanced convolutional classifier) enables⁹
¹⁰a knowledge transfer between the generator and a classifying discriminator that¹⁰
¹¹significantly improves its classification accuracy. The mechanism of these interactions¹¹
¹²is not clear and it could be a subject of further theoretical research.¹²

¹³One of the possible explanations for the classification performance improvement¹³
¹⁴is that the generator introduces noise to the training process of the discrimina-¹⁴
¹⁵tor/classifier and therefore makes it more robust. This explanation is not completely¹⁵
¹⁶sufficient as the same effect occurs in the augmented CNN model which does not¹⁶
¹⁷indicate the same improvement. The same applies to another explanation, which¹⁷
¹⁸points to the data augmentation caused by the generator producing realistic but¹⁸
¹⁹not identical training examples and there improving the robustness of the classifier.¹⁹
²⁰This should also manifest in the augmented CNN model which, however, falls short²⁰
²¹of the CGAN classifier performance.²¹

²²The results of this study suggest a potential for the generative enhancement²²
²³methodology, enhancing other existing well-performing classifiers by embedding²³
²⁴them within a generative architecture. Our next efforts will focus on exploring other²⁴
²⁵candidates for this transformation and evaluating the resulting architectures. Addi-²⁵
²⁶tionally, we plan to perform a physical experiment to verify the real-life application²⁶
²⁷potential of the newly created models. For this purpose, a custom rotary table 3D²⁷
²⁸scanner is being designed to acquire scans of real-life objects which will be used for²⁸
²⁹the model evaluation. These efforts, including the existing hardware and software²⁹
³⁰solutions, are part of the Technicom project [29].³⁰

³¹

³²**Competing interests**³²

³³The authors declare that they have no competing interests.³³

1 Author's contributions

2 J.J. is responsible for the general research supervision, S.J. for research project management and funding acquisition
 3 J.J. and S.J. guided the overall direction of the article and methodology, provided technical guidance and simulation
 4 equipment. M.V. was a major contributor in writing the manuscript. All authors read and approved the final
 5 manuscript.

5 Acknowledgements

6 Authors are thankful for the support from project KEGA "Implementation of research results in the area of modeling
 7 and simulation of cyber-physical systems into the teaching process - development of modern university textbooks" –
 8 7072TUKE-4/2018.

8 Funding

9 This work has been funded by grant KEGA "Implementation of research results in the area of modeling and
 10 simulation of cyber-physical systems into the teaching process - development of modern university textbooks" –
 11 072TUKE-4/2018.

11 Availability of data and materials

12 The datasets analyzed during the current study are available on the public ModelNet website,
 13 <https://modelnet.cs.princeton.edu/>.

14 References

- 15 1. Tasse, F.P., Dodgson, N.: Shape2Vec: Semantic-based descriptors for 3D shapes, sketches and images. ACM
 16 Transactions on Graphics (TOG) **35**(6), 208–120812 (2016). doi:[10.1145/2980179.2980253](https://doi.org/10.1145/2980179.2980253)
- 17 2. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes
 18 via 3d generative-adversarial modeling. In: Advances in Neural Information Processing Systems, pp. 82–90
 19 (2016)
- 20 3. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.:
 21 Generative Adversarial Nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q.
 22 (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc., Palais
 23 des Congrès de Montréal, Montréal, Canada (2014)
- 24 4. Odena, A.: Semi-Supervised Learning with Generative Adversarial Networks. In: Workshop on Data-Efficient
 25 Machine Learning (ICML 2016), New York City, NY, USA (2016)
- 26 5. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for
 27 volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.
 28 1912–1920 (2015)
- 29 6. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for
 30 training GANs. In: Proceedings of the 30th International Conference on Neural Information Processing Systems
 31 NIPS'16, pp. 2234–2242. Curran Associates Inc., Red Hook, NY, USA (2016). [1606.03498](https://arxiv.org/abs/1606.03498)
- 32 7. Chintala, S.: How to Train a GAN? Tips and Tricks to Make GANs Work.
 33 <https://github.com/soumith/ganhacks> (2019)
8. Antoniou, A., Storkey, A., Edwards, H.: Data Augmentation Generative Adversarial Networks. arXiv:1711.04340
 [cs, stat] (2018). [1711.04340](https://arxiv.org/abs/1711.04340)
9. Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning.
 Convolutional Neural Networks Vis. Recognit, 11 (2017)
10. Tanaka, F.H.K.d.S., Aranha, C.: Data Augmentation Using GANs. arXiv:1904.09135 [cs, stat] (2019).
[1904.09135](https://arxiv.org/abs/1904.09135)
11. Odena, A., Olah, C., Shlens, J.: Conditional Image Synthesis With Auxiliary Classifier GANs. In: Precup, D.,
 Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of
 Machine Learning Research, vol. 70, pp. 2642–2651. PMLR, International Convention Centre, Sydney, Australia
 (2017)
12. Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., Malossi, C.: BAGAN: Data Augmentation with Balancing
 GAN. arXiv:1803.09655 [cs, stat] (2018). [1803.09655](https://arxiv.org/abs/1803.09655)

13. Sutskever, I., Jozefowicz, R., Gregor, K., Rezende, D., Lillicrap, T., Vinyals, O.: Towards Principled
Unsupervised Learning. arXiv:1511.06440 [cs] (2015). [1511.06440](https://arxiv.org/abs/1511.06440)
14. Springenberg, J.T.: Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial
Networks. In: 5th International Conference on Learning Representations, Palais des Congrès Neptune, Toulon,
France (2017)
15. Shen, P., Lu, X., Li, S., Kawai, H.: Conditional Generative Adversarial Nets Classifier for Spoken Language
Identification. In: INTERSPEECH (2017). doi:[10.21437/interspeech.2017-553](https://doi.org/10.21437/interspeech.2017-553)
16. Brock, A., Lim, T., Ritchie, J.M., Weston, N.J.: Generative and discriminative voxel modeling with
convolutional neural networks. In: Neural Information Processing Conference, Barcelona, Spain, pp. 1–9 (2016)
17. Khan, S.H., Guo, Y., Hayat, M., Barnes, N.: Unsupervised Primitive Discovery for Improved 3D Generative
Modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.
9739–9748 (2019)
18. Varga, M., Jadlovský, J.: Evaluation of Depth Modality in Convolutional Neural Network Classification of
RGB-D Images. Acta Electrotechnica et Informatica **18**, 26–31 (2018). doi:[10.15546/aei-2018-0029](https://doi.org/10.15546/aei-2018-0029)
19. Min, P.: Bivox. <http://www.patrickmin.com/bivox> (2004)
20. Nooruddin, F.S., Turk, G.: Simplification and Repair of Polygonal Models Using Volumetric Techniques. IEEE
Transactions on Visualization and Computer Graphics **9**(2), 191–205 (2003)
21. Varga, M., Jadlovský, J.: Contribution to 3D Voxel Generative Adversarial Networks. In: SCYR 2019:
Proceedings from Conference, pp. 56–57 (2019)
22. Campagnola, L.: PyQtGraph, <http://www.pyqtgraph.org> (2012)
23. Varga, M.: Klasifikácia 3D obrazov pomocou generatívnych modelov. Writings of dissertation thesis, Technical
University of Košice (2017)
24. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. arXiv:1411.1784 [cs, stat] (2014). [1411.1784](https://arxiv.org/abs/1411.1784)
25. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the
34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp.
214–223. PMLR, International Convention Centre, Sydney, Australia (2017)
26. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: 3rd International Conference for
Learning Representations, San Diego (2015)
27. Han, Z., Shang, M., Liu, Y.-S., Zwicker, M.: View Inter-Prediction GAN: Unsupervised Representation Learning
for 3D Shapes by Learning Global Shape Memories to Support Local View Predictions. In: Proceedings of the
AAAI Conference on Artificial Intelligence, vol. 33, pp. 8376–8384 (2019)
28. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning Representations and Generative Models for 3D
Point Clouds. In: 35th International Conference on Machine Learning, pp. 40–49 (2018)
29. Jadlovský, J., Jadlovská, A., Jadlovská, S., Čerkala, J., Kopčík, M., Čabala, J., Oravec, M., Varga, M., Vošček,
D.: Research activities of the center of modern control techniques and industrial informatics. In: 2016 IEEE
14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 279–285 (2016).
doi:[10.1109/SAMI.2016.7423021](https://doi.org/10.1109/SAMI.2016.7423021)

Figures

Figure 1

High-level architecture of the tree models designed for the proposed experiment. The first model (a) is a plain 3D CNN trained directly on ModelNet10 training examples. The other CNN (b) is trained on an augmented dataset created by combining ModelNet10 examples and the conditional generator (G) output. The generator output is labeled according to its input condition vector. The last model (c) is a 3D conditional GAN network where the discriminator (D) is used as the classifier during the testing phase.

Figure 2

Architecture of the generator network. Input vector z consisting of a Gaussian noise vector and a condition (class) vector is expanded in a fully connected stage and then is deconvoluted three times with batch normalization and ReLU activation function in between.

Figure 3

ROC curves for each of the three designed models. These curves are calculated by micro-averaging the curves for each class.

Figure 4

Confusion matrix of the GAN classifier evaluated on the ModelNet10 testing set. The values are normalized in [0,1] range per each true label. It shows the most misclassified classes are desks, tables, and nightstands. This can most likely be attributed to low resolution and non-identical scale of the voxel models caused by the voxelization tool scaling the mesh models to fit the voxel grid volume.

Figure 5

Output of the conditional generator using random input noise and random condition class vector. This generator has been trained on the ModelNet10 dataset.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- `bmcartbiblio.sty`
- `manuscript.tex`
- `bmcmathphys.bst`
- `bmcart.cls`