applied sciences

Article

Generative Enhancement of 3D Image Classifiers

Michal Varga *, Ján Jadlovský and Slávka Jadlovská *

Department of Cybernetics and Artificial Intelligence, FEI TU of Košice, 04200 Košice, Slovakia; jan.jadlovsky@tuke.sk

* Correspondence: michal.varga@tuke.sk (M.V.); slavka.jadlovska@tuke.sk (S.J.)

Received: 23 September 2020; Accepted: 16 October 2020; Published: 22 October 2020

Abstract: In this paper, we propose a methodology for generative enhancement of existing 3D image classifiers. This methodology is based on combining the advantages of both non-generative classifiers and generative modeling. Its purpose is to streamline the synthesis of novel deep neural networks by embedding existing compatible classifiers into a generative network architecture. A demonstration of this process and evaluation of its effectiveness is performed using a 3D convolutional classifier and its generative equivalent—a 3D conditional generative adversarial network classifier. The results of the experiments show that the generative classifier delivers higher performance, gaining a relative classification accuracy improvement of 7.43%. An increase of accuracy is also observed when comparing it to a plain convolutional classifier that was trained on a dataset augmented with samples created by the trained generator. This suggests a desirable knowledge sharing mechanism exists within the hybrid discriminator-classifier network.

Keywords: generative modeling; image classification; convolutional neural network; deep learning; 3D imaging

1. Introduction

Generative modeling has been successfully applied in order to solve a wide range of tasks, including realistic data generation (numeric time-series [1], text [2], image [3], audio [4], video [5], etc.), extrapolation [6], image super-resolution [7], inpainting [8], 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 that 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 [9,10].

Lately, the Generative Adversarial Network (GAN) is one of the most popular varieties of generative models [11]. 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. Limited datasets are not uncommon in the domain of 3D imaging. While GAN was not originally designed for classification, simple modifications exist to enable this functionality, such as 3D-GAN [10]. One such modification is to use the discriminator itself as a classifier. This can be achieved by extending the existing discriminator output with additional units, one per class, and training the model (semi)supervisedly. [12] While mostly used with 2D image data, a straightforward adjustment of the network (using 3D (de)convolution instead of 2D) enables the use of 3D data in the form of voxel grids or point clouds.

State of the art generative [13–15] and non-generative [16–18] classifiers compete very closely in terms of 3D image classification accuracy, as can be seen in the results of benchmarks, such as ModelNet [19]. While models that utilize the two approaches differ significantly in structure, we see the potential for their combination by enhancing the non-generative classifiers with elements of generative modeling. Aside from improving the classification accuracy, this could likely help with the 3D domain-specific problems, such as small datasets, too generic and synthetic training examples, etc.

This idea is lightly supported in [20], where the authors observe an occurrence of beneficial interaction within the model when combining supervised and unsupervised loss while training a feature matching GAN classifier. This hints towards a potential for such interactions within a generative classifier, which allows further improvement of non-generative state-of-the-art models by embedding them within a generative network architecture.

For these reasons, we propose a general methodology of generative enhancement, which aims to enable any compatible classifier in order to utilize a generative modeling approach to improve its classification accuracy. Although it is not theoretically limited to a particular type of data, this methodology requires a fully differentiable trainable generator. We propose and evaluate this methodology for voxel grid 3D format by applying it to a simple 3D convolutional classifier. We investigate the reproducibility and measurability of the benefits of this transformation by comparing the classification performance of the resulting conditional GAN network to the original 3D Convolutional Neural Network (CNN) classifier, identical to the GAN discriminator. Experiments are also performed in order to investigate possible explanations, such as the introduction of noise or data augmentation by the generator.

The sections of this paper are structured in the following manner: following the introduction in Section 1, Section 2 discusses related work and publications in the field relevant to this paper. Section 3 describes the dataset that was used in the following experiments, as well as the details of its preprocessing and transformation. Section 4 presents the proposed Generative Enhancement methodology. In Section 5, three network architectures, designed to evaluate the proposed methodology, are shown along with the training process. Section 6 contains results and discussion, including classification, object generation, and computational performance evaluation. Finally, Section 7 presents a summary of the paper with ideas for future work.

2. Related Work

As the training of a GAN network is very difficult in general, a substantial amount of research focuses on the training process itself and examines various tweaks and hacks that improve the convergence of the model during training Many of these were utilized in our research [20,21].

Generative networks have been used very successfully as a tool for data augmentation, improving performance of existing (vanilla) classifiers. The benefit of GAN-based data augmentation was shown to even exceed that of standard augmentation methods. This augmentation has proven helpful, especially with small and/or unbalanced datasets. [22–24] It has been shown, that even fully synthetic labeled datasets, generated by deep neural generators [25] or simple 3D renderers [26] can be very beneficial for real-life applications.

Conditional generation combined with a classifying discriminator (auxiliary classifier) has been shown to improve the quality of generated samples when trained on a labeled dataset using a model, called Auxiliary Classifier GAN (ACGAN) [27]. This network includes separate discrimination and classification terms in the objective function. Another approach, used in Balancing GAN (BAGAN) [28] and also in our models, is to combine the terms in a way, where a sample cannot be classified as both fake and of a particular class. This is achieved by creating a ‘fake’ class and discarding the fake-real discrimination function. BAGAN is a generative network that was designed to restore the balance of an unbalanced dataset by generating samples of the minority classes. Additionally, this network tackles the mode collapse issue by initializing the parameters while using an autoencoder.

Data classification is also a domain, where generative models have found their application. Many deep generative models, using both supervised and semi-supervised training, have been used for various learning and classification tasks. The unsupervised training of generative models is also viable and various techniques exist to perform this task, such as Output Distribution Matching cost function [29] or CatGAN [30].

A combination of GAN architecture and variational autoencoders (VAE) have found their use in the classification and retrieval of 3D models, also being able to convert 2D images to 3D models [10]. Similarly, these models are also useful for tasks concerning other types of data, such as spoken language identification systems. [31]

The domain of 3D images introduces several specific use cases for generative networks. One of these is a 3D model reconstruction from single, or multiple 2D images. Multiple architectures have been proposed, being able to generate voxel [10] and point cloud [32] models of the objects presented as a single 2D [10], single 2.5D (RGB-D) image [32], or a series of 2D images [33]. Similarly, 3D images can also be used as an input of a generative model for tasks, such as shape inpaining [34]. Other applications with 3D input include object detection, semantic segmentation, and instance segmentation, usually applied to point cloud scans of the environment [35,36].

Inspired by our findings of the significance of the spatial (depth) information during real-life object classification and scene understanding [37], we decided to focus on the classification of 3D images. Generally, these classification methods can be grouped into view-based and volume, or shape-based techniques, depending on the form of data that are present at the input of the classifier. Many of the most successful classifiers [16–18,38] are based on the multi-view approach, which renders or acquires several views of the classified object and extract features from those 2D projections. Efficient aggregation of the information from multiple views is the key to high classification accuracy. This problem is explored in [17], where the authors process the views as an unordered graph by learning a semantic feature mapping and aggregating the patterns using a novel pair-wise spatial pattern correlation and a multi-attention model. Similarly, in [18], the authors learn global 3D features from multiple views by segmenting the projections into semantically meaningful parts.

Volume and shape-based techniques are also widely used, as can be seen in the ModelNet [19] benchmark leaderboard. The published results of different 3D object classifiers that were trained on the dataset also show that generative models can achieve state-of-the-art performance and sometimes also exceed it. VRN Ensemble [13], a generative model that is based on voxel variational autoencoders and convolutional architecture, achieves one of the best results in the benchmark.

The standard deconvolutional layer is not the only building block that the generator can be constructed with. A novel factorized generative model in [15] generates the 3D models by first building it with generic primitives and then refining the resulting shape. The resulting representation was evaluated while using a classifier which achieved state-of-the-art results on ModelNet10 among the unsupervised models.

A similar approach to our methodology, albeit in 2D, has been presented in [39], called VAC+GAN. Authors took inspiration from ACGAN [27] and developed a versatile classifier, that can be "attached" to any existing GAN network, forming a powerful classification model. This is similar to our reasoning about combining generative and non-generative architectures, utilizing the strengths of both. However, this approach differs from ours in the order of execution. While we propose a methodology of enhancing a non-generative classifier, [39] presents a process of transforming an existing generator into a classifier.

3. Dataset

All of the models presented in this paper have been trained and validated on the ModelNet dataset [19]. Two subsets of the dataset exist, ModelNet10 and ModelNet40, which contain 3D meshes from 10 or 40 classes. The examples in the dataset are already in canonical orientation, meaning that every model's vector of gravity is aligned to the vertical axis. Therefore, no manual adjustments of the mesh objects were necessary.

Preprocessing and Transformation

The ModelNet dataset (Supplementary Materials) contains examples represented as 3D CAD meshes. To be usable as an input for 3D convolution-based networks, it has to be converted to a voxel grid format. For this purpose, our data transformation pipeline has been used [40].

This pipeline utilizes a voxelization tool, called `binvox` by Patrick Min [41,42]. The pipeline then fills the hollow voxel model and scales it down to reduce the amount of data, due to computational performance considerations.

Filling the model is done by labeling the connected regions in the voxel grid and joining everything except the largest connected region, which is usually the free space around the model. This is not fault-proof, because some models are either not watertight or not aligned perfectly to one face of the volume. While faulty results are rare, this is one of the possible areas of improvement in the processing pipeline.

The scaling is performed by a combination of max pooling and interpolated subsampling. This process removes small artifacts and holes that were introduced by the voxelization step while preserving most of the information by using floating-point values in the subsampling pass. The resulting size of one sample is $32 \times 32 \times 32$ floating-point value voxels. The value of a single voxel after pre-processing is the approximation of the fraction of the voxel volume occupied by the 3D object. The voxel values, originally from $[0, 1]$ range, are finally normalized in $[-1, 1]$ range. There is no color information available in the dataset.

For debugging and visualization of the dataset and generated examples, the voxel grids are drawn while using our custom rendering pipeline [43]. The main parts of the pipeline are surface normal estimation using 3D Sobel operators and texture mapping of an equirectangular lightmap to the voxels. The resulting RGBA-valued voxel grid is rendered using the PyQtGraph library [44]. The volume is rendered as a stack of textured planes that represent individual volume slices.

4. Generative Enhancement Methodology

In this study, we present a methodology for constructing novel generatively enhanced classifiers by embedding existing well-performing neural network classifiers in a compatible generative network architecture. By that, the classification performance of the original classifier is improved along with its tolerance for small training datasets. The methodology is designed in order to streamline and formalize this transformation process. The process of modification consists of several steps that are shown in Figure 1.

First a conditional generator is designed, generating samples in a format that are identical to the classified data type (same number of dimensions, size, number of channels, etc.) and from the desired class. This is necessary because the output of the generator is fed into the input of the classifier during training. For example, if the training dataset contains labeled voxel volumes, a generator producing voxel volumes of the same size is required. To make the generator conditional, its input must include the class information (e.g., by concatenating the input noise vector with a one-hot encoded class vector), telling it which class the generated sample should belong to.

Subsequently, the classifier is extended in order to perform the discrimination task (e.g., by appending a 'fake' class to the set of possible classifier outputs). Lastly, adversarial optimization of the network is added to the training loop.

Therefore, this can be conceptualized as a neural network transformation, which turns a traditional convolutional neural network classifier into a conditional GAN classifier. By applying this transformation to different classifiers, entirely new architectures can be produced with potentially improved properties.

The main constraint for this methodology is the classifier must be fully differentiable from one end to the other, so that the gradients can be calculated for all layers, including the input layer. This is required to optimize the generator, which is attached to the classifier's input layer.

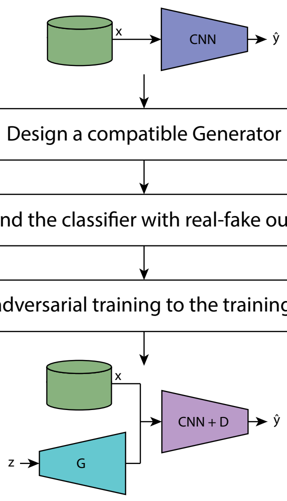
x
CNN
ŷ
Design a compatible Generator
Extend the classifier with real-fake output
Add adversarial training to the training loop
x
CNN + D
ŷ
z
G

Figure 1. Simplified flowchart showing the basic steps of the proposed Generative enhancement methodology. The starting point is a network, which classifies training and testing samples x, assigning them a predicted label $\hat{y}$. Using our methodology, the network is expanded to contain a generator (G), producing original training examples from a Gaussian noise input vector z.

5. Models and Methods

In order 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 that is represented by a plain 3D convolutional neural network (plain CNN) that is only trained 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 that is produced by the generator from a GAN network (augmented CNN). This configuration essentially utilizes the generator as a dataset augmentation mechanism, where new labeled examples with predetermined classes are generated and added to the training set used to train the CNN classifier. The third model is the aforementioned 3D conditional GAN network. Figure 2 shows a high-level diagram of the designed models.

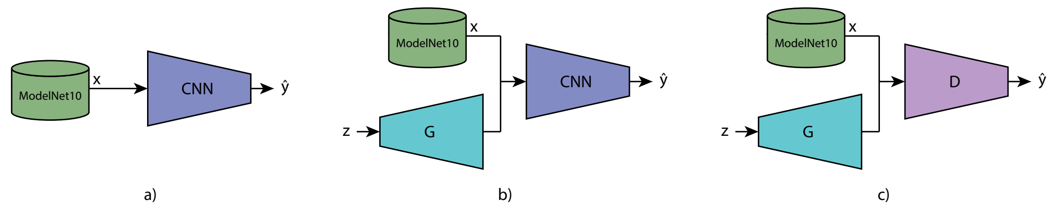

Figure 2. High-level architecture of the tree models designed for the proposed experiment. The first model (**a**) is a plain 3D convolutional (CNN) classifier trained directly on ModelNet10. The other CNN (**b**) is trained on an augmented dataset created by combining ModelNet10 examples and the conditional generator (G) output. The generator output is labeled according to its input condition vector. The last model (**c**) is a three-dimensional (3D) conditional Generative Adversarial Network (GAN) network where the discriminator (D) is used as the classifier during the testing phase.

5.1. 3D Convolutional Neural Network Classifiers

The first two models are identical deep 3D convolutional networks consisting of three convolutional layers. Stride is used instead of pooling layers for downsampling. Batch normalization and Leaky ReLU activation function is applied after 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 in order to convert the output logits to class probabilities.

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

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

In the equation, M stands for the number of examples (in a minibatch or total), C is the number of classes in the dataset, $\mathbf{y}$ is a vector of one-hot encoded training data labels, and $\hat{\mathbf{y}}$ is the classifier prediction of the class vector. The $\mathbf{y}_j^{(i)}$ notation refers to the jth element of the vector that belongs to the ith sample within the minibatch. The optimizer used to train these classifiers is RMSprop.

5.2. 3D Conditional Generative Adversarial Network

The last model is a 3D Conditional Generative Adversarial Network (3D CGAN), which consists of a generator-discriminator pair connected in sequence and trained in opposition to each other. The role of the generator is to transform a latent representation (vector) of an object to its visual representation (voxel grid), while the discriminator tries to distinguish real samples (taken from the training set) from the fake ones (created by the generator). The objective of the adversarial training is to find the equilibrium, where the generator produces samples that are realistic enough to confuse the discriminator [11]. A conditional GAN introduces a condition for the generator, specifying the desired output by providing a one-hot encoded class vector or another form of specification (e.g., keywords or description text), as originally shown in [45]. In our model, the conditioning is achieved by feeding a one-hot encoded class vector to the generator input. However, the discriminator does not receive this information, as it would trivialize its classification objective and make it unusable for classifying unknown samples (as opposed to [45]). The starting point for this architecture has been the deep convolutional 3D GAN model that was published in [10].

The main difference from the standard GAN architecture in our model is the discriminator, which is mostly identical to the plain and augmented CNN models mentioned above. The only difference between the discriminator and two previously described convolutional classifiers is in the output layer, which contains one extra output in addition to the class label, which indicates whether the input is considered to be fake or not. Essentially, it distinguishes one additional "fake" class. The layers of discriminator use the Leaky ReLU activation function, along with batch normalization.

The generative part of the CGAN network (generator), as shown in Figure 3, consists of three deconvolutional (or transposed convolutional) layers with a descending amount of volumetric filters. The upsampling of the activation volumes is achieved by kernel stride, which is equal to 2 in every layer. ReLU activation function is used after each layer, followed by a batch normalization function. The output layer uses *tanh* activation function instead to fit the output into the dataset value range of $[-1, 1]$.

The generator is conditioned by appending the desired one-hot encoded class vector to the latent feature vector, containing 200 feature values that were sampled from a Gaussian distribution $\mathcal{N}(0, 1)$ forming an input vector $\mathbf{z}$. 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 whole 3D CGAN classifier network, as shown in Figure 4, is constructed by connecting the generator and discriminator in sequence. During training, the discriminator receives separate batches of real samples from the training dataset and batches of fake models that were generated by the generator network. During another step of the same training loop, the generator is trained to produce realistic examples of the desired class. The generator loss is calculated at the output of the discriminator, whose parameters are fixed.

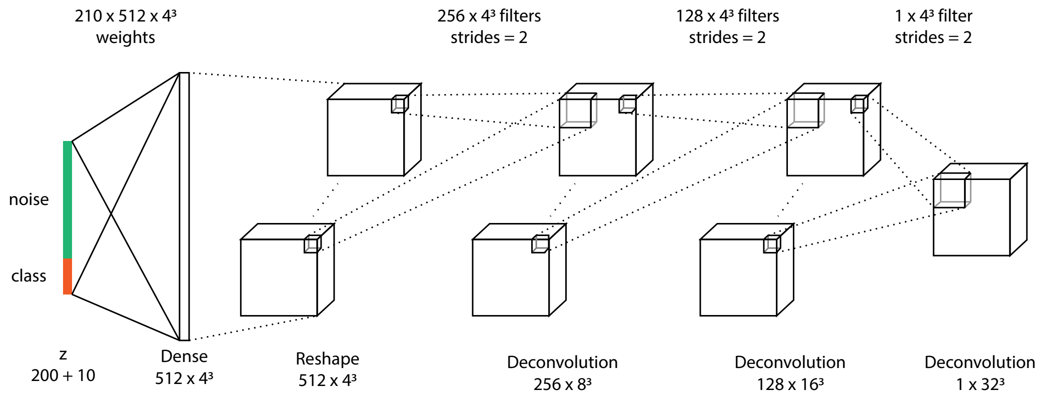

Figure 3. Architecture of the generator network. Input vector z consisting of a gaussian noise vector and a condition (class) vector is expanded in a fully connected stage and then is deconvoluted three times with batch normalization and ReLU activation function in between.

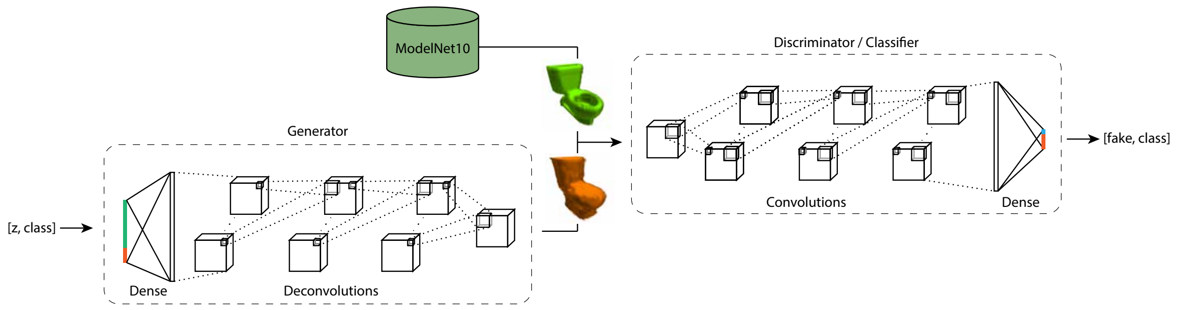

Figure 4. Architecture of the whole 3D conditional generative adversarial network classifier. Generator and discriminator are connected sequentially. The discriminator is fed both real (green) and fake (orange) examples during training.

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

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

An extended target (ground truth) vector $\mathbf{y}^*$ is constructed in order to include both fake-real discrimination error and classification error, as shown in (3). This is necessary for the adversarial training of both the generator and the discriminator so that the generator is taught to produce realistic samples. Without this extension, the generator is only trained to generate noise that will be categorized into the desired class. While this was also found to improve the accuracy of the network, the benefit was smaller when compared to the architecture using the extended ground truth vector. The idea behind this methodology is that the generator assists the discriminator-classifier to learn to utilize realistic features that are discovered and propagated from the generator network. Therefore, it is necessary to make sure that the features learned by the generator are transferrable to the real samples from the dataset.

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

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 essentially identical to the discriminator loss with a real example. The only difference is the true class vector, which is created randomly, instead of sampling it from the dataset, and used both as generator input (concatenated with noise vector) and as the target vector when compared with the discriminator output.

An alternative loss function we experimented with is a modified Wasserstein loss function [46] shown in (4) and (5).

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

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

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 is the discriminator and G the generator function. 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. The ADAM optimizer [47] is used to train both generator and discriminator.

5.3. Training

The training is concurrently performed for all three models. In each training epoch, the training data from the ModelNet10 dataset containing all 10 classes are 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 supervisedly trained in two stages, first with the training data and then with generated 'fake' data. In each stage, it is trained for both of its objectives—discrimination and classification in one weight update step (the loss function takes both objectives into account). This fake-real separation of gradient calculation has been shown to improve training stability [21]. Experiments using both approaches (separating or mixing real and fake data) have been conducted; however, no significant difference has been observed. One-sided label smoothing is also utilized to improve the model's resistance to overfitting.

6. Results and Discussion

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

6.1. Classification

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

The augmented CNN classifier also shows improvement over the plain CNN, but the difference is not as significant as between the plain CNN and GAN. This suggests that the benefits of generative enhancement do not only stem from its augmentative effect on the dataset. Some of the features learned to discriminate between fake and real examples are also likely utilized during the classification, which improves accuracy. An experiment (labeled as 'no-fake') shows a slightly lesser improvement of accuracy when training the model without using the extended ground truth vector shown in (3), therefore non-adversarially.

Table 1. Comparison of classification accuracy and mean average precision (Mean AP) score of generative modelling-based classifier architectures validated on the ModelNet10 dataset [19]. These are the results from the public ModelNet leaderboard. Some of the models do not publish their Mean AP.

	Accuracy [%]	**Mean AP [%]**
VRN Ensemble [13]	97.14	
Achlioptas et al. [14]	95.40	
VIPGAN [48]	94.05	90.69
Primitive-GAN [15]	92.20	
3D-GAN [10]	91.00	
Plain CNN (ours)	83.04	83.75
Augmented CNN (ours)	84.58	86.31
GAN Classifier (ours)	89.21	89.18
GAN Classifier no-fake (ours)	87.11	90.04

The ROC plot in Figure 5 shows another comparison of classification performance of our three models and the confusion matrix in Figure 6 provides some additional insight into classification performance.

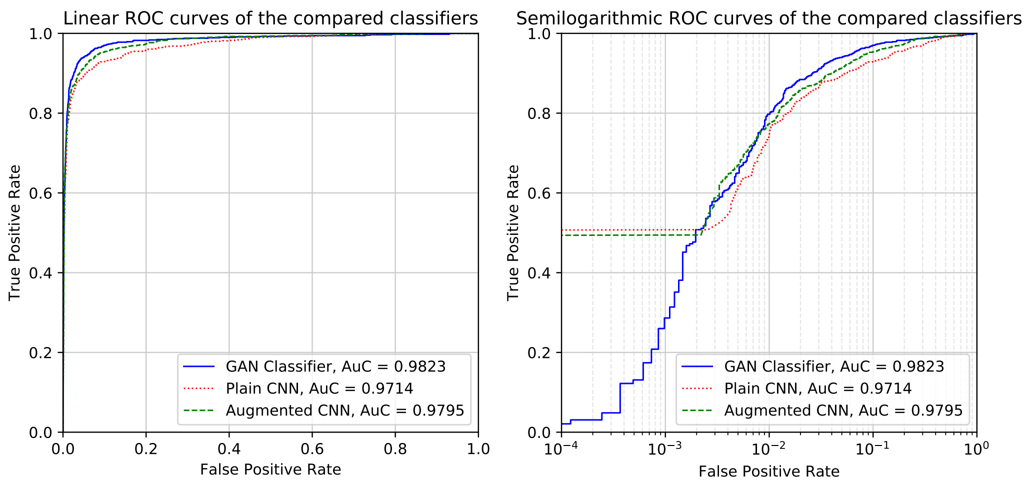

Figure 5. Linear and semilogarithmic ROC curves for each of the three designed models. These curves are calculated by micro-averaging the curves for each class.

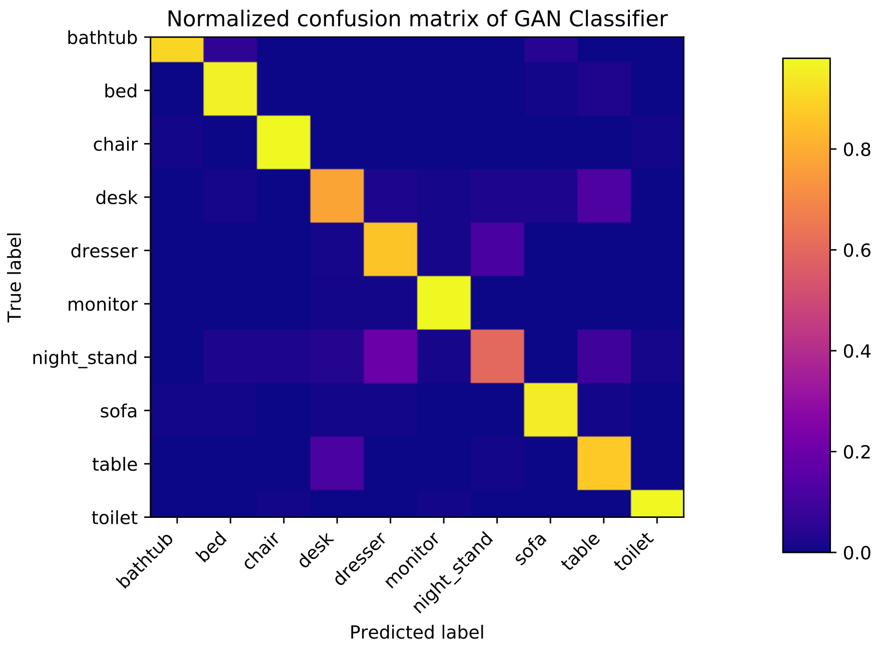

Figure 6. Confusion matrix of the GAN classifier evaluated on the ModelNet10 testing set. The values are normalized in [0, 1] range per each true label. It shows the most misclassified classes are desks, tables, and nightstands. This can most likely be attributed to low resolution and non-identical scale of the voxel models caused by the voxelization tool scaling the mesh models to fit the voxel grid volume.

Although the accuracy of the designed models is not superior to the state of the art, the results show that training a classifier inside a GAN model can produce a significant accuracy improvement when compared to conventional training. In our case, the relative improvement from plain CNN to the GAN-trained classifier was 7.43%. This is likely one of the simplest types of classifiers that is compatible with this methodology, clearly demonstrating its principle and benefits. However, the advantage of this methodology is that it is general enough to be applicable to many of the existing compatible classifiers that already achieve significantly better performance than the models used in our demonstration. The improvement of 7.43% that is shown on the simple 3D CNN is obviously not guaranteed in every case; however, it is likely that some of the more sophisticated classifiers can also benefit from this enhancement, albeit with a lower relative performance increase.

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 that 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 that 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 also likely improve the generative performance, as other researchers recommend using class labels, if available when training a GAN network in general [21].

Additionally, a series of experiments was performed in order to verify the justification of the γ hyperparameter in loss functions (4) and (5) and to find the ideal value. Despite the lower resulting performance of the models that were trained using the Wasserstein loss function, the results (shown in Figure 7) suggest that a slight bias towards classification objective can improve the classification performance. The ideal value of the hyperparameter was found to be around 10. Figure 8 shows the plots of discriminator and generator Wasserstein loss functions (4) and (5).

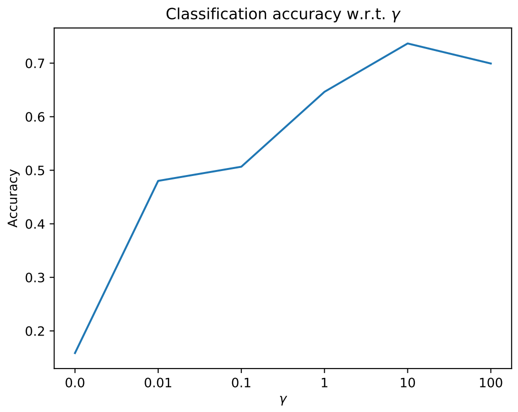

Figure 7. Classification performance of the 3D Conditional Generative Adversarial Network (3D CGAN) classifier trained using the Wasserstein loss function shown in (4) and (5) with different values of γ.

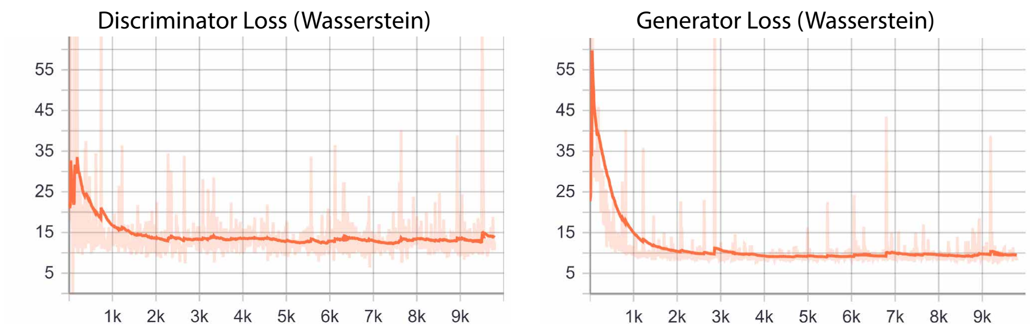

Figure 8. Plots of discriminator and generator loss trained using the Wasserstein loss functions with $\gamma = 10$.The x axis shows number of training steps. The darker lines in the plots show the exponential moving average of the actual values represented by the lighter lines.

6.2. *Object Generation*

We also examined the output of the generator in order to verify the model is working as expected. Although the quality of the generated examples was not our primary goal, it can be an indicator of errors within the architecture design or implementation. Figure 9 shows several generated examples. Mode collapse, a common pitfall for generative models, has been a major problem during our experimentation. It is a situation, when the generator only learns to produce a small subset of the real sample distribution. It manifests itself as low diversity of generated samples, where changes in the input vector z have almost no effect on the output of the generator. 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 visible intra-class variance in the output of the generator. Although some noise and artifacts are also produced, it apparently does not cause any adverse effects during classification aside from the lower visual quality of the output.

While we mostly focused on the classification of the samples, the network could be modified in order to support other tasks, such as instance identification. In our case, the conditional generator is only trained to generate a sample from a particular class, without other requirements. However, by introducing a loss term comparing the input of the generator with, for example, one of the hidden layers of the classifier, we could similarly create a generatively enhanced identifier. Such a network would not only classify a sample, but would also produce a "recipe" (latent vector) for the generator to re-create it. It would allow, for example, generating multiple original examples looking similar to a provided model by applying slight perturbations to the extracted latent vector. Such an approach could be used to enhance other kinds of deep-learning networks, such as those used for instance identification, inpainting, or shape completion.

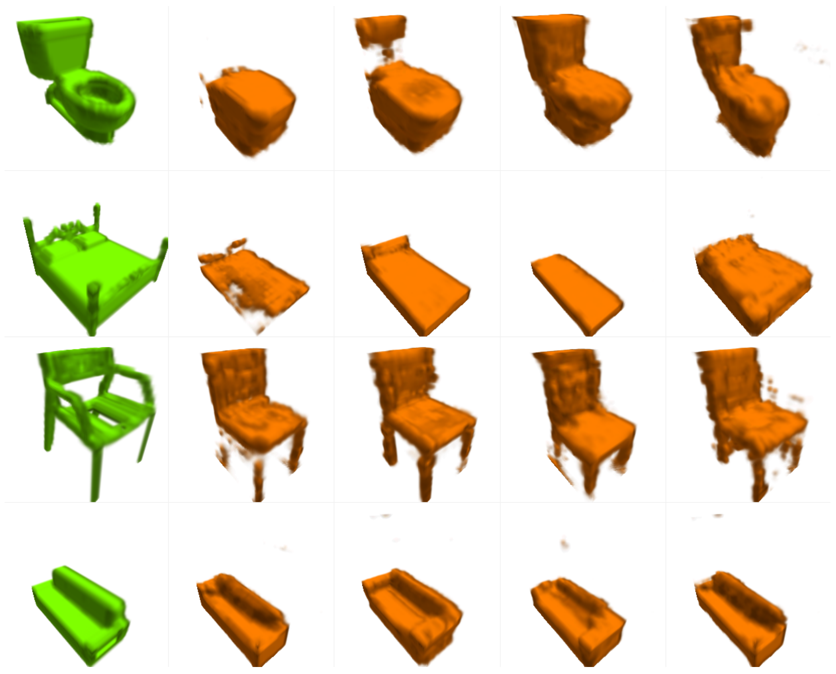

Figure 9. Output of the conditional generator using random input noise. The class condition vector has been constructed so that the rows contain examples from these ModelNet10 classes (top to bottom): toilet, bed, chair, and sofa. The first column contains a random example from the training set. The output of the generator is a 32^3 voxel grid visualized using our voxel rendering pipeline [43].

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 6700 K CPU, 16 GB RAM, Nvidia GeForce GTX 1080), the plain CNN classifier achieved its peak performance at around step 15 k. The augmented CNN classifier took 40 k steps to reach its peak. The GAN classifier reached its maximum around step 88 k. 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 separately measured 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 impact the inference computational performance. The testing set containing 908 samples has been classified in 830 ms (plain CNN), 826 ms (augmented CNN), and 786 ms (CGAN). This translates to 1092, 1098, and 1153 samples per second, respectively.

7. Conclusions and Future Work

This paper proposes a methodology for synthesizing novel 3D image classifiers by generative enhancement of existing non-generative classifiers. We experimentally verified its viability and evaluated its performance in a basic application scenario.

The results of the performed experiments support the initial idea, that a supervisedly trained conditional GAN network (i.e., generatively enhanced convolutional classifier) enables knowledge transfer between the generator and a classifying discriminator that significantly improves its classification accuracy. The underlying mechanism of these interactions is not clear and it could be the subject of further theoretical research.

One of the possible explanations for the classification performance improvement is that the generator introduces noise to the training process of the discriminator/classifier and, therefore, makes it more robust. This explanation is not completely sufficient as the same effect occurs in the augmented CNN model which does not indicate the same improvement. The same applies to another explanation, which points to the data augmentation that is caused by the generator producing realistic but not identical training examples and, therefore, improving the robustness of the classifier. This should also manifest in the augmented CNN model which, however, falls short of the CGAN classifier performance.

The results of this study suggest a potential for the generative enhancement methodology, enhancing other existing well-performing classifiers by embedding them within a generative architecture. Our next efforts will focus on exploring other candidates for this transformation and evaluating the resulting architectures. Additionally, we plan to perform a physical experiment in order to verify the real-life application potential of the newly created models. For this purpose, a custom rotary table 3D scanner is being designed to acquire scans of real-life objects, which will be used for the model evaluation. These efforts, including the existing hardware and software solutions, are part of the Technicom project [49].

Supplementary Materials: The following are available online at http://www.mdpi.com/2076-3417/10/21/7433/s1. The datasets analyzed during the current study are available on the public ModelNet website, https://modelnet.cs.princeton.edu/.

Author Contributions: J.J. is responsible for general research supervision, S.J. for research project management, and funding acquisition. J.J. and S.J. guided the overall direction of the article and methodology, provided technical guidance and simulation equipment. M.V. was a major contributor in writing the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by grant KEGA "Implementation of research results in the area of modeling and simulation of cyber-physical systems into the teaching process—development of modern university textbooks"—072TUKE-4/2018. This work was supported by the Slovak Research and Development Agency under the contract No. APVV-19-0590.

Acknowledgments: Authors are thankful for the support from project KEGA "Implementation of research results in the area of modeling and simulation of cyber-physical systems into the teaching process—development of modern university textbooks"—072TUKE-4/2018. This work was supported by the Slovak Research and Development Agency under the contract No. APVV-19-0590.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Hartmann, K.G.; Schirrmeister, R.T.; Ball, T. EEG-GAN: Generative Adversarial Networks for Electroencephalograhic (EEG) Brain Signals. *arXiv* **2018**, arXiv:1806.01875.
2. Chen, L.; Dai, S.; Tao, C.; Zhang, H.; Gan, Z.; Shen, D.; Zhang, Y.; Wang, G.; Zhang, R.; Carin, L. Adversarial Text Generation via Feature-Mover's Distance. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 4666–4677.
3. Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; Metaxas, D.N. Stackgan: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5907–5915.
4. Pascual, S.; Bonafonte, A.; Serra, J. SEGAN: Speech Enhancement Generative Adversarial Network. *arXiv* **2017**, arXiv:1703.09452.
5. Vondrick, C.; Pirsiavash, H.; Torralba, A. Generating Videos with Scene Dynamics. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 613–621.
6. Wang, Y.; Tao, X.; Shen, X.; Jia, J. Wide-Context Semantic Image Extrapolation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 1399–1408.

7. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690.
8. Jiang, Y.; Xu, J.; Yang, B.; Xu, J.; Zhu, J. Image Inpainting Based on Generative Adversarial Networks. *IEEE Access* **2020**, *8*, 22884–22892. [CrossRef]
9. Tasse, F.P.; Dodgson, N. Shape2Vec: Semantic-Based Descriptors for 3D Shapes, Sketches and Images. *ACM Trans. Graph. (TOG)* **2016**, *35*, 208:1–208:12. [CrossRef]
10. Wu, J.; Zhang, C.; Xue, T.; Freeman, B.; Tenenbaum, J. Learning a Probabilistic Latent Space of Object Shapes via 3d Generative-Adversarial Modeling. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 82–90.
11. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Palais des Congrès de Montréal, Montréal, QC, Canada, 2014; pp. 2672–2680.
12. Odena, A. Semi-Supervised Learning with Generative Adversarial Networks. In Proceedings of the Workshop on Data-Efficient Machine Learning (ICML 2016), New York City, NY, USA, 19–24 June 2016.
13. Brock, A.; Lim, T.; Ritchie, J.M.; Weston, N.J. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. In Proceedings of the Neural Inofrmation Processing Conference, Barcelona, Spain, 5–10 December 2016; pp. 1–9.
14. Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; Guibas, L. Learning Representations and Generative Models for 3D Point Clouds. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 40–49.
15. 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, Long Beach, CA, USA, 15–20 June 2019; pp. 9739–9748.
16. Kanezaki, A.; Matsushita, Y.; Nishida, Y. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5010–5019.
17. Han, Z.; Wang, X.; Vong, C.M.; Liu, Y.S.; Zwicker, M.; Chen, C.P. 3DViewGraph: Learning Global Features for 3D Shapes from a Graph of Unordered Views with Attention. In Proceedings of the 2019 International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
18. Han, Z.; Liu, X.; Liu, Y.S.; Zwicker, M. Parts4Feature: Learning 3D Global Features from Generally Semantic Parts in Multiple Views. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
19. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3d Shapenets: A Deep Representation for Volumetric Shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
20. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2016; NIPS'16; pp. 2234–2242.
21. Chintala, S. How to Train a GAN? Tips and Tricks to Make GANs Work. 2019. Available online: https://github.com/soumith/ganhacks (accessed on 14 October 2020).
22. Antoniou, A.; Storkey, A.; Edwards, H. Data Augmentation Generative Adversarial Networks. *arXiv* **2018**, arXiv:1711.04340.
23. Wang, J.; Perez, L. The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. *Convolutional Neural Netw. Vis. Recognit.* **2017**, *11*, abs/1712.04621 .
24. Tanaka, F.H.K.d.S.; Aranha, C. Data Augmentation Using GANs. *arXiv* **2019**, arXiv:1904.09135.
25. Iqbal, T.; Ali, H. Generative Adversarial Network for Medical Images (MI-GAN). *J. Med. Syst.* **2018**, *42*, 231. [CrossRef] [PubMed]
26. Židek, K.; Lazorík, P.; Pitel', J.; Hošovský, A. An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition. *Symmetry* **2019**, *11*, 496. [CrossRef]

27. Odena, A.; Olah, C.; Shlens, J. Conditional Image Synthesis With Auxiliary Classifier GANs. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; PMLR: International Convention Centre, Sydney, Australia, 2017; Volume 70, pp. 2642–2651.
28. Mariani, G.; Scheidegger, F.; Istrate, R.; Bekas, C.; Malossi, C. BAGAN: Data Augmentation with Balancing GAN. *arXiv* **2018**, arXiv:1803.09655.
29. Sutskever, I.; Jozefowicz, R.; Gregor, K.; Rezende, D.; Lillicrap, T.; Vinyals, O. Towards Principled Unsupervised Learning. *arXiv* **2015**, arXiv:1511.06440.
30. Springenberg, J.T. Unsupervised and Semi-Supervised Learning with Categorical Generative Adversarial Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
31. Shen, P.; Lu, X.; Li, S.; Kawai, H. Conditional Generative Adversarial Nets Classifier for Spoken Language Identification. In Proceedings of the INTERSPEECH 2017, Stockholm, Sweden, 20–24 August 2017. [CrossRef]
32. Fan, H.; Su, H.; Guibas, L.J. A Point Set Generation Network for 3d Object Reconstruction from a Single Image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 605–613.
33. Ren, M.; Niu, L.; Fang, Y. 3D-A-Nets: 3D Deep Dense Descriptor for Volumetric Shapes with Adversarial Networks. *arXiv* **2017**, arXiv:1711.10108.
34. Wang, W.; Huang, Q.; You, S.; Yang, C.; Neumann, U. Shape Inpainting Using 3d Generative Adversarial Network and Recurrent Convolutional Networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2298–2306.
35. Yi, L.; Zhao, W.; Wang, H.; Sung, M.; Guibas, L.J. Gspn: Generative Shape Proposal Network for 3d Instance Segmentation in Point Cloud. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 3947–3956.
36. Wang, W.; Yu, R.; Huang, Q.; Neumann, U. Sgpn: Similarity Group Proposal Network for 3d Point Cloud Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2569–2578.
37. Varga, M.; Jadlovský, J. Evaluation of Depth Modality in Convolutional Neural Network Classification of RGB-D Images. *Acta Electrotech. Inform.* **2018**, *18*, 26–31. [CrossRef]
38. Jiang, J.; Bao, D.; Chen, Z.; Zhao, X.; Gao, Y. MLVCNN: Multi-Loop-View Convolutional Neural Network for 3D Shape Retrieval. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 8513–8520.
39. Bazrafkan, S.; Corcoran, P. Versatile Auxiliary Classifier with Generative Adversarial Network (Vac+ Gan), Multi Class Scenarios. *arXiv* **2018**, arXiv:1806.07751.
40. Varga, M.; Jadlovský, J. Contribution to Generative Modelling and Classification of 3D Images. In *SCYR 2018: Proceedings from Conference*; TUKE: Herl'any, Slovakia, 2018; pp. 61–62.
41. Min, P. Binvox. 2004. Available online: http://www.patrickmin.com/binvox (accessed on 14 October 2020).
42. Nooruddin, F.S.; Turk, G. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Trans. Vis. Comput. Graph.* **2003**, *9*, 191–205. [CrossRef]
43. Varga, M.; Jadlovský, J. Contribution to 3D Voxel Generative Adversarial Networks. In *SCYR 2019: Proceedings from Conference*; TUKE: Herl'any, Slovakia, 2019; pp. 56–57.
44. Campagnola, L. PyQtGraph. 2012. Available online: http://www.pyqtgraph.org (accessed on 14 October 2020).
45. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2014**, arXiv:1411.1784.
46. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. In Proceedings of the 34th International Conference on Machine Learning, , Sydney, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; PMLR: International Convention Centre, Sydney, Australia, 2017; Volume 70, pp. 214–223.
47. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
48. 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, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 8376–8384.

49. Jadlovský, J.; Jadlovská, A.; Jadlovská, S.; Čerkala, J.; Kopčík, M.; Čabala, J.; Oravec, M.; Varga, M.; Voščšek, D. Research Activities of the Center of Modern Control Techniques and Industrial Informatics. In Proceedings of the 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 21–23 January 2016; pp. 279–285. [CrossRef]