Article

Synthetic data generation to speed-up the object recognition pipeline

Damiano Perri[1,2]0000-0001-6815-6659, Marco Simonetti[1,2]0000-0003-2923-5519, and Osvaldo Gervasi[2]0000-0003-4327-520X,

[1] University of Florence, Italy; damiano.perri@unifi.it, m.simonetti@unifi.it
[2] University of Perugia, Italy; osvaldo.gervasi@unipg.it
* Correspondence: damiano.perri@unifi.it

Abstract: This paper provides a methodology for the production of synthetic images for training neural networks to recognise shapes and objects. There are many scenarios in which it is difficult, expensive and even dangerous to produce a set of images that is satisfactory for the training of a neural network. The development of 3D modelling software has nowadays reached such a level of realism and ease of use that it seemed natural to explore this innovative path and to give an answer regarding the reliability of this method that bases the training of the neural network on synthetic images. The results obtained in the two proposed use cases, that of the recognition of a pictorial style and that of the recognition of migrants at sea, leads us to support the validity of the approach, provided that the work is conducted in a very scrupulous and rigorous manner, exploiting the full potential of the modelling software. The code produced, which automatically generates the transformations necessary for the data augmentation of each image, and the generation of random environmental conditions in the case of Blender and Unity3D software, is available under the GPL licence on GitHub. The results obtained lead us to affirm that through the good practices presented in the article, we have defined a simple, reliable, economic and safe method to feed the training phase of a neural network dedicated to the recognition of objects and features, to be applied to various contexts.

Keywords: Unity3D, Blender, Virtual Reality, Syntetic dataset generation, Machine learning, Neural Networks

1. Introduction

Machine learning is now one of the areas where scientific research has focused most. In order to train the neural networks correctly, it is necessary to have available datasets of examples that the network can use to learn and understand how to solve the problem that has been assigned to it. The datasets for the training of the neural networks are generally very large and require considerable efforts to be constructed correctly. Let us think, for example, of the convolutional neural networks: these are used today to extract the features that compose the images and classify them according to the labels that the programmer has predefined. As an example let us imagine a dataset for binary classification of animals, such as dogs or cats. Unless we consider a dataset that is already available on the web it will be necessary to create one specific to the problem to be addressed. Such a dataset would be very complex and expensive to create in the real world, and is a general case of representing three-dimensional environments with completely random lighting conditions and object arrangements, so it may be appropriate and advantageous to create it virtually, thanks to the enormous developments that have taken place in 3D modelling software. This solution allows us to recreate virtual scenarios, generating a high number of images with specific techniques of scene illumination and a random

arrangement of objects to enrich the number of information to be fed to the neural network for its training.

In this paper we present a technique for generating synthetic datasets using the popular three-dimensional environment modelling software Unity3D[1] and Blender[2].

We illustrate two different *use cases*, which we think can be of valuable help to researchers, as they are general cases that can help to solve specific problems. The first *use case* involves generating a dataset for the binary classification of paintings of the Baroque and Impressionist styles.

With this example we want to show and describe how it is possible to generate images of objects (in our case paintings) and how they can be analysed using neural networks. This use case therefore involves framing a three-dimensional model from various angles and under various lighting conditions. The second use case involves generating a dataset for recognising people at sea, specifically migrants.

This example is completely different from the previous one because instead of having an object, we have a scenario, and through graphical modelling environments we can easily recreate different weather and light conditions: for example we can have a scenario with the typical sunlight of the morning, afternoon, evening or night. We can also simulate what people at sea would look like in different weather conditions, for example clear skies or cloudy skies.

A further qualifying aspect of our work is that we have automated the process of data augmentation on virtually generated images by manipulating each image in various aspects.

The sections that make up this paper are divided in the following manner. In the section 2 we analyse articles and manuscripts that have dealt with this problem or have analysed the correlations that these methodologies may have with the world of scientific research. The section 3 describes how we organised the research, analysing both the theoretical and practical development of our work. In section 4, we describe our results and also analyse best practices for generating synthetic datasets using three-dimensional environment modelling software. The section 5 describes the techniques we recommend for generating synthetic datasets. The section 6 reports the conclusions we drew as a result of the experimental analysis described in this paper. The datasets that are generated for this article and the codes we developed to implement them have been made public and are freely accessible via the GitHub page[3]. All the code on the repository is open source, licensed under the GNU General Public License v3.0 and freely usable by anyone.

2. Related works

Supervised Machine Learning for Image Recognition, like any other human endeavour, has yielded significant advantages while simultaneously posing new challenges. Indeed, one of the most significant concerns picture recognition reliability when the data collection comprises a small number of samples on which to base training.

Despite the fact that there are several picture databases online, both private and public, even providing open access (i.e. ImageNet[4], OpenImages[5], SUN Database[6], Microsoft Common Objects in Context - COCO[7], PASCAL VOC Dataset [8], OpenLORIS-Object[9], etc.), there are still many niches of themes that are not depicted or for which there are only a few photos accessible. So, many techniques have been devised to enhance the number of available occurrences, such as geometric transformations [1–3], parameter adjustment (number of pixels, colour mappings, contrast, multi-spectral bands, etc.)

---

1 https://unity.com/
2 https://blender.org/
3 https://github.com/DamianoP/DatasetGenerator
4 https://www.image-net.org
5 https://github.com/openimages
6 http://vision.princeton.edu/projects/2010/SUN/SUN397.tar.gz
7 https://cocodataset.org
8 http://host.robots.ox.ac.uk/pascal/VOC/databases.html
9 https://lifelong-robotic-vision.github.io/dataset/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

[4,5], and synthetic picture generation (via GAN/RAN [6–10] or graphics engines, like Unity3D, GODOT[10] [11], and Unreal[11]).

Another problem deriving from the use of some particular types of datasets is the imbalance of the classes in the classification process (*unbalanced classes*) [12,13]. This phenomenon has been shown to have a negative impact on traditional classifier training. Many of these strategies can efficiently solve the problem of unbalancing classes [14–17] in datasets when their quantity is not exactly uniform, but oversampling minor classes [18–20], undersampling major ones [21–23], the possibility of weighing the network parameters in a different way so as to appropriately rebalance all the classes [24–26], appear to be winning strategies too. By contrast, this issue appears to be decreasing in all binary classification methods [27,28].

Furthermore, synthetic scenario creation is becoming increasingly relevant in a variety of fields [29–35], ranging from object and figure identification and categorisation to automated recognition and tracing [36–39]. The adaptability and reusability of this method is exemplified by the ability to train networks to detect specific items or people in very challenging circumstances. So, the use of powerful graphics engines that are able to reproduce reality, or a scenario to be represented, in a very realistic way is therefore becoming a particularly crucial practise for increasing or balancing the recognition classes in a dataset for CNNs [40–42].

3. Research methodology

When approaching a machine learning challenge, the first step is to identify a dataset that accurately defines the problem and train a classifier using it, such as a decision tree, a neural network, or a Support Vector Machine. Manually constructing a dataset is a time-consuming, sometimes expensive, and even risky task. Consider the following scenarios: you want to train a neural network that recognises men overboard, or you want to train a neural network that recognises animals that pass along a road at night: these are specific scenarios that could present many challenges for a researcher, as well as a significant cost and time benefit from using graphic modelling software. The pipeline that we suggest for approaching these difficulties is depicted in Figure 1, which begins with the development of a synthetic dataset before moving on to the actual one.

3.1. Our proposed pipeline

The pipeline that we proposed is now being evaluated. First and foremost, the virtual setting must be designed and built using software such as Unity3D and Blender. This is a crucial stage, and having some example images and a clear sense of how the setting will be constructed might help. After you have finished creating the scenario, you may start creating the synthetic dataset. Photographs of the virtual world must be taken in this step, using combinations of light, shadows, and items that we deem appropriate for the research. The acquired photographs can then be used to train a neural network, with care taken to divide the images into two sets, the first of which contains 80% of the samples and can be used to compose the training set, and the second of which contains the remaining 20% and can be used to test the neural network with examples not used in the training phase. After training the neural network, the results must be analysed: if the results are poor, the generated dataset must be double-checked, and the scenario modelling may need to be adjusted. If the results are satisfactory, we may assert that our problem can be solved using neural networks, and we can spend time and money looking for and photographing objects in the real world. If financial resources allow, it may be able to replace the synthetic images with real ones and evaluate the dataset generated by the neural network using the continuous learning approach. If the results are positive, the task is completed. Otherwise,

[10] https://godotengine.org/
[11] https://www.unrealengine.com

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

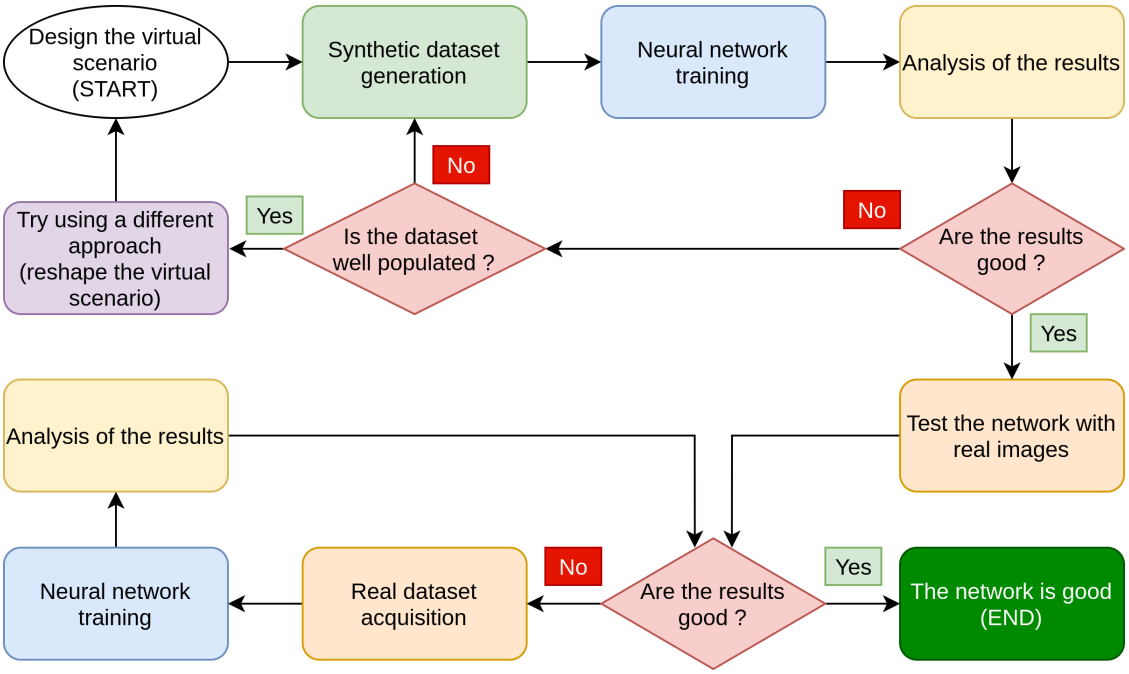


Figure 1. Flow chart summarising the various phases of the training of a neural network using virtual scenarios.

you will have to go back and examine the dataset of real photographs, train the network again, and see if the goals have been met.

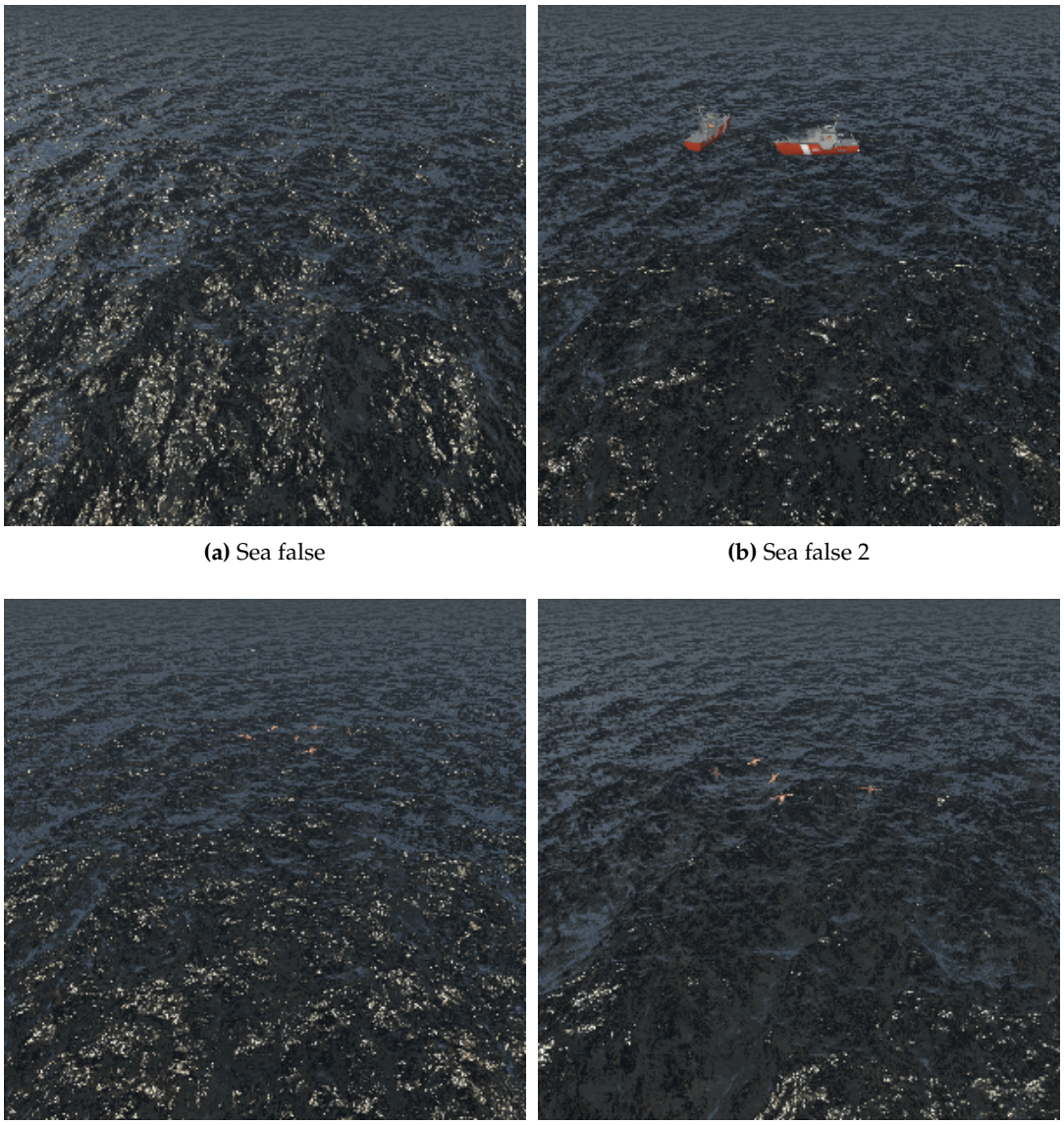
Instead of immediately beginning with the capture of real-world images, we advocate starting with synthetic photographs created using 3D modelling software utilising our pipeline. If the three-dimensional environment is created with care, realism, and detail fidelity, we will have a clear idea of the performances that we will be able to obtain in the real world quickly and with low initial costs, and only then begin the construction phase of a dataset with photographs taken in the real world.

We have discovered a number of advantages using this method. The first benefit is the speed with which we can generate photographs to train the networks, because once we have set up the working environment, we will be able to generate an almost infinite number of images by simply running our algorithm and letting it generate images with random combinations of lights, shadows, and objects. Another advantage is cost; developing a synthetic dataset is far less expensive than creating a genuine dataset, which in our example would include the hiring of a helicopter, actors, and at least one ship. It should also be remembered that by utilising a synthetic dataset, we will be able to determine much more quickly if the technique we are applying is appropriate or not, and, if required, entirely alter strategy and attack the problem from different perspectives. We believe that a synthetic dataset should not be used in place of a dataset made up of real images because it is currently impossible to faithfully reconstruct all of the graphic facets and decals that make up the natural world, but we do believe that it is a useful tool for researchers working on machine learning problems, particularly those involving image classification.

In Figure 2 it is possible to see some photographs of the scenario we created. In Figures 2A and 2B we can see two examples of photos in which there are no men overboard. In Figures 2C and 2D there are men overboard, generated randomly and in a random point of the scene.

In Figure 3 we find 4 examples of paintings, which make up the dataset of objects we have created. In Figures 3A and 3B are represented paintings of the Impressionist period, while in Figures 3C and 3D are represented paintings of the Baroque period.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1



(a) Sea false **(b)** Sea false 2

(c) Sea true **(d)** Sea true 2

Figure 2. Sea scenarios.

4. Discussion of results

We evaluated the synthetic datasets with neural networks after synthesising them with virtual reality software using the methodology outlined in the previous sections. The goal of the test is to determine whether or not a neural network can operate with pictures that are not from the real world and whether or not the results it generates are adequate. The analysis was carried out using two convolutional neural networks. The first network is Alexnet[43] while the second neural network is InceptionResnet-V2[44]. We trained the first neural network via Matlab[12] software while the second network was trained via python code running on the Google Colab[13] cloud environment. Both networks were trained using the transfer learning technique[45]. This approach assumed that the networks would be partially trained before the training began. Before the actual training, the weights

[12] https://it.mathworks.com/products/matlab.html
[13] https://colab.research.google.com/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

(a) impressionism **(b)** impressionism

(c) baroque **(d)** baroque

Figure 3. Sample paintings used in our work.

of the neural connections are preloaded. The values are derived from the training that these networks performed on the public dataset ImageNet[46]. We then deleted the network's head, as well as the initial 1000-class prediction layer, and added a layer to conduct binary classification of our photos, for example, by utilising the sigmoid activation function[47]. Finally, after evaluating the general-purpose networks mentioned above, we constructed a customised neural network by modelling the internal structure of the layers to match our proposed problem. This network comprises 12 million neurons, which is a small amount in comparison to general-purpose networks, yet it has produced good performance in the prediction phase on the validation set.

4.1. Alexnet

Alexnet was the first neural network we looked at, and we used Matlab software to analyse it. Using the transfer learning method, the neural network was imported and analysed. The network's last three layers were eliminated after it was pre-trained on the ImageNet dataset. These layers acted as learning layers for the network, allowing it to recognise the 1000 ImageNet classes. We have replaced these layers with a dense, fully connected layer with a WeightLearnRateFactor of 20 and a BiasLearnRateFactor of 20. A Softmax layer and, lastly, a Classification Layer were attached to this

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

layer. We examined both the dataset depicting paintings and the dataset representing people at sea with the network setup in this way.

Figure 4 shows the confusion matrices obtained by analysing the dataset of the paintings.

Training Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	816 50%	0 0%	816 100% 0%
Impressionism	0 0%	816 50%	816 100% 0%
SUM	816 100% 0%	816 100% 0%	1632 / 1632 100% 0%

(a) Training set Confusion Matrix

Validation Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	204 50%	0 0%	204 100% 0%
Impressionism	10 2.45%	194 47.54%	204 95.09% 4.9%
SUM	214 95.32% 4.68%	194 100% 0%	398 / 408 97.54% 2.45%

(b) Validation set Confusion Matrix

Figure 4. Confusion matrices of the Painting dataset analysed on Alexnet with Matlab.

Figure 5 shows the confusion matrices obtained by analysing the dataset of men at sea.

Training Set

TARGET / OUTPUT	False	True	SUM
False	800 50%	0 0%	800 100% 0%
True	0 0%	800 50%	800 100% 0%
SUM	800 100% 0%	800 100% 0%	1600 / 1600 100% 0%

(a) Training set Confusion Matrix

Validation Set

TARGET / OUTPUT	False	True	SUM
False	179 44.75%	21 5.25%	200 89.5% 10.5%
True	1 0.25%	199 49.75%	200 99.5% 0.5%
SUM	180 99.44% 0.56%	220 90.45% 9.54%	378 / 400 94.5% 5.5%

(b) Validation set Confusion Matrix

Figure 5. Confusion matrices of the Sea dataset analysed on Alexnet with Matlab

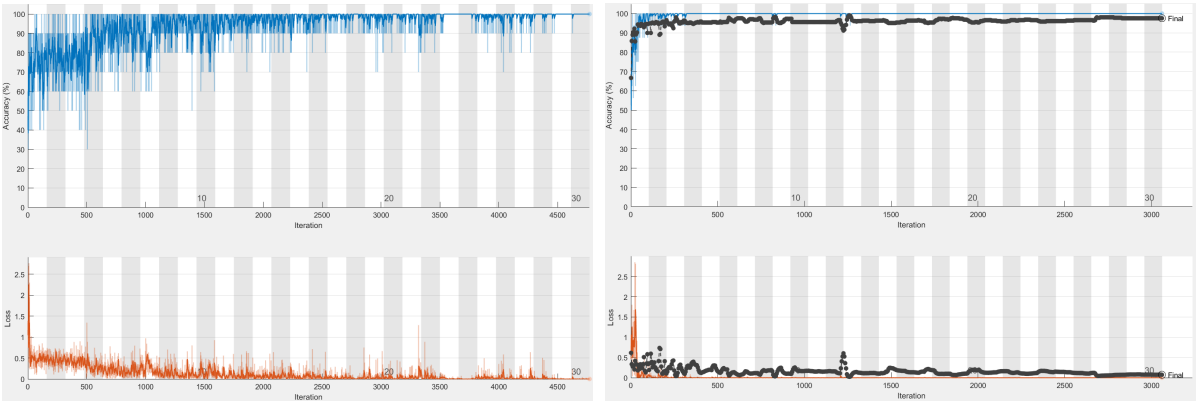
Figure 6 shows the result of Alexnet's training on the two datasets.

The graph showing the network's percentage of accuracy as a function of time can be seen in the upper section of the photos. In the lower section of the photos, the Loss function as a function of time is presented.

The learning curve of the network in the recognition of the training set is shown in blue in Figure 6A, while the dataset of men at sea is studied. As you can see, the network begins to settle after a few early oscillations, becoming increasingly precise.

Figure 6B, on the other hand, depicts the network's learning curve as it analyses the collection of paintings. Blue highlights the various training iterations. We have added periodic checks in the recognition of the validation set as an extra analysis in this example, and the interpolation of the recognition percentages achieved is shown by the black curve. Figure 5B shows the final result in the recognition of the validation set of the man overboard dataset. The findings acquired from the first tested network are, in our perspective, extremely good; in particular, we can see how the validation set paintings are identified with an accuracy of 97.5%. The accuracy of the dataset of men at sea, on the other hand, is lower, with an overall accuracy of 94.5%. When in confusion matrix we consider the cell

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1



(a) Sea dataset **(b)** Paintings dataset

Figure 6. Training curve of the datasets analysed on Alexnet with Matlab

corresponding to the column "true" and the row "true", that is, when there are really men overboard, we can see that 199 of the 200 photos in the sample were properly identified. The Alexnet neural network described above takes up a total of 201.9MB on disk.

4.2. InceptionResNet-V2

We used the Google Colab cloud environment to test our second system, the InceptionResNet-V2 neural network. This network was also preloaded using the transfer learning approach, thus the weights of the neurons were already capable of classifying the 1000 ImageNet classes. We imported the network, removed the network's head, which was made up of dense learning layers, and replaced it with the following layers. The first is a Flatten type layer, whose job was to convert the network's data structure into a float vector. After that, we added two dense layers of 64 neurons with Rectified Linear Unit (ReLU) activation functions to learn the features collected from the network's convolutional layers. Finally, we have added a layer that just contains one neuron and is responsible for binary categorization. Binary crossentropy[14] was employed as the loss function, while Adam[15][48] was chosen as the optimizer. In a cloud environment like Google Colab, the network described above has a total of 60,632,481 parameters, which is a big quantity to maintain and handle. As a consequence, we devised a strategy that enabled us to operate effectively with such a large network. The early layers of the InceptionResNet-V2 network were frozen, and only the final layers were permitted to be trained. The layers that extract picture characteristics and are pre-trained on ImageNet are not impacted in this way. Because of this approach, 54,336,736 parameters are immutable and will not change during the network's training, whereas the remaining 6,295,745 parameters will need to be trained in order to learn how to accurately categorise our datasets. We have additionally set up two more aspects. The first is in the training phase: we assess the network accuracy percentage on the validation set every time an epoch finishes. If the accuracy rate has increased, the network model is saved and this step is considered a checkpoint. In this method, even if the network overfits during training and its accuracy on the validation set deteriorates, we may still trace the optimum combination of parameters gained during training. The second point to consider is when to halt (*early stopping*). We can track the development of the network training phase and verify its accuracy in identifying the dataset using this method. At the conclusion of each period, a check is performed. If the network does not increase recognition accuracy for a predefined number of epochs (generally three), we may say that we have found a local minimum in the range of possible solutions to the problem. This abnormality will be

14 https://keras.io/api/losses/probabilistic_losses/
15 https://keras.io/api/optimizers/adam/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

detected, and network training will be halted, saving time that would otherwise be squandered. The confusion matrices derived by examining the dataset of the paintings are shown in Figure 7. The validation set, as it can be seen, has a very high accuracy percentage of 98.03%.

Training Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	778 47.67%	39 2.38%	817 95.22% 4.78%
Impressionism	0 0%	815 49.93%	815 100% 0%
SUM	778 100% 0%	854 95.43% 4.56%	1593 / 1632 97.61% 2.39%

(a) Training set Confusion Matrix

Validation Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	195 47.79%	8 1.96%	203 96.05% 3.95%
Impressionism	0 0%	205 50.24%	205 100% 0%
SUM	195 100% 0%	213 96.24% 3.76%	400 / 408 98.03% 1.96%

(b) Validation set Confusion Matrix

Figure 7. Confusion matrices of the Painting dataset analysed on InceptionResNet-V2 with Google Colab

Figure 8 shows the confusion matrices obtained by analysing the dataset of men at sea with the InceptionResNet-V2 neural network. In this case the network was able to recognise each image provided through the validation set, reaching an accuracy of 100%.

Training Set

TARGET / OUTPUT	False	True	SUM
False	799 49.93%	1 0.06%	800 99.87% 0.12%
True	0 0%	800 50%	800 100% 0%
SUM	799 100% 0%	801 99.87% 0.12%	1599 / 1600 99.93% 0.06%

(a) Training set Confusion Matrix

Validation Set

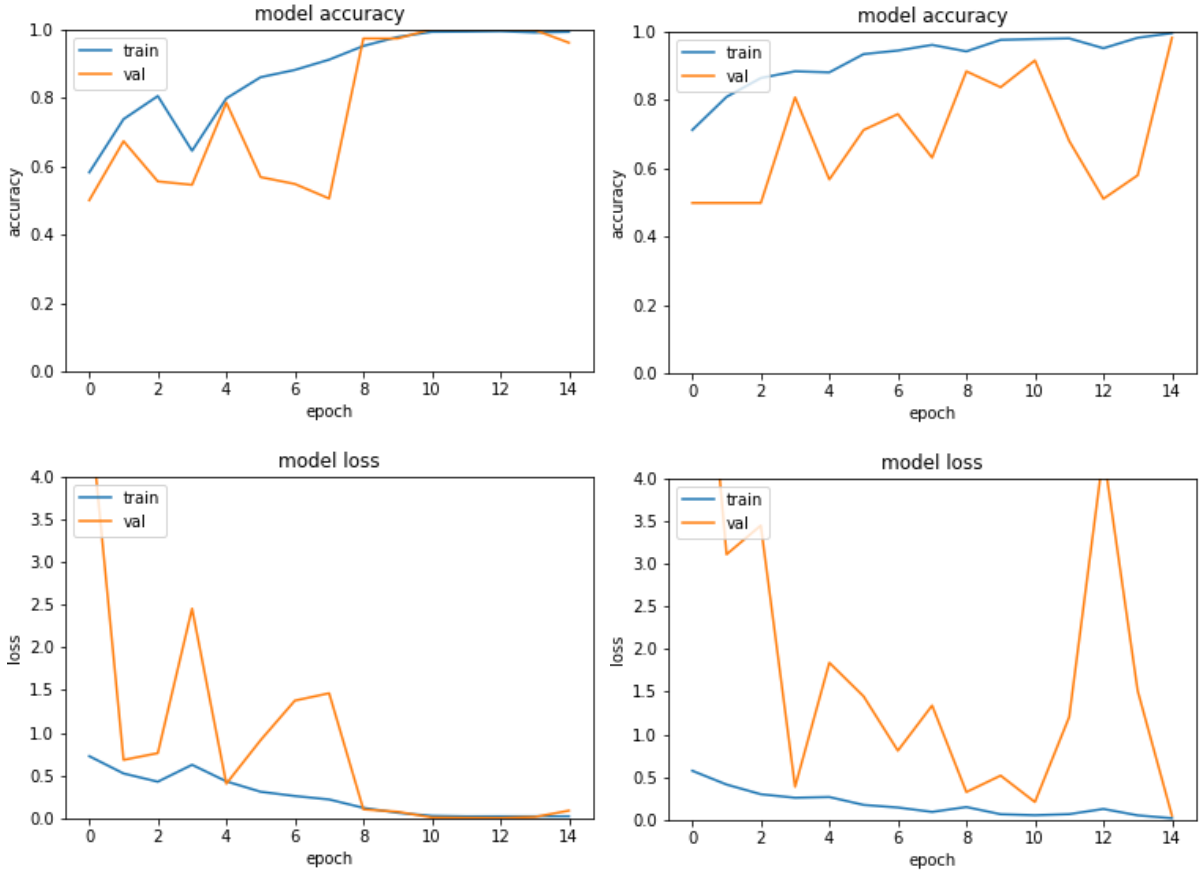
TARGET / OUTPUT	False	True	SUM
False	201 50.25%	0 0%	201 100% 0%
True	0 0%	199 49.75%	199 100% 0%
SUM	201 100% 0%	199 100% 0%	400 / 400 100% 0%

(b) Validation set Confusion Matrix

Figure 8. Confusion matrices of the Sea dataset analysed on InceptionResNet-V2 with Google Colab

Furthermore we report in Figure 9 the graphs obtained with the training of the neural network. The graphs show how the proportion of accuracy has increased over time, peaking at epoch n.10 and then declining. The network then entered an overfitting phase, during which the accuracy of the validation set decreased significantly. We were able to save the condition in which the network was at its highest level of accuracy, which was obtained around epoch n.10, thanks to early stopping and checkpoints. The neural network as described takes up 54.5MB of storage space. The convolutional layers have been frozen, thus no changes in the weight values of the neural interconnections have been made compared to the network trained on ImageNet, allowing for such a small amount of space to be filled.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1



(a) Sea dataset **(b)** Paintings dataset

Figure 9. Training curve of datasets analysed on InceptionResNet-V2 with Google Colab

4.3. Custom Convolutional Neural Network

To finish our study, we constructed a custom neural network for the collection of paintings. The goal of this network's architecture is to achieve a more linear training phase than InceptionResNet-V2. Indeed, we want to avoid the frequent spikes and decays in accuracy that we saw with the general-purpose network, and we want the loss function to be smoother. In this approach, we want to develop a neural network that is theoretically more stable when applied to pictures that are not synthetically created. The figure 10 depicts the network we are presenting.

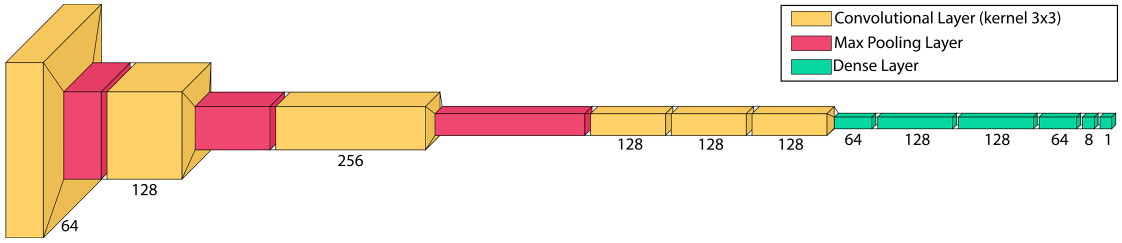


Figure 10. Structure of the custom Convolutional Neural Network

There are three blocks to its construction. The identification of the highest level characteristics is the initial block. We employ a convolutional layer sequence and max pooling to do this. The three first convolutional layers have 64, 128 and 256 filters, respectively, as we go deeper into the network. The second block is made up of three convolutional layers, each with 128 filters. These layers are responsible for learning the finer details of our datasets. A series of dense layers with ReLU activation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

functions composes the third block. These layers are responsible for learning and categorising the characteristics retrieved by convolutional layers. The network's last layer is composed of a single neuron with a sigmoid activation function. Adam was the optimizer for this neural network, and there were a total of 12,209,553 parameters to train. The setup of the neuronal weights of this neural network requires 139.8MB of storage space. Figure 11 shows the confusion matrices created by testing the neural network on the datasets. As it can be seen, the recognition percentages are quite high, with a validation set accuracy of 97.54%. Figures 12A and 12B show the statistics gained during the training

Training Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	804 49.26%	4 0.24%	808 99.5% 0.5%
Impressionism	2 0.12%	822 50.36%	824 99.75% 0.25%
SUM	806 99.75% 0.25%	826 99.51% 0.48%	1626 / 1632 99.63% 0.37%

(a) Training set Confusion Matrix

Validation Set

TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	206 50.49%	6 1.47%	212 97.16% 2.84%
Impressionism	4 0.98%	192 47.05%	196 97.95% 2.04%
SUM	210 98.09% 1.9%	198 96.96% 3.04%	398 / 408 97.54% 2.45%

(b) Validation set Confusion Matrix

Figure 11. Confusion matrices of the Paintings dataset analysed on Custom-CNN with Google Colab

phase, whereas Figure 9D shows the Roc curve. As it can be seen, the learning curve is highly steady, and the validation set's identification percentage is very near to the training set's recognition accuracy. Checkpoints and early stopping were also implemented in this situation, allowing us to save the optimal configuration of neural weights that the training phase could create. The best configuration was found at the end of epoch n.11, however the early stopping of the training at epoch n.14, as the network was approaching overfitting, caused the training to be terminated. Finally, we conducted deeper investigation. We created a test set of paintings that were not included in our synthetic dataset and utilised it to see if the neural network we trained could distinguish the painting style. The analysis gave very good results with an accuracy percentage of 94%. The confusion matrix constructed on the test set as presented is shown in Figure 12C.

5. Best Practices generating a synthetic dataset in virtual environments

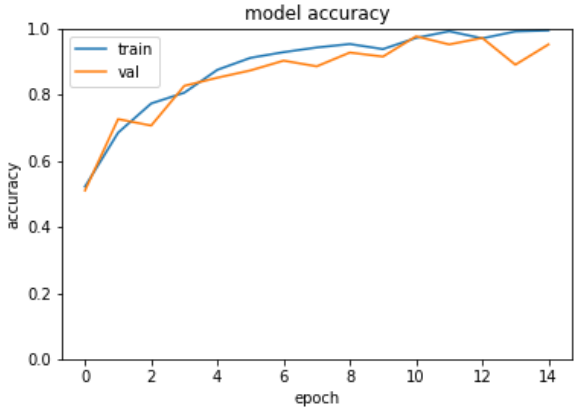
This section describes the techniques we recommend for generating synthetic datasets. In subsection 5.1 we describe the techniques we recommend for generating synthetic scenarios using Unity3D while in subsection 5.2 we describe how to generate synthetic datasets depicting objects. The techniques described can be applied generically to many other graphic modelling software, we used Unity3D only as a use case.

5.1. Representation of the synthetic scenario

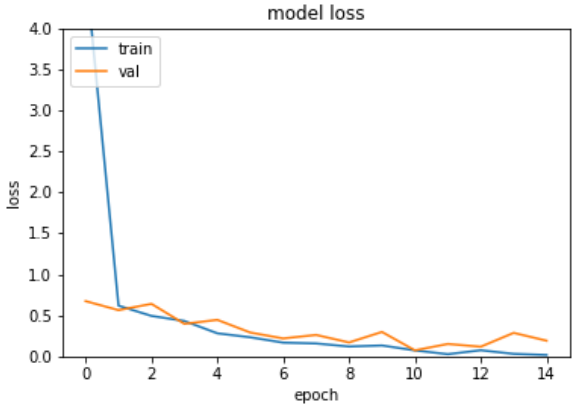
The first step is to use graphic modelling software to properly represent the scenario. One of the most well-known and widely used programmes to do this is Blender [16]. This software product also has a significant community that has created a number of tutorials that may be helpful to individuals who are just getting started in the realm of graphic modelling. After you have received the model, it will be necessary to import it into Unity3D. It will be sufficient to place the model we produced into the

[16] https://www.blender.org/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1



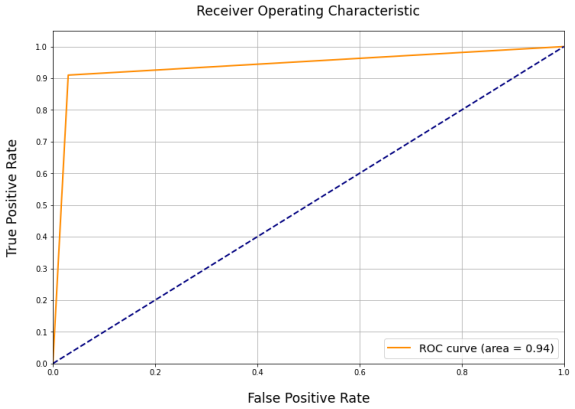
(a) Accuracy function



(b) Loss function

Test Set			
TARGET / OUTPUT	Baroque	Impressionism	SUM
Baroque	97 48.5%	3 1.5%	100 97% 3%
Impressionism	9 4.5%	91 45.5%	100 91% 9%
SUM	106 91.5% 8.5%	94 96.8% 3.2%	188 / 200 94% 6%

(c) Confusion Matrix Paintings Test set



(d) Receiver Operating Characteristic Curve

Figure 12. Confusion matrix and statistical analysis of the Paintings dataset analysed on Custom-CNN with Google Colab

gameobject after constructing an empty gameobject to use as a container. As a result, a camera must be placed in the scene and positioned so that it frames precisely the scene, as illustrated in Figure 13.

Depending on the lighting conditions we want to reproduce, we will also need a directional light and maybe one or more pointlights or spotlight types of lights. After you have completed these preparatory procedures, you will need to write a script that will change the lighting conditions at random, rotate the item, and take photos with the appropriate resolution. We propose including some public variables in the script so that you can define the parameters using the Unity3D graphical interface (the Inspector) rather than having to change the code all the time. Some examples of public variables are the path to the output image folder, the number of photographs to be taken, the object's and light's minimum and maximum rotations, the light's minimum and maximum intensities, and so on. Figure 14a shows an example. The operations flow that we suggest is as follows. Execute the following procedures within a loop that iterates a number of times equal to the number of photos we wish to generate:

1. Random rotation of the object: we produce three random integers that are contained in the minimum and maximum rotation ranges that we previously established and assign them as coefficients of the object's X, Y, and Z rotations.
2. Changing the scene's global lighting: we produce three random integers that represent the potential rotations of the light in the scene; it is critical to establish the ranges of the three variables accurately to guarantee that the representation obtained is believable. A picture with illumination from the bottom up, or even directly into the viewer's eyes, for example, would

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

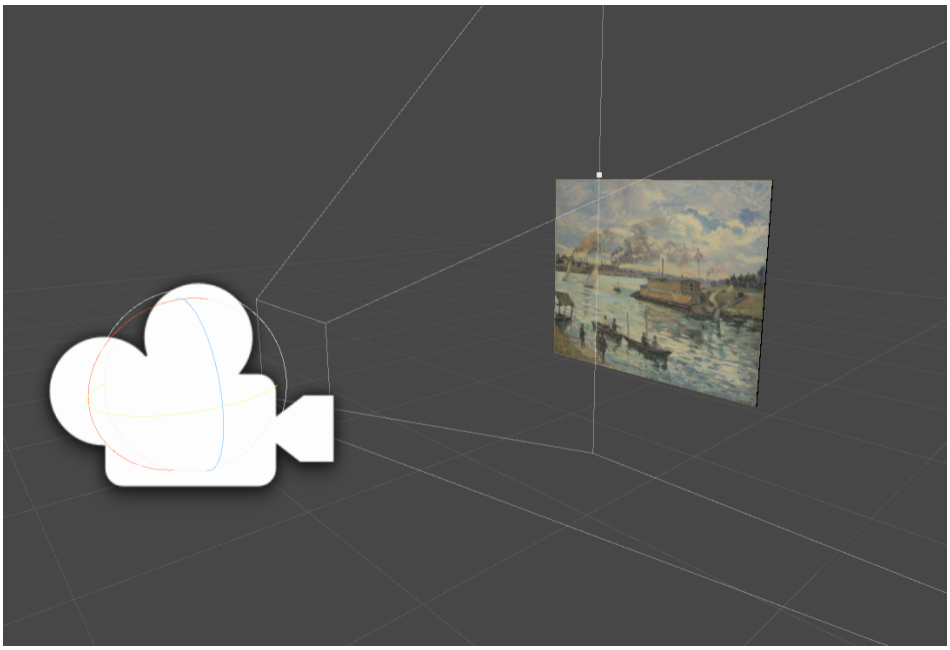
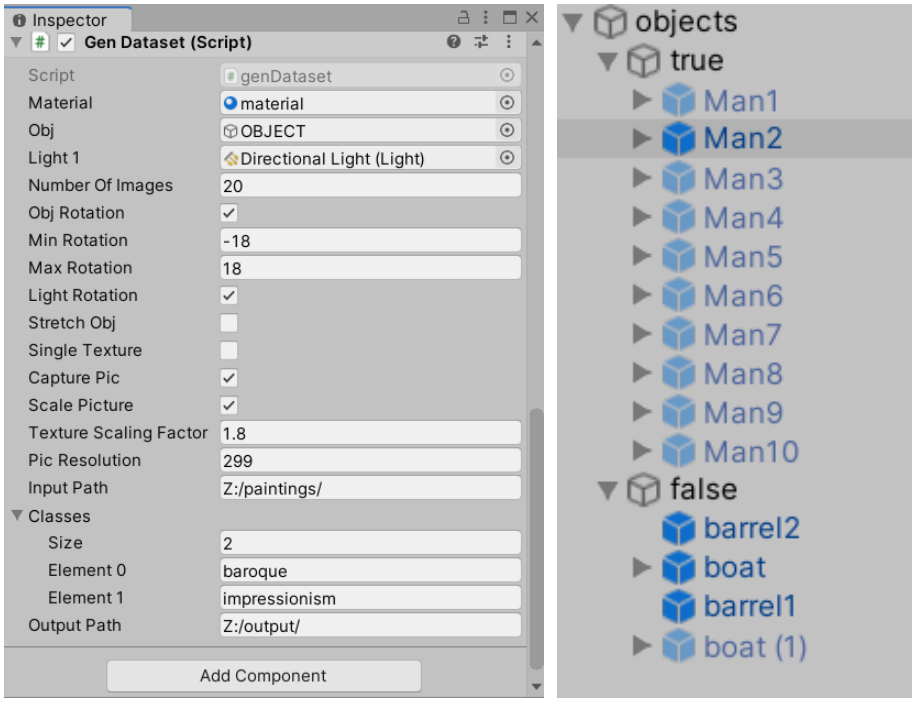


Figure 13. Camera that precisely frames the scene.



(a) Inspector, the script and the parameters **(b)** Objects in the sea

Figure 14. Unity3D interface

be unusual. As a result, we recommend trying until you get the appropriate outcome. These numbers are then allocated as light rotation coefficients once they have been formed. It is also possible to produce a random value that alters the light's intensity and hue.

3. Acquisition of the image: to store a photograph of the scene, generate a rectangle that overlaps the user interface starting at the x_0, y_0 coordinates of (0,0) and finishing at the x, y coordinates of (Screen.width, Screen.height), then extract the RGB values of the pixels included inside the rectangle.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 November 2021 doi:10.20944/preprints202111.0446.v1

4. Scaling the image: after we have gotten the RGB values, we will need to scale the image to fit the size requirements of the neural network we are going to utilise. For example, if we are creating a dataset to train the InceptionResNetV2 network, we may scale the photos directly in Unity3D to 299x299 resolution.
5. Saving the image: once the pixels have been captured and resized, the image must be saved to a file system, for example, in the PNG format. During the saving step, we propose distinguishing the objects using an identifying name and an incremental integer that will be used to create the saved file's name, such as `"impressionism_0000X.png"`.

5.2. Dataset for the classification of images representing objects

The followings are best practices for generating virtual scenario datasets using Unity3D. Now let us assume we want to do a true-false binary classification. First and foremost, the models of the things that will make up the scene must be created. The models must be realistic, and we suggest Blender in this situation as well. It is also important to consider how to construct the examples for the *true* and *false* classes. We must also ensure that the items are not always arranged in the same area of the image; to do so, we need to designate locations where they can be formed and then displayed to the camera. We created a scenario suitable for both photographs associated with the *true* category and photographs associated with the *false* category, in which we represented people at sea, shipwrecked people, and created a scenario suitable for both photographs associated with the *true* category and photographs associated with the *false* category. The images in the *true* category account for half of the whole dataset. The remaining 50% are in the *false* category, which is split into two sub-categories, with half of them representing simply the scenario with the sea and random light conditions. The other half is a representation of the sea, with ships and barrels floating on the surface. First of all, we recommend generating and putting a container object, in our case *"objects"*, into the scene. We have added two more containers to it, one for objects of the *true* class and the other for objects of the *false* class. Three-dimensional models of people, ships, and barrels have been placed within the two containers. Figure 14b provides an example.

As a result, the operations flow that we suggest is as follows. The following procedures should be executed within a loop that iterates a number of times equal to the number of images we wish to generate:

1. Restore the scene to its original state by hiding all items on the scene except the sea, the camera, and the sunlight at the start of each new iteration.
2. Make a random number of the objects active (and so display them within the scene) while producing pictures of the class *true/false*.
3. Alter the location of the objects: for each object in the scene, we generate three random numbers (X, Y, and Z) from the range of coordinates that the camera can frame and update the position of this object to the produced coordinates.
4. Change the rotation of objects: for each object in the scene, we produce three random integers (X, Y, and Z) that are within the desired rotation interval and realistically match the class to be formed. Then, using the provided values, rotate the item under examination.
5. Modify the lighting of the scene as described in point 2 in the previous list.
6. Acquire the scene image as specified in the preceding list's point 3.
7. Resize the image to the size acceptable by the neural network we wish to test, as explained in the preceding list's point 4.
8. Save the picture to the file system as stated in point 5 from the preceding list, be sure to give each image a name that allows you to identify the class to which it belongs.

6. Conclusions and future works

We have demonstrated that synthetic datasets may be a valuable resource for researchers utilising machine learning algorithms to identify objects or scenarios. When dealing with challenges of this

nature, you often have two options: either use datasets that other scientist have made accessible on the internet, or invest time and money in constructing an ad-hoc dataset specific to the topic at hand. It takes time and money to collect materials and images to create a dataset, which generally comprises tens of thousands of shots. Taking the photos required to train a neural network might be risky in some circumstances. We can naturally think, for example, of the training of the network of people at sea, the recognition of animals or pedestrians along the motorways for the automatic braking systems of cars, and so on. The synthetic datasets created by the pipeline and the methodologies presented in this article allow us to accelerate this process and predict which kinds of images will perform better for the task at hand. We also believe that, as a result of the high degrees of realism achieved by computer graphics, the image quality is quite good and will continue to improve, allowing the construction of increasingly realistic datasets. We are going to expand our study in the future and concentrate on particular elements, such as the union of synthetic and realistic datasets, by examining how neural networks trained on synthetic datasets react while adding instances to the original dataset and utilising continuous learning approaches.

7. Author contributions

Conceptualization, Damiano Perri, Marco Simonetti and Osvaldo Gervasi; Data curation, Damiano Perri, Marco Simonetti and Osvaldo Gervasi; Investigation, Damiano Perri, Marco Simonetti and Osvaldo Gervasi; Methodology, Damiano Perri, Marco Simonetti and Osvaldo Gervasi; Software, Damiano Perri, Marco Simonetti; Supervision, Osvaldo Gervasi; Validation, Damiano Perri, Marco Simonetti; Writing – original draft, Damiano Perri, Marco Simonetti, and Osvaldo Gervasi; Writing – review & editing, Damiano Perri, Marco Simonetti, and Osvaldo Gervasi.

Supplementary Materials:
The open source code for the generation of the confusion matrices developed for this paper is available on GitHub https://github.com/DamianoP/confusionMatrixGenerator

Funding: "This research received no external funding"

Acknowledgments:

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
RAN	Recurrent Adversarial Network
UV	Represent the u,v graphic coordinates
VR	Virtual Reality
ROC	Receiver Operating Characteristic
NN	Neural Network

References

1. Wu, F.Y. Remote sensing image processing based on multi-scale geometric transformation algorithm. *Journal of Discrete Mathematical Sciences and Cryptography* **2017**, *20*, 309–321.
2. Wolberg, G. Geometric transformation techniques for digital images: a survey. *Columbia University Computer Science Technical Reports* **1988**. doi:10.7916/D8TH8VRW.
3. Arce-Santana, E.R.; Alba, A. Image registration using Markov random coefficient and geometric transformation fields. *Pattern Recognition* **2009**, *42*, 1660–1671.
4. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* **2017**.

5. Ekstrom, M.P. *Digital image processing techniques*; Vol. 2, Academic Press, 2012.
6. Kwak, H.; Zhang, B.T. Generating images part by part with composite generative adversarial networks. *arXiv preprint arXiv:1607.05387* **2016**.
7. Im, D.J.; Kim, C.D.; Jiang, H.; Memisevic, R. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110* **2016**.
8. Zhang, C.; Feng, Y.; Qiang, B.; Shang, J. Wasserstein generative recurrent adversarial networks for image generating. 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, 2018, pp. 242–247.
9. Jiwoong Im, D.; Dongjoo Kim, C.; Jiang, H.; Memisevic, R. Generating images with recurrent adversarial networks. *arXiv e-prints* **2016**, pp. arXiv–1602.
10. Frid-Adar, M.; Klang, E.; Amitai, M.; Goldberger, J.; Greenspan, H. Synthetic data augmentation using GAN for improved liver lesion classification. 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018). IEEE, 2018, pp. 289–293.
11. Santucci, F.; Frenguelli, F.; De Angelis, A.; Cuccaro, I.; Perri, D.; Simonetti, M. An immersive open source environment using godot. International Conference on Computational Science and Its Applications. Springer, 2020, pp. 784–798.
12. Benedetti, P.; Perri, D.; Simonetti, M.; Gervasi, O.; Reali, G.; Femminella, M. Skin Cancer Classification Using Inception Network and Transfer Learning. International Conference on Computational Science and Its Applications. Springer, 2020, pp. 536–545.
13. Biondi, G.; Franzoni, V.; Gervasi, O.; Perri, D. An Approach for Improving Automatic Mouth Emotion Recognition. Computational Science and Its Applications – ICCSA 2019; Misra, S.; Gervasi, O.; Murgante, B.; Stankova, E.; Korkhov, V.; Torre, C.; Rocha, A.M.A.; Taniar, D.; Apduhan, B.O.; Tarantino, E., Eds.; Springer International Publishing: Cham, 2019; pp. 649–664.
14. Batista, G.E.; Prati, R.C.; Monard, M.C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter* **2004**, *6*, 20–29.
15. Buda, M.; Maki, A.; Mazurowski, M.A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* **2018**, *106*, 249–259.
16. Bellinger, C.; Corizzo, R.; Japkowicz, N. Remix: Calibrated resampling for class imbalance in deep learning. *arXiv preprint arXiv:2012.02312* **2020**.
17. Aggarwal, C.C. An introduction to outlier analysis. In *Outlier analysis*; Springer, 2017; pp. 1–34.
18. Kubat, M.; Holte, R.; Matwin, S. Learning when negative examples abound. European Conference on Machine Learning. Springer, 1997, pp. 146–153.
19. Kubat, M.; Matwin, S.; others. Addressing the curse of imbalanced training sets: one-sided selection. Icml. Citeseer, 1997, Vol. 97, pp. 179–186.
20. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* **2009**, *21*, 1263–1284.
21. Liu, X.Y.; Wu, J.; Zhou, Z.H. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **2008**, *39*, 539–550.
22. Chawla, N.V.; Lazarevic, A.; Hall, L.O.; Bowyer, K.W. SMOTEBoost: Improving prediction of the minority class in boosting. European conference on principles of data mining and knowledge discovery. Springer, 2003, pp. 107–119.
23. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* **2002**, *16*, 321–357.
24. Zhou, Z.H.; Liu, X.Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering* **2005**, *18*, 63–77.
25. Ting, K.M. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering* **2002**, *14*, 659–665.
26. Sun, Y.; Kamel, M.S.; Wong, A.K.; Wang, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern recognition* **2007**, *40*, 3358–3378.
27. Perri, D.; Simonetti, M.; Lombardi, A.; Faginas-Lago, N.; Gervasi, O. Binary classification of proteins by a machine learning approach. International Conference on Computational Science and Its Applications. Springer, 2020, pp. 549–558.

28. Perri, D.; Simonetti, M.; Lombardi, A.; Faginas-Lago, N.; Gervasi, O. A new method for binary classification of proteins with Machine Learning. International Conference on Computational Science and Its Applications. Springer, 2021, pp. 388–397.
29. Meyer, G.W. Wavelength selection for synthetic image generation. *Computer Vision, Graphics, and Image Processing* **1988**, *41*, 57–79.
30. Spindler, A.; Geach, J.E.; Smith, M.J. AstroVaDEr: astronomical variational deep embedder for unsupervised morphological classification of galaxies and synthetic image generation. *Monthly Notices of the Royal Astronomical Society* **2021**, *502*, 985–1007.
31. Perri, D.; Simonetti, M.; Tasso, S.; Gervasi, O. Learning Mathematics in an Immersive Way. In *Software Usability*; IntechOpen, 2021.
32. Prokopenko, D.; Stadelmann, J.V.; Schulz, H.; Renisch, S.; Dylov, D.V. Unpaired synthetic image generation in radiology using gans. Workshop on Artificial Intelligence in Radiation Therapy. Springer, 2019, pp. 94–101.
33. Kuo, S.D.; Schott, J.R.; Chang, C.Y. Synthetic image generation of chemical plumes for hyperspectral applications. *Optical Engineering* **2000**, *39*, 1047–1056.
34. Simonetti, M.; Perri, D.; Amato, N.; Gervasi, O. Teaching math with the help of virtual reality. International Conference on Computational Science and Its Applications. Springer, 2020, pp. 799–809.
35. Svoboda, D.; Ulman, V. Generation of synthetic image datasets for time-lapse fluorescence microscopy. International Conference Image Analysis and Recognition. Springer, 2012, pp. 473–482.
36. Borkman, S.; Crespi, A.; Dhakad, S.; Ganguly, S.; Hogins, J.; Jhang, Y.C.; Kamalzadeh, M.; Li, B.; Leal, S.; Parisi, P.; others. Unity Perception: Generate Synthetic Data for Computer Vision. *arXiv preprint arXiv:2107.04259* **2021**.
37. Al-Masni, M.A.; Al-Antari, M.A.; Park, J.M.; Gi, G.; Kim, T.Y.; Rivera, P.; Valarezo, E.; Choi, M.T.; Han, S.M.; Kim, T.S. Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system. *Computer methods and programs in biomedicine* **2018**, *157*, 85–94.
38. Lan, W.; Dang, J.; Wang, Y.; Wang, S. Pedestrian detection based on YOLO network model. 2018 IEEE international conference on mechatronics and automation (ICMA). IEEE, 2018, pp. 1547–1551.
39. Laroca, R.; Severo, E.; Zanlorensi, L.A.; Oliveira, L.S.; Gonçalves, G.R.; Schwartz, W.R.; Menotti, D. A robust real-time automatic license plate recognition based on the YOLO detector. 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018, pp. 1–10.
40. Wang, W.; Li, Y.; Luo, X.; Xie, S. Ocean image data augmentation in the USV virtual training scene. *Big Earth Data* **2020**, *4*, 451–463.
41. Yun, K.; Yu, K.; Osborne, J.; Eldin, S.; Nguyen, L.; Huyen, A.; Lu, T. Improved visible to IR image transformation using synthetic data augmentation with cycle-consistent adversarial networks. Pattern Recognition and Tracking XXX. International Society for Optics and Photonics, 2019, Vol. 10995, p. 1099502.
42. Lu, T.; Huyen, A.; Nguyen, L.; Osborne, J.; Eldin, S.; Yun, K. Optimized training of deep neural network for image analysis using synthetic objects and augmented reality. Pattern Recognition and Tracking XXX. International Society for Optics and Photonics, 2019, Vol. 10995, p. 109950I.
43. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1; Curran Associates Inc.: Red Hook, NY, USA, 2012; NIPS'12, p. 1097–1105.
44. Szegedy, C.; Ioffe, S.; Vanhoucke, V. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR* **2016**, *abs/1602.07261*, [1602.07261].
45. Torrey, L.; Shavlik, J. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*; IGI global, 2010; pp. 242–264.
46. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.
47. Sharma, S.; Sharma, S.; Athaiya, A. Activation functions in neural networks. *towards data science* **2017**, *6*, 310–316.
48. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization, 2017, [arXiv:cs.LG/1412.6980].