Synthetic Imagery used with Transfer Learning

John Wisnewski ppwxmc@missouri.edi
University of Missouri

– ***Abstract***
When researchers are faced with limited data there are a few approaches to handling this. With the rise of 3D modeling software one approach is creating synthetic images with open source software like Blender and Unreal Engine. These softwares allow for complete freedom and control for creating images and scenes to replicate the dataset. With the potential of unlimited high quality data a transfer learning technique based around a Convolutional Neural Network can be a powerful approach to image classification. Transfer Learning allows for constant improvements and robustness while a CNN is well suited for handling image data.

I. Introduction

The scarcity of real, reliable data is a problem that many researchers across domains face. It can be costly, dangerous, or even impossible to obtain data from real world scenarios. Machine learning algorithms and pipelines require a vast amount of high quality data in order to be effective. In this paper we will be focusing on image data that is hard to obtain in the real world. In order to supply these algorithms with meaningful data a few approaches can be taken. Data augmentation and processing techniques can help researchers create more usable data without collecting more. These techniques are widely used and can provide algorithms with more diverse data. Another technique is creating new images by utilizing 3D modeling software to create synthetic photorealistic images. This is done by using free open source software and replicating reference images. This added data will help fuel the machine learning algorithms and be integral to implementing a transfer learning approach.

II. Problem Definition

It is important to have reliable and high quality data when using machine learning models. The main problem that this project is aiming to solve is increasing the amount of usable high quality data. The approach that is being used is to create synthetic images using open source software like Blender and Unreal engine. The reason why we need to create a synthetic data set is because of limited real world examples due to various constraints that make it costly and difficult to obtain the data.

Once the data problem is solved the next step is to export the images to be used in a machine learning algorithm. Because we are dealing primarily with image data a convolutional neural network was chosen to be the baseline for examination. Then transfer learning techniques are utilized to help facilitate a more robust and precise model. Transfer learning was chosen because new data sets can be made constantly so it could be advantageous to transfer the knowledge of one model to another being trained on a different but similar dataset.

III. Synthetic Image Creation Pipeline

Synthetic imagery is crucial for this problem and use case because it is very difficult to gather vast amounts of real world data. The way to combat this is to create new synthetic data that highly resembles the few examples of real data. In order to create synthetic images it is necessary to have examples of what you are trying to replicate. This dataset contained a small amount of examples to be used as reference. From this starting point it was up to creativity and trial and error to create the necessary objects in Blender and Unreal Engine.

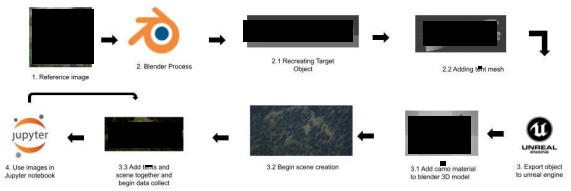

Fig. 1 Data flow diagram

Figure 1 showcases the workflow process. We begin with a reference image shown in the top left corner. The next step is to use Blender to create the 3D mesh that resembles the target object. After the mesh is created it is exported into Unreal Engine. Then in Unreal Engine the meshes get materials applied and are scaled to be the correct size. After the models are finished the scene needs to be created. Again the reference image is used to create the scene and assets from the Unreal Engine marketplace are used. Once the scene is created the 3D mesh is inserted into the scene and the data collection process begins. After data collection the images are used in machine learning algorithms.

The first thing that needed to be done was to create a 3D model of the target object. This was necessary because there were not any existing available 3D models of this object. The object could not have been scanned or otherwise imported into the software so creating a 3D model from scratch was the best approach. Examples are shown on the next page. Multiple 3D models were created with slight

variations which will be shown in the next figure. After these objects were created they could be exported and used in other softwares like Unreal Engine. Figure 2 and 3 show the skeletal frame of the target object and then the 3D model once a tent mesh is applied.

Fig. 2 skeleton frame for object mesh

Fig. 3 3D mesh with [redacted]p applied

Once the 3D models are created they are then imported into Unreal Engine. When the models are imported into Unreal Engine they can be used and altered to generate synthetic scenes. The initial step after importing to UE is to scale the models to the UE scale. This is crucial because the target objects must resemble the real life examples as closely as possible. Once scaled in UE the objects need to be altered to have the same appearance as the reference images. This was done by altering the materials used on the object to replicate the outer appearance of examples found in real world scenarios. The figure below shows five variations of the tents and a few examples of the different materials and patterns applied.

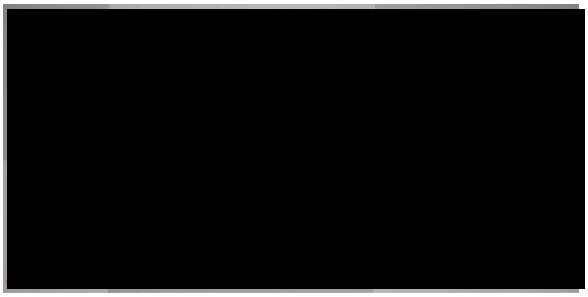

Fig. 4 examples of target object and variations in UE

This also adds a level of variation because the materials can vary in color and pattern along with having multiple styles. In this project there were five variations of the model, with thirty color and pattern variations that could be applied to each model variant. After altering the appearance of the models, a physical scene needed to be created next. This was done by importing free and paid assets from the UE marketplace. The UE marketplace has thousands of 3D models with accompanying materials. Along with the assets that are included in the project there was sufficient materials and objects to create a photorealistic scene.

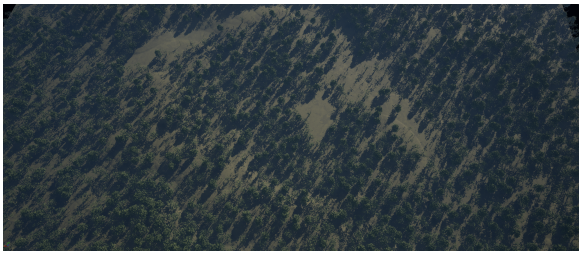

Fig. 5 example of scene creation in UE

The figure above shows an example of the scenes created in Unreal Engine. By using the reference images and background information on the location of where the target objects are located the selection of assets to be used in the scene was narrowed down to the ones most similar or likely to be in the environment. Some of the objects that were included to resemble the reference image were trees, ground cover, ground patterns, and basic terrain manipulation. These objects were randomly placed into the scene using various built in techniques to further increase the variance of the images to provide a broader dataset. The figure below shows the combination of 3D models and scenery created in UE.

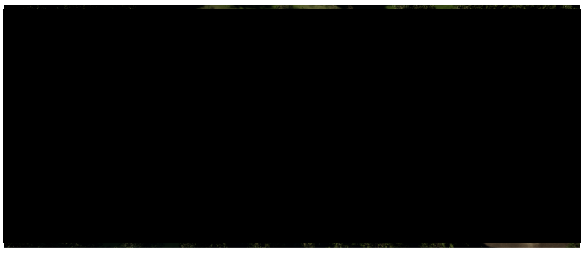

Fig. 6 UE scene with target objects

Once the physical portion of the scene was created the next step was to import lighting and shadows. Shadows were a very important and difficult aspect to this approach. When UE updated to 5.0 and later a new lighting technique called Nanite and Ray Tracing was supported and allowed for

better control and performance for generating lighting and shadows. The lighting and shadows needed to be photorealistic along with the physical part of the scenes which resulted in computational expensive renders. This was essential in generating the scenes and also added necessary variation of the scene itself by affecting shadows and colors of the scene.

The synthetic images were collected by capturing high resolution screenshots of the 3D scenes. During these captures the lighting was changing to add variation to the shadows in the scene. This way an automated process could gather images while continuously changing the lighting variable. This was done to simulate different times of the day which would cast different shadows based upon the angle of the sun. These images were then exported to another local folder to be used in the image classification pipelines.

IV. Experiments and Results

The experiment is set up to use a convolutional neural net to classify images. The pipeline to feed data into this CNN is as follows: 3D model creation, import 3D assets into UE, collect data, run initial model on data, augment data set, load trained model and repeat. This pipeline allows for constant improvements and iterations of models to help improve model performance. The purpose of this project is to provide a structured approach for using machine learning on synthetic imagery. Further modifications to the CNN model and machine learning approach are necessary to further improve the pipeline. The results in this project are to serve as a baseline for further improvements and establish a framework for future development.

The performance of the pipeline can be viewed as promising. The models perform better than randomly classifying and can classify vast amounts of images quickly without using heavy computational resources.The most important factor is the speed that the model provides. When classifying these images it would take a human much longer to carefully scan the images and look for the target object. This is also strenuous and tedious for a human to complete which can lead to poor results. A human will also cost money and require breaks and accommodations. This pipeline's main goal and objective is to provide a framework to automate and facilitate image creation and image classification.

The structure of the CNN consists of an input layer designed for 64x64 pixel RGB images, followed by three convolutional layers with increasing filter sizes (32, 64, 128) and (3, 3) kernels, each paired with a max pooling layer to reduce spatial dimensions and control overfitting. The network then flattens the output and passes it through a dense layer with 512 neurons, all using ReLU activation. The final output layer employs a sigmoid function, suitable for binary classification. Compiled with the Adam optimizer and binary cross-entropy loss, the model is trained and validated on separate data generators for 30 epochs, making it a well-rounded solution for tasks like distinguishing between two classes in image data.

The CNN is trained on a dataset containing 2,532 augmented images. The augmented images contain various levels of noise, pixel occlusion and translation to add variation to the dataset. The CNN was trained for thirty epochs before being evaluated. The image below shows the folder of images used for training. This folder was composed entirely of synthetic images. The synthetic images received various augmentations to further increase the dataset size and an example of those images is shown below.

Fig. 7 Training dataset

```
Accuracy: 0.73
Confusion Matrix:
[[25 18]
 [25 90]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.58      0.54        43
           1       0.83      0.78      0.81       115

    accuracy                           0.73       158
   macro avg       0.67      0.68      0.67       158
weighted avg       0.74      0.73      0.73       158
```

Fig. 8 Initial confusion matrix and classification report

The figure above is the validation results of the model testing against full synthetic images. This was done because the model was trained on synthetic images and it is important to see how the model performs on more synthetic images. The figure below shows the confusion matrix when the model sees real images it has not seen before.

The image below shows these real world validation images. These images were used to replicate real world use cases.

Fig. 9 Validation dataset

The figure below shows the second confusion matrix and classification report of the real world data examples.. Along with the class imbalance there were duplicates of the images but with varying contrast to add variety to the data set.

```
Accuracy: 0.68
Confusion Matrix:
[[  8  35]
 [ 19 106]]
Classification Report:
              precision    recall  f1-score   support

           0       0.30      0.19      0.23        43
           1       0.75      0.85      0.80       125

    accuracy                           0.68       168
   macro avg       0.52      0.52      0.51       168
weighted avg       0.64      0.68      0.65       168
```

Fig. 10 Validation confusion matrix and classification report

This examination is limited but it does show promise for the pipeline and approach. There was high recall for the class containing the object which indicates it is able to recognize when the object is in the image. Opposite to this point the recall for the no object class was poor indicating it is not recognizing when the object is not in the image. This can be an indication for overfitting or some other issue in the training data and algorithm process. Even with these shortfalls the possibilities are evident. Better models exist and can be put in place for this very simple and standard CNN. Along with more synthetic and real data this pipeline can show promising results.

V: Discussion and Conclusion

In conclusion there are very promising results after these experiments. The main goal of establishing a framework and pipeline was achieved but there is a lot of room for improvements and further development.

This project's goal was to showcase a pipeline for transfer learning with synthetic imagery. The key components of the pipeline include 3D model creation, photorealistic scene creation, data collection, data augmentation, implementation of a machine learning algorithm and repeating these steps as necessary. This goal was achieved by creating models in Blender, importing into Unreal Engine, exporting those images and utilizing those images in a Python Jupyter Notebook to conduct machine learning. This pipeline has independent processes that can be fine tuned and expanded on. For example if you were to add more models into Unreal Engine the machine learning aspects could remain unchanged. The same is true if the machine learning methodology was changed you would not have to redo your image collections. By utilizing data augmentation methods like adding noise you can further increase your dataset of synthetic imagery. Those aspects of the project were completed and the structure of the pipeline allows for constant development.

The next areas of future direction will be to complete a human out of the loop data collection process. The current process still requires a human to control and oversee the processes.By removing the human and going 'headless' it is possible to further automate the pipeline. This would allow for more image creation. By taking the human out of the loop and providing checks in the algorithm to quit training if accuracy reaches a certain point it is possible to have semi automated data generation and training.