Computer Vision for Quality Assurance in Procedural Generation in Visual Mediums.

Cyrus Thompson
Northeastern University
thompson.cy@northeastern.edu

Abstract

Procedural generation is a tried and tested approach to content creation in visual mediums for good reason, as generating content automatically has the ability to scale aribitarly for very little cost. Yet the downside is that the content produced is variable, and thus has variable quality. The failures of complex procedural generation systems are often difficult to put in terms of generative constraints but can often be identified easily by a human. The difficulty is that having a human go through and review the content is not always a scalable approach. Thus within this paper we will do a novel exploration of replicating the process of manual human review for quality assurance of procedural generation in visual mediums through computer vision. We will show using a case study of the widely popular game Minecraft that procedural generation in visual mediums by its nature can produce large labeled image datasets with minimal human oversight and models trained on these datasets can be used to do highly scalable and accurate quality checking for procedurally generated content.

Background

Visual content is in high demand and in almost every form is very data dense. This can be seen in the fact that 79% of all internet traffic is video traffic[2]. Thus naturally to satisfy demand content producers in the visual medium have reliably returned to procedural generation to automatically generate content. This can be seen in television shows such as “The Mandalorian” where the landscapes were procedurally generated or movies such as “Lord of the Rings Return of The King” where many of the crowd visual effects were procedural generated, or in video games such as “Minecraft” where the infinite interactable world is procedurally generated. Simply generating content automatically rather than hand crafting is much less costly at scale. Yet inevitably if the content is non-repetitive the procedural generation will produce errors. Fixing these errors on a generative side not only requires an understanding of what produced the error but also an understanding of what can be changed without causing errors somewhere else. Yet if the quality of the content output by a procedural generation system is generally good and a human review finds any particular content to be degenerate in any way they can simply rerun the procedural generation and get new content. The only part of this hypothetical content pipeline that is not scalable is the manual human review of the content. Thus this paper's primary contribution is the exploration of automating that task with computer vision.

Specifically this paper will demonstrate that data generation in visual mediums is a tractable approach and a computer vision machine learning framework that consumes the visual medium and outputs a sense of how good the generation is not only a tractable approach but is effective. For this paper we will specifically be using the game Minecraft for demonstrating this. The reason for this choice is the range of the procedural generation, Minecraft generates a infinite 3d world, the popularity, with Minecraft being the best-selling videogame ever[1], the reproducibility, with all procedural generation in Minecraft being easily reproduced with an associated seeded/version, and for the importance of procedural generation, with all the player interactions being derived from the procedurally generated world.

Data-

An advantage of this approach is that adding this type of automated quality assurance does not require a large change from an existing data-pipeline. Procedural generation in visual mediums output visual representations which can be represented as images. The existing consumer of the visual medium be this development staff or the audience are already receiving the data and are implicitly considering if the quality of the procedural generation is good or bad thus by capturing just a small percentage of this implicit labeling on the data will result in more than enough data that can be fed into a computer vision model to train it and then in turn to classify other later images improving the quality of the procedural generation overall .

Data Definitions and Motivations-

The goal of the framework created in this paper is the ability to consume the visual medium and output labels.

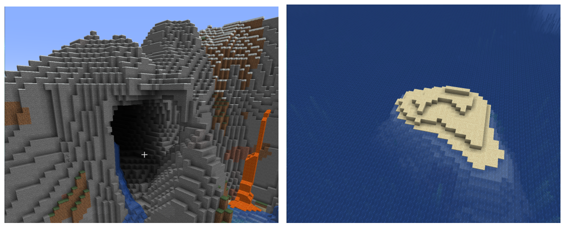


Figure 1. Great and Ok examples

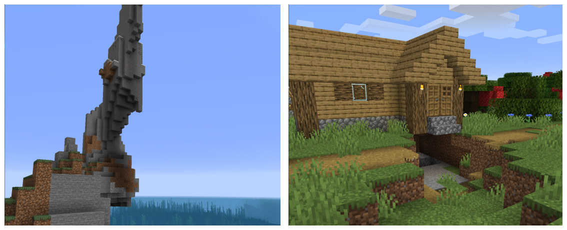


Figure 2. Bad and Degenerate examples

Thus the data must be the images that a user consuming a visual medium would see. So within this paper the data that is used are screenshots from the game Minecraft. Because of the novelty of this problem there are no precedents or benchmarks to derive labels from, thus we decided to have four base labels “Great”, “Ok” and “Bad”, “Degenerate”. “Great” an example of which can be seen in Figure 1 meaning an exceptionally good piece of procedural generated content such as a well formed visually exciting mountain. “Ok” meaning an unexceptional but still acceptable piece of content generation such as a simple wide empty plain.“Bad” an example of which can be seen in Figure 2 meaning a poor but not game disrupting procedural generated content such as an out of place odd looking stone spire. “Degenerate” an example of which can also be seen in Figure 2 meaning a procedural generated content that disrupts gameplay and world believability such as a structure that is unusable because it is merged with the terrain. These four labels are in some cases mapped up into 2 different sets of Binary Labels at training time. {Good, Bad} where Great and Ok are mapped to Good and Bad and Degenerate are mapped to bad. {Not_Degenerate, Degenerate} where Degenerate is mapped to Degenerate and all other labels are mapped to Not_Degenerate. The main motivation for these multi tiered labels was the ability to analyze and report on the benefits of increased label granularity in comparison to accuracy, allowing future work to understand the trade off within this problem space.

Data Generation and Reproducibility-

The methodology for data generation had one primary goal, reproducibility. Because we were generating the data through taking screenshots in Minecraft, to allow for reproducibility we labeled all images generated with Minecraft version number, seed number, time of day, minecraft in game biome, a overarching generation type which is one of ["Structure", "OpenLandscape", "UndergroundLandscape"] and X,Z coordinates of the generation being photographed on top of the multi tier label mentioned in the previous section. With this information any piece of image could be regenerated with ease, thus allowing for future work to explore the dataset produced by this paper in more detail. The decision on what to take screenshots of was one of weighing potential biases. In general the methodology was to try to collect items from the different sub labels in the same vicinity, and with the same composition of terrain. Then simply take a large sample size. All data generation was done by hand within this project.

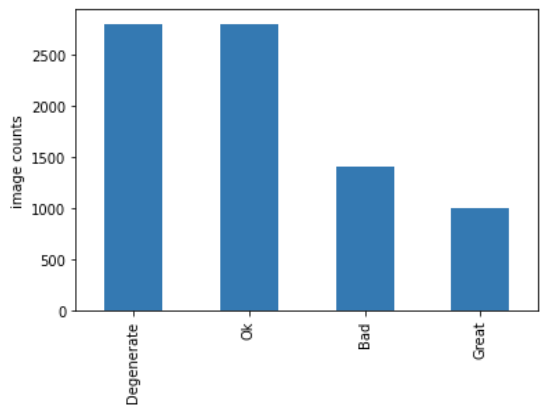


Figure 3. Label Distribution

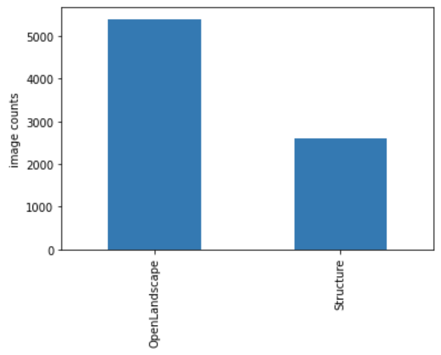


Figure 3.Generation Type Distribution

Our Data-

After a long data generation timeline we produced 8000 different hand labeled images all generated from a single Minecraft seeded world. This data and it’s labels are open source and can be accessed online by anyone. In terms of the data itself we had a fairly good distribution, that is a bit skewed towards “Degenerate” and “Ok” labels. The reason for this skew is that it was easy to find content that was objectively degenerate and objectively ok but it was not easy to find content that was objectively great or objectively bad. In terms of the type of content we were collected from we ended up collecting twice as much from open landscapes than structures simply because of their lack of occurrence of structures.

Information Bleed-

The number one issue we faced with this approach to data generation was information bleed between training, validation and test sets. This is because if two images of the same item from different angles and distances get into the training and the test sets this results in a much higher accuracy and the test sets but this accuracy is not generalizable to other data. Thus the first thing we did when creating training, validation, test split was make sure no image of the same item gets into more than one set by comparing reproducibility labels.

Methods-

This paper takes a breadth instead of depth approach to which computer vision methods are used and how to train the hyper parameters. The main reason for this is the lack of existing work on the topic of this paper. Thus considering the topic is novel, understanding how effective the wide array of existing computer vision approaches are would be useful for any further work on this topic or for anyone considering the work done in this paper before creating a parallel application of their own. The basic approaches which shall be the baseline of effectiveness are the classic simple models found in Scikit Learn specifically Logistic regression and Support Vector Machines. After this baseline has been established we will first use custom convolutional neural nets of various depths trained from scratch. Then using different cutting edge pre trained models we performed transfer learning using our data, specifically we used different generations of resnet for this resnet18, wide resnet50_2 and wide resnet101_2.

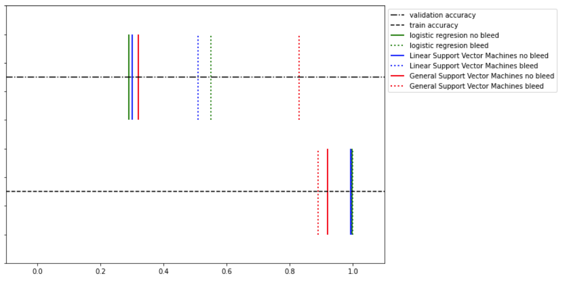
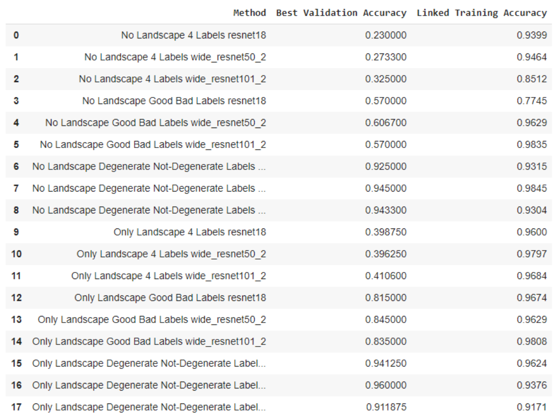


Figure 4.Information bleed effect on validation and training accuracy for the more classic computer vision methods

We trained the above models while varying model specific hyperparameters, varying over the three different label sets and varying over the two different generation types. Which when ran on a google collab machine, with a gpu Accelerator when needed, took over a week in total computational time to complete.

Results-

After an extensive batch of training/testing/validating on the myriad number of computer vision models, a few clear results have surfaced. A preface for the following discussion: we will only be considering the models in the context of having the best possible hyper parameters as we found in training in testing. The first of which is the fact that the data bleed mentioned in a previous section had a substantial impact on validation accuracy as seen in figure 4. Secondly as can be seen in figure 4 the classic methods of computer vision namely Logistic Regression, Linear Support Vector Machines and General Support Vector Machines simply did not perform very well in this problem environment. The biggest difference in terms accuracy caused by data separation was separating structures from open landscapes. By separating them every single model had an



	Method	Best Validation Accuracy	Linked Training Accuracy
0	No Landscape 4 Labels resnet18	0.230000	0.9399
1	No Landscape 4 Labels wide_resnet50_2	0.273300	0.9464
2	No Landscape 4 Labels wide_resnet101_2	0.325000	0.8512
3	No Landscape Good Bad Labels resnet18	0.570000	0.7745
4	No Landscape Good Bad Labels wide_resnet50_2	0.606700	0.9629
5	No Landscape Good Bad Labels wide_resnet101_2	0.570000	0.9835
6	No Landscape Degenerate Not-Degenerate Labels ...	0.925000	0.9315
7	No Landscape Degenerate Not-Degenerate Labels ...	0.945000	0.9845
8	No Landscape Degenerate Not-Degenerate Labels ...	0.943300	0.9304
9	Only Landscape 4 Labels resnet18	0.398750	0.9600
10	Only Landscape 4 Labels wide_resnet50_2	0.396250	0.9797
11	Only Landscape 4 Labels wide_resnet101_2	0.410600	0.9684
12	Only Landscape Good Bad Labels resnet18	0.815000	0.9674
13	Only Landscape Good Bad Labels wide_resnet50_2	0.845000	0.9629
14	Only Landscape Good Bad Labels wide_resnet101_2	0.835000	0.9808
15	Only Landscape Degenerate Not-Degenerate Label...	0.941250	0.9624
16	Only Landscape Degenerate Not-Degenerate Label...	0.960000	0.9376
17	Only Landscape Degenerate Not-Degenerate Label...	0.911875	0.9171

Figure 5 Transfer learning results after 24 epochs

increase in validation accuracy even though they were working with less data. In terms of this split they both got better accuracies in general across the board than when they were together but open landscape did better in a majority of cases. This could be because open landscapes had double the data or it could be because of the fact that what makes a structure bad is very tied to its use thus quality might be very abstract and hard to train a model to see. Next both the Good Bad and the Degenerate Non-Degenerate labels did better than the base 4 sub labels across the board in terms of accuracy and Degenerate Non-Degenerate did better than Good Bad in general. To confirm this we tried multiple different random train test validation splits but they almost all returned similar differences meaning that for our data it is simply easier to detect whether something is degenerate or not rather than good or bad, we think this is useful for the overall purpose of this paper because if you were doing this in a data pipeline and wanted to know if something needed to be rerun knowing if it is degenerate is useful. Besides this in terms of method transfer learning was the most effective in general especially in the binary case and within transfer learning the best model was in general the biggest - wide resnet101_2, but this was not always the case. We believe there are still a lot of other models that could use transfer learning on our dataset and might get different results so we suggest this as a direction of future research. While overall accuracy did leave a bit to be desired we consider the accuracies in the 80-95% range on a large portion of the binary transfer learning models seen in figure 5 to be a success. Though there were versions of the model that performed even higher on further exploration showed they almost always had a skewed train test validation split.

Conclusions-

Visual content is extremely influential in today's world and procedural generation creates a large portion of it. Considering that quality assurance is a basic part of any content's lifecycle and human quality assurance simply does not scale as well as procedural generation does, it should be self-evident that a tool that allows quality assurance to scale alongside procedural generation is needed. This paper is just a case study but should serve as a proof of concept to inspire further work in this field. To this end all code used to produce all the results above is open source and designed to be run in google collaboration by anyone. All the data that was generated and used in this paper is similarly open source and is available for any future projects. There is a lot of interesting work still to be done as this paper had a lot of logistical limitations but we showed that at some level computer vision based quality assurance for procedural generation in visual mediums is possible.

References-

1. Warren, Tom. "Minecraft still incredibly popular as sales top 200 million and 126 million play monthly".The Verge. Retrieved April 5, 2020.
2. “VNI Complete Forecast Highlights.” *cisco.com*, Cisco, 2020, https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf. Accessed 29 4 2021.