Data Pipeline Optimization for Deep Learning applications in Game development

Abderrahim Jami

BSc(Hons) Computer Game Applications Development

School of Design and Informatics

Abertay University

Table of Figures

Table of Tables

Acknowledgments

Special thanks to my supervisor Ruth Falconer for supporting and mentoring me during the undertaking of this Honours project.

Abstract

Artificial intelligence or AI for short has been at the centre of research for many years but only recently was it possible to see theories and hypothesis that were only on paper, materialize, thanks to the compute power available with modern computers. And with that came a new area of study called Deep Learning, many researchers are currently adopting DL techniques and neural networks in many experiments showing how impressive this technology can be. Companies like Microsoft for instance are pushing this resource to be used in games as well as games production and although it is still early to rely completely on deep learning and computer vision for big scale asset creation the industry is certainly moving in that direction.

This study is going to focus on a small sub section of this area which is called data pre-processing. Neural networks need semantic data to be able to retain unique characteristics and thus learn how to either classify and recognize data or to even synthetize it, attempts to achieve this have been made with the use of the Nvidia Spade model to create artificial 2d representations of Minecraft worlds, this dissertation is going to outline the attempts at streamlining the feature extraction from Minecraft worlds by automating certain phases which were previously done manually and then analysing the results to assess whether they are suitable for machine deep learning with the spade model.

Abbreviations, Symbols and Notations

….

Chapter 1

Introduction

Game content in games could be grouped in two distinct categories, one where it is manually crafted by a human being and the other being generated by a machine or an algorithm. Today's games often rely on the use of artists and game designers to author high quality assets, this has been the main source of content in games and qualitatively speaking it's usually preferred due to the ability of the human to discern certain patterns and unique characteristics and to transpose them in something new. Not only that but humans possess something that computer intrinsically lack which is creativity. This approach although dominant in this sector faces substantial limitations which come in the form of absorbent financial strains in the production costs due to additional professional figures required, and/or excessive development times. This consequently restricts human generated content to a subsection of production companies, those only capable of sustaining such expenses.

This is not to say that procedurally generated content is to be considered a poor production choice, in fact as explained by Michael Blatz et. al. (Blatz and Korn,2017) independent teams (or indies) nowadays massively rely on this process to overcome some of the obstacles that small-sized teams have to face while making sure not to force a limit on the creative scope of their projects.

While it is true that in certain tasks computers are still seen as inferior to say, an artist, in the sense that creatively speaking the latter is much more independent than the former, efforts are still being made to further enhance their capabilities with the ultimate goal of supporting humans in content creation.

With the continuous expansion of machine and deep learning in recent years, ai and computers are now capable of performing much more intricate jobs, being able to synthesize visual and auditory assets like facial animations and voices like never before. So, it shouldn't come as a surprise that many game companies like Epic Games are looking into new ways to develop games, enriching the user experience in many forms, not only related to graphics but also in terms of game design and story. The game industry just like many others who are deeply linked to some of these breakthroughs will definitely achieve a new level of both realism and complexity in their products.

A very recent news saw the announcement from epic games of a new tool which will be available in the coming months to consumers called MetaHuman Creator (Epic Games, 2021) capable of assisting asset creators to such a high degree that extremely photorealistic human animated models can be created in a matter of minutes. This is yet another evidence of the power that software and AI coupled with humans can achieve. It will be interesting to see future developments of this.

*Figure 1 (*Siren. *2018)*

Figure 2 (Epic Games, 2021)

Problem Definition

The term videogame implies a product that is intended for entertainment purposes and uses a screen as the primary medium to provide said experience. While it is true that a successful game is only as good as its gameplay, visuals still play a pretty significant role in the overall quality. After all a video game is nothing more than an interactable video. Games that feature incredibly looking graphics are more often than not able to reach a wider audience than a game with barely any.

Due to these reasons companies are more prone to dedicating a chunk of their resources to produce graphic assets and find new ways to speed the process. In comes **Computer Vision**, a sub section of both machine learning and Deep Learning that is often capable of generating the most impressive results, but what is it in practice?

Computer Vision is a type of machine learning whose job is to teach computers how to see like humans do. This means being able to look at a picture and recognize certain elements, patterns, and shapes, not just pixel colours. Identifying and classifying data in such a manner is clearly the primary concern but once they are capable of performing these tasks it becomes clear that computers can also generate images that contain people or animals for instance.

In order to first identify high semantic data like the one described above it is necessary to come up with a classification method. This can vary between **supervised** and **unsupervised**, meaning whether they rely on the **ground truth data** provided by humans to learn or are tasked with discerning features of a specific object by themselves.

Image Segmentation is used extensively in this field to teach classifiers. The idea behind this is to segment or divide certain sections of a given image and assign a predefined class to it, a label. The

final result is a picture where each pixel has a corresponding label that dictates how a certain object is made. This instructs the ML model on how to identify that object. This method has been used in both traditional ML models as well as in Deep Learning to create classifiers. Nvidia for instance used image segmentation in their spade model to create an algorithm capable of synthesizing images. By training the algorithm with millions of pictures of landscapes it was able to identify and generate new ones just by looking at a simple sketch (Park *et al.,* 2019a).

Obviously, **Computer vision** and the **Image Segmentation** technique extend to multiple industries and even if it is one of the sectors that will benefit the most from this field is not restricted to entertainment, we can see this in the **medical** as well as **security** sector and even in the automotive industry.

A broader looks at practical implementations of this will be discussed in the literature review section of this dissertation.

As previously state Nvidia has been pushing the evolution of AI for the last decade or so, focusing on many subjects that revolve around it one of which is computer vision with their GauGan (Spade) model a Generative Adversarial Network (GAN) capable of extracting semantic data from paint-like sketches and synthesizing photorealistic landscapes (Park *et al.,* 2019b).

This DL model has been repurposed in order to generate 2d Representation of Minecraft worlds in the past thanks to the work of Ridam S.S. Rahman (Rahman, 2020)**.**

Currently the feature extraction phase is performed manually and with the use of a multitude of external editors and tools, this is something the author believes can be improved upon. In its current state in a professional context the use of the model as a production tool is limited to the few people familiar enough with the specific process used to prepare the training data. Whenever possible accessibility and ease of use should be considered especially if the software is meant to be deployed as a tool for further DL research. Furthermore, the human factor makes the whole process error prone meaning that the Model might potentially learn from incorrect labelled data or data that is not precise enough.

This work is meant to add a level of optimization to the spade Minecraft model in the hope of perfecting the work previously done by Ridam but before going over the techniques implemented and the work that's been done a discussion on the current state of image classification and image segmentation will be carried out to outline some of the efforts made in this field, how they're effecting our present world and how they will in the future.

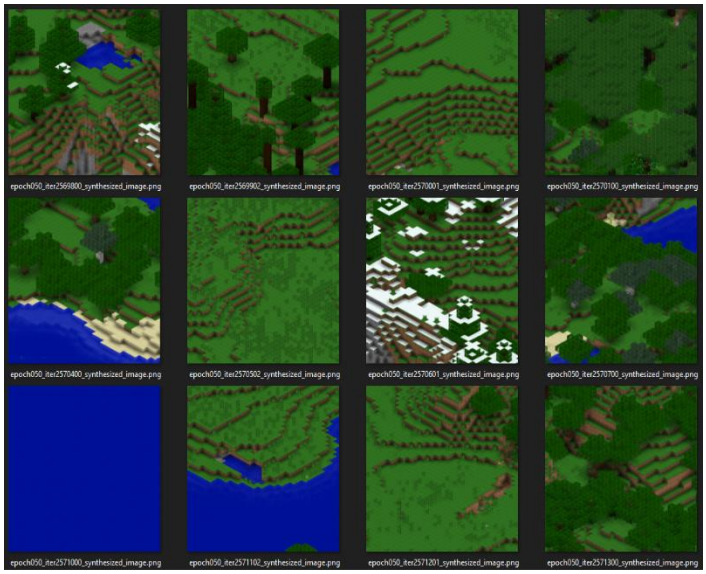


Figure 3 (Rahman, 2020)

Project Aims

This project is going to focus on the process called feature extraction, where high semantic data is collected from, more specifically the data pre-processing which is the phase that represents the preparation of the data before it is fed to the model for training.

A section in this dissertation is also dedicated to illustrating the state of research on the subject

The following points summarize the aims for this project:

- Overview of contemporary research and experiments conducted in the past towards content generation through computer vision and DL as well as practical uses in commercial environments.
- Implementation of optimization techniques which will be discussed in the methodology section
- Collection of samples to be used for evaluation metrics
- Discussion of the produced results both on a qualitative and quantitative level
- Conclusion and examination of possible future work

Background

The **SPADE** model uses segmentation maps to classify data both during the training and as a content generation tool to understand how to synthesize a picture. Before diving into how segmentations maps are used by the spade model, it's important to go over some of its uses both in **commercial** and **research applications**. This will allow the reader to get some insight into how widely used this technique is in Compute Vision applications and most importantly how effective it is in teaching ML **classifiers**.

Naturally the actual implementation of this technique varies depending on the individual attributes of the field it's going to be used in as well as the problem it is attempting to be solved, but it is still possible to categorize and distinguish some of the different algorithms used.

Preface:

Image segmentation algorithms are used to identify high semantic data from raw images (simply pixel colours), meaning that there's no system or infrastructure in place to be able to programmatically categorize the required data in the pictures at a very fine level, and with a high degree of precision, some of these algorithms are going to be explained in the following section but essentially they start by analysing pixels, since that's the first type of data any computer will be able to comprehend when all that is given to them is a picture, in the case of the Minecraft worlds images, these are not simply taken as screenshots directly from the game but rather they are generated thanks to a 2D Renderer called **Mapcrafter**, its job is to take in Minecraft worlds from the game files and create top down or isometric views. Minecraft worlds are divided in regions, chunks and individual blocks the latter being literally the building blocks of the world, the Renderer is able to reconstruct the map by parsing the content of these files and output a map. This gives us a chance to act on more than just pixels but instead classify the blocks to represent specific data like, vegetation, water, houses more precisely. Thanks to this tool being opensource it will be possible to tweak it and make the necessary modifications so that the generated maps will have more distinct features that the spade model can pick up and use as labels (i.e. colours).

So, for our specific case image segmentation algorithms are not going to be used, this is because a better, more efficient solution is at our disposal, however for the sake of argument it is going to be assumed no extra tool will be available, only Minecraft worlds in picture format are given to us and the only options are more traditional image segmentation algorithms.

What are the common algorithms used at the moment to perform the task of image segmentation? It is first possible to divide them in 2 categories

- Classical methods
- AI Methods

Thresholding (or intensity-based Method)

This is the simplest image detection method and it works on a per pixel level. Before performing the image segmentation though, usually the image is converted from colour to greyscale, this is done in order to facilitate the detection process. A threshold values is determined, to do this it's good practice to look at the number of white pixels as opposed to black through an Histogram which makes it easy to detect spikes in whites or blacks (depending on what the algorithm chose to represent the background with).

Once the threshold value is set, a specific label is assigned to pixels that go over it

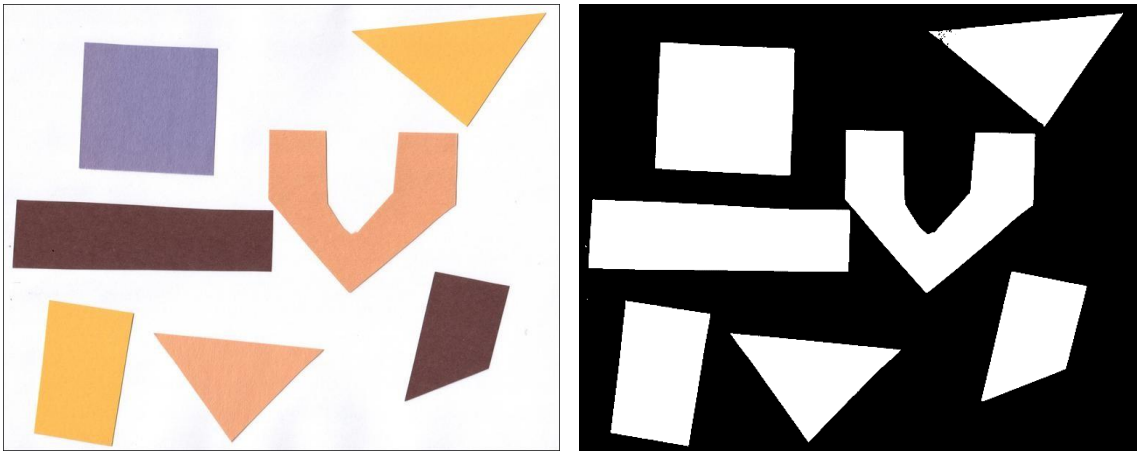


Figure 4 Threshold methods

A variation of this method called **Adaptive Thresholding** uses something called the *Otsu's method* to automatically determine the threshold value, this is especially useful when there are multiple peaks in the histogram which means that choosing one can become challenging.

So, the process could be defined as so:

1) Input image
2) Convert Image from colour to greyscale
3) Apply blur (usually gaussian)
4) Create pixel colours histogram
5) Define Peak
6) Define threshold value
7) Label pixels that exceed the threshold value

Clustering Method

Clustering is used in Image segmentation, but it is not limited to this field. It is mostly used in unsupervised environments; in the case of image segmentation this is when the elements in a picture that need to be isolated are unknown.

The method could be summarized in a data driven approach where colours are grouped based on their presence in a given image.

The most used clustering method in image classification is the K-means classification method, it is used specifically with data whose labels are undefined. Here, the algorithm attempts to generate clusters or groups (in this case labels) base on some kind of similarity between the number of groups represented (defined by the value K)

The latter could be either a value chosen in an algorithmic fashion or pre-determined, this is based on whether the type of labels is known or not

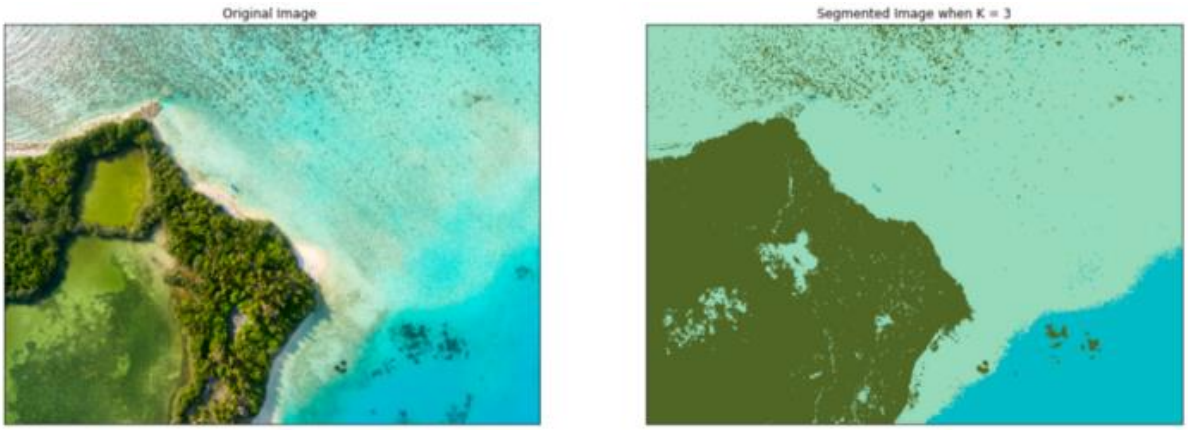


Figure 5 ('Image Segmentation | Types Of Image Segmentation', 2019)

Edge detection Method

Edge detection is another method used in image detection that works essentially by analysing the image and finding sharp differences in colour contrast between each pixel.

Edge detection can be classified in 2 distinct types:

- Gradient methods
- Gaussian methods

Gradient Methods

These are algorithms capable of computing first order derivations, the gradient method is used to detect the edges by looking at the maximum and minimum values for the gradient which is the measure of change. An example of this are Sobel, Prewitt and Robert operators. They all differ in complexity and performance and because of this they possess different advantages and disadvantages.

The sobel operator for instance is fast and simple enough but its implementation is limited to smooth edges, not to mention its sensitivity to noise.

The Prewitt operator is similar to the previous one in that it's able to detect vertical and horizontal edges but lacks the ability to spot diagonal ones.

The Robert Gradient method tries to tackle the issue of diagonal edge detection but at the cost of being very sensitive to noise

Gaussian Methods

Laplacian operator takes its name from the use of the Laplacian to calculate the second order derivation. It also implements a Gaussian function to reduce noise. With this method it's easy to detect edges but it still remains susceptible to noise and can also incur in false edges.

The Canny operator falls into the Gaussian based edge detection algorithm category and it is an improvement on the Laplacian method. Noise in this case doesn't affect the performance of the algorithm and in fact is one of the most widely used edge detection methods. The only downside to this is the high level of complexity as well as increased computation time.

Region Based Methods

This type of algorithms works on the assumption that the neighbouring pixels of a given pixel are similar to each other. By making this comparison it is possible to assume that these pixels are part of the same group, there are 2 distinct types of region-based algorithms, **region splitting** and **region growing**.

Region Growing

First of all, this method requires 2 different user inputs 1 is the number of initial regions in the image. From each region an arbitrary pixel is chosen, this is then compared with its neighbours via a specific similarity factor (this being the second user input) if the corresponding neighbouring pixels are considered similar to the seed then they are included in the seed's region, the process is then repeated with the seed's neighbours in a recursive fashion until either all the pixels in the image are part of a region or the unclaimed pixels are turned into seeds of new regions at which point the process will start until all pixels have been assigned.

A modification of the above technique has been proposed called **Unseeded Region Growing**, the key difference in this case is that the number of initial regions is not required from the user, instead the process starts with a single region represented by a pixel and the same comparison test is computed to determine whether the neighbouring pixels should be part of the first region or not. In the case of a negative response a new region is created, this is repeated until the entire image has been traversed.

Region Splitting

This is considered to be the exact opposite of region growing in that the image is fragmented in a quad tree structure. The entire picture is first considered as one single region. Based on a pre-defined criteria the image is divided in 4 quadrants, the algorithm applies the same process to each section and if the test results negative this is further separated until the comparison results positive, the worst case obviously would be if the regions end up corresponding to each individual pixel, meaning no actual image segmentation has been carried out.

Now, the region-based methods differ from the others because they offer substantially superior results as explained by (Khan and S,). In their comparison research where they assessed the different image segmentation techniques and offered their take on how they compared against each other. One of the downsides of this Region Based approach is the issue of human dependency since they rely on it for seed selection. This can consequently result in erroneous results.

These are some of the more traditional image segmentation techniques that are deployed for the purposed of image classification in computer vision.

Figure 7 Region splitting and Region growing methiods

(Khan and S,)

Figure 6 (Khan and S,)

AI Based Methods

As stated at the start of this chapter however, there is still another type of image segmentation that greatly differs from the ones mentioned above. This is **AI based image segmentation**.

Currently this is the dominant group of techniques for image classification and image segmentation following recent breakthroughs in the field of Deep Learning. The main points in favour of AI Image segmentation are substantially **better results** and little to no **human intervention**. Unfortunately, this type of technology comes with its own set of limitations, primarily the cost of data retrieval and to a lesser degree computational power. These points will be explained further in the following paragraphs.

Artificial Neural Networks or ANN play a crucial role in this. They represent the primary difference between AI based methods and traditional ones, specifically the algorithms are now trained and taught how to carry out a task as opposed to performing a set of instruction hardcoded by their creators.

One of the most used AI architectures are CNN or **Convolutional Neural Networks**, these are models specifically designed for the task of Computer Vision, they loosely resemble the hierarchical receptive field model of the visual cortex and are composed of multiple layers of neurons or nodes, one of these called kernel is a convoluted filter that is tasked with extracting features from a given image. The subsequent layer than applies an activation function to allow for non-linearity in the feature maps and after this a set of pooling layers replace a small section of the feature map with statistical information like mean to reduce spatial resolution. Some examples of Convolutional Neural Networks are VGGNet, AlexNet or GoogLeNet.

Another type of neural networks is **Recurrent Neural Networks**, they are more suited to what are called temporal or ordinal problems, these are tasks where in order to produce an output the previous state needs to be considered and weighted in. An example would be voice search, natural language recognition or image captioning. The main difference between other neural networks is that RNNs collect and store data at each time interval. There is currently a limit to how far they can go back and store data that can have a meaningful impact on the outcome, but some efforts are being made in order to overcome this limitation. **Long Short-Term Memory** architectures sets specific stages in the NN to regulate the flow of information these are: input, output and forget states.

GANs or Generative Adversarial Networks are NN that work on the premise of "collaboration"; since these are actually 2 stage networks, a Generator-Discriminator as the name suggests is a combination of a NNs tasked with generating some sort of data from a given input (Generator) and a Discriminator whose job is to distinguish the outcome of the generator from the ground truth data, this is done through the use of a reconstruction loss function. By working their way through the dataset, they are supposed to improve each other until the loss function is reduced to an absolute minimum. This type of Neural networks are very popular in Computer vision and image synthesis. The SPADE model whose feature extraction the author is currently trying to improve is based on the very same technology.

Encoder-Decoder and Autoencoders they differ from GANs but operate on the same principle of cooperation. they are used to reduce the input they're given into a compressed encoded form, this is the task of the encoder, the decoder performs the decoding of the produced output from the Encoder and computes the result from a loss function to determine how much data has been lost from the original input, the decoder is capable of applying changes to the encoder to reduce the loss function and get a better encoded output through an iterative process. This is the preferred choice when it comes to image to image translation

These types of networks are just some of the deep learning methods that are favoured when it comes to image segmentation/classification and computer vision in general. Fully connected networks, spatial transformers and capsule networks are some of the new introductions in this field.

(Minaee *et al.,* 2020)

Applications of Image Segmentation

Now that we have explored some of the techniques used in image segmentation, we can have a look at where this is used.

Medical Imaging

A very popular use of image segmentation comes from the medical field, recent studies have shown how this technology when used correctly can be deployed to identify possible cancer cells in the bloodstream or organic tissue. This could result in life changing scenarios preventing patients from incurring into serious diseases thanks for computer assisted diagnosis and analysis. Some papers that demonstrate some interest in this subject are for instance Shereen Fouad et. al. (Fouad *et al.,* 2017) where they try to segment digitized histopathological images to identify epithelial and stromal tissue to prevent further development of tumours in tumour like environments.

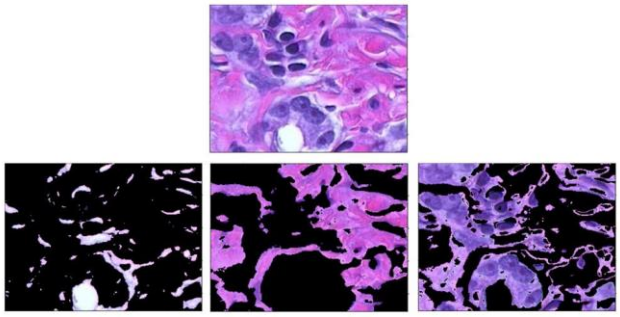


Figure 8 Brain Tumor detection (Wu et al., 2014)

Another example of this is (Wu *et al.,* 2014) where they propose an image segmentation-based method to retrieve valuable information from MR (Magnetic Resonance) images in order to detect brain cancer in some patience. Their premise is that traditional image segmentation methods are unsuited for this task as they have proved how they cannot achieve satisfactory results. Generative and Discriminative models (GANs) are also considered not adequate for this as they don't provide sufficient learning samples. Their paper is a new method that tries to tackle these limitations by using a SVM or support vector machine as well as a conditional random field.

Video Surveillance

Another use of image segmentation comes in the form pedestrian recognition and detection. This is useful as it provides a way for law enforcement institutions to regulate and enforce laws to people as it could quickly identify suspects and crime perpetrators in video surveillance footage. It is intrinsically faster, and less error prone compared to humans and thus more efficient.

Constant research is carried out especially as it ties up with another use of image segmentation that is going to be revolutionary in the following years, autonomous driving.

Papageorgiou et.al (Papageorgiou and Poggio, October 1999) are some of the researchers that are attempting to improve pre-existing pedestrians and object recognition models. Some of these rely on motion cues and are hand crafted. They claim to have developed a new model that doesn't take into account motion information and perform better than their counterparts, these are some of the performance results that have been produced by their experiment:

Figure 9 Images taken from trained AI model for pedestrian detection (Papageorgiou and Poggio, October 1999)

Autonomous Vehicles

This is a relatively new subject that is starting to gain a lot of traction and attraction from car manufactures and tech companies as they invest heavily in this subject. Autonomous cars are an invention that has been fantasized for many years and this can be seen in many pieces of literature and mainstream media over the years. Unfortunately, the technology to power these cars wasn't present until a few years ago, Google has been one of the companies at the forefront of this with their Waymo development subsidiary company which currently employs some 400 self-driving cars around the phoenix area in Arizona. They still rely on human intervention from time to time, but mostly this comes in rare situations where they are unsure about something the cameras mounted on the car frame have spotted, in fact as of November 2019 the cars operate without a remote backup driver. (Bansal, Krizhevsky and Ogale, 2018)

Machine Vision

Granted this category actually encompasses the previous two, there are still some other uses that fall into this macro area. Machine vision is a type of science that combines computer vision and other disciplines like robotics and software engineering. Its main purpose is to provide robots and software a way to see like humans and to process visual information in order to generate an output based on the visual data where it wasn't possible previously.

Some practical applications of this apart from the video surveillance and self-driving cars are industrial automation, automatic inspection robotic sorting guidance and internet of things.

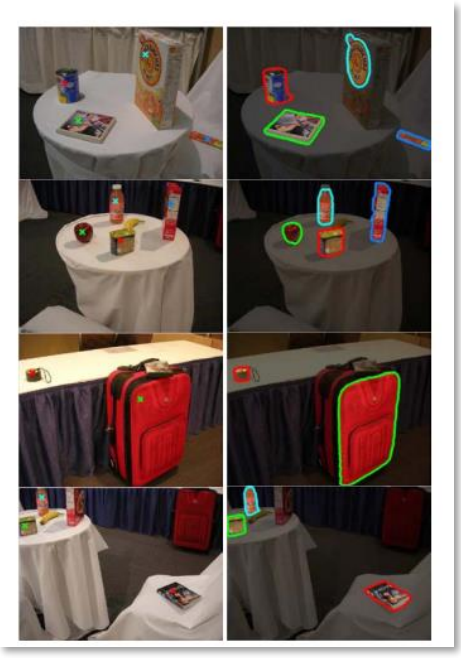


Figure 10 (Mishra, Aloimonos and Fermuller, October 2009)

GANcraft

Very recently Nvidia in collaboration with researchers at Cornell university, developed a new tool called GANcraft that is capable of generating 3D photorealistic images of large 3D Block worlds, similar to Minecraft. As an alternative to image2image translation like in the spade model previously developed by the same company the authors in this case opted for a different approach on the premise that image2image couldn't be viable as "ground truth photorealistic renderings for user created block worlds simply don't exist". This resulted in a model that has been trained in an

unsupervised fashion using what they call ***pseudo-ground truths***. These are images generated from a pre-trained SPADE model that have been used during training. The model takes semantic block worlds as input where each block represents entities like trees, grass dirt, water etc, and produces a

Figure 11 (Hao et al.*, 2021)*

3D photorealistic world that can even be rendered from different viewpoints. This process is very intensive and as stated in the paper generating even 1 frame takes approximately 10 hours using an Nvidia titan X. (Hao *et al.,* 2021)

DatasetGAN Latest Nvidia paper

This new GAN developed by Nvidia has the potential of being the new frontier in data gathering for GAN training. It serves as a fully automatic procedure to source labelled and segmented data essential for training Neural Networks. It is claimed that this new introduction the to the field outperforms semi supervised models and is on par with supervised methods with the advantage of requiring 100 times less labelled data to operate (DatasetGan still requires few images to be labelled manually). This is very much related to the subject this project is trying to focus and given how important dataset and data gathering it is certainly very good news.

(Zhang *et al.,* 2021)

Automation, human labour in the data-driven technologies and its implications

The rise of Machine Learning and the continuing adoption of technologies such as Neural Networks meant also an increase in demand in a new type of resource that has become invaluable to many companies, **data**. Neural Networks and many other types of Deep Learning technologies are directly based on the human brain, and how it operates when tasked with learning a new concept. When a human needs to recognise images of dogs they look for certain features that they have learned to correspond to dogs and dogs alone, in order to do this the subject is very likely to have been in contact with a sufficient amount of dogs or dogs images to be able to spot not just one type of dog but also other breeds. Neural Networks work on the same basis, this means that they also need to be taught first how to recognise characteristics that are unique to dogs, to do this, just like humans over time, they need to look at a considerate amount of images before they can take guesses as to whether an image contains dogs or not. The more images are available to the neural network the better and more precise the DL model gets. There are also some differences between humans and AI in this case though. One of which for instance is the ability of humans to recognise something at

different angles, whether they are facing one direction or the opposite, for humans there's no need to see images that contain the same dog from different directions because they can intrinsically tell. Artificial neural networks on the other hand can't, this means that not only they need millions of images of dogs but they also need to see them at different angles, if the training data contained pictures of the animal facing one direction, exclusively, they eventually wouldn't be able to recognise dogs that are facing the other. There has to be some form of variation that needs to be introduced in the training data before it is fed to the neural network and extreme care needs to be adopted with this phase as it can separate a good classifier with a very precise, human-like one. For these reasons data has become something companies are trying to constantly gather in one form or the other. Companies put data gathering at the centre of their business model oftentimes offering products at no extra cost to incentivise people to use it in exchange of valuable data like average use, chronological webpage history (see free VPNs for instance) and more.

In an attempt to **automate** the **data collection** used by image classifiers and other types of ML models companies like **Amazon** have come up with a way to get people to do **simple repetitive tasks** only humans are able to perform for an hourly salary. This gave many tech giants the possibility of outsourcing data labelling and other tasks that would otherwise require an enormous amount of time had they been done by a single individual. Millions of people can sign up to their website and choose to carry out simple actions like recording their voice while they read some text, categorize as many objects in an image as they can recognize, or transcribe audio to be used by Voice assistants to enhance their natural voice language processing.

And it's not just amazon with their Amazon Mechanical Turk, other silicon valley start-ups have joined in like, Appen, Scale.ai, Playment and the list goes on, this companies offer data to research groups and other businesses so that can be used to improve their deep learning algorithms.

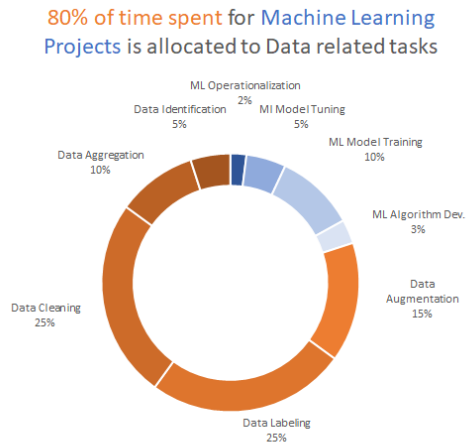


Figure 12 (Heller, 2020)

This excessive need for data over the years has brought to attention some flaws in the system that might result in catastrophic outcomes in a couple of years. Saiph Savage, director of the human-computer interaction lab at West Virginia University has been talking with technologyreview.com about the effects platforms like the Amazon Mechanical Turk can have on its employees (Will Douglas and Heavenarchive, 2020)

She explains how workers are essentially paid below minimum wage without any possible career advancements, in her study she examined the experience from various people working there and attempted to document and analyse the possible effects this can lead to. Working for large tech companies can result in many workers feeling estranged and disconnected as they perform small tasks rarely seeing the effects they directly generate. She insisted that although many of our products rely on these kinds of services to work better, there is still more that could be done to increase the quality of the workplace for the people who make all of this happen especially when there

Figure 13 (Will Douglas and Heavenarchive, 2020)

are so many. According to a study mentioned in the same article more than a million people in the US earn money each month through these kinds of platforms and around 250.000 earn three quarters of their income this way. One of the ways Saiph is attempting to improve the quality of these people's jobs is by developing AI tools to increase their time efficiency. By helping them choose to perform the most lucrative tasks or the ones that are reviewed to be from better employers they can minimize downtime and increase their revenue.

Methodology

Overview

The aim of this project is to improve on the work conducted on generating 2D Images of Minecraft worlds through the use of the SPADE model. This has been carried out by Ridam Rahman in 2020 and it is the authors intention to try and streamline some of the techniques used in order to facilitate the creation of training and testing data for feature extraction as well as removing some of the practices involved in last year's work that could be considered error prone. The following paragraphs will explain the main objectives that the author has set out and that represent the focal areas where improvements have been identified and, in some cases, successfully implemented.

In his paper (Rahman, 2020) the author illustrates how in order to generate images of Minecraft worlds, it is first necessary to train the model with appropriate data samples, in order to do this though, a particular pipeline has been put in place to obtain enough samples required by SPADE. This specific series of procedures revolve around the use of Minecraft Save files, and 3 different pieces of software:

- Mapcrafter: a very powerful Minecraft multi-threaded renderer.
- IrfanView: an image editor and image processing software.
- Microsoft paint and Photoshop

Firstly, the Minecraft maps are passed to Mapcrafter for processing. This will create a representation of each map in isometric view. At this point the maps are split up into smaller samples which will make up the dataset of labelled maps. For each map IrfanView colour correction tool is used to remove the shadows effect that cause each block in the labelled map to have 2 different hues of the same colour.

Now, Mapcrafter is capable of rendering roughly 200+ types of blocks each with its own textures, some even using more than one (One for the top side and one for the sides and bottom section). Since we're limited by the number of entities that we can successfully let the spade model infer from Minecraft maps, having a different label for each block type is highly discouraged. Instead, as previously done by Ridam a subsection of these block types will be ignored leaving only the ones we want the spade model to learn about. To method by which this is achieved on Ridam's work included making use of the mask setting in the configuration files used by Mapcrafter and manually changing the colours of the textures that are to be excluded from the training to be black. This is one of the areas the author of this project believe can be improved upon.

The block textures of the entities remaining are reduced to a dozen colours. These too have been created on paint according to Ridam's work and placed with same names as the original ones in order for Minecraft to be picked up and used when rendering the maps. The following diagrams shows some of the remaining block types and their corresponding label textures, each one representing an entity in the map.

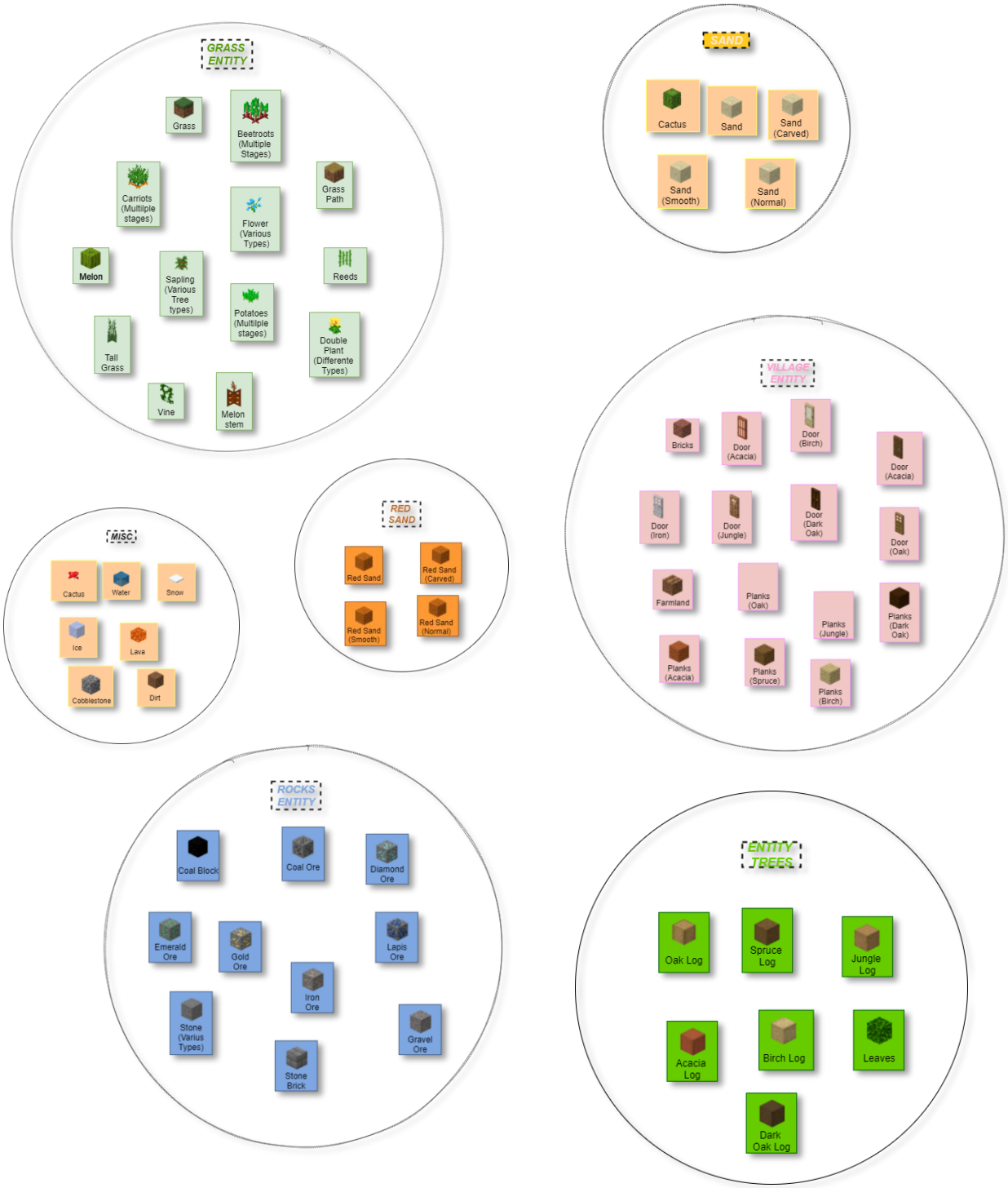


Figure 14 Semantic entities and their corresponding Blocks

This part of the pipeline can strongly benefit from some optimization as this process can be considered time consuming and error prone.

As suggested by Ridam the pipeline can be improved by applying the right changes to the mapcrafter source code. Readapting the code will require a great deal of reverse engineering, but if done right it could streamline part of the feature extraction and allow for a more versatile and faster experience.

Before making any change to it however, it's important to briefly analyse and identify the crucial sections of the code base directly related to the aforementioned changes that the author intends to make.

The changes can be summarized like so:

1. Grouping the relevant textures and replace original ones with the label colours programmatically from mapcrafter
2. Eliminate unwanted graphical artifacts and shadows from the final generated map directly from source code
3. Skip the rendering of block types that are to be excluded from the training natively from mapcrafter

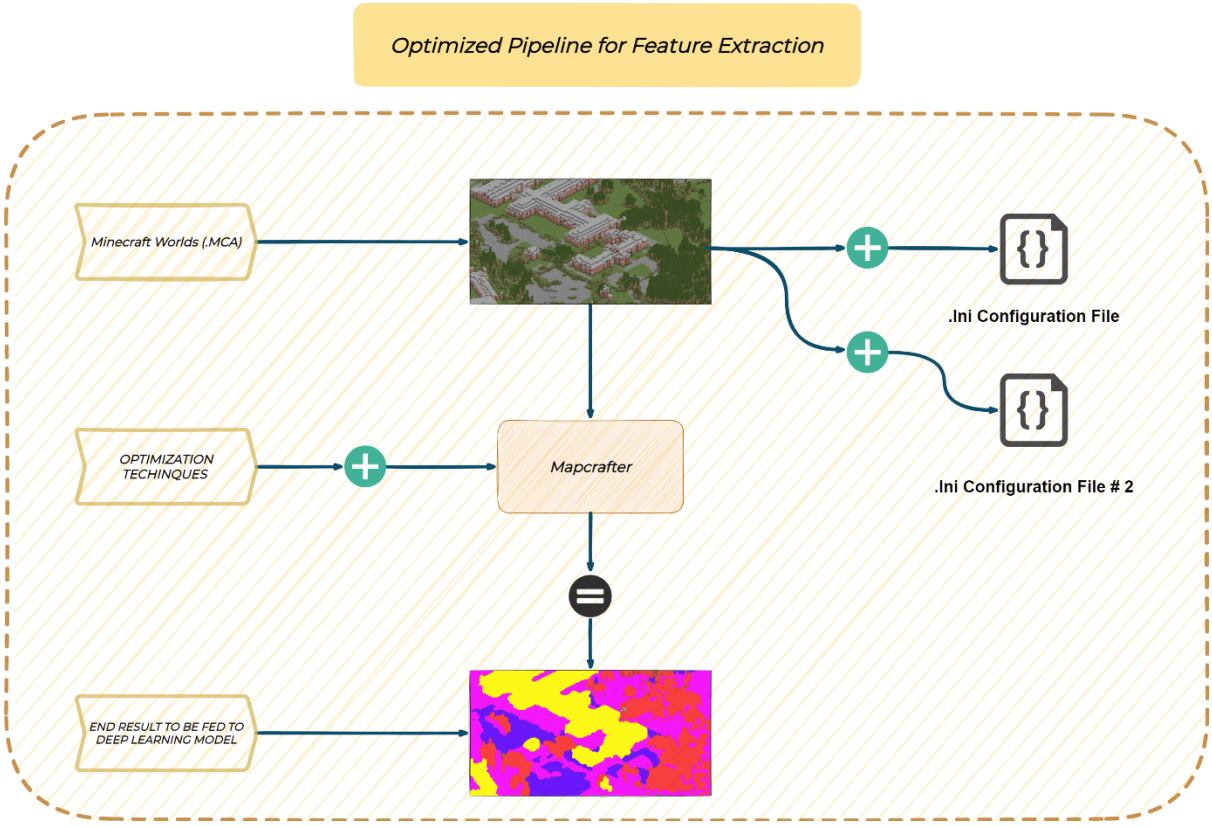


Figure 15 Optimized Pipeline for feature extraction

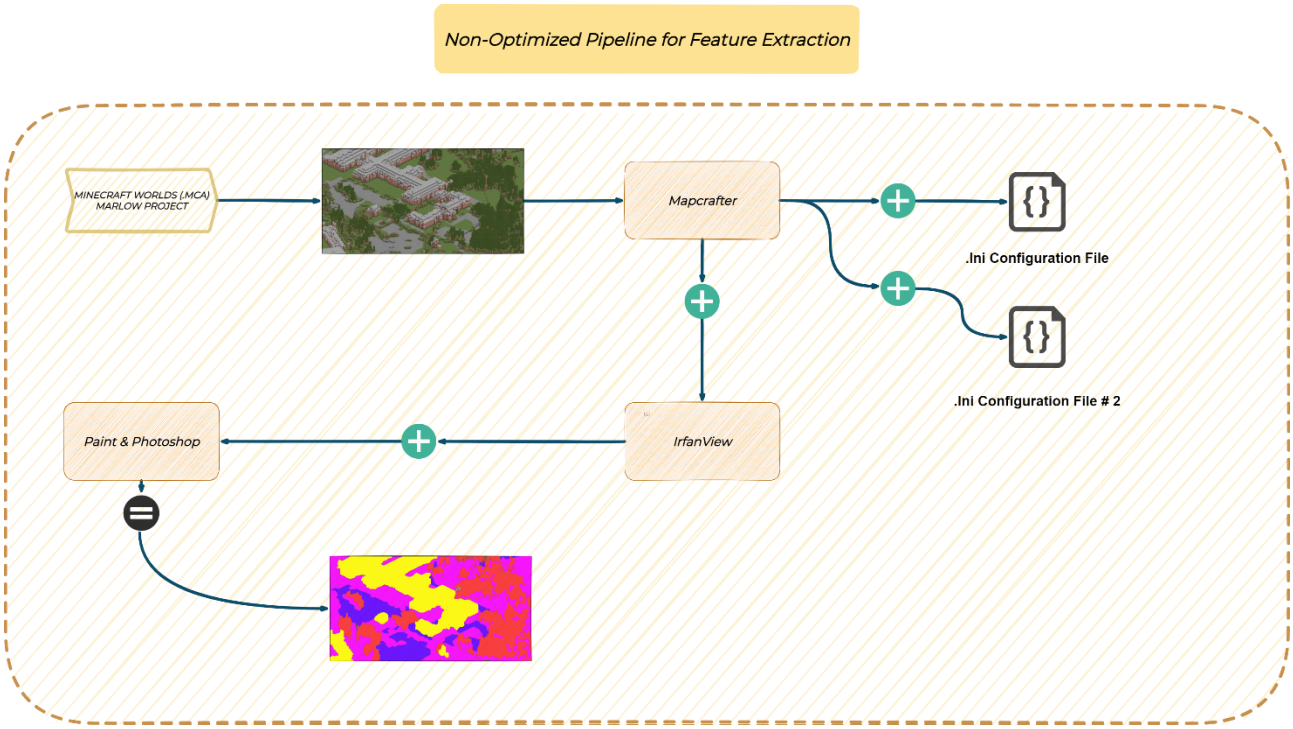


Figure 16 Non-Optimized Pipeline for Feature Extraction

The renderer

The mapcrafter source code touches on many aspects both common to graphical renderers and GUI Applications (In this case however the CMD Line version was used) as such some of the code sections responsible for storing and outputting information like the configuration file management section have been ignored. This is because the main focus has been directed towards the **renderer**, this is the section responsible with loading the necessary textures, creating the Minecraft blocks in both views (isometric and top-down) and generating the maps. In this part of the code base it was possible to isolate the significant classes that help with the creation, assignment of block textures and block types generation.

BlockTextures

The blockTextures class loads the **textures** used by mapcrafter from **disk** and its main goal is to provide said textures to all the classes that require them.

From this the maps are constructed in **tiles** following an **octree** structure and it's possible to see this in the output folder produced by mapcrafter upon finishing the rendering stage. This tree structure is used by another API, **leaflet.js** to display the map in a google maps type of style on any supported Browsers, this means that it's possible to move around the map with the cursor and zoom in and out. When doing this effectively Leaflet is traversing the tree folder structure where each layer

represents a zoom level to display smaller sections of the map at a close distance or bigger sections composed of the children tiles when zooming out.

BlockImages

This file is where most of the changes take place, specifically in the Isometric::BlockImages class which is a render view. Renderviews are exactly what the name implies, they represent different angles from which the map is to be rendered, currently mapcrafter supports 2 different render views: **Isometric** and **Top-Down**.

Isometric::Blockimages is a child of the parent class called **AbstractBlockImages** (a child of BlockImages) which holds the maps where the BlockImages are stored.

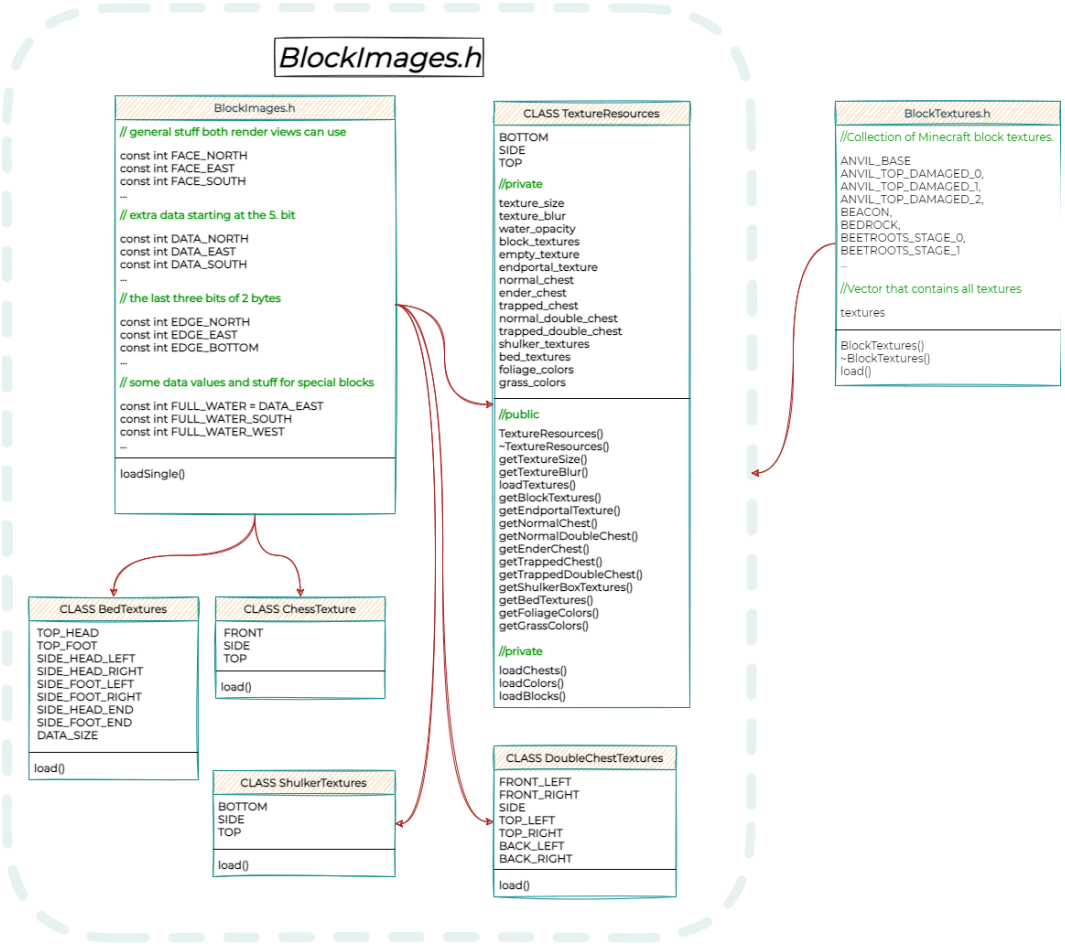


Figure 17 BlockImages.h header file

Among the many functions used to create special kinds of blocks the **GenerateBlocks()** function is where each individual type of block image is first composed and then added to the parent's std::Set

of blocks and from which the other class functions mentioned initially are called from. Here for instance the GenerateBlock function creates a BlockImage, assigns the passed RGBAImage textures for the block to each face of the block and then calls the SetBlockImage parent function which adds the BlockImage to the set previously mentioned.

RGBAImage

This is a child class of "Image" and represents an Image, when dealing with Minecraft blocks a blockImage is a type of RGBAImage which simply represents a given block in the map, it can either be isometric or top-down. It also contains different functions helpful when dealing with pixel manipulation

In this GenerateBlocks function it's possible to simply comment out some of the function calls to prevent mapcrafter from rendering unwanted blocks. This resulted in a much smaller function body where fewer blocks are actually generated (refer to the above bubble diagram to view the block types). Thanks to this change the rendering of essential blocks is performed natively from mapcrafter without having to use the mask function in the configuration map file.

Another important change was the assignment of label colours to the rendered blocks essentially doing something called "Annotation" which is another term for labelling the data. The way this was executed by Ridam was through texture files created in paint, the issue with this comes from the way these were used by mapcrafter. The code looks for the specific names of each texture files meaning that if the code needs to create the grass block, it will use the texture file called grass_top.png. In order to label the blocks, multiply instances of the same label texture were stored in the textures folder however each instance had a different name corresponding to the block type to be associated with that label colour.

The GenerateBlocks() function in this case is still where the next code change has been applied. Additionally, the BlockTextures class has been re-adapted to include 13 new RGBATextures (1 for each label this time) to the BlockTextures vector, these are passed to the GenerateBlocks function and used on the remaining block types left. Although texture files are still used, by changing the source code it's possible to assign the label colours without having to store multiple instances of the same textures thus saving on disk consumption and eliminating file duplicates.

Shadows are unfortunately an unwanted by-product of the rendering process, it makes for some beautiful isometric Minecraft maps but they invalidate the labelled maps and cause more colour presence than needed, for this reason block faces need to be of the same colour regardless of their position in the cube or the light direction.

Mapcrafter offers different render modes. As opposed to renderviews render modes decides what type of lighting is applied to the map, there are 4 pre-sets, these are:

- Daylight
- Nightlight
- Plain
- Cave

The plain pre-set according to the mapcrafter documentation doesn't apply any lighting to the blocks but as mentioned in Ridam's dissertation still doesn't get rid of the shadow effect. Through process of reverse engineering it was possible to establish the source of this behaviour.

The **configureBlockImages** function in the IsometricRenderView class declares an instance of the isometricBlockImages discussed earlier and calls the SetBlockDarkening() function to darken the sides of the cubes depending on the render mode. Similarly, to the previous points commenting out the code responsible for it resolved this issue unfortunately however there is another artifact that was needed to be removed.

Now the map presents dark edges on blocks of the same type that are next to each other but on different heights

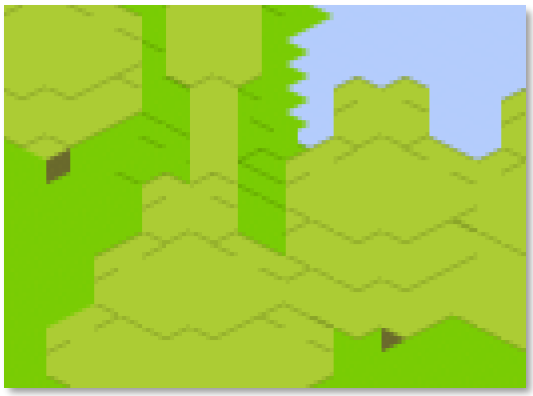


Figure 18 Edge graphical Artifact

This was quickly resolved by going into the IsometricBlockImages class and locating the function responsible for it. In this case, the addblockshadowEdges function which gets called for each of the non-transparent blocks uses 3 RGBAImages which act as masks and colour the edges of the blocks' top side by "blitting" the block's top side and the mask together. Before this however the masks are initialized in the buildCustomTextures function with a darkness value. The alpha channel of this RGBA colour was set to 0 making the mask essentially transparent and thus removing the effect of the blitting.

At this point the map is correctly displaying the labelled blocks and can be passed on to the next stage of the pipeline.

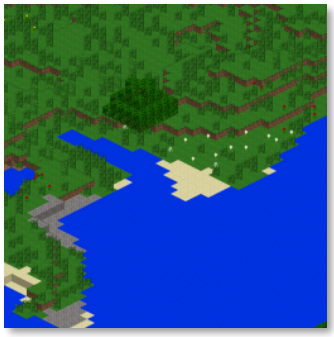


Figure 19 Ground Truth image

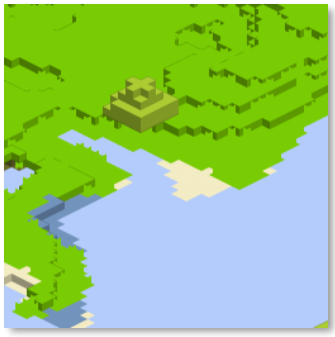


Figure 20 Labelled image with shadows and with edge artifacts

Figure 21 Labelled image without shadows but with edge artifacts

Figure 22 Labelled image with no shadows nor edge artifacts

The python scripts and the palette issue

As stated before, the output from mapcrafter is in the form of an octree folder structure of images where each layer acts as a zoom level. In order to generate a working dataset however these pictures need to be combined together. This produces a high-resolution image of the entire map that can then be split up in smaller samples. Python was used to generate the full map image, this script as well as the one responsible for the splitting comes from other sources but some changes were made to accommodate for our special case.

The first script called **Minecraft.py** traverses the octree in a recursive fashion. the **bounds** variable instructs the code on what section of the map to capture and generate the full map image from. Another useful feature of mapcrafter is the ability to render maps from different rotations however when doing this the bounds had to be modified accordingly.

Upon completion of this phase a new issue was put to light. Since the spade model pays particular attention to the labels used (in this case being RGB colours) the generated map couldn't contain more than the colours chosen to represent the labels. Unfortunately, the PIL library used by the Minecraft.py script to create the full map image ended up using an antialiasing filter. This resulted in sections of the image where colours where blended together using more hues than it was originally intended.

The PIL documentation didn't offer any easy fix and it was left to the author to find a workaround. Ridam encountered a similar problem and one suggested solution was to create a script that would assign to each pixel in the image one of the label colours based on a nearest neighbour colour algorithm in order to get rid of the extra colours. In our case the author's script loops through each pixel and calculates the Euclidian distance between any given pixel and all the label colours, finally it simply changes the former's colour to the closest label colour. This script effectively helped limiting the number of colours in the image palette however this script didn't restore the map to its original state as some sections of the map where the antialiasing filter was applied where still labelled incorrectly

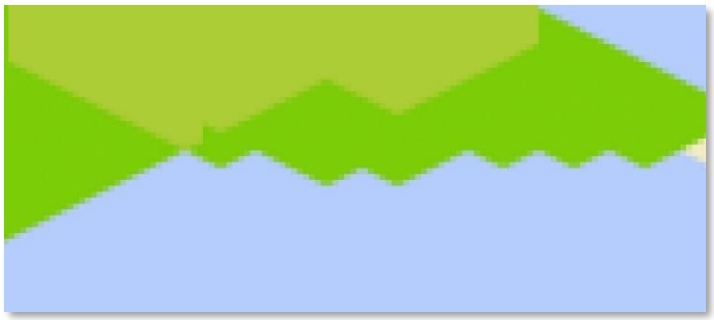


Figure 23 Before using palette script

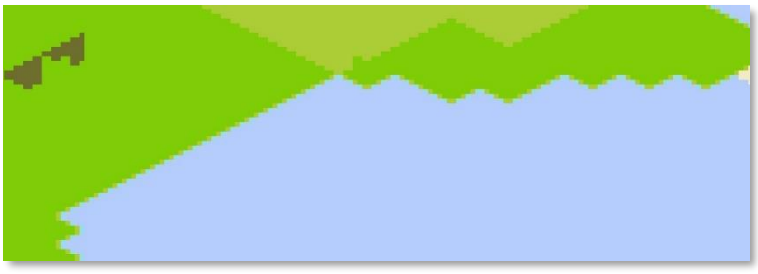


Figure 24 After using palette script

Following this the image was simply split in 256px256p images and passed on to the next phase, the samples colour conversion.

At this stage the dataset contains images that have colours in RGB format however the SPADE requires images with one channel where each pixel represents the index of the semantic label class it belongs to. Another important aspect is that the labels indexes need to be in continuous form. This conversion and the subsequent training and testing split are done thanks to the blog article on Paperspace from Ayoosh Kathuria titled Understanding GauGan Part 2: Training on Custom Datasets (Hao *et al.,* 2021).

Before applying changes to the dataset a text file needs to be created which will contain a list of all the semantic labels and their colours, after this a script parses the content of the text file and assigns the index of each of the labels to the corresponding pixel colours in each of the images in the dataset. Another script takes care of the partitioning between training (80% of the dataset) and testing set (20%). The same script splits the ground truth dataset with the same ratio. A previous attempt has been made by the author to produce the same script and although working correctly when tested on a smaller dataset it didn't behave as expected when used against the mapcrafter dataset.

At this point the end of the pipeline is reached and the data can then be fed to the SPADE model for testing. To test and evaluate the quality of the new and improved pipeline the SPADE model has been training on newly created data, however some difficulties were encountered during this process which will be explained in the next section of this dissertation.

TESTING

Limitations

Sourcing the required Minecraft maps proved to be more of a challenge than previously anticipated. This is a step that is actually prior to the first in the data pre-processing pipeline. As the quality of the maps from which a dataset is created can vary greatly depending on the requirements, finding suitable data in the form of Minecraft save files can become a laborious process. Many communities over the internet provide custom maps to players to play with, however these are more often than not filled with specific shapes, buildings, and sculptures. An important rule in Deep Learning is that a Neural Network is only as good as the data it's been given and in our case the maps needed to have features that resemble the original look of the game more closely, in order to do that the author decided to directly jump into the game and create a world manually. Now, this obviously goes against the level of automation it was intended to be achieved but given the focus of this work revolved primarily around changing the mapcrafter source code to provide labelling capabilities this was deemed the most practical way to get the required map data. (Refer back to the Improvements Chapter for a potential fix).

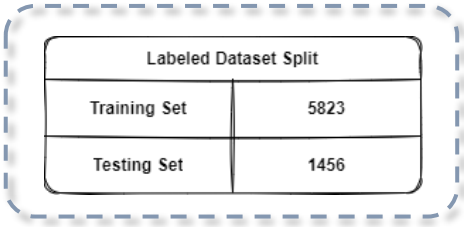
As the adopted method involved a lot of manual operations the data gathered and passed to the improved pipeline originated from one single map. This is an important factor as it can have a substantial effect on the number of generated samples in the final dataset. Because of this in fact, it

is the author's belief that the training phase could have greatly benefitted from a greater number of Minecraft maps and consequently a bigger dataset.

Secondly as explained in the methodology section the labelling of the blocks was invalidated by the Minecraft.py script by applying a form of anti-aliasing on the blocks' edges when creating the full-map image. Although this was partly fixed by the author's python script it wasn't however fully removed and it is believed this could impact the final quality of the training stage.

Finally, another important aspect of the dataset that's been generated is an extension of the first point. In order to train a GAN correctly the labelled semantic entities need to be equally present in the training set: if the number of village or cobblestone blocks are not as many as the grass or trees blocks the chances of having a correctly inferred understanding of the features of the former are very slim.

Training



After the generated map from Minecraft has been processed by the pipeline the new dataset can be fed to the SPADE model.

For training the test.py script was used, and some command line arguments were passed to instruct the spade on what dataset to use and how many GPUs to perform the training with:

- --Name: The name of the experiment
- --Dataset_Mode: can either be custom or on one of the datasets provided by Nvidia
- --No_Instance: Boolean value to indicate that the dataset doesn't contains any instance map
- --Label_dir: the directory that holds the labelled samples used for training
- --Image_dir: the directory that holds the ground truth images used for training (each pair of labelled and ground truth pictures need to have the same filename in both label and ground truth dataset)
- --gpu_IDs: the GPUs to be used for training
- --batchSize the number of images passed to each GPU for training, this needs to be a multiple of the # GPUs

The training of the model took approximately 2 days and was performed on the university talisker Linux server after which testing was conducted.

Testing and Evaluation

For the testing phase 1456 samples were used from both the labelled and the ground truth dataset, the test.py python script was deployed. Some of the CMD line arguments passed are similar to the train.py script but with an additional option:

--which_epoch: this parameter allows the testing to be done on a different epoch than the default one which in this case is just the latest.

An epoch being the total number of times a neural network works through the entire training set. For each epoch learning weights are adjusted meaning that for every epoch the neural network is potentially learning. Thanks to this option it's possible to test the training set over the course of the training process.

As suggested by both Ridam and Ayoosh in his blog qualitative assessment could be taken in the form of user-preference study but a more popular approach of quantitative nature was used instead. The Fréchet inception distance or FID is a metric used in many studies that revolve around the use of GANs. A neural network called the InceptionV3 is at the core of this method. The distance between the feature vectors from both ground truth and the synthesized sets are calculated, and the FID score is derived. A high FID score signifies a poorly trained model whereas a very low score means that the ground truth and the synthesized images are very similar

Testing was carried out with different epochs to illustrate how the training process proved to be successful

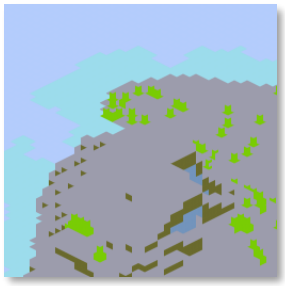
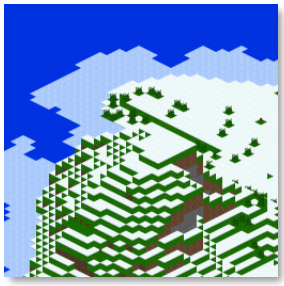
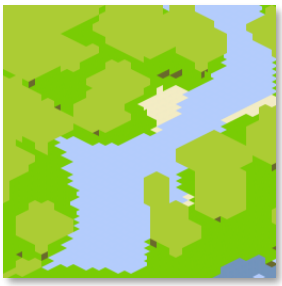
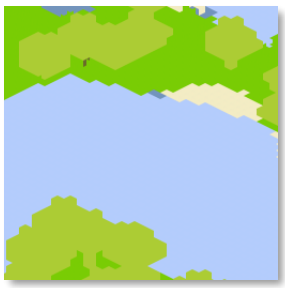
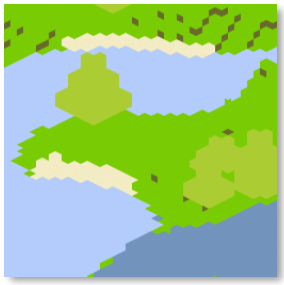
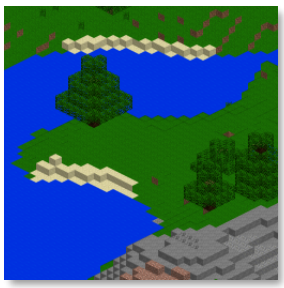
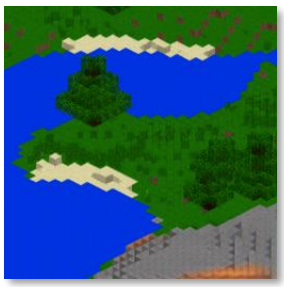
Testing results	
Epochs	FID Calculated from trainin set (real images vs synthesized ones)
10	36.342
30	33.055
40	32.862
50	32.648

Figure 25 FID Testing results

Starting from epoch number 10 downwards it's possible to notice a decrease in the FID score, this implies that the training process has been carried out successfully and that the model correctly captured the necessary features of the semantic entities in the maps.

An additional test was meant to calculate the FID score for the training data and compare how well the model performed when asked to synthesize images that were used for training, however there was no easy way to get the synthesized maps on different epochs for the training.

The following are images taken from the testing results; the ground truth maps were processed by the edited version of mapcrafter the others are maps synthesized by the newly trained SPADE model.

Labelled	Ground Truth	Synthesized
		
		
		
		

Conclusion

The undertaking of this project was aimed at answering the question of "Is it possible to automate the feature extraction through mapcrafter for content generation". The author believes that there is a strong possibility of fully repurposing the mapcrafter source code in order to automatically generate a dataset without human intervention. Unfortunately given the limited amount of time and some challenges that have been encountered during the implementation and testing phase this scenario was partially achieved but having now gained more familiarity with the mapcrafter tool and its inner workings it's safe to say that the current pipeline is an improvement on the one used previously by Ridam.

Although some of the objectives that were set initially were eventually scraped in favour of the ones that were directly related to streamlining and automating the pipeline. Nonetheless the attempt made by the author definitely marks a step towards a fully automated process that can have an immediate impact on data gathering for GANs which is of vital importance when training Neural Networks for content generation in games.

Incomplete work and Suggestion on future work

The labelling process done through mapcrafter is still using texture files, the author believes that it's possible to generate RGBAImages programmatically and fill the pixels for each label image with the fill() function provided by the RGBAImage class. This can decouple the processing of the map by mapcrafter from any kind of texture files stored on disk. As can be seen in the CPP files in the project submission some attempts have been made unfortunately tests proved unsuccessful and the results didn't yield the expected results. Due to time constraints this part of the code couldn't be debugged and eventually deployed in the final Mapcrafter build.

Initially the testing phase was planned to include more variety in the generated datasets. For each of the changes made to the mapcrafter code a new dataset was to be generated and tested against the spade model and finally results compared. Unfortunately, it was only later in the final stages of the project that the author found out about the colour palette issue and how images needed to be converted according to the label colours which prevented the planned testing methodology from finalizing. The datasets were to be created one for each change and some of these datasets were to purposefully include artifacts and shadows that would later be fixed and removed but due to incompatibility with the strict SPADE requirements on labelled images this was deemed unfeasible

The following are some suggestions on future improvements that are believed to take the pipeline a step closer towards full automation:

Project Malmo ('Project Malmo',) is a mod for the Java version of the Minecraft game that helps developers research and train AI bots to perform specific tasks, this can be anything from gathering specific minerals, sculpting certain structures or even traversing an obstacle course. In order to generate Minecraft maps, it is required that the player moves around the map to add chunks into the world, this was part of the work the author has done to gather Minecraft maps. An AI bot could be trained through project Malmo and help with the sourcing of these maps to be used with the new

pipeline. Given how they play an important role in the final quality of the dataset this could be crucial.

During the final stages of the pipeline python scripts were used to generate full map images, fix the colour palettes, split them into smaller samples and convert the pixel format, these could be incorporated into mapcrafter source code so that the steps humans need to perform are even fewer than before by making essentially mapcrafter the only tool needed to generate the dataset.

Bibliography

'Image Segmentation | Types Of Image Segmentation', (2019) *Analytics Vidhya,* -04-01T03:45:17+00:00. Available at: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/ (Accessed: May 25, 2021).

Siren. (2018) Available at: https://cubicmotion.com/case-studies/siren (Accessed: Feb 10, 2021).

'Project Malmo', *Microsoft Research,* . Available at: https://www.microsoft.com/en-us/research/project/project-malmo/ (Accessed: May 25, 2021).

Bansal, M., Krizhevsky, A. and Ogale, A. (2018) 'ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst', .

Blatz, M. and Korn, O. (2017) 'A Very Short History of Dynamic and Procedural Content Generation'*Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, pp. 1-13.

Epic Games (2021) *A sneak peek at MetaHuman Creator: high-fidelity digital humans made easy.* Available at: https://www.unrealengine.com/en-US/blog/a-sneak-peek-at-metahuman-creator-high-fidelity-digital-humans-made-easy (Accessed: Feb 10, 2021).

Fouad, S., Randell, D., Galton, A., Mehanna, H. and Landini, G. (2017) 'Unsupervised morphological segmentation of tissue compartments in histopathological images', *PLoS ONE,* 12(11). doi: 10.1371/journal.pone.0188717.

Hao, Z., Mallya, A., Belongie, S. and Liu, M. (2021) 'GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds', *arXiv preprint arXiv:2104.07659,* .

Heller, M. (2020) *Data Labeling: AI's Human Bottleneck.* Available at: https://medium.com/whattolabel/data-labeling-ais-human-bottleneck-24bd10136e52 (Accessed: May 25, 2021).

Khan, A.M. and S, R. *Image Segmentation Methods: A Comparative Study*.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. and Terzopoulos, D. (2020) 'Image Segmentation Using Deep Learning: A Survey', .

Mishra, A., Aloimonos, Y. and Fermuller, C. (October 2009) *Active segmentation for robotics.* pp. 3133.

Papageorgiou, C. and Poggio, T. (October 1999) *Trainable pedestrian detection.* pp. 35.

Park, T., Liu, M., Wang, T. and Zhu, J. (2019a) 'Semantic Image Synthesis with Spatially-Adaptive Normalization', .

Rahman, R. (2020) *Using Generative Adversarial Networks for Content Generation in Games.* Available at: (Accessed: Feb 10, 2021).

Will Douglas and Heavenarchive (2020) *AI needs to face up to its invisible-worker problem.* Available at: https://www.technologyreview.com/2020/12/11/1014081/ai-machine-learning-crowd-gig-worker-problem-amazon-mechanical-turk/ (Accessed: Apr 29, 2021).

Wu, W., Chen, A.Y.C., Zhao, L. and Corso, J.J. (2014) 'Brain tumor detection and segmentation in a CRF (conditional random fields) framework with pixel-pairwise affinity and superpixel-level features', *International Journal of Computer Assisted Radiology and Surgery,* 9(2), pp. 241-253. doi: 10.1007/s11548-013-0922-7.

Zhang, Y., Ling, H., Gao, J., Yin, K., Lafleche, J., Barriuso, A., Torralba, A. and Fidler, S. (2021) 'DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort', .

Abderrahim Jami

-