Solent University

School of Media Arts and Technology

BSc (Hons) Computing

Academic year 2018-2019

Julius Böcker

"Art and AI, creating an AR drawing tool"

Supervisor : Dr. Cédric Mesnage
Date of submission : May 2019

Acknowledgements

I would like to thank the deep learning community for offering great advice and content on the internet I cannot quote, because they are not proper sources. Also, I would like to thank my supervisor for inspiring me with an exciting project proposal.

Abstract

Inspired by the latest developments in Artificial Intelligence, we wanted to develop an application that can employ new technologies like Generative Adversarial Networks and image classification in combination with art. The application should help to and inspire artists by projecting generated art to the real world with Augmented Reality. The artist should be able to film his artwork, which is recognised by an AI, and get AI generated Art as an overlayer to his canvas. The purpose of this project is to build a proof of concept prototype and gather research over available frameworks and methods that can be used to achieve this goal. The scope of this project is set very high, because multiple techniques that need to be implemented and be able to work together. Moreover, successful presentation of GANs creating art was mostly done by professionals and no examples of working mobile usage of such has been released. In the following we will discuss what steps needs to be taken to build a prototype combining AI, AR and art as a private developer.

Acronyms

AR – Augmented Reality

AI - artificial intelligence

ML– machine learning

NN – neural network

GAN – generative adversarial network

List of Figures

Contents

1. Introduction

1.1 Background

Humanity has always been involved with art and drawing. Even in prehistoric times humans have created cave drawing to express themselves. Art has evolved since then using different mediums like oil paintings, photographs or digital art. The Interaction between the artist and the medium has always played an important role in the history of art.
To the present-day digitalization has great influence even on artist. They can scan or photograph their work to create image files, which can be publish on the internet.
Some artists use drawing pads to create digital art, but analogy art mediums still hold more value to the general public. Yet, some artists who prefer analog still use digital assistance in the form of there smartphones, by looking up references or inspirations on them.
In this digital age, with the immense increase of the popularity of smartphones, it is possible to create more computer assisting drawing tools that help with analog drawing without the need of excessive usage of a digital medium.
By using Augmented Reality, the User could get feedback for their art, by viewing their canvas through their smartphone. Augmented Reality has been an emerging term in the last years with new development of powerful mobile devices. But the concept has been around for over 50 years (Kipper and Rampolla 2013) . Although in a different form e.g. by using head mounted glasses (e.g. HoloLense) to see the real world while simultaneously projecting digital images into it.

In the last years Artificial Intelligence has gained much popularity with new modern approaches to AI (Russell, Stuart J. (Stuart Jonathan), Davis and Norvig

2010). After Deep Blue won against chess world champion in 1997 the company Deep Mind created stronger AI in the recent years to beat human players in even more complex games than chess. For example, AlphaGo's take on the supposedly hardest game of the world, the Chinese game Go. Contemporary AI chess engines have advanced immensely and reached unbeatable levels by humans. Chess Engines like AlphaZero or Stockfish achieve this by playing against each other in thousands of simulations and learning from their mistakes. To gain this experience as a human is impossible in a lifetime.

The idea of AI creating creative work was already discussed around the time of Deep Blue. (Boden 1996) At that time it was possible to create musical symphonies in the style of great composers, which nowadays are so perfect it is hard to tell that they are written by a machine.

Likewise, stunning Images were created by AI in the last years that can be described as real art. In 2014 Generative Adversarial Neural (Goodfellow *et al.* 2014) where developed, that use unsupervised ML to can generate images or music that has not been seen before. Psychedelic images created with googles Deep Dream Generator or paintings resembling old oil masterpieces that sell for nearly half a million dollars are can be credited to the research of GANs.

Figure 1: A picture generated by ai and sold for 432,500$ (Cohn 2018)

These Adversarial Networks are somehow very similar to the chess AIs. They also use a game theory approach and have 2 players fighting each other. One player is a neural network that generates something. At first, nothing more than a shapeless grain. His adversary is a discriminator, a neural network that was trained to classify an image. Like a teacher to a pupil, he won't let the generator pass unless the result could be classified as the desired outcome. But he still gives hints how big the error rate to succeed is. A GAN can learn how to generate an image of an unknown object with enough input data. Just like someone who has never seen a cat, could learn how to draw it after seeing thousands of pictures of cats. At least that's what we wish if we were talented artists. But here our proposed prototype could help someone to draw a cat after his first attempts have been recognized by our neural network.

1.2 Context

Our initial idea was inspired by recent advances in AI with Art. We wanted to let our AI to create something whether it will very abstract or productive. Computer Assisting Drawing tools already exist but are mainly used by architecture or designers who need precise measurements. Most CADs are also commonly used with a mouse or drawing tablet. Yet, we wanted to let the artist interact with a real world medium and let our AI generated results being augmented on it.

AR can be descripted as “a live or direct view of physical real-world environment whose elements are supplemented by computer -generated sensory input such as sound, video, graphics or location data.” (Scoble and Israel 2016)

The tasks of such an AR aided drawing tool would be:

To detect a real-world canvas with the help of sensors. (mainly the camera and the geomagnetic field sensor and the accelerometer)

To collect and pre-process the image data of the canvas, to analyse the given data, to give feedback for user in form of a digital projection on the viewed canvas displayed on the screen of the mobile device.

Furthermore, this project aims to make use of artificial intelligence(ai) to make predictions for the artist. The emerging term ai could lead to confusion. It is not meant as a human like intelligence, but rather a way to get a prediction after a system has been trained on sets of data. With a method of conventional programming the task of getting a good prediction from the data of a canvas is difficult. The inputs of drawing on a canvas are hard to compute and a representation of art is hard to grasp with an algorithm. There, the advantages of machine learning come in. By definition, machine learning finds patterns of data without explicit instruction and might learn how to associate features of inputs such as images with outputs such as labels.(Appenzeller 2017)

These techniques are already used for image detection. Data is computed in different layers of Neural Networks to get answers what objects an image contains. In a sense AR already is related to AI for image recognition of real-world objects to align AR elements and could play an important role in the future to improve AR experiences in general.
The initial proposal of the project differed from the final prototype due to a misunderstanding. In a mock-up of the project art on a canvas was captured by a camera that inputs the information to a neural network and then projects art generated back on the canvas as inspiration for the artist. The picture of a projector was interpreted as a way of projecting the art in an Augmented Reality application onto the canvas. But it was intended to literally use a projector directed onto the canvas. By definition, using a projector to project digital images while still seeing the real world is also Augmented Reality. This confusion could be explained by the general association of AR with mobile phones or holographic lenses.
At the end, using a projector would have made development probably easier. Mobile could have been neglected and AI is not as advanced yet as on PC. Especially in the field of using Generative Neural Networks.
Disadvantages of this method would been worse general accessibility. Nearly everyone could use the prototype on a mobile phone. In later stages of the project this could aid to gather a great amount of data from our users. Furthermore, the projector needs to be adjusted to fit on the canvas and is inconvenient to set up.

2. Literature Review

(Scoble and Israel 2016) present their research on the impact of Augmented Reality and Artificial Intelligence on society and the current state of technology and make predictions for evolutions in the future. Although the topics are very much related to the task of developing an AR drawing assisting tool working with AI, Scoble and Isreal delve more into the social and economic impacts of these technologies and only cover the technical aspects only on the surface. Their book is intended for business decision makers and not for developers. They have deep roots within the digital industry, but some parts appear to be buzzwords from the emerging technologies and hyped up futuristic visions.
Further literature about Augmented Reality was written by (Kipper and Rampolla 2013). They give a detailed definition of what exactly falls under the term and give a brief history of Augmented Reality. Going back to 1962 with a motorcycle simulation by Morton Heilig to present day Mobile Reality Applications. They also give a very good overview of the applications of Augmented Reality and different platforms it can be achieved with. For our project, it helped to outline definitions of AR with simple explanations of its functionality. "…Augmented Reality is taking digital or computer-generated information, whether it be images, audio, video, and touch or haptic sensations and overlaying them over in a real-time environment. Augmented Reality technically can be used to enhance all five senses. But its most common present-day use is visual. Unlike Virtual Reality, Augmented Reality allows the user to see the real world with virtual objects superimposed upon or composited with the real world" (Kipper and Rampolla 2013
In (Yee, Ning and Lipson 2009) a 3D sketching method based on Video See-through Augmented Reality is developed and discussed. Thy are using a head mounted stereo display, an optical tracking system and head mounted cameras. A wand is being used in the hands of the user to draw 3D content.

This setup is kind of reminiscent of contemporary Virtual Reality Devices, because they are also using stationary optical tracking systems to calibrate a headset and the drawing wand that is very similar to an HTC Vive controller. Nonetheless it is Augmented Reality since virtual objects are projected on video of the real world, as defined by (Kipper and Rampolla 2013b).
They state that their sketching application has minimal functionality since the focus on the paper is the proof of concept of a real-time 3d sketching system with live 3D interfaces. As technology used, they printed out markers and used the open source framework ARToolKit, which is still being used today and a consideration for our own project.
To evaluate the feasibility of augmented reality frameworks, (Herpich, Guarese and Tarouco 2017) gave a comparative analysis of them even though their publication also aimed at educational applications. For the educational part, they also considered the ease of development with some frameworks offering an AR editing platform and features that would benefit educational purposes such as word or textual recognition. Even though the educational aspect was not relevant for our project, they tested the frameworks on relevant features for general dvelopement. Text,- Image-, 3D Object-Recognition, also Markerless- Multi-target-, Online/Offline-Recognition and Geolocation-Recognition which could be ignored for our project. In total, 11 frameworks were tested. In previous research for our project, 3 of them were already a consideration. Namely ARToolkit, Vuforia and Wikitude. In their conclusion ARToolkit was rated one of the worst frameworks with 4points and Vuforia (8) and Wikitude (9) were rated as the best. A key advantage for Vuforia pointed out is its markerless tracking with extended tracking, that allows the camera to lose track of a marker and still remembering its position and good textual recognition. Wikitude also offers markerless technology and a web-based AR editing platform. But Herpic's, Guarese's and Tarouco's journal from 2017 did not include a promising framework for our project, google's ARcore that only was officially released in 2018.

(Glover 2018) with an extensive book about Unity Augmented Reality projects covers ARKit, ARCore and Vuforia. It has a very practical approach to setting up and using these frameworks in Unity and provides step by step tutorials for specific AR projects. It also gives theoretical knowledge about AR techniques.
Another promising source for our project was about the potential of Augmented Reality computer-aided drawing (CAD). (Wang and Dunston 2006) claim to have developed an AR CAD prototype for architecture engineering and construction industry to help visualization and spatial cognition in these areas.
For a brief overview of Artificial Intelligence (Warwick 2012) gives insight about basic concepts, how intelligence can be defined and difference of weak AI and modern AI techniques. Expert Systems can also be considered AI having a Top-Down approach familiar to classical programming. Multiple Rules of IF…THEN can be used to get a solution of a problem. But Warwick also describes their limitations to complex problems, like problems of our project e.g. image recognition and creating of art. Later, Warwick describes modern AI that use neural networks similar to the biological intelligence of the brain. Furthermore, he briefly mentions the topics of image transformation and image analyses, which are crucial to our project. Pre-Processing image data to reduce noise and methods for edge detection in images are also mentioned. The book is written for academic purpose and gives very good overviews about certain topics but does not deal with the practicalities of programming.
In her book, (Boden 1996) covers the topic of Artificial Intelligence extensively and even 20 years ago had surprisingly relevant approaches to current state of the art AI. The book is actually a collection of contributions of 11 different authors. In addition to theoretical outlines and promising recent developments at that time. It also deals with philosophical aspects, Human-Computer Interaction and Creativity using AI. HCI is also a very relevant field that needs consideration in later stages of our project. But modern approaches would be more useful that also involve mobile use cases. In the chapter about HCI by Mike Sharples, there were only little overlapping aspects with today's HCI and

UX Design practices. In the last section of this chapter there was an interesting sentence: “Human-computer interaction is a new discipline, and it has had little opportunity to mature, because computers and their users are changing so rapidly. At its worst, HCI is a mishmash of anecdotes and good intentions. But as its best, it blends the psychology and technology of computing, it begins to turn software design from an art into science and offers guidelines for good practice in developing and deploying computer systems.” (Boden 1996)
The more relevant topic for our project Creativity and AI seemed interesting as it became very popular in recent years with Generative Adversarial Networks creating incredible art with GANs being first introduced in 2014 by (Goodfellow et al. 2014). And yet the idea already existed 20 years ago. They describe the concept of creativity, possibilities for ai creating art and works that already tried this. “An AI model can help to show the precise generative potential of the conceptual space concerned and suggest detailed questions not thought of by humanist scholars.” In a subsection about music, computer generated compositions in the styles of Mozart, Stravinsky etc. are discussed. And recently in 2015, ai composed music was performed by a real orchestra in Dubai. She states that further progress will depend on AI research in many areas.
In a blog post, (Desai 2017) writes very detailed and practical about his final project in his Convolutional Neural Networks course about neural style transfer. The style of a piece of art is transferred onto a picture. In contrast to previous neural style transfer methods that start off with random noise and create the picture with many training iterations, this method pretrains a network for specific style, so it can produce stylized images instantly.
(Appenzeller 2017) writes about the latest developments in AI in an article on sciencemag.org. The article gives a broad trivial overview about AI and has a section with brief definitions and explanations about popular AI terms, that are handy to use in a review.
To combine Vuforia’s AR framework and python-based machine learning APIs, we wanted to use the Unity Machine Learning Agents Toolkit. This open source

project offers useful documentation on their github page. In addition to these, they were also very kind and published a reference paper to cite as a researcher. (Juliani et al. 2018) wrote about creating a learning environment for machine learning agents in Unity as a simulation platform. Problematic in regard to our project is only that ml-agents focus on training agents in Unity Scenes that simulate physics and other complex behaviour, whereas our project uses Unity as a mere tool to enable AR Interactions and detection of real-world objects and needs machine learning tools to generate or classify image data.

2.1 Visualization of literature

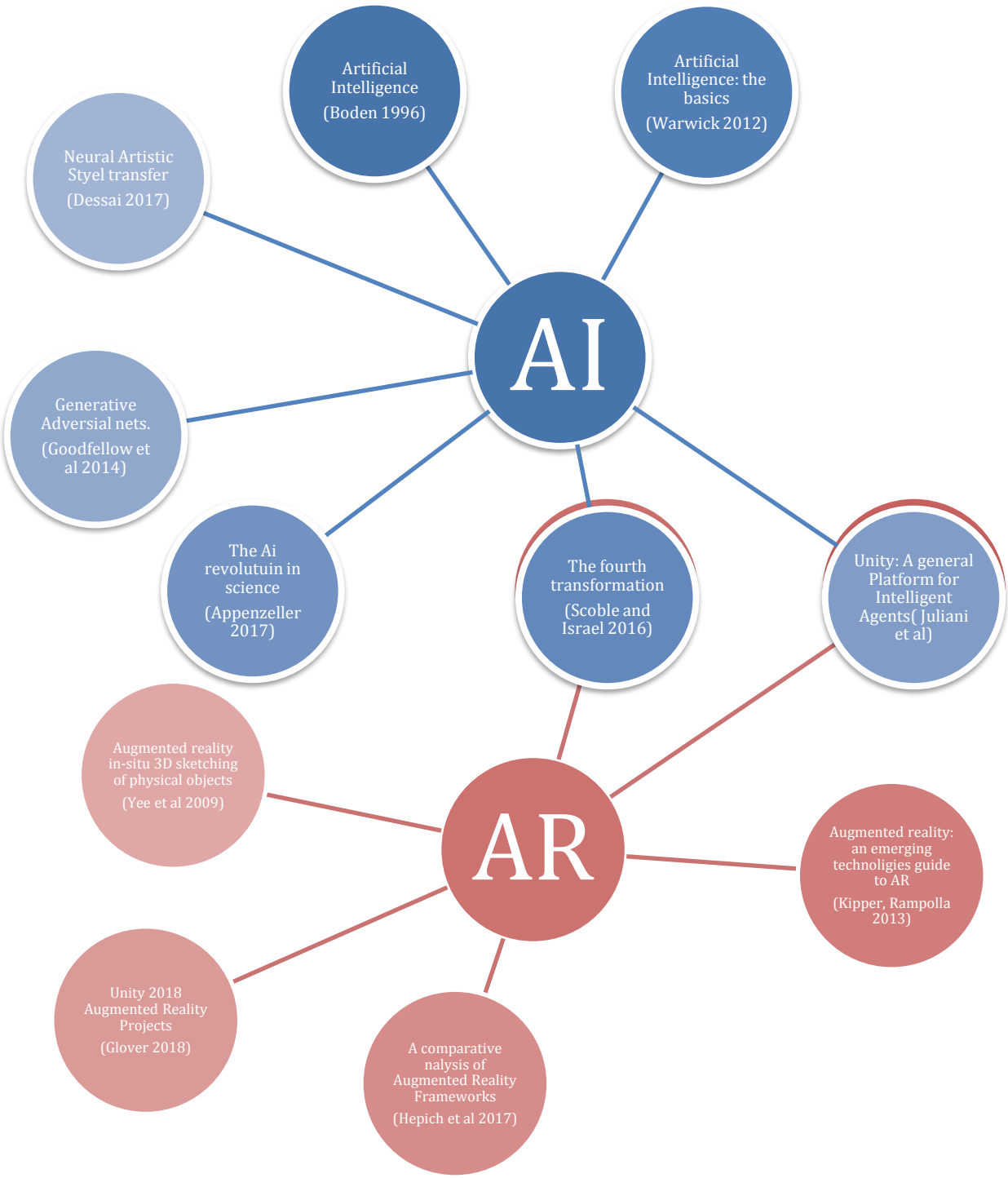


Figure 2: Visualization the sources used

3. Specifications & Requirements

To create an AR Experience, we decided to use a Mobile Device Application, due to the wide spread usage of smart phones in contrast to using HoloLens like devices.
Specifically, we used Android Devices, because they are developer friendly and it fits to the Engineering Mobile Application Unit, which also uses Android, at Solent University.
As Software we used Unity and Android Studio, whereas the latter would be more in line with the studies at Solent.
As tools for Developing, SDKs were used to simplify the process of developing AR. The Vuforia Engine was used to enable easy image and object detection. It can be integrated in both Unity and Android Studio.
For machine learning tools, he Tensorflow library offers tools such as Keras, which can be used to get started with neural networks.
With the use of Unity, the integration of these Python based libraries seemed to be a problem, but Unity Machine Learning Agents Toolkit offers Usage of Python based libraries. (Juliani et al 2018) Also, the TensorflowSharp plugin allows the use of pretrained Tensorflow graphs, so the actual machine learning process could be done separately.

During the implementation phase, further requirements and specifications needed to be done. Because of the high demand in computing power, the hardware for the programs need to be stated.
Mobile device that were used:
Galaxy S7 with 4GB Ram, Quadcore 4x1.15 CPU, Snapdragon Adreno 540 GPU, 12MP rear camera.
Amazon Fire HD 8 Tablet, had performance issues.

PCs being used: Thinkpad edge e545, HP Pavilion 8, Google Cloud Service for Computing

For building machine learning models, Keras was used most of the time instead of just basic Tensorflow. A virtual environment for Python was set up using the Anaconda Python distribution for Windows with Python version 3.7. Some small Python tasks like web scrapping images from Flickr where done with basic Windows Command Tool executing Python Scripts.
To name a few important Python Libraries that needed to be installed:
Pip for installing packages. PIL fork pillow for manipulating and showing image files. NumPy for basic Array functions. Tensorflow and Keras.

3.1 Comparison of Deep Learning Frameworks

In the progress of research and development for the project, Tensorflow was initially singled out as the deep learning framework to use. Mainly because of similar work using Tensorflow as well and being supposedly easy to deploy to an Augmented Reality mobile application. Further into development, the Keras framework was used which sits atop Tensorflow making coding a little bit easier and intuitive. (Hale 2018)
But it is worth looking a bit deeper into deep learning frameworks available and suitable for the project.

- "Theano was developed at the University of Montreal in 2007 and is the oldest significant Python deep learning framework" (Hale 2018)Today it lost a lot of popularity and is updated only rarely. It is low level and compiles rather slowly. While researching for the project it was used with the high-level wrapper Keras. It was a big step stone for further development and is quite similar to Tensorflow. The creators of Theano went on to also work on Tensorflow.

- Tensorflow is by far the most popular deep learning framework right now. Created and maintained by google the framework went open source and is used by many tech companies and individuals. Thus, having rich and detailed documentation and many tutorials available for the community. It also provides tools like TensorBoard for visualization. A big point of criticism is that tensorflow is very low-level and difficult to learn and tedious to debug.
- Keras is built on top of Tensorflow and simplifies many codelines and parameters that need to be written using basic Tensorflow.

4. Methodology

At the initial start, information was gathered of similar state of the art work related to the project. To get a good overview what would be possible and what tools are used. There are projects by private persons and professionals combining AI and Art like Neural Artistic Style Transfer, that transfers famous artistic styles to a regular Photograph. (Desai 2017)
Following this, there was a phase of academic research on these topics to give a more grounded foundation. Since the topics of AR and AI have a broad scale, the research needed to be reduced to specific topics that are beneficial for the project.

The research can be divided into two categories with two subcategories:
Academic books and papers that deal with the general concept of the technologies of a) AR and b) AI, mostly written for persons with no background in computer science.
Development documentations, tutorials, guides and books with algorithms that are intended to give instructions to implement these technologies practically for a) and b).
The former literature could date back any time, since the concepts of AI and AR where introduced decades ago.(Kipper and Rampolla 2013) (Warwick 2012)

The latter literature should to be no older than 5 years. The techniques to use AI have been changing much in the past year (Appenzeller 2017) and AR frameworks are changing rapidly. Using older guides to implement these could be problematic.
To test the initial thesis of developing an AR drawing assisting tool would be tested by creating prototypes and evaluate the functionality. At first the

usability of the product is neglected and just minimal viable product(mvp) should be created.
To train the neural networks with images of art would require extensive user data. It is not very realistic to get enough test subjects to use the AR prototype application. Another approach could be to collect art image data separately with a website that lets users sketch on it.
Data Mining techniques could be used to get art content from the internet (Jiawei Han, Micheline Kamber,Jian Pei. 2011) or training data samples could be used from related work if permitted.

In the developing phase, it was important to get quick results of a functional prototype. This was difficult though with little experience in AI and AR development. With resources available on the internet we tried to learn AI and AR basics and make progress in development at the same time. Tutorials and guide for similar projects can be adapted to fit the needs of our projects. This requires a great deal of work sometimes, but it is easier to do without being an experienced AI developer. This mainly requires transforming data input from our input we got to be the same as the data input for guides and tutorials and extending the program meant to learn the principles to a functional program.

Testing for the project was done with the focus on simple functionality and needed to be fast to apply. At first, we encountered the problem tedious testing and debugging of the mobile app in Unity due to

5. Professional, Legal and Ethical issues

5.1 Legal issues

Unity and Vuforia both are free, but not totally free for commercial use. For Unity there is a limit of 100.000 $ of revenue in the last fiscal year. For Vuforia it is necessary to purchase a one-time license or abonnement to deploy an application. However, the project has an experimental character and serves as a proof of concept rather than a finished product.
To get training data for the neural network, data related to art needs to be collected. These could cause legal authorship problems. The machine learning library from google, TensorFlow, is open source and the itegration to Unity via ML-Agents Toolkit is also open source.

5.2 Ethical issues

Ethically testing is only planned with few participants related to the project and not very concerning. Yet, the topics of Artificial Intelligence and Machine Learning do often raise ethical concerns. But in this project, it is very unlikely that the complexity of Artificial Intelligence could lead to unwanted outcomes that could harm others.

5.3 Legal/professional issues for further development

Ultimately, it could be considered purchasing a license for Unity or Vuforia simply to get rid of the logos that are displayed in the application. If you use a free version of Unity, the logo appears at the launch of your application. This has led to a negative reputation of Unity games, because professional companies have a license and you won't know that their games are made with this game engine. For example, Pokémon Go was made using Unity. A

significant step stone that made Augmented Reality Apps popular to the general public.
One the other Hand, hobbyist game developers or small companies use the free version of Unity, which has led to a negative association with the Unity Logo to subpar games.

Furthermore, the Vuforia Logo appears constantly on the bottom right of the screen, which is unpleasant and hinders development. To get the camera feed of the phone and in the end an image of it for a classification, the Vuforia Camera function was used. This has the advantage of working cross platform on IOS, Android and even on the webcam of a laptop which is very convenient for testing purposes. But the logo also appears on the image using the Vuforia Camera. The edge detection filter also detected the edges of the logo which is problematic for our image saving function. Training a model with pictures of a logo, will distort the results. Even the implementation of the Inception v3 like Image Classification could be distorted, because the images used as input for the classify function also have the Vuforia logo in it.

6. Project Management

6.1 Assessment and discussion of risk and contingency planning

The initial assessment of the project of creating something using AI and Augmented Reality to help artists or create art were a little vague and could lead to many outcomes. Tackling too many areas and underestimating the complexity of the project could lead to no outcomes at all. It is important to rescope the project and evaluate what would be possible and realistic to achieve. Combining the AI and AR in one application could also be a challenge. To get some results it could help to define mini prototypes and goals, that deal with each problem, but still with the idea in mind to combine them together.
To tackle AR drawing:
Creating an application that detects a canvas (markers or drawn symbols could be used to define the edges).
Getting the image data of the canvas in real time (optionally, communicate the data with another program).
Show recommendation in AR to help an artist draw something (could be just one static recommendation)

To tackle AI and Art:
Create image classification of sketches (only on desktop)
Using GANs to create images (first pretrained or with training data from the internet)
Train Neural Networks with our own data (that resamples data we would get from our detected canvas.

Small successes and iterations could help to find solutions for the overall problem and to be more agile in development.

At first, structuring the development process seemed difficult. Multiple features needed to be created in the AI and AR departments as well as the gathering and preparation of data, while research still wasn't finished. We used an agile developing approach with small iterations, since planning the whole project was not possible in the begining.
Inspired by Scrum, a Scrum board for small tasks was used. Results were discussed after small sprints of one week. Although the Scrum concept was only partly realized with only one developer and one product owner involved.

7. Discussion of Design and Implementation

7.1 Implementation of the first image classification AR App prototype

The first prototype was intended as a proof of concept that it is possible to use machine learning image classification in an Augmented Reality App with the specifications that were established.
The machine learning model was trained on the ImageNet Dataset using Keras with the VGG16 model. To use it in Unity with ML-Agents, it needed to be frozen and saved to a file. Since Unity cannot import .h5 files, it needs to be renamed to a .byte file first. Also, the Tensorflow version needed to be the same as in the version of ML-Agents we are using. If this wasn't the right version, we would have needed to open it again and save it with the correct version of Tensorflow.
Getting a result with the Tensorflowsharp plugins runner function required a certain data format as an input. We used Vuforia Camera to get out current front camera of the device. The function getCamTexture() returns a 2D Texture. With the help of the Texturetools helper class we crop the texture to a 224 times 224 pixel texture and convert it as a one dimensional Color32[] array. Color 32 represents the RGBA values of a pixel. Red, Green, Blue and Alpha for the opaqueness ranging from 0 to 255. The pixels are set in the array from left to right, adding row after row from bottom to top. An image with 244 width and 244 height translates to an Array with the length of 244*244 = 59,546. Tensorflow also needs a one-dimensional array as an input but it needed to be converted to fit properly.

We used the Unity Plugin TensorflowSharp by Miguel Deicaza suppling Tensorflow support for the C# language. In his documentation we found the

best practice of using the runner Class. An input needs to be added with runner.addInput() and after fetching with runner.Fetch() we can the results with runner.Run(). More information can be found on the TensorflowSharp documentation. The output returns an array with the possible labels and their probabilities. At first, we only took the highest probability and displayed it using UI Elements of Unity. The classify function was called, every time we have a click input, which translates to a touch on the screen or a mouse click on the game. This allowed us to test the app in Unity playmode with a PC and in the App that was build for Android with Unity.

7.2 Saving images in the right format with an Edge Filter applied

After experimenting with the classification prototype, the problem of identifying the canvas and extracting the data of the artwork came up. Having multiple objects in the picture or unexpected variations would get false results. To neglect different types of art mediums, for example different types of pencils, brushes or canvases, we tried to only use the edges of a picture as input. Likewise, we could also neglect the influence of colour in a picture and focusing of sketches at first. Having the same format of image data for our input of the app and the training of the neural network seemed to be advantageous.

The image filtering was done in Unity to transform the Texture we get from our Camera / Webcam to an edge picture and finally as a byte array of pixel values to use as an input for our TF.Runner function

7.3 Writing our own image classifier for our preprocessed pictures of sketches

A widely used example for machine learning is the classification of hand drawn number digits. In the official Tensorflow guide ,the MNist dataset is used to explain how a basic Classifier is created in Keras. The Dataset can simply be downloaded in the pythonscript, because it is already included in the Tensorflow package. The images of the digits are low resolution grayscale images of 24*24 pixels. Due to the similarity to our image dataset of edge filtered sketches, it seemed possible to use the same model to classify them.

The pictures used where saved with our app in 7.2 as png and jped file. Even though we only used the jpg files, since we have not found evidence that Tensorflow works with 4 channel RGBA pictures. We have drawn pictures of easily recognizable shapes by hand and made about 10 pictures from different angles of them. Our App named them with a counter, but some images were not usable. So, at then end we had imagefiles with random numbers ranging from 0 to 500 for each class. Each class of a shape had to be in its corresponding folder of a train and a test folder for a training and testing set.

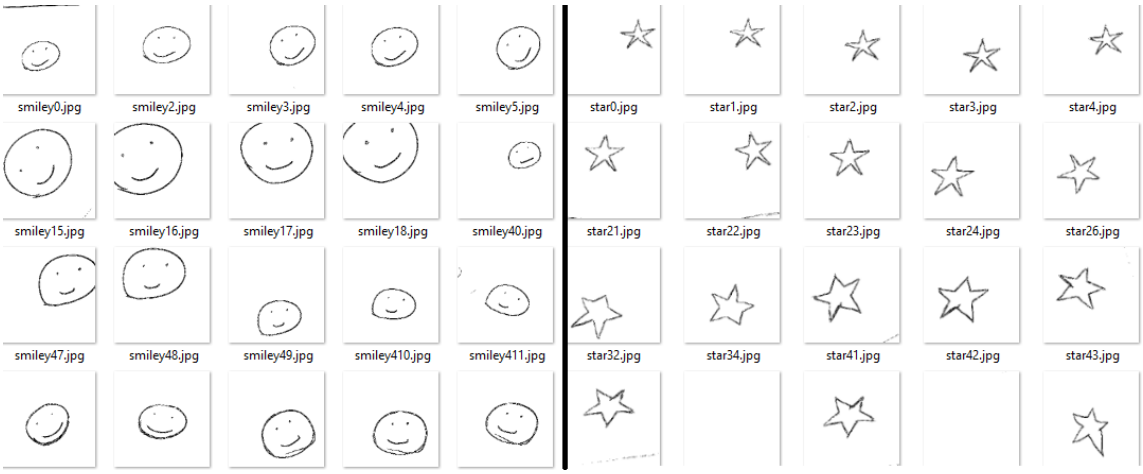
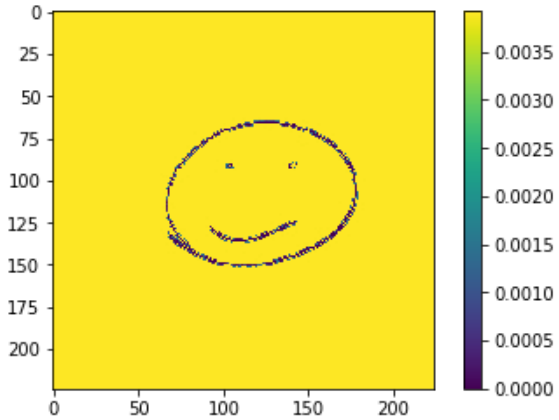


Figure 3:The Dataset of sketch images in their respective folders

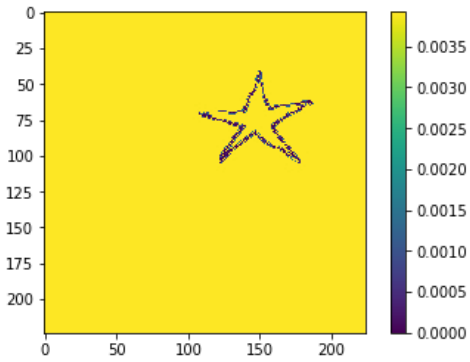
Preparing the data from a file of about 100 images and converting them to an array was a major point to do. The images had to be saved in a NumPy array. This Array also contains NumPy Arrays, which are a one Dimensional representation of our pictures. Some images where rotated and mirrored to augment the number of images used. In the MNIST fashion dataset the pictures of images were arranged in the train array one by one. They had 9 classes, resulting in index 1, 9, 18,… cointaining a shoe for example. The Array for the results was structured in the same pattern. With the number of the class which is linked to a String of the description of it. The array with 9 classes would look like this: [1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,1 …] For our dataset we replicated the same pattern with our classes of “Smiley” “Star” and so on. For every image we had to normalize the rgb pixel values that range from 0-255 to be in a range from 0-1.

```
In [10]: train_images = train_images / 255.0
         test_images = test_images / 255.0

         plt.figure()
         plt.imshow(train_images[3])
         plt.colorbar()
         plt.grid(False)
         plt.show()
```



```
In [13]: test_loss, test_acc = model.evaluate(test_images, test_labels)

         print('Test accuracy:', test_acc)

         12/12 [==============================] - 0s 18ms/sample - loss: 1.1212 - acc: 0.3333
         Test accuracy: 0.33333334
```

```
In [14]: img = test_images[1]

         plt.figure()
         plt.imshow(img)
         plt.colorbar()
         plt.grid(False)
         plt.show()

         predictions = model.predict(test_images)
         predictions[5]
```




```
Out[14]: array([0.00171083, 0.28348085, 0.28042603, 0.43438226], dtype=float32)
```

7.4 Writing a GAN to generate art based on the classifier in 7.2

To write a Generative Adversarial Network, that can generate images from our sketch data, a version of a similar GAN that generated random monster images was modified. In key elements of this program are the generator and the discriminator. In the future we would need to change the discriminator to be the same as our classifier in 7.3. Right now, we only modified that data input to fit the GAN in the form of low resolution sRGB jpeg images. The program would need to run for about 500 hours in the required epochs of 5000 on CPU. Unless we can run it with better Hardware on GPU it can only run for 50 epochs or less resulting in a suboptimal outcome.

7.5 Developing Key words:

Batch: A Batch is a stack of data that should be processed by the machine learning model. If a set data is too big to be processed it has to be split in multiple batches. In our case the number of images needed to classify an object had to be split up in multiple batches.

Model: A model is the form of the neural network. It represents how the neurons are connected in different layers. Every deep learning task needs a different model which needs to be defined at the start with a variety of parameters. In our project we won't build a model ourselves and use an existing one instead.
In a competition to classify images of the ImageNet Dataset a model named VGG16 won the competition which achieved a 7.5% error rate on the top 5 results of large-scale image recognition. We used a similar model for the recognition of the drawn sketches.

Convolutional Neural Network: Convolutional Neural Networks are used to identify objects in images. Convolution is an operation which multiplies an input with a filter very similar to the Sobel operator which is used to detect edges in images. So, in the process of filtering the input image to only show edges might be redundant if we use a convolutional Neural Network, because it applies a sort of edge detection on its own.

8. Results

The Results of the project could be classified as small successes in the field of AR and AI and as a grand result at the end.
The first result was a working implementation of an image classification application that can give an output to the identified objects using Augmented Reality. As stated in the review, the use of two popular AI and AR frameworks Tensorflow and Vuforia and Unity as a Software Environment gave a satisfying result. The Application could identify objects well if no other distracting artefacts are also in the picture. Given that the object was in the ImageNet Database it had an error rate of approximately 20%. For our first expectations that was rather impressive. But the neural network model that our trained model is based on even obtains an error rate of und 10%. This is of course on the ImageNet Dataset it is trained on, which is known in comparison to unknown images our phone makes.
The performance of the mobile app was a bit lackluster. Every time a classification was made a short lag happened. Hence the app only made a classification after a touch on the screen.
Moreover, the app needed to display the result somehow in Augmented Reality to proof the concept of both technologies working together. The easiest way of using AR elements with **Vu**foria is using an image target as a marker. The image needs to be uploaded to the database of Vuforia and integrated to Unity. Then, it can be placed in a scene in Unity and gameobjects could act related to its position. The most basic application would be displaying a 3D cube on top of the image. A more appealing method would be a 3D object that resembles the detected object, like a rotating 3D coin appearing over the image target after a coin was classified. Due to the effort of loading 3D objects in the project for every label of the 20,000 categories of the ImageNet database this is not very

feasible. A simpler solution was to display the labels on the texture of a 3D object like a cube. By definition, this would combine both technologies of AR and AI even though militaristically.

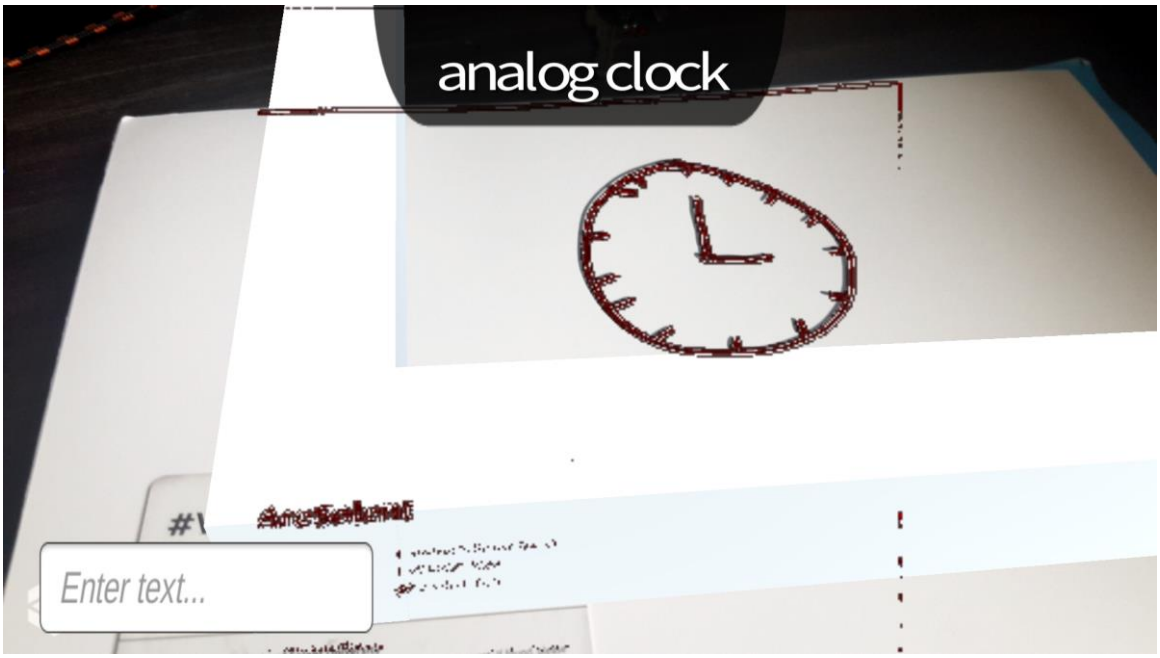


Figure 4: Using the Edge filter and saving an image of a clock that was also classified as a clock

As an additional Augmented Reality feature, we planned to mark the canvas in the real world with objects. This way, the neural network won't get distracted inputs. Not relevant information caught by the camera like the edges of a paper or the wood grain of a table. Marking the canvas could also help calculating the perspective of the canvas relative to the phone. The objects used as markers should be common every day items, that look the same for everyone. Instead of using machine learning to identify objects, Vuforia uses an object scanning method by taking multiple pictures of different angles and mapping them to a 3D object. Machine learning could greatly benefit from using AI object detection. But right now, it is presumably too computationally intensive to use it whilst maintaining a stable frame rate.

For this reason, we used the Vuforia Object Scanner to map recognize common

objects. A target picture for reference needs to be printed out before placing the object on it and circumcising it with the phone camera. The lighting needs to be considered and the properties of the object also. Highly reflective surfaces or repeating patterns are hard to scan. Thus, the initial idea of using a coin as a reference object did not work out.
As another possibility, the letters of the Solent University card could work as a common reference object. Vuforia provides a text detection function, but it works only with a few different font styles. Lastly, a widely used method of using a one-dollar note could work, but with British pound instead. Since a 5 and 10 have transparent or reflective properties the section of Queen Elizabeth's face is best used as a reference.

Figure 5: Using the app with a tripod

To get the most of Vuforia's Augmented Reality features, we plan to use ground plane detection, which can recognize flat horizontal surfaces like a table surface or the floor. This could be beneficial to display AI generated art on top

of a canvas in the right perspective without using reference markers. In contrast to display the art as a simple overlay on the screen.
Additionally, the extended tracking feature of Vuforia, which was a considerate point in choosing this framework could be used to still get an orientation with the help of a reference target. Even after it is not seen by the phone camera anymore.

Attempts to classify sketches with our own training set of self drawn shapes were not very successful. With an error rate of 33%. However, the code compiled without errors.

Generating new sketches started working, but due to the lack of resources it did not produce a satisfying result. To generate better images we would need to run the GAN with 1000 epochs or more. But the program was run with 20 epochs, which already took 2 hours on CPU. The results from generating sketches of eyes only looked like random grain.

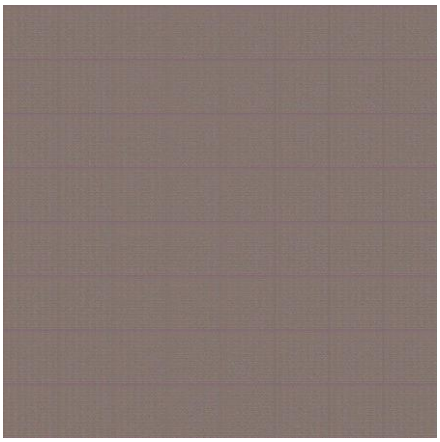


Figure 6: first GAN results

9. Project Progress

9.1 Results and discussion of initial research

At the start, the initial scope of the project was overwhelming. There are many resources of using AI to create art and many tools and libraries to get started. But at first the scope of the project was reduced to develop an application that can recognize a canvas, collect the data from the drawings on it and project guidance for future drawing on this canvas.
Researching Developing Tools for AR and evaluating them resulted in good options. (Herpich, Guarese and Tarouco 2017)

9.2 Assessment and discussion of appropriate guidelines

With the progression of the project, the possible and desired outcomes were sometimes not clear. What exactly should the application do?
Should there be recommendations/ideas showing in AR that will show in a blank canvas, based on other gathered data about the user or other artists users.
Should there be recommendations showing in AR for completing a drawing after the user starts sketching a bit and the AI predicts what he is trying to draw? (Auto Draw?)
Should there be recommendation shown in AR to alter a picture to a specific art style?

With the implementation of the artifact of the project it became unclear how research about the topics of AI and AR is beneficial to the project. How important is a good grasp or complete understanding of these topics? Some

options of implementation need no understanding of the technology, but it definitely helps to understand concepts behind AI and AR.

9.3 Evaluation of techniques, tools and frameworks

The Vuforia Engine offered powerful image and object recognition and seemed easy to implement. It can be integrated in Unity, Android Studio and Visual Studio. Although Android Studio offers more diversity and control in building mobile applications, Unity was easier to build simple apps and testing the capabilities of the Vuforia Engine for the project.
Another great advantage of using Vuforia and Unity is that they are both cross-platform and the resulting app could be deployed in Android or IOS.
Unity also allows to run the program while developing with the use of a webcam that can act as a smartphone camera and allows testing of AR applications without deploying them to a mobile device.

Images that meet specific criteria such as being rich in detail, having good contrast and no repeating patterns can be uploaded to the Vuforia Database and then detected with the application build in Unity after printing them out. Objects can be scanned, using an object scanner application released by Vuforia, and detected as well. Detection of predefined objects seemed to be useful to define the scope and location of a canvas. It still seemed difficult to get dynamic image data of the canvas to analyse.

9.4 Documentation of experimentation/ technical design process

After creating the foundation of an augmented creativity lab, the problem of combining it with machine learning tools emerged. Would there be an easy way to use python-based AI libraries and Unity's c# based scripts together?

Luckily, there is a Project called “Unity Machine Learn Agents” to use machine learning based on Tensorflow in Unity (Juliani *et al*. 2018) It is mainly aimed at training Agents with simulations in Unity that has the advantages of realistic physic and light rendering. With the Unity extension “TensorFlowSharp” related to the Unity ml-agents projects, pre-trained Tensorflow graphs can be used within Unity. So, for the Project, the Ai doesn’t have to be trained with data only from the Unity project itself. But it could also use data from other sources, for example image recognition datasets to identify drawn objects or art style datasets, to transfer art styles. (Desai 2017)

10. Conclusions

Based on the set criteria and specifications in the review report, some of the set goals where achieved and it was established that first prototypes could be built with the discussed tools and frameworks rather easily. It was possible to build an application that shows results AI classification in Augmented Reality and project AI generated images.

But the functionality was lackluster with only a few drawing elements that could be classified and only few results from GAN generated art. Which also could not be dynamically generated with inputs from the phone camera and instead were generated before building the application and appearing as a condition of a classification. However, after more research about GANs has shown, it is currently difficult to deploy a dynamic model on mobile phones. Most projects with GANs are of scientific nature using pytorch which could not be used with our specifications. Moreover, the amount of image data for generation and classification was larger than expected. And even with efforts of web scrapping from image hosting sites, a lot of data preparation needed to be done. In an ideal world, we could gather content from users of our app to create more drawing elements that can be classified and generated. But right now, we have less than 5 elements, which has little use for an artist.

We intended to test our application with actual artists, to see if they could benefit from an AR drawing assistant, which is difficult with the limited functionality. Nevertheless, we could test UI elements and Usability of the concept the was fleshed out at the end.

After all, the project was very research heavy in its nature. Many things were learned about AI, deep learning and AR and even could be translated to

building something with it, despite having nearly no experience in those fields.at the beginning of the project. GANs where especially hard to tackle, as they are very much advanced machine learning models. Also, a great deal of work was about manipulation and preparing image data in python as well as in C# with Unity.

11. Recommendations for further work

In the future, using more advanced concepts of GANs sounds promising. The graphics card company Nvidia made a program called GauGan, in context of their SPADE tool, that can turn scribbles into photorealistic landscapes. Right now, we are not even remotely close to being able to create something like this. We would need to understand the inner workings of a NN better and be able to use the input Vectors of a Network to generate certain features it was trained on. Our adaptation wouldn't need to be photorealistic, but if we could use tell our GAN to create a certain feature, for example a twig of a tree drawing, it would be a huge success. Yet, this could not be deployed to a mobile application, so we would need to use the method with a projector connected to a desktop computer.

Furthermore, improving our current mobile prototype to more functionality is recommended. There are many aspects that could be improved with our current knowledge. The idea of using edge filtered pictures as an input and saving them in the exact same format from the phone to later train our NN with it needs to be improved. Even if the outlines of sketch were looking good after applying the simple edge filter, they were not very precise in detail. This could have let to errors in the NN. The background and the contour still were a bit grainy and had little dots. The edge filter could be improved with an advanced Soble algorithm and Gaußchen blur prior to it.

Likewise, we should make our app for saving images more useable. With better performance and a way to upload saved images directly to a webserver. This could help us to accumulate enough data, if distribute the app just to a few people. Correspondingly our method of webscrapping images from flickr should also be improved. Getting images just with certain keywords as inputs rarely led to satisfying results and the results needed to be filtered by hand, because

often totally misleading images were scraped. In an article on medium, J. Deng describes how to use machine learning to filter images that were from google images, to use them for training in machine learning.(Deng 2019)

Moreover, we need to improve our training scripts and use proper hardware to work with big data in our machine learning models. First attempts only used rather small datasets for testing purposes, which could be compiled with CPU. But later it became evident that this procedure would take too long for bigger datasets. Even generating a picture with a GAN took half a day. It is recommended to use Tensorflow GPU with a decent graphics card or using cloud computing to get enough hardware acceleration. Even Google Colab offers GPU hardware support and could be a solution.

Finally, the tools and frameworks to build the app should be reconsidered. Especially the use of Unity and Vuforia. Although they performed great in building AR elements, they oppose great limitations for deploying AI elements. In the current state we can only use a pretrained frozen tensorflow graph, which returns outputs after receiving inputs. The other functionalities of Unitys ML-Agents is more of game simulation nature and of little use for image processing. It is recommended to use Android Studio for future proof development and making use of Tensorflow Lite, which is adjusted for mobile device AI calculations.

12. Appendices

12.1 Script used to create a simple image classifier after the fashion mnist concept or our edge highlighted skteches:

```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

from PIL import Image
import os


# In[4]:


img = Image.open("./nbs/train/smiley/smiley0.jpg")


# In[86]:


smileyTrainArr = []
counter = 0
for x in range (500):
    exists = os.path.isfile("./nbs/train/smiley/smiley"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/train/smiley/smiley"+str(x) +".jpg").convert('L'
)
        imgArr = np.array(img)
        smileyTrainArr.append(imgArr)
    else:
        counter+=1

print(500-counter)


starTrainArr = []
counter = 0
```

```
for x in range (500):
    exists = os.path.isfile("./nbs/train/star/star"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/train/star/star"+str(x) +".jpg").convert('L')
        imgArr = np.array(img)
        starTrainArr.append(imgArr)
    else:
        counter+=1

for x in range (500):
    exists = os.path.isfile("./nbs/train/star/star"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/train/star/star"+str(x) +".jpg").convert('L')
        imgArr = np.array(img)
        starTrainArr.append(imgArr)
    else:
        counter+=1

print(1000-counter)

heartTrainArr = []
counter = 0
for x in range (500):
    exists = os.path.isfile("./nbs/train/heart/heart"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/train/heart/heart"+str(x) +".jpg").convert('L')

        imgArr = np.array(img)
        heartTrainArr.append(imgArr)
    else:
        counter+=1

print(500-counter)


#test arrays

smileyTestArr = []
counter = 0
for x in range (500):
    exists = os.path.isfile("./nbs/test/smiley/smiley"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/test/smiley/smiley"+str(x) +".jpg").convert('L')

        imgArr = np.array(img)
        smileyTestArr.append(imgArr)
    else:
        counter+=1

print(500-counter)

starTestArr = []
counter = 0
for x in range (500):
    exists = os.path.isfile("./nbs/test/star/star"+str(x) +".jpg")
    if exists:
        img = Image.open("./nbs/test/star/star"+str(x) +".jpg").convert('L')
        imgArr = np.array(img)
               starTestArr.append(imgArr)
```

```
            else:
                counter+=1


        print(500-counter)

        heartTestArr = []
        counter = 0
        for x in range (500):
            exists = os.path.isfile("./nbs/test/heart/heart"+str(x) +".jpg")
            if exists:
                img = Image.open("./nbs/test/heart/heart"+str(x) +".jpg").convert
    ('L')
                imgArr = np.array(img)
                heartTestArr.append(imgArr)
            else:
                counter+=1

        print(500-counter)


        train_images2 = []
        train_labels2 = []

        for x in range (43):
            train_images2.append(smileyTrainArr[x])
            train_images2.append(starTrainArr[x])
            train_images2.append(heartTrainArr[x])
            train_labels2.append(1)
            train_labels2.append(2)
            train_labels2.append(3)

        print(len(train_images2))

        test_images2 = []
        test_labels2 = []

        for x in range (4):
            test_images2.append(smileyTestArr[x])
            test_images2.append(starTestArr[x])
            test_images2.append(heartTestArr[x])
            test_labels2.append(1)
            test_labels2.append(2)
            test_labels2.append(3)

        print(len(test_images2))

        test_images3 = np.array(test_images2)
        test_images3.shape

        train_images3 = np.array(train_images2)
        train_images3.shape

        test_labels3 = np.array(test_labels2)
        test_labels3.shape

        train_labels3 = np.array(train_labels2)
        train_labels3.shape

```

```
        print(train_images2)


        # In[85]:


        img = Image.open("./nbs/train/heart/heart0.jpg").convert('L')
        imgArr = np.array(img)

        print(imgArr[0][1])


        # In[87]:


        fashion_mnist = keras.datasets.fashion_mnist
        (train_images, train_labels), (test_images, test_labels) = fashion_mnist.
    load_data()

        train_labels


        # In[88]:


        train_images = train_images3
        test_images = test_images3

        train_labels = train_labels3
        test_labels = test_labels3


        # In[97]:


        #class_names = ['T-
    shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker'
    , 'Bag', 'Ankle boot']
        class_names = ['Smiley', 'Star', 'Heart']
        train_images.shape
        train_labels


        # In[90]:


        print(train_images)


        # In[91]:


        train_images = train_images / 255.0
        test_images = test_images / 255.0

        plt.figure()
        plt.imshow(train_images[10])
        plt.colorbar()
        plt.grid(False)
```

```
215.        plt.show()
216.
217.
218.        # In[106]:
219.
220.
221.        model = keras.Sequential([
222.            keras.layers.Flatten(input_shape=(224,224)),
223.            keras.layers.Dense(128, activation=tf.nn.relu),
224.            keras.layers.Dense(4, activation=tf.nn.softmax)
225.        ])
226.        model.compile(optimizer='adam',
227.                      loss='sparse_categorical_crossentropy',
228.                      metrics=['accuracy'])
229.        model.fit(train_images, train_labels, epochs=5)
230.
231.
232.        # In[107]:
233.
234.
235.        test_loss, test_acc = model.evaluate(test_images, test_labels)
236.
237.        print('Test accuracy:', test_acc)
238.
239.
240.        # In[116]:
241.
242.
243.        img = test_images[1]
244.
245.        plt.figure()
246.        plt.imshow(img)
247.        plt.colorbar()
248.        plt.grid(False)
249.        plt.show()
250.
251.        predictions = model.predict(test_images)
252.        predictions[5]
```

12.2 First attempts to generate images with GANs from out web scrapped sketches.

It was attempted to make it work in google collab with acces to the google one drive where the pictures where stored. Credit goes to Siraj Raval for demonstrating the usage and https://github.com/moxiegushi for making this work to generate random pokemon. Only few changes were made for our project

```
# -*- coding: utf-8 -*-

"""Copy of gantest2.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1K43TSnFpaWTCbbf7pV3UWDNOWkxPXoU3
"""

"""
Some codes from https://github.com/Newmu/dcgan_code
"""
from __future__ import division
import math
import json
import random
import pprint
import scipy.misc
import numpy as np
from time import gmtime, strftime
from six.moves import xrange

import tensorflow as tf
import tensorflow.contrib.slim as slim

pp = pprint.PrettyPrinter()

get_stddev = lambda x, k_h, k_w: 1/math.sqrt(k_w*k_h*x.get_shape()[-1])

def show_all_variables():
  model_vars = tf.trainable_variables()
  slim.model_analyzer.analyze_vars(model_vars, print_info=True)

def get_image(image_path, input_height, input_width,
              resize_height=64, resize_width=64,
              crop=True, grayscale=False):
  image = imread(image_path, grayscale)
  return transform(image, input_height, input_width,
                   resize_height, resize_width, crop)

def save_images(images, size, image_path):
  return imsave(inverse_transform(images), size, image_path)

def imread(path, grayscale = False):
  if (grayscale):
```

```
    return scipy.misc.imread(path, flatten = True).astype(np.float)
  else:
    return scipy.misc.imread(path).astype(np.float)

def merge_images(images, size):
  return inverse_transform(images)

def merge(images, size):
  h, w = images.shape[1], images.shape[2]
  if (images.shape[3] in (3,4)):
    c = images.shape[3]
    img = np.zeros((h * size[0], w * size[1], c))
    for idx, image in enumerate(images):
      i = idx % size[1]
      j = idx // size[1]
      img[j * h:j * h + h, i * w:i * w + w, :] = image
    return img
  elif images.shape[3]==1:
    img = np.zeros((h * size[0], w * size[1]))
    for idx, image in enumerate(images):
      i = idx % size[1]
      j = idx // size[1]
      img[j * h:j * h + h, i * w:i * w + w] = image[:,:,0]
    return img
  else:
    raise ValueError('in merge(images,size) images parameter '
                     'must have dimensions: HxW or HxWx3 or HxWx4')

def imsave(images, size, path):
  image = np.squeeze(merge(images, size))
  return scipy.misc.imsave(path, image)

def center_crop(x, crop_h, crop_w,
                resize_h=64, resize_w=64):
  if crop_w is None:
    crop_w = crop_h
  h, w = x.shape[:2]
  j = int(round((h - crop_h)/2.))
  i = int(round((w - crop_w)/2.))
  return scipy.misc.imresize(
      x[j:j+crop_h, i:i+crop_w], [resize_h, resize_w])

def transform(image, input_height, input_width,
              resize_height=64, resize_width=64, crop=True):
  if crop:
    cropped_image = center_crop(
      image, input_height, input_width,
      resize_height, resize_width)
  else:
    cropped_image = scipy.misc.imresize(image, [resize_height, resize_width])
  return np.array(cropped_image)/127.5 - 1.

def inverse_transform(images):
  return (images+1.)/2.

      def to_json(output_path, *layers):
        with open(output_path, "w") as layer_f:
          lines = ""
          for w, b, bn in layers:
```

```
104.             layer_idx = w.name.split('/')[0].split('h')[1]
105.
106.             B = b.eval()
107.
108.             if "lin/" in w.name:
109.               W = w.eval()
110.               depth = W.shape[1]
111.             else:
112.               W = np.rollaxis(w.eval(), 2, 0)
113.               depth = W.shape[0]
114.
115.             biases = {"sy": 1, "sx": 1, "depth": depth, "w": ['%.2f' % elem for
    elem in list(B)]}
116.             if bn != None:
117.               gamma = bn.gamma.eval()
118.               beta = bn.beta.eval()
119.
120.               gamma = {"sy": 1, "sx": 1, "depth": depth, "w": ['%.2f' % elem fo
    r elem in list(gamma)]}
121.               beta = {"sy": 1, "sx": 1, "depth": depth, "w": ['%.2f' % elem for
    elem in list(beta)]}
122.             else:
123.               gamma = {"sy": 1, "sx": 1, "depth": 0, "w": []}
124.               beta = {"sy": 1, "sx": 1, "depth": 0, "w": []}
125.
126.             if "lin/" in w.name:
127.               fs = []
128.               for w in W.T:
129.                 fs.append({"sy": 1, "sx": 1, "depth": W.shape[0], "w": ['%.2f'
    % elem for elem in list(w)]})
130.
131.               lines += """
132.                 var layer_%s = {
133.                   "layer_type": "fc",
134.                   "sy": 1, "sx": 1,
135.                   "out_sx": 1, "out_sy": 1,
136.                   "stride": 1, "pad": 0,
137.                   "out_depth": %s, "in_depth": %s,
138.                   "biases": %s,
139.                   "gamma": %s,
140.                   "beta": %s,
141.                   "filters": %s
142.                 };""" % (layer_idx.split('_')[0], W.shape[1], W.shape[0], biase
    s, gamma, beta, fs)
143.             else:
144.               fs = []
145.               for w_ in W:
146.                 fs.append({"sy": 5, "sx": 5, "depth": W.shape[3], "w": ['%.2f'
    % elem for elem in list(w_.flatten())]})
147.
148.               lines += """
149.                 var layer_%s = {
150.                   "layer_type": "deconv",
151.                   "sy": 5, "sx": 5,
152.                   "out_sx": %s, "out_sy": %s,
153.                   "stride": 2, "pad": 1,
154.                   "out_depth": %s, "in_depth": %s,
155.                   "biases": %s,
156.                   "gamma": %s,
```

```
                    "beta": %s,
                    "filters": %s
                };""" % (layer_idx, 2**(int(layer_idx)+2), 2**(int(layer_idx)+2),
                         W.shape[0], W.shape[3], biases, gamma, beta, fs)
            layer_f.write(" ".join(lines.replace("'","").split()))

        def make_gif(images, fname, duration=2, true_image=False):
          import moviepy.editor as mpy

          def make_frame(t):
            try:
              x = images[int(len(images)/duration*t)]
            except:
              x = images[-1]

            if true_image:
              return x.astype(np.uint8)
            else:
              return ((x+1)/2*255).astype(np.uint8)

          clip = mpy.VideoClip(make_frame, duration=duration)
          clip.write_gif(fname, fps = len(images) / duration)

        def visualize(sess, dcgan, config, option):
          image_frame_dim = int(math.ceil(config.batch_size**.5))
          if option == 0:
            z_sample = np.random.uniform(-0.5, 0.5, size=(config.batch_size, dcgan.z_dim))
            samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample})
            save_images(samples, [image_frame_dim, image_frame_dim], './samples/test_%s.png' % strftime("%Y%m%d%H%M%S", gmtime()))
          elif option == 1:
            values = np.arange(0, 1, 1./config.batch_size)
            for idx in xrange(100):
              print(" [*] %d" % idx)
              z_sample = np.zeros([config.batch_size, dcgan.z_dim])
              for kdx, z in enumerate(z_sample):
                z[idx] = values[kdx]

              if config.dataset == "mnist":
                y = np.random.choice(10, config.batch_size)
                y_one_hot = np.zeros((config.batch_size, 10))
                y_one_hot[np.arange(config.batch_size), y] = 1

                samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample, dcgan.y: y_one_hot})
              else:
                samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample})


              save_images(samples, [image_frame_dim, image_frame_dim], './samples/test_arange_%s.png' % (idx))
          elif option == 2:
            values = np.arange(0, 1, 1./config.batch_size)
            for idx in [random.randint(0, 99) for _ in xrange(100)]:
              print(" [*] %d" % idx)
              z = np.random.uniform(-0.2, 0.2, size=(dcgan.z_dim))
              z_sample = np.tile(z, (config.batch_size, 1))
```

```
210.             #z_sample = np.zeros([config.batch_size, dcgan.z_dim])
211.             for kdx, z in enumerate(z_sample):
212.               z[idx] = values[kdx]
213.
214.             if config.dataset == "mnist":
215.               y = np.random.choice(10, config.batch_size)
216.               y_one_hot = np.zeros((config.batch_size, 10))
217.               y_one_hot[np.arange(config.batch_size), y] = 1
218.
219.               samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample, d
    cgan.y: y_one_hot})
220.             else:
221.               samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample})
222.
223.             try:
224.               make_gif(samples, './samples/test_gif_%s.gif' % (idx))
225.             except:
226.               save_images(samples, [image_frame_dim, image_frame_dim], './sampl
    es/test_%s.png' % strftime("%Y%m%d%H%M%S", gmtime()))
227.           elif option == 3:
228.             values = np.arange(0, 1, 1./config.batch_size)
229.             for idx in xrange(100):
230.               print(" [*] %d" % idx)
231.               z_sample = np.zeros([config.batch_size, dcgan.z_dim])
232.               for kdx, z in enumerate(z_sample):
233.                 z[idx] = values[kdx]
234.
235.               samples = sess.run(dcgan.sampler, feed_dict={dcgan.z: z_sample})
236.               make_gif(samples, './samples/test_gif_%s.gif' % (idx))
237.           elif option == 4:
238.             image_set = []
239.             values = np.arange(0, 1, 1./config.batch_size)
240.
241.             for idx in xrange(100):
242.               print(" [*] %d" % idx)
243.               z_sample = np.zeros([config.batch_size, dcgan.z_dim])
244.               for kdx, z in enumerate(z_sample): z[idx] = values[kdx]
245.
246.               image_set.append(sess.run(dcgan.sampler, feed_dict={dcgan.z: z_samp
    le}))
247.               make_gif(image_set[-1], './samples/test_gif_%s.gif' % (idx))
248.
249.             new_image_set = [merge(np.array([images[idx] for images in image_set]
    ), [10, 10]) \
250.                 for idx in range(64) + range(63, -1, -1)]
251.             make_gif(new_image_set, './samples/test_gif_merged.gif', duration=8)
252.
253.       from __future__ import absolute_import, division, print_function, unicode
    _literals
254.
255.       import tensorflow as tf
256.       device_name = tf.test.gpu_device_name()
257.       if device_name != '/device:GPU:0':
258.         raise SystemError('GPU device not found')
259.       print('Found GPU at: {}'.format(device_name))
260.
261.       # Helper libraries
```

```
262.        import numpy as np
263.        import matplotlib.pyplot as plt
264.        import cv2
265.        import os
266.        import scipy.misc
267.        from utils import *
268.
269.        from google.colab import drive
270.        drive.mount('/content/gdrive')
271.
272.        with open('/content/gdrive/My Drive/imagedata/foo.txt', 'w') as f:
273.          f.write('Hello Google Drive!')
274.        !cat /content/gdrive/My\ Drive/foo.txt
275.
276.        img = cv2.imread('/content/gdrive/My Drive/imagedata/foo.txt',0)
277.
278.        !pip install utils
279.
280.        HEIGHT, WIDTH, CHANNEL = 128, 128, 3
281.        BATCH_SIZE = 64
282.        EPOCH = 20
283.        version = 'newSketch'
284.        newSketch_path = './' + version
285.
286.        def lrelu(x, n, leak=0.2):
287.            return tf.maximum(x, leak * x, name=n)
288.
289.        def process_data():
290.            current_dir = os.getcwd()
291.            # parent = os.path.dirname(current_dir)
292.            sketch_dir = os.path.join(current_dir, 'gdrive/My Drive/imagedata')
293.            images = []
294.            for each in os.listdir(sketch_dir):
295.                images.append(os.path.join(sketch_dir,each))
296.            # print images
297.            all_images = tf.convert_to_tensor(images, dtype = tf.string)
298.
299.            images_queue = tf.train.slice_input_producer(
300.                                                [all_images])
301.
302.            content = tf.read_file(images_queue[0])
303.            image = tf.image.decode_jpeg(content, channels = CHANNEL)
304.            # sess1 = tf.Session()
305.            # print sess1.run(image)
306.            image = tf.image.random_flip_left_right(image)
307.            image = tf.image.random_brightness(image, max_delta = 0.1)
308.            image = tf.image.random_contrast(image, lower = 0.9, upper = 1.1)
309.            # noise = tf.Variable(tf.truncated_normal(shape = [HEIGHT,WIDTH,CHANN
    EL], dtype = tf.float32, stddev = 1e-3, name = 'noise'))
310.            # print image.get_shape()
311.            size = [HEIGHT, WIDTH]
312.            image = tf.image.resize_images(image, size)
313.            image.set_shape([HEIGHT,WIDTH,CHANNEL])
314.            # image = image + noise
315.            # image = tf.transpose(image, perm=[2, 0, 1])
316.            # print image.get_shape()
317.
318.            image = tf.cast(image, tf.float32)
319.            image = image / 255.0
```

```
320.
321.           iamges_batch = tf.train.shuffle_batch(
322.                                               [image], batch_size = BATCH_SIZE,
323.                                               num_threads = 4, capacity = 200 + 3*
    BATCH_SIZE,
324.                                               min_after_dequeue = 200)
325.           num_images = len(images)
326.
327.           return iamges_batch, num_images
328.
329.       def generator(input, random_dim, is_train, reuse=False):
330.           c4, c8, c16, c32, c64 = 512, 256, 128, 64, 32 # channel num
331.           s4 = 4
332.           output_dim = CHANNEL  # RGB image
333.           with tf.variable_scope('gen') as scope:
334.               if reuse:
335.                   scope.reuse_variables()
336.               w1 = tf.get_variable('w1', shape=[random_dim, s4 * s4 * c4], dtyp
    e=tf.float32,
337.                                    initializer=tf.truncated_normal_initializer(
    stddev=0.02))
338.               b1 = tf.get_variable('b1', shape=[c4 * s4 * s4], dtype=tf.float32
    ,
339.                                    initializer=tf.constant_initializer(0.0))
340.               flat_conv1 = tf.add(tf.matmul(input, w1), b1, name='flat_conv1')

341.                #Convolution, bias, activation, repeat!
342.               conv1 = tf.reshape(flat_conv1, shape=[-
    1, s4, s4, c4], name='conv1')
343.               bn1 = tf.contrib.layers.batch_norm(conv1, is_training=is_train, e
    psilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn1')
344.               act1 = tf.nn.relu(bn1, name='act1')
345.               # 8*8*256
346.               #Convolution, bias, activation, repeat!
347.               conv2 = tf.layers.conv2d_transpose(act1, c8, kernel_size=[5, 5],
    strides=[2, 2], padding="SAME",
348.                                                  kernel_initializer=tf.truncate
    d_normal_initializer(stddev=0.02),
349.                                                  name='conv2')
350.               bn2 = tf.contrib.layers.batch_norm(conv2, is_training=is_train, e
    psilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn2')
351.               act2 = tf.nn.relu(bn2, name='act2')
352.               # 16*16*128
353.               conv3 = tf.layers.conv2d_transpose(act2, c16, kernel_size=[5, 5],
     strides=[2, 2], padding="SAME",
354.                                                  kernel_initializer=tf.truncate
    d_normal_initializer(stddev=0.02),
355.                                                  name='conv3')
356.               bn3 = tf.contrib.layers.batch_norm(conv3, is_training=is_train, e
    psilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn3')
357.               act3 = tf.nn.relu(bn3, name='act3')
358.               # 32*32*64
359.               conv4 = tf.layers.conv2d_transpose(act3, c32, kernel_size=[5, 5],
     strides=[2, 2], padding="SAME",
360.                                                  kernel_initializer=tf.truncate
    d_normal_initializer(stddev=0.02),
361.                                                  name='conv4')
362.               bn4 = tf.contrib.layers.batch_norm(conv4, is_training=is_train, e
    psilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn4')
```

```
363.                 act4 = tf.nn.relu(bn4, name='act4')
364.                 # 64*64*32
365.                 conv5 = tf.layers.conv2d_transpose(act4, c64, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
366.                                                    kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
367.                                                    name='conv5')
368.                 bn5 = tf.contrib.layers.batch_norm(conv5, is_training=is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn5')
369.                 act5 = tf.nn.relu(bn5, name='act5')
370.
371.                 #128*128*3
372.                 conv6 = tf.layers.conv2d_transpose(act5, output_dim, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
373.                                                    kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
374.                                                    name='conv6')
375.                 # bn6 = tf.contrib.layers.batch_norm(conv6, is_training=is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn6')
376.                 act6 = tf.nn.tanh(conv6, name='act6')
377.                 return act6
378.
379.         def discriminator(input, is_train, reuse=False):
380.             c2, c4, c8, c16 = 64, 128, 256, 512  # channel num: 64, 128, 256, 512
381.             with tf.variable_scope('dis') as scope:
382.                 if reuse:
383.                     scope.reuse_variables()
384.
385.                 #Convolution, activation, bias, repeat!
386.                 conv1 = tf.layers.conv2d(input, c2, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
387.                                                    kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
388.                                                    name='conv1')
389.                 bn1 = tf.contrib.layers.batch_norm(conv1, is_training = is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope = 'bn1')
390.                 act1 = lrelu(conv1, n='act1')
391.                  #Convolution, activation, bias, repeat!
392.                 conv2 = tf.layers.conv2d(act1, c4, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
393.                                                    kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
394.                                                    name='conv2')
395.                 bn2 = tf.contrib.layers.batch_norm(conv2, is_training=is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn2')
396.                 act2 = lrelu(bn2, n='act2')
397.                 #Convolution, activation, bias, repeat!
398.                 conv3 = tf.layers.conv2d(act2, c8, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
399.                                                    kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
400.                                                    name='conv3')
401.                 bn3 = tf.contrib.layers.batch_norm(conv3, is_training=is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn3')
402.                 act3 = lrelu(bn3, n='act3')
403.                  #Convolution, activation, bias, repeat!
404.                 conv4 = tf.layers.conv2d(act3, c16, kernel_size=[5, 5], strides=[2, 2], padding="SAME",
```

```
                                        kernel_initializer=tf.truncated_normal_initializer(stddev=0.02),
                                        name='conv4')
                bn4 = tf.contrib.layers.batch_norm(conv4, is_training=is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn4')
                act4 = lrelu(bn4, n='act4')

                # start from act4
                dim = int(np.prod(act4.get_shape()[1:]))
                fc1 = tf.reshape(act4, shape=[-1, dim], name='fc1')


                w2 = tf.get_variable('w2', shape=[fc1.shape[-1], 1], dtype=tf.float32,
                                     initializer=tf.truncated_normal_initializer(stddev=0.02))
                b2 = tf.get_variable('b2', shape=[1], dtype=tf.float32,
                                     initializer=tf.constant_initializer(0.0))

                # wgan just get rid of the sigmoid
                logits = tf.add(tf.matmul(fc1, w2), b2, name='logits')
                # dcgan
                acted_out = tf.nn.sigmoid(logits)
                return logits #, acted_out

        def train():
            random_dim = 200

            with tf.variable_scope('input'):
                #real and fake image placholders
                real_image = tf.placeholder(tf.float32, shape = [None, HEIGHT, WIDTH, CHANNEL], name='real_image')
                random_input = tf.placeholder(tf.float32, shape=[None, random_dim], name='rand_input')
                is_train = tf.placeholder(tf.bool, name='is_train')

            # wgan
            fake_image = generator(random_input, random_dim, is_train)

            real_result = discriminator(real_image, is_train)
            fake_result = discriminator(fake_image, is_train, reuse=True)

            d_loss = tf.reduce_mean(fake_result) - tf.reduce_mean(real_result)  # This optimizes the discriminator.
            g_loss = -tf.reduce_mean(fake_result)  # This optimizes the generator.


            t_vars = tf.trainable_variables()
            d_vars = [var for var in t_vars if 'dis' in var.name]
            g_vars = [var for var in t_vars if 'gen' in var.name]
            trainer_d = tf.train.RMSPropOptimizer(learning_rate=2e-4).minimize(d_loss, var_list=d_vars)
            trainer_g = tf.train.RMSPropOptimizer(learning_rate=2e-4).minimize(g_loss, var_list=g_vars)
            # clip discriminator weights
            d_clip = [v.assign(tf.clip_by_value(v, -0.01, 0.01)) for v in d_vars]

```

```
453.
454.             batch_size = BATCH_SIZE
455.             image_batch, samples_num = process_data()
456.
457.             batch_num = int(samples_num / batch_size)
458.             total_batch = 0
459.             sess = tf.Session()
460.             saver = tf.train.Saver()
461.             sess.run(tf.global_variables_initializer())
462.             sess.run(tf.local_variables_initializer())
463.             # continue training
464.             save_path = saver.save(sess, "/tmp/model.ckpt")
465.             ckpt = tf.train.latest_checkpoint('./model/' + version)
466.             saver.restore(sess, save_path)
467.             coord = tf.train.Coordinator()
468.             threads = tf.train.start_queue_runners(sess=sess, coord=coord)
469.
470.             print('total training sample num:%d' % samples_num)
471.             print('batch size: %d, batch num per epoch: %d, epoch num: %d' % (bat
    ch_size, batch_num, EPOCH))
472.             print('start training...')
473.             for i in range(EPOCH):
474.                 print("Running epoch {}/{}...".format(i, EPOCH))
475.                 for j in range(batch_num):
476.                     print(j)
477.                     d_iters = 5
478.                     g_iters = 1
479.
480.                     train_noise = np.random.uniform(-
    1.0, 1.0, size=[batch_size, random_dim]).astype(np.float32)
481.                     for k in range(d_iters):
482.                         print(k)
483.                         train_image = sess.run(image_batch)
484.                         #wgan clip weights
485.                         sess.run(d_clip)
486.
487.                         # Update the discriminator
488.                         _, dLoss = sess.run([trainer_d, d_loss],
489.                                             feed_dict={random_input: train_noise,
     real_image: train_image, is_train: True})
490.
491.                     # Update the generator
492.                     for k in range(g_iters):
493.                         # train_noise = np.random.uniform(-
    1.0, 1.0, size=[batch_size, random_dim]).astype(np.float32)
494.                         _, gLoss = sess.run([trainer_g, g_loss],
495.                                             feed_dict={random_input: train_noise,
     is_train: True})
496.
497.                     # print 'train:[%d/%d],d_loss:%f,g_loss:%f' % (i, j, dLoss, g
    Loss)
498.
499.                 # save check point every 500 epoch
500.                 if i%500 == 0:
501.                     if not os.path.exists('./model/' + version):
502.                         os.makedirs('./model/' + version)
503.                     saver.save(sess, './model/' +version + '/' + str(i))
504.                 if i%50 == 0:
505.                     # save images
```

```
506.                     if not os.path.exists(newSketch_path):
507.                         os.makedirs(newSketch_path)
508.                     sample_noise = np.random.uniform(-
    1.0, 1.0, size=[batch_size, random_dim]).astype(np.float32)
509.                     imgtest = sess.run(fake_image, feed_dict={random_input: sampl
    e_noise, is_train: False})
510.                     # imgtest = imgtest * 255.0
511.                     # imgtest.astype(np.uint8)
512.                     save_images(imgtest, [8,8] ,newSketch_path + '/epoch' + str(i
    ) + '.jpg')
513.
514.                     print('train:[%d],d_loss:%f,g_loss:%f' % (i, dLoss, gLoss))
515.             coord.request_stop()
516.             coord.join(threads)
517.
518.         def imsave(images, size, path):
519.           image = np.squeeze(merge(images, size))
520.           return scipy.misc.imsave(path, image)
521.
522.         def save_images(images, size, image_path):
523.           return imsave(inverse_transform(images), size, image_path)
524.
525.         current_dir = os.getcwd()
526.         sketch_dir = os.path.join(current_dir, 'gdrive/My Drive/imagedata')
527.         print(sketch_dir)
528.
529.         train()
```

12.3 Simple webscrapper for Flickr to search by keywords:

```
1. from __future__ import print_function
2. import time
3. import sys
4. import json
5. import re
6. import os
7. import requests
8. from tqdm import tqdm
9. from bs4 import BeautifulSoup
10.
11. KEY = '5ec2c0f582971f49a34002fc4f3f3bbc'
12. SECRET = '6d11ed03fa867bf2'
13.
14. def download_file(url, local_filename):
15.     if local_filename is None:
16.         local_filename = url.split('/')[-1]
17.     r = requests.get(url, stream=True)
18.     with open(local_filename, 'wb') as f:
19.         for chunk in r.iter_content(chunk_size=1024):
20.             if chunk:
21.                 f.write(chunk)
22.     return local_filename
23.
24.
25. def get_photos(q, page=1):
26.     params = {
27.         'content_type': '7',
28.         'per_page': '500',
29.         'media': 'photos',
30.         'method': 'flickr.photos.search',
31.         'format': 'json',
32.         'advanced': 1,
33.         'nojsoncallback': 1,
34.         'extras': 'media,realname,url_l,o_dims,geo,tags,machine_tags,date_taken'
    ,#url_c,url_l,url_m,url_n,url_q,url_s,url_sq,url_t,url_z',
35.         'page': page,
36.         'text': q,
37.         'api_key': KEY,
38.     }
39.
40.
41.     results = requests.get('https://api.flickr.com/services/rest', params=params
    ).json()['photos']
42.     return results
43.
44.
45. def search(q, max_pages=None):
46.     # create a folder for the query if it does not exist
47.     foldername = os.path.join('images', re.sub(r'[\W]', '_', q))
48.
49.     if not os.path.exists(foldername):
50.         os.makedirs(foldername)
```

```
51.
52.     jsonfilename = os.path.join(foldername, 'results.json')
53.
54.     if not os.path.exists(jsonfilename):
55.
56.         # save results as a json file
57.         photos = []
58.         current_page = 1
59.
60.         results = get_photos(q, page=current_page
61.
62.         total_pages = results['pages']
63.         if max_pages is not None and total_pages > max_pages:
64.             total_pages = max_pages
65.
66.         photos += results['photo']
67.
68.         while current_page < total_pages:
69.             print('downloading metadata, page {} of {}'.format(current_page, tot
    al_pages))
70.             current_page += 1
71.             photos += get_photos(q, page=current_page)['photo']
72.             time.sleep(0.5)
73.
74.         with open(jsonfilename, 'w') as outfile:
75.             json.dump(photos, outfile)
76.
77.     else:
78.         with open(jsonfilename, 'r') as infile:
79.             photos = json.load(infile)
80.
81.     # download images
82.     print('Downloading images')
83.     for photo in tqdm(photos):
84.         try:
85.             url = photo.get('url_l')
86.             extension = url.split('.')[-1]
87.             localname = os.path.join(foldername, '{}.{}'.format(photo['id'], ext
    ension))
88.             if not os.path.exists(localname):
89.                 download_file(url, localname)
90.         except:
91.             continue
92.
93.
94. if __name__ == '__main__':
95.     import argparse
96.     parser = argparse.ArgumentParser(description='Download images from flickr')
97.     parser.add_argument('--search', '-
    s', dest='q', required=True, help='Search term')
98.     parser.add_argument('--max-pages', '-
    m', dest='max_pages', required=False, help='Max pages (default none)')
99.     args = parser.parse_args()
100.
101.         q = args.q
102.
103.         print('Searching for {}'.format(q))
104.
```

```
105.            max_pages = None
106.            if args.max_pages:
107.                max_pages = int(args.max_pages)
108.
109.            search(q, max_pages)
```

12.4 Unity scripts for classifying and edge filtering

The Camera Class to get and Image Texture from the Camerafeed from Vuforia for cross platform usage

```
1.  using UnityEngine;
2.  using Vuforia;
3.
4.  public class CameraFeedBehavior : MonoBehaviour {
5.
6.      private Renderer rend;
7.      private Vector3 videoTexSize = Vector3.one;
8.      private Vuforia.Image image;
9.
10.     #region PRIVATE_MEMBERS
11.     private Vuforia.Image.PIXEL_FORMAT mPixelFormat = Vuforia.Image.PIXEL_FORMAT
    .UNKNOWN_FORMAT;
12.     private bool mAccessCameraImage = true;
13.     private bool mFormatRegistered = false;
14.     #endregion // PRIVATE_MEMBERS
15.
16.     #region MONOBEHAVIOUR_METHODS
17.
18.     void Start() {
19.
20.         rend = GetComponent<Renderer>();
21.
22. #if UNITY_EDITOR
23.         mPixelFormat = Vuforia.Image.PIXEL_FORMAT.RGBA8888;
24. #else
25.         mPixelFormat = Vuforia.Image.PIXEL_FORMAT.RGB888; // Use RGB888 for mobi
    le
26. #endif
27.
28.         // Register Vuforia life-cycle callbacks:
29.         VuforiaARController.Instance.RegisterVuforiaStartedCallback(OnVuforiaSta
    rted);
30.         VuforiaARController.Instance.RegisterTrackablesUpdatedCallback(OnTrackab
    lesUpdated);
31.         VuforiaARController.Instance.RegisterOnPauseCallback(OnPause);
32.
33.     }
34.
35.     #endregion // MONOBEHAVIOUR_METHODS
36.
37.     #region PRIVATE_METHODS
38.
39.     void OnVuforiaStarted() {
40.
41.         // Try register camera image format
42.         if (CameraDevice.Instance.SetFrameFormat(mPixelFormat, true)) {
43.             Debug.Log("Successfully registered pixel format " + mPixelFormat.ToS
    tring());
```

```
44.         mFormatRegistered = true;
45.         } else {
46.             Debug.LogError(
47.                 "\nFailed to register pixel format: " + mPixelFormat.ToString()
    +
48.                 "\nThe format may be unsupported by your device." +
49.                 "\nConsider using a different pixel format.\n");
50.
51.             mFormatRegistered = false;
52.         }
53.
54.     }
55.
56.     /// <summary>
57.     /// Called each time the Vuforia state is updated
58.     /// </summary>
59.     void OnTrackablesUpdated() {
60.         if (mFormatRegistered) {
61.             if (mAccessCameraImage) {
62.                 //get camera image
63.                 image = CameraDevice.Instance.GetCameraImage(mPixelFormat);
64.             }
65.         }
66.     }
67.
68.     /// <summary>
69.     /// Called when app is paused / resumed
70.     /// </summary>
71.     void OnPause(bool paused) {
72.         if (paused) {
73.             Debug.Log("App was paused");
74.             UnregisterFormat();
75.         } else {
76.             Debug.Log("App was resumed");
77.             RegisterFormat();
78.         }
79.     }
80.
81.     /// <summary>
82.     /// Register the camera pixel format
83.     /// </summary>
84.     void RegisterFormat() {
85.         if (CameraDevice.Instance.SetFrameFormat(mPixelFormat, true)) {
86.             Debug.Log("Successfully registered camera pixel format " + mPixelFor
    mat.ToString());
87.             mFormatRegistered = true;
88.         } else {
89.             Debug.LogError("Failed to register camera pixel format " + mPixelFor
    mat.ToString());
90.             mFormatRegistered = false;
91.         }
92.     }
93.
94.     /// <summary>
95.     /// Unregister the camera pixel format (e.g. call this when app is paused)
96.     /// </summary>
97.     void UnregisterFormat() {
98.         Debug.Log("Unregistering camera pixel format " + mPixelFormat.ToString()
    );
```

```
99.         CameraDevice.Instance.SetFrameFormat(mPixelFormat, false);
100.                mFormatRegistered = false;
101.            }
102.
103.            public Color32[] GetImage(){
104.                Texture2D camTex = new Texture2D(image.Width, image.Height);
105.                //copy to texture
106.                image.CopyToTexture (camTex);
107.                //crop
108.                var cropped = TextureTools.CropTexture (camTex);
109.                //scale
110.                var scaled = TextureTools.scaled (cropped, 224, 224, FilterMode.B
    ilinear);
111.                //return scaled color32[]
112.                return scaled.GetPixels32();
113.            }
114.
115.            public Texture2D GetImageTexture()
116.            {
117.                Texture2D camTex = new Texture2D(image.Width, image.Height);
118.                image.CopyToTexture(camTex);
119.                return camTex;
120.            }
121.
122.        #endregion //PRIVATE_METHODS
123.        }
```

The Classification Class that uses Tensorflowsharp to get a prediction from our frozen TF Graph credit to Matthew Hallberg

```
1.  using UnityEngine;
2.  using TensorFlow;
3.  using System.Linq;
4.
5.  public class Classification : MonoBehaviour {
6.
7.      [Header("Constants")]
8.      private const int INPUT_SIZE = 224;
9.      private const int IMAGE_MEAN = 117;
10.     private const float IMAGE_STD = 1;
11.     private const string INPUT_TENSOR = "input";
12.     private const string OUTPUT_TENSOR = "output";
13.
14.     [Header("Inspector Stuff")]
15.     public CameraFeedBehavior camFeed;
16.     public TextAsset labelMap;
17.     public TextAsset model;
18.     public MessageBehavior messageBehavior;
19.
20.     private TFGraph graph;
21.     private TFSession session;
22.     private string [] labels;
23.
24.     // Use this for initialization
25.     void Start() {
26. #if UNITY_ANDROID && !UNITY_EDITOR
27.         TensorFlowSharp.Android.NativeBinding.Init();
28. #endif
29.         //load labels into string array
30.         labels = labelMap.ToString ().Split ('\n');
31.         //load graph
32.         graph = new TFGraph ();
33.         graph.Import (model.bytes);
34.         session = new TFSession (graph);
35.     }
36.
37.     private void Update () {
38.         //process image on click or touch
39.         if (Input.GetMouseButtonDown(0)){
40.             ProcessImage ();
41.         }
42.     }
43.
44.     void ProcessImage(){
45.         Debug.Log("click classifi");
46.         //pass in input tensor
47.         var tensor = TransformInput (camFeed.GetImage (), INPUT_SIZE, INPUT_SIZE
    );
48.         var runner = session.GetRunner();
49.         runner.AddInput (graph [INPUT_TENSOR] [0], tensor).Fetch (graph [OUTPUT_
    TENSOR] [0]);
50.         var output = runner.Run();
```

```
51.         //put results into one dimensional array
52.         float[] probs = ((float [] [])output[0].GetValue (jagged: true)) [0];
53.         //get max value of probabilities and find its associated label index
54.         float maxValue = probs.Max ();
55.         int maxIndex = probs.ToList ().IndexOf (maxValue);
56.         //print label with highest probability
57.         string label = labels [maxIndex];
58.         print (label);
59.         messageBehavior.ShowMessage (label);
60.     }
61.
62.     //credit to https://github.com/Syn-McJ/TFClassify-
    Unity for transforming to the right tensor input
63.     public static TFTensor TransformInput (Color32 [] pic, int width, int height
    ) {
64.         float [] floatValues = new float [width * height * 3];
65.
66.         for (int i = 0; i < pic.Length; ++i) {
67.             var color = pic [i];
68.
69.             floatValues [i * 3 + 0] = (color.r - IMAGE_MEAN) / IMAGE_STD;
70.             floatValues [i * 3 + 1] = (color.g - IMAGE_MEAN) / IMAGE_STD;
71.             floatValues [i * 3 + 2] = (color.b - IMAGE_MEAN) / IMAGE_STD;
72.         }
73.
74.         TFShape shape = new TFShape (1, width, height, 3);
75.
76.         return TFTensor.FromBuffer (shape, floatValues, 0, floatValues.Length);

77.     }
78. }
```

The Unity Class to apply a simple Edge Filter to our image and save it to a file. Also showing it as an overlay in red over our current Camera Screen.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using UnityEngine.UI;
5.  using System.Linq;
6.  using System;
7.  using Vuforia;
8.  using System.IO;
9.
10. public class ImageFilter : MonoBehaviour {
11.
12.     Texture2D camTextureSaved;
13.     Texture2D camTexture;
14.     Texture2D destTex;
15.     Texture2D saveToFileTex;
16.
17.     public CameraFeedBehavior camFeed;
18.     public RawImage background;
19.     public AspectRatioFitter fit;
20.     public InputField fileNameField;
21.     public MessageBehavior messageBehavior;
22.
23.     private int imageHeight;
24.     private int imageWidth;
25.     private double timeCounter;
26.     private int imageNameCounter;
27.     private string fileName = "Edge";
28.
29.
30.     // Use this for initialization
31.     void Start () {
32.         background.texture = camFeed.GetImageTexture();
33.         timeCounter = 0;
34.         imageNameCounter = 0;
35.     }
36.
37.     void ProcessImage()
38.     {
39.         camTexture = camFeed.GetImageTexture();
40.         imageHeight = Convert.ToInt32(camTexture.height);
41.         imageWidth = Convert.ToInt32(camTexture.width);
42.         //getting right pxiel ratios
43.
44.         //convert to two D pixel array
45.         Color32[,] twoDImageData = GetTwoDColor(camTexture, imageWidth,imageHeig
    ht);
46.
47.         // apply edge filter
48.         Color32[,] edgeImageData2 = EdgeFilter(twoDImageData, imageWidth, imageH
    eight);
49.
50.
51.         //convert back to one D pixel array
```

```
52.         Color32[] oneDImageData = GetOneDColor(edgeImageData2, imageWidth, image
    Height);
53.
54.         destTex = new Texture2D(imageWidth, imageHeight);
55.         destTex.SetPixels32(oneDImageData);
56.         destTex.Apply();
57.
58.
59.
60.         // Set the current object's texture to show the
61.         // rotated image.
62.         background.texture = (Texture)destTex;
63.
64.     }
65.
66.     Color32[,] EdgeFilter(Color32[,] twoDImagearray, int imageWidth, int imageHe
    ight)
67.     {
68.
69.        Double[,] intensity = new Double[imageHeight, imageWidth];
70.        Color32[,] imageData = new Color32[imageHeight, imageWidth];
71.
72.
73.
74.         //making the image to a gryscale image
75.         for (int h = 0; h < imageHeight; h++)
76.             for (int w = 0; w < imageWidth; w++)
77.             {
78.                 intensity[h, w] = 0;
79.                 intensity[h, w] += twoDImagearray[h, w].r * 0.2126;
80.                 intensity[h, w] += twoDImagearray[h, w].g * 0.7152;
81.                 intensity[h, w] += twoDImagearray[h, w].b * 0.0722;
82.
83.             }
84.
85.         //blurring
86.        // intensity = SimpleBlurGreyScale(intensity, 1);
87.
88.
89.         for (int h = 1; h < imageHeight - 1; h++) {
90.                 for (int w = 1; w < imageWidth - 1; w++)
91.                 {
92.                 // Substracts the intensity values from 4 pixels around the
93.                 // pixel from the the current pixel times 4.
94.                 // If they differ a lot from the current pixel there is and edge

95.                 // and the current pixel is set to light gray.
96.                 // The result has to be an absolute value. For example, if the
97.                 // current pixel is lighter,
98.                 // than the average of the surrounding pixels the result will be

99.                 // a negative value, which can't be an RGB value.
100.                            try
101.                            {
102.                                imageData[h, w].r = Convert.ToByte(Math.Min(255,M
    ath.Abs(4 * intensity[h, w]
103.                                        - (intensity[h + 1, w] + intensity[h - 1,
     w] + intensity[h, w + 1] + intensity[h, w - 1]))));
104.
```

```
105.                                imageData[h, w].g = 0;
106.
107.                                imageData[h, w].b = 0;
108.
109.                            // make black bckground transparant
109.                                if(imageData[h,w].r < 20)
110.                                {
111.                                    imageData[h, w].a = 0;
112.                                }
113.                                else if(imageData[h, w].r < 50)
114.                                {
115.                                    imageData[h, w].a = 200;
116.                                }
117.                                else
118.                                {
119.                                    imageData[h, w].a = 255;
120.                                }
121.
122.                            }
123.                             catch (Exception)
124.                            {
125.                            Debug.Log(Math.Abs(4 * intensity[h, w]
126.                                        - (intensity[h + 1, w] + intensity[h - 1,
    w] + intensity[h, w + 1] + intensity[h, w - 1])));
127.            }
128.                        }
129.
130.                    }
131.
132.                return imageData;
133.            }
134.
135.            Double[,] SimpleBlurGreyScale(Double[,] intensity, int nTimes)
136.            {
137.                Double[,] imageData = new Double[imageHeight, imageWidth];
138.
139.                for (int n = 0; n <= nTimes;n++)
140.                {
141.                    for (int h = 1; h < imageHeight - 1; h++)
142.                    {
143.                        for (int w = 1; w < imageWidth - 1; w++)
144.                        {
145.                            imageData[h, w] = (intensity[h - 1, w] + intensity[h
    + 1, w] + intensity[h, w - 1]
146.                                            + intensity[h, w + 1] + intensity[h, w])
    / 5;
147.
148.                        }
149.                    }
150.                }
151.                return imageData;
152.            }
153.
154.            Color32[,] BlurFilter(Color32[,] twoDImagearray)
155.            {
156.
157.                Color32[,] imageData = new Color32[imageHeight, imageWidth];
158.
159.
```

```
160.                 for (int counter = 0; counter < 2; counter++) // executes filter
161.                                                               // "counter" (n) ti
    mes
162.                 {
163.                     for (int h = 1; h < imageHeight - 1; h++) // goes through pix
    els
164.                                                               // vertically
165.                     {
166.
167.                         for (int w = 1; w < imageWidth - 1; w++) // goes through
    pixels
168.                                                                 // horizontally
169.
170.                         {
171.                             imageData[h,w].r = Convert.ToByte((twoDImagearray[h-
    1,w].r + twoDImagearray[h + 1, w].r + twoDImagearray[h, w-1].r
172.                                 + twoDImagearray[h, w-
    1].r + twoDImagearray[h, w].r) / 5);
173.
174.                             imageData[h, w].g = Convert.ToByte((twoDImagearray[h
    - 1, w].g + twoDImagearray[h + 1, w].g + twoDImagearray[h, w - 1].g
175.                                 + twoDImagearray[h, w - 1].g + twoDImagearra
    y[h, w].g) / 5);
176.
177.                             imageData[h, w].b = Convert.ToByte((twoDImagearray[h
    - 1, w].b + twoDImagearray[h + 1, w].b + twoDImagearray[h, w - 1].b
178.                                 + twoDImagearray[h, w - 1].b + twoDImagearray
    [h, w].b) / 5);
179.
180.                             imageData[h, w].a = 255;
181.
182.                             // original selected pixel rgb value gets overriden w
    ith the
183.                             // average rgb value of the original selected pixel r
    gb
184.                             // value added to
185.                             // neighbouring pixels
186.                         }
187.
188.                     }
189.                 }
190.
191.                 return imageData;
192.             }
193.
194.
195.
196.             Color32[,] CutOutFilter(Color32[,] twoDImagearray, int imageWidth, in
    t imageHeight)
197.             {
198.
199.                 Color32[,] imageData = new Color32[224, 224];
200.
201.                 for (int h = 0; h < 224; h++)
202.                 {
203.
204.                     for (int w = 0; w < 224; w++)
205.                     {
```

```
206.                             imageData[h, w] = twoDImagearray[h+10, imageWidth/2
    - 112];
207.                         }
208.                     }
209.
210.                 return imageData;
211.             }
212.
213.
214.             Color32[,] GetTwoDColor(Texture2D sourceTex, int imageWidth, int imag
    eHeight)
215.             {
216.                 var pix = sourceTex.GetPixels32();
217.                 Color32[,] imageData = new Color32[imageHeight,imageWidth];
218.                 int height = imageHeight-1;
219.                 int width = 0;
220.                 for (int i = 0; i <= imageHeight*imageWidth; i++)
221.                 {
222.                     if (height >= 0  && width < imageWidth)
223.                     {
224.
225.                         imageData[height, width] = pix[i];
226.                         width++;
227.                         if (i % imageWidth - 1 == 0)
228.                         {
229.                             height--;
230.                             width = 0;
231.                         }
232.                     }
233.
234.                 }
235.
236.                 return imageData;
237.             }
238.
239.             Color32[] GetOneDColor(Color32[,] twoDimageArray, int imageWidth, int
     imageHeight)
240.             {
241.                 Color32[] imageData = new Color32[imageWidth * imageHeight];
242.
243.                 int counter = 0;
244.                 for(int h = 0; h <= imageHeight; h++)
245.                 {
246.                     for(int w = 0; w <= imageWidth; w++)
247.                     {
248.                         if (counter < imageHeight * imageWidth && h< imageHeight
    && w < imageWidth)
249.                         {
250.                             imageData[counter] = twoDimageArray[h, w];
251.                             counter++;
252.                         }
253.
254.                     }
255.                 }
256.                 return imageData;
257.             }
258.
259.             Color32[] GetOneDColor224(Color32[,] twoDimageArray, int imageWidth,
    int imageHeight)
```

```
260.            {
261.                Color32[] imageData = new Color32[224 * 224];
262.
263.                int counter = 0;
264.                for (int h = 0; h <= 224; h++)
265.                {
266.                    for (int w = 0; w <= 224; w++)
267.                    {
268.                        if (counter < 224 * 224 && h < 224 && w < 224)
269.                        {
270.                            imageData[counter] = twoDimageArray[h, w];
271.                            counter++;
272.                        }
273.
274.                    }
275.                }
276.                return imageData;
277.            }
278.
279.            Color32[] GetBWjpg(Color32[] oneDImageArray)
280.            {
281.                Color32[] imageData = new Color32[oneDImageArray.Length];
282.
283.                for(int i = 0; i<oneDImageArray.Length; i++)
284.                {
285.                    if(oneDImageArray[i].r < 10)
286.                    {
287.                        imageData[i].r = 255;
288.                        imageData[i].g = 255;
289.                        imageData[i].b = 255;
290.                    }
291.                    else if(oneDImageArray[i].r < 30)
292.                    {
293.                        Byte blackGrade = Math.Min(Convert.ToByte(Math.Abs(255 -
    oneDImageArray[i].r * 5)), (byte)255);
294.                        imageData[i].r = blackGrade;
295.                        imageData[i].g = blackGrade;
296.                        imageData[i].b = blackGrade;
297.                    }
298.                    else
299.                    {
300.                        imageData[i].r = 0;
301.                        imageData[i].g = 0;
302.                        imageData[i].b = 0;
303.                    }
304.                    imageData[i].a = 255;
305.                }
306.
307.                return imageData;
308.            }
309.
310.            void SaveToPng()
311.            {
312.
313.                camTextureSaved = camFeed.GetImageTexture();
314.
315.                int imageHeightSaved = Convert.ToInt32(camTexture.height);
316.                int imageWidthSaved = Convert.ToInt32(camTexture.width);
317.                //getting right pxiel ratios
```

```
318.                double ratio = imageWidthSaved / imageHeightSaved;
319.
320.                imageHeightSaved = 244;
321.                imageWidthSaved = Convert.ToInt32(244 * ratio);
322.
323.                camTexture = TextureTools.CropTexture(camTexture);
324.                camTextureSaved = TextureTools.scaled(camTextureSaved, imageWidth
    Saved, imageHeightSaved, FilterMode.Trilinear);
325.                Debug.Log(imageHeightSaved + "    " + imageWidthSaved);
326.
327.                //convert to two D pixel array
328.                Color32[,] twoDImageData = GetTwoDColor(camTextureSaved, imageWid
    thSaved, imageHeightSaved);
329.
330.                // apply edge filter
331.                Color32[,] edgeImageData = EdgeFilter(twoDImageData, imageWidthSa
    ved, imageHeightSaved);
332.
333.                //cutout for saving to file
334.                Color32[,] cutOutImageData = CutOutFilter(edgeImageData, imageWid
    thSaved, imageHeightSaved);
335.
336.                //convert back to one D pixel array
337.                Color32[] oneDSaveToFileImageData = GetOneDColor224(edgeImageData
    , imageWidthSaved, imageHeightSaved);
338.
339.                saveToFileTex = new Texture2D(224, 224);
340.                saveToFileTex.SetPixels32(oneDSaveToFileImageData);
341.                saveToFileTex.Apply();
342.
343.                fileName = fileNameField.text;
344.                byte[] pngBytes = saveToFileTex.EncodeToPNG();
345.                File.WriteAllBytes(Application.persistentDataPath + "/" + fileNam
    e + imageNameCounter + ".png", pngBytes);
346.
347.
348.                Color32[] BWjpgImageData = GetBWjpg(oneDSaveToFileImageData);
349.                saveToFileTex.SetPixels32(BWjpgImageData);
350.                saveToFileTex.Apply();
351.
352.                byte[] jpgBytes = saveToFileTex.EncodeToJPG(100);
353.                File.WriteAllBytes(Application.persistentDataPath + "/" + "B_W_"
    + fileName + imageNameCounter + ".jpg", jpgBytes);
354.
355.                Debug.Log("Saved to " + Application.persistentDataPath + "/"+ "R_
    T_" + fileName + imageNameCounter + ".png");
356.                messageBehavior.ShowMessage("Saved");
357.                imageNameCounter++;
358.            }
359.            // Update is called once per frame
360.            void Update () {
361.
362.
363.                timeCounter++;
364.                if (timeCounter > 20)
365.                {
366.                    timeCounter = 0;
367.                    ProcessImage();
368.                }
```

```
369.                if (Input.GetMouseButtonDown(0))
370.                {
371.                    if(fileNameField.text != "")
372.                    {
373.                        SaveToPng();
374.                    }
375.                    else
376.                    {
377.                        messageBehavior.ShowMessage("Enter name to Save");
378.                    }
379.
380.
381.                }
382.                if (camTexture)
383.                {
384.                    float ratio = (float)camTexture.width / (float)camTexture.hei
    ght;
385.                    fit.aspectRatio = ratio;
386.                }
387.
388.            }
389.        }
```

13. Reference List

APPENZELLER, T., 2017. *The AI revolution in science* [viewed Feb 12, 2019]. Available from: https://www.sciencemag.org/news/2017/07/ai-revolution-science

BODEN, M.A., 1996. *Artificial intelligence.* 2nd ed. London: APProfessional

COHN, G., 2018. AI Art at Christie's Sells for $432,500. *The New York Times,* -10-27T03:04:22.429Z

DENG, J.J., 2019. How to scrape Google for Images to train your Machine Learning classifiers on. In: *Medium*. -02-07T07:48:33.655Z [viewed May 3, 2019]. Available from: https://medium.com/@intprogrammer/how-to-scrape-google-for-images-to-train-your-machine-learning-classifiers-on-565076972ce

DESAI, S., 2017. Neural Artistic Style Transfer: A Comprehensive Look. In: *Medium.* -09-14T16:04:40.938Z [viewed Feb 13, 2019]. Available from: https://medium.com/artists-and-machine-intelligence/neural-artistic-style-transfer-a-comprehensive-look-f54d8649c199

GOODFELLOW, I. *et al.*, 2014. Generative adversarial nets. *Advances in neural information processing systems.* pp.2672-2680

HERPICH, F., R.L.M. GUARESE and L.M.R. TAROUCO, 2017. A Comparative Analysis of Augmented Reality Frameworks Aimed at the Development of Educational Applications. *Creative Education,* 08, 1433

JIAWEI HAN, MICHELINE KAMBER,JIAN PEI., 2011. *Data Mining: Concepts and Techniques.* 3 ed. ed. US: Morgan Kaufmann Publishers Inc

Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D. (2018). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. https://github.com/Unity-Technologies/ml-agents.

KIPPER, G. and J. RAMPOLLA, 2013. *Augmented reality : an emerging technologies guide to AR.* 1st ed. Waltham, Massachusetts: Syngress

RUSSELL, STUART J. (STUART JONATHAN), E. DAVIS and P. NORVIG, 2010. *Artificial intelligence : a modern approach.* 3rd ed. Upper Saddle River, N.J: Pearson Education

SCOBLE, R. and S. ISRAEL, 2016. The fourth transformation : how augmented reality and artificial intelligence change everything. United States: Patrick Brewster Press

WARWICK, K., 2012. Artificial intelligence : the basics. London: Routledge

14. Bibliography List

WANG, X. and P.S. DUNSTON, 2006. Potential of augmented reality as an assistant viewer for computer-aided drawing. Journal of Computing in Civil Engineering, 20(6)

GLOVER, J., 2018. Unity 2018 Augmented Reality Projects: Build four immersive and fun AR applications using ARKit, ARCore, and Vuforia. Packt Publishing Ltd

YEE, B., Y. NING and H. LIPSON, 2009. Augmented reality in-situ 3D sketching of physical objects. Intelligent UI workshop on sketch recognition. Citeseer,