

**Semantic Segmentation with Deep Neural Networks**

**by**

**Abhishek Kumar[A00268732]**

**BACHELOR’S THESIS**

**to achieve the university degree of**

**B.E(Hons) Software Engineering**

**submitted to**

**ATHLONE INSTITUTE OF TECHNOLOGY**

**Supervisors:**

**Mr. Michael Thornton**

**April 27,2020**

**Abstract**

Semantic segmentation is about marking each pixel in an image with the division to which it belongs. There are many implementations in a wide variety of fields, such as automation, navigation or medical image processing, where pixel-level marks are of paramount significance. In recent years, deep neural networks have demonstrated promising results and have become state-of-the-art for a variety of recognition tasks. In this thesis, I have research and use the deep neural networks for the purpose of semantic image segmentation.

I have created a fully convolutional networks, which is designed to label images for the task of cityscapes image segmentation. I transfer the learned feature representation from a cityscapes image dataset of street images for classification to pixel-wise labeling of cityscapes dataset. Further, I perform a training of the deep neural network at 20 epochs and used a model check point to save the model which have the minimum validation loss.

My proposed semantic segmentation approach is evaluated on a cityscapes dataset and improves the accuracy of the model. My experimental evaluation confirms that end-to-end training of the deep neural network and a conditional random field improves the overall performance of the model. Finally, I input the unlabeled data (test data) into the model which was showing accurate results.

# Declaration

I declare that I am the sole author of this thesis & that all the work presented in it, unless otherwise referenced, is my own. I also declare that this work has not been submitted, in whole or in part, to any other university or college for any degree or qualification.

Abhishek Kumar Mr. Michael Thornton (Supervisor)

27th April 2020

# Acknowledgements

This project has been an immense learning experience for me, I would like to acknowledge, “Mr. Michael Thornton” for giving me this opportunity & constantly helping me in the whole process. His vision & guidance have helped me to learn & explore about this technology. I would also like to thank my felloe batchmates who were supporting and giving their insight on my project which really helped me a lot during the development phase. I would like to thank, “Declan Byrne” who has been an overall figure for me throughout the whole semester.

Table of Contents

[Declaration 3](#_Toc39088150)

[Acknowledgements 4](#_Toc39088151)

[Acronyms 6](#_Toc39088152)

[Chapter 1 7](#_Toc39088153)

[1.1 Semantic Segmentation 8](#_Toc39088154)

[1.1.1 Definition and Applications 8](#_Toc39088155)

[1.1.2 The Challenges 9](#_Toc39088156)

[1.1.3 Recipe for Semantic Segmentation 10](#_Toc39088157)

[1.2 Deep Neural Networks 11](#_Toc39088158)

[1.3 Image Labeling Methods 13](#_Toc39088159)

[1.4 Contributions of This Thesis 14](#_Toc39088160)

[1.5 Outline 14](#_Toc39088161)

[Chapter 2: Literature Review 15](#_Toc39088162)

[2.1 Introduction 15](#_Toc39088163)

[2.2 Brief History of Deep Learning 15](#_Toc39088164)

[2.3 The Current trend of Deep Learning 17](#_Toc39088165)

[2.4 Future of Deep Learning 17](#_Toc39088166)

[Chapter 3: System Design 19](#_Toc39088167)

[3.1 Introduction 19](#_Toc39088168)

[3.2 Workflow 19](#_Toc39088169)

[Fig - Workflow of deep convolutional neural networks (CNNs) for semantic segmentation. 19](#_Toc39088170)

[3.3 Requirements 20](#_Toc39088171)

[3.3.1 Cityscapes Dataset 20](#_Toc39088172)

[3.3.2 Scikit-Learn KMeans 20](#_Toc39088173)

[3.3.3 Tensorflow 20](#_Toc39088174)

[3.3.4 Keras 21](#_Toc39088175)

[3.3.5 Pillow and CV2 21](#_Toc39088176)

[3.3.6 Matplotlib 21](#_Toc39088177)

[3.3.7 Numpy 22](#_Toc39088178)

[3.4 Loading Image 22](#_Toc39088179)

[3.5 Colour Clustering 22](#_Toc39088180)

[3.5 Layer to RGB Transition 23](#_Toc39088181)

[3.6 Colour to Class Transition 24](#_Toc39088182)

[3.7 Data Generator 24](#_Toc39088183)

[3.8 Convolutional Neural Network – Unet 25](#_Toc39088184)

[3.8 Training the model 27](#_Toc39088185)

[3.9 Validation Loss and Accuracy 27](#_Toc39088186)

[3.10 Results 28](#_Toc39088187)

[3.10 ROC Curve 29](#_Toc39088188)

[3.11 Accuracy Class 30](#_Toc39088189)

[Chapter 4: Conclusions 31](#_Toc39088190)

[4.1 Introduction 31](#_Toc39088191)

[4.2 Reflection 31](#_Toc39088192)

[4.3 Recommendation 31](#_Toc39088193)

[Chapter 5: References 32](#_Toc39088194)

[Appendix A: Images 34](#_Toc39088195)

[A.1 Accuracy Class 34](#_Toc39088196)

[A.2 Validation Accuracy & Loss 34](#_Toc39088197)

[A.3 Result 1 35](#_Toc39088198)

[A.4 Result 2 35](#_Toc39088199)

[A.5 Result 3 35](#_Toc39088200)

[A.6 Result 4 36](#_Toc39088201)

[A.7 Result 5 36](#_Toc39088202)

[A.8 Result 6 37](#_Toc39088203)

[A.9 Result 7 37](#_Toc39088204)

[A.10 Result 8 37](#_Toc39088205)

[A.11 Result 9 38](#_Toc39088206)

[Appendix B: System Specifications 38](#_Toc39088207)

# Acronyms

**CNN** Convolutional Neural Network

**CRF** Conditional Random Field

**DNN** Deep Neural Network

**DSM** Digital Surface Model

**IoU** Intersection Over Union

**MAP** Maximum A Posteriori

**MLE** Maximum Likelihood Estimation

**MLP** Multilayer Perceptron

**NIR** Near-infrared

**SGD** Stochastic Gradient Descent

**SVM** Support Vector Machine

# Chapter 1

**Introduction**

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| [**1.1**](#page17) | [**Semantic Segmentation**](#page17) **. . .** | **.....................** | **1** |
| [**1.2**](#page19) | [**Deep Neural Networks**](#page19) **. . .** | **.....................** | **3** |
| [**1.3**](#page21) | [**Image Labeling Methods**](#page21) **. . . . . . . . . . . ..** | | **5** |
| [**1.4**](#page21) | [**Contributions of This Thesis**](#page21) | **.....................** | **6** |
| [**1.5**](#page22) | [**Outline**](#page22) **. . . . . . . . . . . ..** | **.....................** | **6** |
|  |  |  |  |

The first chapter introduces semantic image segmentation, explains what the challenges are and presents a recipe of how to approach semantic segmentation. In addition, a short introduction to deep learning is given before current state-of-the-art methods are discussed. I conclude with a summary of our contributions and gave an outline of this thesis.

# 1.1 Semantic Segmentation

# 1.1.1 Definition and Applications

Semantic segmentation is about automatically extracting information from images. The goal is to assign a category label to each pixel in an image, or in other words, given an input image, I want to know all segmented object, where they are at pixel-level and what category they belong to. For that purpose, it is required to jointly solve localization, segmentation and classification.

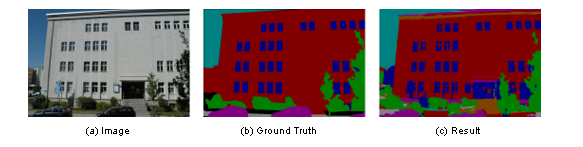


Figure [1.1](#page18) illustrates an example image and a semantic segmentation result created. The objective is illustrated by the ground truth image.

There exist many applications that require semantic information from images at pixel-level, like in the fields of medical image analysis, robotics, surveillance and many more. However, automatically extracting semantic information from images is a challenging task, as I will describe in the next section.

# 1.1.2 The Challenges

Semantic segmentation is still an unresolved problem in computer vision. While recent methods show reasonable results, they are still not able to beat human-level performance.

Humans can recognize objects with less eﬀort, even if the objects change in viewpoints, scale, illumination or when they are translated or rotated. Objects can even be recognized when they are partially occluded from view.

To automatically tackle these challenges using computers, state-of-the-art semantic segmentation methods rely on machine learning techniques to learn the various representations of objects from given images. Nevertheless, actual approaches have also their drawbacks. To achieve the accuracy of state-of-the-art methods, pixel-level annotated images are re-quired, which are limited for many applications or even not available [.](#page102)

In addition, learning the object representation from a finite set of labeled images, the model may achieve a satisfying performance on samples that look similar to those in the training set, but there is no guarantee of the algorithm to generalize well on other images. This problem is also known as the dataset-bias.

For that reasons, semantic segmentation is still a hot topic in research.



FIGURE 1.2: Recipe for semantic segmentation.

# 1.1.3 Recipe for Semantic Segmentation

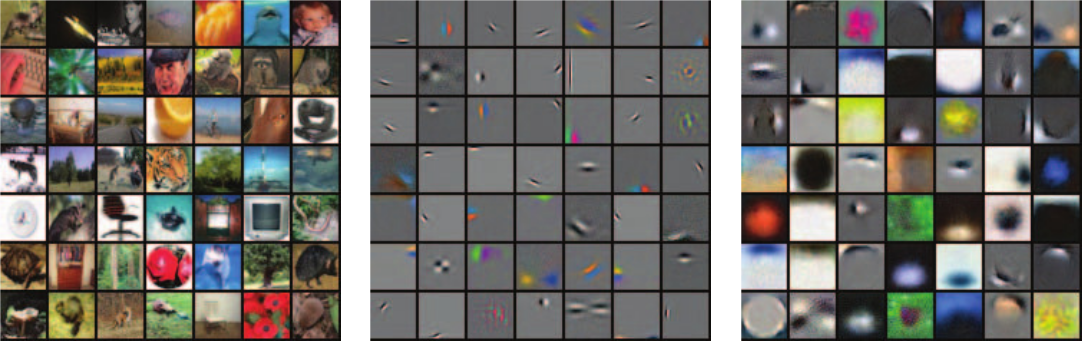
In the last section, I got an insight of the challenges to tackle. Next, I describe a common recipe for semantic segmentation algorithms which is illustrated in figure [1.2.](#page19)

First, a given image is represented in a feature space. The intention of a feature space is to get a lower-dimensional representation of the original image. Second, a classifier is applied to decide which category label each pixel belongs to. Since Ideal with real-world data, the classification results may be noisy or the estimated object boundaries do not coincide with the input image. Therefore, a graphical model is finally applied to counteract these inconsistencies.

In this thesis, I investigate into deep neural networks (DNNs) and conditional random fields (CRFs) to implement these steps. I now continue with an introduction of deep neural networks before I describe the current state-of-the-art image segmentation methods.

# 1.2 Deep Neural Networks

In recent years, deep learning methods using neural networks have shown impressive results in many fields, like in speech recognition [,](#page102) natural language processing or computer vision. Deep neural networks are able to automatically learn low-level and high-level representations from given data [.](#page102) There is no need for hand-crafted features, which often require expensive human eﬀorts and expert knowledge.



(a) Input images (b) Low-Level Features (c) High-Level Features

FIGURE 1.3: Low-level and high-level representation of images using deep neural net-works. Image courtesy [.](#page102)

FIGURE 1.4: Network architecture of one of the first CNNs which was used to perform digits recognition

Examples of learned features are shown in figure [1.3.](#page20) The visualized features demonstrate the ability of the network to capture low-level features such as edges and corners, as well as high-level features to model the complex composition of object categories.

DNNs are built of several layers to represent the images at diﬀerent levels of abstraction. The architecture of one of the first deep learning methods, a convolutional neural network (CNN) to recognize digits.

The drawbacks of deep neural networks are high computational costs and the need for a huge amount of data. The raise of computational power due to graphical processing units and the availability of large-scale datasets lead deep learning methods to be state-of-the-art for several recognition task.

It is an interesting insight that deep models like CNNs [,](#page103) which are used to extract features at diﬀerent levels of abstraction, are biologically inspired by the primary visual cortex (V1) [,](#page103) where simple and complex cells respond to lower and higher level features respectively.

CNNs are also state-of-the-art to perform semantic image segmentation. The next section describes the current best performing image labeling methods.

# 1.3 Image Labeling Methods

In recent years, the progress of deep learning methods improved the accuracy of image labeling algorithms too. The current best performing semantic segmentation method rely on CNNs. I have summarize the diﬀerent variants in this section.

Fully convolutional networks by make use of CNNs to classify each pixel in an image, but instead of applying a CNN for each output pixel, they compute the output map more eﬃciently by applying the network convolutional. In addition, deconvolution layers are added on top of the network to further refine the outputs.

Since labeling images at pixel-level is an expensive task, studies also analyzed weakly-supervised learning setups, in which images with image-level annotations are in-corporate into the training process. Other work by exploits bounding box annotations to reduce the number of required pixel-level annotated images.

The previous methods described are tuned to detect everyday objects in images. I also investigate into semantic segmentation of cityscape dataset. The methods proposed by rely on neural networks for cityscapes images labeling. However, they do not apply them convolutional. Instead, they crop patches from images, which are then classified by a neural network.

In this section, I gave an overview of state-of-the-art image segmentation methods. In the next section, I will summarize the scientific contributions of this thesis.

# 1.4 Contributions of This Thesis

In this section, I summarize the main contributions of this thesis.

**Deep Neural Networks for cityscapes images Labeling**

This is a specific form of CNN, which utilizes concatenate layers to feed data from previous layers into near-output layers. My approach was to improve the cityscapes images benchmark.

**Structured End-to-end Learning**

Fully convolutional networks are trained using a loss function that does not consider dependence between adjacent pixels. For that reason, I study an approach to perform structured end-to-end training of a fully convolutional network and a CRF. In particular, I train the my model at 20 epochs which took 7.5 hrs to complete

# 1.5 Outline

Chapter one of this thesis offers a general introduction of semantic segmentation, deep neural method, challenges and image method labelling.

Chapter two of this study would look at Deep Learning, its past, its present state & the future of deep learning. It will look at the present application of deep learning & its potential.

Chapter three will concentrate on the construction of the project framework. Specifications, an outline of the process for the project will be given.

Chapter four would demonstrate the validation of the program, whether the customer approved the road risk detecting check, in order to decide if the methods used are appropriate.

The results of the work carried out on the project will be seen in Chapter five. It's going to show what I should have done better & the continued research I intend to do for the idea.

# Chapter 2: Literature Review

# 2.1 Introduction

This section presents the literature consulted for the research required to enable the knowledge to commence this project. The literature was reviewed with the intent to review the history to the theory, & the application of deep Learning.

# 2.2 Brief History of Deep Learning

Deep Learning, as a branch of Machine Learning, uses algorithms to analyze data and mimic the mechanism of thought, or to create abstractions. Deep Learning (DL) uses algorithm layers to process data, interpret human expression, and visually identify objects. Knowledge is transmitted through each layer, with the data of the previous layer supplying information to the next layer. The first layer in the network is called the input layer, and the last layer is called the output layer. The layers within them are referred to as hidden layers. Growing layer is usually a simple, standardized algorithm containing a single type of activation function.

Function selection is a further component of Deep Learning. Feature Extraction uses an algorithm to automatically create useful data "properties" for teaching, learning, and comprehension purposes. Typically, the Data Scientist, or programmer, is responsible for the retrieval of functionality.

The history of Deep Learning can be traced back to 1943, when Walter Pitts and Warren McCulloch developed a computational model based on human brain neural networks. They used a combination of algorithms and mathematics that they called "threshold logic" to mimic the process of thinking. Since then, Deep Learning has progressed gradually, with just two major breaks in its growth.

Henry J. Kelley is credited with creating the fundamentals of a continuous 1960 Back Propagation Method. In 1962, Stuart Dreyfus created a simplified version based only on the chain rule. Although the idea of backspread (backspreading of errors for training purposes) did occur in the early 1960s, it was sloppy and unreliable and would not become effective before 1985.

The early efforts to develop Deep Learning algorithms came from Alexey Grigoryevich Ivakhnenko (developed Community Method of Data Handling) and Valentin Grigor Lápa (author of Cybernetics and Forecasting Techniques) in 1965.

They used models of polynomial (complicated equations) activation functions, which were then evaluated statistically. The best statistically selected features were then forwarded from each layer to the next layer (a long, manual process).

In the 1970s, the first AI started the season, the product of commitments that could not be maintained. The effect of this lack of funding has affected both DL and AI science. Luckily, there were people who carried out work without funding.

Kunihiko Fukushima used the first "convolutional neural networks." Fukushima developed neural networks with multiple pooling and convolutional layers. In 1979, an artificial neural network, called Neocognitron, was created using a hierarchical, multilayer architecture. This design allowed the machine to "read" visual patterns. The networks mirrored modern models but were equipped with a technique of amplification of repeated multi-layer activation, which accumulated intensity over time. Additionally, the architecture of Fukushima allowed essential features to be manually changed by the the "weight" of such connections.

# 2.3 The Current trend of Deep Learning

Deep learning is all around us. It is used to decide which online advertisements should appear in real time, recognise and tag friends in images, translate your voice to text, translate text to various languages on the web page, and drive stand-alone vehicles.

Deep learning is often used in less visible environments. Credit card firms use deep learning to identify fraud; enterprises use it to determine when someone will cancel a payment and give customized guidance to clients; banks use it to forecast default and loan risk; hospitals use it to track, diagnose, and manage illnesses.

The number of implementations is nearly infinite. Many choices include text analysis, image captioning, image colorization, x-ray interpretation, weather forecasting, financial forecasts, and more.

Deep learning is now commonly used to automate processes, boost performance, identify patterns, and solve problems.

# 2.4 Future of Deep Learning

Predictions for the Future of Deep Learning claims that DL will be democratized over every app creation platform within the next 5 to 10 years. DL tech is expected to become a regular part of the device toolkit. Reusable DL modules, built into standard DL libraries, will use the instructional features of their previous versions to improve learning. If the advancement of deep learning applications continues, there is an obvious probability that technology will turn into something so complex that the average developer will be totally ignorant of it. Deep learning networks can demystify the memory of your machine.

A Deep Dive into Deep Learning in 2019 reflects on the pervasive role of DL in many areas of AI whether NLP or computer vision applications. Gradually, AI & DL-enabled automation systems, tools & applications penetrate & take over all business industries from ads to customer services, from virtual reality to natural language processing (NLP)—digital effect is everywhere.

Several of the key developments that are bringing deep learning into the future are: the rapid growth of DL science & company deployment reveals its "ubiquitous" presence in all fields of AI — whether NLP or computer vision implementations.

With time & research methods, unsupervised learning strategies can create models that closely mimic human conduct.

There will appear to be a strong disparity between user data protection legislation and the processing needs of vast volumes of customer data.

Deep learning technology's limitations on being able to "pop" are a hindrance to automated decision-making.

Google's acquisition of DeepMind Technologies is a pledge to global advertisers.

The rapid development of DL research & business deployment shows its "ubiquitous" role in all areas of AI — whether NLP or computer vision implementation.

With time & research methods, unsupervised learning strategies can create models that closely replicate human behavior.

There will tend to be a broad gap between consumer data privacy laws and the operational needs of vast amounts of customer data.

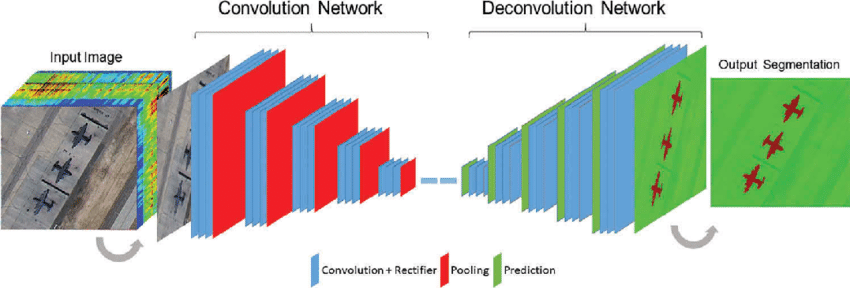
Deep learning constraints on the ability to "pop" are a hindrance to automatic decision-making systems. Google's purchase of DeepMind technology is a promise to multinational advertisers.

# Chapter 3: System Design

# 3.1 Introduction

As stated above, the intention of this project is to create an accurate deep learning model to segment& classify them based on some key features & make the system scalable by using neural network technologies for data processing.

# 3.2 Workflow



# Fig - Workflow of deep convolutional neural networks (CNNs) for semantic segmentation.

U-net was first invented and used for the first time in biomedical image segmentation. Its architecture can be commonly thought of as an encoder network preceded by a decoder network. Unlike classification, where the final outcome of a deep network is the only interesting aspect, semantic segmentation involves not only discrimination at pixel level, but also a method for projecting the discriminatory features acquired at various stages of the encoder onto the pixel space.

The encoder is the first part of the architecture diagram. It is normally a pre-trained classification network like VGG / ResNet where convolution blocks get used followed by a maxpool downsampling to translate the input image into multi-level representations.

The decoder is the second component of the design. The goal is to semantically project the discriminatory features (lower resolution) acquired by the encoder into the pixel space (higher resolution) in order to achieve a dense classification. The decoder consists of sampling and concatenation followed by regular convolution operations.

# 3.3 Requirements

The following are the layers or technologies used to create this system.

1) CityScapes Dataset

2) Scitkit-Learn KMeans

3) Tensorflow

4) Keras

5) Pillow and OpenCV2

6) Numpy

7) Matplotlib

# 3.3.1 Cityscapes Dataset

This dataset has 2975 training images files and 500 validation image files. Each image file is 256x512 pixels, and each file is a composite with the original photo on the left half of the image, alongside the labeled image (output of semantic segmentation) on the right half.

# 3.3.2 Scikit-Learn KMeans

K-means clustering is one of the most widely used unsupervised machine learning algorithms that forms data clusters based on similarity between data instances. The number of clusters must be defined in advance for this particular algorithm to work. The K in the K-means is the number of clusters.

The K-means algorithm begins by arbitrarily choosing the centroid value for each cluster. After that, the algorithm performs three steps: (i)I Find the Euclidean distance between each data instance and the centroids of all clusters; (ii) Assign the data instances to the centroid cluster at the nearest distance; (iii) Calculate new centroid values dependent on the mean coordinate values of all device instances in the same cluster.

# 3.3.3 Tensorflow

It is an end-to-end open source machine learning tool. It has a robust, scalable ecosystem of software, databases & community services that helps researchers to drive the state-of-the-art ML & developers to quickly create & deploy ML-powered applications. It provides multiple layers of abstraction such that someone can select the best one for your needs. Create & train models using the high-level Keras API, which makes it easy to get started with TensorFlow & machine learning.

When one need more autonomy, willing implementation enables rapid experimentation & intuitive debugging. For broad ML training exercises, using the distribution strategy API for distributed testing on various hardware setups without altering the description of the concept. It has also offered a direct route to output. If it's on servers, edge devices or the cloud, it lets train & deploy your model quickly, no matter which language or network people use. Develop & train state-of-the-art models without losing speed & efficiency. It allows simplicity & power with tools such as the Keras Functional API & the Model Sub classing API to build complex topologies. Using eager execution for fast prototyping & simple debugging.

It also provides an ecosystem of popular add-on libraries & conceptual models, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor & BERT.

# 3.3.4 Keras

It is a high-level neural network API, written in Python, capable of running on top of TensorFlow, CNTK, or Theano. This was designed with the intention of allowing rapid experimentation. The trick to conducting good work is to be able to switch from idea to outcome with the least possible time. Allows simple & fast prototyping through user friendliness, modularity & extensibility. Supports all convolutionary networks & recurrent networks, as well as variations of both. Runs on CPU & GPU effortlessly. It is an API designed for humans, not computers. This places the customer interface at the front & the middle. It implements best practices for cognitive load reduction: it delivers reliable & quick APIs, minimizes the amount of user activities needed for specific usage cases, & provides direct & actionable input on user error. Keras main data structure is a pattern, a way to arrange layers. The most common type of model is a sequential model, a linear sequence of layers. For more complex architectures, using the Keras functional API to construct arbitrary layer graphs.

# 3.3.5 Pillow and CV2

Pillow is a library for image manipulation / processing, while OpenCV is a library for computer vision.

Although there is definitely a lot of similarities (i.e. OpenCV includes a bit of image processing functionality) they are very specific in nature.

To make drastic simplification, use Pillow if want to cut and re-size images, and maybe do some scraping, and use OpenCV while you're creating a robot that's trying to "see" stuff.

# 3.3.6 Matplotlib

It is a robust library for making static, animated, & interactive Python visualizations. It makes simple & complex tasks possible. It is a great Python simulation framework for 2D array plots. Matplotlib is a multi-platform data visualization library based on the NumPy array & developed to work with a larger SciPy stack. This was founded in 2002 by John Hunter. Some of the biggest advantages of visualization is that it gives one easy access to large volumes of data with quickly digestible graphics. It consists of multiple plots such as thread, row, scatter, histogram, etc. It comes with a wide range of plots. Plots helps to understand movements, habits, & associations. Usually, they are methods for objective knowledge reasoning. It is a Python programming language plotting library & its NumPy numerical mathematics extension. Provides an object-oriented API to integrate plots into programs using general-purpose GUI toolkits such as Tkinter, wxPython, etc. There is also a state machine-based pylab interface like OpenGL built to closely mimic that of matlab, but its use is discouraged.

# 3.3.7 Numpy

NumPy is a main program for scientific programming with Python. It contains, among other things: a versatile N-dimensional array object with sophisticated (broadcasting) feature tools for combining C / C++ and Fortran code useful linear algebra, Fourier transform, and random number functionality In addition to its obvious science uses, NumPy can also be used as an effective multi-dimensional container for generic data. Arbitral data-types can be described. This helps NumPy to connect smoothly and easily with a wide range of databases.

# 3.4 Loading Image

Loading image from file path, it allows simple data manipulation like flipping and rotating. Dataset has side by side images of raw and color-coded segmented image. It will split the image into separate image.

In Python Pillow is the most common and de facto standard library for loading and working with image info. Pillow is an modified version of the Python Image Repository, or PIL, which provides a variety of basic and advanced tools for image manipulation. In other Python libraries such as SciPy and Matplotlib, it is also the foundation for the basic picture support.

# 3.5 Colour Clustering

The goal is to divide n data points into clusters k. Every of the n data points will be allocated to a nearest-mean cluster. Each cluster's mean is called its "centroid" or "centre."

In general, the implementation of k-means yields k independent clusters of the initial n data points. Data points are "more related" to each other within a cluster than data points which belong to other clusters.

In our case the pixel intensities of an RGB image would be clustered. Thus, given an image of MxN dimension, have a MxN pixels, each consisting of three components: Red, Green and Blue.

Those MxN pixels will be viewed as our data points and clustered using k-means.

Pixels belonging to a given cluster should be more in color compared to pixels belonging to a different cluster.

One caveat of k-means is that will define in advance the number of clusters if it want to produce. There are algorithms that choose the optimum value of k automatically, but these algorithms are beyond this post's reach.

# 3.5 Layer to RGB Transition

I have to preprocess the image before passing to the CNN.

Think that in RGB the 'R' channel senses for "materials darkened by red photons representing red photons" when different. Detection in that case would depend on the lighting ... What color, what angle, what luminosity, etc.

Maybe it might find a translation of the RGB image into another color representation, such as one that distinguishes luminance from chrominance (represented as a 2D vector in color)? This means, light objects are observed irrespective of hue, and colorful objects are less dependent on lighting. Additional representations may discern between hue & saturation (which differentiates between "what colour" and "how pure colour?")

Many RGB pre-processing may involve dividing by the peak diffuse brightness (or approximate peak diffuse brightness if the real peak brightness is cut). People do not care for specular highlights (although maybe people care about your particular problem space?), but usually the detected signal from diffuse white objects in the image (a piece of fabric, paper, painted surface, etc.) will help the CNN perceive objects with less reliance on how brightly the scene is illuminated.

There are algorithms I have seen (but do not know) that are very effective at eliminating, for example, the effects of irregular lighting, allowing for significantly consistent identification of the color of a sphere in the presence of many, differently colored sources of light.

The benefit of pre-processing the RGB to various color spaces is that these can select better the kind of features the CNN will detect and under which conditions such detections would be invariant.

# 3.6 Colour to Class Transition

It converts color clustering output into 13-dimensional representation of class. This will later help the machine to learn the algorithm. Output has shown the layered classification.

The Keras library offers wrapper classes so it can use scikit-learn neural network models built with Keras.

In Keras there is a KerasClassifier class that can be used as an Estimator in scikit-learn, the library's base model sort. The KerasClassifier takes the name as argument for a function. This task will return the neural network model that was built, ready for training.

Below is a method that generates a neural base line network for the issue of iris classification. It provides a basic completely linked network with a single secret layer comprising eight neurons.

The secret layer uses a feature to trigger the rectifier which is a good practice. Because for our iris dataset it used a one-hot encoding, the output layer has to generate 3 output values, one for each class. The output value with the greater value is taken as the model's predicted class.

# 3.7 Data Generator

To make training more memory efficient a generator is used for feeding the data to the deep learning algorithm. This generator creates batches of raw segmented image pair at a moment. It uses image manipulation like random flips and rotation to increase the efficiency size of dataset.

I have added arguments relevant information about the data, such as dimension sizes (e.g. a volume of length) number of channels, number of classes, batch size, or decide whether it want to shuffle our data at generation. I have also store important information such as labels and the list of IDs that will generate at each pass.

Here, the method on\_epoch\_end is triggered once at the very beginning as well as at the end of each epoch

Shuffling the order in which examples are fed to the classifier is helpful so that batches between epochs do not look alike. Doing so will eventually make our model more robust.

Another method that is core to the generation process is the one that achieves the most crucial job: producing batches of data. The private method in charge of this task is called \_\_data\_generation and takes as argument the list of IDs of the target batch.

During data generation, this code reads the NumPy array of each. Now comes the part where I build up all these components together. Each call requests a batch index between 0 and the total number of batches.

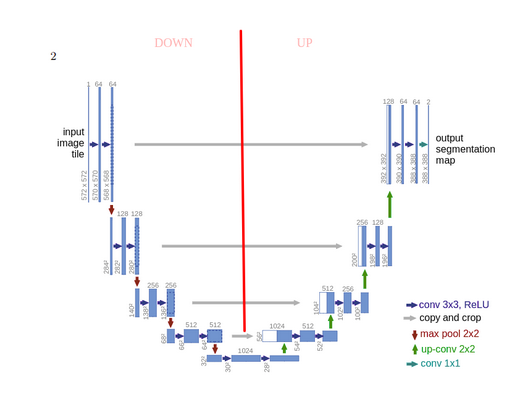
# 3.8 Convolutional Neural Network – Unet

This is a specific form of CNN, which utilizes concatenate layers to feed data from previous layers into near-output layers. The network is built like a double-funnel. Data flow through guarantees that no data is lost. Otherwise the form of the funnel will allow an auto encoder to be quite identical.

Total params: 9,002,410

Trainable params: 9,002,404

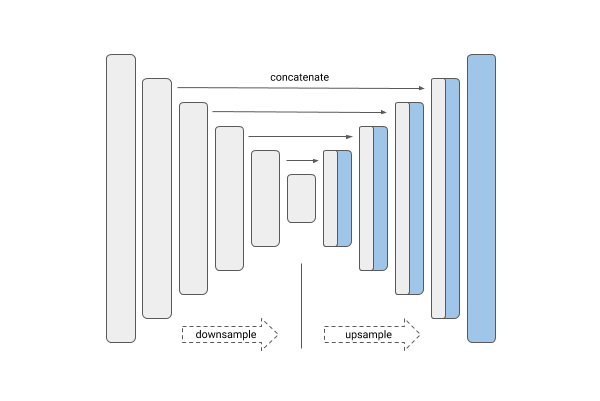
Non-trainable params: 6



Architecture of UNet [*Image source*](https://arxiv.org/pdf/1505.04597.pdf)

The U-Net architecture derives from the first suggested by Long and Shelhamer, the so-called "fullly convolutionary network."

The key concept is to complement successive levels of a normal contracting network, where pooling operations are supplemented by upsampling operators. Therefore these layers improve performance resolution. Moreover, a successive convolutionary layer will then learn to assemble an accurate output based on that input.



A big improvement in U-Net is that the upsampling segment includes a large number of function channels that allow the network to transmit background information to higher resolution levels. As a result, the vast direction to the contracting component is more or less symmetrical, resulting in a u-shaped design. The network only uses the correct portion of each convolution without any entirely linked layers.[2] The missing background is extrapolated by mirroring the input picture to determine the pixels inside the image's boundary area. This tiling technique is critical for applying the network to large files, as otherwise the GPU memory would restrict the resolution.

# 3.8 Training the model

I have trained this model at 20 epochs and used a Model Checkpoint to save the model which have slightest validation loss. I used tensorflow-gpu to train my model better cost of the training.

* On GPU Nvidia940MX an epoch took about 22 minutes. Training early stops at 10-15 epochs.
* For batch size=64 6Gb GPU memory is required.
* Best single model achieved 0.964 accuracy.

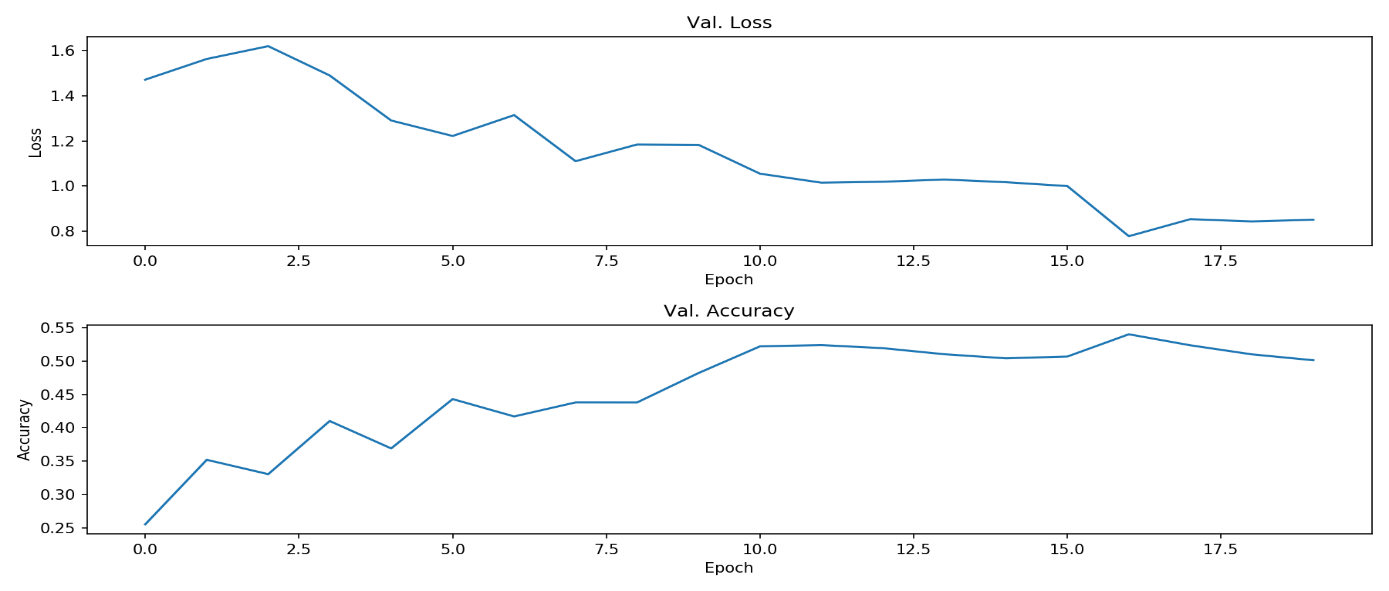
# 3.9 Validation Loss and Accuracy

The History callback is one of the default callbacks that is registered while training all deep learning models. For each period it records the training metrics. This involves loss and accuracy (for classification issues) as well as loss and accuracy, if specified, for the validation dataset.

From calls to the fit) (function used to train the model the history object is returned. Metrics are stored in a dictionary of the returned object's member of history.

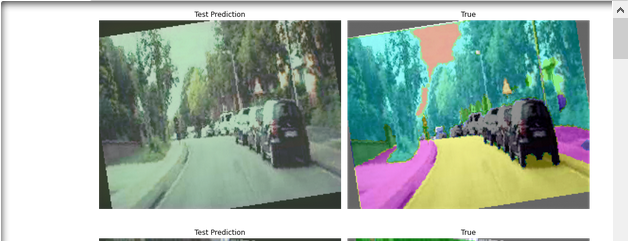
I have collected data in the history object to create plots.

* t’s speed of convergence over epochs (slope).
* Whether the model may have already converged (plateau of the line).
* Whether the mode may be over-learning the training data (inflection for validation line).



# 3.10 Results

I have tested few test images to see how CNN is working. To be very confident that there is no overfitting one should have split from the training dataset even from a test collection.



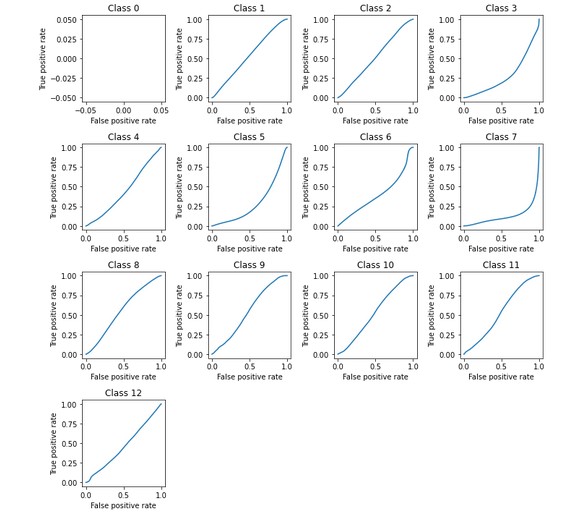




# 3.10 ROC Curve

ROC curve, is a schematic plot that shows a binary classifier system's predictive potential as the threshold for discrimination is varied.

The ROC curve is formed by the plotting at various threshold settings of the true positive rate (TPR) against the false positive rate (FPR). In machine learning, the true-positive rate is also called sensitivity, recall, or chance of detection. Also known as the likelihood of false alarm is the false-positive rate, which can be measured as (1 − specificity). It can also be used as a power story, as a part of the sort I Error of judgment law (when the result is determined from a population sample only, it can be viewed as estimators of certain quantities). Hence, the ROC curve is the sensitivity or recall as a fall-out function. If the probability distributions for both detection and false alarm are known, the ROC curve can be created by plotting the cumulative distribution function (area under the probability distribution of the false alarm probably) versus the cumulative distribution function.



# 3.11 Accuracy Class

Groups of accuracy are described and used under IEC and ANSI standards. Groups are either lettered or percented. Class B, for example, is an IEC-751 temperature accuracy requiring a precision of ± 0.15 degrees Celsius. Class 0.5 is an ANSI C12.20 precision level for electrical meters with absolute accuracy greater than ± 0.5 percent of the full-scale nominal reading.

A class usually determines reliability at a range of scales, with the absolute accuracy at lower values being higher than the average "full scale percentage" accuracy.

Accuracy classes such as 0.15s from the IEC are a high-precision 'special' class.

**Calculation for accuracy of class 1 meter:**

**1600 impulse/KWh and**

**considering, P.F= 1 and LOAD = 100w**

**Revolution time,**

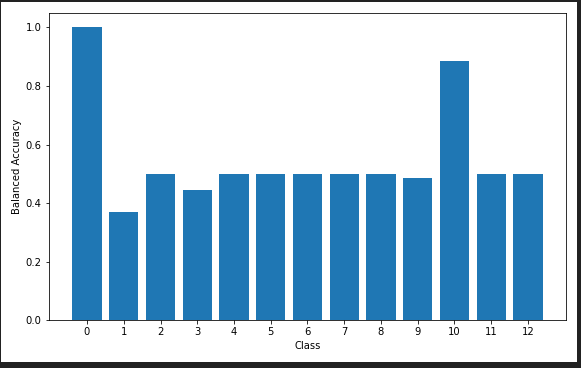
**Rt = (3600×Kh×1)/Load(w) [Kh = 1000/(impulse/Kwh=1600)]**

**Rt = (3600×0.625×1)/100**

**Rt = 22.5sec [Standard]**

**%of error = (Ft-Rt)/Rt**

The positive or negative result indicates whether the meter is fast or slow. If the result is positive then the meter is fast, while negative means the meter is slow.



# Chapter 4: Conclusions

# 4.1 Introduction

This section of the thesis will look back on what the project has achieved, how well it has achieved the goals set & what could have been executed better.

# 4.2 Reflection

The goal of this project was to develop an effective deep learning model to color segment in image based on a variety of main features & to their classification & to make the system scalable by using neural network technology for data processing. Semantic segmentation plays a major role in Autonomous vehicle, Facial segmentation, Robotics, Fashion categorization and Bio-medical Image Diagnosis.

While completing this I had to do lot of research which made me think more about this field. I got to learn about deep learning algorithms, models, base models, dataset which gets used in developing a project.

During the research, I learned how to define & solve the problem. In fact, the project implicitly helped me think individually, discipline myself, be careful, self-confidence & problem-solving skills. In fact, my personal abilities improve as well as I communicate with others. I sought guidance from my boss & professors during the study. They are valuable tips for me to improve myself & stop making the same mistakes again.

Throughout the project, my key learnings were, Time management, Goal management & Analytical Thinking.

# 4.3 Recommendation

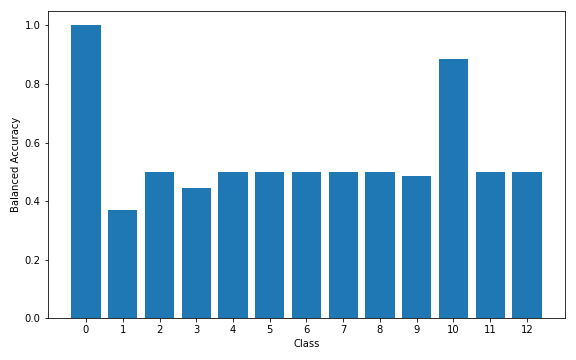
If I were to do this project again, I feel that one of the biggest areas that could be improved was creating an application which could help to detect object using a smart phone. Given more time, other techniques the project can be linked to a government firm, camera can be installed on busy roads, train station for security purposes which could help to get better understanding of

# Chapter 5: References

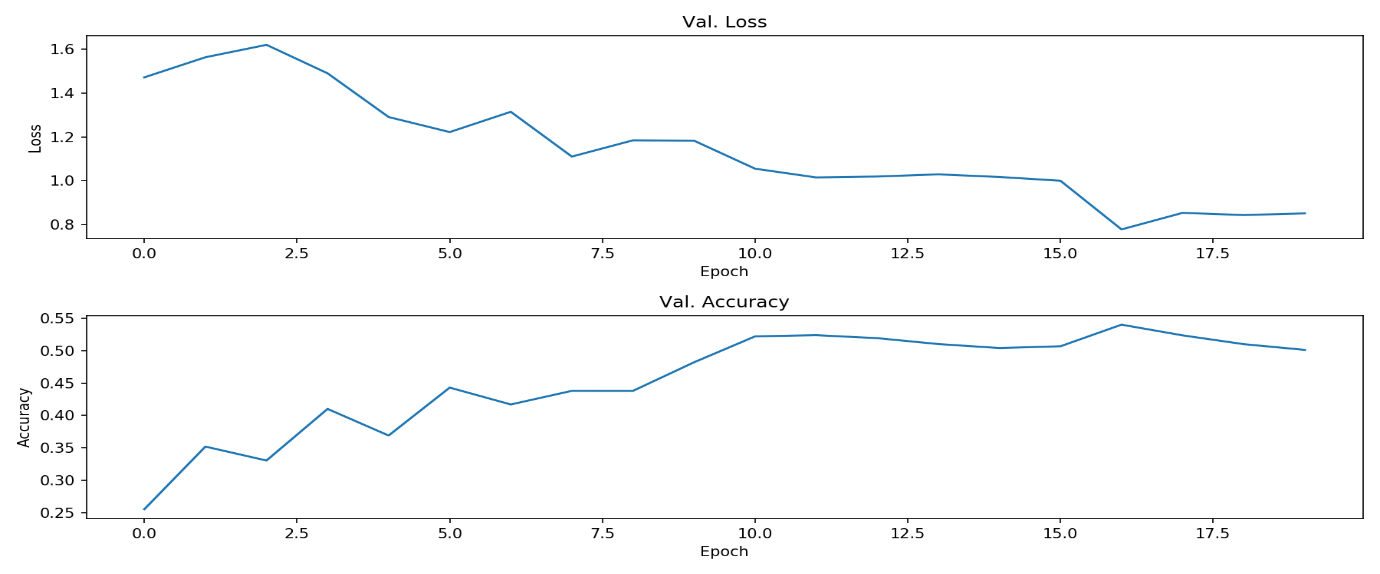
1. Seif, G. (2018, September 19). Semantic Segmentation with Deep Learning. Retrieved 25, 2020, from <https://towardsdatascience.com/semantic-segmentation-with-deep-learning-a-guide-and-code-e52fc8958823>
2. Lamba, H. (2019, February 19). Understanding Semantic Segmentation with UNET. Retrieved April 25, 2020, from <https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>
3. Turgutlu, K. (2019, May 7). Semantic Segmentation — U-Net - Kerem Turgutlu. Retrieved April 25, 2020, from <https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066>
4. Seo, J. D. (2018, June 17). Medical Image Segmentation [Part 1] — UNet: Convolutional Networks with Interactive Code. Retrieved April 25, 2020, from <https://medium.com/@SeoJaeDuk/medical-image-segmentation-part-1-unet-convolutional-networks-with-interactive-code-d07231eb29bf>
5. Gupta, D. (2019, June 6). A Beginner’s guide to Deep Learning based Semantic Segmentation using Keras. Retrieved April 25, 2020, from <https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html>
6. Potter, R. (2019, December 4). How To Label Data For Semantic Segmentation Deep Learning Models? Retrieved April 26, 2020, from <https://medium.com/anolytics/how-to-label-data-for-semantic-segmentation-deep-learning-models-907a996f95f7>
7. Pallawi. (2020, January 19). Semantic segmentation with U-Net- train, and test on your custom data in Keras. Retrieved April 26, 2020, from <https://medium.com/@pallawi.ds/semantic-segmentation-with-u-net-train-and-test-on-your-custom-data-in-keras-39e4f972ec89>
8. Mwiti, D. (2019, June 29). A 2019 Guide to Semantic Segmentation. Retrieved April 20, 2020, from <https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc>
9. MrGloom. (n.d.). mrgloom/awesome-semantic-segmentation. Retrieved March 28, 2020, from <https://github.com/mrgloom/awesome-semantic-segmentation>
10. Cianrn. (n.d.-a). Cianrn/Semantic-Segmentation-Cityscapes. Retrieved March 29, 2020, from <https://github.com/Cianrn/Semantic-Segmentation-Cityscapes>
11. Pal, S. (2019, May 6). Semantic Segmentation: Introduction to the Deep Learning Technique Behind Google Pixel’s Camera! Retrieved March 2, 2020, from <https://www.analyticsvidhya.com/blog/2019/02/tutorial-semantic-segmentation-google-deeplab/>
12. Mody, P. (2018, October 1). Semantic Segmentation: Wiki, Applications and Resources. Retrieved March 25, 2020, from <https://www.kdnuggets.com/2018/10/semantic-segmentation-wiki-applications-resources.html>
13. Ronneberger, O. (2015, May 18). U-Net: Convolutional Networks for Biomedical Image Segmentation. Retrieved March 29, 2020, from <https://arxiv.org/abs/1505.04597>

# Appendix A: Images

# A.1 Accuracy Class



# A.2 Validation Accuracy & Loss



# A.3 Result 1



# A.4 Result 2



# A.5 Result 3



# A.6 Result 4



# A.7 Result 5



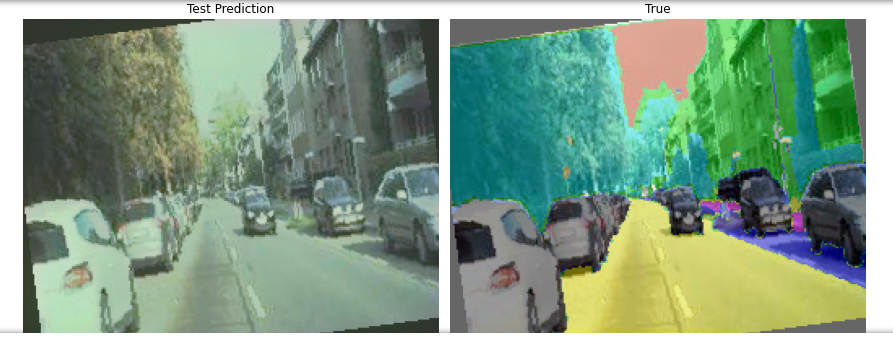
# A.8 Result 6



# A.9 Result 7



# A.10 Result 8



# A.11 Result 9



# Appendix B: System Specifications

|  |  |
| --- | --- |
| 1. OS | 1. Windows 10 Pro 64-bit (10.0, Build 18363) (18362.19h1\_release.190318-1202) |
| 1. Processor | 1. Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (4 CPUs), ~2.9GHz |
| 1. Memory | 1. 16384MB RAM |
| 1. Graphics Card | 1. Dedicated: Graphics Chipset 2. NVIDIA GeForce 940MX 2GB 3. Discrete: Intel(R) UHD Graphics 2GB |