IT Carlow –

BSc.

Software Development

UAV using Convolutional Neural Networks

Research Report

Student Name : Josh Hudziak

Student Email: C00231846@itcarlow.ie

Date : 13/11/2020

Supervisor: Dr. Oisin Cawley

Supervisor Email : oisin.cawley@itcarlow.ie

Abstract

In our present day and as we look to the future, we see drones becoming far more prevalent. It is openly discussed that drones will be delivering packages for consumer use with amazon soon. A drone's capabilities will be far more exceeding just this task. In all environments, no matter how mundane, a drone faces many challenges to navigate in a safe and efficient way, especially if that drone is autonomous. In this report, we explore the options available, using modern technologies and algorithms to solve the above challenges. The modern technology that is the most appealing here is convolutional neural networks.

Contents

Introduction

A UAV is an unmanned aerial vehicle and can be controlled by a person but for the purpose of this project we will create an autonomous UAV. An autonomous UAV must perceive its surroundings and make actions based on these observations. Some UAVs may make these judgments using sensory data such as radar, an example of this being Tesla's autopilot feature which is a hybrid of image processing and radar/ultrasonic sensors [ING16]. Another way to create an autonomous vehicle is by using GPS and localization via a map. However, the goal of this project is to have a UAV using just camera input. This is a very complex challenge as the UAV should be capable of navigating roads, as well as paths and tracks in a safe manner, always under control.

There are a few ways to produce results here, one of them being reinforcement learning on images. This learning does not need labelled datasets. It’s success lies in balancing it’s current knowledge and exploring unknown options. Depending on the algorithm used you may end up with high risk and a fast network. However, this is too risky and may produce unsafe behavior.

Instead here we will create an image classification task using Convolutional Neural Networks and setting it upon a gathered dataset which will be classified in three sections; left, right and straight. Image classification has been trialed and tested producing outstanding results in an array of different tasks [KRI12] *[CIR12]*.Therefore, the CNN will choose the UAVs actions based upon the weight of it’s input’s classification.

However, This technique needs a large dataset to work upon. To do this the dataset will be made from driving, walking and drone flight data which will be labelled according to the track, path or road positioning in the image.

Overview: Technologies and Topics Researched

In this section of the document the overall architectures, technologies and techniques are reviewed. From this review choices can be made as to the best technologies to use. Not only the effectiveness of the technologies but also how they will pair with other technologies which will be needed to implement the project.

A technique must be developed to gather a dataset which is trainable. Further analysis will be conducted as to how to preprocess the data after it has been gathered.

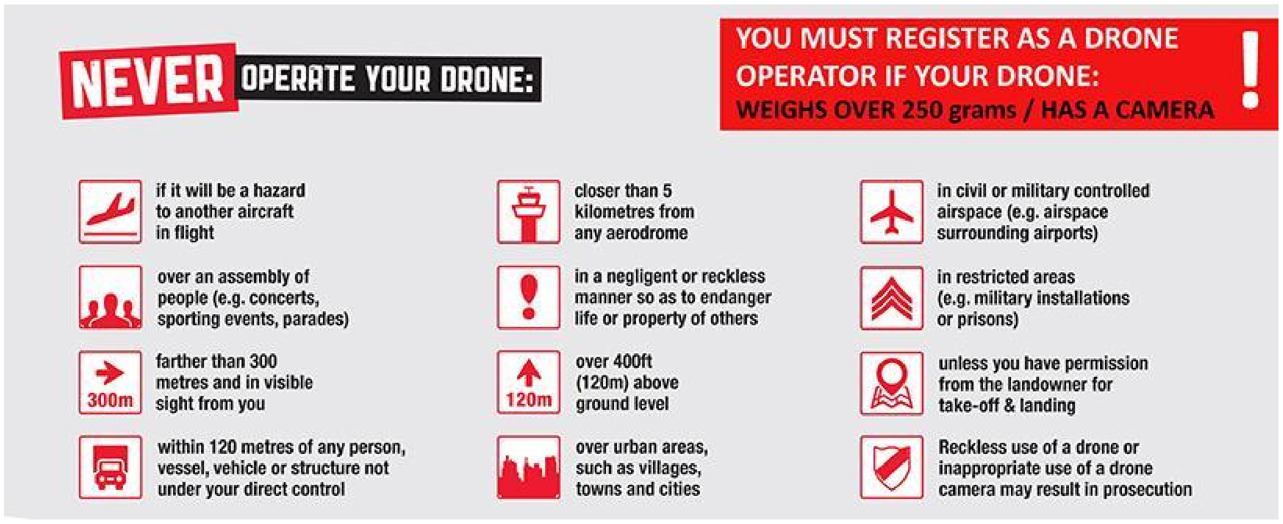
UAV Information

We've already seen in the introduction that a UAV is an unmanned aerial vehicle, therefore a drone will be used to implement a convolutional neural network. There are many different types of drones, you may see fixed wing drones, single rotor drones up to many rotor drones. The most common and consumer friendly being the quadcopter (4 rotors), which is used in this project.

For the case of implementing the CNN, a Parrot Anafi is being used. This drone is a quadcopter that is extremely easy to fly and ultra portable. It has an average battery life of 25min, it also features 4K UHD video with 3 axis image stabilization on a 21 megapixel camera. The drone folds up neatly into its carrying case and it's batteries are swappable and can be recharged with a battery bank on the go because they use a USB-C port [PAR18].

Figure 1: Parrot Anafi, Source: [PAR18]

IAA Drone Flying Regulations



NEVER OPERATE YOUR DRONE:

YOU MUST REGISTER AS A DRONE OPERATOR IF YOUR DRONE: WEIGHS OVER 250 grams / HAS A CAMERA !

- if it will be a hazard to another aircraft in flight
- over an assembly of people (e.g. concerts, sporting events, parades)
- farther than 300 metres and in visible sight from you
- within 120 metres of any person, vessel, vehicle or structure not under your direct control
- closer than 5 kilometres from any aerodrome
- in a negligent or reckless manner so as to endanger life or property of others
- over 400ft (120m) above ground level
- over urban areas, such as villages, towns and cities
- in civil or military controlled airspace (e.g. airspace surrounding airports)
- in restricted areas (e.g. military installations or prisons)
- unless you have permission from the landowner for take-off & landing
- Reckless use of a drone or inappropriate use of a drone camera may result in prosecution

Figure 2: List of Drone regulations 2020, Source: [IAA20]

The Irish Aviation Authority implements drone flight regulations in Ireland, these regulations are in accordance with the European Union Regulations 2019/947. If your drone weighs over 250 grams or has a camera/sensor you will have to register as a drone operator. Registration requires creating an account online, taking the online training and supplying personal and drone details to MySRS. Some of the rules are as follows: Don't fly; within 5km of an aerodrome, over 120m above ground level, over urban areas, further than 300m from the operator, over an assembly of people, within 120m of a vehicle not in the drone operators control [IAA20]. Of course, all these regulations apply to autonomous UAVs as well.

Architecture

Deep Learning

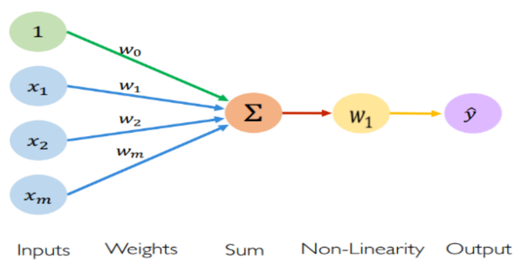
To get to deep learning and then neural networks, traditional machine learning must be understood first. Traditional machine learning algorithms are a defining set of rules or features within data, these rules or features are usually hand engineered and because of this, the algorithms become time consuming, brittle and not very scalable in practise.

Take facial recognition, to define a face you must define the characteristics of a face. A face is composed of a nose, two eyes and a mouth. Now, how do we define those features, they are composed of certain lines, shadings and shapes all within a certain orientation. Here, Deep learning will learn these features all from a large data source. Starting at the lower tier, recognition of lines and shapes and working it's way up to cheekbones, noses and the shapes of eyes and finally an actual face. Although deep learning has been around since the 1950's now it is becoming more prevalent due to 3 main aspects; bigger data sets, better hardware and improved techniques and modelling.

Neural Networks

The basis of a neural network begins with the perceptron (neuron). Each perceptron will have an input or many inputs and each of these inputs will have a corresponding weight. A Sum is gathered from multiplying each input and weight together. There is also a bias added to the sum, this bias is an indication whether the perceptron tends to be active or inactive, the bias therefore has the ability to shift the output.

The sum is then passed through a non-linear activation function, this then produces a final output. The method here is forward propagation of a perceptron. To simplify this equation, take all the inputs and list them in a vector and take all the weights and list them in a vector. Next, get the dot product of the two vectors (inputs and weights) and pass them through a non-linearity function [GOO16]. This is all based on linear algebra, calculus and probability.



$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

$$\text{where: } \boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Figure 3: Forward Propagation Source: [GOO16]

Now to expand upon the non-linear activation function. This function may be any number of formulas such as the Sigmoid function, Hyperbolic Tangent and Rectified linear unit (ReLU). As most every data set is non linear, these functions help us show networks non-linearities.

Creating a network from here is simply scaling. If each perceptron has more than one output, this is called a multi output perceptron which in turn creates a dense layer. From the knowledge of the dense layer we can see how a single layer neural network can be formed. In this network a single hidden layer feeds into outputs, this layer is called the hidden layer because all the values here are learned and cannot be influenced outside of the network, this is in contrast to the inputs which are given to the network. Finally, to create a deep neural network, hidden layers are added to the network, thus each layer connects to another layer, deeper and deeper.

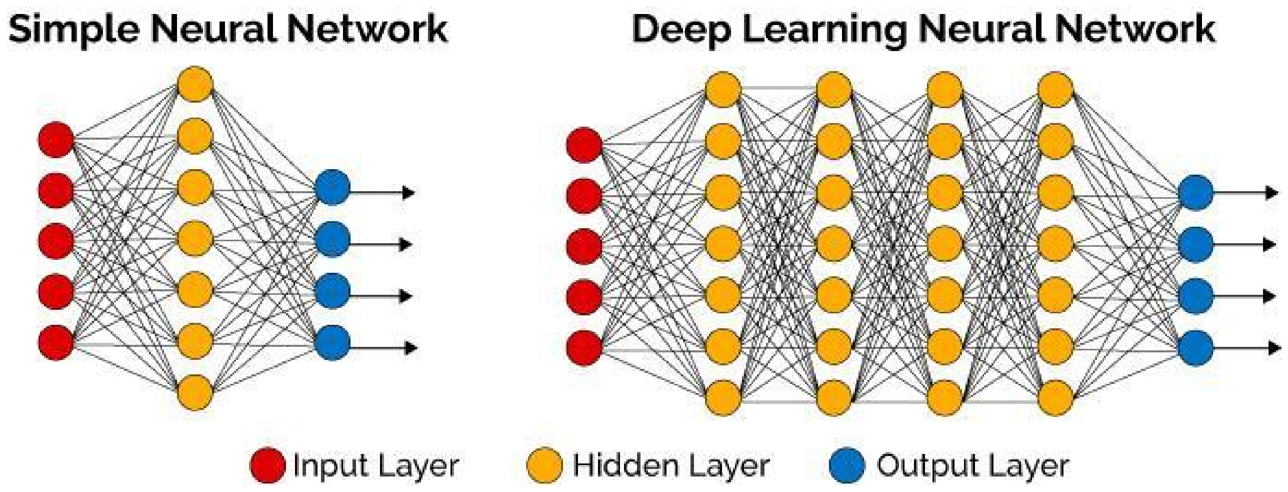


Figure 4: Deep Neural Network Source: [KAM18]

Now that we know what a basic neural network is it needs to be trained to apply it properly. Training the network basically entails telling it when it's wrong, this is done by using a cost/loss measurement. The cost is quantified by the difference of what the network predicted and what the actual answer is. Due to normal datasets being very large there are a multitude of these costs, therefore, the empirical loss or cost function is the average of each individual cost. This works for binary outputs but if the output was dynamic the mean squared error loss with regression models can output dynamic or real numbers.

To optimize the cost function all the weights in the weight vector must be understood, as only a certain set of these weights will grant the lowest cost function. An efficient way of finding this is using gradient descent [KIE52]. If two weights were plotted x-axis, y-axis and the z-axis is the loss, the lowest point on this graph would be the global minima. [LI17] [GOO16]

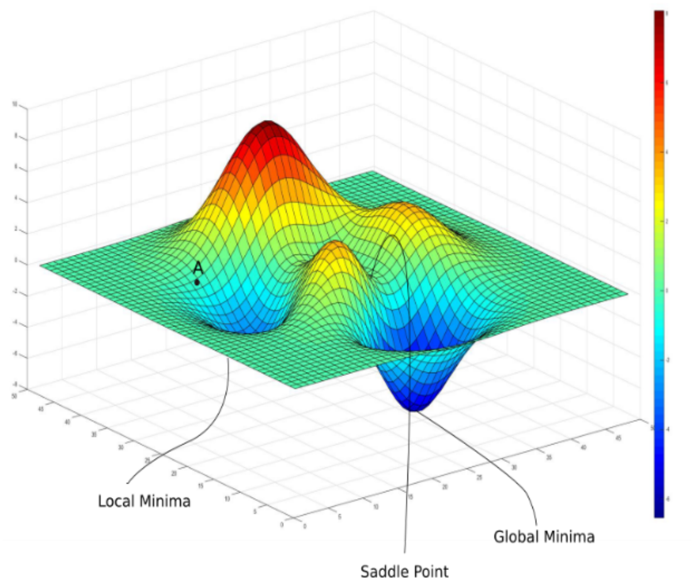


Figure 5: Gradient Descent, Source: [KAT18]

So, given this, how do we implement the network to know what gradient is good or bad. Back propagation uses a core calculus method called the chain rule which states the derivative of f(g(x)) is f'(g(x)).g'(x) [KAH20]. Most deep learning frameworks used have automatic differentiation which applies the back propagation automatically.

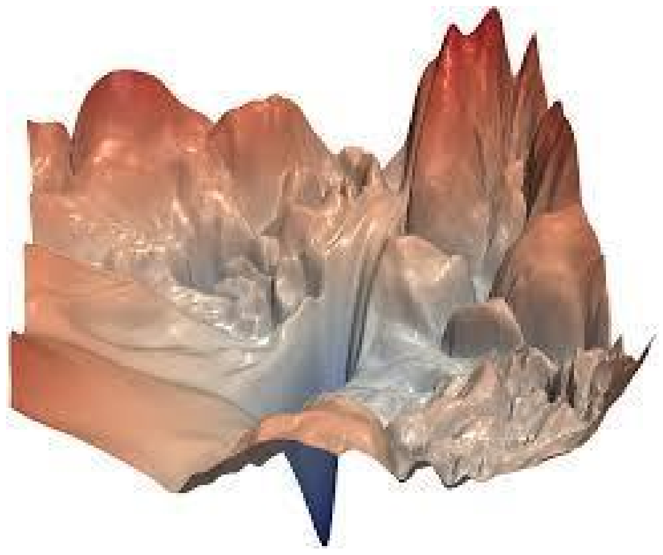


Figure 6: Gradient Descent with millions of weights, Source: [LI17]

One more factor influences the potential of a neural networks training, this is the learning rate which is how big or small a step is taken in the direction of the gradient. As discussed before there is a global minima which is the best result of the gradient descent but there are local minima which the algorithm can become stuck on, due to a low learning rate. However, a learning rate that is too large will create diverging results.

A great way to get the best of both a large and small learning rate is to have an adaptive rate, therefore it is not fixed and will depend upon the size of gradient, size of a weight or how fast the learning occurs. Some adaptive learning rate algorithms include Momentum, Adagrad, Adadelta and Adam [QIA99] [DUC11] [ZEI12] [KIN14].

Unfortunately it is quite computationally expensive to work out these gradient points across a whole dataset. This is why it is best to take a mini batch of data points, compute the gradient with respect to this batch and then take another and work out an estimate at each step. This creates a more accurate estimation of gradient and an increase of the learning rate. Taking it another step further, these batches can be parallelized across a few GPUs or CPUs enabling even faster learning.

To finalize the basics of deep learning there is a problem called generalization, this is actually a problem across all of machine learning. Ultimately in machine learning we use models that accurately describe data. This implies that models should be generalized to test data not just training data.

A technique called regularization can help deal with these problems by stopping the model memorizing the training data [GIR95]. One form of regularization is called drop out where activations are set to 0 during training, this is a random effect and stops the network relying on a perceptron. Another technique is early stopping, where the network is stopped when there is a large divergence in training and test data's cost function. This will stop memorization from occurring. [GOO16]

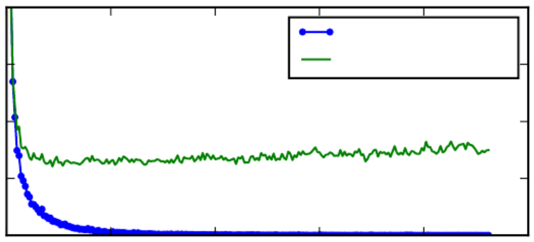
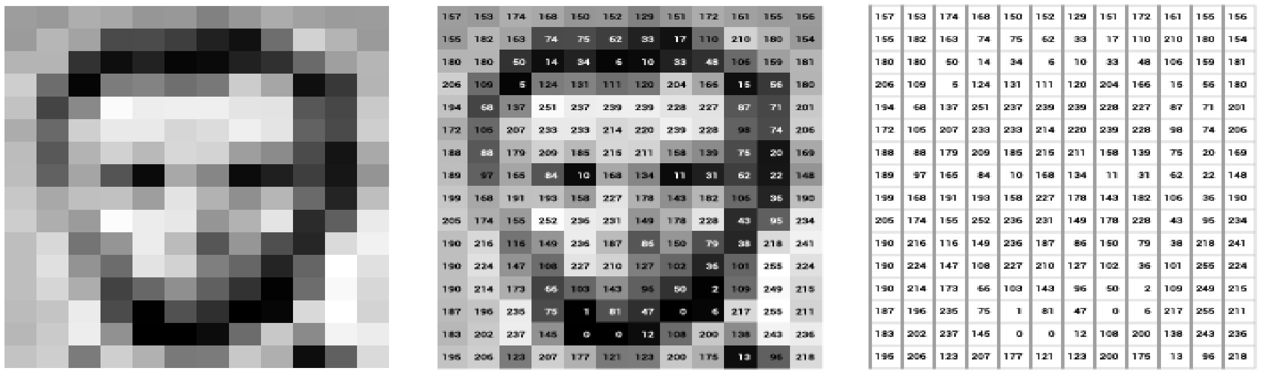


Figure 7: Learning curve between training(blue) and the test(green) showing the training iterations memorizing the dataset and diverging from the test iterations. Source: [GOO16]

Computer Vision and Convolutional Neural Networks

One of the largest known applications of computer vision is facial recognition whether it is emotional recognition or personal facial recognition for authorization. In the previous sub heading, it was shown as an example how a neural network creates an output from a face, defining lines and shapes, eventually defining noses and eyes etc., finally recognizing a face. This specific field has now expanded into health care and medicine. When the facial recognition task is swapped for a disease recognition task, the techniques stay the same while the results are profoundly different.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Figure 8: Pixel data of Lincoln numbered from 0-255 to represent shading. Source: [MIH19]

So, how can a computer see? Obviously it does not see the conventional manner but actually has tailored information as pixels and each pixel has a value from 0 to 255, which form a one dimensional array, this is only applicable to grayscale images.

A computer sees a coloured image in a similar way, it's just three layers of the same image stacked on top of each other but each layer is either Red, Green or Blue. This is where RGB comes from.

Computers decipher an image using techniques such as regression and classification. In classification an algorithm will predict a single label for an image. An example of this is to take an image of a dog and classify it from a list of animals, each animal in the list has a probability of successfully being labelled. This is usually done by classifying features of each listed item. An example of this is a dog is larger than a cat, a dog has 4 legs compared to a fish having none.

This becomes extremely difficult as images play host to such variation like; illumination conditions, scale, clutter, viewpoint variation and intra class variation (e.g. breeds of dog).

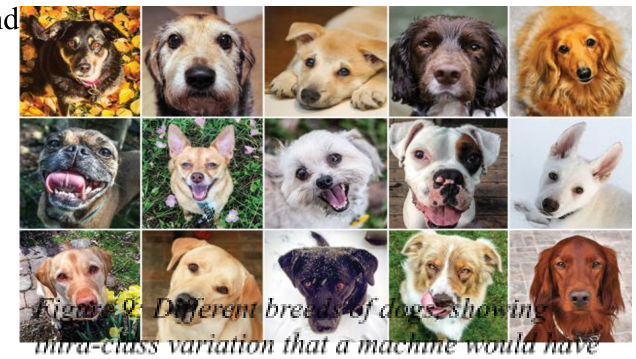


Figure 9: Different breeds of dogs showing intra-class variation that a machine would have trouble processing.

Using a dense neural network of the image's pixels where each pixel value is stored in a one dimensional array would not be feasible because the image loses all structure in the one dimensional array, as well as, each pixel of the input is connected to each neuron in the hidden layer creating a huge amount of parameters to process.

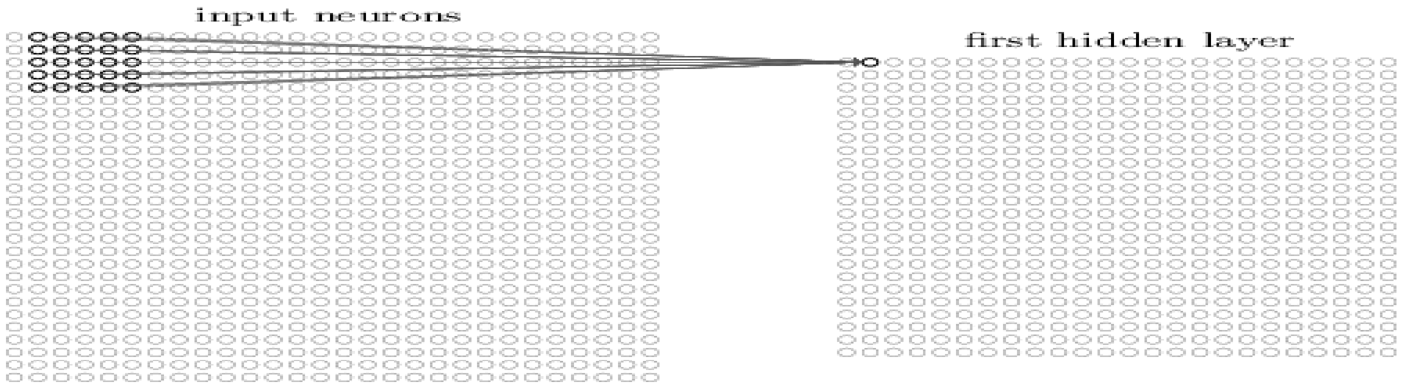


Figure 10: A patch from the 2-dimensional input connected to one neuron of the first hidden layer. Source: [NIE15]

To keep the vital image structure in a 2 dimensional image, patches of the image can be connected to a single neuron within the hidden layer, thus eliminating both problems [NIE15]. Each of these pixels are then weighted and summed. To take into account the whole image, the patch is moved over 2 pixels in the array and connects to the next neuron of that layer. This mathematical technique is known as convolution.

So, what use is this? Well, each of these patches will be compared and as was discussed in the sub-heading, neural networks, an important basis to conclude an output are features. Therefore, these patches make it quite easy to compare features and if two images share a lot of the same features they should cohere to the same object.

Each patch is a mini image that is a two dimensional array of numbers. Overlaying the patch on the image can create an output of that patch in the specific location of the image A sum is then gathered of areas that match pixels this output then creates a feature map. If you use different weighted patches on the original image you may impact the feature map, creating a sharper image, brighter or to detect edges and outlines.

From this technique convolutional neural networks are created. They are then used in three steps; applying patches on the image to generate a feature map, applying a non-linearity function and down-sample each feature map by pooling. If there are a number of features that must be extracted from an image, a layer is added for each feature. A non linearity function often used within CNN is the Rectified Linear Unit (ReLU) function which would be introduced after each convolutional layer. This function in particular is a pixel by pixel action that replaces any negative values with zero [NIE15].

Then pooling makes the network impartial to different image resolutions. The max pooling technique takes the max value of a patch and adds it to a new patch. This shrinks the dimension of the image while keeping its structure.

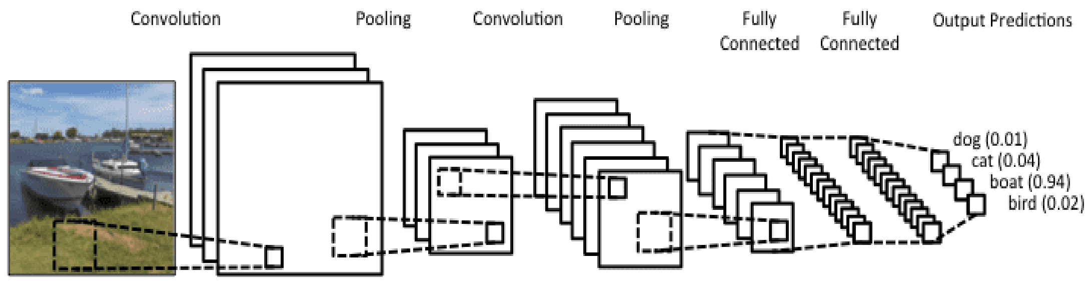


Figure 11: Convolutional Neural Network altogether with classification of the image Source: [BES17]

LeNet

Formed in 1998, known as the first CNN. LeNet was performed on handwritten digits and could classify digits with distortions [LEC95]. The CNN consisted of five layers of convolution, pooling and two fully connecting layers in a feed forward network. Other Networks at the time only processed individual pixels which was a hardware burden. LeCun, the network's creator, used convolution to identify features in many areas of a photograph using very few parameters for the given task.

AlexNet

Named after its creator Alex Krizhevsky, AlexNet won the ILSVRC- 2012 by reaching a test error rate of 15.3%, in comparison second place was 26.2% [KRI12]. This network is widely known as the first deep CNN to show groundbreaking results in image recognition and classification. It reached this standard by implementing deeper layers and using parameter optimization. Reaching 8 layers over LeNets' 5 increases generalization but is more affected by overfitting. AlexNet overcomes overfitting by randomly skipping transformation units while training while also using larger filter sizes. The ReLu function was then added to address the vanishing gradient problem.

VGG

The VGG architecture made it to 19 layers deep by opting for smaller filter sizes [SIM15]. These filter sizes (3x3) lowered computational complexity and still regulates the network by adding (1x1) sized filters between these layers of convolution. The network was optimized by using max pooling after convolution and padding was added which maintained the image integrity. VGG is extremely simple but even with its use of smaller filter sizes, it is computationally expensive as it uses over 138 million parameters, this larger parameter size makes the VGG architecture a bad choice for running on hardware with small amounts of resources.

GoogleNet

As the winner of ILSVRC-2014, GoogleNet has a lower computational complexity than other networks. It puts forward the idea of inception blocks that use merge, transform and splits [SZE15]. The inception block is made up of different sized filtered convolutional layers (1x1, 3x3, 5x5), this allows the network to work on an image at many levels.

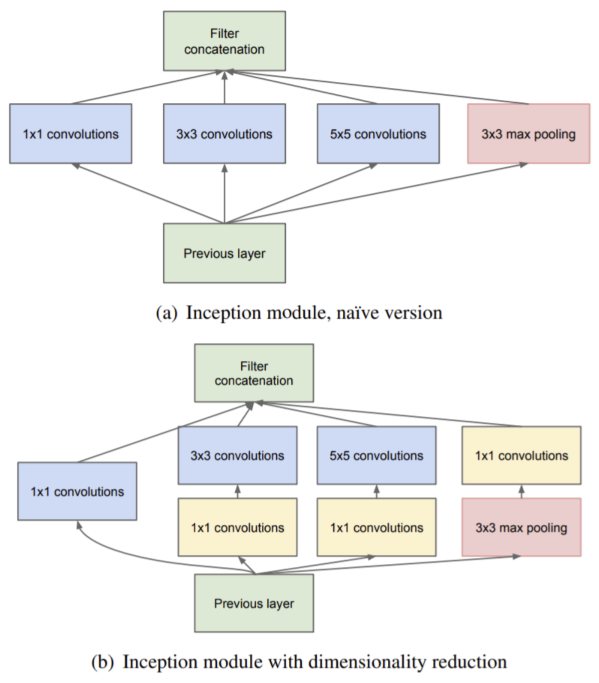


Figure 12: The Inception models in GoogleNet. Source: [SZE15]

The network was running at 138 million parameters, however, using random feature map connections and global average pooling layers, the network was reduced to 4 million parameters. As well as this, the computational complexity was reduced by using a smaller filter (1x1), which bottlenecks the inception blocks.

ResNet

Residual Learning was introduced to the CNN space in the ILSVRC-2015 competition where it won using a 152 layer architecture [HE16]. Not only has ResNet achieved deeper layers but has been shown to be less computationally expensive than VGG and AlexNet [SIM15] [KRI12]. ResNet makes use of a feature from Highway Networks, which has the ability to skip connections and pass on an identity to the next layer.

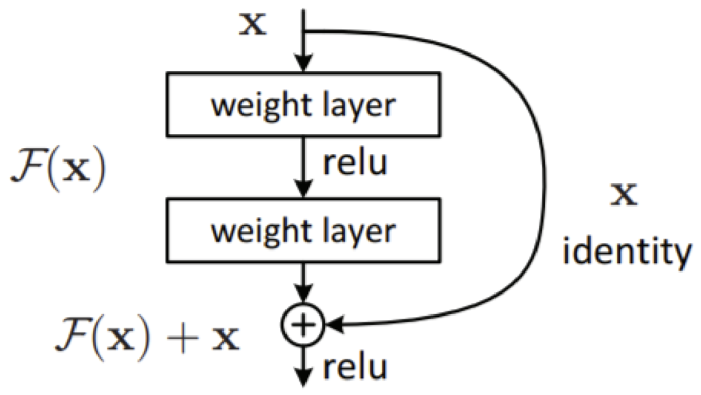

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 13: A residual learning block. Source [HE16]

When dimensions are increased in the network the skip connection can use a (1x1) filter convolution and a stride of 2 to perform identity mapping, this has no additional cost.

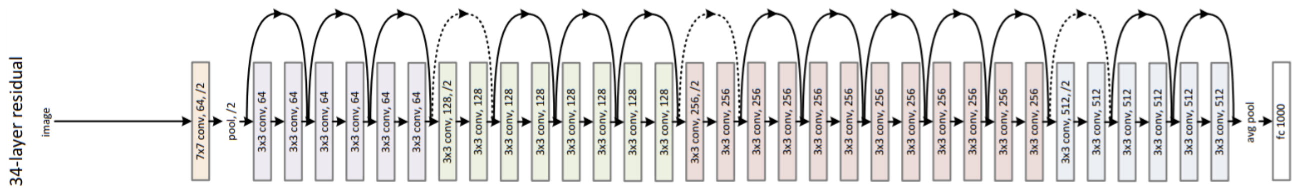


Figure 14: A 34 layer ResNet using normal skip connections (black line) and (1x1) skip connections (dotted line) [HE16].

Architecture Conclusion

To make a UAV navigate paths, roads and tracks all from an image is a tricky task but becomes much easier with the use of convolutional neural networks. This network will learn turning angles by assessing a labelled dataset of driving and walking footage. These turning angles will have 3 categories; Left, Right and Straight. All within correlation of 180°, broken down, that becomes Left = -90° <-> 0°, Straight = 0°, Right = 0° <-> +90°. The greater the path shifts to that part of the image the greater the turning angle becomes in that direction.

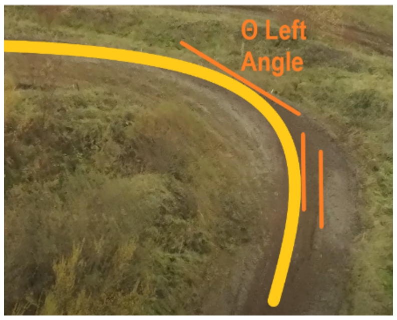


Figure 14: An aerial view of a left turn

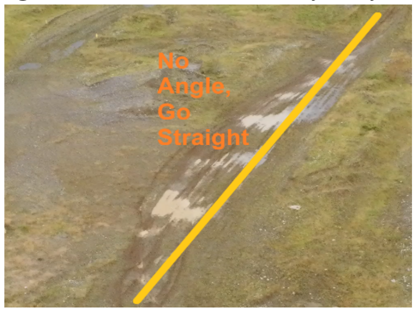


Figure 15: An aerial view of a straight track

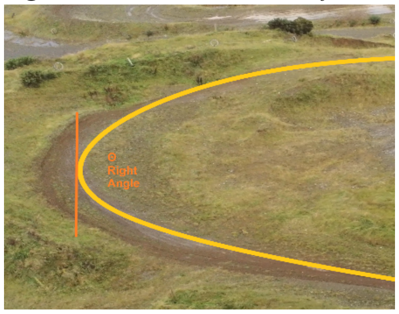


Figure 16: An aerial view of a right turn.

To get the most out of this technique the ResNet [HE16] architecture will be implemented with the ReLU nonlinearity activation function. From here the mean-squared method with categorical cross-entropy may be used to train predictions [LOQ18]. This Network is chosen due to it's smaller computational complexity while still maintaining excellent learning results, avoiding the vanishing gradient problem and if overfitting occurs a dropout function can be easily applied or a layer can be easily taken away.

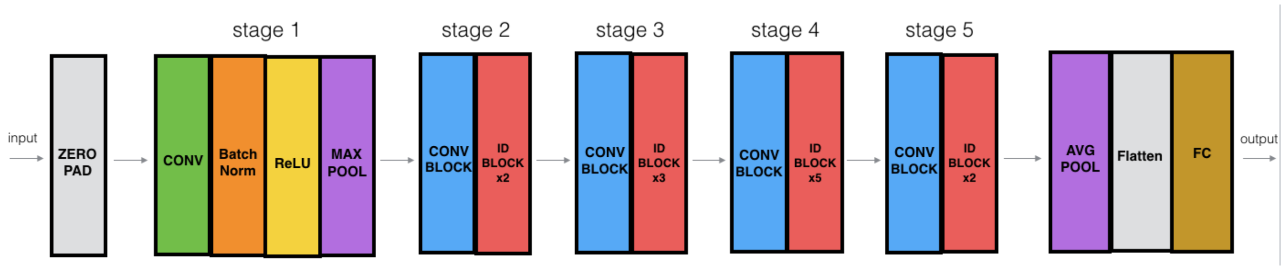


Figure 17: ResNet Architecture, Source: [LOD17]

Dataset

The dataset is one of the most key and influential aspects in development of a successful machine learning project. There are many datasets online which cater for real world computer vision problems. To solve the problem of drone direction a: trail, road or path must be clearly visible. Unfortunately, images are not this plane and have many different aspects such as lighting, clutter and blur. Thus, a very good dataset with a lot of variability is needed.

To heighten the learning outcomes of the task, I hope to develop my own dataset from video's of driving, walking and from the drone. These videos will be taken at 30fps in 1920 x 1080p using the camera on the Parrot Anafi as well as the One Plus 7 Pro. Some of this data will be taken from walking with my phone strapped to my body, driving with my phone in a stable mounted position and also from drone flights on small paths. All images will then be labelled as to what class it lies in Left Turning, Right Turning or Straight.

This should produce a substantial amount of variant images in different lighting, different path quality, and road driving data.

From this dataset subsequent sets will be made in relation to training and a test set. However, care must be taken to have equal representation of image classification in both subsets. The test set should also be completely original to the training set incase of training memorization. The test set does not need to be as large as the training set and will be split 20/80 respectively.

Tools

Now that the Architecture is known, the question is how do we implement this. There are many tools, languages and libraries out there to implement deep learning. Another key aspect of the task at hand is to have everything synchronized and communicating with each other. The Parrot Anafi has its own SDK therefore the tools used must be interchangeable with that.

Sphinx

This is a simulation program which runs Parrot's firmware on a PC in a virtual environment. It uses another software called Gazebo, this software creates complex indoor and outdoor environments [PAR20]. The Sphinx program can visualize flight data, modify virtual behavior, connect to a controller and run remotely. Unfortunately, using this software creates an issue, it can only be installed on ubuntu 18.04. This will prompt the use of Oracle's Virtual Box software which enables running of virtual operating systems [PAR20].

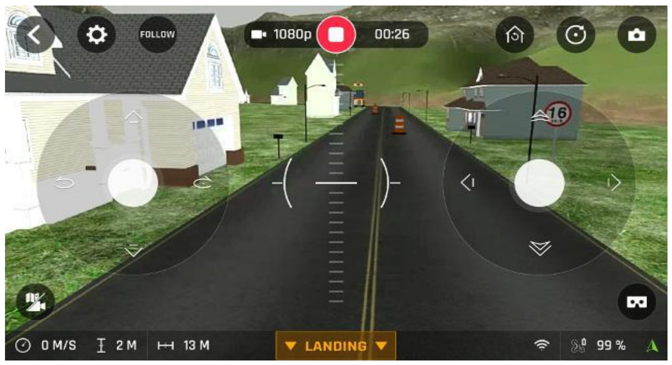


Figure 18: Sphinx Simulation environment. Source: [PAR20]

Olympe

The Olympe program provides a python controller programming interface for any drone [PAR20]. Based on arsdk-ng and primarily used with Parrot's Sphinx program, Olympe can connect and control a drone using Python scripts from a PC. With an extensive API, it's easy to implement commands and receive statuses and event messages from a drone, also providing the ability to record data such as video and metadata.

Python

As much of the software relates to python, it will be used to implement the deep learning architecture as well as drone communication through Olympe. Python is a dynamically typed language and also uses compound data types. Also providing users with a garbage collector it is renowned for being beginner friendly and having extensive data science libraries.

TensorFlow and Keras:

TensorFlow is a large open-source library used in machine learning developed and maintained by Google's brain team. To implement the architecture seen in previous sub-headings we will use TensorFlow for training [TEN20]. In general it is a large library of neural network algorithms and models, most of these algorithms are mathematical solutions. Companies like PayPal, GE Healthcare and Airbnb use it.

Tensorflow makes use of easy to implement layered components which make creating deep networks much easier.

Keras is an easy to use high-level API within TensorFlow. Models are created by matching building blocks together and the library provides a lot of room for innovation and development.

PyTorch

Pytorch is a deep learning library with native support for Python and use of all Python libraries. It is developed by Facebook's AI research lab but is open source. It offers simplicity, optimal memory use and computational graph construction [PYT20].

OpenCV

To process images from the data set we will use OpenCV. This library is used with NumPy to solve computer vision, it uses over 2500 algorithms to detect objects, track them and is also used in facial recognition. It has many interfaces to make it available on the most popular programming languages such as C++, Python and Java [OPE20].

Summary and Conclusion

The final product of this research will be a UAV using Convolutional Neural networks that can safely navigate paths, trails and roads. It is hoped that the UAV will generalize it's input and be applicable in any directional based scenario such as a hallway or lane due to its proposed diverse dataset. The CNN will process an image input to give a general weight of direction due to pathway positioning.

This technology, in a UAV, has many real life use cases such as search and rescue, road mapping and surveying. Without the need for any man input and easily manipulated to other tasks it is quite a powerful tool.

To train the model that will implement autonomy to the UAV, Tensorflow/Keras will be used. These libraries have easy to use APIs while also being extremely fast with great visualization and optimization characteristics. To complement Parrot's Olympe software, Python will be used as the main coding language. All these technologies will be working in an Ubuntu virtual environment as this is the only running environment Olympe can work in.

The model itself will take after the ResNet model due to it's lower computational complexity which can be benefited from the drones weaker hardware, if it is to be embedded.

References

[BES17] Besbes, A., 2017. *Introduction to CNNs*. Retrieved from Ahmedbesbes: https://www.ahmedbesbes.com/blog/introduction-to-cnns

[CIR12] Ciregan, D., Meier, U. and Schmidhuber, J., 2012, June. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3642-3649). IEEE.

[DUC11] Duchi, J., Hazan, E. and Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

[GIR95] Girosi, F., Jones, M. and Poggio, T., 1995. Regularization theory and neural networks architectures. *Neural computation*, *7*(2), pp.219-269.

[GOO16] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016. *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press.

[HE16] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[IAA20] IAA, 2020. *Drone regulation and guidance*. Retrieved from IAA: https://www.iaa.ie/general-aviation/drones/drone-regulations-guidance

[ING16] Ingle, S. and Phute, M., 2016. Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, *3*(9), pp.369-372.

[KAH20] Khan, 2020. *chain rule introduction*. Retrieved from khanacademy: https://www.khanacademy.org/math/ap-calculus-ab/ab-differentiation-2-new/ab-3-1a/v/chain-rule-introduction

[KAM18] Kampakis, S., 2018. *What deep learning is and isnt*. Retrieved from The Data Scientist: https://thedatascientist.com/what-deep-learning-is-and-isnt/

[KAT18] Kathuria, A., 2018. *intro to optimization in deep learning gradient descent*. Retrieved from paperspace: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[KIE52] Kiefer, J. and Wolfowitz, J., 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, *23*(3), pp.462-466.

[KIN14] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[KRI12] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, pp.1097-1105..

[LEC95] LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P. and Vapnik, V., 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, *261*(276), p.2.

[LI17] Li, H., Xu, Z., Taylor, G., Studer, C. and Goldstein, T., 2017. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

[LOD17] Lodaya, T. 2017. *keras-signs-resnet*. Retrieved from Github: https://github.com/tejaslodaya/keras-signs-resnet/blob/master/README.md

[LOQ18] Loquercio, A., Maqueda, A.I., Del-Blanco, C.R. and Scaramuzza, D., 2018. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, *3*(2), pp.1088-1095.

[MIH19] Mihajlovic, I. 2019. *Everything you ever wanted to know about computer vision*. Retrieved from towards data science: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

[NIE15] Nielsen, M.A., 2015. *Neural networks and deep learning* (Vol. 25). San Francisco, CA: Determination press.

[OPE20] opencv, 2020. *about*. Retrieved from OpenCV: https://opencv.org/about/

[PAR18] Parrot, 2018. *Anafi*. Retrieved from Parrot: https://www.parrot.com/en/drones/anafi

[PAR20] Parrot, 2020. *Olympe*. Retrieved from Parrot: https://developer.parrot.com/docs/olympe/overview.html

[PAR20] Parrot, 2020. *sphinx*. Retrieved from Parrot: https://developer.parrot.com/docs/sphinx/whatissphinx.html

[PAR20] Parrot, 2020. *Virtual Box*. Retrieved from Parrot: https://developer.parrot.com/docs/sphinx/virtualbox.html

[PYT20] Pytorch, 2020. *Get started*. Retrieved from Pytorch: https://pytorch.org/get-started/locally/

[QIA99] Qian, N., 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), pp.145-151.

[SIM15] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[SZE15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[TEN20] TensorFlow, 2020. *About*. Retrieved from TensorFlow: https://www.tensorflow.org/about

[ZEI12] Zeiler, M.D., 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Plagiarism Declaration

Declaration

- I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name: Josh Hudziak

Student Number: C00231846

Signature: Josh Hudziak

Date: 30-04-21