City University of London

MSc in Artificial Intelligence

Project Report

2021/22

Ablation Study on Faster-RCNN for Hepatocellular Carcinoma Detection

Benjamin Dwumah

Supervised by: Dr. Alex Ter-Sarkisov

December 21, 2022

**Declaration**

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: Benjamin Dwumah

**ABSTRACT**

Hepatocellular carcinoma (HCC) is one of the most prevalent causes of cancer incidences and deaths. Despite many years of research and the creation of new medical interventions, patients with HCC continue to have poor treatment outcomes. Patients with HCC suffer from unfulfilled concerns like risk prediction, individualised treatments, accurate prognosis and early diagnosis. In recent years, there has been a massive growth in Artificial Intelligence (AI) applications in medical research, and the field of HCC is no exception. Deep learning algorithms are among the most advanced AI-based machine learning algorithms for processing and analysing complicated multimodal data, from routine diagnostic factors to high-resolution medical images. In this research project, I present my experiment results and review for early diagnosis of HCC using deep learning techniques, specifically Computer Vision. I have done detailed experiments on the object detection model Faster Region-Based Convolutional Neural Network (Faster R-CNN) for detecting HCC. I experimented with different backbones for the Faster R-CNN model. I concluded that the backbone plays a significant role in the Faster-RCNN architecture for good accuracy results and performance. The codebase is available at: <https://github.com/Ben74x/Indiv_Proj>

Keywords: Hepatocellular carcinoma (HCC), Artificial intelligence, Deep learning, Computer Vision, Object Detection, Faster R-CNN.

**1. Introduction**

Hepatocellular carcinoma (HCC) is an aggressive primary liver cancer that develops in the setting of chronic parenchymal liver diseases and is among the top causes of cancer incidence and mortality worldwide (Bray et al., 2018; Yang et al., 2019). While the hardship of having HCC has decreased with effective antiviral therapy, HCC cases associated with metabolic syndrome are expected to rise further due to the significant increase in the commonness of non-alcoholic fatty liver disease (NAFLD) in the general population (Stepanova et al., 2017).

Decades of research in HCC have resulted in the development of a screening protocol, non-invasive imaging-based diagnostic modalities, and different treatment modalities, including surgical, locoregional and systemic therapies (Llovet et al., 2021; Yang and Heimbach, 2020). Nevertheless, the results of patients with HCC continue to be poor. There are areas of critical unmet demand in early detection, accurate prognostication, risk prediction, and individualized treatments.

A lot of health data is produced by HCC patients. While this is exciting for researchers, guaranteeing that such large amounts of data are converted into actionable insights can be difficult. Artificial intelligence (AI) is considered to be capable of synthesising and analysing multimodal data with extraordinary levels of accuracy, and the use of AI to several areas of medicine, including hepatology, has grown rapidly in recent years (Ahn et al., 2021). For a wide range of tasks and clinical applications, such as image classification, detection and segmentation, etc, AI-based concepts offer a variety of techniques. Recent developments in AI, specifically in the field of medical image analysis, provide a vast array of automated tools for obtaining precise measurements of biomarkers, exposing delicate features, categorising tissue characteristics, and conducting radiomics for in-depth analysis of raw imaging data. The introduction of deep learning techniques has made the AI revolution of the past ten years conceivable. This research analyses how the object detection model, Faster R-CNN, performs in the detection of HCC lesions.

The main question for this research study is: *“Can customized Faster R-CNN models detect HCC lesions better than the base model?”*

**1.1 Objectives, Project Product, and Beneficiaries**

In this project, my goal is to research, understand, and experiment the Faster R-CNN model to see if it is appropriate for use in the detection of HCC by contrasting the base model with modified versions. The project's final product is a fully operational Faster-RCNN model that has been trained on data from liver cancer patients with HCC and can be used in real-world scenarios.

The following are the project deliverables:

* Extensive research on the Faster R-CNN model for HCC lesion detection
* Results of using the top-performing model to find HCC lesions in ultrasound scans

By incorporating the top performing algorithm in this project into their applications, researchers from many fields of study, as well as medical professionals working in the field of cancer, can gain from this project. Additionally, this project offers thorough information concerning the effectiveness of the Faster RCNN algorithm and its contributing factors to the performance of HCC detection, which can aid in future research. Furthermore, anyone with a foundation in machine learning will also gain from developing their knowledge of deep learning, particularly computer vision.

**1.2 Structure of the Project Report**

In Section 2, I discuss Artificial Intelligence and its applications in the health sector. I also detail machine learning, its types and limitations. I then discuss deep learning and give a brief overview of the computer vision and its applications in the health sector. Finally, I describe the concept of object detection and talk about the Faster R-CNN algorithm discussing all the essential parameters.

In Section 3, I discuss the method of experiments involving the data for the project, the model, other external tools and evaluation metrics.

In Section 4, my findings from the experiments are presented along with graphs, diagrams and figures.

In Section 5 and Section 6, I introduce the project results and re-evaluate the research question with regard to the experiments conducted. I then discuss the results and conclude my project report.

**2. Context**

**2.1 Artificial Intelligence**

**2.1.1. Definition**

In a 2004 study, John McCarthy gave the following definition of artificial intelligence (AI), despite the fact that there have been numerous other definitions over the past few decades, "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable" (McCarthy 2004, p. 2).

Turing, known as the "Father of Computer Science," posed the question, "Can machines think?" in this paper. From there, he proposed a test which is now popularly known as the "Turing Test". In the test, a human interrogator attempts to differentiate between a computer and a text response from a human. While this test has been heavily scrutinised since its publication, it remains an essential part of the history of AI in addition to an ongoing concept within philosophy due to its use of linguistic ideas.

In recent times, a book by Peter Novig and Stuart Russell, Artificial Intelligence: A Modern Approach has become one of the leading learning materials in the study of AI. In it, they explore four potential objectives or definitions of AI, differentiating between computer systems based on their reasoning and thinking vs acting:

* Human approach:
  + Systems that think like humans
  + Systems that act like humans
* Ideal approach:
  + Systems that think rationally
  + Systems that act rationally

Comparing Alan Turing’s definition to Peter Novig and Stuart Russell own, his definition would fall under the human approach which is systems that act like humans.

In its basic form, AI can be defined as an area incorporating robust datasets and computer science to solve problems. It has the subfields of machine learning and deep learning, which are widely discussed in the context of artificial intelligence.

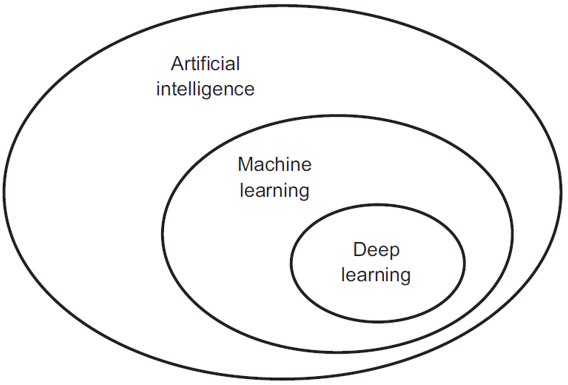


Fig. 1: Artificial Intelligence and its subfields

AI is grouped into two types which are Weak AI and Strong AI. Weak AI, known as Narrow AI, has been trained and focused on performing specific tasks. The majority of AI applications today are driven by weak AI. The type of AI is anything but weak; it powers many applications like self-driving cars, voice assistants, personalized marketing, facial recognition systems, and gamified therapy. Strong AI comprises Artificial Super Intelligence (ASI) and Artificial General Intelligence (AGI). A computer with an intellect comparable to humans, a self-aware awareness, and the capacity to learn, reason, and make plans for the future would be said to have AGI, also known as general AI. ASI, also known as Superintelligence, would be more intelligent and capable than the human brain. Even though there are now no real-world applications for strong AI and it is only theoretical, experts in the field of artificial intelligence are continuously studying its potential.

**2.1.2. AI APPLICATIONS IN HEALTHCARE**

Healthcare delivery may change as a result of AI. It can boost output and care delivery efficiency, enabling healthcare systems to serve more people with more effective treatment. AI can assist healthcare professionals in having a better experience, permitting them to spend more time providing direct patient care and lowering burnout. One of the biggest success stories in our time is healthcare. Life expectancy has increased globally due to significant advancements in medical technology. However, as people live longer, healthcare systems must contend with expanding patient demand, rising expenditures, and a stretched-thin staff. Population ageing, shifting patient expectations, a change in lifestyle preferences, and the endless cycle of innovation are just a few of the inescapable drivers that drive demand. The effects of an ageing population stand out among these. One-fourth of the people in North America and Europe will be over 65 by 2050, which means the healthcare systems will be required to cope with more patients with complicated demands. It is expensive to manage these patients, and systems must change from a philosophy of periodical care to one that is considerably more proactive and centred on long-term patient care.

The expenditure on healthcare is growing. Healthcare systems will have difficulty staying sustainable unless significant structural and innovative changes are made. Health systems require a larger workforce as well. However, while the world economy could generate 40 million new healthcare jobs by 2030, the World Health Organization predicts a 9.9 million physician, nurse, and midwife shortage over the same time period. Not only must we attract, train, and sustain more medical professionals, but we must also ensure that their time is spent where it adds the most value, which is caring for patients. AI, which is based on automation, has the potential to revolutionise healthcare and assist in addressing some of the issues raised above. AI can improve care outcomes as well as the efficiency and effectiveness of care delivery. It can also improve healthcare practitioners' daily lives by allowing them to spend more time caring for patients, increasing staff morale and retention. It can even help bring life-saving treatment methods to market faster. Simultaneously, concerns have been expressed about the influence AI may have on patient populations, professionals, and health systems, as well as the risks involved; there are ethical debates about how AI should be used.

According to Spatharou, Hieronimus and Jenkins (2020), a growing number of governments have set forth goals for AI in healthcare, and several are making significant investments in the field. Venture capital (VC) expenditure for the top 50 companies in healthcare-related AI reached $8.5 billion, and huge tech companies, startups, pharmaceutical and medical device companies are all involved in the developing AI healthcare ecosystem.

**2.2. Machine Learning**

**2.2.1 Definition**

Machine learning is a subset of artificial intelligence. Machine learning focuses on data driven learning, whereas artificial intelligence focuses on general intelligent behaviour. Machine learning is the process by which a computer learns from data. The program takes in data, and from the data, the program learns. In the process, the program creates a model, which can then be used to make predictions about future data. The predicted values are usually, but not always, probabilistic in nature. That is, the model is a mathematical model which gives probabilities for events in the data. However, the model is only sometimes probabilistic. For example, decision trees are not probabilistic, but they are a model of the data.

A more detailed definition of machine learning would be “A computer program is said to learn from experience E with respect to some class T and performance P, if its performance at tasks in T, as measured by P, improves with experience E” (Mitchell 1997, p. 2).

Machine learning is a vast and vital field of study. The practical significance of machine learning cannot be overstated. Many of the applications we use every day are machine learning applications. Email apps, for example, use machine learning to distinguish between spam and legitimate emails. It is also used in search engines to prioritise web pages, in speech recognition applications to recognise speech, and in image recognition applications to recognise images.

**2.2.2 Types of Machine Learning**

Machine learning algorithms can solve a wide range of problems. Machine learning algorithms are currently trained using four distinguished methods: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

Machine learning algorithms can be trained in a variety of ways, each with its own benefits and drawbacks, like any other method. We must first examine the types of data that each type of machine learning consumes in order to comprehend the benefits and drawbacks of each type. Labelled data and unlabelled data are the two types of data used in machine learning. Although labelled data has both the input and output parameters in a properly machine-readable form, labelling the data initially takes a significant amount of human effort. In unlabelled data, only one or none of the parameters are present in machine-readable form. This eliminates the need for human work but calls for more complex solutions. Furthermore, certain machine learning algorithms have extremely specific applications; yet, the four primary approaches are still in use today.

Supervised learning is the most prevalent kind of machine learning. In supervised learning, the algorithm analyses labelled data. In other words, the data has some marking, and for each input, the appropriate output is given to the application. Numerous applications include supervised learning. Email applications, for instance, employ supervised learning to categorise emails as spam or not. Applications for image recognition identify images using supervised learning. Applications for speech recognition classify sounds using supervised learning. Constructing a model that links inputs to outputs is the aim of supervised learning.

Learning from unlabelled data is referred to as unsupervised learning. The algorithm gathers information and makes learning from it. However, the algorithm is not informed of the proper output. The algorithm understands relationships between data points in an abstract way; human input is not necessary. Unsupervised learning methods are flexible because of the development of these hidden structures. Unsupervised learning algorithms can modify their underlying structures dynamically to respond to the data rather than using a predetermined and stated problem statement. This provides more post-deployment development than supervised learning techniques. There are numerous uses for unsupervised learning. Unsupervised learning may be used by an email application, for instance, to group spam emails.

Machine learning techniques such as semi-supervised learning allow the algorithm to learn from labelled and unlabelled input. The use of semi-supervised learning is widespread. Semi-supervised learning, for instance, might be used by an email application to categorise emails as spam or not. The algorithm may use unlabelled data to identify spam email clusters. Then, it might classify emails as spam or not using labelled data. Better models can be learned via semi-supervised learning than with supervised learning alone.

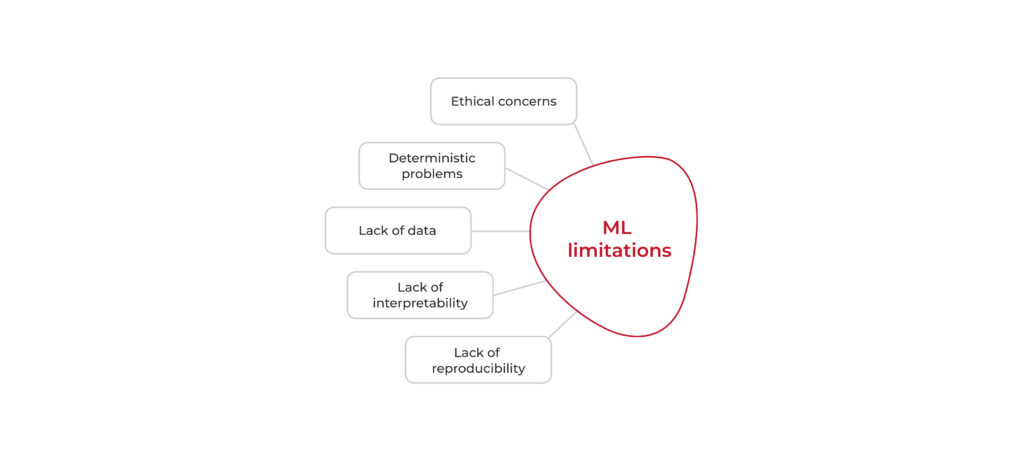
Reinforcement learning is a subset of machine learning in which the algorithm learns through "trial and error" experiments. The algorithm starts with an initial state and then acts. The algorithm is then rewarded or punished for its actions. The process is then repeated, beginning with the initial state until the algorithm has explored the space of possible actions. Reinforcement learning is used in a wide range of applications. For example, a chess programme might use reinforcement learning to teach itself how to play the game. The goal of reinforcement learning is to learn a policy that determines each state's best course of action.

**2.2.3 Machine Learning Limitations**

Developers of machine learning systems have been able to make these systems think more like humans in recent years, executing difficult tasks and coming to choices after doing in-depth analysis. Robots doing a variety of professions for humans is not a sci-fi movie premise anymore; it is a reality. However, there are still a number of machine learning algorithms' limitations, despite the advancements ML developing teams have achieved in this area.

Although machine learning is incredibly helpful for many applications, it isn't always the best option. Implementing machine learning applications isn't always essential, it's not always a good idea, and it sometimes even makes things worse. The world has been greatly touched by ML. According to Harari (2015) in his famous book HOMO DEUS, we are gradually moving towards an ideology known as dataism, which means individuals will embrace data and algorithms more than their own personal convictions. There have been cases already on this ideology on how people trust GPS instructions to take them to their destinations. People have occasionally hit roadblocks after blindly following a navigation device's recommendations without once consulting a map or the people around them.

When developing a project that needs to process a lot of data, machine learning offers a creative way of accomplishing it. But one thing we must take note of is that, there are crucial factors that should be taken into account before deciding to use ML as a tool to build applications. One needs to be aware of the potential drawbacks and limitations of this formidable technology before putting it into use. There are five basic areas in which ML issues can be categorized as seen in the figure below.

Fig. 2: Limitations of Machine Learning

* **Ethical Concerns**: There are many benefits in using machine learning algorithms. The use of these algorithms to automate tasks, evaluate vast volumes of data, and make complex decisions has helped humanity a lot. However, these algorithms do have certain disadvantages. Bias can exist in these algorithms at any stage of development and bias cannot be completely eliminated because these algorithms are created and trained by humans. Many ethical issues are still unresolved. Taking self-driving cars as an example, who is responsible if something goes wrong. In the event of an accident who should be blamed – the driver, the automobile company or the software. One thing is certain: ML cannot independently decide on challenging moral or ethical issues. Measures have been put in place to tackle this issue with the introduction of explainable AI. However, we will need to develop a complete system in the near future to address ethical issues with machine learning technology.
* **Deterministic Problems**: Machine learning is an advanced tool which is used in a variety of fields. For example, in the meteorology field, machine learning algorithms can assist in the calibration and correction of sensors to measure pressure, temperature and humidity for the environment. Also, models can be built to simulate atmospheric emissions to predict pollution. Although machine learning algorithms are able to do such complex tasks, they are unable to comprehend the physics of a weather system. Machine learning models are capable of making these predictions but the computations of intermediate fields like density may result in negative values that defy the laws of physics. Machine learning is not capable of understanding the cause and effect relationships. Although, a neural network can connect input and output data, it cannot determine why there are related.
* **Lack of data**: Given the complex architectures of machine learning algorithms, they usually need a lot of data for training in order to function well. The amount of data needed by ML models increases with its complex architectures. Some could choose to reuse data when training these algorithms but this never produce satisfactory outcomes. Furthermore, the absence of good data is a further issue. This is distinct from merely lacking data. Consider a scenario in which your neural network needs additional data and you provide it with a sufficient amount of low-quality input. The accuracy of the model may be severely hampered as a result.
* **Lack of interpretability**: Interpretability is a significant issue with machine learning algorithms. Consider the situation where you are building a model for a financial company to identify fraudulent transactions. The model has to be able to defend how it categorizes transactions in this situation. For a task like this, a machine learning application might do well in terms of accuracy and responsiveness, but it might not be able to prove its outcomes. If machine learning techniques are to be used in practice, it is crucial that they become interpretable.
* **Lack of reproducibility**: A complex and expanding problem in machine learning called lack of reproducibility is made worse by lack of model testing procedures and transparency in code. New models are created and are quickly implemented in practical applications. Nevertheless, despite the fact that the models are created to incorporate the most recent scientific advancements, they might not function in actual situations. Various industries and professions can use reproducibility to use the same model and find solutions to issues more quickly. Safety, dependability, and the capacity to spot bias can all be impacted by a lack of reproducibility.

There is no denying that AI has provided humans with a number of exciting new opportunities. Some have, however, also come to believe that machine learning applications are capable of resolving any issue facing mankind. It is optimal for machine learning systems to be used on tasks that would normally be completed by humans. If you don't ask it to be imaginative, intuitive, or utilize common sense, it can do well. Machine learning applications are capable of learning from concrete data quiet effectively, but they lack the human capacity to comprehend the world and how it works. An ML application, for instance, can be trained to know what a cup looks like but it is unable to comprehend that the cup contains coffee. Machine learning algorithms make people's life better and their work more efficient, but it cannot completely replace them because it is not capable of performing many responsibilities. While ML has several benefits, there are also some drawbacks.

**2.3 Deep Learning**

**2.3.1 Definition**

Deep learning is a subset of machine learning concerned with creating and implementing algorithms and models capable of processing and analysing vast volumes of data. Before deep learning, we depended on classic machine learning techniques such as logistic regression, support vector machine (SVM), Bayes classifier, decision trees, etc. These classic techniques are also referred to as flat algorithms. Normally, a pre-processing stage called feature extraction is required when using these classic techniques. These traditional machine learning techniques can employ the depiction of the given raw data created by feature extraction to complete a task. For instance, we can now divide the data into classes. Feature extraction is typically a difficult process that calls for in-depth understanding of the problem area. For best results, this pre-processing layer needs to be modified, examined, and improved across numerous iterations. In deep learning, the feature extraction stage is not needed. The layers have the ability to independently learn the underlying depiction of the raw data.

Understanding that deep learning is fuelled by enormous amounts of data is essential to understanding why it has gained such popularity. The evolution of big data has opened up several possibilities for deep learning advancements. According to an article by Garling, 2015, Andrew NG explained AI in these terms: **“*I think AI is akin to building a rocket ship. You need a huge engine and a lot of fuel. If you have a large engine and a tiny amount of fuel, you won’t make it to orbit. If you have a tiny engine and a ton of fuel, you can’t even lift off. To build a rocket you need a huge engine and a lot of fuel.*”**

Using deep learning to summarize this, one can say that deep learning algorithms are the rocket and data is the fuel.

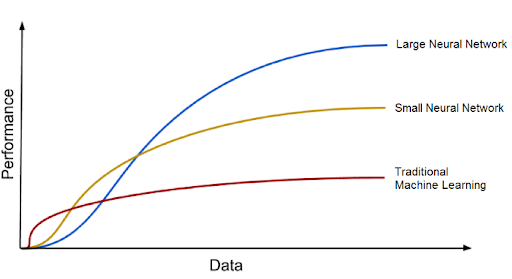


Fig. 3: Deep learning models against data

Deep learning methods continually examine data using a predetermined logical structure in an effort to reach conclusions that are comparable to those reached by humans. This is done by using neural networks, which is a multi-layered architecture of algorithms.

**2.3.2 Origin of Neural Networks**

A brain’s biological neurons serve as a model for artificial neural networks (ANNs). In simplified terms, artificial neural networks mimic several fundamental biological neural network functions. In order to draw comparisons between natural and artificial neural networks, let's first examine biological neural networks. A biological neural network is made up of many neurons.

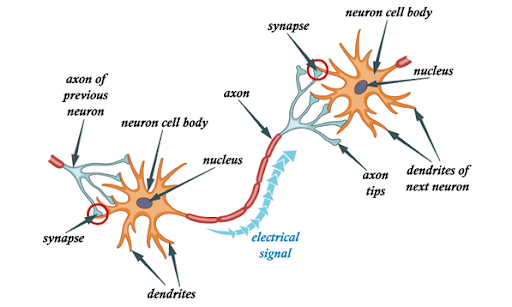
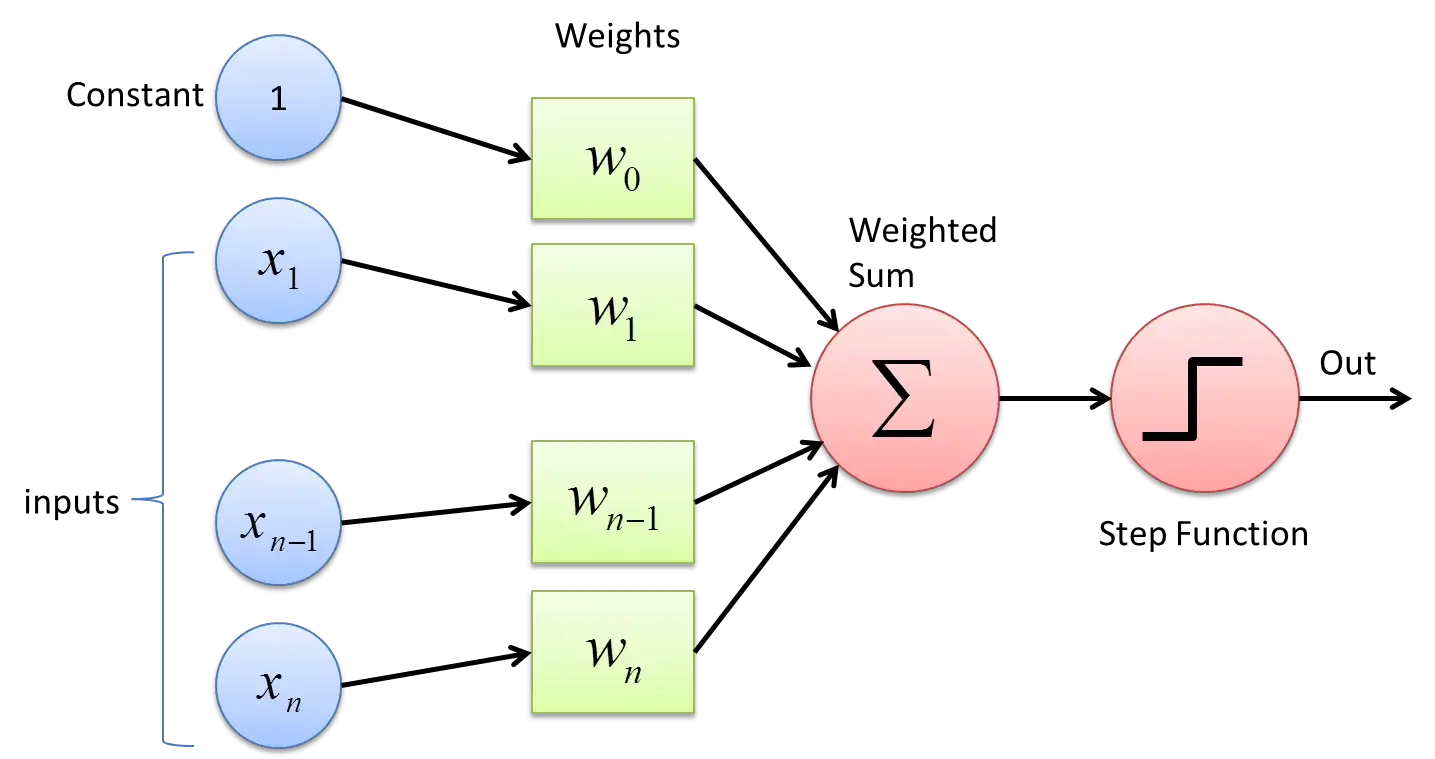
****

Fig. 4: A biological neural network model

A neuron consists of a cell body, dendrites and an axon (Woodruff, 2018). Dendrites are thin structures that protrude from the cell body. Axon is also a cellular extension that emerges from the cell body. The majority of neurons send impulses via the axon and pick up signals with the dendrites. Signals travel from one neuron’s axon to another neuron’s dendrite at the large percentage of synapses. Since voltage gradients are kept in place in the membranes of all neurons, all neurons are electronically excitable. The neuron produces an electrochemical pulse known as an action potential if the voltage fluctuates by a significant enough amount over a brief period of time. Rapidly moving down the axon, this potential trigger synaptic connections.

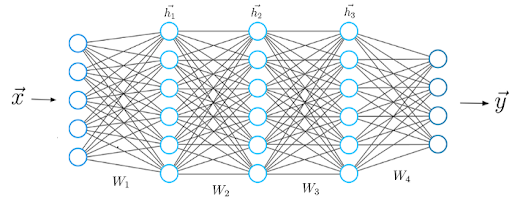
**2.3.3 Perceptron**

A Perceptron is a supervised learning algorithm for binary classifiers (DeepAI, 2019). Binary classifiers determine whether an input, which is often represented by a collection of vectors, falls into a certain category. In summary, a perceptron is a single-layer neural network which consists of input values, weights and bias, net sum, an activation function (DeepAI, 2019). Perceptron works by multiplying all of the input values by their weights. The weighted sum is then calculated by adding the sum of all these multiplied values. The output of the perceptron is then generated by applying the weighted sum to the activation function. The vital job of the activation function is to make sure that the output is transferred between necessary values like (0,1) or (-1,1).

Fig. 5: Perceptron

**2.3.4 Artificial Neural Networks**

A neural network typically consists of a group of linked nodes. These nodes are known as neurons. These synthetic neurons are based in part on the actual brain cells as mentioned in 2.3.2. In deep learning, a neuron is nothing more than a graphical depiction of numerical values. For a biological neuron, an axon is any link between two neurons. In deep learning, these connections are called weights, which are numerical values, serve to represent the link between the artificial neurons. The design of the human brain served as the inspiration for the neural network's architecture. We can train neural networks to recognize patterns and categorize various types of information, just like how our brains do it naturally. The probability of detecting and producing a proper result is increased by using the different layers of neural networks as a kind of filter that operates from coarse to fine. Like this, the human brain functions. The brain seeks to make comparisons with familiar objects whenever we get new information. Deep neural networks also make use of the same idea.

Fig. 5: Artificial neural network

**2.3.5 Feedforward Networks**

A feed forward neural network is a type of ANN in which there is no recurrence in the connections between the nodes. Since input is only processed in one channel, the feed forward model is the most straightforward type of neural network. Although the data may flow via several buried nodes, it always proceeds forward and never backward. A single layer perceptron is a common example of a feed-forward neural network in its most basic configuration. A number of inputs are introduced into the layer in this model and multiplied by the weights. The weighted input values are then summed together to produce a total. The value produced is frequently 1, and if the sum of the values is below the cut-off, the output value is -1. The threshold is typically set at zero. In classification tasks, the single layer perceptron is a crucial feed forward neural network model. Single layer perceptron can also contain some features of machine learning. The neural network may compare the outputs of its nodes with the desired values using a property known as the delta rule, which enables the network to train its weights to create output values that are more accurate. This learning and training procedure results in a gradient descent. Although the procedure for changing weights in multi-layered perceptron is almost comparable, it is more formally known as back-propagation. In these circumstances, the network's hidden layers are each changed in accordance with the output values generated by the final layer.

From Fig. 6, we can see the architecture of a feedforward neural network. The input layer is supplied with input ***x***, which is data from which the neural network learns. The neurons in the input layer are equal to the number of elements in the vector ***x***. To simplify it, every input neuron corresponds to a single vector element. The output layer, the final layer, generates a vector ***y*** that represents the outcome of the neural network. The values of the neurons in the output layer are represented by the elements in this vector. The network must carry out specific mathematical computations in the layers in between input and output layers in order to generate a prediction vector ***y***. These are called hidden layers.

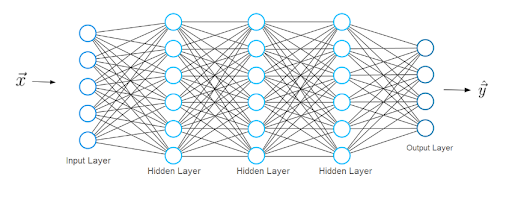


Fig. 6: Feedforward neural network

We can properly investigate the process of learning now that we have a better understanding of neural network architecture. Let's go cautiously. The neural network generates a prediction vector, that we'll refer to as ***h***, for an input feature vector called ***x***.

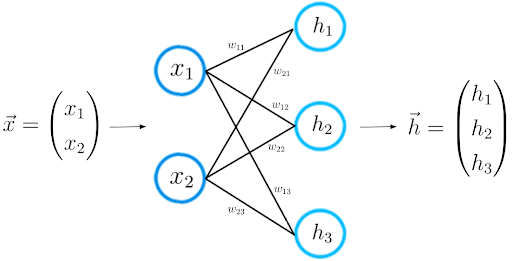


Fig. 7: Structure of forward propagation

This process is also known as step forward propagation. We calculate the dot product between the input vector ***x*** and the weight matrix ***W***, which connects the two neuron layers. The outcome vector, that we refer to as ***Z***, is produced by this dot product.

Equation 1: Forward Propagation

**2.3.6 Activation Functions**

An essential component of a neural network's design is its activation functions. How effectively the network model comprehends the training dataset will depend on the activation function that is selected for the hidden layer (Brownlee, 2021). The kind of predictions the model can end up making will depend on the activation function that is selected for the output layer. Therefore, the activation function for each deep learning project must be carefully chosen. In a neural network, an activation function describes how a node or nodes in a layer of the network shape the weighted sum of the input into an output (Brownlee, 2021). It is sometimes referred to as the transfer function. Also, it may be referred to as a "squashing function" if the output range of the activation function is constrained.

Numerous activation functions have nonlinear behaviour, which is referred to as "nonlinearity" in the layer design. Although networks are built to utilise the exact activation function across all nodes in a layer, practically the activation function is used either preceding or following the internal processing of each node in the network. Typically, the same activation function is used by all hidden layers. The type of prediction needed by the model will determine what activation function is used in the output layer, which is typically different from the hidden layers. A given input value can be used to calculate the first-order derivative for activation functions that are typically differentiable. Given that neural networks are frequently trained using the backpropagation of error algorithm, which needs the derivative of the prediction error to update the model's weights, this is necessary. In deep learning, we use three activation functions: Logistic (Sigmoid), Hyperbolic Tangent (Tanh), and Rectified Linear Activation (ReLU).

* ***Logistic (Sigmoid) Activation Function***: It is a special form of logistic function which is denoted by . One application of a sigmoid function is to transform a real value into one that can be understood as a probability because all sigmoid functions have the property that they map the entire number line into a small range, such as between 0 and 1, or -1 and 1 (Wood, 2019). The equation of the function is given by:

Equation 2: Sigmoid Function

* ***Hyperbolic Tangent (Tanh) Activation Function***: This activation function is very identical to the sigmoid function and they even share the same S-shape. In this function, any real value may be used as an input, and it returns values between -1 and 1. When the input value is more positive, it is closer to 1, however, when the value is more negative, it is closer to -1. The function is calculated as:

Equation 3: Tanh Function

* ***Rectified Linear Activation (ReLU) Activation Function***: It is possibly the most frequently used activation function for hidden layers. It is well known because it is easy to use and it is also successful at tackling the drawbacks of the other activation functions. Although it has minor drawbacks like suffering from saturated units, it is less prone to vanishing gradients that impedes the training of models. The function is determined by:

Equation 4: ReLU Function

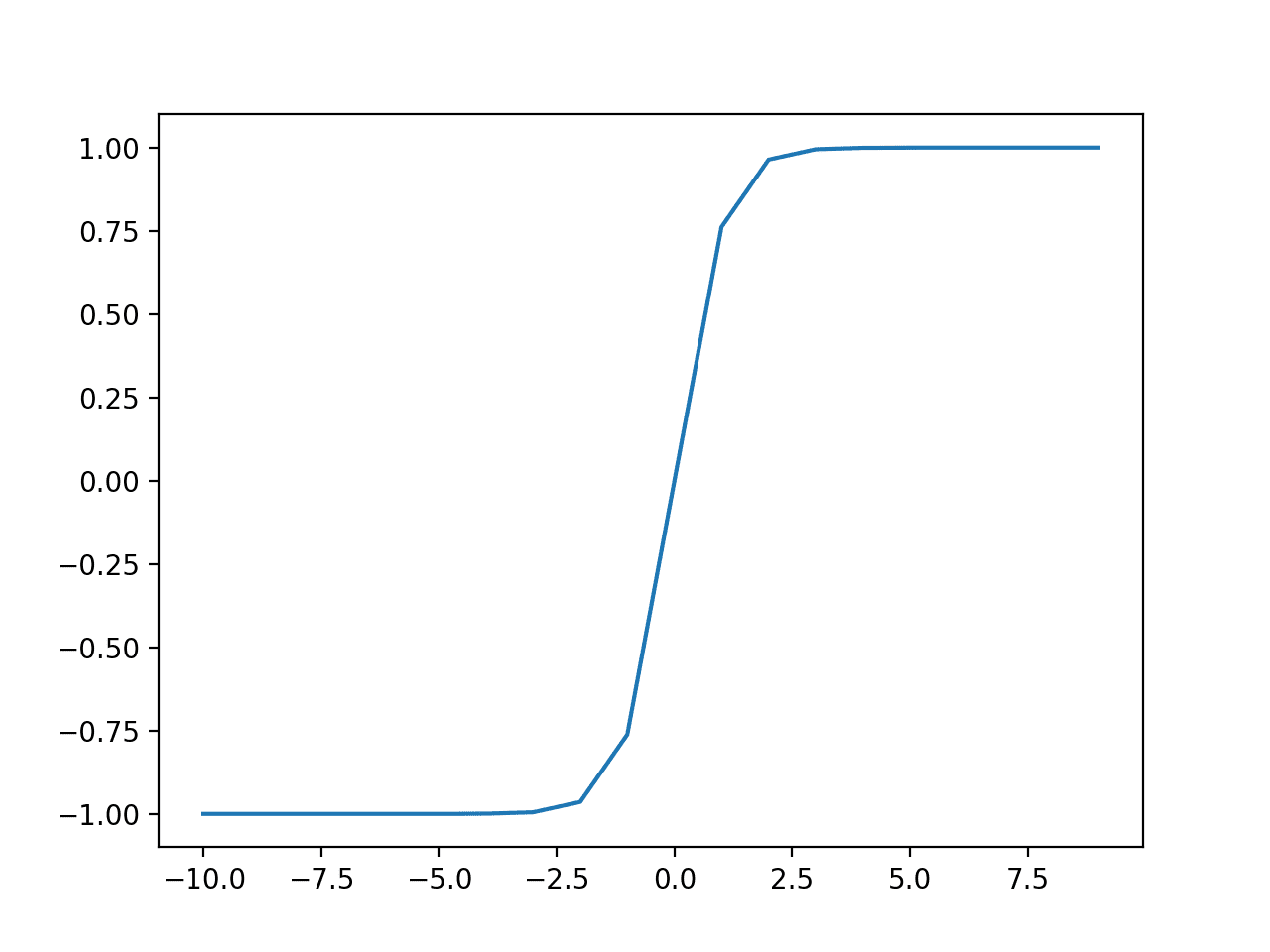
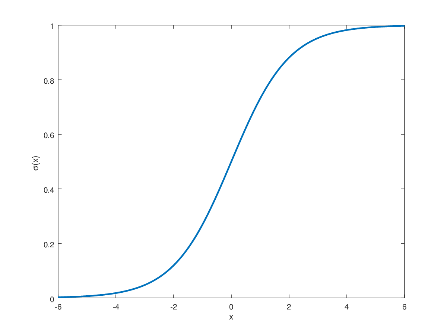


Fig. 8: Sigmoid Function Fig. 9: Tanh Function Fig. 10: ReLU Function

**2.3.7 Data Augmentation**

The standard, volume, and contextual significance of training data all play a significant role in how accurate deep learning models are. But one of the biggest problems with developing these models is a lack of data. Gathering this kind of data can be costly and labour-intensive in production use cases. Companies use data augmentation, a cheaper and more efficient technique, to build high-precision models more quickly and reduce reliance on gathering and preparing of training examples. Data augmentation is the process of creating new data points from original data in order to artificially increase the amount of data. In order to amplify the dataset, this may involve making small adjustments to the data or using machine learning algorithms to produce new data points in the subspace of the original data.

Practically every state-of-the-art deep learning application makes extensive use of data augmentation techniques. By creating fresh and varied examples for training datasets, augmented data enhances the efficiency and outcomes of deep learning models.

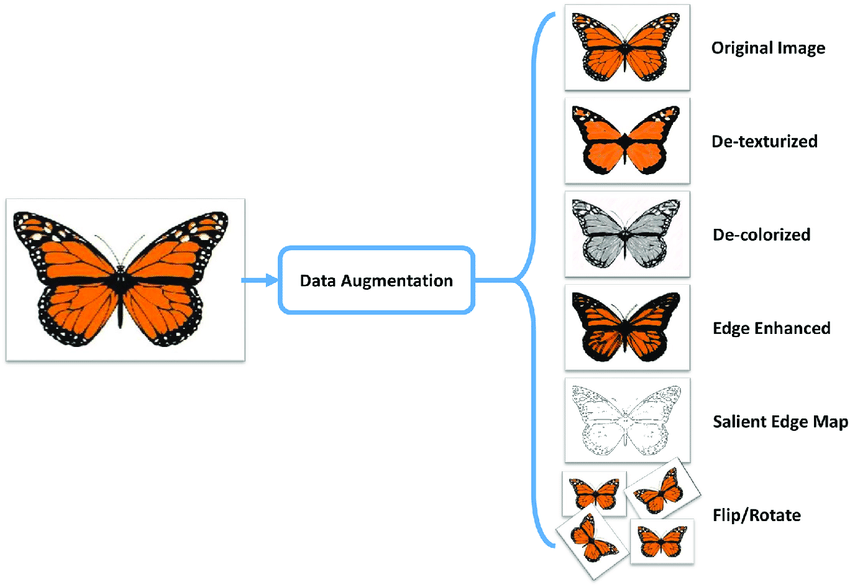


Fig. 11: Data Augmentation Image

**2.3.8 Regularization**

Building a model that performs fairly on training data and test dataset has proven to be a universal challenge in machine learning. Regularization, a term used to describe machine learning techniques, is the process of reducing test error, typically at the expense of increasing training error. Any modification or alteration to the learning algorithm commonly referred to as generalisation error that helps lower its error over a test dataset and not on the training dataset is known as regularisation (Tewari, 2021).

The majority of regularisation techniques in the case of deep learning models centre on regularising estimators. The figure of bias vs variance tradeoff below provides some additional insight into the subtleties of this subject and demarcation.

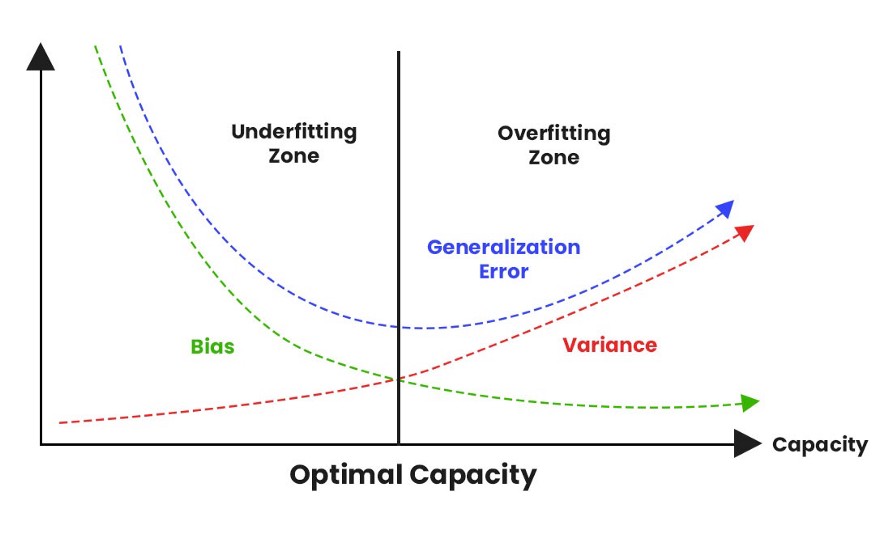


Fig. 12: Tradeoff of Bias vs Variance

When an estimator is regularised, it trades more bias for less variance. Effective regularisation is one that strikes the best balance in both bias and variance, with the end result being a notable decrease in variance at the least possible cost to bias. This would imply low variance without significantly raising the bias value, to put it another way. L1, L2, and dropout are three very well-liked and effective regularisation techniques.

The L2 regularisation, also referred to as Ride Regression or weight decay, is the most widely used type of regularisation technique (Oppermann, 2020). Both the mathematical foundation for this regularisation and the mathematical justification for why this approach reduces overfitting are lengthy and intricate. The loss function of the neural network is applied by a regularisation term Ω, during the L2 regularisation.

Equation 5: Regularisation Term

The term Ω is the L2 norm of the weight matrices, which is the cumulative total across all squared weight values of the weight matrix. The regular loss function selected is combined with Ω which is weighted by the scalar alpha divided by 2. This results in a new loss function expression:

Equation 6: Regularisation loss during L2 regularisation

We also add another hyperparameter to the neural network called alpha, also known as the regularisation rate. Simply put, alpha controls how much our model is regularised. We can determine the gradient of the new loss function and incorporate that gradient into the weights update rule:

Equation 7: Gradient descent during L2 Regularization

The expression that results from some reformulations of the update rule closely resembles the weights’ update rule during regular gradient descent.

Equation 8: Gradient descent during L2 Regularisation - Final

The addition of the regularisation term results in an extra subtraction from the existing weights, which is the only difference. In other terms, we are decreasing our weights every time an update is made, regardless of the gradient of the loss function. In L1 regularisation, which is also referred to as Lasso regression, we also use the regularisation term Ω. The weight parameters’ absolute values in the weight matrix make up this term.

Equation 9: L1 regularisation term

Just like L2 regularisation, we take the loss function and add it to the regularization term multiplied by alpha.

Equation 10: Loss function during L1 Regularisation

Taking the derivative of the loss function, we get the expression in equation 11 which is the result of adding the old loss function’s gradient and alpha multiplied by the sine of the weight.

Equation 11: Loss function’s gradient during L1 Regularisation

When L2 regularisation is used, the weight values are encouraged to approach zero, whereas when L1 regularisation is used, the weight values are encouraged to be zero. Small weights lessen the influence of the hidden neurons. In that case, the neural network's overall complexness is decreased, and those hidden neurons are no longer critical. Additionally, less complicated models usually avoid modelling data noise, preventing overfitting. When selecting the regularisation term alpha, one must be careful. The objective is to achieve the ideal balance between accuracy and the model's low complexity. Although your model will be straightforward if your alpha value is too high, you face the possibility of underfitting. To make accurate predictions, your model won't have learned enough from the training set of data. Your model will be more complicated and you face the possibility of overfitting if the value of your alpha is too low. Your model won't be able to generalise to new data because it will have learned too much about the specifics of the training set of data.

The dropout regularisation is a well-known and potent regularisation method aside L2 and L1 regularisation. The process for regularising dropouts is very straightforward. In summary, dropout refers to the probability of a neuron of a neural network getting turned off during training.

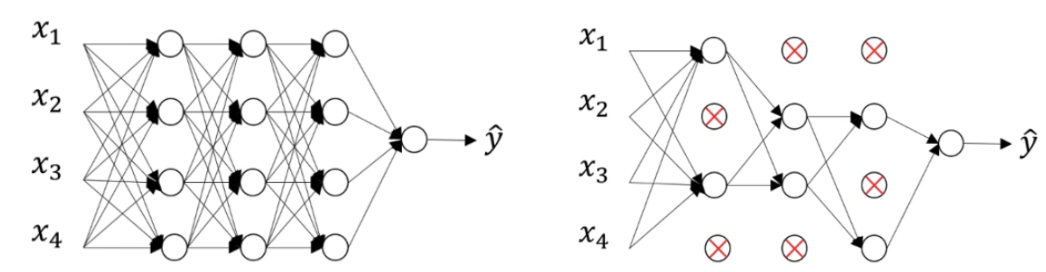


Fig. 13: Neural network without dropout Fig. 14: Neural network with dropout

If a random neuron were to be turned off during training utilising dropout with a probability of 0.5, the feedforward neural network in figure 12 which has no dropout would become the neural network in figure 13. In this instance, you can see that roughly 50% of the neurons are not functioning and are not included in the neural network. Additionally, the neural network is less complex, as you can see. Less complexity produced by a less complex neural network can lessen overfitting. At each step of forward propagation and update of the weight, neurons with a probability P of activation are deactivated.

**2.3.9 Optimization**

Optimization is a process of altering a neural network's properties, such as its weights and learning rate, in order to minimise losses (Doshi, 2020). The optimizers one employs determine how one should modify the weights or learning rates of a neural network to minimise losses. The goal of optimization algorithms is to minimise losses and deliver the most precise results.

The simplest but most popular optimization algorithm is gradient descent. Algorithms for classification and linear regression make extensive use of it. The gradient descent algorithm is also used for backpropagation in neural networks. Gradient descent, which is a first order optimization method is dependent on a loss function’s first derivative.

Equation 12: Gradient descent

Gradient descent is simple to calculate, use, and comprehend, but it can get stuck at the local minima, which is never a good thing. Additionally, after calculating gradient on the entire dataset, weights are modified. Therefore, it could a while for this to converge to the minima if the dataset is too large. Last but not least, computing gradient across the entire dataset necessitates a lot of memory.

Gradient Descent has a variation that aims to address these issues which is referred to as Stochastic Gradient Descent (SGD). It tries to perform more frequent parameter updates for the model. The model parameters in this are changed after the loss on each training iteration is computed (Srinivasan, 2019). For instance, SGD will modify the model parameters 3000 times in one loop of the dataset rather than once as in Gradient Descent if the dataset has 3000 rows.

Equation 13: Stochastic gradient descent

Because the model parameters are updated frequently, they have high variance and they fluctuate in loss functions at various intensities. SGD and conventional gradient descent are both improved by mini-batch gradient descent (Brownlee, 2017). After each batch, the model's parameters are updated. As a result, the dataset is split up into different batches, and the parameters are adjusted between each batch.

Equation 14: Mini-Batch gradient descent

There are various categories of optimizers, and each has its difficulties. To summarise the significant differences from the ones already mentioned, momentum accelerates convergence while lowering the high variance of SGD (Lee, Gasswint and Henning, 2021). Fixed learning rate is a problem that is resolved by AdaGrad (Khandelwal, 2019). The Adam optimizer corrects high variance and vanishing learning rates while maintaining rapid convergence times (Kingma and Ba, 2014). Adam is the ideal optimizer, in summary. Adam is the optimizer to use if one intends to train the neural network more quickly and effectively. Use optimizers with a dynamic learning rate for sparse data.

**2.3.10 Loss Functions**

The objective function, as used in an optimization algorithm, is the function that assesses a potential solution. In other words, we might be looking for a feasible solution with the maximum and minimum score in order to maximise or minimise the objective function. With neural networks, we typically aim to minimise the error. As a result, the objective function is frequently called a loss function or a cost function, and the amount determined by the loss function is just called loss (Goodfellow, Yoshua Bengio and Courville, 2016).

The loss function has a crucial task in that it must faithfully reduce every component of the model to a single number, with adjustments in that number serving as a signal of a more accurate model (Reed and J, 1999). The cross-entropy loss and the mean squared error (MSE) loss are the only two loss functions that you really need to be familiar with in order to solve virtually any problem you come across in deep learning.

* Mean Squared Error (MSE) Loss:

Equation 15: Mean squared error loss function

* Cross-Entropy Loss:

Equation 16: Cross-entropy loss function

**2.4 Computer Vision**

**2.4.1 Overview**

The study of computer vision focuses on developing systems that can process, examine, and comprehend visual data, such as images or videos, in a manner similar to that of humans (Babich, 2020). The idea behind computer vision is to programme computers to analyse and comprehend images down to the pixel level. Technically, machines try to retrieve, manipulate, and interpret visual data using specialised software algorithms. Computer vision quite often resembles how the human brain functions (DiCarlo, Zoccolan and Rust, 2012). According to a widely accepted theory, our brains use patterns to understand individual objects. Systems for computer vision are developed using this idea. Computer vision systems can be used for tasks such as object detection (Remanan, 2020), image classification (Sanghvi, 2020), and object localisation (Karagiannakos, 2019).

Today's computer vision algorithms are centred around pattern recognition. We use a vast portion of visual data to train computers. These machines analyse images, label the objects they contain, and look for patterns. For instance, if we send a computer a million pictures of flowers, it will analyse them, find patterns common to all flowers, and then build a model "flower" as a result of its analysis. As a result, each moment we send them images, the system will be able to recognise whether a specific image is a flower.

**2.4.2 Convolutional Neural Networks**

Convolutional neural networks (CNNs) are deep learning neural networks for processing structured data arrays, like images (Wood, 2019). CNNs are frequently used in computer vision and it has advanced to a state-of-art technique in many visual applications. Convolutional neural networks are very effective for computer vision tasks because they recognise details in images pixel by pixel such as the lines, circles, face, etc. Many convolutional layers are stacked on top of one another in convolutional neural networks, and each layer is capable of recognising more complex shapes. A convolutional neural network uses convolutional layers to process input images and recognise highly complex features, mimicking the structure of the human eye.

**2.4.3 Architecture of CNNs**

A convolutional neural network's architecture is a multi-layered feed-forward neural network created by sequentially stacking numerous hidden layers on top of one another. Convolutional neural networks can learn hierarchical features because of this sequential design. Convolutional layers are usually preceded by activation layers, several of which are then followed by pooling layers, as the hidden layers. LeNet-5, a pioneering convolutional neural network developed by Yann LeCun, was published in 1998 which serves as a straightforward convolutional neural network that facilitates comprehension of the fundamental design principles (Lecun et al., 1998). There are three layers in CNNs which are convolution layer, pooling layer and the fully connected layer.



Fig. 13: CNN architecture

* Convolution layer: The foundational component of the CNN is the convolution layer. It carries the majority of the computational load on the network. This layer creates a dot product between two matrices, one of which is the kernel which is a set of learnable parameters and the other of which is the constrained area of the receptive field. Compared to an image, the kernel is smaller in space but deeper. This means that the height and width of the kernel will be sparsely small if the image consists of three RGB channels, but the depth will go up to all three channels.

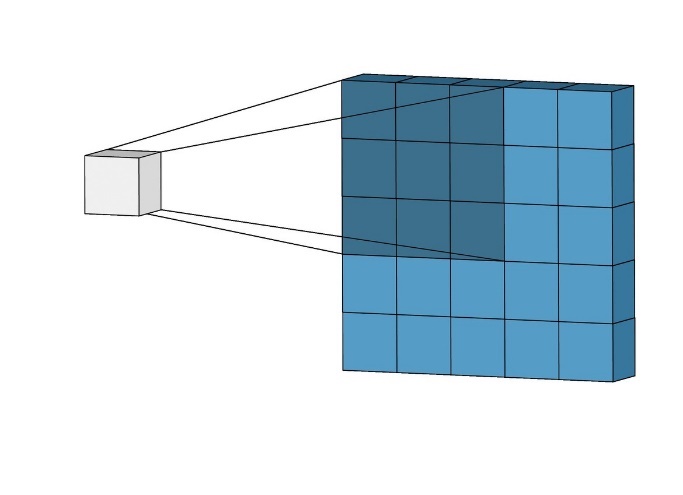


Fig. 14: Convolution illustration

The kernel moves across the image's height and width during the forward pass, creating an image illustration of that receptive region. As a result, a 2-dimensional illustration of the image called an activation map is created, revealing the kernel's response at each location in the image. A stride is a name for the kernel's slidable size. The output volume can be calculated using the formula below if we have an input of size W x W x D, and Dout kernels with a spatial size of F, stride S, and amount of padding P.

Equation 17: Convolution layer formula

* Pooling Layer: By calculating an aggregate statistic from the nearby outputs, the pooling layer substitutes for the network's output at specific locations. This aids in shrinking the representation's spatial size, which lowers the number of weights and computation needed. Each piece of the representation is subjected to the pooling operation separately. A weighted mean utilising the length from the central pixel is one of the pooling functions, along with the average of the rectangular neighbourhood and the L2 norm of the rectangular neighbourhood. The most widely used process is the max pooling, which is the highest output from the neighbourhood.

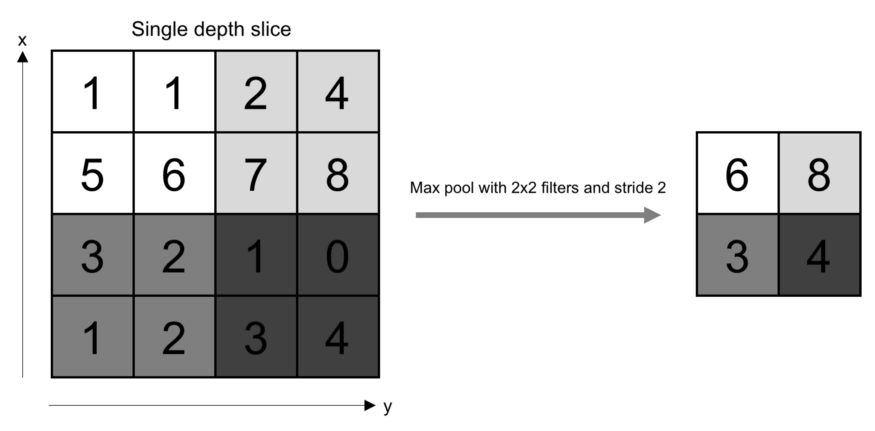


Fig. 15: The operation of pooling

The following formula can be used to calculate the size of the output volume if we've got an activation map with the size of W x W x D, a pooling kernel of spatial size F, and stride S:

Equation 18: Formula of padding layer

Wout x Wout x D will be the output volume size from the formula. With pooling, an object would always be recognisable regardless of where it is on the frame because there is some translation invariance.

* Fully Connected Layer: As in a typical fully convolutional neural network, neurons in this layer are fully connected to all neurons in the layer above and below. Because of this, it can be calculated using a matrix multiplication accompanied by a bias effect, as per usual. The depiction in between the input and the output is mapped with the aid of the fully connected layer.

**2.4.4 Applications of Computer Vision in the health sector**

Recent improvements in image classification and object detection, two subclasses of computer vision, have greatly benefited medical imaging. A number of studies have shown encouraging outcomes in challenging medical diagnostics projects which includes pathology, radiology, or dermatology. Physicians benefit from computer vision systems by getting second opinions and highlighting questionable areas in images. Although image recognition is unquestionably the most common use of computer vision in healthcare, it is not the only application. Recently, the growing demands for lab tests has stretched laboratories to their breaking point. Results are delayed by several days for the public. The machines used in clinical laboratories for automated analysis draw users in with their obvious advantages, such as faster processes and a decrease in human error. According to a report from the business research company (2021), the estimate market for laboratory automation systems worldwide will reach GBP5.2 billion by 2025.

Health professionals have some reservations about the idea of dividing duties among the doctor and AI. Many of them think that patients must have their diagnoses confirmed by a doctor in order to be certain that everything is accurate. The obligations between the physician or medical facility and the developer of AI software are another area of concern. Long-term operation and maintenance of the latter is anticipated to be failure-free. Finally, when creating computer vision-based solutions for the medical sector, doctors' involvement is required. Computer vision systems applications in health care include cancer detection, tumour detection, medical imaging and training, machine assisted diagnosis, etc.

**2.5 Object Detection**

Object detection is a computer vision method used to find instances of objects in pictures or videos. To generate useful results, object detection algorithms frequently use machine learning or deep learning. Humans can quickly identify and pinpoint objects of interest when viewing images or videos. Using a computer, object detection aims to simulate this intelligence. There are numerous methods available for performing object detection. CNN-based methods that are widely used in deep learning, like R-CNN, learn to detect objects in images. To begin using deep learning for object detection, one can choose between two main strategies:

* **Custom object detector**: One must create the architecture of the network to learn features for the objects if you want to train a custom object detector from scratch. To train the CNN, you also have to assemble a sizable set of labelled data. A custom object detector can produce amazing results. Nevertheless, you must manually configure the CNN's layers and weights, which takes time, effort and training data.
* **Pretrained object detector:** Transfer learning is a technique used by many deep learning object detection workflows that allows users to begin with a pretrained network and afterwards fine-tune it to their preference. Since object detectors have been trained on thousands or even millions of images already, this method can produce results more quickly.

You must choose between using a two-stage network or a single-stage network for your object detector, whether you build one from scratch or use one that has already been pretrained. In two-stage networks, a region proposal, or a subset of the image that may contain an object, is identified in the first stage. Examples of two-stage network include RCNN and its family. The objects contained in the region proposals are classified in the second stage. Although two-stage networks are generally slower than single-stage networks, they can produce results for object detection that are very accurate. In single-stage networks, the CNN uses anchor boxes to generate predictions for regions all over the image. The predictions are then decoded to produce the expected bounding box for the object. Despite being much quicker than two-stage networks, single-stage networks may not reach the level of accuracy of two-stage networks, particularly in scenes with small objects.

**2.6 Faster R-CNN Model**

**2.6.1 Overview**

Faster R-CNN was developed in 2015 by Ren et al and it was originally introduced as the third version of the R-CNN family. It makes use of region of interest (RoI) layers and the region proposal network (RPN). Utilizing the region proposal network, it learns the region of interests and eliminates the need for selective searching. The RPN can also be seen as an attentional mechanism. The areas of the network that generate region proposals and bounding box predictions all share convolutional feature maps, and the entire network can be trained from beginning to end. The addition of anchor boxes, a collection of crop regions with various sizes and aspect ratios which are assigned to each region of interest, was one factor in making this work effectively. One could envision a network that computes various scaling and squashing operations or an architecture in which images of various sizes are sent into the network. Anchor boxes don't function like this. They enable effective feature sharing earlier in the network because they are applied at the end of the RPN. This is an ensemble for each RoI because each anchor box predicts the bounding box values for the RoI and the regressors do not distribute weights among anchor boxes.

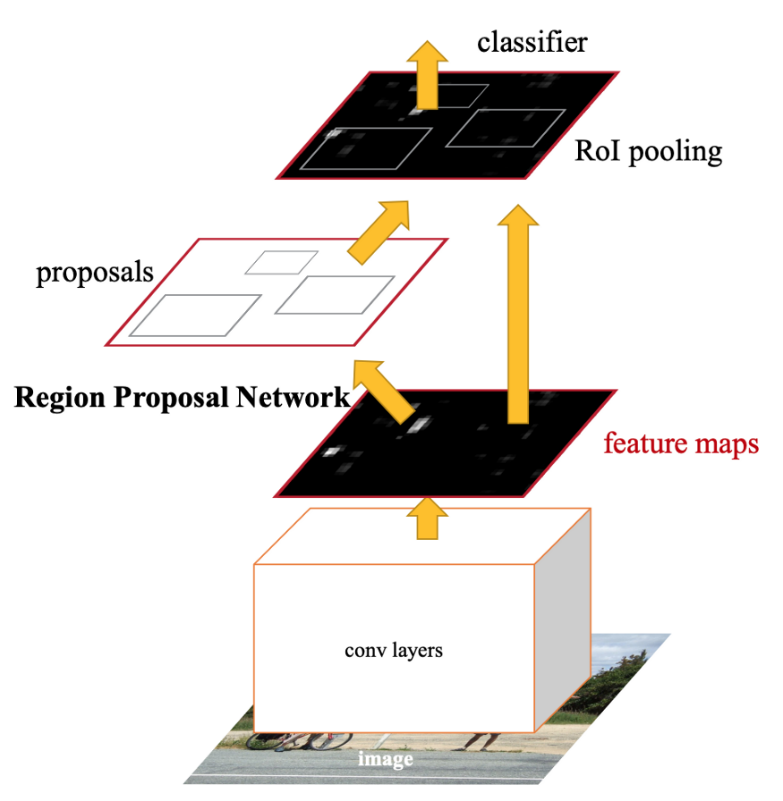


Fig. 16: Faster R-CNN architecture

**2.6.2 Region Proposal Network (RPN)**

RPN is a fully convolutional neural network that predicts both the object bounds and objectness scores at every area in an image (Ren et al., 2015). To produce good region proposals, the RPN undergoes end-to-end training. Sharing convolutional features allows RPN to be combined into a single network. The RPN component instructs the unified network where to look by using attention mechanisms. RPNs were created to effectively predict region proposals with a variety of scales and aspect ratios. With regard to scales and aspect ratios, RPNs make use of anchor boxes as references. This method avoids having to list images or filters with various scales or aspect ratios by using a pyramid-like structure of regression references.

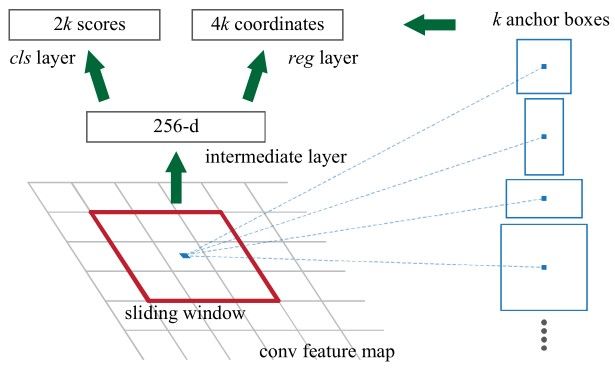


Fig. 17: Region Proposal Network

**2.6.3 Region of Interest (RoI) Pooling**

Fast R-CNN was the publication that first proposed region of interest pooling, a neural-net layer that significantly speeds up both training and testing (Girshick, 2015). The detection accuracy is also kept at a high level. The RoI layer accepts these two inputs:

* A fixed-size feature map produced by a convolutional network with multiple convolutions and maximum pooling layers.
* A matrix containing region of interest

**2.6.4 Anchor Boxes**

These are bounding boxes that are positioned throughout the entire feature map and act as the areas where the RPN will look for objects. As shown in the figure below, a regular grid is first created from the feature map's divisions before those anchor boxes are placed. We then insert a handful of anchor boxes into every one of these grid cells, centring them on the cell's centre point. We can decide what kind of size and aspect ratio, and the number of anchor boxes we want. Based on the objects that we hope to detect, these parameters must be adjusted.

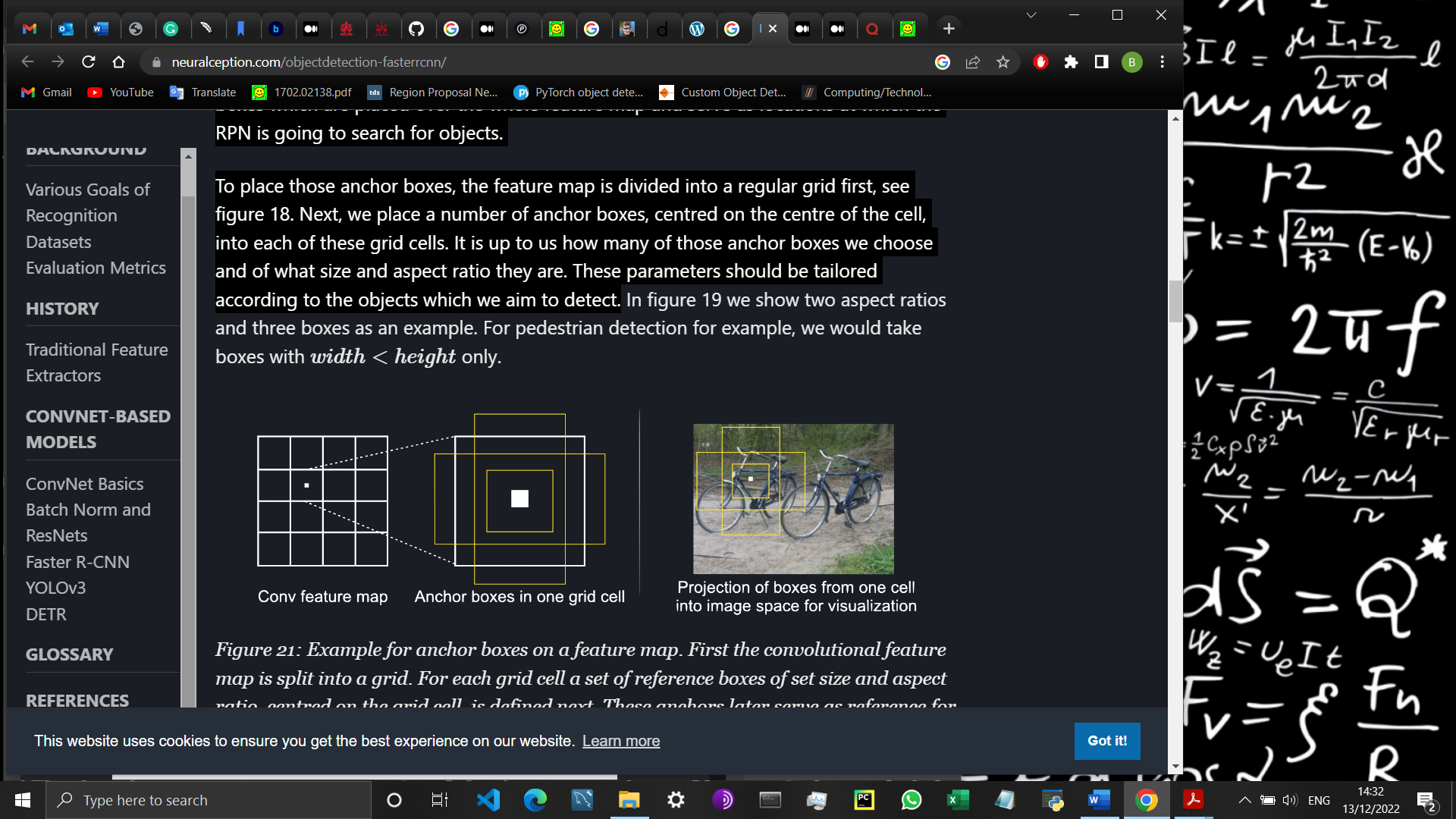


Fig. 18: Anchor box on a feature map

One essential component of Faster R-CNN's performance is the use of anchor boxes. The R-CNN family's earlier iterations applied region proposals established on segmentation’s selective search algorithm (Uijlings et al., 2013). While getting region proposals in this manner is logical, it is computationally expensive. For a 227x227-pixel image, selective search generates about 2000 proposals in about two seconds. Since the convolutional feature map is only computed once, using anchor boxes on it enables the allocation of computation between the modules.

**2.6.5 Filtering Proposals**

The number of potential regions that an object could be in is still overestimated by the anchor box method. Numerous boxes try to identify the same object result from the boxes' intentional overlap, as shown on figure 19's left side. Therefore, a large number of these overlapping boxes must be filtered out and eliminated.

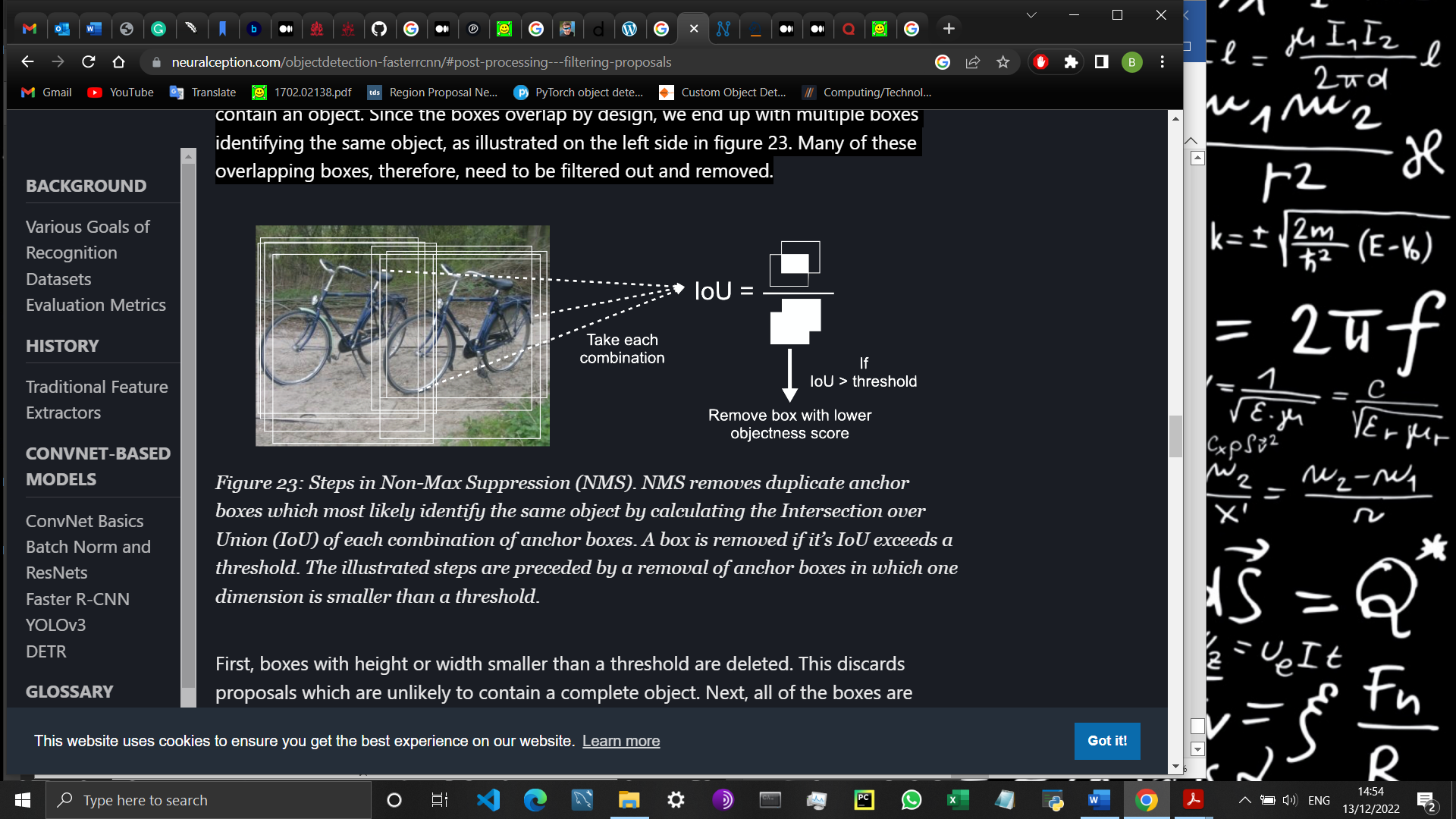


Fig. 19: Non-Max Suppression (NMS)

To explain it in details, boxes that are smaller than a cutoff in height or width are first deleted. This eliminates proposals that are unlikely to include an entire object. The boxes are then sorted by objectness score, highest to lowest, and NMS is used. NMS first calculates the Intersection over Union (IoU) of the boxes to identify those with high overlap. The IoU takes in two boxes, determining their overlap, and then divides that amount by the combined area of both boxes. Two anchor boxes are much more inclined to represent distinct objects if their IoU values are low. On the other hand, two anchor boxes with high overlap and low IoU are more likely to depict the same object. The anchor box with the lower objectness score is eliminated if two boxes have a high IoU. However, partially occluded objects, like those in figure 19, can still cause issues. In such cases, a Soft-NMS can be used in their place. In this case, a penalty to the objectness score is generated rather than removing the anchor box.

**3. Methods**

**3.1 Dataset**

The dataset for the project was gotten from the cancer imaging archive (Clark et al., 2013). It is an open-source informational tool to support initiatives in research, development, and education that make use of cutting-edge cancer imaging technology. It is also backed by the national cancer institute in the United States of America. The dataset contains different types of liver cancer ultrasound images (Eisenbrey, Lyshchik and Wessner, 2021). The images are 1859 with majority of the cancer type being HCC. All the images have a copy of it with the area of the lesion identified. There is one flaw in this dataset and that is, it is not labelled. The labelling was done by me using Napari which is a quick, engaging, multi-dimensional software for images (Napari Team, 2020). Large, multi-dimensional images can be browsed, annotated, and analysed using this programme. The labelling did not require a professional in the field of radiology because the position of the lesions was already pinpointed in copies of it as seen in the figures below. All I had to do was draw bounding boxes around the original images which took some time. I was able annotate 1035 out of the 1859 images.

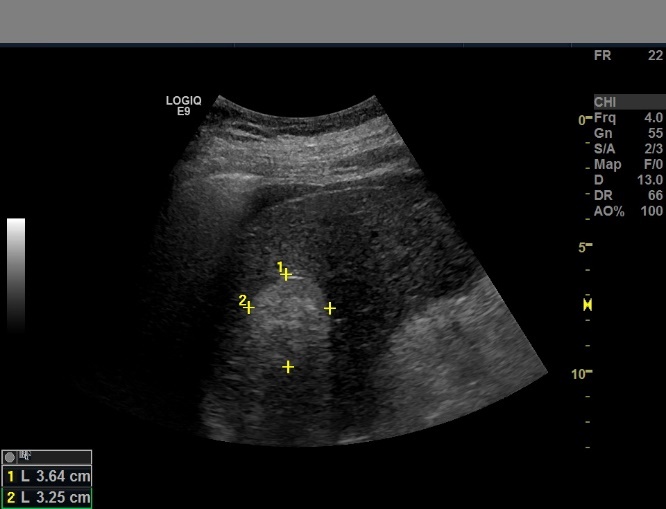
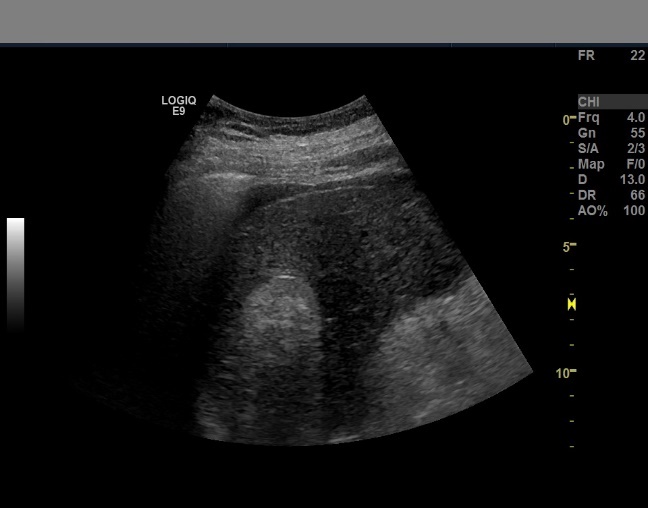


Fig. 20: Raw ultrasound image Fig. 21: Copy of image with lesion

Aside labelling the data, a script was used to rename all of in an appropriate manner to make training and testing easier. The script can be found in the code repository as “*rename\_files\_script.ipynb”*.

**3.2 Dependencies**

Several tools and dependencies were used to implement this project. To begin with the programming language, python was used. It is the best programming language for projects involving machine learning and artificial intelligence. Furthermore, it is simple, consistent, and has excellent libraries and frameworks. I used Pytorch Lightning as the framework for the project which is built on PyTorch. It is a Torch-based deep learning framework that is relatively new. It was created by Facebook's AI research team and released as open source (PyTorch, 2019). It is primarily utilized in applications involving computer vision and natural language processing. I selected this framework because it is known for its simplicity, usability, effective memory usage, and dynamic computational graphs. Additionally, it feels pythonic as programmers sometimes refer to it, which speeds up computation and makes coding easier to manage. I also used Napari which is a visualisation software as mentioned in section 3.1. I used it to annotate the dataset and also for visualisations. I used Neptune to track and log all my experiments and models (Neptune, 2022). It is a metadata application that provides the tracking and registering of machine learning models. It is mostly used by engineers and researchers. I used the anaconda distribution for these dependencies and tools and all training was done in the robin milner lab.

**3.3 Implementation**

A customized dataset class was built to feed the Faster RCNN model. The class has arguments like ***transform*** to apply data transformations, ***use\_cache*** for multiprocessing, ***mapping*** to encode string labels and ***convert\_to\_format*** to convert bounding boxes to the right format. For data augmentation techniques, the module ***transformations.py*** contains all the classes and functions. The prominent ones include ***ComposeDouble*** which allows you to stack various transformations. There’s also the ***AlbumentationWrapper*** class which contains the albumentation module to augment the images.

For the model implementation, I used the ResNet family as the backbone. Further details can be found in the tables below.

|  |  |  |
| --- | --- | --- |
| Backbone | Trained Parameters | Total Estimated Model Parameters |
| ResNet-18 | 40.3 MB | 161.201 MB |
| ResNet-34 | 50.4 MB | 201.633 MB |
| ResNet-50 | 165 MB | 660.464 MB |
| ResNet-101 | 184 MB | 736.433 MB |
| ResNet-152 | 199 MB | 799.007 MB |

Table 1: ResNet family – Without FPN

|  |  |  |
| --- | --- | --- |
| Backbone | Trained Parameters | Total Estimated Model Parameters |
| ResNet-18 | 28.3 MB | 113.11 MB |
| ResNet-34 | 38.4 MB | 153.519 MB |
| ResNet-50 | 41.3 MB | 165.217 MB |
| ResNet-101 | 60.2 MB | 240.977 MB |
| ResNet-152 | 75.8 MB | 303.367 MB |

Table 2: ResNet family – With FPN

All the backbones are pretrained and as seen from the table, not all parameters were used for the training. To train a model with one of these backbones, I used the ***get\_fasterRCNN\_resnet()*** module with various parameters and hyperparameters: batch size, anchor size, learning rate, min\_size, precision, max\_size, aspect ratios, etc. The min\_size and max\_size is the minimum and maximum size the image has to be rescaled before being fed to the backbone. Batch size is the sample size that go through the network. The anchor size and aspect ratios are the width and length of the anchor boxes and the learning rate determines the modification of the model with regards to the estimated error and weight update. The full scale of the parameters and hyperparameters can be seen in the table below.

|  |  |
| --- | --- |
| Parameter / Hyperparameter | Value |
| Batch Size | 1 or 2 depending on backbone |
| Learning Rate | 0.001 |
| Precision | 2 |
| Epochs | 150 |
| FPN | Boolean: Activates FPN if true |
| Anchor Size | ((32, 64, 128, 256, 512)) – Without FPN  ((32,), (64,), (128,), (256,)) – With FPN |
| Aspect Ratio | ((0.5, 1.0, 2.0),) |
| Min Size | 1024 |
| Max Size | 1024 |
| IoU Threshold | 0.5 |

Table 3: Parameters and Hyperparameters for the Faster R-CNN model

The code repository has instructions on how to run the experiments. The code and login details for Neptune is also available in the appendix. The login details for Neptune can be used to see all the details of the experiments including the charts. Further details on how to use and navigate Neptune is also available in the appendix.

**4. Results**

A total of 15 experiments were run for the Faster-RCNN model with various ResNet backbones and hyperparameters. The initial 5 experiments were done with just 120 images. The results were pretty bad as seen in the figure below. The experiments were done on the ResNet family without FPN and due to the bad performance, I increased the dataset and trained again for better performance. From the figure below, we can see that the mAP was low for all 5 experiments. However, there was a connection in the backbones with ResNet-18 being the lowest performing model and ResNet-152 being the top. From the way ResNet architecture is built, it is supposed to perform like that with an exception in ResNet-50. This case was different for other experiments.

Fig. 22: First set of experiments with 120 images

Further experiments were done with an increased dataset. The images were increased from 120 to 1035 and there was significant improvement in the accuracy. The whole ResNet family was used as backbone to train the Faster RCNN model again with the new dataset. An additional parameter which is FPN was added and training was done for that one too. This can be seen in the figures below.

Fig. 23: Improved data experiments Fig. 24: Final experiments with improved data

All the experiment for Faster R-CNN with their distinct parameters and hyperparameters can be found in the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Backbone | Batch\_Size | With FPN | IoU | LR | epochs | mAP |
| ResNet-18 | 2 | False | 0.5 | 0.001 | 150 | 0.0071 |
| ResNet-34 | 2 | False | 0.5 | 0.001 | 150 | 0.022 |
| ResNet-50 | 2 | False | 0.5 | 0.001 | 150 | 0.044 |
| ResNet-101 | 1 | False | 0.5 | 0.001 | 150 | 0.032 |
| ResNet-152 | 1 | False | 0.5 | 0.001 | 150 | 0.073 |

Table 4: First Experiment with 120 images – model comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Backbone | Batch\_Size | With FPN | IoU | LR | epochs | mAP |
| ResNet-18 | 2 | False | 0.5 | 0.001 | 150 | 0.399 |
| ResNet-34 | 2 | False | 0.5 | 0.001 | 150 | 0.484 |
| ResNet-50 | 2 | False | 0.5 | 0.001 | 150 | 0.421 |
| ResNet-101 | 1 | False | 0.5 | 0.001 | 150 | 0.052 |
| ResNet-152 | 1 | False | 0.5 | 0.001 | 150 | 0.030 |

Table 5: Second Experiment with 1035 images – model comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Backbone | Batch\_Size | With FPN | IoU | LR | epochs | mAP |
| ResNet-18 | 2 | True | 0.5 | 0.001 | 150 | 0.487 |
| ResNet-34 | 2 | True | 0.5 | 0.001 | 150 | 0.4745 |
| ResNet-50 | 2 | True | 0.5 | 0.001 | 150 | 0.446 |
| ResNet-101 | 1 | True | 0.5 | 0.001 | 150 | 0.439 |
| ResNet-152 | 1 | True | 0.5 | 0.001 | 150 | 0.455 |

Table 6: Third Expirement with 1035 images – model comparison

**5. Discussion**

In this project, the aim was to build customized Faster R-CNN model to detect HCC lesions as stated in the project proposal. To complete this objective, I did comprehensive study on the Faster R-CNN model and its backbones. I decided to work with the ResNet family as the backbone because of different and multiple architecture types. As seen in the tables in section 4, we can conclude that the type of backbone used in Faster R-CNN model affects its performance. Taking the first set of experiments, we saw that Faster R-CNN with ResNet backbone without FPN does not perform well on small medical imaging datasets. The highest accuracy that was achieved was 4 percent with Resnet-152 backbone. We can also say that, although it doesn’t perform well on small datasets, it achieves a better accuracy with ResNet-152 as the backbone, as compared to the rest of the ResNet family. Taking the second set of experiments, we can say that Faster R-CNN perform better with not so small dataset. A dataset with 1035 images is not really big and at the same time not small. From table 5, we can conclude that, the type of ResNet backbone affects the performance on the model with the type of parameters and hyperparameters that you use. From the table we can see that ResNet-34 as the backbone outperformed ResNet-50 and the others. In principle, ResNet-50 is far better than ResNet-34 so as a feature extractor, it should boost the model performance significantly but that was not the case. ResNet-101 and ResNet-152 as the backbones did not perform well. From the table, we can see that the only difference in the parameters and hyperparameters is the batch size. It could be that the batch size influenced the bad performance because large batch sizes frequently converge more quickly and produce better results. Taking the third set of experiments, we can say that ResNets + FPN as backbone performs better than ResNets as backbone. From table 6, we can see that the mAP range is quite similar for all the backbones and general performance is better as compared to the other experiments.

The research objective was to compare customized Faster-RCNN to the base model. The based model was trained on the pascal voc dataset with VGG-16 backbone and achieved a mAP of 59.2 percent (Ren et al., 2015). The highest mAP achieved on my dataset is 48.7 percent for ResNet-18 + FPN backbone which is not bad.

**6. Evaluation**