

A Deep Learning-Based Software Solution for CT Scan Classification



Janusz Snieg
SNI20774609

25085325@students.lincoln.ac.uk

School of Computer Science
College of Science
University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of BSc(Hons) Computer Science

Supervisor: Dr. James Wingate

Word count: 11563 | Word count total: 12565

May 2023

Acknowledgements

This thesis is in memoriam and dedication to my grandfather who suddenly passed away to sudden haemorrhage and whom I never got a chance to meet. I miss you.

My belated grandparents that taught me the values of living and how to appreciate your everyday life.

I want to personally thank my tutor, supervisor and everyone that helped me through these tough three years of university – without you I would not succeed.

My close friends back home and friends I made at the university, thank you for all memories made, eternal friendships and help that I received.

My beloved girlfriend that during those two years made sure I didn't lose sanity.

Lastly, thank you to my parents and my dogs Lucy and Elza for directing me in the right decision and luckily, I chose to go to the university to not only complete my hunger but also make everyone proud.

Abstract

This paper examines the role of deep learning applied in medical setting, notably with an attempt to detect; by highlighting regions of interest of intracranial haemorrhages for the purpose of diagnosis and binary classification. Haemorrhagic strokes are a leading mortality cause affecting 13.7 million people with 5.5 million deaths annually. Data was accumulated from public and academic database source *Kaggle*, no human participants were identified nor used and the scans collected are owned by a third-party. To carry out the solution with regulated momentum a waterfall methodology was adapted for the target of completing the artefact. Results were gathered by training the deep-learning classifier on firstly, ~200 samples on 200 epoch which yield wrong class predictions. Secondly, a larger dataset was used ~7000 samples each split to 0.7 training and 0.3 testing split sets which achieved high accuracy, precision, and recall metrics at 20 epochs. Lastly, same samples were used at lower epoch ~5, to introduce some bias and increase justification argument and transparency of the model, on average the metrics deemed more realistic with value set at 0.98 recall and 0.99 everywhere else. To conclude, artefact successfully justified the aim for this paper but also unravels unanswered questions in adapting existing machine models for different problem settings. It is insistent to keep researching this field as deep learning posses' issues that need to be addressed.

Table of Contents

1.	Introduction.....	12
1.1	Motivation.....	12
1.2	Rationale	16
1.3	Referencing	17
1.	Literature Review.....	18
2.1	Radiomics	18
2.2	Image Segmentation	19
2.3	Computed Tomography	20
2.4	Machine Learning.....	21
2.5	Convolutional Neural Networks	23
2.6	Aims & Objectives	24
2.	Requirements Analysis	28
3.1	Functional Requirements	28
3.2	Non-functional Requirements.....	29
4.	Design & Methodology.....	30
4.1	Project Management	30
4.2	Risk Analysis	31
4.3	Toolsets and Machine Environments.....	33
4.3.1	Python (programming language).....	33
4.3.2	Frameworks.....	33
4.4	Data Acquisition and Annotation.....	34
4.5	Haemorrhage Detection	35
4.6	Architecture of a CNN Model.....	37
4.6.1	Loss Functions.....	42
4.6.2	Data Normalisation	43
4.6.3	Parameter Tuning	43
4.7	How is the system evaluated?	44
5.	Implementation	47
5.1	Splitting Data	47

5.2	Directory Walkthrough	48
5.3	Image Visualisation (Randomised).....	49
5.4	Data Transformation & Loader.....	50
5.4	Model Classifier.....	52
5.4.1	Training & Testing Steps	54
5.5	Training & Plotting Metrics.....	56
5.6	Prediction	57
5.7	Correlation Matrix	59
6.	Results & Discussion	60
7.	Conclusion	68
	References.....	70
	Appendix.....	76

List of Figures

Fig. 1. Machine learning categories and their definition summaries (Choy, G., et al. 2018. 4)

Fig. 2. Workflow of the Classifier in Supervised Learning Model (Salim D. 2021. 5).

Fig. 4. Stage process of applied radiomics to stroke neuroimaging. Feature extractions A and B are showing intracerebral haemorrhage and ischemic stroke (Chen, Q., et al. 2021. 11).

Fig. 5. Routine CT scan is on the left and low dosage on the right. (Smith-Bindman, R., 2010. 28)

Fig. 6. Waterfall as a software development methodology for this project.

Fig 7. An overview of the project life cycle presented in GANTT format.

Fig. 8 Training speed for 6 dependencies applying 32-bit floats (Paszke, A., et al. 2019. 42).

Fig. 9. Putamenal Hypertensive Haemorrhage (Rymer, M.M. 2011. 34).

Fig. 10. Intracerebral Haemorrhage on CT image, method for detection is ABC/2 formula (Macellari, F., et al. 2014. 35).

Fig. 11 An overview of a convolutional neural network architecture and training phases (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Fig. 12. Image portrayed by a computer is an array of numbers. Label C on the right make up the image with numbers ranging from 0-255 which correspond to individual brightness of a pixel (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Fig. 13. A visual explanation of how feature map is obtained in the element-wise product (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Fig. 14. Input dimension of 5 by 5 is identical to the one in the feature map thanks to padding (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Fig. 15. A visual representation of Rectified Linear Unit. While $x < 0$ output activation function but when $x > 0$ outputs linear (Abien, F. and Agarap 2019. 33).

Fig. 16. 1st cell representing required dependencies to be installed if missing.

Fig. 17. Cell representing version and directory checkers.

Fig. 18. (a) & (b) A walkthrough through the specified directories.

Fig. 19. Randomised image output for visualisation testing.

Fig. 20. Data transform into Tensor & Torch-vision version checker.

Fig. 21. Plotting transformed images into Tensor.

Fig. 22. A comparison between original input and transformed output of Haemorrhagic scans.

Fig. 23. Transformed images into Image Folder and Data Loaders.

Fig. 24. Retrospective labels in correspondence to directories.

Fig. 25. Classifier model inspired by TinyVGG and model's output structure.

Fig. 26. A single image prediction test.

Fig. 27. Summary of the architecture, process within convolutional layers and more.

Fig. 28. Training step with plotting output showing loss and accuracy per batch.

Fig. 29. Test step with plotting output showing loss and accuracy per batch.

Fig. 30. Training function to train the model with empty dictionaries for metrics return.

Fig. 31. Instance of the convolutional neural network, loss function, optimiser and model training.

Fig. 32. Plotting the loss and accuracy curves.

Fig. 33. Custom image transform, converts it into PIL image, resizes and transforms to Tensor.

Fig. 34. Prediction function and function's call-back.

Fig. 35. Predicted random image not located within nor train or test directories.

Fig. 36. Correlation Matrix cell before its' execution.

Fig. 37. Model trained on ~200 samples on 10 epoch.

Fig. 38. Model trained on ~200 samples on 100 epoch.

Fig. 39. Model trained on ~200 samples on 200 epoch.

Fig. 40. A wrong-class prediction of a haemorrhagic stroke outlined by darker centre build of fluid of the brain.

Fig. 41. Model trained on ~7000 samples on 20 epoch.

Fig. 42. A right-class prediction of a haemorrhagic stroke when trained on large dataset with high probability.

Fig. 43. A right-class prediction of no haemorrhagic stroke when trained on large dataset with high probability.

Fig. 44. A classification report returning Precision, Recall and F1 Score over test dataset.

Fig. 45. A confusion matrix plotted using Seaborn.

Fig. 46. A balance between overfit and underfit when trained on 5 epochs & ~7000 samples.

Fig. 47. A correlation matrix and classifications' report on 5 epochs & ~7000 samples.

Fig. 48. Correct prediction of a haemorrhage scan "h_4.jpg" with probability 0.617.

Fig. 49. Updated GANTT chart summarising entire project span.

Fig. 50. Training process with accountable epochs, losses, and accuracies on 20 epochs.

List of Tables

Table 1. Table of individual SMART objectives.

Table 2. A risk analysis table with risks that can come as project progresses.

List of Equations

Eq. 1 Accuracy of a classifier (Cho, J., et al. 2016. 38).

Eq. 2. Dot product calculated within the convolution layer.

Eq. 3 Mathematical definition of ReLU (Abien, F. and Agarap 2019. 33).

Eq. 4 Back-propagation mathematical equation (Alzubaidi, L., et al. 2021. 41).

Eq. 5 Binary classification expression of a total data model (Hicks, S.A., et al. 2022. 43).

Eq. 6 Confusion Matrix (Hicks, S.A., et al. 2022. 43).

Eq. 7 Accuracy Formula (Hicks, S.A., et al. 2022. 43).

Eq. 8 Recall Formula (Hicks, S.A., et al. 2022. 43).

Eq. 9 Precision Formula (Hicks, S.A., et al. 2022. 43).

Chapter 1

1. Introduction

1.1 Motivation

A stroke which affects a human's brain is a neurological disorder commonly branded by blockage of blood vessels or the lack of it. A clot is formed in the brain and interrupts the blood flow carrying oxygen and other nutrients to safeguard the functionality of a person's brain, by clogging up the arteries and causing vessels to break it leads to severe bleeding which results in sudden death as brain cells have no supply to oxygen – but also can lead to depression, dementia, or other neurological progresses. International Classification of Disease (ICD-11) released in 2018 a stroke consists of a class that describes stroke as a blood vessel disease. Previously, their code was proven wrong as data collected misrepresented the dangerousness of the burden begun by a stroke. As a result, government bodies did not obtain elegant funding to research this neurological harm to life. However, as it was re-classified with sufficient data, analysis and supporting improvements were made to tackle the problem with various diagnosis approaches and potential solutions (Kuriakose, D. and Xiao, Z.-C. 2020. 29). Within the report, a stroke is the second major leading cause of death worldwide affecting 13.7 million people with 5.5 million death rate per annum. 87% is the estimated percentage of ischemic infarctions (clots) that increased between 1990 and 2016 due to declined mortality but improved clinical interventions. A patient who experienced their stroke first-time make-up up the margin of strokes. Lower to middle-income countries suffer from the doubled events of strokes from periods of 1990-2016 but declined by almost half, a staggering 42% in high-income countries within the same time span. Global Burden of Disease Study (30) asserted that although the event of stroke has decreased, age, sex, or geographic location of a patient their social-economic burden of a stroke has increased over time which only means that access to healthcare in less developed countries is limited therefore an event of a stroke could be pertinent. The risk of stroke doubles as a person passes the 55 years old remark and it is worthwhile noting that era's an influx of stroke occurring between ages of 20-54 from 12.9% to 18.6% between 1990 and 2016 internationally but fortunately, attributable death rates decreased by 36.2%. One of the highest rates of strokes happening was recorded in Eastern

Europe, 181-218 per 100,000 life years. Although stroke heavily depends on someone's age, gender also is a factor. It is high in younger women due to pregnancy aspects e.g., preeclampsia, contraceptive use and hormonal therapy with the severity based on average was estimated to be 10 for women and 8.2 for men in accordance with the National Institutes of Health Stroke Scale. The chances of patient dying because of a stroke is heavily associated with women than men, although women live longer than men severity and circumstances of strokes increase with apparent decline amongst women when reaching out for support. Men, on the other hand, a frequent cause of stroke is smoking tobacco, excessive alcohol consumption and arterial disorders.

Machine learning, a branch of artificial intelligence is a computer program supported by supervised, unsupervised and reinforced algorithms where a machine performs a task in predicting outputs given inputs as the system learns over time with comprehensive datasets to decide to forecast data presented upturn over time. This thesis's particular focus is on medical imaging, therefore, understanding both medical imaging and machine learning aspects is crucial. By a medical reports congregation and comprehensive investigation models' performance, the model will continue to improve by recognising patterns and its extraction. Machine learning is used throughout different industries e.g., commercial, advertising, product/entertainment recommendation, data mining & etcetera (Ray, S. 2019. 3).

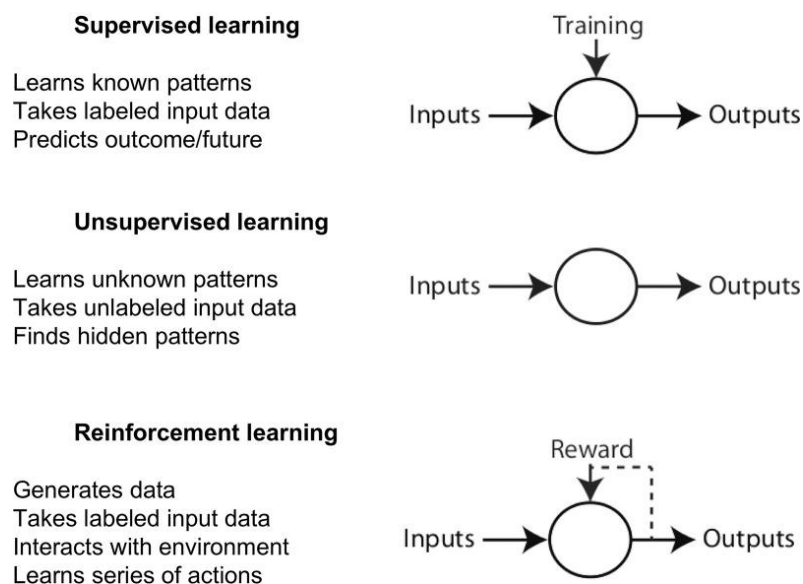


Fig. 1 Machine learning categories and their definition summaries (Choy, G., et al. 2018. 4)

Supervised learning fundamentally supports a machine model by providing data labels in training phase e.g., supervisor trains a new employee recognising issues and giving advice to iteratively develop upon. An expert(s) labels the output using appropriate ground truth, implying that the data is assumed to be true. Additionally, supervised approach uses classification and regression tactics (Choy, G., et al. 2018. 4). A classification-based method aims to foresee unknown values by knowing values on input when the output is categorised it is referred as a classification problem. Classification is supported on both structured and unstructured datasets and initialises it according to specific classes. A binary classifier is where only one output is expected if the model expects to classify an input to classes A, B or C it is a multiclassification problem. As machine trains repeatedly the new trained data point is assigned to a certain class, it does not have to be necessarily true/right first time as machine models train overtime improving their measuring metrics that'll be discussed later in the paper. Fig. 2 depicts step-by-step starting with data collection and pre-processing the data as it cleanses noise and duplicates to eradicate anomalies within the model. Brute-force is commonly used way of pre-processing machine data. A cross-validation technique is splitting the data into training, testing phases and potentially validation sets where it quantifies the performance and is measured on a validation set. Whereas regression refers to the discovery of correlation between variables and predictions of continuous values of the variables. Continuous value is captured by measuring e.g., a height of an animal. It cannot be taken but can be measured. Regression is typically divided into simple and multiple linear regression where a line is drawn to divide variables however, with multiple linear there's more than two variables and decides upon non-linear and linear data (Tougui, I. et al. 2021. 6) (Salim D. 2021. 5).

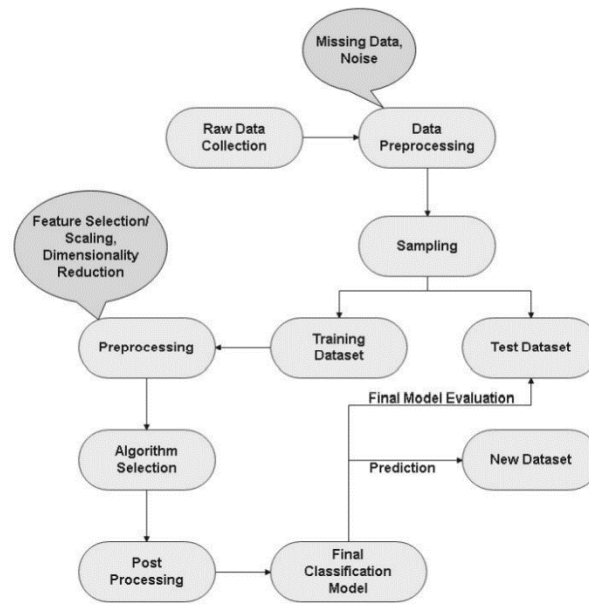


Fig. 2 Workflow of the Classifier in Supervised Learning Model (Salim D. 2021. 5).

Unsupervised learning just like any other machine learning model its sole task is to find hidden structures in the data as unsupervised approach has no data labels. (Choy, G., et al. 2018. 4.) Algorithms due to no data labels or supervision of sorts classify datapoints by identifying patterns within the data. Datasets are grouped according to parallels and distinctions. Commonly clustering is frequently in use in the industry but principal component analysis (lowers dimensionality of the data), anomaly detection, association and auto coders too exist. (Salim D. 2021. 7).

Reason for deep learning framework in the clinical setting is presented in (Miotto, R., et al. 2018. 48), where a convolutional network was spent in scanning MRI of low-field knee to automatically segment cartilage and predict a risk of osteoarthritis, although, two-dimensional it returned better results than manual three-dimensional scale features. Not to mention, obtaining performances on classification of biopsy-proven clinical scans of diverse types of skin cancer over a dataset sum to 130,000 images; this was also supported by 21 board-certified dermatologists. It is an obligation to train classifiers with available publicly models to test their capabilities and extend their usage to other problems i.e., haemorrhages. Numerous findings in the existing literature demonstrate the profound potential of deep learning in analysing healthcare data. Specifically, the utilization of multi-layer neural networks has substantially enhanced the predictive capacity for various clinical applications within the medical field. Moreover, deep architectures, with their hierarchical learning structure, possess

the ability to effectively merge disparate data sets encompassing diverse data types. This integration capability, combined with the emphasis on representation learning rather than solely focusing on classification accuracy, holds significant promise for achieving broader generalization in healthcare data analysis.

1.2 Rationale

A deep learning model although hungry in terms of data needed, extracts a larger number of deep features from hidden layers incorporated into the radiomics model. Deep learning evidently has the talent to transform how information is extracted from medical imaging, however, its' accuracy remains uncertain, although radiological investigations are carried out by radiologists themselves which impose risks for mishandling as everything is prone to error, pressure on radiology staff and time is taken to address a haemorrhage stroke before mortality rate decreases, thus deep learning hopes to address this issue. A satisfied integration of deep learning to a full-time role heavily depends on a criterion of accuracy that is “non-inferior to healthcare professionals”. In other words, accuracy must depict the clinician's expectation in addition to saving cost, time, satisfied, and maintenance of ethical conduct (Aggarwal, R., et al. 2021. 37). Medical images have sizes that vary and in types of disease patterns e.g., haemorrhage size can be solved by feeding more data or curriculum learning that grips training phase by allowing the model to learn more complex structures. The output of feature learning allows for more abstract information of medical images and predictive patterns compared to manual inspection. The outcome of this artefact is to essentially be able to diagnose haemorrhage stroke by trained deep learning model to overcome the cost of annotating data which increases when the machine model was not trained, so the hope is to help reduce the costs of training a deep-learning model allowing even professionals or public to use the model. A fully convolutional neural network produces low quality images in comparison to the input images due to down-sampling to lower dimensions of images (Kim, M., et al. 2019. 36).

The in-depth analysis of both architecture and solution for the purpose of diagnosis will help to unravel and quantify the need for deep learning in medicine and propose yet, another satisfiable ratio of accuracy and data needed for future consideration.

1.3 Referencing

It is worth noting that the standard for referencing is Harvard.

Chapter 2

2. Literature Review

2.1 Radiomics

Radiomics is a fundamental stone to begin understanding analysis of medical images to see reasoning and enhance decision-making in the clinic, thus this review will focus on its relevant background to extent comprehension of the artefact and highlight flaws and pros. Radiomics is a quantitative method with an objective to enhance given data by a mean of mathematical analysis. Broadly applied to oncology images where radiomics show details that are normally invisible to a human eye hence are not accessible by manual inspection of images generated. Although, radiomics and AI show similarities they are not automated but rather use an AI approach to analyse an image through texture extraction or pixel interrelationships. Radiomics in return computes discrepancies in image intensity, shapes, or textures and this only provides existing data with more data (van Timmeren, J.E., et al. 2020. 10). A haemorrhage or commonly known as stroke is a leading factor of international disability and mortality cause with 80 million estimated survivors in 2016. Neuroimaging e.g., computed tomography as seen in Fig. 4 (CT) is the backbone in early diagnosis and potential prevention of a stroke. There's outgoing research of using radiomics in detecting haemorrhages with hope to root out analogous benefits that compared to oncology. There are outgoing issues with radiomics application for instance lack of duplicability due to multiple software or hardware sources that have different default parameters for pre and post processing techniques their result differ drastically or separate manufacturers. (Chen, Q., et al. 2021. 11). Normalisation techniques to enhance the repeatability such as radiomics quality score to ensure that best procedures were chosen (Lambin P. et al. 2017. 12) and image biomarker standardisation initiative (Zwanenburg A., et al. nd. 13) which puts a standard on workflow at analysis are put out there. Nevertheless, despite the issues some studies show promising results. A study in 2022 (Xie, G., 2022. 14) achieved decent AUC score when using radiomics with CT to foresee a haemorrhagic transformation (HT) respectively attaining 0.845 and 0.750 in return the RAD score managed to predict the risk of a HT and with accuracy between 68.8 to 81.6%. Evidently, it is not a reliant accuracy however, this

highlights a huge potential for applying radiomics into CT and can only be improved on in the future.

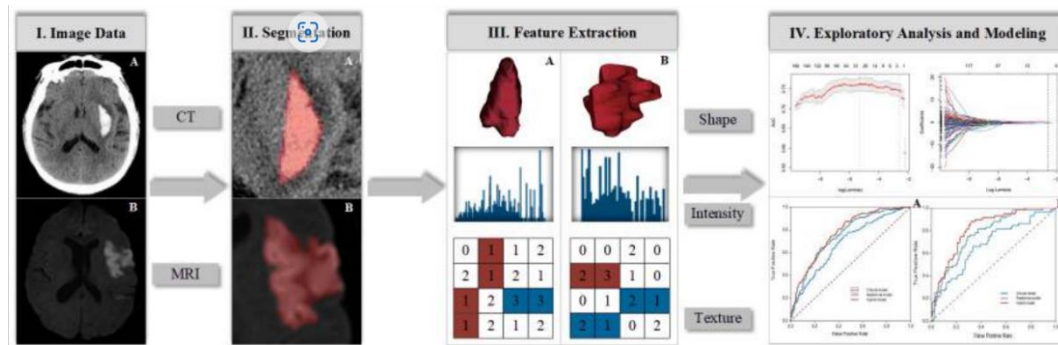


Fig. 4 Stage process of applied radiomics to stroke neuroimaging. Feature extractions A and B are showing intracerebral haemorrhage and ischemic stroke (Chen, Q., et al. 2021. 11).

2.2 Image Segmentation

Every medical image consists of interest areas and have guaranteed noise before being processed. This study (Santosh Kumar, G.U., et al. 2021. 15) exploited filtering techniques along with segmentation images to heighten the problem, furthermore, applying machine learning to segment the image. Paper continues onto the disadvantages of manual inspection which is prone to paucity of details and is time consuming. Filters such as mean, median, gaussian or fuzzy filters e.g., ATMED or TMED. An image is sensitive to faulty sensors or insufficiency of light therefore encouraging issues within the image which can limit the features, brightness or contrast thus using said filters they hope to eliminate and extract features. Segmentation by classification of boundaries, features, edges and regions of interest, region-based segmentation works on similar pictures and fuzzy c-means (FCM) connection of local complement memberships and local data distances is widely used for unsupervised approach however, they result in blurrier pixels. Consequently, with generational enhancements in technology medical experts turn towards machine learning approaches that plan to augment correct and reliable parameters for the best, optimal solution (Sharma, N., et al. 2010. 16).

2.3 Computed Tomography

A computer-aided design (CAD) with computed tomography imagery or datasets can increase the radiologist's efficacy and output true diagnosis for lung cancer as an example. A computed tomography is a well-known technology within the medical sector that uses X-rays and computers to create 3D images of the human body or concentrated parts. CT scans unlike their predecessor pure X-rays stop at dense tissues like bones; CT's output detailed views of soft tissues i.e., blood vessels, muscle tissues or brain parts which is significant for the purpose of this thesis as we investigate blood haemorrhages. Furthermore, computed tomography produces a cross-sectioned of the body to extend and uncover details invisible as two-dimensional plane offers very little detail like a modern scanner is capable of engineering up to thousand slices of an individual patient's body thus covering larger volume within a shortened time (Aggarwal, P., et al. 2011. 27). Major fear of capturing medical images is the exposure to radiation which is needed because clinic needs to penetrate human's tissue without actual physical harm to the patient. Inside the article of (Smith-Bindman, R., 2010. 28) a patient's health declined after all-together exposure to MRI and CT scanning. A Ms. C., a 59-year-old schoolteacher began to lose her hair shortly followed by vertigo, confusion and when she was admitted to the emergency room fatigue, memory loss and malaise began afterward. Close review of her CT scanning disclosed unnecessarily large radiation dose to her brain of 6 Gy which is hundred times more intensive than an average CT scan followed by 10 times the dose from the brain-perfusion mean scan and more dosage of radiation treatment for brain cancer. Around 378 other patients through the states have also witnessed those dosages and now are in class lawsuit, in result Food & Drug Administration (FDA) issued a *recommendation* to warily check CT steps leading up to the procedure. Radiation does range from their intensity depending on what part of patient's body is examined. Image quality is associated with radiation dosage given by the manufacturer, and it is an on-going competition, technical advances for instance boosting up the speed of imaging has boosted dosages. Fig. 5 depicts differences between routine dose given and gradually building up to scan an abdomen on much higher dosage to recruit more detail off the output. Radiologists were made unaware of default dose settings used when carrying out daily procedures hence standardized regulations should be put into place to not lower the accuracy of diagnosis but lower radiation dosage to avoid not only lawsuits but also harm to a human-being. National Research Council (CRC) found that patients who were exposed to computed tomography

with high dose were more potent to increased risk of cancer however, there is hesitation towards the significance of the risk which also depends on the individual as no one's body is exact in contrast to everyone. There is no body to govern liable for monitoring or collecting the dosage settings and studies found that physicians have limited understanding of radiation doses and cancer risks associated, as radiologists decide on how CTs are done. Even the technologists have no steady education regarding what doses are excessive like with prior example Ms. C. visit shown abnormal readings, but the physician did not see it as abnormal. Way to tackle the issue is to lower down the dosage of radiation given by 50% which in fact does not reduce diagnostic accuracy however, as remarked no organisation standardises enforced dosages in the US. Must the FDA interfere, and it could, facilities must benchmark the procedures as many European agencies already had this could highlight and expand the issue with dosage and make it a priority in the higher levels of courts – facilities failing to meet these requirements consequently should lose their CT certificate.

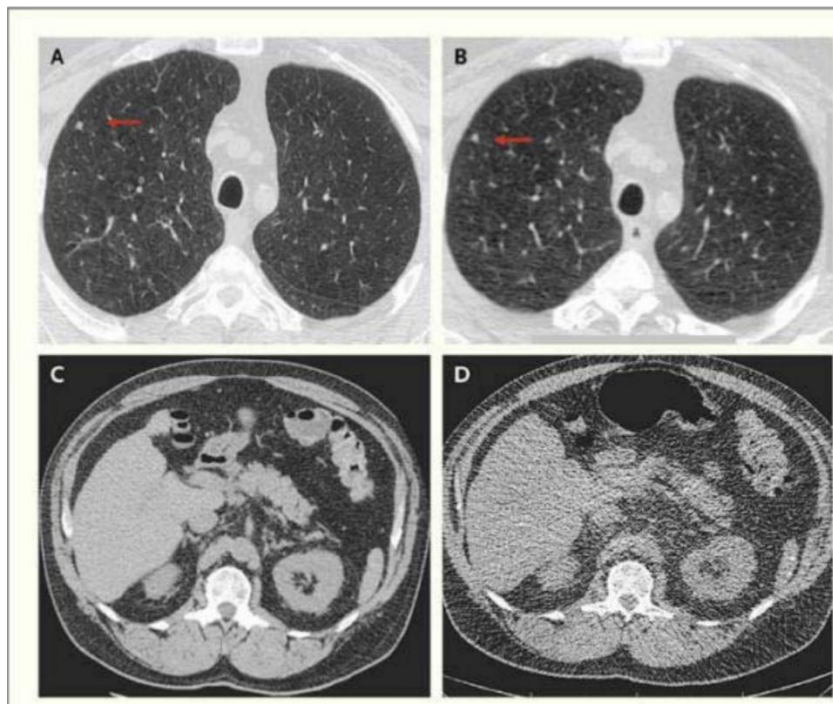


Fig. 5. Routine CT scan is on the left and low dosage on the right. (Smith-Bindman, R., 2010. 28)

2.4 Machine Learning

Machine learning as above mentioned is a system that tunes parameters thus learning to recognise patterns over time – intelligent predictions given a vast data set. In late

algorithms it was noticed that machine's accuracy towards findings is far more accurate than expert's opinion. It is a powerful tool thanks to the amassed computational power, compatible with sizable datasets and advances in algorithm design. A size of order 10 to the power of 6 a deep-learning approach is more suitable which tries to simulate the behaviour of the human brain hence given the amount of data that can be trained, validated, and tested hugely is dependent on the right approach as it may underperform in innumerable cases. Data points present fewer samples denote classic methods of feature extraction prove more reliable i.e., linear regression or decision trees that classify data sets into areas to fixed rules. An engineer needs to understand difference and appropriate usage between supervised or unsupervised learning for the artifact's solution we have gone through with supervised classification as patients with haemorrhage medical scans are compared with healthy patients. Advanced system exploiting unsupervised learning; Google's AlphaGo Zero (Silver D., et al. 2017. 23) and its neural networks won a competition by learning winning moves in the game of *Go*. It used reinforcement learning where a model is trained by rewards. Within 3 days, AlphaGo Zero exceeded the level of the supervised model from 2016, Lee variation and by 40 days of learning independently it managed to become the best *Go* player of all time without human intrusion. That is not the only case, (Kalose, A., et al. n.d. 24) used reinforced learning to come up with the best battle plan to win the game of Pokémon or another example (Gnanasekaran, A., et al. n.d. 25) a viable way of wining was used in the game Pacman and many other field cases. Standardized output choices are classification (whether an object is a dog or a cat) or regression which is a continuous variable (height measure), those are predictive outputs. Within papers we solemn come across notations i.e., given n items of data x denoted i and associated outcomes y denoted by i where i is the index from 1 to N . We're given a model which is defined by $f(x, \theta)$ where x is the data and θ is the model parameters which are tinkered with as model trains – those are all standards within the field of study. Goal is to iterate as computational powers permits with the aim to find optimal θ . Loss function sometimes called an objective or cost, it is an assessment of how closely the model prediction is within the correct value of y solemnly lower value yields better results. Deep learning uses cross entropy and mean squared error is used in regression. Upon training stage deemed done the model effectiveness is evaluated using reserved data sets or data set that wasn't present in the training phase, binary artefacts normally use sensitivity or specificity given correlation matrix. Problem arises with overfitting where model is trained to fit training data but upon seeing test data sets it does not match the outcome as in training phase i.e.,

parameters are leaning in favour of training phase rather than testing phase hence fewer data sets struggle with this but more data samples there are this issue tends to be lowered but never disappears and vice versa with underfitting. To avoid overfitting a penalty can be introduced for model's complexity increase or some type of metric that stops training when - model is overfit. As an example, k-fold cross-validation, training data is split into K segments where model is processed K times each with segment missing. A dropout method is commonly used for the neural nets. High bias for underfit model but low variance and low bias and high variance for overfit model, those are statistical concepts fundamental to reach an understanding of models' performance – those are approximations not mathematically defined equations. Bias is an error *measurement* of the model vs. the true model whereas variance measures how much model changes when given different training data and there's typically a balance between the two (Nichols, J.A., et al. 2018. 22). However, with every invention there comes issues within the system. (Anguita, D., et al. 2010. 26).

Computer vision and image processing paradigms suffer under constant limitations amidst heavier workloads regarding large datasets to produce high fidelity pictures for both machine learning and medical analysis. Recent studies found that embedded systems require plentiful computational resources considering their size and fitness for analysis but also deep neural networks especially with its enhancements, cause catastrophic forgetting to reuse old data at intervals with new data (Janusz Snieg. 2023. 49).

2.5 Convolutional Neural Networks

Moving toward image classification there are ideals that are standardised; thus, this following review heavily focus on convolutional neural networks as a primary candidate for the potential solution. This network utilises a deep-learning model (Vaz, J.M. and Balaji, S. 2021. 17) named convolutional neural network (CNN) through collective neurons that are interconnected sequentially, data is processed in multiple dimensions. Design orientates around learning simpler patterns and slowly transitioning to complicated patterns. Problem with deep learning is with number of parameters hence addition of multiple layers aids and solves the issue. Weight sharing and local connectivity. Weight sharing entails identical weights across the nodes in a layer in addition, local connectivity occurs when a node receives input from local values in an array and outputs only parts of the input vector as a result. In simple terms, features are automated hence can be highlighted.

A singular node detects local features from the input with down-sampling as stated minimises number of parameters required. Neural network is inspired by animal neural systems i.e., human (Hijazi, S., et al. 2015. 18). Rose flower research (Anjani, I.A., et al. 2021. 19) and their CNN model gotten 96.33% accuracy on datasets that were tested. Issue resides with images that if its direction or original scale properties are present, CNN potentially faces an obstacle to identify the object as a result the image is downscaled to lower resolution for adaptability with computational power being saved which proves to be energy efficient. As training phase is about to be launched, data is augmented which flips the picture or rotates it – changing its horizontal and vertical direction by a small factor to train the dataset. The system recognises the picture by a pixel cluster that are arranged in very distinctive patterns however, as components they are not recognised due to lack of coordinate frame where human eye does – coordinate frame is a list of orientation and features of an object (S, Bhuiya. 2020. 20). To conclude, convolutional network to be trained flawlessly, a big dataset, stationary 2D image and good hardware are a necessity. However, the decision boundary (a line where classes are divided by, and nodes are plotted on either side if they match the characteristics) has a potential to overstrain if training data doesn't possess the ownership of its samples belonging to the appropriate class (Alzubaidi, L., et al. 2021. 21). Goal is to receive an output given input overtime; it should produce from an output from prior epochs based on large datasets. This approach has viability given by numerous of research papers et cetera thus, should be considered as the main contender or one of main to tackle the state-of-the-art.

2.6 Aims & Objectives

The aim of this artefact solution is to create a system that can detect a haemorrhagic stroke. With an attempt to do this by localising and outlining the regions of interest of every CT image by applying a deep-learning model for the purpose of diagnosis and binary classification.

	Objective 1	Objective 2	Objective 3	Objective 4
Specific	Identify applicable datasets	Train the model to achieve	Research common deep learning	Accumulate sufficient or max amount

	consisting of haemorrhagic strokes and stroke-free patients to train the model with efficient image quality.	reliable accuracy and metrics medically appropriate that would only solidify clinician's trust into the model.	approaches to anomaly detection for the purpose of medical environment.	of evaluation metrics for the purpose of in-depth analysis.
Measurable	Minimum acceptable dataset size is 1000 images to achieve higher accuracy percentage. 48 by 48 as a minimum or 448 x 448 as a maximum.	More metrics present will only strengthen the results.	Compare or choose one model to build a neural network that can correctly classify images.	Classification report offered by "sklearn" library.
Assignable	An academic or legitimate source is required and Kaggle is a main contender for datasets.	Torch as a dependency has lots of standard metrics to be used for evaluation.	It is vital if not necessary to research to gain an understanding of available ways of constructing neural networks.	Image classification may pose some issues regarding the adaptability of variables to be passed through parameters

				for the purpose of classification report, it can result in needing to transform datatypes.
Realistic	If images have white spots on the patient's brain this is likely to indicate a haemorrhagic stroke in comparison to a stroke-free scan. Data cannot be corrupt, can have noise as it can be reduced, it must have legit source and must not break patient's confidentiality to not risk the	Realistically, this is only few lines of code and adaptation.	It is relevant to research already existing models that aim to solve the solution or propose new findings for further research, finding its weaknesses particularly in gaining clinician's trust.	Appropriate knowledge and workings around the library are required to understand the need for arrays, tensors which are three dimensional arrays.

	ethical aspect of this solution as no human participants are present. If database has high quality images less of it may be required and vice versa.			
Time-bound	1 Week.	1 day.	1 month	1 day

Table 1. Table of individual SMART objectives.

Aims and objectives are significantly not the same, almost new due to deeper analysis into finding the appropriate solution for the artefact, with conclusions drawn from literature review new objectives were required.

Chapter 3

3. Requirements Analysis

3.1 Functional Requirements

There are important steps for consideration when addressing a deep learning solution in medical imaging. Anatomical structures vary from patient to patients noticeably i.e., skull structure or complex brain areas. It is possible a vast database is collectively required to achieve desired accuracy and to not over-step this boundary. How large does the training set needs to be to get satisfiable accuracy? Accuracy of artefact's binary classifier is dependent on data that is high-quality and its' presence. Learning curve in modelling classifier's performance as a function of the training data sample avows prediction for sample size needed to train a classification model. Normally represented as an inverse power law function. The accuracy of a classifier y ; is conveyed as a function of the training set size x ; with unknown parameter $\mathbf{b} = b_1, b_2$ which denotes to learning rate and decay rate. Model fit thinks that the accuracy grows asymmetrically to 100% (Cho, J., et al. 2016. 38).

Eq. 1 Accuracy of a classifier (Cho, J., et al. 2016. 38).

$$y = f(x; \mathbf{b}) = 100 + b_1 \cdot x^{b_2}$$

An image is required to have sufficient quality to be considered as a candidate for the binary classification. Image resolution before any pre-processing or processing phases can be manually inspected by a human observer to determine how low-quality image can be before it becomes unrecognisable to a human spectator. In comparison with the performance lower quality vs. higher quality, detection of hernia or pulmonary nodules saw highest fractional improvements in AUC at higher image resolutions. Optimally, the image quality should therefore be aimed at 256 x 256 at minimum (Sabottke, C.F. and Spieler, B.M. 2020. 39).

3.2 Non-functional Requirements

Performance metrics measures how well model has done in the prediction of dataset. A binary classifier is a prediction for a sample to be classified to true and false class diagnosing a build-up of blood fluids or healthy patient or 1 for true and 0 for false or, a multilabel classifier chooses only one and only one of more than two predefined classes i.e., a haemorrhage may be a combination of epidural, subdural, intraventricular, and/or intraparenchymal haemorrhage labels assigned to each slice. Assigning a threshold e.g., 0.5 to the binary classifier gives a chance for the model to portray which class is the predicted image closest to (Erickson, B.J. and Kitamura, F. 2021. 40).

Model to achieve an objective where accumulation of metrics is required, explainability is a requirement to reach that objective thus completing a step in achieving the aim. Explainability insinuates “extent to which the internal mechanics of ML-enabled system can be explained in human terms”. It’s prudent due to clinician’s lack of knowledge in this field excluding medicine and based on assumptions but even a patient should understand what data represents. Justifiability connotes “ability to show the output of an ML-enabled system to be right or reasonable”. What is meant by this; there is room for error as computers are designed to assist clinicians in either supporting their reasoning for concern or predict and diagnose to reduce morality, therefore a model which makes mistakes is acceptable because it’ll work fairly; “ability of a system to operate in fair and unbiased manner” and transparency “ability of the system to clarify the reasoning for its decisions to a human user”. Transparency is paramount to justify computers’ reason for said prediction, preferably an outputted image should return a highlighted region of interest where a build-up of blood fluids occurred (Khan, M., et al. 202. 50).

Chapter 4

4. Design & Methodology

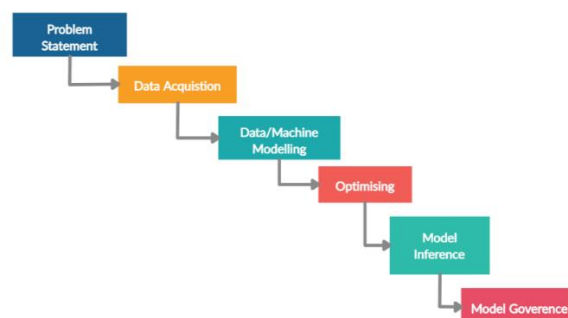
4.1 Project Management

Waterfall is a software development methodology that embraces sequential stages that mimic waterfall in the nature. Main concern regarding this methodology is its' heavily dependent on previous iteration in the stages, because like a waterfall it is not possible to swim upwards due to the current. Waterfall is not fitting unless problem statement is opted upon, agreed, and defined (Wilfred Van Casteren 2017. 44). However, (Samuli Laato., et al. 2022. 45) argues that "... but the waterfall model is more readable [...]" and "I prefer knowing the steps involved in the development process."

There's a good justification on why to pursue waterfall methodology, because of its' simplicity and problem statement is clearly outlined with research, aims, objectives and requirements to reach those objectives to fulfil an aim considering the medical field – which is fundamentally important as clinicians need to trust the model.

Firstly, software solution needs a clearly defined problem statement. Data then is acquitted from a reliable source, appropriate machine model was chosen, an optimiser to tweak parameters to yield best possible performance metrics to rightly evaluate performance and lastly, model governance is to sustain model with up-to-date data and frequently test it before final completion (Fig. 6).

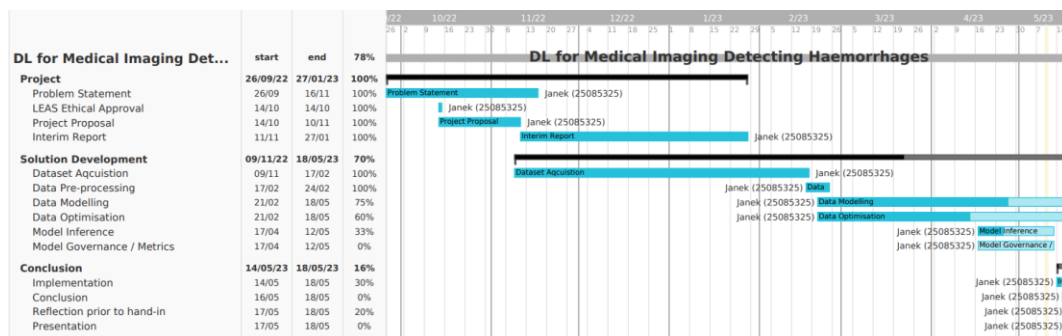
Fig. 6 Waterfall as a software development methodology for this project.



A Gantt chart is a helpful way to visualize a project's schedule using a bar chart. They're easy to read and commonly used to show the timing of different project activities. These charts display the start and end dates of the main parts of a project, including both the individual tasks and the overall project phases. In some cases, Gantt charts also show how these tasks depend on one another.

Inclusive, Gantt charts help project managers see all the major stages of their project briefly, with each stage represented as a bar on the chart. The time scale for the project is shown along the top of the chart. Gantt charts are a quick and easy way for project managers to estimate how long the different tasks will take. Sometimes, it can be useful to start with the deadline for the entire project and work backwards from there to make sure the timeline is feasible. Once the main goals of the project are established, a detailed Gantt chart can be created to help keep track of progress as presented in the **Figure 7** below which outlines the scope and milestones for the project, its' latest up-to-date version will appear in the conclusion (Albrecht, J. 2018. 47).

Fig 7. An overview of the project life cycle presented in GANTT format.



4.2 Risk Analysis

The field of risk has two main objectives: firstly, to use risk assessments and risk management to study and mitigate the risks associated with specific activities (such as operating an offshore installation or making an investment). Secondly, to conduct broader research and development on risk-related concepts, theories, frameworks, approaches, principles, methods, and models to better understand, assess, characterize, communicate, and manage/govern risk in a general sense (Aven, T. 2016. 46.).

Table 2. A risk analysis table with risks that can come as project progresses.

Risk	Image Noise	Problem-Solution Alignment	Excessive use of resources	Unintended behaviour & consequences
Impact	An image has noise which can interrupt classifier with erroneous output or misclassify labels. This could prove disastrous in detecting haemorrhages.	Applying machine model to the wrong problem.	ML or DL heavily depend on data that is available, vast dataset means better accuracy but lengthens the time taken to train our model taking hardware limitations into considerations too, a CUDA enabled GPU is preferred for parallelisation which reduces time but also there's an issue debugging in case of errors.	Vulnerability to adversarial examples and feedback loops which can add biases in training data.
Control	Apply a pre-processing on data or clarify that all inputs are clear and visible.	Research concluded evidently suggests that AI is the only way forward in combating issues with medical	Closely measure overfitting, underfitting, variance	Retrain the model if there's an ambiguity, update database of images, assess the

		imaging, this model created for the purpose of solution has been carefully considered personally and via supervisor communication.		data for biases or representativeness.
Severity (1-5)	5 [high]	1 [low]	3 [medium]	3 [medium]

4.3 Toolsets and Machine Environments

4.3.1 Python (programming language)

Python is a high-level programming language which is interpreted and supports object-oriented paradigms. The simplicity of syntax and accessibility towards dependencies i.e., PyTorch or Tensorflow which are widely available to the public, professionals, and academics. Direct support of Jupyter Notebook is also possible where segments or cells of code can be run to display a table or cells of images very useful for deep learning as it allows for simple debugging in case of runtime error or logic errors.

IDE or known as integrated development environment where a programming language can be run in, debugged or compiled example of those is PyCharm, Visual Studio Code or Atom.

4.3.2 Frameworks

PyTorch is a featured framework designed for manufacturing deep learning models commonly applied to image recognition and language processing notably in self-driving cars by Tesla which detect objects when car is in auto-pilot mode avoiding collision. This framework is natively written in Python with majority basis residing in C++, thus is pythonic, fast and data definitions with addition of other dependencies to represent data plainly i.e., Pandas or Matplotlib. Through immediate execution of dynamic three-dimensional arrays or

tensors computation with automatic differentiation and accelerating the GPU hardware it maintains the performance in comparison with fastest current DL libraries. Researchers or engineer experts are capable of manually adjusting the architecture in search of superior performance and furthermore, the internal architectonics are intelligible thus tolerating an implementation of additional features to adapt the model to new situation and keeping up the pace in the field. A bidirectional data exchange is present with outside libraries a mechanism for converting arrays to tensors is an example of such exchange e.g., ``torch.from_numpy()`` and ``numpy()``. No data is copied only an object is described by how to interpret a memory region which is shared on both sides by virtue of cheap computation and size does not impact this process, which only proves how the system was designed to be extensible. Tensors have versioning system helping to track direct modification of the data, certifying right data is used. A future has a bright light ahead of it because, developers are pursuing a way to optimise models outside Python interpreter which is known to be slow in comparison to C; PyTorch JIT. By exploiting Python bindings as they're commenced by YAML meta-data files open a door for communities to create bindings to multiple other languages and think of a use for them e.g., StarCraft. Multi-processing dependency from Torch automatically moves elements in tensors sent to other processes to share memory, vastly remodelling performance in users' favour and weakens process isolation in return produces closure that is comparable to regular threaded programs **Fig. 8** expositions the benchmark amongst other available dependencies (Paszke, A., et al. 2019. 42).

Framework	<i>Throughput (higher is better)</i>					
	AlexNet	VGG-19	ResNet-50	MobileNet	GNMTv2	NCF
Chainer	778 ± 15	N/A	219 ± 1	N/A	N/A	N/A
CNTK	845 ± 8	84 ± 3	210 ± 1	N/A	N/A	N/A
MXNet	1554 ± 22	113 ± 1	218 ± 2	444 ± 2	N/A	N/A
PaddlePaddle	933 ± 123	112 ± 2	192 ± 4	557 ± 24	N/A	N/A
TensorFlow	1422 ± 27	66 ± 2	200 ± 1	216 ± 15	9631 ± 1.3%	4.8e6 ± 2.9%
PyTorch	1547 ± 316	119 ± 1	212 ± 2	463 ± 17	15512 ± 4.8%	5.4e6 ± 3.4%

Fig. 8 Training speed for 6 dependencies applying 32-bit floats (Paszke, A., et al. 2019. 42).

4.4 Data Acquisition and Annotation

Data is acquitted from unwavering source from public site named Kaggle (*Kaggle.com*), it is a community-based network where users can find datasets, they wish to use

in building their own machine models, publish datasets for public use, work with other data scientists and machine engineers but also its' a basis to enter challenges and win prizes.

To pre-format the data, a script in Python was written utilising “split-folders” dependency, which splits a folder(s) into train and testing folders – script can be found in “split_data.py”.

A small dataset is located within the directory “sml_labels” possessing 200 CT scans split by half for haemorrhage and normal outcomes, it was collected by this user (F. Kitamura. 2018. 54) from Google Images.

A directory “x_lrg_formatted” is formatted into haemorrhagic folder and normal or no haemorrhagic, folders are used as related classes. Dataset consists of ~6794 files in .jpg extension format, downloaded and pre-formatted for the purpose of the solution from (Helwan, A., El-Fakhri, G., Sasani, H., & Uzun Ozsahin, D. 2018. 52), collected from Near East Hospital, Cyprus.

4.5 Haemorrhage Detection

It is imperative to fundamentally understand the meaning of computed tomography images to distinguish a patient who undergone sudden stroke and patient who haven't undergone it.

Patients who develop intracerebral haemorrhage originates lesion expansion over the course in a day. Lesions can migrate in the cause of primary bleeding and dissects through less-dense white subject and into ventricles resulting in increased and felt pressure. Physical examination is necessary to assess not only the vital signs but also if intubation is required for safety of the imaging however, this is only worth to mention to further visualise the procedure but for the purpose of this solution it's only a food for thought. Computed tomography helps to differentiate between acute intracerebral haemorrhage or ischemic stroke. Imaging identifies the size and location of the haemorrhage, estimated volume of the stroke is determined by multiplication of the maximum length, maximum width and number of transverse CT scans cuts divisible by 2 or using ABC/2 method where A is the dominating haemorrhage diameter; B, diameter at 90 degrees and C, the estimated number of CT slices with haemorrhage multiplied by slice breadth. Quantomo method is reliable in comparison to ABC/2 method because it can detect smaller changes. CT can determine rough age of hematomas, by

calculation of the density of the lesions measured in Hounsfield units, where typical value for blood presence is 30 between 45, grey substance is in range of 37 and 45 and, white substance between 20 and 30. In regards to this topic, when examining acute intracerebral hemorrhage (ICH), fluid blood levels can be identified as a horizontal boundary between a layer of hypodense bloody serum situated above hyperdense settled blood. Detecting fluid blood levels in acute ICH can be somewhat sensitive (59%) in indicating the presence of coagulopathy, which refers to an abnormal prothrombin time and partial thromboplastin time. However, they are highly specific (98%) in confirming the condition. In cases of thrombolysis-related ICH, blood/fluid levels are also common and are associated with higher haemorrhage volumes. Within the first 72 hours after onset, a hypodense area can be observed surrounding lesions due to the edema that encircles the brain tissue, and a significant mass effect may be apparent as well. Between three to 20 days after onset, the lesion area typically decreases in size and becomes less intense, losing approximately 1.5 Hounsfield units per day as depicted in (Fig 9). The periphery of the lesion usually acquires an uneven profile, which takes on a pseudoabscess (ring-like) appearance when viewed under contrast (Macellari, F., et al. 2014. 35). Regularly the site for hypertensive intracerebral haemorrhage is the putamen but there's possibility of it occurring in other sectors (Fig. 10). A lobar haemorrhage is indorsed to amyloid angiopathy in the elderly, but hypertension is an alternative factor and sadly carry poor prognosis (Rymer, M.M. 2011. 34).

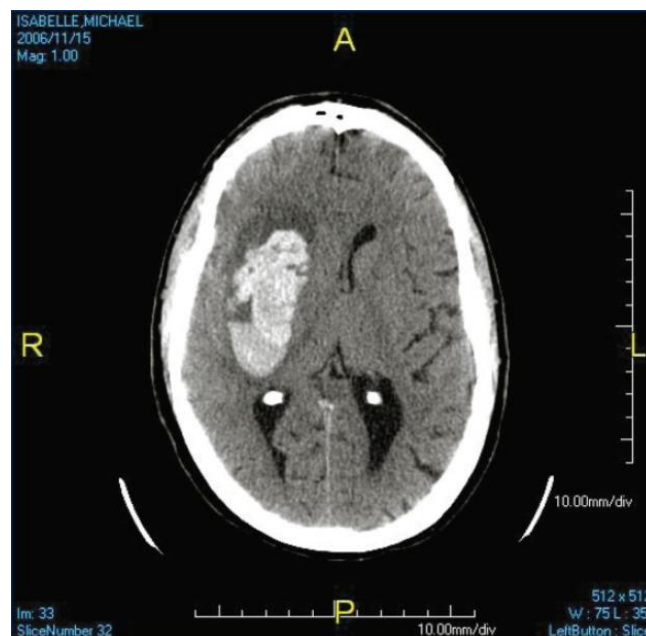


Fig. 9. Putamenal Hypertensive Haemorrhage (Rymer, M.M. 2011. 34).

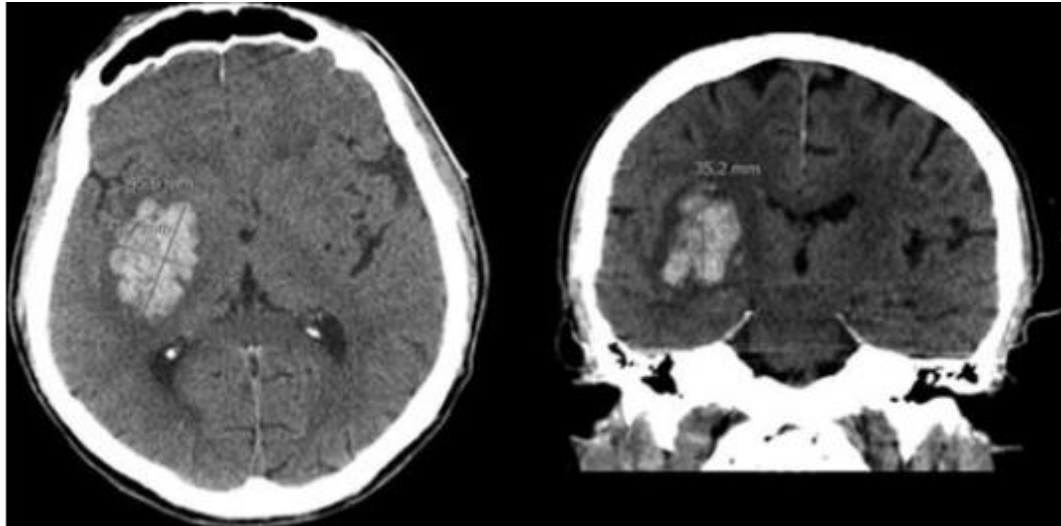


Fig. 10. Intracerebral Haemorrhage on CT image, method for detection is ABC/2 formula (Macellari, F., et al. 2014. 35).

4.6 Architecture of a CNN Model

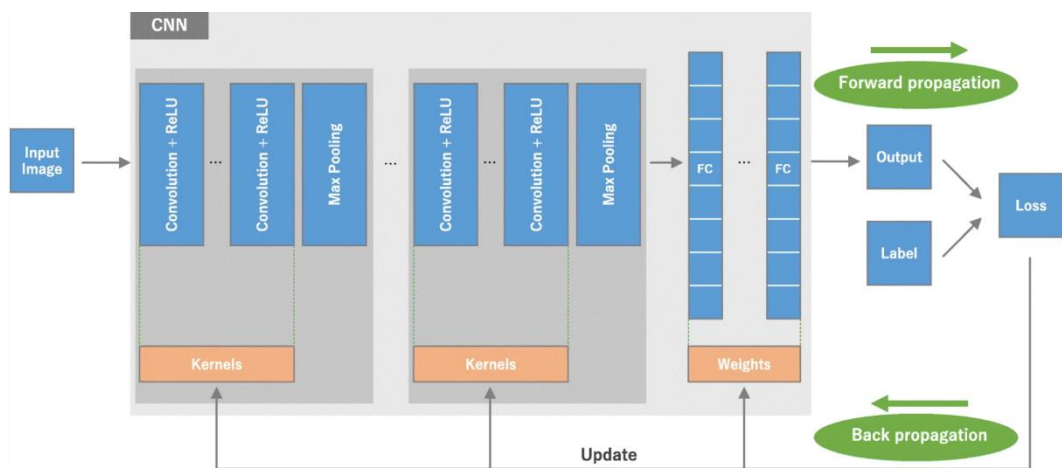


Fig. 11 An overview of a convolutional neural network architecture and training phases (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

The project's artifact utilises a deep-learning model called convolutional neural network to classify images based on present haemorrhage strokes outputting true or false, whether it is present or not.

The (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32) article fully explains the terminology and architecture of a convolutional neural network. We'll begin with basic terminology and further down the sub-chapter this thesis uses the paper to explain the

CNN. A “parameter” is a variable responsible for automatic learning when carrying out the training phase of a model. A “hyperparameter” is a variable that needs to be manually set prior to the training process. A “kernel” is referring to the series of learnable parameters applied in the convolution operations. Lastly, a term “weight” is used in-between with parameter however, this term is frequently known when referring to a parameter outside of layers i.e., kernel. **Fig 11** is an overview of said convolutional neural network.

Convolutional neural network is a building block made from several layers. Heavily inspired by the animal visual cortex hence it has a pattern-like grid for images designed to automatically withdraw features seen normally invisible to a human eye. Pixels within the digital images are stored as two-dimensional array of numbers and a small grid of parameters named kernel. A feature extractor, optimised over course of time is applied at each image position wherever feature may be which is very renowned in image processing. Training is a process of optimising the kernels to initially minimise the difference between outputs and ground truth labels through backpropagation – a type of optimising algorithm. Fig. 12 represents a way that computers depict digital images. Input of each layer in the model is defined by three dimensions; height, width, and depth (or channel number) where height is equal to the width (Alzubaidi, L., et al. 2021. 41).

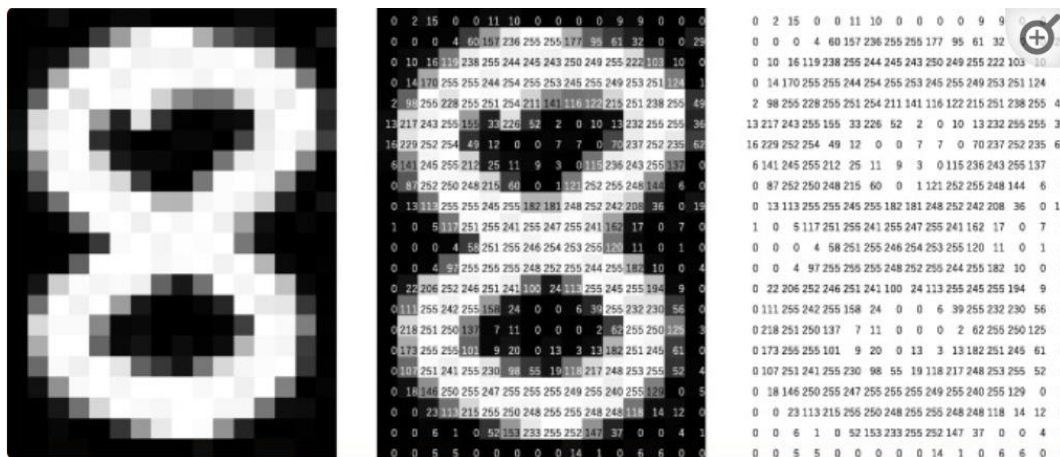


Fig. 12. Image portrayed by a computer is an array of numbers. Label C on the right make up the image with numbers ranging from 0-255 which correspond to individual brightness of a pixel (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

There’s a repetition of stacked layers i.e., convolutional layers or pooling layer followed by one or more fully connected layers. A convolutional layer is a stone base of the model

architecture responsible for feature extraction by combining linear and non-linear operations i.e., convolution operation and activation function (Fig. 13). A small array of numbers or a kernel, applied through the input is called a tensor. Element-wise product from each element of the kernel and input tensor is calculated at individual position of the tensor and totalled to obtain the output structure in the analogous position of the output tensor called a feature map (Fig. 13). Convolutional layer calculates a dot product between its input and the weights however, the inputs are undersized areas of the initial size as seen in **Eq. 3**. Calculation of feature maps is repeated by using multiple kernels to create a subjective number of feature maps which represent distinct attributes of a tensor in return can be considered as a feature extractor. Key hyperparameters are the size and number of kernels normally 3 by 3 or sometimes larger, and this defines convolution operation. Randomised values are assigned to the weights in the kernels and with next training phase those weights are adjusted to allow kernel to extract more features (Alzubaidi, L., et al. 2021. 41).

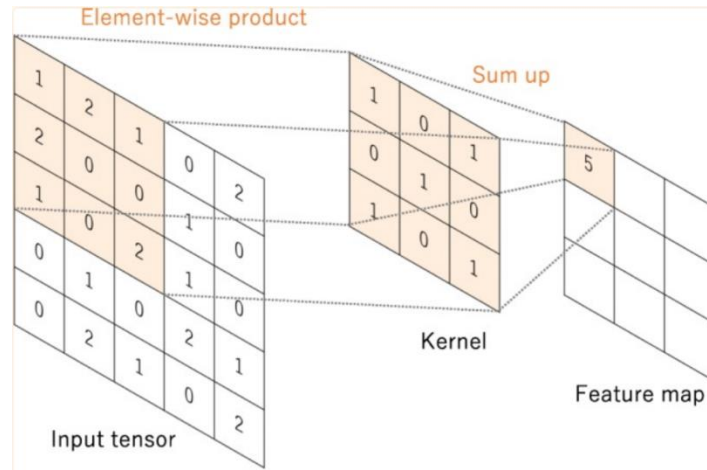


Fig. 13. A visual explanation of how feature map is obtained in the element-wise product (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Eq. 2. Dot product calculated within the convolution layer.

$$h^k = f(W^k * x + b^k)$$

Padding is a solution to allow the center of each kernel to intersect the outer regions of the input tensor which reduces height and width of the feature map. However, padding solves this by adding rows and columns of zeroes on each side of input tensor to fit the kernel centre on outermost element and keep identical plane dimension through the convolution operation.

If padding or zero-padding did not exist feature maps would minimise after the convolution operation (Fig. 14). A stride is a distance between two successive kernels positions A and B, default understanding of what value to use is 1 however, down sampling is sometimes achieved by increasing this value or alternatively pooling operation is used.

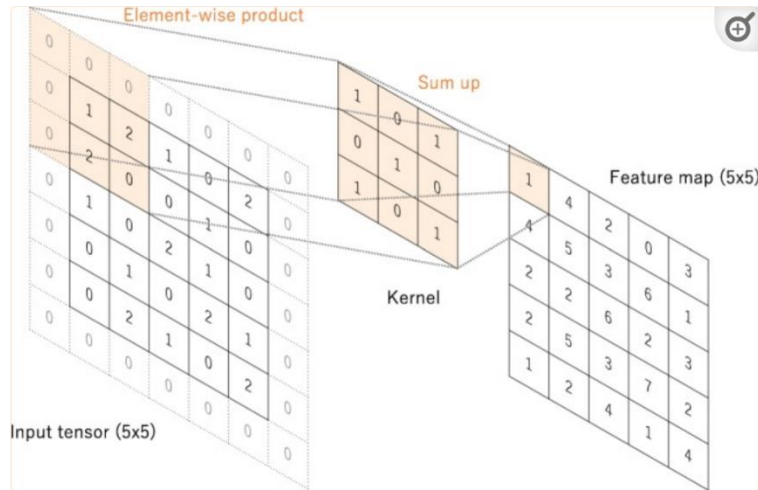


Fig. 14. Input dimension of 5 by 5 is identical to the one in the feature map thanks to padding (Yamashita, R., Nishio, M., Richard and Togashi, K. 2018. 32).

Weight sharing is a key element of a convolution operation. Letting the local feature patterns extract via kernel translation as kernels move across all image positions and find local patterns. Spatial hierarchy is understood by feature patterns by down-sampling in union with a pooling operation resulting in substantial field of view. Number of parameters are lowered to increase model efficiency to learn in retrospect to fully connected neural networks. Those are all characteristics that make up weight sharing. Convolution layer requires to identify the kernels that are appropriately suited to tackle a task based on given training data. Within the layer kernel is the only parameter that automatically adjusts its values as training phase proceeds in comparison to hyperparameters that are required to be set manually prior to the training. Outputs of a convolution layer are therefore passed through a non-linear activation function and remember those are mathematical representation of a biological neuron activity mimicking animals' visual cortex. Most frequently used non-linear activation function is rectified linear unit i.e., "ReLU" which sums up the function by this definition:

Eq. 3 Mathematical definition of ReLU (Abien, F. and Agarap 2019. 33).

$$f(x) = \max(0, x).$$

Rectified linear units is a thresholder for values that outputs 0 when $x < 0$ and outputs a linear function when $x > 0$ (Abien, F. and Agarap 2019. 33) and (Fig. 9) is a clearer representation.

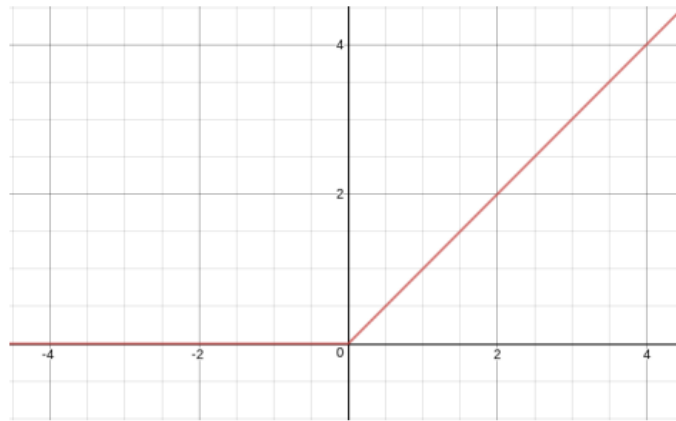


Fig. 15. A visual representation of Rectified Linear Unit. While $x < 0$ output activation function but when $x > 0$ outputs linear (Abien, F. and Agarap 2019. 33).

Continuing with an explanation of the pooling layer, it is a down-sampling method with aim of reducing the in-plane dimensionality of the feature maps to introduce a translation invariance small shifts and distortions and decrease the number of subsequent parameters that can be learnt. Max pooling by far is the popular way of pooling which extracts patches from the input feature maps, outputs the max. value in each patch and discards all the other values. Pooling function is applied to an adjacent area of size. Down sampling every feature map leads to the reduction in the network parameters in return accelerating the training process and in turn gives better control of the overfitting issue (Alzubaidi, L., et al. 2021. 41). Max pooling with present filter of size 2 by 2 in addition with a set stride value at 2 is used in practice, down-sampling feature maps by a factor of 2 where depth remains unhinged, but width and height adjust. Global average pooling is an extreme way of down-sampling, where a feature map with size columns by rows is down-sampled into a 1 by 1 array by averaging of all the elements in each feature map whereas the depth again, remains unchanged. This is only brought before the fully connected layers. Pooling and global average pooling share same advantage but global enables CNN to accept inputs of a variable size.

The output feature map of the last convolution operator or pooling layers are per norm transformed into a one-dimensional array of numbers (vectors) and connected to one or many

fully connected layers also known as dense layers where each input is allied to every output by a learnable weight. When the features are extracted, they are mapped by a subdivision of fully connected layers to the final outputs of this neural network. Final fully connected layer should or has the same amount of output nodes as the number of classes – in this case, stroke or no stroke pursued by non-linear function i.e., ReLU. However, the last fully connected layer and its activation function are different from the rest. A function cannot be assigned at random hence there's needed to choose appropriate activation function. If the problem is multiclass classifier, softmax function is applied to normalise the output real values from the last fully connected layer to target class probabilities where value range from 0 to 1 and all values total 1. One of the objectives/aims hopes to tackle multiclass solution however, binary classification is a realistic target thus its' last layer activation function is sigmoid.

When we use a weight sharing feature in a network, it means we don't have to train as many parts of the network, which can be helpful. This can make the network better at understanding new data it hasn't seen before and prevent it from memorizing the training data too well, which can be a problem. By doing this, the network becomes more organized and is better able to recognize important parts of the data (Alzubaidi, L., et al. 2021. 41).

4.6.1 Loss Functions

Loss functions are present in the output layer for predicted error calculations created along the training samples in the machine model. This value reveals the contrast between the actual output value and the predicted value. Model will optimise itself through the learning process. First parameter of a loss function is the estimated output, second parameter is the actual output or label. There are several loss function types, however, cross-entropy or softmax loss function is commonly used in measuring the model performance. Below bullet points highlight key features (Alzubaidi, L., et al. 2021. 41).

- Output probability – $\{0, 1\}$.
- Substitution of the square error loss function in multi-class problems.
- Uses softmax activations to generate the output by probability.

4.6.2 Data Normalisation

Model and data must execute well on both training and testing splits however, that is difficult to achieve, and trade-offs must be put in place. Hence this section focuses on different ways of generalising data.

A batch normalisation is for tackling performance of the model where it deals with output activations following Gaussian distribution reducing the in each layer variation in the activation distribution defines “internal covariance shift” of the activation layers. By reason of continuous weight update causes a big shift above mentioned which can happen if the images are not consistent or are from numerous dissimilar mutually sources thus increasing training taken by consuming extra time for convergence hence batch normalisation is applied in the architecture. Batch normalisation is advantageous thanks to; (a) prevenance of vanishing gradient (b) control poor weight initialization (c) reducing time for network convergence (d) little influence on regularisation thus over-fitting is reduced (e) struggles to decrease training dependency on hyperparameters (Alzubaidi, L., et al. 2021. 41).

4.6.3 Parameter Tuning

Optimiser selection is crucial for the capability of a learning process. Two issues arise; (a) learning algorithm optimiser (b) use of many enhancements along with the algorithm to improve output value. Gradient-based learning is the usual selection, network parameters ought to update via absolute training epochs meanwhile looking for local optimising answer to minimise the error. A learning rate although hyperparameter is defined as the step size of the parameter updating, it needs to be carefully set as value closer to null or big ought to bring irregularities within the model deeming it “useless”. Gradient Descent minimises the training error by updating network parameters with each epoch. An objective function or slope must be computed by a first-order derivative. Parameter is updated by back-propagation performance, where gradient at every neuron is back-propagated to all neurons in the preceding layer; represented as a mathematical equation in **Eq. 4** (Alzubaidi, L., et al. 2021. 41).

Eq. 4 Back-propagation mathematical equation (Alzubaidi, L., et al. 2021. 41).

$$w_{ij}^t = w_{ij}^{t-1} - \Delta w_{ij}^t, \quad \Delta w_{ij}^t = \eta * \frac{\partial E}{\partial w_{ij}}$$

4.7 How is the system evaluated?

Model life cycle is split to three sections, training model on input dataset for the task at hand, validating on data that is not part of the training data to assess performance on data yet not seen. As training phase concludes, it is tested on test dataset where final metrics *should* be calculated (Hicks, S.A., et al. 2022. 43).

Eq. 5 Binary classification expression of a total data model (Hicks, S.A., et al. 2022. 43).

$$p(X, \alpha) = \alpha p_P(X) + (1 - \alpha) p_N(X),$$

A binary classification problem in here is expressed by mathematical equation (Eq. 4). Where X is data sample(s), p_P/N expresses positive/negative class allocations and α , or *alpha* is the mixture parameter of the positive class where its' formula is $(N \times p) / (N \times p + N \times n)$ with $N \times p / n$ is the total of positives or negative class data samples. Following performance metrics or evaluators utilise four elements within the confusion matrix (Eq. 6) (Hicks, S.A., et al. 2022. 43).

Eq. 6 Confusion Matrix (Hicks, S.A., et al. 2022. 43).

$$\mathbf{M} = \begin{pmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{pmatrix}$$

TP or True Positive refers to the number of correctly classified positive samples, e.g., a person was supposed to visit Hyde Park and they have. **TN** or True Negative refers to the number of incorrectly samples classified as positive samples, e.g., this patient was hurt but wasn't. **FP** or False Positive is referring to the number of data incorrectly marked as positive. **FN** or False Negative denotes back to the number of sets incorrectly classified as negative. Accuracy is measured as a ratio of correctly classified samples and the sum of samples in the evaluation dataset. However, the downside of this measure, it can be misleading required as different class proportions can be assigned to rampant class thus achieving high accuracy but this decreases trust in the model; range is 1 where all positive and negative samples were predicted right and 0 is none of the samples were correctly aligned to belonging classes (Eq. 6) (Hicks, S.A., et al. 2022. 43).

Eq. 7 Accuracy Formula (Hicks, S.A., et al. 2022. 43).

$$ACC = \frac{\# \text{ correctly classified samples}}{\# \text{ all samples}} = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall or sensitivity refers to the rate of positive samples correctly classified and assessed as a ratio of correctly positive and all samples ascribed to the positive class. Bounded to [0, 1], 1 is a perfect prediction of positive class whereas 0 is an incorrect prediction of all positive data points. Medical studies greatly benefit from this metric where it is desired to miss as few positive instances as possible, translating into high recall (Hicks, S.A., et al. 2022. 43).

Eq. 8 Recall Formula (Hicks, S.A., et al. 2022. 43).

$$REC = \frac{\# \text{ true positive samples}}{\# \text{ samples classified positive}} = \frac{TP}{TP + FN}$$

Specificity denoted by the negative class version of the recall or sensitivity is the rate of negative datapoints (samples) correctly classified. It is computed in ratio form between correctly classified negative samples and all samples classified as negative (Hicks, S.A., et al. 2022. 43). Naturally all performance metrics range [0, 1] respectively.

Precision is the proportion of the get back samples in relevance and is calculated as a ratio of correctly classified samples and all samples assigned to that class. Again, [0, 1] is present and 1 defines all samples in the class rightfully predicted and 0 defines no correct predictions, refer to **Eq. 9**.

Eq. 9 Precision Formula (Hicks, S.A., et al. 2022. 43).

$$PREC = \frac{\# \text{ samples correctly classified}}{\# \text{ samples assigned to class}} = \frac{TC}{TC + FC},$$

C in formula directly overhead terms *class*, and within the binary classifier it can be either a positive or negative sample. Positive case of precision is also known as Positive Predictive Value (PPV) and negative case is Negative Predictive Value (NPV) their meanings, formulas and conversation regarding F1 score, Matthew's correlation coefficient or a Threat Score, we refer reader back to (Hicks, S.A., et al. 2022. 43).

Many metrics are present within the field of deep learning to calculate the feat of a model hence using only a handful can yield unexpected results, referring to the risk analysis, is not acceptable in a clinical setting. Thus, luxuriating more than two metrics and interpreting those holistically befalls a foundation. A class whether positive or negative should have its' own metric computation. Although, a performance metric is indispensable as from non-engineer perspective this titivates trust but still should be frowned upon. Training and testing splits are presented as the model must shun from the bias (Hicks, S.A., et al. 2022. 43).

Chapter 5

5. Implementation

This section demonstrates solution from a technical perspective, showing the different components of the Python software, including code snippets and screenshots from Jupyter Notebook as figures to provide detail where necessary.

5.1 Splitting Data



```
split_data.py > ...
1 import splitfolders
2
3 path = "C:/Users/janek/haemorrhage_project/datasets/x_lrg_formatted/all"
4 print("Splitting folder {} into train/test folders".format(path))
5 splitfolders.ratio(path, output="output",
6 seed=1337, ratio=(.8, .2), group_prefix=None, move=False)
```

Fig. 16. Python script to split the data, only if data has been already formatted into folder-labels.

By utilising a library installed by “pip”, it was necessary to split the dataset to corresponding train and testing directory prior to training the classifier, because of quick testing without this approach there was influx of wrongly classified images and huge loss in test data samples.

Path variable is initialised with path to the folder that needs to be split into training and testing datasets. Split folder takes in parameters to include path, produce an output folder with returned splits, seed to randomise the choice of images, ratio for splitting 80% of samples into training and 20% into validation or test; for the purpose of this classifier, it’s named test. Group prefix denotes to the length of the group and move causes the file as a substitute of copying.

5.2 Directory Walkthrough

```
""" Installs dependencies if some are missing. """
import pkg_resources

def install_dependencies():
    # https://stackoverflow.com/questions/44210656/how-to-check-if-a-module-is-installed-in-python-and-if-not-install-it-within
    required_dependencies = {'torch', 'torchvision', 'torchaudio', 'numpy', 'pandas' }
    installed_dependencies = { pkg.key for pkg in pkg_resources.working_set }

    missing = required_dependencies - installed_dependencies

    if missing:
        print(f"Dependencies MISSING...! Installing.. {required_dependencies} ...\n")
        python = sys.executable
        subprocess.check_call([python, '-m', 'pip', 'install', *missing])
    if not missing:
        print("No missing dependencies, proceeding as normal.")

install_dependencies()

No missing dependencies, proceeding as normal.
```

Fig. 17. 1st cell representing required dependencies to be installed if missing.

```
""" Importing Torch library. """

import torch
from torch import nn

""" Note: this notebook requires torch >= 1.10.0 """
print(" Torch detected ... available for use ... version ... {}".format(torch.__version__))

Torch detected ... available for use ... version ... 2.0.0

""" Setup device-agnostic code, if CUDA is detected.. """
device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

from pathlib import Path
data_path = Path("C:/Users/jasja/_haemorrhage_project/_datasets_/")
image_path = data_path / "sml_labels/head_ct"

if image_path.is_dir():
    print("{} -> directory exists.".format(image_path))

C:/Users/jasja/_haemorrhage_project/_datasets_/sml_labels/head_ct -> directory exists.
```

Fig. 18. Cell representing version and directory checkers.

```
""" Walks through dir_path (database) as a str returning its contents. """
import os
def walk_through_directory(dir_path: str) -> None:
    for dirpath, dirnames, filenames in os.walk(dir_path):
        print(f"There are {len(dirnames)} directories and {len(filenames)} images in '{dirpath}'.")

""" Return 'n' directories and 'n' images in a specified directory. """
walk_through_directory(data_path)

There are 5 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/'.
There are 0 directories and 2 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/images_to_predict'.
There are 1 directories and 8 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg'.
There are 82 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT'.
There are 2 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\049'.
There are 0 directories and 33 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\049\bone'.
There are 0 directories and 42 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\049\brain'.
There are 2 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\050'.
There are 0 directories and 26 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\050\bone'.
There are 0 directories and 38 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\050\brain'.
There are 2 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\051'.
There are 0 directories and 32 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\051\bone'.
There are 0 directories and 51 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\051\brain'.
There are 2 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\052'.
There are 0 directories and 30 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\052\bone'.
There are 0 directories and 39 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\052\brain'.
There are 2 directories and 0 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\053'.
There are 0 directories and 33 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\053\bone'.
There are 0 directories and 43 images in 'C:/Users/jasja/_haemorrhage_project/_datasets_/lrg/Patients_CT\053\brain'.
```

(a)


```
walk_through_directory(image_path)

There are 2 directories and 0 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct'.
There are 2 directories and 0 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\test'.
There are 0 directories and 26 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\test\haemorrhage'.
There are 0 directories and 21 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\test\no_haemorrhage'.
There are 2 directories and 0 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train'.
There are 0 directories and 74 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train\haemorrhage'.
There are 0 directories and 79 images in 'C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train\no_haemorrhage'.

""" Setups train and testing paths of the image_path for all images. """
import numpy as np

train_directory=image_path / "train"
test_directory=image_path / "test"
print("Train dir -> {}\nTest dir -> {}".format(train_directory, test_directory))

Train dir -> C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train
Test dir -> C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\test
```

(b)

Fig. 19. (a) & (b) A walkthrough through the specified directories.

5.3 Image Visualisation (Randomised)

```
import random
import glob
from PIL import Image

"""
Firstly, we set a seed to random images that can be tweaked to suit randomisation better.
image_path_list -> Get all image paths..
random_image_path -> Gets random image path..
img -> Opens the image..
image_class -> Withhelds the class that the image belongs to, normally the database was manually formatted.

Return: all the info above in print format and display the randomised image.
"""

random.seed(41)

image_path_list = list(image_path.glob("*/**/*.png"))
random_image_path = random.choice(image_path_list)
img = Image.open(random_image_path)
image_class = random_image_path.parent.stem

print("Random image path -> {}".format(random_image_path))
print("Image class -> {}".format(image_class))
print("Image height -> {}".format(img.height))
print("Image width -> {}".format(img.width))
img

Random image path -> C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train\haemorrhage\076.png
Image class -> haemorrhage
Image height -> 378
Image width -> 280
```



Fig. 20. Randomised image output for visualisation testing.

5.4 Data Transformation & Loader

```
import torch
import torchvision
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
print(" Torchvision detected ... available for use ... version ... {}".format(torchvision.__version__))

Torchvision detected ... available for use ... version ... 0.15.0

"""
    Write a transform for an image...
    Resize the images to 64 by 64 for less computation time...
    Turn the image into three-dimensional array i.e., Tensor.
    -> This also converts all pixel values from 0 to 255 to be between 0. and 1.0.
"""
data_transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor()
])
```

Fig. 21. Data transform into Tensor & Torch-vision version checker.

```
"""
    Plots a series of random images from image_paths using matplotlib.
    Will open `n` image_paths, transform these with transform and plot them side by side.

    `permute` -> changes the shape of image to suit matplotlib
    to... [H, W, C] or [height, width and colour_channel]
"""
import matplotlib.pyplot as plt
seed: int=46

def plot_transformed_images(image_paths, transform, n=3, seed=seed):
    random.seed(seed)

    random_image_paths=random.sample(image_paths, k=n)
    for image_path in random_image_paths:
        with Image.open(image_path) as f:
            fig, ax = plt.subplots(1, 2)
            ax[0].imshow(f)
            ax[0].set_title("Original \nSize: {}".format(f.size))
            ax[0].axis("off")

            transformed_image = transform(f).permute(1, 2, 0)
            ax[1].imshow(transformed_image)
            ax[1].set_title("Transformed \nSize: {}".format(transformed_image.shape))

            fig.suptitle("Class: {}".format(image_path.parent.stem), fontsize=16)

plot_transformed_images(image_path_list,
                        transform=data_transform,
                        n=3)
```

Fig. 22. Plotting transformed images into Tensor.

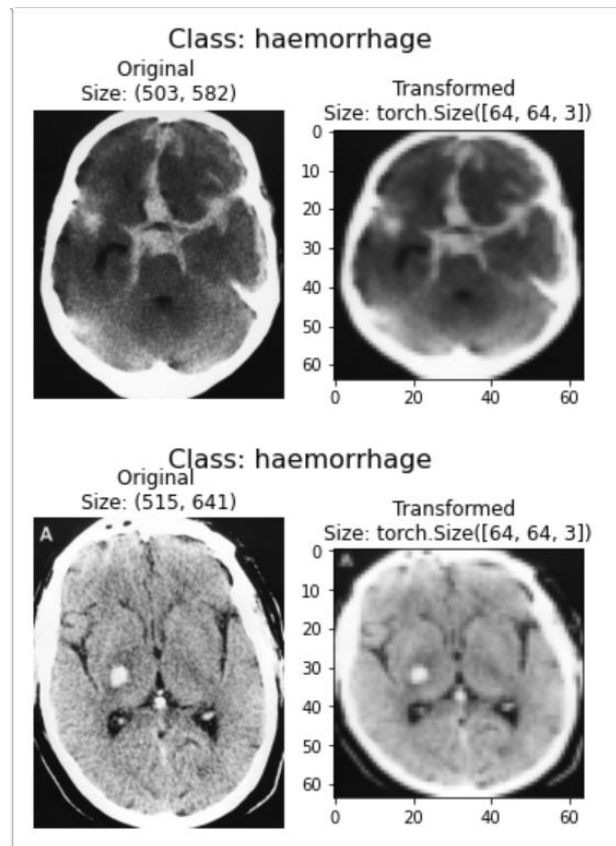


Fig. 23. A comparison between original input and transformed output of Haemorrhagic scans.

```

"""
    Use ImageFolder to create Dataset(s)
    Target folder of images -> root
    Transforms to perform on data..
    Transforms to perform on labels, but this is not necessary for these examples.
"""
    NUM_OF_WORKERS -> equals to the amount of CPUs present on a singular machine.
"""

from torchvision import datasets
train_data = datasets.ImageFolder(root=train_directory,
                                  transform=data_transform)
test_data = datasets.ImageFolder(root=test_directory,
                                  transform=data_transform)

print("Train data:\n{}\nTest data:\n{}".format(train_data, test_data))

print(f"Train Data Length: {len(train_data)}\nTest Data Length: {len(test_data)}")

import os
from torch.utils.data import DataLoader

BATCH_SIZE=32
NUM_OF_WORKERS=os.cpu_count()
print(f"\nCreating DataLoader's with batch size {BATCH_SIZE} and {NUM_OF_WORKERS} workers.")

train_dataloader = DataLoader(train_data,
                              batch_size=BATCH_SIZE,
                              shuffle=True,
                              num_workers=NUM_OF_WORKERS)

test_dataloader = DataLoader(test_data,
                             batch_size=BATCH_SIZE,
                             shuffle=False,
                             num_workers=NUM_OF_WORKERS)

train_dataloader, test_dataloader
print(f"Length of train dataloader: {len(train_dataloader)} batches of {BATCH_SIZE}")
print(f"Length of test dataloader: {len(test_dataloader)} batches of {BATCH_SIZE}")

```

```

Train data:
Dataset ImageFolder
  Number of datapoints: 153
  Root location: C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\train
  StandardTransform
Transform: Compose(
  Resize(size=(64, 64), interpolation=bilinear, max_size=None, antialias=warn)
  ToTensor()
)
Test data:
Dataset ImageFolder
  Number of datapoints: 47
  Root location: C:\Users\jasja\__haemorrhage_project__\__datasets__\sml_labels\head_ct\test
  StandardTransform
Transform: Compose(
  Resize(size=(64, 64), interpolation=bilinear, max_size=None, antialias=warn)
  ToTensor()
)
Train Data Length: 153
Test Data Length: 47

Creating DataLoader's with batch size 32 and 4 workers.
Length of train dataloader: 5 batches of 32
Length of test dataloader: 5 batches of 32

```

Fig. 24. Transformed images into Image Folder and Data Loaders.

```

"""Get class names as a dictionary i.e., label name with retrospective key."""
class_dict = train_data.class_to_idx
class_dict

{'haemorrhage': 0, 'no_haemorrhage': 1}

```

Fig. 25. Retrospective labels in correspondence to directories.

5.4 Model Classifier

```

"""
  Model architecture inspired from TinyVGG, URL source:
  https://poloclub.github.io/cnn-explainer/
"""
class ConvNet(nn.Module):
    def __init__(self, input_shape: int, hidden_units: int, output_shape: int) -> None:
        super().__init__()
        self.conv_block_1 = nn.Sequential(
            nn.Conv2d(in_channels=input_shape,
                      out_channels=hidden_units,
                      kernel_size=3,
                      stride=1,
                      padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=hidden_units,
                      out_channels=hidden_units,
                      kernel_size=3,
                      stride=1,
                      padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,
                         stride=2)
        )
        self.conv_block_2 = nn.Sequential(
            nn.Conv2d(hidden_units, hidden_units, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(hidden_units, hidden_units, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )

```

```

        self.classifier=nn.Sequential(
            nn.Flatten(), # This is because each layer of the network compresses and changes the shape of the input.
            nn.Linear(in_features=hidden_units*16*16,
                      out_features=output_shape)
        )

    def forward(self, x: torch.Tensor):
        x=self.conv_block_1(x)
        x=self.conv_block_2(x)
        x=self.classifier(x)
        return x

model_0 = ConvNet(input_shape=3,
                  hidden_units=10,
                  output_shape=len(train_data.classes)).to(device)

model_0

ConvNet(
  (conv_block_1): Sequential(
    (0): Conv2d(3, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv_block_2): Sequential(
    (0): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=2560, out_features=2, bias=True)
  )
)

```

Fig. 26. Classifier model inspired by TinyVGG and model's output structure.

```

In [35]: """This is a test cell-solution to clarify that classifier indeed is working as intended."""
img_batch, label_batch = next(iter(train_data_loader))

img_single, label_single = img_batch[0].unsqueeze(dim=0), label_batch[0]
print(f"Single image shape: {img_single.shape}\n")

model_0.eval()
with torch.inference_mode():
    pred = model_0(img_single.to(device))

print(f"Output logits: {pred}\n")
print(f"Output prediction probabilities: {torch.softmax(pred, dim=1)}\n")
print(f"Output prediction label: {torch.argmax(torch.softmax(pred, dim=1), dim=1)}\n")
print(f"Actual label: {label_single}")

Single image shape: torch.Size([1, 3, 64, 64])

Output logits: tensor([[ -0.0306, -0.0175]])

Output prediction probabilities: tensor([[0.4967, 0.5033]])

Output prediction label: tensor([1])

Actual label: 0

```

Fig. 27. A single image prediction test.

```

class_names=train_data.classes
print("Class names for training data directory.. {} \nsize train: {} \nsize test: {}".format(class_names, len(train_data), len(test_data)))

Class names for training data directory.. ['haemorrhage', 'no_haemorrhage']
size train: 153
size test: 47

"""Infrastructure of a model and details of each layer, what goes in and what goes out."""
try:
    import torchinfo
except:
    !pip install torchinfo
    import torchinfo

from torchinfo import summary
summary(model_0, input_size=[1, 3, 64, 64])

```

```

Out[37]: =====
Layer (type:depth-idx)      Output Shape      Param #
=====
ConvNet
├─Sequential: 1-1           [1, 2]            --
│   └─Conv2d: 2-1           [1, 10, 32, 32]   --
│       └─ReLU: 2-2         [1, 10, 64, 64]   280
│           └─Conv2d: 2-3   [1, 10, 64, 64]   --
│               └─ReLU: 2-4 [1, 10, 64, 64]   910
│                   └─MaxPool2d: 2-5 [1, 10, 32, 32]   --
│                       └─Sequential: 1-2 [1, 10, 16, 16]   --
│                           └─Conv2d: 2-6 [1, 10, 32, 32]   910
│                               └─ReLU: 2-7 [1, 10, 32, 32]   --
│                                   └─Conv2d: 2-8 [1, 10, 32, 32]   910
│                                       └─ReLU: 2-9 [1, 10, 32, 32]   --
│                                           └─MaxPool2d: 2-10 [1, 10, 16, 16]   --
│                                               └─Sequential: 1-3 [1, 2]            --
│                                                   └─Flatten: 2-11 [1, 2560]          --
│                                                       └─Linear: 2-12 [1, 2]            5,122
=====
Total params: 8,132
Trainable params: 8,132
Non-trainable params: 0
Total mult-adds (M): 6.74
=====
Input size (MB): 0.05
Forward/backward pass size (MB): 0.82
Params size (MB): 0.03
Estimated Total Size (MB): 0.90
=====

```

Fig. 28. Summary of the architecture, process within convolutional layers and more.

5.4.1 Training & Testing Steps

```

"""Takes a model, DataLoader, loss function and optimiser and conducts training model on the DataLoader."""
def train_step(model: torch.nn.Module,
               dataloader: torch.utils.data.DataLoader,
               loss_fn: torch.nn.Module,
               optimizer: torch.optim.Optimizer):
    # Puts the model into training mode;
    model.train()

    # Default values for train loss and train accuracy scores;
    train_loss, train_acc = 0, 0

    # Loops through data loader and data batches;
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        y_pred = model(X)

        loss = loss_fn(y_pred, y)
        train_loss += loss.item()

        optimizer.zero_grad()

        loss.backward()

        optimizer.step()

    # Calculate and sum accuracy metrics across all batches;
    y_pred_class = torch.argmax(torch.softmax(y_pred, dim=1), dim=1)
    train_acc = (y_pred_class == y).sum().item() / len(y_pred)

    # Adjust metrics to get a mean loss and accuracy per single batch;
    train_loss = train_loss / len(dataloader)
    train_acc = train_acc / len(dataloader)
    return train_loss, train_acc

```

Fig. 29. Training step with plotting output showing loss and accuracy per batch.

```

"""Difference vs. train_step is, there's no optimizer thus won't perform gradient descent."""
def test_step(model: torch.nn.Module,
              dataloader: torch.utils.data.DataLoader,
              loss_fn: torch.nn.Module):
    # Put model in evaluation mode
    model.eval()

    # Setup test loss and test accuracy values
    test_loss, test_acc = 0, 0

    # Turn on inference context manager
    with torch.inference_mode():
        # Loop through DataLoader batches
        for batch, (X, y) in enumerate(dataloader):
            # Send data to target device
            X, y = X.to(device), y.to(device)

            test_pred_logits = model(X)

            loss = loss_fn(test_pred_logits, y)
            test_loss += loss.item()

            # Calculate and accumulate accuracy
            test_pred_labels = test_pred_logits.argmax(dim=1)
            test_acc += ((test_pred_labels == y).sum().item() / len(test_pred_labels))

    # Adjust metrics to get average loss and accuracy per single batch
    test_loss = test_loss / len(dataloader)
    test_acc = test_acc / len(dataloader)
    return test_loss, test_acc

```

Fig. 30. Test step with plotting output showing loss and accuracy per batch.

```

"""This is a progress bar that adapts to appropriate file extension or platform."""
from tqdm.auto import tqdm

# Taking all necessary parameters
def train(model: torch.nn.Module,
          train_dataloader: torch.utils.data.DataLoader,
          test_dataloader: torch.utils.data.DataLoader,
          optimizer: torch.optim.Optimizer,
          loss_fn: torch.nn.Module = nn.CrossEntropyLoss,
          epochs: int = 10):
    # Creates an empty dictionaries to return metrics
    results = {"train_loss": [],
              "train_acc": [],
              "test_loss": [],
              "test_acc": []}

    for epoch in tqdm(range(epochs)):
        train_loss, train_acc = train_step(model=model,
                                           dataloader=train_dataloader,
                                           loss_fn=loss_fn,
                                           optimizer=optimizer)
        test_loss, test_acc = test_step(model=model,
                                       dataloader=test_dataloader,
                                       loss_fn=loss_fn)

    print(
        f"Epoch: {epoch+1} | "
        f"train_loss: {train_loss:.4f} | "
        f"train_acc: {train_acc:.4f} | "
        f"test_loss: {test_loss:.4f} | "
        f"test_acc: {test_acc:.4f}"
    )

    # Update results dictionary
    results["train_loss"].append(train_loss)
    results["train_acc"].append(train_acc)
    results["test_loss"].append(test_loss)
    results["test_acc"].append(test_acc)

    # Return the filled results at the end of the epochs
    return results

```

Fig. 31. Training function to train the model with empty dictionaries for metrics return.

```

y_preds = []

targets = [x[1] for x in test_data.samples]

model_convnet.eval()
with torch.inference_mode():
    for X, y in tqdm(test_dataloader, desc="Making predictions "):
        X, y = X.to(device), y.to(device)
        y_logit = model_convnet(X)
        y_pred = torch.softmax(y_logit, dim=1).argmax(dim=1)
        y_preds.append(y_pred.cpu())

y_pred_tensor = torch.cat(y_preds)

```


Making predictions : 100%  42/42 [01:03<00:00, 1.52it/s]

Fig. 32. Prediction for a batch of images for use in correlation matrix.

5.5 Training & Plotting Metrics

```

seed: int = 38
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)

# No. of epochs the training phase will undergo
NUM_EPOCHS: int=10

# Recreating an instance of the convolutional neural network
model_0 = ConvNet(input_shape=3,
                  hidden_units=10,
                  output_shape=len(train_data.classes)).to(device)

# Setup loss function and optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(params=model_0.parameters(), lr=0.001)

#Start the timer
from timeit import default_timer as timer
start_time = timer()

# Train model_0
model_0_results = train(model=model_0,
                       train_dataloader=train_dataloader,
                       test_dataloader=test_dataloader,
                       optimizer=optimizer,
                       loss_fn=loss_fn,
                       epochs=NUM_EPOCHS)

# End the timer and print out how long it took
end_time = timer()
print(f"Total training time: {end_time-start_time:.3f} seconds")

```

Fig. 32. Instance of the convolutional neural network, loss function, optimiser, and model training.

Adam optimiser is an efficient stochastic optimisation only requiring little memory allocation with first-order gradients, computing singular adaptive learning rates for different parameters from the approximates of first and second moments of the gradients. Adam is a derived acronym from “adaptive moment estimation” (Kingma, D. and Lei Ba, J. (2017. 51).


```

from typing import Tuple, Dict, List

def plot_loss_curves(results: Dict[str, List[float]]):
    """Plots training curves of a results dictionary.

    Args:
        results (dict): dictionary containing list of values, e.g.
            {"train_loss": [...],
             "train_acc": [...],
             "test_loss": [...],
             "test_acc": [...]}
    """

    # Get the Loss values of the results dictionary (training and test)
    loss = results['train_loss']
    test_loss = results['test_loss']

    # Get the accuracy values of the results dictionary (training and test)
    accuracy = results['train_acc']
    test_accuracy = results['test_acc']

    # Figure out how many epochs there were
    epochs = range(len(results['train_loss']))

    # Setup a plot
    plt.figure(figsize=(15, 7))

    # Plot Loss
    plt.subplot(1, 2, 1)
    plt.plot(epochs, loss, label='train_loss')
    plt.plot(epochs, test_loss, label='test_loss')
    plt.title('Loss')
    plt.xlabel('Epochs')
    plt.legend()

    # Plot accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs, accuracy, label='train_accuracy')
    plt.plot(epochs, test_accuracy, label='test_accuracy')
    plt.title('Accuracy')
    plt.xlabel('Epochs')
    plt.legend()

    # Returns the graph
    plot_loss_curves(model_0_results)

```

Fig. 32. Plotting the loss and accuracy curves.

5.6 Prediction

```

from PIL import Image

cstm_img_transform=transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((64, 64)),
    transforms.ToTensor()
])

```

Fig. 33. Custom image transform, converts it into PIL image, resizes and transforms to Tensor.

```

predict_path=data_path / "images_to_predict" / "pred_1.jpeg"

import torch.nn.functional as nnf
"""Prediction of the class/label function."""
def predict_and_plot_img(model: torch.nn.Module,
                        image_path: str,
                        class_names: List[str] = None,
                        transform=None,
                        device: torch.device = device):

    # Randomise a choice of the prediction img
    #tmp=List(predict_path.glob(*/*.jpg))
    #rnd_img_path=random.choice(tmp)

    trg_img=torchvision.io.read_image(str(image_path)).type(torch.float32)

    trg_img=trg_img/255

    if transform:
        trg_img=transform(trg_img)

    model.to(device)

    model.eval()
    with torch.inference_mode():
        trg_img=trg_img.unsqueeze(dim=0)

        trg_img_pred=model(trg_img.to(device))

        #trg_img_pred_probs=torch.softmax(trg_img_pred, dim=1) # multi-class classifier

        #https://stackoverflow.com/questions/60182984/how-to-get-the-predict-probability
        trg_img_pred_probs=nnf.softmax(trg_img_pred, dim=1) # binary classifier

    trg_img_pred_label=torch.argmax(trg_img_pred_probs, dim=1)

    plt.imshow(trg_img.squeeze().permute(1, 2, 0))
    if class_names:
        title=f"Prediction: {class_names[trg_img_pred_label.cpu()]} | Probability: {trg_img_pred_probs.max().cpu():.3f}"
    else:
        title=f"Prediction: {trg_img_pred_label} | Probability: {trg_img_pred_probs.max().cpu():.3f}"
    plt.title(title)
    plt.axis(False)

"""Prediction"""
predict_and_plot_img(model=model_convnet,
                    image_path=predict_path,
                    class_names=class_names,
                    transform=cstm_img_transform,
                    device=device)

```

Fig. 34. Prediction function and function's call-back.

Prediction: haemorrhage | Probability: 1.000

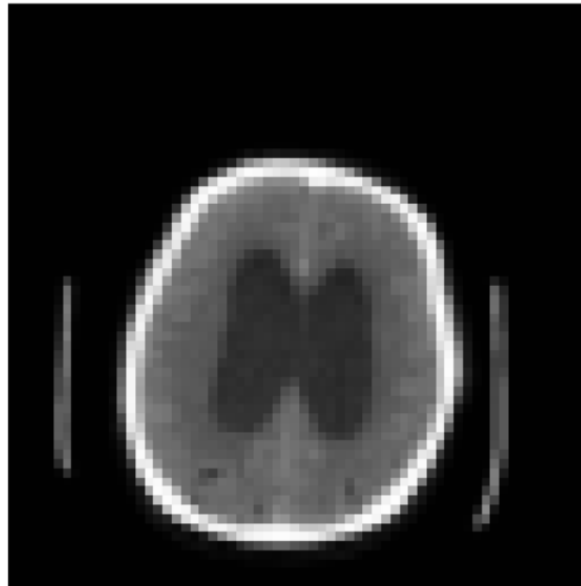


Fig. 35. Predicted random image not located within nor train or test directories.

5.7 Correlation Matrix

```
#https://stackoverflow.com/questions/19233771/sklearn-plot-confusion-matrix-with-labels

from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

print(classification_report(test_data.targets, y_pred_tensor, target_names=class_names))
cm=confusion_matrix(test_data.targets, y_pred_tensor)

import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax); #annot=True to annotate cells, fmt='g' to disable scientific notation

# Labels, title and ticks
ax.set_xlabel('Predicted labels'); ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['haemorrhage', 'normal']); ax.yaxis.set_ticklabels(['normal', 'haemorrhage']);

#ConfusionMatrixDisplay(cm).plot()
```

Fig. 36. Correlation Matrix cell before its' execution.

Chapter 6

6. Results & Discussion

This section presents the findings of the work and discusses them in the context of original aims & objectives, as well as requirements specification. For research-oriented projects, this chapter will present the data in an appropriate format (tables, charts, visualisations) and discuss what the data shows.

As a test the number of epochs which is type hinted as an integer is initialised to 10, **Fig. 36** brings to the view the process. Trained instance is nowhere near unvarying to solidify clinician's trust in the model. Data is underfitted but also overfitted, because of small dataset the network could not extract enough regions of interest to successfully predict as seen in **Fig. 40** and no sufficient epochs were initialised for reason to test the classifier.

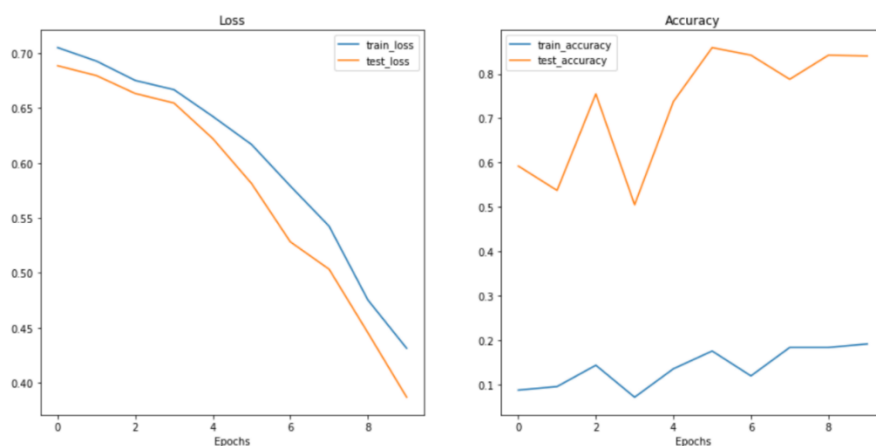


Fig. 37. Model trained on ~200 samples on 10 epoch.

As a result of poor trained model, epoch was increased systematically up to 100 as shown in **Fig. 37** and to 200 as shown in **Fig. 38**. Model was not overfitted nor underfitted. The sweet spot for manual early stopping was found to be at 80 epochs as loss of both trained and tested data levelled out with minimal loss on both sides.

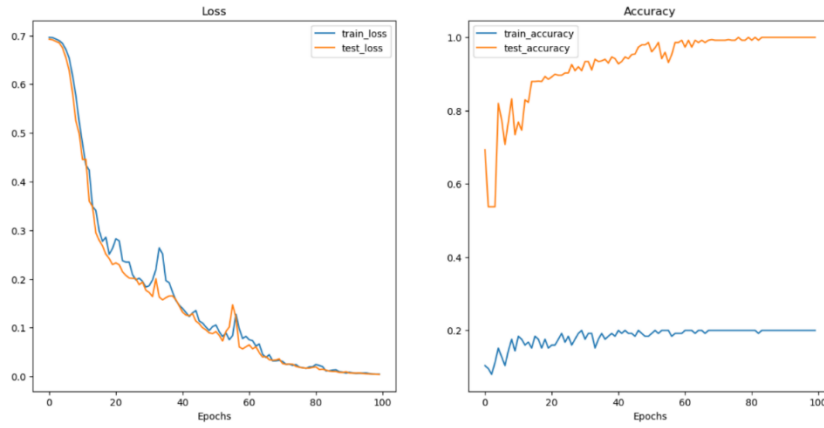


Fig. 38. Model trained on ~200 samples on 100 epoch.

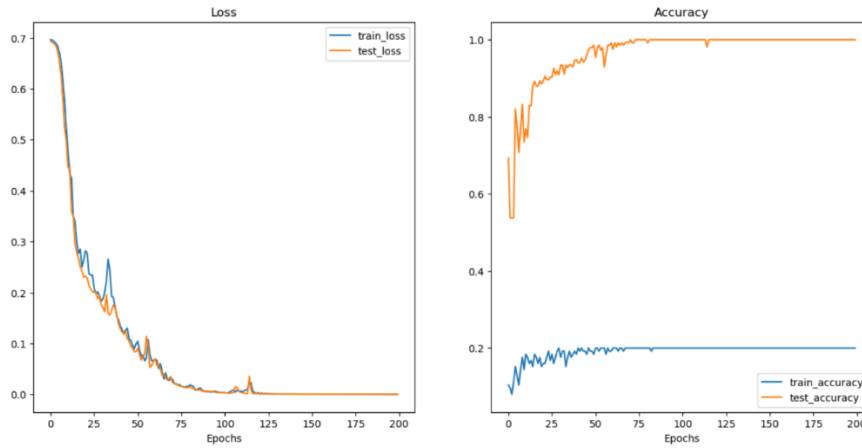


Fig. 39. Model trained on ~200 samples on 200 epoch.

However, a problem arose when predicting unseen data from the “images_to_predict” directory. A haemorrhagic scan of a patient digitised from copyright-free source foresaw a wrong prediction to a wrong class as seen in **Fig. 39**. Network had not enough data to extract features from out of the ordinary sectors with anomalies and its’ confidence in the probability was also immensely misjudged, to reflect from previous chapter a build-up of blood fluid can proceed to occupy areas of patient’s brain or skull. Next step was to format and transform larger size of a dataset with intrinsically more slices of both skull and brain to extract those regions of interest as seen in **Fig. 40**.

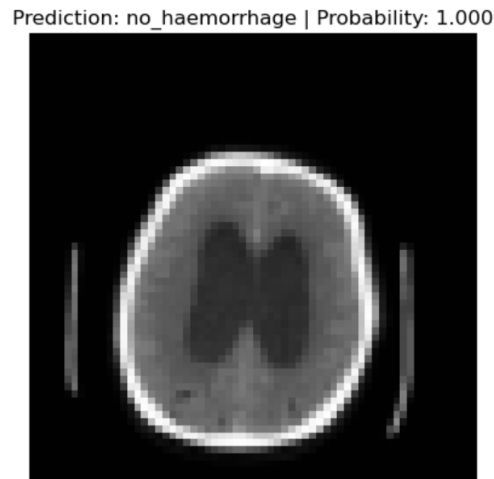


Fig. 40. A wrong-class prediction of a haemorrhagic stroke outlined by darker centre build of fluid of the brain.

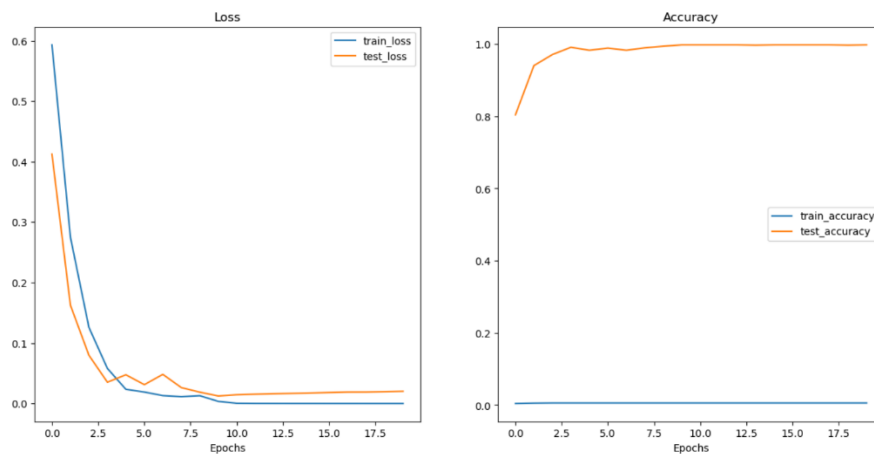


Fig. 41. Model trained on ~7000 samples on 20 epoch.

Improvement was seen in predicting the class on unseen data when trained using ~7000 samples on 20 epoch, where the optimum place for an early stop. There's an upshot of balanced overfitting and underfitting in comparison to **Fig. 38** or **Fig. 41** where twain graphs seen indiscrepancies only justifying distrusting the model for a clinical setting. Accuracy, on the other hand is good as the model is class-balanced, where each class dataset has same number of samples as its adversary.

When predicting two scans from haemorrhagic and non-haemorrhagic labels the prediction was right with high probability as seen in **Fig. 42** and **Fig. 43**.

Prediction: haemorrhage | Probability: 1.000

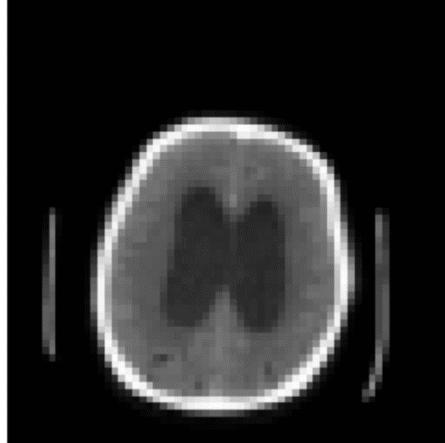


Fig. 42. A right-class prediction of a haemorrhagic stroke when trained on large dataset with high probability.

Prediction: no_haemorrhage | Probability: 1.000

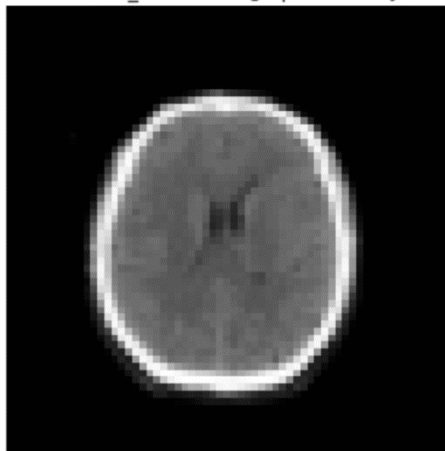


Fig. 43. A right-class prediction of no haemorrhagic stroke when trained on large dataset with high probability.

	precision	recall	f1-score	support
haemorrhage	1.00	1.00	1.00	522
no_haemorrhage	1.00	1.00	1.00	821
accuracy			1.00	1343
macro avg	1.00	1.00	1.00	1343
weighted avg	1.00	1.00	1.00	1343

Fig. 44. A classification report returning Precision, Recall and F1 Score over test dataset.

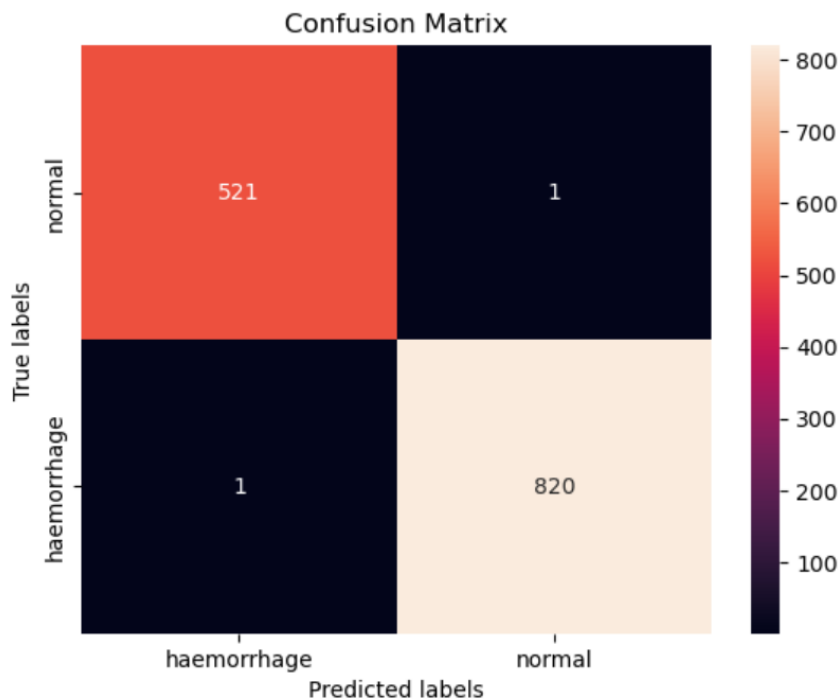


Fig. 45. A confusion matrix plotted using Seaborn.

As it is necessary to display all possible metrics to evaluate the model roots, library imported from “sklearn” allows for such metrics. Precision is the ratio between the true positives and all the positives, to contextualise, it is a measure of patients that were correctly identified as having an intracranial haemorrhagic stroke. **Fig. 45**, has a high precision score of 1.00 or 100% which affluences that model is predicting that patient has a haemorrhagic stroke but should not be relied on.

Recall denotes how many patients truly had a haemorrhagic stroke. It scored similarly well to precision (1.00 / 100%) which could mean that although the patient

suffered a haemorrhage, treatment should've been conducted solely due to the model predicting with high recall.

Last examination was counselled on large dataset but with less epoch value, **Fig. 46** and **Fig. 47** show a good trade-off between overfit and underfit data approximately at 3.0 – 4.0 epochs. Program required to have an ability of a system to operate in fair and unbiased manner, witnessing figures made known is successful at that. There's little bias but is necessary to justify the model as it needs to be right or reasonable with moderate to understandable transparency.

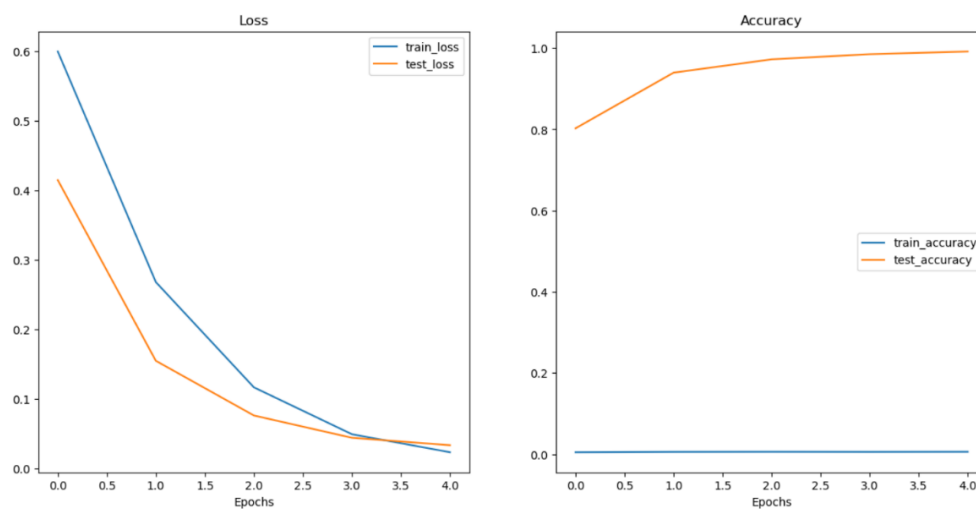


Fig. 46. A balance between overfit and underfit when trained on 5 epochs & ~7000 samples.

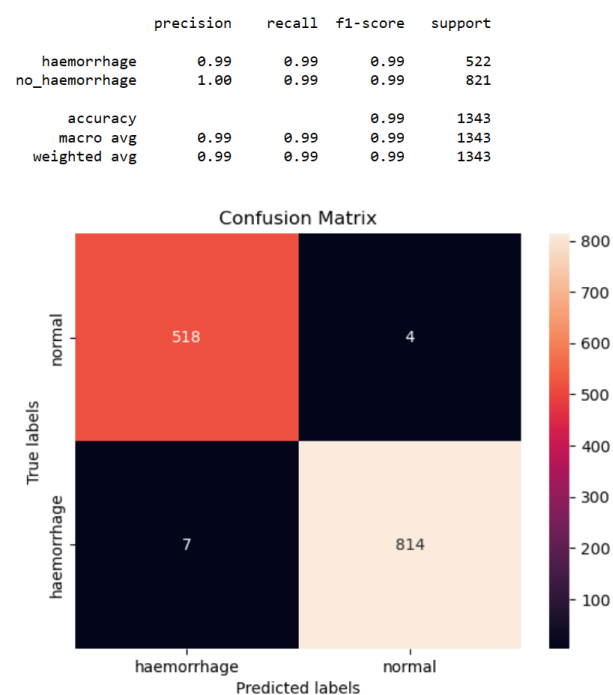


Fig. 47. A correlation matrix and classifications' report on 5 epochs & ~7000 samples.

When predicting on the trained model with large dataset on an image with visible haemorrhage, there's an extensive build of blood fluid levels, prediction is haemorrhage with probability at 0.617 as shown in **Fig. 48**.

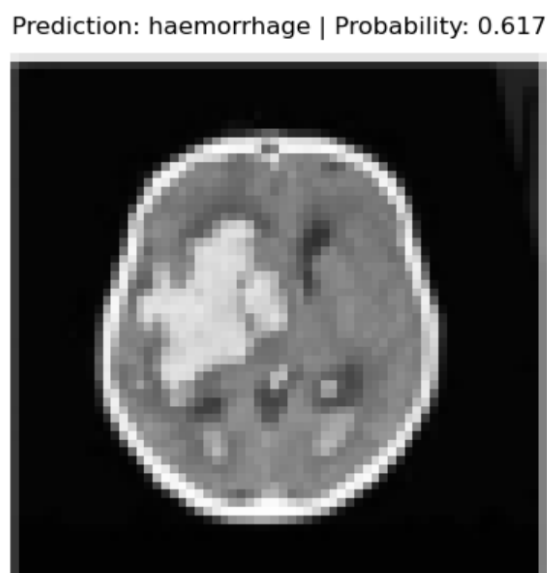


Fig. 48. Correct prediction of a haemorrhage scan "h_4.jpg" with probability 0.617.

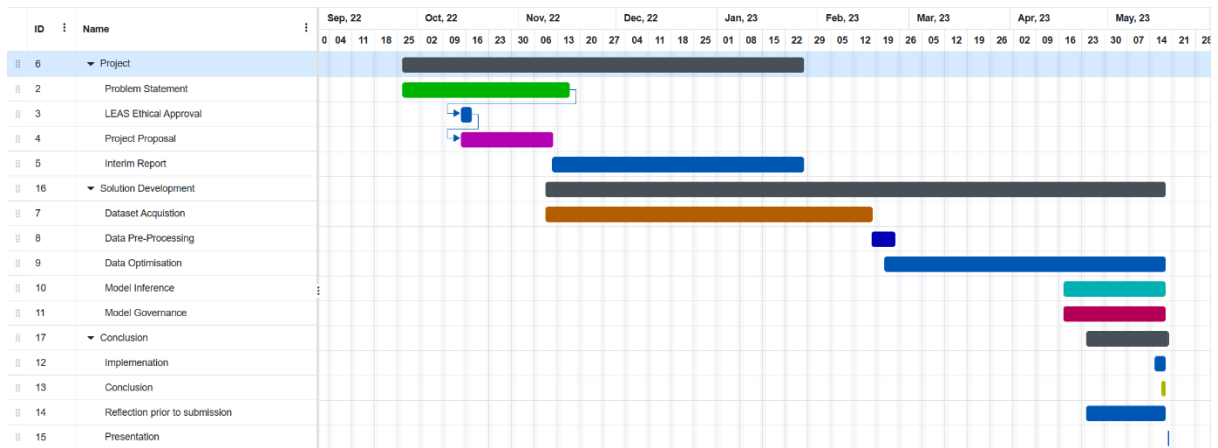


Fig. 49. Updated GANTT chart summarising entire project span.

Chapter 7

7. Conclusion

To conclude, the aim of the solution was achieved, model correctly predicted images and model was able to detect and highlight regions of interest for the purpose of every CT by applying a deep learning model, convolutional neural network, for the purpose of diagnosis and binary classification. Model achieved on average 98% probability of correct prediction with few exceptions when trained on ~7000 data samples of CT slices and 20 epochs. Trained classifier achieved 1.00 in recall, precision and F1 score with worthy mention to accuracy on both testing and training data but when subjugated with 5 epochs, classifier achieved .98 in recall and .99 elsewhere. Withal when trained on less epoch's metrics were more reasonable. However, there's an anomaly of all scores being so high which can disapprove clinician's trust towards the model however, after predicting 6 images 3 with haemorrhage and 3 without haemorrhage the model was able to classify all correctly at pixelated resolution: 64 by 64. Large dataset abiding of multiple slices of haemorrhage CT scans standardised number of features and singularities for the classifier.

The objective 4 was accumulated as per specification, as in medical setting it is vital to have as many metrics as there are for building trust, evaluation, and performance analysis but also reliability in prediction.

Epoch: 1		train_loss: 0.6006		train_acc: 0.0043		test_loss: 0.4193		test_acc: 0.7989
Epoch: 2		train_loss: 0.2782		train_acc: 0.0055		test_loss: 0.1712		test_acc: 0.9411
Epoch: 3		train_loss: 0.1254		train_acc: 0.0057		test_loss: 0.1045		test_acc: 0.9635
Epoch: 4		train_loss: 0.0648		train_acc: 0.0060		test_loss: 0.0524		test_acc: 0.9821
Epoch: 5		train_loss: 0.0296		train_acc: 0.0060		test_loss: 0.0465		test_acc: 0.9851
Epoch: 6		train_loss: 0.0124		train_acc: 0.0060		test_loss: 0.0411		test_acc: 0.9888
Epoch: 7		train_loss: 0.0158		train_acc: 0.0060		test_loss: 0.0233		test_acc: 0.9933
Epoch: 8		train_loss: 0.0126		train_acc: 0.0060		test_loss: 0.0359		test_acc: 0.9888
Epoch: 9		train_loss: 0.0082		train_acc: 0.0060		test_loss: 0.0170		test_acc: 0.9985
Epoch: 10		train_loss: 0.0008		train_acc: 0.0060		test_loss: 0.0173		test_acc: 0.9978
Epoch: 11		train_loss: 0.0004		train_acc: 0.0060		test_loss: 0.0178		test_acc: 0.9985
Epoch: 12		train_loss: 0.0003		train_acc: 0.0060		test_loss: 0.0191		test_acc: 0.9978
Epoch: 13		train_loss: 0.0002		train_acc: 0.0060		test_loss: 0.0192		test_acc: 0.9978
Epoch: 14		train_loss: 0.0002		train_acc: 0.0060		test_loss: 0.0201		test_acc: 0.9970
Epoch: 15		train_loss: 0.0002		train_acc: 0.0060		test_loss: 0.0202		test_acc: 0.9970
Epoch: 16		train_loss: 0.0001		train_acc: 0.0060		test_loss: 0.0210		test_acc: 0.9978
Epoch: 17		train_loss: 0.0001		train_acc: 0.0060		test_loss: 0.0217		test_acc: 0.9978
Epoch: 18		train_loss: 0.0001		train_acc: 0.0060		test_loss: 0.0219		test_acc: 0.9978
Epoch: 19		train_loss: 0.0001		train_acc: 0.0060		test_loss: 0.0224		test_acc: 0.9978
Epoch: 20		train_loss: 0.0001		train_acc: 0.0060		test_loss: 0.0236		test_acc: 0.9978
Total training time: 1341.580 seconds								

Fig. 50. Training process with accountable epochs, losses, and accuracies on 20 epochs.

Fig. 50 presents a moment where losses on testing increase exponentially when it should not. Using early-stopping, network seems to train continuously better, the error or loss decreases but during specific epoch it rises again. Perchance to fight overfitting is to reduce the number of dimensions of the parameter space or reduce dimensions size. This could in fact oppose the gradual increase of test error where it shouldn't (Lutz Prechelt 1998. 53).

Primary objective referred to collecting ~1000 images for the purpose of diagnosis however, after conducting an observation on the smaller dataset it was found that the classifier did not rightfully predict haemorrhage stroke rather thought there was no stroke, but it was present. It was only necessary to collect a wider database with more slices on each level to extract those features carefully and precisely. None of the data came to be corrupt, nor damaged, nor noise-impacted and model can be appropriately saved into a torch compatible file extension for further use.

Images were converged with good pixel intensities and resolution notably in the large dataset with 512 by 512 on width and height, this allowed the model classifier to equally transform all images into smaller dimensions and extract those regions of interest.

Prediction model struggles to transform images of contrastive image formats, e.g., jpeg which is like jpg (if not almost the same) it crashes upon testing with an error of wrong value of input channels, which is out of ordinary because it is a binary image. As a future work, it'd be advisory to adapt the classifier for the purpose of this solution to support not only all formats; to be able to transform them and clinical pictures as seen in (Radiological Society of North America. 2023. 55) with file extension “.dcm” that is used in clinics.

References

- Aad, Georges et al. (2012). ‘Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC’. In: *Physics Letters B* 716.1, pp. 1–29 (cit. on p. 2).
- Chatrchyan, Serguei et al. (2012). ‘Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC’. In: *Physics Letters B* 716.1, pp. 30–61 (cit. on p. 2).
- [3] Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. [online] doi:<https://doi.org/10.1109/comitcon.2019.8862451>.
- [4] Choy, G., Khalilzadeh, O., Michalski, M., Do, S., Samir, A.E., Pinykh, O.S., Geis, J.R., Pandharipande, P.V., Brink, J.A. and Dreyer, K.J. (2018). Current Applications and Future Impact of Machine Learning in Radiology. *Radiology*, [online] 288(2), pp.318–328. Doi:<https://doi.org/10.1148/radiol.2018171820>.
- [5] Salim D. (2021) Supervised Learning – A Systematic Literature Review. [online] <https://osf.io/tysr4/download>.
- [6] Tougui, I., Jilbab, A. and Mhamdi, J.E. (2021). Impact of the Choice of Cross-Validation Techniques on the Results of Machine Learning-Based Diagnostic Applications. *Healthcare Informatics Research*, [online] 27(3), pp.189–199. doi:<https://doi.org/10.4258/hir.2021.27.3.189>.
- [7] Salim D. (2021). *Unsupervised Learning - A Systematic Literature Review*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/357380639_Unsupervised_Learning_-_A_Systematic_Literature_Review.
- [8] Rodriguez, M.Z., Comin, C.H., Casanova, D., Bruno, O.M., Amancio, D.R., Costa, L. da F. and Rodrigues, F.A. (2019). Clustering algorithms: A comparative approach. *PLOS ONE*, [online] 14(1), p.e0210236. doi:<https://doi.org/10.1371/journal.pone.0210236>.

- [9] Salman, S. and Liu, X. (2019). *Overfitting Mechanism and Avoidance in Deep Neural Networks*. [online] Available at: <https://arxiv.org/pdf/1901.06566.pdf>.
- [10] van Timmeren, J.E., Cester, D., Tanadini-Lang, S., Alkadhi, H. and Baessler, B. (2020). Radiomics in medical imaging—‘how-to’ guide and critical reflection. *Insights into Imaging*, [online] 11(1). doi:<https://doi.org/10.1186/s13244-020-00887-2>.
- [11] Chen, Q., Xia, T., Zhang, M., Xia, N., Liu, J. and Yang, Y. (2021). Radiomics in Stroke Neuroimaging: Techniques, Applications, and Challenges. *Aging and disease*, [online] 12(1), p.143. doi:<https://doi.org/10.14336/ad.2020.0421>.
- [12] Lambin P, Leijenaar RTH, Deist TM, Peerlings J, de Jong EEC, van Timmeren J, et al. (2017). Radiomics: the bridge between medical imaging and personalized medicine. *Nat Rev Clin Oncol*, 14:749-762.
- [13] Zwanenburg A, Leger S, Vallieres M, Lo S Image biomarker standardisation initiative. *arXiv preprint arXiv:1612.07003*.
- [14] Xie, G., Li, T., Ren, Y., Wang, D., Tang, W., Li, J. and Li, K. (2022). Radiomics-based infarct features on CT predict hemorrhagic transformation in patients with acute ischemic stroke. *Frontiers in Neuroscience*, [online] 16. doi:<https://doi.org/10.3389/fnins.2022.1002717>.
- [15] Santosh Kumar, G.U., Rajini Kanth, T.V., Raju, S.V. and Malyala, S. (2021). Advanced Analysis of Cardiac Image Processing Using Hybrid Approach. *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*. [online] doi:<https://doi.org/10.1109/icaect49130.2021.9392390>.
- [16] Sharma, N., Ray, A., Shukla, K., Sharma, S., Pradhan, S., Srivastva, A. and Aggarwal, L. (2010). Automated medical image segmentation techniques. *Journal of Medical Physics*, [online] 35(1), p.3. doi:<https://doi.org/10.4103/0971-6203.58777>.
- [17] Vaz, J.M. and Balaji, S. (2021). Convolutional neural networks (CNNs): concepts and applications in pharmacogenomics. *Molecular Diversity*, [online] 25(3), pp.1569–1584. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8342355/> [Accessed 23 Mar. 2023].

- [18] Hijazi, S., Kumar, R., & Rowen, C. (2015). Using convolutional neural networks for image recognition. Cadence Design Systems Inc.: San Jose, CA, USA.
- [19] Anjani, I.A., Pratiwi, Y.R. and Norfa Bagas Nurhuda, S. (2021). Implementation of Deep Learning Using Convolutional Neural Network Algorithm for Classification Rose Flower. *Journal of Physics: Conference Series*, [online] 1842(1), p.012002. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1842/1/012002/meta>.
- [20] Sandeep Bhuiya (2020). *Disadvantages of CNN models*. [online] OpenGenus IQ: Computing Expertise & Legacy. Available at: <https://iq.opengenus.org/disadvantages-of-cnn/>.
- [21] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, [online] 8(1). doi:10.1186/s40537-021-00444-8.
- [22] Nichols, J.A., Herbert Chan, H.W. and Baker, M.A.B. (2018). Machine learning: applications of artificial intelligence to imaging and diagnosis. *Biophysical Reviews*, [online] 11(1), pp.111–118. doi:<https://doi.org/10.1007/s12551-018-0449-9>.
- [23] Silver D., et al. Mastering the Game of Go Without Human Knowledge. *Nature*. 2017; 550(7676):354-359
- [24] Kalose, A., Kaya, K. and Kim, A. (n.d.). *Optimal Battle Strategy in Pokémon using Reinforcement Learning*. [online] Available at: <https://web.stanford.edu/class/aa228/reports/2018/final151.pdf>.
- [25] Gnanasekaran, A., Faba, J. and An, J. (n.d.). *Reinforcement Learning in Pacman*. [online] Available at: <http://cs229.stanford.edu/proj2017/final-reports/5241109.pdf>.
- [26] Anguita, D., Ghio, A., Greco, N., Oneto, L. and Ridella, S. (2010). Model selection for support vector machines: Advantages and disadvantages of the Machine Learning Theory. *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [online] doi:<https://doi.org/10.1109/ijcnn.2010.5596450>.

- [27] Aggarwal, P., Vig, R., Bhadoria, S. and Dethle, C.G., 2011. Role of segmentation in medical imaging: A comparative study. *International Journal of Computer Applications*, 29(1), pp.54-61.
- [28] Smith-Bindman, R., 2010. Is computed tomography safe. *N Engl J Med*, 363(1), pp.1-4.
- [29] Kuriakose, D. and Xiao, Z.-C. (2020). Pathophysiology and Treatment of Stroke: Present Status and Future Perspectives. *International Journal of Molecular Sciences*, [online] 21(20), pp.7609–7609. doi:<https://doi.org/10.3390/ijms21207609>.
- [30] Collaborators G.S. Global, regional, and national burden of stroke, 1990-2016: A systematic analysis for the Global Burden of Disease Study 2016. *Lancet Neurol*. 2019;18:439–458.
- [31] Chen J.C. Geographic determinants of stroke mortality: Role of ambient air pollution. *Stroke*. 2010;41:839–841. doi: 10.1161/STROKEAHA.110.578476.
- [32] Yamashita, R., Nishio, M., Richard and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights Into Imaging*, [online] 9(4), pp.611–629. doi:<https://doi.org/10.1007/s13244-018-0639-9>.
- [33] Abien, F. and Agarap (2019). *Deep Learning using Rectified Linear Units (ReLU)*. [online] Available at: <https://arxiv.org/pdf/1803.08375.pdf>.
- [34] Rymer, M.M. (2011). Hemorrhagic stroke: intracerebral hemorrhage. *Missouri medicine*, [online] 108(1), pp.50–4. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6188453/> [Accessed 4 May 2023].
- [35] Macellari, F., Paciaroni, M., Agnelli, G. and Caso, V. (2014). Neuroimaging in Intracerebral Hemorrhage. *Stroke*, [online] 45(3), pp.903–908. doi:<https://doi.org/10.1161/strokeaha.113.003701>.
- [36] Kim, M., Yun, J., Cho, Y., Shin, K., Jang, R., Bae, H. and Kim, N. (2019). Deep Learning in Medical Imaging. *Neurospine*, [online] 16(4), pp.657–668. doi:<https://doi.org/10.14245/ns.1938396.198>.

- [37] Aggarwal, R., Sounderajah, V., Martin, G., Ting, D.S.W., Karthikesalingam, A., King, D., Ashrafian, H. and Darzi, A. (2021). Diagnostic accuracy of deep learning in medical imaging: a systematic review and meta-analysis. *npj Digital Medicine*, [online] 4(1). doi:<https://doi.org/10.1038/s41746-021-00438-z>.
- [38] Cho, J., Lee, K., Shin, E., Choy, G. and Do, S. (2016). *HOW MUCH DATA IS NEEDED TO TRAIN A MEDICAL IMAGE DEEP LEARNING SYSTEM TO ACHIEVE NECESSARY HIGH ACCURACY?* [online] Available at: <https://arxiv.org/pdf/1511.06348.pdf>.
- [39] Sabottke, C.F. and Spieler, B.M. (2020). The Effect of Image Resolution on Deep Learning in Radiography. *Radiology: Artificial Intelligence*, [online] 2(1), p.e190015. doi:<https://doi.org/10.1148/ryai.2019190015>.
- [40] Erickson, B.J. and Kitamura, F. (2021). Magician's Corner: 9. Performance Metrics for Machine Learning Models. *Radiology: Artificial Intelligence*, [online] 3(3), p.e200126. doi:<https://doi.org/10.1148/ryai.2021200126>.
- [41] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, [online] 8(1). doi:<https://doi.org/10.1186/s40537-021-00444-8>.
- [42] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury Google, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., Devito, Z., Raison Nabla, M., Tejani, A., Chilamkurthy, S., Ai, Q., Steiner, B. and Facebook, L. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. [online] Available at: <https://arxiv.org/pdf/1912.01703.pdf>.
- [43] Hicks, S.A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M.A., Halvorsen, P. and Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. *Scientific Reports*, [online] 12(1). doi:<https://doi.org/10.1038/s41598-022-09954-8>.
- [44] Wilfred Van Casteren (2017). *The Waterfall Model and the Agile Methodologies : A comparison by project characteristics - short*. [online] ResearchGate. Available at:

https://www.researchgate.net/publication/313768860_The_Waterfall_Model_and_the_Agile_Methodologies_A_comparison_by_project_characteristics_-_short.

[45] Samuli Laato, Matti Mäntymäki, Matti Minkkinen and Dennehy, D. (2022). *Integrating Machine Learning With Software Development Lifecycles: Insights From Experts*. [online] ResearchGate. Available at:

https://www.researchgate.net/publication/360318448_Integrating_Machine_Learning_With_Software_Development_Lifecycles_Insights_From_Experts.

[46] Aven, T. (2016). Risk assessment and risk management: Review of recent advances on their foundation. *European Journal of Operational Research*, [online] 253(1), pp.1–13. doi:<https://doi.org/10.1016/j.ejor.2015.12.023>.

[47] Albrecht, J. (2018). GIS Project Management. *Comprehensive Geographic Information Systems*, [online] pp.446–477. doi:<https://doi.org/10.1016/b978-0-12-409548-9.09612-3>.

[48] Miotto, R., Wang, F., Wang, S., Jiang, X. and Dudley, J.T. (2018). Deep learning for healthcare: review, opportunities and challenges. [online] 19(6), pp.1236–1246. doi:<https://doi.org/10.1093/bib/bbx044>.

[49] Janusz Snieg. (2023). Assessment Item 1. [University of Lincoln].

[50] Khan, M., Habibullah, G., Gay, J. and Horkoff (2022). Non-Functional Requirements for Machine Learning: An Exploration of System Scope and Interest. [online] doi:<https://doi.org/10.1145/3526073.3527589>.

[51] Kingma, D. and Lei Ba, J. (2017). *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. [online] Available at: <https://arxiv.org/pdf/1412.6980.pdf>.

[52] "Helwan, A., El-Fakhri, G., Sasani, H., & Uzun Ozsahin, D. (2018). Deep networks in identifying CT brain hemorrhage. *Journal of Intelligent & Fuzzy Systems*, 35(2), 2215-2228."

[53] Lutz Prechelt (1998). Early Stopping — But When? [online] pp.53–67. doi:https://doi.org/10.1007/978-3-642-35289-8_5.

[54] FelipeKitamura (2018). *Head CT - hemorrhage*. [online] Kaggle.com. Available at: <https://www.kaggle.com/datasets/felipekitamura/head-ct-hemorrhage> [Accessed 17 May 2023].

[55] Radiological Society of North America. (2023). *RSNA Intracranial Hemorrhage Detection* / *Kaggle*. [online] Available at: <https://www.kaggle.com/competitions/rsna-intracranial-hemorrhage-detection/data> [Accessed 18 May 2023].

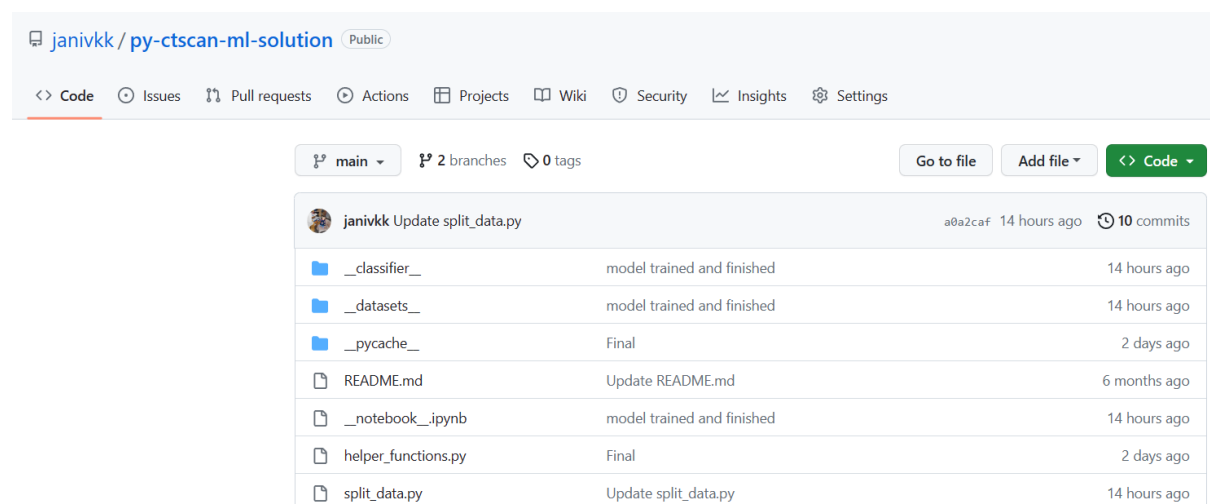
Appendix

Errors that I came across with:

RuntimeError: mat1 and mat2 shapes cannot be multiplied (1x10240 and 2560x2) when doing 128 by 128

img should be PIL Image. Got <class 'torch.Tensor'>

GitHub Repository:



The screenshot shows the GitHub repository page for 'janivkk / py-ctscan-ml-solution'. The repository is public and has 2 branches and 0 tags. The main branch is selected. The repository contains several files and folders, including __classifier__, __datasets__, __pycache__, README.md, __notebook__.ipynb, helper_functions.py, and split_data.py. The most recent commit is 'Update split_data.py' by janivkk, made 14 hours ago. The repository also has 10 commits in total.

File/Folder	Description	Commit Time
__classifier__	model trained and finished	14 hours ago
__datasets__	model trained and finished	14 hours ago
__pycache__	Final	2 days ago
README.md	Update README.md	6 months ago
__notebook__.ipynb	model trained and finished	14 hours ago
helper_functions.py	Final	2 days ago
split_data.py	Update split_data.py	14 hours ago

