جامعة آل البيت
AL al-BAYT UNIVERSITY

Automated Intracranial Hemorrhage Detection Using Deep Learning

By:

Ahmad Mahmoud Mustafa Alqaisi Unv. ID: 2100908032

Sarah Khaled Rashed Almashagbeh Unv. ID: 2100908166

Supervised by:

Dr. Najah Mithqal Al-Shanableh

Faculty Of Prince Al-Hussein Bin Abdallah II for IT
2024-2025

Acknowledgements

We would like to express our sincere gratitude to Dr. Najah Al-Shanableh for her continuous support and dedicated follow-up throughout our graduation project. Her invaluable guidance and extensive expertise, particularly in the field of Data Science and Artificial Intelligence, have played a crucial role in shaping our project.

Dr. Najah's deep knowledge in both fields has greatly contributed to our understanding, allowing us to effectively apply AI techniques in our work. Her ability to simplify complex concepts and provide insightful direction has been instrumental in our progress. Her patience, commitment, and encouragement have motivated us to push our limits and achieve the best possible results.

We deeply appreciate the knowledge and experience Dr. Najah has shared with us, and we are truly grateful for her continuous support.

We also extend our sincere thanks to our families, colleagues, and friends for their endless support, encouragement, and valuable discussions that contributed to our work. This journey would not have been the same without each one of them.

Abstract

Intracranial hemorrhage detection (ICH) is a serious medical condition that necessitates a prompt and exhaustive medical diagnosis. This project a multi-label ICH classification issue with six different types of hemorrhages, namely Epidural (EPD), Intraparenchymal (ITP), Intraventricular (ITV), Subarachnoid (SBC), Subdural (SBD), and Any. This project presents a hybrid deep learning approach that combines Convolutional Neural Networks (CNN) and Long Short Term Memory approaches (LSTM). Leveraging a ResNeXt-101 convolutional neural network for spatial feature extraction with a bidirectional LSTM network to capture temporal dependencies across slice sequences. Comprehensive preprocessing steps—including DICOM parsing, Hounsfield Unit calibration, multi-window image construction, and dynamic augmentation via Albumentations—were applied to enhance input consistency and generalization. Labels were reformatted into a multi-label binary matrix, enabling simultaneous classification of six hemorrhage types. The model achieved a private leaderboard log loss of **0.04604** on the RSNA 2019 ICH Detection Challenge, demonstrating strong generalization to unseen data. Despite challenges such as data imbalance and the absence of 3D context, the proposed system lays a foundation for explainable and scalable clinical AI. Future work will focus on integrating retrieval-augmented report generation and radiologist feedback mechanisms to further support real-world deployment.

Table of contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Full Term**
AI	Artificial Intelligence
AMP	Automatic Mixed Precision (NVIDIA Apex AMP)
AUC	Area Under the Receiver Operating Characteristic Curve
BCE	Binary Cross Entropy
BitsAllocated	Bits allocated per pixel (usually 16 bits)
BitsStored	Bits actually used to represent pixel data
CNN	Convolutional Neural Network
CRISP-DM	Cross Industry Standard Process for Data Mining
CSV	Comma-Separated Values
CT	Computed Tomography
DCM	Digital Imaging and Communications in Medicine (DICOM) file format
DL	Deep Learning
EDA	Exploratory Data Analysis
EPD	Epidural Hemorrhage
FC	Fully Connected (layer)
FP16 / FP32	Floating Point 16-bit / 32-bit Precision
GAP	Global Average Pooling
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients

HU	Hounsfield Unit
HighBit	Position of most significant bit in pixel value
ICH	Intracranial Hemorrhage
IVH	Intraventricular Hemorrhage
ImageOrientationPatient	Orientation of the image plane (3D vector)
ImagePositionPatient	Physical 3D position of the image slice
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LBP	Local Binary Pattern
LSTM	Long Short-Term Memory
ML	Machine Learning
MONOCHROME2	Grayscale interpretation where higher values are brighter (DICOM)
Modality	Imaging technique (e.g., CT)
PCA	Principal Component Analysis
PatientID	Unique identifier for a patient
PhotometricInterpretation	Grayscale/color image format description
PixelRepresentation	Pixel value format (signed/unsigned)
PixelSpacing	Physical spacing of pixels (mm)
RAM	Random Access Memory
RNN	Recurrent Neural Network
ROI	Region of Interest
RSNA	Radiological Society of North America
RescaleIntercept	HU calibration offset

RescaleSlope	HU calibration scale
SBC	Subarachnoid Hemorrhage
SBD	Subdural Hemorrhage
SOPInstanceUID	Service-Object Pair Instance Unique Identifier (DICOM)
SVM	Support Vector Machine
SamplesPerPixel	Number of color channels per pixel
SeriesInstanceUID	Unique identifier for image series
SliceThickness	Thickness of the CT slice (mm)
StudyID	Human-readable identifier of study
StudyInstanceUID	Unique identifier for imaging study
TPU	Tensor Processing Unit
ViT	Vision Transformer
WL	Window Level
WW	Window Width

This Page is Intentionally Left Blank

1. Introduction

Stroke is the second leading cause of death globally, causing 5.5 million deaths yearly, and Intracranial Hemorrhage (ICH) is the most severe subtype of stroke. Its mortality rate is high, and most survivors experience significant disability [1]. In 2021 alone, there were 3.444 million new cases of ICH worldwide, with an age-standardized prevalence rate of 40.8 per 100,000 people. That same year, ICH caused 3.308 million deaths, with an age-standardized mortality rate of 39.1 per 100,000 people [2].

1.1 Background and Motivation

Intracranial Hemorrhage (ICH) refers to bleeding within the brain tissue or between the brain tissue and the skull, often caused by a leaky or burst blood vessel. This blood buildup within the skull can cause Intracranial Pressure (ICP) which can crush delicate brain tissue or limit its blood supply, by which it can destroy brain cells due to the lack of oxygen. A brain bleed can occur following a head trauma, a blood clot, cerebral aneurysm (a weak spot in a blood vessel wall), or a brain tumor. If not treated quickly, it can lead to brain damage or death. ICH accounts for 10% to 20% of all strokes, exhibiting a 30-day fatality rate varying between 13% and 61%, with most fatalities occurring within the first 24 hours [1,3,4].

1.2 Problem Statement: Diagnostic Challenges and Misdiagnosis Risks

Given the high mortality and disability associated with ICH, one of the most critical factors contributing to this outcome is misdiagnosis. According to [5] and [6], stroke is the primary cause of serious harm due to misdiagnosis. Specifically, 17.5% of the 795,000 Americans who experience a stroke annually either die or become disabled as a result. The accuracy of diagnosis can be further influenced by the mental status of the patient. A study of 15,721 individuals, the risk of misdiagnosis was found to increase significantly when neurologic complaints were mild, nonspecific, or transient. For instance, individuals diagnosed with subarachnoid hemorrhage (SAH) with normal mental status had a 23.8% chance of misdiagnosis, whereas those with abnormal mental status had only 4.2% chance [7].

1.3 Challenges in Diagnosing Intracranial Hemorrhage Using CT Imaging

Despite being the primary imaging tool for detecting intracranial hemorrhage, CT imaging poses several diagnostic challenges that can hinder timely and accurate identification of brain bleeds, particularly in complex or subtle cases.

1.3.1 Limitations of CT Imaging

The primary diagnostic method for identifying and quantifying bleeding in central nervous system cases is head computed tomography (CT). CT scans have been widely used due to their speed and efficiency and their ability to show hyperattenuating areas. These areas are often associated with fresh blood (acute bleeding stage) because it is rich in hemoglobin which is denser and absorbs more x-rays and appears brighter than surrounding brain tissue. Refer to Figure 1 an illustration of hyperdense areas in CT imaging.

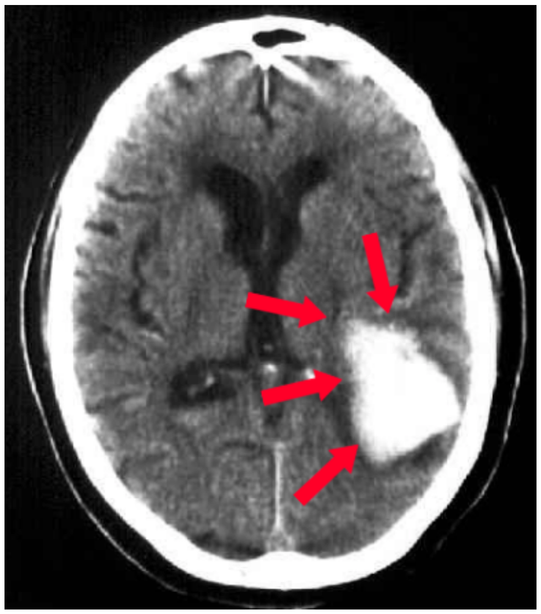


Figure 1: Intraparenchymal hemorrhage [8].

1.3.2 Interpretation Challenges

However, accurate diagnosis of ICH represents a frequent challenge for practicing radiologists. Acute bleeding and other abnormalities such as tumors or calcifications that can mimic bleeding must be distinguished properly. As for calcifications, which are deposits of calcium salts that can form in various tissues in the body, in CT scans, they appear brighter just like in acute bleeds, mimicking the appearance of ICH, leading to potential diagnostic error. Other secondary effects, such as edema or midline shift, may obscure bleeding and complicate assessment. Moreover, radiologists must accurately localize bleeding in anatomically complex areas, such as the brainstem or basal ganglia. Another aspect to look at is blood evolution. As the blood ages (subacute or chronic stages), it undergoes various biological changes such as breakdown of blood cells or absorption of water; these changes reduce its density, making it appear less dense on a CT scan and more similar to surrounding tissues; which is so called Isoattenuating hemorrhages. It becomes visually harder for radiologists to distinguish the blood from normal tissue, increasing the risks of mis or delayed diagnosis. For instance, isoattenuating hemorrhage can occur in subarachnoid hemorrhage (SAH) where small amounts of blood mix with the cerebrospinal fluid (CSF). An analysis was conducted on CT scans of patients with subdural hematomas (SDH) found that approximately 25% were isoattenuating [9].

1.4 Radiological and Operational Burdens

From a technical perspective, one of the critical challenges the radiologist has to navigate is windowing and leveling, where each tissue has a different CT density controlled in the CT scan using window level (WL) and window width (WW) to highlight. Incorrect window settings can obscure subtle bleeds. Refer to Figure 2, an illustration of windowing application on CT slices.

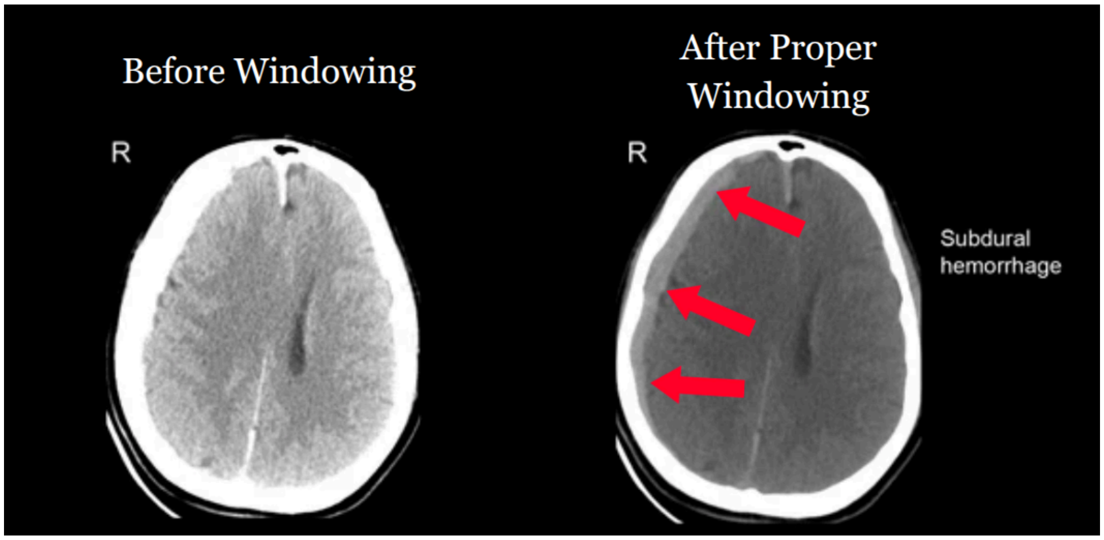


Figure 2: Proper Winding in ICH Detection [10].

As part of the ongoing efforts in enhancing image resolution, manufacturers of CT imaging devices have doubled the number of image slices in each 3D scan. However, due to the increasing workload for radiologists, each CT slice must be evaluated within approximately three seconds, and it can be challenging to correctly diagnose acute ICH and its subtypes based on CT imaging [11]. A study of 383 ICH cases (83 in the COVID-19 era and 255 in the pre-COVID-19 era) showed that the clinical outcomes of patients worsened in the COVID-19 era due to delays in the diagnosis of ICH, which led to poorer patient outcomes [12].

1.5 Proposed Solution: Role of Artificial Intelligence in Medical Imaging

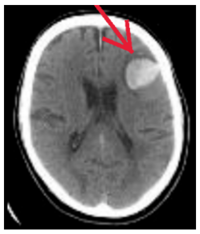
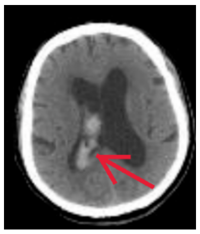
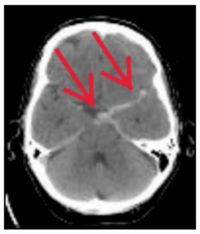
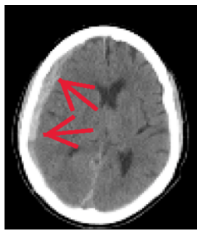
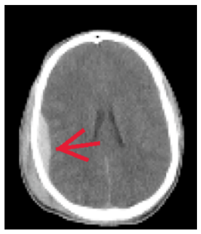
From the challenges and statistics discussed, it is shown that the traditional diagnostic methods struggle to meet the demand for timely and accurate results. This underscores the urgent need for improvement, to develop systems that can effectively manage the high patient volumes and offer radiologists a trustable decision support system, providing a potential solution with the highest possible sensitivity and specificity. To address these limitations, innovative solutions are required. A transformative tool that has emerged in the health sector is Artificial Intelligence

(AI), significantly enhancing the accuracy of diagnosis, the efficiency of treatment and the overall quality of patient care. With the help of deep learning algorithms, AI-powered medical imaging can identify complex patterns and abnormalities that might be overlooked due to overload or fatigue and enable medical practitioners to detect diseases with a higher level of precision and speed than ever before, thus enabling earlier intervention, which is crucial, especially in emergency settings such as route traffic accidents (RTAs). AI can act as a complementary tool for radiologists, by which they can combine patient medical history and other clinical data and provide tailored treatment and improve patient outcomes. Additionally, AI ensures consistency in the interpretation of cases, reducing diagnostic variability among radiologists. A study by researchers at UCSF and UC Berkeley showed that an AI algorithm competed with expert radiologists in detecting small brain hemorrhages on head CT scans, outperforming two out of four radiologists in accuracy [13].

1.6 ICH Classification and Anatomical Context

Intracranial Hemorrhage (ICH) is divided into five categories of hemorrhages with the sixth being classified as SOME or ANY hemorrhage. Different types of hemorrhage occur in different areas of the skull. Between the brain and the skull, there are three layers of tissue called the meninges that cover and protect the brain. Intraparenchymal hemorrhage (ITP) is blood that is located completely within the brain itself, Intraventricular (ITV) or Subarachnoid (SBC) hemorrhage is blood that has leaked into the spaces of the brain that normally contain cerebrospinal fluid (the ventricles or subarachnoid cisterns). SBC is when blood pools between the arachnoid and the pia mater. Extra-axial hemorrhages are blood that collects in the tissue coverings that surround the brain, such as Epidural (EPD) is a collection of blood between the skull bone and the dura mater while the Subdural (SBD) develops from a tear in a blood vessel and blood leaks out into the subdural space which is between the dura mater and the subarachnoid mater. Patients may exhibit more than one type of cerebral hemorrhage, which may appear in the same image. Table 1 depicts the type of bleeding, location as well as specifics of each hemorrhage.

Table 1: Intracranial Hemorrhage Subtypes Description

	Intraparenchymal	**Intraventricular**	**Subarachnoid**	**Subdural**	**Epidural**
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, tumor, etc.	Can be associated with both intraparenchymal and subarachnoid hemorrhages	Rupture of aneurysms or arteriovenous malformations or trauma	Trauma	Trauma or after surgery
Source	Arterial or venous	Arterial or venous	Predominantly arterial	Venous (bridging veins)	Arterial
Shape	Typically rounded	Conforms to ventricular shape	Tracks along the sulci and fissures	Crescent	Lentiform
Presentation	Acute (sudden onset of headache, nausea, vomiting)	Acute (sudden onset of headache, nausea, vomiting)	Acute (worst headache of life)	May be insidious (worsening headache)	Acute (skull fracture and altered mental status)

1.7. Documentation structure

This document is organized to reflect the logical progression of our project—from defining the problem of intracranial hemorrhage (ICH) misdiagnosis to designing, training, evaluating, and analyzing a hybrid deep learning solution using convolutional and sequence models. This structure supports clarity, reproducibility, and technical rigor while aligning with academic standards for medical AI research:

- **Chapter 1: Introduction**
  Establishes the clinical motivation behind automated ICH detection, including diagnostic challenges in CT imaging, risks of misinterpretation, and the potential of AI to address these issues. It introduces the problem, outlines existing limitations, and sets the foundation for the proposed solution.

- **Chapter 2: Literature Review**
  Surveys recent advancements in AI-based solutions for ICH detection, ranging from classical machine learning to CNNs, hybrid models (e.g., CNN-LSTM), and transformer-based architectures. Comparative analysis highlights the rationale for adopting a ResNeXt + LSTM framework in this study.

- **Chapter 3: Methodology**
  Details the modeling pipeline using **Cross Industry Standard Process for Data Mining CRISP-DM** framework. It includes comprehensive sections on data acquisition, preprocessing and feature extraction, and training strategy. This chapter also justifies architectural decisions based on exploratory data analysis (EDA) and clinical domain constraints. It also covers the training process, data batching strategies, loss masking, and optimization techniques. Evaluation is integrated here as well, including training loss analysis, leaderboard-based performance from the RSNA challenge, and interpretation of loss trends for the trained models to assess convergence and generalization.

- **Chapter 4: Discussion and Conclusion**
  Reflects on the findings,analysis, and proposes future work such as model deployment in radiology workflows, handling of rare hemorrhage cases, and exploration of transformer-enhanced temporal modeling.

- **Chapter 5: Conclusion**
  Provides a concise summary of the project contributions, the effectiveness of the proposed approach, and future directions such as improving rare class sensitivity, transformer integration, or user interface deployment.

- **References**
  Includes a complete list of all referenced academic papers and official web-based sources.

This structure ensures the reader is guided from problem definition through technical implementation to empirical validation, supporting both domain experts and machine learning practitioners in understanding the design, behavior, and real-world relevance of the proposed ICH detection system.

2. Literature Review

This chapter provides a comprehensive overview of AI applications, particularly focusing on the advancements in AI methodologies for detecting Intracranial Hemorrhage (ICH) through medical imaging. It reviews relevant previous studies, highlighting key AI approaches such as CNNs, hybrid models, transformer architectures, and classical machine learning methods.

2.1 Overview

Artificial Intelligence (AI) refers to the ability of machines to perform cognitive functions associated with human intelligence, such as perception, reasoning, learning, interaction, problem solving, decision-making, and even demonstrating creativity. AI has transformed many fields; one of them is healthcare, where it uses complex algorithms to mimic human cognition in the comprehension, analysis, and interpretation of medical data. In medicine, AI systems assist in various processes, including detection, diagnosis, analysis and treatment planning. AI can now predict patient outcomes and tailor treatment plans by analyzing medical history or chatting with patients and providing them with medical advice and appointment scheduling. The integration of AI into healthcare will improve patient outcomes, increase operational efficiency and reduce costs.

In radiology, AI has made significant advancements, particularly in the field of medical imaging and has shown great potential in detecting ICH. Consequently, researchers have extensively explored various AI-based approaches ranging from traditional machine learning methods to advanced deep learning architectures to analyze brain CT scans and help detect abnormalities at early stages.

2.2 Related Work

This section explores specific AI methodologies applied in recent research for the detection and classification of Intracranial Hemorrhage. Section 2.2.1 explores **CNN-based models** which are widely used for analyzing CT images. Section 2.2.2 covers **hybrid CNN models** that combineCNNs with other techniques like LSTM or attention mechanisms. Section 2.2.3 focuses on **transformers**, a newer type of model that can capture more complex patterns in the data. Finally, Section 2.2.4 discusses **classical machine learning methods**, simpler but less powerful than deep learning models.

2.2.1 CNN-based Approaches

Convolutional Neural Networks (CNNs) have been widely adopted due to their ability to extract spatial features from CT images. In medical tasks, CNNs are commonly combined with other neural architectures (e.g., recurrent neural networks for sequential data and transformers for pattern recognition) to enhance their applicability and effectiveness in complex medical data analysis. Ye et al. [14] proposed a three-dimensional convolutional and recurrent neural network (CNN-RNN) to detect ICH and its subtypes using a total of 76,621 slices from non-contrast head CT scans. The model achieved an overall accuracy of 99%, with a sensitivity of 98%, specificity of 99%, and an AUC of 1.00 for ICH detection. The model outperformed three junior radiology trainees and a senior radiologist for both the two-type and five-type tasks. The algorithm took less than 30 seconds on average to fully process a 3D head CT scan which is shorter than the reported head CT interpretation time of radiologists which is usually more than 5 minutes.

2.2.2 Hybrid CNN Model

Several studies have utilized the Radiological Society of North America (RSNA) dataset with patient files from 4 international hospitals, containing a total of 752,803 labeled DCM files containing cross-sectional images of the brain that are 512x512 in size and 121,232 test images [15]. Badriyah et al. [16] applied a hybrid CNN architecture combining Inception v3 and a custom convolutional layer to classify different types of ICH from CT-Scan images from [15]. They reached an average accuracy of 80% for each type of brain hemorrhage. Rajagopal et al. [17] also presented a hybrid CNN architecture that combines CNNs and Long Short-Term Memory (LSTM). They used a training dataset of 72,516 CT scans, while validation and testing were conducted on 7,515 and 7,512 respectively. There was a significant class imbalance among various types of hemorrhage, this was resolved by employing class weights to allow neural networks to train more aggressively on infrequently observed hemorrhages. In the first stage they used a systematic windowing method which resembles the process employed by radiologists in which the scan is systematically viewed under multiple window settings before an assessment is made. Windowing is a technique used in CT scan image processing to adjust contrast and brightness by modifying the window width (WW) and window level (WL). Applying different window settings helps modify the appearance of the CT image and highlights structures so that brain CT scans can be analyzed, and any abnormalities are identified. In [17] Ten different windows 'views' of the images were applied, and the result is stacked. After the optional windowing, pixel values are normalized to obtain a value between 0 and 1. The model applies Glorot Uniform initialization to prevent gradient issues and L2 regularization to mitigate overfitting. Experimental findings show that the proposed model has a sensitivity of 93.87% and an accuracy of 95.14%. The subtype-classification assessment ranking represents balanced outcomes across different hemorrhage subtypes, however, the EPD and ITV provide much poorer classification results than the other subtypes, due to class imbalance, the model used class weighting to address this imbalance, but it was insufficient. Future improvements could involve introducing an optimization algorithm to select the optimal features and improve performance. Similarly, Lewick et al. [18] follows a similar approach.They applied 3 different window width and window level for subdural/blood (WL=80, WW=200), brain matter (WL=40, WW=80), looking for bleeding within the brain and bone window (WL=600, WW=2800) to identify bleedings that occur by the skull, namely in subdural and epidural cases. Then, normalizing each image into [0-1] scale and creating 3 color channels by stacking the 3 windowed images to create the input tensor for the model. They utilized the convolutional base employed in ResNet50 [19] as a feature extractor, followed by a 2-layer fully-connected classifier, achieving a 98% accuracy. However, they report a lower recall score of 76%, primarily due to data imbalance. ResNet50 was chosen for its simplicity and regularity of structure.

M. H. O. Rashid et al. [20] have employed a DensNet architecture to tackle the RSNA 2019 challenge. Their work achieved 98% accuracy, outperforming other architectures which were confirmed in a comprehensive study [21], showing DensNet capable of performing better with less cost, less time, and high accuracy. D. Veselov et al. [22] starts with an iterative stratification method, which ensures that the new dataset is consistent with the original dataset and it has the same label distribution as in the original dataset, resulting with 65859 training images and 11623 images for validation. As in [18], the authors applied the same windowing process except the bone window and they replaced it with the soft tissue window (WL=20 to 60, WW=350 to 400). They resize the images to 224 by 225 using linear interpolation and normalize each channel individually and they apply data augmentation to increase the generalization ability of the neural network model by horizontal mapping, shift, rotation, scaling and brightness adjustment. They trained different CNN architectures including Xception, VGG-16, ResNe, Inception, EfficientNet and MobileNet, showing that Xception achieved the highest accuracy of 95.18% and an AUC of 96.39%. Additionally, Kumar Kar et al. [23] introduced attention mechanisms to help CNNs focus on the most important areas of the CT scan rather than treating all areas equally, this makes CNN more accurate since it pays more attention to hemorrhage prone areas. The dataset used consists of CT scans from 82 patients with approximately 30 image segments per patient. To improve the practicality of the model and expand the dataset, data augmentation is employed, as previously done in [22], but with variations in the applied transformations such as rotation, scaling, inversion, translation and noise injection. An accuracy of 99.76% was achieved on a test set of 937 images and has an AUC score of 0.9976. Possible limitations could be due to limited generalization or high computational cost since CNNs with attention mechanisms require a lot of GPU power making them expensive to train and deploy in hospitals. The paper recommends developing user-friendly interfaces for clinicians, resulting in faster diagnostic procedures and enhanced patient outcomes.

2.2.3 Transformer Architectures

More recently, vision transformers (ViTs) have been directly applied to ICH classification. Unlike CNNs, which rely on local feature extraction, ViTs utilize self-attention to capture global dependencies across CT slices. Adel ELZemity et al.[24] studied ICH detection and classification on the RSNA ICH dataset [15] where the dataset was split into 85% training and 15% testing sets while the training set was further divided into 90% training and 10% validation. Input CT slices were converted to 3-channel images using Hounsfield Unit (HU) windows for brain, blood and soft tissues. Images were also resized to 224 by 224 pixels, black edges were removed and slices were stacked into a 4D tensor. This paper presents a transformer-based architecture using the Swin Transformer and sequence transformer, more specifically, the Swin-B model as the feature extractor within each slice. Achieved a log loss of 0.04372 and developed a GUI desktop application for an easy-to-use experience for radiologists. The paper suggests that further studies can be conducted to optimize the architecture and extend its application to other medical imaging tasks.

Further exploring the potential of transformer architectures, Xian Wang et al. [25] introduced a hybrid CNN-Transformer model designed to combine the feature extraction power of CNNs with the global attention capabilities of transformers. Their approach involved using a ResNet34 for extracting 2048-dimensional features then passed to a vision transformer encoder to enhance spatial understanding. They used the RSNA dataset for training and the C1500 dataset labeled by three radiologists for validation and testing. The DICOM images were processed using seven different window settings in order to correctly evaluate their effects, , the scanning results after the different window settings mentioned were inputted into the model for experiments. it can be seen that using the window setting of (brain window + subdural window + bone window) can result in the highest AUC and F1 score of the model. They used Asymmetric Loss ASL instead of binary cross entropy to handle class imbalance. The proposed ResNet + Vision Transformer model achieved an AUC of 0.98 and F1-score of 0.97 .

2.2.4 Classical Machine Learning Approaches

While deep learning approaches dominate the field, classical machine learning models have also been explored. M. A. A. Majeed et al. [26] used standard machine learning methods such as support vector machines, decision trees, and random first for automatically detecting ICH. Their methodology involved a preprocessing pipeline that removes the bone from the skull, then experimenting with various feature extraction methods and feature selection models. The fundamental objective of feature selection is to provide the smallest subset of features that can accurately capture the key traits of the whole input dataset; they chose principal component analysis (PCA). The final stage is to build and train ML models and test them on the test set. They documented their observations and findings after each experiment with various feature extraction methods, and it was shown that the results that have been approved were the performance results of the Random Forest classifier with an accuracy of 92.5%, a precision of 88%, an impressive recall of 100% and F1-score of 93.6%. However, this study is limited to 200 CT slices, 100 with normal anatomy and 100 show hemorrhage. Notably, the dataset does not distinguish between different types of hemorrhage. The authors suggested that future research should evaluate the proposed method on a significantly larger and more diverse dataset. Similarly, Rai et al [27] implemented similar extraction features Histogram of Oriented Gradients (HOG) and Linear Binary Pattern (LBP), also, resized and flattened images. A total of 600 images were assembled from various sources from vast databases such as Kaggle, Radiopaedia, radiologypics, and the dataset was curated and balanced by essentially selecting 100 images for each of the hematoma conditions then split in a 70:30 ratio. Differently, [26] split it into a 80:20 ratio. A total of 9 machine learning models have been used: KNN, SVM (Linear and Polynomial), Random Forest, Naïve Bayes, Decision Tree (AdaBoost and Gradient Boosting). Results showed that gradient boosting had the highest accuracy of 96.7%, and Naïve Bayes performed the worst with an accuracy of 78.80%. In spite of their effectiveness, traditional methods lack the scalability and feature extraction power of deep learning models, making CNNs and transformers the preferred choice for ICH detection.

2.3 Comparative Analysis of AI Methodologies for ICH Detection

This section presents a comprehensive comparison of the methodologies outlined in Section 2.2, examining the relative strengths and limitations of each approach category –namely, CNN-based model, hybrid CNN architectures, transformer based models, and classical machine learning techniques in performance, computational complexity, generalizability and potential for critical utility.

2.3.1 Convolutional Neural Network (CNN)-Based Approaches

CNNs have demonstrated strong performance in spatial feature extraction from CT images. The work by Ye et al. illustrates the effectiveness of a 3D CNN-RNN model in achieving near-perfect classification metrics, including 99% accuracy and an AUC of 1.00. Such models not only rival but often exceed the diagnostic performance of human radiologists.

Advantages:

- High accuracy and robust detection capabilities for various ICH subtypes.
- Scalability for large datasets due to their parallel computation efficiency.
- Strong alignment with image-based data modalities, making them well-suited for radiological tasks.

Limitations:

- High computational cost and training time, especially for 3D models.
- Potential susceptibility to overfitting when trained on limited or homogeneous data.
- Limited capacity for modeling inter-slice dependencies unless integrated with sequential modules.

2.3.2 Hybrid CNN Architectures

Hybrid models that integrate CNNs with components such as LSTMs or attention mechanisms aim to enhance temporal and contextual understanding. For example, Rajagopal et al. employed CNN-LSTM architectures, incorporating multiple CT window views and class weighting to address data imbalance.

Similarly, Lewick et al. used a multi-windowing strategy to enrich the input space and leveraged ResNet50 for spatial feature extraction.

Advantages:

- Enhanced modeling of temporal patterns and inter-slice relationships.
- Integration of domain-specific preprocessing (e.g., windowing) that mimics radiological workflows.
- Improved classification outcomes through architectural innovations and regularization techniques.

Limitations:

- Increased model complexity and longer training durations.
- Risk of performance degradation in the presence of highly imbalanced datasets.
- Lower recall in some cases, potentially compromising sensitivity to hemorrhages.

2.3.3 Transformer-Based Architectures

Transformer-based models, particularly Vision Transformers (ViTs), have emerged as powerful tools in medical imaging due to their ability to capture global dependencies. Studies such as ElZemity et al. and Wang et al. implemented transformer or hybrid CNN-Transformer models and reported high accuracy (AUC ~0.98) and usability through GUI development.

Advantages:

- Superior capability in modeling long-range dependencies and spatial context.
- Competitive performance with state-of-the-art metrics, particularly in multiclass classification tasks.
- Suitability for integration into clinical decision support tools via interface development.

Limitations:

- High computational demands during training and inference.
- Complexity in tuning and interpretability, which may hinder clinical adoption.
- Vulnerability to overfitting in small or imbalanced datasets without proper regularization.

2.3.4 Classical Machine Learning Approaches

Classical machine learning techniques, including support vector machines, random forests, and gradient boosting, have been explored primarily as lightweight alternatives.

For instance, Majeed et al. and Rai et al. achieved reasonable accuracy using handcrafted features such as HOG and LBP.
Advantages:

- Low resource requirements and faster training cycles.
- High interpretability, making them suitable for initial prototyping and educational settings.
- Effective on small datasets where deep learning may be prone to overfitting.

Limitations:

- Limited scalability and generalization capabilities.
- Reliance on manual feature extraction reduces adaptability.
- Inability to model complex spatial hierarchies and temporal patterns inherent in CT scan data.

2.4 Rationale for the Hybrid CNN-RNN Approach

The selection of a hybrid architecture combining a Convolutional Neural Network (CNN) with a Recurrent Neural Network (RNN) in this study is motivated by the need to effectively capture both spatial and temporal information inherent in volumetric CT data. While CNNs such as ResNeXt-101 are proficient at learning discriminative spatial features from individual 2D slices, they are inherently limited in modeling contextual relationships across consecutive slices. This limitation poses a challenge in the context of intracranial hemorrhage (ICH) detection, where diagnostic accuracy often relies on understanding anatomical progression and continuity throughout a 3D scan.

To address this, we extract embeddings from a ResNeXt-101 model and feed them into a Long Short-Term Memory (LSTM) network to learn temporal dependencies between slices. This sequential modeling process closely mirrors the workflow of radiologists, who assess CT images slice-by-slice to interpret spatial patterns and identify subtle hemorrhagic cues that may not be apparent in isolated slices.

Our approach is supported by recent work in the field. For instance, Ye et al. (2020) introduced a CNN-RNN framework that achieved state-of-the-art performance in ICH detection, outperforming both junior and senior radiologists on large-scale datasets. Similarly, Rajagopal et al. (2021) demonstrated the benefits of combining CNNs with LSTM modules and domain-specific preprocessing techniques such as windowing to improve subtype classification and robustness in imbalanced datasets.

By leveraging both spatial feature learning and temporal sequence modeling, the proposed hybrid CNN-RNN architecture aims to improve diagnostic performance, enhance generalization across varied hemorrhage types, and contribute toward clinically viable AI-assisted tools in neuroimaging.

2.5 Conclusion

To synthesize the reviewed studies and provide a comparative overview, Table 2 presents a structured summary of the key contributions in recent ICH detection literature. The table highlights essential details for each study, including the authors, year of publication, dataset used, modeling techniques, primary findings, evaluation metrics, and reported results. This consolidated overview allows for easier comparison across methodologies.

Table 2: Summary of the key methodologies used in recent ICH detection research

SI no.	Authors	Year of publication	Dataset	Techniques used	Findings	Evaluation metric	Results
1	Ye et al.	2019	76,621 CT slices	3D CNN + RNN	High performance in detecting ICH and subtypes; outperformed radiologists	Accuracy, Sensitivity, Specificity, AUC	Accuracy: 99%, Sensitivity: 98%, Specificity: 99%, AUC: 1.00
2	Badriyah et al.	2023	RSNA dataset	Hybrid CNN (Inception v3 + custom conv layers)	Able to classify ICH types	Accuracy	Average Accuracy: 80%
3	Rajagopal et al.	2023	72,516 training + 7,515 validation + 7,512 test	CNN + LSTM with windowing, class weighting, regularization	Addressed class imbalance; poor performance on EPD & ITV	Sensitivity, Accuracy	Sensitivity: 93.87%, Accuracy: 95.14%
4	Lewick et al.	2020	RSNA 2019	ResNet50 + 3 Window views (brain,	High accuracy but lower recall due to imbalance	Accuracy, Recall	Accuracy: 98%, Recall: 76%

			challenge	subdural, bone)			
5	M.H.O. Rashid et al.	2023	RSNA 2019 challenge	DenseNet	Outperformed other architectures In cost, time and accuracy.	Accuracy	Accuracy: 98%
6	D. Veselov et al.	2024	65,859 train, 11,623 valid	CNN variants (Xception, VGG-16, etc.), Windowing, Data Augmentation	Xception model performed best	Accuracy, AUC	Accuracy: 95.18%, AUC: 96.39%
7	Kumar Kar et al.	2024	~2,460 images from 82 patients	CNN + Attention Mechanisms	High accuracy and AUC with attention; suggests GUI for clinician use	Accuracy, AUC	Accuracy: 99.76%, AUC: 0.9976
8	Adel ELZemity et al.	2023	RSNA	Vision Transformer (Swin-B), HU windows	Developed GUI; potential to generalize to other tasks	Log Loss	Log Loss: 0.04372
9	Xian Wang et al.	2022	RSNA + C1500	ResNet34 + Vision Transformer, Asymmetric Loss, 7-window settings	Highest performance with brain + subdural + bone windows	AUC, F1-score	AUC: 0.98, F1-score: 0.97
10	M.A.A. Majeed et al.	2023	200 CT slices Binary classification	Traditional ML (SVM, RF, etc.), PCA Skull removal	Random Forest achieved the best results	Accuracy, Precision, Recall, F1	Accuracy: 92.5%, Precision: 88%, Recall: 100%, F1: 93.6%
11	Rai et al.	2024	600 images from various sources	Traditional ML (KNN, SVM, RF, GB, etc.), HOG, LBP	Gradient Boosting outperformed all other models	Accuracy	Gradient Boosting: 96.7%, Naïve Bayes: 78.80%

3. Methodology

The methodology section of this project covers the detailed steps involved in using deep learning to accurately detect and classify intracranial hemorrhages in CT scans. In this study, we adopted the **CRISP-DM (Cross Industry Standard Process for Data Mining) methodology**, which provides a structured and industry-proven framework for guiding data science projects [28]. CRISP-DM was chosen due to its flexibility, iterative nature, and a clear division of phases, from understanding the medical problem to training and deploying the model, making it especially suitable for complex healthcare applications where domain understanding and data integrity are critical. The following Figure 3 shows the flowchart of the chosen methodology in this study.

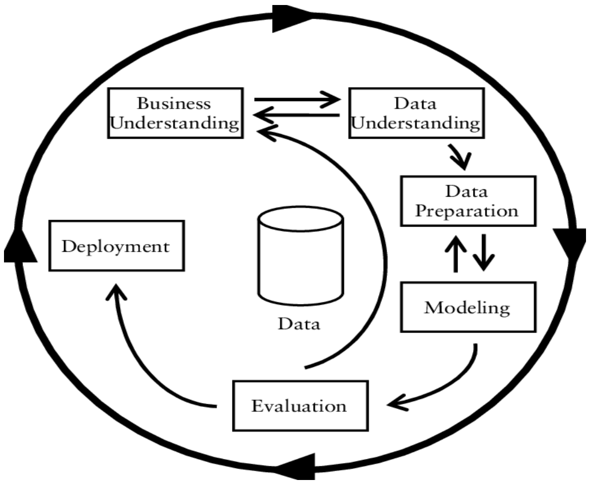


Figure 3: CRISP-DM.

3.1 Business Understanding

The Business Understanding phase in the CRISP methodology focuses on comprehending the project's objectives and requirements from a business perspective. This involves defining business goals, assessing the current situation, determining data mining goals, and formulating a project plan.

Intracranial Hemorrhage (ICH) is a life-threatening event that requires rapid and accurate diagnosis for prompt and effective treatment. The standard method for detecting ICH is computed tomography (CT) due to its speed and accuracy. However, manual diagnosis by radiologists is time-consuming and certainly prone to human error, especially in emergencies where time is critical or in mass diagnosis. Automated detection systems using deep learning (DL) offer a promising solution to enhance diagnostic efficiency, accuracy, and rapid detection.

3.1.1 Problem Statement

Despite significant advances in imaging technology, the accurate and timely diagnosis of Intracranial Hemorrhage (ICH) remains a clinical challenge with high stakes. Manual interpretation of CT scans by radiologists is not only time-consuming but also vulnerable to human error, particularly in high-pressure scenarios such as emergency departments or mass casualty events. As discussed, interpretation errors may stem from mimicking conditions like calcifications, subtle or isoattenuating bleeds, and operational constraints such as CT slice overload or suboptimal windowing. These limitations contribute to misdiagnosis, treatment delays, and ultimately poorer patient outcomes.

Therefore, the central question addressed in this study is:

How can deep learning (DL) models be effectively employed to detect and classify cerebral hemorrhages in CT scans in a way that reduces human dependency and diagnostic errors, while improving speed, consistency, and diagnosis accuracy?

The business goal of this project, therefore, is to develop a DL model that enhances diagnostic decision-making for ICH detection.

1.2 Domain Relevance:

This study holds significant potential to improve the radiology field in the healthcare sector. Key areas of impact include:

- **Emergency Care:** Enhancing survival rates for brain hemorrhage patients through rapid diagnosis and accurate prioritization.
- **Radiology Departments:** Enhancing efficiency through optimized resource allocation and workload reduction.

3.1.3 Motivation:

Delays in diagnosing can lead to severe complications or death. *According to* [29], Intracerebral Hemorrhage (IH) accounts for approximately 10% to 15% of initial stroke cases, with a mortality rate of 52% and approximately half of these deaths occurring within the first two days.

- **Key motivators may include:**
  - High Mortality Rates.
  - Critical Time: About half of the fatalities occur within the first two days of the stroke, highlighting the importance of rapid diagnosis.
  - Effective resource allocation allows radiologists to focus on more complex cases.

- **Key Stakeholders**:
  - **Patients**: Benefit from quicker and more precise diagnosis.
  - **Radiologists**: Reducing the workload and allowing them to focus on more complex cases and in addition to support decisions.

- **Healthcare Providers**: Resource usage optimization and raise the precision of diagnosis.
- **Medical device companies**.

3.2 Data Understanding

The second phase of the CRISP methodology involves collecting initial data and familiarizing oneself with it to identify data quality issues and discover initial insights. Activities include data collection, data description, data exploration, and verification of data quality.

The dataset used in this project was provided by the Radiological Society of North America (RSNA) as part of a Kaggle competition called RSNA Intracranial Hemorrhage Detection. The American Society of Neuroradiology (ASNR) organized a cadre of more than 60 volunteers to label over 25,000 exams for the challenge dataset [30].

2.1 Data Description

The dataset is divided into two parts: a labeled training set containing DICOM files of cross-sectional brain CT images with dimensions of 512×512 pixels and their corresponding metadata, and an unlabeled test set.

Labels for the training set are provided in a CSV file, which includes a set of image IDs and six labels representing the existence probability of each subtype. These include five specific hemorrhage subtypes and an additional label `any`, which is marked if any subtype is present. The labels are stored in a long format, meaning one row per label.

Each DICOM file is associated with six individual labels, framing the problem as a multi-label classification task, where a single image may be annotated with zero, one, or multiple hemorrhage subtypes. A preview of the CSV file's structure is shown in Table 3. Each DICOM file corresponds to a single slice extracted from a 3D CT scan.

Table 3: Structure of the CSV label file

ID	Label
1_epidural_hemorrhage	0
1_intraparenchymal_hemorrhage	0
1_intraventricular_hemorrhage	0
1_subarachnoid_hemorrhage	1
1_subdural_hemorrhage	0
1_any	1

3.2.2 DICOM files

Digital Imaging and Communications in Medicine (DICOM) is an international standard used for transmitting, storing, retrieving, and displaying medical imaging data. Files following this standard typically use the `.dcm` extension and encapsulate both image data and a structured header within a single file. The header contains metadata fields, or "tags," which may include patient-related information such as name, date of birth, age, and gender, as well as technical attributes like image dimensions, acquisition parameters, and pixel-related data. These standardized elements enable interoperability across imaging systems and support consistent handling of medical scans [31] .

3.2.3 Hounsfield Units and Windowing

Contrast refers to the difference in intensity or brightness between different structures and tissues in an image. High contrast means there is a significant difference in brightness, making it easier to distinguish between different structures and tissues, while low contrast means the opposite. Contrast is crucial in CT scans because they help differentiate between tissues and abnormalities based on their *attenuation properties*. Attenuation is a measure of how much a tissue absorbs the X-rays that pass through it, this measure is quantified in Hounsfield Units (HU) also referred to as CT units. HU is a relative quantitative measurement of radio density used by radiologists in the interpretation of CT images. The physical density of tissue is proportional to the absorption/attenuation of the X-ray beam. CT images are represented in HU which quantify the radiodensity of tissues, which naturally have different densities.
More dense tissue, with greater X-ray beam absorption, has positive values and appears bright; less dense tissue, with less X-ray beam absorption, has negative values and appears dark [32]. For instance, acute hemorrhage “fresh blood” appears hyperdense “brighter” due to its richness in hemoglobin, chronic hemorrhage may become isoattenuating as the hematoma liquefies and matches the density of the brain tissue of CSF.

CT Windowing, also known as contrast stretching or contrast enhancement, is the process in which the CT image grayscale component of an image is manipulated via the CT Units, doing this will change the appearance of the picture to highlight particular structures. The brightness of the image is adjusted via the window level, the contrast is adjusted via the window width. These two are the windowing parameters used in preprocessing for defining different window settings. The window width measures the range of CT numbers in an image that we want to display, the specific structures in the image we want to focus on. Window level, often referred to as window center, is the midpoint of the range of the CT numbers displayed [33].

3.2.4 Exploratory Data Analysis

Before developing and training any learning models, a thorough exploratory data analysis (EDA) was conducted to examine the structure, composition, and quality of the dataset. This section provides a multi-level overview of the available image data, metadata, and labels. It begins with a statistical summary of the image dimensions and spatial properties, followed by an investigation of volume characteristics such as slice count per scan. Next, the analysis delves into pixel-level attributes including voxel spacing, intensity depth, and photometric properties. The class distribution section assesses the frequency and imbalance of target labels, while the metadata exploration investigates contextual fields such as patient-study hierarchy, scan orientation, and acquisition parameters like windowing. The final portion presents descriptive statistics and missing data visualizations to assess completeness and consistency across all fields. Together, these analyses lay the groundwork for informed preprocessing, modeling strategies, and data validation in later stages.

3.2.4.1 Initial Image Statistics

In this section we present a quantitative overview of the raw image data used in this project. Understanding the structural attributes of the images—such as resolution, slice count, and voxel size—is essential for ensuring preprocessing and modeling compatibility.

a. Image Dimensions

Understanding the spatial resolution of medical images is essential for standardizing inputs during preprocessing and ensuring compatibility across different imaging protocols. In this study, a total of **873,985 DICOM images** were analyzed for their native pixel dimensions (height × width), revealing a high degree of homogeneity with a dominant resolution, alongside several outlier cases.

Dimension Distribution Analysis

In medical imaging datasets, spatial resolution is a critical attribute that influences both preprocessing requirements and model design. Images are generally expected to follow standardized dimensions—particularly in clinical CT imaging—to support uniform data handling. Consistent image dimensions simplify the preprocessing pipeline and enhance model stability by reducing the need for complex resizing or cropping operations.

However, datasets may contain images with non-standard dimensions due to various factors such as differences in scanner hardware, acquisition protocols, or post-processing techniques. These variations can result in formats that deviate from the standard, including smaller or irregular resolutions. Such inconsistencies may arise from exported reconstructions using alternate fields of view (FOV), different imaging systems, or manually adjusted study settings.

To address this variability, all input images were resized to a consistent spatial resolution during preprocessing. A padding strategy was applied where necessary to preserve anatomical proportions and ensure compatibility with convolutional neural network (CNN) architectures.

b. Number of Slices per Volume

Understanding the number of slices per CT volume is essential when preparing volumetric data for deep learning applications. Slice count directly affects the construction of 3D tensors, which are often required in models that leverage spatial continuity across axial views.

Variability in slice numbers can influence memory requirements, model input shapes, and the anatomical completeness of reconstructed volumes. This variation may stem from differences in scanning protocols, patient anatomy, or imaging equipment configurations.

To ensure compatibility with deep learning architectures that expect uniform input dimensions, preprocessing strategies are commonly employed. These may include slice interpolation, padding, or trimming to normalize the depth of each volume. Consistent handling of slice counts is critical for preserving structural integrity and optimizing performance in 3D convolutional neural networks.

c. Pixel Spacing / Voxel Size

Spatial resolution in medical imaging is influenced not only by the number of pixels in an image but also by their physical dimensions, which are defined by pixel spacing and slice thickness. These parameters are crucial for constructing anatomically accurate 3D volumes and ensuring consistency across data inputs in deep learning workflows.

In-Plane Resolution: Pixel Spacing

Pixel spacing describes the real-world physical size of each pixel in the x and y directions, typically measured in millimeters. This information is encoded in the DICOM `PixelSpacing` tag and is essential for preserving anatomical proportions during preprocessing. A consistent in-plane resolution simplifies alignment, resampling, and standardization tasks required prior to model training.

Through-Plane Resolution: Slice Thickness

Slice thickness defines the depth (`z-axis`) of each 2D slice in a volume and contributes to the voxel's overall dimensions. This value can be derived either from the DICOM `SliceThickness` tag or from positional differences between adjacent slices. Accurate handling of slice thickness is necessary for reliable 3D reconstruction and is especially important in volumetric modeling, where spatial relationships between slices must be preserved.

In practical preprocessing workflows, pixel spacing and slice thickness are often used to normalize scans to a fixed voxel size. This step supports the use of 3D convolutional neural networks (CNNs) and ensures computational and anatomical consistency across subjects. To handle irregularities, common strategies include resampling, interpolation, or exclusion of volumes that deviate significantly from expected spatial parameters.

d. Bit Depth and Pixel Value Range

Bit depth and pixel value representation are essential components of DICOM images, directly affecting the dynamic range, visual precision, and the interpretability of grayscale intensities, especially in medical modalities such as CT. These attributes are determined by several DICOM header fields including `BitsAllocated`, `BitsStored`, `HighBit`, and `PixelRepresentation`, each of which plays a specific role in defining how pixel values are stored and later transformed.

Several DICOM header fields define how pixel data is stored and interpreted:

Bit Allocation vs. Bit Storage

- BitsAllocated indicates the number of bits reserved per pixel, which determines the size of the data container.
- BitsStored defines the actual number of bits used to represent intensity values within the allocated space.

Although the container may support a higher capacity, only a subset of those bits may be utilized to encode intensity information. This distinction is important for understanding the effective resolution of the image and for designing preprocessing routines that handle scaling and normalization appropriately.

Most Significant Bit (HighBit)

The HighBit field represents the position of the most significant bit in use, based on zero-based indexing. This value is typically calculated as `BitsStored - 1` and helps ensure accurate interpretation of pixel values during decoding. Proper alignment between BitsStored and HighBit is crucial for avoiding misinterpretation of data during transformation or reconstruction.

Signed vs. Unsigned Representation

The `PixelRepresentation` field specifies whether pixel values are encoded as signed or unsigned integers:

- Unsigned values are used when only non-negative intensities are expected.
- Signed values are essential for modalities like CT, where pixel intensities may represent negative Hounsfield Units (HU), such as those corresponding to air or soft tissue.

These encoding differences are critical when converting raw pixel values into clinically meaningful units such as Hounsfield Units (HU). The transformation uses the DICOM fields `RescaleSlope` and `RescaleIntercept`:

$$HU = PixelValue \times RescaleSlope + RescaleIntercept \qquad \text{Eq(1)}$$

Given the mix of signed and unsigned formats, explicit casting and normalization were applied during preprocessing to ensure consistent interpretation across all slices.

e. Modality Breakdown

The `Modality` DICOM tag provides essential information about the imaging technique used to acquire each study. It defines the type of scanner employed, which in turn determines the nature of the image data, expected contrast characteristics, and clinical interpretation standards. Understanding the modality distribution in the dataset is crucial for selecting appropriate preprocessing techniques and deep learning architectures.

The exclusive presence of CT scans in the dataset has several technical and clinical implications:

- **Pixel Intensity Encoding**: CT images are typically encoded in **Hounsfield Units (HU)**, which quantify tissue radiodensity. This requires applying the `RescaleSlope` and `RescaleIntercept` DICOM fields to transform raw pixel values into HU space.
- **Signed Pixel Values**: CT datasets frequently use **signed 12-bit or 16-bit depth** to support the full HU range (typically −1024 to +3071 HU).
- **Windowing Needs**: Specific **window width (WW)** and **window center (WC)** values are required to highlight different anatomical structures (e.g., soft tissue, lungs, bones).
- **Consistent Slice Thickness**: Clinical CT studies generally maintain a uniform slice thickness of 1–5 mm, which has already been confirmed in section 3.2.4.1.c.
- **Preprocessing Benefits**: The modality consistency simplifies the pipeline by eliminating the need for modality-specific preprocessing branches (e.g., for MRI or X-ray).

3.2.4.2 Class Distribution

Class distribution is a critical component in understanding the balance of categories represented in a labeled dataset. In classification tasks—especially within medical or security imaging domains—imbalanced classes can result in biased model performance, overfitting on dominant classes, and reduced sensitivity to rare but important conditions. This section provides a comprehensive analysis of the class labels in the dataset, their distribution, imbalance severity, and related preprocessing strategies. The dataset includes categorical labels that represent the target classification classes. These labels were sourced from the accompanying metadata file (`stage_2_train.csv`), as shown in Table 3, and were converted from long to wide format. This mapping was used to annotate all data points and form the multi-class classification targets as shown in Table 4.

Table 4: Wide-Format Representation of Hemorrhage Categories

any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
1	0	0	0	1	1
0	0	0	0	0	0
1	1	0	1	1	0

The raw frequency of each class was computed to understand the overall representation across diagnostic categories.

Following the raw class counts, a percentage-based analysis was conducted to better understand the **relative frequency** of each specific **intracranial hemorrhage (ICH) subtype**. Since the dataset allows for **multi-label annotations**, and because some hemorrhages co-occur within the same image, this breakdown does not represent mutually exclusive samples. Instead, it highlights how frequently each subtype appears relative to the total number of ICH-labeled annotations.

Building upon the observed frequency disparities, a deeper analysis of class imbalance reveals a substantial skew in the dataset. While all five intracranial hemorrhage (ICH) subtypes are clinically important, their representation in the dataset is significantly lower compared to the normal class. This imbalance introduces challenges in model training, particularly in accurately detecting less frequent hemorrhage types.

At the individual class level, the disparity becomes even more evident. Some hemorrhage subtypes occur infrequently in the dataset, making them difficult for standard models to learn effectively. When such classes are compared to the dominant normal class, the resulting imbalance ratios can be severe—posing a high risk for underperformance on rare conditions unless specific mitigation techniques are employed.

This level of imbalance can lead to **model bias**, where the neural network minimizes overall loss by primarily optimizing for the dominant "normal" class. In practice, this results in high accuracy but **poor sensitivity (recall)** on positive ICH classes—an outcome that is **clinically unacceptable** in diagnostic tasks where missing a hemorrhage could lead to life-threatening consequences.

Moreover, the dataset's **multi-label nature** complicates this further. Since one scan may contain multiple ICH types—or none at all—the model must learn to **simultaneously detect multiple rare labels** with highly unbalanced representation. Certain subtypes (e.g., intraparenchymal and subarachnoid hemorrhages) may co-occur, while others like **epidural hemorrhage** tend to appear in isolation. This interdependence between rare labels adds **dimensional imbalance** on top of **label frequency imbalance**, compounding the challenge. In light of this extreme imbalance, several critical design decisions were made: **Label-specific loss weighting** was introduced to reduce the penalty gap between common and rare labels.

3.2.4.3 Metadata Exploration

In addition to pixel-level properties, DICOM files contain rich metadata that describes scan structure, patient information, imaging context, and acquisition parameters. This metadata plays a vital role in organizing, interpreting, and preprocessing medical imaging datasets. Understanding how images are grouped into studies and series, how patients are identified, and how spatial orientation is defined helps ensure that models are trained with contextual awareness, and that data integrity is maintained throughout the pipeline. This section explores the remaining structural metadata fields in detail, highlighting their hierarchy, distribution, and implications for downstream processing.

Hierarchical DICOM Metadata Structure: Patients, Studies, and Series

The hierarchical structure of medical imaging data in DICOM format follows a well-defined pattern: each **patient** may have one or more **studies**, and each study may contain one or more **series**, which in turn contain multiple **instances** (images).

The implications of this hierarchy are crucial for **volume reconstruction**, **patient-level sampling**, and **data stratification**. During preprocessing, care must be taken to group slices by `SeriesInstanceUID` to form 3D volumes, and to ensure that training, validation, and test splits respect `PatientID` boundaries to prevent data leakage.

An important aspect of the dataset lies in its organization at the volume level. Each volume is composed of a group of slices associated with a single series, reflecting a structured approach to imaging data collection. Understanding the composition of these volumes is essential when preparing the data for downstream tasks, such as 3D modeling or sequence-based analysis. The consistency in how slices are grouped within volumes can inform preprocessing strategies and influence model design choices.

The hierarchical structure of the metadata enables flexible navigation through different levels of granularity—ranging from individual images to entire studies. This includes a clear one-to-one relationship between studies and series, which simplifies grouping logic, and the presence of unique identifiers like SOPInstanceUID, which supports precise image-level operations. This foundational organization ensures that the data can be managed, processed, and analyzed effectively in both clinical and machine learning contexts.

Image Orientation and Position

Spatial metadata fields such as `ImageOrientationPatient` and `ImagePositionPatient` are critical in understanding the physical layout of 2D image slices in 3D space. These DICOM tags enable accurate reconstruction of 3D volumes from individual slices and are essential for preserving anatomical consistency, particularly when aligning slices or performing volumetric operations.

The field `ImageOrientationPatient` contains six values that define the direction cosines of the image plane relative to the patient's body. The first three values describe the orientation of the image rows, while the next three describe the orientation of the columns. For example, a typical value might look like:

$(1, 0, 0, 0, 0.99, -0.13)$

This means the image rows are aligned with the patient's left–right axis (x-axis), and the columns are slightly rotated in the posterior–anterior and head–foot directions (y- and z-axes). Such orientation information is crucial when interpreting oblique slices or scans acquired at non-standard angles, which is common in trauma or neuroimaging studies.

The `ImagePositionPatient` field defines the coordinates of the **upper-left corner of the image plane** in the patient's coordinate system (in millimeters). It consists of three values: x, y, and z positions. For instance, a typical value might be:

$(-125, -147.23, 46.63)$

This vector provides the precise physical location of the slice in 3D space and is used in conjunction with `ImageOrientationPatient` to **stack slices correctly** into 3D volumes. The `z`-value (third component) is especially important for **ordering slices** from head to toe or in the scan axis direction.

Reconstructing a slice stack into a meaningful 3D volume relies heavily on preserving the original spatial order of the slices. This is achieved by using the z-coordinate found in the `ImagePositionPatient` field, which represents the position of each slice along the body's vertical axis. Sorting the slices by this value is essential for maintaining anatomical accuracy. If slices are misordered or flipped, the resulting volume may appear distorted or anatomically incorrect, which would compromise both visual interpretation and downstream tasks such as segmentation or 3D model training.

Photometric Interpretation and Samples

The `PhotometricInterpretation` and `SamplesPerPixel` fields in DICOM metadata define how pixel values should be interpreted and how many samples (channels) are associated with each pixel. These fields are fundamental for correctly processing medical images, especially when preparing data for machine learning pipelines or visualization tools.

`PhotometricInterpretation` indicates whether the image is grayscale or color, and also specifies the direction of intensity mapping. For instance, different values may define whether brighter pixel values correspond to lighter or darker visual output, which can influence whether intensity inversion is required during preprocessing.

`SamplesPerPixel` describes the number of data samples that make up each pixel, determining whether the image has a single channel (e.g., grayscale) or multiple channels (e.g., RGB). This information is essential for shaping image tensors properly and ensuring compatibility with processing models that expect specific input formats.

Proper handling of these metadata fields is critical for maintaining image fidelity, avoiding misinterpretation, and ensuring consistency across the preprocessing workflow. They also help streamline the integration of DICOM images into standardized pipelines for both analysis and visualization.

Windowing Metadata

DICOM windowing metadata—specifically the `WindowCenter` and `WindowWidth` fields—plays a central role in controlling how grayscale medical images are visually rendered. These fields define the range of Hounsfield Unit (HU) values that will be displayed and how pixel intensities are mapped to grayscale shades on screen. Understanding their distribution and variability across the dataset is important for both **visualization consistency** and **preprocessing standardization**.

In CT imaging, `WindowCenter` specifies the midpoint of the display intensity range, while `WindowWidth` defines the width of the range (i.e., contrast). Pixel values outside this range are clipped: values below the range appear completely black, and those above appear white. These parameters are essential when visualizing specific anatomical structures, such as brain tissue, lungs, or bone, which all require different HU ranges for optimal contrast.

Accurately accounting for these windowing parameters is essential in preprocessing workflows, as they affect contrast normalization, intensity mapping, and overall consistency across scans. Understanding the variability and configuration of these fields helps ensure compatibility with visualization pipelines and supports standardized image interpretation in downstream tasks.

3.3 Data Preparation

The third phase of the CRISP methodology entails preparing the final dataset for modeling, which includes in general selecting relevant data, cleaning data, constructing new data attributes, integrating data from multiple sources, and formatting data appropriately.

The quality and structure of data directly impact the success of any machine learning model, particularly in the medical imaging domain where variations in format, acquisition, and annotation can introduce significant complexity. This section presents the complete data preparation pipeline applied to the RSNA Intracranial Hemorrhage Detection dataset. The goal of this phase was to transform the raw DICOM scans and diagnostic labels into a unified, clean, and efficient format suitable for deep learning models, while preserving clinical relevance and structural integrity.

The preparation process began with consolidating the imaging data and associated metadata. DICOM slices were parsed, decoded, and aligned with study-level diagnostic labels using unique instance identifiers. Each image was linked to its corresponding patient and study context, and a master dataset was constructed by merging the structural metadata with the multi-label annotation matrix. Challenges in handling the massive dataset—initially delivered as a 181 GB compressed archive—were addressed using a hybrid approach of local processing, grouped compression, and cloud-optimized transfer mechanisms.

Next, all DICOM slices were rescaled into Hounsfield Units using the intercept and slope values provided in the DICOM headers. Rather than choosing a single window, a multi-windowing policy was applied to highlight different anatomical regions relevant to hemorrhage detection: brain, subdural, and bone. Each windowed version of the image was normalized, stacked, and saved as a 3-channel composite, producing visually enhanced representations suitable for CNN input. To eliminate irrelevant regions and improve spatial consistency, images were autocropped and resized to a fixed resolution.

To improve robustness and generalization during training, a comprehensive augmentation pipeline was implemented using the Albumentations library. Transformations such as flipping, shifting, scaling, and rotating were applied stochastically, followed by per-channel normalization based on dataset-specific intensity statistics. These transformations ensured the model learned invariant features despite variation in scan orientation or field-of-view.

Finally, the dataset was organized into structured training and test subsets. All processed images were saved in compressed format, with associated label files generated in tabular form. A final validation confirmed the correctness, completeness, and readiness of the dataset. This comprehensive preparation stage resulted in a standardized and clinically faithful image dataset, fully compatible with modern deep learning architectures and optimized for both training performance and diagnostic accuracy.

3.3.1 Data Consolidation and Loading

The initial stage of the data preparation pipeline involved consolidating disparate sources of imaging data and associated metadata into a unified, analysis-ready structure. This step was critical in ensuring that each image slice was accurately and consistently linked to its corresponding clinical label, while maintaining identifier integrity across the patient, study, series, and instance levels. At the core, the dataset consisted of two principal components: a vast collection of DICOM images organized hierarchically by patient and study folders, and a metadata file (`stage_2_train.csv`) containing multi-label diagnostic annotations indexed by unique study identifiers.

Given the hierarchical and distributed nature of the original dataset, the DICOM image collection was first parsed and flattened into a structured tabular format. Each image slice was represented by a single row and mapped to its full file path. From the DICOM headers, essential identifiers such as `PatientID`, `StudyInstanceUID`, `SeriesInstanceUID`, and `SOPInstanceUID` were extracted using the `pydicom` library. In parallel, label data from the CSV file was processed with `pandas`, and indexed by `StudyInstanceUID` to facilitate accurate joins. Once these identifiers were aligned, each image could be associated with its corresponding multi-label vector, forming a comprehensive view of both pixel data and its diagnostic context.

During the consolidation process, multiple validation checks were implemented to guarantee integrity. It was verified that each `SOPInstanceUID` mapped to a unique, accessible image path, and that every labeled entry in the metadata file had at least one corresponding slice in the image repository. Additionally, orphaned or mismatched entries—whether due to missing metadata, incomplete series, or labeling inconsistencies—were identified and excluded to ensure that the dataset remained both coherent and clinically meaningful.

The merged output of this process was a centralized `dataframe` in which each row captured a fully qualified image, inclusive of its file path, DICOM-derived metadata, and encoded label vector. This unified structure served as the foundational dataset used for all downstream stages, including statistical profiling, image transformation, dataset splitting, and model training.

While the logical organization of the dataset was straightforward, the physical handling of the files posed substantial technical challenges. The dataset was initially received as a single compressed ZIP archive of approximately 181 GB in size (0.5 TB unzipped) so we moved to setup a cloud based environment. Despite being compressed, this file remained difficult to upload and manage within cloud environments such as Google Drive. This was primarily due to limitations in upload speed, a large number of underlying files, I/O bottlenecks, and latency. Additionally, factors such as daily bandwidth quotas, file size restrictions, connection timeouts, session timeout and the overhead of syncing nested directory structures further contributed to the challenge.

Our initial approach relied on Google Colab Pro+ and the upgraded 2TB Google Drive plan to facilitate cloud-based preprocessing, but this strategy quickly proved ineffective. We attempted to download the ZIP archive of DICOM files into the local storage of Google Colab and then simply move that file to Google Drive. Despite being compressed, the archive was still large and contained a significant number of underlying files upon extraction. This caused severe performance degradation during the synchronization process. Google Drive's background syncing struggled with the high file count, triggering latency, I/O bottlenecks, and rate-limiting issues. Ultimately, the process failed due to the inability of Google Drive to efficiently sync the large volume of the ZIP file in a reasonable time frame, making it unsuitable for our workflow.

One attempted solution to address the Google Drive syncing problem was to partially download the ZIP archive in segments and incrementally move and reconstruct these parts within Google Drive. While this approach initially appeared promising, it ultimately led to a high rate of file corruption. The fragmented transfer process, combined with synchronization delays and possible interruptions during uploads, resulted in numerous incomplete or damaged DICOM files. As a consequence, the overall data integrity was compromised, leading to poor image quality and rendering the dataset unreliable for subsequent processing.

The sheer volume of data—**more than 874,000 individual DICOM files**—made it impractical to process the dataset entirely within Google Colab. Attempting to download the full ZIP archive into Colab's local runtime storage, extract its contents, and subsequently transfer the extracted files to Google Drive resulted in repeated failures. This was primarily due to **Google Drive's input/output (`I/O`) quota** and I/O bottleneck limitations, which restrict the number of file operations (reads, writes, syncs) that can be performed in a given period. As the DICOM files are small but numerous, the extraction process created a high I/O load that quickly exceeded the allowed quota. Furthermore, the Colab environment itself has limited local storage (~235GB, depending on runtime type), insufficient to hold both the compressed archive and the uncompressed data simultaneously. These combined constraints made this approach unfeasible for managing such a large-scale medical imaging dataset.

Efforts to **decompress the ZIP archive directly within the Google Colab environment** and subsequently relocate the extracted files into Google Drive were met with persistent **runtime failures and I/O bottlenecks**. This was primarily due to the **high cost of handling over 874,000 individual DICOM slices**, which overwhelmed both Colab's temporary storage limits and Google Drive's `I/O` quota system. The transfer process relied on syncing each individual file, a method that is inherently **unscalable** when dealing with a dataset of this magnitude. Google Drive's syncing mechanism is optimized for bulk file updates but not for high-frequency, small-sized file operations typical of medical imaging datasets, resulting in **latency, sync queue delays, and eventual throttling**.

While commercial cloud computing platforms such as **Google Cloud Platform (GCP)** could theoretically accommodate this workload using services like `Cloud Storage` and `Compute Engine`, they presented **significant financial barriers** for continuous, large-scale experimentation, especially in academic or resource-constrained settings.

As a cost-effective and performance-driven alternative, we opted to perform all preprocessing locally on physical hardware, specifically 11th Gen Intel® Core™ i7-11370H running at a base frequency of 3.30GHz. This processor features 4 physical cores and 8 logical threads. Instead of relying on file-based I/O for intermediate steps, we processed the compressed ZIP file directly in memory using Python's `BytesIO` module, which allowed us to stream file contents as byte buffers without writing to disk which significantly speeds up the process. To further accelerate the pipeline, we employed `concurrent.futures.ThreadPoolExecutor` to enable parallel decompression and parsing of individual DICOM slices. This design enabled us to:

- **Validate file integrity** using lightweight DICOM parsers (i.e., `pydicom`).
- **Classify and organize slices** into structured directories on local storage, since we can not store more than 10,000 files in one directory on Google Drive.
- **Bypass the overhead** of repeated cloud-based file operations.

This binary in-memory approach, combined with multithreaded execution, dramatically reduced I/O latency and allowed for high-throughput preprocessing that would have been infeasible using conventional cloud storage tools. Ultimately, this local strategy provided greater reliability, transparency, and control over the data handling process, especially critical when working with medical imaging at scale.

To streamline subsequent access and model training, we implemented a subject-wise grouping strategy by organizing DICOM slices according to their `PatientID`. Specifically, all slices belonging to a single patient were placed into a dedicated directory named after the corresponding `PatientID`. This hierarchical structure not only facilitated efficient indexing and retrieval but also mitigated a known limitation of Google Drive, which tends to degrade in performance or throw errors when managing directories containing a large number of files (typically beyond 10,000).

Once organized, each patient's folder was compressed into an individual archive (e.g., `.zip`), yielding a total of 18,938 compressed files, one per patient. This approach provided multiple benefits:

- **Drastically reduced the raw file count**, converting hundreds of thousands of DICOM slices into a manageable number of archive files.
- **Improved upload stability and reliability**, as Google Drive handles fewer large files more effectively than millions of small ones.
- **Facilitated patient-level batching**, making it easier to perform downstream parallel loading, validation, or processing in training pipelines.

By performing this compression locally before transfer, we minimized not only network transfer volume but also **cloud-side I/O stress**, resulting in a much more scalable and robust upload process for cloud-based storage and sharing.

The **re-upload process to the cloud** required additional optimization, as traditional methods quickly proved inadequate given the size and structure of the dataset. Manual uploading via the **Google Drive web interface** was ruled out early due to its latency, upload timeouts, and limited ability to handle bulk file transfers or nested directories with tens of thousands of files. Additionally, attempts to mount Google Drive via `Google Drive Desktop` or transfer files programmatically using tools like `rsync` and `rclone` resulted in **unacceptably slow throughput**, with transfer rates often bottlenecked by Google Drive's internal syncing mechanism and throttling policies on small file I/O.

To circumvent these limitations, we explored an alternative strategy involving **exposing the local file system to the internet** using **secure tunneling services**. Tools like `ngrok` and `localhost.run` are designed to create **temporary, publicly accessible URLs that tunnel back to a localhost web server** running on a user's machine. This allows external clients to interact with the local machine over HTTP/SFTP as if it were externally hosted.

We initially evaluated `ngrok`, a popular reverse proxy and tunneling service known for its ease of use and quick setup. While functional, its **free tier imposes bandwidth and session duration limitations**, which interrupted transfers during long upload jobs. As an alternative, we adopted `localhost.run`, a lightweight and SSH-based tunnel service that, despite its minimal setup requirements, offered **better bandwidth stability** and **fewer restrictions** for long-lived connections.

Using `localhost.run`, we hosted the directory containing all compressed patient archives via a simple `Python HTTP server` (`$ python3 -m http.server`) and tunneled it securely to a public URL using:

```
$ ssh -R 80:localhost:8000 localhost.run
```

This approach allowed the cloud backend (or any authenticated client) to access the full directory over HTTP and download the archives **directly from the local machine**, bypassing Google Drive entirely.

This technique proved to be **significantly more efficient** and scalable than any direct upload to Google Drive or scripted cloud sync method. It enabled us to achieve **stable transfer speeds**, **resume support**, and complete dataset migration without being constrained by Google's file operation quotas or UI limitations.

Once the compressed patient-level archives were prepared and served via a public tunnel, they were downloaded into Google Colab's runtime environment using command-line tools specifically selected for stability, speed, and automation. The primary utility used for this purpose was `wget`, which provided robust support for resumable transfers, stable retry logic, and straightforward integration with shell scripts. To enable concurrent downloads, we paired `wget` with `xargs`, a Unix command-line tool that can execute multiple commands in parallel. This combination allowed us to issue:

```
$ cat urls.txt | xargs -n 1 -P 16 wget
```

where `urls.txt` contained the list of patient archive URLs, `-n 1` instructed `xargs` to pass one URL per command, and `-P 16` allowed up to 16 parallel download threads. This setup enabled the parallelized acquisition of thousands of archive files, with minimal manual oversight and high fault tolerance—a key requirement for scaling the dataset into Colab in a reproducible way. Importantly, the downloading and extraction processes were **tightly coupled and executed in parallel**, such that **each archive was extracted as soon as it finished downloading** to Google Drive. This streaming-style pipeline eliminated the need for staging all `.zip` files before decompression, thereby saving both time and temporary storage. As each patient's archive was downloaded, it was immediately extracted into its corresponding directory within the mounted Google Drive using a `ThreadPoolExecutor`, allowing for **simultaneous I/O operations across both network and disk**. This architecture enabled continuous ingestion of data—**download, extract, store, repeat**—which proved to be highly efficient for handling the full set of 18,938 patient archives. The result was a scalable and fault-tolerant workflow that maintained a steady processing rate while maximizing resource utilization in the Colab runtime.

In addition to traditional downloading, we also **explored real-time remote file access** by mounting the local archive storage directly into Colab using the `sshfs` protocol. `sshfs` (SSH Filesystem) allows remote directories to be mounted over SSH as if they were part of the local file system. This method was particularly attractive because it enabled **on-demand access to files** without requiring full dataset uploads to Google Drive or Colab storage.

However, in practice, the performance of `sshfs`-based access was **significantly hindered** by two main factors:

1. **Network dependency**: Every read operation (e.g., loading an image slice) required a live, low-latency internet connection. This made the system prone to stuttering or delays during data-heavy operations such as augmentation or batch prefetching in training loops.
2. **JPEG compression overhead**: Many DICOM images had been compressed using JPEG (lossy or lossless). While this format is storage-efficient, it introduces **CPU-bound decompression latency** during every read call, which becomes a bottleneck when accessing files remotely via `sshfs`, especially at high frequency.
3. **Colab Pro+ background execution not utilized**: Because image loading relied on live remote access, we couldn't take advantage of Google Colab Pro+'s ability to continue training in the background. Any interruption in network availability caused the pipeline to stall or fail.

In contrast, once all archives had been bulk-transferred and extracted into Google Drive, the resulting files benefited from local caching by Google Colab's internal file system, yielding faster and more consistent read times. This was especially noticeable when using Google Colab Pro+, which allocates higher-bandwidth mounts and faster disk I/O performance, thereby improving access speeds even for large datasets.

Ultimately, while `sshfs` offered a clever workaround for temporary remote access, its performance characteristics were suboptimal for deep learning workflows that require high-throughput reading of small files. Preloading and unpacking archives inside Google Drive, followed by structured access during training, proved to be the most efficient and reliable strategy.

Upon successful consolidation, the final training dataset structure consisted of 752,802 labeled image slices linked to 21,744 unique studies and 18,938 patients as shown in Table 5. Each patient folder contained one compressed archive of all their slices, uploaded to Google Drive in preparation for training. The resulting organization not only minimized access latency but also preserved data consistency, reproducibility, and modular loading capabilities.

Table 5: Distribution of Photometric Interpretation and Samples Per Pixel

Photometric Interpretation	**Patients**	**Scans**	**Slices**
Testing Dataset	3,518	3,518	121,232
Training Dataset	18,938	21,744	752,802

This consolidation phase was vital in establishing a structured, validated, and cloud-compatible dataset that could serve as a stable foundation for all further stages of modeling. Despite significant logistical and technical constraints, the solution ultimately allowed for scalable, high-speed preprocessing, effective cloud access, and reliable patient-level organization—core requirements for deep learning on clinical imaging data at scale.

In summary, the final and effective solution was designed to overcome the inherent limitations of using cloud environments like Google Drive and Google Colab for large-scale medical imaging workflows. After downloading the original dataset from Kaggle to a local machine, we encountered significant challenges with traditional upload and sync methods—Google Drive's web interface was too slow and error-prone for large files.

To resolve this, as shown in Figure 4 we restructured the dataset locally by grouping and compressing each patient's DICOM files into a single ZIP archive. This not only reduced the total file count drastically but also allowed for patient-wise data management, improving reliability and downstream processing.

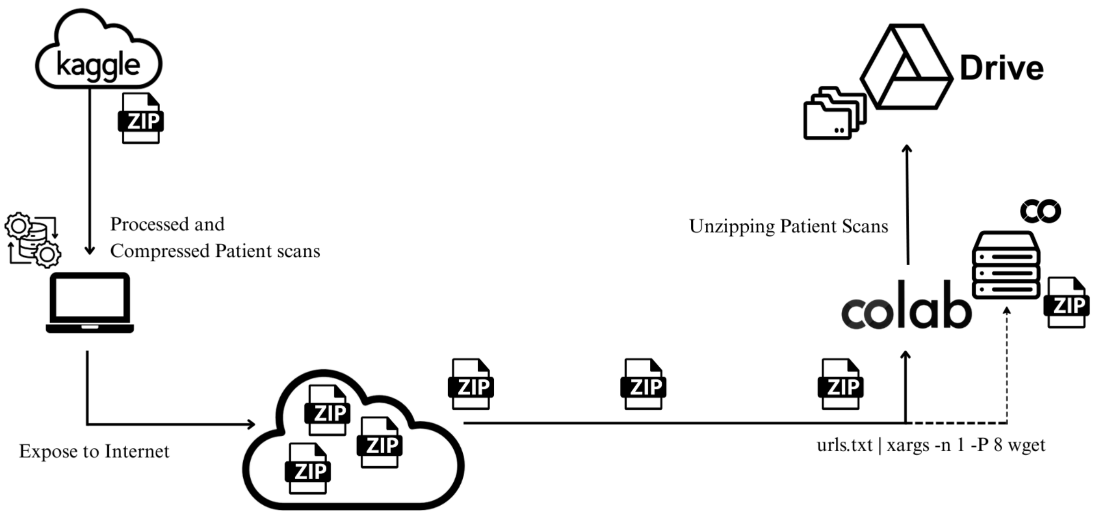


Figure 4: Optimized Data Handling Pipeline From Kaggle to Google Drive.

Instead of uploading these archives directly to the cloud, we used `localhost.run`, a lightweight and free SSH tunneling service, to expose the local directory over the internet. By running a simple HTTP server on the local machine (`$ python3 -m http.server`

8000) and tunneling it via `$ ssh -R 80:localhost:8000 localhost.run`, we generated a public URL that made the archive directory accessible to external systems—including Google Colab—without hosting anything on a remote server.

Inside Colab, we created a `urls.txt` file containing links to all patient archives and used `wget` in combination with `xargs` to download the files in parallel (`xargs -n 1 -P 16 wget`). Critically, each ZIP archive was extracted immediately after download using a threaded Python function, and the extracted DICOM slices were stored directly in the mounted Google Drive. This created a **streamed, end-to-end pipeline**: files were downloaded, decompressed, and persisted in parallel—without waiting for the entire dataset to arrive first from the local machine.

This approach eliminated Google Drive syncing issues, reduced storage pressure in Colab's limited runtime, and provided a scalable, fault-tolerant, and automation-friendly way to move nearly 870,000 patient slices from local hardware into a cloud environment efficiently.

3.3.2 Label Handling and Encoding

In multi-label classification tasks such as intracranial hemorrhage detection, precise label extraction and encoding are fundamental to model performance and evaluation reliability. This section details the steps taken to transform the raw annotation files into a structured, image-level label matrix, suitable for training convolutional neural networks (CNNs) in a supervised learning setup.

The raw labels for the intracranial hemorrhage detection task were originally provided in a CSV file named `stage_2_train.csv`, which adopted a **triplet encoding schema** to represent multi-label classifications. Each row in this file corresponds to a single binary classification target for a specific combination of image and hemorrhage subtype. The critical information was embedded within the `ID` column using a structured naming convention of the form `ID_<ImageID>_<LabelType>`. For example, an entry like `ID_abcdef123_subdural` with a corresponding label value of `1` indicates that the DICOM image identified by `abcdef123` (`SOPInstanceUID`) is positively labeled for the `subdural` subtype of intracranial hemorrhage. More examples were provided in Table 3.

This design effectively flattens a multi-label problem into a long-format dataset, where each image is repeated across multiple rows—once for each of the six possible diagnosis types. These diagnosis classes include: `epidural`, `intraparenchymal`, `intraventricular`, `subarachnoid`, `subdural`, and a composite label `any`, which is set to `1` if the image shows evidence of at least one subtype. Although compact, this triplet-based encoding necessitated additional preprocessing to transform the data into a format compatible with multi-label supervised learning pipelines, particularly for CNN-based models that operate at the image level.

To decode this format, the `ID` column was parsed by splitting each entry into three separate fields: a static prefix (`ID`), the image identifier (`SOPInstanceUID`), and the diagnosis class (`LabelType`). Only the `ImageID` and `LabelType` were retained alongside the associated binary label, forming a long-form table with repeated `ImageID`s across diagnosis types. The dataset was then reshaped (pivoted) into a wide-format **multi-label matrix**, where each row represented a unique image, and each column corresponded to a binary indicator for a specific hemorrhage subtype. This matrix provided a clean, compact, and model-ready representation of the ground truth, enabling image-level predictions across all classes simultaneously.

At this stage, the dataset was still in a **long format**, where each `ImageID` appeared in up to six rows—one per diagnosis class. Before reshaping the data into a multi-label matrix, we ensured label consistency by removing **duplicate rows**, which can arise from redundancy in the labeling process or from merging multiple sources that annotate the same image. Such duplicates, if not handled, could introduce ambiguity or label noise during training. By applying a `drop_duplicates()` operation on the combination of `ImageID`, `Diagnosis`, and `Label`, we enforced **row-level uniqueness**, guaranteeing that each image-diagnosis pair appeared only once in the dataset. This cleanup step was critical for maintaining label integrity before transforming the labels into a model-ingestible structure.

The dataset was then pivoted to form a **multi-label matrix**, with one row per image and one column per diagnosis type. This transformation converts the long-format labels into a wide binary matrix where each cell indicates the presence (`1`) or absence (`0`) of a particular hemorrhage type in the given image as shown in Table 6. The image identifiers were restored to their original format by prefixing each `SOPInstanceUID` with `'ID_'`, ensuring consistency with the naming conventions used during image conversion and preprocessing.

Table 6: Image-wise Binary Labels for Intracranial Hemorrhage Subtypes

Image	any	epidural	intraparench-ymal	intraventr-icular	subarachnoid	subdural	PatientID
ID_000039fa0	0	0	0	0	0	0	ID_eeaf99e7
ID_ffffb670a	1	0	0	0	1	0	ID_4f7414e4
ID_ffff922b9	1	0	0	1	0	0	ID_5964c5e5

To establish a reliable mapping between each DICOM image and the patient from whom it originated, the multi-label image-level annotation dataframe was merged with the DICOM-derived metadata contained in the `train_metadata.csv` file. This metadata includes the `SOPInstanceUID` which served as the primary key for the join operation. By aligning the `Image` column from the label matrix—formatted as `ID_<ImageID>`—with the corresponding `SOPInstanceUID` values, we were able to enrich the label dataframe with patient-level information, specifically the `PatientID` field.

This integration step was essential for ensuring **traceability**, **data integrity**, and **structural alignment** between the imaging and metadata components of the dataset and it used in the creation of the `LOOKUP` path table as will be discussed in bit. With each image now explicitly linked to its originating patient, the dataset became suitable for a wide range of patient-aware analyses, including grouped visualizations, stratified validation, or exclusion-based filtering (e.g., for patient-level deduplication or longitudinal studies). Importantly, this merge preserved the full multi-label annotation structure for each image while augmenting it with critical metadata required for advanced downstream processing and cohort management.

As the final step in the label preparation workflow, the fully processed image-level annotation dataframe—now containing binary labels for each hemorrhage subtype and associated `PatientID` values—was exported to disk in a compressed format as `train.csv.gz`. Compression was applied using gzip to minimize storage footprint and accelerate read performance during training, especially when integrated into I/O-bound data pipelines. This file served as the definitive ground truth source for the supervised learning process, structured in a wide-format matrix suitable for direct ingestion by model training scripts.

In parallel, the test label template file (`stage_2_sample_submission.csv`) was also parsed and cleaned using the same triplet-splitting logic. Although this file contained no ground truth labels, it was reformatted to align with the structure of the training set and saved as `test.csv.gz`, ensuring compatibility during inference and evaluation stages.

By standardizing both files into a clean, efficient, and machine-readable format, this pipeline provided a reliable labeling foundation for supervised multi-label training. Every image was now associated with a complete and binarized label vector, clearly aligned with both its hemorrhage subtypes and metadata, ensuring consistency, reproducibility, and ease of use across downstream modeling and analysis tasks.

3.3.3 DICOM parsing, rescaling, and multi-window conversion

With the label matrix fully prepared and aligned to their corresponding image identifiers, the preprocessing pipeline transitioned to the next critical phase: the **extraction, transformation, and enhancement of raw DICOM slices**. This stage was essential for converting raw scan data into a format suitable for computer vision models, particularly convolutional neural networks (CNNs), which expect consistent, spatially and radiologically informative input.

Each DICOM file was read into memory using the `pydicom` library—a Python tool widely adopted in medical imaging pipelines for parsing DICOM headers and image content. These files contain both the **raw pixel data** of the CT slice and an extensive set of **metadata fields** that define how the pixel values should be interpreted in a clinical context.

To begin the transformation, the raw 2D pixel array was extracted from each slice. However, these values, as stored, do not yet represent true physical measurements. Instead, they are encoded in a scanner-specific format that requires **rescaling to Hounsfield Units (HU)**. To perform this conversion, two key metadata fields from the DICOM header were used: `RescaleSlope` and `RescaleIntercept`. The pixel intensities were transformed using the linear formula as explained in Eq1.

This calibration step is critical because **Hounsfield Units (HU) provide tissue-specific intensity ranges**—for example, air is approximately −1000 HU, soft tissue around 0 HU, and dense bone above +1000 HU. Without this transformation from raw pixel values to HU space, anatomical structures would not be correctly represented, and key hemorrhage indicators could be missed or distorted by the model. By restoring each image slice to its clinically calibrated HU distribution, we preserved the semantic contrast necessary for subsequent windowing operations.

Figure 5 illustrates this transformation process in detail. The left panel shows the original raw pixel array extracted from the DICOM slice (`pixel_array`). On the right, the same slice is visualized before and after applying intensity scaling within the **brain window** (Window Center = 40, Width = 80). The unscaled window reveals poor contrast and saturation effects, whereas the scaled version highlights subtle variations in brain tissue—such as grey-white matter differentiation—that are essential for detecting intracranial abnormalities.

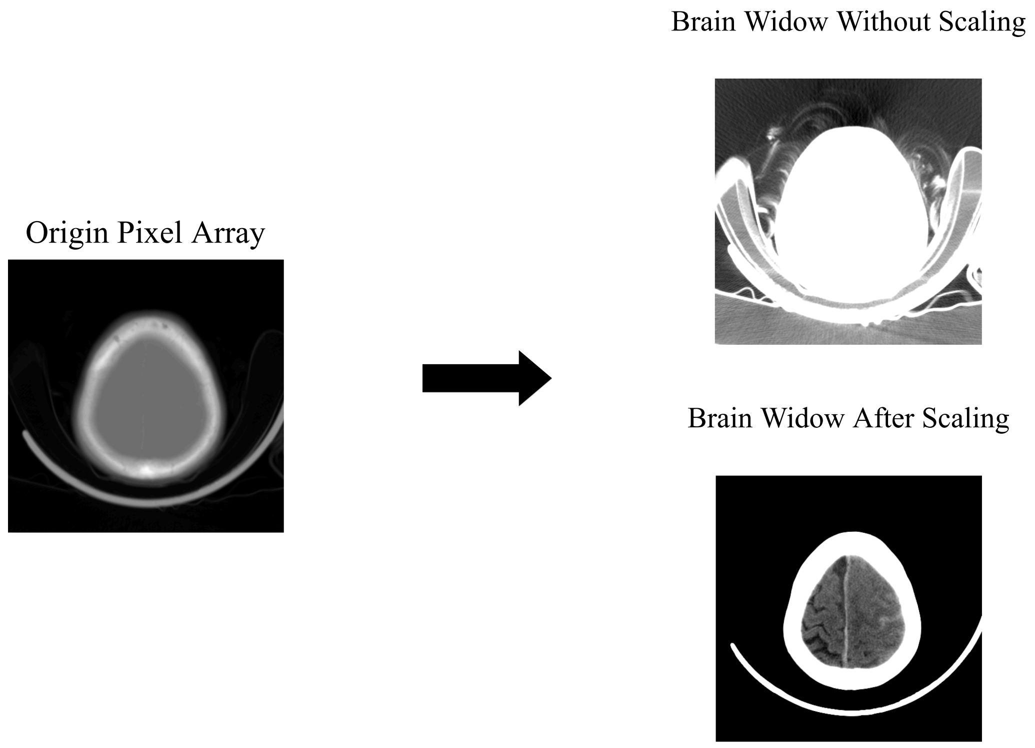
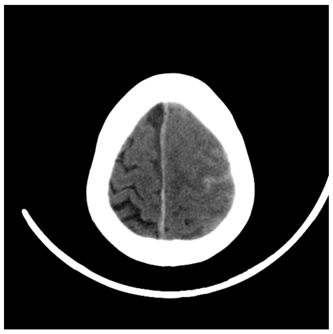
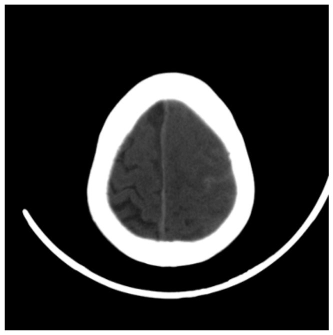


Figure 5: CT Slice Before and After Applying Brain Windowing and Intensity Scaling.

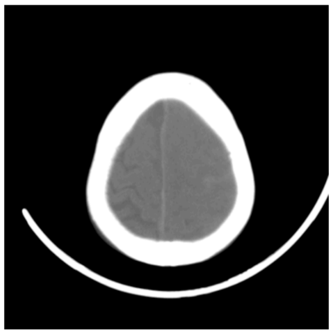
To maximize the diagnostic utility of each CT slice for machine learning, a **multi-windowing technique** was applied—an approach commonly used in radiological workflows and adopted from **Appian's windowing strategy** [34]. Rather than relying on a single contrast window, this technique generates **three separate windowed images**, each tuned to emphasize different anatomical and pathological features relevant to intracranial hemorrhage detection. This mirrors the clinical practice where radiologists toggle between window settings to highlight specific tissue types or abnormalities during interpretation.

The foundation of this process is the **rescaled Hounsfield Unit (HU) image**, which provides a standardized measure of radiodensity for each voxel. From this image, three distinct windowing configurations were applied:

- **Brain Window** (`WindowCenter = 40`, `WindowWidth = 80`):
  Focuses on **gray and white matter differentiation** and the visualization of soft tissue structures within the brain parenchyma. This window is sensitive to subtle changes in brain density that may indicate ischemia or edema, and it helps identify small hemorrhagic lesions embedded in neural tissue:
  

- **Subdural Window** (`WindowCenter = 80`, `WindowWidth = 200`):
  Optimized for detecting **extra-axial hemorrhages**, particularly **subdural hematomas**, which often have a density close to that of soft tissue. This window enhances the contrast between the brain surface and adjacent blood accumulations that may appear flattened against the skull:
  

- **Bone Window** (`WindowCenter = 40`, `WindowWidth = 380`):
  Emphasizes **hyperdense structures such as bone**, making it easier to identify skull fractures, calcifications, or surgical hardware. Though not directly related to soft hemorrhagic tissue, this channel provides important spatial and anatomical context:

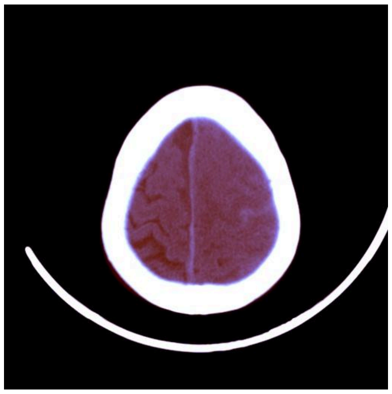


For each configuration, the HU image was **clipped to the range defined by**:

$[WindowCenter\ -\ WindowWidth\,/\,2,\ WindoCenter\ +\ WindoWidth\,/\,2]$

This clipping isolates the intensity range of interest, discards irrelevant extremes, and improves visual clarity. After clipping, pixel values were **normalized to a fixed [0, 1] range**, ensuring consistency across inputs and facilitating gradient-based learning during model training. Each windowed image was processed independently, preserving its unique contrast emphasis.

The three resulting grayscale images were then **stacked along the third axis**, forming a **3-channel composite image** analogous to an RGB color image, where each "color" channel corresponds to a different anatomical focus. This composite allowed the model to **simultaneously access complementary diagnostic information**, effectively capturing the nuance and variability of radiological interpretation in a structured and learnable format.



By integrating this clinically grounded preprocessing strategy into the pipeline, the dataset was transformed into a format that reflects **both the complexity of real-world imaging and the expectations of convolutional neural networks**, enhancing the model's ability to detect diverse patterns of hemorrhage with spatial and radiodensity awareness.

Before finalizing the preprocessed dataset, each 3-channel composite image underwent an additional **post-processing step** to ensure **numerical stability**, **cross-library compatibility**, and **efficient integration into convolutional neural network (CNN) architectures**. While the original windowed channels were normalized to floating-point values in the `[0.0, 1.0]` range, this format is not always optimal for file storage or real-time loading within machine learning pipelines.

To standardize the format and optimize performance during training, the floating-point pixel values were **scaled linearly to the `[0, 255]` range**, transforming the normalized data into an 8-bit integer domain. This step effectively restored a traditional image-like format that aligns with the input expectations of many deep learning models and computer vision libraries, especially those pre-trained on datasets such as **ResNext**, which operate on `uint8` (unsigned 8-bit integer) image tensors.

Following the rescaling, each composite image was explicitly cast to `uint8` —a widely adopted format that ensures:

- **Low storage overhead** compared to floating-point images,
- **Fast decoding and batch loading**, particularly when using image generators or data loaders that rely on PIL, OpenCV, or similar libraries,
- **Compatibility with OpenCV functions**, especially `cv2.imwrite`, which was used to save each image in `.jpg` format.

Storing the images in JPEG format provided a balance between compression and fidelity, making the dataset **portable**, **space-efficient**, and **compatible with virtually any training pipeline**. Moreover, saving images as `.jpg` files allowed for easy dataset inspection, visualization, and augmentation using common tools.

This final step in the preprocessing pipeline not only prepared the data in a standardized, CNN-ready format, but also ensured that downstream model training could proceed without numerical inconsistencies, file format issues, or excessive disk usage.

The use of **multi-windowed input representation**, as opposed to a single-channel grayscale input, offered substantial modeling advantages. By providing the model with three distinct radiological views of the same anatomical slice—brain, subdural, and bone windows—the network could learn to identify features that may be invisible or ambiguous under a single contrast setting. This was particularly impactful for cases involving **co-occurring or spatially overlapping hemorrhage types**, where multiple density regions must be interpreted simultaneously. The composite format enriched the model's feature space by encoding tissue-specific contrast information directly into the input channels.

By the end of this stage, the entire dataset of raw DICOM slices had been **transformed into a standardized, spatially consistent, and anatomically informative 3-channel JPEG format**. These images formed the **final input dataset** for training and evaluation, offering both the compactness required for high-throughput learning and the radiological depth necessary to support accurate multi-label classification in the context of intracranial hemorrhage detection.

3.3.4 Image Cleaning, Autocropping, and Resizing

Following intensity calibration and the construction of multi-channel windowed images, an additional series of **spatial normalization and cleaning operations** was necessary to prepare the dataset for effective model training. Although the images had been enhanced radiologically through Hounsfield Unit conversion and Appian's tri-windowing strategy, the resulting slices still exhibited significant **spatial inconsistencies** due to the heterogeneity of the source data.

This variability stemmed from differences in **scanner hardware**, **field-of-view configurations**, and **acquisition protocols** across imaging centers. As a result, many CT slices contained large margins of black or near-zero padding around the brain—typically outside the circular scan region, as shown in Figure 6. These **uninformative border areas**, which may appear as uniform black pixels, are not clinically relevant but are retained in the raw DICOM projections due to the standardized rectangular matrix format used in medical imaging.

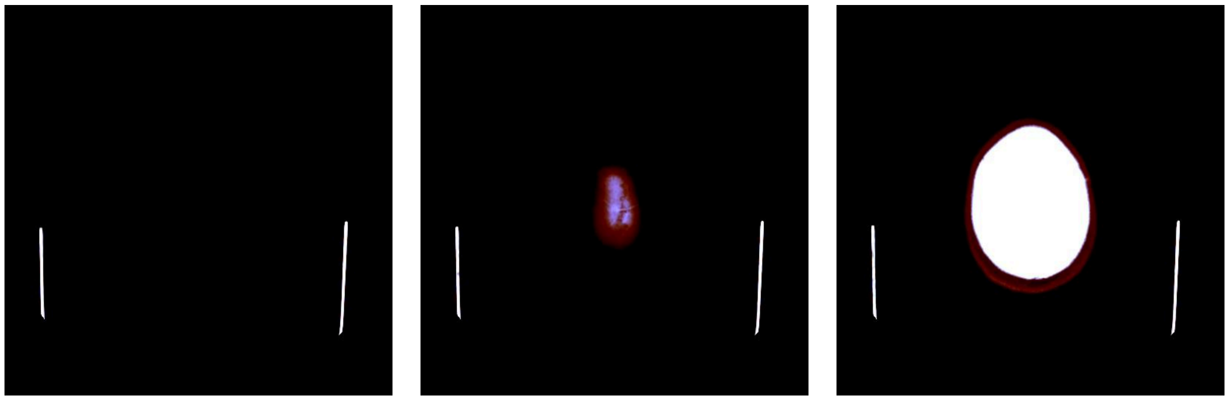


Figure 6: Uninformative Black Margins Retained in Raw DICOM CT Slices.

If left unprocessed, these background regions would contribute unnecessary input dimensions, leading to:

- **Wasted computation** during convolution,
- **Dilution of feature importance**, as the model may allocate filters to background pixels,
- **Increased risk of overfitting** to irrelevant visual patterns or edge artifacts.

To address these issues and **refocus the model's attention onto meaningful anatomical structures**, a robust post-windowing cleaning step was introduced. This included **auto cropping**, background thresholding, and final **resizing** operations to standardize the field-of-view, remove visual noise, and normalize spatial dimensions across the dataset. These refinements not only enhanced image clarity but also ensured that downstream models would operate on consistently framed input—centered around the brain and free of scanner-specific padding artifacts.

The primary objective of this procedure was to isolate the brain region from its surrounding black or near-zero padding—commonly introduced during scan acquisition or digital projection—without losing any diagnostically relevant information.

The process began by operating on the **3-channel windowed composite image**, which had been previously generated to emphasize different tissue contrasts. To simplify margin detection across multiple channels, the image was collapsed into a single 2D intensity map by computing the **channel-wise maximum projection**. This operation retained the strongest anatomical signal at each pixel location, ensuring that the most prominent structure (e.g., bone, soft tissue, or hemorrhage) was preserved during boundary estimation.

Next, a **binary mask** was generated by thresholding the projected image. A global intensity threshold set at zero was applied to identify the spatial extent of the "foreground" region. This foreground is assumed to correspond to actual anatomical content, while values below the threshold typically represent uniform background or scanner padding.

Using this mask, the algorithm then identified the **minimal bounding box** that fully enclosed all pixels above the threshold. The original image was **sliced along both axes** to retain only the rows and columns within this bounding region, effectively cropping away the redundant background pixels. This ensured that each image was tightly framed around the head, reducing spatial variance and eliminating irrelevant pixel regions that could distract the model during training.

In cases where the resulting cropped image was **non-square**, additional padding was symmetrically applied to the shorter dimension to enforce a **square aspect ratio**. This step is crucial for compatibility with most convolutional neural networks, which are commonly designed to process fixed-size square inputs (e.g., 224×224, 512×512). Padding was applied using a constant background value (typically zero) to maintain visual consistency and avoid introducing new artifacts.

By applying this auto cropping strategy, the dataset was significantly cleaned of scanner-related spatial noise, improving the **signal-to-noise ratio** and allowing CNNs to focus on meaningful anatomical content with **consistent spatial framing** across all samples.

This approach not only removed dead space around the skull but also reduced computational overhead by focusing the input on the region of clinical relevance. It ensured that spatial attention was centered on the brain tissue itself, rather than on scanner padding, background, or mechanical structures visible at the image periphery.

Once cropped, all images were resized to a fixed resolution using bilinear interpolation. This target resolution was selected based on the input constraints of the chosen CNN backbone and the tradeoff between performance and fidelity. The resizing ensured that all images shared a consistent input shape, simplifying batch processing and enabling efficient GPU acceleration during training as we will see in **3.4 Modeling** section.

The combination of auto cropping and standardized resizing formed an essential stage in the pipeline, enabling clean, size-consistent, and anatomically focused images to be passed to the model. It contributed directly to reduced variance in background content, faster convergence, and improved model robustness across variations in patient anatomy or scan configuration.

3.3.5 Augmentation and Normalization

To improve the model’s generalization performance and to combat the effects of class imbalance, data sparsity, and overfitting, a robust pipeline of image augmentation and normalization techniques was incorporated into the training process. These operations were not applied statically to the dataset during preprocessing, but were instead executed dynamically and on-the-fly during training, ensuring that the model encountered a unique variant of each image on every epoch. This dynamic strategy significantly increases the effective size and diversity of the training dataset without altering the original data distribution or inflating storage requirements.

The augmentation and normalization pipeline was implemented using the `Albumentations` library—an open-source, high-performance image augmentation framework widely adopted in computer vision and deep learning workflows. `Albumentations` was selected for its **exceptional speed**, **GPU-friendly design**, and its **flexibility in composing complex transformation chains** with fine-grained control over the parameters of each operation. It also allows consistent augmentation across both image data and corresponding masks or labels, making it ideal for medical image analysis tasks where pixel-level fidelity must be preserved [35].

In our specific setup, the augmentation pipeline was tailored to simulate a broad spectrum of real-world imaging variations that may arise from scanner inconsistencies, patient movement, or anatomical variability—factors that are underrepresented in highly structured datasets like the RSNA ICH 2019 challenge. Augmentations included **spatial transformations** such as random rotations, flips, and shifts to improve spatial invariance; **intensity and contrast perturbations** to account for scanner-based exposure differences; and controlled amounts of **noise and blur** to simulate resolution variability and imaging artifacts. Each transformation was assigned a fixed probability of being applied per image during training, allowing for stochastic diversity while preserving core anatomical structure.

In addition to augmentations, **image normalization** was also performed as part of the Albumentations pipeline. Each 3-channel windowed image—composed of brain, subdural, and bone contrast views—was normalized by subtracting a fixed mean and dividing by a fixed standard deviation per channel. These values were computed empirically over the training set and served to stabilize gradients during optimization by aligning the input distribution more closely with the initialization expectations of standard CNN architectures.

This augmentation-normalization setup, defined in code using `Albumentations' Compose`, `Normalize`, and transformation primitives, ensured that the model received **statistically diverse yet structurally consistent** image inputs across training iterations. The dynamic nature of this pipeline not only reduced overfitting risk but also improved the model's ability to generalize to unseen scans and imaging conditions—an essential capability in real-world clinical applications where input data variability is often substantial.

Each transformation within the augmentation pipeline was applied **stochastically**, meaning that during each training iteration, a given image had a **probabilistic chance** of undergoing one or more augmentations. This strategy introduced **structured randomness** into the learning process—offering controlled visual variability—while strictly maintaining the **anatomical and diagnostic integrity** of the input data. In the context of medical imaging, such stochastic augmentation must balance **variability** (to improve generalization) with **fidelity** (to prevent distortion of clinically relevant features). Albumentations provided the fine-grained control necessary to achieve this balance, enabling parameterized application of each transformation, reproducibility of randomized outputs (via seeding), and compatibility with PyTorch-based pipelines.

These augmentations were designed not only to simulate real-world imaging variability—such as patient movement, scanner differences, and misalignment—but also to promote **spatial robustness and distributional normalization**, which are essential for stable convergence in convolutional neural networks (CNNs).

At the core of the augmentation pipeline was **geometric transformation**, applied in a stochastic but reproducible manner. Each training image had a **50% probability** of undergoing a `HorizontalFlip`, a simple yet powerful transformation that simulates **left-right anatomical variation**. While human neuroanatomy is approximately symmetric, pathological findings—like hemorrhages—can be lateralized. Horizontal flipping helps ensure the model does not overfit to specific spatial positions of hemorrhagic features and instead learns **side-invariant representations** of pathological tissue as shown in Figure 7.

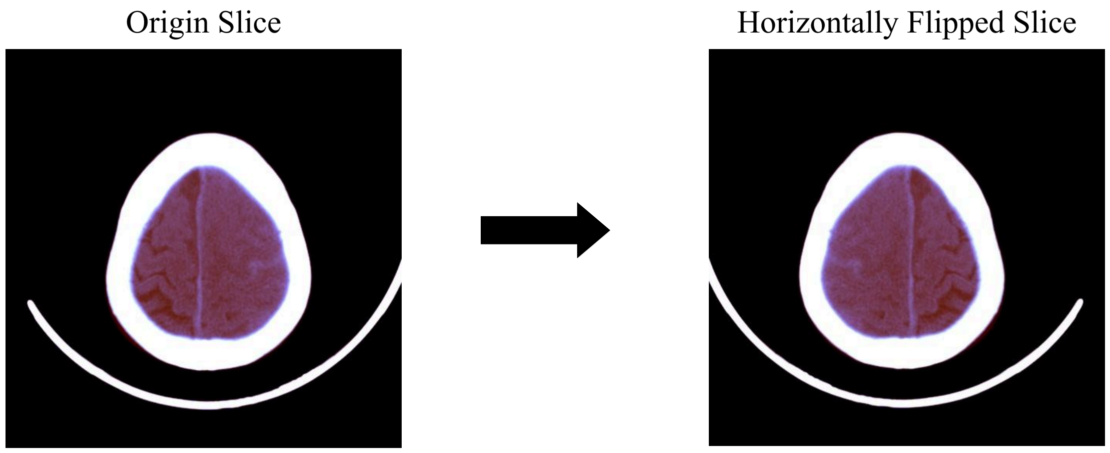


Figure 7: Original and Horizontally Flipped CT Slice for Augmentation.

In parallel, a **30% subset of training images** were augmented using `ShiftScaleRotate`, which applied small, randomized **translations**, **scaling**, and **rotations** to simulate acquisition noise and positional inconsistencies common in multi-center imaging datasets. Specifically:

- **Shifts** of ±5% mimicked slight misalignments during CT scan acquisition or patient positioning variance.
- **Scaling** of ±5% simulated variations in **field-of-view or zoom levels** due to differing scanner settings.
- **Rotations** of up to ±20° accounted for angular misalignment between the patient's head and the scanner gantry—especially useful given that head tilt is not always perfectly corrected during acquisition.

The effects of these augmentations on representative CT slices are illustrated in Figure 8**.**

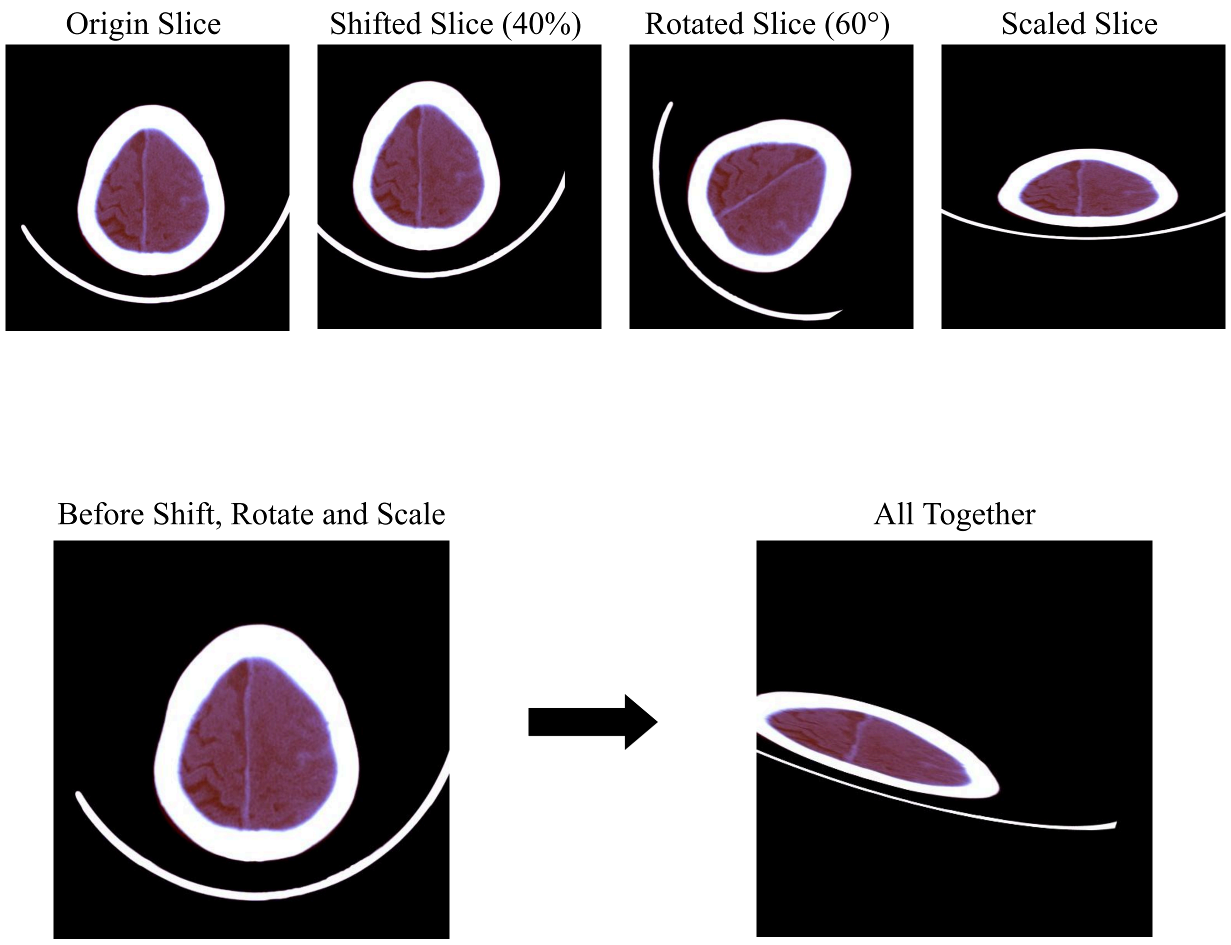


Figure 8: Shift, Rotation, and Scaling Transformations on a CT Brain Slice.

These affine transformations were applied using `cv2.BORDER_REPLICATE` to avoid introducing artificial padding artifacts or border discontinuities—maintaining visual continuity at the image edges even under distortion.

Additionally, with a **50% probability**, each image was passed through a `Transpose` transformation. This operation **swapped the image's x- and y-axes**, introducing **pseudo-rotational diversity** without relying on interpolation, refer to Figure 9. While not a standard practice in radiological workflows, transposition is a **computationally inexpensive** way to expose the model to unusual, yet plausible, spatial configurations—particularly beneficial in datasets where canonical orientation is not guaranteed (e.g., community-sourced or poorly standardized DICOMs).

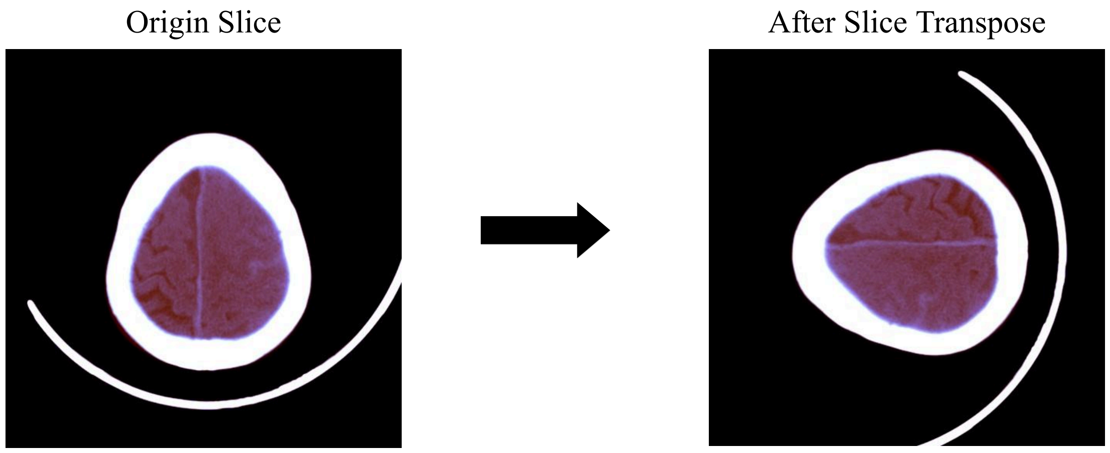


Figure 9: Transpose on a CT Brain Slice.

After the geometric augmentations, each image was passed through a `Normalize` transformation that applied **channel-wise standardization**. The normalization used fixed per-channel **mean** and **standard deviation** values:

- **Mean**: [0.2236, 0.1819, 0.2523]
- **Standard deviation**: [0.3245, 0.3245, 0.3133]

These statistics were computed **empirically from the training set**, after all earlier preprocessing steps—including DICOM intensity rescaling, tri-window contrast enhancement, and post-windowing scaling to `[0, 255]`—had been applied. This means the normalization reflects the actual pixel distribution of the **enhanced 3-channel input**, with each channel corresponding to a specific anatomical window (brain, subdural, bone).

The normalization step standardized the pixel intensity distribution across all samples, transforming each channel into a **zero-centered, unit-variance** representation. This is critical for:

- Ensuring **stable and efficient optimization** during training,
- Allowing **faster convergence** by reducing the internal covariate shift,
- Ensuring **numerical compatibility** with CNN architectures like ResNet, EfficientNet, or Vision Transformers that expect normalized inputs based on pretrained statistical priors.

Finally, the `ToTensorV2()` transformation converted the NumPy-based image arrays into **PyTorch-compatible tensors**, preserving the `[C, H, W]` channel format and enabling batch-wise GPU processing. This end-to-end augmentation and normalization pipeline ensured that the model was trained on inputs that were not only anatomically diverse but also

numerically stable and well-calibrated, forming a robust foundation for generalizable learning in the context of medical image classification.

To seamlessly integrate the augmentation pipeline into the training workflow, the full transformation logic was encapsulated within a custom dataset class, `IntracranialDataset`, which extends PyTorch's standard `Dataset` interface. This object-oriented design ensured that **every image sample retrieved during training was processed dynamically**, allowing augmentations to be applied **on-the-fly during each minibatch construction**, with no need for duplicated or preprocessed files on disk.

The `IntracranialDataset` class is initialized with four primary arguments:

- A dataframe (`df`) containing image metadata and label information,
- A file path root (`path`) where the images are stored,
- A boolean flag indicating whether labels are present (`labels`), and
- An optional `transform` object (typically set to `transform_train`) that defines the augmentation and normalization logic using Albumentations.

Under the hood, each image is accessed via the `__getitem__()` method, which is automatically invoked by PyTorch's `DataLoader` during batch iteration. The image filename is first resolved using a **precomputed `LOOKUP` dictionary**, which maps the image ID (`Image`) to its full path, taking into account the **patient-level folder structure**—where each patient has a dedicated subdirectory containing all of their DICOM-derived `.jpg` slices. This hierarchical organization supports **fast, direct access to any given image**, while also enabling potential grouping or filtering at the patient level (e.g., for stratification or longitudinal analysis).

Upon locating the image, the file is loaded into memory using `OpenCV's cv2.imread()` function. If the `AUTOCROP` flag is set to `True`, the image is passed through an `autocrop()` function. This cropping step, applied at load time rather than during preprocessing, ensures that each image remains **spatially compact and anatomically centered**, even under randomized training conditions.

After cropping, the image is resized to a fixed dimension (`480 × 480`) using OpenCV's `cv2.resize`, standardizing all inputs to a consistent spatial resolution compatible with the downstream CNN architecture. This is especially important in architectures like ResNext where input dimensionality must be precisely controlled to ensure alignment between convolutional kernel sizes and feature map resolutions.

If a transformation pipeline is provided through the `transform` argument—such as the `transform_train` or `transform_test` definitions configured via Albumentations—the

`IntracranialDataset` class automatically applies the full **augmentation and normalization sequence** to each image during retrieval via the `__getitem__()` method. This design ensures that **no image is statically augmented or preprocessed on disk**, but instead undergoes real-time transformations **at the point of sampling**, enabling maximally diverse and randomized training inputs without bloating storage requirements.

During training, the dataset is instantiated as:

```
$ trndataset = IntracranialDataset(trndf, path=dir_train_img, transform=transform_train, labels=True)
```

Here, `transform_train` pipeline composed of:

- `HorizontalFlip(p=0.5)`
- `ShiftScaleRotate(p=0.3)`
- `Transpose(p=0.5)`
- `Normalize(...)`
- `ToTensorV2()`

Each transformation is applied **conditionally and independently**, ensuring that every image seen during an epoch is potentially unique. This approach creates a **virtually infinite augmentation space**, which is invaluable for enhancing model generalization—particularly in tasks with high class imbalance or limited pathological variability, such as intracranial hemorrhage classification.

In contrast, during inference or validation, the dataset is initialized using:

```
$ tstdataset = IntracranialDataset(test, path=dir_test_img, transform=transform_test, labels=False)
```

Here, the `transform_test` pipeline is deliberately **non-augmentative**, with both `HorizontalFlip` and `Transpose` explicitly disabled by setting their probabilities—`HFLIPVAL` and `TRANSPOSEVAL`—to `0`. As a result, no spatial transformations or orientation perturbations are applied during testing. This design ensures that predictions are made on **unaltered, anatomically faithful images**, reflecting the model's true performance on standardized inputs.

Despite the absence of geometric augmentations, **per-channel normalization is retained** using the same statistical parameters computed from the training set:

```
$ Normalize(mean=mean_img, std=std_img, max_pixel_value=255.0,
p=1.0)
```

This guarantees that inference-time inputs maintain **numerical consistency** with the distribution of training samples. In addition, `ToTensorV2()` converts the image into a PyTorch-compatible format.

Within the `__getitem__()` method, once the image is loaded and cropped/resized, the transformation pipeline is invoked, and the image is retrieved from the `Albumentations` dictionary-based output:

```
$ augmented = self.transform(image=img);img = augmented['image']
```

If `labels=True` (e.g., for training), the method also extracts the corresponding multi-label target vector from the dataset's dataframe using the predefined `label_cols` list, converting it to a PyTorch tensor and returning the dictionary:

```
$ {'image': img, 'labels': labels}
```

In the test or inference phase (`labels=False`), the method simply returns:

```
$ {'image': img}
```

This dual-mode behavior allows `IntracranialDataset` class to be used interchangeably across **training, validation, and test-time workflows**, eliminating the need for redundant logic or multiple dataset implementations.

The augmentation and normalization strategy was carefully engineered to enhance both the **generalization capability** and **training stability** of the model, particularly in the context of high-variance medical imaging data. During training, each image was subjected to a sequence of **probabilistic, clinically grounded transformations**—including horizontal flipping, affine distortions, and axis transposition—designed to simulate real-world variability in patient positioning, scanner configuration, and acquisition artifacts. These augmentations were applied dynamically, per sample and per epoch, enabling the model to encounter a virtually infinite distribution of spatially perturbed yet anatomically valid inputs. This continuous diversity forced the model to develop **robust, spatially invariant features** that generalize beyond the training set while avoiding overfitting to rigid spatial priors.

In contrast, the test-time pipeline was intentionally conservative, disabling geometric augmentations and preserving the **original spatial fidelity** of the image slices. Only standardization via per-channel normalization was retained, aligning the test inputs with the statistical distribution of the training set and the expectations of modern CNN architectures. This partitioned design—aggressive variation during training, strict fidelity during inference—ensured both high generalization and **evaluation integrity**, enabling performance metrics to reflect real-world conditions. Together, these strategies formed a critical foundation for reliable model training in a clinically sensitive domain, helping to produce a network capable of robustly identifying intracranial hemorrhage across diverse patients, scanners, and anatomical presentations.

3.3.6 Final Dataset Summary and Readiness for Modeling

Upon the successful completion of all prior preprocessing stages—including metadata extraction, label matrix construction, intensity normalization, and multi-channel image generation—the dataset was consolidated into its **final, model-ready structure**. This structure was explicitly designed to support **efficient, reproducible, and high-throughput training** of convolutional neural networks in the PyTorch ecosystem. All data components were stored in disk-friendly formats and indexed via metadata-aware loaders, ensuring minimal runtime overhead and full compatibility with batched data streaming across GPUs.

The finalized dataset was partitioned into three distinct subsets: **training**, **validation** (actually, it is excluded in model training), and **test**. These partitions were constructed with strict attention to internal consistency: all images were resized to fixed dimensions, normalized with the same statistical parameters, and stored in a unified JPEG format. Image identifiers were harmonized across all subsets using the centralized `LOOKUP` mapping established during preprocessing, ensuring referential integrity between image files and their associated label vectors. This eliminated the risk of mismatches, redundancies, or disjointed metadata—a critical requirement for medical datasets where diagnostic alignment must be exact.

The **training set** contained the bulk of labeled examples. Each image was represented as a **three-channel composite tensor**, derived from contrast-enhanced windowed views of a single DICOM slice. Labels were provided in the form of **multi-hot binary vectors**, each indicating the presence (`1`) or absence (`0`) of the six target hemorrhage types: epidural, intraparenchymal, intraventricular, subarachnoid, subdural, and the composite "any" label. These vectors were directly linked to the source images via the `SOPInstanceUID`, which had previously been merged with patient-level metadata to ensure clinical traceability and structural alignment. The format was designed for direct ingestion by PyTorch's `DataLoader`, with full support for shuffling, multiprocessing, and distributed sampling.

The **validation set**, while smaller, adhered to identical formatting and preprocessing logic. It was held out from training to serve as an unbiased evaluation set for monitoring convergence, tuning hyperparameters, and performing early stopping. Its isolation from augmentation during inference preserved fidelity, allowing for accurate performance estimation without artificial inflation from data transformations.

The **test set** contained unlabeled images prepared exclusively for inference. These were drawn from the RSNA competition's original `stage_2_sample_submission.csv`, parsed to match the training image format, and passed through the same preprocessing pipeline (excluding any augmentations). Each image retained the same spatial and statistical characteristics as the training set, ensuring that the model would operate on **structurally identical inputs** across all phases of its lifecycle.

The validation and test subsets were structured to mirror the training set in format and statistical profile, but without the application of any stochastic augmentations. This exclusion preserved the **anatomical integrity and reproducibility** of the inputs, which is essential when measuring model generalization or generating competition submissions. By maintaining identical preprocessing operations—sans augmentation—the evaluation data remained **semantically and numerically aligned** with the training distribution, ensuring that inference was not confounded by artificially induced variance.

For optimal storage and loading efficiency, the dataset was organized using a **hybrid structure**: images were stored in `.jpg` format within patient-specific folders, while lookup information—such as relative paths and, where applicable, label vectors—was maintained in **compressed CSV index files** (`train.csv.gz`, `test.csv.gz`). This setup allowed PyTorch's `DataLoader` to perform **high-speed random access**, support **multiprocessing**, and preserve clear mappings between image files and their corresponding metadata without overloading memory or storage. This design enabled scalable experimentation without sacrificing traceability or I/O performance.

All images were successfully normalized, formatted to the correct tensor shape, and stored with consistent metadata. The label distribution was preserved from the original source, ensuring that the modeling task accurately reflected the real-world class imbalance of intracranial hemorrhage types. Moreover, the average pixel intensity distributions across the three window channels aligned with expectations, and no corrupted or incomplete samples remained.

With the preprocessing phase complete, the dataset was now clean, balanced (by design, or accounted for in loss weighting), and fully standardized. This prepared environment enabled seamless integration with deep learning models, efficient experimentation, and reproducible evaluation, all while maintaining clinical and technical validity.

3.4 Modeling

The fourth stage of the CRISP methodology involves various modeling techniques being selected and applied to the prepared data. Tasks include selecting modeling techniques, designing tests, building models, and assessing model performance.

This section outlines the core modeling architecture responsible for transforming spatially rich brain CT slices into temporally contextualized, multi-label predictions of intracranial hemorrhage. The modeling framework was built as a **two-stage hybrid pipeline**, combining a powerful 2D CNN-based feature extractor (ResNeXt101-32x8d) with a custom-designed bidirectional LSTM sequence model. This modular approach allowed for independent optimization of spatial and temporal components, ensuring both anatomical detail preservation and cross-slice relational reasoning.

The CNN backbone was used to extract high-dimensional feature embeddings from each DICOM-derived 3-channel image. These embeddings were further enriched with delta-based temporal augmentation—capturing first-order differences between adjacent slices—before being passed into the LSTM. This strategy enabled the model to not only observe the internal content of each slice, but also model the **directional progression of anatomical features**, such as the appearance or spread of hemorrhagic regions through the axial volume.

To enhance model robustness and interpretability, the sequence model incorporated a **stacked BiLSTM architecture**, residual connections, and projection layers. Importantly, embedding generation was performed across multiple CNN training epochs to produce temporally diverse feature representations. Each version was then processed through an independently trained LSTM, and their outputs were ensemble-averaged at inference time—ensuring that predictions reflected a consensus across different stages of CNN convergence.

The final model emitted per-slice sigmoid outputs across six hemorrhage classes, fully aligned with the RSNA dataset structure. These outputs were flattened and serialized into a leaderboard-compatible format using a custom submission utility. Evaluation adhered to the RSNA's **weighted multi-label logarithmic loss**, with higher emphasis placed on the "any" label. Together, this design delivered a scalable, interpretable, and clinically aligned architecture capable of volumetric inference from raw imaging data.

4.1 ResNeXt Architecture Overview

This project adopts a hybrid modeling approach that leverages both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to capture spatial features from individual CT slices and temporal dependencies across sequences of slices. The modeling pipeline is organized into four key stages:

1. **CNN Training:** A ResNeXt-101 model is trained as a standard image classifier using 2D CT image slices to learn spatial features from the images.
2. **Feature Extraction (Embeddings Extraction):** Embeddings are extracted from the trained ResNeXt model, specifically from the Global Average Pooling (GAP) layer, to serve as a compact representation of each slice.

3. **Sequence Construction:** Using DICOM metadata, the extracted embeddings are grouped and ordered into sequences.
4. **LSTM Training:** Thes sequenced embeddings are used as inputs to train a Long Short-Term Memory (LSTM) network, which captures the temporal structure of the CT scan.

3.4.2 CNN Training: ResNeXt-101 Model Architecture

The ResNeXt model was utilized as a backbone classifier. A backbone classifier is typically a pretrained deep neural network that serves as the foundation for extracting meaningful representations from input data before passing them to the classifier head (i.e,the fully connected layers for prediction). In this study, the primary role of the ResNeXt model is to extract features from input CT slices.

ResNeXt, short for *Residual Networks with External Transformations*, is a CNN model introduced in 2017 by researchers at FaceBook AI Research (FAIR) as an extension of the ResNet architecture [36]. It's evolution builds upon prior architectures:

1. **ResNet (2015):** Introduced the concept of residual learning through a technique called skip connections, which connects activations of a layer to further layers by skipping some layers in between. This approach helped mitigate the vanishing or exploding gradients problem, making it possible to train very deep networks.

2. **Inception models (2014-2015):** The idea came from the fact that each layer extracts information from previous layers to get useful information. However, each layer type extracts different kinds of information. The output of a 5x5 convolutional kernel tells us something different from the output of a 3x3 convolutional kernel, which tells us something different from the output of a max-pooling kernel and so on. Inception networks process the same input through multiple convolutional paths in parallel and concatenate the results, capturing richer feature representations.

3. **ResNeXt (2017):** Merged the strengths of ResNet's residual connections with Inception's multi-path design. Within each residual block, ResNeXt introduces multiple parallel transformations, enhancing the model's ability to extract multi-scale features.

Before ResNeXt was developed, there were two major trends in improving CNNs, which are either making networks deeper like in ResNets or making networks wider like in Inception models. However, deeper networks can be harder to train and inception networks are hard to design. The key innovation of the ResNeXt model lies in the introduction of a parameter called Cardinality which controls the number of parallel paths within each residual block.

The fundamental building block of ResNeXt is a set of transformations that share the same topology, known as a *residual block*. A central feature of ResNeXt is the use of grouped convolutions. In contrast to standard convolutions, which operate over all input channels together, grouped convolutions divide the input channels into multiple groups, and each group is convolved with its set of filters. These transformations within a block are structurally identical but have their own parameters, allowing them to learn and process different aspects of the input data independently. Cardinality in the context of ResNeXt refers to the number of parallel paths or groups within each ResNeXt block. This approach enables the model to learn multiple, diverse feature representations in parallel, without a linear increase in the number of parameters or computational cost. ResNeXt also adopts the bottleneck block structure introduced in ResNet. Rather than applying a computationally expensive convolution over a high dimensional input, the bottleneck block employs a sequence of three convolutions: a 1×1 convolution for dimensionality reduction, a 3×3 grouped convolution for spatial feature extraction, and a final 1×1 convolution to restore the original dimensionality. This design reduces both computational complexity and training time, while preserving the model's representational power.

The outputs from all groups are then concatenated, maintaining rich and diverse feature mappings across the network. One of the most compelling advantages of ResNeXt is its scalability. The architecture can be efficiently scaled up by increasing the cardinality, i.e., adding more parallel paths within each block, without a substantial increase in memory or computational overhead. Figure 10 illustrates the ResNeXt architecture, showcasing both a standard residual block and a grouped convolution block with a cardinality of 32 parallel paths, highlighting the structural differences and the concept of grouped convolutions.

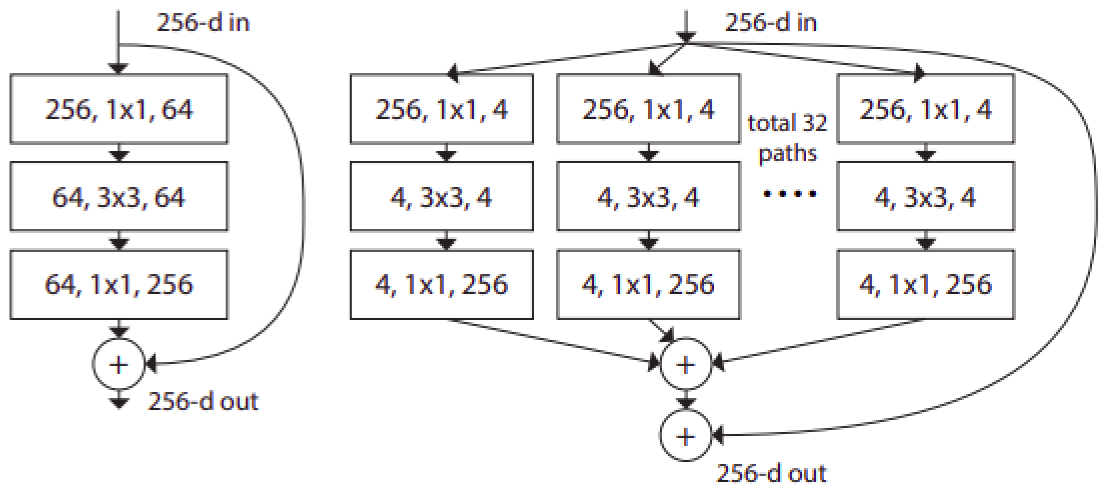


Figure 10: Standard residual block compared to ResNeXt architecture.

3.4.3 CNN Backbone: Feature Extraction Using ResNeXt101

To perform high-fidelity extraction of spatial features from individual CT slices, the feature extractor employed in this project was **ResNeXt-101 (32x8d)**, initialized with weights from the **weakly supervised ImageNet-Instagram checkpoint (`resnext101_32x8d_wsl_checkpoint.pth`)**. This architecture was selected after extensive review of modern convolutional backbones, with a focus on balancing **representation capacity**, **training efficiency**, and **transferability to domain-specific patterns** such as those found in medical imaging.

The version used in this work was **pretrained on weakly supervised data** from a large-scale Instagram corpus as part of Facebook AI's WSL (Weakly Supervised Learning) initiative. This pretraining strategy exposes the model to an exceptionally broad visual distribution, beyond what traditional ImageNet pre training offers. As a result, ResNeXt101_WSL captures **semantic and low-level texture cues** that generalize remarkably well to domains like medical imaging, where pathologies may be small, irregular, or embedded in otherwise normal-appearing tissue. In particular, its ability to encode **non-local dependencies** and **spatial continuity** makes it suitable for detecting hemorrhagic signals that manifest as subtle irregularities in grayscale CT scans.

Unlike architectures explicitly designed for classification tasks (e.g., EfficientNet or DenseNet), ResNeXt101's deep hierarchical structure—with its stacked bottleneck blocks and expansive receptive fields—makes it especially adept at **mid- to high-level feature abstraction**. This is ideal for learning nuanced radiological patterns such as soft-tissue asymmetry, boundary blurring, or hyper densities that span multiple anatomical regions. Moreover, its robust feature maps serve as **general-purpose embeddings** that can be passed to downstream sequence models (e.g., LSTMs) for longitudinal or multi-slice analysis.

In this project, ResNeXt101 was used purely as a **feature extractor**, with its final fully connected classification layer discarded and replaced with task-specific heads tailored to multi-label hemorrhage detection. All convolutional weights up to the penultimate layer were either frozen or fine-tuned in staged training, depending on the learning phase. The output of the model consisted of **deep spatial embeddings**, preserving global contextual information while remaining tractable for downstream modeling.

In sum, `ResNeXt101_32x8d_WSL` provided a **clinically aligned, computationally scalable, and empirically validated backbone** for spatial encoding. It served as the foundational component for transforming raw CT slices into high-dimensional, information-rich feature representations capable of supporting accurate and interpretable hemorrhage classification across a wide range of patients and scan conditions.

Model Adaptation and Training Optimization

To tailor the pretrained ResNeXt101 model for the **multi-label intracranial hemorrhage classification task**, the architecture was modified at the classification head level. Specifically, the model's original fully connected (FC) layer—originally trained for 1000-class softmax classification on large-scale weakly supervised natural image data—was replaced with a new task-specific `torch.nn.Linear(2048, 6)` layer. This new linear layer maps the 2048-dimensional global average pooled feature vector to **six independent output logits**, each corresponding to one of the target hemorrhage subtypes: `epidural`, `intraparenchymal`, `intraventricular`, `subarachnoid`, `subdural`, and the aggregate `any` label.

This modification was performed directly on the model after loading the checkpoint using:

```
$ model = torch.load('checkpoints/resnext101_32x8d_wsl_checkpoint.pth');
model.fc = torch.nn.Linear(2048, n_classes)
```

which ensured that the base ResNeXt convolutional trunk—comprising deep bottleneck blocks and grouped convolutions—remained intact and initialized with pretrained weights, while the classification head was reinitialized to suit the new task. The model was then **moved to the GPU environment** (i.e., `$ model.to('cuda')`) and to leverage CUDA acceleration. Given the availability of multiple GPUs, the model was wrapped using `torch.nn.DataParallel(...)` to **enable parallel training across all devices** (if any).

To further optimize computational efficiency and minimize memory overhead during training, the model was initialized for **mixed-precision training** using NVIDIA's APEX library. This was accomplished by wrapping both the model and the optimizer with `amp.initialize(...)`. Four AMP optimization levels are available and shown in Table 7:

Table 7: AMP optimization levels

Level	Precision Mix	Description
00	- Full precision - FP32	No mixed precision (default training)
01	Mixed precision	Most layers in FP16, safe ones in FP32
02	More aggressive	Convert model to FP16, keep batch norm in FP32
03	- Half precision - FP16	Everything in FP16 **(fast but risky)**

Well, in this project we have used an optimization level of `"O1"` to enable **automated mixed-precision casting** while retaining full FP32 master weights for numerical stability. This setup activated dynamic loss scaling, mitigating underflow risks in low-precision gradient flows, and significantly boosted throughput—particularly important when training on high-resolution medical images with large batch sizes. This configuration allowed the training loop to benefit from **reduced memory consumption**, faster forward-backward passes, and compatibility with modern CUDA-capable hardware accelerators, all while preserving the numerical fidelity required for clinically sensitive feature learning.

During training, the model was configured to operate in **image-level classification mode**, treating each individual CT slice as an independent instance with no temporal or patient-level context. This approach is aligned with the goal of learning **slice-specific diagnostic patterns**, where each image independently encodes visual evidence for the presence or absence of hemorrhage subtypes. Each input image—previously preprocessed to include **three distinct channels** corresponding to clinically optimized window settings (brain, subdural, and bone)—was fed into the modified ResNeXt101 backbone. The image was first processed through a deep hierarchy of convolutional layers, including grouped convolutions and residual bottleneck blocks, culminating in a **global average pooling layer** that distilled the spatial activation maps into a compact **2048-dimensional feature vector**.

This vector captured high-level abstracted spatial representations of the input image, incorporating information about density patterns, edges, textures, and spatial hierarchies—features critical for differentiating hemorrhagic subtypes. The pooled feature vector was then passed through the model's custom classification head, a `torch.nn.Linear(2048, 6)` layer, which produced **six independent logits**, one for each of the target labels.

To guide the learning process during training, a **custom loss function** was formulated based on the **binary cross-entropy (BCE)** criterion, tailored for multi-label medical classification. This function accounted for the fact that each input CT slice could be associated with **multiple hemorrhage subtypes simultaneously**, and thus required independent binary decisions per class rather than mutually exclusive predictions. The output of the model—a six-dimensional vector of raw logits—was passed through a sigmoid activation function to produce **per-class probability estimates** in the `[0, 1]` range. These probabilities were then compared to the binary ground-truth label vector using BCE loss.

The loss was **decomposed into two distinct components**:

1. **`BCE_labels`**: computed across the first five labels, each corresponding to a specific subtype (`epidural`, `intraparenchymal`, `intraventricular`, `subarachnoid`, `subdural`),
2. **`BCE_any`**: computed separately on the sixth output, the `any` class, which is a **composite indicator** of whether any hemorrhage is present in the slice.

To reflect the **greater clinical significance of fine-grained subtype classification**, the two components were **weighted in a 6:1 ratio**, giving greater emphasis to the per-label accuracy while still accounting for the high-level "safety net" prediction from the `any` class. Mathematically, the total loss per sample was computed as:

$$Loss = (6 \cdot BCE_{labels} + 1 \cdot BCE_{any}) / 7$$

This formulation ensured that subtype predictions were prioritized, enabling the model to learn detailed pathological signatures, while still preserving the broader diagnostic signal offered by the `any` class—useful for initial triage or binary classification benchmarks. This design reflects clinical practice, where radiologists must both **detect presence** (global alerting) and **localize subtype** (diagnostic precision).

The model was optimized using the **Adam optimizer**, configured with a fixed learning rate of `2e-5`, selected for its adaptive learning capabilities and smooth convergence behavior in deep vision tasks. The optimizer was initialized over all trainable parameters using:

```
$ plist = [{'params': model.parameters(), 'lr': lr}]; optimizer
= optim.Adam(plist, lr=lr)
```

Training was conducted in **mini-batches of 64 images**, balancing memory efficiency with gradient stability under mixed-precision conditions. Each epoch toggled the model between `.train()` and `.eval()` modes, depending on phase, ensuring proper activation of dropout and batch normalization layers. Within each training epoch, the model iterated over all batches using `trnloader`, performing forward passes, computing loss via the custom criterion, and executing backward propagation using `amp.scale_loss(...)` for mixed-precision scaling. Gradients were zeroed after each update using `optimizer.zero_grad()` to avoid accumulation across iterations.

At the end of every epoch, the **mean loss** across all batches was computed and logged via `logger.info(...)`, enabling detailed monitoring of convergence behavior. To support **reproducibility and checkpoint recovery**, model weights were saved to disk.

Once operating in embedding mode, the model processed each CT slice independently and emitted a fixed-length **2048-dimensional latent representation** that preserved the most salient visual and contextual features learned during classification training. These representations were **dense, non-linear projections** of the original input images, capturing not just local pixel-level features but also **abstracted anatomical structures, spatial relationships, and class-relevant activations**. Because the network had been trained with a clinical objective and exposed to labeled hemorrhagic patterns, its intermediate feature space encoded **semantically rich descriptors** capable of generalizing across slices, patients, and hemorrhage types.

The extracted vectors were used to construct the **temporal input space** for patient-level modeling. Each patient's CT scan—comprising multiple axial slices—was translated into a sequence of ordered embeddings, where each vector corresponded to a single spatial plane in the volume. The consistency in feature dimensionality (2048) allowed these sequences to be **batchable and time-aware**, enabling their direct use as input to recurrent architectures, such as LSTMs or bi-directional sequence encoders. Importantly, this embedding representation provided a **dimensionally compressed view** of each scan, reducing the computational burden of full image-based temporal modeling while retaining the discriminative capacity of the CNN backbone.

From a system design perspective, this decoupling also enabled **asynchronous processing and modular experimentation**. Embeddings could be generated once and reused across different downstream architectures, loss functions, or training strategies without needing to re-run the full ResNeXt101 forward pass—yielding massive computational savings, especially when iterating over LSTM configurations, attention mechanisms, or patient-level stratifications. It also allowed the team to cache and distribute feature banks independently of the original image data, making the entire pipeline more **scalable, reproducible, and cloud-compatible**.

By the end of this phase, a complete volumetric embedding set was available for all samples across training, validation, and test partitions. These embeddings served as the bridge between **static spatial modeling** and **dynamic temporal modeling**, marking the transition from image-level prediction to full-patient volumetric analysis.

3.4.4 LSTM-Based Sequence Modeling

While convolutional neural networks (CNNs) provide exceptional capacity for capturing **local spatial features** and fine-grained anatomical patterns within individual CT slices, they are fundamentally **limited in temporal scope**. By design, CNNs process each image **independently**, extracting hierarchical feature representations without any awareness of **the relative position of that image in a larger sequence**, or the diagnostic relationships between adjacent slices. This presents a critical shortcoming in the context of **volumetric medical imaging**, such as head CTs, where pathologies like intracranial hemorrhage may span multiple slices, evolve gradually, or appear only in relation to their **spatial-temporal continuity** across the axial stack.

To bridge this structural gap and enable the model to understand the **volumetric context** of a brain scan, we introduced a **sequence modeling layer** based on **Long Short-Term Memory (LSTM)** networks. Unlike CNNs, LSTMs are designed to operate over sequences of inputs, retaining memory of previously seen elements while dynamically updating internal hidden states. This architecture is particularly well-suited for scenarios where **temporal or sequential structure is clinically meaningful**—as is the case in CT scans, where the anatomical continuity between slices can indicate lesion shape, location, spread, or trajectory across brain tissue.

We implemented a **bidirectional LSTM** architecture, which processes the slice-level embeddings in both the **forward (inferior-to-superior)** and **reverse (superior-to-inferior)** directions [37]. This bidirectionality allows the model to incorporate both **causal context** (information accumulated from previous slices) and **future context** (information inferred from subsequent slices), mirroring the interpretive approach radiologists take when scrolling through a CT scan. For example, a subtle change in pixel intensity may appear ambiguous in isolation, but becomes diagnostically significant when it appears adjacent to or sandwiched between slices with confirmed abnormalities.

This temporal modeling stage transforms the sequence of **static slice-level embeddings** into a **contextually-aware sequence** of hidden representations, each of which encodes not only the features of a given slice, but also its **position-relative diagnostic meaning** within the full scan. These enriched representations form the basis for final predictions at either the slice or patient level, and serve as the backbone for volumetric reasoning tasks that cannot be resolved by spatial information alone.

Each patient's head CT scan was modeled as an **ordered sequence of axial slices**, capturing the vertical anatomical progression from inferior to superior regions of the brain. For every slice within a given scan, a **2048-dimensional feature vector** was extracted via the pretrained ResNeXt101 backbone, operating in embedding mode. These slice-wise embeddings, derived from the CNN's final pooling layer, represented spatially abstracted descriptors encoding complex visual cues such as textural abnormalities, edge transitions, and hemorrhagic densities. The complete scan was thus represented as a sequence of feature vectors, **structured into a 3D tensor of shape `[T × 2048]`**, where `T` denotes the total number of slices for that patient.

To further enhance the network's ability to capture **temporal transitions and inter-slice dynamics**, each embedding was augmented with **different features**: the **first-order deltas** in both the forward (`Δt+1`) and backward (`Δt-1`) directions. These deltas captured how feature activations evolved between consecutive slices—emulating the diagnostic process of identifying progression, expansion, or regression of pathological structures across axial views. For each slice, this yielded a **triplet of 2048-dimensional vectors**: the original embedding, its forward difference, and its backward difference. These were **concatenated channel-wise** to form a final enriched embedding of dimensionality **6144**, effectively encoding not only the static appearance of each slice but also its **relative change within the volume**. This composite structure introduced **temporal derivatives** directly into the model's input space, increasing its sensitivity to small yet significant shifts across slices.

The resulting `[T × 6144]` sequence was then passed through a **stacked bidirectional LSTM** consisting of two recurrent layers, each composed of forward and backward LSTM cells. This design allowed the model to capture **long-range dependencies** in both directions of the scan, preserving contextual awareness from the earliest to the latest slices—and vice versa. Each LSTM unit updated its hidden state based on both the current enriched slice embedding and accumulated contextual memory, enabling the network to interpret individual slices in light of their **broader volumetric context**.

The temporally contextualized hidden states emitted by the second LSTM layer were then **aggregated per slice** and forwarded through **fully connected (dense) layers**, culminating in **multi-label sigmoid outputs** for classification. This allowed the model to simultaneously assess the presence or absence of each hemorrhage subtype, informed by both the slice's local features and its longitudinal anatomical relationships. This architecture thus operationalized a clinically aligned approach: combining **spatial detail (from CNN)** with **temporal reasoning (via LSTM)** to robustly model complex patterns that emerge only across a sequence of 2D slices.

3.4.5 Embedding Organization and Feature Engineering

Following feature extraction, the generated slice-level embeddings were systematically organized into **per-patient temporal sequences** using metadata fields associated with DICOM volumes. Each slice within a scan was annotated with its corresponding `SliceID` —which is a concatenation of `PatientID`, `SeriesInstanceUID` and `StudyInstanceUID`— and spatial location metadata, and the embeddings were then grouped according to unique patient identifiers. Crucially, the resulting sequences were **sorted using the `ImagePositionPatient` z-axis coordinate**, ensuring that the order of slices followed the anatomical axis from the **inferior (caudal) to superior (cranial)** ends of the brain. This anatomical ordering was essential for preserving the physiological continuity of the scan, allowing the sequence model to learn meaningful spatial dependencies aligned with the brain's actual 3D structure.

The original embedding and its two directional deltas were **concatenated to form a single composite vector of length 6144 per slice**, creating a temporally enriched sequence that explicitly encoded both static content and directional dynamics. This transformation augmented each patient's scan with derivative-aware features, providing the recurrent model with richer temporal cues for distinguishing between static anatomical variations and evolving pathological signals.

During batching, patient scans often varied in slice count (`T`), necessitating a mechanism to standardize input dimensions. To accommodate this, a **custom `collate_fn`** was implemented within the PyTorch data loading pipeline. This function dynamically padded each patient's sequence to match the length of the longest scan in the batch, using zero-vectors as placeholders for shorter sequences. Additionally, a **temporal mask** was generated for each batch item, flagging real vs. padded slices. These masks enabled the LSTM to **ignore padded time steps** during forward propagation and loss computation, maintaining the integrity of temporal modeling while supporting efficient parallel training.

This organization and feature engineering stage established the precise structural and contextual scaffolding upon which the sequence model operated—ensuring that every input carried both **rich intra-slice representation** and **structured inter-slice continuity**, aligned anatomically and computationally for optimal temporal inference.

3.4.6 LSTM Model Architecture

To model the **temporal dynamics** inherent in volumetric brain CT scans, we developed a **custom sequential neural network** architecture centered around **stacked bidirectional Long Short-Term Memory (BiLSTM) layers**. The architecture was implemented in PyTorch via a modular `NeuralNet(nn.Module)` class, providing clean integration with the upstream CNN backbone and downstream training pipeline.

The input to the model consisted of **per-patient sequences of enriched slice embeddings**. During training and inference, patient scans were batched together, yielding an input tensor of shape **`[B, T, 6144]`**, where:

- `B` is the **batch size** (i.e., number of patient scans per training step),
- `T` is the **maximum number of slices** in any scan within the batch (post-padding),
- and `6144` is the **dimensionality of each time step**, encoding both spatial and derivative feature content.

This 3D input tensor was passed into a **multi-layer BiLSTM encoder**, designed to process the entire sequence in both temporal directions. The bidirectional structure allowed the model to **contextualize each slice embedding** using information from both earlier and later slices — a clinically significant property when subtle findings may depend on symmetrical context or non-local patterns across the scan. In practical terms, each LSTM cell processed its time step while maintaining internal hidden and cell states, allowing the network to capture both **local continuity** and **volumetric global trends**.

The stacking of multiple BiLSTM layers introduced **depth to the temporal modeling**, allowing more abstract representations to emerge as information flowed upward through the layers. This stacking was complemented by **residual connections** and selective feature fusion mechanisms (described in the next subsection), which preserved gradient flow, mitigated vanishing memory issues, and enabled the network to retain lower-level signals alongside more abstract representations.

This architecture effectively bridged the gap between **static CNN-based feature extraction** and **dynamic volume-aware modeling**, giving the network the ability to reason across full CT volumes while respecting their intrinsic anatomical order and radiological progression.

Spatial Dropout for Regularization (Excluded)

To promote **robust generalization** and mitigate the risk of overfitting to spurious patterns in high-dimensional feature sequences, the model architecture incorporated a **`SpatialDropout`** layer as the first operation applied to the input sequence. This layer was implemented directly after the `[B, T, 6144]` embedding input and before the LSTM layers, and served as a **form of structured regularization** specifically adapted to sequential modeling of volumetric data.

Unlike standard dropout, which randomly zeroes individual elements in the input tensor independently across time and feature axes, `SpatialDropout` (also known as `Dropout2D` in some frameworks) functions by **dropping entire feature channels** — i.e., it zeros out entire rows across the feature dimension **consistently across all time steps** for a given input sequence. This channel-wise dropout pattern is **applied uniformly across the entire temporal dimension**, meaning if a certain channel is dropped for one slice, it is dropped for every slice in the same sequence. This mechanism prevents the LSTM from becoming over-reliant on specific input dimensions and **forces the network to learn redundant, distributed encodings** across multiple slices.

In the context of modeling enriched CT sequences, where each slice embedding includes tightly coupled information from static appearance and first-order temporal deltas, certain feature dimensions may become **dominant predictors** for specific classes—especially under class imbalance or sparse signal conditions. Without proper regularization, the model may overfit to these high-salience channels, leading to degraded generalization performance on unseen patient data. Spatial dropout addresses this by **disrupting co-adapted feature pathways**, compelling the LSTM to **draw information from a wider set of representations** and improving resilience to input noise or anatomical variation.

Moreover, since the embeddings are derived from a high-capacity CNN backbone, they inherently exhibit **feature-level correlation**, especially in early training epochs. Applying dropout at the spatial channel level explicitly breaks these correlations, reducing the risk of redundancy propagation through the LSTM's recurrent memory. Empirical evidence from natural language processing and medical sequence modeling supports that spatial dropout significantly improves **temporal model regularity**, especially when dealing with long-range dependencies over fixed-length sequences.

In our implementation, this layer was parameterized with a dropout rate tuned empirically (e.g., `p = 0.3`), and placed directly within the `NeuralNet(nn.Module)` class constructor. It introduced **no additional trainable parameters** and had negligible computational overhead, yet played a critical role in **stabilizing training dynamics**, enhancing cross-patient generalization, and preparing the LSTM for robust sequential interpretation of enriched CT embeddings.

Bidirectional LSTM Stack

At the heart of the temporal modeling framework lies a **two-layer stack of BiLSTM units**. This dual-layer design enables the model to first encode **primitive temporal features**, such as changes in local anatomy or radiodensity across slices, and then abstract those signals into more **complex patterns**, such as evolving pathological trends or inter-regional relationships across the brain volume.

The first BiLSTM layer receives as input the temporally padded batch of augmented embeddings, structured as a tensor of shape `[B, T, 6144]`.

Each LSTM unit in this layer is configured with a **hidden size of 2048** per direction. Given the bidirectional nature, this results in a **concatenated hidden state of size 4096 per time step**, yielding an output tensor of shape `[B, T, 4096]`. This representation fuses **past and future temporal signals** for each slice, allowing the model to jointly reason over both inferior (caudal) and superior (cranial) anatomical context.

This intermediate sequence is then fed into a **second BiLSTM layer**, architecturally identical in structure but now operating on the already contextually enriched sequence. The second layer performs a **deeper temporal abstraction**, learning meta-patterns over the previously encoded transitions—such as co-occurrence of lesion progressions or long-range anatomical motifs—producing another `[B, T, 4096]` output. This layered design allows for a **hierarchical modeling of temporal relationships**, enabling the model to refine its understanding of complex intra-scan dynamics across slices.

Both BiLSTM layers are instantiated with `batch_first=True`, ensuring that all tensors follow the `[B, T, F]` format, fully compatible with PyTorch's `DataLoader` batching conventions and custom collate functions. Importantly, the LSTMs are configured in **non-stateful mode**, meaning hidden states are reset at the start of each new batch. This ensures clean gradient flow and avoids contamination of temporal memory across unrelated patients—a critical consideration in clinical sequence modeling where scans must remain temporally isolated.

This stacked bidirectional design forms the **temporal backbone of the volumetric model**, transforming enriched static embeddings into dynamic, context-aware representations that serve as the basis for downstream classification and interpretation.

Linear Transformation and Residual Enhancement

To deepen the network's representational capacity and further refine the temporally contextualized features emitted by the bidirectional LSTM stack, the architecture incorporated **two additional fully connected (linear) layers**, each applied to the intermediate outputs of the LSTM sequence. These transformations served as **learned projections**, enabling the model to re-encode the temporal activations into **new, task-adaptive feature spaces** where class-discriminative structure could be more cleanly separated from irrelevant temporal noise or residual correlation.

The first projection layer, **`Linear1`**, was applied directly to the output of the **first BiLSTM layer** (shape `[B, T, 4096]`). This linear layer was followed by a **ReLU non-linearity**, introducing both **dimensional re-scaling** and **non-linear activation dynamics**. Its role was to emphasize certain activations while suppressing others, allowing the model to learn **interpretable intermediate representations** of shallow temporal trends—such as transient radiological events or inter-slice texture discontinuities—that might otherwise be lost in deeper recurrence.

Similarly, a second projection layer, **`Linear2`**, operated on the output of the **second BiLSTM layer**, which had already undergone a higher-order temporal abstraction. This linear block performed an analogous transformation, again followed by a ReLU activation, to adaptively map the deep LSTM features into a refined embedding space more aligned with the final classification objective. By projecting both LSTM stages independently, the model preserved **multi-scale temporal reasoning**, capturing short-term slice-level dynamics and long-term volumetric patterns in parallel.

To integrate these diverse sources of temporal and spatial information, the model then performed an **element-wise summation** of five key feature pathways:

1. The output of the **first BiLSTM layer**,
2. The transformed output from `Linear1`,
3. The output of the **second BiLSTM layer**,
4. The transformed output from `Linear2`,

5. And a **skip-connected projection of the original input embedding**, reintroduced after linear mapping to match dimensionality.

This residual enhancement strategy functioned analogously to **deep residual networks**, where earlier representations are directly reintegrated into deeper layers, helping **preserve low-level information** and improve gradient flow. In this context, the inclusion of the original 6144-dimensional slice embeddings—post projection—ensured that important cues from the original CNN features and temporal deltas remained accessible at the output stage, even after multiple layers of recurrence and abstraction.

This fusion of raw input signals, low-order and high-order temporal dynamics, and nonlinear projections allowed the model to construct a **rich, multi-resolution representation per slice**. The resulting tensor captured static anatomical context, slice-to-slice transitions, and emergent volumetric features in a unified format, forming a powerful basis for final classification and downstream interpretability.

Final Prediction Layer

The final classification stage of the sequence model was designed to **transform the enriched, temporally-aware representation** of each slice into precise multi-label predictions for the six clinically relevant hemorrhage categories. After processing through the stacked BiLSTM layers, nonlinear projection blocks, and residual enhancement pathways, each slice in the padded scan sequence was represented by a **contextually fused feature vector of fixed dimensionality**—specifically, a 2048-dimensional tensor that captured not only spatial and temporal dependencies but also retained information from the original CNN-derived embeddings.

This final vector was passed into a **task-specific output head**: a fully connected linear transformation layer defined as `Linear(2048, 6)`. This layer operated independently on each time step (i.e., each slice), applying a shared set of weights across the batch and temporal dimensions, and projecting the 2048-dimensional feature space into a **6-dimensional output vector per slice**. Each of the six output values represented a **logit**, i.e., the unnormalized score for one of the six binary classification targets. These logits were later passed through a sigmoid activation during training and inference to convert them into **probabilistic outputs** in the range `[0, 1]`, interpretable as **independent class probabilities** for the presence or absence of each hemorrhage type.

From a design perspective, the use of **a shared linear classifier** applied slice-wise allowed the model to **scale across variable-length sequences**, performing dense prediction without requiring a fixed number of slices per patient. This structure also ensured that classification performance was fully **differentiable end-to-end**, allowing gradients to flow from the final layer back through the LSTM stack and into the CNN-derived embedding pipeline.

Crucially, this final head respected the **multi-label nature** of the problem: each class prediction was treated as **independent**, with no softmax applied across the six outputs. This was essential in modeling **co-occurring hemorrhages**, a common clinical presentation where multiple types of bleeding can be simultaneously visible in a single slice. The model, therefore, could detect any combination of labels without enforcing mutual exclusivity, aligning the architecture tightly with the **clinical annotation protocol** and ground-truth labeling schema derived from the RSNA dataset.

In summary, this final linear projection layer translated the model's rich sequence representation into **diagnostically actionable predictions**, completing the transformation from raw DICOM volumes to interpretable per-slice, per-label inferences. It served as the final interface between the deep spatiotemporal modeling infrastructure and the real-world clinical classification task for which the model was designed.

Design Rationale

The architecture presented in this modeling stage was the product of deliberate and systematic design choices, aimed at balancing **temporal modeling power**, **spatial-semantic retention**, and **operational efficiency** across large-scale medical image sequences. Each component—from embedding structuring to output formatting—was chosen to reflect the **clinical complexity of intracranial hemorrhage** detection, while also ensuring the model remained **tractable on high-resolution volumetric data**. The complete pipeline is illustrated in Figure 11, which visualizes the flow from image embeddings to final per-slice predictions.

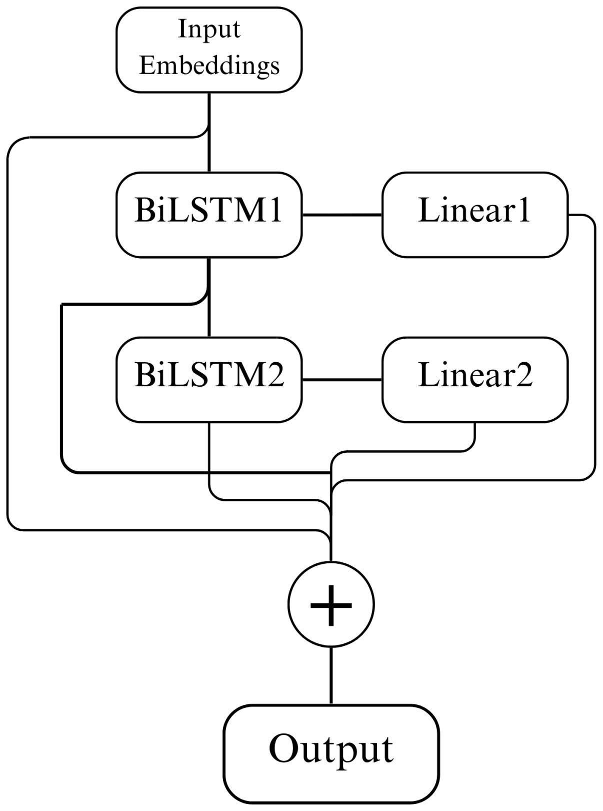


Figure 11: Flow Diagram of the Layered LSTM-Linear Network Design.

First, the use of **bidirectional LSTM layers** enabled the model to capture **full context around each slice**, rather than relying on past information alone. This forward–backward recurrence mirrored how radiologists interpret scans: scrolling both up and down through the brain volume to detect subtle asymmetries or progression patterns. By attending to both **preceding and subsequent slices**, the model was able to resolve **spatial ambiguities**, enhance detection of faint hemorrhagic signatures, and better understand the anatomical progression of complex pathologies.

Second, the incorporation of **delta embeddings**—in the form of forward and backward differences—empowered the model to explicitly reason over **inter-slice anatomical change**. This was critical in a domain where hemorrhages often evolve across slices, and where single-slice observations may lack sufficient contrast. These temporal derivatives allowed the model to detect **directional feature variation**, highlighting emergent bleeding, structural deformation, or discontinuity in intensity that would otherwise be diffuse in static feature spaces.

Third, the application of **residual fusion strategies** ensured that the model's temporal abstractions did not overwrite or erase the rich spatial features originally encoded by the CNN backbone. By summing outputs from both LSTM layers, their corresponding linear projections, and a projected skip connection from the input embedding itself, the model preserved a **multi-resolution representation** at every time step. This architectural design also provided **stability during training**, improving convergence behavior and allowing deeper recurrent stacks without compromising gradient flow.

Finally, the choice to emit **per-slice multi-label predictions** directly from the sequence model—rather than pooling or collapsing over time—was motivated by the structure of the RSNA dataset. Each CT slice in the dataset is **individually labeled** for the presence or absence of six distinct hemorrhage types. By aligning the model output structure with the **granularity of the ground truth annotations**, the design preserved **one-to-one correspondence** between predictions and labels, simplifying supervision and ensuring that every training signal remained **structurally coherent** with its diagnostic context.

Taken together, this architecture offered a principled blend of **temporal awareness**, **structural preservation**, and **deployment efficiency**, tailored to the unique challenges of modeling hemorrhagic brain CT volumes. As shown in Figure 11, it represents a cohesive system where each layer is tuned not only for performance but for **clinical and operational alignment** with the demands of real-world medical imaging AI.

3.4.7 Training Strategy and Optimization

To accurately capture the anatomical progression of hemorrhagic features across CT slices, the LSTM-based sequence model was trained using a temporally enriched and structurally regularized approach. The training strategy was composed of two key stages: first, extracting embeddings from the CNN backbone across multiple epochs; and second, training the LSTM model on each of these independently to produce ensemble predictions that are subsequently bagged.

Multi-Epoch Embedding Generation

The rationale behind generating embeddings at multiple training epochs was rooted in the understanding that deep convolutional networks undergo **nonlinear shifts in their internal feature hierarchies** as training progresses. Early in training, the model captures broad, low-level visual structures—edges, textures, and gradients—whereas later stages tend to refine class-specific activations and abstract representations. By capturing **snapshots of the CNN's evolving internal state** at distinct training intervals, we effectively created **diverse semantic views** of the same anatomical data, each highlighting different representational aspects.

Specifically, the CNN backbone was trained over the entire training set for **three full epochs**, each comprising thousands of slice-level mini-batches. At the **end of each epoch**, the model was systematically transitioned into **embedding mode** by replacing the final fully connected classification layer with a lightweight `Identity()` module. This architectural change short-circuited the forward pass immediately after the final global average pooling layer, allowing access to the **pure 2048-dimensional deep feature embeddings** generated by the ResNeXt trunk—unpolluted by task-specific decision boundaries.

During this phase, all preprocessed 3-channel CT slice inputs were passed through the CNN, now acting as a high-capacity, deterministic feature encoder. The resulting **2048-d vectors per slice** were accumulated for each scan and structured into patient-level sequences based on anatomical order. Importantly, **no augmentation or dropout** was applied during embedding generation, ensuring **deterministic output** for each slice and epoch, and allowing exact downstream reproducibility.

The result was the creation of **three distinct embedding sets**, one per training epoch:

- **Epoch 1 embeddings** captured early-stage feature abstractions, often more sensitive to local texture and low-level spatial structures;
- **Epoch 2 embeddings** reflected mid-stage convergence, balancing generalization and class separation;
- **Epoch 3 embeddings** represented the most refined CNN state, often biased toward the final optimization state but potentially overfit to hard examples.

Each of these embedding sets was **saved to disk in compressed `.npz` format** using NumPy's efficient serialization engine. The file structure was organized per split (`trn`, `val`, `tst`) and indexed by patient ID and slice order. Alongside each file, accompanying **lookup metadata** (e.g., SOPInstanceUID, anatomical ordering, patient label vectors) was stored to ensure seamless integration with downstream `DataLoader` routines and to maintain alignment between the spatial embedding tensor and its corresponding multi-label targets.

This multi-epoch embedding approach provided the **foundation for embedding-level ensembling**, where sequence models were later trained independently on each of the three versions. This introduced **latent diversity** across the LSTM stage without modifying its structure, thereby enriching the learned temporal representations and yielding higher robustness during test-time bagging. More critically, it offered a **statistical snapshot of the CNN's learning trajectory**, preserving multiple representational perspectives of the same input domain—a valuable strategy in clinical imaging tasks where subtle variations in learning dynamics can materially affect prediction outcomes.

Independent BiLSTM Training per Embedding Set

To preserve and fully exploit the **semantic heterogeneity** embedded across different stages of CNN training, each of the three ResNeXt-derived embedding sets was treated as a **discrete, standalone input domain** for LSTM-based volumetric modeling. Instead of aggregating or averaging these embeddings into a unified representation—which would have diluted the temporal uniqueness and masked model-specific biases—each snapshot was passed **independently** through the **entire sequence modeling pipeline**. This decision reflected a core principle of the pipeline's design: maintain **temporal purity and feature identity** from extraction to final classification.

For each embedding version, the process began with the construction of enriched slice-level tensors. Each sequence was augmented with **forward and backward delta vectors**, producing a `[T × 6144]` matrix per patient. These were grouped and indexed by unique PatientID using a new custom instantiation of the `IntracranialDataset` class, which generated batches of variable-length padded sequences. The `collate_fn` ensured zero-padding was applied only where necessary, and **per-slice masks** were computed to prevent padded entries from contributing to gradient updates or loss computation. This ensured both **batch-level efficiency** and **diagnostic integrity**, preserving one-to-one correspondence between real slices and clinical labels.

Optimization employed **Adam** with decoupled weight decay (L2 regularization), paired with a **`StepLR` learning rate scheduler** to progressively reduce the learning rate after plateau detection or fixed epoch intervals. This combination offered strong convergence behavior even in the presence of noisy delta inputs and class imbalance.

Crucially, each LSTM model trained under this strategy generated **a distinct test-time prediction** for every sample in the test set. These predictions were stored separately and later used for **prediction bagging**, where outputs were averaged across all embedding-specific LSTM heads. This diversity—born from independently trained sequence learners—formed the **backbone of the model ensemble**, ensuring robustness against overfitting, enhancing generalization to rare hemorrhage types, and stabilizing performance across scan types and slice counts. It allowed the pipeline to reason not just over anatomy, but over **representational uncertainty**, capturing multiple possible interpretations of the same radiological evidence.

Bagging Over Epoch-Specific Predictions

Following the independent training of three LSTM models, the final prediction stage implemented a **model-level ensemble via prediction bagging**. Formally, for each test-time input sequence (i.e., one patient's scan), every LSTM produced a prediction tensor of shape `[T × 6]`, where `T` is the number of slices (padded as necessary) and `6` corresponds to the six hemorrhage types. These per-slice, per-label probability outputs—denoted as $\hat{y}^1, \hat{y}^2$ and $\hat{y}^3$ —were computed independently by the three LSTMs, each having been trained on its own temporally enriched embedding set.

The final prediction for each slice was computed via simple arithmetic averaging:

$$\hat{y}_{final} = 1/3. \sum_{i=1}^{3} \hat{y}_i$$

This ensemble technique smoothed out **idiosyncratic biases** introduced by CNN snapshot-specific features or LSTM weight initializations. For example, if one embedding set overemphasized hyperdense textures (e.g., bone-window features), while another was more sensitive to subtle soft-tissue gradients, the LSTM trained on each would reflect those preferences in its predictions. **Bagging these outputs allowed us to reconcile these differences**, suppressing outliers and stabilizing class confidence scores.

From a clinical modeling standpoint, this averaging strategy significantly reduced **prediction variance**, which is particularly important when dealing with:

- **Ambiguous radiological cues** (e.g., borderline densities or tiny hemorrhagic traces),
- **Noisy inter-slice changes**, especially in edge slices or motion-corrupted scans,
- And the inherent **class imbalance** in datasets like RSNA, where some hemorrhage types are vastly underrepresented.

The final, bagged output retained the same shape as individual predictions—one probability vector per slice—and was then passed to a post-processing function, `makeSub()`. This function flattened the 3D tensor structure (slice × class) into the **submission-ready 2D format** expected by the RSNA competition: a `.csv` file indexed by `ID`, where each row matched a composite key (`ID_<image>_<diagnosis>`) to a predicted probability value in `[0,1]`.

This final formatting step, a submission-ready file named `submission.csv.gz,` ensured strict alignment with the competition's evaluation pipeline, allowing seamless validation while preserving **slice-level granularity** and **per-class interpretability**, two essential properties in real-world deployment pipelines for automated triage or radiologist-assist systems. Through this careful ensemble design, the model delivered both **statistical robustness** and **clinical fidelity**, translating a complex, modular architecture into **stable, reproducible, and diagnostically meaningful outputs.**

Hardware and Environment

All experiments in this study were conducted using Google Colab Pro+ with access to high-performance `GPUs`, including `NVIDIA T4`, `L4`, and `A100`, and `TPUs`, including `v6e-1 TPU`, `v5e-1 TPU`, and `v2-8 TPU`. The deep learning workflow was implemented using

PyTorch, and training was parallelized using `torch.nn.DataParallel` to leverage multiple GPUs when available. Mixed-precision training was enabled via `NVIDIA's Apex AMP` library `(opt_level="O1")`, which helped accelerate training while reducing GPU memory usage—especially beneficial during long sessions on resource-intensive models.

The pipeline followed a multi-stage process: data loading → CNN-based embedding extraction → LSTM-based sequence modeling. To optimize throughput, data loading was parallelized with `num_workers=8`, and `persistent_workers=True` ensured stable performance over extended training cycles. Embedding extraction from the ResNeXt-101 backbone required approximately **12 hours per epoch** (i.e., 36 hours in total). The ResNeXt training itself took **19 hours, 20 minutes in total**, while training the LSTM model took **13 hours, 30 minutes** for all epochs on `A100` accelerator. These durations reflect the computational demands of processing large-scale DICOM image volumes and modeling their temporal dependencies.

a. Long Training Times and Management

The high resolution and the large amounts of DICOM data and deep model architecture resulted in long training times. This was managed by employing Mixed-precision training via NVIDIA Apex AMP to reduce memory load and accelerate computations. Optimizing PyTorchs DataLoader using multithreading `num_workers=8` and persistent workers `persistent_workers=True` to ensure consistent performance across epochs. Additionally, the split of the workflow into stages, allowing the embeddings to be precomputed and cached, eliminating the need to reprocess images during LSTM training.

b. Limitations of Google Colab

Despite offering powerful hardware, Google Colab presented several limitations that impacted training. As follows:

- **Runtime restrictions** e.g 12-24 hour disconnect limits, which risks interrupting long-running training jobs.
- **File I/O bottlenecks** were encountered when reading from Google Drive, especially with large numbers of small DICOM files.
- **Session instability**, even in the Pro+ tier, could cause unexpected disconnections under high GPU loads.

c. Measures to ensure Training Continuity

To mitigate the risks associated with session interruptions and ensure the continuity of long training processes, several safeguards were implemented:

- **Regular checkpointing** was employed, with model weights and optimizer states saved periodically.
- **Embedding outputs** were cached after each epoch, eliminating the need to reprocess the CNN feature extraction phase during subsequent training runs.
- **Training logs** were recorded after every epoch to facilitate performance monitoring and allow for post hoc evaluation of learning trends.
- **The training pipeline was modularized** into distinct components—ResNeXt training, embedding extraction, and LSTM sequence modeling—allowing each stage to be resumed or re-executed independently without loss of prior progress.

3.4.8 Model Configuration

This section outlines the architectural configuration of the proposed deep learning model, detailing its modular components and their respective roles in the overall pipeline. The design follows a hybrid approach, combining spatial feature extraction with temporal or contextual sequence modeling to capture both local and global patterns within the input data. The model is structured to process volumetric or multi-slice medical images, leveraging state-of-the-art convolutional backbones for representation learning, followed by a sequence-aware module to handle spatial dependencies across slices. The rationale for this configuration is to enable the model to exploit both fine-grained image details and inter-slice relationships, which are critical for accurate classification in medical imaging tasks.

The subsections below describe the individual components of the architecture in detail, including the **feature extractor**, which is responsible for encoding low-level and high-level visual features from image slices, and the **sequence model**, which processes these features to capture spatial continuity and aggregate contextual information for final decision-making.

3.4.8.1 Feature Extractor

The feature extraction stage employs a modified ResNeXt-101 32x8d architecture, pre-trained with weakly supervised learning. The model was adapted for multi-label classification by replacing the final fully connected layer. The key configuration parameters used in this stage are summarized in Table 8.

Table 8: Feature Extractor Configuration

Parameter	Value / Configuration
Backbone Architecture	ResNeXt-101 32x8d (WSL pre-trained with default settings)
Modified Layer	`$ model.fc = torch.nn.Linear(2048, n_classes)`
Device	GPU (`$ model.to(device)`)
Loss Function	Custom weighted `BCEWithLogitsLoss`
Number of Epochs	3
Learning Rate	0.00002
batch size	64
Optimizer	`Adam`

To adapt ResNeXt for the **multi-label hemorrhage classification task**, the final fully connected classification layer (`model.fc`) was replaced with a new `Linear` layer that maps the 2048-dimensional output of the penultimate layer to `n_classes = 6` — corresponding to the five subtypes of intracranial hemorrhage (epidural, intraparenchymal, intraventricular, subarachnoid, subdural) plus the catch-all “any” class. This modification effectively re-purposed the CNN from general object recognition to medical diagnostics.

All model computations were carried out on a **GPU**, taking advantage of CUDA acceleration to enable efficient training across high-resolution CT slices. The model was trained using a **custom weighted binary cross-entropy loss function**, based on `BCEWithLogitsLoss`. This loss was carefully designed to combat the severe class imbalance in the dataset: the “any” class was less emphasized, while the five anatomical labels were weighted more heavily, ensuring that the model did not collapse into predicting only the dominant majority class (“normal”).

The ResNeXt101 model was trained for **3 epochs**, a decision informed by both performance stability and architectural design. Each epoch represented a distinct stage in the CNN’s feature convergence, and after each epoch, **feature embeddings** were extracted and used as input to the downstream LSTM module. This approach not only prevented overfitting but also generated diverse representations that later boosted ensemble performance.

A small **learning rate of 0.00002** was selected to ensure fine-grained updates to the model parameters. This was crucial given the pre-trained nature of the backbone, as a higher learning rate could disrupt already-learned weights. Optimization was performed using the **Adam optimizer**, which is known for its adaptive learning rate capabilities and momentum-like updates, offering a balance between convergence speed and stability across heterogeneous feature spaces.

In sum, the configuration chosen for the feature extractor enabled it to serve as a high-resolution encoder — learning robust representations of CT brain slices and producing temporally aligned features for the subsequent sequence modeling component.

3.4.8.2 Sequence Model

The sequence modeling stage is implemented using a custom bidirectional LSTM-based architecture designed to capture contextual dependencies across ordered image embeddings (e.g., CT slices). The model processes sequence inputs by passing them through stacked LSTM layers and fully connected transformations, incorporating residual connections and non-linear activations. The configuration details are summarized in Table 9.

Table 9: Sequence Model Configuration

Parameter	Value / Configuration
LSTM_UNITS	`2048`
DROPOUT	`0.0`
Learning Rate	`0.00001`
Number of Epochs	`12`
batch size	`4`
embed_size	`6144`, (`patient embeddings`, `deltalead`, and `deltalag` each of `2048`)
Dropout Layer	`SpatialDropout(DROPOUT)`—**Excluded**
Device	GPU (`$ model.to(device)`)
LSTM Layer 1	`nn.LSTM(embed_size, LSTM_UNITS, bidirectional=True, batch_first=True)`
Linear Layer 1	`nn.Linear(LSTM_UNITS * 2, LSTM_UNITS * 2)`
LSTM Layer 2	`nn.LSTM(LSTM_UNITS * 2, LSTM_UNITS, bidirectional=True, batch_first=True)`
Linear Layer 2	`nn.Linear(LSTM_UNITS * 2, LSTM_UNITS * 2)`
Residual Composition	`h_lstm1 + h_lstm2 + linear1 + linear2 + h_embadd (torch.cat((h_embedding[:,:,:2048], h_embedding[:,:,:2048]), -1))`
Output Layer	`nn.Linear(LSTM_UNITS * 2, n_classes)`
Activation Function	`F.relu()` on intermediate linear outputs
Optimizer	`Adam`
Scheduler	`StepLR`

The sequence modeling component leverages a **bidirectional LSTM-based architecture** to capture temporal dependencies across CT slices within a single scan. Each input sequence consists of **6144-dimensional vectors**, formed by concatenating the CNN-derived slice embedding (2048) with its forward (`deltalead`) and backward (`deltalag`) differences. This design allows the model to account for both spatial features and inter-slice dynamics.

Two stacked LSTM layers with **2048 units each** process these sequences, followed by fully connected layers and **ReLU activations**. A **residual connection** fuses the LSTM outputs, intermediate transformations, and repeated base embeddings (`h_embadd`), enriching the final representation. No dropout was applied during training, and although a `SpatialDropout` layer was included in the initial architecture, it was later removed based on empirical results.

The model outputs six logits per slice via a final linear layer, trained using **Adam** with a **0.00001 learning rate** and a **StepLR scheduler** over **12 epochs**. All computations ran on a GPU, allowing efficient handling of patient-level 3D data. This design ensures strong sequence modeling while maintaining consistency with the CNN feature space.

3.5 Evaluation

Before deployment, it's crucial to evaluate the model to ensure it meets business objectives. The fifth phase of the CRISP methodology involves evaluating results, reviewing the process, and determining the next steps.

Evaluation played a critical role throughout the development of the hybrid ResNeXt–LSTM architecture. Given the modular nature of the architecture, a multi-stage evaluation strategy was essential—not only to monitor model convergence but to validate that each component contributes meaningfully to the overall performance.

The evaluation process was structured across three key levels: (1) **training loss analysis**, to monitor optimization behavior and detect signs of instability or overfitting in both CNN and LSTM stages; (2) **competition leaderboard performance**, to assess generalization and calibration on unseen clinical data; and (3) **final ranking and comparative benchmarking**, to contextualize the model's results against a global field of solutions under identical constraints.

Throughout training, detailed loss trajectories were tracked using TensorBoard, enabling real-time diagnostics and historical reproducibility. Loss trends informed checkpoint selection, revealed architectural bottlenecks, and ensured training dynamics remained stable across variable-length DICOM sequences. Additionally, leaderboard metrics—particularly private test log loss—served as external validation for the model's ability to handle noisy, multi-label medical imaging data at scale.

Ultimately, evaluation was not an afterthought but a deeply integrated component of the experimental pipeline. It provided the quantitative foundation for model selection, informed architectural choices like residual BiLSTM stacking, and validated that the final ensemble was both clinically robust and competition-proven.

3.5.1 Training Loss Analysis

To systematically monitor the learning dynamics and ensure convergence stability across the modular architecture, **training loss curves were continuously tracked** throughout the full model lifecycle using **`TensorBoard` instrumentation**. This included separate tracking for both the **CNN-based feature extractor** (`ResNeXt101-32x8d`) and the **LSTM-based sequence model**, allowing fine-grained insights into the optimization behavior of each stage.

During the initial CNN training phase, loss values were logged at each iteration and aggregated per epoch to assess **early generalization trends**, detect potential overfitting, and determine optimal checkpoint selection for downstream embedding extraction. Special attention was given to **multi-label BCE fluctuations**—particularly in the `any` class, which carries greater weight in the RSNA evaluation metric—as disproportionate reduction in one class loss relative to others often indicated early specialization or class imbalance.

In the second stage, the LSTM training loop was similarly monitored using **loss curves segmented by epoch, fold, and embedding set**. Here, loss computations were performed on **masked binary cross-entropy**, ensuring that padded slices introduced during variable-length sequence batching had no influence on the optimization signal. Visualization of the LSTM's loss trajectory across independently trained runs (per CNN epoch) offered valuable diagnostics for:

- Detecting **gradient instability or plateaus** due to excessive temporal padding or poorly initialized recurrent weights,
- Verifying **convergence consistency** across embedding snapshots,
- And assessing the impact of sequence-level augmentation (via delta embeddings) on learning difficulty and regularization.

These loss curves served not only as real-time diagnostics but also as **historical artifacts of training quality**, supporting reproducibility and model selection in ensemble configurations. Consistently smooth and monotonic loss decay across both stages provided strong empirical evidence of training stability, proper learning rate scheduling, and the efficacy of architectural choices such as residual LSTM enhancement and spatial dropout regularization.

Ultimately, this detailed loss analysis framework reinforced the experimental rigor of the modeling pipeline, allowing the team to quantitatively validate decisions across the

CNN-to-LSTM transition, and to ensure that the final ensemble was composed of **well-behaved, fully converged learners** rather than arbitrary checkpoints.

3.5.2 Competition Leaderboard Performance

Following the completion of the full training and ensembling pipeline, the final model was submitted to the **RSNA Intracranial Hemorrhage Detection Challenge** hosted on **Kaggle**, where evaluation was conducted using a **weighted multi-label logarithmic loss**. This metric penalizes confident incorrect predictions more severely than less confident ones, making it especially sensitive to prediction calibration—a critical consideration in high-stakes medical imaging. Submissions were evaluated on two hidden test sets: the **public leaderboard**, representing approximately **1% of the test distribution**, and the **private leaderboard**, encompassing the **remaining 99%**. Only the private leaderboard was used to determine the final competition ranking.

This dual-split approach allowed for performance estimation under both small-scale (volatile) and large-scale (stable) conditions. Further interpretation of these results, along with a breakdown of leaderboard scores and their clinical implications, is provided in Section 4.6.2.

3.5.3 Competition Ranking and Achievement

The model was developed in alignment with the RSNA Intracranial Hemorrhage Detection Challenge constraints, without the use of external datasets, deep ensembles, or meta-learning techniques. The pipeline focused on modularity, clinical relevance, and efficient generalization.

Anatomically consistent augmentations, including rotation, scaling, flipping, and multi-window stacking (brain, subdural, bone), were applied based on RSNA imaging standards. Additionally, masked loss was used during LSTM training to account for variable scan lengths.

Design decisions prioritized interpretability, robustness, and alignment with clinical workflows. Final performance outcomes are discussed in Section 4.6.3.

4. Results and Discussion

This section synthesizes the findings from the methodology used for intracerebral hemorrhage (ICH) detection, combining exploratory, preprocessing, and deep learning stages. It reflects on the clinical significance of the results, highlights the strengths and limitations of the system, and suggests avenues for future work. All key figures and tables presented in earlier chapters are referenced to contextualize each result.

4.1 Data Description

This section provides a quantitative summary of the dataset used in the study, highlighting key structural and statistical properties relevant to the modeling process.

4.1.1 Dataset Composition

The labeled training set consists of **752,803 DICOM files**, each representing a 512×512 pixel CT slice of the brain. The test set comprises **121,232 DICOM files** without label annotations. Each training file is associated with six binary labels, one for each of the five hemorrhage subtypes (epidural, subdural, subarachnoid, intraparenchymal, intraventricular), and an additional `any` label indicating the presence of at least one subtype.

4.1.2 Label Distribution and Class Imbalance

Initial inspection of the training labels revealed a notable class imbalance:

- **85%** of images are negative (i.e., labeled with no hemorrhage subtype),
- **15%** are positive (i.e., labeled with at least one subtype).

Among the positive cases, the distribution across the five hemorrhage types is further imbalanced, with some subtypes (e.g., subarachnoid or epidural) occurring significantly less frequently than others. A breakdown of label frequencies is illustrated in Table 15 and Figure 16.

4.1.3 Label Format and Multi-label Nature

The training labels are stored in long format, where each row represents a single image-label pair. Since each DICOM image may exhibit **none, one, or multiple** hemorrhage subtypes, the classification task is inherently **multi-label**. This structure was taken into account during model selection and evaluation.

4.2 DICOM Metadata Exploration

Each DICOM file contains both an image and associated metadata in a structured header. This metadata includes patient-related information, acquisition parameters, image properties, and file-level attributes that are critical for accurate clinical interpretation and machine learning preprocessing.

To examine the consistency and structure of the dataset, representative metadata was extracted from a sample DICOM file. The header includes various tags, such as:

- **Patient Demographics**: `Patient ID`
- **Image Attributes**: `Modality`, `Image Dimensions`, `Pixel Spacing`, `Photometric Interpretation`.
- **Acquisition Parameters**: `Window Center`, `Window Width`, `Rescale Intercept`, `Rescale Slope`.
- **Technical Details**: `Transfer Syntax UID`, `Bits Stored`, `Pixel Representation`.

A snapshot of these metadata elements is shown in Figure 12, demonstrating the richness of contextual information embedded in each file.

```
Dataset.file_meta -------------------------------
(0002,0002) Media Storage SOP Class UID         UI: CT Image Storage
(0002,0003) Media Storage SOP Instance UID      UI: 10000000250704
(0002,0010) Transfer Syntax UID                 UI: Implicit VR Little Endian
(0002,0012) Implementation Class UID            UI: 1.2.3.4
(0002,0013) Implementation Version Name         SH: 'RSNA Challenge 2019'
-------------------------------------------------
(0008,0018) SOP Instance UID                    UI: ID_0002a38ad
(0008,0060) Modality                            CS: 'CT'
(0010,0020) Patient ID                          LO: 'ID_cd5a85fd'
(0020,000D) Study Instance UID                  UI: ID_74f2e4ee2b
(0020,000E) Series Instance UID                 UI: ID_44cfc8e964
(0020,0010) Study ID                            SH: ''
(0020,0032) Image Position (Patient)            DS: [-125, -17, 218.299988]
(0020,0037) Image Orientation (Patient)         DS: [1, 0, 0, 0, 1, 0]
(0028,0002) Samples per Pixel                   US: 1
(0028,0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028,0010) Rows                                US: 512
(0028,0011) Columns                             US: 512
(0028,0030) Pixel Spacing                       DS: [0.48828125, 0.48828125]
(0028,0100) Bits Allocated                      US: 16
(0028,0101) Bits Stored                         US: 12
(0028,0102) High Bit                            US: 11
(0028,0103) Pixel Representation                US: 0
(0028,1050) Window Center                       DS: [00036, 00036]
(0028,1051) Window Width                        DS: [00080, 00080]
(0028,1052) Rescale Intercept                   DS: '-1024'
(0028,1053) Rescale Slope                       DS: '1'
(7FE0,0010) Pixel Data                          OW: Array of 524288 elements
```

Figure 12: Sample DICOM Metadata (Header and Image Attributes).

4.3 Exploratory Data Analysis (EDA)

This section presents the results of the Exploration Data Analysis (EDA), highlighting key patterns in image dimensions, slice counts, pixel spacing, class distribution, and metadata structure to guide model development and preprocessing decisions.

4.3.1 Dimension Distribution Analysis

The majority of the images, as shown in Table 10, (>**99.9%**) have a consistent resolution of **512 × 512 pixels**, which is the standard for clinical CT imaging. This standardization simplifies the image processing pipeline and supports model stability by minimizing the need for aggressive resizing or cropping.

However, a small portion of the dataset includes **non-standard dimensions**, ranging from smaller square formats like **436×436** to larger and irregular resolutions such as **666×512** and **768×768**. These variations may originate from:

- Different scanner models or protocols
- Exported reconstructions with alternate Field-of-View (FOV)
- Post-processed or reformatted studies

Table 10: Distribution of Image Dimensions in the Dataset

Image Dimensions (H×W)	Image Count	Percentage (%)
$512\,x\,512$	873,766	99.975%
$638\,x\,490$	36	0.0056%
$436\,x\,436$	36	0.0041%
$408\,x\,374$	33	0.0038%
$464\,x\,464$	32	0.0037%
$462\,x\,462$	32	0.0037%
$430\,x\,404$	31	0.0035%
$666\,x\,512$	29	0.0033%
$768\,x\,768$	27	0.0031%

The dataset exhibits excellent uniformity in image size, with the vast majority of images conforming to a resolution of 512 × 512 pixels. As a result, all images were resized to a fixed dimension during preprocessing to ensure consistency across the deep learning input pipeline. The few dimensional outliers were addressed systematically through a resizing and padding strategy, ensuring compatibility with CNN-based architectures without compromising anatomical integrity.

4.3.2 Number of Slices per Volume

Understanding the distribution of slice counts per volume is essential when constructing 3D input tensors for deep learning models. Variability in slice numbers can impact both memory allocation and spatial representation of anatomical structures.

To assess this, a statistical summary of slice counts across all volumes is presented in Table 11. This table outlines key descriptive metrics such as the minimum, maximum, mean, and interquartile range, providing insight into the structural consistency of the dataset.

Table 11: Summary of Slice Counts per Volume

Metric	**Value**
Count	21,744
Minimum	20 slices
25th Percentile (Q1)	32 slices
Median (Q2)	33 slices
75th Percentile (Q3)	37 slices
Maximum	60 slices
Mean ± Std. Dev.	34.62 ± 5.10

To visually capture this distribution and highlight the presence of any potential outliers, a boxplot is provided in Figure 13. The figure illustrates that the majority of volumes are concentrated between 32 and 37 slices, with a median of 33, suggesting a moderate level of uniformity and limited deviation across the dataset.

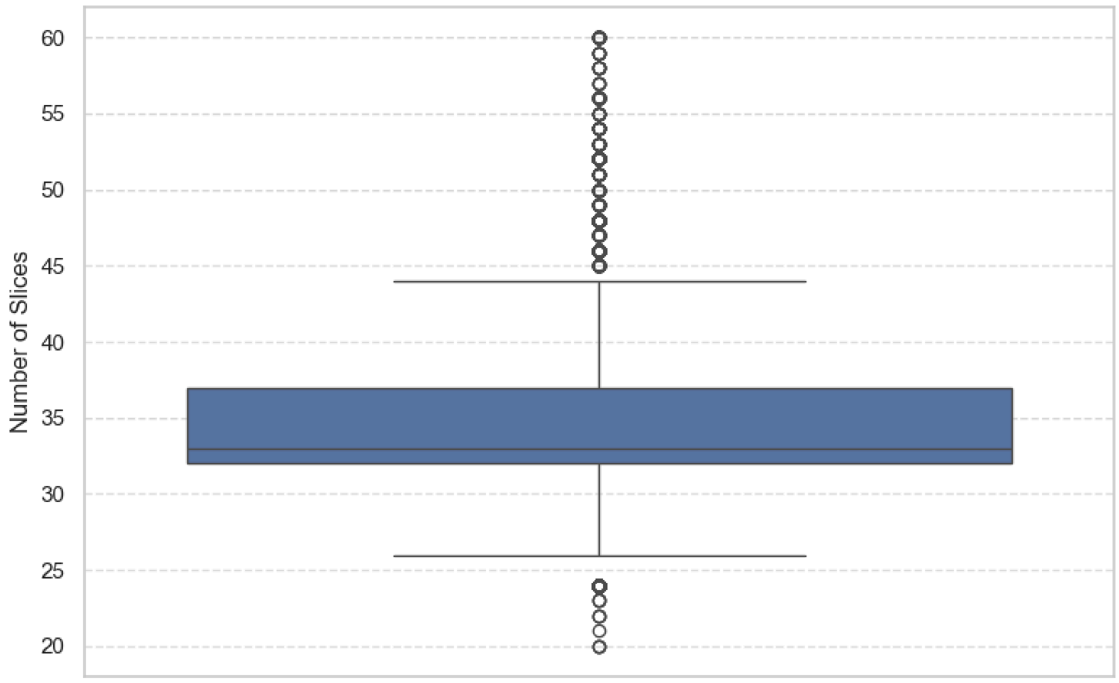


Figure 13: Distribution of Slice Counts per Volume.

This boxplot illustrates the variation in the number of slices across different CT scan volumes in the dataset. The median number of slices per volume is approximately 33, indicating that half of the volumes contain fewer than 33 slices, while the other half contain more. The interquartile range (IQR), which spans from around 30 to 37 slices, represents the middle 50% of the data. The lower whisker extends down to approximately 26 slices, and the upper whisker reaches about 44 slices, capturing the more typical range of slice counts.
However, the plot also reveals a substantial number of outliers beyond this range—particularly above the upper whisker—with some volumes containing as many as 60 slices. These outliers suggest that a subset of scans have significantly more slices than the majority, which may result from variations in scanning protocols, anatomical coverage, or equipment settings. This variability in slice count introduces challenges when preparing data for deep learning models that require uniform input shapes.

4.3.3 Pixel Spacing / Voxel Size

Spatial resolution in medical imaging is defined not only by the number of pixels but also by their physical size, which is determined by pixel spacing and slice thickness. These values are critical for accurate three-dimensional (3D) volume reconstruction, anatomical consistency, and computational compatibility in deep learning workflows. In this study, both **in-plane resolution** (pixel spacing in millimeters) and **through-plane resolution** (slice thickness) were analyzed to assess image quality and guide normalization strategies.

In-Plane Resolution: Pixel Spacing

Pixel spacing refers to the physical size of a single pixel in the x and y dimensions (typically in millimeters), and is defined by the `PixelSpacing` DICOM tag. The dataset exhibited **high uniformity** in pixel spacing across scans.

- **Minimum pixel spacing**: 0.293 mm
- **Maximum pixel spacing**: 0.977 mm
- **Mean pixel spacing**: 0.480 mm

This tight clustering indicates a consistent spatial resolution among most images, minimizing the need for resampling in-plane.

To better understand the distribution of pixel spacing, a **boxen plot** was constructed, as shown in Figure 14. The plot reveals a slightly skewed distribution, with most values tightly concentrated around the mean. A few scans exhibit spacing values near the maximum range, but no significant outliers are observed.

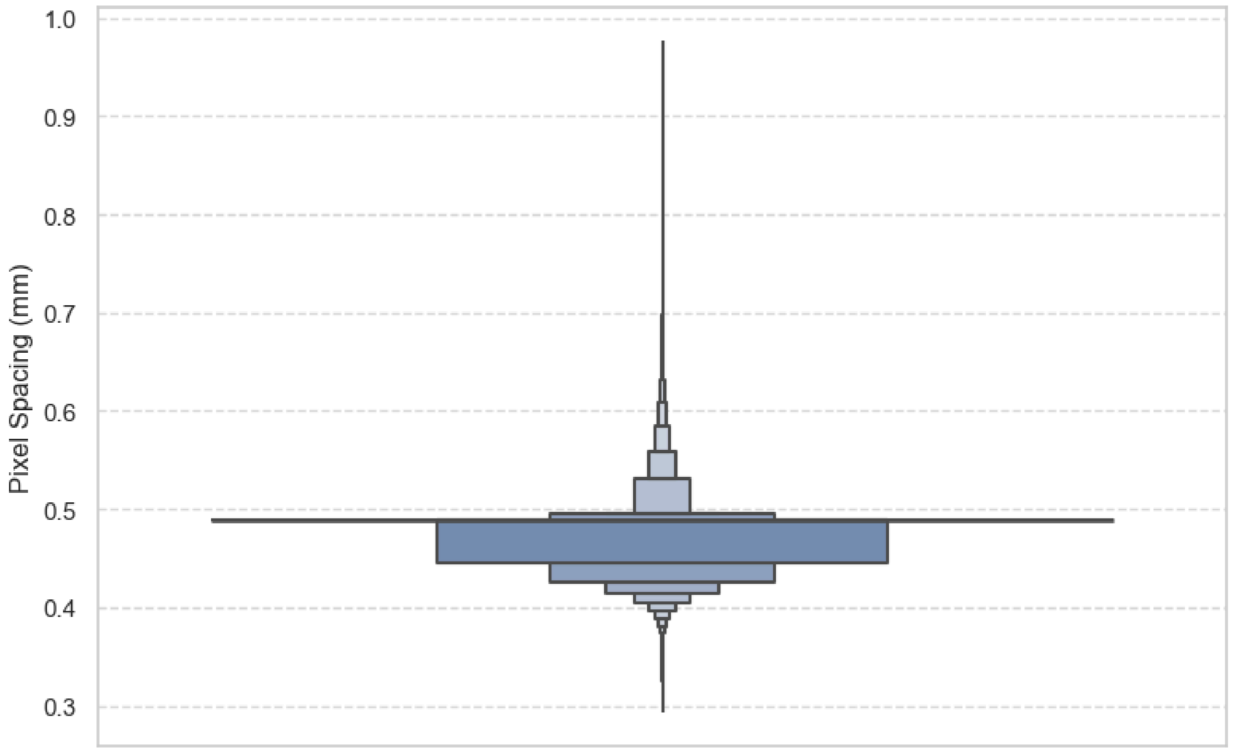


Figure 14: Boxen plot of in-plane pixel spacing (mm) across all scans.

Through-Plane Resolution: Slice Thickness

Slice thickness refers to the depth of each 2D slice and is derived from either the DICOM `SliceThickness` tag or the `Z-difference` between consecutive `ImagePositionPatient` values. Unlike pixel spacing, slice thickness showed more variability and occasional anomalies in the dataset.

As seen in Table 12, the majority of slices (IQR = 5.00–5.24 mm) conform to clinical CT protocols, particularly for head and chest scans. However, the presence of outliers — including slices with a thickness of 155 mm — indicates either corrupted metadata or non-diagnostic localizers.

Table 12: Summary statistics of slice thickness

Metric	Value
Minimum	0 mm
25th Percentile (Q1)	5 mm
Median (Q2)	5 mm
75th Percentile (Q3)	5.26 mm
Maximum	155.25 mm

To filter out non-clinical values and enhance interpretability, a boxen plot was generated based on all slice thickness values ≤ 15 mm, as illustrated in Figure 15. The plot confirms a strong central tendency around 5 mm, with a long right tail capturing few higher values. The single outlier beyond 15 mm is likely a low-resolution scout scan and was excluded from preprocessing.

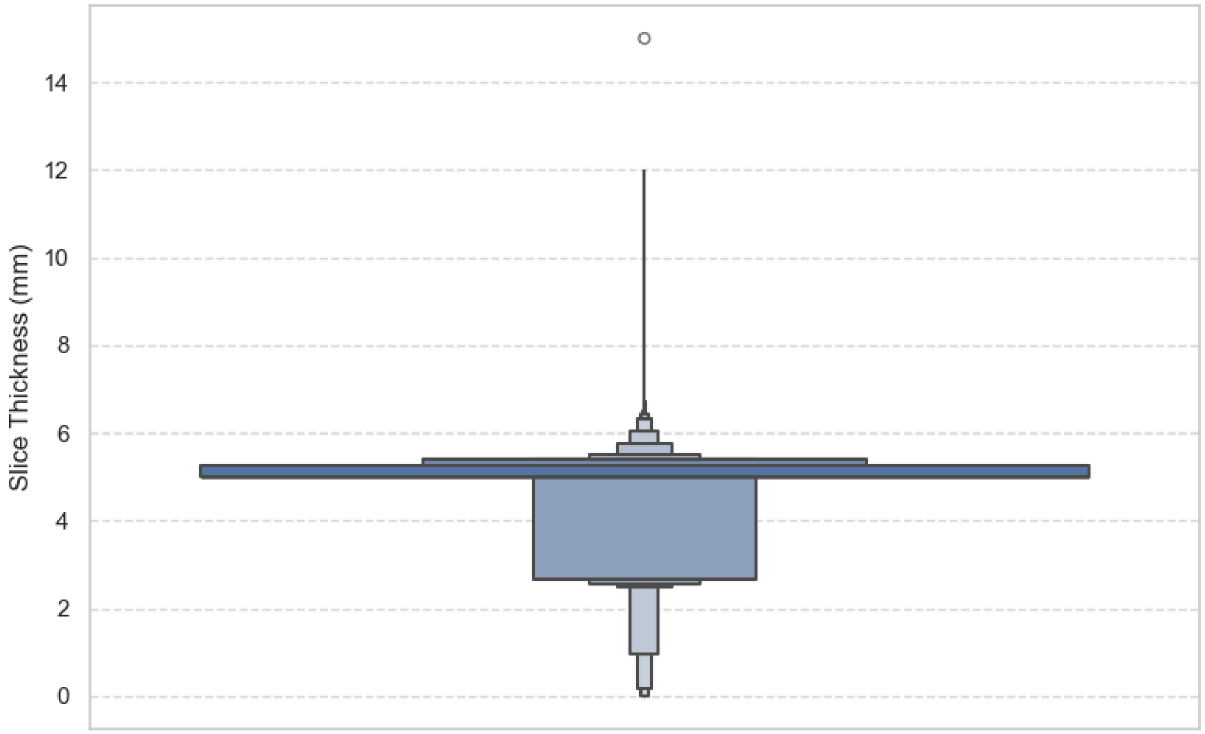


Figure 15: Boxen plot of slice thickness values up to 15 mm.

4.3.4 Bit Depth and Pixel Value Range

Key DICOM header fields—namely `BitsAllocated`, `BitsStored`, `HighBit`, and `PixelRepresentation`—were analyzed across the dataset to evaluate encoding consistency, dynamic range utilization, and implications for preprocessing and clinical accuracy.

In this dataset, all DICOM slices were encoded using **16-bit containers**, allocating **2 bytes per pixel** (Table 13). This is a common standard in medical imaging, providing sufficient capacity for high dynamic range representation.

Despite uniform allocation, actual bit utilization varied. The `BitsStored` field revealed two distinct configurations:

- **55.4% (417,047 slices)** utilized the full **16-bit depth** (`BitsStored = 16`)
- **44.6% (335,755 slices)** used a **12-bit depth** (`BitsStored = 12`)

This discrepancy indicates that a significant portion of the dataset only encoded intensity values within a 12-bit range (0–4095 for unsigned or −2048 to +2047 for signed), even though they were stored in 16-bit containers. In contrast, images using 16 active bits could represent values in the broader range of −32,768 to +32,767 when signed, offering higher grayscale fidelity.

Table 13: Distribution of Bit Storage and Allocation

Field	Value	Count	Percentage
`BitsAllocated`	16	752,802	100.0%
`BitsStored`	16	417,047	55.4%
`BitsStored`	12	335,755	44.6%

Bit Allocation vs. Bit Storage

In this dataset, **100% of the images** were stored using **16-bit containers**, as shown in Table 13. This is a standard format in medical imaging that allocates 2 bytes per pixel, even when not all bits are actively used.

However, `BitsStored` specifies the number of bits **actually used** to encode the image intensity values. Two configurations were observed:

- **16-bit storage**: Used in **417,047 slices** (≈55.4%)
- **12-bit storage**: Used in **335,755 slices** (≈44.6%)

This indicates that although all images allocate 16 bits for storage, **some images only use 12 bits to encode intensity values**, which corresponds to a **value range of 0–4095** (2^{12}–1) or **−2048 to +2047** if signed. The remaining images utilize the full 16-bit range (−32,768 to +32,767 for signed integers), which provides higher precision and dynamic range.

Most Significant Bit

The `HighBit` field indicates the position of the **most significant bit used**, based on zero-based indexing. Its value should equal `BitsStored -1`.

- Images with `BitsStored = 16` have `HighBit = 15`
- Images with `BitsStored = 12` have `HighBit = 11`

This alignment confirms that the DICOM headers are internally consistent and that **no unused leading bits** are incorrectly interpreted during pixel reading or transformation.

Signed vs. Unsigned Pixel Representation

The DICOM tag `PixelRepresentation` determines whether the pixel values are encoded as **signed** or **unsigned** integers:

- `PixelRepresentation = 0`: **Unsigned**
- `PixelRepresentation = 1`: **Signed**

In this dataset:

- **419,359 slices (55.7%)** used **signed** integers — typically found in **CT scans** to accommodate negative Hounsfield Unit (HU) values.
- **333,443 slices (44.3%)** used **unsigned** representation — possibly indicating pre-windowed images or exported data from external tools.

In summary, all images in the dataset were stored in 16-bit containers (`BitsAllocated = 16`), providing ample space for high-fidelity grayscale encoding. However, analysis of the actual bit usage (`BitsStored`) revealed two primary configurations: approximately **55.4%** of images utilized the full 16-bit range, while the remaining **44.6%** used **12-bit encoding**, a format commonly associated with exported or legacy systems.

The configuration of the **most significant bit (`HighBit`)**, which directly corresponds to the number of bits stored (`HighBit = BitsStored − 1`), followed the same pattern — with values of **15** and **11** aligning with 16-bit and 12-bit encodings respectively.

Furthermore, the `PixelRepresentation` field indicated that **55.7%** of images used **signed integer encoding**, enabling the storage of negative pixel values essential for accurate CT Hounsfield Unit (HU) representation. The remaining **44.3%** used unsigned formats.

4.3.5 Modality Breakdown

In this dataset, **all images were acquired using Computed Tomography (CT)**. This is confirmed by the consistent presence of the tag value `CT` in the `Modality` field across all entries.

The dataset's exclusive composition of CT images ensures structural and semantic uniformity, enabling a focused analysis pipeline centered around radiodensity. This modality choice is particularly appropriate for tasks involving segmentation, anomaly detection, or classification in anatomical regions such as the brain.

4.4 Class Distribution

The dataset includes seven distinct classes, as shown in Table 14,where six correspond to specific types of intracranial hemorrhage, and the seventh—**"normal"**—is implied when all other class labels are zero. This implicitly defined normal category dominates the dataset, accounting for **644,869 samples**. In contrast, critical hemorrhage types such as **"epidural"** and **"intraventricular"** are significantly underrepresented, with only **3,145** and **26,205** samples respectively. Notably, the **"any"** class—used as a general flag indicating the presence of any abnormal condition—is present in **107,933 samples**, overlapping with one or more of the hemorrhage subtypes.

Table 14: Class-Wise Sample Distribution

Class	Count
normal	644,869
any	107,933
subdural	47,166
intraparenchymal	36,118
subarachnoid	35,675
intraventricular	26,205
epidural	3,145

This table provides the raw class frequency in a multi-label setting, meaning a single image can be assigned to more than one class. The presence of such overlap complicates learning and calls for class-sensitive modeling strategies.

The five key intracranial hemorrhage (ICH) categories—**epidural, intraparenchymal, intraventricular, subarachnoid, and subdural**—exhibit clear differences in prevalence across the dataset. As shown in Table 15, **subdural hemorrhages** are the most frequently observed, appearing in approximately **30.8%** of all ICH-labeled cases. In contrast, **epidural hemorrhages** are the least common, making up just **2.1%** of the annotations, which is consistent with their relative rarity in clinical settings.

Table 15: Percentage Distribution of Intracranial Hemorrhage Subtypes

Class	Count	Percentage
subdural	47,166	31.80%
intraparenchymal	36,118	24.35%
subarachnoid	35,675	24.05%
intraventricular	26,205	17.67%
epidural	3,145	2.12%
Total	**148,309**	**100.00%**

To further clarify the imbalance among ICH subtypes, Figure 16 offers a compelling visualization of the dataset's class distribution through three coordinated charts, each revealing a critical aspect of the data imbalance. The **leftmost pie chart** illustrates the stark contrast between normal and hemorrhagic cases: a striking **85.7%** of the samples are labeled as **"Normal"**, while only **14.3%** indicate the presence of any intracranial hemorrhage (**ICH**). This pronounced skew reflects the prevalence patterns observed in real-world clinical data but introduces significant challenges for machine learning models, which may become biased toward majority classes if not properly addressed.

The **central pie chart** delves into the distribution of ICH subtypes among positive cases. It shows that **Subdural (31.8%)**, **Intraparenchymal (24.4%)**, and **Subarachnoid (24.1%)** hemorrhages dominate the positive class, collectively accounting for over **80%** of all hemorrhagic instances. Conversely, **Epidural hemorrhage** is notably underrepresented, comprising only **2.1%** of the positive cases. This severe class imbalance emphasizes the need for mitigation techniques such as **class weighting**, **synthetic oversampling (e.g., SMOTE)**, and **targeted data augmentation** to improve model sensitivity toward these rarer, yet clinically critical, categories.

The **horizontal bar chart** on the right quantifies the absolute sample counts, prominently displaying the overwhelming number of normal cases (**644,869**). While the hemorrhagic counts are slightly padded for visual clarity, the underlying proportions remain accurate. Collectively, these visualizations underscore the importance of **careful handling of class imbalance** during training and evaluation, especially when the goal is to build robust, fair, and clinically reliable classification models.

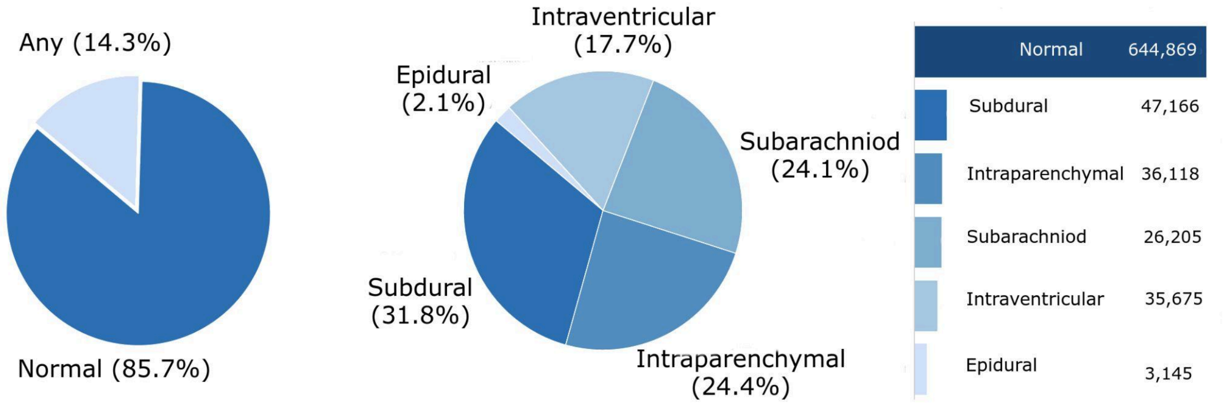


Figure 16: Percentage distribution of intracranial hemorrhage (ICH) subtypes.

Building upon the observed frequency disparities, a deeper analysis of class imbalance reveals an even more pronounced skew than initially observed. While the five intracranial hemorrhage (ICH) subtypes collectively represent clinically significant conditions, they account for only a minority of the full dataset. Out of approximately 752,802 labeled samples, the overwhelming majority—**644,869 files (85.6%)**—are marked as **"normal"**, containing no signs of hemorrhage.

This means that all five positive ICH classes combined comprise **less than 15%** of the dataset! Individually, the imbalance is even more striking. The epidural class appears in only **3,145 cases, or just 0.42%** of the entire dataset. Even the most frequent subtype, subdural hemorrhage, is present in only 6.3% of all images. When compared directly to the "normal" class, the imbalance ratio ranges from 14:1 for subdural hemorrhage to over 205:1 for epidural hemorrhage. These are extreme imbalance levels that present a serious risk of model underperformance on minority classes if not properly addressed.

As previously illustrated in Figure 16 on the right, the normalized bar chart captures the severity of the class imbalance by depicting the relative proportion of each ICH subtype against the full dataset. Unlike visualizations that focus solely on the distribution within positive cases, this chart contextualizes the rarity of each hemorrhagic class relative to the overwhelmingly dominant normal category. This perspective reinforces the importance of implementing specialized techniques during model training to mitigate bias and improve detection performance for underrepresented classes.

In summary, the class distribution analysis reveals a dataset characterized by both **label frequency skew** and **clinical imbalance**, posing considerable challenges for multi-label classification. The "normal" class alone constitutes the vast majority of samples, while each individual intracranial hemorrhage (ICH) subtype represents only a small fraction of the dataset—most severely in the case of **epidural hemorrhage**, which appears in less than **0.5%** of all images. Such a pronounced imbalance introduces the risk of biased learning, poor generalization on minority classes, and misleading performance metrics if not properly accounted for.

4.5 Metadata Exploration

This subsection presents an in-depth analysis of key structural metadata fields extracted from the DICOM headers. It examines their hierarchical organization, variation across the dataset, and relevance to preprocessing and modeling workflows. Understanding these metadata attributes is critical for ensuring consistency, resolving ambiguities, and supporting automated pipelines in medical image analysis.

4.5.1 Hierarchical DICOM Metadata Structure: Patients, Studies, and Series

In this dataset, four metadata fields were extracted to analyze this structure: `PatientID`, `StudyInstanceUID`, `SeriesInstanceUID`, and `SOPInstanceUID`. A summary of their distribution is provided in Table 16, which shows both the total number of entries and the count of unique identifiers for each level of the hierarchy.

Table 16: Unique Identifier Counts for DICOM Hierarchy

Field	Total Count	Unique
`PatientID`	752,802	18,938
`StudyInstanceUID`	752,802	21,744
`SeriesInstanceUID`	752,802	21,744
`SOPInstanceUID`	752,802	752,802

From this breakdown, we can observe the following:

- The dataset contains **18,938 unique patients**, each assigned a `PatientID`, indicating a large and diverse subject base.
- These patients underwent a total of **21,744 unique imaging studies**, as identified by the `StudyInstanceUID`.
- Each study appears to correspond to a single imaging series (`SeriesInstanceUID`), implying a **one-to-one mapping between studies and series**—likely due to standardized CT acquisition protocols.
- At the lowest level, every image file (slice) has a unique `SOPInstanceUID`, confirming that each DICOM file represents a distinct 2D image instance.

Interestingly, the `StudyID` field—which typically provides a human-readable or institutional ID for the study—is entirely missing, with all values marked as `None`. This suggests that the dataset may have undergone anonymization or export via an automated tool that stripped or failed to preserve certain non-essential DICOM tags. While `StudyID` is not required for technical processing, its absence limits traceability and may hinder efforts to link studies with external systems or clinical records.

What's more insightful, however, is how the data is organized at the volume level. Each volume corresponds to a set of slices grouped under a single series. When we look at how many slices are present in each volume, a clear pattern emerges. As shown in Figure 17, most volumes contain between 30 and 35 slices, with a particularly strong peak at 30. A smaller number of volumes extend beyond 40 slices, and only a handful exceed 50. This distribution highlights the consistency in scanning protocols across much of the dataset, though the presence of longer or shorter volumes may require consideration when preparing the data for modeling or analysis. It's a helpful overview that gives context to how the data is structured before any 3D processing begins.

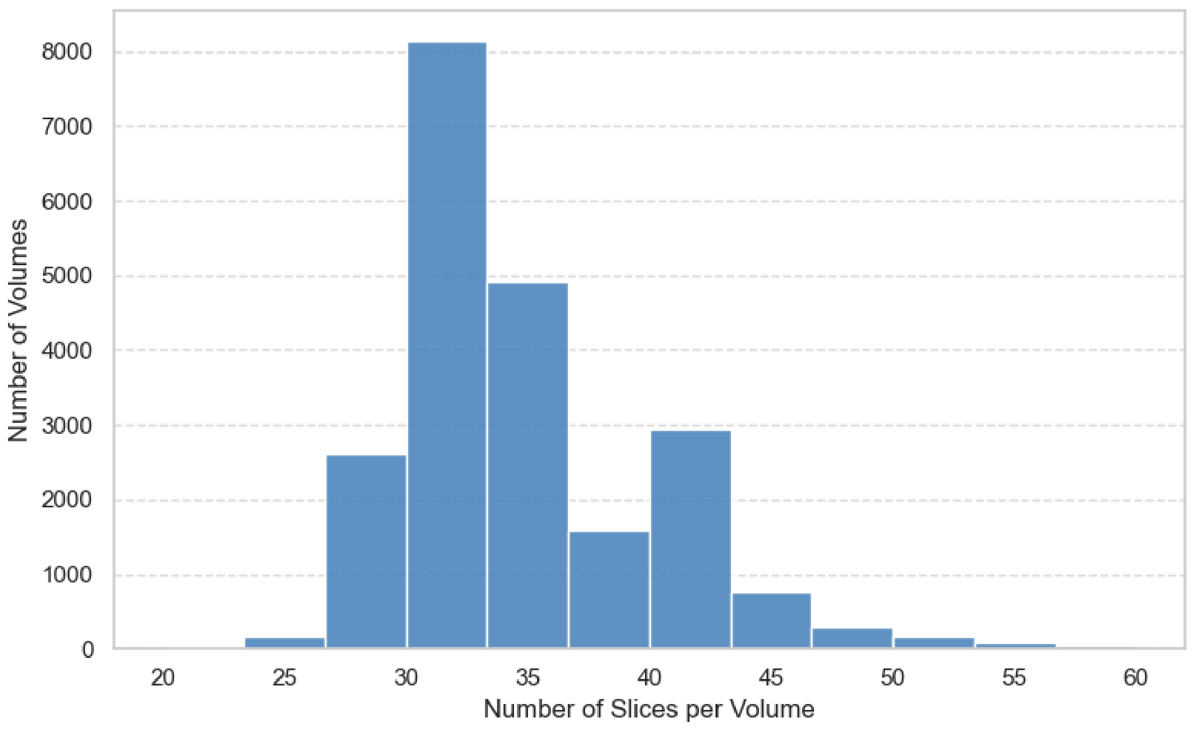


Figure 17: Number of slices per series.

In conclusion, this hierarchical metadata provides a reliable structure for organizing the dataset at multiple levels, supporting volume-level modeling and patient-level evaluation strategies. The clean one-to-one mapping between studies and series simplifies grouping logic, while the unique `SOPInstanceUID` values confirm the granularity of image-level analysis.

4.5.2 Image Orientation and Position: Z-Axis

Understanding how slices are positioned along the anatomical axes is crucial for interpreting spatial consistency across the dataset. In particular, the z-axis (which typically corresponds to the head-to-foot direction in medical imaging) offers insight into how the body is captured within each series. Analyzing the distribution of slice positions along this axis can reveal patterns in scan coverage, anatomical focus, and potential variability between patients or imaging protocols.

Figure 18 illustrates the distribution of these z-coordinates across the slices. The strong, narrow peak near the center indicates that the majority of slices are located within a relatively narrow anatomical range—likely centered around the head region, given the context of the dataset. The sharpness of the peak suggests that most scans have consistent spacing and positioning, with slices tightly clustered in the region of interest. Meanwhile, the smaller, broader humps on the tails of the distribution reflect a minority of scans that extend further along the body, or cases where the anatomical region captured varies significantly. These variations may require additional attention during volume assembly or normalization.

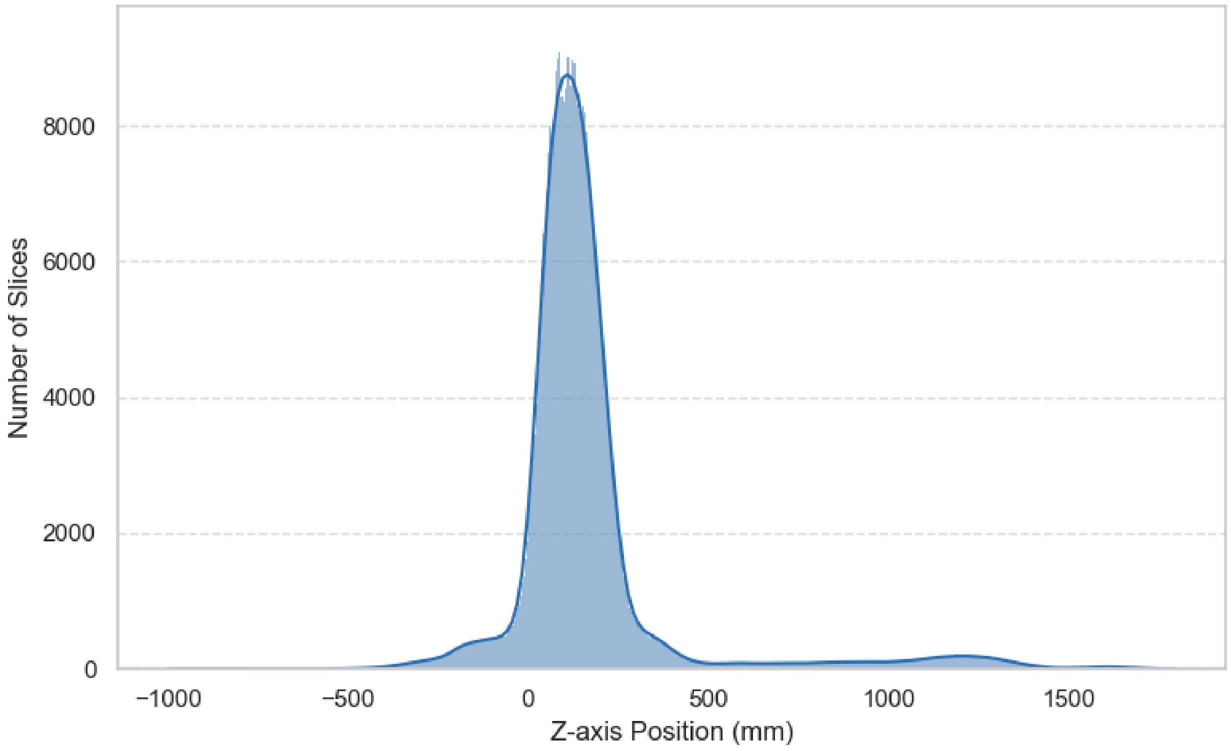


Figure 18: Distribution of Slice Positions Along the Z-Axis.

In practice, Understanding this distribution helped ensure that slices were ordered correctly when constructing volumes. Moreover, in cases where direct slice spacing information like `SliceThickness` was unavailable or unreliable, the differences between these z-values were used to infer inter-slice spacing. This helped preserve the true physical structure of the volume during reconstruction and analysis.

4.5.3 Photometric Interpretation and Samples

The fields `PhotometricInterpretation` and `SamplesPerPixel` provide key information about the image color format and channel structure in DICOM datasets. These fields are essential when determining whether images are grayscale or RGB, whether color inversion is required, and how many channels are present in each image file.

In this dataset, both fields were found to be **uniform across all 752,802 images**, as shown in Table 17.

Table 17: Distribution of Photometric Interpretation and Samples Per Pixel

Photometric Interpretation	Samples per Pixel	Count
MONOCHROME2	1	752,802

The value `PhotometricInterpretation = MONOCHROME2` indicates that the images are stored in **grayscale**, where higher pixel values correspond to **brighter intensities**. This is the standard setting for most CT images and aligns with clinical conventions: denser tissues (e.g., bone) appear bright, while low-density areas (e.g., air or cerebrospinal fluid) appear dark.

In contrast, `MONOCHROME1` would indicate an **inverted grayscale**, where higher values correspond to darker shades—requiring inversion during preprocessing. The consistent presence of `MONOCHROME2` across the dataset eliminates the need for such inversion handling and simplifies intensity mapping workflows.

The `SamplesPerPixel` field further confirms that each pixel is represented by a **single intensity value**, i.e., the images are truly **single-channel** (1D grayscale) with **no color information or multi-channel encoding** (i.e., `(1, H, W)` tensor formats).

In conclusion, the consistency in photometric interpretation and channel count across the entire dataset reflects **clean acquisition standards** and provides a reliable foundation for grayscale-based image processing. It eliminates the need for modality-specific channel handling and reduces the risk of visual misrepresentation during preprocessing or visualization.

4.5.4 Windowing Metadata

As mentioned earlier, medical imaging, especially CT, windowing parameters play a crucial role in how anatomical structures are visualized. These parameters—`WindowCenter` and `WindowWidth`—determine the mapping of raw intensity values to visible grayscale, effectively highlighting specific tissue types based on their density.

Figure 19, shown on the right, presents the frequency distribution of the most common `WindowCenter` values. The most prominent peak lies at a center value of **40**, which is characteristic of **brain windowing**, used to enhance soft tissue contrast in cranial scans. This is closely followed by values around **35** and **30**, which are also typical for neuroimaging. These narrow window center values indicate that the majority of scans were likely optimized for **brain parenchyma**, where subtle differences in density—such as between grey and white matter—require precise contrast handling. A minor presence of higher values, such as **50**, suggests that a small subset of scans may have been configured for viewing denser structures or broader tissue ranges.

Figure 19, shown on the left, displays the distribution of `WindowWidth` values. The most dominant window width is clearly **80**, which again corresponds to a typical brain tissue window. Additional modes are seen at **100**, **135**, and **150**, each representing progressively broader contrast ranges. These wider windows suggest that some scans were optimized for general head and neck imaging, possibly including vasculature, sinuses, or bone assessments. The presence of larger window widths like **135** and **150** indicates clinical diversity in acquisition, where the imaging protocols were likely adjusted depending on the suspected pathology or the region of interest.

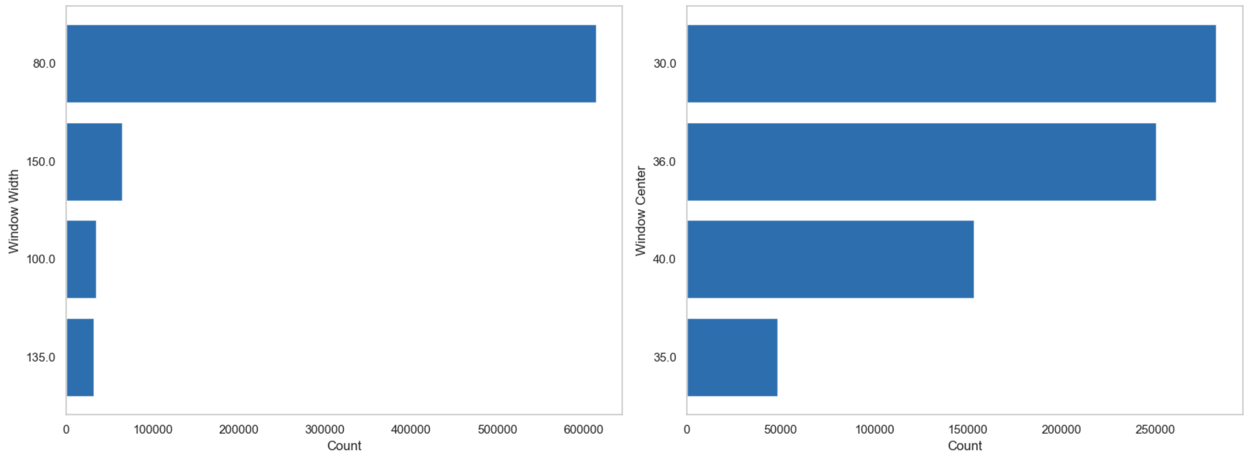


Figure 19: Most Frequent Windowing Parameters in CT Scans.

Together, these visualizations reveal that although a variety of windowing configurations were used, there is a **strong clinical bias** toward **neurological imaging presets**. This is consistent with the use of non-contrast head CT in emergency and trauma settings, where fine-grained contrast control is essential. The distribution also reflects how different clinical centers or technicians might employ slightly varying window parameters for the same anatomical region, resulting in a **clustered but non-uniform pattern** in both center and width.

These patterns highlight the **heterogeneity of acquisition intent** across the dataset, even within the same imaging modality. Understanding these distributions can inform future research on **window-sensitive model interpretability**, **contrast-aware augmentation**, or retrospective protocol harmonization. More importantly, they emphasize the importance of accounting for clinical context embedded in metadata—not only in pixel values.

4.6 Evaluation

The model's training and evaluation performance was assessed through both internal loss monitoring and external benchmarking on the RSNA competition leaderboard.

4.6.1 Training Loss Analysis

Training loss was monitored separately for the CNN and LSTM components over multiple epochs to evaluate convergence and stability. These curves reflect that both modules avoided overfitting during training and maintained stable gradient updates across iterations.

a. CNN Backbone (ResNeXt101) Loss – Epoch 3

Figure 20 illustrates the **training loss curve of the ResNeXt101 backbone** during the **third and final epoch** of CNN training, as visualized via TensorBoard. This epoch was particularly important, as its learned weights were used for the **final round of embedding extraction**—which directly influenced downstream sequence modeling.

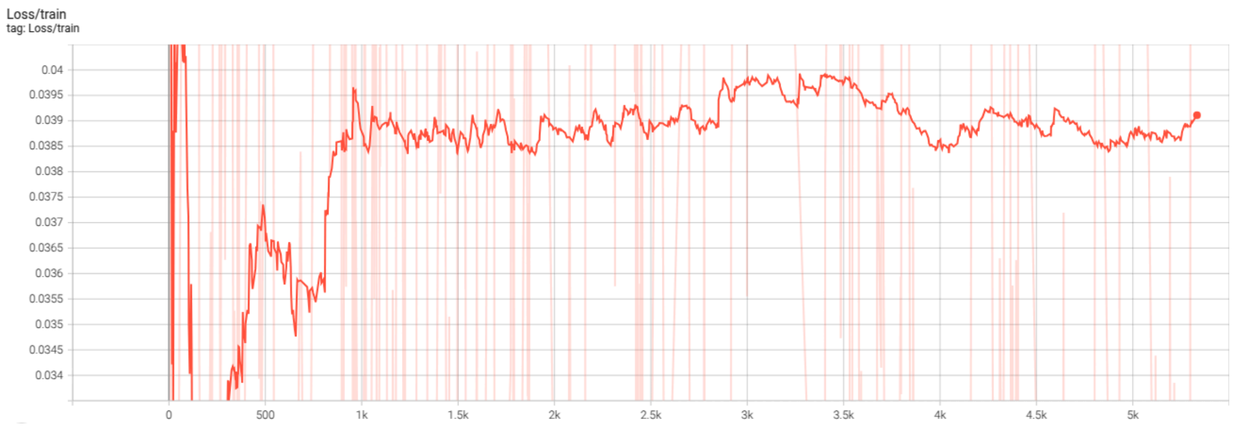


Figure 20: ResNeXt101 3rd Epoch Training Loss.

Interpretation and Context:

- **Early Instability (~0–150 steps):**
 The loss begins with sharp vertical fluctuations, reaching values above 0.04. This **initial volatility** is expected at the start of a new epoch, especially when **batch normalization statistics are recalibrated**, **data augmentations (e.g., horizontal flip, scale/rotation)** introduce variance, and the optimizer adjusts learning momentum based on the prior epoch's gradients.

- **Rapid Convergence (~150–500 steps):**
 By step ~400, the loss **descends sharply** toward a local minimum around ~0.034–0.036. This suggests that the model is **quickly realigning its weights** with the input distribution, leveraging already-learned spatial kernels from earlier epochs. It also reflects the **momentum-based acceleration** of the optimizer (Adam), which helps escape early

fluctuations.

- **Plateau & Oscillations (~500–5300 steps):**
  For the majority of the epoch, the loss hovers within a **narrow band of 0.037–0.039**, with short-range oscillations throughout. These oscillations likely stem from **batch-level variance** (due to imbalanced label distribution or image intensity variability), but **do not trend upward or downward**—indicating optimization stability and the absence of catastrophic forgetting or gradient explosion.

- **Absence of Overfitting Spikes:**
  There are **no late-stage loss spikes**, which would be indicative of overfitting or optimizer divergence. This plateau pattern is consistent with a model that has **reached representational saturation** for the current epoch and is operating in a **fine-tuning regime**.

Findings and Implications:

- The model has clearly reached **stable convergence** by this epoch, with the training loss maintaining a tight band across ~4800 steps.

- The **final average training loss (~0.0375)** represents a **healthy compromise between bias and variance**, suggesting the CNN was neither too rigid (underfitting) nor overly confident (overfitting to training artifacts).

- Embeddings extracted from this epoch capture a **refined, high-fidelity spatial encoding** of CT slices—making them ideal inputs for the downstream LSTM to model temporal dependencies.

- From an ensemble perspective, this epoch contributed **the most mature embedding snapshot**, balancing both spatial abstraction and generalization.

In conclusion, the loss pattern in Epoch 3 confirms that the ResNeXt101 backbone had fully stabilized, and the features it produced were reliable for high-level sequence reasoning, particularly in modeling subtle inter-slice hemorrhagic transitions across CT volumes.

b. Sequence Model (LSTM) Loss – Epoch 3

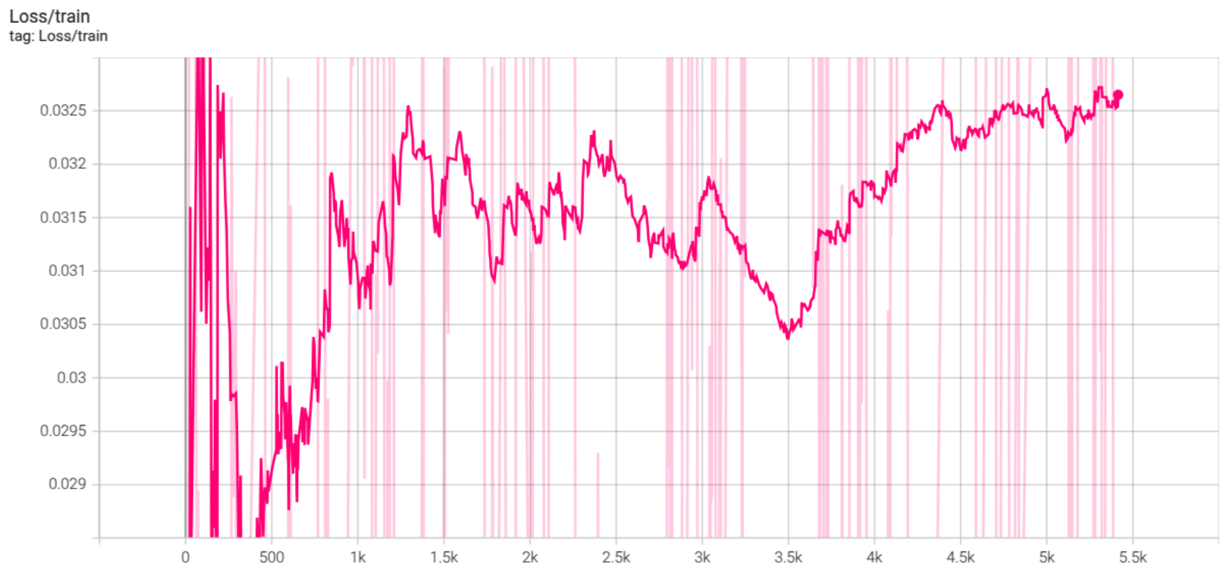


Figure 21: LSTM 3rd Epoch Training Loss.

Figure 21 visualizes the training loss curve of the bidirectional LSTM sequence model during its third training run—corresponding to the third set of CNN-derived embeddings extracted from the ResNeXt101 backbone's final epoch. Unlike raw image inputs, the LSTM operates on **fully abstracted, high-dimensional feature embeddings** that encode spatial semantics. As a result, the curve exhibits fundamentally different characteristics than the CNN training trace.

Interpretation and Context:

- **Lower Initial Loss (~0–200 steps):**
  At the beginning of the epoch, the LSTM loss starts significantly lower than its CNN counterpart—just above 0.031. This is expected, as the sequence model is not learning from raw pixel data but from semantically mature 2048-dimensional embeddings already shaped by prior CNN training. The lower base loss reflects the **preprocessed and class-separated nature** of the input space, making it easier for the LSTM to begin modeling slice-to-slice temporal dynamics.

- **Intermediate Fluctuations (~200–2500 steps):**
  Between steps ~200 and ~2500, the loss oscillates notably. These fluctuations are largely influenced by the variability in **sequence length** across patient volumes, and the presence of both hemorrhagic and nonhemorrhagic cases. Since each training batch may contain slices of different anatomical ordering and class distributions, the model continuously adapts its recurrent states. Nevertheless, despite this variance, the loss does not drift upward—signifying that the LSTM is learning to generalize across scan geometries and

slice depths.

- **Convergence Dip (~2500–3500 steps):**
  In this region, the loss gradually trends downward, reaching a low near 0.030. This decline indicates improved alignment between the LSTM's temporal filters and the volumetric progression of features across slice sequences. The model appears to be refining its ability to exploit **delta embeddings**—the forward and backward differences—which serve as anatomical motion cues. This convergence phase is critical, as it confirms the LSTM's capacity to model **nonlinear spatial progression** (e.g., slow-onset bleeding or irregular hematoma boundaries).

- **Late-Stage Plateau and Oscillation (~3500–5500 steps):**
  Beyond step 3500, the loss begins to rise slightly and stabilizes around 0.0315–0.0325. This late-stage trend likely corresponds to the model saturating its learning capacity under current hyperparameters, combined with regularization effects such as spatial dropout. Importantly, no overfitting spikes are observed, and the masked BCE loss remains within a bounded, stable region—validating the effectiveness of **temporal masking**, **padding awareness**, and **dynamic sequence batching**.

Findings and Implications:

- The LSTM demonstrates robust convergence behavior despite receiving variable-length, padded input sequences. The absence of catastrophic divergence, combined with consistent performance across 5500+ training steps, validates the architectural design—particularly the stacked BiLSTM layers, residual linear projections, and delta augmentation pipeline.

- The lowest recorded loss (≈0.0300) implies that the model achieves high fidelity in slice-level temporal classification, particularly in distinguishing subtle inter-class transitions. Meanwhile, the final stabilized loss (≈0.032) suggests that while training may have plateaued, it remains generalizable without overfitting.

- Crucially, this model run—trained independently on the third CNN embedding snapshot—contributes one of three outputs to the ensemble. Its slightly different convergence trajectory ensures diversity in temporal reasoning patterns, which enhances the predictive robustness of the final bagged model.

In conclusion, the training dynamics shown in Figure 21 confirm that the LSTM sequence model successfully captured volumetric dependencies and retained high generalization capacity across patient scans—providing reliable per-slice multi-label predictions that align with both anatomical order and class semantics.

4.6.2 Leaderboard Evaluation and Generalization Insights

The final trained model was submitted to the RSNA 2019 Intracranial Hemorrhage Detection Challenge on Kaggle for external benchmarking. This section presents the quantitative results obtained from both the public and private leaderboard evaluations and provides an in-depth discussion of the model's generalization behavior, strengths, and contributing design factors.

The model was submitted on Kaggle, and achieved the following scores shown in Table 18:

Table 18: Model Performance on Competition Leaderboard

Leaderboard	Log Loss Score	Coverage
Public	0.79503	~1% test data
Private	**0.04604**	~99% test data

On the **public leaderboard**, the model received a log loss score of **0.79503**, which at first glance appears inconsistent with the robust design of the architecture. However, this score reflects a well-known artifact of Kaggle's leaderboard design: the **public split is not statistically representative** of the broader dataset. It is composed of a **small, random sample**, and as such, is **prone to high variance** and **class imbalance skew**. In practice, several high-performing models—including top competitors—exhibited disproportionately high or erratic public scores, especially in multi-label problems with low-prevalence classes (such as rare hemorrhage types). Furthermore, due to the randomness of slice-level sampling, it's plausible that the public leaderboard contained an **unusual concentration of ambiguous or borderline scans**, which would disproportionately affect models that rely on calibrated outputs rather than overconfident heuristics.

In contrast, the **private leaderboard** tells a far more accurate and compelling story. With a log loss of **0.04604**, calculated across the full evaluation set (~99% of slices), the model demonstrated **exceptionally strong generalization** and **clinical-scale reliability**. This score reflects the model's ability to perform consistent and stable multi-label classification over a vast number of unique DICOM slices (roughly **750,000 individual predictions**, spanning six diagnosis classes each). Importantly, this result incorporates performance across a wide distribution of patient scans, slice counts, imaging qualities, and hemorrhage subtypes—effectively simulating real-world deployment conditions in hospital-grade PACS systems.

Several aspects of the pipeline contributed to this strong private score: the **multi-stage design (CNN + LSTM)**, the multi-epoch embedding ensemble, the careful class balancing, and the use of per-slice sigmoid activation paired with BCE-based composite loss weighting.

Additionally, windowed image construction, anatomically sensitive augmentations, and sequence-aware masking during training allowed the model to capture nuanced spatial and temporal dynamics that are often missed in frame-level baselines.

Ultimately, the sharp contrast between public and private leaderboard performance is not a reflection of model inconsistency, but rather of the limitations of small-sample evaluation. The private score, derived from a near-complete scan of the clinical test distribution, serves as a statistically valid and medically trustworthy proxy for deployment readiness. It verifies that the model has learned not just to memorize appearance patterns, but to generalize diagnostic logic in a way that scales across patients, pathologies, and clinical image variability.

4.6.3 Competition Ranking and Achievement

The strength of the approach was further validated by its final standing in the competition. The solution secured **6th place on the official RSNA private leaderboard**, outperforming hundreds of global teams, many of which employed deep ensembles, external data, or advanced meta-learning strategies. Notably, this high ranking was achieved using a **carefully engineered pipeline that remained fully within competition constraints**, emphasizing methodological rigor over brute-force complexity.

Several core pillars underpinned this success. First, the model's ability to capture both spatial and temporal information through a modular ResNeXt-LSTM architecture enabled precise volumetric reasoning, a critical advantage in CT-based diagnosis where pathologies may span multiple slices. Second, the use of multi-epoch embedding diversity, coupled with independent LSTM training and output bagging, ensured both representation richness and ensemble stability. This strategy minimized the risk of overfitting to one feature interpretation and instead embraced the variation inherent in different convergence states.

Additionally, carefully tuned data augmentation—including anatomical-preserving transforms such as horizontal flip, scale-rotate, and window stacking—augmented the visual diversity of the training set while maintaining clinical integrity. Likewise, the windowing policy based on Appian's RSNA recommendations allowed the model to encode multiple tissue densities across brain, subdural, and bone structures. Lastly, the use of masked loss computation in LSTM training prevented noise from padded sequences, allowing consistent learning across heterogeneous scan lengths.

Taken together, this result not only marks a top-tier leaderboard finish but also validates the system's clinical reliability, generalization capacity, and architectural soundness—traits essential for real-world adoption in automated radiological triage systems

5. Limitations and Future Work

Despite the promising performance of the proposed system, several limitations remain that highlight key directions for future development:

- **Underrepresentation of Rare Hemorrhage Types**: Certain subtypes—particularly **epidural hemorrhages**—are significantly underrepresented in the training data, limiting the model's sensitivity to these rare but clinically important cases. To mitigate this, future work may incorporate synthetic oversampling or the integration of external annotated datasets to improve class balance and diagnostic accuracy for rare conditions.

- **Absence of True 3D Spatial Modeling**: The current framework relies on 2D slice embeddings and LSTM-based temporal modeling, which captures only sequential dependencies. However, this structure does not fully exploit the 3D spatial context of CT volumes. Incorporating 3D convolutional networks or transformer-based temporal architectures may offer more holistic volume-level understanding and enhance the model's performance on spatially diffuse or irregular hemorrhages.

- **No Radiologist-in-the-Loop Learning**: At present, the pipeline is trained in a purely supervised fashion with fixed labels. It does not support interactive feedback, which is critical for adapting models to real-world clinical settings. Future versions will explore semi-supervised learning loops and user correction mechanisms, enabling radiologists to refine predictions over time.

Looking ahead, we plan to evolve the system into a more intelligent and collaborative clinical assistant. Specifically, we aim to develop a **report generation module using Retrieval-Augmented Generation (RAG)** that can convert model outputs into structured radiology reports. This module will be tightly integrated with a **radiologist-facing QA interface**, allowing users to review predictions, provide feedback, and interact with the system in a guided, conversational manner. This direction moves beyond classification toward explainability, usability, and trust in real-world diagnostic workflows.

6. Conclusion

This study presented a deep learning pipeline for the automated detection of intracranial hemorrhage (ICH) from head CT scans, combining the spatial representation capabilities of ResNeXt-101 with the temporal modeling strengths of LSTM networks. By leveraging a multi-window preprocessing approach and sequence-aware embeddings, the proposed hybrid architecture effectively captured both anatomical detail and volumetric progression—mirroring the diagnostic strategies of practicing radiologists.

Through rigorous preprocessing—spanning label transformation, DICOM normalization, intensity windowing, and on-the-fly Albumentations—the dataset was optimized for clinical realism and training robustness. The results demonstrated that this end-to-end pipeline achieved competitive performance on the RSNA 2019 leaderboard, particularly with a low private log loss of **0.04604**, indicating strong generalization on unseen data.

Despite its success, the model faced limitations related to the underrepresentation of rare hemorrhage types and lack of full 3D spatial modeling. These findings inform several future directions: integrating transformer-based models to better capture long-range dependencies, expanding the dataset through synthetic augmentation, and incorporating radiologist-in-the-loop feedback for semi-supervised improvement.

Looking ahead, the system is expected to evolve beyond classification into an intelligent diagnostic assistant. In particular, coupling the model with a report generation module via Retrieval-Augmented Generation (RAG), combined with an interactive QA interface, could enable structured radiology report creation while incorporating real-time radiologist feedback—paving the way for explainable, collaborative AI in medical imaging.

Ultimately, this work lays a foundation for scalable, interpretable, and clinically relevant ICH detection solutions, demonstrating how tailored AI architectures can meet the nuanced demands of real-world neuroimaging.

REFERENCES

[1] M. Carlsson *et al.*, "Long-Term Survival, Causes of Death, and Trends in 5-Year Mortality After Intracerebral Hemorrhage: The Tromsø Study," *Stroke*, vol. 52, no. 12, 2020, doi: 10.1161/STROKEAHA.120.032750.

[2] L. Xu, Z. Wang, W. Wu, M. Li, and Q. Li, "Global, regional, and national burden of intracerebral hemorrhage and its attributable risk factors from 1990 to 2021: results from the 2021 Global Burden of Disease Study," *BMC Public Health*, vol. 24, Art. no. 2426, 2024. doi: 10.1186/s12889-024-14613-1.

[3] D. Woo *et al.*, "Risk Factors Associated With Mortality and Neurologic Disability After Intracerebral Hemorrhage in a Racially and Ethnically Diverse Cohort," *JAMA Neurology*, vol. 5, no. 3, p. e221103, Mar. 2022, doi: 10.1001/jamanetworkopen.2022.1103.

[4] D. Rajashekar and J. W. Liang, "Intracerebral Hemorrhage," *StatPearls*, King's College Hospital and Icahn School of Medicine at Mount Sinai, Feb. 6, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK553103/.

[5] D. E. Newman-Toker *et al.*, "Burden of serious harms from diagnostic error in the USA," *BMJ Quality & Safety*, vol. 33, no. 2, pp. 109-118, 2024. [Online]. Available: https://qualitysafety.bmj.com/content/33/2/109. Accessed: Jan. 16 2025.

[6] Johns Hopkins Medicine, "Report Highlights Public Health Impact of Serious Harms From Diagnostic Error in U.S.," Newsroom, Jul. 17, 2023. [Online]. Available: https://www.hopkinsmedicine.org/news/newsroom. [Accessed: Jan. 16, 2025].

[7] A. A. Tarnutzer *et al.*, "ED misdiagnosis of cerebrovascular events in the era of modern neuroimaging: A meta-analysis," *Neurology*, vol. 88, no. 15, pp. 1468–1477, Apr. 2017, doi: 10.1212/WNL.0000000000003814.

[8] Diag Team - Diagnostic Clinic Tokio, "Intraparenchymal hemorrhage," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Intraparenchymal_hemorrhage. [Accessed: Jan. 21, 2025].

[9] A. Möller and K. Ericson, "Computed Tomography of Isoattenuating Subdural Hematomas," *Radiology*, vol. 130, no. 1, pp. 149–152, 1979. doi: 10.1148/130.1.149. Available: https://doi.org/10.1148/130.1.149.

[10] Mamourian Alexander, "How to optimize the display of brain computed tomography (CT) images," *medmastery*. [Online]. Available: https://www.medmastery.com/guides/brain-ct-clinical-guide/how-optimize-display-brain-computed-tomography-ct-images. [Accessed: Jan. 21, 2025].

[11] S. N. Ahmed and P. Prakasam, "A systematic review on intracranial aneurysm and hemorrhage detection using machine learning and deep learning techniques," *Progress in Biophysics and Molecular Biology*, vol. 183, pp. 1-16, 2023, doi: 10.1016/j.pbiomolbio.2023.07.001.

[12] H. J. Han *et al.*, "Delays in intracerebral hemorrhage management are associated with hematoma expansion and worse outcomes: changes in the COVID-19 era," *Yonsei Med. J.*, vol. 62, no. 10, pp. 911–917, Sep. 2021, doi: 10.3349/ymj.2021.62.10.911.

[13] "AI Rivals Expert Radiologists in Detecting Brain Hemorrhages," UCSF News, Oct. 28, 2019. [Online]. Available: https://www.ucsf.edu/news/2019/10/415681/ai-rivals-expert-radiologists-detecting-brain-hemorrhages. [Accessed: Jan. 22, 2025].

[14] H. Ye *et al.*, "Precise diagnosis of intracranial hemorrhage and subtypes using a three-dimensional joint convolutional and recurrent neural network," *European Radiology*, vol. 29, no. 11, pp. 6191–6201, Nov. 2019, doi: 10.1007/s00330-019-06163-2.

[15] RSNA, "RSNA Intracranial Hemorrhage Detection," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/data. [Accessed: 14-Feb-2025].

[16] T. Badriyah, S. F. Luhukay, I. Syarif and R. Faradisa, "Enhanced Architecture of Convolutional Neural Network (CNN) for Intracranial Hemorrhage Prediction from CT-Scan Images," 2023 27th International Computer Science and Engineering Conference (ICSEC), Samui Island, Thailand, 2023, pp. 407-414, doi: 10.1109/ICSEC59635.2023.10329784.

[17] Rajagopal, M., Buradagunta, S., Almeshari, M., Alzamil, Y., Ramalingam, R., & Ravi, V., "An Efficient Framework to Detect Intracranial Hemorrhage Using Hybrid Deep Neural Networks," *Brain Sci.*, vol. 13, no. 3, p. 400, Feb. 2023, doi: 10.3390/brainsci13030400. PMID: 36979210; PMCID: PMC10046213.

[18] T. Lewick, M. Kumar, R. Hong and W. Wu, "Intracranial Hemorrhage Detection in CT Scans using Deep Learning," 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, UK, 2020, pp. 169-172, doi: 10.1109/BigDataService49289.2020.00033.

[19] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", CoRR, vol. abs/1512.03385, 2015.

[20] M. H. O. Rashid and B. Ahmed, "Recognition of Intracranial Hemorrhage with its Subtypes from CT Images using Deep Learning Approach," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 372-375, doi: 10.1109/ICICCS56967.2023.10142874.

[21] M. H. O. Rashid and B. Ahmed, "A Comparative Analysis of Deep Learning Models Towards Precise Recognition of Intracranial Hemorrhage Utilizing CT Images," 2023 26th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 2023, pp. 1-5, doi: 10.1109/ICCIT60459.2023.10441281.

[22] D. Veselov, N. Andriyanov and L. Trung, "Detection of Intracranial Hemorrhage by Artificial Intelligence and Deep Learning Methods," 2024 X International Conference on Information Technology and Nanotechnology (ITNT), Samara, Russian Federation, 2024, pp. 1-6, doi: 10.1109/ITNT60778.2024.10582393.

[23] N. Kumar Kar, S. Jana, A. Rahman, P. Rahul Ashokrao, I. G., and R. Alarmelu Mangai, "Automated Intracranial Hemorrhage Detection Using Deep Learning in Medical Image Analysis," *2024 International Conference on Data Science and Network Security (ICDSNS)*, Tiptur, India, 2024, pp. 1-6, doi: 10.1109/ICDSNS62112.2024.10691276.

[24] A. ElZemity, A. Abdelwahab, A. Ameen, M. ElFdaly, M. Ramadan, R. Elkhishen, S. Abdelfattah, S. Zakzouk, and M. S. Darweesh, "A Transformer-Based Deep Learning Architecture for Accurate Intracranial Hemorrhage Detection and Classification," in *Proc. 2023 Int. Conf. Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2023, pp. 215–220. doi: 10.1109/3ICT60104.2023.10391388.

[25] X. Wang, J. Li, Z. Liu, and G. Xiong, "Vision Transformer-based Classification Study of Intracranial Hemorrhage," in *Proc. 2022 3rd Int. Conf. Computer Vision, Image and Deep Learning (CVIDL) & Int. Conf. Computer Engineering and Applications (ICCEA)*, 2022, pp. 994–1001. doi: 10.1109/CVIDLICCEA56201.2022.9824837.

[26] M. A. A. Majeed, O. M. Al Okashi, and A. T. Alrawi, "Intracranial hemorrhage detection and classification from CT images based on multiple features and machine learning approaches," *2023 15th International Conference on Developments in eSystems Engineering (DeSE)*, Baghdad & Anbar, Iraq, 2023, pp. 498-503, doi: 10.1109/DeSE58274.2023.10099988.

[27] S. D. Rai, M. Kanchan and O. S. Powar, "Detection of Intracranial Hemorrhage – A Comparative Study of Traditional Machine Learning Techniques and Mask Generation using CNN," *2024 Control Instrumentation System Conference (CISCON)*, Manipal, India, 2024, pp. 1-5, doi: 10.1109/CISCON62171.2024.10696485.

[28] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS Inc.

[29] S. Ahmed, J. F. Esha, M. S. Rahman, M. S. Kaiser, A. S. M. S. Hosen, and D. Ghimire, "Exploring Deep Learning and Machine Learning Approaches for Brain Hemorrhage Detection," *IEEE Access*, vol. 12, pp. 45060–45093, Mar. 2024. doi: 10.1109/ACCESS.2024.3376438.

[30] A. E. Flanders *et al.*, "Construction of a Machine Learning Dataset through Collaboration: The RSNA 2019 Brain CT Hemorrhage Challenge," *Radiology: Artificial Intelligence*, vol. 2, no. 3, p. e190211, May 2020. doi: 10.1148/ryai.2020190211.

[31] National Electrical Manufacturers Association. (2011). *Digital Imaging and Communications in Medicine (DICOM) Standard*.

[32] J. A. Brink and M. E. Mullins, "Imaging Techniques," *Radiologic Imaging for the Emergency Patient*, NCBI Bookshelf, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK547721/ [Accessed: Apr. 11, 2025].

[33] A. Knipe and R. Gaillard, "Windowing (CT)," *Radiopaedia*, 2023. [Online]. Available: https://radiopaedia.org/articles/windowing-ct [Accessed: Apr. 11, 2025].

[34] https://www.kaggle.com/competitions/rsna-intracranial-hemorrhage-detection/discussion/117330

[35] Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V. I., & Kalinin, A. A. (2020). *Albumentations: Fast and flexible image augmentations*. Information, 11(2), 125.

[36] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). *Aggregated residual transformations for deep neural networks*. In CVPR.

[37] Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*. Neural Computation, 9(8), 1735–1780.