Universiteit Leiden

Exploring Synthetic Non-Contrast Brain CT Generation and Deep-Learning Classification for Intracranial Haemorrhage Detection

Saar Calame (s3209199)

Supervisors:
Prof. dr. K.J. Batenburg & N. Fragkos

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 21, 2026

Abstract

Intracranial haemorrhage (ICH) is a potentially life-threatening condition that requires rapid and accurate diagnosis using non-contrast brain computed tomography (CT) scans. Although deep-learning models have demonstrated strong performance in automated ICH detection, their development may be limited by the availability of sufficiently large, diverse, and accurately annotated medical imaging datasets. Synthetic image generation has been proposed as a possible way to expand such datasets.

This thesis investigates the generation of synthetic non-contrast brain CT images from an available training set. It also evaluates the performance of a deep-learning classifier for ICH detection under different training and decision-threshold settings. A balanced subset of the RSNA Intracranial Haemorrhage Detection dataset was used for slice-level binary classification. A ResNet-18 classifier was evaluated using a real-data baseline, an augmented training configuration, and validation-based threshold tuning. In parallel, several single-class and conditional GAN variants were explored to assess whether anatomically plausible synthetic CT slices could be generated.

The baseline classifier achieved an AUROC of 0.811 on the held-out test set. The augmented training configuration achieved a test AUROC of 0.881, compared to 0.811 for the baseline, indicating stronger separation between haemorrhage-positive and haemorrhage-negative slices. At the default decision threshold, the augmented classifier achieved high sensitivity but low specificity. After selecting the decision threshold on the validation set, the augmented model achieved an accuracy of 0.796, a sensitivity of 0.791, and a specificity of 0.801 on the held-out test set.

The generative models gradually learnt coarse head-like structure, but the generated images continued to lack realistic internal brain anatomy, sufficient sample diversity, and reliable class-specific features. They were therefore not used for synthetic-only or mixed real-synthetic classifier training. The generative component of this thesis is consequently interpreted as a qualitative feasibility study.

These findings show that the augmented training configuration achieved stronger held-out classification performance than the baseline and that validation-based threshold selection produced a more balanced operating point. Future work should focus on improving the anatomical realism and diversity of the generated images before investigating whether such synthetic data are useful for downstream classification tasks.

Contents

1 Introduction **1**
- 1.1 Thesis overview . . . 3

2 Background **3**
- 2.1 Intracranial haemorrhage detection . . . 3
- 2.2 CT imaging principles . . . 5
- 2.3 Deep learning for medical imaging . . . 5
- 2.4 Classifier architecture: ResNet-18 . . . 6
- 2.5 Synthetic data generation . . . 8
  - 2.5.1 Generative Adversarial Networks and conditional GANs . . . 9
  - 2.5.2 Diffusion models . . . 11

3 Related Work **13**
- 3.1 Deep learning for intracranial haemorrhage detection . . . 13
- 3.2 Synthetic data in medical imaging . . . 13
- 3.3 GAN-based medical image synthesis . . . 14
- 3.4 Diffusion models and recent generative approaches . . . 15
- 3.5 Positioning of this thesis . . . 15

4 Dataset and Preprocessing **16**
- 4.1 RSNA ICH dataset . . . 16
- 4.2 Subset selection and class balancing . . . 16
- 4.3 Preprocessing pipeline . . . 16
- 4.4 Data splitting . . . 17

5 Methodological Overview **17**

6 Exploratory Study **21**
- 6.1 Exploratory goals . . . 21
- 6.2 Computational resources . . . 21
- 6.3 Exploration of the classification pipeline . . . 22
- 6.4 Initial exploration of generative modelling . . . 25
- 6.5 Observed failure modes in the initial cGAN . . . 25
- 6.6 Architectural and training modifications in the cGAN setting . . . 27
- 6.7 From conditional to single-class generation . . . 29
- 6.8 Stabilisation in the single-class GAN setting . . . 30
- 6.9 CPU sanity check of the cropped single-class GAN . . . 32
- 6.10 Extended CPU training of the cropped single-class GAN . . . 33
- 6.11 GPU replication of the cropped single-class GAN . . . 36
- 6.12 Upsample-based generator experiment . . . 38
- 6.13 Two-class cGAN experiments on the full RSNA training split . . . 39
  - 6.13.1 Version 1 . . . 39
  - 6.13.2 Version 2 . . . 41
- 6.14 Lessons from the exploratory study . . . 43

7 Structured Experiments **44**
7.1 Experimental design . . . 44
7.2 Baseline classifier . . . 44
7.3 Augmented classifier . . . 44
7.4 Threshold tuning . . . 45
7.5 Evaluation metrics . . . 46
7.6 Scope of the structured experiments . . . 46

8 Experiments and Results **46**
8.1 Real-data baseline . . . 46
8.2 Real-data baseline with augmentation . . . 47
8.3 Threshold tuning . . . 47
8.4 Implications of the exploratory generative study . . . 48
8.5 Comparative analysis . . . 48

9 Discussion and Limitations **49**
9.1 Interpretation of the classification findings . . . 49
9.2 Interpretation of the generative findings . . . 50
9.3 Relationship between the classification and generative results . . . 52
9.4 Limitations of the classification experiments . . . 52
9.5 Limitations of the generative experiments . . . 53
9.6 Clinical and practical limitations . . . 54
9.7 Overall interpretation . . . 54

10 Conclusions and Further Research **54**
10.1 Conclusions . . . 54
10.2 Further research . . . 55

References **57**

A Data Preparation and Classification Scripts **60**
A.1 Balanced subset selection and extraction . . . 60
A.2 Preprocessing pipeline for the balanced RSNA subset . . . 61
A.3 Patient-level split construction . . . 62
A.4 Baseline classifier training script . . . 62
A.5 Augmented classifier training script . . . 63
A.6 Threshold tuning and ROC evaluation . . . 64

B Exploratory Generative Experiments on the MacBook **65**
B.1 Overview of the early two-class conditional GAN experiments . . . 65
B.2 Initial conditional generator and discriminator definitions . . . 65
B.3 GAN-specific negative-only subset construction . . . 67
B.4 GAN-specific preprocessing, central-slice filtering, and cropping . . . 67
B.5 Transitional conditional architecture trained on negative-only data . . . 68
B.6 Single-class GAN training on cropped haemorrhage-negative slices . . . 70
B.7 Single-class generator and discriminator definitions . . . 70

C Single-Class GAN Experiments on the Workstation **72**
C.1 CPU sanity-check configurations . . . 72
C.1.1 2-epoch CPU sanity check script . . . 72
C.1.2 5-epoch CPU sanity check script . . . 73
C.2 20-epoch CPU configuration . . . 73
C.3 40-epoch CPU configuration . . . 73
C.4 Overview of the CPU single-class GAN experiments . . . 74

D GPU-Based Single-Class GAN Experiments **75**
D.1 40-epoch GPU training script . . . 75
D.2 40-epoch GPU upsample training script . . . 75
D.3 Overview of the GPU single-class GAN experiments . . . 77

E Two-Class Conditional GAN Experiments on the Workstation **77**
E.1 Key adaptations from `train_cgan.py` to the GPU-based two-class cGAN run . . . 78
E.2 Key adaptations from the first two-class cGAN run to the second . . . 81
E.3 Overview of the two-class cGAN experiments . . . 82

1 Introduction

Intracranial haemorrhage (ICH) is bleeding within the cranial cavity and is a potentially life-threatening neurological condition that often requires rapid diagnosis and treatment. Depending on its anatomical location, ICH may occur in the epidural, subdural, subarachnoid, intraparenchymal, or intraventricular form. These subtypes differ in appearance, severity, and clinical management, and haemorrhages can vary substantially in size, shape, location, and intensity [12, 32, 31]. In acute care, non-contrast computed tomography (CT) is commonly used as a first-line imaging modality because it is fast, widely available, and effective for detecting acute intracranial bleeding [12, 14]. Early and accurate detection is important because imaging findings influence treatment decisions and patient outcomes.

Automated ICH detection is commonly formulated as an image-classification problem in which a model predicts whether a CT image contains haemorrhage. Convolutional neural networks can learn hierarchical image features directly from CT data and have demonstrated strong diagnostic performance for ICH detection in systematic reviews and meta-analyses [14, 17]. These findings show that deep-learning models can identify clinically relevant patterns in non-contrast brain CT images.

However, reported performance depends strongly on the available dataset, preprocessing pipeline, annotation quality, evaluation strategy, and the clinical setting. A model that performs well on an internal test set may not generalise equally well to unseen patients, scanners, or institutions [21, 14, 17]. Developing robust medical-imaging models also requires sufficiently large, diverse and accurately annotated datasets. Such datasets are difficult to obtain because expert annotation is expensive, sharing medical data is restricted by privacy and ethical requirements, and clinically relevant findings can be unevenly represented [21]. When the available training data are limited or insufficiently diverse, deep-learning models may overfit and fail to generalise reliably.

Synthetic image generation has been proposed as one way to expand the amount and diversity of medical imaging data. Generative models learn patterns from an available training distribution and generate artificial images intended to resemble real data. In medical imaging, synthetic data have been investigated for applications that include data augmentation, anonymisation, class balancing, domain adaptation, and increased dataset diversity [27, 19, 29, 7]. Generative Adversarial Networks (GANs) are among the most widely studied approaches for this purpose [10, 16, 1].

Conditional GANs are particularly relevant to labelled medical-image generation because both the generator and the discriminator can be conditioned on a target class. In the context of ICH detection, this could in principle allow the separate generation of haemorrhage-positive and haemorrhage-negative CT slices [4, 28, 34]. However, conditional medical-image generation is challenging because the model must learn both realistic global anatomy and image features that are consistent with the requested clinical label. This is especially difficult when the pathology is subtle, heterogeneous, or spatially localised, as is often the case for intracranial haemorrhage in individual CT slices [12, 14, 17].

Existing research therefore identifies an important distinction between perceptual realism and practical usefulness. A synthetic medical image may appear superficially plausible while still containing anatomical errors, artificial intensity patterns, limited sample diversity, or inconsistencies between the image and its assigned label [19, 29, 30]. GAN training can also be affected by instability and mode collapse, in which the generator produces only a narrow range of similar outputs [30, 34]. These limitations may prevent synthetic images from representing the real data

distribution sufficiently well for use in downstream model training. Therefore, visual plausibility alone does not demonstrate that synthetic images are suitable for classification or other clinical machine-learning tasks [19, 29].

Before the downstream utility of synthetic brain CT images can be evaluated, it must first be established whether a generative model can produce images with plausible global head structure, realistic internal anatomy, sufficient diversity and, in the conditional setting, consistency with the requested haemorrhage label. At the same time, a controlled real-data classification baseline is needed to determine how well deep-learning-based ICH classification performs on the available dataset and how this performance is affected by training-time data augmentation and the selected decision threshold.

The aim of this thesis is therefore twofold. First, it investigates the extent to which GAN-based models, with particular emphasis on conditional GANs, can generate anatomically plausible non-contrast brain CT slices from an available training set. Second, it evaluates the performance of a deep-learning classifier for slice-level ICH detection under different training and decision-threshold settings. The following research questions are addressed:

RQ1: To what extent can GAN-based models generate anatomically plausible non-contrast brain CT slices from the available training data?

RQ2: How does the performance of a deep-learning classifier for slice-level intracranial haemorrhage detection change when conservative data augmentation and validation-based decision-threshold tuning are applied?

To address RQ1, several conditional and single-class GAN variants were explored using CT data from the RSNA Intracranial Haemorrhage Detection Challenge [22]. The generated images were qualitatively assessed in terms of global head structure, internal anatomical realism, sample diversity, training behaviour, and, for conditional models, consistency with the requested haemorrhage label.

To address RQ2, a ResNet-18 classifier was evaluated on a balanced subset of the RSNA dataset. ResNet-18 was selected because it combines sufficient representational capacity with relatively modest computational requirements and has been used successfully in medical-imaging applications [11, 3, 9]. A fixed patient-level split was used to separate the training, validation, and test data. The completed classification experiments consist of a real-data baseline, conservative training-time data augmentation, and threshold tuning based on validation. Performance is evaluated on the held-out real test set using AUROC, accuracy, sensitivity, and specificity.

The broader motivation of this work is to determine whether synthetic medical images can eventually support downstream classification. The original research direction therefore included synthetic-only and mixed real-synthetic classifier training. However, during the exploratory study, the generated images did not reach the required level of anatomical realism, diversity, and class consistency. Their downstream utility could consequently not be evaluated fairly within the current study. Investigating whether improved synthetic images can support ICH classification is therefore positioned as a direction for future research.

The contribution of this thesis lies in the combination of a documented exploratory investigation of synthetic CT generation with a controlled evaluation of real-data classification. This approach provides insight into the capabilities and limitations of GAN-based generation for non-contrast brain CT, while also comparing deep-learning-based ICH classification under baseline and augmented training configurations and different decision thresholds.

1.1 Thesis overview

The remainder of this thesis is structured as follows. Section 2 provides the necessary background on intracranial haemorrhage detection, CT imaging, deep-learning classification, and generative modelling. Section 3 discusses related research on automated ICH detection and synthetic medical-image generation. Section 4 describes the RSNA dataset, subset construction, preprocessing pipeline, and patient-level data split. Section 5 introduces the overall methodological structure of the thesis. Section 6 presents the exploratory classification and generative experiments that shaped the final experimental scope. Section 7 describes controlled classification experiments, including baseline, conservative augmentation, and validation-based threshold tuning. Section 8 presents and compares the resulting classifier performance. Section 9 discusses the findings and limitations of the study. Finally, Section 10 summarises the main conclusions and outlines directions for future research.

This bachelor thesis was conducted at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, under the supervision of Prof. dr. K.J. Batenburg and N. Fragkos.

2 Background

2.1 Intracranial haemorrhage detection

Intracranial haemorrhage (ICH) refers to bleeding within the cranial cavity and is a serious neurological condition that often requires rapid diagnosis and treatment. Depending on the anatomical location of the bleeding, ICH can be divided into several subtypes, including epidural, subdural, subarachnoid, intraparenchymal, and intraventricular haemorrhage. These subtypes differ in location, appearance, clinical severity, and possible treatment strategies [12, 32, 31].

On acute non-contrast CT, the haemorrhage generally appears hyperdense relative to the surrounding brain tissue. Epidural haemorrhage typically appears as a biconvex or lens-shaped extra-axial collection adjacent to the skull. Subdural haemorrhage is more commonly seen on the inner surface of the skull as a crescent-shaped collection. Subarachnoid haemorrhage can be recognised as hyperdense blood within the cortical sulci, fissures, or basal cisterns. Intraparenchymal haemorrhage appears within the brain tissue itself, often as a focal hyperdense region, while intraventricular haemorrhage is visible as blood within the ventricular system [12, 32]. Representative examples of these appearances are shown in Figure 1. This visual heterogeneity makes slice-level automated detection difficult, because the model must distinguish haemorrhage from both normal anatomical variation and other hyperdense structures.

The examples in Figure 1 illustrate typical appearances, but the visual appearance can vary substantially between patients and slices. Haemorrhages may differ in size, shape, location, and intensity, and multiple subtypes may occur within the same patient or CT slice. Small haemorrhages can also be difficult to distinguish from other hyperdense structures.

In acute care settings, non-contrast computed tomography (CT) is commonly used as a first-line imaging modality for suspected intracranial bleeding. CT is fast, widely available, and effective in detecting acute haemorrhage, which makes it particularly suitable for emergency diagnosis. Early detection is clinically important because imaging findings help determine the location and severity of haemorrhage and can directly influence treatment decisions and patient outcomes [12, 14].

Automated ICH detection is commonly formulated as an image classification task in which

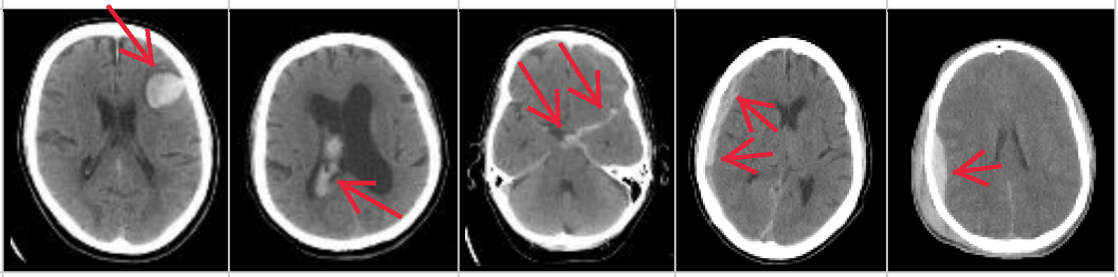
	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, 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)

Figure 1: Illustration of the main intracranial haemorrhage subtypes. Source: RSNA Intracranial Haemorrhage Detection Challenge [22].

a model predicts whether a CT image contains haemorrhage. In the Intracranial Haemorrhage Detection Challenge of RSNA, each CT slice is labelled for the five haemorrhage subtypes shown in Figure 1, together with an *any* label indicating the presence of at least one subtype [22]. In this thesis, the original multi-label problem is simplified to binary classification using the *any* label, distinguishing haemorrhage-positive from haemorrhage-negative slices.

Despite the clinical relevance of this task, automated ICH detection remains a challenge. Haemorrhage can vary in size, shape, location, and intensity, and some findings may be subtle on individual two-dimensional slices. In addition, medical imaging datasets are often affected by limited data availability, high annotation costs, privacy restrictions, and class imbalance. These challenges can make it difficult to train models that generalise reliably to unseen patients [21, 14, 17].

2.2 CT imaging principles

CT imaging represents tissue attenuation using Hounsfield Units (HU). On this scale, air, water, bone, blood, and soft tissue occupy different intensity ranges. This quantitative representation makes CT images suitable for computational analysis, because pixel values have a physical interpretation related to tissue density [2, 18, 6].

However, the full HU range is too wide to visualise all relevant anatomical structures with equal contrast. Therefore, CT images are commonly displayed using windowing. Windowing selects a specific HU interval and maps it to the visible image intensity range. Structures outside this interval are clipped, while structures inside this interval are displayed with increased contrast. This process is controlled by the window level, which determines the centre of the displayed intensity range, and the window width, which determines the size of that range [2, 18, 6].

For non-contrast brain CT, a brain window is typically used to emphasise soft-tissue structures and haemorrhage-relevant intensity differences. This is important for the preprocessing pipeline used in this thesis. The raw DICOM pixel values are the numeric values stored in the DICOM image data prior to display conversion. DICOM is the standard format for medical images and stores the scan together with metadata such as acquisition settings and intensity calibration information [8, 20]. For CT, these pixel values are typically converted to Hounsfield Units using the rescale slope and rescale intercept stored in the DICOM metadata, after which the brain window is applied before normalisation and resizing [8, 20, 6]. As a result, both the classifier and the generative model operate on the same preprocessed two-dimensional CT slice representation rather than directly on raw DICOM values.

This preprocessing step is important because the appearance of CT images strongly depends on the chosen intensity window. If the window is too broad, subtle differences between brain tissue and haemorrhage may become less visible. If the window is too narrow, relevant information may be clipped. A consistent preprocessing pipeline is therefore necessary to ensure that all real and synthetic images are represented in the same intensity space.

2.3 Deep learning for medical imaging

Deep learning has become an important approach in medical image analysis. In particular, convolutional neural networks (CNNs) are widely used for image classification tasks because they can learn hierarchical visual features directly from image data [5, 26]. A convolution is a small trainable filter that is applied repeatedly across the image. In early layers, these filters typically

learn simple local patterns such as edges, contrast transitions, and textures. In deeper layers, the network can combine these simpler patterns into more task-specific representations, such as anatomical structures or pathology-related image features. This makes CNNs well suited for medical image classification problems such as haemorrhage detection.

For ICH detection, CNN-based models can be trained to distinguish haemorrhage-positive from haemorrhage-negative CT slices. Recent systematic reviews have shown that deep learning methods can achieve strong diagnostic performance for ICH detection on non-contrast CT scans [14, 17]. These results suggest that deep learning models can capture clinically relevant image patterns from CT data.

Nevertheless, deep learning models usually require large and representative datasets. This requirement is difficult to satisfy in medical imaging, where expert annotations are expensive and data sharing is restricted by privacy and ethical considerations [21]. When the available dataset is small or insufficiently diverse, a model may overfit to the training data and fail to generalise to new patients. Overfitting occurs when a model learns patterns that are specific to the training data rather than patterns that generalise well to unseen examples. As a result, training performance continues to improve while validation performance stagnates or worsens. This limitation is especially relevant for medical applications, where reliable performance in unseen cases is more important than high performance in the training set.

In this thesis, a CNN classifier is used to establish a real-data reference for slice-level ICH detection. The original experimental design also considered using this classifier to compare real-data, synthetic-only, and mixed real-synthetic training. However, because the generated images did not reach sufficient anatomical realism, diversity, and class consistency across the tested generative configurations, this downstream synthetic-data evaluation was not performed.

2.4 Classifier architecture: ResNet-18

The classifier used in this thesis is based on ResNet-18, a convolutional neural network architecture originally developed for large-scale image recognition [11]. Like other convolutional neural networks, ResNet-18 learns the features of the image hierarchically: early layers detect local patterns such as edges and intensity transitions, while deeper layers combine these patterns into more abstract and task-specific representations [11].

Formally, let

$$x \in [0,1]^{N \times N} \tag{1}$$

denote a preprocessed single-channel CT slice, where N is the spatial side length of the square image in pixels. Let

$$y \in \{0,1\} \tag{2}$$

denote the binary haemorrhage label, where $y = 1$ indicates haemorrhage and $y = 0$ indicates no haemorrhage. Because the standard ResNet-18 implementation expects a three-channel input, the single-channel slice is repeated across channels:

$$\tilde{x} = R(x) \in [0,1]^{3 \times N \times N}, \tag{3}$$

where $R(\cdot)$ denotes the channel-repetition operation.

The classifier can then be written as a function

$$f_\theta : [0,1]^{3 \times N \times N} \to \mathbb{R}, \tag{4}$$

where $f_\theta(\tilde{x})$ is a scalar logit. The corresponding predicted probability of haemorrhage is obtained by applying the sigmoid function:

$$p_\theta(y = 1 \mid \tilde{x}) = \sigma(f_\theta(\tilde{x})) \,. \tag{5}$$

The binary prediction is then obtained by thresholding this probability:

$$\hat{y} = \mathbb{I}[p_\theta(y = 1 \mid \tilde{x}) \geq \tau] \,, \tag{6}$$

where $\tau \in [0, 1]$ is the decision threshold and $\mathbb{I}(\cdot)$ denotes the indicator function.

A defining characteristic of ResNet-18 is the use of residual connections, also known as skip connections. Instead of learning only a direct mapping $H(x)$, a residual block learns a residual function $F(x)$ such that

$$H(x) = F(x) + x, \tag{7}$$

where $F(x)$ denotes the residual mapping learnt by the convolutional layers within the block. The input is therefore passed through a sequence of convolutional layers and is also added directly to the block output. This improves the flow of information and gradients through the network and makes optimisation more stable, especially in deeper architectures [11].

In this thesis, the ResNet-18 backbone is used as a feature extractor, followed by a modified final classification layer. If

$$h_\theta(\tilde{x}) \in \mathbb{R}^{512} \tag{8}$$

denotes the feature vector after the final global average pooling layer, then the binary classification head is

$$f_\theta(\tilde{x}) = w^\top h_\theta(\tilde{x}) + b, \tag{9}$$

with $w \in \mathbb{R}^{512}$ and $b \in \mathbb{R}$. This replaces the original multi-class ImageNet classification layer with a single-output linear layer suitable for binary haemorrhage classification.

ResNet-18 was selected because it provides a suitable balance between representational capacity and computational efficiency. It is expressive enough to learn the haemorrhage-related features of CT slices, while remaining relatively lightweight compared to deeper architectures. This made it appropriate to establish a reliable classification baseline within the available computational resources. ResNet-18 has also been used successfully in medical imaging applications, including CT-based image retrieval and broader diagnostic imaging tasks, which supports its suitability as a baseline architecture for this thesis [11, 3, 9].

Figure 2 shows the classifier pipeline used in this thesis.

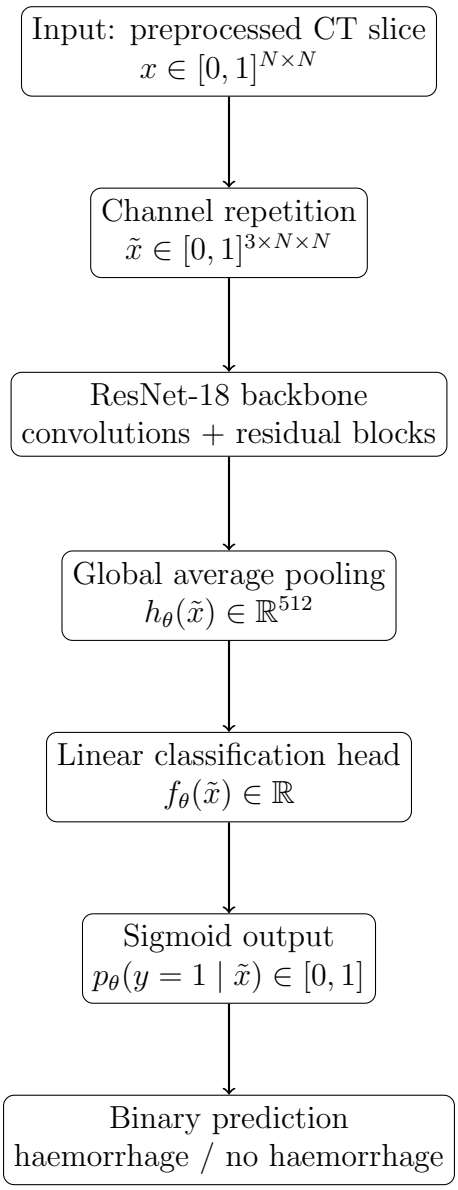


Figure 2: Schematic overview of the ResNet-18 classifier used in this thesis. A preprocessed single-channel CT slice is repeated across three channels, passed through the ResNet-18 backbone, reduced to a 512-dimensional feature vector by global average pooling, and mapped to a scalar logit, which is converted into a haemorrhage probability by the sigmoid function.

2.5 Synthetic data generation

Synthetic data generation has been proposed as a way to reduce some of the limitations of medical imaging datasets, such as limited sample size, class imbalance, and privacy constraints [19, 7]. Instead of relying only on existing patient images, generative models can be trained to produce artificial images that resemble real medical data. These synthetic images may be useful for data augmentation, class balancing, anonymisation, and increasing the diversity of training datasets [27,

19, 29, 7].

Formally, let

$$x \in \mathcal{X} \tag{10}$$

denote a medical image, where $\mathcal{X}$ is the image space. In this thesis, $\mathcal{X}$ corresponds to the space of preprocessed CT slices. A generative model aims to learn a distribution

$$p_\theta(x) \approx p_{\text{data}}(x), \tag{11}$$

where $p_{\text{data}}(x)$ is the true data distribution and $p_\theta(x)$ is the distribution of the learnt model. In the class-conditional setting, the task becomes learning

$$p_\theta(x \mid y), \tag{12}$$

where $y \in \{0, 1\}$ denotes the binary haemorrhage label. After training, new synthetic images can be sampled from the learnt distribution $p_\theta(x)$ or, in the conditional case, from $p_\theta(x \mid y)$.

Synthetic data generation should be distinguished from ordinary data augmentation. Data augmentation is a regularisation strategy in which modified versions of existing training images are generated on the fly during training, without learning a generative model of the full data distribution. In this study, only mild transformations were explored for augmentation, because strong transformations can easily produce anatomically implausible brain CT slices.

Although synthetic images can increase the size and diversity of a training dataset, visual realism alone does not guarantee usefulness for a downstream classification task [19, 29]. Generated images may contain artefacts, unrealistic anatomical patterns, reduced diversity, or subtle distribution differences compared to real CT images [30, 19, 29, 7]. Such issues may limit their value or even harm the performance of the model when synthetic images are added to the training data [19, 29, 7].

For this reason, synthetic medical images should ultimately be evaluated not only by visual inspection but also by their effect on task performance [19, 29]. The original experimental design therefore considered classifiers trained on real data, synthetic data, and mixed real-synthetic data. However, the exploratory generative study showed that the generated images did not meet the predefined quality requirements for downstream training. The current generative evaluation is consequently limited to a qualitative feasibility assessment, while downstream synthetic-data evaluation remains a direction for future work. The completed structured experiments focus on the real-data baseline, the augmented training configuration, and the validation-based threshold tuning.

2.5.1 Generative Adversarial Networks and conditional GANs

A Generative Adversarial Network (GAN) consists of two neural networks that are trained simultaneously in a competitive setup [10]. The first network, the generator, takes as input a latent noise vector and produces a synthetic image. The second network, the discriminator, receives either a real image from the dataset or a generated image from the generator and predicts whether the image is real or synthetic. During training, the generator attempts to fool the discriminator and the discriminator attempts to improve its ability to distinguish real samples from generated samples [10, 30]. Through this adversarial process, the generator ideally learns to approximate the distribution of the real training data.

Formally, let

$$x \in \mathcal{X} = [-1, 1]^{1 \times N \times N} \tag{13}$$

denote a preprocessed single-channel CT slice in the image space used for GAN training. Here, N denotes the spatial side length of the square image in pixels. Let

$$y \in \{0, 1\} \tag{14}$$

denote the binary haemorrhage label, where $y = 0$ indicates haemorrhage-negative and $y = 1$ indicates haemorrhage-positive. Let

$$z \in \mathbb{R}^{d_z} \tag{15}$$

denote a latent noise vector sampled from a simple prior distribution $p_z(z)$, for example a standard Gaussian distribution.

In the unconditional setting, the generator can be written as a function

$$G_{\theta_G} : \mathbb{R}^{d_z} \to \mathcal{X}, \tag{16}$$

which maps the latent vector z to a synthetic image

$$\hat{x} = G_{\theta_G}(z). \tag{17}$$

The discriminator is then a function

$$D_{\theta_D} : \mathcal{X} \to (0, 1), \tag{18}$$

where $D_{\theta_D}(x)$ represents the estimated probability that the input image is real rather than generated.

The original GAN objective is the minimax problem [10]

$$\min_{\theta_G} \max_{\theta_D} \; \mathbb{E}_{x \sim p_{\text{data}}}[\log D_{\theta_D}(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D_{\theta_D}(G_{\theta_G}(z)))]. \tag{19}$$

In a conditional GAN (cGAN), both the generator and the discriminator additionally receive a class label [28, 34]. In this thesis, the class label corresponds to the binary haemorrhage label,

$$G_{\theta_G} : \mathbb{R}^{d_z} \times \{0, 1\} \to \mathcal{X}, \tag{20}$$

so that the synthetic image is generated as

$$\hat{x} = G_{\theta_G}(z, y). \tag{21}$$

Similarly, the discriminator becomes

$$D_{\theta_D} : \mathcal{X} \times \{0, 1\} \to (0, 1), \tag{22}$$

so that $D_{\theta_D}(x, y)$ denotes the estimated probability that image x is real under class condition y.

The conditional adversarial objective is then

$$\min_{\theta_G} \max_{\theta_D} \; \mathbb{E}_{x,y \sim p_{\text{data}}}[\log D_{\theta_D}(x, y)] + \mathbb{E}_{z \sim p_z,\, y \sim p_y}[\log(1 - D_{\theta_D}(G_{\theta_G}(z, y), y))]. \tag{23}$$

where p_y denotes the label distribution used during training or sampling.

The architectural idea behind this setup is that the generator should learn not only global CT-like structure, but also class-consistent structure. In other words, the generator should produce a synthetic image that is both anatomically plausible and compatible with the requested haemorrhage label. The discriminator then evaluates whether the generated image looks realistic under that label. This makes conditional generation attractive for a downstream task in which separate synthetic examples are needed for different target classes [34].

The conditional setting is relevant because the downstream classification task also distinguishes between haemorrhage-negative and haemorrhage-positive slices. A cGAN can therefore, in principle, generate separate synthetic examples for each class. At the same time, conditional generation is more difficult than unconditional generation, because the model must not only learn realistic anatomy, but also ensure that the generated image is consistent with the requested class label. In medical imaging, GANs and cGANs have been explored for image synthesis, augmentation, anonymisation, classification support, and segmentation support [27, 16, 1, 30, 4].

Figure 3 shows a schematic overview of the conditional GAN used in this thesis. The generator receives a latent vector z and a class label y, and produces a synthetic CT slice $\hat{x}$. The discriminator receives a real or synthetic image together with the same label y, and predicts whether the image is real or fake under that condition.

2.5.2 Diffusion models

Diffusion models are another family of generative models that have recently become important for image synthesis [13, 33]. Instead of training two networks in an adversarial game, diffusion models learn to generate data by reversing a gradual noising process. Starting from a clean image, the noise is added step by step until the image becomes nearly indistinguishable from the random noise. A neural network is then trained to reverse this process and reconstruct a plausible sample from noise.

Formally, let

$$x_0 \in \mathcal{X} \tag{24}$$

denote a real image in the image space $\mathcal{X}$. A diffusion model defines a forward noising process

$$x_0 \to x_1 \to x_2 \to \cdots \to x_T, \tag{25}$$

in which each successive variable is a noisier version of the previous one. After sufficiently many steps, x_T approaches a simple noise distribution, typically a Gaussian distribution.

The generative task is then to learn the reverse process, that is, to model reverse transitions of the form

$$p_\theta(x_{t-1} \mid x_t, t), \tag{26}$$

so that generation can start from noise x_T and proceed step by step back towards a plausible synthetic image x_0. In practice, many diffusion models are trained to predict the noise that was added at each step, rather than to predict the clean image directly [13].

Compared with GANs, diffusion models often provide more stable training and can produce highly realistic images. They have shown strong performance in general image generation and have also been applied to medical imaging, including brain image synthesis and latent diffusion approaches [24]. They are relevant to this thesis because they represent an important alternative to GAN-based synthesis and may offer higher image quality or greater robustness.

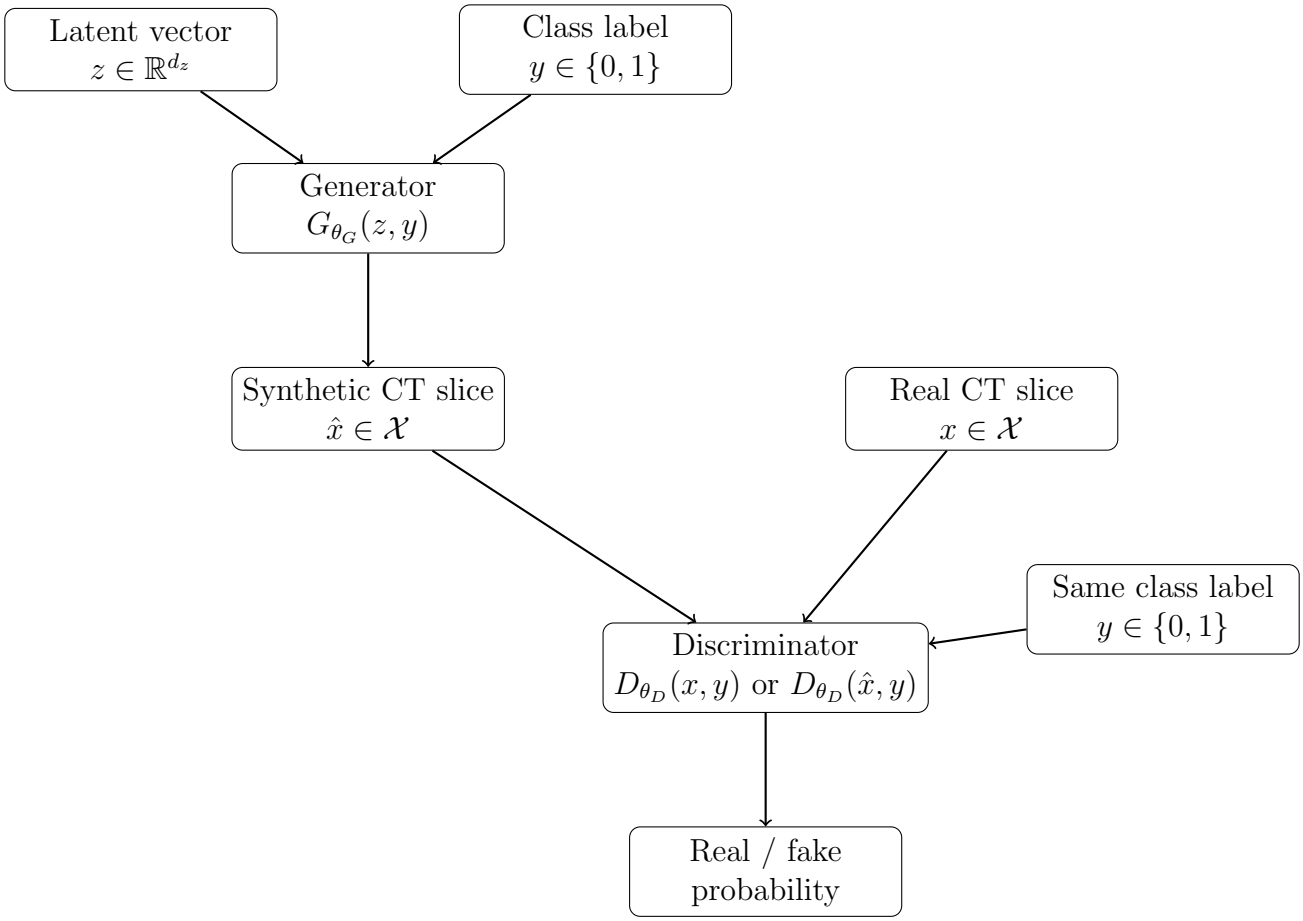


Figure 3: Schematic overview of the conditional GAN. The generator G_{θ_G} maps a latent vector z and class label y to a synthetic CT slice $\hat{x}$, while the discriminator D_{θ_D} receives either a real or synthetic image together with the same label y and predicts whether the input is real or generated under that condition.

However, diffusion models are also typically more computationally demanding than GANs, both during training and during image generation. This thesis therefore focuses on conditional GANs because they provide a comparatively simple and computationally manageable framework for class-conditional synthetic image generation. Nevertheless, diffusion models remain an important direction for future work if higher-quality synthetic CT images are required.

3 Related Work

3.1 Deep learning for intracranial haemorrhage detection

Deep learning has been widely investigated for the automated detection of intracranial haemorrhage in non-contrast CT scans. Recent systematic reviews and meta-analyses report that deep learning methods can achieve strong diagnostic performance for ICH detection, suggesting that convolutional neural networks are capable of learning clinically relevant image patterns from CT data [14, 17]. These studies show that automated methods have considerable potential as decision-support tools in acute care settings, where rapid haemorrhage detection is clinically important.

At the same time, the reported performance of deep learning models strongly depends on the dataset, the preprocessing pipeline, the quality of the annotation, the evaluation strategy, and the clinical setting. Models that perform well on internal test data may not generalise equally well to unseen patients, different scanners, or different institutions. This is a common challenge in medical imaging, where data availability, privacy restrictions, and expert annotation costs limit the size and diversity of training datasets [21]. As a result, improving generalisation remains an important research problem for automated ICH detection.

The RSNA Intracranial Haemorrhage Detection Challenge provided a large public dataset that has been widely used for the development and evaluation of ICH detection algorithms [22]. The original challenge formulated the task as a multi-label classification problem, with labels for different haemorrhage subtypes and an additional label indicating the presence of any haemorrhage. In contrast, this thesis simplifies the task to binary slice-level classification using the *any* label. This makes it possible to focus directly on whether synthetic images can support the distinction between haemorrhage-positive and haemorrhage-negative CT slices.

3.2 Synthetic data in medical imaging

Synthetic data generation has been proposed as a way to address several structural limitations of medical imaging datasets. In healthcare, synthetic data may support data augmentation, anonymisation, class balancing, and increased dataset diversity [7, 19, 29]. These potential benefits are especially relevant in medical imaging, where large, diverse, and accurately labelled datasets are difficult to obtain.

Recent reviews discuss synthetic data as an opportunity and a challenge for machine learning in healthcare [15, 23]. Synthetic images can increase the amount of available training data, but their value depends on whether they introduce meaningful and task-relevant variation rather than simply producing visually plausible examples. Koetzier et al. [19] and Sizikova et al. [29] similarly emphasise that synthetic medical images should be carefully evaluated, since visual plausibility alone does not guarantee clinical or machine-learning usefulness.

A central issue in this literature is the distinction between perceptual realism and downstream utility. A generated CT image may appear realistic to a human observer, while still containing artefacts, unrealistic anatomical structures, or subtle distribution shifts that affect model training. Therefore, synthetic data should not only be visually assessed, but also evaluated through their effect on downstream tasks such as classification or segmentation [19, 29]. This thesis adopts this task-oriented perspective as a guiding principle. The generated CT slices were first assessed to determine whether they were sufficiently realistic, diverse, and class-consistent to justify downstream classifier experiments. Because these requirements were not met, their effect on haemorrhage classification performance was not evaluated in the current study.

3.3 GAN-based medical image synthesis

Generative Adversarial Networks (GANs) are one of the most widely used families of generative models for medical image synthesis. Since their introduction by Goodfellow et al. [10], GANs have been applied to a wide range of image-generation tasks in medical imaging, including data augmentation and anonymisation [27, 1]. In particular, Shin et al. [27] demonstrated that GAN-based synthesis can be used to supplement real medical imaging data for both augmentation and anonymisation purposes.

Several reviews have examined the role of GANs in medical image analysis. Jeong et al. [16] reviewed GAN applications for medical image classification and segmentation, while Alamir and AlGhamdi [1] provided a broader survey of GAN-based methods in medical image analysis. Together, these reviews show that GANs have been used for image generation, augmentation, segmentation support, classification support, and domain adaptation. At the same time, recent studies and reviews also emphasise important limitations, including training instability, mode collapse, image artefacts, and uncertainty about whether visually plausible synthetic images are actually useful for downstream learning tasks [30, 19, 29, 34].

Mode collapse is a failure mode in GAN training in which the generator produces only a narrow range of visually similar outputs instead of reflecting the broader diversity of the real data distribution [34, 30]. This can reduce the diversity of synthetic data and limit its usefulness for medical image applications.

For the purposes of this thesis, conditional GANs (cGANs) are especially relevant. As introduced formally in Section 2.5.1 and illustrated schematically in Figure 3, a cGAN extends the original GAN framework by conditioning both the generator and the discriminator on a class label. In the notation used in the background chapter, the generator produces an image of the form given in Eq. 21, while the discriminator evaluates whether an image is real or generated under the same class condition as defined in Eq. 22. The corresponding conditional adversarial training objective is given in Eq. 23. This makes cGANs attractive for supervised medical imaging tasks in which synthetic samples are needed for specific disease categories rather than only for the overall image distribution [28, 4].

In medical imaging, class conditioning is especially relevant when diagnostically meaningful image variation is tied to disease status and may be spatially localised, as is the case for intracranial haemorrhage on CT [12, 14, 17]. A class-conditional generator is therefore, in principle, more closely aligned with downstream classification tasks than an unconditional generator, because it can be instructed to generate targeted examples for a requested class [4, 28, 34]. In the context of this thesis, the conditioning label corresponds to the binary haemorrhage target, so a cGAN

could in principle be used to generate either haemorrhage-negative or haemorrhage-positive CT slices. Compared to unconditional generation, this provides a more direct route to synthetic data generation for a binary classification problem [4, 28].

However, the literature also suggests that this conditional setting is particularly challenging in medical imaging. The model must not only generate anatomically plausible images, but must also ensure that the generated structures are consistent with the requested label. This becomes difficult when pathology is subtle, spatially heterogeneous, or represented by relatively local image features, as is often the case for intracranial haemorrhage on individual CT slices [12, 14, 17]. Therefore, a cGAN trained for ICH detection must learn both the global appearance of head CT and the class-specific visual cues associated with haemorrhage. This helps explain why cGANs are theoretically attractive for the present problem and why successful training and meaningful downstream utility cannot be assumed in advance.

Recent literature on synthetic medical imaging consistently emphasises that visually plausible GAN outputs are not automatically useful for downstream learning [19, 29, 30]. If the generator produces limited variation, unrealistic pathology, or images that differ systematically from real CT slices, the synthetic data may fail to improve generalisation or may even harm classifier performance. This concern is especially relevant in brain computed tomography, where subtle structural and intensity differences may be diagnostically important. For this reason, the GAN-generated images in this thesis were assessed against qualitative requirements relevant to their intended downstream use, including anatomical realism, sample diversity, and class consistency. Their actual effect on classifier performance was not evaluated because the generated images did not meet these requirements.

3.4 Diffusion models and recent generative approaches

Diffusion models have recently become an important alternative to GANs for image synthesis. These models generate images by learning to reverse a gradual noise process and have shown strong performance on many visual-generation tasks [13, 33, 25]. In medical imaging, diffusion-based approaches have also been explored for the generation of brain images. For example, Pinaya et al. [24] investigated latent diffusion models for brain imaging generation.

Diffusion models are relevant to this thesis because they represent a powerful and increasingly important direction in synthetic medical image generation. Compared with GANs, they often offer more stable training and can produce highly realistic outputs. However, they are also typically more computationally demanding during both training and sampling. Since the generative study was conducted under limited computational resources, GAN-based methods were chosen as the main generative framework. Diffusion-based methods remain an important direction for future work, particularly if higher-quality or more diverse synthetic CT images are needed.

3.5 Positioning of this thesis

The literature shows that deep learning can achieve strong performance for ICH detection and that synthetic medical images may help address data scarcity and class imbalance. At the same time, previous work highlights a key unresolved issue: visual realism of synthetic images does not automatically translate into improved downstream model performance [7, 19, 29]. This is especially relevant in medical imaging, where subtle artefacts or distribution mismatches can affect clinically meaningful classification tasks.

This thesis is positioned at the intersection of automated ICH classification and synthetic medical-image generation. The study compares a real-data baseline with an augmented real-data training configuration, while exploring whether GAN-based models can produce synthetic CT slices with sufficient anatomical realism, diversity, and class consistency to justify future synthetic-only or mixed real-synthetic classifier experiments.

The main contribution of this thesis is therefore not the proposal of a new generative architecture or an empirical demonstration of synthetic-data utility. Instead, it combines a structured real-data classification reference with a documented exploratory assessment of synthetic CT generation. This approach identifies the limitations that prevented the evaluation of downstream synthetic-data and clarifies the quality requirements that generated images would need to meet before their usefulness for ICH classification can be tested fairly.

4 Dataset and Preprocessing

4.1 RSNA ICH dataset

This study uses the RSNA Intracranial Haemorrhage Detection dataset, a large public dataset of non-contrast head CT slices labelled for the presence of intracranial haemorrhage and several haemorrhage subtypes. Each image is associated with slice-level labels for epidural, intraparenchymal, intraventricular, subarachnoid, and subdural bleeding, as well as the *any* label indicating whether there is any type of haemorrhage.

For the experiments in this thesis, the multi-label annotation setup is simplified into a binary classification problem by using the *any* label as the target. This allows the study to focus on the distinction between haemorrhage-positive and haemorrhage-negative CT slices.

Representative examples of CT slices from the RSNA Intracranial Haemorrhage Detection dataset are shown in Figure 4. The examples in the figure include one haemorrhage-negative slice and three haemorrhage-positive slices with different visual appearances. This illustrates both the diversity of the dataset and the difficulty of the classification task, since haemorrhage can vary substantially in size, location, and appearance across slices.

4.2 Subset selection and class balancing

Because the full dataset is very large, an initial balanced subset was constructed for exploratory experiments. A total of 4,000 CT slices were selected, consisting of 2,000 haemorrhage-positive slices and 2,000 haemorrhage-negative slices. This subset size was chosen to make early experiments computationally manageable while still allowing a meaningful first evaluation of the full preprocessing and classification pipeline. In addition, the subset was intended to support rapid iteration during the exploratory phase, given the limited hardware available for local training. The relevant subset-selection and extraction steps are shown in Appendix A.1.

4.3 Preprocessing pipeline

All selected images were stored as DICOM files and preprocessed before training. First, the raw pixel values were converted to Hounsfield Units (HU) using the rescale slope and rescale intercept values stored in the DICOM metadata. Since this study focuses on non-contrast brain CT, a

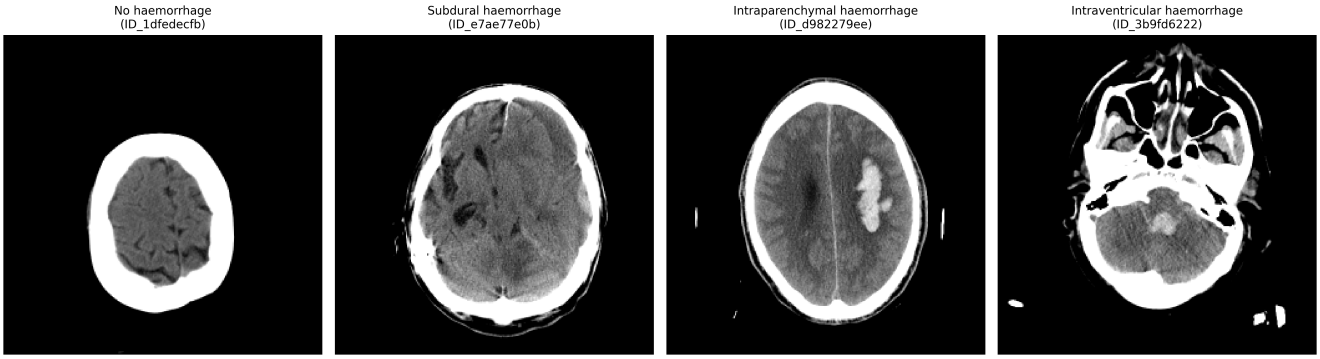


Figure 4: Example preprocessed CT slices from the RSNA Intracranial Haemorrhage Detection dataset used in this thesis. The figure shows one haemorrhage-negative slice and three haemorrhage-positive slices with different haemorrhage subtypes. The corresponding image IDs are shown in parentheses.

standard brain window was applied to emphasise pathologically relevant structures. The window centre was set at 40 HU and the window width at 80 HU.

After windowing, the image intensities were normalised to the range $[0, 1]$ and resized to a fixed spatial resolution of 256×256 pixels. The processed images were stored as NumPy arrays to allow efficient loading during model training. The main preprocessing steps are shown in Appendix A.2.

4.4 Data splitting

To mitigate the risk of data leakage, the dataset was partitioned at the patient level rather than at the individual image level. This procedure ensured that slices originating from the same patient did not appear simultaneously in both the training and evaluation subsets. The resulting patient-level split approximated a 70%/15%/15% distribution for the training, validation, and test sets, respectively, and comprised 2,793 training images (69.8%), 609 validation images (15.2%), and 598 test images (15.0%).

The class distribution remained approximately balanced across all splits. In the training set, 1,405 slices were negative and 1,388 were positive. In the validation set, 299 slices were haemorrhage-negative and 310 haemorrhage-positive. In the test set, 296 slices were negative and 302 positive. The construction of the patient-level split is shown in Appendix A.3.

5 Methodological Overview

This thesis is organised into two complementary parts. The first part is an exploratory methodological study, in which different preprocessing strategies, generative model variants, and stabilisation techniques are investigated to assess whether synthetic non-contrast brain CT slices can be generated with sufficient quality for downstream use. The second part consists of structured classification experiments in which completed real-data classifier settings are evaluated in a controlled and reproducible way on a fixed patient-level split.

This distinction is important because the project did not follow a purely linear path. The classification pipeline reached a stable and reproducible stage relatively quickly, while the generative modelling pipeline remained exploratory and required multiple iterations. Rather than presenting only the final outcome, this thesis therefore also documents the decision-making process that shaped the final experimental scope.

Computational efficiency was not treated as a primary evaluation outcome. GPU acceleration was introduced as a practical means of reducing training time and making the later generative experiments more feasible. The quality of the generated images was assessed separately in terms of anatomical realism, sample diversity, training behaviour, and, for conditional models, consistency with the requested haemorrhage label.

A generative model was considered suitable for downstream synthetic-data experiments only if the generated slices showed sufficient anatomical realism, sample diversity, and, in the conditional setting, consistency with the requested haemorrhage label. Since these criteria were not met during the exploratory phase, the generated images were not used for synthetic-only or mixed real-synthetic classifier training. This decision is treated as a methodological outcome of the thesis, rather than as an unrelated omitted experiment.

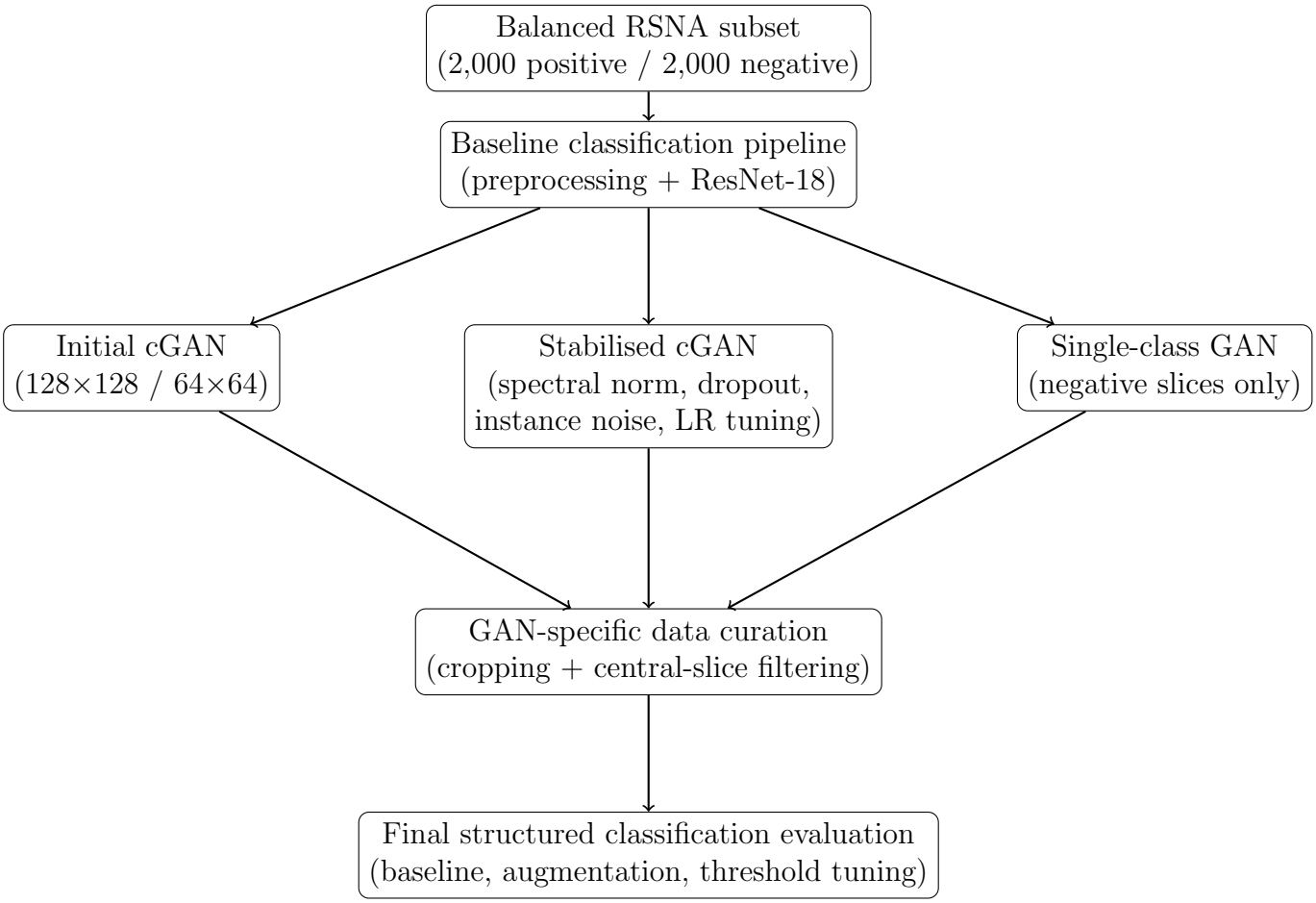


Figure 5: Overview of the early local exploratory workflow in this thesis. Starting from a balanced real-data subset, the study first established a baseline classification pipeline and subsequently explored multiple generative modelling variants on the MacBook, including conditional GANs, stabilised cGANs, single-class GANs, and GAN-specific preprocessing with cropped negative-only central slices. These exploratory steps informed the later controlled Workstation experiments and the final scope of the structured evaluation.

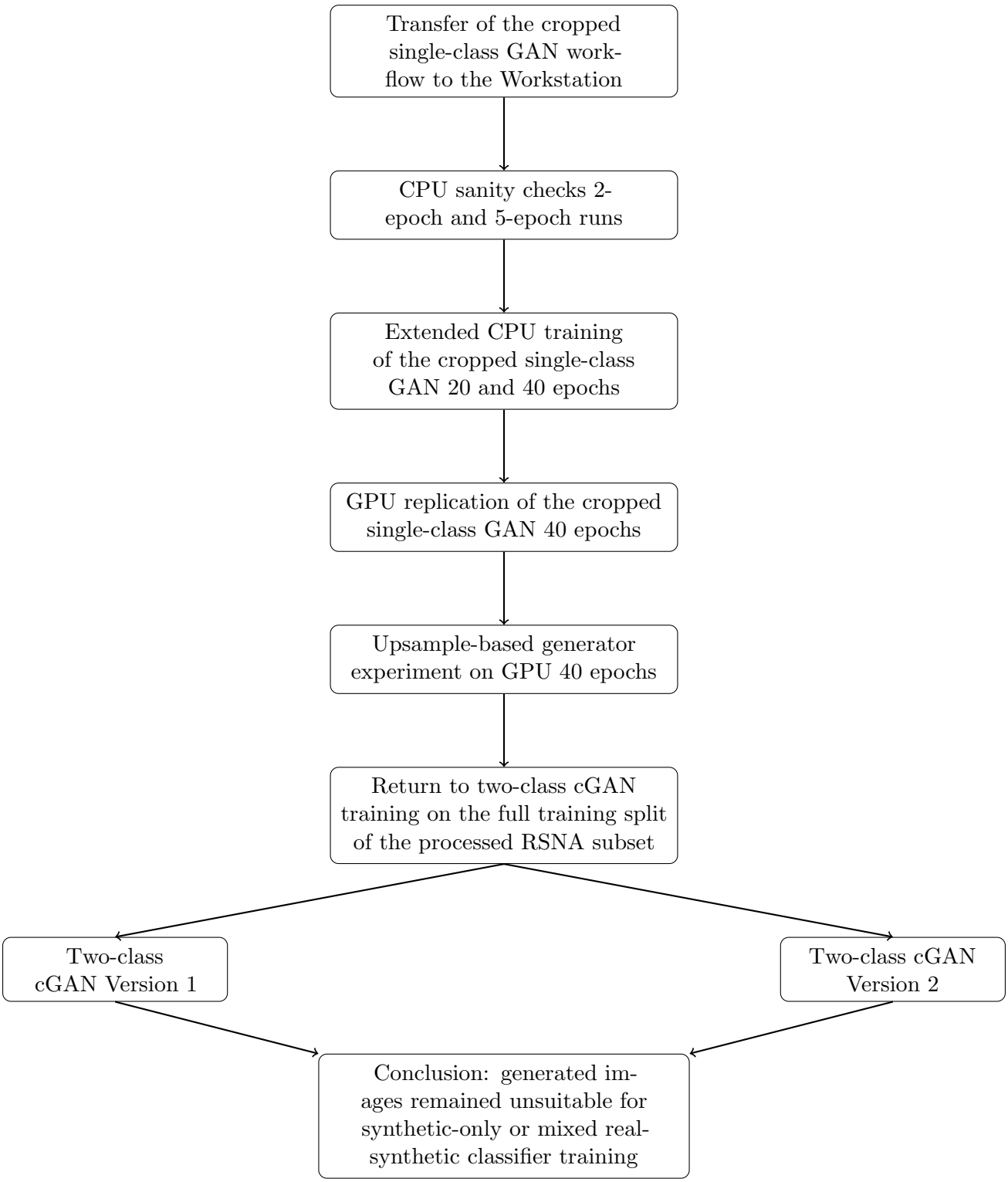


Figure 6: Workflow of the Workstation phase of the exploratory generative study. The transferred single-class GAN workflow was followed by CPU sanity checks, extended CPU training, GPU replication, an upsample-based generator experiment, and two class-conditional GAN experiments. None of these runs produced images with sufficient anatomical realism, diversity, and class consistency for downstream classifier training.

Figures 5 and 6 together summarise the methodological structure of the thesis. Figure 5 shows the early local exploratory workflow, while Figure 6 shows the later Workstation phase of the exploratory generative study. The classification and generative components both start from the same underlying dataset and preprocessing principles, but they provide different types of evidence. The classification experiments result in quantitative predictive-performance comparisons on the held-out patient-level test set, whereas the generative modelling experiments are assessed mainly through qualitative inspection of generated images, sample diversity, class consistency, and training behaviour. Computational efficiency is considered separately as a practical factor in the transition from CPU-based to GPU-accelerated training, rather than as a primary evaluation outcome.

The final structured experiments therefore focus on the completed real-data classification settings: the baseline classifier, the augmented classifier, and threshold tuning. The exploratory generative experiments remain part of the thesis because they explain why the planned synthetic-only and mixed real-synthetic classifier experiments were not carried out. In this way, the absence of downstream synthetic-data training is not treated as a missing result, but as a consequence of the methodological findings obtained during the exploratory phase.

6 Exploratory Study

6.1 Exploratory goals

The exploratory part of this thesis has two purposes. The first was to establish a reliable real-data classification pipeline for the detection of binary intracranial haemorrhage. The second was to investigate whether GAN-based models could generate non-contrast brain CT slices of sufficient quality to support downstream synthetic-data experiments.

The two parts of the exploratory study were evaluated differently. The classification pipeline was considered successful if it produced a stable baseline that could be quantitatively evaluated on a fixed patient-level split. The generative pipeline was qualitatively assessed, using generated image realism, sample diversity, training behaviour, and, for conditional models, consistency with the requested haemorrhage label.

The generated images were evaluated qualitatively through visual inspection of sample grids, real-versus-fake comparisons, sample diversity, and class consistency. No expert clinical assessment or quantitative generative image-quality metric was included.

Synthetic images were considered suitable for downstream synthetic-only or mixed real-synthetic classifier training only if they showed plausible global head structure, convincing internal CT anatomy, sufficient variation between samples, and clear class consistency in the conditional setting. Since these criteria were not met during the exploratory phase, the generated images were not used for downstream classifier training. This decision is part of the methodological outcome of the thesis.

6.2 Computational resources

The exploratory experiments were carried out using two computing configurations.

The **MacBook** configuration consisted of a 14-inch MacBook Pro with an Apple M5 chip, 24 GB of memory, and macOS Tahoe 26.5.2. PyTorch used the Metal Performance Shaders backend when available. This configuration was used for early development, preprocessing checks, script adaptation, and the initial exploratory GAN experiments.

The **Workstation** configuration consisted of a Dell OptiPlex Tower Plus 7020 running Ubuntu 24.04.4 LTS, with an Intel Core i7-14700 processor, 32 GB of RAM, and an NVIDIA GeForce RTX 4060 GPU. This configuration was used for the longer CPU experiments and the GPU-based GAN runs.

Throughout the remainder of this thesis, these configurations are referred to as the **MacBook** and the **Workstation**.

6.3 Exploration of the classification pipeline

The classification branch started from the balanced RSNA subset described in Section 4. The preprocessing pipeline converted raw DICOM slices to Hounsfield Units, applied a brain window with centre 40 HU and width 80 HU, normalised the images to $[0, 1]$, resized them to 256×256 pixels, and stored them as NumPy arrays for efficient loading. A patient-level split was used to prevent slices from the same patient from appearing in the training, validation, and test sets.

A ResNet-18 classifier was selected as the initial baseline model because it provided sufficient representational capacity while remaining computationally manageable. The final fully connected layer was replaced by a single output neuron for binary haemorrhage classification. Since the preprocessed CT slices were stored as single-channel images, each slice was repeated across three channels to match the expected input format of the standard ResNet implementation. The model was trained using binary cross-entropy with logits and the Adam optimiser with a learning rate of 10^{-4}. The main implementation choices are shown in Appendix A.4.

The baseline experiment confirmed that the preprocessing, patient-level splitting, and training pipeline could learn haemorrhage-related information from the real CT slices. The test-set ROC curve in Figure 7 shows a meaningful threshold-independent separation between haemorrhage-positive and haemorrhage-negative slices.

However, training behaviour also showed signs of overfitting. As shown in Figure 8, training loss continued to decrease while validation loss increased and fluctuated during later epochs. This indicated that the model was fitting the training data more strongly than it was learning features that generalised to unseen patients.

The observed overfitting motivated the exploration of conservative training-time data augmentation. The augmentation strategy included small affine transformations, mild brightness and contrast perturbations, and low-amplitude Gaussian noise. These transformations were selected to increase variation in the training data while preserving the medical plausibility of the CT slices. Strong transformations, such as large rotations or flips, were avoided because they could produce anatomically implausible images or alter clinically relevant spatial relationships. The corresponding implementation is shown in Appendix A.5.

The augmented classifier showed improved threshold-independent discrimination, as illustrated by the test-set ROC curve in Figure 9. The augmented training configuration achieved a higher AUROC than the baseline, indicating stronger ranking of haemorrhage-positive slices relative to haemorrhage-negative slices.

Figure 10 shows the corresponding training and validation behaviour. Compared with the baseline experiment, the augmented training configuration showed more stable validation behaviour. This configuration combined conservative transformations, validation-based early stopping, and weight decay of 10^{-4}. Because these changes were introduced together, the observed difference cannot be attributed exclusively to data augmentation.

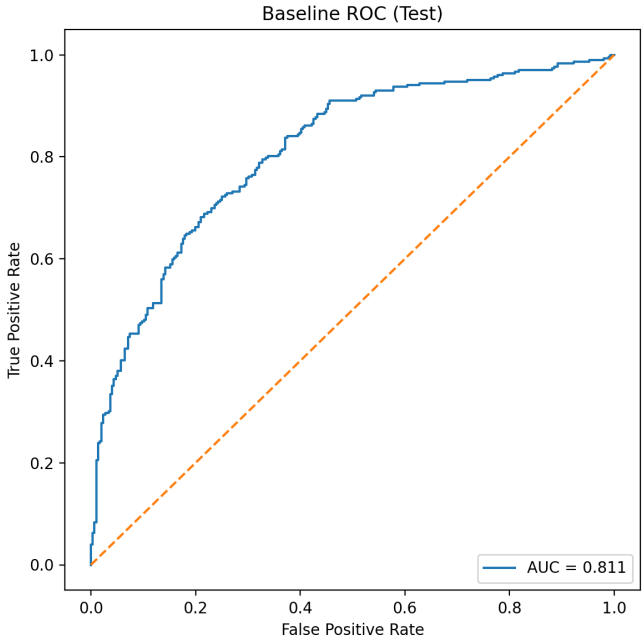


Figure 7: Receiver operating characteristic curve of the baseline ResNet-18 classifier on the held-out test set. The model achieved an AUROC of 0.811, indicating meaningful discrimination between haemorrhage-positive and haemorrhage-negative slices.

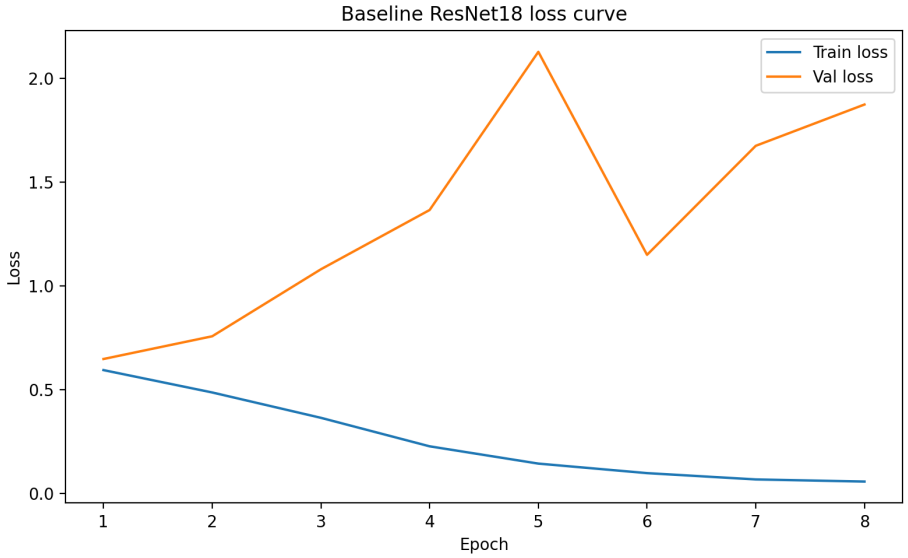


Figure 8: Training and validation loss curves for the baseline classifier. Training loss decreases steadily, while validation loss increases and fluctuates during later epochs, indicating overfitting.

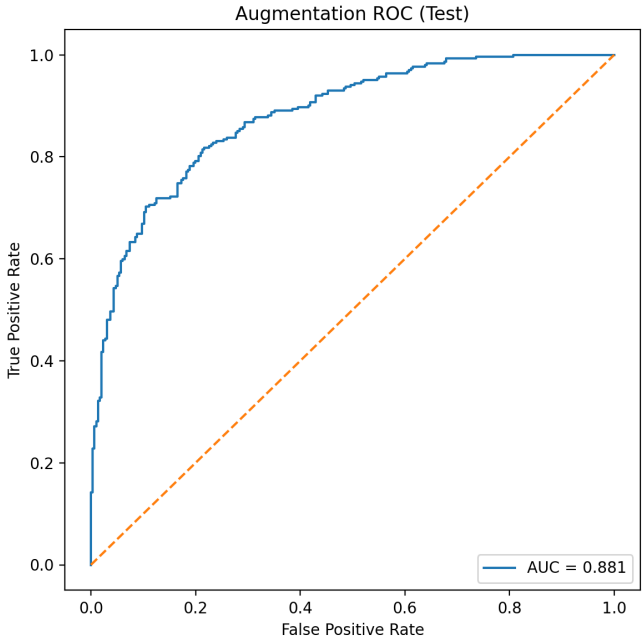


Figure 9: Receiver operating characteristic curve of the augmented ResNet-18 classifier on the held-out test set. The model achieved an AUROC of 0.881, indicating improved threshold-independent discrimination compared with the baseline classifier.

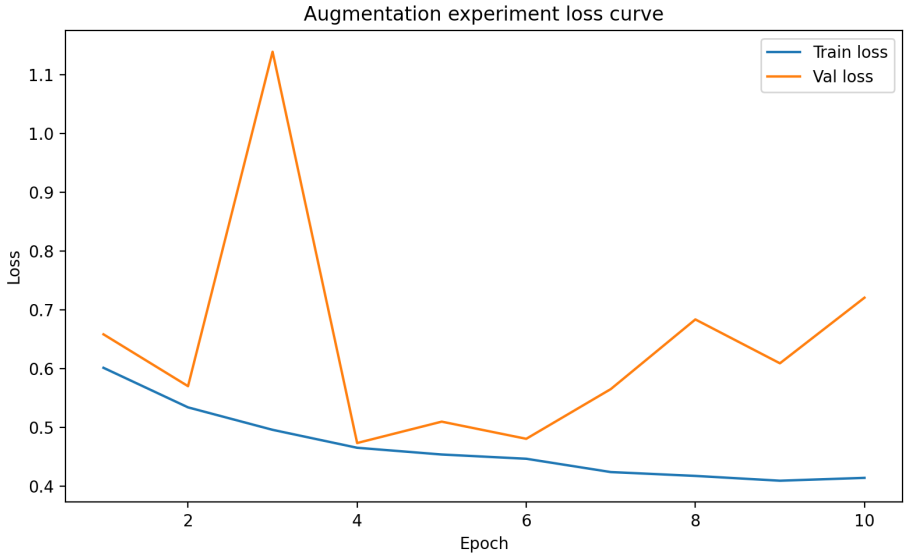


Figure 10: Training and validation loss curves for the augmented classifier. Compared with the baseline experiment, the augmented training configuration shows more stable validation behaviour. This configuration included conservative data augmentation, validation-based early stopping, and weight decay of 10^{-4}.

The exploratory classification work therefore established two real-data settings for the structured evaluation: the baseline ResNet-18 classifier and the augmented ResNet-18 classifier. Their final performance at the default and validation-selected decision thresholds is compared quantitatively in the structured experimental results. The threshold-tuning procedure used for this later comparison is shown in Appendix A.6.

6.4 Initial exploration of generative modelling

The first generative approach explored in this study was a conditional Generative Adversarial Network (cGAN). This choice followed directly from the intended downstream task: because the classifier distinguishes between haemorrhage-positive and haemorrhage-negative CT slices, the generative model needed to be able to produce class-specific synthetic examples. In the cGAN setup, the generator therefore received both a latent noise vector and the binary haemorrhage label, while the discriminator received an image together with the corresponding label and predicted whether the image was real or generated under that condition.

The aim of this initial setup was not to optimise synthetic image quality immediately, but to test whether a simple class-conditional adversarial model could learn the basic structure of non-contrast brain CT slices and produce visually distinguishable outputs for the two classes. The initial architecture used a transposed-convolutional generator and a convolutional discriminator. This provided a practical starting point for exploratory experiments, while keeping the model simple enough to train and inspect under the available computational constraints. The main configurations explored during the early conditional GAN phase are summarised in Appendix B.1. The corresponding conditional generator and discriminator definitions are shown in Appendix B.2.

Figure 11 provides a schematic overview of the conditional GAN setup used in the exploratory phase of this study.

The initial cGAN experiments were not performed on the original 256×256 images, but on downsampled slice representations. This decision was made for both practical and methodological reasons. Training GANs at high spatial resolution is computationally expensive and often unstable, especially during early exploratory experiments. In addition, reducing resolution simplifies the generation problem, which is useful when the first goal is to determine whether the model can learn the coarse anatomical structure at all. Therefore, the first runs were performed at 128×128 resolution, followed later by additional experiments at 64×64 resolution.

6.5 Observed failure modes in the initial cGAN

The first cGAN runs showed three recurring failure modes. First, the generated images were noisy and lacked convincing internal anatomical structure. Second, sample diversity was limited: different latent inputs often produced visually similar outputs, indicating early signs of mode collapse. Third, the conditioning signal was weak, because haemorrhage-positive and haemorrhage-negative generations were not clearly distinguishable. Together, these observations suggested that the initial cGAN was able to learn some coarse global structure, but not a sufficiently rich or class-consistent representation of the CT slice distribution.

These limitations motivated a more systematic exploration of architectural and training modifications. Rather than abandoning the cGAN immediately, the next step was to investigate whether

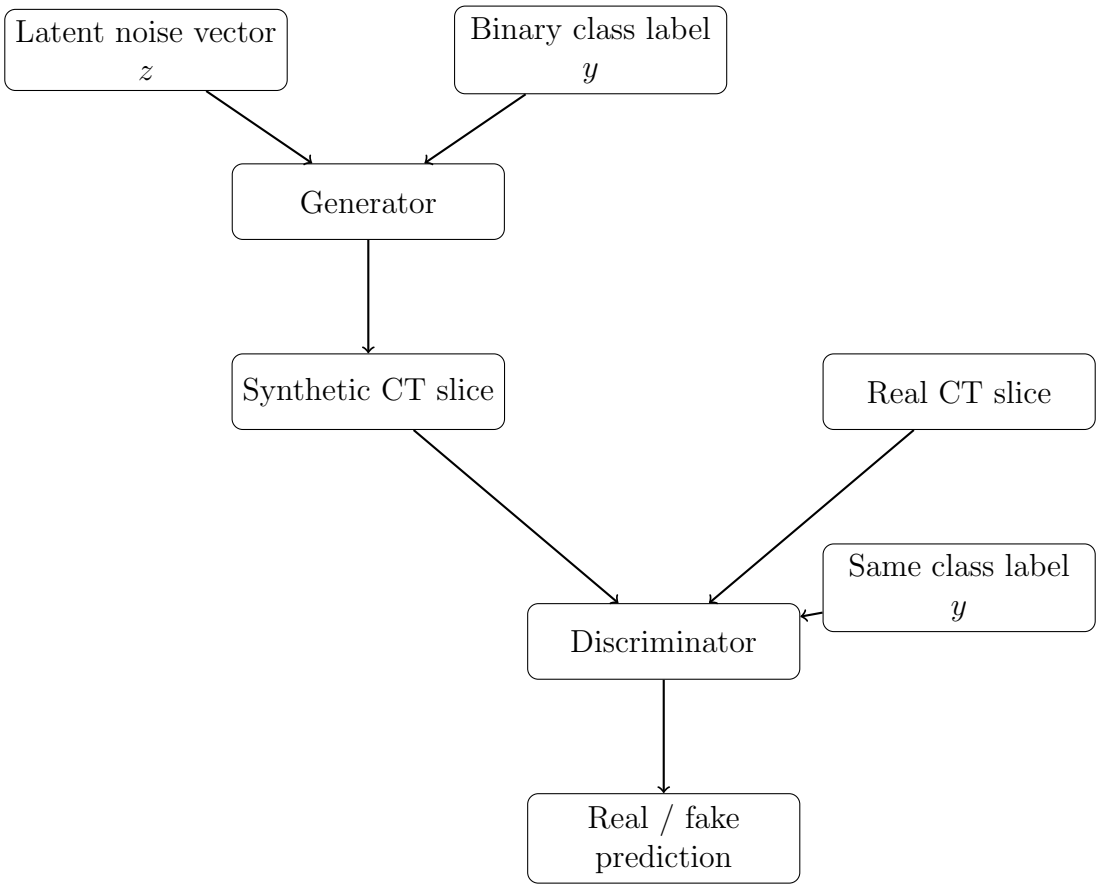
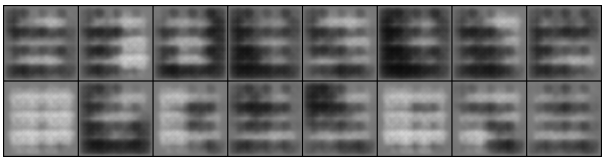
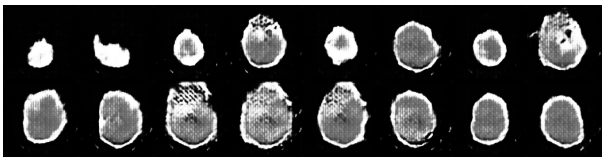


Figure 11: Schematic overview of the conditional GAN setup used in the exploratory study. The generator receives a latent noise vector and a binary class label and produces a synthetic CT slice. The discriminator receives either a real or synthetic CT slice together with the same class label and predicts whether the input is real or generated.

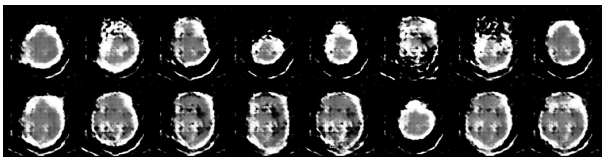
the observed instability could be reduced by changing image resolution, discriminator capacity, regularisation, and optimisation strategy.



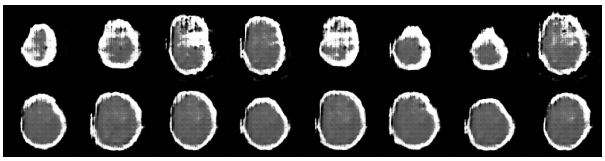
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 30

Figure 12: Qualitative progression of samples generated by the initial conditional GAN. Although the model gradually learnt coarse head-like structure, the outputs remained noisy, showed limited internal anatomical detail, and exhibited restricted diversity. In addition, the class-conditioning effect remained weak, which is consistent with the failure modes described in the text.

Figure 12 illustrates why the initial conditional setup was not sufficient for downstream synthetic-data experiments. Although the generated samples improved from the early epochs, they did not reach the level of realism, diversity, or class consistency required for use as synthetic training data.

6.6 Architectural and training modifications in the cGAN setting

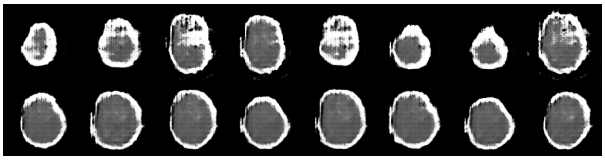
After the initial cGAN runs showed noisy outputs, limited diversity, and weak class conditioning, several architectural and training modifications were explored. The aim was to determine whether the limitations were mainly caused by unstable adversarial training, an overly strong discriminator, or the complexity of the image-generation task.

First, the image resolution was reduced from 128×128 to 64×64. This simplified the generative task, reduced computational cost, and allowed more frequent inspection of training behaviour. The reasoning was that the model should first be able to learn the coarse anatomical structure at a lower resolution before attempting higher-resolution synthesis.

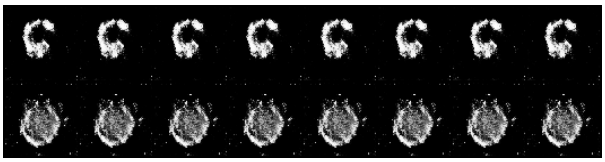
Second, the discriminator was simplified. Early runs suggested that the discriminator separated real images and generated images too easily, leaving the generator with limited useful feedback. To reduce this imbalance, the discriminator capacity was lowered. In addition, regularisation techniques were introduced, including spectral normalisation and dropout. Spectral normalisation constrains the spectral norm of the discriminator's weight matrices, which helps to control how strongly the discriminator can respond to small changes in its input. In practice, this can reduce overly sharp discriminator behaviour and make adversarial training more stable. Discriminator dropout was explored for a similar reason. Dropout is a regularisation technique in which part of the network is randomly deactivated during training. In the discriminator, this can reduce overconfidence and encourage a more stable learning signal for the generator.

Instance noise was added as another stabilisation technique. Instance noise means that a small amount of random noise is added to both real and generated images before they are passed to the discriminator. By slightly smoothing the distinction between the real and generated distributions, instance noise can make the adversarial game less abrupt during the early stages of training. These changes were intended to make the adversarial task less brittle and to reduce dominance of early discriminator.

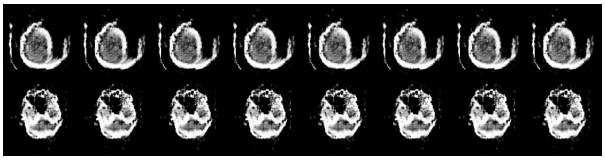
Finally, the optimisation procedure was adjusted. Separate learning rates were used for the generator and the discriminator, because the two networks did not appear to learn at the same pace. Fixed-noise monitoring was also introduced, meaning that the same latent vectors were used to generate sample grids across epochs. This made it easier to assess whether the model was genuinely improving, collapsing to similar outputs, or simply producing different noise patterns over time. The main differences between the early conditional GAN variants are summarised in Appendix B.1. The corresponding conditional generator and discriminator definitions are provided in Appendix B.2.



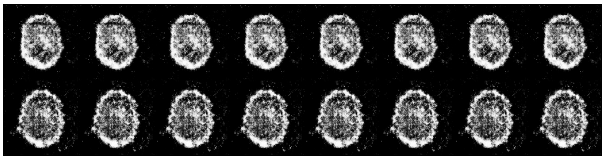
(a) Initial cGAN (Epoch 30)



(b) Reduced-resolution cGAN (Epoch 30)



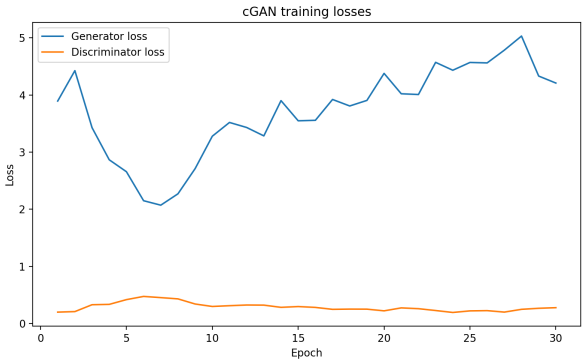
(c) Stabilised cGAN v2 (Epoch 30)



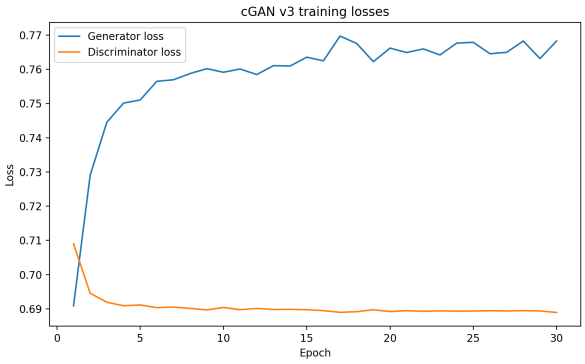
(d) Stabilised cGAN v3 (Epoch 30)

Figure 13: Representative outputs from successive cGAN variants explored during the architectural and training-modification phase. Reducing the resolution and introducing stabilisation techniques improved overall training behaviour and produced clearer coarse head structure, but the later variants still showed limited diversity and insufficient internal anatomical realism.

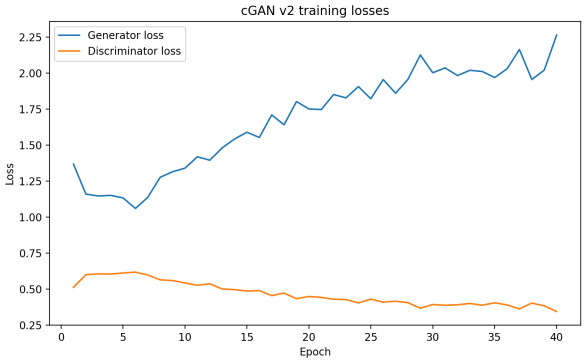
Figure 13 shows that the later cGAN variants produced more stable and structured outputs than the initial model. However, the improvement was limited to the coarse global structure. The generated slices still lacked convincing internal CT anatomy, and the class-conditioning effect remained too weak for reliable haemorrhage-positive and haemorrhage-negative generation.



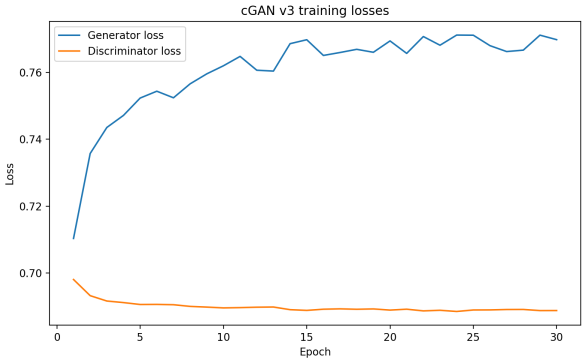
(a) Initial cGAN



(b) Reduced-resolution cGAN



(c) Stabilised cGAN v2



(d) Stabilised cGAN v3

Figure 14: Generator and discriminator loss curves for representative cGAN variants. The later variants show more stable training dynamics, but the loss curves alone do not establish anatomical realism or downstream usefulness.

The loss curves in Figure 14 indicate that the later variants changed the adversarial training dynamics and reduced the most severe instability. However, qualitative inspection remained essential because stable adversarial losses did not necessarily correspond to realistic CT synthesis.

Overall, the architectural and training modifications improved the stability of the cGAN experiments, but did not solve the main generative problem. The models learnt coarse head-like structure more reliably than before, but internal anatomical detail remained weak, sample diversity was limited, and class-conditional differences were not sufficiently clear. The cGAN branch therefore remained unsuitable for synthetic-only or mixed real-synthetic classifier training at this stage.

6.7 From conditional to single-class generation

Because class-conditioned generation remained difficult to stabilise, the next exploratory step was to simplify the generative task. Instead of generating both haemorrhage-positive and haemorrhage-negative slices, subsequent experiments focused only on haemorrhage-negative slices. The reasoning was that the model should first be able to learn plausible non-ICH head CT structure before attempting the more difficult task of class-conditioned pathology generation.

This led to the construction of a separate GAN-specific subset containing only haemorrhage-negative slices. Restricting the data to one class reduced label complexity and made it possible to isolate the problem of modelling normal head and brain structure without the additional difficulty of subtle pathological variation. The corresponding subset-construction and preprocessing steps are shown in Appendices B.3 and B.4.

The transition to single-class generation was accompanied by additional GAN-specific data curation. Exploratory inspection suggested that the GAN was sensitive not only to image content, but also to background variation, skull positioning, and slice location. Head-centred cropping was therefore introduced to remove irrelevant background and increase the relative prominence of the anatomical region. Central-slice filtering was then applied to retain slices with more consistent and informative head content, because highly peripheral slices were more variable and often contained less useful anatomical structure for generative modelling.

As an intermediate step, the existing conditional generator and discriminator components were trained on the cropped haemorrhage-negative subset using a fixed class label. Although the model retained its conditional architecture, every training sample used the same haemorrhage-negative condition, making the experiment effectively single-class. The corresponding configuration is shown in Appendix B.5.

The later experiments replaced this transitional setup with an explicitly unconditional generator and discriminator. In this configuration, neither model received a class label. The corresponding training procedure and model definitions are shown in Appendices B.6 and B.7.

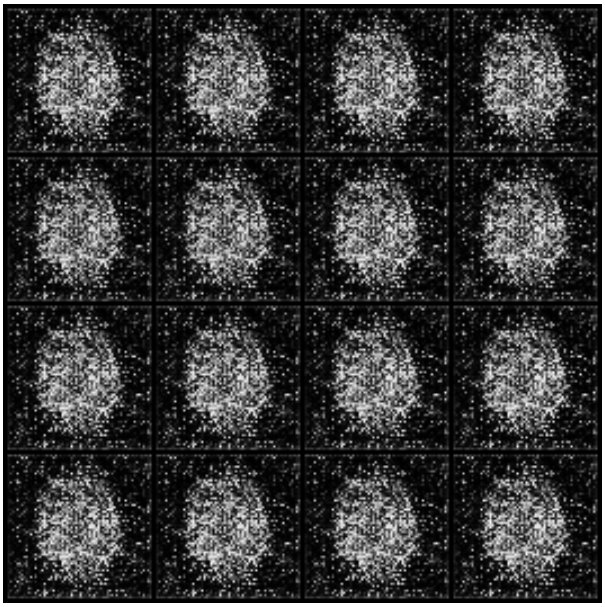
Figure 15 shows why this simplification was methodologically useful. Removing class-conditioning and reducing variation in the training data helped the model learn clearer global structure, but did not solve the underlying realism problem. The generated images still lacked sufficient internal anatomical detail, diversity, and visual plausibility for downstream classifier training.

6.8 Stabilisation in the single-class GAN setting

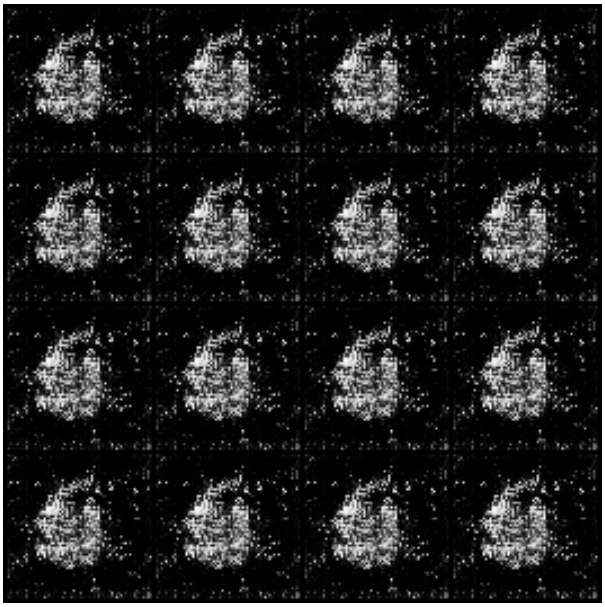
After moving to a haemorrhage-negative single-class setup, additional training modifications were explored to improve stability and sample quality. These included hinge-loss training, instance-noise scheduling, spectral normalisation in the discriminator, and feature matching in the generator objective.

The purpose of these changes was to reduce unstable adversarial behaviour and to encourage more globally consistent outputs. Hinge loss was used as an alternative adversarial objective, often used instead of the original binary cross-entropy GAN loss, because it can provide more stable gradients and improve training behaviour. Feature matching was added so that the generator was encouraged not only to fool the discriminator, but also to match intermediate discriminator feature statistics of real images. Instance-noise scheduling was used to smooth the real-versus-fake distinction during training, and spectral normalisation was retained to constrain discriminator behaviour. Real-versus-fake image grids were saved during training to monitor qualitative progression. The corresponding implementation of the stabilised local single-class GAN is shown in Appendices B.6 and B.7.

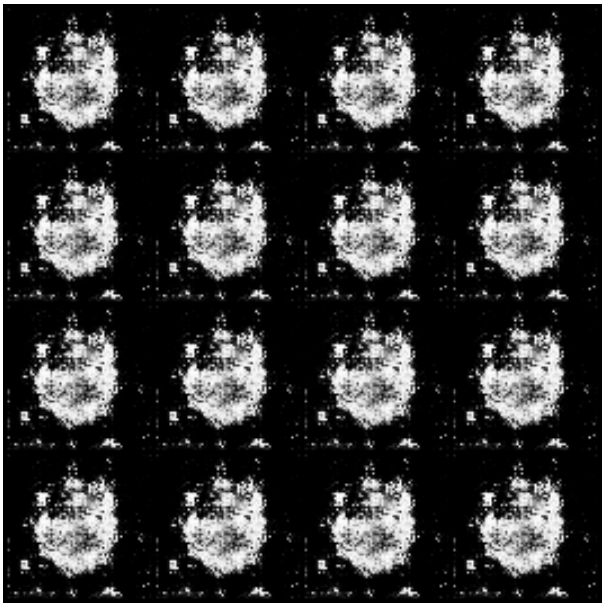
These modifications led to a partial qualitative improvement. Later single-class GAN variants produced more consistent coarse head-shaped outputs and, in some cases, a clearer skull outline. However, the generated images still contained visible artefacts, limited diversity, and weak internal anatomical structure. The single-class setup therefore improved the exploratory direction, but the



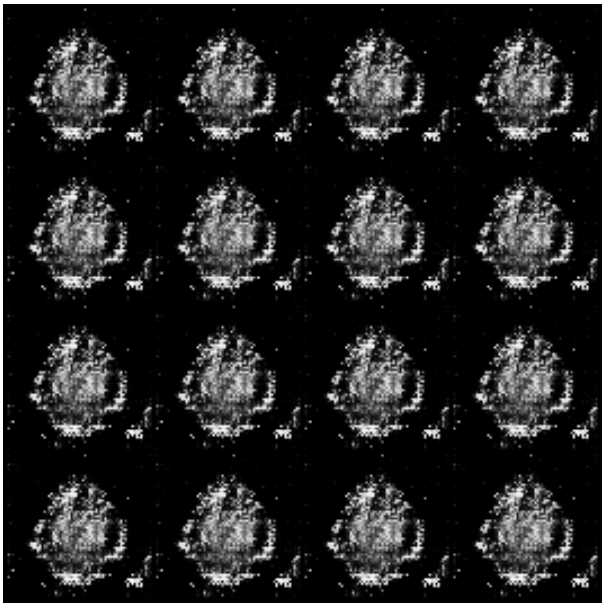
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 30

Figure 15: Qualitative progression of the first single-class GAN trained only on haemorrhage-negative CT slices. Restricting the task to one class simplified the generation problem and led to somewhat clearer global head structure, although substantial anatomical limitations remained.

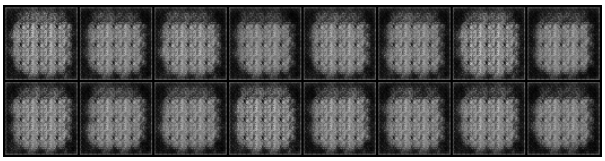
outputs remained unsuitable for synthetic-only or mixed real-synthetic classifier training.

6.9 CPU sanity check of the cropped single-class GAN

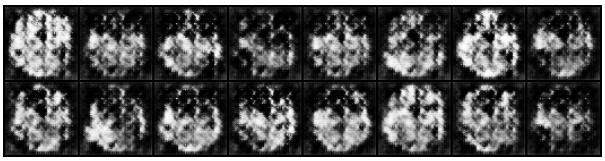
Before continuing with longer exploratory GAN runs on the Workstation, two short CPU sanity checks were performed. These runs were not intended to evaluate image realism, but to verify that the cropped single-class GAN pipeline remained operational after transferring the project to the university environment. In particular, they tested whether data loading, training, checkpointing, logging, and output saving worked correctly with the Leiden file paths and CPU execution.

Both sanity checks were based on the original `src/training/train_gan_singleclass.py` script, with a Leiden-specific base directory `/local/s3209199`, CPU execution, and a reduced batch size of 8. The runs used the cropped negative-only GAN dataset in `data/processed/rsna_gan_cropped`, with labels from `central_labels.csv`. The shared settings of the exploratory Leiden runs are summarised in Appendix C.4, Table 2.

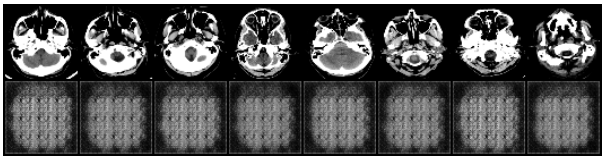
The initial 2-epoch run was completed successfully and produced the expected sample grids, real-versus-fake grids, checkpoints, configuration file, history file, and loss curve. A second 5-epoch run was then performed to check whether the same workflow remained stable over a slightly longer training duration. This run also completed successfully and saved outputs at the first and final epoch.



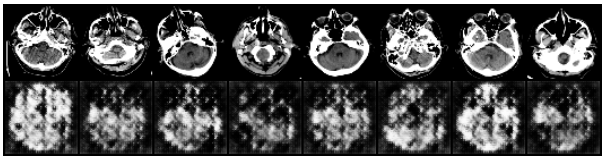
(a) Generated samples at epoch 1



(b) Generated samples at epoch 5



(c) Real-versus-fake comparison at epoch 1



(d) Real-versus-fake comparison at epoch 5

Figure 16: Outputs from the 5-epoch CPU exploratory run of the cropped single-class GAN on the Workstation.

Figure 16 shows the visual outputs from the 5-epoch sanity check. The samples are still too early in training to support conclusions about anatomical realism, but the run confirmed that the full experimental workflow was functioning correctly. The model could be trained on the Workstation, outputs could be saved, results could be transferred back to the MacBook, and figures could be integrated into the thesis. This provided the practical basis for the subsequent 20-epoch and 40-epoch exploratory runs.

6.10 Extended CPU training of the cropped single-class GAN

After two short sanity checks confirmed that the cropped single-class GAN pipeline functioned correctly on the Workstation, the same configuration was trained for 20 and 40 epochs. The purpose was to assess whether additional training would move the generator beyond the early artefact-dominated outputs and produce more plausible non-contrast head CT structure. Both experiments used the cropped haemorrhage-negative dataset and the shared settings summarised in Appendix C.4, Table 2. The corresponding configurations are provided in Appendices C.2 and C.3.

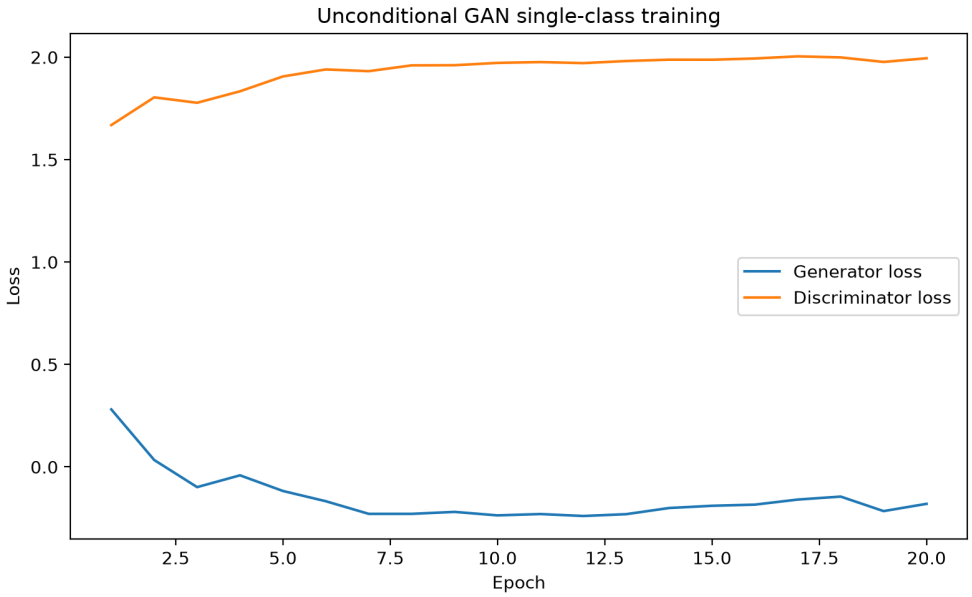
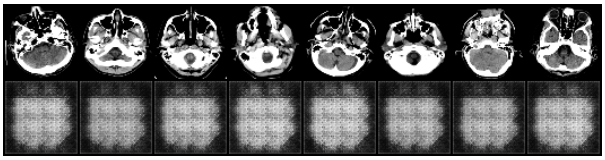


Figure 17: Generator and discriminator loss curves for the 20-epoch CPU run of the cropped single-class GAN on the Workstation. The losses indicate that adversarial training remained active throughout the run, but qualitative inspection is required to assess image realism.

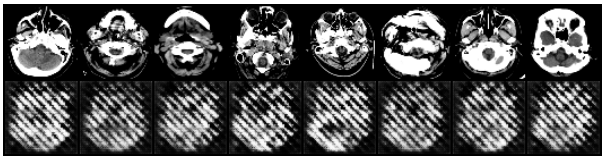
During the 20-epoch run, the generator loss decreased from 0.2808 in epoch 1 to -0.1802 in epoch 20, while the discriminator loss increased from 1.6687 to 1.9949, shown in Figure 17. This indicates that the adversarial training dynamics changed over the course of the run, with the generator becoming more successful according to its training objective. However, GAN loss values are difficult to interpret directly and do not necessarily correspond to visual realism. The generated images were therefore assessed primarily through qualitative inspection, as shown in Figure 18.

During the 20-epoch run, the generated images gradually changed from bright stripe- and grid-like artefacts to more consistent head-shaped structures. As shown in Figure 18, the first signs of coarse global head structure appeared around epoch 10 and became clearer by epoch 20. The changing generator and discriminator losses confirmed that adversarial training remained active, although these values were not treated as direct measures of image quality.

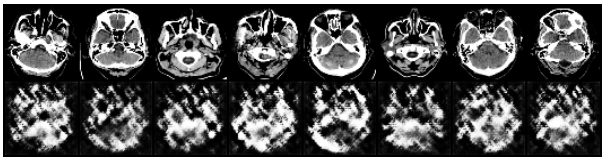
Despite this progression, the epoch-20 outputs still lacked convincing internal anatomy. The images remained noisy, visible artefacts persisted, and the contrast relationships between the skull, the brain tissue, and intracranial structures were not reproduced realistically. The model had therefore learnt part of the global shape of the cropped CT distribution, but not enough anatomical



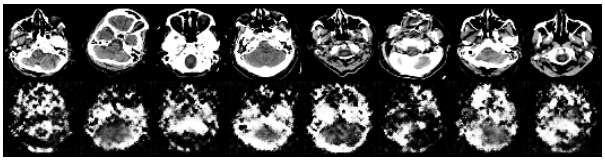
(a) Epoch 1



(b) Epoch 5



(c) Epoch 10



(d) Epoch 20

Figure 18: Real-versus-fake progression during the 20-epoch CPU run of the cropped single-class GAN on the Workstation. The generated outputs gradually evolve from highly artefactual images toward coarse head-like structure, but remain substantially different from the real CT slices.

detail for downstream classifier training.

The same configuration was subsequently trained for 40 epochs to assess whether the qualitative improvement observed in the 20-epoch run would continue. Figure 19 shows the corresponding loss curve. As in the 20-epoch run, the losses suggest that adversarial training remained active, but they do not by themselves demonstrate that the generated images became anatomically realistic. The generated outputs were therefore assessed primarily through qualitative inspection.

Figure 20 shows that most qualitative development again occurred during the first half of training. The generated images became more consistent in global shape between approximately epochs 5 and 20. However, the later epochs did not produce a clear improvement in anatomical realism. Strong artefacts, noisy internal structure, limited diversity, and unrealistic contrast relationships remained visible, indicating that the model had reached a limited qualitative plateau within this configuration.

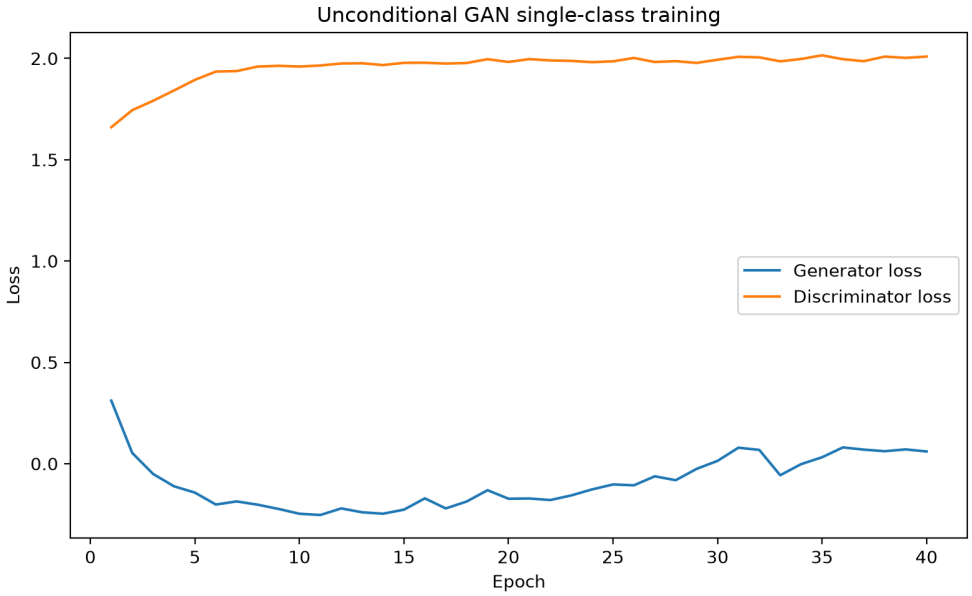
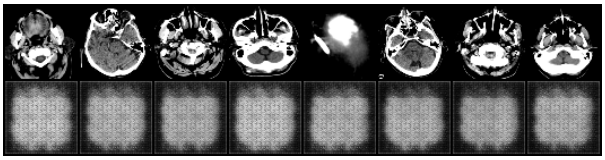
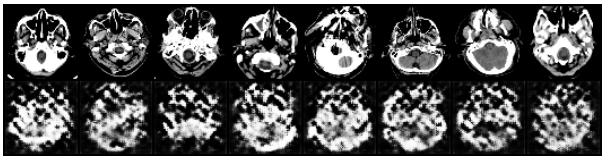


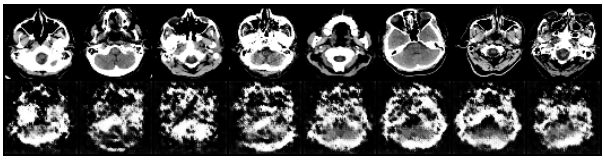
Figure 19: Generator and discriminator loss curves for the 40-epoch CPU run of the cropped single-class GAN on the Workstation. The losses suggest that training remained active, but the later epochs do not correspond to a clear qualitative improvement in the generated images.



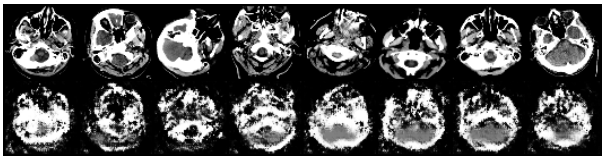
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 40

Figure 20: Real-versus-fake progression during the 40-epoch CPU run of the cropped single-class GAN on the Workstation. The model learns a coarse head-like structure during training, but the later epochs do not show a clear qualitative breakthrough beyond the level already reached around epoch 20.

At epoch 40, the generated outputs still contained strong artefacts, noisy internal structure, limited diversity, and unrealistic contrast patterns. In particular, the model did not reproduce the anatomical relationships between the skull, brain tissue, and intracranial structures convincingly. Extending the training duration from 20 to 40 epochs therefore did not solve the main realism

problem.

These experiments indicate that the cropped single-class GAN reached a limited plateau within the tested configuration. Longer training improved coarse global structure, but did not produce anatomically plausible CT slices. Further progress therefore required a methodological change rather than another extension of the same CPU-based run, motivating the subsequent GPU and generator-architecture experiments.

6.11 GPU replication of the cropped single-class GAN

The cropped single-class GAN was subsequently replicated for 40 epochs using GPU acceleration on the Workstation. The dataset, architecture, training objective, number of epochs, and core optimisation settings were kept the same as in the preceding 40-epoch CPU experiment.

GPU acceleration was not expected to improve image quality by itself. Its main advantage was the shorter training time, which made longer runs and subsequent architectural experiments computationally feasible. The purpose of this experiment was therefore to verify that the cropped single-class GAN pipeline could be reproduced under CUDA-based GPU acceleration and to establish a more efficient computational setup for the subsequent experiments.

Although no systematic quality improvement was expected from changing the execution device alone, the generated outputs did not have to be identical because GAN training is stochastic and CPU- and GPU-based computations may introduce small numerical differences.

This experiment used the same cropped negative-only dataset and core model settings as the previous 40-epoch CPU run, but replaced CPU execution with CUDA-based GPU acceleration. A separate script copy, shown in Appendix D.1, was used to keep the outputs of the GPU run separate from the earlier CPU experiments. The main settings remained an image size of 64×64, a latent dimension of $Z = 64$, a batch size of 8, a generator learning rate of 2×10^{-4}, a discriminator learning rate of 5×10^{-5}, feature matching with $\lambda = 2.0$, and an initial instance-noise standard deviation of 0.02. The most relevant settings are summarised in Table 3.

The generator loss changed from 0.2411 in epoch 1 to -0.1014 in epoch 40, while the discriminator loss increased from 1.6761 to 2.0068. Figure 21 shows that adversarial training remained active throughout the run. As in the CPU experiments, the loss values were not interpreted as direct measures of anatomical realism.

The qualitative progression in Figure 22 was similar to that observed during CPU training. The earliest outputs were dominated by regular grid-like artefacts. By epochs 10 and 20, the generator had learnt a more consistent coarse head shape. Further training up to epoch 40 did not produce convincing internal brain anatomy or realistic contrast relationships.

The GPU run led to the same overall qualitative conclusion as the preceding CPU experiment. The generator learnt coarse global head structure, but did not produce anatomically realistic CT slices. This outcome was consistent with the expectation that changing the execution device alone would mainly affect runtime rather than generation quality.

The main contribution of the GPU run was therefore computational. It reduced the practical training time and made the subsequent generator-architecture and class-conditional experiments more feasible. The remaining qualitative limitations motivated changes to the modelling setup, including the introduction of an upsample-based generator architecture.

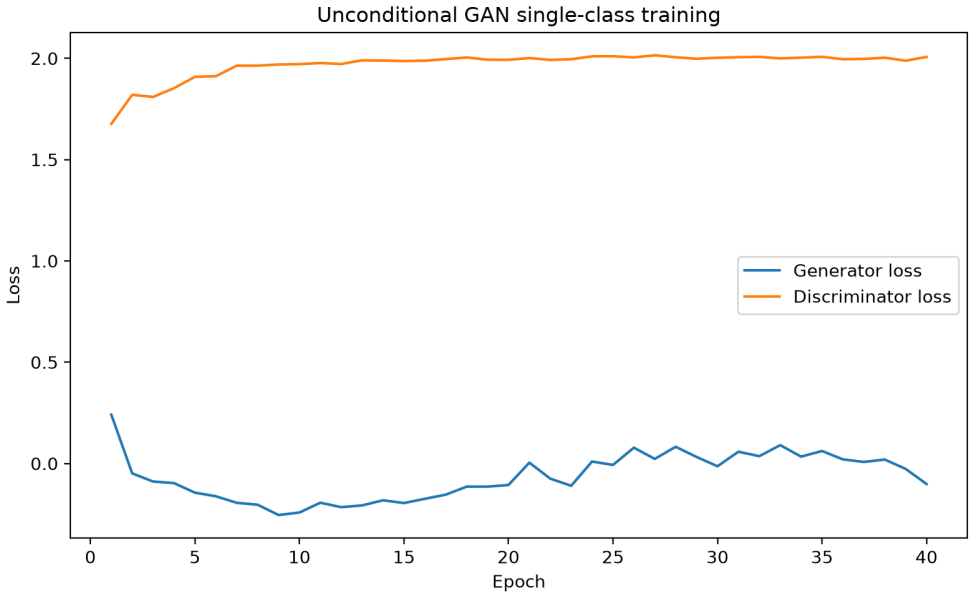
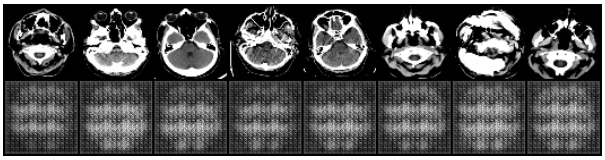
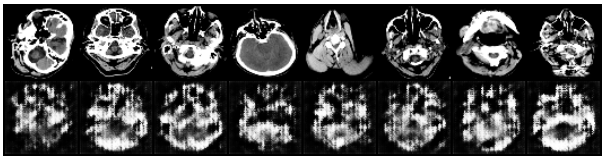


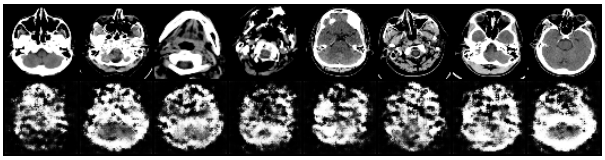
Figure 21: Generator and discriminator loss curves for the 40-epoch GPU run of the cropped single-class GAN on the Workstation. The losses indicate the numerical training behaviour of the model, but do not provide a direct measure of anatomical realism.



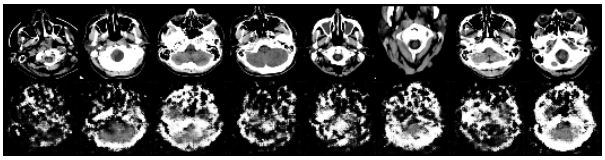
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 40

Figure 22: Real-versus-fake progression during the 40-epoch GPU run of the cropped single-class GAN on the Workstation. The generator learnt coarse global head structure, but the outputs remained affected by strong artefacts, noisy internal structure, limited diversity, and unrealistic contrast relationships.

6.12 Upsample-based generator experiment

To investigate whether the checkerboard-like artefacts observed in the earlier single-class GAN runs were related to transposed convolutions, a second 40-epoch GPU experiment was performed using an upsample-based generator. In this architecture, each transposed-convolutional block was replaced by explicit upsampling followed by a standard convolution. The aim was to determine whether this modification would produce smoother outputs and improve anatomical plausibility.

The dataset, discriminator, training duration, and main optimisation settings were kept the same as in the previous GPU experiment. Compared with the previous run, the main methodological change was the generator architecture: transposed convolutions were replaced by upsampling followed by standard convolutions. All other key settings were kept the same. The full configuration is provided in Appendix D.2 and the most relevant settings are summarised in Table 3.

The generator loss changed from 0.0636 in epoch 1 to 0.0971 in epoch 40, while the discriminator loss changed from 1.8317 to 1.9398. Figure 23 shows that adversarial training again remained active, although the loss values did not indicate a clear improvement in image quality.

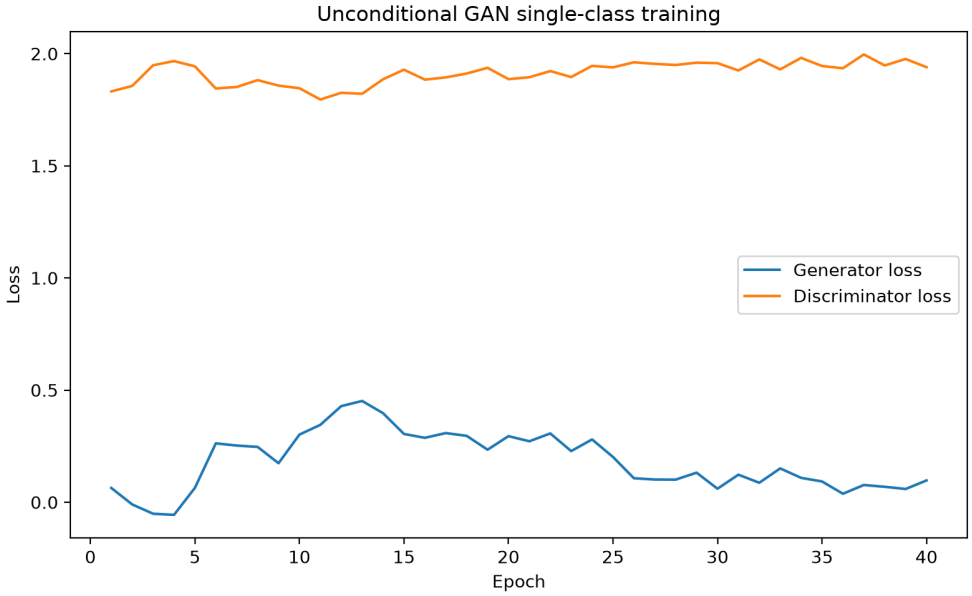
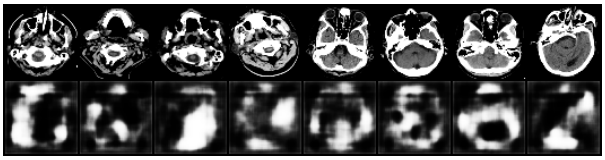


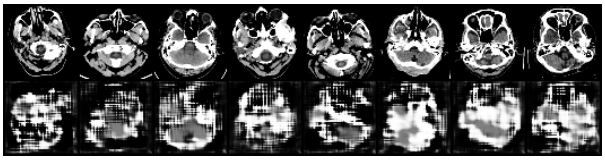
Figure 23: Generator and discriminator loss curves for the 40-epoch GPU run with an upsample-based generator. The losses remained relatively stable, but this did not correspond to a clear improvement in anatomical realism.

The outputs in Figure 24 show that the architectural change affected the appearance of the generated images. The earliest outputs were less uniformly checkerboard-like than those of the transposed-convolutional generator, indicating that the upsampling strategy influenced the artefact pattern. During training, the model again developed coarse head-shaped structure. However, the later outputs still contained grid- and block-like artefacts, limited variation, and unrealistic internal anatomy.

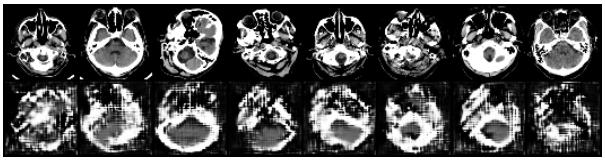
Replacing transposed convolutions with upsampling and standard convolutions therefore changed the visual behaviour of the generator, but did not resolve the main realism problem. The model



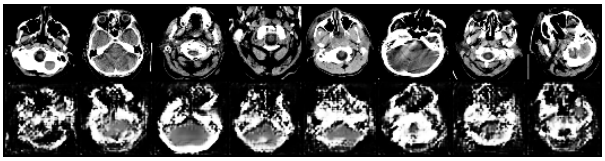
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 40

Figure 24: Real-versus-fake progression during the 40-epoch GPU run with an upsample-based generator. The architectural change altered the artefact pattern, but the generated images continued to lack realistic internal anatomy, contrast structure, and sample diversity.

continued to learn global head shape more successfully than meaningful internal anatomical structure. This result indicates that the limitations of the single-class GAN were not attributable solely to the use of transposed convolutions and motivated the subsequent return to class-conditional generation with a modified cGAN setup.

6.13 Two-class cGAN experiments on the full RSNA training split

After the single-class experiments failed to produce convincing internal CT anatomy, two additional experiments returned to class-conditioned generation using a cGAN. Both models were trained for 40 epochs on the complete patient-level RSNA training split of 2,793 images, consisting of 1,405 haemorrhage-negative slices and 1,388 haemorrhage-positive slices. The purpose was to determine whether explicit conditioning on the ICH status could produce anatomically plausible and visually distinct examples for the two classes.

6.13.1 Version 1

The first two-class cGAN experiment differed from the previous single-class runs mainly in its use of class conditioning. Instead of training an unconditional or effectively single-class GAN, this model was trained as a conditional GAN with two classes: haemorrhage-positive and haemorrhage-negative. Both the generator and the discriminator were conditioned on the binary haemorrhage label. All other key settings were kept broadly comparable to the previous runs, unless stated otherwise. The full configuration is provided in Appendix E.1, and the most relevant settings are summarised in Table 4.

The losses in Figure 25 remained within a relatively narrow range during training, with generator loss changing from 0.7055 to 0.7734 and discriminator loss from 0.7039 to 0.6889. This indicated numerically stable training behaviour, but did not establish that the generator had learnt the real CT distribution. Qualitative inspection remained necessary to assess anatomical realism, diversity, and class consistency.

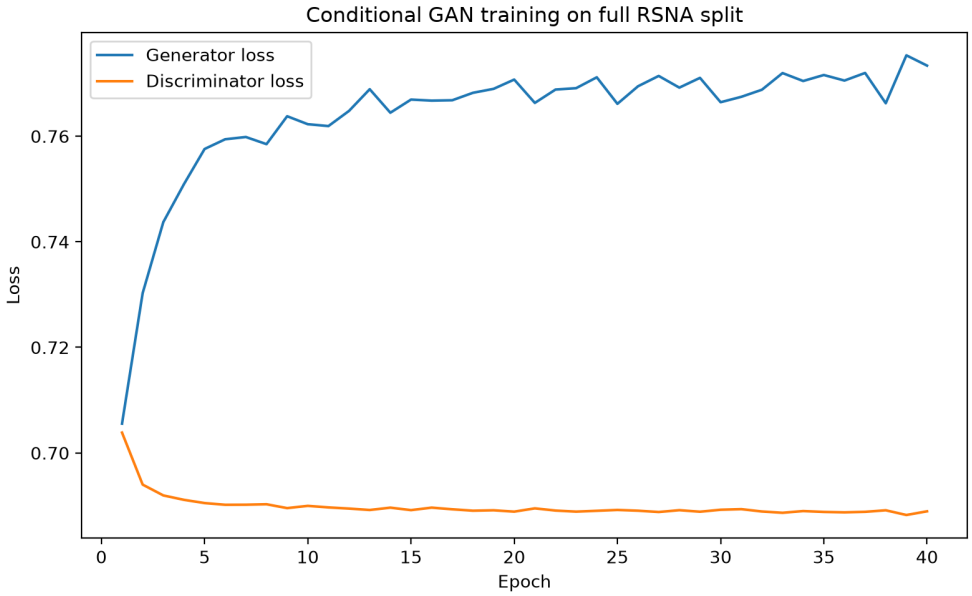
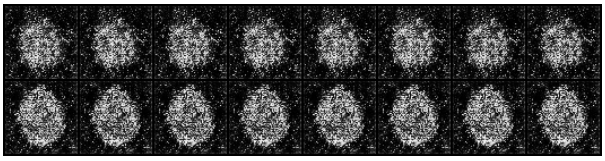
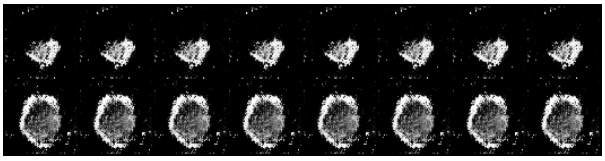


Figure 25: Loss curve of the 40-epoch GPU-based two-class cGAN run on the full RSNA training split. The generator loss increased gradually while the discriminator loss decreased slightly and remained close to a stable plateau, indicating numerically stable adversarial training.

To illustrate the evolution of the generated outputs more directly, Figure 26 shows sample grids from an early and a late stage of training. At epoch 1, the generated images mainly consisted of noisy, highly simplified head-like blobs. By epoch 40, the samples showed somewhat more consistent global structure and slightly clearer head contours. However, they still lacked realistic internal anatomy and remained visually far from true non-contrast brain CT slices.



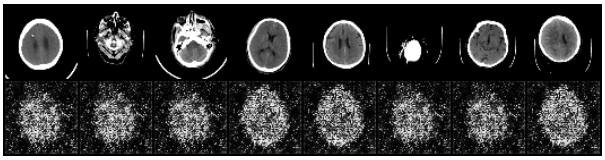
(a) Epoch 1



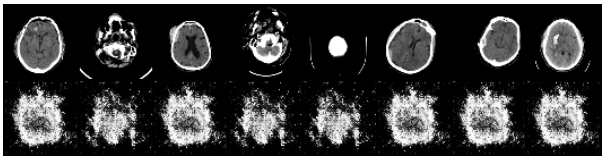
(b) Epoch 40

Figure 26: Generated sample grids from the 40-epoch GPU-based two-class cGAN run on the full RSNA training split. From epoch 1 to epoch 40, the model developed somewhat more consistent coarse head-like structure, but the generated images remained noisy and anatomically unrealistic.

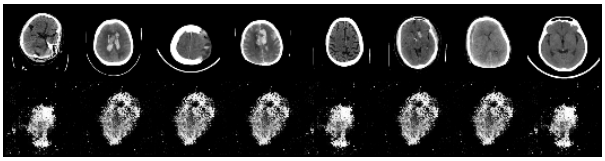
Figure 27 shows that the generated images developed more consistent global head-like structure over time. Compared with the earlier single-class experiments, the outputs were less dominated by strong checkerboard artefacts. However, they remained highly artificial, with noisy intensity patterns, limited internal detail, and weak resemblance to real non-contrast brain CT slices. No clearly interpretable visual distinction emerged between the generated haemorrhage-positive and haemorrhage-negative examples.



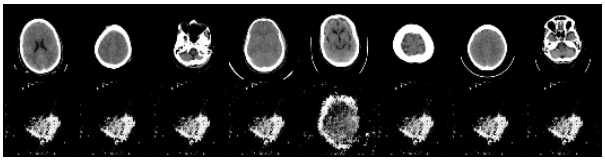
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 40

Figure 27: Real-versus-fake progression during the first 40-epoch two-class cGAN experiment. The model learnt coarse global head structure, but the generated images remained noisy and anatomically unrealistic, without clear visual separation between the two requested classes.

6.13.2 Version 2

A second two-class cGAN experiment tested whether a less constrained configuration would provide a more useful learning signal. The aim of this v2 experiment was therefore not merely to repeat the earlier run, but to test whether a slightly simpler and less tightly regulated conditional setup would lead to more realistic and class-consistent non-contrast brain CT slices. Relative to the first run, the latent dimension was reduced from $Z = 128$ to $Z = 64$, real-label smoothing was removed, the initial instance-noise standard deviation was reduced from 0.03 to 0.02, the number of generator updates per batch was reduced from two to one, and discriminator dropout was increased from 0.30 to 0.35. The remaining main settings and training data were unchanged and visible in Table 4. The complete configuration is provided in Appendix E.2.

The generator loss changed from 0.6307 in epoch 1 to 0.6879 in epoch 40, while the discriminator loss changed from 0.7022 to 0.6933. As shown in Figure 28, both losses quickly settled into a narrow range. This indicates stable but relatively flat numerical training behaviour; it does not establish that the generator learnt anatomically realistic or class-consistent outputs.

Figure 29 shows that the generated outputs became more structured during training. The diffuse, noisy shapes visible at epoch 1 developed into more consistent head-like forms by epoch 40. However, the samples remained repetitive and lacked realistic internal brain anatomy.

The comparison with real CT slices in Figure 30 confirms that the improvement was limited to coarse global structure. Even at epoch 40, the generated images showed weak internal detail, unrealistic intensity patterns, and little resemblance to the real slices. The modified conditional setup also failed to produce a clearly interpretable visual distinction between haemorrhage-positive and haemorrhage-negative outputs.

Compared with the first two-class cGAN experiment, the modified configuration did not improve anatomical realism, diversity, or class consistency. Reducing the latent dimension and relaxing several stabilisation settings therefore did not resolve the main limitations of the conditional model.

Neither two-class configuration produced images of sufficient quality for downstream use. Numerically stable training and the emergence of coarse global structure were not accompanied

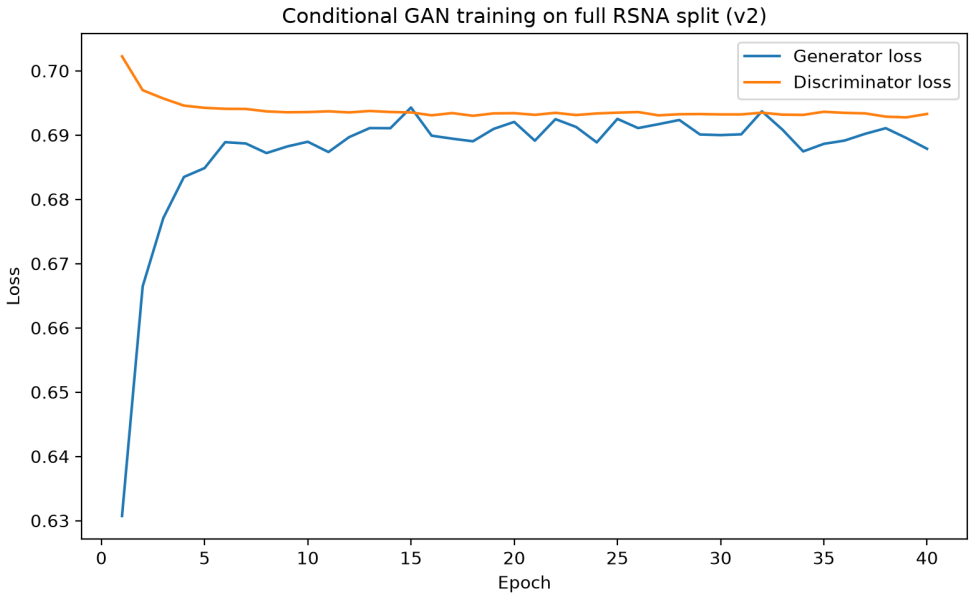
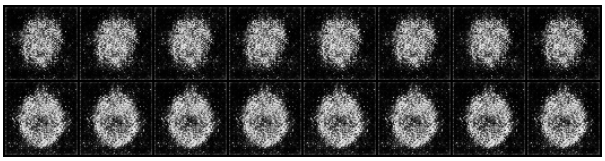
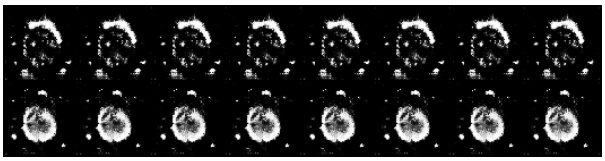


Figure 28: Generator and discriminator loss curves for the second 40-epoch two-class cGAN experiment. Both losses quickly settled into a narrow range, indicating stable but relatively flat numerical training behaviour.

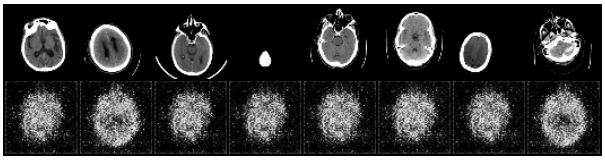


(a) Epoch 1

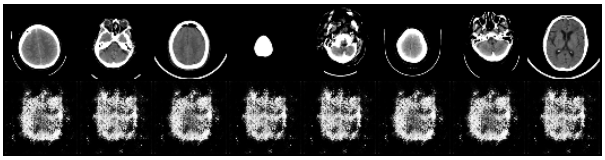


(b) Epoch 40

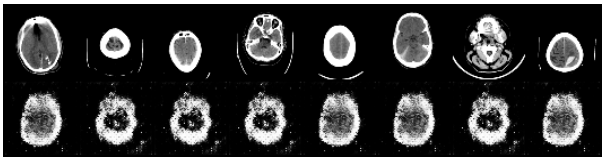
Figure 29: Generated sample grids from the second 40-epoch two-class cGAN experiment. The outputs developed more consistent global structure but remained repetitive and anatomically unrealistic.



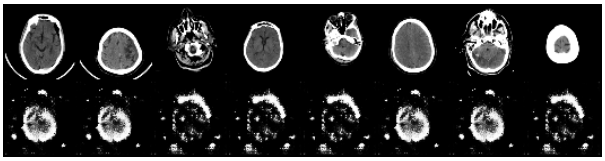
(a) Epoch 1



(b) Epoch 10



(c) Epoch 20



(d) Epoch 40

Figure 30: Real-versus-fake progression during the second 40-epoch two-class cGAN experiment. The generated images developed coarse head-like structure but remained clearly different from the real CT slices and showed no reliable class-specific features.

by realistic internal anatomy or reliable label conditioning. Within the tested 64×64 cGAN architecture and training strategy, the generated images were therefore unsuitable for synthetic-only or mixed real-synthetic classifier training.

6.14 Lessons from the exploratory study

The exploratory study yielded several findings that shaped the final scope of the thesis. First, the real-data classification pipeline reached a stable and reproducible setup, whereas the generative pipeline required repeated changes to preprocessing, architecture, and training strategy. Second, reducing variation in background, skull positioning, and slice location created a more consistent training distribution for the GAN experiments. Third, the generative models learnt coarse global head structure more readily than realistic internal brain anatomy. Fourth, simplifying the task from conditional to single-class generation improved global structure, but did not produce synthetic CT slices with sufficient realism or diversity.

The generative experiments also showed that longer training, a modified upsampling strategy, and adjusted class-conditioning settings did not resolve the central realism problem. GPU acceleration improved the computational feasibility of the experiments but did not change the overall qualitative conclusion.

Based on these findings, the structured experimental phase focuses on the completed real-data classification settings: the baseline classifier, conservative data augmentation, and threshold tuning. Synthetic-only and mixed real-synthetic classifier training were not performed because the generated images did not meet the predefined requirements for anatomical realism, diversity, and class consistency. The generative component is therefore reported as a qualitative feasibility study and provides the methodological basis for this final experimental scope.

7 Structured Experiments

7.1 Experimental design

The structured experimental part of the thesis uses a fixed patient-level split, a fixed preprocessing pipeline, and a fixed classifier architecture in order to make performance comparisons meaningful. At this stage, the completed experiments consist of a real-data baseline classifier, an augmented classifier, and a threshold tuning analysis. Threshold tuning is the process of selecting a decision threshold other than the default value of 0.5 in order to obtain a more suitable balance between sensitivity and specificity. The original intention was also to perform synthetic-only and mixed real-synthetic experiments. However, because the available synthetic images did not yet reach sufficient realism and diversity, these experiments were not performed in the current study.

Each reported classifier configuration represents a single training run. The experiments were not repeated across multiple random seeds, and no confidence intervals or statistical significance tests were calculated. The reported differences should therefore be interpreted as observed differences within the current dataset split rather than as estimates of variability across repeated experiments.

7.2 Baseline classifier

As an initial reference experiment, a ResNet-18 convolutional neural network was used as the baseline classifier. ResNet-18 was selected because it is a standard and well-established image classification architecture that is sufficiently expressive for the current slice-based problem while remaining computationally manageable.

The network was adapted for binary classification by replacing the final fully connected layer with a single output neuron. Since preprocessed CT slices are stored as single-channel images, each slice was repeated across three channels before being passed to the network, allowing the standard ResNet architecture to be used without structural modification. The model outputs a single logit, which is converted into a predicted probability of haemorrhage using the sigmoid function. The architecture itself was introduced earlier in Section 2.4 and Figure 2; in the present section, the focus is on how that classifier was instantiated in the structured experiments.

Binary cross-entropy with logits was used as the loss function. The model parameters were optimised with the Adam optimiser using a learning rate of 10^{-4}. The baseline model was trained for a fixed number of epochs, and the best checkpoint was selected using validation AUROC. The area under the receiver operating characteristic curve (AUROC) is a threshold-independent metric that measures how well a model separates positive and negative cases across all possible decision thresholds. This baseline serves as the primary real-data reference for all further comparisons.

7.3 Augmented classifier

The baseline classifier provided a useful reference point, but also showed signs of overfitting. Although training performance improved strongly over time, validation performance improved less consistently and fluctuated between epochs. This suggested that the model was learning patterns that were specific to the training set rather than features that generalised well to unseen data. To address this problem, a second classifier was trained using conservative training-time data augmentation.

Data augmentation is a regularisation strategy in which modified versions of the original training images are generated on the fly during training. The purpose is to increase the effective variation in the training data without changing the underlying class label. In this study, augmentation was introduced as a natural next step after the baseline experiment, because it allows the same classifier architecture to be trained on a more diverse set of inputs while keeping the train-validation-test split unchanged. This makes it possible to test whether better generalisation can be achieved without changing the core model.

The same ResNet-18 architecture, train-validation-test split, loss function, Adam optimiser, and learning rate were used as in the baseline experiment. However, the augmented configuration additionally used weight decay of 10^{-4}. Weight decay was included as an additional regularisation measure in response to the signs of overfitting observed in the baseline experiment. It discourages excessively large parameter values during optimisation, which can help reduce overfitting and improve generalisation to unseen data. Random transformations were applied only to the training images, while the validation and test images were left unchanged. The augmented experiment also used early stopping based on validation AUROC, whereas the baseline followed its original fixed-epoch training procedure.

The augmentation strategy was intentionally mild in order to preserve medical plausibility. The applied transformations included small affine transformations, limited brightness and contrast perturbations, and low-amplitude Gaussian noise. Small affine transformations were used to simulate minor differences in positioning and acquisition, while brightness and contrast perturbations were used to introduce slight intensity variation. Gaussian noise was added to make the model more robust to small fluctuations in image appearance. Strong geometric transformations, such as large rotations or flips, were avoided because they could disrupt anatomical consistency or produce unrealistic brain CT images.

In this setting, early stopping based on validation AUROC was used to reduce unnecessary overtraining. Early stopping is a regularisation technique in which training is stopped once performance on the validation set no longer improves. This helps prevent the model from continuing to optimise the training data at the expense of generalisation. The augmented classifier therefore represents a more regularised real-data training configuration than the baseline model. The experiment retains the same classifier architecture while combining medically plausible training-time augmentation, validation-based early stopping, and weight decay.

7.4 Threshold tuning

The classifier does not directly output a final class label, but a probability-like score that reflects how likely a slice is to contain haemorrhage. To convert this score into a binary prediction, a decision threshold must be chosen. A commonly used default is 0.5, meaning that predictions above 0.5 are classified as positive and predictions below 0.5 as negative. However, this default is not necessarily optimal, especially in medical classification tasks where the balance between sensitivity and specificity is clinically important. Sensitivity is the proportion of true positive cases that are correctly identified by the model. Specificity is the proportion of true negative cases that are correctly identified by the model.

For this reason, an additional threshold-tuning analysis was carried out for both classifiers. The purpose of this step was to determine whether the observed sensitivity-specificity trade-off was caused by the model's actual class-separation ability or simply by the choice of decision threshold.

In other words, threshold tuning makes it possible to separate the quality of the learnt ranking from the operating point at which the classifier is used.

For each trained model, prediction probabilities were first computed on the validation set. A threshold sweep was then performed over the interval $[0, 1]$, and the threshold maximising Youden's statistic J was selected:

$$J = \text{sensitivity} + \text{specificity} - 1.$$

Youden's J statistic is defined as sensitivity plus specificity minus one. It is commonly used to select a threshold that balances the correct identification of positive and negative cases. It was chosen because it gives equal importance to sensitivity and specificity and therefore provides a simple way to identify a more balanced operating point.

The selected threshold was then applied to the held-out test set. Threshold selection was based exclusively on validation-set predictions, so no test-set labels or test-derived threshold criterion were used to optimise the final decision rule. This prevents direct tuning of the decision threshold on the test set. However, because test-set ROC curves and AUROC values had already been inspected during the exploratory phase, the test set was not fully untouched throughout the complete development process. This limitation is discussed in Section 9.4.

This threshold-tuning analysis therefore complements the AUROC-based evaluation. AUROC measures threshold-independent ranking performance, while sensitivity, specificity, and accuracy depend on the chosen threshold. Considering both types of evaluation makes it possible to distinguish between improved class separation and a simple shift in the decision boundary.

7.5 Evaluation metrics

The primary evaluation metric is AUROC, because it measures class separation independently of a specific decision threshold. In addition, accuracy, sensitivity, and specificity are reported to describe threshold-dependent behaviour. Sensitivity is especially important in the context of intracranial haemorrhage detection, because missing a true haemorrhage may have serious clinical consequences. Specificity is also relevant, because too many false alarms reduce practical usefulness in a clinical workflow. Together, these metrics provide a balanced description of classifier behaviour.

7.6 Scope of the structured experiments

The structured experiments include only the completed real-data baseline and augmented classifiers. The synthetic-only and mixed real-synthetic settings remain methodological goals rather than completed experiments. Their absence is not simply a missing step, but follows directly from the findings of the exploratory study: under the current conditions, the generative models did not yet produce synthetic CT slices of sufficient quality for a fair downstream comparison.

8 Experiments and Results

8.1 Real-data baseline

The first structured classification experiment evaluated the baseline ResNet-18 classifier established during the exploratory phase in Subsection 6.3. In the present chapter, the focus is on its final

performance under the fixed structured evaluation setup. The classifier was trained on the fixed patient-level training split, and the best checkpoint was selected based on validation AUROC. Final performance for the structured comparison was reported on the held-out test set.

At the default decision threshold of 0.5, the baseline model achieved a test AUROC of 0.811, an accuracy of 0.731, a sensitivity of 0.692, and a specificity of 0.770. These results show that the baseline classifier achieved meaningful discrimination between haemorrhage-positive and haemorrhage-negative slices.

The corresponding ROC curve and training behaviour were presented earlier in the exploratory study (Figures 7 and 8). Those figures showed that the model learnt a meaningful decision boundary, but also exhibited signs of overfitting. This motivated the evaluation of conservative data augmentation in the next experiment.

8.2 Real-data baseline with augmentation

The second structured experiment evaluated the augmented ResNet-18 classifier that was introduced after overfitting had been observed during the exploratory phase, as described in Subsection 6.3. The classifier architecture, preprocessing pipeline, patient-level split, loss function, optimiser type, and learning rate were kept unchanged. The augmented configuration differed from the baseline through the use of conservative training-time data augmentation, validation-based early stopping, and weight decay of 10^{-4}.

At the default threshold of 0.5, the augmented classifier achieved a test AUROC of 0.881, an accuracy of 0.717, a sensitivity of 0.950, and a specificity of 0.480. The augmented training configuration achieved a higher AUROC than the baseline, increasing from 0.811 to 0.881 and indicating stronger threshold-independent separation between haemorrhage-positive and haemorrhage-negative slices.

The corresponding test-set ROC curve is shown in Figure 9. The training and validation curves in Figure 10 show the training behaviour of the augmented model. Although the augmented classifier achieved a higher AUROC, the default threshold produced a strongly unbalanced operating point. Sensitivity increased to 0.950, but specificity decreased to 0.480. This suggests that the augmented model assigned relatively high haemorrhage scores to many slices, including a substantial number of haemorrhage-negative cases, so the default threshold of 0.5 was too low for this model's output-score distribution. In other words, the model improved its overall ranking ability, but the default decision threshold no longer produced a balanced sensitivity-specificity trade-off. This motivated the threshold-tuning analysis.

8.3 Threshold tuning

Because the default decision threshold of 0.5 produced very different sensitivity-specificity trade-offs for the baseline and augmented models, an additional threshold-tuning analysis was performed. For both models, the classification threshold was selected on the validation set by maximising Youden's statistic J, defined as sensitivity + specificity -1. The selected threshold was then applied to the held-out test set.

Youden's J statistic is a simple threshold-selection criterion that combines sensitivity and specificity into a single quantity. A larger value of J indicates a more balanced operating point, which makes it useful when both false negatives and false positives are relevant.

For the baseline model, threshold tuning had only a limited effect on overall performance. The selected threshold was 0.17, yielding a test accuracy of 0.731, a sensitivity of 0.775, and a specificity of 0.686. Compared with the default threshold, this mainly shifted the operating point toward higher sensitivity at the cost of lower specificity, while leaving the overall balance nearly unchanged.

For the augmented model, threshold tuning had a much larger impact. The selected threshold was 0.84, which gave a test accuracy of 0.796, a sensitivity of 0.791, and a specificity of 0.801. This shows that the poor specificity observed at threshold 0.5 was largely caused by an unfavourable operating point rather than by poor class separation. In other words, the augmented training configuration produced stronger ranking performance, but the default threshold was not appropriate for its output-score distribution.

Model	Threshold	AUROC	Accuracy	Sensitivity	Specificity
Baseline	0.50	0.811	0.731	0.692	0.770
Baseline	0.17	0.811	0.731	0.775	0.686
Augmentation	0.50	0.881	0.717	0.950	0.480
Augmentation	0.84	0.881	0.796	0.791	0.801

Table 1: Comparison of baseline and augmentation experiments on the held-out patient-level test set, before and after threshold tuning. AUROC is threshold-independent and therefore remains unchanged for a given model.

Table 1 shows that the augmented training configuration achieved stronger threshold-independent discrimination than the baseline. After validation-based threshold tuning, the augmented classifier also achieved a better test-set balance between sensitivity and specificity than the baseline classifier. Among the evaluated settings, the augmented model at threshold 0.84 achieved the highest accuracy and the most balanced sensitivity-specificity trade-off.

8.4 Implications of the exploratory generative study

The exploratory study had two aims: to establish a reliable real-data classification pipeline and to investigate whether GAN-based models could generate CT slices of sufficient quality for downstream use, as stated in Subsection 6.1. As summarised in Subsection 6.14, the generative experiments did not produce synthetic CT slices with sufficient internal anatomical realism, diversity, or class consistency for downstream classifier training.

For that reason, synthetic-only and mixed real-synthetic training were not performed in the current study. This is not treated as a missing experiment, but as a methodological outcome of the exploratory phase. The structured results in this chapter therefore focus on the completed real-data classification experiments.

8.5 Comparative analysis

Within the scope of the completed structured experiments, the main comparison is between the baseline and augmented real-data classifiers. Table 1 shows that the augmented training configuration achieved stronger threshold-independent discrimination than the baseline, with the test AUROC increasing from 0.811 to 0.881.

At the default threshold of 0.5, however, the augmented model produced a highly sensitivity-oriented operating point, with sensitivity rising to 0.950 and specificity falling to 0.480. This shows that AUROC improvement alone is not sufficient to characterise practical classifier behaviour. The threshold-tuning analysis therefore played an important role in interpreting the results.

After validation-based threshold tuning, the augmented classifier achieved the strongest overall performance of the evaluated settings, with a test accuracy of 0.796, a sensitivity of 0.791, and a specificity of 0.801. Compared with the threshold-tuned baseline, this result reflects both better class separation and a better balance between false negatives and false positives.

The main finding of the structured experimental phase is therefore that the augmented training configuration achieved stronger held-out discrimination than the baseline, while validation-based threshold tuning produced a more balanced sensitivity-specificity trade-off. Because the augmented experiment also used validation-based early stopping and weight decay, the observed performance difference cannot be attributed exclusively to data augmentation.

9 Discussion and Limitations

This thesis addressed two related research questions. The first concerned the extent to which GAN-based models can generate anatomically plausible non-contrast brain CT slices from the available training data. The second concerned the performance of a deep-learning classifier for slice-level intracranial haemorrhage detection under different training and decision-threshold settings. The results show that the real-data classification pipeline learnt a meaningful distinction between haemorrhage-positive and haemorrhage-negative slices, whereas the explored generative models learnt mainly coarse global head structure and did not reproduce sufficiently realistic internal anatomy. These outcomes are discussed separately below, followed by the main methodological and clinical limitations of the study.

9.1 Interpretation of the classification findings

The baseline ResNet-18 classifier achieved a test AUROC of 0.811. This result demonstrates that the preprocessing pipeline, patient-level data split, and classifier were able to learn haemorrhage-related information from the selected RSNA subset. The baseline should nevertheless be interpreted as an internal reference experiment. Its training and validation curves showed that training loss continued to decrease while validation performance became less stable, indicating that the model increasingly fitted characteristics of the training data that did not generalise equally well to unseen patients.

This behaviour is consistent with the broader challenge of applying deep learning to limited medical-imaging datasets. Although convolutional neural networks can achieve strong performance for intracranial haemorrhage detection, their generalisation depends on the size and diversity of the training data, the preprocessing strategy, and the evaluation setting [21, 14, 17]. In the present study, only a balanced subset of 4,000 slices was used. This made the experiments computationally manageable, while also increasing the risk of overfitting and limiting the range of anatomical and pathological variation available during training.

The augmented training configuration achieved a higher observed test AUROC than the baseline, increasing from 0.811 to 0.881. The augmented configuration combined small affine transformations, mild intensity perturbations, low-amplitude Gaussian noise, validation-based early stopping, and

weight decay of 10^{-4}. The resulting performance suggests that this more regularised training configuration supported stronger held-out discrimination. Since augmentation, early stopping, and weight decay were introduced together, their individual contributions cannot be separated in the present experiment.

The results at the default decision threshold require a more careful interpretation. At a threshold of 0.5, the augmented classifier achieved a sensitivity of 0.950 and a specificity of 0.480. This operating point produced many positive predictions. However, the higher AUROC shows that the model had improved its overall ranking of positive and negative cases. The poor specificity at threshold 0.5 therefore did not indicate poor discrimination; it indicated that this threshold was unsuitable for the score distribution produced by the augmented model.

Validation-based threshold tuning clarified this distinction. At the selected threshold of 0.84, the augmented classifier achieved an accuracy of 0.796, a sensitivity of 0.791, and a specificity of 0.801. These values represent the strongest and most balanced performance among the evaluated settings. The result illustrates why threshold-independent and threshold-dependent metrics should be considered together. AUROC describes how well the classifier ranks positive cases above negative cases, while sensitivity, specificity, and accuracy describe its behaviour at a particular operating point.

The substantially higher validation-selected threshold for the augmented model shows that the default threshold of 0.5 was not appropriate for its output-score distribution. Probability calibration was not evaluated, so the classifier outputs should not be interpreted as calibrated clinical probabilities.

Furthermore, the threshold selected by maximising Youden's J statistic should not be interpreted as a clinically optimal threshold. This procedure gives equal weight to sensitivity and specificity. In a clinical setting, the relative consequences of false negatives and false positives may justify a different trade-off. Missing a true haemorrhage may have more serious consequences than generating an additional false alarm, but excessive false-positive predictions may also reduce the usefulness of an automated system. Selection of a clinical operating point would therefore require explicit consideration of the intended workflow, disease prevalence, and the consequences of both types of error.

Overall, the classification experiments answer the second research question by showing that the augmented training configuration achieved stronger threshold-independent discrimination than the baseline and that validation-based threshold selection substantially affected the final sensitivity-specificity balance. The results provide evidence that the implemented pipeline can support controlled slice-level ICH classification experiments. They do not establish clinical performance or generalisation beyond the present dataset and experimental setup.

9.2 Interpretation of the generative findings

The explored GAN-based models gradually learnt aspects of the global appearance of non-contrast head CT slices. Across several configurations, the generated outputs developed from noise and regular artefact patterns into more consistent head-like shapes. Head-centred cropping, central-slice filtering, single-class training, longer training, and changes to the generator architecture influenced the visual appearance or training behaviour of the models. GPU acceleration primarily reduced training time and made the later experiments more feasible.

These improvements remained limited to coarse global structure. The generated slices did

not convincingly reproduce the internal relationships between the skull, brain tissue, ventricular structures, and other intracranial anatomy. Sample diversity also remained limited, and the conditional models did not produce a reliable visual distinction between haemorrhage-positive and haemorrhage-negative outputs. Consequently, the generated images did not satisfy the predefined requirements for anatomical realism, diversity, and class consistency.

The generative results are consistent with limitations described in the synthetic medical-imaging literature. GAN training can be affected by instability, mode collapse, image artefacts, and limited coverage of the real data distribution [30, 34]. These problems are particularly important in medical imaging because coarse visual resemblance does not guarantee that diagnostically meaningful anatomical and pathological structures have been reproduced correctly [19, 29].

Several characteristics of the present task may have contributed to the observed results. First, the generative experiments were performed primarily at a resolution of 64×64 pixels. This reduced computational requirements and simplified the initial learning problem, but it also limited the amount of anatomical detail that could be represented. Second, the training data contained substantial variation in patient anatomy, slice position, skull shape, and image appearance. Although cropping and central-slice filtering reduced some of this variation, the models still had to learn a complex distribution from a relatively small number of images.

Third, haemorrhage can be subtle, spatially localised, and heterogeneous in appearance. A conditional generator therefore had to learn both the overall anatomy of a brain CT slice and the smaller class-specific structures associated with haemorrhage. The weak distinction between generated positive and negative examples suggests that the conditioning signal was insufficient to learn these local pathological differences reliably.

The experiments also demonstrate that numerically stable adversarial losses do not establish successful image generation. Several later runs showed relatively stable generator and discriminator losses, while the generated images remained anatomically unrealistic. Qualitative inspection was therefore necessary to identify limitations that were not visible in the loss curves. This supports the broader argument that generative medical-image models require evaluation of anatomy, diversity, and label consistency in addition to training stability.

A further concern for future downstream experiments is the potential domain shift between synthetic and real CT images. Synthetic images may contain intensity patterns, textures, anatomical structures, or artefacts that differ systematically from real scans [19, 29, 30]. Since the generated images in this study were not used for classifier training, synthetic-to-real domain shift was not measured.

The absence of synthetic-only and mixed real-synthetic classifier experiments follows directly from these findings. Training a classifier on images with unrealistic anatomy, restricted diversity, or unreliable labels would not provide a valid evaluation of synthetic-data utility. The decision to stop before downstream training therefore protects the interpretation of the classification experiments. It also changes the contribution of the generative part of the thesis: the results establish the limitations of the tested GAN configurations rather than demonstrating that synthetic images improve ICH classification.

The first research question can therefore be answered within the scope of the tested models as follows: GAN-based models were able to generate coarse head-like CT representations, but the investigated configurations did not generate anatomically plausible, sufficiently diverse, and reliably class-consistent slices. Their usefulness for downstream classification remains an open question for future work with improved generative methods.

9.3 Relationship between the classification and generative results

The two experimental branches provide complementary findings. The classification experiments show that the available real-data subset contains sufficient information to train a functioning slice-level classifier and that the augmented training configuration achieved stronger discrimination than the baseline. Because this configuration combined conservative augmentation with validation-based early stopping and weight decay, the observed difference cannot be attributed exclusively to data augmentation. The generative experiments show that learning a realistic image distribution is substantially more difficult than learning a binary decision boundary on the same underlying data.

This difference is plausible because the two models solve different problems. The classifier only needs to identify features that separate the two classes. A generative model must reproduce the broader distribution of image structure, intensity, anatomical variation, and, in the conditional setting, pathology-specific features. A classifier may therefore achieve useful internal discrimination even when the same dataset is insufficient for high-quality image synthesis.

The stronger performance of the augmented training configuration should also be distinguished from learnt synthetic-data generation. The applied transformations modified existing real slices while preserving their underlying anatomy and labels, and the configuration additionally used validation-based early stopping and weight decay. By contrast, the GANs attempted to generate complete new images from a learnt distribution. The performance difference observed between the baseline and augmented configurations therefore does not imply that GAN-generated images would provide a comparable benefit.

The original motivation of the thesis was to investigate whether synthetic images could support downstream ICH classification. The current results establish two prerequisites for such future work: a functioning real-data classifier and a clear set of minimum quality requirements for generated images. The second prerequisite was not met by the tested GAN configurations, so downstream synthetic-data evaluation was postponed rather than performed with unsuitable inputs.

9.4 Limitations of the classification experiments

The classification results are subject to several limitations.

Dataset scope. The experiments used a curated balanced subset of 4,000 slices rather than the full RSNA dataset. This subset represents only part of the anatomical and pathological variation available in the complete dataset. The results should therefore be interpreted within the selected experimental subset.

Task formulation. The task was formulated as binary slice-level classification using the RSNA *any* label. It does not distinguish between haemorrhage subtypes and treats each two-dimensional slice independently. Information from neighbouring slices and the complete three-dimensional scan was therefore outside the scope of the study.

Internal evaluation. The models were evaluated on one patient-level split from a single dataset. No external test dataset was used, and the experiments did not examine generalisation to different scanners, acquisition protocols, hospitals, or patient populations. The results therefore represent internal held-out performance rather than evidence of broader clinical generalisation.

Run-to-run variability. Each reported classifier configuration represents a single training run. Since the experiments were not repeated across multiple random seeds and no confidence intervals or statistical significance tests were calculated, the uncertainty and run-to-run variability of the

reported differences are unknown.

Training-procedure differences. Augmentation was not the only difference between the two classifier training procedures. The augmented experiment also used early stopping based on validation AUROC and weight decay of 10^{-4}, whereas the baseline was trained using its original fixed-epoch setup without weight decay. The observed difference therefore cannot be attributed exclusively to data augmentation. A more tightly controlled comparison would apply the same checkpoint-selection, stopping, and weight-decay settings to both models.

Test-set reuse. Although thresholds were selected on the validation set, the test-set ROC curves and AUROC values were already inspected during the exploratory classification phase and were later reused in the structured results. The test set was therefore not fully untouched throughout the complete development process. The final metrics should consequently be interpreted as internal hold-out results rather than as a completely independent final estimate.

Probability calibration. The evaluation did not assess probability calibration. Reliability diagrams, expected calibration error, or Brier scores could help determine whether the classifier outputs correspond to meaningful probability estimates. This is particularly relevant because the augmented model required a substantially higher decision threshold than the baseline model.

9.5 Limitations of the generative experiments

The generative study was exploratory and was not designed as a controlled comparison of generative architectures. Multiple preprocessing, architectural, regularisation, and optimisation choices were changed throughout the study. This iterative process was useful for identifying recurring failure modes, but it prevents strong causal conclusions about the effect of any individual modification.

The generated images were assessed primarily through visual inspection of sample grids, real-versus-fake comparisons, sample diversity, class consistency, and training behaviour. No blinded evaluation by radiologists or other clinical experts was performed. The study also did not use a formal anatomical scoring system or quantitative image-quality metrics. The conclusion that the images were unsuitable for downstream use is supported by clear visual deficiencies, but the evaluation remains subjective.

At the same time, common quantitative generative metrics may not fully capture clinically relevant image quality. A low distributional distance does not necessarily demonstrate correct anatomy or pathology. A more complete evaluation would therefore combine quantitative distributional metrics with expert assessment, diversity analysis, tests of label consistency, and eventual downstream task performance.

The experiments were also constrained by the available data and computational resources. Most generative runs used 64×64 images, relatively small batch sizes, and limited training durations. These choices made experimentation feasible, but reduced the capacity to model fine anatomical structures. More advanced methods, larger training sets, higher spatial resolution, or three-dimensional models may produce different results.

No synthetic images were used to train a classifier, so this thesis cannot make an empirical claim about whether GAN-generated data improve or harm downstream ICH classification. The study only establishes that the images generated under the tested conditions were not suitable for a fair downstream comparison.

9.6 Clinical and practical limitations

This thesis evaluates an experimental research pipeline rather than a clinical decision-support system. The reported results are limited to slice-level performance on the selected RSNA test subset and should not be interpreted as evidence of clinical utility.

The classifier operates on isolated preprocessed slices and does not consider complete CT studies, patient history, acquisition metadata, radiological workflow, or clinical presentation. No radiologist comparison, prospective evaluation, or workflow analysis was performed.

The reported sensitivity and specificity therefore describe performance within the selected experimental test set. They should not be interpreted as estimates of safety or diagnostic accuracy in clinical practice.

9.7 Overall interpretation

Within the scope of the current study, the classification branch was more successful than the generative branch. The ResNet-18 pipeline learnt a meaningful binary distinction from the real CT slices, and the augmented training configuration achieved a higher observed AUROC than the baseline. Validation-based threshold tuning then produced a more balanced operating point for the augmented classifier.

The GAN-based models learnt coarse features of the CT distribution, but did not reproduce the detailed anatomy, diversity, and class-specific structure required for downstream training. This negative result is methodologically relevant because it prevents an unsupported claim about synthetic-data utility and identifies image quality as the main barrier to the originally intended synthetic-only and mixed real-synthetic experiments.

The conclusions should therefore remain limited to the tested dataset, models, and experimental conditions. The results support continued investigation of synthetic brain CT generation, while showing that downstream usefulness should only be evaluated after the generated images have met stronger anatomical, diversity, and label-consistency requirements.

10 Conclusions and Further Research

10.1 Conclusions

This thesis investigated two related questions: the extent to which GAN-based models can generate anatomically plausible non-contrast brain CT slices, and how a deep-learning classifier for slice-level intracranial haemorrhage detection performs under baseline and augmented training configurations with validation-based threshold tuning.

Regarding the first research question, the explored GAN-based models were able to learn coarse global characteristics of head CT images. Across the conditional and single-class experiments, the generated outputs developed from noise and regular artefact patterns into more consistent head-like structures. However, the models did not reproduce convincing internal brain anatomy, sufficient sample diversity, or reliable class-specific features. Numerically stable generator and discriminator losses also did not correspond to anatomically realistic outputs. Within the tested architectures, preprocessing choices, training procedures, and computational constraints, the generated images were therefore not suitable for synthetic-only or mixed real-synthetic classifier training.

The generative part of the thesis should consequently be interpreted as a qualitative feasibility study. It demonstrates that learning coarse global CT structure was possible, while also showing that anatomical realism, diversity, and label consistency remained the main barriers to downstream use. Since no classifier was trained on synthetic images, this thesis does not establish whether GAN-generated CT data improve or reduce ICH-classification performance.

Regarding the second research question, the baseline ResNet-18 classifier achieved a test AUROC of 0.811 on the selected patient-level test split. The augmented training configuration achieved a higher test AUROC of 0.881, indicating stronger threshold-independent separation between haemorrhage-positive and haemorrhage-negative slices. Because the augmented experiment also used validation-based early stopping and weight decay, the observed difference cannot be attributed exclusively to data augmentation.

The results also demonstrate the importance of the decision threshold. At the default threshold of 0.5, the augmented classifier achieved high sensitivity but low specificity. After selecting the threshold on the validation set using Youden's J statistic, the augmented classifier achieved a test accuracy of 0.796, a sensitivity of 0.791, and a specificity of 0.801. Threshold selection therefore produced a more balanced operating point, while leaving the threshold-independent AUROC unchanged.

These findings show that the implemented real-data classification pipeline was able to learn a meaningful distinction between haemorrhage-positive and haemorrhage-negative slices. They also show that model discrimination and threshold-dependent behaviour must be evaluated separately. The results remain limited to a balanced subset of the RSNA dataset, one patient-level split, single training runs, and two-dimensional slice-level classification. They should therefore be interpreted as internal experimental findings rather than evidence of clinical utility or broader generalisation.

Overall, the classification branch produced the strongest completed result of this thesis. The generative branch identified clear methodological limitations that prevented a valid downstream synthetic-data comparison. Together, these findings establish a real-data classification reference and clarify the conditions that future synthetic CT images would need to satisfy before their usefulness for ICH classification can be evaluated fairly.

10.2 Further research

Future research should first investigate whether improved generative methods can produce synthetic CT images with more realistic internal anatomy, greater sample diversity, and stronger consistency with the requested haemorrhage label. Possible improvements include higher-resolution or progressively trained generation, more effective conditioning mechanisms, alternative generator and discriminator architectures, and larger or more carefully curated training datasets.

Alternative generative approaches should also be considered. Diffusion models may provide more stable training and higher image quality than the GAN configurations explored in this thesis, although they generally require greater computational resources. Future work could compare conditional GANs and diffusion-based models using the same data, preprocessing pipeline, and evaluation criteria.

Generative evaluation should be extended beyond visual inspection. Future studies could combine quantitative image-distribution metrics with structured anatomical assessment, diversity analysis, class-consistency evaluation, and expert review. Such evaluation would help determine whether generated images reproduce clinically relevant structure rather than only coarse visual

similarity.

Only after the generated images have demonstrated sufficient realism, diversity, and label consistency should their downstream utility be evaluated. This could involve comparing classifiers trained on real data, synthetic data, and mixtures of real and synthetic data, while evaluating all models on the same held-out real-data test set. Such experiments should also examine whether classifiers learn synthetic-specific artefacts or whether performance transfers reliably to real CT images.

The classification experiments could be strengthened by applying identical training-control procedures to the baseline and augmented configurations. In particular, both models should use the same checkpoint-selection, early-stopping, and weight-decay settings so that the individual effect of data augmentation can be assessed more directly. Repeating each configuration across multiple random seeds and reporting confidence intervals would provide information about run-to-run variability and the uncertainty of observed performance differences.

Future classification studies should also use larger portions of the RSNA dataset and consider haemorrhage-subtype classification rather than only the binary *any* label. Three-dimensional models or approaches that combine information from neighbouring slices could incorporate volumetric context that is absent from the current two-dimensional design. Probability calibration and alternative threshold-selection strategies could additionally be investigated to better characterise the relationship between model scores and operating-point behaviour.

Finally, broader generalisation would need to be evaluated using external datasets from different institutions, scanners, and acquisition settings. Clinical interpretation would additionally require patient-level assessment, expert comparison, and evaluation within the intended diagnostic workflow. These steps fall outside the scope of the present thesis but are necessary before conclusions about clinical usefulness can be drawn.

References

[1] Manal Alamir and Manal AlGhamdi. The role of generative adversarial network in medical image analysis: An in-depth survey. *ACM Computing Surveys*, 55(5):1–36, 2022.

[2] Haidara Almansour, Jan Brendel, and Daniel Wessling. Computed tomography. In *ESR Modern Radiology eBook*. European Society of Radiology, 2024. Available at: https://doi.org/10.26044/esr-modern-radiology-04.

[3] Swarnambiga Ayyachamy, Varghese Alex, Mahendra Khened, and Ganapathy Krishnamurthi. Medical image retrieval using resnet-18. In *Medical Imaging 2019: Imaging Informatics for Healthcare, Research, and Applications*, volume 10954, pages 233–241. SPIE, 2019.

[4] Angona Biswas, Nasim Md Abdullah Al, Al Imran, Anika Tabassum Sejuty, Fabliha Fairooz, Sai Puppala, and Sajedul Talukder. Generative adversarial networks for data augmentation. In *Data Driven Approaches on Medical Imaging*, pages 159–177. Springer, Cham, 2023.

[5] Chao Chen, Nor Ashidi Mat Isa, and Xin Liu. A review of convolutional neural network based methods for medical image classification. *Computers in Biology and Medicine*, 185:109507, 2025.

[6] Tami D. DenOtter and Johanna Schubert. Hounsfield unit. In *StatPearls*. StatPearls Publishing, Treasure Island (FL), 2026. Available at: https://www.ncbi.nlm.nih.gov/books/NBK547721/.

[7] Xingyue Fu, Xiaoshuang Li, Eduardo Delamare, Adam G. Dunn, Lei Bi, and Jinman Kim. A systematic review of generative artificial intelligence techniques for synthetic medical image datasets: Quality, models, public availability and applications. *Computer Methods and Programs in Biomedicine*, 280:109331, 2026.

[8] Romane Gauriau, Christopher Bridge, Lina Chen, Felipe Kitamura, Neil A. Tenenholtz, John E. Kirsch, Katherine P. Andriole, Mark H. Michalski, and Bernardo C. Bizzo. Using DICOM metadata for radiological image series categorization: A feasibility study on large clinical brain MRI datasets. *Journal of Digital Imaging*, 33(3):747–762, 2020.

[9] Wei Gong, Deep K. Vaishnani, Xuan-Chen Jin, Jing Zeng, Wei Chen, Huixia Huang, Yu-Qing Zhou, Khaing Wut Yi Hla, Chen Geng, and Jun Ma. Evaluation of an enhanced ResNet-18 classification model for rapid on-site diagnosis in respiratory cytology. *BMC Cancer*, 25(1):10, 2025.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Jeremy J. Heit, Michael Iv, and Max Wintermark. Imaging of intracranial hemorrhage. *Journal of Stroke*, 19(1):11–27, 2016.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[14] Ping Hu, Tengfeng Yan, Bing Xiao, Hongxin Shu, Yilei Sheng, Yanze Wu, Lei Shu, Shigang Lv, Minhua Ye, Yanyan Gong, Miaojing Wu, and Xingen Zhu. Deep learning-assisted detection and segmentation of intracranial hemorrhage in noncontrast computed tomography scans of acute stroke patients: A systematic review and meta-analysis. *International Journal of Surgery*, 110(6):3839–3847, 2024.

[15] Mahmoud Ibrahim, Yasmina Al Khalil, Sina Amirrajab, Chang Sun, Marcel Breeuwer, Josien Pluim, Bart Elen, Gökhan Ertaylan, and Michel Dumontier. Generative AI for synthetic data across multiple medical modalities: A systematic review of recent developments and challenges. *Computers in Biology and Medicine*, 189:109834, 2025.

[16] Jiwoong J. Jeong, Amara Tariq, Tobiloba Adejumo, Hari Trivedi, Judy W. Gichoya, and Imon Banerjee. Systematic review of generative adversarial networks (GANs) for medical image classification and segmentation. *Journal of Digital Imaging*, 35(2):137–152, 2022.

[17] Armin Karamian and Ali Seifi. Diagnostic accuracy of deep learning for intracranial hemorrhage detection in non-contrast brain CT scans: A systematic review and meta-analysis. *Journal of Clinical Medicine*, 14(7):2377, 2025.

[18] Sachin Khanduri. Computed tomography physics. In *Textbook of Radiology for X-Ray, CT, MRI, BSc, BRIT and MSc Technicians*, pages 101–109. Jaypee Brothers Medical Publishers, New Delhi and London, 2nd edition, 2022. ISBN: 9789354655128.

[19] Lennart R. Koetzier, Jie Wu, Domenico Mastrodicasa, Aline Lutz, Matthew Chung, W. Adam Koszek, Jayanth Pratap, Akshay S. Chaudhari, Pranav Rajpurkar, Matthew P. Lungren, and Martin J. Willemink. Generating synthetic data for medical imaging. *Radiology*, 312(3):e232471, 2024.

[20] Michele Larobina. Thirty years of the DICOM standard. *Tomography*, 9(5):1829–1838, 2023.

[21] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 2018.

[22] Radiological Society of North America. RSNA intracranial hemorrhage detection challenge. Kaggle competition and dataset, https://www.kaggle.com/competitions/rsna-intracranial-hemorrhage-detection, 2019.

[23] Vasileios C. Pezoulas, Dimitrios I. Zaridis, Eugenia Mylona, Christos Androutsos, Kosmas Apostolidis, Nikolaos S. Tachos, and Dimitrios I. Fotiadis. Synthetic data generation methods in healthcare: A review on open-source tools and methods. *Computational and Structural Biotechnology Journal*, 23:2892–2910, 2024.

[24] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F. Da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Brain imaging generation with latent diffusion models. In *Deep Generative Models*, volume 13609 of *Lecture Notes in Computer Science*, pages 117–126, Cham, 2022. Springer.

[25] Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T. Barron, Amit H. Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, C. Karen Liu, Lingjie Liu, Ben Mildenhall, Matthias Nießner, Björn Ommer, Christian Theobalt, Peter Wonka, and Gordon Wetzstein. State of the art on diffusion models for visual computing. *Computer Graphics Forum*, 43(2):e15063, 2024.

[26] D. R. Sarvamangala and Raghavendra V. Kulkarni. Convolutional neural networks in medical image understanding: A survey. *Evolutionary Intelligence*, 15(1):1–22, 2022.

[27] Hoo-Chang Shin, Neil A. Tenenholtz, Jameson K. Rogers, Christopher G. Schwarz, Matthew L. Senjem, Jeffrey L. Gunter, Katherine P. Andriole, and Mark Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In Orcun Goksel, Ipek Oguz, Ali Gooya, and Ninon Burgos, editors, *Simulation and Synthesis in Medical Imaging*, volume 11037 of *Lecture Notes in Computer Science*, pages 1–11, Cham, 2018. Springer.

[28] A. Robert Singh, Sri Durgesh R, and Addwin Vibro D. Feature controlled synthetic medical image generation using conditional generative adversarial network. In *2025 International Conference on Inventive Computation Technologies (ICICT)*, pages 136–142. IEEE, 2025.

[29] Elena Sizikova, Andreu Badal, Jana G. Delfino, Miguel Lago, Brandon Nelson, Niloufar Saharkhiz, Berkman Sahiner, Ghada Zamzmi, and Aldo Badano. Synthetic data in radiological imaging: Current state and future outlook. *BJR|Artificial Intelligence*, 1(1):ubae007, 2024.

[30] Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalande. GANs for medical image synthesis: An empirical study. *Journal of Imaging*, 9(3):69, 2023.

[31] Steven Tenny, Joe M. Das, and William Thorell. Intracranial hemorrhage overview. In *StatPearls*. StatPearls Publishing, Treasure Island (FL), 2026. Updated 2024. Available at: https://www.ncbi.nlm.nih.gov/books/NBK470242/.

[32] Pedro Vilela and Martin Wiesmann. Nontraumatic intracranial hemorrhage. In *Diseases of the Brain, Head and Neck, Spine 2020–2023: Diagnostic Imaging*, IDKD Springer Series, chapter 5, pages 45–57. Springer, Cham, 2020.

[33] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[34] Jeong Taek Yoon, Kyung Mi Lee, Jang-Hoon Oh, Hyug-Gi Kim, and Ji Won Jeong. Insights and considerations in development and performance evaluation of generative adversarial networks (GANs): What radiologists need to know. *Diagnostics*, 14(16):1756, 2024.

A Data Preparation and Classification Scripts

This section contains the main code excerpts used to construct the balanced RSNA subset, preprocess the CT slices, create the patient-level data split, train the baseline and augmented classifiers, and perform validation-based threshold tuning. The order follows the dataset, classification, and structured-experiment chapters of the thesis.

A.1 Balanced subset selection and extraction

The exploratory classification pipeline started from a balanced subset of the RSNA challenge data. The script below selected 2000 haemorrhage-positive and 2000 haemorrhage-negative slices from the original metadata file and stored the resulting image identifiers for later extraction and preprocessing.

```
RANDOM_STATE = 42
N_POS = 2000
N_NEG = 2000

df = pd.read_csv(csv_path)

parts = df["ID"].str.rsplit("_", n=1, expand=True)
df["Image"] = parts[0]
df["Subtype"] = parts[1]

pivot = (
    df.pivot_table(
        index="Image",
        columns="Subtype",
        values="Label",
        aggfunc="max"
    )
    .reset_index()
)

pivot["any_ich"] = pivot["any"].astype(int)

pos = pivot[pivot["any_ich"] == 1].sample(N_POS, random_state=RANDOM_STATE)
neg = pivot[pivot["any_ich"] == 0].sample(N_NEG, random_state=RANDOM_STATE)

subset = (
    pd.concat([pos, neg])
    .sample(frac=1, random_state=RANDOM_STATE)
    .reset_index(drop=True)
)

subset.to_csv(out_path, index=False)
```

The selected DICOM files were then extracted from the original compressed RSNA archive using the stored subset image identifiers.

```
subset = pd.read_csv(subset_csv)

wanted = [
```

```
    f"rsna-intracranial-hemorrhage-detection/stage_2_train/{img}.dcm"
    for img in subset["Image"]
]

with zipfile.ZipFile(zip_path, "r") as zf:
    all_names = set(zf.namelist())
    for member in wanted:
        if member in all_names:
            zf.extract(member, path=out_dir)
```

A.2 Preprocessing pipeline for the balanced RSNA subset

The following preprocessing script converted raw DICOM slices to Hounsfield Units, applied a standard brain window, resized the images to 256×256 pixels, and stored the processed outputs as NumPy arrays together with binary haemorrhage labels.

```
def load_dicom(path):
    dcm = pydicom.dcmread(path)
    img = dcm.pixel_array.astype(np.float32)

    intercept = float(getattr(dcm, "RescaleIntercept", 0.0))
    slope = float(getattr(dcm, "RescaleSlope", 1.0))
    img = img * slope + intercept

    return img

def window_brain(img, center=40, width=80):
    lower = center - width / 2
    upper = center + width / 2
    img = np.clip(img, lower, upper)
    img = (img - lower) / (upper - lower)
    return img

for _, row in tqdm(labels_df.iterrows(), total=len(labels_df)):
    image_id = row["Image"]
    dcm_path = RAW_DIR / f"{image_id}.dcm"

    if not dcm_path.exists():
        continue

    img = load_dicom(dcm_path)
    img = window_brain(img)
    img = cv2.resize(img, (256, 256), interpolation=cv2.INTER_LINEAR)

    np.save(OUT_IMG_DIR / f"{image_id}.npy", img)

    processed_labels.append({
        "image_id": image_id,
        "any_ich": int(row["any_ich"])
    })

pd.DataFrame(processed_labels).to_csv(OUT_LABELS, index=False)
```

A.3 Patient-level split construction

To prevent data leakage between training and evaluation sets, the exploratory classification pipeline used a patient-level split. Patient and study identifiers were first read from the DICOM metadata, after which patients were assigned to train, validation, and test partitions.

```
for image_id in df["Image"]:
    dcm_path = RAW_DIR / f"{image_id}.dcm"
    dcm = pydicom.dcmread(dcm_path, stop_before_pixels=True)

    patient_id = str(getattr(dcm, "PatientID", "unknown"))
    study_id = str(getattr(dcm, "StudyInstanceUID", "unknown"))

    patient_ids.append(patient_id)
    study_ids.append(study_id)

df["patient_id"] = patient_ids
df["study_id"] = study_ids
```

```
patient_df = (
    df.groupby("patient_id", as_index=False)
      .agg(patient_label=("any_ich", "max"))
)

pos_patients = patient_df[patient_df["patient_label"] == 1]["patient_id"].
                                        to_numpy()
neg_patients = patient_df[patient_df["patient_label"] == 0]["patient_id"].
                                        to_numpy()

rng.shuffle(pos_patients)
rng.shuffle(neg_patients)

def split_group(arr, train=0.70, val=0.15, test=0.15):
    n = len(arr)
    n_train = int(n * train)
    n_val = int(n * val)
    train_ids = arr[:n_train]
    val_ids = arr[n_train:n_train + n_val]
    test_ids = arr[n_train + n_val:]
    return train_ids, val_ids, test_ids
```

```
df["split"] = df["patient_id"].apply(assign_split)

split_df = df[["Image", "any_ich", "patient_id", "study_id", "split"]].rename(
    columns={"Image": "image_id"}
)

split_df.to_csv(OUT_SPLIT, index=False)
```

A.4 Baseline classifier training script

The exploratory baseline classifier used a ResNet-18 architecture adapted for binary classification. Preprocessed single-channel CT slices were repeated across three channels before being passed to

the network.

```
DEVICE = torch.device("mps" if torch.backends.mps.is_available() else "cpu")

BATCH_SIZE = 32
EPOCHS = 8
LR = 1e-4
```

```
class RSNADataset(Dataset):
    def __init__(self, df):
        self.df = df.reset_index(drop=True)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        image_id = row["image_id"]
        y = np.float32(row["any_ich"])

        x = np.load(IMG_DIR / f"{image_id}.npy").astype(np.float32)
        x = np.expand_dims(x, axis=0)
        x = np.repeat(x, 3, axis=0)
        x = torch.tensor(x, dtype=torch.float32)
        y = torch.tensor([y], dtype=torch.float32)

        return x, y
```

```
def build_model():
    model = models.resnet18(weights=None)
    model.fc = nn.Linear(model.fc.in_features, 1)
    return model.to(DEVICE)
```

```
criterion = nn.BCEWithLogitsLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=LR)
```

A.5 Augmented classifier training script

A second exploratory classifier setting introduced conservative training-time data augmentation. The architecture remained the same as in the baseline experiment, while random affine perturbations, mild intensity variation, and Gaussian noise were added during training.

```
class AddGaussianNoise:
    def __init__(self, std=0.01):
        self.std = std

    def __call__(self, x):
        noise = torch.randn_like(x) * self.std
        x = x + noise
        return torch.clamp(x, 0.0, 1.0)
```

```
train_transform = transforms.Compose([
    transforms.RandomAffine(
        degrees=10,
        translate=(0.03, 0.03),
```

```
        scale=(0.97, 1.03),
        fill=0.0
    ),
    transforms.ColorJitter(brightness=0.08, contrast=0.08),
    AddGaussianNoise(std=0.01),
])
```

```
PATIENCE = 3
optimizer = torch.optim.Adam(model.parameters(), lr=LR, weight_decay=1e-4)
```

```
if val_metrics["auroc"] > best_val_auc:
    best_val_auc = val_metrics["auroc"]
    epochs_without_improvement = 0
    torch.save(model.state_dict(), best_path)
else:
    epochs_without_improvement += 1

if epochs_without_improvement >= PATIENCE:
    print(f"\nEarly stopping triggered after epoch {epoch}.")
    break
```

A.6 Threshold tuning and ROC evaluation

After classifier training, a separate evaluation script was used to compute ROC curves and to select a tuned decision threshold using Youden's statistic on the validation set.

```
def compute_metrics(y_true, probs, threshold=0.5):
    preds = (probs >= threshold).astype(int)
    auc = roc_auc_score(y_true, probs)
    acc = accuracy_score(y_true, preds)
    tn, fp, fn, tp = confusion_matrix(y_true, preds).ravel()

    sensitivity = tp / (tp + fn) if (tp + fn) > 0 else 0.0
    specificity = tn / (tn + fp) if (tn + fp) > 0 else 0.0
    balanced_accuracy = (sensitivity + specificity) / 2.0
    youdens_j = sensitivity + specificity - 1.0
```

```
def sweep_thresholds(y_true, probs):
    rows = []
    thresholds = np.linspace(0.0, 1.0, 201)

    for t in thresholds:
        rows.append(compute_metrics(y_true, probs, threshold=t))

    return pd.DataFrame(rows)
```

```
best_idx = threshold_table["youdens_j"].idxmax()
best_row = threshold_table.loc[best_idx]
best_threshold = float(best_row["threshold"])
```

B Exploratory Generative Experiments on the MacBook

This section follows the chronology of the early generative study. The local exploration began with conditional GAN experiments, including an initial higher-resolution configuration, a reduced-resolution configuration, and later stabilised variants. The explored changes included discriminator simplification, spectral normalisation, dropout, instance noise, separate generator and discriminator learning rates, and fixed-noise monitoring. The complete scripts for every early variant are not reproduced here; the shared conditional architecture is shown first. The appendix then documents the transition to negative-only data, GAN-specific filtering and cropping, a transitional fixed-label conditional setup, and the explicitly unconditional single-class GAN.

B.1 Overview of the early two-class conditional GAN experiments

The first generative experiments used a two-class conditional GAN in which both the generator and the discriminator received the binary haemorrhage label. The purpose of these experiments was to determine whether a relatively simple conditional adversarial model could learn both global head CT structure and visually distinguishable characteristics of haemorrhage-positive and haemorrhage-negative slices.

The initial experiment used transposed-convolutional generation at a resolution of 128×128 pixels. Subsequent experiments reduced the image resolution to 64×64 pixels to simplify the generation task and reduce computational cost.

Later conditional variants also modified the discriminator and the training procedure. The discriminator capacity was reduced, while spectral normalisation and dropout were introduced to limit discriminator dominance. Instance noise was added to both real and generated images before discriminator evaluation. Separate generator and discriminator learning rates were used, and fixed latent vectors were introduced to monitor the development of generated samples consistently across epochs.

These experiments correspond to the initial cGAN, the reduced-resolution cGAN, and the stabilised cGAN variants presented in Section 6.6. The variants retained the same general class-conditional framework, while differing in image resolution, discriminator capacity, regularisation, and optimisation settings. The conditional generator and discriminator structures used during this phase are shown in Appendix B.2.

B.2 Initial conditional generator and discriminator definitions

The early MacBook cGAN experiments used conditional generator and discriminator components based on class embeddings. The exact experimental variants differed in image resolution and training configuration, but retained the general conditional structure shown below.

```
class ConditionalGenerator(nn.Module):
    def __init__(self, z_dim=128, num_classes=2, base_channels=32):
        super().__init__()
        self.label_emb = nn.Embedding(num_classes, 16)

        in_dim = z_dim + 16

        self.net = nn.Sequential(
```

```
            nn.ConvTranspose2d(in_dim, base_channels * 16, 4, 1, 0, bias=False)
                                                    ,
            nn.BatchNorm2d(base_channels * 16),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 16, base_channels * 8, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 8),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 8, base_channels * 4, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 4, base_channels * 2, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 2),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 2, 1, 4, 2, 1, bias=False),
            nn.Tanh(),
        )
```

```
class ConditionalDiscriminator(nn.Module):
    def __init__(self, num_classes=2, img_size=64, base_channels=24, dropout=0.
                                            30):
        super().__init__()
        self.img_size = img_size
        self.label_emb = nn.Embedding(num_classes, img_size * img_size)

        self.net = nn.Sequential(
            spectral_norm(nn.Conv2d(2, base_channels, 4, 2, 1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels, base_channels * 2, 4, 2, 1,
                                                    bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels * 2, base_channels * 4, 4, 2,
                                                     1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels * 4, base_channels * 8, 4, 2,
                                                     1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),

            spectral_norm(nn.Conv2d(base_channels * 8, 1, 4, 1, 0, bias=False))
                                                    ,
        )
```

B.3 GAN-specific negative-only subset construction

To simplify the generative task, a GAN-specific subset containing only haemorrhage-negative slices was constructed. The corresponding image identifiers were sampled from the original metadata and extracted from the archive in the same way as for the balanced classification subset.

```
RANDOM_STATE = 42
N_NEG = 2000

pivot["any_ich"] = pivot["any"].astype(int)

neg = pivot[pivot["any_ich"] == 0].sample(N_NEG, random_state=RANDOM_STATE).
                                        reset_index(drop=True)

neg.to_csv(out_path, index=False)
```

```
wanted = [
    f"rsna-intracranial-hemorrhage-detection/stage_2_train/{img}.dcm"
    for img in subset["Image"]
]

with zipfile.ZipFile(zip_path, "r") as zf:
    all_names = set(zf.namelist())
    for member in wanted:
        if member in all_names:
            zf.extract(member, path=out_dir)
```

B.4 GAN-specific preprocessing, central-slice filtering, and cropping

The negative-only GAN subset was preprocessed in the same general way as the classification data, but was subsequently filtered to retain more content-rich central slices. In a later stage, head-centred cropping was introduced before resizing.

```
img = load_dicom(dcm_path)
img = window_brain(img)
img = cv2.resize(img, (256, 256), interpolation=cv2.INTER_LINEAR)
np.save(OUT_IMG_DIR / f"{image_id}.npy", img)
```

```
foreground_mask = img > 0.08
foreground_fraction = foreground_mask.mean()

h, w = img.shape
center_crop = img[h//4:3*h//4, w//4:3*w//4]
center_mean = float(center_crop.mean())
```

```
filtered = stats_df[
    (stats_df["foreground_fraction"] >= 0.15) &
    (stats_df["foreground_fraction"] <= 0.60)
].copy()
```

The cropped version used an additional head-centred cropping step:

```
def crop_to_head(img, threshold=0.08, pad=12):
    mask = (img > threshold).astype(np.uint8)
    num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(mask,
                                                    connectivity=8)

    if num_labels <= 1:
        return img

    largest_label = 1 + np.argmax(stats[1:, cv2.CC_STAT_AREA])

    x = stats[largest_label, cv2.CC_STAT_LEFT]
    y = stats[largest_label, cv2.CC_STAT_TOP]
    w = stats[largest_label, cv2.CC_STAT_WIDTH]
    h = stats[largest_label, cv2.CC_STAT_HEIGHT]

    x0 = max(0, x - pad)
    y0 = max(0, y - pad)
    x1 = min(img.shape[1], x + w + pad)
    y1 = min(img.shape[0], y + h + pad)

    cropped = img[y0:y1, x0:x1]
    return cropped if cropped.size > 0 else img
```

For the cropped subset, the filtering criteria were also tightened:

```
filtered = stats_df[
    (stats_df["foreground_fraction"] >= 0.25) &
    (stats_df["foreground_fraction"] <= 0.55) &
    (stats_df["center_mean"] >= 0.20) &
    (stats_df["center_mean"] <= 0.65)
].copy()
```

B.5 Transitional conditional architecture trained on negative-only data

After the early two-class conditional GAN experiments, the generative task was simplified by restricting the training data to cropped haemorrhage-negative slices. As an intermediate step, the existing conditional generator and discriminator components were retained, but every sample was assigned the fixed class label zero. The architecture was therefore still technically conditional, while the training setup was effectively single-class.

This transitional experiment used 64×64 images, a latent dimension of $Z = 128$, a batch size of 16, and 30 training epochs.

```
IMG_DIR = BASE / "data" / "processed" / "rsna_gan_cropped" / "images"
LABELS_CSV = BASE / "data" / "processed" / "rsna_gan_cropped" / "central_labels
                                                .csv"

DEVICE = torch.device("mps" if torch.backends.mps.is_available() else "cpu")

IMG_SIZE = 64
BATCH_SIZE = 16
EPOCHS = 30
Z_DIM = 128
```

```
NUM_CLASSES = 2
LR_G = 2e-4
LR_D = 5e-5
BETA1 = 0.5
LABEL_SMOOTH_REAL = 0.9
INSTANCE_NOISE_STD = 0.03
N_CRITIC = 1
G_STEPS = 2
```

The dataset returned every cropped haemorrhage-negative image with the same class label:

```
class RSNAGanDataset(Dataset):
    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        image_id = row["image_id"]

        x = np.load(
            IMG_DIR / f"{image_id}.npy"
        ).astype(np.float32)

        x = torch.tensor(
            x,
            dtype=torch.float32
        ).unsqueeze(0)

        x = F.interpolate(
            x.unsqueeze(0),
            size=(IMG_SIZE, IMG_SIZE),
            mode="bilinear",
            align_corners=False
        ).squeeze(0)

        x = (x - 0.5) / 0.5

        # All samples use the haemorrhage-negative condition.
        label = torch.tensor(0, dtype=torch.long)

        return x, label
```

The fixed labels were still supplied to both conditional networks during training:

```
for real_imgs, labels in loader:
    real_imgs = real_imgs.to(DEVICE)
    labels = labels.to(DEVICE)

    z = torch.randn(
        real_imgs.size(0),
        Z_DIM,
        device=DEVICE
    )

    fake_imgs = G(z, labels)

    real_logits = D(real_imgs, labels)
    fake_logits = D(fake_imgs.detach(), labels)
```

Because all values in `labels` were zero, the model used only the haemorrhage-negative condition. This configuration formed an intermediate step between the earlier two-class cGANs and the later explicitly unconditional single-class GAN.

B.6 Single-class GAN training on cropped haemorrhage-negative slices

Following the transitional fixed-label experiment, the model was reformulated as an explicitly unconditional single-class GAN. The dataset contained only cropped haemorrhage-negative central slices, and neither the generator nor the discriminator received a class label.

The experiment used 64×64 images, a latent dimension of $Z = 64$, a batch size of 16, and 30 training epochs. The training procedure combined hinge loss, feature matching, spectral normalisation, discriminator dropout, scheduled instance noise, and fixed-noise monitoring.

```
DATA_DIR = BASE / "data" / "processed" / "rsna_gan_cropped"
LABELS_CSV = DATA_DIR / "central_labels.csv"

DEVICE = torch.device("mps" if torch.backends.mps.is_available() else "cpu")

IMG_SIZE = 64
BATCH_SIZE = 16
EPOCHS = 30
Z_DIM = 64
```

```
LR_G = 2e-4
LR_D = 5e-5
BETA1 = 0.5
INSTANCE_NOISE_STD = 0.02
FM_LAMBDA = 2.0
N_CRITIC = 1
G_STEPS = 1
```

```
d_loss_real = torch.mean(F.relu(1.0 - real_logits))
d_loss_fake = torch.mean(F.relu(1.0 + fake_logits))
d_loss = d_loss_real + d_loss_fake
```

```
g_adv = -torch.mean(gen_logits)
real_feats = D.forward_features(real_imgs)
fake_feats = D.forward_features(gen_imgs)
fm_loss = feature_matching_loss(real_feats, fake_feats)
g_loss = g_adv + FM_LAMBDA * fm_loss
```

```
def schedule_instance_noise(epoch):
    return INSTANCE_NOISE_STD * max(0.0, 1.0 - (epoch - 1) / EPOCHS)
```

B.7 Single-class generator and discriminator definitions

The single-class GAN experiments used the following unconditional generator and discriminator structures.

```
class Generator(nn.Module):
    def __init__(self, z_dim=64, base_channels=32):
        super().__init__()

        self.net = nn.Sequential(
            nn.ConvTranspose2d(z_dim, base_channels * 16, 4, 1, 0, bias=False),
            nn.BatchNorm2d(base_channels * 16),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 16, base_channels * 8, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 8),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 8, base_channels * 4, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 4, base_channels * 2, 4, 2, 1,
                                                    bias=False),
            nn.BatchNorm2d(base_channels * 2),
            nn.ReLU(True),

            nn.ConvTranspose2d(base_channels * 2, 1, 4, 2, 1, bias=False),
            nn.Tanh(),
        )
```

```
class Discriminator(nn.Module):
    def __init__(self, base_channels=24, dropout=0.25):
        super().__init__()

        self.net = nn.Sequential(
            spectral_norm(nn.Conv2d(1, base_channels, 4, 2, 1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels, base_channels * 2, 4, 2, 1,
                                                    bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels * 2, base_channels * 4, 4, 2,
                                                     1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout2d(dropout),

            spectral_norm(nn.Conv2d(base_channels * 4, base_channels * 8, 4, 2,
                                                     1, bias=False)),
            nn.LeakyReLU(0.2, inplace=True),

            spectral_norm(nn.Conv2d(base_channels * 8, 1, 4, 1, 0, bias=False))
                                                    ,
```

```
)
```

C Single-Class GAN Experiments on the Workstation

After the local workflow had been established, the cropped single-class GAN was transferred to the Workstation. The experiments progressed from short CPU sanity checks to 20-epoch and 40-epoch CPU runs. The shared settings, experiment-specific configurations, and chronological overview are provided below.

This section summarises the configurations used for the exploratory cropped single-class GAN runs. The complete implementation was based on `src/training/train_gan_singleclass.py`. Only the experiment-specific configuration is reproduced here; the full training implementation is available in the accompanying project repository.

All runs used the cropped haemorrhage-negative dataset with central-slice filtering. Unless stated otherwise, the runs used the following shared settings:

```
BASE = Path("/local/s3209199")
DATA_DIR = Path("data/processed/rsna_gan_cropped")
IMG_DIR = DATA_DIR / "images"
LABELS_CSV = DATA_DIR / "central_labels.csv"

IMG_SIZE = 64
BATCH_SIZE = 8
Z_DIM = 64

LR_G = 2e-4
LR_D = 5e-5
BETA1 = 0.5
INSTANCE_NOISE_STD = 0.02
FM_LAMBDA = 2.0
N_CRITIC = 1
G_STEPS = 1
```

Two short CPU runs were used to verify that data loading, training, checkpointing, logging, and figure generation worked correctly in the Workstation environment.

C.1 CPU sanity-check configurations

C.1.1 2-epoch CPU sanity check script

```
EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_cpucheck"
)

DEVICE = torch.device("cpu")

EPOCHS = 2
```

C.1.2 5-epoch CPU sanity check script

```
EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_5ep"
)

DEVICE = torch.device("cpu")

EPOCHS = 5
```

C.2 20-epoch CPU configuration

The first extended CPU experiment used the same model and optimisation settings as the sanity checks, with training extended to 20 epochs.

```
EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_leiden_e20"
)

DEVICE = torch.device("cpu")

EPOCHS = 20
```

Outputs were saved at epochs 1, 5, 10, 15, and 20.

C.3 40-epoch CPU configuration

The second extended CPU experiment increased the training duration to 40 epochs while retaining the same dataset, architecture, and optimisation settings.

```
EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_leiden_e40"
)

DEVICE = torch.device("cpu")

EPOCHS = 40
```

Outputs were saved at epochs 1, 5, 10, 15, 20, 25, 30, 35, and 40.

C.4 Overview of the CPU single-class GAN experiments

Run	Script	Main Changes	Shared Settings	Main Observation	Conclusion
2-epoch CPU sanity check	C.1.1	Device set to CPU; batch size reduced to 8; epochs reduced to 2; Leiden base path used.	Cropped negative-only dataset; central labels; image size 64×64, latent dimension $Z = 64$, generator learning rate 2×10^{-4}, discriminator learning rate 5×10^{-5}, feature matching loss with $\lambda = 2.0$, initial instance-noise standard deviation 0.02, one discriminator step per iteration, and one generator step per iteration.	The full cropped single-class GAN pipeline executed successfully on the Workstation. Data loading, training, output saving, and logging all worked correctly.	Technical validation only, the pipeline could run on the Workstation and that data loading, training, saving, logging, and checkpointing worked correctly; no substantive conclusion about image realism.
5-epoch CPU sanity check	C.1.2	Epochs increased to 5; save condition adjusted to store outputs at the first and final epoch.	Same as above.	The run produced saved outputs at epochs 1 and 5. Early generated images moved slightly beyond the most trivial artefacts, but remained strongly dominated by stripe and checkerboard-like structure.	Provided a first view of early training dynamics and confirmed that the workflow of running, saving, retrieving, and integrating outputs functioned correctly. The pipeline is stable enough for short exploratory runs, but the generated images are still far from anatomically realistic.
20-epoch CPU run	C.2	Epochs increased to 20; outputs saved at epochs 1, 5, 10, 15, and 20.	Same as above.	The generator gradually learnt coarse head-like structure between epochs 10 and 20, but internal anatomy remained noisy and artefact-rich. Real-versus-fake comparisons still showed a large gap with real CT slices.	The generator moved beyond early artefacts and learnt coarse head-like structure, but generated images remained noisy, artefactual, and anatomically unrealistic. Longer training improved global structure, but did not yet produce anatomically convincing synthetic CT slices.
40-epoch CPU run	C.3	Epochs increased to 40; outputs saved every 5 epochs and at the final epoch.	Same as above.	Compared with the 20-epoch run, the model continued training without collapse, but the qualitative outputs did not show a clear breakthrough beyond the level reached around epoch 20. Artefacts and unrealistic internal structure remained prominent.	Longer training did not produce a clear qualitative breakthrough beyond the level reached around epoch 20. The model appeared to plateau within this setup. Simply extending training within the same setup is unlikely to solve the realism problem; further progress will probably require a methodological change.

Table 2: Overview of the main exploratory cropped single-class GAN runs performed on the Workstation.

D GPU-Based Single-Class GAN Experiments

The next phase replicated the cropped single-class GAN on GPU and then replaced the transposed-convolutional generator with an upsample-based generator. These experiments are presented in the same order as in the exploratory study.

This section contains the main configuration and architecture excerpts used for the exploratory cropped single-class GAN runs on GPU. These scripts correspond to the runs summarised in Table 3.

D.1 40-epoch GPU training script

```
BASE = Path("/local/s3209199")
DATA_DIR = BASE / "data" / "processed" / "rsna_gan_cropped"
IMG_DIR = DATA_DIR / "images"
LABELS_CSV = DATA_DIR / "central_labels.csv"

EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_leiden_gpu_e40"
)
CHECKPOINT_DIR = EXPERIMENT_DIR / "checkpoints"
METRICS_DIR = EXPERIMENT_DIR / "metrics"
FIGURES_DIR = EXPERIMENT_DIR / "figures"

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

IMG_SIZE = 64
BATCH_SIZE = 8
EPOCHS = 40
Z_DIM = 64

LR_G = 2e-4
LR_D = 5e-5
BETA1 = 0.5
INSTANCE_NOISE_STD = 0.02
FM_LAMBDA = 2.0
N_CRITIC = 1
G_STEPS = 1
```

D.2 40-epoch GPU upsample training script

```
...

EXPERIMENT_DIR = (
    BASE
    / "experiments"
    / "generation_gan_singleclass_64_cropped_leiden_gpu_e40_upsample"
)
...
```

```
class UnconditionalGenerator(nn.Module):
    def __init__(self, z_dim=Z_DIM, base_channels=32):
        super().__init__()

        self.fc = nn.Sequential(
            nn.Linear(z_dim, base_channels * 16 * 4 * 4, bias=False),
            nn.BatchNorm1d(base_channels * 16 * 4 * 4),
            nn.ReLU(True),
        )

        self.net = nn.Sequential(
            nn.Upsample(scale_factor=2, mode="nearest"),
            nn.Conv2d(base_channels * 16, base_channels * 8, kernel_size=3,
                                                stride=1, padding=1, bias=
                                                False),
            nn.BatchNorm2d(base_channels * 8),
            nn.ReLU(True),

            nn.Upsample(scale_factor=2, mode="nearest"),
            nn.Conv2d(base_channels * 8, base_channels * 4, kernel_size=3,
                                                stride=1, padding=1, bias=
                                                False),
            nn.BatchNorm2d(base_channels * 4),
            nn.ReLU(True),

            nn.Upsample(scale_factor=2, mode="nearest"),
            nn.Conv2d(base_channels * 4, base_channels * 2, kernel_size=3,
                                                stride=1, padding=1, bias=
                                                False),
            nn.BatchNorm2d(base_channels * 2),
            nn.ReLU(True),

            nn.Upsample(scale_factor=2, mode="nearest"),
            nn.Conv2d(base_channels * 2, base_channels, kernel_size=3, stride=1
                                                , padding=1, bias=False),
            nn.BatchNorm2d(base_channels),
            nn.ReLU(True),

            nn.Conv2d(base_channels, 1, kernel_size=3, stride=1, padding=1,
                                                bias=False),
            nn.Tanh(),
        )

    def forward(self, z):
        x = self.fc(z)
        x = x.view(z.size(0), -1, 4, 4)
        return self.net(x)

...

def weights_init(m):
    if isinstance(m, (nn.Conv2d, nn.ConvTranspose2d, nn.Linear)):
        nn.init.normal_(m.weight.data, 0.0, 0.02)
```

```
        if m.bias is not None:
            nn.init.constant_(m.bias.data, 0.0)
    elif isinstance(m, (nn.BatchNorm2d, nn.BatchNorm1d)):
        nn.init.normal_(m.weight.data, 1.0, 0.02)
        nn.init.constant_(m.bias.data, 0.0)

...
```

D.3 Overview of the GPU single-class GAN experiments

Run	Script	Main Changes	Shared Settings	Main Observation	Conclusion
40-epoch GPU run	D.1	Device changed from CPU to GPU; same cropped single-class setup retained.	Cropped negative-only dataset; central labels; image size 64×64, latent dimension $Z = 64$, batch size 8, generator learning rate 2×10^{-4}, discriminator learning rate 5×10^{-5}, feature matching with $\lambda = 2.0$, initial instance-noise standard deviation 0.02, one discriminator step per iteration, and one generator step per iteration.	Compared with the CPU-based 40-epoch run, the model again learnt coarse head-like structure, but the outputs still contained strong artefacts, noisy internal structure, limited diversity, and unrealistic contrast relationships.	GPU acceleration improved the computational efficiency of the run, but did not produce a clear qualitative breakthrough. The remaining limitations therefore appear to be methodological rather than purely computational.
40-epoch GPU run with upsample-based generator	D.2	Generator architecture changed from transposed convolutions to upsampling followed by standard convolutions; same cropped single-class GPU setup retained otherwise.	Cropped negative-only dataset; central labels; image size 64×64, latent dimension $Z = 64$, batch size 8, generator learning rate 2×10^{-4}, discriminator learning rate 5×10^{-5}, feature matching with $\lambda = 2.0$, initial instance-noise standard deviation 0.02, one discriminator step per iteration, and one generator step per iteration.	The modified generator changed the visual artefact pattern, especially in the earliest phase of training. The outputs appeared less uniformly checkerboard-like at first, but later epochs still showed substantial block- and grid-like artefacts, limited diversity, and poor internal anatomical realism.	The architecture change influenced the form of the artefacts, but did not yield a clear qualitative breakthrough. The generated images remained insufficiently realistic for downstream classifier training, suggesting that a broader methodological change is required.

Table 3: Overview of the main exploratory cropped single-class GAN runs performed on the Workstation.

E Two-Class Conditional GAN Experiments on the Workstation

The final generative phase returned to true two-class conditional generation on the full RSNA training split. The first GPU-based cGAN configuration is followed by the adaptations introduced

in the second configuration and a comparative overview of both runs.

This section contains the main code adaptations and configuration excerpts used for the exploratory two-class cGAN runs on GPU. These scripts correspond to the runs summarised in Table 4.

E.1 Key adaptations from `train_cgan.py` to the GPU-based two-class cGAN run

The exploratory two-class cGAN run on the Leiden GPU machine was based on the earlier `train_cgan.py` script, but several important changes were made to turn it from an effectively single-class normal-only experiment into a true two-class conditional GAN trained on the full RSNA training split. The most relevant code adaptations are shown below.

1. Base paths, dataset source, experiment directory, and device The original script used the local MacBook project path, the cropped GAN-specific dataset, and an MPS/CPU device fallback. In the Leiden version, these were replaced by Leiden-local paths, the full processed RSNA dataset, a new experiment directory, and CUDA-based GPU acceleration.

```
BASE = Path("/local/s3209199")

DATA_DIR = BASE / "data" / "processed" / "rsna"
IMG_DIR = DATA_DIR / "images"
SPLIT_CSV = DATA_DIR / "split.csv"

EXPERIMENT_DIR = BASE / "experiments" / "generation_cgan_64_rsna_leiden_gpu_e40
                                        "
CHECKPOINT_DIR = EXPERIMENT_DIR / "checkpoints"
METRICS_DIR = EXPERIMENT_DIR / "metrics"
FIGURES_DIR = EXPERIMENT_DIR / "figures"

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

2. True two-class dataset instead of hardcoded single-class labels In the original `train_cgan.py`, the dataset always returned label 0, which made the experiment effectively single-class despite using conditional model components. In the adapted Leiden version, the dataset was changed to read the true binary `any_ich` labels from the RSNA split file, while restricting training to the predefined train split only.

```
class RSNACGANTrainDataset(Dataset):
    def __init__(self, df):
        self.df = df.reset_index(drop=True)

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        image_id = row["image_id"]
        label = int(row["any_ich"])
```

```
        x = np.load(IMG_DIR / f"{image_id}.npy").astype(np.float32)
        x = torch.tensor(x, dtype=torch.float32).unsqueeze(0)
        x = F.interpolate(
            x.unsqueeze(0),
            size=(IMG_SIZE, IMG_SIZE),
            mode="bilinear",
            align_corners=False,
        ).squeeze(0)

        x = (x - 0.5) / 0.5
        y = torch.tensor(label, dtype=torch.long)
        return x, y


def make_loader():
    df = pd.read_csv(SPLIT_CSV)
    df = df[df["split"] == "train"].copy()
    return DataLoader(
        RSNACGANTrainDataset(df),
        batch_size=BATCH_SIZE,
        shuffle=True,
        drop_last=True,
    )
```

3. Scheduled instance noise instead of fixed instance noise The original script used a fixed instance-noise standard deviation throughout training. In the adapted version, instance noise was scheduled to decay linearly over epochs, so that early training remained smoother while later epochs used less artificial perturbation.

```
def add_instance_noise(x, std=0.03):
    if std <= 0:
        return x
    noise = torch.randn_like(x) * std
    return torch.clamp(x + noise, -1.0, 1.0)


def schedule_instance_noise(epoch):
    return INSTANCE_NOISE_STD * max(0.0, 1.0 - (epoch - 1) / EPOCHS)
```

4. Fixed-noise monitoring for both classes To monitor both classes consistently during training, the fixed-noise sampling function was changed so that the saved sample grid always contained eight non-ICH and eight ICH-conditioned generations.

```
def make_fixed_noise():
    fixed_z = torch.randn(16, Z_DIM, device=DEVICE)
    fixed_y = torch.tensor([0] * 8 + [1] * 8, dtype=torch.long, device=DEVICE)
    return fixed_z, fixed_y
```

5. Real-versus-fake visualisation adapted to class-conditioned generation The visualisation function was adjusted so that fake images were generated with the same labels as the selected real images in the comparison grid.

```
def save_real_vs_fake_grid(G, real_imgs, real_labels, epoch):
    G.eval()
    with torch.no_grad():
        n = min(8, real_imgs.size(0))
        z = torch.randn(n, Z_DIM, device=DEVICE)
        fake = G(z, real_labels[:n])

        real_grid = (real_imgs[:n] + 1.0) / 2.0
        fake_grid = (fake + 1.0) / 2.0
        grid = torch.cat([real_grid, fake_grid], dim=0)

        out_path = FIGURES_DIR / f"real_vs_fake_epoch_{epoch:03d}.png"
        vutils.save_image(grid, out_path, nrow=n, normalize=False)
        print(f"Saved real-vs-fake grid to {out_path}")
    G.train()
```

6. Training loop with scheduled noise Within the training loop, the scheduled instance-noise level was updated each epoch and applied consistently to both real and generated images before they were passed to the discriminator.

```
for epoch in range(1, EPOCHS + 1):
    g_losses = []
    d_losses = []
    last_real_batch = None
    last_label_batch = None
    noise_std = schedule_instance_noise(epoch)

    for real_imgs, labels in loader:
        real_imgs = real_imgs.to(DEVICE)
        labels = labels.to(DEVICE)

        last_real_batch = real_imgs
        last_label_batch = labels

        batch_size = real_imgs.size(0)
        real_targets = torch.full((batch_size, 1), LABEL_SMOOTH_REAL, device=
                                                    DEVICE)
        fake_targets = torch.zeros(batch_size, 1, device=DEVICE)

        for _ in range(N_CRITIC):
            z = torch.randn(batch_size, Z_DIM, device=DEVICE)
            fake_imgs = G(z, labels)

            real_noisy = add_instance_noise(real_imgs, std=noise_std)
            fake_noisy = add_instance_noise(fake_imgs.detach(), std=noise_std)

            real_logits = D(real_noisy, labels)
            fake_logits = D(fake_noisy, labels)
```

```
            d_loss_real = criterion(real_logits, real_targets)
            d_loss_fake = criterion(fake_logits, fake_targets)
            d_loss = 0.5 * (d_loss_real + d_loss_fake)

            opt_D.zero_grad()
            d_loss.backward()
            opt_D.step()

        for _ in range(G_STEPS):
            z = torch.randn(batch_size, Z_DIM, device=DEVICE)
            gen_imgs = G(z, labels)
            gen_noisy = add_instance_noise(gen_imgs, std=noise_std)
            gen_logits = D(gen_noisy, labels)

            g_loss = criterion(gen_logits, real_targets)

            opt_G.zero_grad()
            g_loss.backward()
            opt_G.step()
```

7. Updated configuration logging The final configuration output was also updated to reflect that the experiment used the full processed RSNA dataset and only the predefined training split.

```
with open(METRICS_DIR / "config.json", "w") as f:
    json.dump({
        "data_dir": str(DATA_DIR),
        "split_csv": str(SPLIT_CSV),
        "train_split_only": True,
        "img_size": IMG_SIZE,
        "batch_size": BATCH_SIZE,
        "epochs": EPOCHS,
        "z_dim": Z_DIM,
        "num_classes": NUM_CLASSES,
        "lr_g": LR_G,
        "lr_d": LR_D,
        "beta1": BETA1,
        "label_smooth_real": LABEL_SMOOTH_REAL,
        "instance_noise_std": INSTANCE_NOISE_STD,
        "g_steps": G_STEPS,
        "n_critic": N_CRITIC,
        "dataset": "rsna processed full set (train split only)",
    }, f, indent=2)
```

E.2 Key adaptations from the first two-class cGAN run to the second

The second exploratory two-class cGAN run was based directly on the first GPU-based two-class cGAN run, but several hyperparameters were adjusted to test whether a slightly milder adversarial setup would improve qualitative image generation. The intention was not to introduce a fundamentally new architecture, but to examine whether the disappointing qualitative results of the first run were partly caused by an overly constrained or overly balanced training configuration.

The main changes are shown below.

1. New experiment directory The second run used a separate experiment directory so that all outputs, checkpoints, and metrics would be stored independently from the first cGAN run.

```
EXPERIMENT_DIR = BASE / "experiments" / "
                                        generation_cgan_64_rsna_leiden_gpu_e40_v2
                                        "
CHECKPOINT_DIR = EXPERIMENT_DIR / "checkpoints"
METRICS_DIR = EXPERIMENT_DIR / "metrics"
FIGURES_DIR = EXPERIMENT_DIR / "figures"
```

2. Reduced latent dimension The latent dimension was reduced from $Z = 128$ to $Z = 64$. This was intended to simplify the generation problem and test whether a smaller latent space would make it easier for the generator to learn more coherent global structure.

```
Z_DIM = 64
```

3. Removal of real-label smoothing In the first run, the real targets for the discriminator were smoothed to 0.9. In the second run, this smoothing was removed and the real targets were set to 1.0. This change was introduced to test whether a slightly stronger and more direct learning signal would improve generator training.

```
LABEL_SMOOTH_REAL = 1.0
```

4. Reduced initial instance noise The initial instance-noise standard deviation was reduced from 0.03 to 0.02. This was intended to retain some early stabilisation while making the training signal slightly less blurred.

```
INSTANCE_NOISE_STD = 0.02
```

5. Fewer generator updates per batch The number of generator updates per batch was reduced from 2 to 1. This was done to make the generator-discriminator update schedule more conservative and to test whether the earlier setup had encouraged the generator to settle too quickly into a flat equilibrium.

```
G_STEPS = 1
```

6. Slightly increased discriminator dropout The discriminator dropout was increased from 0.30 to 0.35. This adjustment was intended to regularise the discriminator slightly more strongly and reduce the chance that it would become too confident too early.

```
class ConditionalDiscriminator(nn.Module):
    def __init__(self, num_classes=NUM_CLASSES, img_size=IMG_SIZE,
                                                base_channels=24, dropout=0.35):
        super().__init__()
        ...
```

E.3 Overview of the two-class cGAN experiments

Run	Script	Main Changes	Shared Settings	Main Observation	Conclusion
40-epoch GPU cGAN run	E.1	Switched from single-class GAN to a two-class conditional GAN trained on the full RSNA train split.	Image size 64×64, batch size 16, $Z = 128$, 40 epochs, $lr_G = 2 \times 10^{-4}$, $lr_D = 5 \times 10^{-5}$, label smoothing, instance noise.	Training was numerically stable and produced coarse head-like structures, but the generated images remained noisy and anatomically unrealistic; class conditioning did not yield a clear qualitative separation.	Conditioning improved stability, but not enough to make the generated images suitable for downstream classifier training.
40-epoch GPU cGAN run (v2)	E.2	Reduced latent dimension from 128 to 64, removed real-label smoothing, reduced instance noise, reduced generator steps per batch, and increased discriminator dropout slightly.	Image size 64×64, batch size 16, 40 epochs, $lr_G = 2 \times 10^{-4}$, $lr_D = 5 \times 10^{-5}$, two-class conditioning, full RSNA train split.	Training was numerically very stable and converged rapidly toward a narrow equilibrium, but the generated images remained repetitive, noisy, and anatomically unrealistic.	A milder cGAN setup did not improve qualitative realism. The result strengthens the conclusion that the current cGAN configuration is not sufficient for generating useful synthetic CT data.

Table 4: Overview of the exploratory two-class cGAN runs performed on GPU on the Workstation.