Hogeschool Rotterdam

Idiap Research Institute

Beyond the Last-Frame

Temporal Deep Learning for Automated Grading of Retinal Inflammation in Fluorescein Angiography

by

Tymo van Rijn

Student number 1057297

A graduation portfolio

Supervised by Dr. André Anjos, Dr. Oscar Jiménez del Toro

MedAI Group, Rue Marconi 19, Martigny, Switzerland

21st June 2026

Foreword and Reader's Guide

This portfolio is the final deliverable of my graduation internship at **Idiap Research Institute** in Martigny, Switzerland, where I spent 6 months with the MedAI Group. It concludes my Bachelor's programme at Hogeschool Rotterdam.

Doctors use a procedure called fluorescein angiography to "film" what happens inside a patient's eye after a fluorescent dye is injected into the bloodstream. Over several minutes the dye lights up and moves through the retina, and the way it spreads, brightens, or stays put is what a clinician uses to diagnose disease. At Idiap, an AI system already exists that can grade these examinations – but only looks at one still image and throws away the time information. My project was to extend that system so it can read the **whole sequence** of frames, and to find out whether doing so actually leads to better grading.

My personal contribution to this project was the following; I designed and built the temporal pipeline end-to-end: I recovered the missing timing information from the raw frames using an OCR-based pipeline, I worked out empirically how frames should be grouped in time, I trained and evaluated sequence models on top of the existing images backbone, and I integrated the results back into the group's research framework with proper tests and continuous integration. The final results improve the multiclass classification score (ROC-AUC) from 0.82 to 0.87, with the largest gains on the pathologies where time behavior matters most.

This work sits at the intersection of clinical eye imaging and deep learning, so some chapters are necessarily technical. I have tried to introduce each concept the first time it appears, and chapter 3 gives a plain-language introduction to both the medical side and the AI models I build on.

A 15-minute spoken overview is available as a recording of my Idiap TAM presentation: TAM Presentation.

Thank you for taking the time to read this portfolio.

Tymo van Rijn

Contents

Abstract

Fluorescein angiography (FA) is a dynamic imaging procedure in which clinicians read pathologies by how fluorescence *behaves over time*: leakage spreads, staining brightens in place, window defects stay geographically stable. Yet the current MedNet grading pipeline at Idiap collapses each examination to a single last frame, achieving a macro ROC-AUC of 0.82 on multiclass HyperF_Type classification while discarding the temporal evolution that defines several of the pathologies it grades.

This project engineered a temporal pipeline for variable-length FA examinations on the public 3rd Aptos dataset [1]. Missing frame timing was recovered via a benchmark-driven OCR pipeline (Gemini 2.5 Flash [2], validated on 150 hand-checked frames), and data-driven LDA probing set phase boundaries at 103 s / 518 s, outperforming the textbook convention on every separability metric. Twelve frames (four per phase) are passed through a RETFound-Green ViT [3, 4] and a two-layer GRU [5], with the existing classification head kept unchanged so that any gains are attributable to the temporal block.

With the backbone frozen, no temporal aggregation reliably beats a single well-chosen frame — a genuine negative result. With backbone and GRU fine-tuned jointly, temporal modelling improves every graded task: macro HyperF_Type AUC rises from 0.82 to 0.87, with the largest gains on Pooling ($0.76 \rightarrow 0.83$) and Window Defect ($0.78 \rightarrow 0.85$) — exactly the pathologies defined by behaviour over time. The win requires co-adapting backbone and sequence; sequencing frozen frames does not suffice. External validation on the HOJG clinical dataset is prepared and scoped for the MedAI Group to run next.

Glossary

The following terms appear throughout this portfolio. Where a concept is introduced and explained in more depth inside a specific chapter, that chapter is noted. Medical and clinical concepts — including fluorescein angiography, the four hyperfluorescence types, and the two datasets — are introduced in Chapter 3.

AUC / ROC-AUC
: Area Under the Receiver Operating Characteristic Curve. A number between 0 and 1 that measures how reliably a model ranks diseased cases above healthy ones, regardless of which decision threshold is applied. A value of 1.0 is perfect; 0.5 is equivalent to random guessing. *Macro* AUC averages the score equally across all classes rather than weighting by class size, which prevents large classes from dominating the number.

Backbone
: The main image-processing network in a deep learning pipeline. It converts a raw image into a compact numerical summary (an *embedding*) that captures the visual content in a form the rest of the model can use. In this project the backbone is RETFound-Green, introduced in Chapter 3.

CI (Continuous Integration)
: An automated system that runs a project's tests every time code is changed, preventing regressions from silently entering the codebase. Used here via GitLab CI to guard the temporal pipeline. Described in Chapter 5.

Class imbalance
: A situation in which some categories in a dataset are substantially more frequent than others. In this project, certain hyperfluorescence types are much rarer than others. If ignored, a model can appear to perform well simply by defaulting to the majority class, while being useless on the rare, clinically important ones.

Embedding
: A fixed-length vector of numbers that represents an image after it has been processed by a backbone. The RETFound-Green backbone produces a 384-dimensional embedding per frame; these vectors are the inputs to the GRU sequence model.

F1-score
: A metric that balances precision (how often a positive prediction is correct) and recall (how often a true positive case is actually found). Reported alongside AUC because AUC alone can mask poor performance on rare disease types in an imbalanced dataset.

Fine-tuning Continuing training of a pre-trained model on a new dataset so that its weights adapt to the specific task. Contrasted throughout this portfolio with *frozen* evaluation, where backbone weights are held fixed.

Foundation model A large neural network pre-trained on a massive dataset — often without human-labelled data — that can be adapted to many downstream tasks. Using a foundation model avoids training from scratch on small medical datasets, which is prone to overfitting. RETFound-Green is the foundation model used in this project. Introduced in Chapter 3.

Frozen (weights) A model whose parameters are held fixed and not updated during training. The frozen experiments in Chapters 6 and 4 ensure that any observed performance change comes purely from the sequence architecture, not from the backbone learning anything new.

GRU (Gated Recurrent Unit)
A type of recurrent neural network designed to process ordered sequences. It reads inputs one step at a time — here, one frame embedding per step — and maintains a running summary (the *hidden state*) updated at each step. The final hidden state is used for classification. Chosen over an LSTM for efficiency and over a Transformer because sequences here are short (12 frames after binning). Discussed in Chapter 8.

LDA (Linear Discriminant Analysis)
A statistical method that finds the single projection direction that best separates two or more classes. Used in this project as a cheap *linear probe*: fitted on top of frozen backbone embeddings to compare temporal configurations without running expensive full training. Described in Chapter 6.

Linear probe A simple classifier — typically a single linear layer or LDA — fitted on top of a frozen backbone's embeddings. Because the backbone is not updated, the probe score reflects what information is already present in the representation rather than what could be learned. Used throughout the Analysis chapter to compare configurations cheaply.

LSTM (Long Short-Term Memory)
A recurrent neural network architecture, and the direct predecessor of the GRU. Considered as an alternative temporal model and used in early experiments; replaced by the GRU on the recommendation of supervisors for better efficiency at the sequence lengths used here.

OCR (Optical Character Recognition)
Software that reads and extracts text from images. Used in this project to recover elapsed-time timestamps burned into the corner of

Aptos frames, since the dataset does not include structured timing metadata. Described in Chapter 6.

Overfitting When a model learns the training data too closely — including its noise — and performs well on training examples but poorly on new, unseen cases. A recurring concern given the relatively small dataset size, addressed through class-weighted training, seed-averaged reporting, and evaluation on a held-out test set.

PCA (Principal Component Analysis)
A technique that reduces the number of dimensions in a dataset while retaining as much variance as possible, sometimes used as a denoising step. Tested in a controlled study as a pre-processing step before sequence modelling; found not to improve performance and excluded from the final pipeline. Described in Chapter 6.

Temporal binning The process of dividing a variable-length frame sequence into fixed time windows (phases) and selecting representative frames from each. Used here to convert examinations of varying lengths into a fixed 12-frame input for the GRU. Described in Chapters 6 and 8.

Transformer A neural network architecture based on *attention*: rather than processing a sequence step by step, it computes relationships between all positions simultaneously. State-of-the-art for many tasks but requires more data and compute than a GRU for short sequences. Considered as an alternative to the GRU and discussed in Chapter 8.

ViT (Vision Transformer)
A Transformer architecture adapted for images. It divides an image into small patches, treats each patch as a token, and applies attention across all patches to build a global representation. The RETFound-Green backbone used in this project is a ViT-Small. Introduced in Chapter 3.

Chapter 1

Introduction and Project Context

The Idiap Research Institute, located in Martigny, Switzerland, is a premier non-profit organisation specialising in artificial intelligence, machine learning, and computer vision. Within Idiap, the MedAI group develops advanced algorithmic methodologies for medical imaging in tight collaboration with clinical partners, such as the Hôpital Ophtalmique Jules-Gonin (HOJG) in Lausanne.

Within this high-stakes research environment, the project tackles a critical limitation in current ophthalmic diagnostic pipelines. Fluorescein angiography (FA) is a dynamic, time-based imaging procedure used to track blood flow and detect vascular leakage in the retina. However, the current state-of-the-art deep learning pipelines utilised by the MedAI group treat FA examinations as isolated, static images. By relying on single-frame Vision Transformer (ViT) backbones [6], the existing architecture discards the temporal evolution of the fluorescent dye — information that is clinically indispensable.

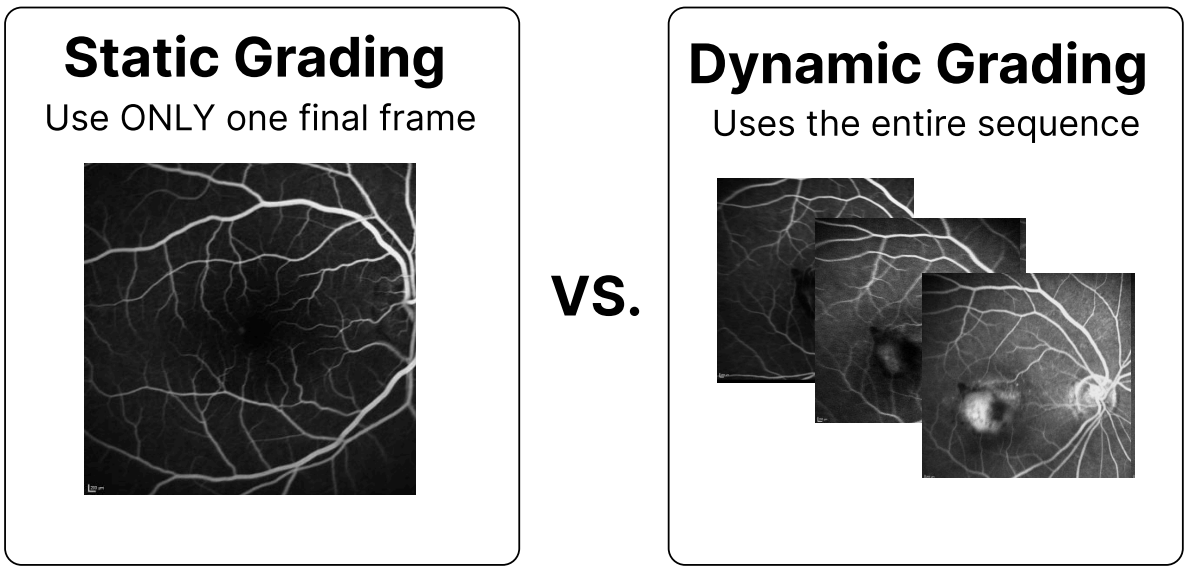


Figure 1.1: A conceptual overview contrasting the single-frame 'Static Grading' baseline against the proposed sequence-based 'Dynamic Grading' approach.

The overarching objective of this project was to engineer a temporal data representation and implement a sequential deep learning architecture capable of processing variable-length FA examinations, thereby improving diagnostic performance, robustness, and algorithmic interpretability.

Chapter 2

Problem Statement and Research Questions

2.1 Problem Statement

Current Fluorescein Angiography classification pipelines insufficiently utilise temporal information present in FA examinations, potentially limiting diagnostic performance, robustness, and interpretability.

2.2 Sub-Questions

The chapters that follow are organised around answering these questions.

Q1 Does the available benchmark data — labels, class balance, frame-level dynamics, and timing metadata — support temporal modelling, and what limitations follow from how time is recorded?

Q2 When metadata lacks reliable timing, can we recover usable elapsed time and decide which frames are safe to use?

Q3 How should we group frames in time (binning, aggregation), and can a lightweight probe thin out options before expensive training?

Q4 Does unsupervised dimensionality reduction (PCA) of frozen embeddings improve sequence classification, or should embeddings stay at backbone dimension unless a supervised objective is introduced?

Q5 Does model performance transfer from Aptos to the external hospital (HOJG) dataset?

Chapter 3

Clinical and Technical Background

To comprehend the architectural decisions made during this project, it is essential to understand both the clinical realities of retinal imaging and the foundations of the deployed machine learning models.

3.1 Uveitis and Fluorescein Angiography

Fluorescein angiography (FA) is easiest to picture as a short film after a dye injection: the camera keeps taking images while the dye moves through the choroid and retinal vessels, then washes through the circulation again over several minutes. A clinician is not looking for a single "pretty frame"; they are watching how brightness moves and whether it stays inside vessels or escapes into tissue.

Clinicians often describe this timeline in phases, because the same fundus image can mean different things depending on how early or late it is — that goes to show that temporality has a role in the classification task. In uveitis care, FA is often used to evaluate blood–retinal barrier integrity and patterns of vascular leakage; the rest of this section explains what clinicians are looking at on the sequence when they interpret those patterns.

During the sequence, clinicians separate several hyperfluorescence patterns by how they behave over time:

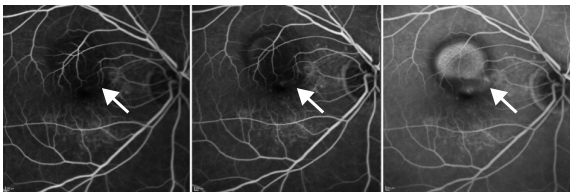
Leakage Dye escapes from vessels; the bright area tends to grow and blur over time.

Pooling Dye collects in a defined space; it may brighten and fill a cavity, often with a sharper border than leakage, and the dynamics differ from simple staining.

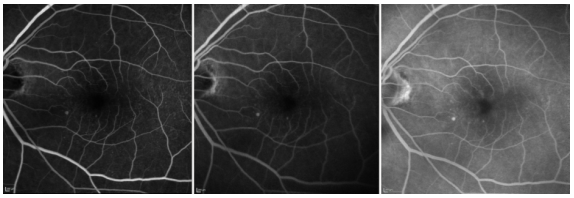
Staining Tissue takes up dye and gets brighter, but the lesion does not behave like expanding leakage — often described as increasing brightness without the same spreading or leaking pattern.

Window defect A patch where the normal masking pigment is missing, so the brighter underlying choroidal signal appears early and stays geographically stable across the sequence.

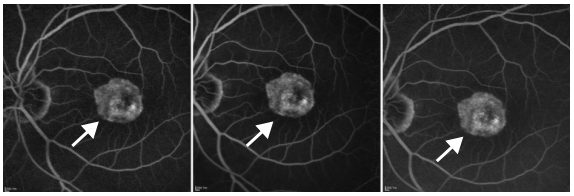
(a) Leakage: the bright area grows and blurs across the sequence.



(b) Pooling: dye fills a defined cavity with a sharper, stable border.



(c) Staining: brightness accumulates without the spatial expansion of leakage.



(d) Window defect: choroidal signal appears early and stays geographically stable.

Figure 3.1: The four hyperfluorescence patterns classified in this project, each shown as a short sequence of FA frames from the Aptos dataset. The key diagnostic information is temporal: the same appearance in a single frame can correspond to different pathologies depending on how it evolves across the sequence.

3.2 Dataset Context

Table 3.1: Complementary role of the two datasets used in this project.

Dataset	**Clinical focus**	**Role in this project**
HOJG	Uveitis-related retinal inflammation	Main clinical target dataset for grading inflammatory signs
3rd Aptos (AngioReport)	Broader angiographic abnormalities and fluorescence patterns	Public dataset for broader FA analysis and temporal exploration

The project utilises two distinct datasets to evaluate the temporal algorithms, bridging the gap between public benchmarks and private clinical applications.

3rd Aptos Competition Dataset [1] A public dataset featuring tens of thousands of angiographic images categorised into various conditions. For this project, it is used primarily to classify hyperfluorescence types: Leakage, Pooling, Staining, Window Defect, and None.

HOJG Dataset [7] A highly curated, private clinical dataset containing 543 patients and 1,042 eyes provided by the Hôpital Ophtalmique Jules-Gonin. It is annotated using the rigorous ASUWOG (Angiography Scoring for Uveitis Working Group) grading system [8] to quantify inflammatory signs such as macular oedema, optic disc hyperfluorescence, and capillary leakage.

3.3 Foundation Models

Training deep neural networks from scratch on specialised medical imagery is notoriously prone to overfitting due to severe data scarcity. To circumvent this, the MedAI group leverages

Foundation Models — large-scale neural networks pre-trained on massive datasets via self-supervised learning [3].

Specifically, this project builds upon RETFound and RETFound-Green [4]. RETFound-Green is a compact Vision Transformer (ViT-Small) pre-trained on 75,000 Colour Fundus Photography (CFP) images using a novel Token Reconstruction pretext task. When an FA image is passed through RETFound-Green, the network divides the image into non-overlapping patches, flattens them, and processes them through multi-head self-attention layers to output a highly compressed, 384-dimensional feature vector that encapsulates the medical semantics of that single frame.

The baseline architecture established by previous work in the group (specifically the thesis of Roberto Pulvirenti [9]) achieved strong classification results using only the final frame of an examination. However, predicting dynamic pathologies such as progressive leakage from a single static vector is inherently limited. The technical mandate of this project was to sequence these 384-dimensional vectors chronologically to mathematically represent the transit of the dye over time.

3.4 The Software Environment: MedNet and the Idiap Grid

Before describing the work itself, it is worth introducing the software environment it was built in, because nearly every design and engineering decision in this project was shaped by it. MedNet is the in-house deep-learning framework that the MedAI group uses for its medical-imaging research [10].

Built on top of PyTorch, it provides a shared, standardised structure for training models, running evaluations, and managing experiment outputs, so that work done by one researcher can be inspected, reproduced, and built upon by another. It is maintained by the MedAI group — especially by Dr. André Anjos — and the group standardises on it for a simple reason: a common framework keeps experiments comparable and reviewable, and it spares each new researcher from rebuilding training and evaluation infrastructure from scratch. Results produced through MedNet are tied to a shared evaluation pipeline rather than to one person's local scripts.

The second piece of environment is the shared GPU grid inside Idiap. Compute is a finite, shared resource: jobs are scheduled alongside those of every other researcher, so an experiment that is wasteful with memory or training time is not just slow for me — it is a cost to the whole group. This constraint reappears throughout the later chapters as a hard requirement and gate on what counts as a realistic proposal.

3.5 Data Governance and the Public/Private Split

The two datasets in this project are not interchangeable. The difference between them is not only clinical; it is a matter of data governance that shaped how each one could be used.

The HOJG dataset is identifiable clinical patient data (543 patients from the Hôpital Ophtalmique Jules-Gonin) and is governed by Swiss data-protection law, the GDPR, and the agreements between Idiap and the hospital. In practical terms this means it cannot leave Idiap's infrastructure and cannot be redistributed; it is also why certain things in this portfolio, such as the raw frames, are pointed to rather than shipped in the deliverables folder.

The Aptos dataset, however, is public. That is precisely why it is the dataset used for open exploration and for anything that touches external tools or services: experimenting on public data carries none of the disclosure risk that the same experiment would carry on patient

imagery. Focusing on Aptos also makes results reproducible and positions the work better for eventual publication as a paper or journal article.

This public/private split was therefore treated as a boundary rather than an accident of availability. The public data was used for almost every kind of experiment; the private clinical data was reserved for external validation of the model.

3.6 Single-Frame Baseline

This project did not start from a blank page. Previous research in the group (specifically the thesis of Roberto Pulvirenti [9]) had already established a strong reference pipeline for the same grading task, and it is the result this work has to be measured against.

That baseline used only the last frame of each examination for training, collapsing the full angiographic sequence into a single static image. On the multiclass HyperF_Type classification task it reached a ROC-AUC of 0.82 — a genuinely strong score and an important detail for framing this project honestly: "temporal" is not competing against a weak straw-man; it has to earn its place against an already-capable single-frame model.

This is what sets the bar for the Results chapter: any temporal approach must justify its additional complexity and compute by improving on that ROC-AUC 0.82 under the same constraints, not in the abstract.

Results

This project extended the existing FA grading pipeline with a temporal component, integrated it into the MedNet framework, and showed that jointly training the image model and the sequence model improves grading performance from 0.82 to 0.87 AUC on the main classification task. The five research questions that structured the work are addressed as follows.

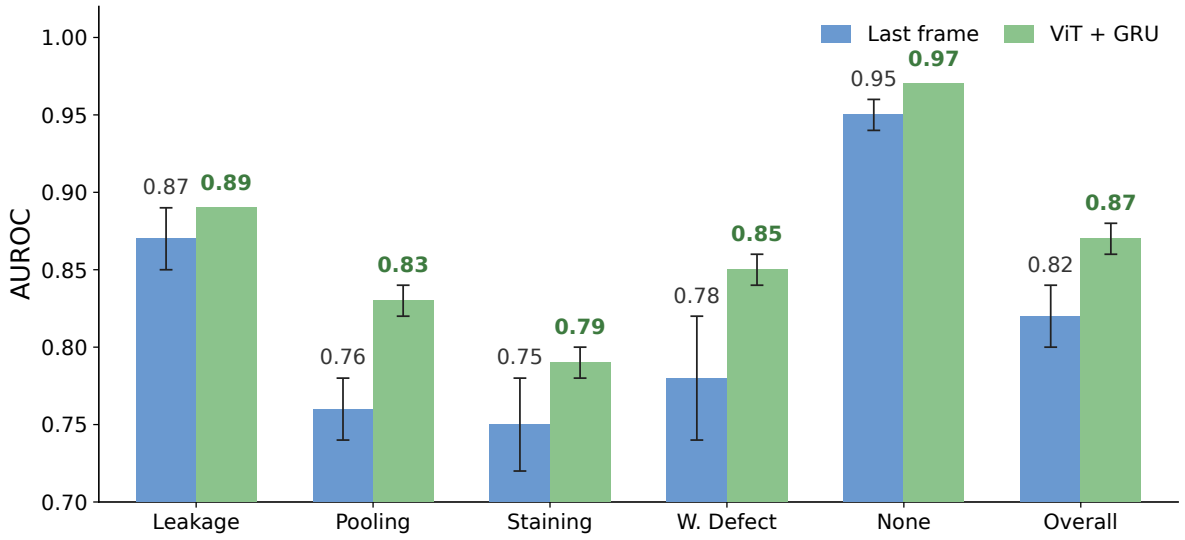


Figure 4.1: AUC comparison between the last-frame fine-tune baseline (blue) and the joint ViT + GRU fine-tune (green) across all graded tasks, mean over five seeds. The temporal model improves every task, with the largest gains on Window Defect and Pooling.

Q1 — Does the data support temporal modelling? Yes, with conditions. Images within an examination show genuine visual progression, and class labels are informative. However, timing metadata is unreliable without additional processing, and class imbalance requires weighted training and per-class evaluation. These findings are addressed in the *Analysis* competency (Chapter 6, data analysis and timestamp extraction sections) and shaped the evaluation strategy throughout.

Q2 — Can missing timing be recovered reliably? Yes. A benchmark-driven OCR pipeline was built and validated against 150 hand-checked frames, selecting the best-performing tool with 96% accuracy on FA frames. This produced 32,834 validated timestamps and a frame- eligibility filter as a side-effect. Addressed in *Analysis* (Chapter 8, OCR benchmark) and *Realisation* (Chapter 9, data pipeline).

Q3 — How should frames be grouped in time? A data-driven approach using a lightweight statistical probe outperformed the standard clinical convention on every metric, setting phase boundaries at 103 s / 518 s. The probe was also used to compare backbone and binning configurations cheaply before committing to expensive training. Addressed in *Analysis*

(Chapter 6, phase analysis and Temporal Separability Benchmark) and *Design* (Chapter 8, binning strategy).

Q4 — Does dimensionality reduction before sequence modelling help? No. A controlled study found no improvement across any tested setting. This negative result is documented and acted upon as a concrete advisory output: the approach was excluded from the pipeline. Addressed in *Analysis* (Chapter 6, PCA study) and *Advise* (Chapter 7).

Q5 — Does performance transfer to the hospital dataset? This question could not be answered within the internship window due to data-governance timelines. The pipeline is designed and ready for this validation step, which is set up for the MedAI group to run next. The scoping decision and its reasoning are addressed in *Analysis* (Chapter 6) and *Advise* (Chapter 7, next steps).

Chapter 5

Professional Skills & Manage and Control

5.1 Professional Skills

5.1.1 Communication

During my internship at Idiap, I developed my professional communication skills in a multidisciplinary research environment in which technical researchers and clinical stakeholders contributed to the same overall objective. This required me to adapt both the content and the form of my communication to the audience. I had bi-weekly Zoom meetings with Jules-Gonin hospital and other non-technical stakeholders. When communicating with these less-technical people, I had to explain machine-learning concepts and findings in clear, accessible language. In discussions with medical-AI researchers, I needed to communicate at a more technical level and justify methodological choices more precisely.

An important part of this development was learning to communicate progress accurately, concisely, and at the right moment. This was necessary to keep my work aligned with the expectations of the project and to avoid spending time on directions that were not sufficiently relevant or well supported. I communicated through several channels, including Mattermost, weekly meetings, and research presentations. During these weekly meetings, I presented the work completed, the status of the project, and the next steps. These presentations correspond to the slide files in `02_ANALYSIS/progress_slides/`. Examples of written and visual communication with the team are shown in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 1, Fig. 3, Fig. 4 and Fig. 8.

A clear learning point in this process was that effective communication is not only about being complete, but also about creating the right emphasis. By actively asking for feedback after presentations, I learned that my explanations were generally clear, but that I sometimes tried to communicate too much information at once. As a result, the main message was not always immediately visible. I responded to this feedback by adapting the structure of my presentations: I started stating the central takeaway earlier, reduced unnecessary detail on the first slides, and used the remaining slides to support the main point more logically. This development can be illustrated by comparing an early presentation with a later one, for example `02_ANALYSIS/progress_slides/week01-progress-slides.pdf` compared to `02_ANALYSIS/progress_slides/week05-06-progress-slides.pdf`.

A major milestone in this development was presenting my findings at the Idiap Tuesday Afternoon Meeting (TAM). Unlike the weekly MedAI meetings, the TAM is an institute-wide forum, which required me to adapt my narrative for a broader audience of researchers who were not intimately familiar with ophthalmic imaging or the specific RETFound-Green architecture. I had to distil the clinical problem and my GRU-based solution into a clear, 15-minute overarching story, applying the exact lessons I learned about upfront emphasis and reducing unnecessary detail.

To provide examiners with direct evidence of my oral presentation skills and to offer a visual summary of this project, a recording of this presentation is available at: https://youtu.be/

9NKKttZj698. Watching this video is recommended as a high-level overview of the work before diving into the technical chapters.

A further milestone was the acceptance of a submitted abstract to SAIMI 2026 (Swiss AI in Medicine Initiative conference, 21 June). This required translating the full research arc — clinical problem, temporal methodology, and experimental findings — into a poster format accessible to a multidisciplinary medical AI audience. Designing this poster forced a discipline of visual communication: every choice of layout, metric, and graphic had to stand alone without a presenter narrating it. The acceptance demonstrates that the communication of this project's findings met the standards of an external scientific peer-review process. The latest draft of this poster is available in `01_Professional_Skills_&_Manage_Control/saimi_poster_draft.pdf`.

5.1.2 Collaboration

In addition to communication, I strengthened my collaboration skills by working proactively within an interdisciplinary team. My role required continuous alignment with supervisors and other colleagues, both to ensure technical correctness and to keep the work relevant to the broader research objective. Rather than working in isolation, I regularly asked questions, shared intermediate findings, and discussed uncertainties before continuing with major implementation or experimental decisions. This helped me maintain alignment with the team and reduced the risk of developing solutions that were technically interesting but insufficiently useful for the project.

This collaborative attitude was especially important because the project sat at the intersection of software development, machine learning, and clinical application. Effective collaboration therefore depended not only on technical competence, but also on recognising my role within the project, taking ownership of appropriate tasks, and involving others at the right moments. By doing so, I contributed more actively to the shared progress of the team and learned how trust and clarity are built in a research setting. Evidence of this proactive collaboration is shown in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 1 and Fig. 2.

This collaborative approach was pivotal in helping me identify design challenges and pivot towards more effective solutions. For example, I initially developed an LSTM pipeline [11] to process frame sequences. During a weekly presentation, I shared this design with the MedAI team; André pointed out that a GRU architecture [5] would be more efficient for our requirements. Rather than treating this as a setback, I utilised this feedback as a constructive opportunity to optimise the design. I conducted further research into GRUs, validated the advantages of the switch, and successfully refactored the pipeline.

This cycle — identifying a technical bottleneck, engaging in open dialogue with team members, and proactively pivoting to a better approach — demonstrates my commitment to both constructive collaboration and rigorous problem-solving.

Overall, this internship strengthened my professional behaviour in two important ways. First, I became more aware of how communication must be adapted to different stakeholders to achieve the desired impact. Second, I learned that strong collaboration depends on initiative, alignment, and openness to feedback, especially in an interdisciplinary research environment.

5.2 Manage & Control

5.2.1 Structure

Manage & Control in this project was not limited to making a schedule; it concerned maintaining a structured, traceable, and quality-oriented software and research process throughout the internship. While I adhered to the company's established standards for research and software development, I also took the initiative to implement additional personal measures to ensure my contribution remained rigorous.

A first important aspect of this was planning and replanning. In the first few weeks, I maintained a daily logbook in which I also wrote weekly reflections, found in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/PROCESS-LOGBOOK.PDF`. This logbook was solely for my own use. At the start of each day, or the evening before, I defined concrete goals for the next working period. At the end of the day, I documented the work completed, the results obtained, and the logical next steps. This gave me a clear overview of progress and helped me work deliberately rather than reactively. In addition, the weekly reflections allowed me to evaluate what went well, which problems occurred, what feedback I had received, and which action points I wanted to carry into the following week. In this way, the logbook functioned not only as a record of work, but also as an active control instrument for planning, monitoring, and adjustment. Evidence of this process is shown in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 5, Fig. 6, and Fig. 10.

At a certain point, the logbook did not feel necessary anymore, since I thought I could maintain the structure in my own head. This was far from true. I returned to writing my goals for the week and day on a piece of paper kept on my desk, which made it easier to refer to them and check items off.

5.2.2 Development-Street

A second aspect of Manage & Control was the structured use of GitLab for version control and repository management. I used GitLab as the central environment for managing project code and kept the repositories actively up to date. New features, scripts, or experimental changes were developed in separate branches before being integrated into the main branch.

This applied, for example, to components such as embedding-extraction scripts, temporal binning logic, and benchmark-related experiments. Working in this way ensured that existing code remained stable while new functionality could be developed and tested in isolation. It also made the development process more traceable, because results could be linked back to specific code versions and intermediate decisions remained visible in the repository history.

This branch-based workflow supported both control and collaboration, since changes could be reviewed and discussed on the basis of a clear development history rather than informal local modifications. This is illustrated with a GitLab repository overview in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 12 and a branch and commit-history example in Fig. 13.

On top of this branch-based workflow, the temporality branch contributes a CI job (`tests-temporal` in `.gitlab-ci.yml`) that runs the temporal unit tests automatically whenever changes touch the temporal source paths, with a `when: manual` fallback for on-demand runs. Execution uses `pixi run -e test-ci-alternative` — the same environment as local development and the reproduction repository, so there is no environment drift between where code is developed and where CI verifies it. This converts the two-layer test strategy from a one-time validation into a regression guard that fires on every relevant merge request, without burning shared CI minutes on unrelated repository activity. The CI job itself is described in detail in Chapter 9.

The most complete demonstration of this development street in action is a bug fix that was accepted into the MedNet `main` branch itself. While validating the caching behaviour of the temporal pipeline, I noticed that parallel dataset caching took roughly twice as long as expected. Profiling traced the cause to `CachedDataset`: a leftover `executor.submit` loop caused every sample to be loaded and transformed twice, doubling caching time, holding roughly 20 % more peak memory, and wasting a full pass of disk reads and CPU preprocessing on every cached run (Figure 5.1). I reported this as issue `#102`, implemented the fix on a dedicated branch, and submitted merge request `!79` through the team's standard contribution workflow. The merge-request pipeline ran the full test suite (2,604 tests, 81 % line coverage) before the framework maintainer reviewed the change and merged it into `main`, which automatically closed the issue and deleted the source branch.

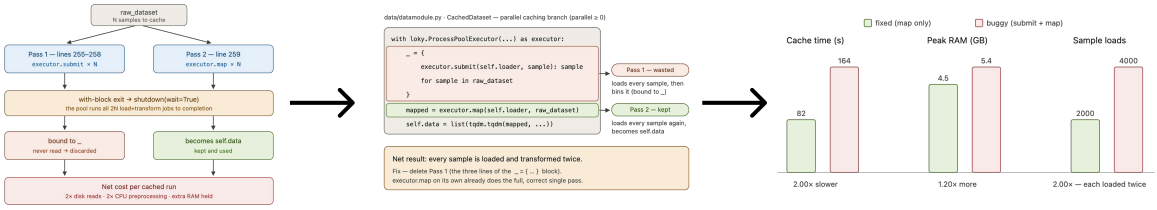


Figure 5.1: The `CachedDataset` double-loading defect fixed through merge request `!79` to MedNet. Left: the redundant `executor.submit` pass, whose results are bound to `_` and discarded, next to the `executor.map` pass that is kept. Middle: the offending lines in `data/datamodule.py` and the three-line fix. Right: measured impact — caching time doubles (82 s vs. 164 s), peak memory grows by a factor of 1.2, and every sample is loaded and transformed twice.

This cycle — detect, quantify, report, fix, and integrate — shows the development street doing exactly what it exists to do in a team setting: the automated pipeline and the maintainer review together are what made it safe to accept a contribution directly into shared infrastructure. It also marked a shift in my role within the team. Rather than only consuming MedNet as infrastructure for my own experiments, I helped guarantee its quality for everyone building on it, which the maintainer acknowledged in the review thread. Because the Idiap GitLab instance is not publicly accessible, the merged merge request, the passing pipelines, the diff, and the maintainer's response are documented as screenshots in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 16, Fig. 17, and Fig. 18.

5.2.3 Quality Assurance

A third aspect was quality assurance. In a research-oriented software project, quality does not only concern whether code runs, but also whether the workflow is reproducible, understandable, and reliable enough to support valid conclusions. I therefore worked in a stepwise validation process. Before launching larger experiments on the research infrastructure, I first checked whether outputs were logically correct and internally consistent.

When modifying data-loading procedures, sequence construction, or embedding extraction, I validated intermediate outputs before scaling up to more expensive runs. This reduced unnecessary compute usage and increased the chance of detecting implementation errors at an early stage. In addition, I maintained a clear repository structure in which scripts, documentation, and experiment-related components were separated logically. This improved maintainability and reduced ambiguity about where specific functionality belonged. An example of this

structured repository organisation is shown in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 14.

5.2.4 Flexibility

Another important part of Manage & Control was the ability to adjust the planning based on evidence. In a research setting, the next step cannot always be fixed in advance, because experimental outcomes influence which direction is most relevant. I therefore worked with short planning cycles and adjusted priorities when results or feedback justified doing so. For example, when an experimental direction produced weaker-than-expected results, or when another direction appeared more promising, I adapted the focus of the following week accordingly. These adjustments were documented in the logbook and reflected in the weekly analysis presentations. This made my planning flexible, but still controlled and accountable. Evidence of this replanning process is shown in `01_PROFESSIONAL_SKILLS_&_MANAGE_CONTROL/HIGHLIGHT-EVIDENCE.DOC` Fig. 15.

This way of working was particularly important because the project was carried out within an existing research environment rather than as an isolated student prototype. My contribution had to remain understandable, reusable, and sufficiently controlled for others in the team. By combining planning, structured reflection, version control, branch-based development, clear repository organisation, and incremental validation, I managed the project in a disciplined and traceable way and supported the quality of both the software process and the research output.

Chapter 6

Analysis

6.1 The question behind the question

The official goal was straightforward: improve automated grading for retinal inflammation on fluorescein angiography. Underneath it sat a more basic issue. The existing pipeline treated each examination as a single static image — the last frame. Before writing temporal model code, I needed to know whether that was wrong in clinical terms, and whether the data here could support a different choice.

To keep the analysis traceable for assessors, the rest of this chapter follows one line of reasoning from domain to decision. It first records how the work was steered (literature and supervision), then states what had to be true clinically (clinical domain and clinician-oriented requirements). After that, it turns empirical: whether Aptos supports temporal learning in practice (data), whether timing can be trusted (timestamp extraction and quality), how frames should be grouped in time (binning), and how competing temporal configurations can be compared without training every model to completion (Temporal Separability Benchmark with LDA probe). A separate controlled study tests whether unsupervised compression of frozen embeddings helps (PCA). The chapter ends by stating only the conclusions that downstream advice and implementation are expected to respect.

6.2 How this analysis was directed

The analysis had to be steered before the code: given how FA is read, what this dataset contains, and what the host pipeline allows, which options are worth treating seriously? I used three things together — clinical sense, data checks, and supervision — but the literature side was deliberately lightweight: not "read everything," but keep a small, organised trail of what mattered.

That trail lived in Notion. Figure 1 (`02_ANALYSIS/FIGURES/FIGURE_1.PNG`) shows the paper database: titles, dates, and keywords so I could see immediately whether I was still anchored in longitudinal, clinical imaging research or drifting into generic vision papers. Opening a row was where the real reading happened. Figures 2 and 3 (`02_ANALYSIS/FIGURES/FIGURE_2.PNG`, `02_ANALYSIS/FIGURES/FIGURE_3.PNG`) are two examples of the same habit — when the abstract was dense, I wrote notes in my own words and pinned down unfamiliar terms, so the paper became something I could actually reuse later rather than a tab I had once scrolled through.

Separately from the Notion paper log, I kept a long-form notebook of handwritten and sketched notes on the methods used later in the pipeline — for example ViT-style patch embeddings as they relate to frozen frame vectors, recurrent models such as LSTM [11] and GRU [5], Transformers [12], and probes such as LDA [13] and PCA [14]. These notes were for building a genuine understanding before running experiments; they are not additional empirical results. An exported PDF is included as supporting evidence in `02_ANALYSIS/STUDIED_CONCEPTS.PDF`.

That is how most of the general understanding behind the next sections was built. Supervisor and group feedback then chose which of those directions became formal experiments; the chapter that follows follows that same order.

6.3 Clinical Domain Analysis

Before the internship I had already read into the relevant biology — how dye moves through the circulation, what the retina is doing in broad terms, and how common pathologies differ — so I had a basic sense of why certain appearances show up. During the project I added the clinical-imaging side: what a fluorescein angiography sequence is meant to show, and how readers use change over time (early vs. late frames, stable vs. expanding vs. leaking patterns) to tell findings apart.

That reading was not separate from the technical work; it shaped what I treated as the actual task. It made it clearer which information belongs in the model input, which comparisons between time points matter, and where a single-frame shortcut would miss what the examination is for. The goal was to understand the problem well enough to argue for those choices, not only to learn vocabulary.

6.4 Clinician Requirement Analysis

The first move was not to open a dataset. It was to understand what a clinician is doing when they read a fluorescein angiography sequence, because any model that ignores that is solving the wrong problem.

A fluorescein angiography examination is not a photograph. It is closer to a short film. A fluorescent dye is injected intravenously and then tracked as it moves through the retinal vasculature over ten to fifteen minutes. Clinicians read this sequence by watching how the fluorescence changes: whether it stays put, slowly expands, or bleeds outward into surrounding tissue. A window defect appears early and stays the same size. Leakage starts small and progressively blurs. Staining accumulates brightness without expanding.

These patterns are defined by their behaviour over time — not by any single moment. This made the problem clear. A model that sees only the last frame is not just missing some information; it is missing the entire basis of the clinical diagnosis for several pathology types. That is not a limitation to optimise around. It is a structural flaw worth fixing. With that clinical picture in place, the quantitative checks could follow.

6.5 Stakeholder and Requirement Analysis

Before any modelling decision could be judged “good” or “bad”, I had to be clear about for whom, because this project served several stakeholders at once and they did not all want the same thing. Treating “improve the grading model” as a single requirement would have hidden exactly the trade-offs that turned out to matter most.

Five groups had a stake in the outcome. The HOJG clinicians are the eventual consumers of the grading: they care about clinical validity and about being able to understand why the model decided what it did, more than about a marginal metric gain. The MedAI researchers (my supervisors André and Oscar specifically) care that the work integrates cleanly with the existing RETFound/MedNet pipeline, that results are reproducible, and that experiments are economical on shared compute. Roberto, who built the single-frame baseline [9], is the owner of the preprocessing knowledge this project depends on and the author of the result I had to beat. Future researchers in the group are a stakeholder too: anything I built had to be

reusable and inspectable, not a private prototype. And finally, although they never sit in a meeting, the patients behind the HOJG data are a stakeholder in the strictest sense — their identifiable imagery imposes hard constraints on what I was allowed to do with it.

Translating those stakeholders into requirements gives the quality attributes that every later chapter is measured against:

Table 6.1: Stakeholder requirements and corresponding quality attributes. These attributes act as gates throughout the remaining chapters.

Stakeholder	**What they need**	**Quality attribute**
HOJG clinicians	Decisions they can trust and interpret	Clinical validity, interpretability
MedAI researchers	Beat the ROC-AUC 0.82; fit the existing stack	Performance, compatibility, reproducibility
Shared-grid users	Experiments that do not monopolise compute	Scalability, compute efficiency
Future researchers	Code they can reuse and extend	Maintainability, reproducibility
Patients (HOJG)	Their data handled lawfully and safely	Security, privacy

This table is more or less the contract the rest of the analysis answers to. When a later section "passes a gate" or "fails a gate", the gate is one of these attributes.

6.6 Data Analysis

I then had to check whether the available data could support temporal modelling. The dataset used for model development is the 3rd Aptos Competition Dataset [1], consisting of 1,921 examinations across 16 metadata fields, with images labelled by hyperfluorescence type: Leakage, Pooling, Staining, Window Defect, and None. The analysis in `02_ANALYSIS/APTOS-DATA-EXPLORATION.IPYNB` worked through this systematically.

The first thing worth checking was whether the labels were informative at all. They were — but not uniformly. Class imbalance was non-trivial, with some hyperfluorescence types substantially underrepresented. That matters because a model that learns to predict the majority class most of the time can look deceptively good on aggregate metrics while being useless for the cases that matter most clinically. This finding shaped two downstream decisions directly: class-weighted training to prevent the model from ignoring minority classes, and a commitment to interpreting per-class performance rather than averages alone.

The second thing worth checking was whether images within an examination changed across the sequence — because if they did not, there was nothing temporal for a model to learn. Comparing first and last frames within exams confirmed genuine visual progression across the sequence. The *temporal* signal was there. The question was how to extract it.

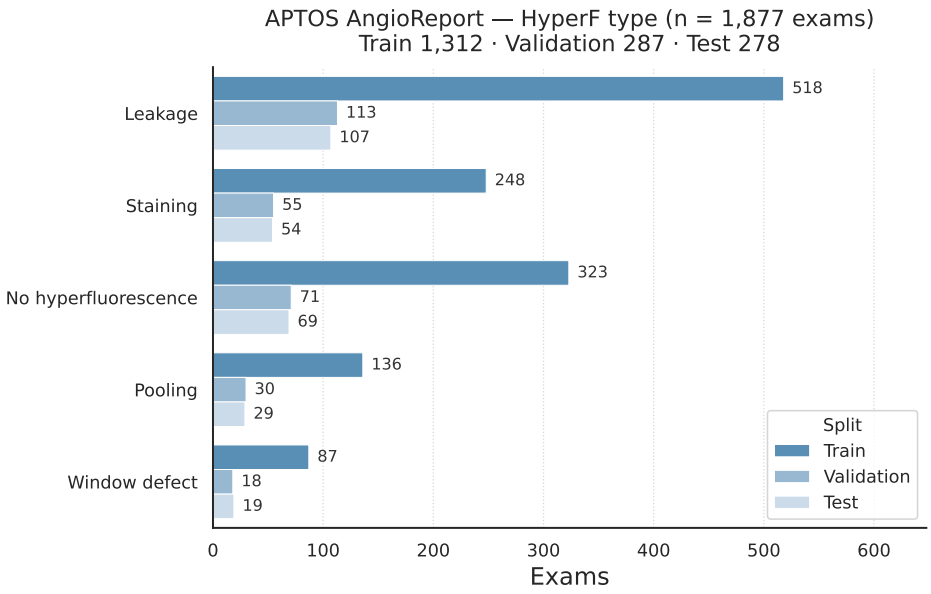


Figure 6.1: HyperF_Type class distribution in the Aptos dataset (n = 1,921 examinations, but 1,877 were eligible after analysis). The imbalance between classes motivates class-weighted training and per-class metric reporting rather than aggregate averages.

6.7 Timestamp Extraction Analysis

A temporal model is only meaningful if frame timing is reliable. Early in the project I treated timestamp handling as a technical detail, but this turned out to be a weak assumption. The Aptos dataset does not provide structured per-frame timing metadata, so elapsed time had to be reconstructed from timestamp text burned into each image. This made timestamp quality a first-order analysis problem, not a preprocessing footnote.

The idea was to use Optical Character Recognition (OCR) to solve this problem — OCR makes it possible to extract text from images, which was exactly what was needed here. I had no prior knowledge of OCR going in, so some additional research was needed to find the best approach for this dataset.

A few things had to be taken into account during the search for a suitable OCR service. First, accuracy had to be high, since these timestamps would be reused in future research and needed to be correct. Second, with more than 30,000 frames to process, the workflow had to be fast enough not to be a burden on the shared GPU grid. Third, it had to be reproducible, since the same problem could arise on other datasets at Idiap.

It is worth noting that clinicians working with HOJG data would never need this step — the HOJG dataset already contains elapsed-time metadata as part of the examination software. This was a specific solution for Aptos.

OCR services like TrOCR, AWS Textract, and Google Cloud Vision were considered, but proved too large to run on the grid; processing 150 frames alone took up to 60 minutes with those services, making 30,000 frames completely unfeasible. Looking at more lightweight options, EasyOCR and Tesseract stood out — both are Python libraries that run on CPU, are easy to reproduce, and place no burden on the GPU grid.

As a first approach I used EasyOCR, but it proved insufficiently robust: too many frames required fallback handling because no correctly extracted timestamp could be found. Based on that, I changed direction and developed a stricter OCR-based timestamp extraction and

Figure 6.2: Example Aptos FA frame. The elapsed time since dye injection is burned into the image ribbon as pixel text (visible in the bottom-left corner) but is not available as structured metadata. OCR was used to extract these timestamps at scale.

validation workflow, then benchmarked it explicitly against a manually checked ground-truth set of 150 frames (`02_ANALYSIS/OCR_BENCHMARK_ANALYSIS.PDF`):

Table 6.2: OCR accuracy and coverage on the 150-frame hand-checked benchmark.

Engine	**Accuracy**	**Coverage (frames with usable timestamp)**
EasyOCR	57.3% (86/150)	127/150
Tesseract	50.7% (76/150)	100/150
Gemini 2.5 Flash [2]	96.0% (144/150)	144/150

This changed the project direction: timestamp extraction was no longer treated as a minor preprocessing step, but as a core quality gate for temporal modelling. After switching to the Gemini-based workflow (timestamp strip extraction, schema-constrained JSON output, and explicit status tracking), I introduced a conservative training rule: only frames with `status = ok` and numeric `time_fa_seconds` are considered timestamp-eligible for temporal training.

Frames without a reliable timestamp turned out to be Indocyanine Green Angiography (ICGA) frames rather than FA frames, meaning they should not have been used in the first place. Excluding them, Gemini 2.5 Flash reached 100% accuracy (144/144) on the FA subset. With this new information I also tackled the problem of incorrect frame types entering the pipeline: when no timestamp was found, it was always the case that the frame was not an FA image, so it should not be used. This reduced noise in temporal supervision and made the provenance of each retained timestamp auditable.

6.8 Frame Timeline Analysis

The post-extraction timing behaviour is analysed in `02_ANALYSIS/TIMELINE_DISTRIBUTION.PDF`, the main follow-up analysis after OCR. That analysis shows where stamped frames sit on the FA timeline, how sampling density behaves within exams, and how exam coverage differs across early, mid, and late temporal regions.

The analysis revealed substantial variability: some examinations span up to 90 minutes, while most last at most 14 minutes (`02_ANALYSIS/TIMESTAMP-METADATA-QUALITY.PDF`). When I plotted the sampled density along each exam's stamped FA span, it showed that most frames cluster near the beginning or the end of the examination; relatively few are sampled from the temporal centre.

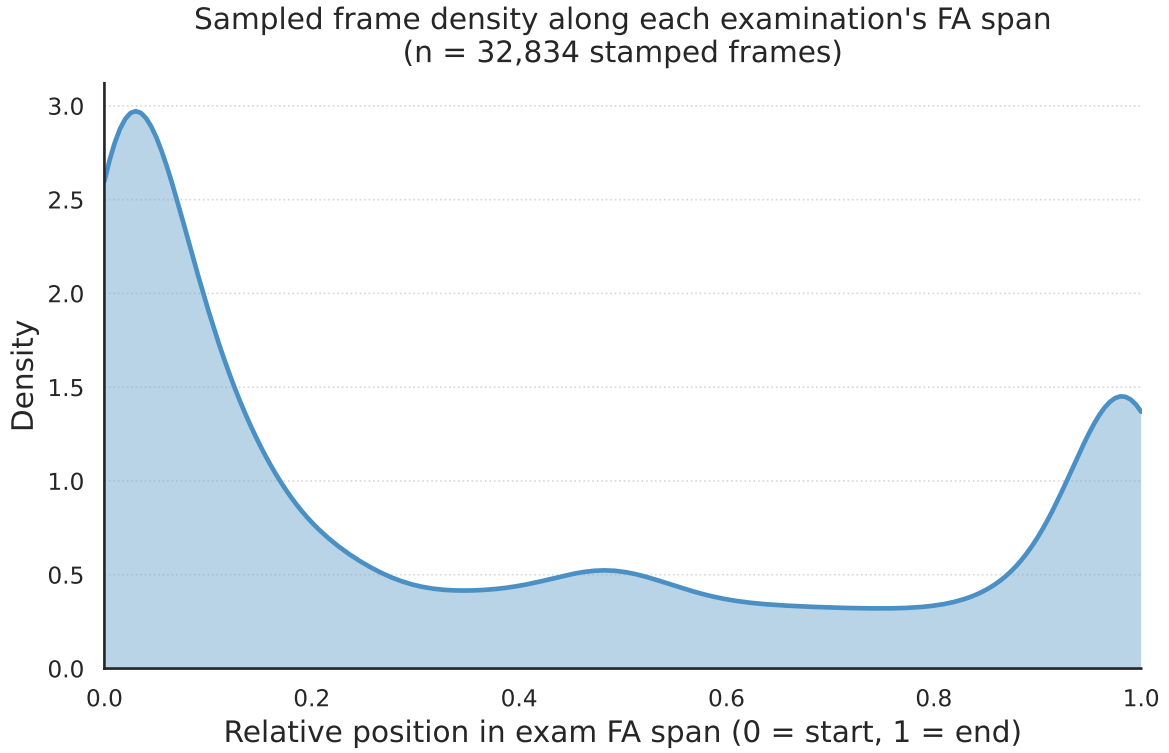


Figure 6.3: Sampled frame density along each examination's FA span. Frames cluster near the beginning and end of the sequence; the temporal centre is sparsely covered. This distribution shaped the choice of phase boundaries and the bin-then-select strategy.

So now that I knew the *when* (timestamps), the question became the *what*: were there distinguishable phases in the examination, and could they be identified empirically?

6.9 The Shared Measurement Instrument: LDA Probing

Before the next two subsections, it helps to name the tool they share. Whenever this chapter needs to compare candidate configurations empirically without running a full training cycle, it uses the same lightweight probe: a one-dimensional Linear Discriminant Analysis (LDA) [13] fitted on frozen embeddings. The backbone stays frozen throughout, so no weights change and the probe measures what is already in the representation, not what could be learned from it.

LDA finds the single projection axis that maximises separation between classes, and three metrics are then read off that axis. *Balanced accuracy* corrects for unequal class sizes and gives a fair performance score. *Overlap* is the fraction of the projected distributions that visibly intermix — lower is better. The *Fisher score* is the ratio of between-class variance to within-class variance (F = (between-class variance)/(within-class variance)); a score of zero means the classes sit on top of each other in the projected space, while higher values mean cleaner separation.

Together, the three metrics give a consistent and cheap signal about whether a given configuration preserves temporal structure. This probe is used twice below: first to calibrate phase boundaries against the Aptos frame distribution, and then inside the Temporal Separability Benchmark to rank binning strategies. Recognising it as one reused instrument rather than two separate techniques is important for following the reasoning — each application asks a different question, but the same measurement logic answers it.

6.10 Data-Driven Phase Analysis

So the timeline was now usable: each frame had an elapsed time, and unreliable frames had been filtered out. The next question was how that continuous timeline should be cut into phases for downstream temporal modelling. A model that uses phase labels (early, mid, late) only makes sense if those phases correspond to genuinely different signal in the embedding space; otherwise the labels are decoration.

Starting point: a clinical convention, not data. The first set of phase boundaries did not come from this dataset. They came from a small case-report on intravitreal bevacizumab for central serous chorioretinopathy (which is a mouthful, but they were essentially also using FA and had to define phases for the examination), which illustrates the FA timeline with three example phases captured at roughly 30 s (Early), 65 s (Mid), and 330 s (Late). Using midpoints between those reference times gave initial boundaries of 47.5 s and 197.5 s. This is a perfectly reasonable starting position with clinical provenance, but it is important to be explicit that it is not data-driven: it is a convention derived from three example frames in one pilot study, applied to a dataset of 1,877 exams.

Whether the convention transfers to Aptos is an empirical question, not a clinical one. The clinical timing of dye transit is a property of physiology; how that timing maps onto the distribution of frames in Aptos depends on how the original images were sampled. If Aptos overrepresents one phase and underrepresents another, boundaries that look reasonable on a textbook timeline can be poor descriptors of the actual data.

Testing the convention. To test whether the clinical convention transferred to Aptos, I applied the LDA probe with phase labels assigned to each frame purely from its elapsed time and a candidate pair of boundaries. Evaluation used a 5-fold GroupKFold split by examination, so no exam appears in both train and test — this matters because frames within an exam are highly correlated and a naive split would leak information.

Manual boundaries resulted in heavily skewed phase counts (only 3,589 frames in Early against 16,309 in Late) and the LDA probe scored a balanced accuracy of 0.643, a Fisher score of 1.56, and an overlap of 0.336. The LD1 histogram in Figure 6.4 shows the corresponding cluster geometry: Early and Mid sit on top of each other, while Late is the only cluster clearly displaced. In plain terms, the boundary the paper drew between Early and Mid is not where the embeddings change.

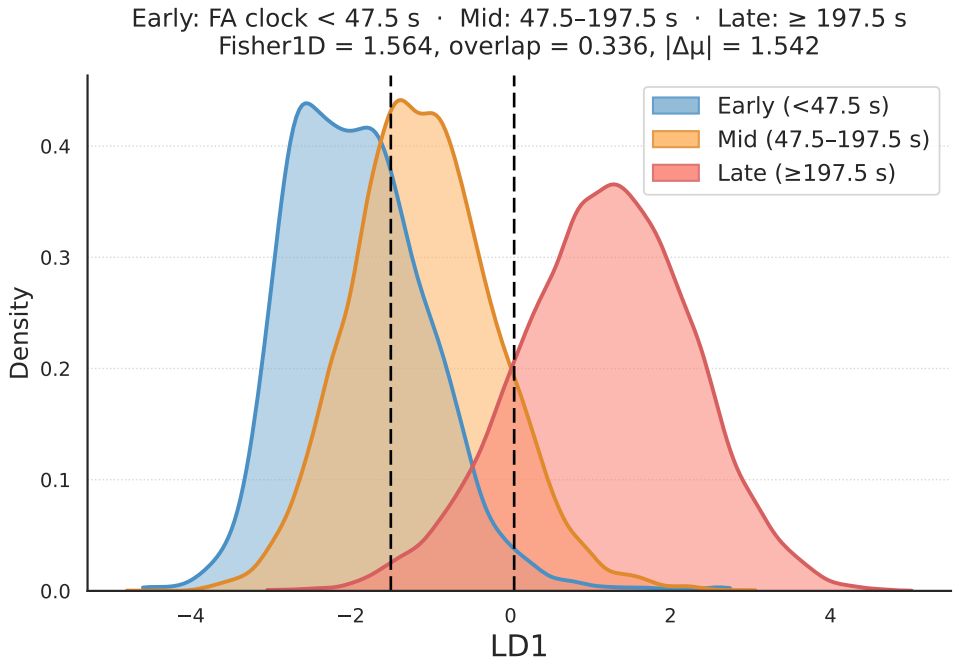


Figure 6.4: LD1 projection with manual phase boundaries (47.5 s / 197.5 s). Early and Mid overlap heavily; only Late is clearly displaced.

Data-driven search. I then ran the same probe over a grid of candidate boundaries derived from the time distribution itself: 28 quantile-spaced anchors between the 5th and 95th percentile of elapsed time, all 325 valid ordered pairs, with a minimum-phase-fraction constraint of 5% so no phase could collapse to a near-empty class. The argmax — meaning the combination with the highest scores — sat at 103 s / 518 s, with a balanced accuracy of 0.694, a Fisher score of 1.69, and an overlap of 0.315, and phase counts of 9,890 / 11,455 / 9,894. The LD1 histogram in Figure 6.5 shows the data-driven cluster geometry: Early and Mid are now more separate and distinguishable.

The data-driven boundaries are better than the paper-derived ones on every metric tested, though the gap is moderate rather than dramatic. The phases used for all downstream work are therefore the data-driven ones (103 s / 518 s), with the manual boundaries retained as a reference baseline. The next section takes this conclusion forward and asks the broader question of how time should be discretised at all — binning strategies and bin counts — given that even the correct three-phase split has visible residual overlap on LD1.

It is important to note that these data-driven phases are specific to the Aptos dataset — for other datasets, the spreading of the dye may occur earlier or later. However, the LDA phase probing procedure is fully reproducible and can readily be applied to external datasets to select the best phase boundaries for that data.

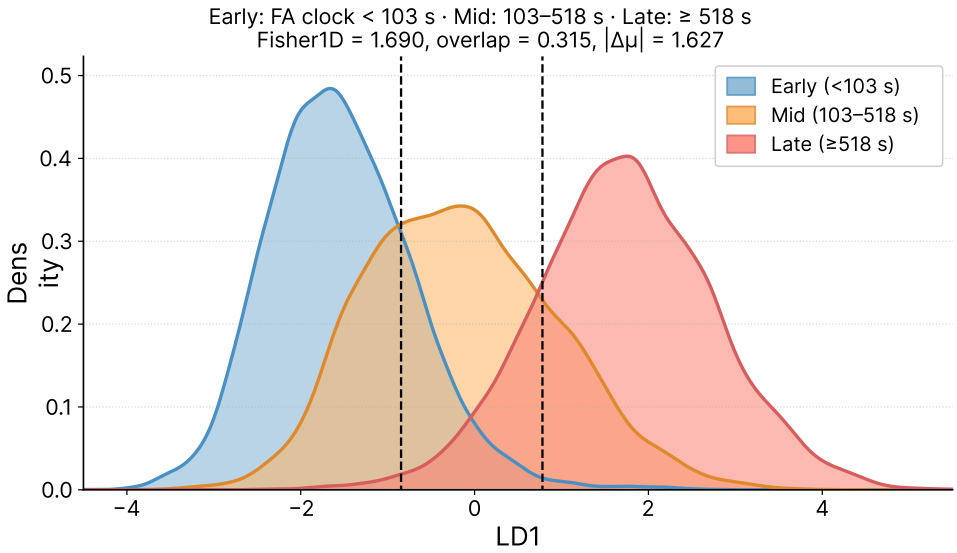


Figure 6.5: LD1 projection with data-driven phase boundaries (103 s / 518 s). All three phases are more separated. Fisher score improves from 1.56 to 1.69.

6.11 Temporal Separability Benchmark (TSB): LDA Probe for Configuration Ranking

Now that I knew the *when* and the *what*, it was time to test how to pick frames for the downstream classification task, because some sequence of frames had to be fed to the temporal model one way or another. The way I did this was by comparing different backbone families against each other and different frame-picking strategies against each other. The main idea of this Temporal Separability Benchmark (TSB) is captured in `02_ANALYSIS/TSB-STUDY-PROPOSAL.PDF`. I first sent this to my supervisor André and waited for his approval before starting, which I received.

Before starting the benchmark, I had to get the requirements clear. When choosing a backbone family, a few things had to be taken into account. There are 384-dimensional backbones and 1152-dimensional backbones — the bigger ones take substantially longer to train. Since we are working on a shared GPU grid, trading a little accuracy for efficiency was a worthwhile trade-off. Another consideration was that the MedAI team had been using the RETFound-Green backbone for prior research in this field and had a working relationship with the person who created it, which was worth something. And lastly, using more frames takes up significantly more memory, and the MedNet classification pipeline cannot handle arbitrarily long sequences, so fewer frames were preferred.

The full benchmark is documented in `02_ANALYSIS/PROBING_TEMPORAL_SEPARABILITY.PDF`. The same LDA probe served as the core measurement instrument, allowing low-cost comparison across backbones and temporal configurations before committing to expensive full training runs. The benchmark showed that temporal structure was present while also indicating diminishing returns from increasingly fine binning in several settings. At the same time, it was not treated as a single-command decision engine for backbone selection — it informed the analysis, but final implementation had to satisfy the additional constraints above.

At the end of the analysis, even though RETFound-Green did not obtain the best probe scores in absolute terms, its results were strong enough to justify staying with it: it is highly efficient, uses substantially less compute than the larger variants, and is already embedded in

the group's shared infrastructure. There was a trade-off in peak performance, but the team agreed it was worth it.

6.12 PCA on Frozen Embeddings

Separately from the TSB, I ran a controlled study — documented in `02_ANALYSIS/PCA-EMBEDDING-GRU-STUDY.PDF` — to investigate whether linear dimensionality reduction using Principal Component Analysis (PCA) [14] on frozen frame embeddings could act as a denoising step and improve HyperF-type classification when full examinations (all frames) are used.

Across tested backbones and variance-retention settings, PCA did not improve macro ROC-AUC or AP relative to raw embeddings and often performed similarly or slightly worse. Although negative, this was decision-relevant: it removed an attractive but non-beneficial branch from the solution space and kept the implementation path simpler.

6.13 MedNet Framework Analysis

When the project moved from a single-frame classification setup to multi-frame sequences with a temporal head, the limiting factor shifted from model choice to operational feasibility: training had to remain tractable on the shared GPU infrastructure while MedNet continued to serve as the primary framework.

The technical treatment is documented in `02_ANALYSIS/MEDNET_FRAMEWORK_ANALYSIS.PDF`. Rather than cataloguing MedNet end-to-end, that document answers a focused question: where GPU memory is spent during training, how requirements scale with batch size and numbers of frames, and why out-of-memory (OOM) errors can persist even when gradient checkpointing (a technique explained in the document) is enabled. The structure is question-driven to keep the analysis goal-oriented, so I would not go too deep or too shallow.

The emphasis is on algorithm and tool analysis (time and space): activation-related storage dominates parameter storage in this regime, and scaling is driven primarily by batch size and temporal depth. The main conclusion is straightforward: the practical bottleneck is not that ViT-Small is unusually large — quite the opposite compared to ViT-Large — but that a per-frame backbone multiplies work inside one optimisation step. MedNet was compared with external ecosystems such as MONAI [15], which showed some interesting techniques for handling larger numbers of samples. These ideas were taken into consideration by the team, but transferring projects to the MONAI ecosystem was not a realistic option — the MedAI group is deliberately building their own framework in MedNet, and moving projects out of it would go against that direction.

6.14 Model Error Analysis

After obtaining the first results, I needed to understand where the actual classification mistakes were being made. For this purpose I designed a Python script that takes the `predictions.json` file — automatically generated by MedNet after training and evaluation, containing the softmax logits per examination — and visualises the entire examination sequence alongside the ground truth and model prediction.

The output is an interactive `.html` file (`02_ANALYSIS/ERROR_ANALYSIS.HTML`). HTML was chosen so that everyone could easily access and interact with it: since the images are embedded as base64, there is no need to have the raw frames on your own disk, making it immediately accessible to anyone on the team. The file can also be used interactively to observe how the model's predictions change as it processes more frames in the sequence.

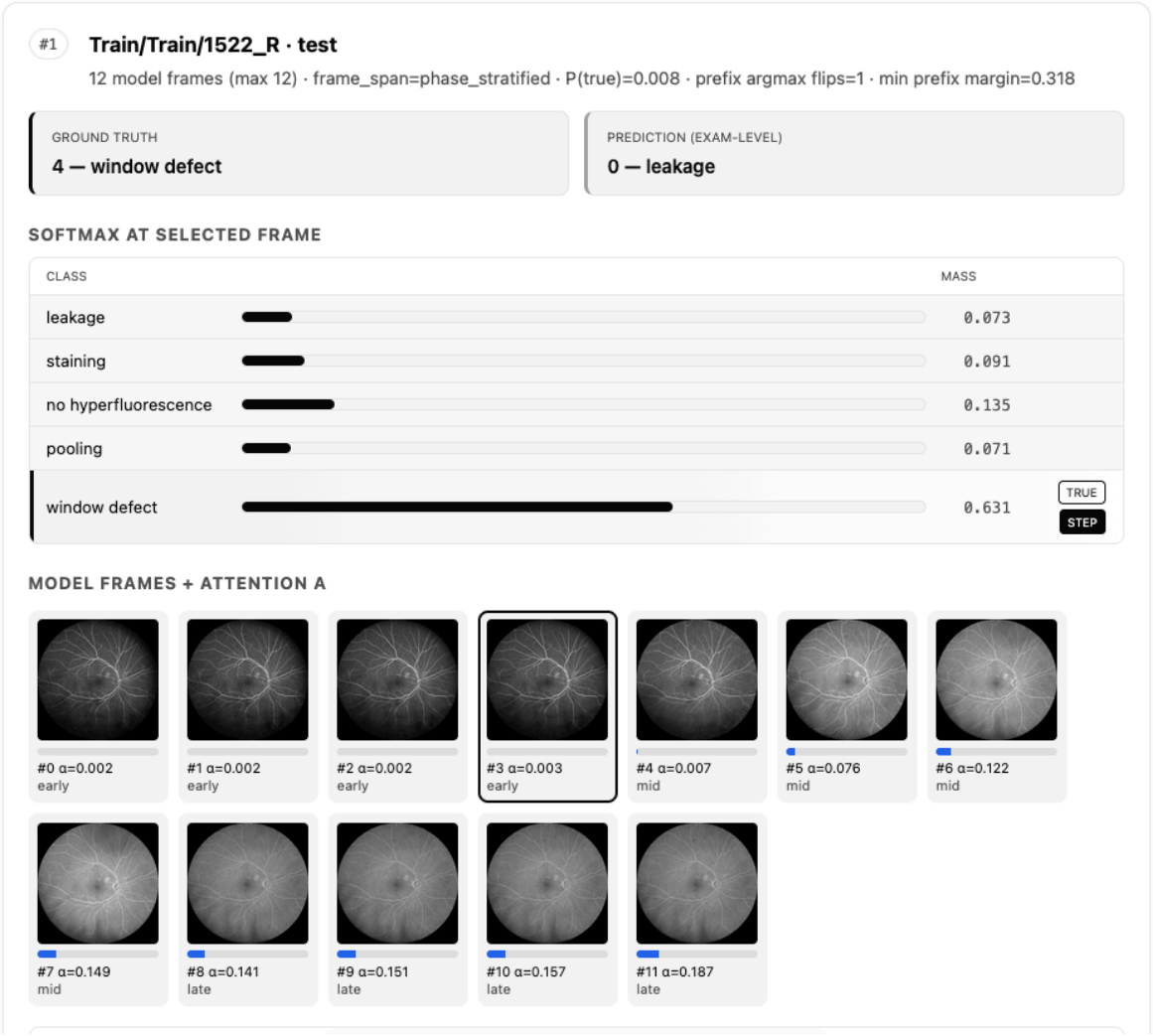


Figure 6.6: The interactive error analysis tool showing a misclassified examination (ground truth: window defect; exam-level prediction: leakage). At the selected frame the softmax correctly ranks window defect highest (0.631), yet the exam-level label is wrong. The attention weights α (which are not covered in this portfolio) reveal why: they climb from near-zero in the early phase to 0.187 by the final late frame, meaning the model weighted late-phase evidence most heavily — the opposite of what window defect requires, since this pattern appears early and remains geographically stable across the sequence.

The `.html` file alone was not enough — it had to be manually checked and interpreted. I had to make a clear separation between two types of error. The first type was irrecoverable: some frames were blurred or otherwise corrupted in ways that made correct classification impossible regardless of model design.

The more important type was the one I could act on — what my supervisor André called *actionable* mistakes. When reviewing the misclassified examinations, one pattern stood out: the model had consistent difficulty identifying pathologies that occupied only a small area of the image, for example leakage visible in just a few pixels.

This led to further investigation. Our FA frames are originally 334×334 pixels, but the Vision Transformer expects 224×224 inputs. The current pipeline resizes frames by interpolation,

which discards fine spatial detail and may prevent the model from detecting small, subtle lesions. I decided to point this out to the team as advice; it is discussed further in Chapter 7.

6.15 HOJG: Scope and Transfer

The fifth research question asks whether performance transfers from Aptos to the external hospital dataset, HOJG. This requires me to be direct about how far this project could take that question.

HOJG is the main clinical target, but it is also the hardest dataset to access, and for good reason. As the data governance section explained, it is identifiable patient data that cannot leave Idiap's secured infrastructure. In practice this meant that running anything against HOJG required my codebase to be mirrored into a separate, access-controlled environment and the necessary clearances to be obtained. Completing that mirror was a slow process — the privacy safeguards that protect the patients are precisely what made the data hard to reach. That process took longer than the project window allowed, so within this internship I was not able to produce HOJG results. This is a scoping reality, not a result: Q5 remains open for future research.

What this changes is how the rest of the project should be read. The whole method was deliberately developed on Aptos as a transferable benchmark: public, reproducible, and far better suited to eventual publication as a paper or journal article. The phase-calibration probe, the binning strategy, and the entire MedNet GRU pipeline were all built to be re-runnable on a new dataset rather than fitted to Aptos alone. The temporal method is ready for HOJG, the infrastructure access has almost been fully established, and the external validation step will be carried out by the MedAI team in the future.

6.16 Analysis Conclusions

The chapter ultimately resolves into a single chain of decisions. Temporal modelling is clinically justified because FA interpretation is inherently dynamic. On Aptos, timestamp quality is a gating variable, so temporal supervision must be restricted to explicitly valid time reads. Temporal phase boundaries should be calibrated to dataset behaviour rather than copied unmodified from external conventions. Multi-backbone temporal probing remains valuable for insight, but final implementation choices must also satisfy reproducibility and integration constraints.

For that reason, the project stayed with RETFound-Green in the implementation context, while still reporting TSB findings as meaningful exploratory evidence rather than discarded work.

That is the core analytical story: not a straight-line success narrative, but a controlled sequence of revisions in which weak assumptions were exposed, corrected, and converted into more defensible downstream decisions.

Chapter 7

Advice

This chapter takes the Analysis chapter as its starting point and translates those findings into recommendations that are actionable in the MedAI setting. The key analytical shift that must carry into advice is the timestamp pivot: earlier OCR approaches were not reliable enough for temporal supervision, but after benchmark-driven method revision and stricter eligibility filtering, timestamp quality became sufficient for controlled temporal experiments on Aptos.

Advice therefore distinguishes clearly between two contexts: Aptos as a reconstructed-timestamp benchmark setting, and HOJG as a native-metadata clinical setting. The goal of this chapter is not to repeat empirical results, but to recommend which strategy class is most defensible under practical constraints. In addition to formal advisory documents, these recommendations were iteratively discussed in weekly MedAI presentations, where current findings were translated into concrete next-step decisions for the following week(s).

7.1 What the Customer needed

Before any recommendation could be written, the requirements had to be understood clearly enough to constrain the answer. The MedAI group needed a temporal deep-learning pipeline for FA grading that could run reproducibly on their existing GPU infrastructure, integrate cleanly with the RETFound family of foundation models already in use, and remain interpretable enough to support discussion with clinical partners. Three quality criteria shaped every subsequent recommendation.

First, compute cost and scalability: the cluster is shared, so proposals that multiply training cost without a believable gain are not actionable. Second, infrastructure compatibility: replacing the backbone was not a realistic recommendation within this project's scope, because it breaks continuity with the group's established pipeline. Third, interpretability: an opaque bump in a summary metric is weak evidence if it cannot be connected to how clinicians reason about phases and dynamics.

Finally, there was a sobering baseline reality: a strong single-frame reference already existed [9], so temporal modelling had to earn its keep against those same gates.

7.2 From first draft to revised advice

The first advisory document (`03_ADVISE/ADVISORY-WEEK04-OPTIONS-MATRIX.PDF`) was produced after the earliest LSTM experiments had returned promising results. It compared three options: temporal binning with an LSTM [11], a full-sequence LSTM, and a GRU [5], and presented a recommendation matrix with estimated effort, expected gain, and risk per option. As a starting point it served its purpose: it made the options concrete and created something to discuss and improve.

Through supervision it became clear that the scope could be broader. The document had taken the existing binning approach as the natural frame of reference and asked which variant

to pursue, without stepping back to ask whether the overall strategy was the most defensible one. The project criteria — compute cost and interpretability — were named upfront but not yet systematically applied in the comparison. Expanding the scope before committing to a direction was the right next move.

Feedback received — supervisor Oscar, week 5–6 supervision meeting

> *The initial advisory trajectory was focused too narrowly on optimising binning parameters and had not sufficiently considered alternative temporal encoding strategies from the recent literature. The recommendation was to broaden the scope before committing to a design direction. This feedback directly produced the literature review in* `02_ANALYSIS/LITERATURE-TEMPORAL-ENCODING.PDF` *and the revised advisory report.*

This reframing sharpened the question considerably. Rather than "which binning variant is best?", the revised advice asked "which strategy for encoding time is most defensible given the domain, the available data, and the group's constraints?" — and answered it by applying the project gates explicitly to each option.

7.3 Alternatives and their Assessment

Working through the literature with that broader question produced three strategy classes that were realistic for this environment. The evaluation below assesses each explicitly against the three core project gates: compute/scalability, backbone compatibility, and clinical interpretability. The goal is not to choose "which paper sounds newest", but to identify which strategy survives these gates.

Temporal binning. This approach turns a variable-length exam into a short sequence of bin embeddings by mean-pooling or centre-selecting frames within fixed or quantile-based time windows, which are then fed to a recurrent model. This would behave very well operationally: binning introduces only minimal preprocessing overhead, scales easily to larger or external datasets, and keeps the backbone and head interfaces intact, so it passes the compute gate. Interpretability is moderate — bins naturally encode relative order, but they abstract away elapsed time, which can drift from how clinicians reason about "early vs. late" on a clock.

Explicit time-embedding. This strategy maps the explicit elapsed time since dye injection to a learned vector, injecting that signal alongside each frame embedding prior to the sequence model. Computationally this is strong: it adds minimal training cost compared to large architectural rewrites, and it would be possible to keep using the same backbone. Interpretability is better than binning because the model directly encodes a clinically meaningful, continuous notion of time rather than relative bin order.

Heavier temporal objectives. These include multi-scale sequence modelling or contrastive training across time. This class fails on compute: it introduces substantial complexity, requires more components, and demands longer training cycles. It is also unclear whether the current RETFound-Green backbone is compatible with multi-scale objectives, since multi-scale sequence modelling asks for different outer input/output interfaces. Interpretability also fails: the increased complexity makes it much harder to explain model behaviour to clinical partners.

Table 7.1 summarises the assessment.

Table 7.1: Assessment of temporal encoding strategies against the three project gates.

Strategy	**Compute**	**Backbone compat.**	**Interpretability**
Temporal binning	Pass	Pass	Moderate
Explicit time-embedding	Pass	Pass	Pass
Heavier objectives	Fail	Unclear	Fail

7.4 The recommendation

The advisory report (`03_ADVISE/ADVISORY-TEMPORAL-ENCODING.PDF`) recommends a two-track approach, derived directly from the analysis facts above, not from preferences in isolation.

Track 1 keeps temporal binning as a controlled baseline. It already satisfies all three gates, matches what was empirically exercised in depth, and removing it would weaken any subsequent claim about alternatives.

Track 2 implements a time-embedding strategy. It is the only option that scores acceptably on all three customer criteria simultaneously: low compute cost, no infrastructure disruption, and the highest interpretability of the three strategies considered. It also addresses the structural weakness of the binning baseline: the model encodes actual elapsed time rather than relative frame position, which is what the clinical task requires.

Multi-scale and contrastive approaches are deferred: not because they are wrong in theory, but because they fail the gates now and would complicate interpretation before the simpler time signal is validated — especially on HOJG, which has clean native timing metadata.

So although Track 2 may seem the most promising direction, implementing it in the existing architecture requires substantial effort and understanding. The right starting point was therefore Track 1, establishing a validated temporal baseline before building on it. When I proposed the two tracks at the bi-weekly MedAI group meeting, André and Oscar agreed: research is about making iterative progress, start small, end big. So Track 1 was picked first.

Feedback received — supervisors André and Oscar, bi-weekly MedAI meeting

> *The interpretability gate had been interpreted incorrectly. Interpretability in this context means the model's ability to show directly where it found evidence and how it reached its decision — for example via Gradient-weighted Class Activation Mapping (Grad-CAM), which overlays a heatmap on the input image showing which regions the model attended to. This is distinct from whether the model's reasoning aligns with clinical practice. The rest of the analysis and the two-track recommendation were received positively.*

Incorporating this feedback clarified an important distinction. I had not heard of Grad-CAM before this meeting. Stating "interpretability" as a gate without grounding it in a specific mechanism — such as Grad-CAM or attention visualisation — leaves the criterion open to misinterpretation. Future advisory work should specify which interpretability tools are expected, not just name the concept.

7.5 How the PCA study fits next to the temporal advice

The main advice concerns how to represent time, but I also tested whether reducing embedding dimensionality before sequence modelling would provide a practical benefit. The PCA study [14] (`02_ANALYSIS/PCA-EMBEDDING-GRU-STUDY.PDF`) did not improve macro ROC-AUC or AP relative to raw embeddings across any tested setting.

This negative result directly supports the advisory criteria: PCA adds complexity and hyperparameter surface without delivering the interpretability or performance gain needed to justify inclusion in the current pipeline. The recommended default is therefore to keep backbone-dimensional embeddings unless future objectives explicitly motivate supervised compression.

7.6 MedNet Framework Advice

This advice is derived directly from the MedNet Framework Analysis: the same constraints, expressed as recommendations rather than derivations. Each option was checked against the same three gates used elsewhere in the project: compute cost and scalability on a shared cluster, compatibility with MedNet and RETFound-style backbones, and interpretability of the modelling choices for discussion with colleagues and clinical partners.

The two primary recommendations are conservative by design. First, cached frozen ViT features should be used when rapid iteration is required, training the temporal component on stored embeddings rather than recomputing the backbone every epoch. Second, MedNet should be extended so temporal batches become an explicit contract — data loading, collation, and a clear model wrapper — rather than relying on ad hoc local modifications that are difficult to review and reproduce, which is especially important for future researchers picking up this work.

The recommendations were presented to the MedAI group as a presentation; the slides and speaker notes are in `03_ADVISE/MEDNET_FRAMEWORK_ADVISE.PDF`. Several questions came up during the discussion that I had to answer on the spot.

I was asked why we could not simply reduce the batch size to solve the memory problem. I answered that lowering batch size eases peak VRAM but does not change the linear scaling with the number of frames, and that recovering an effective batch size through gradient accumulation usually costs additional time on a shared cluster — so it is a mitigation, not a structural fix, but it was a fair point.

Another question concerned whether a switch to MONAI [15] was feasible. I explained that MONAI is a strong ecosystem, but that switching the whole stack would imply re-validation, re-training habits, and significant integration work. Extending MedNet is the more realistic path for current Idiap workflows, while MONAI remains a sensible choice for genuinely greenfield projects — projects that carry no prior infrastructure constraints.

Finally, someone asked whether the ViT could be fine-tuned later. I said yes, but that fine-tuning should come in a second stage after the temporal setup is validated on cached embeddings. In practice, that view matched what happened next: cached embeddings were very useful for linear probing and light configuration work, whereas stronger experiments still required partial backbone fine-tuning, so caching was a deliberate first step rather than the end state.

On the basis of this discussion, I was assigned work on a dedicated MedNet branch for temporal Aptos integration, so that changes could evolve without destabilising the mainline used by other researchers.

7.7 External dissemination and validation (TAM, SAIMI, OMIA13)

As part of the advisory phase, I presented this work at the Idiap Tuesday Afternoon Meeting (TAM), documented in `03_ADVISE/TAM_PRESENTATION.PDF`. The TAM audience spans researchers across all groups at Idiap, so the session functioned as a cross-disciplinary advice dissemination moment rather than an internal pipeline update. The presentation communicated

not just results but decision logic: why last-frame assumptions are limiting, why timestamp quality must be treated as a gating condition, and how temporal strategy choices should be constrained by interpretability and operational feasibility.

I made the presentation as accessible as possible for the whole audience. Towards the end it seemed to have worked: I got a lot of discussions going. It turned out that by showing how I tackled this temporal problem, I was actually advising other people across Idiap on how to approach their own temporal problems — something I had not anticipated at all. Temporality is apparently not just a challenge in the MedAI group.

People came to me saying they had never used a GRU as a temporal model in combination with a transformer before, but that it seemed like a plausible solution for their own data. One piece of feedback proved directly actionable for my own work: a colleague suggested that changing the ViT token aggregation from average pooling to max pooling can improve performance, particularly in medical AI settings. One week later, after implementing this change, test AUC improved from 0.85 to 0.87.

Beyond the TAM, the abstract summarising the project's main findings and recommendations was submitted to and accepted by SAIMI (Swiss AI in Medicine Initiative, June 2026), validating that the advisory conclusions — specifically the two-track temporal approach and the timestamp quality gate — are considered relevant and novel by an independent scientific committee.

Furthermore, the findings are being prepared for submission to OMIA13, the Ophthalmic Medical Image Analysis workshop within MICCAI, one of the most competitive venues in medical image analysis. This prospective publication path demonstrates that the advisory and analytical outputs of this project are of publishable quality and that the recommendations made to the MedAI group are grounded in findings that merit dissemination to the broader research community.

7.8 Mitigating Information Loss in Pre-processing

The error analysis (`02_ANALYSIS/ERROR_ANALYSIS.HTML`) indicates that reducing input resolution is associated with loss of subtle visual cues. These cues matter clinically: small lesions such as pixel-scale leakage are difficult to detect even for experienced clinicians. My recommendation to the MedAI team is therefore not to downscale inputs for deployment without a dedicated check that performance and error patterns still hold on the current evaluation setup. If downscaling is unavoidable later, it should be an explicit decision with documented preprocessing and re-validation, not something that quietly slips into the pipeline.

I first did the analysis earlier that day, then reached out to Roberto on Mattermost to share my interpretation and proposed solution (`03_ADVISE/advice-information-loss.png`). I wanted to make clear that I thought there was a structural issue in the pipeline and to propose a change, while also questioning whether it was actually feasible in practice — because I knew Roberto had more knowledge about the current preprocessing pipeline than I did.

As shown in the Mattermost conversation, Roberto was quick to acknowledge that the concern was not unfounded. He agreed that downscaling the frames could blur useful information, but also raised fair counter-arguments about memory consumption and inference latency. He was kind enough to point out several possible options, which gave me a lot to think about.

In the end it came down to one point: this is always going to be a trade-off. We would gain in detection of subtle pathologies but lose in compute efficiency and latency. Looking at the requirements stated earlier in this portfolio — where compute efficiency carries significant

weight — using full-resolution inputs to fix these marginal model errors was not justified. Thanks to the feedback from Roberto, I put this idea aside in favour of higher-return investments, but the concern is documented here so future researchers can revisit it with a clear starting point.

7.9 Next steps and handover

This project ends with the temporal pipeline validated on a public benchmark and ready for the next phase. The following recommendations are intended directly for whoever continues this work — specifically the PhD student joining the MedAI group in June 2026, who will be working on the same project.

Most urgent: clinical validation on HOJG. The single most important next step is running the temporal pipeline on the HOJG hospital dataset. The infrastructure access has been established, the pipeline requires only a new split file to run on new data, and the phase calibration procedure can be re-applied directly to HOJG's native timing metadata — which is reliable, unlike Aptos. This answers research question Q5 and is the step that turns a benchmark result into a clinically meaningful one.

Track 2: explicit time-embedding. The revised advisory recommended two tracks. Track 1 (temporal binning with a GRU) was implemented and validated. Track 2 — injecting the actual elapsed time since dye injection as a learned signal alongside each frame embedding — was deferred but remains the more principled long-term direction. It encodes what clinicians actually use (elapsed time on a clock) rather than relative frame order, and it passes all three project gates. Now that Track 1 provides a validated baseline, Track 2 is the natural next experiment.

Interpretability: Grad-CAM across phases. Supervisor feedback clarified that interpretability in this context means showing *where* in the image the model found its evidence — for example via Gradient-weighted Class Activation Mapping (Grad-CAM). This was not implemented within the internship window but is a clear next step, both for clinical trust and for understanding what temporal information the model actually uses across phases.

Input resolution. The error analysis identified that downscaling frames from 334×334 to 224×224 pixels may cause the model to miss small lesions such as pixel-scale leakage. This trade-off between detection quality and compute cost was documented but not resolved. It is worth a dedicated evaluation once HOJG validation is underway.

Handover resources. To support a smooth transition, the weekly progress slides covering the full internship arc are available in `02_ANALYSIS/PROGRESS\_SLIDES/`. These form a chronological record of every major decision, experimental result, and change of direction — not just the conclusions, but the reasoning behind them as it developed. For someone picking up the project mid-stream, they are a faster way to understand *why* things are the way they are than reading the portfolio from the start.

A copy of this portfolio will also be handed over directly: at the beginning of my time at Idiap, Roberto gave me his thesis and it helped enormously. The TAM presentation recording (https://youtu.be/9NKKttZj698) is also recommended as a first watch, giving a 15-minute spoken overview of the full project arc.

7.10 Reflection

The central learning in this advisory round was methodological discipline. Early advice optimised within an assumed solution class and treated the evaluation criteria too loosely. The revised approach starts from explicit constraints, evaluates alternatives against those constraints, and states trade-offs directly.

A second lesson was equally valuable. Working at an institute like Idiap can induce imposter syndrome — the feeling that everyone around you is substantially more capable and that your own contributions are not worth offering. When you feel that way, walking up to someone and giving them advice starts to feel like a burden to them. But in practice, the opposite turned out to be true: even the researchers I looked up to the most genuinely appreciated the analyses and recommendations, and engaged constructively with them.

As a result, I learned to trust the technical work and to engage proactively — rather than waiting to be asked — and that approach drove the project forward in ways that would not have happened otherwise.

Chapter 8

Design

This chapter describes how the temporal pipeline was designed after Analysis and Advice, but before full integration work in Realisation. Chapter 6 established that temporal modelling is clinically justified and that timestamp quality can be used under strict eligibility rules. Chapter 7 then narrowed the strategy space to a two-track path: temporal binning as a controlled baseline, and explicit time-aware modelling as the primary direction. Design translates those conclusions into architecture choices, quality trade-offs, and a validation strategy.

8.1 Architecture Overview

The design task was to extend a single-frame ViT classification baseline into a temporal pipeline that processes ordered sequences of fluorescein angiography frames per examination, without exceeding the shared cluster's memory budget. The resulting architecture consists of five sequential blocks, each serving a distinct function and each shaped by an explicit quality concern. The full architecture with tensor specifications is documented in `04_DESIGN/TEMPORAL-PIPELINE-ARCHITECTURE.PDF`.

Block 1 — Image loading and preprocessing. Frames are grouped by examination using the `temporal_hyperf_type.json` split file (shown in 9.3.1). This file defines the train, validation, and test partitions, as well as the HyperF_Type label and elapsed-time metadata for every frame. By making the split file the single source of truth for data organisation, the pipeline avoids hardcoded assumptions about any specific dataset. The fixed split file also eliminates partition variance across runs, which directly supports reproducibility.

Block 2 — RETFound-Green ViT backbone [4]. A pretrained Vision Transformer extracts a 384-dimensional embedding per frame. The backbone is frozen (weights are not updated) during simple probing experiments and unfrozen during full fine-tune training. Embeddings are computed once and cached to disk, so repeated training runs that do not require backbone unfreezing do not re-extract features. This reduces both GPU memory and compute costs, supporting scalability. The cached embeddings also guarantee that every experiment starts from identical feature representations, removing a source of run-to-run variance and supporting reproducibility.

Block 3 — Phase-based binning and frame selection. Each examination's frames are assigned to three data-driven temporal phases (Early, Mid, Late) based on elapsed time from the angiographic injection. Within each phase, four representative frames are selected, producing a fixed 12-embedding sequence. This bin-then-select strategy is the core design contribution and is discussed in detail in Section 8.2. The bin structure also makes it possible to discuss which temporal phase contributes most to a prediction, which would be difficult if the model consumed all frames in an examination at once.

Block 4 — Two-layer GRU sequence encoder [5]. A two-layer Gated Recurrent Unit processes the 12-step embedding sequence and outputs a hidden state of dimension $H = 256$ that summarises the temporal progression of the examination. The GRU was chosen over a Transformer [12] because the short, fixed sequence length makes the recurrent option more parameter-efficient on the shared cluster. Variable-length examinations are handled through padding and `pack_padded_sequence` (a PyTorch module), ensuring the model never treats padded frames as real observations.

Block 5 — Linear classification head. The unchanged classification head from the single-frame baseline maps the GRU's hidden state to five HyperF_Type class probabilities `(Batch, 5)`. Keeping this block identical to the baseline ensures that any performance change is strictly attributable to the temporal architecture rather than a classifier change.

Isolating the temporal contribution to a single new block (the GRU) means that performance changes can be localised immediately: if the temporal model underperforms the baseline, the issue is in Blocks 3–4, not in the head. This directly supports the debuggability of the model architecture.

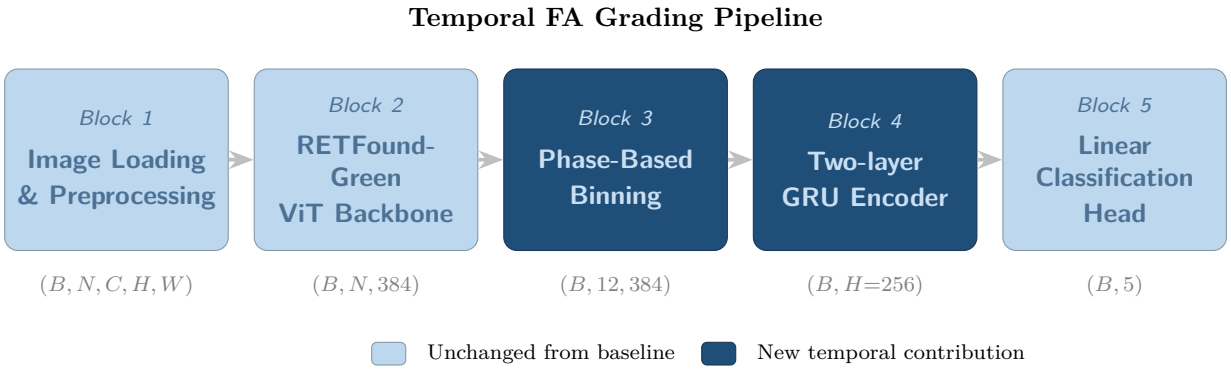


Figure 8.1: The five-block temporal pipeline. Blocks 1–2 and 5 are unchanged from the single-frame baseline (light blue); Blocks 3–4 are the new temporal contribution (dark blue). Tensor dimensions are shown at each stage.

8.2 Design Decisions and Alternatives

With the end-to-end architecture defined, this section documents the design decisions that shaped its internal components. Each decision is presented as a comparison between considered alternatives, with an explicit rationale for the chosen option.

8.2.1 GRU over LSTM

The initial design, documented in `04_DESIGN/DESIGN-DECISIONS.PDF` and reviewed with supervisor André, specified an LSTM [11] as the recurrent encoder. After the design review, the implementation switched to a GRU [5]. The GRU uses two gates instead of three, reducing the parameter count for the same hidden dimension. On a fixed 12-step sequence, the additional gating capacity of an LSTM provided no measurable benefit, while the GRU's lower memory footprint is a concrete advantage on the shared cluster. Critically, the tensor contract remained unchanged: both architectures accept `(Batch, Seq, Embedding_Dim)` and output `(Batch, Hidden_Dim)`, so the switch required no changes to upstream or downstream blocks. The fewer parameters per recurrent step support the efficiency and scalability of the model architecture.

8.2.2 Temporal Binning Strategy

The central design question was how to represent temporal information from a variable-length examination — which can contain up to 200 frames — as a fixed-length input to the GRU. Three strategies were considered.

Mean pooling. All embeddings within a phase are averaged into a single vector, producing a 3-step sequence (one per phase). This approach is simple and handles variable frame counts naturally. However, averaging blurs discriminative temporal changes near phase boundaries, which is exactly the kind of signal a temporal model should learn. This risk was identified during supervisor feedback in the week 3–4 design review (see Section 8.4).

Centre-frame selection. A single representative frame is selected per phase: the first frame for Early, the middle frame for Mid, and the last frame for Late. This preserves temporal sharpness and avoids the averaging problem, but with only 3 embeddings, the GRU receives minimal temporal context.

Multi-frame selection (chosen approach). Four frames are selected per phase using uniform spacing within the phase, producing a fixed 12-embedding sequence. This approach retains temporal progression, provides multiple observations per phase for the GRU to learn from, and keeps the sequence length within the cluster's memory budget. Iterative experiments confirmed that supplying more frames improved classification results, and 12 embeddings did not trigger out-of-memory errors on the shared server. Because the sequence length is fixed at 12 frames regardless of how many raw frames an examination contains, cost scales with phase count (currently 3), not with raw frame count, supporting scalability.

As presented in the TAM presentation (`03_ADVISE/TAM_PRESENTATION.PDF`), this bin-then-select strategy provided a data-driven balance between temporal fidelity, interpretability, and computational feasibility, and was therefore carried forward as the preferred implementation path. Mean pooling was retained as an experimental comparison baseline.

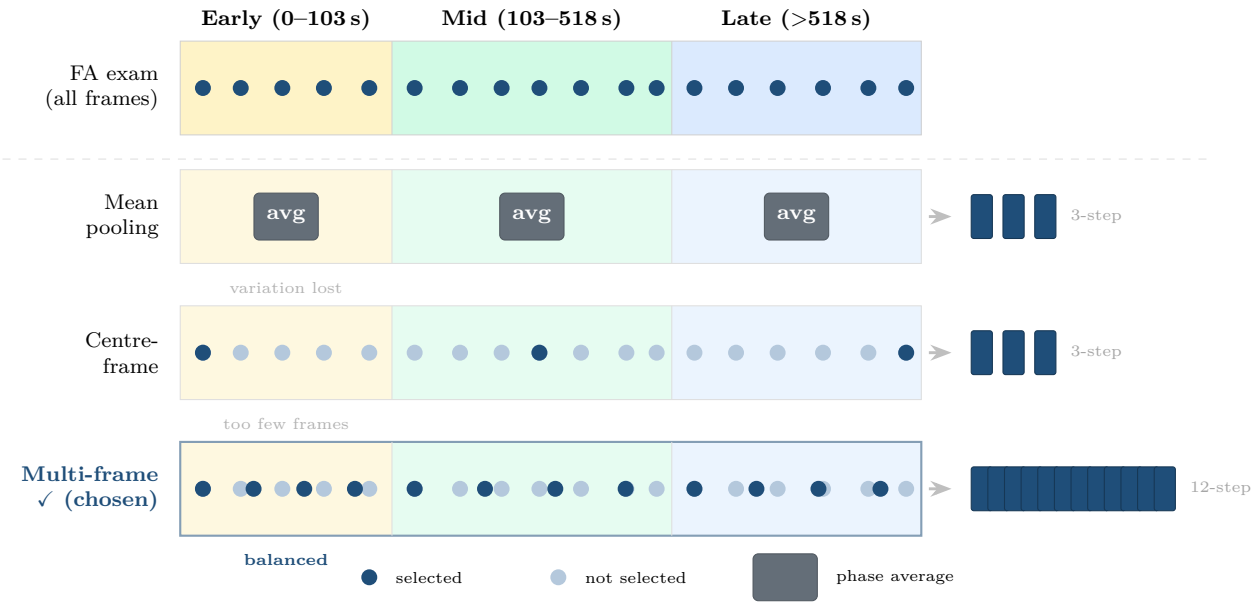


Figure 8.2: Three temporal binning strategies applied to the same FA examination. *Mean pooling* collapses all embeddings per phase into one vector, losing within-phase temporal dynamics. *Centre-frame* selects a single representative per phase, producing only 3 embeddings. *Multi-frame selection* draws four uniformly spaced frames per phase, yielding a 12-step sequence that preserves temporal progression while staying within the shared cluster's memory budget.

8.2.3 OCR Timestamp Recovery

Temporal modelling requires elapsed-time metadata for every frame. In the HOJG clinical dataset, this information is available in structured DICOM metadata fields. In the Aptos public dataset, which was used as the development sandbox, timestamps are burned into the frame image ribbon as pixel text but are not available as structured data. To recover these timestamps, a design decision was needed about which OCR approach to use. Three options were considered.

Local OCR model (e.g. Tesseract). This would keep all processing local, but required setup and tuning for the specific font and layout of Aptos frame ribbons. Given that the Aptos dataset carries no patient risk, the local-only constraint was not a hard requirement for this data.

Manual annotation. Accurate but completely infeasible at scale given the number of frames in the dataset.

Gemini Vision API [2] (chosen approach). A cloud-based vision model that required no local setup and produced reliable results on the Aptos ribbon format. Since the Aptos dataset is public and contains no patient-identifiable information, sending frames to an external API introduced no privacy risk.

The critical design decision was not the tool choice itself, but the architectural scoping of the OCR path. The Gemini API integration exists exclusively in the Aptos data preparation pipeline. It is not importable or callable from the HOJG data loader. For HOJG, timestamps are read directly from the DICOM metadata field, so no frame pixel data is ever sent to an external service. This scoping is enforced at the code level, not by policy: a developer cannot

accidentally invoke the OCR service on clinical images by misconfiguring a flag, because the code path simply does not exist in the HOJG data loader. This directly supports security and privacy by design.

Validation of the OCR output was performed against a manually annotated evaluation set: I went through 150 frames myself, checked the elapsed timestamps, and stored the results in a `.json` file so that the output of any OCR technique could be measured against it.

8.3 Test Strategy

The test strategy validates the pipeline at two levels: structural correctness of the implementation, and evaluation of model performance against the single-frame baseline.

8.3.1 Evaluation Metrics

The design was aligned with evaluation from the start so that temporal models could be compared fairly to the single-frame baseline. All configurations use the same fixed train/validation/test split defined by `hyperftype.json` on the Idiap project storage. The held-out test set contains 211 examinations. Primary metrics are ROC-AUC and F1-score.

These metrics were not chosen simply to match the baseline. ROC-AUC validates the model's fundamental ability to correctly rank cases across all possible thresholds, independent of the decision boundary chosen at inference time, making it suitable for comparing models before a clinical operating point has been chosen. F1-score grounds the evaluation in the clinical reality by balancing precision and recall at a definitive decision boundary. Standard multiclass averaging can allow high-volume mild cases to mask poor performance on clinically critical rare classes; F1-score specifically penalises models that default to predicting the majority class, which is critical given the dataset's severe class imbalance. Metric selection was not inherited blindly from the baseline: borrowing metrics without justification is a fast track to misleading results.

8.3.2 Structural Validation: Prototype Script

Before any integration into MedNet, a self-contained prototype script (`04_DESIGN/PROTOTYPE-TEMPORAL-PIPELIN` `PY`) was used to validate end-to-end tensor flow and sequence reshaping. The prototype constructs synthetic exam tensors, applies temporal selection, simulates backbone output at the correct embedding dimension (384), executes recurrent sequence processing through the GRU, and verifies output shapes at every stage. This prototype caught an early reshaping mismatch before integration, reducing debugging cost later. The prototype therefore served as a concrete debuggability support step.

8.3.3 Two-Layer Unit Test Strategy

Design validation also included a deliberate two-layer unit test strategy implemented as focused test modules inside the MedNet library. The split into two files reflects a structural choice: model-level bugs and data-level bugs tend to fail silently in different places, so they benefit from being caught by separate test modules rather than a single end-to-end script that could mask one failure with another.

Model layer: `MEDNET/TESTS/TEMPORAL/TEST\_VITGRU.PY`. These tests validate the freeze and unfreeze policy for backbone blocks, confirming that only the intended layers become trainable when a partial unfreeze is requested. They also run a full forward pass through a mocked `timm` [16] backbone to verify that phase embeddings, variable-length masking, and the GRU all wire together correctly without a shape mismatch. The output shape and numerical finiteness

of the logits are both asserted, catching silent failures in the temporal tensor plumbing that a training loss curve alone would not surface cleanly.

Data layer: `MEDNET/TESTS/TEMPORAL/TEST\_SEQ\_ANGIOREPORT.PY`. These tests cover three distinct concerns: (1) that the frame index selection logic handles boundary conditions correctly (empty sequences, requesting more bins than available frames, requesting a single frame); (2) that the split parser accepts both the legacy flat format and the newer temporal JSON format without loss of information; and (3) that the custom collate function pads variable-length sequences consistently, with lengths and phase identifiers all aligned correctly across a batch. The third concern is especially high-risk because a silent padding error would corrupt the GRU's sequence signal without raising any exception.

Together, these two modules cover the critical paths most likely to introduce regressions during refactoring, and they run quickly enough to be executed before any downstream training job is launched. As described in Chapter 5, both modules are wired into MedNet's GitLab CI pipeline via a path-triggered `tests-temporal` job, converting them from a one-time validation into a standing regression guard.

8.4 Design Review and Iteration

This section documents how the design was reviewed against requirements before realisation began, what changed as a result, and how changes were verified.

What was reviewed. The design document (`04_DESIGN/DESIGN-DECISIONS.PDF`), including the architecture, the temporal binning strategy, and the choice of recurrent encoder, was presented to supervisor André during the week 3–4 meeting. The review focused on whether the proposed pipeline would preserve meaningful temporal signal through the aggregation and encoding stages.

Feedback received. **Feedback received — supervisor André, week 3–4 meeting**

> *Mean-pooling frames within a bin risks averaging away exactly the temporal variation the model is intended to learn. The recommendation was to use a single representative frame per bin instead and to evaluate both strategies systematically rather than assuming one is always better.*

The recurrent encoder choice was also discussed. After a conversation about the short sequence length (12 timesteps) and the shared-cluster memory constraints, the decision was made to switch from an LSTM to a GRU, which provides comparable sequence modelling capacity with fewer parameters.

What changed as a result. The default aggregation strategy was changed from mean pooling to multi-frame selection (four frames per phase). Mean pooling was retained as a comparison baseline for the experimental evaluation, but the pipeline's default path was updated. The recurrent encoder was changed from LSTM to GRU. Both changes preserved the tensor interface between pipeline blocks, so no upstream or downstream modifications were required.

8.5 Security, Privacy and Deployment

8.5.1 Security and Privacy

The ultimate goal of this project is to apply the temporal pipeline to the HOJG dataset, which contains real, identifiable patient images protected by Swiss data-protection law and the

GDPR. This data cannot leave Idiap's secure network. The design addresses this constraint through architectural separation, not through policy alone.

The pipeline enforces a strict public/private separation at the data-loading interface. The temporal data loaders, the model, the training loop, and the evaluation infrastructure are all dataset-agnostic: they accept any split file conforming to the `temporal_hyperf_type.json` schema. No component in these shared layers contains dataset-specific code paths or external API calls.

The Gemini Vision API OCR integration, used to recover timestamps from the public Aptos dataset, exists exclusively in the Aptos data preparation module. It is not importable or callable from the HOJG data loader. For HOJG, timestamps are read directly from the hospital's own DICOM metadata fields, so no frame pixel data is ever sent to an external service. By making this boundary a matter of code architecture rather than convention, the design ensures that no clinical image data can reach an external service, even through misconfiguration.

8.5.2 Deployment and Portability

In this research context, deployment does not mean shipping a container image to a production server. It means integration into the MedNet framework so that the temporal pipeline becomes available to the MedAI group as a shared, reproducible, and maintainable tool.

The choice to contribute directly to MedNet on a dedicated `temporality` branch (pending review and merge), rather than building a standalone repository, was a deliberate design decision. A standalone repository would have been faster to build, but it would have required future researchers to maintain a separate environment and learn a separate codebase. By integrating into MedNet, the temporal pipeline inherits MedNet's shared evaluation guarantees, its reproducibility infrastructure, and its visibility to the whole MedAI group.

The dataset-agnostic design described in Section 8.1 also directly supports deployment portability. Running the same pipeline on the HOJG dataset requires only a new split file that conforms to the same JSON schema. The model architecture, training loop, and evaluation code remain identical. This enables the clinical validation step that follows this internship without requiring any rewrite of pipeline components.

8.6 Quality Characteristics

Table 8.1 maps each quality characteristic to how the design supports it. The full rationale is documented in `04_DESIGN/DESIGN-DECISIONS.PDF`.

Table 8.1: Quality characteristics and how the design supports each one.

Characteristic	**How the design supports it**
Reproducibility	Fixed split file defines train/val/test partition; backbone frozen and embeddings computed once and cached; all configurations share the same held-out test set of 211 examinations.
Scalability	Cost scales with phase count (currently 3), not with raw frame count per examination. The 12-embedding fixed sequence keeps the GRU within shared-cluster memory limits.
Interpretability	Phase-level bin structure makes it possible to discuss which temporal phase contributes most to a prediction, unlike a model consuming all frames at once.
Compatibility	Backbone and classification head are unchanged from the single-frame baseline; the temporal block is a drop-in layer between them. Running on HOJG requires only a new split file.
Debuggability	Fixed 12-step sequences make tensor shapes predictable. Prototype script validates shapes end-to-end before integration. Two-layer unit test strategy catches model-level and data-level bugs separately.
Security & Privacy	Public/private dataset segregation at the data-loader interface. External OCR architecturally scoped to public data only; HOJG loader has no code path to invoke an external service on clinical images.

Chapter 9

Realisation

The realisation competency is not only about producing working code. It also requires showing that the implementation follows the design, connects to existing systems, uses existing frameworks correctly, and is made available in a way that can be tested, reused, and evaluated. For this reason, this chapter describes both the technical functionality of the final pipeline and the development process that led to it.

9.1 From exploratory downstream script to integrated pipeline

The first working version of the temporal model was implemented in `05_Realisation/downstream/run_downstream.py`. This script allowed the core modelling idea to be tested in isolation: take precomputed frame embeddings, arrange them as short temporal sequences, pass them through a GRU [5], and classify the examination from the final hidden state. The script supports two sequence strategies — a `first_last` strategy and a uniform binning strategy — with the GRU receiving sequences of embedding vectors rather than raw images. It uses padding and `pack_padded_sequence` so the model can process variable-length sequences without treating padded frames as real observations, then maps the final GRU hidden state to five HyperF_Type classes using a linear classification head.

This script was an important first step because it proved that the temporal modelling idea could run end-to-end. However, it was still exploratory software: configuration choices such as the model name, sequence strategy, number of bins, hidden size, and learning rate were all defined directly in the script, making it useful for fast experiments but unsuitable as a reusable pipeline.

The final temporal pipeline improved on this by moving the idea into the MedNet project structure. Instead of only consuming precomputed embeddings, the integrated pipeline starts from image folders, groups frames by examination, loads full frame sequences, preprocesses them, applies the ViT backbone per frame, passes the resulting frame embeddings through a GRU, and outputs class probabilities for the whole examination. This is the essential difference between the two versions: the first tested the temporal head; the second implements the full image-to-prediction pipeline.

9.2 Implemented system

A thorough description of the final pipeline is provided in `05_Realisation/temporal_pipeline`, written so that anyone continuing this research can make good use of the code and reproduce the results.

The pipeline is more reusable than the exploratory script because responsibilities are separated across the project structure. Grouping frames by examination is handled by the temporal data code. Frame loading and sequence construction are handled by the raw data loader. Variable-length batching is handled by the custom collate function. Model transforms and hyperparameters are placed in configuration files. The ViT-GRU model contains the actual

image-to-sequence-to-classification logic. The experiment runner handles training, prediction, evaluation, and output management.

This structure improves maintainability in concrete ways. If a future experiment needs a different frame selection strategy, only the data layer changes. If a different backbone or GRU size is needed, only the model configuration changes. If evaluation needs to be repeated, the output folder already contains all settings, checkpoints, logs, predictions, evaluation files, and metric plots.

The pipeline uses existing frameworks throughout rather than reinventing standard components. PyTorch [10] is used for tensors, batching, padding, packing sequences, the GRU, and loss computation. Torchvision handles preprocessing and augmentation. Timm [16] provides the ViT backbone. MedNet serves as the surrounding training and evaluation framework, keeping the implementation consistent with the existing Idiap codebase and easy for others to inspect and extend.

9.3 Making available: a two-repository deployment structure

9.3.1 Framework contribution: MedNet branch `temporality`

The core temporal components — the sequential data loaders, the ViT+GRU model class, and the enriched `temporal_hyperftype.json` split file — are implemented directly inside the MedNet framework on a dedicated branch:

https://gitlab.idiap.ch/medai/software/mednet/-/tree/temporality

This GitLab instance is internal to Idiap and requires an account; readers without access can consult the complete branch contents, which are included as a file directory in `05_Realisation/mednet-temporality/`.

```
src/mednet/config/temporal/data/angioreport/hyperftype_seq.py
src/mednet/config/temporal/data/angioreport/temporal_hyperftype.json
src/mednet/config/temporal/models/vitgru.py
src/mednet/data/temporal/hyperftype_split.py
src/mednet/data/temporal/seq_angioreport.py
src/mednet/models/temporal/model.py
src/mednet/models/temporal/vitgru.py
tests/temporal/test_seq_angioreport.py
tests/temporal/test_vitgru.py
```

Figure 9.1: Files added to the MedNet framework on the `temporality` branch. All contributions are contained within `temporal/` subdirectories, keeping them isolated from existing pipelines. The split between `config/`, `data/`, `models/`, and `tests/` mirrors MedNet's own internal structure, making the new components immediately recognisable to other researchers in the group.

This branch follows MedNet's internal code conventions and is maintained separately from the mainline in dedicated `temporal/` directories, so that changes can be reviewed and validated without destabilising the pipelines of other researchers. Once the implementation has been confirmed to align with MedNet's in-house format standards, it will be merged into the main branch.

At that point, the temporal Aptos pipeline becomes a first-class part of the shared group infrastructure, inheriting all of MedNet's evaluation guarantees rather than remaining a personal experiment repository.

Working inside the framework at this depth also led to a contribution that has already been merged upstream: while validating caching behaviour, I found and fixed a defect in MedNet's `CachedDataset` that loaded every sample twice in parallel mode (merge request `!79`, closing issue `#102`). The fix passed the full merge-request pipeline (2,604 tests) and was reviewed and merged into `main` by the framework maintainer. The diagnosis, the measured impact, and the review process are described in Section 5.2.2.

The `temporal_hyperf_type.json` file contributed through this branch transforms the raw Aptos dataset into an enriched, temporally structured, and cleaned dataset, adding validated timestamps, phase labels, and examination-level groupings. This is a reusable data artefact: any future researcher working with the Aptos dataset in the MedNet ecosystem can use this file directly, without repeating the OCR extraction and quality filtering work.

```
{
  "train": [
    {
      "exam": "Train/Train/2_L",
      "label": 0,
      "frames": [
        { "file": "0.jpg", "elapsed_seconds":   58.0, "elapsed_display":  "0:58.00"
↪  },
        { "file": "1.jpg", "elapsed_seconds":   59.0, "elapsed_display":  "0:59.00"
↪  },
        { "file": "2.jpg", "elapsed_seconds":   88.0, "elapsed_display":  "1:28.00"
↪  },
        { "file": "3.jpg", "elapsed_seconds":  798.0, "elapsed_display": "13:18.00"
↪  },
        { "file": "4.jpg", "elapsed_seconds":  801.0, "elapsed_display": "13:21.00"
↪  },
        { "file": "5.jpg", "elapsed_seconds": 1045.0, "elapsed_display": "17:25.00"
↪  }
      ]
    },
    {
      "exam": "Train/Train/2_R",
      "label": 1,
      "frames": [ ... ]
    },
    ...
  ],
  "val":  [ ... ],
  "test": [ ... ]
}
```

Listing 1: Excerpt from `temporal_hyperf_type.json`. Each examination entry groups its frames in acquisition order and attaches a validated elapsed time to every frame. The `label` field carries the HyperF_Type class (0–4). This schema is the single source of truth for all temporal training runs and transfers to any new dataset without code changes.

During integration, the framework's parallel data-loading path exhibited counter-intuitive behaviour: enabling parallelism *doubled* loading time rather than reducing it. Tracing the call stack revealed that `CachedDataset` submitted every sample via `executor.submit` and then immediately re-submitted the same samples via `executor.map`, causing each image to be loaded and transformed twice (Figure 9.2).

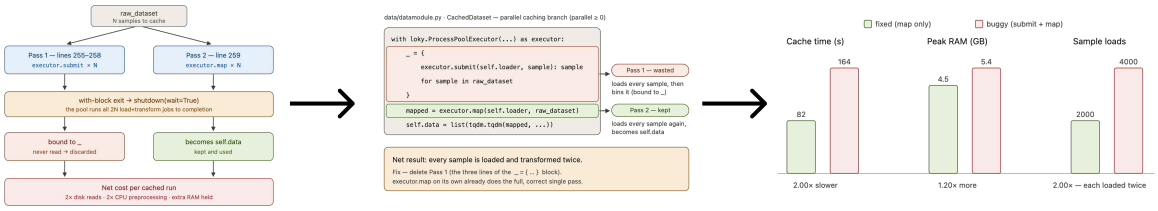


Figure 9.2: Root-cause analysis of a double-load bug in MedNet's parallel caching path. Removing the redundant `submit` pass (Pass 1) halved wall-clock cache time, reduced peak RAM by 1.2×, and eliminated the duplicate sample loads.

The fix — removing the three lines comprising the redundant `submit` pass — was contributed back to the `main` branch of Mednet.

9.3.2 Experiment reproduction repository: `beyond-the-last-frame`

To make the specific experimental results reported in this project fully reproducible, a separate self-contained repository documents the exact commands, configuration files, and environment setup required to reproduce the ViT+GRU HyperF_Type experiments. The README walks through five concrete steps: installing dependencies via `pixi`, configuring the dataset path for MedNet, placing the RETFound-Green checkpoint, running the full temporal fine-tuning experiment, and generating the interactive HTML error analysis report. Each step corresponds to a single copy-pastable command, so a new researcher can go from a clean environment to a complete set of evaluation results without any undocumented manual steps.

The output folder structure — containing settings, checkpoints, logs, predictions, evaluation metrics, and the misclassification HTML — makes each run fully auditable afterwards.

This separation was deliberate. The MedNet branch is the professional contribution to the group's shared infrastructure; the reproduction repository is the accountability artefact that proves the results are real and reachable. Together they satisfy both the integration requirement (connecting to an existing system) and the deployment requirement (making the work available in a form others can actually use). The reproduction repository itself cannot be shared externally as it is the property of Idiap.

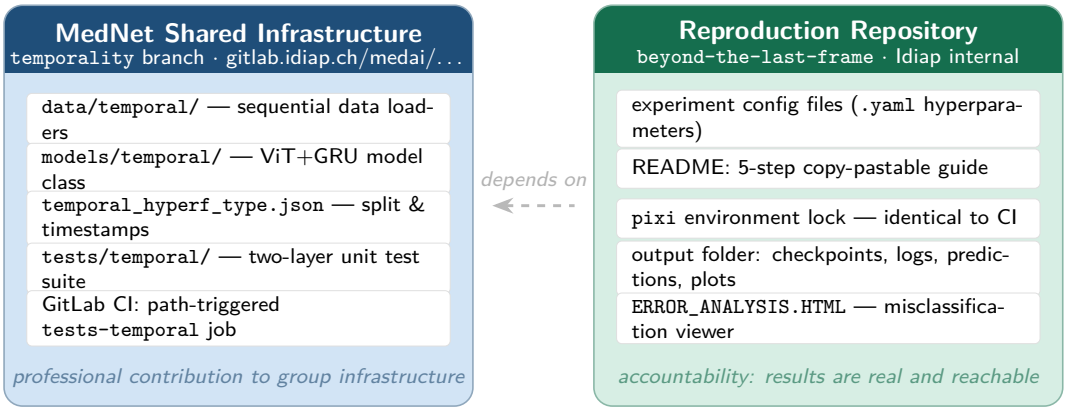


Figure 9.3: Two-repository deployment structure. The MedNet `temporality` branch is the professional contribution to the group's shared infrastructure, providing reusable data loaders, the ViT+GRU model, and CI-tested evaluation components. The `beyond-the-last-frame` repository is the self-contained accountability artefact: it depends on the MedNet branch and documents the exact steps to reproduce every experimental result reported in this portfolio.

The README (included at `05\_Realisation/README.md`) documents every configurable option as a flag on the experiment entry point, requiring no code edits to reproduce or extend any result:

Table 9.1: Configurable flags in the reproduction repository. Every experimental variant reported in this portfolio corresponds to a specific combination of these flags.

Choice	**Flag**	**Values (default bold)**
Frame selection	`-frame-span`	**phase_stratified**, uniform, last
Temporal aggregation	`-temporal-pooling`	**last** (GRU), attention (softmax pool)
ViT patch read-out	`-global-pool`	**avg**, max, token (CLS)
Trainable depth	`-backbone-unfreeze-last-n`	0–12 (**12** = full fine-tune)
Phase boundaries	`-bound-early-mid` / `-bound-mid-late`	**103 s** / **518 s**

Running the default experiment (phase-stratified frames, full backbone fine-tune) requires a single command:

```
pixi run seq-experiment -- \
  --modality experiment \
  --device cuda:0 \
  --pretrained-weights retfoundgreen_statedict.pth \
  --output-folder results/temporal-hyperftype
```

Each run writes a `settings.json` recording the exact configuration, making every result independently auditable.

9.4 Testing and evaluation readiness

The implementation was validated at three levels.

Level 1 — exploratory validation. `05_Realisation/downstream/run_downstream.py` showed that the GRU-based temporal classifier could train and evaluate on precomputed embeddings. This gave a fast way to test temporal modelling choices before integrating them into the larger pipeline.

Level 2 — structural validation. The final temporal pipeline was validated by tracing the complete flow from split JSON to grouped exam folders, sequence loading, batching, preprocessing, ViT embedding extraction, GRU classification, prediction, and evaluation output. The generated output folder makes each run auditable because it stores all important files and plots.

Level 3 — test automation. The two-layer unit test strategy designed in Chapter 8 was implemented as described, with both modules residing in the MedNet library. The tests confirmed that the model-level and data-level components behaved correctly in isolation before any expensive downstream training was launched.

These tests are wired into MedNet's GitLab CI pipeline via a dedicated `tests-temporal` job in `.gitlab-ci.yml`. The job is path-triggered: it runs automatically whenever changes touch `src/mednet/data/temporal/`, `src/mednet/models/temporal/`, `src/mednet/config/temporal/`, or `tests/temporal/`, with a `when: manual` fallback for on-demand runs. Execution uses `pixi`

`run -e test-ci-alternative` — the same environment as local development and the reproduction repository — so there is no environment drift between where code is developed and where CI verifies it.

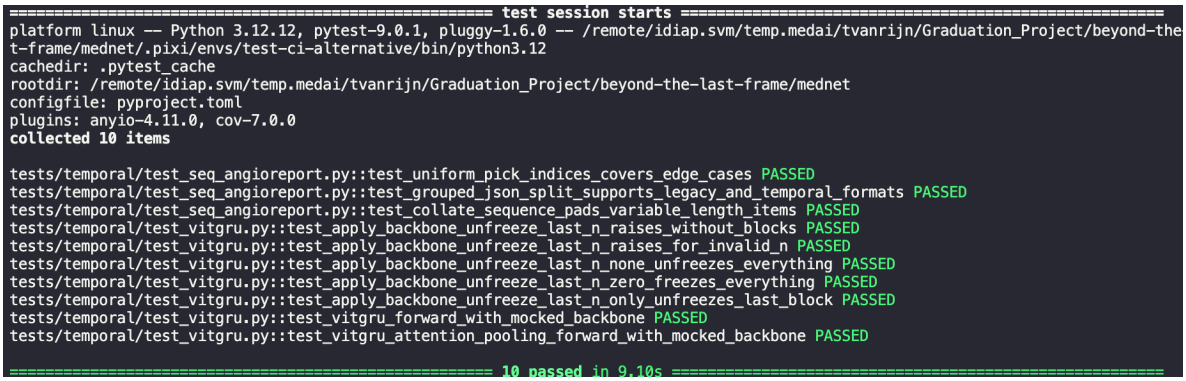


Figure 9.4: All temporal unit tests passing locally via `pixi run -e test-ci-alternative`, the same environment used by the GitLab CI `tests-temporal` job. The test names map directly to the model-layer and data-layer concerns described in Chapter 8.

Together, the structural validation, the two-layer unit tests, and the CI job establish that the implementation is engineered correctly: the pipeline runs end-to-end, components behave as specified in isolation, and regressions are caught automatically before they reach the mainline.

For the Realisation competency, the important result is that the designed temporal architecture was implemented as a working, reusable, and evaluable software pipeline — and that it delivers. The experimental results documented in Chapter 4 confirm this: the ViT+GRU fine-tuned pipeline achieves a macro HyperF_Type AUC of 0.87, a five-point improvement over the ROC-AUC 0.82, demonstrating that the implemented system fulfills the design requirements and earns its place against an already-capable reference.

9.5 Reflection

The main lesson from this realisation phase is that working code is not automatically good professional software. The first downstream script was valuable because it quickly tested the idea, but it mixed configuration, data preparation, model definition, and training all in one file. That made it fast to modify but difficult to scale or hand over to someone else.

The final pipeline is stronger because it is integrated into the existing MedNet structure and separates the main responsibilities cleanly. It extends the project from single-frame classification to exam-level temporal classification while still using the existing training and evaluation infrastructure, meaning the work is immediately accessible to the rest of the group rather than being a private experiment.

This is what makes it stronger evidence for the realisation competency: it is not just an experiment that runs, but a pipeline that can be reused, inspected, tested, and evaluated by others.

Chapter 10

Conclusion and Reflection

This project set out to answer one question: does looking beyond the last frame of a fluorescein angiography examination lead to better automated grading? The answer is yes, but only under one specific condition. Simply feeding more frames into a frozen image model produces no reliable improvement. Training the image model and the sequence model together, so they co-adapt, improves grading across every disease category tested, with the largest gains on Pooling ($0.76 \rightarrow 0.83$) and Window Defect ($0.78 \rightarrow 0.85$) — exactly the conditions defined by how fluorescence behaves over time rather than what it looks like at the end.

The pipeline is integrated into the MedAI group's shared framework, validated on a public benchmark, and ready for clinical evaluation on hospital patient data. The SAIMI 2026 abstract acceptance and the OMIA13 submission confirm that the findings are considered novel and relevant beyond Idiap.

Personally, the most important shift during this internship was learning to engage proactively in a research environment rather than waiting to be invited to contribute. The most technical lesson was that negative results — the frozen experiments, the PCA study — are not wasted work. They are exactly what makes the positive result interpretable. Without them, the claim that temporal modelling helps would be much harder to defend.

Chapter 11

Deliverables

The graduation folder is named `Graduation_Project_Tymo_van_Rijn` and is organised into six numbered folders, one per competency. The contents of each folder are described below.

01_Professional_Skills_&_Manage_Control Contains three files: the highlights evidence document with annotated screenshots of communication and collaboration moments (Word document); the process logbook covering the full internship period (PDF); and the SAIMI 2026 poster draft (PDF).

02_Analysis Contains the bulk of the analytical evidence. This includes the Aptos data exploration notebook, the timestamp analysis script, and a separate data exploration notebook (project source code); the OCR benchmark analysis, literature review on temporal encoding, LSTM bin-count evaluation, MedNet framework analysis, PCA–GRU embedding study, Temporal Separability Benchmark study, TSB study proposal, timeline distribution analysis, timestamp metadata quality report, and studied concepts reference document (all PDF); the interactive model error analysis viewer (HTML); five analysis figures (images); and six weekly progress slide decks from week 1 through week 12 (PDF).

03_Advise Contains the week 4 options matrix advisory, the revised temporal encoding advisory, the MedNet framework advice presentation, and the TAM presentation (all PDF); and the Mattermost screenshot documenting the information-loss advice exchange with Roberto (image).

04_Design Contains the design decisions document and the pipeline architecture document (PDF); and the prototype tensor-shape validation script (project source code).

05_Realisation Contains the temporal pipeline documentation (PDF); and the exploratory downstream script that was the first working proof of concept (project source code).

References

[1] Asia Pacific Tele-Ophthalmology Society. *3rd APTOS Big Data Competition: AngioReport Fluorescein Angiography Dataset*. Online dataset. Public benchmark used for hyperfluorescence-type classification. Referred to in this work as "Aptos" or "AngioReport". 2023.

[2] Google DeepMind. *Gemini 2.5 Flash*. https://deepmind.google/technologies/gemini/. Used for OCR-based recovery of per-frame timestamps from Aptos imagery. 2025.

[3] Y. Zhou, M. A. Chia, S. K. Wagner, M. S. Ayhan, D. J. Williamson, R. R. Struyven, T. Liu, M. Xu, M. G. Lozano, P. Woodward-Court, Y. Kihara, A. Altmann, A. Y. Lee, E. J. Topol, A. K. Denniston, D. C. Alexander and P. A. Keane. "A foundation model for generalizable disease detection from retinal images". In: *Nature* 622.7981 (2023), pp. 156–163. DOI: 10.1038/s41586-023-06555-x.

[4] J. E. Wang, T. MacGillivray and M. O. Bernabeu. "Training a high-performance retinal foundation model with half-the-data and 400 times less compute". In: *arXiv preprint* (2024). arXiv: 2405.00117 [eess.IV]. URL: https://arxiv.org/abs/2405.00117.

[5] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[7] V. Amiot, O. Jiménez-del Toro, Y. Guex-Croisier, M. Ott, T.-E. Bogaciu, S. Banerjee, J. Howell, C. Amstutz, C. Chiquet, C. Bergin, I. Meloni, M. Tomasoni, F. Hoogewoud and A. Anjos. "Automatic transformer-based grading of multiple retinal inflammatory signs in uveitis on fluorescein angiography". In: *American Journal of Ophthalmology* (2025). Predecessor single-image grading work from the MedAI group at Idiap. DOI: 10.1016/j.ajo.2025.04.027.

[8] I. Tugal-Tutkun, C. P. Herbort, M. Khairallah and the Angiography Scoring for Uveitis Working Group (ASUWOG). "Scoring of dual fluorescein and ICG inflammatory angiographic signs for the grading of posterior segment inflammation (dual fluorescein and ICG angiographic scoring system for uveitis)". In: *International Ophthalmology* 30 (2010), pp. 539–552. DOI: 10.1007/s10792-008-9263-x.

[9] R. Pulvirenti. "Single-frame deep learning for grading retinal inflammation on fluorescein angiography". Internal MSc thesis, MedAI group, Idiap Research Institute. 2024.

[10] A. Paszke, S. Gross, F. Massa et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. URL: https://pytorch.org.

[11] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin. "Attention is All You Need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017.

[13] R. A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[14] H. Hotelling. "Analysis of a Complex of Statistical Variables into Principal Components". In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[15] MONAI Consortium. *MONAI: Medical Open Network for AI*. https://monai.io. Discussed as an alternative ecosystem to MedNet in Sec. 7.6. 2020–.

[16] R. Wightman. *PyTorch Image Models (timm)*. https://github.com/huggingface/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.