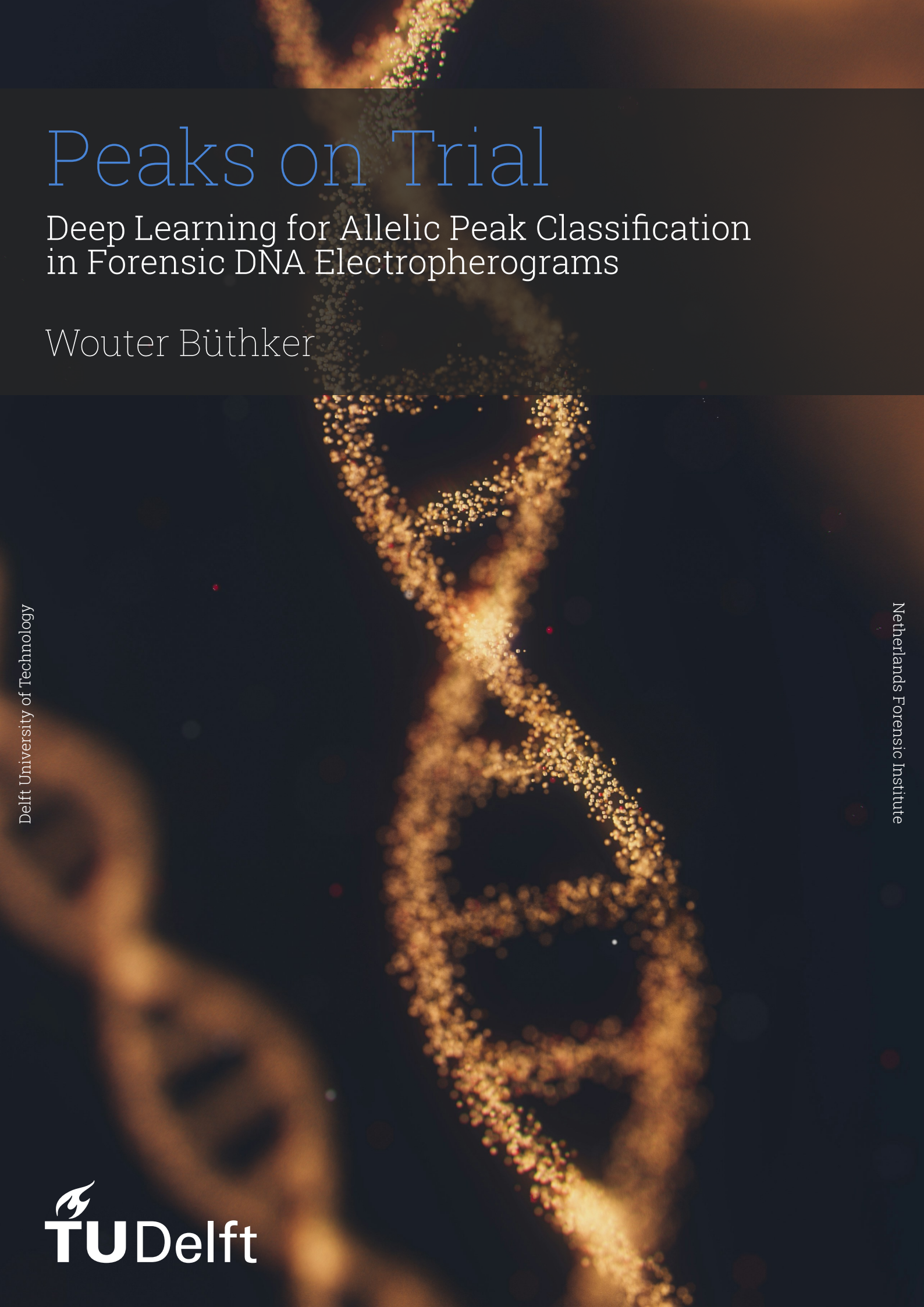
Peaks on Trial

Deep Learning for Allelic Peak Classification in Forensic DNA Electropherograms

Wouter Büthker



Delft University of Technology

Netherlands Forensic Institute

TU Delft

Netherlands Forensic Institute
Ministry of Justice and Security

Peaks on Trial

Deep Learning for Allelic Peak Classification in Forensic DNA Electropherograms

by

Wouter Büthker

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday 23 February, 2025 at 10:00

Document version:	February 16, 2026	
Project duration:	June 17, 2025 – February 23, 2026	
Cover Image:	By ANIRUDH on Unsplash	
Thesis committee:	Assoc. prof. dr. ir. T. Abeel,	TU Delft, Thesis Advisor
	Asst. prof. dr. ir. T. Höllt,	TU Delft, Thesis Committee Member
	Dr. ir. R. Ypma,	Netherlands Forensic Institute

An electronic version of this thesis is available at `https://repository.tudelft.nl/`.

TU Delft

Abstract

Allele calling is a critical step in forensic DNA analysis, and better automation could increase throughput and consistency in casework. However, few studies systematically compare machine-learning architectures for allele calling, and limited high-quality training data constrain progress.

We designed and evaluated a peak-based allele-calling pipeline that makes peak-level classifications rather than profile-level segmentations. The pipeline comprises three models: the Peak Model, Autoencoder Model, and Combined Model; and we evaluated them on datasets with ground-truth annotations and with imperfect analyst annotations. We compared performance against state-of-the-art baselines.

The Combined Model outperformed DNANet on NFI research data with ground-truth annotations (pixel F1 0.934 vs. 0.923; $p = 0.001$). Ablation experiments showed that each component contributed to performance. Autoencoder pretraining improved accuracy when training data were scarce (fewer than 1,000 DNA profiles). Error analysis further indicated that small peaks are hardest to classify as allelic (true DNA) versus artefactual compared with medium and high peaks.

Overall, peak-based classification improves allele-calling performance over current models and clarifies key failure regimes, bringing fully automated allele calling closer to forensic deployment.

Contents

List of Figures

List of Tables

1 Introduction

1.1. Motivation and context

More than 80 000 violent crimes were committed in The Netherlands in 2022 [22]. Forensic science applies methods from chemistry, biology, physics, and digital analysis to turn trace evidence such as fibres, fingerprints, DNA, and electronic data into court-admissible proof. Beyond supporting criminal investigations, forensic evidence can also prevent wrongful convictions.

One of the most powerful tools for identifying people is DNA analysis. It is considered the "gold standard" in forensic science [28]. In a typical workflow, biological material collected at a crime scene is processed in a laboratory, yielding a DNA profile represented as an electropherogram (EPG). An electropherogram is measured in relative fluorescence units (RFU) and consists of multiple one-dimensional signals, with each peak corresponding to either an allele or an artefact. Alleles are highly variable between individuals and can be used to identify individuals or their families. Alleles are found at specific locations, or loci, on human DNA. Electropherograms are analysed and reviewed by forensic analysts, who distinguish the true alleles from measurement noise and other artefacts. This step, of deciding which peaks represent the true genetic signal and which are artefacts, is called *allele calling*. Finally, the resulting profile is compared against the (inter)national databases or other samples, and a potential match can be used as an investigative lead or be quantified for use in court.

A critical practical challenge is that allele calling is time-intensive and often involves manual review and rule-based decision-making, which can introduce variability between analysts and limit scalability. This indicates a clear need for computational methods that improve consistency and throughput while maintaining the reliability requirements of forensic casework. Recent machine learning approaches suggest that data-driven interpretation may be feasible, motivating a closer examination of the current state of the art and its remaining limitations [51–53, 57, 62].

Automating parts of this workflow is challenging because the electropherogram contains measurement noise and artefacts, and decisions ultimately feed into investigative leads and may contribute to court reporting. Consequently, computational methods must improve efficiency while respecting the reliability requirements of forensic casework.

Assumed background knowledge This thesis assumes knowledge of standard deep learning concepts and foundational forensic DNA and capillary electrophoresis concepts (e.g., alleles, loci, STR kits). Readers who are new to forensic genetics or DNA analysis are strongly recommended to read appendix A: *Biological and forensic foundations* before continuing.

1.2. State of the art and literature synthesis

This section synthesises the current state of automated electropherogram analysis, with a focus on machine learning. We first review existing tools and research methods for detecting alleles and artefacts. We then summarise the neural architectures used for electropherograms and, because research in this field is limited, we also consider closely related peak-based signals to inform design choices. Finally, we compare available datasets and discuss how differences in STR kits, laboratory conditions, and labels constrain training and benchmarking.

1.2.1. Current methods for automated electropherogram analysis

Automating electropherogram analysis aims to enhance reproducibility, reduce manual workload, and standardise interpretation across laboratories. While rule-based expert systems and probabilistic genotyping software represent the current operational standard, they often require extensive parameter tuning and manual review for complex mixtures [14].

Compared to adjacent domains such as analytical chemistry and biomedical signal processing, research on machine learning for electropherogram analysis remains limited. A substantial part of the deep learning literature on electropherogram interpretation originates from the work of Taylor et al., including early demonstrations of neural-network-based allele calling [51, 53, 57]. Importantly, this line of work formulates the interpretation of electropherograms as a segmentation problem, in which every scan point is classified. Subsequent studies have investigated model behaviour across conditions and datasets [56, 60], and have addressed practical requirements such as explainability [25] and the scarcity of high-quality training data [54]. More recently, Wit et al. introduced a novel deep learning approach that likewise adopts a segmentation-based formulation, aiming to classify every scan point in an electropherogram [62].

Besides these efforts, several studies have explored automation for electropherogram interpretation using alternative (non-segmentation) strategies [2, 10, 35, 36]. However, many of these approaches are evaluated under narrow experimental settings, and the field has not yet converged on broadly applicable tools that can be reliably deployed across different laboratory conditions [8, 47]. Finally, there are also commercial software solutions for automated electropherogram interpretation. An example is AutoAnalyzer, which is currently in use in production systems [13].

A difficulty in assessing progress is that results are often not directly comparable across publications. Differences in materials and laboratory conditions, variation in STR kits, and the use of private datasets strongly affect both model performance and reported metrics. As noted in section A.1, measurements of the same DNA sample can vary substantially across STR kits, which complicates cross-study benchmarking. Public data availability further limits comparison: the most substantial public dataset (PROVEDIt) contains profiles generated across multiple kits and experimental conditions, often forcing researchers to select subsets aligned with their use case or to combine public data with private collections. As a result, evaluation settings differ substantially between studies. Section 1.2.3 further explains the implications of differences in datasets.

1.2.2. Neural network architectures for electropherogram analysis and similar domains

This subsection provides an overview of neural network architectures applied to electropherogram (EPG) analysis and to closely related signal processing tasks where peak detection, classification, and denoising are central. As discussed in section 1.2.1, the literature on deep learning for electropherogram analysis is relatively small compared to adjacent domains. For that reason, in addition to electropherogram-specific work, we also consider evidence from analytical chemistry and biomedical signal processing, where the data often consists of 1D traces with peak-like shapes and where model design choices are frequently driven by similar constraints (noise, baseline drift, artefacts, and limited labelled data).

To motivate this comparison, Figure 1.1 shows example traces from electrophoresis (EPG), high-performance liquid chromatography (HPLC), and electroencephalography (EEG). Electrophoresis and HPLC both exhibit localised peaks on a baseline, making them comparable in terms of detection and peak shape. At the same time, electrophoresis and EEG share structural similarities in how they are commonly represented and processed: both involve multiple simultaneously recorded signals (e.g., dye channels in EPGs, electrode channels in EEG), where dependencies between channels can be informative. Across these domains, there is a recurring focus on identifying relevant events (peaks or artefacts), distinguishing true signal from noise, and producing outputs that are stable enough for downstream decision making.

Deep learning approaches have largely superseded traditional machine learning in complex time-series classification tasks due to their capacity for automatic feature extraction [29, 33]. Within the specific context of electropherogram analysis, Tan et al. [50] compared five machine learning methods: Random Forest, Logistic Regression, Gaussian Naive Bayes, Support Vector Machine, and Multilayer Perceptron (MLP); demonstrating that Multilayer Perceptrons achieved superior performance. However, the authors note: "due to the limited sample size, the observed performance differences between the models may not reach statistical significance" [50]

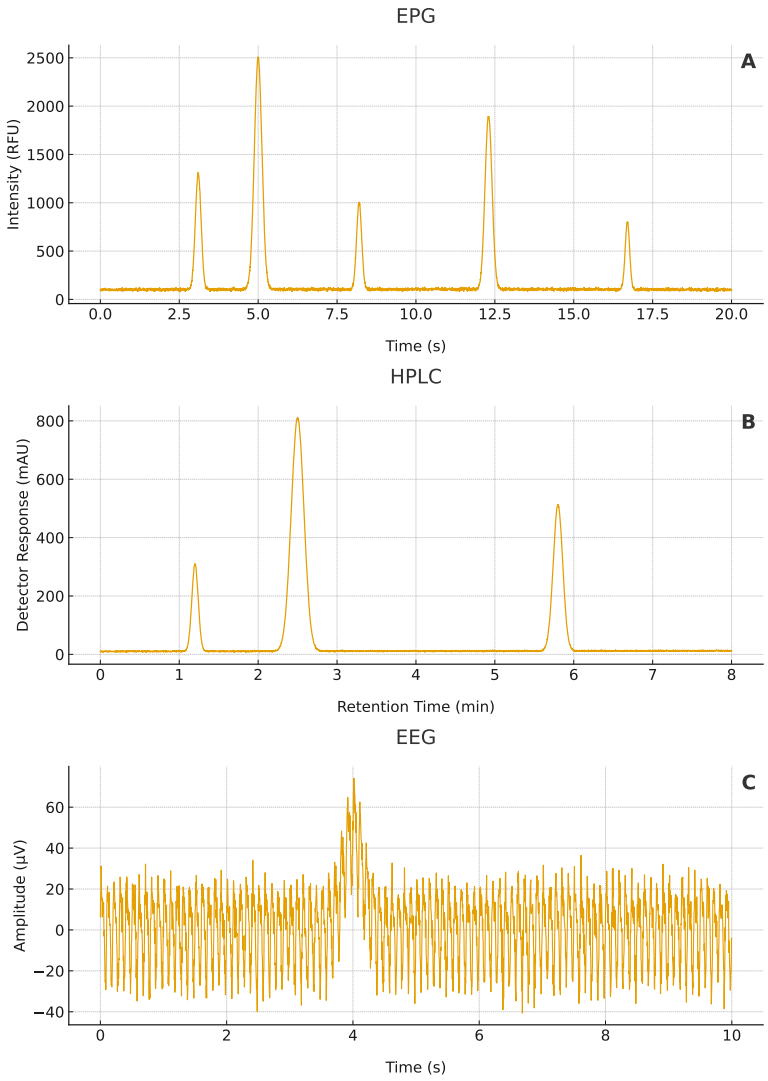


Figure 1.1: Example data from electrophoresis (A), compared to high-performance liquid chromatography (B) and electroencephalography (C) data samples.

Multilayer perceptrons

MLPs are among the earliest neural models applied to electropherogram interpretation. Taylor and Powers use an MLP with a single hidden layer as an initial demonstration of neural-network-based allele calling [57]. In subsequent work, Taylor et al. split the input profile by dye channel and marker to train multiple independent MLPs [53]. This decomposition simplifies the learning problem and aligns with how electropherograms are conventionally interpreted, but it also introduces practical costs: training and maintaining several separate models increases complexity and requires additional computational power, and MLPs can become parameter-inefficient.

Convolutional neural networks

Convolutional neural networks (CNNs) are a natural next step for electropherogram traces because peaks are local structures and many relevant cues are expressed as local patterns in the signal. Taylor propose a CNN-based model for electropherogram analysis, although their implementation was constrained by the desire to transfer learned weights to the earlier MLP-based system, and parts of the processing pipeline remained separated by dye and marker [51]. In contrast, Wit et al. [62] processes an entire profile using a single CNN, adopting a U-Net-style architecture to produce signal-level predictions and to capture both local peak morphology and broader contextual information along the trace [62]. Similar CNN-based approaches are widely used in peak-centric signal domains such as analytical chemistry [9,

19, 32, 38] and biomedical signal processing [5, 21]. In analytical chemistry in particular, it is common to separate peak detection from peak classification [19, 32, 43].

Autoencoders and representation learning
A major practical bottleneck in electropherogram analysis is the scarcity of high-quality labels. Autoencoders are relevant in this setting because they allow representation learning from unlabelled traces by training the model to reconstruct its input [1]. Learned latent representations can then be used as features for downstream tasks such as peak classification or quality assessment [1, 49]. Autoencoders are also used as denoising components, where the model is trained to recover clean signals from corrupted inputs, which can be useful when noise and baseline artefacts dominate the raw trace [7, 30]. Finally, variational autoencoders introduce a probabilistic latent space, which can be beneficial when modelling uncertainty or generating realistic variations of observed signals [1]. These properties make autoencoders an attractive direction when labelled electropherogram data is limited or when robustness to noise is a primary concern.

Recurrent neural networks
Recurrent neural networks (RNNs) are designed for sequential data and have a long history in biomedical time-series processing. In peak-based datasets, however, RNNs are less common than CNNs because many decisions are dominated by local morphology rather than long-range temporal context. Still, RNN-based models are used in EEG processing to detect and correct signal artefacts [7, 31], and recurrent layers can in principle be used to stabilise predictions over time by incorporating local context from neighbouring samples. In the electropherogram literature, Taylor [51] explicitly recommend investigating recurrent models [51].

Transformers
Transformers are best known for natural language processing, but attention-based models have also shown strong performance on other sequential data with long-term dependencies [9, 34]. Furthermore, transformers have shown strong results for sequential biomedical data, including tasks such as arrhythmia classification from ECG signals [5]. In principle, attention mechanisms could be valuable for electropherogram analysis because they allow the model to condition predictions on non-local context, potentially capturing interactions between peaks, channels, or marker regions. Hybrid models that combine CNNs and Transformers are increasingly common in related domains, with CNNs capturing local structure and Transformers modelling longer-range dependencies [21]. At the same time, Transformers typically require substantial training data and careful regularisation, which can be challenging in forensic settings where labelled electropherogram data is scarce and where robustness across kits and laboratory conditions is required.

Conclusion
Overall, the current electropherogram literature demonstrates that both peak-level and signal-level neural modelling approaches are feasible, with CNN-based architectures appearing particularly well aligned with peak-centric traces and segmentation-style outputs [62]. At the same time, the broader signal processing literature suggests multiple promising directions when labels are limited, or robustness is critical, including representation learning with autoencoders and hybrid architectures that combine local feature extractors with sequence-level context. These considerations inform the architectural choices explored in this thesis and help position them relative to existing approaches.

1.2.3. Available DNA data and implications
Automated genotyping and DNA mixture interpretation require large, well-annotated mixture datasets for development and validation, yet only a few such collections are publicly available.

Several practical barriers limit data sharing. First, DNA profiles are sensitive: even after pseudonymisation, profiles can enable re-identification or reveal familial relationships [27, 61]. Second, generating controlled mixture collections is expensive and slow, especially when targeting broad coverage of contributor counts, mixture ratios, STR kits, and laboratory conditions. Third, annotation is labour-intensive. Linking each electropherogram to donor genotypes and mixture composition requires careful bookkeeping and expert review, and differences in protocols, equipment, and STR kits reduce cross-study comparability. As a result, many studies rely on private datasets or combine small public subsets with internal collections, which complicates benchmarking and external validation [51, 62].

Table 1.1 summarises the datasets most relevant to this thesis. They differ in scale, experimental design (mixture ratios and replicates), laboratory setup (STR kits and capillary electrophoresis (CE) equipment), and label availability (ground-truth and analyst annotations). These choices determine which training objectives are feasible and how confidently results transfer across studies.

Table 1.1: Comparison of available datasets

Aspect	PROVEDIt [3]	NFI research data [12]	NFI casework data	GAN [54]
Scale & Scope	25,164 total profiles: 5,160 mixtures; 144 lab conditions;	354 accepted profiles; 120 mixtures, 3 replicates per mixture	70,722 profiles; mostly mixtures	Potential unlimited profiles, trained on 1,078 profiles
Contributors per Sample	1 to 5 contributors, across all ratio combinations (up to 99:1).	2–5 contributors (20 discrete mixture ratios).	20-40% has a single contributor; max 5 contributors; various ratios	Training data includes 0–5 contributors; output supports arbitrary contributor counts (examples up to 10 [55])
Amount of Donors	119 donors (69 DEM and 50 WBM donors)	30 donors (DEM)	Mostly unique donors	-
STR Kits & Loci	PowerPlex 16 HS (16 loci); Identifiler Plus (16 loci); GlobalFiler™ (24 loci).	PowerPlex Fusion 6C (27 loci).	PowerPlex Fusion 6C (27 loci)	GlobalFiler™ (24 loci)
Capillary Electrophoresis Equipment	ABI 3130 & ABI 3500	ABI 3500xL	ABI 3500xL	ABI 3500
Degradation & Inhibition	70% profiles synthetically compromised	No degradation, all high-quality extracts.	Profiles might be naturally compromised	Can be partially specified
Replicates	Single PCR per sample	Triplicate PCR for each mixture	Standard 1; ~8% triple replicates	Can generate multiple outputs per sample via different random secondary inputs.
Ground truth annotations	Yes	Yes	No	Must be specified
Analyst annotations	No	Yes	Yes	No
Multiclass annotations	No	Yes (not for replicates)	Yes (for approximately 600 profiles)	No
Publicly available	Yes [3]	Yes, as part of DNANet [62]	No	Code and weights available on request

In general, data availability is not a single bottleneck but a trade-off between scale, representativeness, and label quality. PROVEDIt offers many profiles and substantial variation in conditions [3], but it includes STR kits that differ from those used at the Netherlands Forensic Institute (NFI). PROVEDIt

therefore supports studies of domain shift and cross-kit generalisation, but performance estimates may not carry over to other kits.

In contrast, the NFI research dataset is smaller [12], but includes both ground-truth labels and analyst annotations. This combination enables controlled evaluation and analysis of agreement between model predictions and expert interpretation. Replicates further support consistency analyses, although the limited scale restricts coverage of rare conditions.

The NFI casework dataset provides scale and reflects operational variability, but it is not publicly available and lacks ground-truth annotations. This setting is valuable for capturing real-world signal quality and artefacts, including degradation and inhibition, but incomplete ground-truth labels limit fully supervised training and prevent strong claims about absolute allele-calling accuracy. Methods that rely less on dense labels become more relevant when the goal is robust decision support rather than fully automated interpretation.

Synthetic data can partially address label scarcity and improve coverage of rare but important conditions. For example, Taylor and Humphries use a generative adversarial network (GAN) to simulate electropherograms, but their evaluation focuses on a single STR kit (GlobalFiler), which limits conclusions about cross-kit use [54]. Mešić report modest gains from incorporating synthetic data [39], suggesting that augmentation can help, but that its impact likely depends on how well generated traces reproduce kit- and laboratory-specific characteristics.

Appendix D further illustrates the differences between STR kits, including variations in locus coverage and measurement characteristics, which are particularly relevant when interpreting generalisation results across datasets.

1.3. Research gaps, problem statement and research questions

Despite progress in automated electropherogram analysis, key barriers still limit reliable, deployable electropherogram interpretation: robustness on difficult profiles, generalisation across kits and laboratory conditions, scarce ground truth, and the step from peak labels to allele calls. This section summarises these gaps, defines the problems addressed in this research, and translates them into the research questions that structure the remainder of this work.

1.3.1. Overview of research gaps

Despite recent advancements, several key challenges remain in the automated analysis of electropherograms. The field has made substantial progress in peak classification and allelic peak prediction in recent years, but numerous gaps remain that hinder widespread deployment of these methods. These challenges include data availability, the ability to generalise across different laboratory conditions and STR kits, and performance issues when handling complex mixtures or degraded samples. Additionally, while the potential for self-supervised learning exists, it has not been thoroughly explored in the context of electropherogram analysis. This section outlines the primary gaps that remain in the field.

Robustness on challenging profiles

Performance for determining whether peaks are allelic is acceptable for many routine profiles [51, 62], but it degrades in challenging settings, such as profiles with many low-RFU peaks. Existing studies suggest that this drop partly comes from a mismatch between the training data and the rare, difficult casework profiles. This motivates the use of data augmentation [51]. Synthetic generation has been proposed as a means to simulate these challenging profiles via GAN-based methods [54], but evidence for consistent improvements across realistic evaluation settings remains limited [39]. This gap motivates research into architectures and training objectives.

Generalisation across kits and laboratory conditions

Electropherogram traces depend on the STR kit and laboratory-specific factors such as amplification settings, instrument configuration, and analysis parameters. Table 1.1 illustrates how loci differ between commonly used kits, which implies that even when the underlying DNA is the same, the observed signal distribution can shift substantially. Current deep learning approaches are typically trained and validated within a single kit and a narrow range of conditions, limiting interchangeability between laboratories and making deployment costly whenever protocols change. The central gap is therefore not only achieving high performance on one site, but developing methods and validation strategies that

can tolerate realistic domain shift (across kits, instruments, and lab conditions) without requiring full re-labelling and end-to-end retraining for every setting.

Dataset scarcity and benchmarking constraints
Progress is constrained by the small number of publicly available STR mixture resources with usable ground truth for supervised evaluation, with PROVEDIt being a key example [3]. Even when data exist, differences in STR kits, equipment, annotation practices, and task definitions complicate meaningful cross-paper comparisons, undermining progress and making it difficult to interpret claims about state-of-the-art performance. Recent work explicitly calls for standardised benchmarks and evaluation practices to support reproducible comparison [62], but creating such benchmarks requires methods that can be evaluated between kits and conditions rather than within a single data silo. This gap motivates research into learning paradigms that can use partially labelled or unlabelled corpora, as well as protocols for evaluation under domain shift.

From peak labels to allele calling
While several approaches focus on classifying peaks as allelic versus artefactual [51, 62], the downstream step of allele calling remains difficult to perform with open methods at a level of practicality comparable to commercial software. In this step, continuous size measurements and peak information must be converted into discrete allele designations per locus (e.g., 10/12 at D5S818), and errors here directly impact the interpretability and downstream use of the profile. Current open approaches rely on heuristics for binning [62], suggesting a gap for learning-based allele calling. Even when the immediate goal is not end-to-end genotyping, this gap matters because peak-level predictors are only operationally useful if they integrate into a reliable allele calling workflow.

Estimating the number of contributors
Before interpreting a mixture, analysts typically assess the number of contributors (NoC), which conditions downstream mixture interpretation and affects the plausibility of competing explanations. Recent work shows that NoC estimation can be supported by machine learning methods, including explainable approaches and specialised models designed for forensic mixtures [11, 55, 59]. However, performance and usability gaps remain, particularly regarding support for PCR replicates, robustness analysis, and explanations that remain stable under realistic noise and artefact conditions [55]. This motivates further work on NoC estimation that is not only accurate but also accompanied by sensitivity analyses that make limitations explicit to practitioners.

Explainability as actionable evidence
Deep learning methods can be applied to electropherogram interpretation; however, limited transparency remains a significant barrier to their adoption in routine casework, where analysts must justify their decisions and document uncertainty. To date, dedicated work on explainability for electropherogram-focused deep learning remains limited [25]. Explainability, therefore, needs to operate at the level of actionable evidence: how sensitive the decision is to baseline noise and common artefacts, and how confidence behaves across loci, contributors, and replicates. A further gap is the lack of standardised practices for reporting explanations and uncertainty alongside performance, which makes it hard to compare methods not only by accuracy, but also by the degree to which they support defensible forensic reasoning.

1.3.2. Problem statement: improving allele calling with novel architectures and self-supervised pretraining

Interpreting electropherograms into discrete allele calls is a critical and time-consuming step in forensic DNA profiling. Although recent deep learning methods can classify peaks as allelic or artefactual with strong performance under standard conditions, evaluated model architectures for this task remain limited. In addition, ground-truth-labelled training data is scarce, and labels available in practice can be incomplete or noisy, limiting the development and evaluation of robust models.

This thesis addresses these limitations by designing a novel neural network architecture for allele calling from electropherograms and evaluating its performance in comparison to existing approaches. A secondary objective is to investigate whether self-supervised pretraining on electropherogram traces can improve allele calling performance when high-quality labels are limited or unreliable.

1.3.3. Research questions

The following research questions were established based on these research gaps and opportunities.

1. How do different machine learning model architectures perform with respect to prediction performance in analysing electropherograms?
2. Does self-supervised pretraining on electropherograms improve allele calling performance versus training using potentially low-quality labels?

1.4. Solution and contributions

Automated allele calling must distinguish allelic peaks from artefacts in STR electropherograms despite noise, mixture complexity, and limited high-quality annotations. Prior learning-based work often frames allele calling as a segmentation problem over the full signal. In this research, we instead pursue a peak-based formulation: we first extract candidate peaks and then classify each candidate peak as allelic or non-allelic. This structure enables explicit peak-level decisions while allowing the model to incorporate both local peak morphology and global profile context.

1.4.1. Proposed solution

We develop three machine learning models that progressively incorporate more context:

1. **Peak Model (local context).** We extract candidate peaks from an electropherogram and classify each peak from a fixed-width window around the peak. This model tests whether local peak morphology and local neighbourhood cues suffice for accurate allele calling.
2. **Autoencoder (global representation).** We train an autoencoder on full electropherograms to learn a compressed latent representation without requiring allele labels. The latent code aims to preserve global profile properties that cannot be inferred from local windows alone.
3. **Combined Model (local + global context).** We integrate local peak features with the learned global representation by combining the Peak Model encoder with the pretrained autoencoder encoder. This model tests whether global profile context improves allele calling beyond peak-only classification.

Together, these models directly address **RQ1** by comparing architectures that use only local context versus those that incorporate global context. They address **RQ2** by enabling self-supervised pretraining of the global branch and measuring its impact on downstream allele calling.

1.4.2. Contributions

This thesis makes the following contributions:

- **Novel peak-based allele calling architectures.** We introduce a Peak Model for per-peak classification and a Combined Model that integrates local peak information with a learned global electropherogram representation.
- **Self-supervised global context for allele calling.** We demonstrate how an autoencoder trained on unlabelled electropherograms can provide a reusable global representation and quantify when pretraining improves allele calling performance.
- **Comprehensive validation and benchmarking.** We benchmark the proposed models against state-of-the-art approaches and perform ablation studies to justify key architectural components and preprocessing choices.
- **Analysis of data characteristics that drive errors.** We investigate how peak height and locus identity affect performance, identifying low-RFU peaks as a particularly challenging regime and quantifying locus-dependent differences.

These contributions collectively provide an empirically grounded answer to the research questions and clarify both the potential and the current limitations of machine-learning-based allele calling.

1.5. Thesis outline

Chapter 1 motivates automated allele calling from STR electropherograms in a forensic setting. It reviews existing electropherogram analysis pipelines and neural network approaches, then identifies the limitations imposed by current methods and available annotated data. The chapter closes by formulating the problem statement, identifying research gaps, outlining research questions, and detailing the contributions of this thesis.

Chapter 2 describes the task formulation and the full experimental workflow. It details the data sources, annotation procedure, and train/test split strategy, and documents the preprocessing pipeline, including newly introduced preprocessing steps. It goes in depth on methodology shared across models, including evaluation metrics, baseline definitions, and statistical testing, to support reproducibility. Finally, the chapter then defines the three models: Peak (classification), Autoencoder (reconstruction), and Combined (segmentation); and specifies their architectures and training protocols.

Chapter 3 reports the empirical findings in a message-driven structure. It first compares high-level architectural choices, then evaluates the Peak Model against DNANet under ground-truth evaluation. It then analyses locus-embedding behaviour and the relation between locus properties and performance, and evaluates reconstruction quality of the learned latent code relative to fixed-basis compression. Next, it quantifies the effect of global profile context and self-supervised pretraining on allele calling performance, and analyses when model rankings depend on the label source and evaluation target. Finally, it studies peak height (RFU), identifying low-RFU peaks as a shared failure regime across methods, and closes with a general discussion of implications, limitations, and future directions.

Chapter 4 summarises the main contributions and findings, answers the research questions, and outlines practical implications and future work.

The appendices provide supporting and reproducibility material: biological and forensic foundations (appendix A), ethical considerations (appendix B), full experimental results (appendix C), an STR kit comparison (appendix D), and complete hyperparameters and software/package versions (appendix E).

2

Materials and methods

2.1. Goal and problem formulation

Let $\mathbf{x} \in \mathbb{R}^{C \times T}$ denote an input DNA profile signal (electropherogram) sampled over T pixels and C dye channels. In this work, $C = 6$ for PowerPlex Fusion 6C (PPF6C), and profiles were resampled to a fixed length $T = 4096$ as described in section 2.2.3.

A deterministic, non-trainable peak extraction routine $g(\cdot)$ was applied to obtain a set of detected peaks

$$\mathcal{P} = g(\mathbf{x}) = \{p_i\}_{i=1}^{N},$$

where each peak p_i was represented by (i) its centre location τ_i (pixel index), (ii) its dye channel c_i, (iii) an extracted peak-centred window $\mathbf{x}_i \in \mathbb{R}^{C_p \times W}$ (see section 2.2.4), and (iv) its locus l_i.

Given $\mathbf{x}$ and $\mathcal{P}$, the proposed architecture produced three outputs:

- A peak-processing branch embedded each detected peak window as $\mathbf{h}_i = f_{\text{peak}}(p_i)$ and predicted a per-peak class distribution $\hat{\mathbf{y}}_i = f_{\text{cls}}(\mathbf{h}_i)$ for each $p_i \in \mathcal{P}$.
- An autoencoder branch mapped $\mathbf{x}$ to a latent representation $\mathbf{z} = f_{\text{enc}}(\mathbf{x})$ and reconstructed the input as $\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z})$.
- A combined model fused global signal information and per-peak information to predict per-peak class distributions

  $$\hat{\mathbf{y}}_i = f_{\text{comb}}\left(\mathbf{z}, \{\mathbf{h}_i\}_{i=1}^{N}\right),$$

  and, by converting these predictions back to a sparse segmentation representation, produced a segmentation-style output compatible with pixel-level evaluation and segmentation baselines (see section 2.2.4).

We focused on per-peak allele/background classification and did not address downstream allele calling or likelihood ratio computation in this research.

Training proceeded in three stages. First, we trained and evaluated the standalone peak model (f_{peak}, f_{cls}). Second, we trained the autoencoder (f_{enc}, f_{dec}) without annotations. Third, we trained a combined model in which the fusion module f_{comb} refined per-peak predictions using the autoencoder latent $\mathbf{z}$ and the peak embeddings $\mathbf{h}_i$.

2.2. Data preparation and preprocessing

2.2.1. Data acquisition and train/test splits

As discussed in section 1.2.3, four datasets were available for this research (see also table 1.1). We used two datasets: NFI casework data and NFI research data [12]. Only profiles generated with the STR kit PowerPlex Fusion 6C (PPF6C) were included.

The NFI casework dataset included 70 722 profiles with analyst annotations. The dataset reflected the variation encountered in forensic practice (e.g., varying contributor numbers, mixture ratios, degradation, and artefacts). We partitioned this dataset into an 80/20 train/test split using a deterministic, hash-based file assignment. This split could include multiple measurements from the same unknown donor in both

the train and test sets, because a file represents a single measurement of a sample. Therefore, we may experience data leakage, where the same donors appear in both the training and test sets. Since donors in our data are unidentified, we are unable to split by donor. We consider this risk acceptable given the wide variety of donors and the large number of profiles in the dataset.

After splitting into train/test, each subset was split into four approximately equal parts (A, B, C, and D) based on the physical machine that was used. We decided to keep this split to aid in loading subsets into memory during training and preprocessing. We assume the data distribution across the different machines is equal.

The NFI research dataset [12] contained 350 profiles with both true ground-truth labels and analyst annotations. The dataset was provided in six parts with independent donors. We used five parts for training and one part for testing to prevent donor leakage. The sixth subset was always used for testing.

Privacy, storage, and access controls. DNA profiles are personally identifiable under EU law. Pseudonymisation was applied to reduce identifiability while preserving suitability for analysis; no directly identifying information (names or comparable identifiers) was included with the profiles. The data were stored and processed in accordance with Dutch and European laws. The NFI enforced a policy that data remained on-premises, and access required completing NFI data safety procedures. As a TU Delft researcher, access required an AIVD security screening (Verklaring Geen Bezwaar)[1]. Additional ethical considerations and risk mitigation related to the use of casework data are discussed in appendix B.

2.2.2. Data annotation and labelling

The NFI casework data lacked true ground-truth genotypes. Therefore, we used analyst annotations. Analysts provided two annotation regimes based on thresholds: Low Threshold (LT) and Analytical Threshold (AT) (see table A.1). Peaks below the relevant threshold were not annotated. We primarily used LT annotations where available, because they reduced the number of ambiguous "unlabelled" peaks that could be either allelic or artefactual. However, only a fraction of casework profiles included LT annotations, and restricting to LT profiles could bias the training distribution (LT profiles contain fewer overloaded profile artefacts). Therefore, we also trained models using AT-annotated profiles in experiments that required broader coverage.

In the NFI research dataset, both ground-truth annotations and analyst annotations were available. To achieve the best possible performance and to fairly evaluate it, we used the ground-truth annotations. The only exception is when we evaluated against ground-truth annotations while simulating analyst annotations as a prediction model, aiming to evaluate the performance of human analysts.

Annotation format and conversion to pixel labels. All analyst annotations were stored as allele calls at specified loci (e.g., allele numbers per locus). We converted these allele lists into a pixel-level segmentation label representation using an existing algorithm [62]. In this representation, the top pixel of each allelic peak was marked as (1) and all other pixels were marked as (0). Exact implementation details can be found in the source code[2].

Multi-class annotations. During this research, we contributed to a third annotation type ("multi-class annotations"). These annotations were manually created by experts under a protocol different from the analyst annotation workflow: allelic peaks below thresholds were included, and non-allelic peaks were labelled by artefact class (e.g., stutter, spectral bleedthrough/pull-up). Peaks were annotated as full peak regions rather than only peak tops. Approximately 600 profiles were annotated, split across the casework and research datasets. We consider these annotations to be of much higher quality than the analyst annotations and offer advantages over ground-truth annotations, such as insight into specific types of artefacts.

During this research, we contributed to the creation of this type of annotation. We contributed to the development of the annotation protocol and the tool used to create these annotations. When some annotations were available, we used these annotations to train DNANet to pre-annotate peaks and support subsequent manual annotation.

[1] https://english.aivd.nl/topics/security-screening/the-security-screening
[2] https://github.com/NetherlandsForensicInstitute/DNANet/tree/7a7b063bb821243ddc1b0632bc1ebfdd055eda5c

2.2.3. Pre-existing preprocessing pipeline

In this research, we build upon the DNANet codebase[3] created by Wit et al. [62].

The ABI 3500xL Genetic Analyzer produces `.HID` files containing electropherogram traces. Raw measurements have dtype `int16` and shape ((C, N)), where (C=6) and (N) is the number of scan points (typically around 9300). To prevent integer overflow during preprocessing, we loaded traces into `int32`, applied preprocessing, and then clipped back to `int16` after baseline subtraction. After preprocessing, the data is converted to `float32` for efficient GPU processing.

Baseline subtraction was applied immediately after loading, since downstream steps relied on fixed RFU thresholds (see section 2.2.4 for details of the baseline subtraction used for raw traces).

Internal standard validation and sizing. After baseline subtraction, we validated the internal standard and computed a mapping from scan points to base pairs. The 6th dye in a PPF6C electropherogram corresponds to the internal standard, which contains DNA fragments of known sizes. These fragments are used to translate from scan points to base pairs in size. In some cases, this requires a non-linear translation. Therefore, we applied a cubic spline for interpolation that passes exactly through every peak (>180 RFU) in the internal standard. This method is consistent with the techniques used in production by analysts [48][p. 20].

Resampling to a fixed length. We resampled each profile to a fixed shape ((C, 4096)). Pixel index (0) corresponded to base pair 65 and pixel index (4095) corresponded to base pair 475. This clipped primer flare outside the target range and yielded a fixed input size suitable for ML training.

Annotation parsing and allele-to-pixel mapping

We loaded and parsed the corresponding annotations and ladder information. Annotations were stored as allele numbers per locus (e.g., 13 or 15). We translated allele numbers to their expected bin locations in the resampled profile and created a pixel-level label representation by marking only the top pixel within the bin (see Wit et al. [62] for details).

The translation from allele to pixel location used a heuristic that could introduce errors, and this could also affect the inverse translation from predicted allelic pixels to allele numbers. Because of this limitation, our primary evaluation was performed at the pixel level (see section 2.3.1). Evaluating at the pixel level removes our reliance on this heuristic and can yield fairer, more accurate results. However, a limitation of this method is that it reduces the interpretability of results.

If any preprocessing steps failed (e.g., internal standard validation failure or missing annotations), we excluded the corresponding profile from the final dataset. When preprocessing was complete, we saved processed profiles to a disk cache to speed up subsequent data loading.

2.2.4. Newly introduced preprocessing techniques

This section describes preprocessing components introduced or modified in this thesis. Unless stated otherwise, these steps were applied after the core pipeline in section 2.2.3.

Peak extraction

We detected candidate peaks independently per dye channel (except in the internal standard) using a local-maximum criterion: any local maximum above a fixed RFU threshold (*threshold*) was considered a peak. Flat tops (plateaus) were treated as a single peak, placed at the centre index of the plateau (rounded down). We padded windows at boundaries with a constant value (0). We apply the peak selection threshold after baseline subtraction but before (log or linear) scaling.

For each detected peak (p_i) at centre location (τ_i) in dye channel (c_i), we extracted a fixed-width, peak-centred window from the electropherogram:

$$\{p_i\}_{i=1}^{N} = g(\mathbf{x}) = \mathrm{crop}(\mathbf{x}, c_i, \tau_i) \in \mathbb{R}^{C_p \times W}. \tag{2.1}$$

Here, (W) is the window width in pixels and ($C_p \in 1, 2$) depends on whether *include_max_pool_dyes* was enabled (see below).

We extracted annotations for peak windows by applying the same cropping procedure to the pixel-level annotation segmentation image.

[3] https://github.com/NetherlandsForensicInstitute/DNANet/tree/7a7b063bb821243ddc1b0632bc1ebfdd055eda5c

Cross-dye context for pull-up artefacts. To account for spectral bleedthrough (pull-up) artefacts, we introduced the hyperparameter *include_max_pool_dyes*. When enabled, we expanded the peak window to ($\mathbf{x}_i \in \mathbb{R}^{2\times W}$): the first channel retained the primary dye signal ($\mathbf{x}^{(c_i)}$), and a second channel contained the max-pooled signal over all other dyes: $\max_{k\neq c_i} \mathbf{x}^{(k)}$. This aggregated context captured high-intensity signals in other dyes that could indicate pull-up. It should be able to capture spectral bleedthrough, since it generally occurs only due to very high peaks in other dyes. Another option we considered was including all dyes in the window. A disadvantage of this is that the to-be-classified peak could become ambiguous without switching the dye order (dyes are ordered by wavelength). For this reason, and because the max-pool strategy should capture all important information, we decided it was most suitable for our use case.

Filtering peaks below annotation thresholds for analyst annotations Analyst annotations are threshold-dependent (LT/AT). Since peaks below the relevant threshold were not annotated, they may be ambiguous training targets, and we can not be sure they are not allelic, even though they are annotated as (0) background. We optionally enabled *filter_peaks* to exclude (unannotated) peaks below the threshold. The thresholds depended on the dye and the annotations. The exact thresholds used are shown in table A.1. This filtering is applied before any other preprocessing (such as scaling).

Raw vs analysed traces

`.HID` files contained both "raw" and "analysed" data streams alongside instrument parameters [6]. Prior work used the analysed data [62]. The analysed data appeared to include proprietary transformations (e.g., baseline subtraction, smoothing, clipping), but the manufacturer did not disclose the full processing pipeline, and we observed unexpected artefacts in some analysed traces. Therefore, we evaluated models trained on both raw and analysed streams as part of our ablation studies using the hyperparameter *data_loading_strategy*. For models originally trained on analysed data, we always supplied analysed data.

Raw traces often exhibited elevated baselines and baseline drift, which affected downstream steps that used fixed RFU thresholds (e.g., internal standard parsing). Therefore, when using raw traces, we applied baseline subtraction per dye using a rolling percentile method: for each scan point, we computed the 20th percentile within a sliding window of width 100 scan points and subtracted this baseline estimate. This approach is consistent with analyst workflows; more details about the algorithm (*Superior baseline subtraction*) can be found in the GeneMarker manual. [48, p. 19].

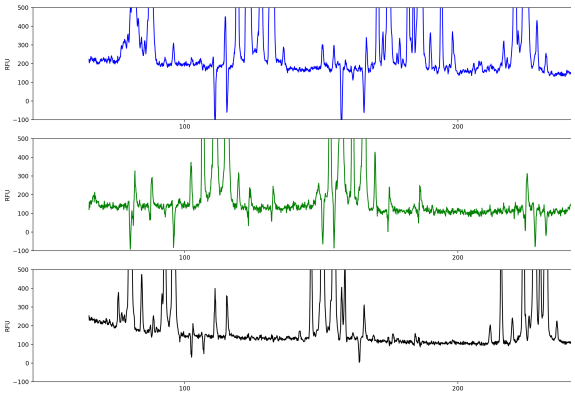
Scaling

Electropherogram signals covered a wide numeric range (see figure 3.5). To stabilise optimisation and reduce sensitivity to absolute RFU magnitude, we evaluated multiple input scaling strategies.

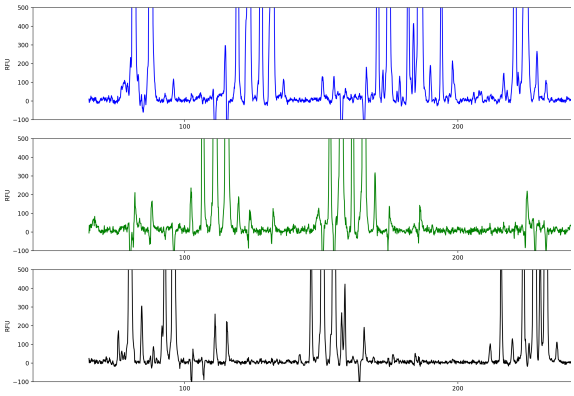
Per-profile and global standardisation were considered but not chosen due to the difficulty of these techniques with our large datasets. Global standardisation requires computations over the entire dataset (which does not fit in memory on our hardware), while per-profile standardisation requires bookkeeping of the per-profile scaling to undo it (which is important for our autoencoder model).

We additionally considered linear scaling and logarithmic scaling.

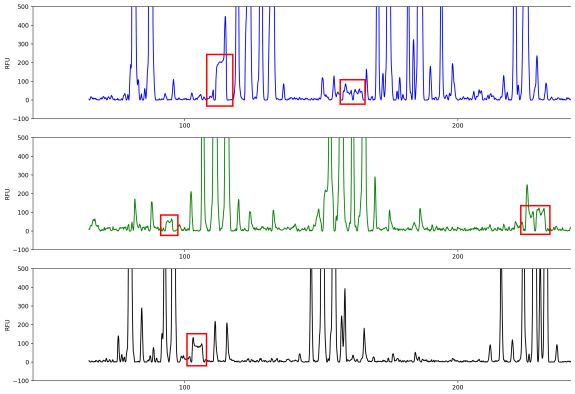
- **Logarithmic compression.** We applied log scaling to preserve relative differences at low RFU:
$$\hat{x} = \ln(1 + \max(x, 0))$$
- **Linear scaling with clipping.** We also evaluated linear scaling with clipping:
$$\hat{x} = \frac{\text{clip}(x, 0, max\_RFU)}{max\_RFU},$$
where *max_RFU* is treated as a tunable hyperparameter. We hypothesise that very high RFU peaks are almost always allelic and that the exact peak height provides no additional information beyond a certain threshold. Furthermore, some model architectures work best with values between 0 and 1.
- **Combined scaling.** We first clip x to $[0, max\_RFU]$, then apply logarithmic compression and normalize by $\ln(1 + max\_RFU)$. This method combines the advantages from the first two, by preserving differences at low RFUs, while also compressing values to (0, 1).
$$\hat{x} = \frac{\ln\big(1 + \text{clip}(x, 0, max\_RFU)\big)}{\ln(1 + max\_RFU)}$$



(A) Example snipped loaded using the raw data without any modification.



(B) Example snipped loaded using the raw data with baseline subtraction applied.



(C) Example snipped loaded using the analysed strategy. Several artefacts caused by this strategy have been highlighted. We can see that neighbouring negative peaks raise the area between them, appearing as an artefactual peak.

Figure 2.1: Example profile snippet showing different loading strategies. The y-axis is fixed from -100 to 500 RFU.

Smoothing

To examine the effect of noise reduction (and to approximate the smoothing observed in some analysed loading strategy profiles), we evaluated the smoothing of input traces.

We smoothed each dye channel of the electropherogram with a hard low-pass filter implemented with `numpy.fft.rfft` and `numpy.fft.irfft`. We define the hyperparameter *keep_fraction* as the amount of fraction of the spectrum to keep.

This method was chosen because it is consistent with analysts' production use. Details about the algorithm can be found in the GeneMarker manual [48, p. 19]. Analysts generally use a *keep_fraction* of 0.4 in production.

Training throughput optimisation and caching

To reduce the overhead of repeated preprocessing during training, we cached extracted and preprocessed peaks to disk, similar to the cached electropherograms caching method described in section 2.2.3.

In the combined model setting, training required both extracted peaks and full electropherograms, so we optimised peak extraction to reduce per-step overhead. We rewrote peak extraction to use PyTorch operations, enabling it to run on the GPU. We also replaced a brute-force locus lookup in the training loop with a precomputed lookup table, enabling (O(1)) locus assignment per peak. While no proper scientific experiments were done, these optimisations resulted in a speed-up of approximately 30x

Creation of segmentation image

To compare peak-wise classifiers to segmentation-based baselines, we converted per-peak predictions ($\{\hat{\mathbf{y}}_i\}_{i=1}^{N}$) into a sparse segmentation tensor ($\hat{\mathbf{s}} \in \{0,1\}^{C\times T}$).

We initialised ($\hat{\mathbf{s}} = \mathbf{0}$) and overwrote the peak-centre pixel locations ((c_i, τ_i)) for peaks predicted as allelic. For binary classification, we marked a pixel as allelic (1) when the predicted allele probability exceeded 0.5. Marking the pixel with the same probability as the peak might be a preferable strategy. Peaks cannot overlap, so we did not handle collisions.

This representation matched the evaluation strategy used in prior work [62].

2.3. Experimental setup and evaluation

2.3.1. Metrics

We evaluate all methods using metrics that align with the task. In forensic human identification, false positives and false negatives are similarly undesirable. We therefore use the F1 score as the primary metric.

F1 is the harmonic mean of precision and recall, so it penalises both false positives and false negatives equally. It ranges from 0 (no overlap between predictions and labels) to 1 (perfect agreement). We compute it as

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ F1 &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{2.2}$$

where (TP) is the number of true positives, (FP) the number of false positives, and (FN) the number of false negatives.

We ultimately want improvements in allele calling to translate into more informative likelihood ratios. However, assessing how allele-call errors propagate to likelihood ratios is outside the scope of this research. We could evaluate final allele calls against labels derived from analyst annotations or ground-truth genotypes. Unfortunately, in practice, allele-level evaluation is sensitive to the heuristic that maps alleles to pixels (section 2.2.3), potentially biasing metric estimates. We therefore report pixel-level F1 scores, comparing predicted pixel labels to available pixel annotations (even when those annotations may contain errors). Only when pixel-level annotations are unavailable, we report the allele-level F1 score.

In addition to the overall F1 score, we stratify performance by peak height (RFU). Peak height affects the strength of evidence in downstream likelihood-ratio calculations, so we prioritise correctly classifying high-RFU peaks. Finally, we report F1 per locus to quantify locus-specific differences in interpretation difficulty.

2.3.2. Baselines

We compared our models to DNANet [62], a state-of-the-art segmentation model on which this work builds. The DNANet code is publicly available[4]. We do not retrain the model and use the publicly available weights. We use the original train and test splits to prevent data leakage during evaluation.

We also compared to AutoAnalyzer (GeneMarker), a commercial tool used in production [13]. The internal method is proprietary and not fully disclosed. However, all software versions and settings necessary to reproduce our comparison are provided in the citation.

We used available analyst annotations to simulate a human analyst in our evaluations. This was only possible for the NFI research data, where both ground-truth and analysts' annotations were available. Details on the implementation are described in the DNANet paper[62].

We did not compare with the system of Taylor [51] due to technical limitations and because it was trained on a different STR kit, rendering a comparison unfair.

2.3.3. Statistical testing and significance

We compared per-profile performance scores across models using two-sided permutation tests and applied Holm correction across model pairs.

[4] https://github.com/NetherlandsForensicInstitute/DNANet/tree/7a7b063bb821243ddc1b0632bc1ebfdd055eda5c

For a model pair ((a,b)), we defined the statistic as the mean difference

$$\Delta(a, b) = \text{mean}(x - y),$$

where (x) and (y) were vectors of per-profile scores for models (a) and (b). A positive $(\Delta(a, b))$ indicated a higher mean score for (a).

We generated all unique unordered model pairs $((i, j))$ with $(i < j)$. For each pair, we ran a two-sided permutation test using SciPy's permutation test with (`permutation_type` = "samples") and (`vectorized` = `True`), and we recorded $((a, b, \Delta(a, b), p))$.

We corrected all raw p-values across pairs using Holm's procedure (via `statsmodels` `multipletests` with (`method` = "holm")) and reported corrected p-values. This can lead to overcorrection of p-values, since we do not report all pairwise comparisons.

2.3.4. Ablation methodology

We used ablation studies to evaluate the effect of individual parameters and architectural choices. We used 5-fold cross-validation with profile-level splits to assess stability across data partitions. For each fold, we trained each configuration twice with different seeds, yielding 10 runs per configuration.

Because collected performance metrics were non-normally distributed, we used non-parametric statistical tests. We used the Kruskal-Wallis test to assess overall differences among configurations and Dunn's post hoc test to identify specific pairwise differences. We applied Holm-Bonferroni correction across comparisons within a single experiment.

2.3.5. Implementation details

Experiments were run on the following system:

- GPU: NVIDIA RTX 4500 with driver 580.126.09
- CPU: Intel Xeon W-2225
- RAM: 64 GB
- OS: Ubuntu 22.04

We used Python 3.10.12. The full list of packages and versions is reported in section E.2.

For all models, we used the Adam optimiser. The *learning_rate* and *weight_decay* (L2 penalty) are specified in the hyperparameters (section E.1). We used $(\beta = (0.9, 0.999))$ and $(\epsilon = 10^{-8})$. We optionally used the PyTorch `ExponentialLR` scheduler (*use_scheduler*) with *gamma* specified in section E.1. When *use_evaluation_metric* was enabled, we early-stopped based on validation F1 for the peak and combined models and based on validation MSE for the autoencoder. Early stopping triggered when the validation metric did not strictly improve by more than *min_delta* for five epochs.

We used MLflow for logging and monitoring experiments. Unless stated otherwise, we used a 0.3 validation split in our experiments. Training times varied depending on the dataset size.

2.4. Model 1: Classification using the Peak Model

Given an electropherogram, the Peak Model aims to classify each peak in the DNA profile as allelic or artefactual. This model takes as an input an extracted peak, as described in section 2.2.4. The model can output its latent representation of the peak to the combined model. However, the model can also output a classification per peak and can be trained as a standalone model. We chose to use this standalone model as a stepping stone to the combined model and to facilitate faster iteration on changes.

2.4.1. Model architecture

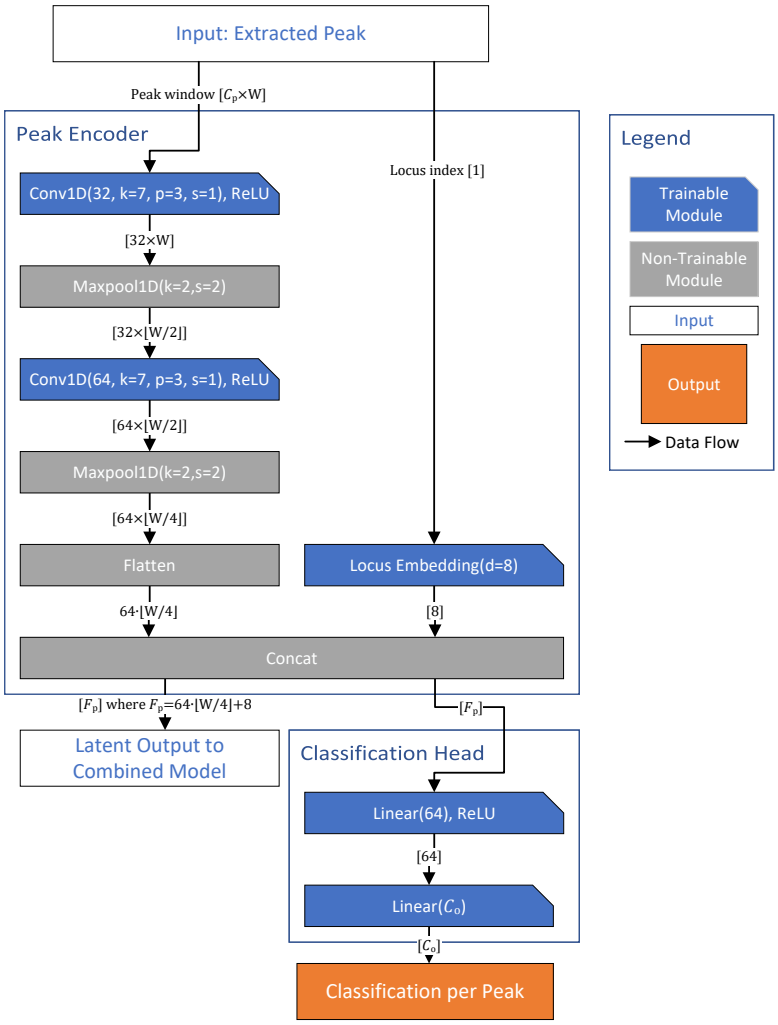


Figure 2.2: Architectural design of the Peak Model

We selected a 1D convolutional architecture because we identified local peak shape and nearby stutter patterns as the most important features. These features can be learned by convolutional filters over the peak-centred window (see section 1.2.2). We chose to use 1D convolutions over 2D convolutions, since 2D convolutions assume that channels (dye lanes) have an ordering and that a local neighbourhood has meaning along the channel axis. While channels are ordered by their measurement wavelength, local neighbourhoods do not have meaning along the channel axis. Artefacts such as spectral bleed-through seem to appear randomly among channels, not just in their neighbouring dyes.

Inputs and outputs The model takes as input an extracted peak ($\mathbf{p}_i \in \mathbb{R}^{C_p \times W}$) (see section 2.2.4). As shown in figure 2.2, the peak encoder consisted of a convolution-based encoder and an MLP-based

classification head. When using the model as a part of the combined model, the classification head is unused, and the model outputs latent representation $\mathbf{h}_i = f_{\text{peak}}(p_i)$. When used as a standalone model, the classification head outputs $\hat{\mathbf{y}}_i = f_{\text{cls}}(\mathbf{h}_i)$ for output classes (C_o). We used ($C_o = 2$) (background vs allele) in all experiments.

Locus embedding To capture locus-specific differences without training separate per-locus models [51, 53, 57], we concatenated a learned locus embedding to the flattened peak features. We hypothesised that a learned embedding can capture the differences between loci without the inefficiencies of training separate models. Experiments to explore this hypothesis can be found in section 2.4.2.

We assigned each of the 27 PPF6C loci to an integer index and added a *no_marker* index for peaks outside all locus ranges. We defined locus ranges in base-pair coordinates from the left boundary of the minimum allele bin to the right boundary of the maximum allele bin. If locus ranges overlapped after sizing (see section 2.2.3), we split the overlap at the midpoint between ranges. A peak was assigned to a locus if its centre (τ_i) fell within the locus range; otherwise it was assigned *no_marker*.

Peak annotations and loss We derived peak labels by checking whether the peak centre pixel ((c_i, τ_i)) coincided with an allelic pixel in the annotation segmentation image. We trained using cross-entropy loss. Handling class weight imbalance was deemed out of scope for this research. Unless stated otherwise, layer parameters not explicitly specified follow PyTorch defaults. The used PyTorch version is described in section E.2.

2.4.2. Ablation study protocol and hyperparameter space

We developed the architecture iteratively [26] and then evaluated robustness through an ablation study. The optimal hyperparameter configuration is listed in listing E.1, and ablation study details are reported in section 2.3.4. We trained these ablations on LT-annotated profiles from the NFI casework training data (A, B, C, and D), totalling approximately 900 profiles.

We evaluated the hyperparameters in table 2.1 to show that the model is robust relative to the base model in section 2.4.1.

Parameter	Possible values	Explanation
data_loading_strategy	raw, analysed	Compares the raw and analysed strategies discussed in section 2.2.4.
max_RFU	none, 5000, 15000, 33000	Evaluates linear normalisation by a maximum RFU as described in section 2.2.4. The value 5000 was chosen because most peaks above this threshold are allelic (see figure 3.6); 15000 is an intermediate setting; 33000 approximates the theoretical maximum.
log_scale	false, true	Enables or disables log scaling as described in section 2.2.4.
keep_fraction	0.2, 0.4, 0.6, 0.8	Tests smoothing strengths. Values 0.2 and 0.4 are used in practice [48]; 0.6 and 0.8 were included to explore weaker smoothing. Implementation details are provided in section 2.2.4.
include_max_pool_dyes (C_p)	1, 2	Tests $C_p = 1$ versus $C_p = 2$ as described in section 2.2.4.
window_size (W)	20, 120, 180	Controls the number of pixels extracted around the peak centre. $W = 20$ corresponds to approximately ±1 basepair (bp) and aims to capture only peak shape; $W = 120$ corresponds to ±6 bp and aims to capture the peak plus potential single stutters; $W = 180$ corresponds to ±9 bp and aims to capture the peak plus potential double stutters (section 2.4).
filter_peaks	false, true	Enables or disables peak filtering as described in section 2.2.4.
threshold	15, 25, 35, 45	RFU threshold for peak detection. A threshold of 15 is the lowest value for which peaks still visually resemble true peaks; peaks above 45 are very clear peaks (section 2.2.4).
channels	[16,32], [32,64], [64,128], [16,32,64], [32,64,128], [32,64,128,256]	Defines the Conv1D channel widths (and depth) in the peak encoder; this changes both the number of Conv1D/downsampling blocks and the number of filters per Conv1D layer.
kernel_size	1, 3, 5, 7, 9	Kernel size used for all Conv1D layers in the peak encoder.
pooling	attention, average, flat	Pooling operator used to aggregate encoder features. figure 2.2 shows the flatten pooling variant.
activation	ReLU, GELU, tanh	Activation function used in the Conv1D layers.
use_batchnorm	false, true	Adds (or omits) a BatchNorm1D layer after each Conv1D layer.
head_dropout_p	0, 0.1, 0.25, 0.5	Dropout probability between the final linear layers.
conv_dropout_p	0, 0.1, 0.25, 0.5	Dropout probability inserted between each Conv1D and downsampling layer.
downsample	maxpool, conv	Replaces max-pooling with strided convolution for downsampling.

Table 2.1: Overview of hyperparameters explored for the Peak Model ablation study.

Locus effect investigation protocol To investigate the learned locus embedding, we extracted the embedding matrix from the trained model and applied PCA to project the locus embeddings onto two dimensions. We used this projection to visually inspect whether loci formed clusters. We evaluated this projection with expert forensic DNA analysts to determine if some clusters might have biological relevance.

2.4.3. Baseline comparison protocol

We compared the Peak Model to DNANet and the AutoAnalyzer. Details on these methods are described in section 2.3.2. We trained the Peak Model on a combined dataset comprising the following: NFI AT+LT casework train A (limited to 5000 profiles); NFI LT casework train B, C, and D; and NFI research train data. This results in approximately 6000 profiles. We chose to combine these datasets to maximise the value of our high-quality annotations, while also including different dataset types to increase data

variability. We used a profile-level validation split of 0.3 for early stopping. We evaluated on the NFI research test set (approximately 60 profiles). For each profile, we generated predictions and computed an F1 score. We compared these F1 scores with those of the other baseline models, and p-values were calculated according to the protocol in section 2.3.3.

2.5. Model 2: Reconstruction with the Autoencoder Model

Given an electropherogram, the autoencoder learns a latent representation that preserves the signal structure needed to reconstruct the original profile. The encoder maps the full profile to a compressed latent representation, which the Combined Model later uses as global context. When trained in a standalone setting, the decoder reconstructs the electropherogram from this latent representation. We pretrain the autoencoder before training the Combined Model to provide a stable global representation and reduce the amount of labelled data needed for peak classification.

2.5.1. Model architectures

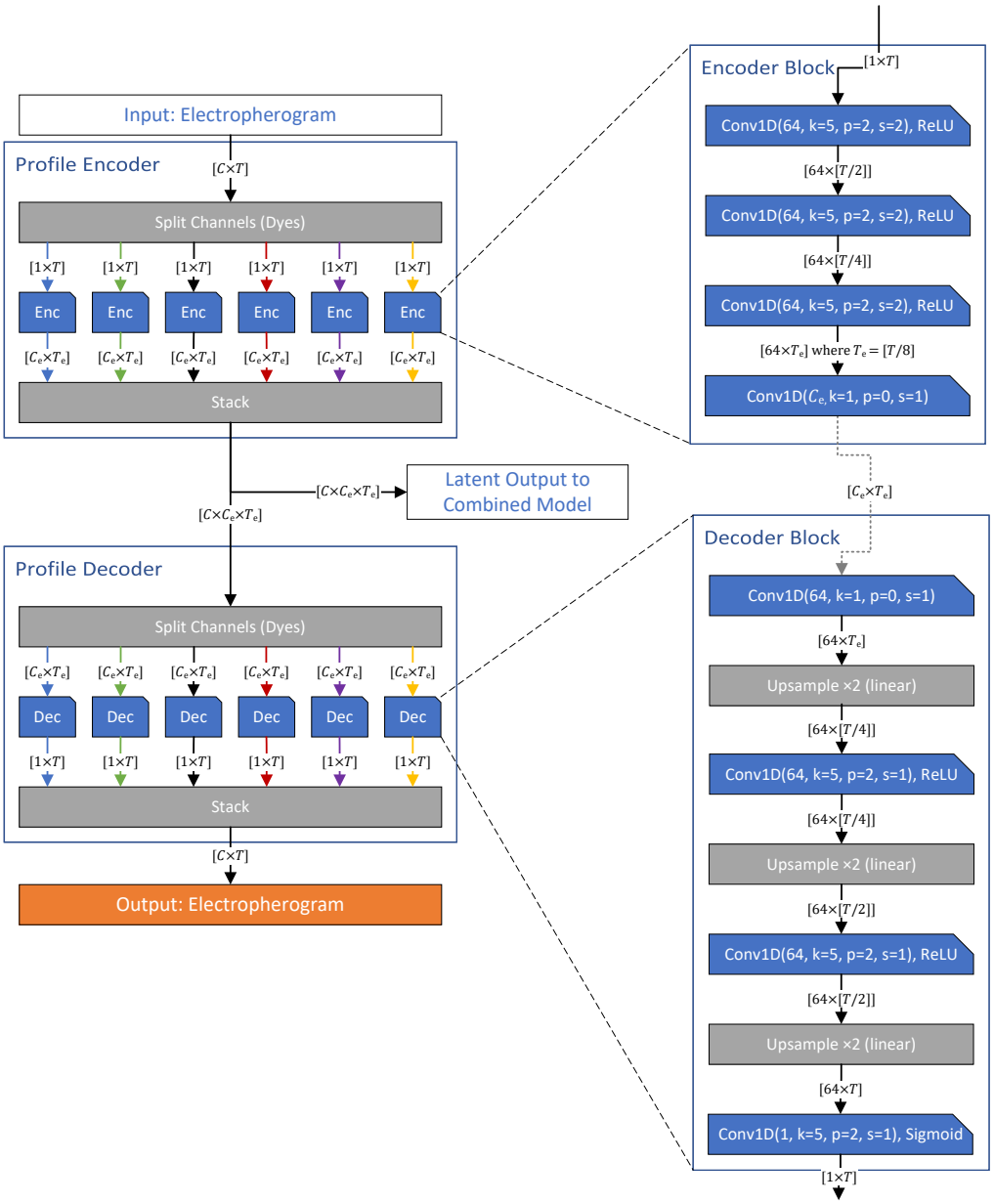


Figure 2.3: Architectural design of the Autoencoder Model. Weights between the encoder and decoder blocks are not shared in the final model. A variant with weight sharing across dye channels was also evaluated.

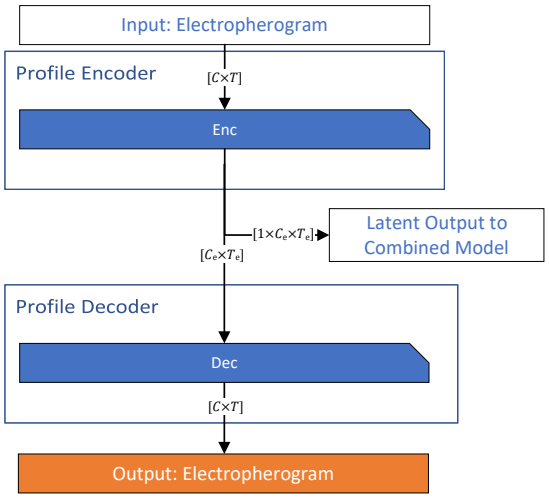


Figure 2.4: Architectural design of the Autoencoder Model with a multichannel 1D convolutional setup. Encoder and Decoder blocks are identical to figure 2.3.

We implemented a 1D convolutional autoencoder that compresses multi-channel electropherograms into a compact latent representation and reconstructs them at the original resolution. The exact model architecture is shown in figure 2.3. We chose a 1D convolution-based architecture because it has been successful with peak-based data in other fields (see section 1.2.2). In our final model, each encoder and decoder block has its own weights and is trained on a different dye channel. However, we also test a variant with shared weights. Furthermore, we test a variant in which the dye channels are not split, performing a single, multi-channel convolution. This architecture is described in the diagram in figure 2.4.

Inputs, outputs and loss The model takes as an input an electropherogram ($\mathbf{x} \in \mathbb{R}^{C \times T}$). When using the model as a part of the combined model, the decoder is unused, and the model outputs a latent representation $\mathbf{z} = f_{\text{enc}}(\mathbf{x})$ When used as a standalone model, the decoder outputs $\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z})$. T is assumed to be divisible by 8. The model applies 8x compression. We use MSE as a loss function. MSE matches our needs exactly; the priority is on the precise location and height of high-RFU peaks, while baseline noise is inconsequential.

2.5.2. Ablation study protocol and hyperparameter space

We evaluated architectural variants in table 2.2 to confirm that the autoencoder design is robust using the ablation study details specified in section 2.3.4. The optimal hyperparameter configuration is listed in listing E.2. We trained these ablations on profiles from the NFI casework training data A, totalling approximately 20 000 profiles.

2.5.3. Baseline comparison protocol

We compared the autoencoder to a Fourier and discrete wavelet transform (DWT) compression baselines to assess whether the learned latent representation effectively reconstructed electropherograms. We chose these baselines because they are commonly used to compress (peak-based) 1D signals. Reconstruction performance was measured using RMSE per profile, and statistical significance was assessed using paired permutation tests with Holm correction (see section 2.3.3). We use RMSE rather than MSE to estimate the mean error in RFUs. We train our model on the NFI casework train data A, approximately 20 000 profiles.

Fourier baseline (truncated real FFT) For each channel $c \in \{1, \ldots, C\}$, let $\mathbf{x}_c \in \mathbb{R}^T$ denote the length-T signal. We computed the real-valued discrete Fourier transform

$$\mathbf{X}_c = \text{rFFT}(\mathbf{x}_c) \in \mathbb{C}^{T/2+1},$$

Parameter	Possible values	Explanation
depth	3, 4, 5	Number of `Conv1D` layers in both the encoder and decoder.
log_scale	false, true	Enables or disables log scaling as described in section 2.2.4.
max_RFU	None, 33000	Applies linear normalisation by a maximum RFU as described in section 2.2.4. None disables this normalisation; 33000 approximates the theoretical maximum RFU.
model_architecture	`cnn_1d`, cnn_1d_shared, cnn	Compares alternative convolutional designs: a channel-specific 1D CNN with separate weights for every dye channel (figure 2.3); a channel-specific 1D CNN with weight-sharing across channels (figure 2.3); and a multi-channel 1D CNN (figure 2.4).

Table 2.2: Overview of hyperparameters explored for the autoencoder ablation study.

using the default FFT normalisation (forward unnormalised; inverse scaled by $1/T$). We retained the K lowest-frequency coefficients and set the remaining coefficients to zero:

$$\tilde{\mathbf{X}}_c[k] = \begin{cases} \mathbf{X}_c[k], & k = 0, \dots, K-1, \\ 0, & k = K, \dots, T/2, \end{cases}$$

and reconstructed the channel by the inverse real FFT,

$$\hat{\mathbf{x}}_c = \mathrm{irFFT}(\tilde{\mathbf{X}}_c) \in \mathbb{R}^T.$$

The full reconstruction $\hat{\mathbf{x}}$ concatenated $\hat{\mathbf{x}}_c$ over channels. We stored complex coefficients as paired real and imaginary parts to keep the representation real-valued.

Discrete wavelet transform baseline (truncated DWT). For each channel $\mathbf{x}_c \in \mathbb{R}^T$, we computed a 1D discrete wavelet decomposition using a Daubechies-4 wavelet. We used the maximum feasible decomposition level for length T and the chosen wavelet filter (as returned by the wavelet library). We converted the multilevel coefficient set (approximation and detail coefficients across levels) to a single 1D coefficient array $\mathbf{w}_c \in \mathbb{R}^M$ using a fixed coefficient-to-array mapping, and retained only the first K entries:

$$\tilde{\mathbf{w}}_c[j] = \begin{cases} \mathbf{w}_c[j], & j = 1, \dots, K, \\ 0, & j = K+1, \dots, M. \end{cases}$$

We then inverted the mapping to recover the structured coefficients and reconstructed via the inverse DWT to obtain $\hat{\mathbf{x}}_c \in \mathbb{R}^T$. When the inverse transform produced a length different from T, we cropped or zero-padded to exactly T samples.

Both baselines were non-trainable and served as signal-only latent representations that could substitute for ($\mathbf{z} = f_{\mathrm{enc}}(\mathbf{x})$) when benchmarking the combined model.

2.6. Model 3: Segmentation using the Combined Model

Given an electropherogram, the Combined Model aims to classify each detected peak as allelic or artefactual by combining two sources of information. First, it uses the Peak Model to describe each peak based on its local shape and nearby context. Second, it uses the autoencoder to capture global patterns in the full profile. The Combined Model merges these local and global representations to make a final classification for every peak. We evaluated three ways to merge the peak and profile information: a multilayer perceptron (MLP), Feature-Wise Linear Modulation (FiLM)-based modulation, and an attention-based combiner.

2.6.1. Model Architectures

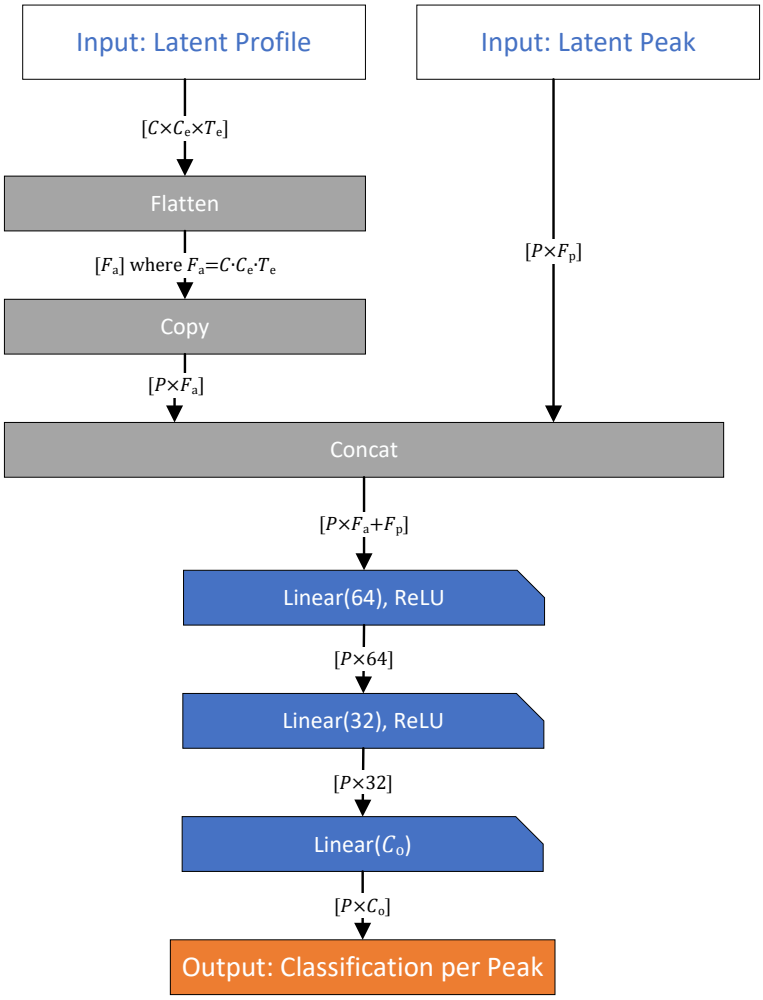


Figure 2.5: Architectural design of the Combined Model. We take as input the latent outputs from the Peak and Autoencoder Model.

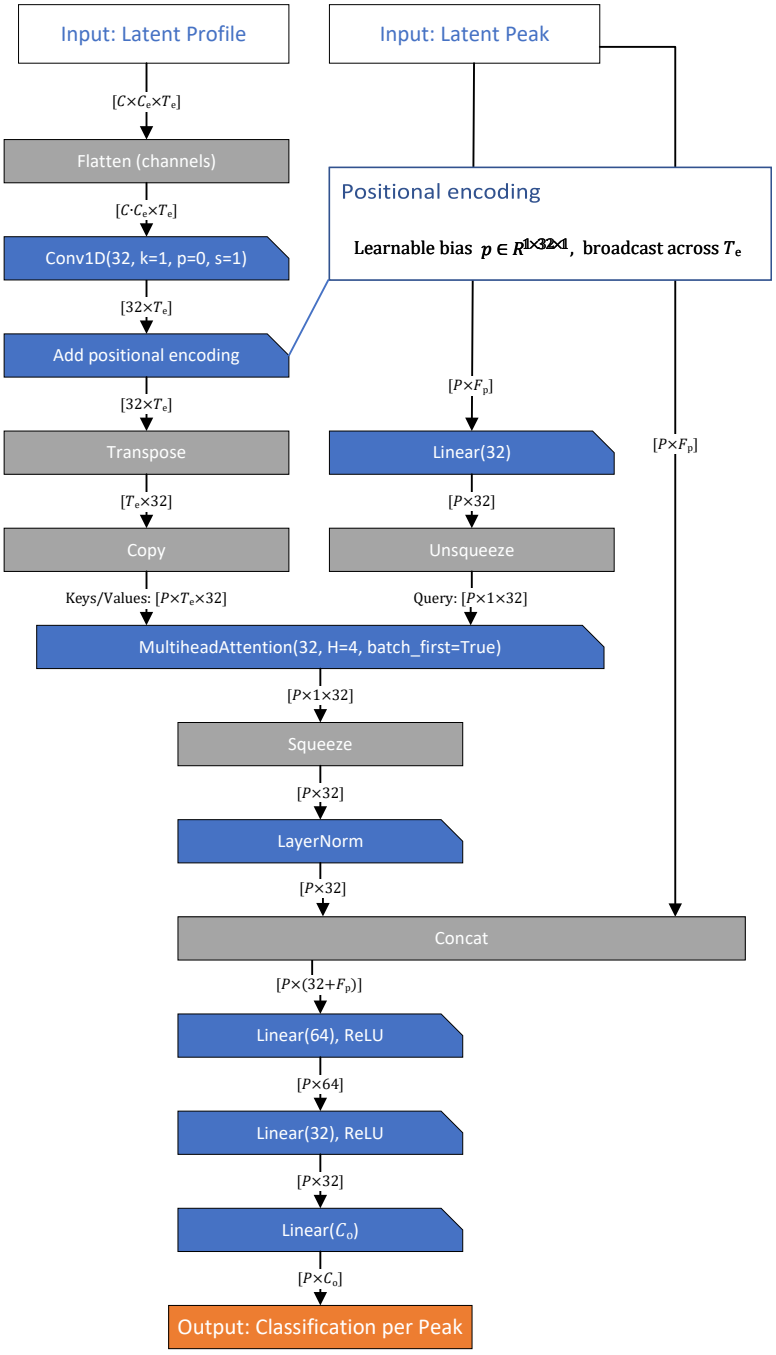


Figure 2.6: Architectural design of the Combined Model with an attention-based combiner

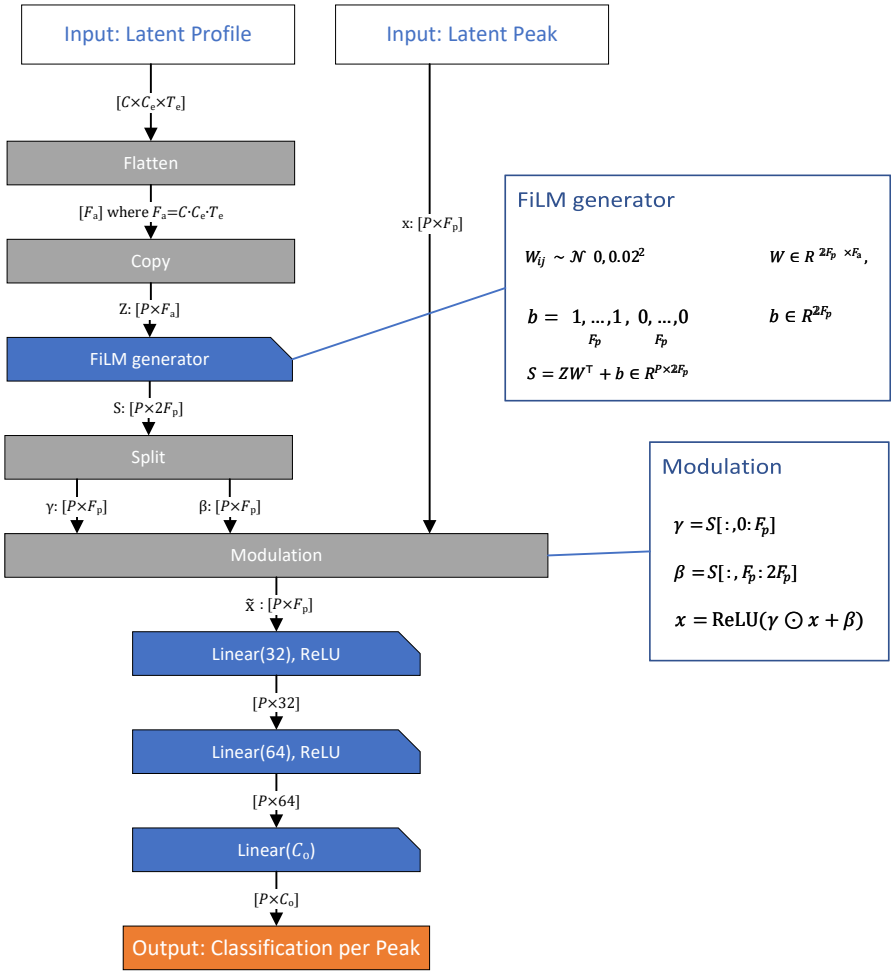


Figure 2.7: Architectural design of the Combined Model with FiLM modulation

We implemented three different architectures for the combined model. The first is an MLP-based combiner. This is our final model, shown in figure 2.5. We chose this model for its adaptability and strengths in identifying important features. Our second model uses FiLM-based modulation to combine the global and local signal [41]. The corresponding architecture is shown in figure 2.7. This model was chosen due to its strengths in combining peak-based local and global signals [17]. Our final model uses an attention-based combination (figure 2.6). The local features are encoded as query, while the global features represent the keys/values. This architecture has shown success in the biomedical domain of peak processing [20]. The weights of the global profile encoder branch of the Combined Model can optionally be frozen.

Inputs and outputs Given an electropherogram ($\mathbf{x}$), the Combined Model classifies each detected peak ($p_i \in \mathcal{P}$) as allelic or artefactual by combining local peak features with global profile context. The peak-processing branch embeds each detected peak window as ($\mathbf{h}_i = f_{\text{peak}}(p_i)$). In parallel, the autoencoder branch maps ($\mathbf{x}$) to a latent representation ($\mathbf{z} = f_{\text{enc}}(\mathbf{x})$). The combiner ($f_{\text{comb}}$) fuses ($\mathbf{z}$) with ($\{\mathbf{h}_i\}_{i=1}^{N}$) to produce context-aware peak classifications, ($\hat{\mathbf{y}}_i = f_{\text{comb}}(\mathbf{z}, \mathbf{h}_i)$).

Annotations and loss For the Combined Model, we calculate the loss over a full electropherogram. As annotations, we use the segmentation style annotations used in DNANet [62]. To calculate the loss, we first create a segmentation prediction from our peak-level classifications, as described in section 2.2.4. Using the segmentation-style annotations and predictions, we calculate the cross-entropy loss. We

chose to use cross-entropy loss in contrast to the previously used dice loss to allow for the possibility of multi-class prediction [62].

2.6.2. Ablation study protocol and hyperparameter space

We evaluated alternative fusion strategies to verify that the combined-model design was reasonable (see section 2.3.4). The optimal hyperparameter configuration is listed in listing E.3. We trained these ablations on LT-annotated profiles from the NFI casework training data (A, B, C, and D), totalling approximately 900 profiles.

Parameter	Possible values	Explanation
combiner_strategy	mlp, attention, film	Architecture of the Combined Model
freeze_autoencoder	false, true	Freezes the weights of the global profile encoder branch ($\mathbf{z} = f_{\text{enc}}(\mathbf{x})$) during training
hidden_dims	[16,32], [32,64], [64,128], [16,32,64], [32,64,128], [32,64,128,256]	Defines the MLP channel widths (and depth) in the combiner head; this changes both the number of linear layers and the hidden channels in each layer.
data_limit	None, 10000, 1000, 500	Limits the number of profiles included in the training data; None includes all available profiles in the set. 500 is supposed to represent a small but high-quality data set, such as the ground-truth or multiclass annotations (section 1.2.3)

Table 2.3: Overview of hyperparameters explored for the Combined Model ablation study.

2.6.3. Baseline comparison protocol

We benchmarked the combined model against DNANet [62], the commercial AutoAnalyzer, and the Peak Model. More details about the baselines can be found in section 2.3.2. We included the Peak Model to investigate the effect of the encoded global profile as an additional feature.

We trained and evaluated the Combined Model on the same datasets as we used for the Peak Model baseline comparison (section 2.4.3). Additionally, we evaluated our models on the NFI casework data with analyst annotations. This was done to demonstrate how closely our models match analysts' annotations. Finally, we evaluate on the NFI research test data with ground-truth annotations. However, we use the Allele F1 score (see section 2.3.1) instead of the Pixel F1 score. This also allows us to include the analyst annotations as one of our models and compare the performance of our models to human analysts. We computed per-profile F1 scores and assessed statistical significance using paired permutation tests with Holm correction as described in section 2.3.3.

2.6.4. Autoencoder contribution experimental protocol

To test whether the autoencoder contributed to prediction performance, we designed two experiments. The first experiment is designed to evaluate whether the global context improves prediction performance. The second experiment investigates whether pretraining the autoencoder has a positive effect or whether using an untrained encoder for the global profile is sufficient.

To analyse the impact of pretraining, we trained the combined model with a pretrained autoencoder and an untrained autoencoder under different training data sizes. We trained these variants on NFI casework training subset A (approximately 20 000 AT profiles). We chose to use this larger dataset to better evaluate the effect of dataset size, even though some of our smaller datasets have higher-quality annotations. We used the dataset limits shown in table 2.3. All other training details match section 2.3.4.

2.6.5. Peak height effect experimentation protocol

We analysed performance as a function of peak height by computing F1 scores over sliding RFU windows of width 40 RFU with a step size of 1 RFU. This window size provided enough samples per window, while remaining small enough to differentiate between thresholds (table A.1). RFUs were measured at the top of peaks, after all preprocessing. We applied this analysis to the NFI research test set for the combined model, DNANet, AutoAnalyzer, and human analysts.

2.6.6. Locus performance comparison protocol

We also investigated whether some loci are more difficult to predict than others. We compared the combined performance of the following models at each locus: DNANet, Combined Model, Peak Model, and AutoAnalyzer. We trained our models as described in the baseline comparison protocol. We evaluated on the NFI research data test set with ground-truth annotations. Since this dataset does not contain annotations for AMEL and the male-only loci from PPF6C, we excluded those. For every model, we calculated the F1 score per locus. From these scores per locus, we calculated the mean and standard deviation to show the difficulty of every locus over the different models.

3

Results and discussion

3.1. High-level model architectures

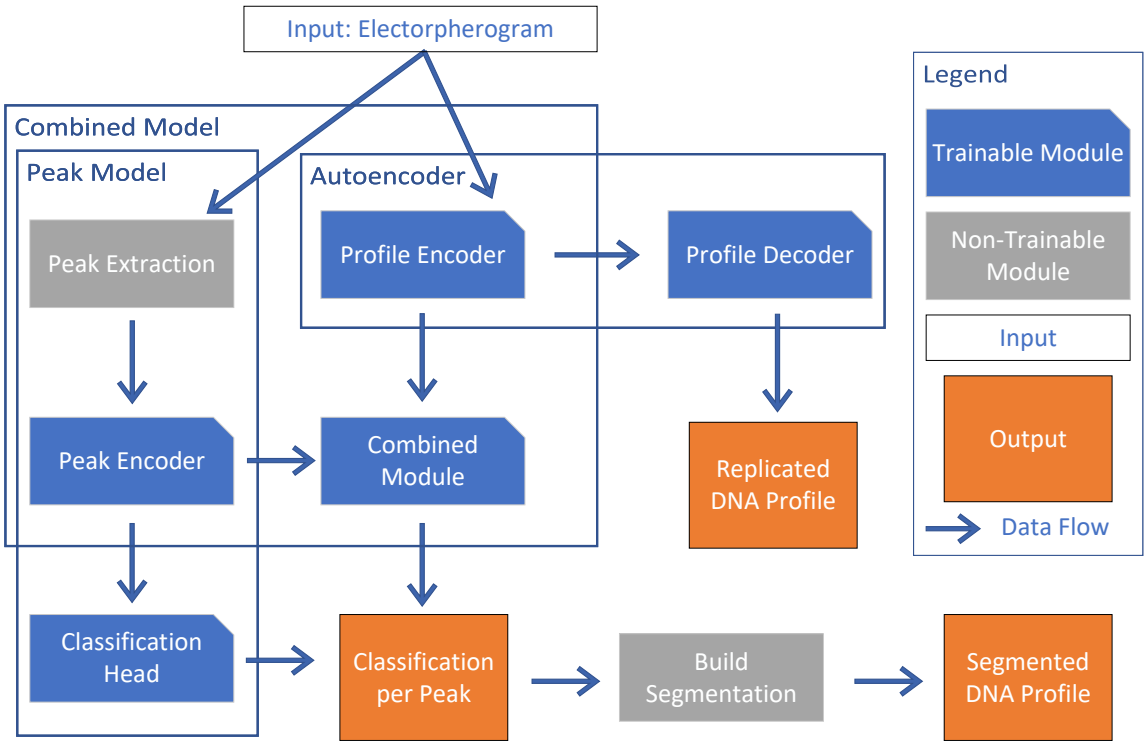


Figure 3.1: High-level overview of our 3 novel models: the Peak Model, Autoencoder, and Combined Model. The Combined Model uses elements from the Peak Model and Autoencoder to achieve accurate electropherogram allele calling. Blue blocks denote trainable (machine learning) modules, grey blocks denote non-trainable modules, and arrows indicate data flow from the electropherogram input to the model outputs.

Given an electropherogram, we aim to classify each peak in the DNA profile as allelic or artefactual. Prior work typically framed allele calling as a segmentation task that separates allelic signal from the remaining profile [51, 62]. We instead follow a two-stage workflow: first, we extract candidate peaks, and then classify each peak as allelic or non-allelic. This approach was inspired by research in other fields [19, 37, 49] (as discussed in section 1.2.2) and by the manual workflow by analysts (explained in section A.2.3).

We first extract fixed-width windows around all peaks in an electropherogram. The peak window aims to capture local details of the peak, such as its shape, height, and stutter presence. However, allele

calling often requires information beyond the local neighbourhood: analysts also use global profile context, including the number of contributors, peak height similarity, peak shifts, possible degradation, and profile-wide artefact patterns.

We therefore use a two-branch architecture that combines local peak features with global profile features. The local branch encodes each peak window, while the global branch encodes the full electropherogram into a compact representation that provides context for per-peak classification. In our setting, the global input is high-dimensional, whereas the local input is much smaller. The model must therefore learn to encode the global signal into features that are useful for downstream allele calling.

Because high-quality peak-level annotations are scarce (section 1.2.3), we pretrain the global branch with an autoencoder objective. This lets the encoder learn a structured representation of the electropherogram without requiring allele labels. We then reuse this pretrained encoder as the global context path in the combined model, which we train on limited (high-quality) annotated data.

Figure 3.1 shows a high-level overview of the model architecture and which components belong to which model. Section 2.4 describes the peak model, Section 2.5 describes the autoencoder, and Section 2.6 describes how we combine both branches for per-peak allele classification.

3.2. The Peak Model matches DNANet performance on ground-truth labels

To establish a competitive baseline for per-peak allele classification, we benchmarked the Peak Model against existing tools on data with ground-truth labels. We hypothesised that local peak windows capture most of the discriminative information for allele calling, so a peak-only model should achieve performance comparable to that of existing learning-based methods. Furthermore, hypothesise that we have found a reasonable architecture through an ablation study.

Comparison with state of the art To evaluate whether local peak windows can make reasonable predictions, we compared the Peak Model to DNANet and AutoAnalyzer on the NFI research data with ground-truth labels (table 3.1). We observed that the Peak Model and DNANet achieve nearly identical F1 scores (0.923 vs 0.922; $p = 0.650$; paired permutation test, see section 2.3.3), whereas AutoAnalyzer achieves a higher F1 score (0.936; $p = 0.001$). This suggests that local peak windows suffice to match DNANet on this benchmark, but the Peak Model does not close the gap to AutoAnalyzer under ground-truth evaluation. A limitation of the ground-truth research data is that it may not accurately reflect casework data, as the research data has not been degraded or contaminated.

Model	Pixel F1 score	p (vs. Peak Model)
Peak Model	0.923	—
DNANet	0.922	0.650
AutoAnalyzer	**0.936**	0.001

Table 3.1: Performance on NFI research data with ground-truth labels (Pixel F1 score). P-values are calculated using a paired permutation test, see section 2.3.3

Architecture evaluation and ablation study To justify the Peak Model design, we conducted an ablation study to identify which architectural and preprocessing choices drive performance. We hypothesised that the selected parameter configuration is not significantly worse than any alternative configuration obtained by varying one parameter at a time within the considered ranges. We first consider the ablation study as a whole, then take a more thorough look at specific parameters and their effects on performance.

To evaluate the Peak Model hyperparameters specified in table 2.1, we varied one component at a time and compared the performance to the selected configuration (table 3.2). We observed that performance did not significantly increase when we chose alternative hyperparameters. Furthermore, we observed that many changes did not produce significant differences (e.g., activation function, batch normalisation, smoothing, and several channel/stride choices), whereas a small set of settings consistently reduced performance when altered, including dropout, peak filtering, pooling strategy, and the analytical threshold/window-size settings. This pattern suggests that the Peak Model is not fragile to minor architectural variation, but it depends on preserving informative peak context and using appropriate

aggregation and filtering. From this ablation study, we cannot determine how parameters interact, nor can we rule out the possibility that other parameter combinations yield better performance.

Parameter	Performance drop vs. selected configuration
Activation function	Not significant
Batch normalization	Not significant
Data loading strategy	Not significant
Data smoothing	Not significant
Downsample method	Significant
Dropout conv	Significant
Dropout head	Not significant
Filter peaks	Significant
Hidden channels	Not significant
Include locus embedding	Significant
Kernel size	Not significant
Log scale	Not significant
Maxpool dyes	Not significant
Max RFU	Not significant
Pooling strategy	Significant
Threshold	Significant*
Window size	Significant*

Table 3.2: Ablation results for the Peak Model. Asterisks indicate settings that were significant for some alternative values but not for all. Full results are reported in section C.1.

3.3. Locus embedding encodes important features, no relations between locus biology and performance are found

In previous work, separate models or branches were trained to capture the differences in different loci [51, 57]. We hypothesised that we could capture these differences in loci using a learned embedding. Furthermore, we hypothesised that we can identify biological features of loci based on our model.

To the effect of the locus embedding, we trained the model with and without the locus embedding and compared performance (table 3.3). We observed a significant performance drop when removing the embedding (F1 from 0.939 to 0.924; $p < 0.001$), indicating that locus identity provides useful context for classification. We did not train separate models to capture these differences, as done in previous work, and thus cannot determine which strategy is more effective.

Use locus embedding	F1 score mean ± std	p (vs True)
True	0.939 ± 0.002	-
False	0.924 ± 0.001	<0.001

Table 3.3: Performance of Peak Model with or without locus embedding on NFI casework data with analyst annotations (Pixel F1 score). P-values are calculated using Kruskal-Wallis and Dunn's post hoc tests, see section 2.3.4.

Locus embedding interpretation To interpret what the embedding captures, we visualised the learned locus representations. In figure 3.2, we show the embeddings using a two-dimensional principal component analysis (PCA) projection. The projection shows no clear global clustering that aligns with known biological groupings of loci, based on properties such as repeat motifs, stutter ratios, and length. Because PCA compresses a high-dimensional representation into two components, we treat this plot as a qualitative diagnostic rather than a biological analysis. During an expert review of figure 3.2 (personal communication), two local neighbourhoods were highlighted as potentially related. The left group (D8S1179, SE33, D12S391, D22S1045) consists of loci measured in the red dye channel in PPF6C, suggesting that embedding may capture dye-specific signal characteristics rather than shared biology. The right group (TH01, TPOX, AMEL, and DYS570) contains loci with low allelic diversity in our dataset: many individuals share the same allele or a small set of alleles. This could make their observed peak patterns more similar and place them close in the embedding space. We did not test this interpretation

quantitatively; therefore, we report it as hypothesis-generating and leave biological validation to future work. The projection also contains a small number of visually isolated loci (e.g., D3S1358 and FGA). For these loci, we found no consistent biological explanation among the attributes considered, suggesting that the embedding may capture technical or distributional properties of the data rather than biological similarity. Furthermore, the repeat motif does not aid in explaining the geometry. In particular, Penta E and Penta D are both 5-nucleotide repeat loci, yet they lie far apart in the projection, indicating that other locus-specific or technical factors likely dominate the learned representation.

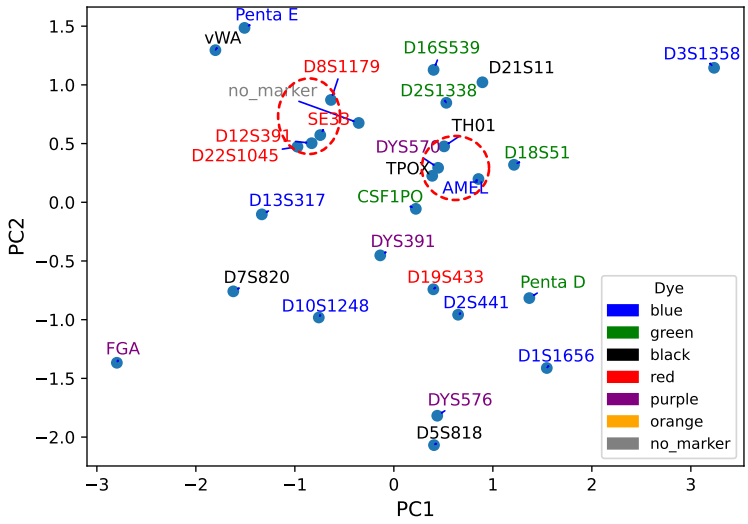


Figure 3.2: Locus embedding visualised using principal component analysis. The red dashed circles represent potentially related neighbourhoods.

Prediction difficulty across loci To examine whether any loci dominate the overall difficulty of predictions, we compared F1 scores per locus across different models (figure 3.3). See also figure C.1 for the individual model scores per locus, and table C.25 for pairwise p-value calculations between the loci. We observed performance differences across loci, but we are unable to explain the relationship between the slightly more difficult loci.

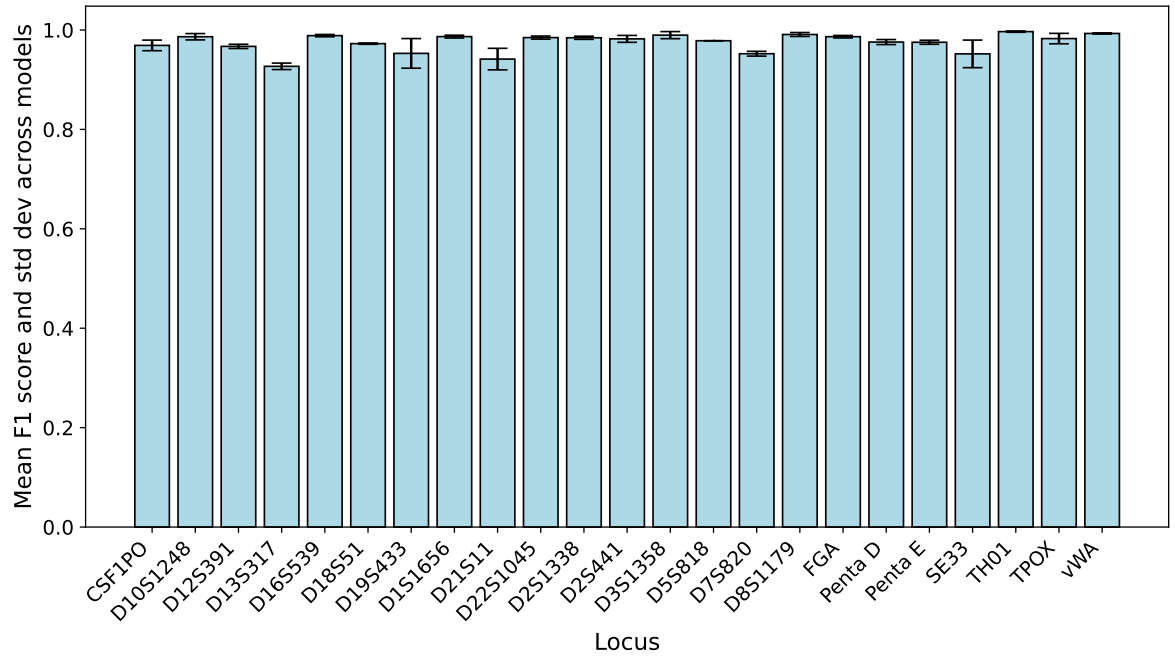


Figure 3.3: Mean F1 scores and standard deviations across models (DNANet, Combined Model, AutoAnalyzer, Human Analyst) for each locus. We can see that some loci are slightly more difficult to predict.

3.4. A learned latent code reconstructs electropherograms more accurately than fixed-basis compression

Electropherograms encode global information (e.g., contributor complexity, baseline noise, and degradation) that local peak windows cannot capture (see sections A.2.3 and A.2.4). We therefore hypothesised that a learned latent representation can compress a full electropherogram while preserving peak positions and intensities, providing a compact global context for the Peak Model.

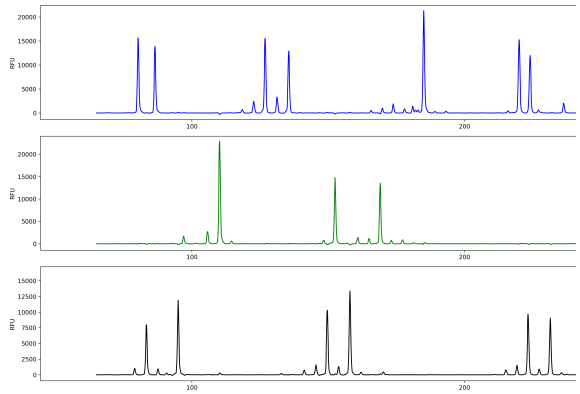
Autoencoder performance compared to baselines To evaluate whether the autoencoder can accurately represent the profile, we compared our autoencoder to two fixed-basis baselines (Fourier and discrete wavelet transform, DWT) at 8× compression and measured reconstruction error using the Root Mean Square Error (RMSE) (table 3.4). The autoencoder achieved the lowest error (RMSE 52 ± 27), substantially outperforming Fourier (195 ± 95) and DWT (245 ± 122; both $p < 0.001$ vs. the autoencoder). Relative to these baselines, the autoencoder reduced RMSE by 73% (vs. Fourier) and 79% (vs. DWT), indicating that a learned code captures electropherogram structure more efficiently than conventional frequency- and wavelet-based encodings. An RMSE of 52 RFU is small relative to typical allelic peak heights. In practice, a 50 RFU deviation matters mainly near the detection threshold, where it can determine whether a marginal peak is detected at all; once a peak is clearly above baseline, errors of this magnitude are unlikely to change allele calling. Because our most important features for reconstruction are the peak locations and heights, other methods may achieve higher compression or better reconstruction performance.

Compression method	RMSE mean ± std	Δ RMSE (vs. autoencoder)	p (vs. autoencoder)
Autoencoder	**52 ± 27**	—	—
Fourier	195 ± 95	143	<0.001
DWT	245 ± 122	193	<0.001

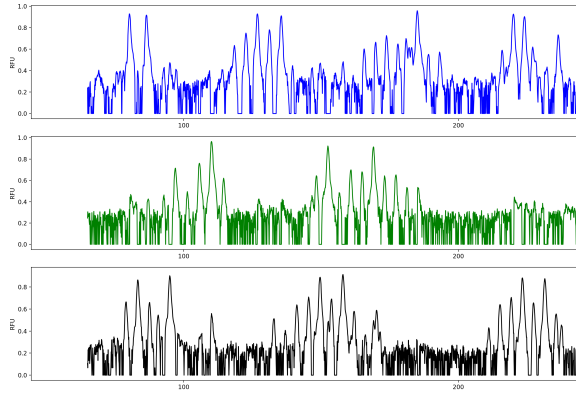
Table 3.4: Reconstruction performance at 8× compression. Lower RMSE indicates better reconstruction. P-values are calculated using a paired permutation test, see section 2.3.3

To assess qualitative fidelity, we compared original and reconstructed traces for a representative profile snippet (figure 3.4). By visual inspection, we can see that the reconstructed electropherogram

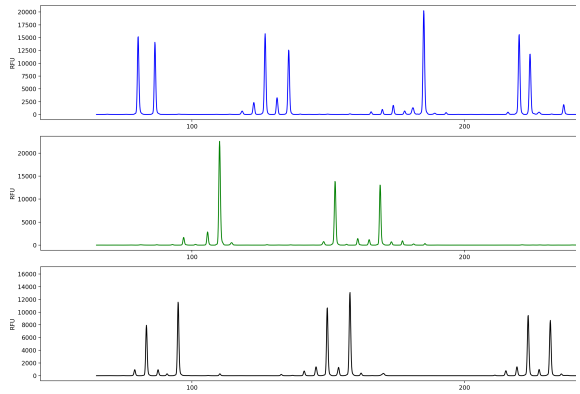
closely matches the original, preserving peak positions and relative peak heights. The main visible deviation is mild smoothing in the preprocessed domain, which does not remove or shift prominent peaks, as shown in the example (figure 3.4). This qualitative agreement supports the interpretation that the latent code retains peak-level structure while compressing global context.



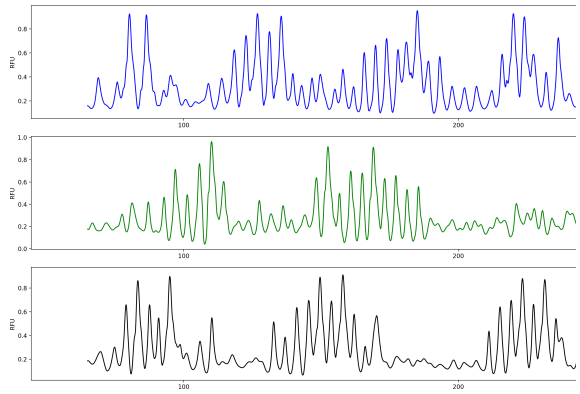
(A) Step 1: Original Profile



(B) Step 2: Original profile after preprocessing



(C) Step 4: Reconstructed profile after denormalisation



(D) Step 3: Preprocessed profile after autoencoder passthrough

Figure 3.4: Qualitative comparison of autoencoder performance on a profile snippet. We can see that the reconstructed profile (**C**) looks almost identical to the original profile (**A**). Comparing the preprocessed profiles (**B** and **D**), we can see the autoencoder applies smoothing while keeping important features.

Autoencoder architecture and weight sharing We hypothesised that, because dyes are largely independent of each other, reconstruction performance would be best with a separate autoencoder for each dye channel. A second open question is whether each channel requires its own convolutional filters or whether a single set of filters can generalise across channels. We hypothesised that a 1D convolutional neural network (CNN) would better match the underlying signal structure than 2D convolutions, and that sharing weights across channels would not substantially degrade reconstruction quality.

To test these hypotheses, we compared three autoencoder architectures: a channel-specific 1D CNN (figure 2.3), a channel-specific 1D CNN with shared channel weights (figure 2.3), and a multichannel 1D CNN (figure 2.4). Table C.17 shows that the multichannel 1D CNN performs substantially worse, increasing the reconstruction error from 1447 ± 154 to 43022 ± 4101 MSE ($p < 0.001$). This suggests that dyes in an electropherogram are mostly independent of one another and should be treated as such.

To evaluate whether channel-specific filters are necessary, we compared the CNN with shared and separate weights. Sharing weights increased the mean squared error (MSE) (from 1447 ± 154 to 1737 ± 734), but this difference was not statistically significant ($p = 0.509$). The large variance for shared weights implementation further suggests that weight sharing may be less stable across runs, even if average performance is comparable. A potential reason for this large variance is the fact that the internal standard dye has different properties from the first five dyes. We did not investigate the internal standard dye or whether it might differ from the sample dyes. Reconstruction performance per dye

has not been tested. Overall, these results support using channel-specific 1D convolutions and indicate that a shared weights 1D architecture can be viable when model simplicity or parameter efficiency is prioritised, but it does not provide a consistent performance advantage in our setting. This partially confirms our hypothesis that reconstruction performance would be best with a separate autoencoder for each dye channel.

3.5. Global profile context with a pre-trained autoencoder improves ground-truth performance over peak-only classification

Local peak windows capture much of the information needed for allele calling, but ambiguous peaks often require profile-level context (e.g., contributor complexity, baseline noise, and degradation). We therefore hypothesised that adding a global electropherogram representation improves classification beyond peak-only models.

Comparison of Combined Model with Peak Model and the state of the art To test the effect of our global encoding, we compared the Combined Model against the Peak Model, DNANet, and AutoAnalyzer on NFI research data with ground-truth labels (table 3.5). The Combined Model outperforms DNANet (0.934 vs. 0.922; $p = 0.001$) and the Peak Model (0.934 vs. 0.923; $p = 0.001$), and it performs comparably to AutoAnalyzer (0.934 vs. 0.936; $p = 0.650$). These results support the major message that global profile context adds discriminative information beyond local peak shape under ground-truth evaluation. We did not investigate the contents of the compressed latent layer, and it is unknown whether the network learns any other useful features (such as the number of contributors or the presence of artefacts) besides the location and height of peaks. While the Combined Model significantly outperforms DNANet and the Peak Model, it is unknown whether this performance difference is meaningful in practice. We did not evaluate the effect of these models on downstream likelihood ratio calculations.

Model	Pixel F1 score	p (vs. Combined Model)
Combined Model	**0.934**	—
DNANet	0.922	0.001
AutoAnalyzer	**0.936**	0.650
Peak Model	0.923	0.001

Table 3.5: Performance on NFI research data with ground truth annotations (Pixel F1 score). P-values are calculated using a paired permutation test, see section 2.3.3

Combined Model ablation study To justify the Combined Model design, we performed an ablation study to identify which architectural and preprocessing choices affect performance. We evaluate two parameters: *hidden_channels* and *freeze_autoencoder*. Tables C.21 and C.22 show us that no parameters resulted in significant differences in performance.

Combined model architectures To evaluate the effect of the combination strategy on the combined model's performance, we test three separate combiner strategies: a multilayer perceptron (MLP), feature-wise linear modulation (FiLM) and an attention-based combiner. table 3.6 shows us the MLP and FiLM perform similarly, while the attention-based combiner slightly underperforms. From this, we can conclude that complex combination methods may not be necessary for our data and context.

model_combiner_strategy	Pixel F1 score mean ± std	p (vs. mlp)
mlp	**0.891** ± 0.004	-
film	**0.893** ± 0.003	0.477
attention	0.868 ± 0.008	0.001

Table 3.6: Performance of different model architectures on NFI casework data with analyst annotations (Pixel F1 score). P-values are calculated using Kruskal-Wallis and Dunn's post hoc tests, see section 2.3.4.

Self-supervised pre-training and annotated data requirements We investigate whether self-supervised pre-training of an autoencoder on electropherograms improves allele calling, particularly when training labels are scarce. Previous research suggests that pre-training helps models generalise, especially with noisy or incomplete data [42, 63]. However, it remains unclear whether this benefit extends to electropherograms and how it varies with the amount of available training data. Our hypothesis is that pre-training on electropherograms enhances allele calling accuracy when only small quantities of (high-quality) annotated data are available.

To test this, we trained the Combined Model with and without autoencoder pre-training and evaluated performance across different training data limits (table 3.7). Pre-training improves F1 at 500 and 1000 labelled samples (0.924 vs. 0.912, $p = 0.010$; and 0.931 vs. 0.923, $p = 0.008$), while performance converges when substantially more labels are available (10,000 or no limit). These results support the major message that self-supervised pre-training primarily benefits the low-data regime, where representation learning substitutes for missing annotations. When sufficient data is available, these benefits disappear.

data limit	Pre-trained F1 score mean ± std	Untrained F1 score mean ± std	p (pre-trained vs. untrained)
500	**0.924** ± 0.006	0.912 ± 0.006	0.010
1000	**0.931** ± 0.004	0.923 ± 0.004	0.008
10000	0.937 ± 0.004	0.936 ± 0.002	0.347
None	0.936 ± 0.002	0.937 ± 0.001	0.597

Table 3.7: Comparison of F1 scores between pre-trained and untrained models across different data limits on NFI casework data. Data limit corresponds to the number of profiles included in the training data set, None corresponds to approximately 20 000 profiles. P-values are calculated using a paired permutation test, see section 2.3.3.

3.6. Model rankings depend on the labels and evaluation target

To assess how data and label sources influence conclusions about model quality, we evaluated the same tools on a different dataset with a different annotation type. We hypothesised that tools trained to replicate analyst decisions would score higher on analyst labels, even if their ranking differs under ground-truth evaluation.

To test this, we evaluated the same methods on analyst-annotated NFI casework data (table 3.8). We observed that DNANet outperforms the Combined model (0.966 vs. 0.942; $p = 0.001$), while AutoAnalyzer performs the lowest (0.897; $p = 0.001$). This reversal in ranking relative to the ground-truth benchmark indicates that the evaluation target matters. DNANet shows higher performance, which can be explained by the fact that it was trained on similar data, annotated according to the same analyst protocol. From this, we can conclude DNANet is better at imitating analyst annotations. However, it could also mean that DNANet achieves higher performance across the casework data. We were unable to test this hypothesis because no ground-truth annotations are available for casework data.

Model	Pixel F1 score	p (vs. Combined Model)
Combined Model	0.942	—
Peak Model	0.935	0.048
DNANet	**0.966**	0.001
AutoAnalyzer	0.897	0.001

Table 3.8: Performance on NFI casework data with analyst annotations (Pixel F1 score). P-values are calculated using a paired permutation test, see section 2.3.3

To benchmark model performance against expert judgment, we computed allele-level F1 scores on the NFI research dataset with ground-truth annotations. Table 3.9 shows that the human analyst achieves the highest F1 (0.975), exceeding all automated methods. Among automated approaches, DNANet performs best (0.973) and exceeds the other models on this benchmark. While we do evaluate on a different metric, the reasons for DNANet's advantage are unclear from the available data and warrant further analysis.

Model	Allele F1 score	p (vs. Human Analyst)
Human Analyst	**0.975**	—
Peak Model	0.926	0.002
Combined Model	0.937	0.002
DNANet	0.973	0.026
AutoAnalyzer	0.936	0.002

Table 3.9: Performance on NFI research data with ground-truth annotations (Allele F1 score). P-values are calculated using a paired permutation test, see section 2.3.3

3.7. Low-RFU peaks are harder to classify, regardless of method

Peaks with low fluorescence intensity (low RFU) are more likely to be artefactual and therefore harder to classify correctly than high-RFU peaks. This hypothesis is important because electropherogram measurements contain many low-intensity peaks; if performance degrades in this regime, overall allele calling accuracy will be disproportionately affected. Furthermore, peak heights have substantial effects on downstream likelihood-ratio calculations, and the relationships between peak heights and model errors warrant investigation.

Distribution of data To investigate whether low-RFU peaks constitute a substantial fraction of the data (and whether this depends on dye), we examined the distribution of RFU values per dye (figure 3.5). Note that these distributions are generated from NFI casework data train A, B, C, and D after baseline subtraction using the raw loading strategy. Across the five sample dyes, the RFU distribution is strongly skewed toward values near 0, with progressively fewer peaks at higher RFU. In contrast, the orange internal lane standard shows a distinct, more controlled distribution, consistent with its synthetic and standardised nature. We also observe dye-dependent frequencies of negative RFU values. Because negative RFUs typically arise and spectral bleedthrough effects rather than true signal, their presence indicates that spectral bleedthrough artefacts are common.

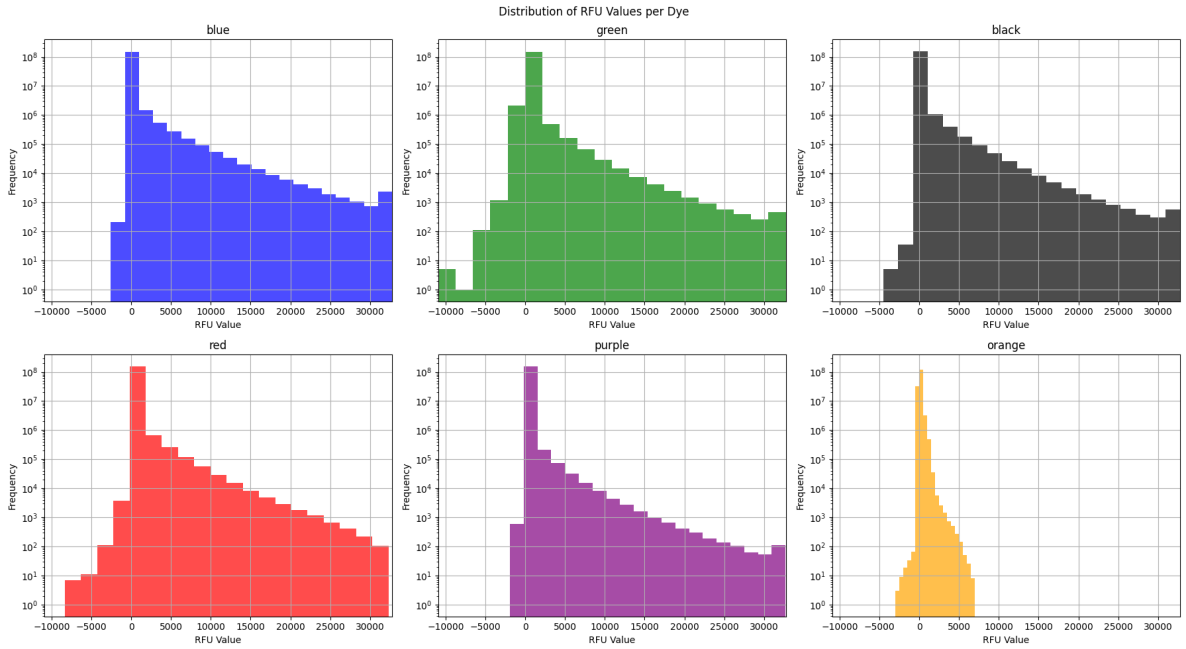


Figure 3.5: Distribution of RFU values per dye. The X-axis represents the discrete RFU values, and the Y-axis shows the frequency at which they occur logarithmically. We can see that most values are near zero, and that higher peaks are rarer than lower ones. Negative values also occur but are relatively uncommon.

Peak height in relation to fraction of annotated values To evaluate how peak height relates to the probability that a peak is truly allelic, we quantified the fraction of peaks annotated as allelic across

RFU bins (figures 3.6 and 3.7). Here, we use the same data as in figure 3.5 and consider all pixels in an allelic bin as annotated. We can see that for RFU values above roughly 5000, over 90% of values are annotated as allelic, whereas peaks below 0 are never annotated as allelic, and the orange internal standard contains no allelic annotations, as expected. Zooming in on 0-3000 RFU (figure 3.7) further shows a monotonic increase in allelic fraction with increasing RFU until the curve stabilises at higher RFU. At extremely high RFU values (most evident in the red dye), not all peaks are annotated as allelic, consistent with occasional high-intensity artefacts (e.g., saturation effects). Because the highest-RFU bins contain relatively few observations (figure 3.5), these tail estimates may also be noisier.

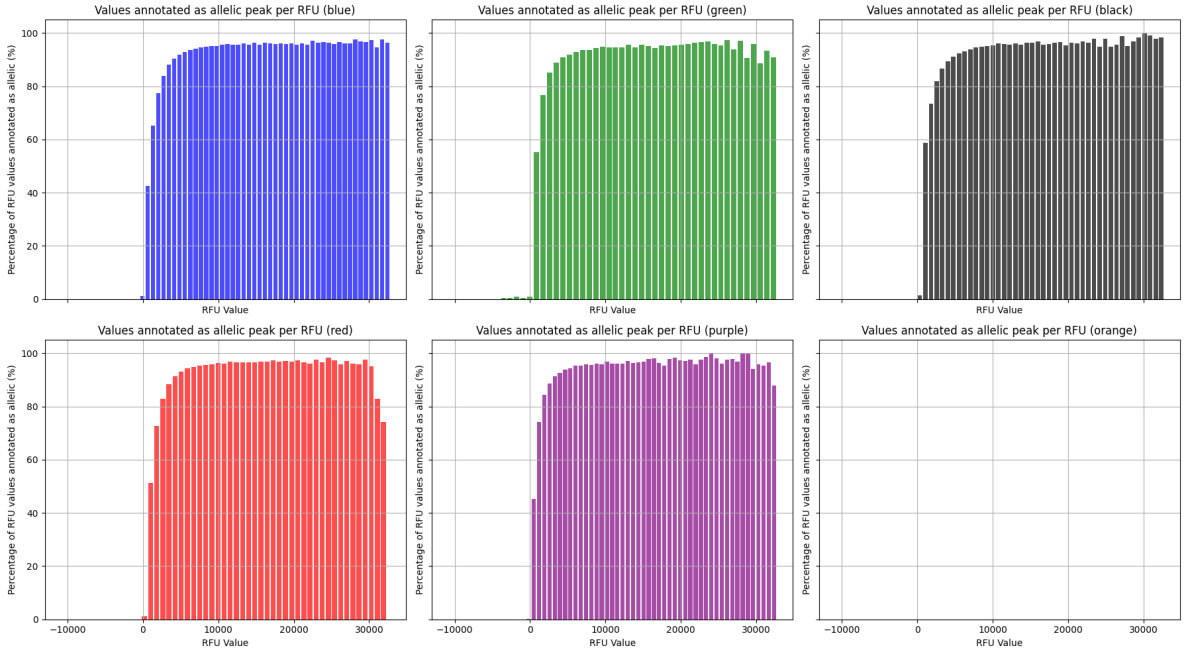


Figure 3.6: Percentage of values annotated per RFU for each dye. The X-axis shows the full range of discrete RFU values, while the Y-axis shows the percentage of RFU values considered allelic.

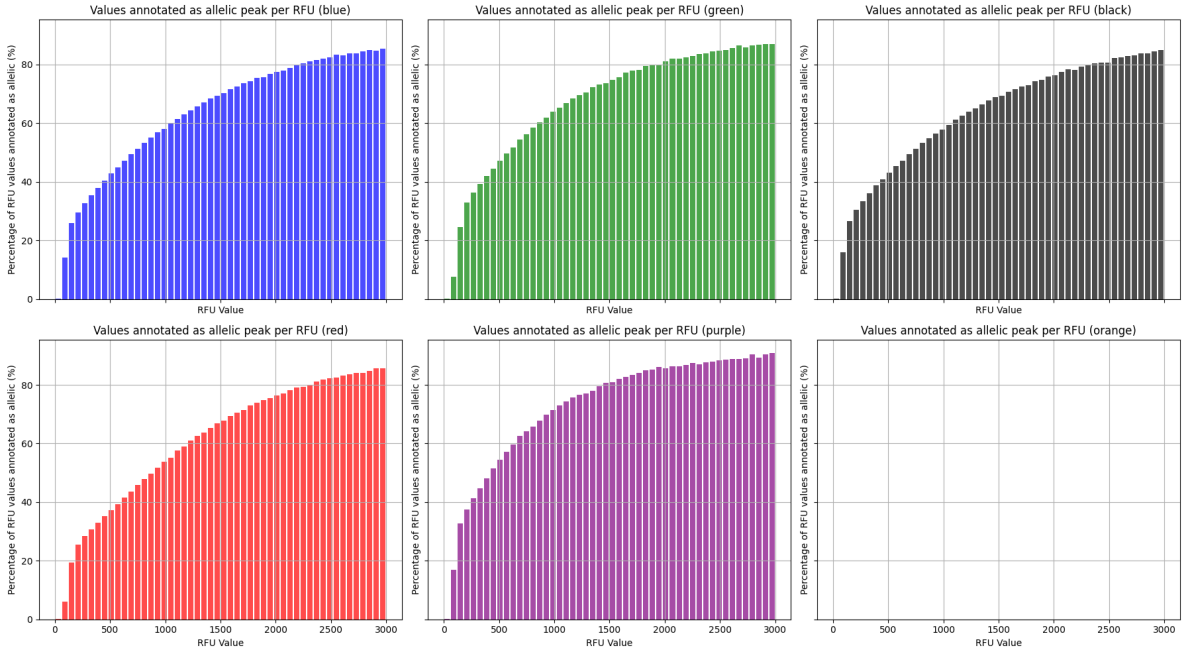


Figure 3.7: Percentage of values annotated for each dye. The X-axis represents the discrete RFU values from 0 to 3000, while the Y-axis shows the percentage of RFU values considered allelic.

Model performance for different peak heights To test whether low RFU systematically reduces classification performance across allele calling methods, we compared method performance as a function of peak height using a sliding-window F1 score (Figure 3.8). All methods show markedly reduced performance for peaks below approximately 300 RFU, after which performance increases with RFU and stabilises. Notably, the methods differ only marginally in the low-RFU regime, indicating that the difficulty is largely shared across approaches rather than being specific to a particular model. Together, these results support the hypothesis that low-RFU peaks are intrinsically challenging to classify, consistent with a lower signal-to-noise ratio and a higher prevalence of artefacts at low intensity.

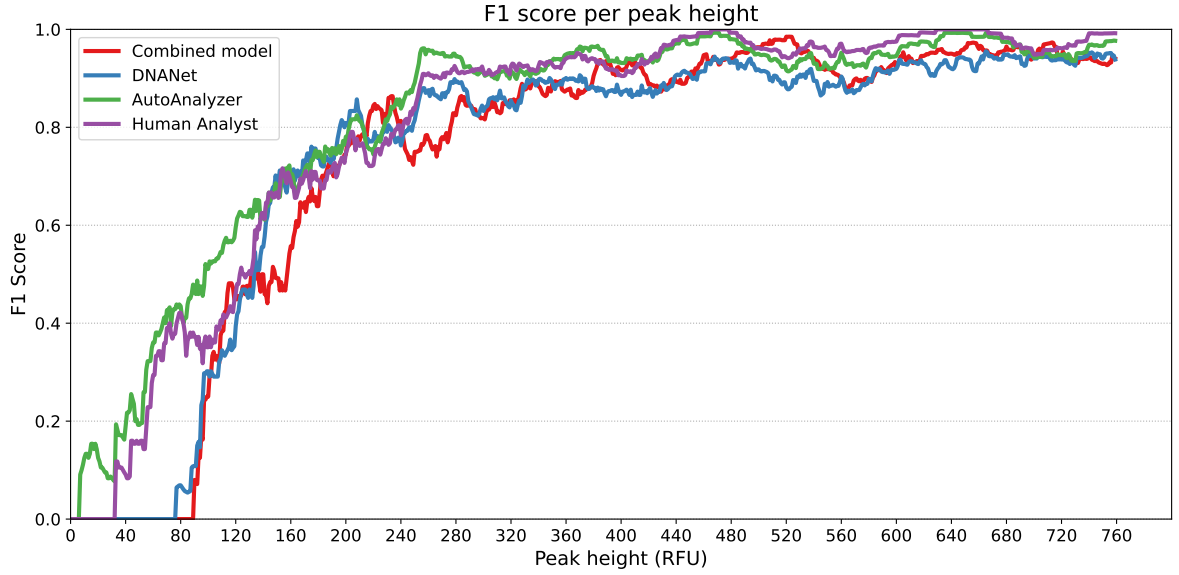


Figure 3.8: F1 score of different allele calling methods per peak height on NFI research data with ground-truth labels (Pixel F1 score). F1 scores for models are calculated over a sliding window of 40 RFU (x=0 corresponds to 0-40 RFU).

To assess whether model predictions are driven primarily by peak height rather than by additional

signal characteristics, we compared performance at a fixed RFU level with the prevalence of allelic peaks at that RFU. At 300 RFU, all methods achieve an F1 score above 0.8 (figure 3.8), while only approximately 40% of peaks are annotated as allelic at this intensity (figure 3.7). If peak height alone determined predictions, performance at 300 RFU would be expected to track this relatively low allelic fraction more closely. Instead, the strong performance despite moderate allelic prevalence indicates that the methods exploit additional discriminative features beyond peak height, such as locus- or dye-specific context and peak shape or spacing cues.

Overall, peak height can be extremely important for allele calling: the low-RFU regime contains many observations, is less likely to be truly allelic, and produces a consistent performance drop across methods, motivating downstream analyses that focus on separating true low-intensity alleles from low-intensity artefacts.

3.8. Discussion: implications, limitations, and future directions

Previous sections already present the major message-driven discussions for each experiment. Here, we synthesise those findings to answer the research questions, clarify what the results do and do not justify, and outline the practical and scientific implications.

Answer to RQ1: architecture choices for allele calling RQ1 asked how different model architectures perform for electropherogram allele calling. Taken together, the results show that a peak-based formulation is competitive and that global profile context can improve performance when the model encodes it effectively. The Peak Model demonstrates that local peak windows can support strong peak-level decisions (see section 3.2). The Combined Model then shows how adding a learned global representation can improve over peak-only classification under the ground-truth evaluation target (table 3.5). The architecture analyses further constrain how to build this global path: channel-specific 1D convolutions provide better performance than a single multi-channel convolution (table C.17), and simple combination mechanisms suffice in this setting (table 3.6). Overall, the evidence supports the claim that a peak-based architecture, extended with global context, can be more effective for allele calling than previous segmentation-based architectures.

Answer to RQ2: when self-supervised pretraining helps RQ2 asked whether self-supervised pretraining improves allele calling relative to training on labelled data alone. The data-limit study shows a clear conditional answer: pretraining helps when labelled training data is scarce, but the benefit vanishes once enough labelled profiles are available (table 3.7). This positions self-supervised pretraining as a tool for sample efficiency and robustness in low-label regimes, rather than as a universal path to higher asymptotic performance. In practice, this distinction matters because high-quality peak-level annotations are expensive and difficult to obtain, whereas unlabelled electropherograms are comparatively abundant.

Implications for evaluation and forensic use Two findings shape how these results should be interpreted. First, model rankings depend on the evaluation target: ground-truth research data and analyst-annotated casework data reward different behaviours (tables 3.5 and 3.8). Performance, therefore, reflects both modelling quality and alignment with the annotation protocol. Second, all methods fail most often on low-RFU peaks (section 3.7). This failure regime matters in practice because low-intensity peaks are common and can drive the interpretive decision.

These methods could support forensic workflows under two conditions: (i) the full pipeline, including sizing and allele calling, is perfected and validated end to end, and (ii) the system is restricted to peaks outside the low-RFU failure regime. Under these constraints, a machine-learning component could be integrated into the NFI's Snelle ID Lijn[1] for investigative use. An Australian team has already demonstrated the practical deployment of applying machine learning for this task, validating a neural-network reader for STR typing and reporting that it has replaced human readers in some forensic laboratories [60]. For evidentiary reporting in court, where decisions carry possibly major legal consequences, expert analyst review remains unquestionably necessary.

[1] https://www.forensischinstituut.nl/over-het-nfi/algoritmeregister/snelle-id-lijn

Limitations and future work A major limitation is the lack of ground-truth casework annotations, which prevents a direct estimate of accuracy in the target distribution and forces evaluation on research data whose characteristics may not match those of casework. Analyst annotations on casework data provide scale but contain errors, which confounds performance estimates. A recent alternative, multiclass annotations on a limited subset of existing casework profiles, offers higher-quality labels than routine analyst calls (though still short of true ground truth) and additionally distinguishes specific artefact types; future work should prioritise evaluation on this dataset. Beyond data quality, our evaluation also omits the evidential impact: F1 summarises pixel-level agreement, but the practical consequences of an error depend on whether it changes downstream likelihood ratios. Finally, the lack of a public, standardised benchmark limits comparability with other studies and slows cumulative progress.

4

Conclusion

This research studied automated allele calling from STR electropherograms, focusing on two questions: how architectural choices affect prediction performance (RQ1) and whether self-supervised pretraining improves performance when labels are limited or noisy (RQ2). The work contributes novel peak-based and context-aware models, along with a validation strategy that links architectural components to empirical evidence.

Key contributions and novelty The primary contribution is a two-branch, peak-based method for allele calling implemented through three models. First, the *Peak Model* classifies candidate peaks using local fixed-width windows, providing a strong and efficient baseline. Second, the *Autoencoder* learns a compressed latent representation of full electropherograms, enabling global context modelling without allele labels. Third, the *Combined Model* integrates local peak features with the pretrained global representation to improve classification when profile-level context is required. A second contribution is a structured evaluation and ablation strategy that benchmarks these models against established tools and uses results to justify key design choices. A third contribution is an analysis of data characteristics, notably peak height and locus identity, that clarifies where current methods succeed and where they fail.

Answer to RQ1: How do different machine learning model architectures perform with respect to prediction performance in analysing electropherograms? Local peak windows capture much of the signal needed for allele calling: the Peak Model matched DNANet on ground-truth research data. Incorporating global profile context improved performance beyond peak-only classification. The Combined Model outperformed other learning-based methods on ground-truth annotations and matched the strongest baseline on this benchmark, showing that the novel peak-based architectures can achieve state-of-the-art performance.

Answer to RQ2: Does self-supervised pretraining on electropherograms improve allele calling performance versus training using potentially low-quality labels? Self-supervised pretraining of the global encoder improved allele calling performance when only small labelled datasets were available and provided no measurable benefit once ample labelled data were used. This indicates that pretraining is most valuable in the low-label regime, where representation learning can compensate for missing supervision. This positions pretraining as a practical strategy for reducing dependence on costly high-quality annotations.

Outlook The most important next step is validation under forensic decision criteria. Future studies should quantify downstream effects on likelihood-ratio calculations and evaluate performance on casework-like profiles with ground truth or near-ground-truth annotations, ideally including multiclass artefact labels. Further work should also test generalizability across STR kits and laboratory settings and support community progress by creating a standardised benchmark dataset. Together, these steps would enable the responsible translation of peak-based, allele calling into operational workflows.

Global profile context improved allele calling under ground-truth evaluation The Combined Model, which conditions peak classification on a pre-trained autoencoder code, outperformed both the Peak Model and DNANet on NFI research data (Pixel F1 0.934 vs. 0.923 and 0.922; both p=0.001) and matched AutoAnalyzer (0.934 vs. 0.936; p=0.650). The architecture did not require elaborate fusion: MLP and FiLM combiners performed similarly, while attention underperformed on analyst-annotated casework data. Self-supervised pretraining mattered most when labels were scarce: it improved performance at 500 and 1000 training profiles but offered no benefit once training data became abundant. Together, these results indicate that a global context provides discriminative information beyond local peak shape, but the practical impact of pretraining the autoencoder depends on the amount of available training data. Future work could investigate the features encoded in the latent representation and investigate its use for other tasks, such as estimating the number of contributors in a profile.

Low-RFU peaks formed a shared failure regime across all methods Most values in a DNA profile are near zero, and the height of peaks reflects how likely they are to be annotated. Consistent with this distribution, all methods showed markedly reduced F1 below roughly 300 RFU, then improved and plateaued as peak height increased. The methods differed only marginally in this low-RFU region, suggesting that the difficulty reflects signal-to-noise limits and a higher artefact rate rather than model-specific weaknesses. At fixed RFU, performance exceeded what peak height alone would predict, implying that models also exploit peak shape, spacing, and locus- or dye-context. This makes low-RFU discrimination the main bottleneck for operational robustness.

Model rankings depended strongly on the label source and evaluation target Under analyst-annotated casework evaluation, DNANet ranked highest and the Combined Model ranked lower, reversing the ordering seen on ground-truth research data; this pattern indicates that high scores can reflect agreement with an annotation protocol rather than superior biological accuracy. Because no ground-truth casework labels exist, we cannot directly estimate real-world accuracy in the target distribution or resolve whether the ranking reflects true performance differences. A human analyst achieved the highest allele-level F1 on the ground-truth benchmark, and DNANet came closest among automated methods, but the drivers of DNANet's advantage remain unclear. These findings motivate evaluations on higher-quality casework annotations and metrics that connect allele-calling errors to downstream likelihood ratios.

5

Acknowledgements

I would like to thank my supervisor, **Rolf Ypma**, for his weekly guidance, constructive feedback, and the many enjoyable coffee chats that made the process both productive and fun. I am also grateful to my professor, **Thomas Abeel**, for his supervision and for providing the academic framework that supported this work.

I thank the members of the **Bio-DataLab** for the stimulating research environment, helpful discussions, and collegial support throughout the project. I thank **Duncan Taylor** for his time and feedback. I am grateful to the forensic DNA experts **Liza Robles** de Medina and **Francisca Duijs** for sharing their expertise, answering my sometimes impossible questions, and helping me connect the technical work to forensic practice.

Finally, I would like to thank my **friends** and **family** for their support and for their help in reviewing this thesis.

References

[1] Walid M. Abdelmoula et al. "Peak learning of mass spectrometry imaging data using artificial neural networks". In: *Nature Communications* 12.1 (Sept. 20, 2021), p. 5544. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25744-8. URL: https://www.nature.com/articles/s41467-021-25744-8 (visited on 06/24/2025).

[2] Jonathan D. Adelman et al. "Automated detection and removal of capillary electrophoresis artifacts due to spectral overlap". In: *ELECTROPHORESIS* 40.14 (July 2019), pp. 1753–1761. ISSN: 0173-0835, 1522-2683. DOI: 10.1002/elps.201900060. URL: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/10.1002/elps.201900060 (visited on 07/14/2025).

[3] Lauren E. Alfonse et al. "A large-scale dataset of single and mixed-source short tandem repeat profiles to inform human identification strategies: PROVEDIt". In: *Forensic Science International: Genetics* 32 (Jan. 2018), pp. 62–70. ISSN: 18724973. DOI: 10.1016/j.fsigen.2017.10.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497317302144 (visited on 07/01/2025).

[4] Takashi Anazawa, Shuhei Yamamoto, and Ryoji Inaba. "An ultra-small nine-color spectrometer with a two-layer biparted ten-dichroic-mirror array and an image sensor". In: *Scientific Reports* 12.1 (Oct. 3, 2022), p. 16518. ISSN: 2045-2322. DOI: 10.1038/s41598-022-20814-3. URL: https://www.nature.com/articles/s41598-022-20814-3 (visited on 12/05/2025).

[5] Yaqoob Ansari et al. "Deep learning for ECG Arrhythmia detection and classification: an overview of progress for period 2017–2023". In: *Frontiers in Physiology* 14 (Sept. 15, 2023), p. 1246746. ISSN: 1664-042X. DOI: 10.3389/fphys.2023.1246746. URL: https://www.frontiersin.org/articles/10.3389/fphys.2023.1246746/full (visited on 06/24/2025).

[6] *Applied Biosystems Genetic Analysis Data File Format*. Sept. 2009. URL: http://arabidopsis.biology.cofc.edu:3838/shinymarker/ABIF_File_Format.pdf.

[7] David Aquilué-Llorens and Aureli Soria-Frisch. *EEG Artifact Detection and Correction with Deep Autoencoders*. Feb. 12, 2025. DOI: 10.48550/arXiv.2502.08686. arXiv: 2502.08686[cs]. URL: http://arxiv.org/abs/2502.08686 (visited on 07/21/2025).

[8] Mark Barash et al. "Machine learning applications in forensic DNA profiling: A critical review". In: *Forensic Science International: Genetics* 69 (Mar. 2024), p. 102994. ISSN: 18724973. DOI: 10.1016/j.fsigen.2023.102994. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497323001692 (visited on 06/16/2025).

[9] Armen G. Beck et al. "Recent Developments in Machine Learning for Mass Spectrometry". In: *ACS Measurement Science Au* 4.3 (June 19, 2024), pp. 233–246. ISSN: 2694-250X, 2694-250X. DOI: 10.1021/acsmeasuresciau.3c00060. URL: https://pubs.acs.org/doi/10.1021/acsmeasuresciau.3c00060 (visited on 06/24/2025).

[10] Myrte van Belkom. "Exploring deep learning to improve allelic peak calling in forensic DNA analysis". Master's Thesis. Delft: TU Delft, Aug. 2021. URL: https://resolver.tudelft.nl/f0a4d8ec-0d95-441c-ba35-65b087447929.

[11] Corina Benschop, Anouk Backx, and Titia Sijen. "Automated estimation of the number of contributors in autosomal STR profiles". In: *Forensic Science International: Genetics Supplement Series* 7.1 (Dec. 2019), pp. 7–8. ISSN: 1875-1768. DOI: 10.1016/j.fsigss.2019.09.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1875176819302598 (visited on 07/12/2025).

[12] Corina C.G. Benschop et al. "An assessment of the performance of the probabilistic genotyping software EuroForMix: Trends in likelihood ratios and analysis of Type I & II errors". In: *Forensic Science International: Genetics* 42 (Sept. 2019), pp. 31–38. ISSN: 18724973. DOI: 10.1016/j.fsigen.2019.06.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497319301711 (visited on 06/16/2025).

[13] Corina C.G. Benschop et al. "Development and validation of a fast and automated DNA identification line". In: *Forensic Science International: Genetics* 60 (Sept. 2022), p. 102738. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2022.102738. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497322000795 (visited on 07/12/2025).

[14] Corina C.G. Benschop et al. "DNAxs/DNAStatistX: Development and validation of a software suite for the data management and probabilistic interpretation of DNA profiles". In: *Forensic Science International: Genetics* 42 (Sept. 2019), pp. 81–89. ISSN: 18724973. DOI: 10.1016/j.fsigen.2019.06.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497319302285 (visited on 01/09/2026).

[15] Corina C.G. Benschop et al. "Multi-laboratory validation of DNAxs including the statistical library DNAStatistX". In: *Forensic Science International: Genetics* 49 (Nov. 2020), p. 102390. ISSN: 18724973. DOI: 10.1016/j.fsigen.2020.102390. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497320301629 (visited on 01/09/2026).

[16] *Besluit DNA-onderzoek in strafzaken*. July 1, 2024. URL: http://wetten.overheid.nl/jci1.3:c:BWBR0012791 (visited on 08/19/2025).

[17] Sawyer Birnbaum et al. "Temporal FiLM: Capturing Long-Range Sequence Dependencies with Feature-Wise Modulation". In: (2021).

[18] Bernd Brinkmann et al. "Mutation Rate in Human Microsatellites: Influence of the Structure and Length of the Tandem Repeat". In: *The American Journal of Human Genetics* 62.6 (June 1998), pp. 1408–1415. ISSN: 00029297. DOI: 10.1086/301869. URL: https://linkinghub.elsevier.com/retrieve/pii/S0002929707627827 (visited on 08/28/2025).

[19] Christoph Bueschl et al. "PeakBot: machine-learning-based chromatographic peak picking". In: *Bioinformatics* 38.13 (June 27, 2022). Ed. by Olga Vitek, pp. 3422–3428. ISSN: 1367-4803, 1367-4811. DOI: 10.1093/bioinformatics/btac344. URL: https://academic.oup.com/bioinformatics/article/38/13/3422/6590644 (visited on 07/21/2025).

[20] Arthur Buzelin et al. "A CNN-based Local-Global Self-Attention via Averaged Window Embeddings for Hierarchical ECG Analysis". In: (2025).

[21] Yinan Cai, Zhao Meng, and Dian Huang. "DHCT-GAN: Improving EEG Signal Quality with a Dual-Branch Hybrid CNN–Transformer Network". In: *Sensors* 25.1 (Jan. 3, 2025), p. 231. ISSN: 1424-8220. DOI: 10.3390/s25010231. URL: https://www.mdpi.com/1424-8220/25/1/231 (visited on 07/21/2025).

[22] CBS. *Geregistreerde criminaliteit; tijdreeks vanaf 1948*. Mar. 4, 2025. URL: https://www.cbs.nl/nl-nl/cijfers/detail/83723NED (visited on 07/30/2025).

[23] Promega Corporation. *PowerPlex 16 HS System Technical Manual*. Oct. 2023. URL: https://www.promega.com/-/media/files/resources/protocols/technical-manuals/tmd/powerplex-16-hs-system-protocol.pdf (visited on 12/02/2025).

[24] Promega Corporation. *PowerPlex Fusion 6C System Technical Manual*. Jan. 2024. URL: https://www.promega.com/-/media/files/resources/protocols/technical-manuals/tmd/powerplex-fusion-6c-system-protocol.pdf (visited on 12/02/2025).

[25] Lauren Elborough, Duncan Taylor, and Melissa Humphries. *A novel application of Shapley values for large multidimensional time-series data: Applying explainable AI to a DNA profile classification neural network.* Oct. 1, 2024. DOI: 10.21203/rs.3.rs-5001559/v1. URL: https://www.researchsquare.com/article/rs-5001559/v1 (visited on 06/30/2025).

[26] Varun Godbole et al. *Deep Learning Tuning Playbook*. 2023. URL: http://github.com/google-research/tuning_playbook.

[27] C. Heeney et al. "Assessing the Privacy Risks of Data Sharing in Genomics". In: *Public Health Genomics* 14.1 (2011), pp. 17–25. ISSN: 1662-4246, 1662-8063. DOI: 10.1159/000294150. URL: https://karger.com/article/doi/10.1159/000294150 (visited on 01/13/2026).

[28] T. Hicks and R. Coquoz. "Forensic DNA Evidence". In: *Encyclopedia of Biometrics*. Ed. by Stan Z. Li and Anil Jain. Boston, MA: Springer US, 2009, pp. 573–579. ISBN: 978-0-387-73002-8 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_106. URL: http://link.springer.com/10.1007/978-0-387-73003-5_106 (visited on 07/30/2025).

[29] Hassan Ismail Fawaz et al. "Deep Neural Network Ensembles for Time Series Classification". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019 International Joint Conference on Neural Networks (IJCNN). Budapest, Hungary: IEEE, July 2019, pp. 1–6. ISBN: 9781728119854. DOI: 10.1109/IJCNN.2019.8852316. URL: https://ieeexplore.ieee.org/document/8852316/ (visited on 09/02/2025).

[30] Hongchao Ji and Jing Tian. "Deep denoising autoencoder-assisted continuous scoring of peak quality in high-resolution LC–MS data". In: *Chemometrics and Intelligent Laboratory Systems* 231 (Dec. 2022), p. 104694. ISSN: 0169-7439. DOI: 10.1016/j.chemolab.2022.104694. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169743922002052 (visited on 07/21/2025).

[31] Bhabesh Kalita, Nabamita Deb, and Daisy Das. "AnEEG: leveraging deep learning for effective artifact removal in EEG data". In: *Scientific Reports* 14.1 (Oct. 16, 2024). ISSN: 2045-2322. DOI: 10.1038/s41598-024-75091-z. URL: https://www.nature.com/articles/s41598-024-75091-z (visited on 07/21/2025).

[32] Edward D. Kantz et al. "Deep Neural Networks for Classification of LC-MS Spectral Peaks". In: *Analytical Chemistry* 91.19 (Oct. 1, 2019), pp. 12407–12413. ISSN: 0003-2700, 1520-6882. DOI: 10.1021/acs.analchem.9b02983. URL: https://pubs.acs.org/doi/10.1021/acs.analchem.9b02983 (visited on 07/21/2025).

[33] Vernon J Lawhern et al. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". In: *Journal of Neural Engineering* 15.5 (Oct. 1, 2018), p. 056013. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/aace8c. URL: https://iopscience.iop.org/article/10.1088/1741-2552/aace8c (visited on 09/02/2025).

[34] Kun Li, Yingchao Zhang, and Yuanlu Li. "A false peak recognition method based on deep learning". In: *Chemometrics and Intelligent Laboratory Systems* 238 (July 2023), p. 104849. ISSN: 0169-7439. DOI: 10.1016/j.chemolab.2023.104849. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169743923000990 (visited on 07/21/2025).

[35] Michael A. Marciano and Jonathan D. Adelman. "PACE: Probabilistic Assessment for Contributor Estimation— A machine learning-based assessment of the number of contributors in DNA mixtures". In: *Forensic Science International: Genetics* 27 (Mar. 2017), pp. 82–91. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2016.11.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497316302162 (visited on 07/14/2025).

[36] Michael A. Marciano, Victoria R. Williamson, and Jonathan D. Adelman. "A hybrid approach to increase the informedness of CE-based data using locus-specific thresholding and machine learning". In: *Forensic Science International: Genetics* 35 (July 1, 2018), pp. 26–37. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2018.03.017. URL: https://www.sciencedirect.com/science/article/pii/S1872497317303137 (visited on 06/16/2025).

[37] Ryan A. Mccarthy and Ananya Sen Gupta. "Employing and Interpreting a Machine Learning Target-Cognizant Technique for Analysis of Unknown Signals in Multiple Reaction Monitoring". In: *IEEE Access* 9 (2021), pp. 24727–24737. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3056955. URL: https://ieeexplore.ieee.org/document/9351935/ (visited on 04/23/2025).

[38] Arsenty D. Melnikov, Yuri P. Tsentalovich, and Vadim V. Yanshole. "Deep Learning for the Precise Peak Detection in High-Resolution LC–MS Data". In: *Analytical Chemistry* 92.1 (Jan. 7, 2020), pp. 588–592. ISSN: 0003-2700, 1520-6882. DOI: 10.1021/acs.analchem.9b04811. URL: https://pubs.acs.org/doi/10.1021/acs.analchem.9b04811 (visited on 08/05/2025).

[39] Amar Mešić. "Robust DNA Profiling with Synthetic Electropherograms". Master's Thesis. Delft: TU Delft, Oct. 2025. URL: https://resolver.tudelft.nl/uuid:d07c1be2-cfa1-44d5-892f-c2d110e0c9a0.

[40] Nederlands Forensisch Instituut. *DNA-databank voor strafzaken*. 2025. URL: https://dnadatabank.forensischinstituut.nl/ (visited on 08/19/2025).

[41] Ethan Perez et al. *FiLM: Visual Reasoning with a General Conditioning Layer*. Dec. 18, 2017. DOI: 10.48550/arXiv.1709.07871. arXiv: 1709.07871[cs]. URL: http://arxiv.org/abs/1709.07871 (visited on 01/15/2026).

[42] Colorado J Reed et al. "Self-Supervised Pretraining Improves Self-Supervised Pretraining". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 1050–1060. ISBN: 978-1-6654-0915-5. DOI: 10.1109/WACV51458.2022.00112. URL: https://ieeexplore.ieee.org/document/9706748/ (visited on 02/03/2026).

[43] Anne Bech Risum and Rasmus Bro. "Using deep learning to evaluate peaks in chromatographic data". In: *Talanta* 204 (Nov. 2019), pp. 255–260. ISSN: 0039-9140. DOI: 10.1016/j.talanta.2019.05.053. URL: https://linkinghub.elsevier.com/retrieve/pii/S0039914019305375 (visited on 07/21/2025).

[44] Roland1952. *Comparison of a single-stranded RNA and a double-stranded DNA with their corresponding nucleobases*. Mar. 23, 2010. URL: https://en.wikipedia.org/wiki/Nucleic_acid#/media/File:Difference_DNA_RNA-EN.svg (visited on 07/31/2025).

[45] Thermo Fisher Scientific. *AmpFlSTR Plus PCR Amplification Kit User Guide*. 2012. URL: https://documents.thermofisher.com/TFS-Assets/LSG/manuals/4440211_AmpFlSTR_IdentifilerPlus_UG.pdf (visited on 12/02/2025).

[46] Thermo Fisher Scientific. *GlobalFiler PCR Amplification Kit User Guide*. 2013. URL: https://tools.thermofisher.com/content/sfs/manuals/4477595.pdf (visited on 12/02/2025).

[47] Francesco Sessa et al. "Artificial Intelligence and Forensic Genetics: Current Applications and Future Perspectives". In: *Applied Sciences* 14.5 (Mar. 4, 2024), p. 2113. ISSN: 2076-3417. DOI: 10.3390/app14052113. URL: https://www.mdpi.com/2076-3417/14/5/2113 (visited on 06/24/2025).

[48] SoftGenetics. *GeneMarker User Manual*. Version 3.1. Oct. 2019. URL: http://www.softgenetics.com/PDF/GeneMarkerHID_3-1_UserManual_web.pdf.

[49] Ethan Stancliffe and Gary J. Patti. "PeakDetective: A Semisupervised Deep Learning-Based Approach for Peak Curation in Untargeted Metabolomics". In: *Analytical Chemistry* 95.25 (June 27, 2023), pp. 9397–9403. ISSN: 0003-2700, 1520-6882. DOI: 10.1021/acs.analchem.3c00764. URL: https://pubs.acs.org/doi/10.1021/acs.analchem.3c00764 (visited on 07/21/2025).

[50] Mengyu Tan et al. "Explainable artificial intelligence in forensic DNA analysis: Alleles identification in challenging electropherograms using supervised machine learning methods". In: *Forensic Science International: Genetics* 78 (June 2025), p. 103289. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2025.103289. URL: https://www.sciencedirect.com/science/article/pii/S1872497325000699 (visited on 04/28/2025).

[51] Duncan Taylor. "Using a multi-head, convolutional neural network with data augmentation to improve electropherogram classification performance". In: *Forensic Science International: Genetics* 56 (Jan. 2022), p. 102605. ISSN: 18724973. DOI: 10.1016/j.fsigen.2021.102605. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497321001423 (visited on 06/16/2025).

[52] Duncan Taylor and John Buckleton. "Combining artificial neural network classification with fully continuous probabilistic genotyping to remove the need for an analytical threshold and electropherogram reading". In: *Forensic Science International: Genetics* 62 (Jan. 2023), p. 102787. ISSN: 18724973. DOI: 10.1016/j.fsigen.2022.102787. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497322001284 (visited on 06/30/2025).

[53] Duncan Taylor, Ash Harrison, and David Powers. "An artificial neural network system to identify alleles in reference electropherograms". In: *Forensic Science International: Genetics* 30 (Sept. 2017), pp. 114–126. ISSN: 18724973. DOI: 10.1016/j.fsigen.2017.07.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497317301448 (visited on 04/23/2025).

[54] Duncan Taylor and Melissa Humphries. *Simulating realistic short tandem repeat capillary electrophoretic signal using a generative adversarial network*. Aug. 28, 2024. DOI: 10.48550/arXiv.2408.16169. arXiv: 2408.16169[cs]. URL: http://arxiv.org/abs/2408.16169 (visited on 06/30/2025).

[55] Duncan Taylor and Melissa A. Humphries. *deepNoC: A deep learning system to assign the number of contributors to a short tandem repeat DNA profile*. Dec. 13, 2024. DOI: 10.48550/arXiv.2412.09803. arXiv: 2412.09803[cs]. URL: http://arxiv.org/abs/2412.09803 (visited on 06/30/2025).

[56] Duncan Taylor, Michael Kitselaar, and David Powers. "The generalisability of artificial neural networks used to classify electrophoretic data produced under different conditions". In: *Forensic Science International: Genetics* 38 (Jan. 2019), pp. 181–184. ISSN: 18724973. DOI: 10.1016/j.fsigen.2018.10.019. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497318303806 (visited on 06/30/2025).

[57] Duncan Taylor and David Powers. "Teaching artificial intelligence to read electropherograms". In: *Forensic Science International: Genetics* 25 (Nov. 2016), pp. 10–18. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2016.07.013. URL: https://www.sciencedirect.com/science/article/pii/S1872497316301338 (visited on 04/23/2025).

[58] Olivier Tytgat. "BRINGING SCIENCE TO THE SCENE: NOVEL STRATEGIES FOR PORTABLE FORENSIC DNA PROFILING". Master's Thesis. Ghent: Ghent University, Jan. 2022. URL: https://www.researchgate.net/publication/362991666_Bringing_Science_to_the_Scene_Novel_strategies_for_portable_DNA_profiling (visited on 07/31/2025).

[59] Marthe S. Veldhuis et al. "Explainable artificial intelligence in forensics: Realistic explanations for number of contributor predictions of DNA profiles". In: *Forensic Science International: Genetics* 56 (Jan. 2022), p. 102632. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2021.102632. URL: https://linkinghub.elsevier.com/retrieve/pii/S187249732100168X (visited on 07/12/2025).

[60] Luke Volgin et al. "Validation of a neural network approach for STR typing to replace human reading". In: *Forensic Science International: Genetics* 55 (Nov. 2021), p. 102591. ISSN: 1872-4973. DOI: 10.1016/j.fsigen.2021.102591. URL: https://linkinghub.elsevier.com/retrieve/pii/S1872497321001289 (visited on 07/12/2025).

[61] "Walking the tightrope between data sharing and data protection". In: *Nature Medicine* 28.5 (May 2022), pp. 873–873. ISSN: 1078-8956, 1546-170X. DOI: 10.1038/s41591-022-01852-w. URL: https://www.nature.com/articles/s41591-022-01852-w (visited on 01/13/2026).

[62] Abel KJG de Wit et al. "Making AI accessible for forensic DNA profile analysis". In: *bioRxiv* (June 2025). _eprint: https://www.biorxiv.org/content/early/2025/06/05/2025.06.02.656996.full.pdf. DOI: 10.1101/2025.06.02.656996. URL: https://www.biorxiv.org/content/early/2025/06/05/2025.06.02.656996.

[63] Evgenii Zheltonozhskii et al. "Contrast to Divide: Self-Supervised Pre-Training for Learning with Noisy Labels". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 387–397. ISBN: 978-1-6654-0915-5. DOI: 10.1109/WACV51458.2022.00046. URL: https://ieeexplore.ieee.org/document/9706754/ (visited on 02/03/2026).

A

Biological and forensic foundations

This chapter provides the essential biological, genetic, and forensic background necessary to support the reader's understanding of this research. Fundamental concepts of computer science and deep learning are assumed to be known. First, a short introduction to DNA and STRs is given in section A.1. Then, the concepts are examined as applied in forensic science, providing more detailed information relevant to this research in section A.2.

A.1. DNA biochemistry and genetics

Deoxyribonucleic acid (DNA) stores genetic information used for the development and functioning of living cells. DNA is a polymer built from four nucleotides: A(denine), C(ytosine), G(uanine), and T(hymine). It consists of two antiparallel strands forming a double helix (figure A.1). Nucleotides pair specifically as base pairs: A with T and G with C.

DNA is organised into chromosomes. Humans have 23 pairs of chromosomes; for each pair, one chromosome is inherited from the mother and one from the father. All genetic information stored on all the chromosomes combined is called the genome. Chromosomes contain many functional DNA sequences called genes, as well as non-coding regions. A *locus* is a specific position (address) on a chromosome; different DNA sequence variants observed at the same locus are called *alleles*.

Short Tandem Repeats (STRs) are loci where a short DNA sequence (typically 2–6 base pairs) is repeated multiple times. A simple visualisation of STRs can be seen in figure A.2. Individuals may differ in the number of repeats at an STR locus. These variants are the STR *alleles*, commonly represented by their repeat counts (e.g., 12, 13) in forensics. STR loci occur throughout the genome. In forensic DNA analysis, a standard set of tens of STR loci is amplified and measured to form an STR profile. STRs have

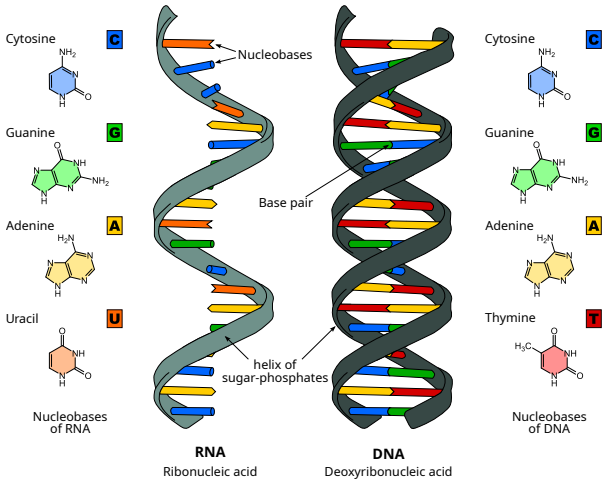


Figure A.1: Visualisation of the structure of RNA and DNA. From: [44]

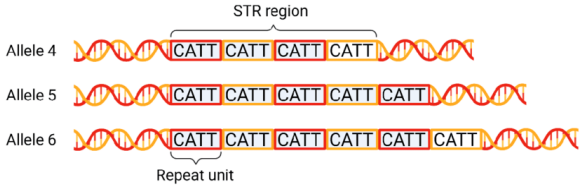


Figure A.2: Simplified visualisation of short tandem repeats. From: [58]

higher mutation rates than many other regions in the DNA [18]. This property can be used to compare STR profiles and determine whether a DNA sample came from the same source (person) as another sample, or to derive family ties.

A.2. Forensic DNA analysis use in practice

To better understand why we should care about forensic DNA analysis and how it is used in practice, we briefly outline the full forensic DNA analysis process, from the crime to sentencing. Afterwards, we take a more thorough look at some parts relevant to this research. Figure A.3 shows the complete process of forensic DNA analysis. This process is explained in section A.2.1. The laboratory techniques (step 2.1 and 2.2) are explained in more detail in section A.2.2. Finally, section A.2.3 explains the current process of digital DNA analysis (steps 3.1 and 3.2) in more detail.

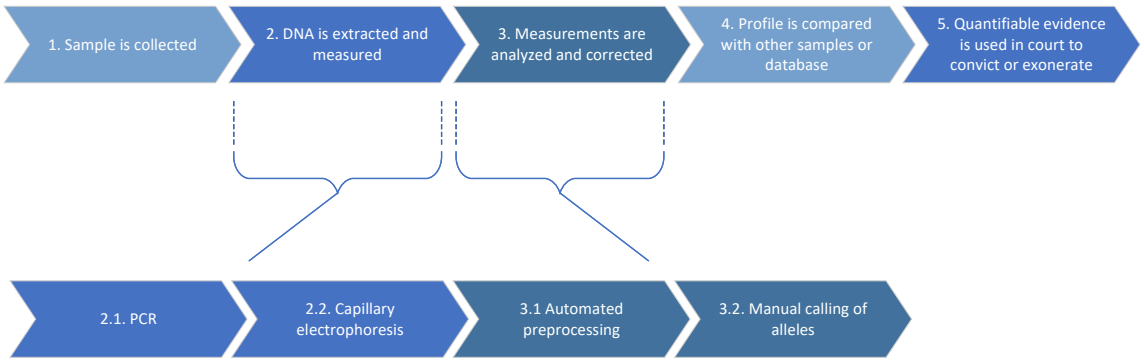


Figure A.3: Big picture overview of the process of Forensic DNA Analysis

A.2.1. Big picture: from crime to sentencing

When a crime occurs, the police secure the scene, interview witnesses, and collect biological traces such as blood, saliva, skin cells, or hair. These samples might contain DNA from multiple people, or even animals. Furthermore, these samples can contain only low concentrations of DNA or be low quality. The contributors to a trace are unknown, and the objective of forensic DNA analysis is to identify them. Police can also collect reference DNA samples voluntarily from already identified individuals. References are generally high-quality measurements that contain DNA from only a single, known contributor.

After the samples are collected, they are documented to preserve the chain of custody and transferred to the Netherlands Forensic Institute (NFI). In the laboratory, DNA is extracted, and STR loci are amplified and measured to generate a DNA profile. Details of this process are described in section A.2.2.

Forensic analysts then interpret the resulting profile and either compare it to reference profiles from suspects or victims, or search it against the Dutch National DNA Database [40]. In some cases, the profile can also be searched against DNA databases from other countries. A match (or database hit) can provide an investigative lead or serve as potential proof for sentencing in court. For court use, the evidential strength is quantified and reported using statistical software [14, 15].

During the criminal trial, forensic experts present their findings to the court, where they are considered alongside other forms of evidence, such as witness testimony or digital traces [16]. Based

on this evidence, the judge can make an informed decision. DNA analysis can contribute to securing convictions, but also to excluding innocent individuals from suspicion.

A.2.2. Laboratory techniques for forensic DNA analysis

Laboratory processing turns a biological trace into an STR profile. First, DNA is extracted by breaking open cells to release it, then cleaned to remove substances that can interfere with later steps. The extracted DNA is then used for amplification.

Next, the chosen STR loci are copied many times using polymerase chain reaction (PCR). In practice, laboratories use commercial STR kits that contain primers for a fixed set of STR loci. Different kits can target different loci and use different chemicals, so the results of different kits are not always directly comparable. Appendix D shows the differences between four commonly used STR kits. Primers are short, single-stranded DNA sequences complementary to the targeted DNA sequence. They create a starting point from which the targeted region can be copied.

PCR repeats three temperature-controlled steps: denaturation separates the DNA strands, annealing allows primers to bind to the target loci, and extension copies the DNA. This cycle is repeated about 25–30 times, increasing the amount of target DNA to a level sufficient for measurement, even if only a small amount was present initially.

Once we have extracted and amplified sufficient DNA, we can begin the measurements. Capillary electrophoresis (CE) is the process of measuring the length of DNA fragments. When a high voltage is applied, negatively charged DNA fragments migrate toward the anode. Shorter fragments traverse the capillary more quickly than longer ones, producing size-based separation. A laser excites the fluorescent dyes on the fragments as they pass a detection window, and the emitted light is recorded as an electropherogram (EPG). A simple diagram of this process can be seen in figure A.4. Generally, primers containing unique fluorescent dyes are used to enable the simultaneous measurement of multiple loci. The unit of measurement is relative fluorescent units (RFU). Dyes are ordered by fluorescence emission spectrum. The colour labels commonly assigned to them do not correspond to their actual wavelengths [4].

Each injection in electrophoresis includes an internal size standard to convert migration time to the length of a DNA segment in base pairs. An allelic ladder is used to convert segment lengths into allele designations (the number of repeats) at each locus. Analysts then apply thresholds, assess peak height ratios for heterozygotes, and evaluate mixture indicators to call genotypes. Details about the work of analysts are discussed in section A.2.3.

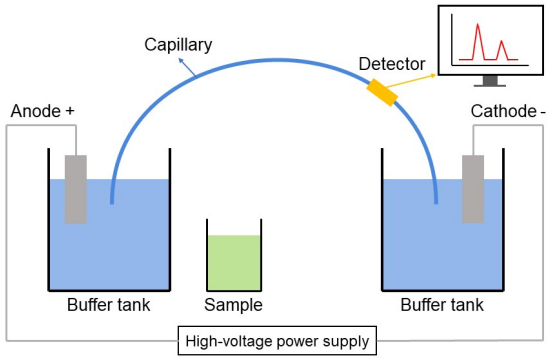


Figure A.4: Capillary Electrophoresis. A charge is applied to move (negatively charged) DNA material through the capillary. Small DNA fragments move faster through the capillary. When the detector measures DNA, peaks appear in the output signal. From: [1]

A.2.3. Digital DNA interpretation techniques

After the DNA has been measured, it must be analysed and interpreted. We briefly describe the protocol that is used at the NFI. The result of capillary electrophoresis is called an *electropherogram*. As discussed in section A.2.2, it measures multiple signals simultaneously, using different dyes. An example using

[1] https://www.ste-mart.com/capillary-electrophoresis.htm

Dye	AT	LT
FL-6C (blue)	95 RFU	45 RFU
JOE-6C (green)	140 RFU	40 RFU
TMR-6C (black)	85 RFU	45 RFU
CXR-6C (red)	135 RFU	80 RFU
TOM-6C (purple)	95 RFU	40 RFU
Max heterozygote imbalance	3%	2%

Table A.1: Thresholds used for manual DNA profile analysis on PPF6C. Peaks below the threshold are skipped and never annotated as allelic.

the STR kit PPF6C is shown in figure A.5. We can see six separate one-dimensional signals that were measured simultaneously. The last dye (yellow) contains the internal size standard.

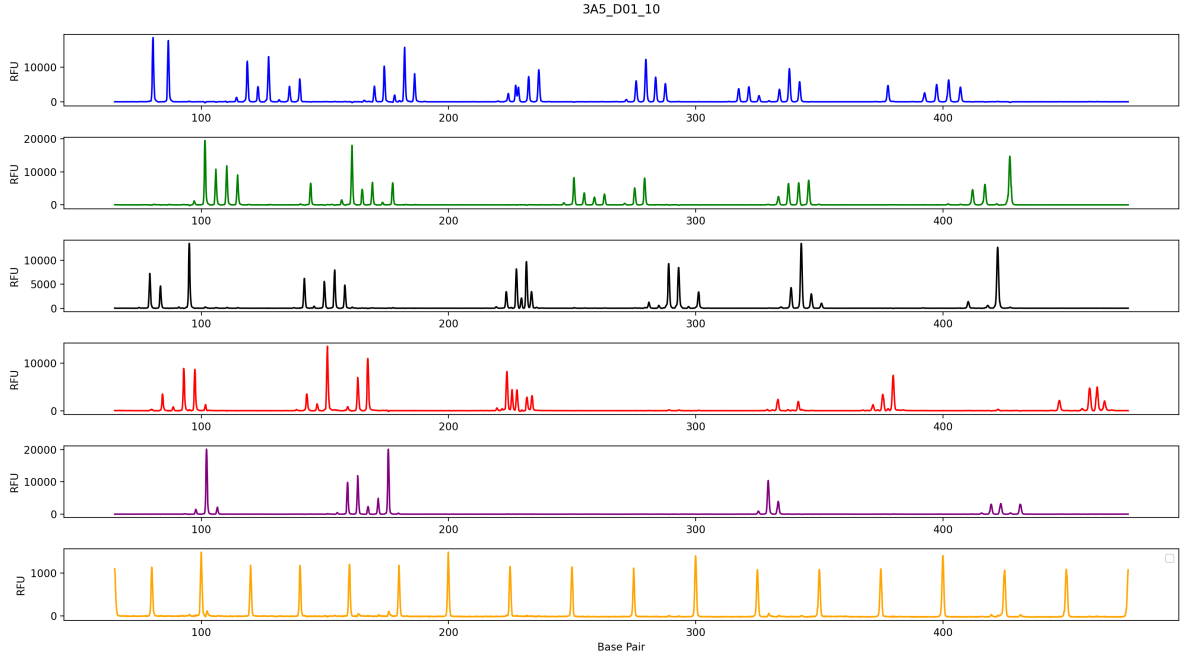


Figure A.5: An example electropherogram. Six dyes are measured simultaneously. The location of peaks indicates the length in base pairs of the detected (DNA) material. The height of the peaks indicates the amount of detected material in relative fluorescence units (RFU).

Analysts use commercial software, such as GeneMarker [48], to analyse an electropherogram. GeneMarker applies multiple preprocessing steps before presenting the data to analysts; these steps include smoothing, baseline subtraction, and artefact removal [48, p. 14]. The main task in digital DNA analysis is allele calling. This process involves identifying the alleles at each locus. Unfortunately, an electropherogram often contains many artefacts, making it difficult to determine whether a peak originates from an allele or an artefact. Section A.2.4 explains the types of artefacts in more detail.

To ensure consistency and reduce workload, set thresholds are used in the analysis. Peaks below the analytical threshold (AT) are not considered allelic and are discarded. When a profile is of low quality or insufficiently informative, the low threshold (LT) is used. Table A.1 shows these thresholds for the STR kit PPF6C. We can see that the threshold varies per dye. In addition to these set thresholds, a fractional threshold is used. The fractional threshold is based on the highest peak in the profile.

To support consistent allele calling, GeneMarker uses predefined allele bins supplied with the PPF6C kit. For each locus, the allelic ladder defines the expected fragment size for each allele; the software converts these into small intervals (bins) around each expected allele. During analysis, each peak that exceeds the relevant threshold is assigned to the nearest bin, provided its size lies within the bin tolerance. The bin label (for example, allele 10 or 14) is then reported as the called allele for that locus. Peaks that do not fall within any bin, or that fall in ambiguous regions between bins, are manually

assigned bins by analysts.

Ambiguous peaks require manual review. The shape of the peak is an important factor for analysts in determining whether it is allelic. They expect allelic peaks to be smooth and approximately Gaussian, and they check whether each peak fits the local pattern at that locus (e.g., common artefact positions and peak-height relationships). However, the global profile context can also have an important impact; some artefact types often occur multiple times throughout a profile. The number of contributors can also be an important factor in the allele call.

When an analyst has assigned every allele to a bin, a list of alleles and their corresponding peak heights is created. This list can be used for comparison with other profiles, as described in section A.2.1.

A.2.4. DNA profile data artefact types

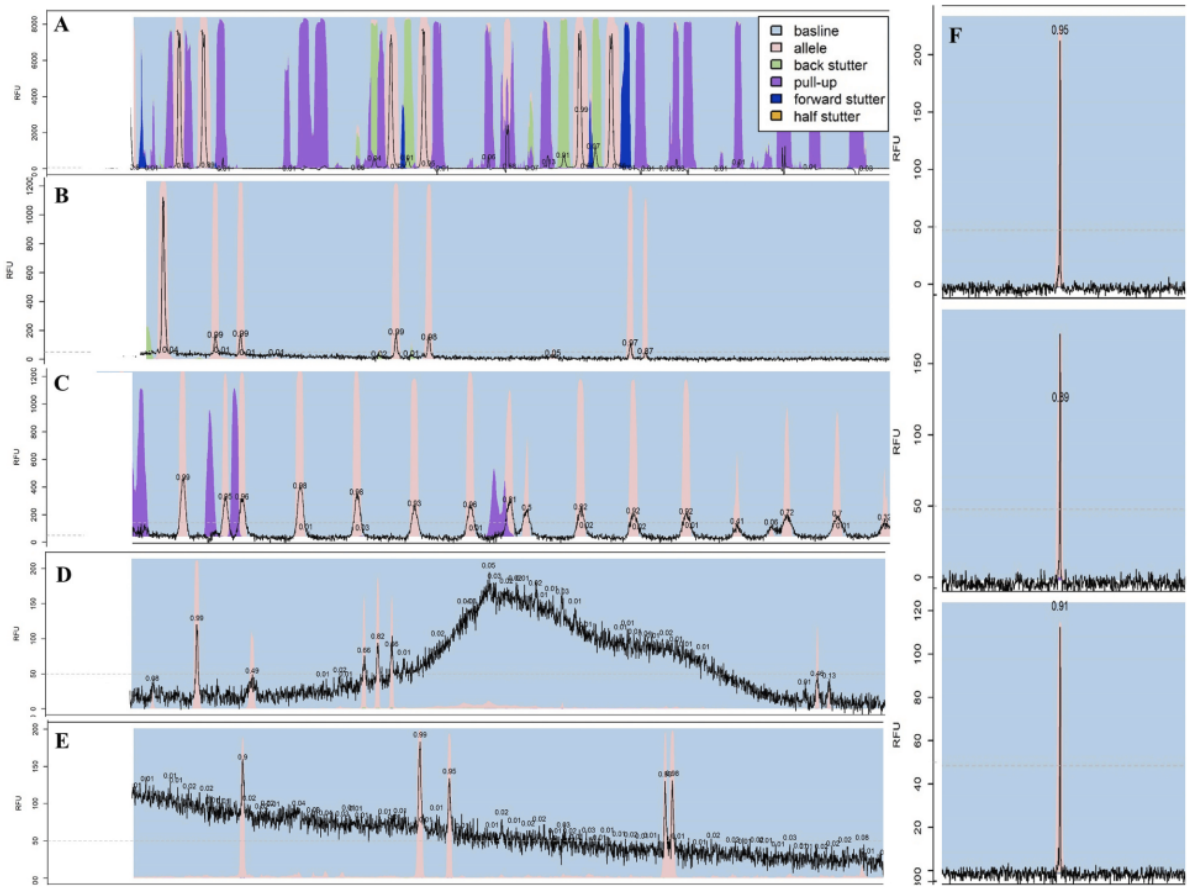


Figure A.6: Selection of dye lanes from DNA profiles showing a range of electrophoretic signal in the training data; A) overloaded DNA profile, B) high degradation, C) broadening peaks over the scan range, D) raised areas of baseline drift, E) slow tailing off of baseline coming out of primer flare region, and F) a spike through the dye lanes. From: [51]

When analysing an electropherogram, different types of artefacts can occur. To differentiate between these artefacts and allelic peaks, we need to understand their characteristics. Some of these artefacts are biological, and some originate from the equipment used. The most common problems are:

- **Baseline noise:** Random, low-amplitude fluctuations in the fluorescence signal.
- **Baseline drift:** A gradual change in the baseline level over time. (figure A.6 D and E)
- **Stutters:** By-products of PCR in which peaks are generated one (a half, or two) repeat unit larger (forward stutter) or smaller (backward stutter) than the true allele; these typically appear at low relative heights (e.g. 5–15% of the main peak) and at predictable base pair distances from the allele. Backward stutters are the most common types of artefacts.
- **Spectral bleed through:** When a strong peak in one dye channel "leaks" into another channel

because of overlapping fluorophore emission spectra or imperfect filter separation, it produces false peaks at the same scan point.

- **Allele dropout:** Failure to detect a true allelic peak.
- **Peak collapse:** Multiple peaks merge into a single broadened peak.
- **Random spikes:** Spikes are typically narrow and sharp peaks that can appear for any number of reasons. Spikes can sometimes appear in a single dye but are easily identified by their presence across multiple dyes. (figure A.6 F)
- **DNA degradation:** Because longer parts of DNA can be more difficult to amplify, peaks can decrease in height gradually over time in a sample. (figure A.6 B)
- **Overloaded profile:** A sample containing too much DNA can cause an overloaded profile where many (non-allelic) peaks appear, peaks overlap, and it is difficult to distinguish between contributors. (figure A.6 A)
- **Dye blobs:** Dye blobs are low but wide peaks, often visible in multiple dyes.
- **Shoulder peaks (± A):** Shoulder peaks are peaks adjacent (a single base pair distance) to a true allele caused by incomplete PCR. Shoulder peaks can sometimes also cause peak collapse.
- **Foreign DNA:** Peak that looks allelic but often falls outside of expected bins. It can be caused by DNA originating from animals or bacteria.

B

Ethical considerations of use of forensic DNA profiles

This research uses forensic DNA data that was collected without explicit consent. In forensic practice, obtaining consent is often impossible or inappropriate: crime scene traces originate from unknown contributors; reference samples can be collected through legal authorisation; victim identification and unidentified remains cases involve incapacitated or deceased persons; and some cold case evidence was gathered long before contact with potential donors was feasible. Processing such data can be ethically justified under the principles of necessity, proportionality, and clear public-interest aims, such as preventing and resolving serious crime.

This research aims to improve forensic DNA analysis by making it more effective, faster, and cheaper. Therefore, the research has a clear public interest: it can result in more crimes being solved, where DNA analysis is currently not viable due to resource or time constraints.

The biggest risk is the model's potential for incorrect predictions, which could lead to wrongful convictions or exonerations. To mitigate these risks, the model will be thoroughly validated across several fronts and compared with human analysts before being deployed in production. Full validation of the model, including its impact on likelihood ratios, is outside the scope of this research.

We considered substituting real casework data with synthetic or alternative data sources. However, a domain shift would likely degrade performance and increase the risk of error, potentially leading to wrongful convictions. We consider that careful use of real-world case data reduces overall risk compared with training on non-representative data.

The risk of accidental or malicious publication of the training data is addressed in section 2.2.1.

This research does not introduce any new risks to the DNA donors. We analyse only electropherogram signal characteristics relevant to the research questions; no biographical or identifying information is used or available to the authors.

We will document datasets, preprocessing, and model versions and make non-sensitive artefacts available where possible. Model outputs are not used in production; the final interpretation of the profiles remains with qualified forensic scientists.

In conclusion, while this research necessarily involves pseudonymised data collected without explicit consent, we have implemented safeguards to mitigate these risks. By handling data and maintaining transparency in our research processes, we strive to ensure that the public-interest benefits of this work outweigh the associated risks.

C

Experimental results

C.1. Peak model ablation results

data_max_rfu_value	F1 score mean ± std	Δ F1 score (vs 5000.0)	p (vs 5000.0)
5000.0	0.940 ± 0.002	-	-
15000.0	0.939 ± 0.002	0.001	0.827
33000.0	0.938 ± 0.003	0.001	0.774
None	0.941 ± 0.004	0.001	0.958

Table C.1: Ablation study of data_max_rfu_value

data_smooth_keep_factor	F1 score mean ± std	Δ F1 score (vs None)	p (vs None)
None	0.940 ± 0.003	-	-
0.2	0.938 ± 0.002	0.001	0.533
0.4	0.939 ± 0.003	0.000	1.000
0.6	0.940 ± 0.003	0.000	1.000
0.8	0.940 ± 0.002	0.000	1.000

Table C.2: Ablation study of data_smooth_keep_factor

data_threshold	F1 score mean ± std	Δ F1 score (vs 25)	p (vs 25)
25	0.938 ± 0.003	-	-
15	0.935 ± 0.002	0.003	0.217
35	0.930 ± 0.004	0.007	0.008
45	0.925 ± 0.003	0.013	<0.001

Table C.3: Ablation study of data_threshold

data_window_size	F1 score mean ± std	Δ F1 score (vs 120)	p (vs 120)
120	0.938 ± 0.002	-	-
180	0.939 ± 0.001	0.001	0.648
20	0.923 ± 0.002	0.016	<0.001

Table C.4: Ablation study of data_window_size

data_include_max_pool_dyes	F1 score mean ± std	Δ F1 score (vs False)	p (vs False)
False	0.939 ± 0.003	-	-
True	0.940 ± 0.001	0.001	-

Table C.5: Ablation study of data_include_max_pool_dyes

model_activation	F1 score mean ± std	Δ F1 score (vs gelu)	p (vs gelu)
gelu	0.939 ± 0.002	-	-
relu	0.939 ± 0.002	0.000	1.000
tanh	0.939 ± 0.002	0.000	1.000

Table C.6: Ablation study of model_activation

model_use_batchnorm	F1 score mean ± std	Δ F1 score (vs False)	p (vs False)
False	0.939 ± 0.002	-	-
True	0.938 ± 0.003	0.001	0.226

Table C.7: Ablation study of model_use_batchnorm

data_data_loading_strategy	F1 score mean ± std	Δ F1 score (vs analyzed)	p (vs analyzed)
analyzed	0.939 ± 0.002	-	-
superior	0.939 ± 0.003	0.000	0.650

Table C.8: Ablation study of data_data_loading_strategy

model_downsample	F1 score mean ± std	Δ F1 score (vs conv)	p (vs conv)
conv	0.941 ± 0.001	-	-
maxpool	0.939 ± 0.002	0.002	<0.001

Table C.9: Ablation study of model_downsample (5 random seeds)

model_conv_dropout_p	F1 score mean ± std	Δ F1 score (vs 0.0)	p (vs 0.0)
0.0	0.939 ± 0.002	-	-
0.1	0.938 ± 0.001	0.002	0.302
0.25	0.927 ± 0.004	0.012	0.003
0.5	0.883 ± 0.009	0.057	<0.001

Table C.10: Ablation study of model_conv_dropout_p

model_head_dropout_p	F1 score mean ± std	Δ F1 score (vs 0.0)	p (vs 0.0)
0.0	0.939 ± 0.002	-	-
0.1	0.939 ± 0.002	0.001	1.000
0.25	0.939 ± 0.002	0.001	1.000
0.5	0.939 ± 0.001	0.001	1.000

Table C.11: Ablation study of model_head_dropout_p

data_filter_peaks	F1 score mean ± std	Δ F1 score (vs False)	p (vs False)
False	0.939 ± 0.003	-	-
True	0.914 ± 0.003	0.025	<0.001

Table C.12: Ablation study of data_filter_peaks

model_channels	F1 score mean ± std	Δ F1 score (vs [16, 32, 64])	p (vs [16, 32, 64])
16, 32, 64	0.940 ± 0.002	-	-
16, 32	0.939 ± 0.002	0.001	1.000
32, 64, 128, 256	0.940 ± 0.001	0.000	1.000
32, 64, 128	0.939 ± 0.002	0.001	1.000
32, 64	0.939 ± 0.002	0.001	1.000
64, 128	0.940 ± 0.001	0.000	1.000

Table C.13: Ablation study of model_channels

model_kernel_size	F1 score mean ± std	Δ F1 score (vs 7)	p (vs 7)
7	0.939 ± 0.002	-	-
1	0.938 ± 0.001	0.001	1.000
3	0.939 ± 0.002	0.000	1.000
5	0.938 ± 0.003	0.001	1.000
9	0.939 ± 0.002	0.000	1.000

Table C.14: Ablation study of model_kernel_size

data_log_scale	F1 score mean ± std	Δ F1 score (vs False)	p (vs False)
False	0.939 ± 0.002	-	-
True	0.941 ± 0.004	0.002	0.058

Table C.15: Ablation study of data_log_scale

model_pooling	F1 score mean ± std	Δ F1 score (vs attn)	p (vs attn)
flat	0.939 ± 0.002	-	-
attn	0.742 ± 0.019	0.179	0.022
avg	0.565 ± 0.021	0.374	<0.001

Table C.16: Ablation study of model_pooling

C.2. Autoencoder ablation study

model_architecture	MSE score mean ± std	Δ MSE score (vs cnn_1d)	p (vs cnn_1d)
cnn_1d	1447 ± 154	-	-
cnn_1d_shared	1737 ± 734	290	0.509
cnn	43022 ± 4101	41575	<0.001

Table C.17: Ablation study of model_architecture

model_depth	MSE score mean ± std	Δ MSE score (vs 3)	p (vs 3)
3	1471 ± 133	-	-
4	2455 ± 258	984	<0.001

Table C.18: Ablation study of model_depth

model_preprocessing_log_scale	MSE score mean ± std	Δ MSE score (vs True)	p (vs True)
True	1436 ± 158	-	-
False	1508 ± 119	72	0.173

Table C.19: Ablation study of model_preprocessing_log_scale

model_preprocessing_max_rfu_scale_value	MSE score mean ± std	Δ MSE score (vs 33000)	p (vs 33000)
33000	1431 ± 88	-	-
None	202369 ± 13277	200938	<0.001

Table C.20: Ablation study of model_preprocessing_max_rfu_scale_value

C.3. Combined Model ablation study

model_freeze_autoencoder	F1 score mean ± std	Δ F1 score (vs False)	p (vs False)
False	0.890 ± 0.003	-	-
True	0.890 ± 0.003	0.000	-

Table C.21: Ablation study of model_freeze_autoencoder

model_hidden_dims	F1 score mean ± std	Δ F1 score (vs [64, 32])	p (vs [64, 32])
64, 32	0.889 ± 0.004	-0.002	-
16, 32, 64	0.887 ± 0.004	-	1.000
16, 32	0.889 ± 0.004	-0.002	1.000
32, 16	0.889 ± 0.003	-0.002	1.000
32, 64, 128	0.887 ± 0.003	0.000	1.000
32, 64	0.891 ± 0.003	-0.004	0.388
64, 32, 16	0.887 ± 0.003	0.000	1.000

Table C.22: Ablation study of model_hidden_dims

data_limit	F1 score mean ± std	Δ F1 score (vs None)	p (vs None)
None	0.936 ± 0.002	-	-
1000.0	0.931 ± 0.004	0.005	0.069
10000.0	0.937 ± 0.004	-0.000	0.789
500.0	0.924 ± 0.006	0.013	<0.001

Table C.23: Ablation study of data_limit with a pretrained autoencoder

data_limit	F1 score mean ± std	Δ F1 score (vs None)	p (vs None)
None	0.937 ± 0.001	-	-
1000.0	0.923 ± 0.004	0.014	0.005
10000.0	0.936 ± 0.002	0.001	0.592
500.0	0.912 ± 0.006	0.024	<0.001

Table C.24: Ablation study of data_limit with an untrained autoencoder

C.4. Locus pairwise comparison over models

Table C.25: Pairwise Dunn post-hoc p-values between loci.

Locus	CSF1PO	D10S1248	D12S391	D13S317	D16S539	D18S51	D19S433	D1S1656	D21S11	D22S1045	D2S1338	D2S441	D3S1358	D5S818	D7S820	D8S1179	FGA	Penta D	Penta E	SE33	TH01	TPOX	vWA
CSF1PO	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.277	1	0.678
D10S1248	1	1	1	0.428	1	1	1	1	0.872	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D12S391	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.09	1	0.243
D13S317	1	0.428	1	1	0.141	1	1	0.487	1	1	1	1	0.1	1	1	0.057	0.507	1	1	1	0.003	1	0.009
D16S539	1	1	1	0.141	1	1	1	1	0.309	1	1	1	1	1	0.42	1	1	1	1	1	1	1	1
D18S51	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.681	1	1
D19S433	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.154	1	0.384
D1S1656	1	1	1	0.487	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.043	1	0.124
D21S11	1	0.872	1	1	0.309	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.028	1	0.083
D22S1045	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.641	1	1
D2S1338	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.526	1	1
D2S441	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.79	1	1
D3S1358	1	1	1	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.001	1	0.004
D5S818	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.576	1	1
D7S820	1	1	1	1	0.42	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.038	1	0.112
D8S1179	1	1	1	0.057	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.001	1	0.004
FGA	1	1	1	0.507	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.035	1	0.103
Penta D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.409	1	1
Penta E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.409	1	1
SE33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TH01	0.277	1	0.09	0.003	1	0.681	0.154	0.043	0.028	0.641	0.526	0.79	0.001	0.576	0.038	0.001	0.035	0.409	0.409	1	1	1	1
TPOX	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
vWA	0.678	1	0.243	0.009	1	1	0.384	0.124	0.083	1	1	1	0.004	1	0.112	0.004	0.103	1	1	1	1	1	1

C.5. Locus F1 scores per model

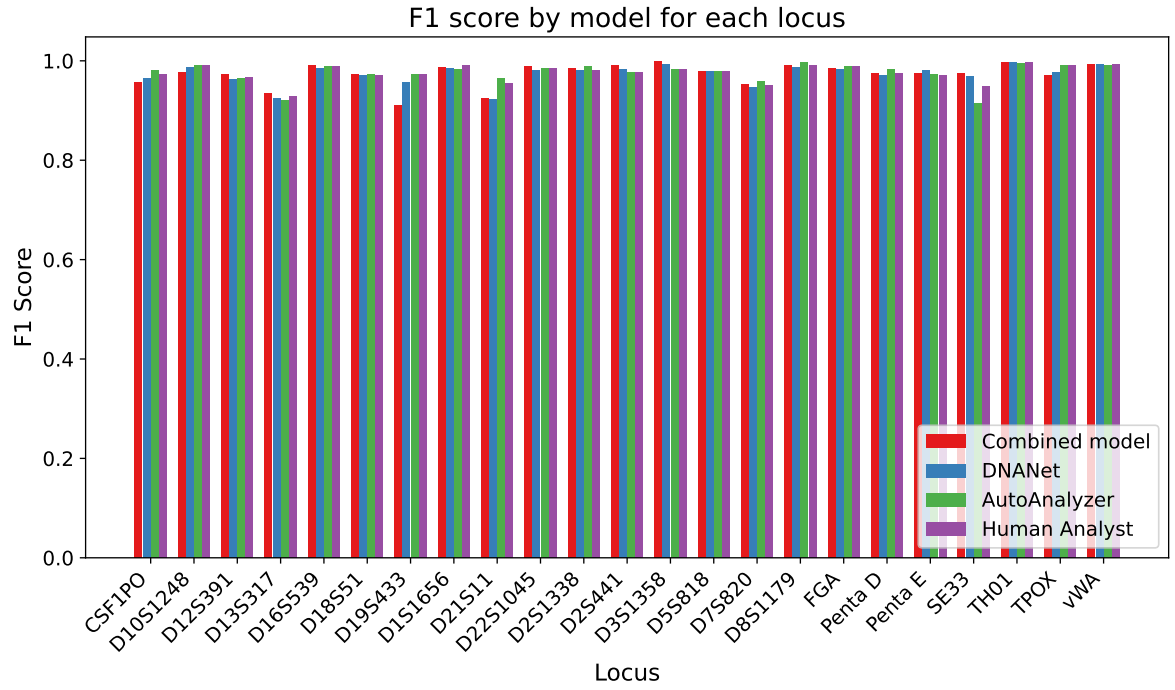


Figure C.1: Per locus F1 scores for four models: Peak Model, Combined Model, DNANet, AutoAnalyzer

D

STR kit comparison

Table D.1: Dye label and colour for each STR locus in four STR kits (or – if the locus is absent). Data from: [23, 24, 45, 46].

Locus	PowerPlex Fusion 6C	GlobalFiler	PowerPlex 16 HS	Identifiler Plus
Amelogenin	FL-6C (blue)	VIC (green)	TMR (yellow)	PET (red)
CSF1PO	JOE-6C (green)	6-FAM (blue)	JOE (green)	6-FAM (blue)
D10S1248	FL-6C (blue)	SID (purple)	–	–
D12S391	CXR-6C (red)	SID (purple)	–	–
D13S317	FL-6C (blue)	TAZ (red)	JOE (green)	VIC (green)
D16S539	JOE-6C (green)	6-FAM (blue)	JOE (green)	VIC (green)
D18S51	JOE-6C (green)	VIC (green)	Fluorescein (blue)	NED (yellow)
D19S433	CXR-6C (red)	NED (yellow)	–	NED (yellow)
D1S1656	FL-6C (blue)	SID (purple)	–	–
D21S11	TMR-6C (yellow)	VIC (green)	Fluorescein (blue)	6-FAM (blue)
D22S1045	CXR-6C (red)	TAZ (red)	–	–
D2S1338	JOE-6C (green)	SID (purple)	–	VIC (green)
D2S441	FL-6C (blue)	NED (yellow)	–	–
D3S1358	FL-6C (blue)	6-FAM (blue)	Fluorescein (blue)	VIC (green)
D5S818	TMR-6C (yellow)	TAZ (red)	JOE (green)	PET (red)
D7S820	TMR-6C (yellow)	TAZ (red)	JOE (green)	6-FAM (blue)
D8S1179	CXR-6C (red)	VIC (green)	TMR (yellow)	6-FAM (blue)
DYS391	TOM-6C (purple)	VIC (green)	–	–
DYS570	TOM-6C (purple)	–	–	–
DYS576	TOM-6C (purple)	–	–	–
FGA	TOM-6C (purple)	NED (yellow)	TMR (yellow)	PET (red)
Penta D	JOE-6C (green)	–	JOE (green)	–
Penta E	FL-6C (blue)	–	Fluorescein (blue)	–
SE33	CXR-6C (red)	TAZ (red)	–	–
TH01	TMR-6C (yellow)	NED (yellow)	Fluorescein (blue)	VIC (green)
TPOX	TMR-6C (yellow)	6-FAM (blue)	TMR (yellow)	NED (yellow)
vWA	TMR-6C (yellow)	6-FAM (blue)	TMR (yellow)	NED (yellow)
Yindel	–	VIC (green)	–	–

E

Hyperparameters and package versions

E.1. Hyperparameters

Listing E.1: Hyperparameters for the Peak Model

```
data:
  datasets:
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
    include_size_standard: true
    limit: 5000
    name: train_case_data_A
    skip_invalid_ladders: true
    use_cache: true
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
    include_size_standard: true
    limit: null
    name: train_case_data_B_LT
    skip_invalid_ladders: true
    use_cache: true
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
    include_size_standard: true
    name: train_case_data_C_LT
    skip_invalid_ladders: true
    use_cache: true
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
    include_size_standard: true
```

```
      name: train_case_data_D_LT
      skip_invalid_ladders: true
      use_cache: true
    - adjustment_of_annotations: top
      data_loading_strategy: superior
      ground_truth_as_annotations: true
      include_size_standard: true
      limit: null
      name: train_RD_data
      skip_invalid_ladders: true
      use_cache: true
    name: combined
model:
    autoencoder: null
    autoencoder_checkpoint: null
    combiner_strategy: mlp
    context: null
    default_label: noise
    freeze_autoencoder: false
    hidden_dims:
    - 32
    - 64
    name: combined_classification
    peak_classifier:
      activation: relu
      channels:
      - 32
      - 64
      conv_dropout_p: 0
      downsample: maxpool
      head_dropout_p: 0
      include_marker: true
      include_max_pool_dyes: false
      kernel_size: 7
      labels:
      - noise
      - allele
      loss_fn: cross_entropy
      model: v2
      name: dnanet_peak_classification
      pooling: flat
      use_batchnorm: false
      window_size: 120
    peak_classifier_checkpoint: null
    task: segmentation
    threshold: 25
training:
    balancer: null
    batch_size: 32
    gamma: 0.95
    learning_rate: 0.00015
    min_delta: 0
    num_epochs: 200
    save_best: true
    steps_per_epoch: null
    task: segmentation
    tensorboard: false
    use_evaluation_metric: true
    use_scheduler: true
    weight_decay: 0.0005
```

Listing E.2: Hyperparameters for the Autoencoder Model

```yaml
data:
  data_loading_strategy: superior
  hash:
  - 0
  - 7
  include_size_standard: true
  name: train_case_data_B
  skip_invalid_ladders: true
  use_cache: true
model:
  architecture: cnn_1d
  compression: 8
  depth: 3
  device: null
  hidden_dims: 64
  input_dyes: 6
  kernel_size: 5
  loss_function: mse
  name: dnanet_autoencoder
  preprocessing_log_scale: false
  preprocessing_max_rfu_scale_value: 33000
  preprocessing_smooth_keep_factor: null
  signal_length: 4096
  task: replication
  use_batchnorm: false
  use_sigmoid: true
training:
  balancer: null
  batch_size: 128
  gamma: 0.8
  learning_rate: 0.0001
  min_delta: 0
  num_epochs: 150
  save_best: true
  steps_per_epoch: null
  task: segmentation
  tensorboard: false
  use_evaluation_metric: true
  use_scheduler: false
  weight_decay: 0.0005
```

Listing E.3: Hyperparameters for the Combined Model

```yaml
data:
  datasets:
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
    include_size_standard: true
    limit: 5000
    name: train_case_data_A
    skip_invalid_ladders: true
    use_cache: true
  - adjustment_of_annotations: top
    data_loading_strategy: superior
    hash:
    - 0
    - 7
```

```
      include_size_standard: true
      limit: null
      name: train_case_data_B_LT
      skip_invalid_ladders: true
      use_cache: true
    - adjustment_of_annotations: top
      data_loading_strategy: superior
      hash:
      - 0
      - 7
      include_size_standard: true
      name: train_case_data_C_LT
      skip_invalid_ladders: true
      use_cache: true
    - adjustment_of_annotations: top
      data_loading_strategy: superior
      hash:
      - 0
      - 7
      include_size_standard: true
      name: train_case_data_D_LT
      skip_invalid_ladders: true
      use_cache: true
    - adjustment_of_annotations: top
      data_loading_strategy: superior
      ground_truth_as_annotations: true
      include_size_standard: true
      limit: null
      name: train_RD_data
      skip_invalid_ladders: true
      use_cache: true
    name: combined
  model:
    autoencoder: null
    autoencoder_checkpoint: scratch/AmiableOutgoingFlounderOfTolerance
    combiner_strategy: mlp
    context: null
    default_label: noise
    freeze_autoencoder: false
    hidden_dims:
    - 64
    - 32
    name: combined_classification
    peak_classifier:
      activation: relu
      channels:
      - 32
      - 64
      conv_dropout_p: 0
      downsample: maxpool
      head_dropout_p: 0
      include_marker: true
      include_max_pool_dyes: true
      kernel_size: 7
      labels:
      - noise
      - allele
      loss_fn: cross_entropy
      model: v2
      name: dnanet_peak_classification
      pooling: flat
```

```
    use_batchnorm: false
    window_size: 120
  peak_classifier_checkpoint: null
  task: segmentation
  threshold: 25
training:
  balancer: null
  batch_size: 32
  learning_rate: 0.00015
  min_delta: 0
  num_epochs: 200
  save_best: true
  steps_per_epoch: null
  task: segmentation
  tensorboard: false
  use_evaluation_metric: true
  use_scheduler: true
  weight_decay: 0.0005
```

E.2. Package versions

Table E.1: List of Packages and Their Versions

Package	Version	Package	Version
absl_py	1.4.0	adjustText	1.3.0
aiohappyeyeballs	2.4.4	aiohttp	3.11.10
aiosignal	1.3.1	alembic	1.14.0
altair	5.5.0	appdirs	1.4.4
array_record	0.5.1	astunparse	1.6.3
async-timeout	5.0.1	attrs	24.2.0
beautifulsoup4	4.12.3	blinker	1.9.0
bokeh	3.6.2	cachetools	5.5.0
certifi	2024.8.30	charset-normalizer	3.4.0
click	8.1.7	cloudpickle	3.1.0
confidence	0.15	construct	2.10.70
contourpy	1.3.1	coolname	2.2.0
coverage	7.6.9	cycler	0.12.1
databricks-sdk	0.38.0	datasets	3.1.0
Deprecated	1.2.15	dill	0.3.8
distinctipy	1.3.4	dm-tree	0.1.8
docker	7.1.0	docstring_parser	0.16
easyocr	1.7.2	etils	1.11.0
exceptiongroup	1.2.2	execnet	2.1.1
filelock	3.16.1	fire-nfi	0.7.3
flake8	7.1.1	Flake8-pyproject	1.2.3
Flask	3.1.0	flatbuffers	24.3.25
fonttools	4.55.2	frozenlist	1.5.0
fsspec	2024.9.0	fuel-nfi	0.10.7
gast	0.6.0	gitdb	4.0.11
GitPython	3.1.43	google-auth	2.36.0
google-pasta	0.2.0	googleapis-common-protos	1.66.0
graphene	3.4.3	graphql-core	3.2.5
graphql-relay	3.2.0	greenlet	3.1.1
grpcio	1.68.1	gunicorn	23.0.0
h5py	3.12.1	huggingface-hub	0.26.5
idna	3.10	imageio	2.36.1

Package	Version	Package	Version
imgaug	0.4.0	immutabledict	4.2.1
importlib_metadata	8.5.0	importlib_resources	6.4.5
inflect	6.2.0	iniconfig	2.0.0
isort	5.13.2	itsdangerous	2.2.0
Jinja2	3.1.4	joblib	1.4.2
jsonschema	4.23.0	jsonschema-specifications	2024.10.1
keras	3.7.0	kiwisolver	1.4.7
lazy_loader	0.4	Levenshtein	0.26.1
libclang	18.1.1	lightning-utilities	0.11.9
llvmlite	0.43.0	lxml	5.3.0
Mako	1.3.8	Markdown	3.7
markdown-it-py	3.0.0	MarkupSafe	3.0.2
matplotlib	3.9.3	mccabe	0.7.0
mdurl	0.1.2	ml-dtypes	0.4.1
mlflow	2.17.2	mlflow-skinny	2.17.2
more-itertools	10.8.0	mpmath	1.3.0
multidict	6.1.0	multiprocess	0.70.16
mypy	1.13.0	mypy-extensions	1.0.0
namex	0.0.8	narwhals	1.22.0
networkx	3.4.2	ninja	1.11.1.2
nmslib	2.1.2	numba	0.60.0
numpy	1.26.4	nvidia-cublas-cu12	12.4.5.8
nvidia-cuda-cupti-cu12	12.4.127	nvidia-cuda-nvrtc-cu12	12.4.127
nvidia-cuda-runtime-cu12	12.4.127	nvidia-cudnn-cu12	9.1.0.70
nvidia-cufft-cu12	11.2.1.3	nvidia-curand-cu12	10.3.5.147
nvidia-cusolver-cu12	11.6.1.9	nvidia-cusparse-cu12	12.3.1.170
nvidia-nccl-cu12	2.21.5	nvidia-nvjitlink-cu12	12.4.127
nvidia-nvtx-cu12	12.4.127	opencv-python	4.10.0.84
opencv-python-headless	4.10.0.84	opentelemetry-api	1.28.2
opentelemetry-sdk	1.28.2	opentelemetry-semantic-conventions	0.49b2
opt_einsum	3.4.0	optree	0.13.1
packaging	24.2	pandas	2.2.3
patsy	1.0.2	pillow	11.0.0
pip	25.3	pluggy	1.5.0
promise	2.3	propcache	0.2.1
protobuf	3.20.3	psutil	6.1.0
psycopg2-binary	2.9.10	py-cpuinfo	9.0.0
pyarrow	17.0.0	pyasn1	0.6.1
pyasn1_modules	0.4.1	pybind11	2.6.1
pyclipper	1.3.0.post6	pycodestyle	2.12.1
pydantic	1.10.19	pydeck	0.9.1
pyflakes	3.2.0	Pygments	2.18.0
pynndescent	0.5.13	pyparsing	3.2.0
pytesseract	0.3.13	pytest	8.3.4
pytest-cov	6.0.0	pytest-xdist	3.6.1
python-bidi	0.6.3	python-dateutil	2.9.0.post0
pytz	2024.2	PyWavelets	1.8.0
PyYAML	6.0.2	RapidFuzz	3.10.1
referencing	0.35.1	regex	2024.11.6
requests	2.32.3	rich	13.9.4
rpds-py	0.22.3	rsa	4.9
ruff	0.8.2	safetensors	0.4.5
scikit-image	0.24.0	scikit-learn	1.5.2
scikit-posthocs	0.11.4	scipy	1.14.1
seaborn	0.13.2	setuptools	44.1.1

Package	Version	Package	Version
shapely	2.0.6	simple-parsing	0.1.6
six	1.17.0	smmap	5.0.1
soupsieve	2.6	SQLAlchemy	2.0.36
sqlparse	0.5.2	statsmodels	0.14.5
streamlit	1.41.1	sympy	1.13.1
tabulate	0.9.0	tenacity	9.0.0
tensorboard	2.18.0	tensorboard-data-server	0.7.2
tensorflow	2.18.0	tensorflow-datasets	4.9.7
tensorflow-io-gcs-filesystem	0.37.1	tensorflow-metadata	1.14.0
tensorflow-similarity	0.17.1	termcolor	2.5.0
threadpoolctl	3.5.0	tifffile	2024.9.20
tokenizers	0.21.0	toml	0.10.2
tomli	2.2.1	torch	2.5.1
torch-dct	0.1.6	torchmetrics	1.2.1
torchvision	0.20.1	tornado	6.4.2
tqdm	4.67.1	transformers	4.47.0
triton	3.1.0	typing_extensions	4.12.2
tzdata	2024.2	ultralytics	8.3.48
ultralytics-thop	2.0.13	umap-learn	0.5.7
urllib3	2.2.3	watchdog	6.0.0
Werkzeug	3.1.3	wheel	0.45.1
wrapt	1.17.0	xxhash	3.5.0
xyzservices	2024.9.0	yarl	1.18.3
zipp	3.21.0		