Timing as a side-channel vulnerability: neural network analysis of generalized PIN prediction

by

Mason Clendenen

B.S., University of Northern Iowa, 2021

B.S., Iowa State University, 2021

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Integrated Studies
College of Technology and Aviation

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2025

Approved by:

Major Professor
Dr. Michael Pritchard

Copyright

Abstract

This study explores a potential side-channel vulnerability in Personal Identification Number (PIN) entry systems (PEDs), by analyzing keystroke timing patterns with machine learning techniques. PIN-based authentication, ubiquitous in securing physical and digital access, struggles to balance usability and security, with prior research focusing on direct threats like visual observation and proposing countermeasures such as extended PINs, or haptic feedback. This research hypothesizes that human timing behavior during 4-digit PIN entry may reveal detectable patterns exploitable by machine learning to predict PINs, posing a novel risk distinct from traditional attack vectors. Using a supervised learning model trained on latency data from a virtual keypad, the study assesses whether these patterns generalize across users, testing a 10,000-class classification problem with out-of-sample k-fold cross-validation. Results show limited pattern detection—a top-1 accuracy of 0.115% and a top-10 of 1.198%, exceeding random guessing but insufficient for practical targeted attacks—with a bias toward top-row digits (e.g., 73, 77), possibly reflecting psychological preferences for numbers like 3 and 7. While generalization remains limited by participant variability, and dataset constraints, the findings suggest a context-specific vulnerability more relevant to large-scale, non-targeted scenarios than individual breaches. This work underscores the need to consider subtle behavioral leaks in PED design, advocating for continued innovation to address emerging machine learning-driven threats.

Table of Contents

List of Figures

List of Tables

Nomenclature

AI Artificial Intelligence

GUI Graphical User Interface

HCI Human-Computer Interaction

HMI Human-Machine Interface

KFold K-Fold Cross-Validation

LSTM Long Short-Term Memory

ML Machine Learning

NN Neural Network

PED PIN Entry Device

PIN Personal Identification Number

QWERTY Keyboard layout named for the first six keys on the top row (Q, W, E, R, T, Y)

SCA Side-Channel Attack

Top-k Metric for ranking prediction accuracy

VoIP Voice over Internet Protocol

Acknowledgments

I would like to thank all my participants who supported this research by volunteering their time and effort to provide their data. From my colleagues responding to my cold-call email for volunteers to my family and friends, as well as my professors who contributed their time and feedback. Your collaboration was transformative in helping me build this idea and experiment, and it also taught me many of the foundational concepts I needed to complete this thesis. It was a truly fun and enlightening project, and your help is greatly appreciated.

Thank you to my wife Kaitlin and my family. Considerable time and dedication went into this, and without your support, this would not have been possible.

Dedication

Dedicated to my wife Kaitlin and my daughter Penelope. Thank you for your support, encouragement, and positivity throughout this process. Finishing this thesis and degree while welcoming our little girl into the world presented its challenges, but we had fun along the way and made many memories I will never forget.

Chapter 1

Introduction

Personal Identification Number (PIN) entry devices (PEDs)—such as door keypads, ATM terminals, and smartphone PINs—are foundational authentication tools securing physical spaces, financial transactions, and digital access worldwide. The PIN-based access control market, projected to reach USD 2.26 billion in 2025 with a 5.5% CAGR through 2030, underscores their enduring prevalence across industries (Intelligence, 2025). Rooted in traditional design, these systems rely on simple 4-digit codes entered via tactile keypads, offering authorized users quick access while aiming to block unauthorized entry. Historically, PEDs have been vulnerable to threats like brute-force attacks (Markert et al., 2020; Nisbet and Kim, 2016), hardware tampering (Drimer et al., 2009), and shoulder surfing (Roth et al., 2004). To counter these, designers have implemented features like longer digit sequences, restricted visual interfaces, and lockout mechanisms after failed attempts. However, these enhancements often complicate the user experience, increasing errors and frustration. This tension between security and usability is a well-referenced idea in PIN entry system design (Chakraborty et al., 2019), often leading to trade-offs that impact both system effectiveness and user satisfaction. An overemphasis on cybersecurity may overlook usability, leading to impractical systems (Kour et al., 2024), or encourage insecure practices like the use of simple, common PINs (Kopecky, 2017).

PED security research has traditionally tackled threats like skimming, tampering, and

shoulder surfing, prompting solutions such as encrypted keypads and tamper-resistant hardware (Hattali et al., 2020). For instance, the Nyang et al. (2018) protocol uses multi-step thumb gestures on touchscreens to obscure PIN entry and reduce shoulder surfing, though its complexity can frustrate users. Extensive research has been conducted on enhancing such measures with haptic and auditory feedback, minimizing visual exposure while guiding users through tactile or sound cues, thus balancing security and usability (Bianchi et al., 2012). However, the rise of machine learning and artificial intelligence (AI) calls into question the effectiveness of these traditional solutions in striking a balance between enhanced security and maintaining usability. Unlike conventional attacks targeting hardware or software flaws, machine learning exploits subtle human behavior patterns, transforming innocuous data into significant security risks (Amrouche et al., 2022; Spreitzer et al., 2018). Recent studies demonstrate machine learning's ability to amplify side-channel attacks (SCAs), analyzing indirect signals like power consumption or acoustic emissions to extract cryptographic keys (Chiari et al., 2024; Kubota et al., 2021). In a striking example, Batina et al. (2019) showed that machine learning can analyze electromagnetic side-channel data to reverse-engineer the architecture of other machine learning models, highlighting its capacity to extract sensitive information from unexpected sources. This suggests that in PEDs, machine learning could similarly exploit keystroke timing patterns—shaped by individual behavior and influenced by features like auditory feedback—to infer PINs, revealing vulnerabilities traditional designs never anticipated.

This thesis investigates one such emerging threat: whether the timing of keystrokes during 4-digit PIN entry can serve as a side-channel attack vector when analyzed with machine learning techniques. This question shifts the focus from conventional PED security concerns, necessitating a reevaluation of these devices through the lens of modern computational power. This chapter lays the groundwork for that investigation. It begins with background on the role of data in machine learning and AI, highlighting human-generated data and its risks. It then examines data leakage and side-channel attacks to frame the research question. The chapter concludes with the study's objectives, scope, and a review of related work, emphasizing its novelty and significance.

1.1 Background

As defined by (Grassi et al., 2017), side-channel attacks are "an attack enabled by leakage of information from a physical cryptosystem." SCAs represent a sophisticated class of security threats that exploit unintended information leakage from a system's inherent design. This leakage can manifest as variations in timing, electromagnetic emissions, or power consumption—data points that reveal operational details not protected by conventional cryptographic safeguards (Agrawal et al., 2003). In systems requiring human interaction, such as authentication mechanisms, this leakage extends beyond hardware to include behavioral traces generated by users. These traces, often subtle and overlooked, provide a rich dataset that machine learning can analyze to uncover patterns invisible to traditional analysis, amplifying the potential for exploitation.

1.1.1 The Double-Edged Sword of Biometric Authentication

As defined by (of Homeland Security, 2025), "A biometric is a measurable biological (anatomical and physiological) and behavioral characteristic that can be used for automated recognition." Biometrics are typically divided into two categories: **hard biometrics**, such as fingerprints, iris patterns, or facial geometry, which are stable and enduring (Kurban and Yildirim, 2024), and **soft biometrics**, such as gait, voice, or typing rhythm, which are more dynamic and context-dependent (Dantcheva et al., 2011). The integration of machine learning into biometric systems has revolutionized their application, enabling the extraction of intricate patterns from both hard and soft traits; a well-known example is machine learning and face recognition (Sharma et al., 2020). However, does this same capability introduce a paradox? The very features that make biometrics valuable for identification may also render them vulnerable to theft or misuse.

Biometric authentication promises a seamless alternative to traditional passwords or PINs, leveraging traits inherent to the user to verify identity. Yet, this convenience comes with inherent risks, creating a double-edged sword (Balamurugan, 2024). If a biometric signature is compromised, it differs fundamentally from a stolen password: it cannot be

easily replaced (Kurban and Yildirim, 2024). For hard biometrics, this permanence is particularly troubling—once a fingerprint or iris scan is replicated, the breach is irrevocable, leaving the individual's identity perpetually exposed (Gomez-Barrero and Galbally, 2020). Soft biometrics, while less permanent, introduce their own challenges. The boundaries of soft biometrics are inherently fluid, subject to flux and overlap across individuals, as traits like reaction time in human motor control often follow statistical distributions that span broad populations rather than uniquely identifying single users (Whelan, 2008). As mentioned by Balamurugan (2024), these challenges can be mitigated by looking at multi-modal biometric factors, a hybrid of soft and hard biometrics, to further increase the robustness of these AI- and machine learning-based applications in security (Venkata et al., 2022). Without these advanced solutions, the data used to secure a system can become a target—stolen silently without the user's knowledge—even from indirect inference.

1.1.2 Soft Biometrics and Machine Learning: Promise and Peril

Soft biometrics, when paired with machine learning, exemplify both the potential and the pitfalls of this technology. Consider gait recognition: machine learning algorithms can analyze video footage to identify individuals based on the rhythm and style of their walk, even in crowded or distant settings (Song et al., 2019). This capability has applications in surveillance and security but relies on consistent behavioral patterns that may not always hold. Likewise, handwriting analysis has been found to reveal age-related traits, with deep learning models detecting age from stroke patterns, and pressure, though such features often overlap across populations (AL-Qawasmeh et al., 2023). These examples highlight machine learning's power to harness soft biometrics, yet their variability underscores potential vulnerabilities. Similarly, keystroke dynamics—the study of timing and pressure patterns in typing—has emerged as a promising soft biometric for authentication. Research has demonstrated its ability to distinguish users based on the unique cadence of their keystrokes, offering a passive, continuous verification method (Piugie et al., 2022).

However, the reliability of keystroke dynamics as an authentication method is challenged

by both external factors and sophisticated attacks. Emotional states, physical fatigue, or variations in keyboard type introduce noise, increasing uncertainty and compromising identification precision (Lee et al., 2015). To address this, designers might be tempted to adopt a generalized model, widening the acceptance window to accommodate such influences. Yet, this approach introduces significant vulnerabilities: recent studies demonstrate that keystroke dynamics can be spoofed, with attackers extracting synthetic typing patterns from screen-recorded video by analyzing feedback symbols (e.g., dots, or asterisks) using high-frame-rate footage, thus bypassing authentication remotely (Fitriani et al., 2022). Generalization exacerbates this risk, as a broader tolerance allows synthetic patterns to pass without needing to precisely replicate a specific user's behavior. Moreover, evidence suggests that soft biometric patterns often exhibit sufficient similarity across individuals to blur profile boundaries (Dantcheva et al., 2016). This is underscored by the handwriting analysis just discussed, where age-related traits were shown to overlap across populations (AL-Qawasmeh et al., 2023), complicating individual identification. This reveals a critical paradox: while a generalized design may enhance reliability by mitigating noise, it simultaneously increases susceptibility to attacks by enabling exploitation of common behavioral traits rather than requiring targeted mimicry. In the context of PIN entry devices (PEDs), where standardized keypads and repetitive 4-digit inputs inherently limit behavioral variability, this overlap may intensify, amplifying the vulnerabilities of such generalized models.

1.1.3 Keystroke Dynamics in PIN Entry Devices

Fitts' Law, a foundational theory in human-computer interaction (HCI), provides a predictive model for the time required to move to a target based on its distance and size (Fitts, 1954). Widely adopted in user interface design—spanning graphical user interfaces (GUIs) and physical human-machine interfaces (HMIs) like keyboards—Fitts' Law posits that movement time increases with distance and decreases with target size. In the context of PIN entry devices (PEDs), such as ATMs, or point-of-sale terminals, each digit press represents a discrete movement to a key on a standardized numeric keypad. With fixed key sizes and

distances, the theory, extended to two-dimensional tasks (MacKenzie and Buxton, 1992), suggests that the timing between consecutive keypresses should be predictable for a given PIN sequence. For example, pressing "1" followed by "2" (adjacent horizontally) should take less time than "1" followed by "7" (a vertical leap). This predictability, rooted in human motor behavior, could lead to consistent timing profiles across users, as the physical layout of PEDs imposes similar constraints on all interactions.

This consistency, however, becomes a double-edged sword in the constrained scenario of PIN entry, where users repeatedly input short, 4-digit codes using only ten digits. Unlike traditional keystroke dynamics in free-text typing, where variability in phrasing and rhythm offers some protection, the limited input range and repetitive nature of PEDs restrict keystroke patterns. Studies suggest that under such controlled conditions, timing profiles may converge, creating overlaps that machine learning models can exploit (Serwadda et al., 2016). While initially explored in touchscreen contexts, such as swipe-based authentication, these findings hint at broader implications for discrete input systems like PEDs. An attacker with access to timing data—potentially harvested via side-channel leakage like power consumption, electromagnetic emissions, or acoustic cues, as demonstrated in early timing attack research (Kocher, 1996a)—could infer PINs by analyzing these patterns. Rather than targeting a specific user, the attacker might leverage shared behavioral traits across populations, turning a standardized security mechanism into an attack vector.

The potential vulnerability deepens when considering the multi-dimensional nature of keypad interactions. While Fitts' Law is traditionally one-dimensional, focusing solely on distance, PEDs involve two-dimensional movements (horizontal and vertical). This extension may introduce biases in timing data—for instance, horizontal transitions might be faster than vertical ones due to ergonomic factors like finger reach or keypad design (Kantowitz and Elvers, 1988). Such biases could produce timing patterns that are not unique to individuals but shared across large groups, echoing the "fuzzy" boundaries observed in soft biometrics, where traits exhibit sufficient similarity to blur individual distinctions (Dantcheva et al., 2016). This raises a pivotal question: do these timing patterns represent individual biometrics, exploitable only with user-specific training, or do they expose broader vulnerabilities by

reflecting generalizable human tendencies? The answer hinges on whether machine learning models can detect and exploit these broad patterns, a tension that underpins the present study.

Timing, therefore, emerges as a critical factor in PED security, with predictable intervals between keypresses—governed by Fitts' Law and amplified by standardized layouts—offering a potential side-channel attack vector that bypasses traditional cryptographic protections. While prior work highlights this risk in touchscreen settings, its applicability to physical PEDs remains unexplored. The following section outlines a research approach to investigate whether keystroke timing in PIN entry can be systematically exploited using machine learning techniques, setting the stage for a deeper examination of this potential vulnerability and its implications for system design.

1.2 Research Overview

The convergence of side-channel data, soft biometrics, and machine learning in PIN entry systems raises a compelling question: can timing data from keystroke dynamics in PEDs be used to predict PINs, exposing a novel vulnerability in human-computer interaction? This research investigates the extent to which population-shared timing patterns in PIN entry can be exploited, assessing the feasibility of such an attack and its implications for biometric security. By exploring this intersection, the study aims to reveal a potential weakness that could erode trust in these ubiquitous devices, paving the way for more robust defenses. Specifically, it examines whether keystroke latency during 4-digit PIN entry presents a side-channel vulnerability exploitable by machine learning. The research question is: **Does time latency data between keystrokes for a 4-digit PIN present a side-channel vulnerability with machine learning technology?** This is addressed through four main objectives:

1. **Collect Timing Data**: Record keystroke latency via a virtual model simulating PIN entry.

2. **Build a Machine Learning Model**: Create and train a machine learning model to predict PINs from latency patterns.

3. **Evaluate Risk**: Analyze the model's success rate and generalizability to assess if timing data poses a widespread threat.

4. **Look for Patterns**: Compare and contrast participant latency data to the correct predictions to understand and theorize model behavior.

The scope focuses on a virtual keyboard setup, chosen for its controllability and resource feasibility. This differs from physical PEDs like ATMs or wall keypads but allows a precise test of the hypothesis. The study doesn't delve into timing pattern origins (e.g., biological factors) or other attack vectors like sound, which initially inspired the idea. The research probes an emerging risk at the intersection of human behavior, machine learning, and PEDs, with potential impacts on security design and user awareness, especially as machine learning shifts cybersecurity toward proactive threat prediction (Infosec Institute, 2024). The next chapter provides more detail on the virtual setup, model design, training, and validation methods. The next section reviews related work to contextualize this approach.

1.3 Related Work

Research on side-channel attacks targeting PIN entry devices (PEDs) is less extensive than studies on new designs or usability surveys, with efforts split between free-text keystroke dynamics and static PIN entries. Within this domain, several studies align closely with this thesis's focus on PIN entry vulnerabilities, offering insights into timing pattern analysis and machine learning's role in exploiting behavioral data—whether individualized or population-shared—during PIN input.

Early investigations into timing-based side-channel attacks on PEDs relied on statistical methods rather than machine learning, yet remain relevant due to their focus on traditional keypads. (Yan et al., 2010) explored timing attacks on PIN input devices like those in VoIP

systems, or ATMs. Their 2010 study analyzed inter-keystroke timing from 4-digit PINs, hypothesizing that predictable latency patterns, driven by human motor behavior and keypad layout, could leak information. Using statistical models on simulated PIN entry data, they correlated latency with digit transitions (e.g., shorter for "1" to "2" vs. longer for "1" to "7"). Their results showed partial success in narrowing PIN guesses, though the attack's effectiveness was constrained by noise (e.g., user variability). Unlike this thesis, their approach didn't leverage modern machine learning techniques for pattern generalization across users, nor did it test a virtual setup, limiting its scope to physical devices and user-specific data. Similarly, (Zhang and Fischer-Hübner, 2011) examined timing attacks in Voice over IP (VoIP) networks, where PINs access automated services. Their 2011 study used inter-keystroke delays from encrypted VoIP packets, exploiting real-time packet generation per keypress. Employing a Markov model on a limited set of digit transitions, they reduced the brute-force search space by 80% (from 10,000 to 2,000 PINs), prioritizing guess reduction over top-1 accuracy. This network-based attack, viable despite encryption due to packet interception, contrasts with this research's machine learning-driven, physical PED focus. These statistical approaches established timing as a viable leak, paving the way for machine learning enhancements.

Recent studies leverage machine learning to exploit smartphone PIN entry, closely relating to this thesis through behavioral data analysis. (Mehrnezhad et al., 2018) developed PINlogger.js, a machine learning framework predicting PINs from smartphone sensor data—orientation, acceleration, and inertial measurements—without privileged access. Their multi-user model achieved a 70.75% first-try success rate for 4-digit PINs, rising to over 90% within three attempts, using a reduced set of 50 PINs (not 10,000). While showcasing machine learning's power on subtle patterns, it targets smartphones, not traditional PEDs, and relies on rich sensor data unavailable in older keypads, unlike this study's timing-only focus. (Beltramelli et al., 2022) also used smartphone motion sensors—accelerometer, and gyroscope readings sampled during keypresses—to train a neural network (NN) for single-digit classification, followed by statistical modeling for full 4-digit PINs. Assuming PIN repetition, their hybrid approach contrasts with this thesis's 10,000-class full-PIN predic-

tion. Their model, tested across users but trained user-specifically, didn't fully generalize to population-shared patterns, diverging from this study's user-agnostic goal and traditional PED context.

Closer to this research, studies apply machine learning to traditional PEDs with timing features. Panda et al. (Panda et al., 2020) predicted PINs from auditory feedback in ATMs, recording keypress sounds and extracting features like press-release timing, inter-keystroke intervals, and acoustic signatures. Their 2020 study recorded keypress sounds from numeric keypads, extracting features such as time between press and release, inter-keystroke intervals, and acoustic signatures. Training on multi-user datasets, their models achieved moderate success in PIN prediction, confirming exploitable patterns in auditory cues. However, their methodology diverges from this thesis: it incorporated multiple timing metrics beyond inter-keystroke latency and prioritized multi-user training over a user-agnostic approach. Notably, their multi-user model underperformed in testing compared to user-specific models, suggesting limited generalizability—a key contrast to this study's aim to identify population-shared vulnerabilities. They also explored a dual role for behavioral data, proposing user-specific timing models as a security enhancement, which underscores its potential as both a risk and a safeguard. (Harrison et al., 2023), while smartphone-focused, merits mention for its timing relevance and related nature to the previous study, using deep learning to classify keystrokes via microphone audio with 95% accuracy. Targeting free-text typing, not PINs, it required proximity and high-quality recordings, differing from this thesis's static, traditional PED scope.

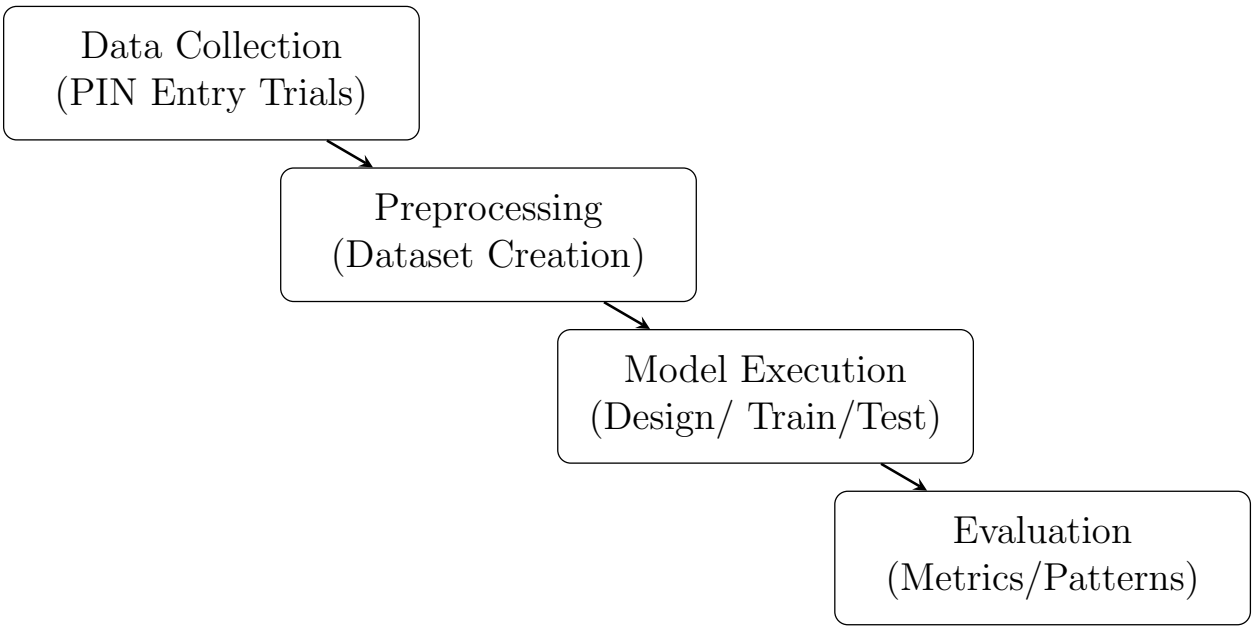
Existing research addresses facets of this thesis's scenario—exploiting timing data from 4-digit PINs in traditional PEDs with user-agnostic machine learning—but not its entirety. Older studies skipped machine learning, relying on statistics; machine learning-based works paired timing with other features (sensors, audio) or limited scope (50 PINs, single digits). Spanning free-text and static datasets, they show timing's pattern sensitivity but diverge in design from this study's broad, timing-only approach across 10,000 classes. This thesis investigates whether keystroke latency alone offers a novel perspective on an underexplored vulnerability. The methodology chapter details this approach.

Chapter 2

Methodology

This chapter details the methodology employed to study time latency patterns in PIN entry devices using machine learning. It begins with an overview of how the real-world PIN entry device scenario was abstracted into a virtual model to simplify complexities. The flow of sections depicted in Figure 2.1, delve into the data collection process, preprocessing of data for the dataset, overall model design, model training strategy, and model evaluation criteria—all of which support the overarching research objectives.

Figure 2.1: Overview of the research pipeline, from PIN entry trials to performance evaluation.



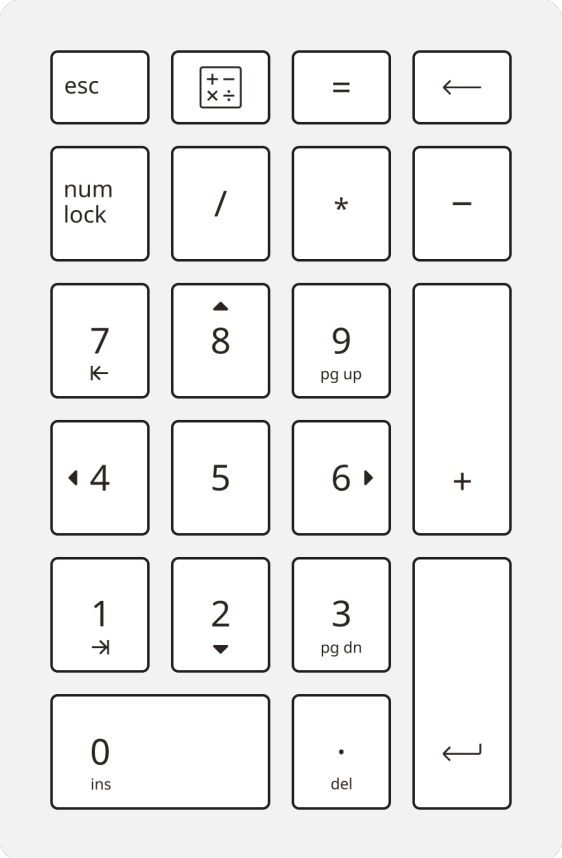
2.1 Experiment Overview

One possible approach for this research would be to develop and deploy hardware and software for a threat device in a real-world scenario using a common PIN entry device, as described in Chapter 1. This device could then collect timing data and host a trained machine learning model for out-of-sample performance testing. However, this approach introduces unnecessary resource demands and uncontrolled variables—such as signal conditioning complexities, processing latencies, and environmental factors—that are not essential to the research question, despite its high fidelity to real-world conditions. With that in mind, as well as development time and financial considerations, it was decided to abstract and model this PIN entry scenario virtually in a desktop environment.

The virtual environment model was derived by functionally decomposing the PIN entry process commonly found with PEDs. This involves initializing the device for PIN code entry and then entering a multi-digit PIN on a 10-digit number pad. Initialization might involve a specific start button, swiping a badge, or radio frequency proximity sensors to identify a user. This behavior is a critical feature for this study; it provides a consistent initial reference point for the first timing value. In real-world scenarios, initialization movements are less constrained and distinct from digit entry, so the virtual model was carefully designed for this. For initialization, the space bar was chosen for its larger size, which introduces slight variability akin to real-world PEDs, while providing a consistent reference point with a distinct distance from the numerical keys. A 10-digit number pad then mirrors the conventional layout on most PEDs today for digit entry, constrained to a specific design as shown in Figure 2.2.

The layout constraint minimizes biases in movement direction. While Fitts' law outlines a simple mathematical relationship in 1-dimensional space, it may overlook 2-dimensional directional influences as noted in Subsection 1.1.3. If present, differences in number pad layout could lead to inconsistent timing patterns due to conflicting movement directions, potentially masking directional biases. For example, an upward movement for one participant might be downward for another, skewing timing data for the same PIN combination.

Figure 2.2: Illustration of the standard 10-digit layout used as a reference for the virtual keypad in this study. Adapted from Framework (2024).



The human behavior aspect for the PIN entry interaction was also constrained for the virtual model. PEDs are typically vertical with a 10-digit number pad, and their small keys, positioning, unfamiliarity, and sequential tasks can lead users to push the keys with a single index finger (Wang and Shih, 2009). Unlike a full desktop QWERTY keyboard, where many users have training to type efficiently with both hands and all fingers, participants in the virtual model might revert to this multi-finger style on the number pad. Therefore, to reflect typical PED usage, the model constrains participants to use only their dominant hand's index finger for all keystrokes in the virtual model.

Finally, the virtual environment enables efficient data collection, model training, and evaluation on desktop computers, which offer greater computational power than edge devices—crucial for processing the large timing datasets and training machine learning models in this study. The next sections will cover data collection, the processing of raw timing data

into a machine learning dataset, the ML model design, and the metrics used to evaluate the model's ability to predict PIN codes based on latency patterns.

2.2 Data Collection

The data collection process adhered to the virtual model requirements outlined in Section 2.1. Participants were instructed to use a full keyboard with a dedicated 10-digit number pad and enter keys using their dominant hand's index finger. To address the research question—timing patterns between keystrokes—latency data was collected for 2-digit combinations rather than full 4-digit combinations. This approach balances data scope with feasibility: with 100 possible 2-digit combinations (compared to 10,000 for 4-digit PINs), it avoids the impractical resource demands and participant fatigue of collecting data for all possible PINs while capturing essential timing data. Additionally, it maintains a generalized scope, as selecting specific 4-digit combinations would require subjective assumptions about potential patterns, potentially limiting broad applicability. Furthermore, to ensure data reliability—critical for analyzing sequential keystroke timing—multiple redundant trials were warranted for each combination. This is feasible with the 2-digit approach, whereas a 4-digit approach would have made redundant trials impractical due to the increased number of trials. Thus, the 2-digit trial approach was adopted, and a custom data collection program was developed to streamline the process.

A custom Python script was created to manage the data collection process. It begins by building a trial queue—a list of 2-digit trials and initialization trials (space bar to a numerical digit). Each unique trial is repeated five times to ensure data reliability through redundancy. The list is then randomized to prevent order effects, such as fatigue or learning biases from typing the same trial consecutively.

When a participant runs the program, they are prompted to enter their name for identification. After providing their name, they receive an instruction printout (as detailed in Section 2.1), which outlines the data entry requirements. Once the participant presses 'Enter' to proceed, the program initiates the data entry process, iterating through the randomized

trial queue described above. For each trial, the program calls the `keyLatencyTimeTrial` function to measure the latency between the two keystrokes. The first critical part of the timing behavior is using the built-in keyboard module's `keyboard.wait()` function to halt the program until the specified start key is pressed, at which point a high-resolution timestamp is captured using `time.perf_counter()`. After the end key is pressed, a second timestamp is recorded, and the latency is calculated as the difference between the two. These results—start key, end key, and latency—are stored in a list of dictionaries. Upon completion, the data is saved to a CSV file named with the participant's identifier, where each row includes the columns `start_key`, `end_key`, and `latency`, enabling efficient analysis of individual participant data. A source code excerpt of the function is shown in Figure 2.3.

Figure 2.3: Python function for measuring keystroke latency

```
    def keyLatencyTimeTrial(start_key, end_key):
        global trialCount, totalTrialsNeeded
        print(f"Trial {trialCount}: Press {start_key} then {end_key}.")

        # Wait for the start key to be pressed
        keyboard.wait(start_key)
        start_time = time.perf_counter()

        # Wait for the stop key to be pressed
        keyboard.wait(end_key)
        stop_time = time.perf_counter()

        elapsed_time = stop_time - start_time
        return elapsed_time
```

The next section will describe how the raw data was used in building the full data set for the ML model. This includes the visualizations and analysis used to digest the data, as well as the processing into final form.

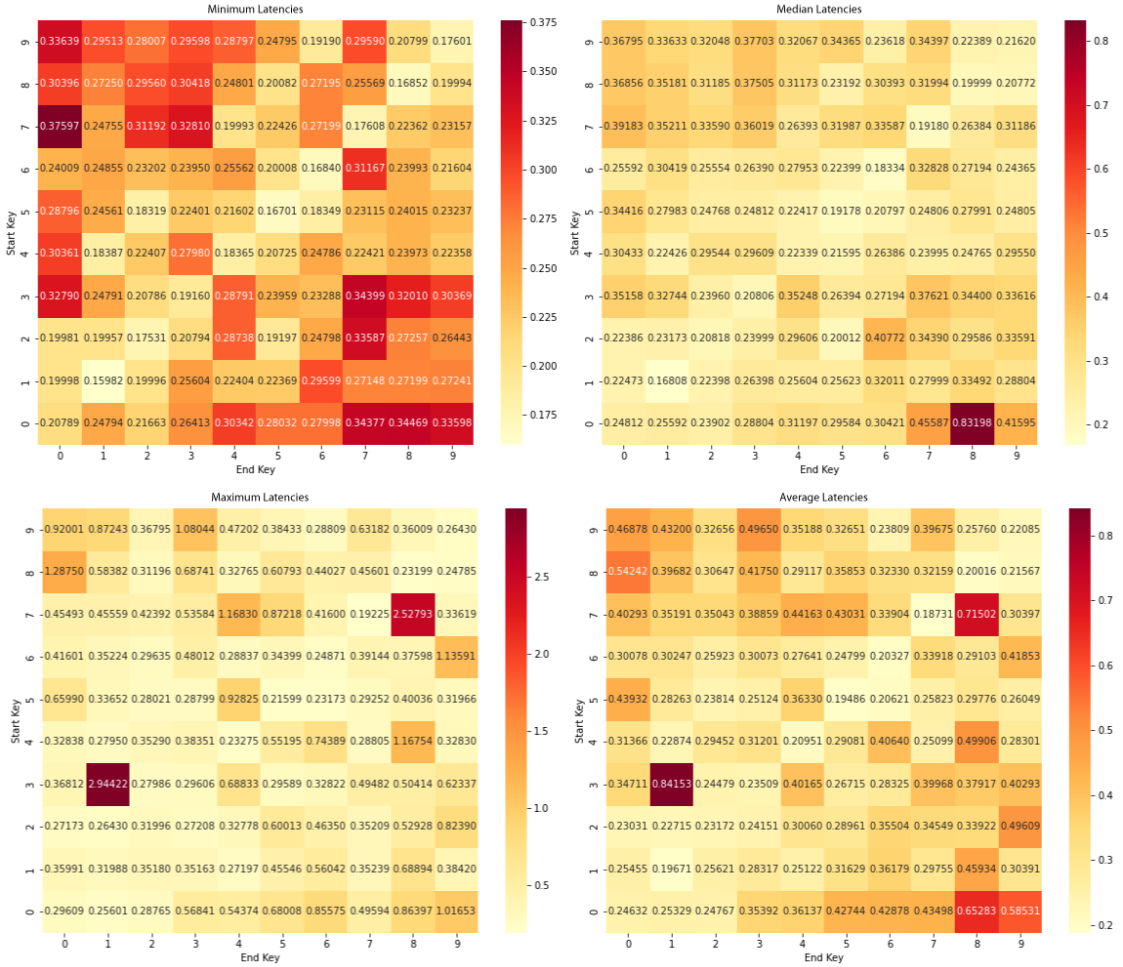
2.3 Preprocessing and Dataset Creation

The raw data for this study were collected from 24 participants with diverse ages and occupations, reflecting a range of typing abilities and experiences. These participants completed the data trials described in Section 2.1, generating latency measurements visualized using

heat maps to explore keystroke timing patterns across 2-digit combinations. This section details the preprocessing steps to transform this raw data into a finalized dataset for the machine learning model, including analysis and visualization techniques.

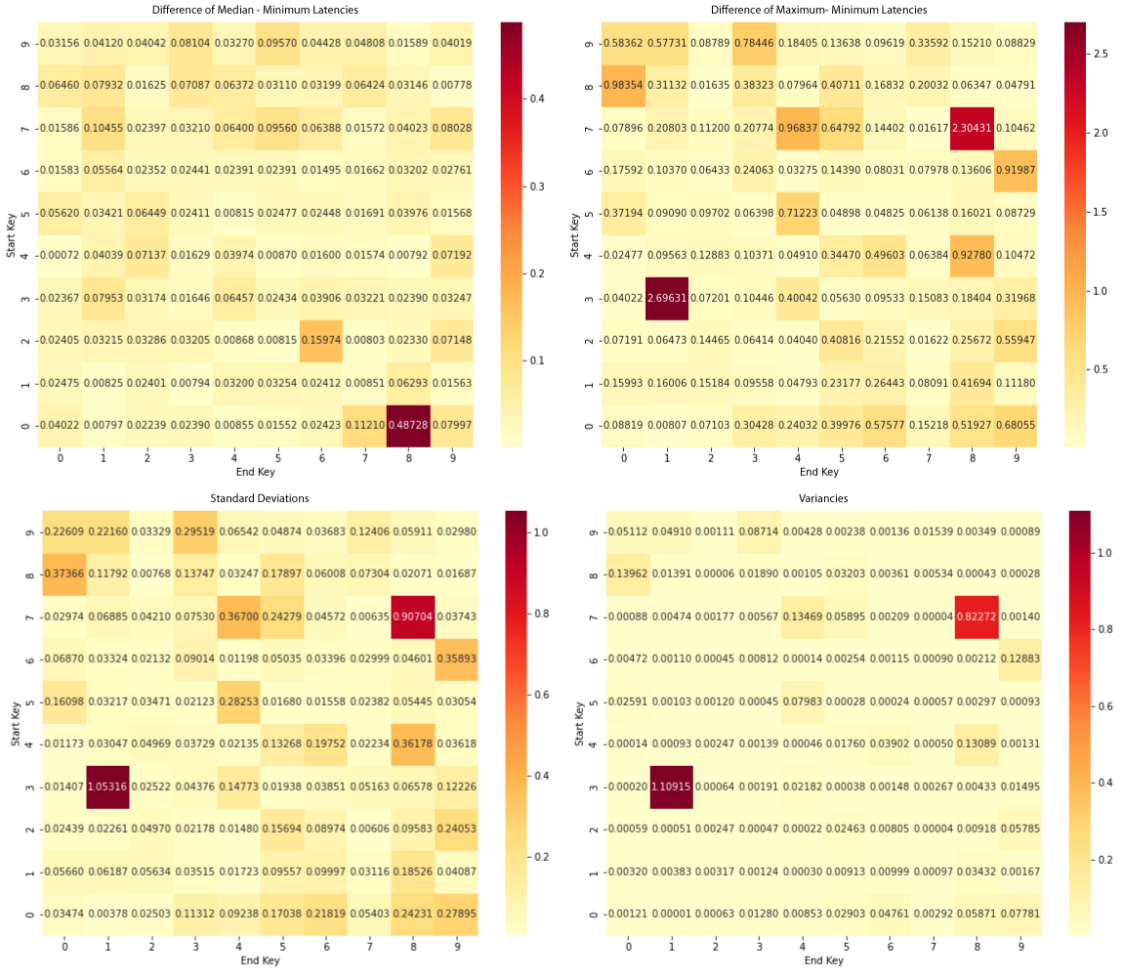
Heat maps visualized latency data in a grid format, with the y-axis representing the start key, the x-axis the end key, and each cell the latency time for that 2-digit combination. Multiple heat maps were generated to analyze the participants' raw latency times, including minimum, median, maximum, and mean values. The plots allowed visual comparisons, providing key insights into the data distribution through the data trials. Figure 2.4 is from participant 9's data collection, as will be discussed, participant 9 was one of the cases that demonstrated challenges in the collected data.

Figure 2.4: Heatmaps with latency time data for participant 9 including minimum, median, maximum, and mean latencies



Heat map analysis revealed that same-digit keypresses (e.g., 0-0, 1-1) exhibited shorter latency times compared to different-digit combinations. Fitts' Law predicts significantly reduced latency for such presses due to zero finger movement, but empirical studies suggest a more modest speed gain (MacKenzie and Soukoreff, 2002). This subtlety is reflected in Participant 9's minimum latency heat map showing a lighter diagonal indicating shorter times, and is more evident in participant 23's minimum latency heatmap (Appendix A, Figure A.3), where the diagonal is clearer. This pattern, reflecting participant 23's consistent timings, contributed to results detailed later(Section 3).

Figure 2.5: Heatmaps showing latency time analysis for participant 9, with each plots data representation titled.



Beyond same-digit patterns, additional heatmaps visualized calculations across redundant trials—such as the difference between median and minimum, maximum and minimum,

standard deviation, and variance—to reveal outliers, errors, and inconsistencies guiding dataset creation. An example of these heatmaps is shown in Figure 2.5 from participant 9. The key takeaway was the presence of large outliers for certain keys, likely due to participants entering incorrect numbers or unfamiliarity with keys. As displayed by participant 9's plot showing the difference between median and minimum times, unusual combinations had large disparities comparable to the full range difference. Ultimately, the mean, median, and maximum latency times were too affected by outliers and errors to be reliable. In contrast, the minimum latency was theorized to reflect confident, muscle-memory-based inputs, better capturing consistent keystroke timing. Consequently, the minimum latency time for each unique 2-digit trial was selected for the final dataset, enabling preprocessing to proceed.

To support supervised learning detailed in Section 2.4.2, the final dataset required two arrays: X, containing concatenated latency times for each 4-digit PIN (including the initialization combination), and Y, the labeled 4-digit PINs in integer form. However, heat map analysis revealed varying typing speed baselines across participants' minimum times, a bias that could impair the model's ability to learn general patterns. Thus, each participant's minimum times were scaled using the normalization algorithm in Equation 2.1 before concatenation into the final dataset.

$$t_{\text{scaled}} = \frac{t - t_{\text{min}}}{t_{\text{max}} - t_{\text{min}}} \tag{2.1}$$

Normalization produced latency times scaled relative to each participant's typing speed, with 1 as their slowest 2-digit combination latency and 0 as their fastest. With preprocessing complete, the final dataset arrays were constructed. The X array is a two-dimensional structure with 10,000 rows (one per possible 4-digit PIN combination) and 4 columns (each representing the sequential latency from initialization to the final input). The Y array is 1-dimensional, with each row containing the corresponding 4-digit PIN in integer form. Both arrays sequentially loop through the participants' data, with the first 10,000 rows from participant 1's data, the next 10,000 from participant 2's, and so forth, yielding 240,000 total samples—24 per 4-digit PIN combination. This dataset was then ready for conversion

into tensors and used in the machine learning model for training, validation, and testing, as detailed in subsequent sections.

2.4 Model Design

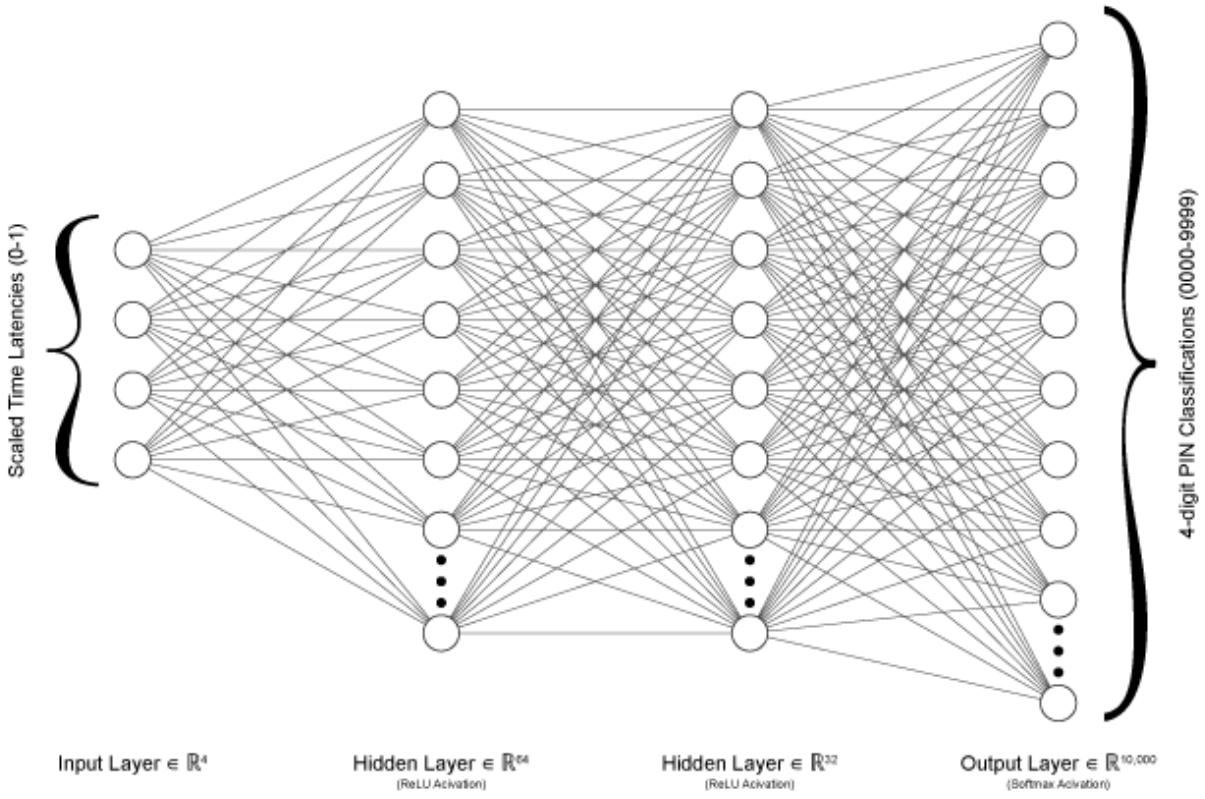
The design of the machine learning model for this study prioritizes simplicity and generality to investigate the presence of keystroke timing patterns in the preprocessed dataset. Rather than employing specialized architectures—such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), long short-term memory networks (LSTMs), or transformers—which are tailored to exploit specific temporal or structural features, a shallow, 4-layer neural network was selected. This choice reflects the exploratory nature of the research: instead of imposing preconceived notions about the nature of timing patterns that more complex models might target, a basic feedforward architecture allows for an unbiased assessment of whether general patterns exist within the latency data. Feedforward neural networks excel at modeling non-linear relationships, making them well-suited to detect latent timing signatures, such as key-pair variations, without assuming their form (Haykin, 2009). Simpler models, such as linear regression or support vector machines, were not considered, as they rely on linear assumptions or predefined kernels that could limit pattern discovery in this novel side-channel context. By limiting architectural complexity, the model serves as a foundational testbed to detect potential timing signatures without assuming their form or structure. The following subsections detail the specific architecture, including layer configuration, activation functions, and node selection strategy, as well as the learning algorithm employed to train the model.

2.4.1 Neural Network Architecture

The neural network architecture designed for this study consists of a shallow, fully connected feedforward network tailored to detect keystroke timing patterns from the preprocessed latency data. Figure 2.6 illustrates the 4-64-32-10,000 architecture, providing a visual overview

of the layer structure and data flow.

Figure 2.6: Illustration of fully connected neural network architecture in this study.



The input layer comprises 4 nodes, corresponding to the normalized latency times for each 4-digit PIN sequence (including the initialization combination), as detailed in Section 2.3. These values, scaled between 0 and 1, feed into two hidden layers. The first hidden layer contains 64 nodes, selected to capture a variety of low-level timing relationships—such as key-pair latency variations—while aligning with power-of-2 sizing for computational efficiency. The second hidden layer reduces to 32 nodes, compressing these features into a more compact representation to balance model capacity and mitigate computational complexity, given the large output layer. Both hidden layers employ ReLU (Rectified Linear Unit) activation functions, chosen for their widespread use in modern neural networks and their advantages in optimization: ReLU promotes faster convergence by maintaining gradient flow and reduces saturation compared to traditional functions like sigmoid or tanh. The output layer consists of 10,000 nodes, each representing a possible 4-digit PIN, with softmax activation applied

to produce a probability distribution over the classes. This enables the model to predict the most likely PIN based on timing patterns. This architecture provides a baseline for pattern exploration, with its training detailed next.

2.4.2 Learning Algorithm and Training Parameters

The learning algorithm and training parameters for the neural network were selected to ensure effective training while maintaining a general approach consistent with the study's exploratory objectives. Given the model's output—a softmax probability distribution over 10,000 possible 4-digit PINs—the cross-entropy loss function was chosen. This loss function quantifies the divergence between predicted probabilities and true labels, a standard choice for multi-class classification that aligns with softmax's probabilistic nature. For optimization, the Adam (Adaptive Moment Estimation) algorithm was employed due to its widespread adoption and robust performance. Adam adapts learning rates for each parameter by leveraging moving averages of both gradients and their second moments, promoting faster and more stable convergence compared to standard gradient descent. This adaptability reduces the need for extensive hyperparameter tuning—aligning with the study's focus on pattern investigation rather than model optimization—while enhancing efficiency and resilience to noisy gradients. Training used a batch size of 64, selected to balance memory usage and gradient smoothness given the dataset's size, and ran for up to 100 epochs, with early stopping after 15 epochs of stagnant validation loss to curb overfitting. A learning rate of 0.001 was set as a standard starting point, striking a balance between training speed and convergence reliability. These hyperparameters—batch size, epoch count, early stopping, and learning rate—were intentionally kept conventional, drawing from commonly recommended values to ensure a practical trade-off between training speed, memory constraints, and generalization, without prioritizing exhaustive tuning for peak performance. This configuration supports the study's objective: to assess whether a simple neural network with general settings can uncover timing patterns in PIN latency data, as explored further in the subsequent section on model validation and outputs that will support that analysis.

2.5 Model Validation, Metrics, and Outputs

The validation and evaluation of the neural network were designed to rigorously assess its ability to predict 4-digit PINs from keystroke timing patterns. Given the dataset's constrained size of 240,000 samples—24 per class across 10,000 unique PINs, a stratified k-fold cross-validation approach with $k = 5$ was selected to provide a robust and reliable performance estimate despite the limited per-class sample size. This method partitions the dataset into five folds, with each fold generally using 80% (192,000 samples) for training and 20% (48,000 samples) for validation, approximating the widely used 80/20 split typical of larger datasets. Stratification maintains that each fold preserves the balanced class distribution—critical given only 24 samples per PIN—by proportionally sampling from each class, unlike random splitting which risks uneven representation. However, the dataset and training subset were shuffled at random for generalization. The choice of $k = 5$ strikes a balance between computational feasibility and evaluation stability: fewer folds (e.g., $k = 3$) might overfit to smaller validation sets, while more (e.g., $k = 10$) increase training time without proportional gains for this dataset size. This approach supports the research objective by thoroughly testing the model's generalization to unseen timing patterns across multiple data splits.

Performance evaluation encompassed a suite of metrics and outputs to quantify model efficacy and explore learned patterns. The average best validation loss across the five folds was computed, reflecting the model's optimization progress and informing early stopping (as in Section 2.4), with lower loss indicating better fit to validation data. Additionally, average best top-1, top-5, and top-10 prediction accuracies were calculated: top-1 accuracy measures exact PIN prediction; top-5 accuracy evaluates whether the correct PIN ranks among the five highest-probability predictions; and top-10 accuracy assesses its presence within the top ten—practical metrics for a 10,000-class problem, where accuracies beyond top-1 suggest partial pattern detection by ranking the correct PIN among top candidates. Cross-fold metric outputs include box plots of best validation loss and top-1 accuracy across folds to visualize performance variability, as well as per-fold learning curves for loss and top-1 accuracy to

monitor training dynamics. To enable deeper analysis tied to the research question of timing patterns, correctly predicted PINs in validation and training were collected, along with indices for each participant facilitating traceability between the correct predictions and the participant they corresponded to. This provided outputs such as counts of correct 4-digit PIN predictions (per fold and total) for each participant, and the most common 2-digit number combinations the model predicted. Report of the metric results and outputs is detailed further in Chapter 3.

Chapter 3

Results

This chapter presents the findings of the model objectively, without interpretation. Supporting figures and tables are included in Appendix B as referenced, with further analysis in Chapter 4. The results begin with performance metrics, followed by supplementary outputs, including derived results from post-output analysis.

3.1 Model Performance Metrics

The model's performance was evaluated across five stratified folds to quantify its efficacy in predicting 4-digit PINs from keystroke timing patterns within a 10,000-class problem. Average best validation loss and prediction accuracies, reported in Table 3.1, reflect the model's typical performance given the dataset's sparsity—only 24 samples per class. The average best validation loss was 8.469, with top-1 accuracy at 0.115%, top-5 at 0.603%, and top-10 at 1.198%, demonstrating improved ranking of correct PINs as the top-k scope widens.

To assess performance consistency, Figure 3.1 presents box plots of best validation loss and top-1 accuracy across folds, highlighting variability and outliers across folds. The per-fold learning curves, which detail training dynamics across epochs, all resulted in similar plots. Thereof, only Fold 1 (best accuracy fold) and Fold 4 (best loss fold) learning curve

Table 3.1: Average best performance metrics across five stratified folds, evaluating the model's typical ability to predict 4-digit PINs from keystroke timing patterns.

Metric	Average Best Value
Validation Loss	8.469
Top-1 Accuracy (%)	0.115
Top-5 Accuracy (%)	0.603
Top-10 Accuracy (%)	1.198

plots are provided in Appendix B (Figures B.1, B.2).

Figure 3.1: Box plots of best validation loss (left plot) and top-1 accuracy (right plot) across the 5 folds, showing performance variability.

Given the study's aim to detect any learnable timing patterns, Table 3.2 reports the best metrics achieved across folds, with Fold 4 yielding the lowest validation loss at 8.429, Fold 1 peaking at 0.127% top-1 and 0.623% top-5 accuracy, and Fold 2 reaching 1.254% top-10 accuracy—revealing the model's maximum potential per metric.

Notable findings emerged when looking at each fold's best loss and accuracy data in Appendix B (Table B.1). First, Fold 4's best validation loss (8.429) coincided with the worst top-1, second-worst top-5, and median top-10 accuracy, as well as appearing as a loss outlier in Figure 3.1. Second, Fold 1's best top-1 (0.127%) and top-5 (0.623%) accuracy paired with

Table 3.2: Best performance metrics, highlighting the model's maximum potential.

Metric	Best Value	Fold
Validation Loss	8.429	4
Top-1 Accuracy (%)	0.127	1
Top-5 Accuracy (%)	0.623	1
Top-10 Accuracy (%)	1.254	2

median top-10 accuracy and the second-worst loss. Finally, Fold 2's best top-10 accuracy (1.254%) had the second-worst top-1 and top-5 accuracy, with a median best loss.

3.2 Supplementary Outputs of Correct Predictions

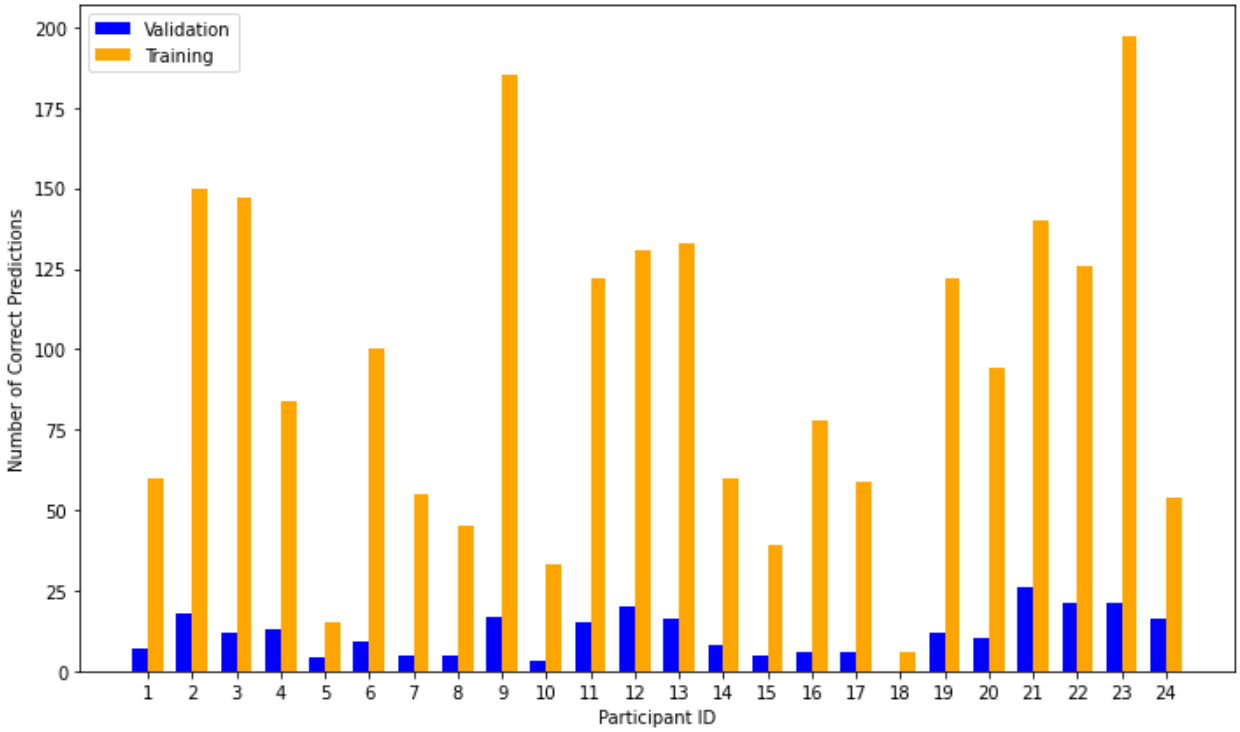
As described in Section 2.5, the experiment provided several key supplementary outputs to aid the investigation of timing patterns. This included correct 4-digit PIN predictions in the validation and training phases traced to folds and participants. From those, per fold counts and participant rankings could be achieved, with results presented in Subsection 3.2.1. Further, the 2-digit combination matches within those correct 4-digit PIN predictions could also be counted, results of which are in Subsection 3.2.2. These results provide insight into the model's performance and the specific patterns it detects in the keystroke timing data, laying the groundwork for understanding its strengths and limitations.

3.2.1 Correct 4-digit PIN Counts

Across all folds, the model achieved 275 correct predictions in the validation set, averaging approximately 11 PINs for each participant, and 2,235 correct predictions in the training set, averaging approximately 93 correct predictions for each participant. Per participant, the correct predictions were counted for each fold, providing a total correct count for training and validation. The count distribution was symmetric in both phases, with half the participants (12) above and half below the average correct predictions in validation (11.46) and training (93.13). Appendix Tables B.2 and B.3 detail the per-participant counts for validation and training, respectively. Additionally, for high-performing participants in both training and

validation, their percentage of correct predictions relative to their total samples in each phase was calculated to assess their relative accuracy. Figure 3.2 illustrates the total correct prediction counts.

Figure 3.2: Total correct PIN predictions per-participant in validation (blue) and training (orange).



In validation, Participant 21 led with 26 correct predictions (0.26% correct relative to total participant validation samples), followed by Participants 22 and 23 (21 each). In training, Participant 23 led with 197 correct predictions (0.49% correct relative to total participant training samples), followed by Participant 9 (185) and Participant 2 (150). Notably, Participant 18 had 0 correct predictions in validation and was the lowest in training with only 6 correct predictions. Participants who performed well in validation generally did so in training, but rankings differed: Participant 21 (top in validation) ranked 4th in training, and Participant 22 ranked 8th. Notably, Participant 24 showed an inverse trend, ranking 8th in validation (16 correct) but 19th in training (54 correct), making them the only participant to be above average in validation but below average in training.

For per-fold-specific trends, Participant 21 had a significant outlier of 12 correct pre-

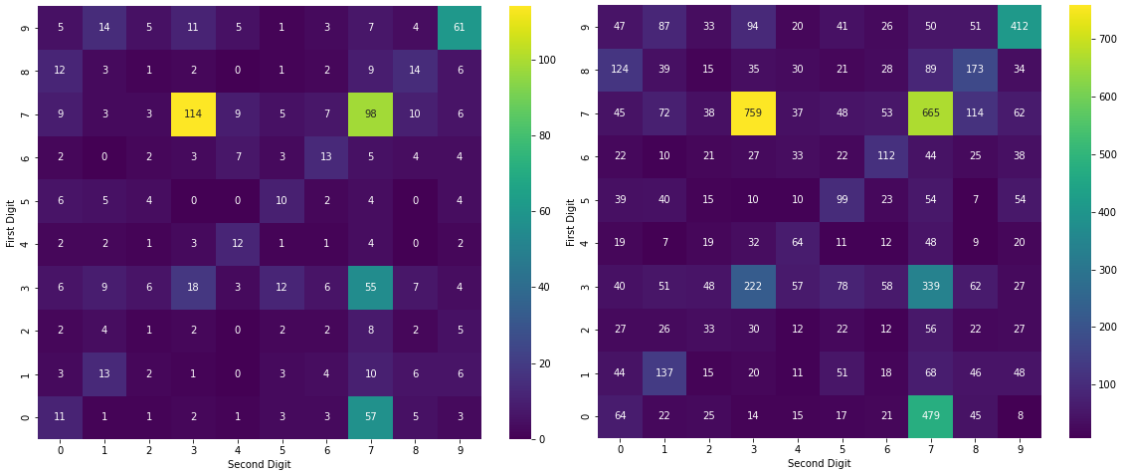
dictions in Fold 2 (validation), compared to 3–4 in the other folds, representing a ∼71% increase over the next-highest fold count of 7. The highest count in training for Fold 2 was from Participant 9 at a count of 45, which is ∼32% higher than the next highest count of 34.

3.2.2 2-digit Combination Counts

The model favored combinations where at least one digit was from the top row (7, 8, 9) of the 10-digit keypad, especially those involving the digit 7 or repeats (e.g., 77). The top 5 combinations accounted for 46.7% of the total correct predictions in validation, with only 6.1% of those being repeats (e.g., 7373). The training results mirrored these patterns, with the distribution being slightly more general: the top 5 accounted for 39.5% of the total, and only 3.1% were repeats. Furthermore, the diagonal of the validation and training heatmaps showed that repeat combinations had a relatively elevated count compared to their neighboring cells. In contrast to high-performing counts, in the validation data, 8 combinations had zero correct predictions, each including at least one digit from the middle row (4, 5, 6) of the 10-digit keypad. This was similar in training, with the bottom 10 combination counts (non-zero) being combinations with digits from the middle row, although these combinations still had some, albeit low, correct predictions. The total counts for each 2-digit combination are visualized with heatmaps, similar to the original data collection analysis in Section 2.2. Figure 3.3 presents the total counts for each combination for the training and validation phases, with the first number on the y-axis, and the second number on the x-axis.

These patterns exhibited variability in top rankings across folds and participants, though the preference for top-row digits generally held on both a per-fold and total-count basis. For validation, the top 5 combination rankings, their totals, and per-fold counts are shown in Table 3.3, while per-participant combination counts for top-performing participants are presented in Table 3.4, revealing shifts in order but consistent trends with the overall validation patterns. Training results followed similar trends, with detailed per-fold and per-participant

Figure 3.3: Heatmap of total correct predictions for two-digit combinations across validation (left plot) and training (right plot).



combination counts available in Appendix B (Tables B.5 and B.4, respectively).

Table 3.3: Top-5 combinations per-fold, with the final column summarizing the top combinations across all folds.

Rank	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
1	77	73	77	73	73	73
2	73	77	73	99	77	77
3	07	37	99	07	07	99
4	99	99	37	77	37	07
5	37	07	07	37	99	37

Table 3.4: Top 2-digit combinations for each of the top 3 participants across folds. For folds with the same highest count for multiple combinations, all tying combinations are provided.

Rank	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
21	77	73	73	73	73	73
22	73	93, 99, 37	73, 77	33	77	73
23	73	77	See Note	08	73, 07	73

Note: For Participant 23 in Fold 3, a 9-way tie occurred for the top counts, involving combinations 91, 99, 73, 74, 77, 57, 33, 37, and 15.

The next chapter delves deeper into these findings, evaluating the extent to which they support the hypothesis of a side-channel vulnerability, analyzing the model's behavior in the

context of keypad layout and participant variability, and identifying limitations and future research directions.

Chapter 4

Discussion

This chapter discusses the results presented in Chapter 3 in the context of the research objective: to investigate time latency patterns as a potential side-channel vulnerability for 4-digit PINs on PIN Entry Devices (PEDs) using machine learning. The hypothesis posits that a simple machine learning model can learn generalized keystroke timing patterns to predict 4-digit PINs. The discussion chapter first compares the model's performance to an established baseline, then analyzes supplementary outputs to assess pattern detection, evaluates the hypothesis, identifies limitations, and proposes future research directions.

4.1 Relative Model Performance

To evaluate the model's performance, random guessing is used as an appropriate baseline for the 10,000-class problem (4-digit PINs), where each PIN has an equal probability of selection. This baseline represents the null hypothesis that keystroke timing data contains no exploitable information, akin to random brute-force guessing. For this baseline, the probability of a correct top-1, top-5, or top-10 prediction is 0.01%, 0.05%, and 0.1%, respectively, with a baseline loss of $\ln(10,000) \approx 9.21$, derived from the cross-entropy loss for a uniform distribution over 10,000 classes (Section 2.4.2). In side-channel research, even modest improvements over random guessing can indicate a vulnerability, as they suggest detectable

patterns (Kocher, 1996b). Typical neural network tasks, such as image classification with 10–100 classes, often achieve accuracies of 80–90% due to denser datasets and fewer outputs (Goodfellow et al., 2016). Such benchmarks are inapplicable to this problem space, given the limited dataset size (240,000 samples, approximately 24 per class) and the large classification space, which dilute absolute accuracy.

The model's average accuracies across five stratified k-folds were a top-1 accuracy of 0.115%, top-5 accuracy of 0.603%, and top-10 accuracy of 1.198%. Compared to random guessing, the top-1 accuracy is 11.5 times higher, the top-5 accuracy is 12.06 times higher, and the top-10 accuracy is 11.98 times higher. The best top-1 accuracy (0.127% in Fold 1) is 12.7 times higher than random guessing, the best top-5 accuracy (0.623% in Fold 1) is 12.46 times higher, and the best top-10 accuracy (1.254% in Fold 2) is 12.54 times higher. The average validation loss of 8.469, with a best of 8.429 (Fold 4), is lower than the random guessing baseline, indicating the probability distribution output is better than chance.

These performance metrics suggest that the model detects timing patterns that enable it to predict PINs more effectively than random guessing, supporting the hypothesis of a side-channel vulnerability. The improvement in top-k accuracies as k increases indicates that the likelihood of the correct PIN being among the top predictions rises with a broader prediction range. Further models or analyses could leverage this to narrow the options to a specific top-k range in a large-scale cryptographic attack to identify vulnerable PINs. This is particularly relevant when considering research on the statistical frequency of common 4-digit PINs (Markert et al., 2020) and the effects of compounding probability (Blitzstein and Hwang, 2014). However, the absolute accuracies remain low, and the best validation loss (8.429 in Fold 4) did not correspond to the best accuracies, suggesting that minimizing loss does not always translate to increased precision. While variability across folds (e.g., Fold 1's best top-1 accuracy vs. Fold 4's best loss) might suggest sensitivity to data splits, the relatively small margins in box plots (Figure 3.1) indicate stable performance. This suggests that, on a large scale, there may be a certain probability of identifying a vulnerable PIN; however, targeting a specific individual's PIN offers little chance of successful exploitation.

4.2 Model Behavior from Supplementary Outputs

The supplementary outputs provide insight into the specific patterns the model detects. The model achieved 275 correct predictions in validation (averaging 11 per participant) and 2,235 in training (averaging 93 per participant), revealing a significant training-validation gap. This gap suggests overfitting, where the model learns patterns specific to the training data but struggles to generalize to unseen validation data. This is supported by the learning curves found in Appendix B (Figures B.1, B.2).

Participant-specific results highlight individual variability. Participant 21 led in validation with 26 correct predictions (0.26% accuracy relative to their validation samples), while Participant 23 led in training with 197 correct predictions (0.49% accuracy). Participant 18 had 0 correct predictions in validation and only 6 in training, indicating that some participants' timing patterns are far less predictable. This raises the question of whether there are different potential groupings of generalized keystroke timing behavior. Outliers, such as Participant 21's 12 correct predictions in Fold 2 (71% higher than their next-highest fold count), further emphasize per-fold-specific and participant-specific effects. The lack of alignment between training and validation rankings (e.g., Participant 21 was 4th in training despite being 1st in validation) suggests that the model's success is tied to individual typing behaviors rather than universal patterns, even though the participants' data were randomly distributed with the shuffled stratified K-fold training and validation approach.

The 2-digit combination analysis reveals the model's focus on specific patterns. The top-5 combinations (e.g., 73, 77, 99) accounted for 46.7% of correct predictions in validation and 39.5% in training, with repeats (e.g., 7373) being rare (6.1% in validation, 3.1% in training). Combinations involving top-row digits (7, 8, 9) and especially the digit 7 were prevalent, while those with middle-row digits (4, 5, 6) performed poorly, with 8 such combinations having zero correct predictions in validation. This may be because the movement characteristics from top-row numbers to other rows are more distinctive than those from the middle row. This is intuitively reasonable, as moving to or from the middle-row numbers exhibits the greatest symmetry in direction and distance compared to the extremes of the top or bottom

rows. The elevated counts along the heatmap diagonal (e.g., 77, 99) indicate that repeat combinations are more predictable, presumably because there is no direction or distance to travel, resulting in shorter latencies compared to other number combinations. Variability in top combinations across folds (e.g., 77 in Fold 1, 73 in Fold 2) and participants (e.g., 73 for Participant 21, ties for Participant 22) further underscores the model's reliance on context-specific patterns.

An intriguing observation emerges when considering psychological biases toward certain numbers. Early research, such as Simon and Primavera (1971), found that the digit 7 is disproportionately favored in subjective number selection, possibly due to cultural or cognitive factors. More recently, a popular exploration by Veritasium (2023) suggests that combinations like 73 persist in informal surveys of preferred two-digit numbers, hinting at a broader human affinity for digits like 3 and 7. Although participants in this study entered predetermined PINs rather than choosing numbers freely, the model's convergence on combinations such as 73 and 77 suggests that these digits may carry distinct timing signatures across participants, potentially due to unconscious familiarity influencing keystroke consistency. With a small dataset, the model may have prioritized these latent commonalities, suggesting that a larger dataset might uncover additional patterns while still reflecting such biases.

These findings partially support the hypothesis that a simple model can learn generalized timing patterns. The model detects patterns, as evidenced by its improvement over random guessing and focus on specific combinations. However, the patterns are not fully generalized, as they are concentrated in a small subset of combinations (e.g., 46.7% in the top-5) and vary significantly across folds and participants. The bias toward top-row digits and repeats, coupled with poor performance on middle-row digits, suggests that the model learns patterns driven by keypad layout or participant habits rather than universal timing signatures.

4.3 Limitations of the Study

The following section outlines the limitations that likely impacted the study's results. Given the model's sensitivity to participant and fold-specific datasets, assessing the dataset's short-

comings is crucial. Further, the model architecture was a more generalized approach that lacked any exploration of architectures or hyperparameter tuning. The following subsections describe these limitations.

4.3.1 Data Collection and Preprocessing Limitations

First, the data collection process lacked strong control measures for ensuring adherence to the virtual model. To maximize the number of participants, the data collection program was provided for download, along with concise requirements and instructions. As a result, some participants were local and could be observed, while others were not, making it impossible to monitor the data entry conditions for non-local participants. The non-local participants, who were not observed, likely used different keyboards, potentially introducing variability in timing data due to differences in hardware (e.g., key travel distance, layout) or behavior (e.g., not using the dominant hand's index finger for keystrokes). Of the participants who were observed, an issue was noted: participants read their combination instructions key by key, first reading the start number, inputting that number, then reading the end number and inputting that number. This could be attributed to the instructions being presented as a simple command-line print rather than an intuitive GUI design, but regardless, this behavior would disrupt the fluid typing expected when entering a familiar 4-digit PIN. A more controlled setup, such as using a uniform GUI, visual observation recording, and the same keyboard for all participants, could have improved dataset consistency.

Second, the timing measurements relied on Python's `perf_counter` function, which, while high-resolution, operates on a general-purpose operating system (e.g., Windows) rather than a real-time operating system (RTOS). Given the critical role of precise timing in this study, small variations due to OS jitter or scheduling delays—such as thread prioritization or background processes—could introduce noise, widening data variability and uncertainty. An RTOS, with its deterministic timing guarantees, might have reduced this jitter, potentially yielding more consistent latency measurements. This uncontrolled factor likely exacerbated the participant-specific variability and overfitting observed, as the model may have struggled

to distinguish true timing patterns from system-induced noise.

Third, the dataset was limited in quantity, with only 24 samples per class (240,000 total samples for 10,000 classes), resulting in sparsity that hindered generalization. This low data rate likely contributed to the model's overfitting, as seen in the training-validation gap. Additionally, participant variability significantly affected performance, with some participants (e.g., 21, 23) showing more predictable patterns than others (e.g., 18), suggesting that individual typing behaviors played a large role in the model's success. Without stronger control and observation, however, it is difficult to understand if the individuals' typing patterns were impacted by the data collection program, the process, or the physical keyboard arrangement.

Lastly, the nature of the model being 4-digit PIN combination predictions, while the data collection involved 2-digit latency measurements, may have had more of an effect than originally thought. The behavior of sequential inputs in a familiar 4-digit combination may differ from concatenating measurements together, as was done in the preprocessing defined in Section 2.3. Regarding Fitts' Law, a study by Roberts et al. (2016) found that users can adopt optimized movement strategies that deviate from the theoretical equation. Without having any validation or data collection of sequential 4-digit PIN latencies, this study cannot assess the effects of this on the model as it relates to the real-world application.

4.3.2 Modeling Approach and Design Limitations

While data-related challenges significantly affected the study, the modeling approach also introduced constraints, as discussed below.

The model architecture—a simple neural network—eliminated the exploration of more advanced architectures that may have been better suited for such a complex classification problem. This simplicity, combined with the high dimensionality, likely limited the model's ability to capture generalized latency patterns. Without hyperparameter tuning, it is difficult to assess whether this simple model architecture reached its full potential. Given the sensitivity to participant data and folds, it would have been beneficial to avoid shuffling the stratified K-fold dataset, thus preventing data leakage and allowing validation on participant

data and patterns entirely unseen during training.

Also, the input features being only the adjusted latency times provided a small number of features relative to the high number of output classes. This lack of additional features, such as total end-to-end time or latency deviations, may have further constrained the model's ability to detect broader timing signatures that could be exposed by the raw recorded latencies.

4.4 Future Research Directions and Application

Future research can address these limitations through several improvements. On the data collection side, implementing a standardized GUI with tighter controls (e.g., observation, uniform keyboards) could reduce variability and improve data consistency. Using an RTOS or a dedicated timing hardware setup could minimize OS-induced jitter, enhancing the precision of latency measurements and potentially revealing more robust timing patterns. Expanding the dataset by collecting more samples per class or using statistical augmentation (e.g., adding noise to timing data) could mitigate sparsity, providing more training examples for the model to learn generalized patterns. Further research into the sequential impacts by collecting 4-digit samples for model validation could test the applicability of concatenating measured latencies of 2-digit combinations.

Model-side enhancements offer significant potential. First, running a trade space on hyperparameters would identify the current architecture's full capability. Before trying different architectures, additional features, such as total end-to-end time or latency deviations, could be explored to provide richer input data, capturing higher-level timing patterns. Next, training strategies to reduce overfitting, such as weight penalties (e.g., L2 regularization) or dropout, could encourage the model to learn more general timing features, addressing the training-validation gap. Finally, different, more advanced architectures, such as hierarchical neural networks (HNNs) or long short-term memory networks (LSTMs), are better suited for sequential data and could improve pattern detection by modeling digit-to-digit transitions or temporal dependencies. Specifically, the hierarchical approach could investigate number-to-number patterns more effectively.

An innovative approach involves using Fitts' Law (incorporating directional bias coefficients) to create a synthetic dataset, which can then be validated against collected participant data to test theoretical predictions. By fitting Fitts' Law to a participant like 23, who exhibited predictable patterns (e.g., 197 correct predictions in training), movement times between keys can be modeled based on keypad geometry (e.g., distances between keys). This model can generate synthetic timing data for all 10,000 PINs, reflecting Participant 23's typing biases (e.g., distinctive transitions for top-row digits). The synthetic dataset can be used for training, with Participant 23's raw data as validation to test the model's accuracy. Extending this approach to multiple participants could validate its generalizability, potentially addressing data sparsity and providing a theoretical basis for timing patterns. This could also inform feature engineering, such as adding movement time as an input feature, or guide data augmentation strategies.

These directions offer opportunities to enhance the model's ability to detect generalized timing patterns, strengthen the side-channel attack potential, and further explore the physical and behavioral factors influencing keystroke timing patterns on PEDs.

4.5 Summary

This discussion has highlighted the model's ability to detect timing patterns in keystroke data, improving over random guessing but falling short of fully generalized predictions. The supplementary outputs revealed a focus on specific patterns, such as top-row digits and repeats, but also underscored participant-specific and per-fold-specific variability. An intriguing observation is the model's convergence on combinations like 37, 77, and 73, which may reflect underlying psychological biases, suggesting that human number preferences could subtly influence detectable timing patterns, even in predetermined PIN entry. Limitations in data collection, dataset size, and model design likely contributed to these challenges, as detailed in the preceding sections. The proposed future research directions, including improved data collection, advanced architectures, and synthetic data generation using Fitts' Law, offer pathways to address these limitations and further explore timing-based vulnerabilities in

PEDs. The next chapter will summarize the overall findings of this study and provide final remarks on the implications of keystroke timing as a side-channel vulnerability.

Chapter 5

Conclusion

This study aimed to investigate time latency patterns as a potential side-channel vulnerability for 4-digit PINs on PIN Entry Devices (PEDs) using a simple machine learning model, with the hypothesis that such a model could learn generalized keystroke timing patterns to predict 4-digit PINs. Through an exploratory evaluation of a 10,000-class classification problem, this research has provided insights into the feasibility of detecting timing-based vulnerabilities in keystroke data, while also highlighting significant challenges in achieving generalized predictions.

The results indicate that the model detects some keystroke timing patterns, as evidenced by its performance metrics surpassing random guessing baselines. The average best validation loss of 8.469 (compared to 9.21 for random guessing) and top-k accuracies (top-1 at 0.115%, top-5 at 0.603%, and top-10 at 1.198%, compared to 0.01%, 0.05%, and 0.1%, respectively) demonstrate that the model learns patterns beyond chance, with improvements ranging from 11.5 to 12.7 times better than random guessing. However, these absolute probabilities remain very low, with the top-1 accuracy indicating a correct prediction in only 1 out of every 870 attempts on average, and the top-10 accuracy in 1 out of every 83 attempts. Consistent with Chapter 1's premise that machine learning can exploit subtle human behavior patterns, these improvements suggest timing-based vulnerabilities exist, though not yet at a level posing significant cryptographic risk as anticipated in the introduction. While

these results demonstrate that timing-based vulnerabilities are indeed feasible, the model's limited success indicates that it did not learn robust or generalized patterns sufficient for practical exploitation in most cryptographic scenarios.

Supplementary outputs provided further insight into the model's behavior. The model achieved 275 correct predictions in validation (averaging 11.46 per participant) and 2,235 in training (averaging 93.13 per participant), revealing a significant training-validation gap suggestive of overfitting. Participant-specific variability was pronounced, with some participants (e.g., Participant 21 with 26 correct predictions in validation) showing more predictable patterns than others (e.g., Participant 18 with 0 correct predictions in validation). The two-digit combination analysis revealed a focus on specific patterns, particularly combinations involving top-row digits (7, 8, 9) and repeats (e.g., 77), which accounted for 46.7% of correct predictions in validation. In contrast, combinations with middle-row digits (4, 5, 6) performed poorly, indicating that the model's success is tied to keypad layout and participant-specific typing behaviors rather than universal timing signatures. As anticipated by Fitts' Law in Chapter 1's background, this bias toward top-row combinations suggests keypad layout shapes timing predictability, though participant variability limited its universality.

These findings provide only partial support for the hypothesis that a simple machine learning model can learn generalized timing patterns to predict 4-digit PINs. While the model detects some patterns that improve prediction accuracy over random guessing, these patterns are not fully generalized, as they are concentrated in a small subset of combinations and vary significantly across folds and participants. The concentration of correct predictions on top-row digits reflects the overlap in soft biometric patterns noted in Chapter 1, though the lack of broader generalization suggests that this vulnerability is specific to certain contexts rather than universally applicable. This suggests that the side-channel vulnerability identified in this study is context-specific, influenced by factors such as individual typing habits and the specific keypad layout used (with 7, 8, 9 on the top row).

In the context of cryptographic threats, the low probabilities make a targeted attack on a specific individual (e.g., an administrator) highly unlikely. However, the findings have more

significant implications for large-scale, non-targeted attacks, such as gaining access to a room with many authorized users. In such scenarios, the attacker does not need to compromise a specific PIN but rather any PIN within the population. The larger the population, the more likely it is that at least one individual exhibits a general typing pattern that the model can exploit, such as a preference for top-row digits or repeats. For example, in a population of 100 users, the model's top-10 accuracy of 1.198% suggests a cumulative probability of success that is higher than that of targeting a single individual, though still limited by the low absolute probabilities and the context-specific nature of the patterns. This highlights a potential risk in access control systems with large user bases, where the diversity of typing behaviors increases the chance of finding a vulnerable PIN.

The findings of this study do not indicate an immediate need for organizations to take security action, such as replacing PEDs that may expose timing data leakage through feedback features (e.g., auditory beeps). However, on a personal level, individuals could consider improving their security by making a conscious effort to maintain variability when entering their PIN into an authentication-passcoded device. For example, varying the speed or rhythm of keystrokes could reduce the predictability of timing patterns, thereby mitigating the risk of exploitation in scenarios where timing data might be exposed. This aligns with the security-usability tension noted in Chapter 1, where enhancing security through variable typing may slightly complicate the user experience but could mitigate timing-based risks.

Several limitations hindered the study's ability to provide stronger support for the hypothesis. The data collection process lacked strong controls, with some participants unobserved and using different keyboards, introducing variability in the timing data. The dataset's sparsity (24 samples per class for 10,000 classes) and the preprocessing approach of concatenating two-digit latency measurements to represent 4-digit PINs further constrained the model's ability to generalize. Additionally, the simple neural network architecture, without hyperparameter tuning or advanced features, limited the model's capacity to capture broader keystroke timing patterns.

Given the exploratory nature of this study, the results are promising but inconclusive, suggesting that further research is needed to either strengthen or weaken support for the

hypothesis. Improving data collection controls, expanding the dataset, and collecting sequential 4-digit PIN latencies could enhance dataset quality and generalizability. Model-side enhancements, such as exploring advanced architectures (e.g., hierarchical neural networks or LSTMs), incorporating additional features (e.g., total end-to-end time), and using synthetic data generation with Fitts' Law, offer promising directions to improve pattern detection and address the identified limitations. Additionally, the unexpected prominence of combinations like 73 and 77, possibly linked to psychological biases toward digits 3 and 7 as noted in Simon and Primavera (1971) and Veritasium (2023), highlights an intriguing intersection of human behavior and machine learning that warrants further exploration.

In conclusion, this research demonstrates that keystroke timing patterns can be detected to some extent, providing partial evidence of a side-channel vulnerability in PEDs. Fulfilling Chapter 1's objectives, Objective 3's risk evaluation revealed a modest threat, while Objective 4's pattern analysis pinpointed layout-specific biases, though data and model constraints limited the full realization of the goal of achieving user-agnostic predictions across a 10,000-class problem. However, the low probabilities and lack of fully generalized patterns indicate that significant challenges remain before such vulnerabilities could be practically exploited, particularly in targeted attack scenarios. The findings are more relevant in the context of large-scale attacks, where the model's ability to exploit general typing patterns in a population increases the likelihood of success. Unlike the multi-feature studies in Chapter 1, this timing-only focus yielded limited generalization, highlighting the specific data requirements of this approach. These results contribute to the understanding of timing-based vulnerabilities in security systems and underscore the need for further research to refine data collection and modeling approaches, paving the way for more robust investigations into keystroke dynamics on PEDs.

Bibliography

Agrawal, D., Archambeault, B., Rao, J., and Rohatgi, P. (2003). The em side–channel(s). In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, pages 29–45. Springer. Discusses electromagnetic side-channel attacks exploiting variations in electromagnetic emissions to extract cryptographic keys.

AL-Qawasmeh, N., Khayyat, M., and Suen, C. Y. (2023). Age detection from handwriting using different feature classification models. *Pattern Recognition Letters*, 167:60–66.

Amrouche, A., Boubchir, L., and Yahiaoui, S. (2022). Side channel attack using machine learning. In *2022 Ninth International Conference on Software Defined Systems (SDS)*, pages 1–5.

Balamurugan, M. (2024). Biometric authentication: A double-edged sword for security? *International Journal of Science and Research (IJSR)*, 13:170–173.

Batina, L., Bhasin, S., Jap, D., and Picek, S. (2019). Csi nn: reverse engineering of neural network architectures through electromagnetic side channel. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, pages 515–532, USA. USENIX Association.

Beltramelli, L., Conti, M., and Lovisotto, G. (2022). Machine learning for pin side-channel attacks based on smartphone motion sensors. *IEEE Transactions on Information Forensics and Security*, 17:2276–2287.

Bianchi, A., Oakley, I., and Kwon, D. (2012). Counting clicks and beeps: Exploring numerosity based haptic and audio pin entry. *Interacting with Computers*, 24:409–422.

Blitzstein, J. K. and Hwang, J. (2014). *Introduction to Probability*. CRC Press.

Chakraborty, N., Li, J., Mondal, S., Chen, F., and Pan, Y. (2019). On overcoming the identified limitations of a usable pin entry method. *IEEE Access*, 7:124366–124378.

Chiari, G., Galli, D., Lattari, F., Matteucci, M., and Zoni, D. (2024). A deep-learning technique to locate cryptographic operations in side-channel traces. In *2024 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE.

Dantcheva, A., Elia, P., and Ross, A. (2016). What else does your biometric data reveal? a survey on soft biometrics. *IEEE Transactions on Information Forensics and Security*, 11(3):441–467.

Dantcheva, A., Velardo, C., and D'Angelo, A. (2011). Bag of soft biometrics for person identification. *Multimedia Tools and Applications*, 51:739–777.

Drimer, S., Murdoch, S. J., and Anderson, R. (2009). Failures of tamper-proofing in pin entry devices. *IEEE Security and Privacy*, 7(6):39–45.

Fitriani, Y., Akbar, S., and Suryotrisongko, H. (2022). Spoofing keystroke dynamics authentication through synthetic typing pattern extracted from screen-recorded video. *Journal of Big Data*, 9(1):114.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391. Reprinted in Journal of Experimental Psychology: General, 121(3), 262–269, 1992.

Framework (2024). Default keymap for the numpad. https://knowledgebase.frame.work/default-keymap-for-the-numpad-rJZv44owa. Published on 1 February 2024, Accessed: 2025-04-07.

Gomez-Barrero, M. and Galbally, J. (2020). Reversing the irreversible: A survey on inverse biometrics. *Computers & Security*, 90:101700.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grassi, P., Garcia, M., and Fenton, J. (2017). Digital identity guidelines. Published on 22 June 2017.

Harrison, J., Toreini, E., and Mehrnezhad, M. (2023). A practical deep learning-based acoustic side channel attack on keyboards. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 270–280. IEEE.

Hattali, S., Hussain, S., and Frank, A. (2020). Design and development for detection and prevention of atm skimming frauds. *Indonesian Journal of Electrical Engineering and Computer Science*, 17:1224–1224.

Haykin, S. (2009). *Neural Networks and Learning Machines*. Pearson, 3rd edition.

Infosec Institute (2024). The future of machine learning in cybersecurity. Published on 13 February 2024, Accessed: 2025-04-06.

Intelligence, M. (2025). Pin-based access control market size & share analysis - industry research report - growth trends and forecasts (2025 - 2030). Accessed: 2025-04-06.

Kantowitz, B. H. and Elvers, G. C. (1988). Fitts' law with an isometric controller: Effects of order of control and direction of movement. *Human Factors*, 30(6):727–736.

Kocher, P. C. (1996a). Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz, N., editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer.

Kocher, P. C. (1996b). Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. *Advances in Cryptology — CRYPTO '96*, 1109:104–113.

Kopecky, S. (2017). Cyber security paradox from a user's view point. In *2017 Computing Conference*, pages 783–787.

Kour, R., Karim, R., Dersin, P., and Venkatesh, N. (2024). Cybersecurity for industry 5.0: trends and gaps. *Frontiers in Computer Science*, 6.

Kubota, T., Yoshida, K., Shiozaki, M., and Fujino, T. (2021). Deep learning side-channel attack against hardware implementations of aes. *Microprocessors and Microsystems*, 87:103383.

Kurban, O. C. and Yildirim, T. (2024). A comparative analysis of multi-biometrics performance in human and action recognition using silhouette thermal-face and skeletal data. *Neural Networks*, 170:1–17.

Lee, P.-M., Tsui, W.-H., and Hsiao, T.-C. (2015). The influence of emotion on keyboard typing: An experimental study using auditory stimuli. *PLOS ONE*, 10(6):1–16.

MacKenzie, I. S. and Buxton, W. (1992). Extending fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*, pages 219–226. ACM.

MacKenzie, I. S. and Soukoreff, R. W. (2002). Using Fitts' Law to model key repeat time in text entry models. Poster presented at Graphics Interface 2002.

Markert, P., Bailey, D., Golla, M., Durmuth, M., and Aviv, A. (2020). This pin can be easily guessed: Analyzing the security of smartphone unlock pins. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 286–303.

Mehrnezhad, M., Toreini, E., Shahandashti, S. F., and Hao, F. (2018). Stealing pins via mobile sensors: Actual risk versus user perception. *International Journal of Information Security*, 17:291–313.

Nisbet, A. and Kim, M. (2016). An analysis of chosen alarm code pin numbers & their weakness against a modified brute force attack. In Johnstone, M., editor, *The Proceedings of 14th Australian Information Security Management Conference*, pages 21–29, Edith Cowan University, Perth, Western Australia. Held on 5–6 December 2016.

Nyang, D., Kim, H., Lee, W., Kang, S.-b., Cho, G., Lee, M.-K., and Mohaisen, D. (2018). Two-thumbs-up: Physical protection for pin entry secure against recording attacks. *Computers & Security*, 78:1–15.

of Homeland Security, U. D. (2025). Biometrics. Accessed: 2025-04-06.

Panda, S., Liu, Y., Hancke, G. P., and Qureshi, U. M. (2020). Behavioral acoustic emanations: Attack and verification of pin entry using keypress sounds. *Sensors*, 20(11).

Piugie, Y. B. W., Di Manno, J., Rosenberger, C., and Charrier, C. (2022). Keystroke dynamics based user authentication using deep learning neural networks. In *2022 International Conference on Cyberworlds (CW)*, pages 220–227.

Roberts, J., Blinch, J., Elliott, D., Chua, R., Lyons, J., and Welsh, T. (2016). The violation of fitts' law: an examination of displacement biases and corrective submovements. *Experimental Brain Research*, 234:2167–2179.

Roth, V., Richter, K., and Freidinger, R. (2004). A pin-entry method resilient against shoulder surfing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 236–245, New York, NY, USA. Association for Computing Machinery.

Serwadda, A., Phoha, V. V., Wang, Z., Kumar, R., and Shukla, D. (2016). Toward robotic robbery on the touch screen. *ACM Trans. Inf. Syst. Secur.*, 18(4).

Sharma, S., Bhatt, M., and Sharma, P. (2020). Face recognition system using machine learning algorithm. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1162–1168.

Simon, W. E. and Primavera, L. H. (1971). The number seven in subjective probability. *Perceptual and Motor Skills*, 33(2):373–374.

Song, C., Huang, Y., Huang, Y., Jia, N., and Wang, L. (2019). Gaitnet: An end-to-end network for gait based human identification. *Pattern Recognition*, 96.

Spreitzer, R., Moonsamy, V., Korak, T., and Mangard, S. (2018). Systematic classification of side-channel attacks: A case study for mobile devices. *IEEE Communications Surveys & Tutorials*, 20(1):465–488.

Venkata, R. N., Nair, S. A. H., and Kumar, K. (2022). Hybrid biometric based person identification using machine learning. *Journal of Pharmaceutical Negative Results*, pages 1605–1614. Accessed: 2025-04-06.

Veritasium (2023). Why is this number everywhere? https://www.youtube.com/watch?v=d6iQrh2TK98. Accessed: 2025-04-07.

Wang, E. M.-Y. and Shih, S. S.-Y. (2009). A study on thumb and index finger operated interface for personal mobile devices: Mobile phone keypad and joystick. In *2009 16th International Conference on Industrial Engineering and Engineering Management*, pages 766–770.

Whelan, R. (2008). Effective analysis of reaction time data. *The Psychological Record*, pages 475–482.

Yan, Q., Han, J., Li, Y., and Deng, R. H. (2010). Timing attacks on pin input devices. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 720–722.

Zhang, G. and Fischer-Hübner, S. (2011). Timing attacks on pin input in voip networks (short paper). In *Proceedings of the 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 75–84.

Appendix A

Supporting Data Collection Visuals

This appendix presents supporting plots from relevant and notable participants referenced throughout the thesis. Their minimum latency time heatmaps are provided to facilitate comparisons and analyses between the statements in Chapters 3 and 4 and the data samples of these participants.

Figure A.1: Heatmap for Participant 18's minimum latency time data used for training and validation, where no predictions were correctly made in validation, and the participant ranked lowest in training predictions.

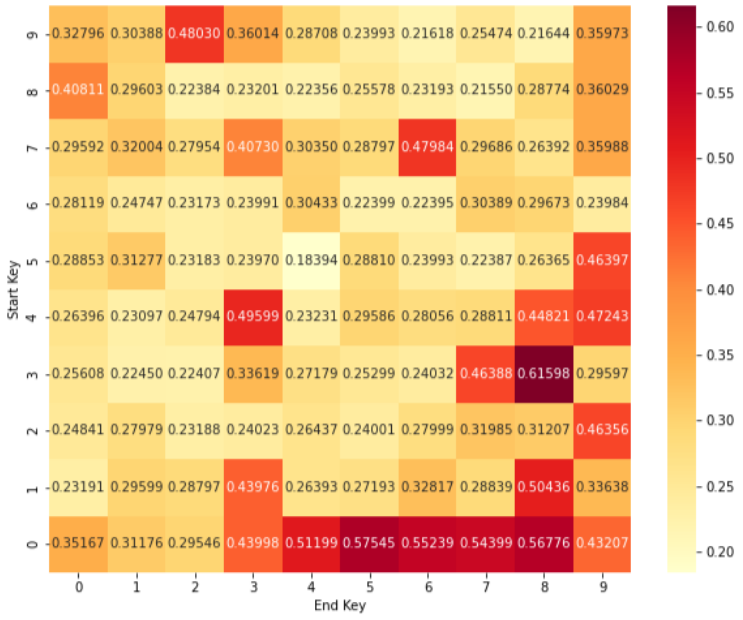


Figure A.2: Heatmap for Participant 21's minimum latency time data used for training and validation, where their PINs were among the highest-ranked correct predictions in validation.

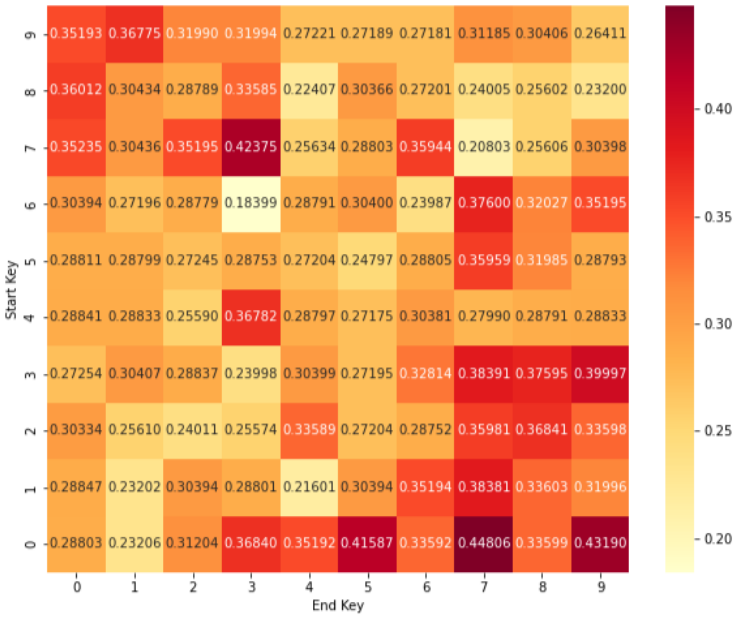
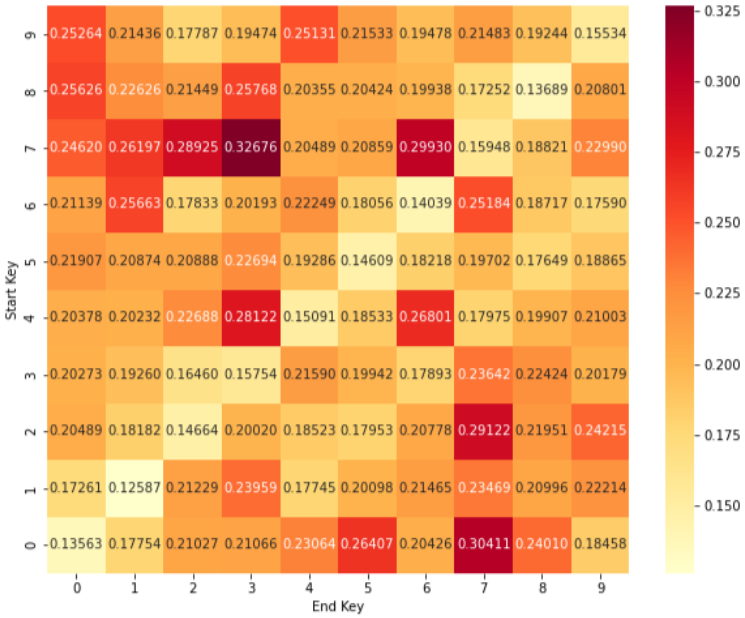


Figure A.3: Heatmap for Participant 23's minimum latency time data used for training and validation, where their PINs were among the highest-ranked correct predictions in training.



Appendix B

Detailed Supporting Model Results

This appendix presents supporting plots and tables for the research results. They are provided for further review and analysis, as they pertain to Chapter 3 and are discussed in Chapter 4. This includes novel data as well as expanded data condensed in the main body of the thesis.

Table B.1: Best validation loss and accuracies per fold in the five-fold cross-validation, providing detailed performance metrics for reference.

Fold	Best Loss	Best Top-1 (%)	Best Top-5 (%)	Best Top-10 (%)
1	8.484	0.127	0.623	1.227
2	8.476	0.106	0.600	1.254
3	8.468	0.113	0.573	1.114
4	8.429	0.104	0.602	1.223
5	8.489	0.123	0.615	1.148

Figure B.1: Learning curve for Fold 1, where the best top-1 accuracy was achieved, showing early learning followed by overfitting over an epoch time series.

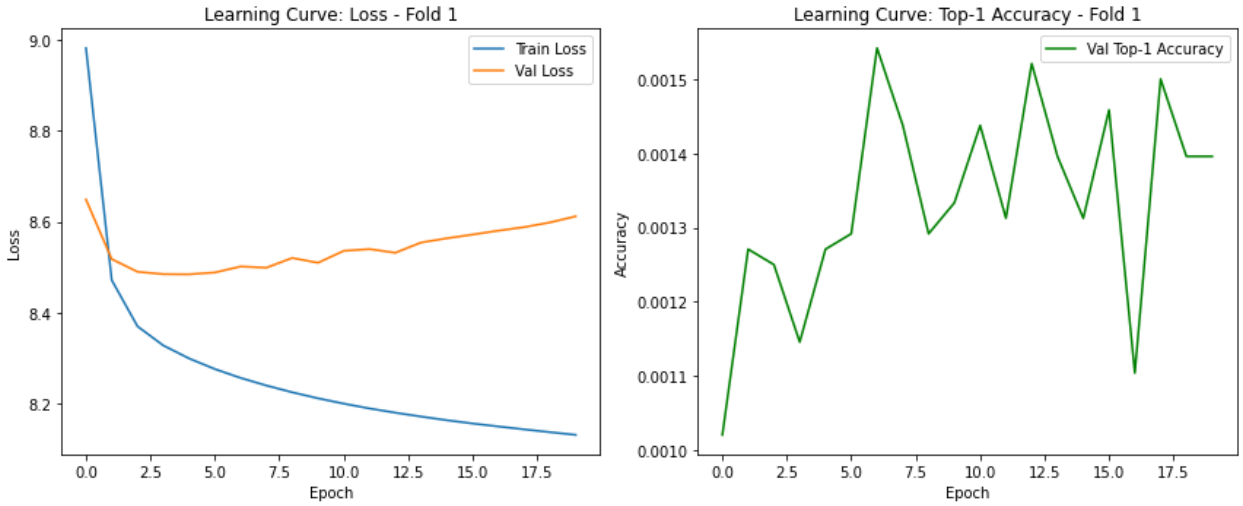


Figure B.2: Learning curve for Fold 4, where the best validation loss was achieved, showing early learning followed by overfitting over an epoch time series.

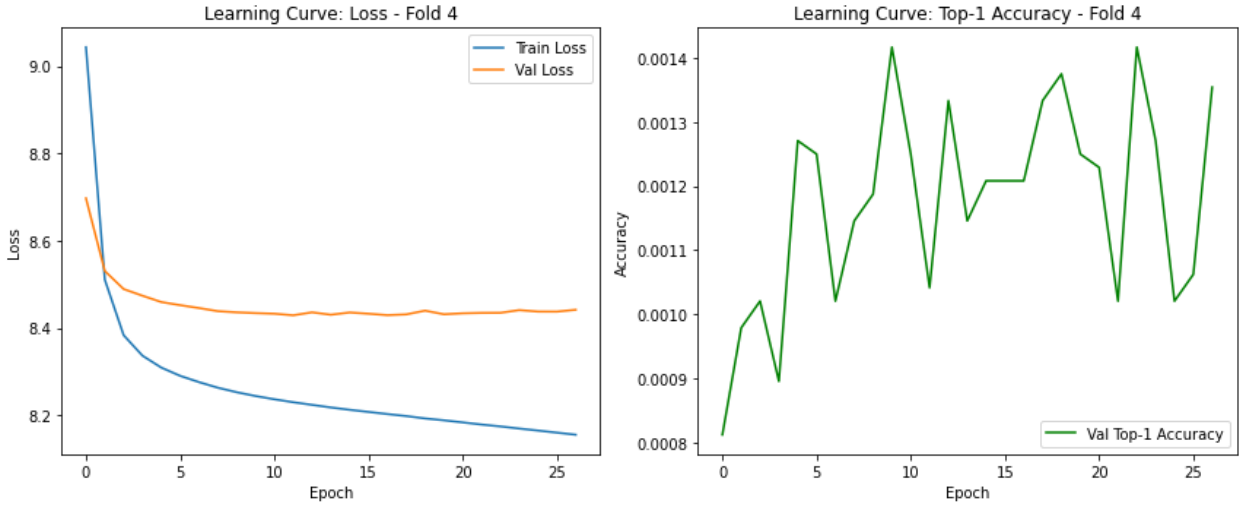


Table B.2: Correct PIN Prediction Counts for the Validation Set Across All Folds

Rank	Participant	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Total
1	21	3	12	4	3	4	26
2	23	6	1	3	6	5	21
3	22	5	1	4	5	6	21
4	12	6	3	7	1	3	20
5	2	5	3	6	1	3	18
6	9	4	2	2	4	5	17
7	13	2	4	3	3	4	16
8	24	5	4	2	4	1	16
9	11	3	1	3	2	6	15
10	4	2	1	3	5	2	13
11	3	4	0	4	0	4	12
12	19	1	4	2	3	2	12
13	20	2	0	3	2	3	10
14	6	3	1	1	1	3	9
15	14	2	0	0	4	2	8
16	1	3	2	0	1	1	7
17	16	0	3	2	0	1	6
18	17	1	2	2	1	0	6
19	7	1	2	0	1	1	5
20	8	1	1	0	1	2	5
21	15	1	3	1	0	0	5
22	5	1	0	1	1	1	4
23	10	0	1	1	1	0	3
24	18	0	0	0	0	0	0

Table B.3: Correct PIN Prediction Counts for the Training Set Across All Folds

Rank	Participant	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Total
1	23	43	32	39	43	40	197
2	9	33	45	32	42	33	185
3	2	27	34	40	27	22	150
4	3	33	26	24	33	31	147
5	21	32	24	25	33	26	140
6	13	14	23	30	33	33	133
7	12	23	28	24	26	30	131
8	22	22	22	26	28	28	126
9	11	15	26	34	23	24	122
10	19	13	23	35	25	26	122
11	6	17	20	17	24	22	100
12	20	18	16	22	16	22	94
13	4	15	24	18	14	13	84
14	16	16	16	15	13	18	78
15	14	9	16	11	10	14	60
16	1	12	15	11	14	8	60
17	17	10	15	12	11	11	59
18	7	18	14	8	8	7	55
19	24	9	9	15	9	12	54
20	8	9	14	7	8	7	45
21	15	3	9	7	10	10	39
22	10	6	5	8	9	5	33
23	5	2	3	3	4	3	15
24	18	1	0	2	1	2	6

Table B.4: Top 5 combinations per fold in training, with the final column summarizing the top combinations across all folds.

Rank	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
1	77	73	73	73	73	73
2	73	77	77	99	97	77
3	07	99	07	07	07	07
4	99	07	37	77	99	99
5	37	37	99	37	37	37

Table B.5: Top 5 combinations per fold in training, with the final column summarizing the top combinations across all folds.

Rank	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
21	77	73	73	73	07,77	73
22	77	73	73	73	73	73
23	77	73	73	73	77	73