

Article

Automated Acoustic Side-Channel Attack on Keyboard Inputs via Combined Video–Audio Analysis

Dario Vranješ , Ivo Stančić , Marin Bugarić  and Toni Perković * 

Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB), University of Split, 21000 Split, Croatia; dario.vranjes.00@fesb.hr (D.V.); istancic@fesb.hr (I.S.); mbugaric@fesb.hr (M.B.)

* Correspondence: toperkovic@fesb.hr

Abstract

Acoustic side-channel attacks (ASCAs) exploit unintended sound emitted by keyboards to infer typed input, but existing methods generally assume manually labelled training data and controlled environments, limiting their applicability to realistic scenarios such as online lectures. We develop a pipeline that automatically labels keystroke-sound samples captured from online coding tutorials: video frames are processed with optical character recognition (OCR) to extract the ground-truth character sequence, audio is segmented into clips centred on detected click events, and the two streams are aligned. A convolutional neural network (CNN) is trained on mel-spectrogram features, with transfer learning used to adapt the pretrained model to a target user with minimal samples. The classifier is trained on all 68 physical keys present in the recordings; of these, 50 produce a character or whitespace and the remaining 18 are control, navigation, and modifier keys. On a held-out test set, the CNN achieves 98.1% top-1, 99.4% top-2, and 100% top-3 accuracy. Transfer learning retains strong performance with as few as 13 samples per key. Pairing OCR-derived ground truth with acoustic CNN classification removes the labelling bottleneck that has limited previous ASCAs, and the transfer-learning stage makes the attack viable with minimal per-victim data. All code, trained models, and labelled datasets are released to support reproducible research.

Keywords: acoustic side-channel; keyboard; convolutional neural network; video analysis; mel-spectrogram; password security

1. Introduction

Microphones embedded in laptops, smartphones, and headsets—together with ubiquitous teleconferencing and live-streaming—have expanded the practical attack surface for acoustic side-channel attacks (ASCAs). In this setting, an adversary infers sensitive user input by analysing the unintended sound emissions produced during typing, without direct visibility of the keyboard or the screen. Classic work demonstrated that different keys can be distinguished from their acoustic signatures when a classifier is trained on labelled keystroke recordings, showing the fundamental feasibility of keyboard sound recognition under controlled conditions [1]. Subsequent research highlighted that the threat extends beyond carefully curated laboratory data: language constraints and statistical structure can be exploited to reconstruct text and even guess passwords with relatively few attempts, amplifying the real-world impact of imperfect acoustic inference [2,3].

More recent studies show rapid progress driven by modern machine learning and more realistic recording channels. In particular, VoIP and video-conferencing environments



Academic Editors: Indranil Roy and Ansuman Bhattacharya

Received: 17 June 2026

Revised: 24 July 2026

Accepted: 5 August 2026

Published: 7 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

enable remote acoustic eavesdropping because keystroke clicks can be transmitted (sometimes after compression and noise suppression) to meeting participants or to an attacker who can capture the audio stream [4]. Deep learning has further improved accuracy and robustness across devices and channels: CNN-based approaches report high recognition rates on laptop keyboards recorded by nearby smartphones and remain effective when the audio is conveyed via platforms such as Zoom [5]. Related work also investigates end-to-end transcription settings (joint detection and recognition) and robustness to overlapping keystrokes and background noise [6], as well as improved acoustic modelling for keystroke eavesdropping with robust features and classification pipelines [7,8]; more recent directions combine visual transformers and large language models to correct mispredictions under noise [9], or exploit user-specific typing patterns instead of raw click acoustics [10]. Progress on the acoustic channel parallels rapid advances in multimodal video understanding—for example, diffusion-guided video large multimodal models [11]—which increasingly lower the barrier to exploiting the *visual* channel of the online recordings that our labelling pipeline also relies on. The threat has broadened beyond the acoustic channel alone: recent surveys catalogue an AI-driven expansion of acoustic side-channel attacks [12], while defensive work has begun to fight back with learned adversarial perturbations that deny the eavesdropper a usable signal [13].

Despite this progress, a key bottleneck remains scalable ground-truth acquisition. Many high-performing systems depend on labelled training data that pairs each acoustic click with the corresponding character [1,5,8]. In realistic scenarios—e.g., online coding sessions, programming tutorials, remote exams, or live-streamed demonstrations—the required labels are not readily available at scale: manual annotation is prohibitive, and sensitive input is often masked or ephemeral. This limits both reproducibility and the ability to study variability across typists, keyboards, languages, and conferencing pipelines.

This paper addresses the labelling bottleneck by introducing an automated pipeline that extracts labelled keystrokes from video recordings that contain both (i) the screen content and (ii) the corresponding audio. We leverage OCR over video frames to recover the character sequence as it appears on the display and align it with detected keystroke events in the audio track, producing large labelled datasets with minimal manual effort. Using these data, we train and evaluate a CNN-based classifier on mel-spectrogram representations and assess performance under realistic conferencing conditions. To enable effective learning with a limited number of available samples, transfer learning is employed by initialising the network from a previously trained, generalised model. In this way, previously learned audio representations are leveraged, reducing the amount of task-specific training data required while maintaining robust performance. Compared to prior work that either assumes labelled data collection per target [1,5] or focuses on different threat models (e.g., dictionary recovery, end-to-end transcription, or purely VoIP capture) [3,4,6], our pipeline explicitly targets scalability and realism in dataset creation while maintaining a practical acoustic attack setting.

Motivating Analysis: Acoustic Differences Between Similar Keys

To illustrate why keystroke classification is feasible, we compare the acoustic signatures of visually similar keys. Figure 1 shows mel-spectrograms for two key pairs: the lowercase letter *i* versus the digit 1 and the comma versus the period. In both cases, the upper panel displays the first key and the lower panel the second key. Distinct patterns are apparent despite the keys' visual similarity. For the *i* versus 1 pair, the digit exhibits stronger energy at mid-high frequencies (1–5 kHz) and a broader time–frequency footprint, whereas the letter *i* concentrates energy at lower frequencies and decays more quickly.

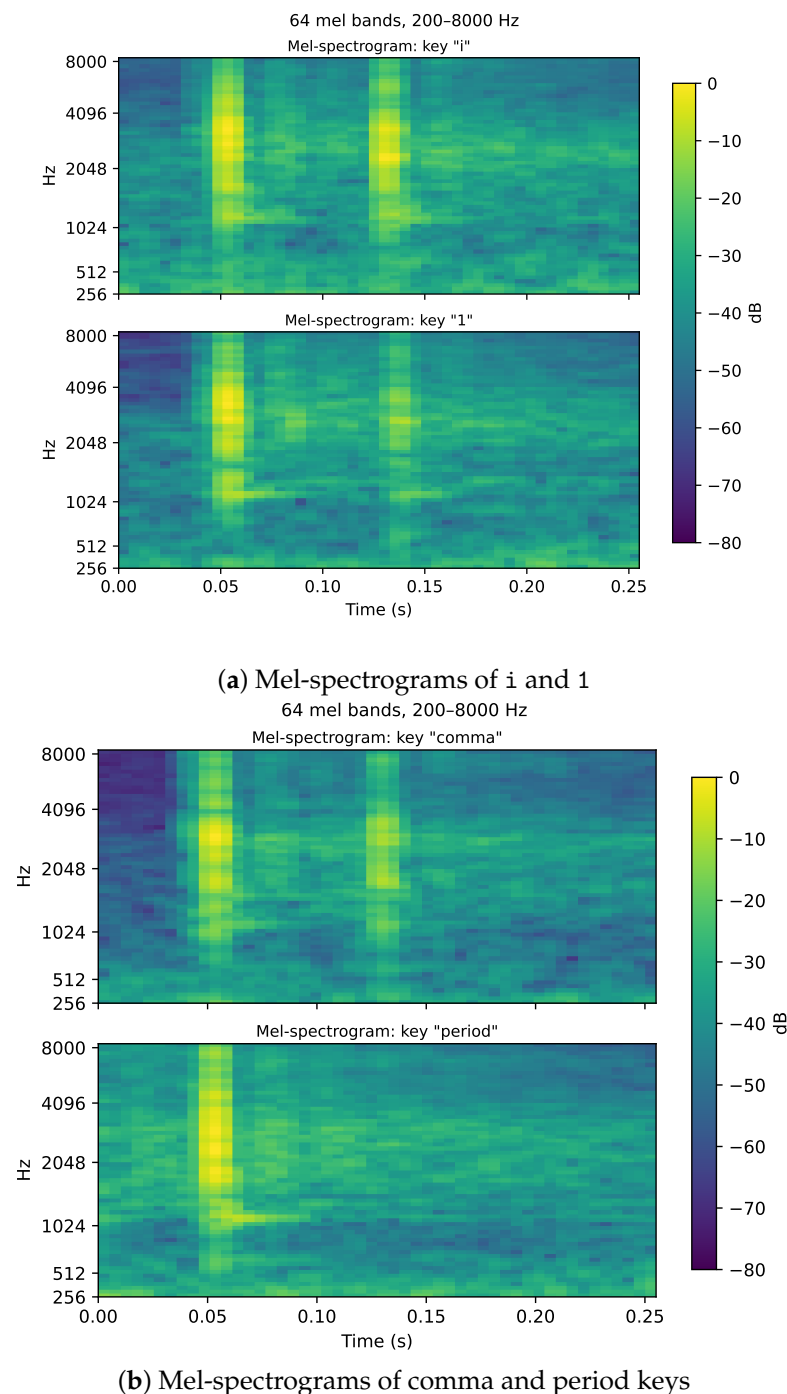


Figure 1. Mel-spectrogram comparisons for representative key pairs. Each subplot shows the mel-spectrograms for two keys. Despite the keys' visual similarity, distinct energy distributions across frequency and time are visible, highlighting the discriminative information captured by mel-spectrogram features.

To move beyond a qualitative comparison, we computed a set of standard, amplitude-invariant spectral statistics—spectral centroid, spectral rolloff, spectral bandwidth, dominant frequency, and zero-crossing rate (ZCR)—for every recorded sample of each key, and we summarised the separation between paired keys with the effect size (Cohen's d^*) in Table 1. Because these statistics are amplitude-invariant, they are unaffected by the unit-RMS normalisation applied during pre-processing.

The i/1 pair is highly separable: the digit has a higher spectral centroid (4836 vs. 4241 Hz), zero-crossing rate (0.035 vs. 0.025), spectral bandwidth (4219 vs. 3881 Hz) and dominant

frequency (40 vs. 15 Hz)—all with large effect sizes ($|d| \geq 1.5$)—together with a broader time–frequency footprint (effective duration 176 vs. 136 ms). These differences produce a per-band mel drift with an L2 norm of 34.9 dB, concentrated near 1 and 5 kHz. The comma/period pair is more subtle: its strongest separators are the ZCR (higher for the comma, $d = 0.60$) and the effective duration (longer for the period, $d = 0.52$), accompanied by a 25.9 dB mel drift near 1.1–1.3 kHz. Amplitude-based measures such as peak amplitude and PSD mean are gain-dependent and therefore unreliable across recordings, so we do not use them as discriminators. A complete worked example of the mel-spectrogram conversion, including a single-band hand calculation, is given in Appendix A. These results provide quantitative evidence that individual keystrokes emit distinguishable acoustic patterns, motivating our use of mel-spectrograms as a discriminative feature representation.

Table 1. Spectral statistics for visually similar key pairs (mean over 125 samples per key) and the effect size (Cohen’s d) between paired keys. All statistics are amplitude-invariant. Large effects ($|d| \geq 0.8$) are shown in bold.

Feature	Pair i/1			Pair Comma/Period		
	i	1	d	Comma	Period	d
Spectral centroid (Hz)	4241	4836	1.82	4327	4231	−0.36
Spectral bandwidth (Hz)	3881	4219	1.68	3983	3994	0.06
Zero-crossing rate	0.025	0.035	1.53	0.025	0.022	−0.60
Dominant frequency (Hz)	14.8	39.6	1.48	32.7	24.1	−0.43
Spectral rolloff 85% (Hz)	8992	9237	0.50	9029	9132	0.17
Effective duration (ms)	136	176	0.96	126	149	0.52
Per-band mel drift (L2, dB)	34.9 (cos 0.998)			25.9 (cos 0.9995)		

2. Background and Related Work

2.1. Acoustic Side-Channel Attacks on Keyboards

Side-channel attacks exploit information leaked by physical implementations (e.g., timing, power, EM radiation, or sound) to infer secrets without breaking cryptography directly. For keyboards, early demonstrations established that keystroke acoustics contain discriminative patterns and that supervised models can learn key-specific signatures when labelled samples are available [1]. A major leap in practicality came from approaches that reduce reliance on explicit labels by combining clustering with linguistic constraints. Zhuang et al. showed that character recovery can be bootstrapped from a short recording and refined using language statistics, enabling high character recovery and facilitating password guessing with few attempts [2]. Berger et al. further strengthened the password-recovery narrative by formalising dictionary-style attacks that combine acoustic classification with structured search over likely strings [3]. Halevi and Saxena [14] subsequently quantified how typing style, keyboard model and the choice of acoustic representation affect attack accuracy, and most recently Fürst and Aßmuth [15] showed that natural-language passphrases can be partially or fully recovered with purely unsupervised acoustic clustering. These works collectively show that even imperfect keystroke classification can be operationally dangerous when combined with language priors; an up-to-date taxonomy of acoustic side-channel attacks on keyboards is given by Taheritajar et al. [16], and a complementary AI-centric survey covering 2020–2025 is provided by Quattrone and Badr [12].

2.2. Remote Channels: VoIP and Video-Conferencing

A particularly relevant threat model is remote eavesdropping via communication software. Compagno et al. demonstrated that keystroke sounds transmitted through VoIP calls can be exploited for inference, highlighting conferencing platforms as an attack

vector [4]. More recently, deep learning approaches report strong performance when training and testing on audio captured through conferencing pipelines, suggesting that compression and denoising do not necessarily eliminate the side-channel [5,8]. In parallel, end-to-end transcription work emphasises that realistic settings must handle imperfect keystroke detection, overlap, and environmental noise, shifting evaluation from isolated classification toward pipeline robustness [6]. Robust modelling of these factors is essential for understanding the true feasibility of attacks outside controlled labs.

2.3. Feature Representations and Augmentation

Most modern keyboard acoustic side-channel attack (ASCA) pipelines convert raw waveforms into time–frequency representations to expose discriminative structure. Mel-spectrograms are widely used because they compactly represent spectral dynamics in a format well-suited to convolutional architectures: the mel scale itself is grounded in human pitch-perception experiments [17], and Davis and Mermelstein [18] showed empirically that mel-frequency cepstral features capture perceptually relevant aspects of the short-term spectrum better than linear-frequency alternatives, and they are empirically effective in audio classification tasks [19,20]. To improve generalisation across recording conditions, we adopt SpecAugment, which applies stochastic time/frequency masking directly in the spectrogram domain and is known to reduce overfitting in speech/audio models [21].

2.4. Deep Learning Models for Keystroke Recognition

CNN-based classifiers remain a strong baseline for keystroke recognition from spectrogram inputs due to locality and translational invariance in the time–frequency plane [5,7], a property of convolutional architectures first formalised for image recognition by LeCun et al. [22] and subsequently shown to be highly effective on mel-spectrogram inputs for speech and general audio [19,20]. For optimisation, Adam is commonly used for stable convergence on noisy gradients and large datasets [23]. While alternative architectures (e.g., recurrent or attention-based models) can improve robustness in some settings [6] and transfer from large-scale pretrained audio models offers a complementary direction [24], our focus is on a practical and reproducible pipeline where the main novelty is scalable dataset creation rather than architectural complexity. In particular, Hama Rawf [25] reports that a convolution–attention (CoAtNet) backbone combined with data augmentation improves robustness across six keyboard and platform conditions, illustrating the accuracy and robustness gains available from heavier architectures; we nonetheless retain a compact CNN because our contribution is the labelling pipeline rather than the backbone.

2.5. Positioning Our Contribution

Table 2 positions our work against representative prior studies. In contrast to work that assumes labelled keystroke audio collection per target device/user [1,5,8], work that emphasises dictionary or end-to-end transcription under different assumptions [3,6], and purely video-based keystroke-inference approaches such as *ClearShot* [26], which reconstruct text from visual keyboard observation alone without exploiting the acoustic channel, we contribute an automated OCR-aligned labelling pipeline that can scale dataset generation from screen-recorded videos with audio. Our use of video is therefore fundamentally different: the OCR output serves only as ground-truth labels for a subsequent acoustic classifier, rather than as the primary inference signal. This enables broader evaluation across keyboards, typists, and conferencing channels, and supports reproducible experimentation at a scale that is difficult to achieve via manual labelling. We deliberately retain a convolutional backbone rather than adopting a Transformer-based alternative [9]: our principal contribution is the data pipeline, and a compact CNN operating on mel-spectrograms is

a well-understood, strong baseline for keystroke-sound recognition [5,7,19,20] that keeps the scientific focus on the value of automatically labelled data rather than on architectural novelty, while remaining compatible with the modest hardware used throughout this study.

Table 2. Comparison of representative acoustic keyboard side-channel works and their assumptions.

Work	Recording Channel	Labels Required?	Core Method	Primary Target	Key Limitation/Note
Asonov & Agrawal (2004) [1]	Near-field mic (lab)	Yes	NN + spectral features	Key classification	Requires labelled data per keyboard/user
Zhuang et al. (2005) [2]	Near-field mic	No (bootstraps)	Clustering + language constraints	Text/password recovery	Language dependence; assumptions on typed text statistics
Berger et al. (2006) [3]	Near-field mic	Typically yes (classifier)	Dictionary attack + acoustic classes	Password guessing	Search-space/model assumptions; attacker priors matter
Compagno et al. (2017) [4]	VoIP/Skype-like channel	Yes (training)	Acoustic eavesdropping over VoIP	Text/password inference	Channel effects; practicality depends on call/audio conditions
Slater et al. (2019) [6]	Near-field (real tasks)	Yes (dataset)	End-to-end detection + transcription	Free-text transcription	Complex pipeline; robustness emphasis; still needs labelled data
Bai et al. (2021) [7]	Near-field mic	Yes	Robust features + ML/DL	Key classification	Generalisation remains challenging across keyboards/users
Harrison et al. (2023) [5]	Near-field + Zoom	Yes	CNN on spectrograms	Key classification	Label acquisition remains the bottleneck
Wang et al. (2023) [8]	Video-conferencing	Yes	DL-based eavesdropping	Key classification	Assumes labelled data and stable conferencing conditions
This work	Screen video + audio (realistic recordings)	No manual labels	OCR-aligned auto-labelling + CNN	Key classification (scalable)	Scales dataset creation across settings; focuses on reproducibility

2.6. Automatic Labelling and Scaling the Attack

Most existing works assume that the attacker can capture keystroke sounds along with the corresponding ground-truth text [1,5,7,8]. In practice, such labelled datasets are difficult to obtain because typed passwords or sensitive input are typically masked, and manual annotation of long typing sessions is prohibitively expensive. Our contribution addresses this gap by exploiting video streams of the victim’s display: the attack assumes that the adversary can record both the screen (for example, during an online lecture where code is typed) and the associated audio. By employing OCR on video frames, we extract the sequence of typed characters and align them with detected keystroke sounds, producing a labelled keystroke-audio dataset without any manual transcription. The same pipeline scales trivially to additional keyboards, typists, and languages: each new recording session can be processed end-to-end in minutes, whereas manual annotation of a comparable amount of audio takes hours per session. Importantly, the video stream is used only as a labelling signal during training-data acquisition; at deployment time, the attacker needs only the acoustic channel.

3. Materials and Methods

3.1. Feature Extraction with Mel-Spectrograms and SpecAugment

The raw waveform of a key press consists of a short click with two energy peaks corresponding to pressing and releasing the key. As illustrated in Figure 2, raw audio is not a suitable input for neural networks because it is high dimensional and sensitive to misalignment. Instead, we compute the short-time Fourier transform and convert the frequency axis to the perceptual mel scale. A mel-spectrogram better represents human auditory perception and emphasises frequencies critical to hearing; it also produces a visually recognisable two-dimensional representation that is well suited for convolutional neural networks. To further improve generalisation, we apply SpecAugment, which randomly masks 10% of the time and frequency bins of the spectrogram to simulate time warping and frequency shifts.

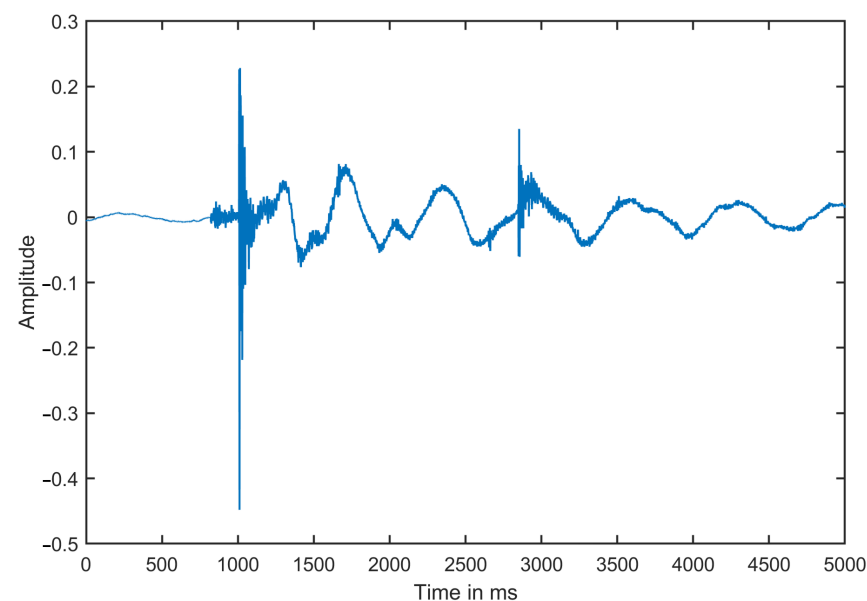


Figure 2. Waveform of a single key press. The raw audio signal contains two peaks for pressing and releasing the key, but the high-dimensional waveform is not directly suitable for CNN input.

3.2. Convolutional Neural Networks for Keystroke Classification

A convolutional neural network (CNN) [22] is employed to classify mel-spectrogram representations of audio signals. CNNs are well suited for this task because they exploit local connectivity and weight sharing, enabling the learning of discriminative time–frequency patterns while providing robustness to small shifts along the frequency axis [19,20]. Such properties are particularly important for audio analysis, where minor spectral variations naturally arise due to environmental conditions, recording devices, and variability in keystroke sounds across different users and typing conditions.

The proposed architecture, illustrated in Figure 3, follows a hierarchical feature-extraction design that progressively captures higher-level representations from the input mel-spectrogram.

Feature extraction is performed using two convolutional blocks specifically designed for mel-spectrogram analysis. In the first block, three convolutional layers with 32 filters are applied. A standard 3×3 convolution is used to capture joint time–frequency patterns, followed by 3×1 and 1×3 convolutions, which decouple temporal and spectral feature learning. This asymmetric convolutional arrangement allows temporal dynamics (e.g., transient events) and spectral structures (e.g., harmonic content) to be modelled more effectively than using only square kernels, while also reducing parameter complexity.

All convolutional layers employ ReLU activation and same padding to preserve spatial resolution, and bias terms are omitted in the early layers to avoid redundancy.

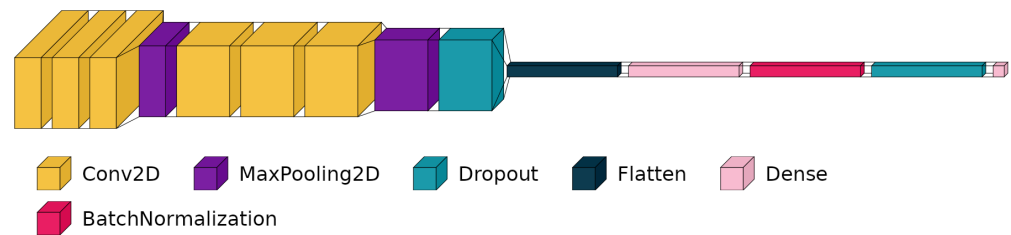


Figure 3. Visual representation of the convolutional neural network architecture used for keystroke classification.

A max-pooling layer with a pooling size of (2×1) is applied to reduce temporal resolution while maintaining frequency resolution, reflecting the fact that temporal invariance is more desirable than frequency invariance in mel-spectrogram representations. The second convolutional block follows the same architectural principle but increases the number of filters to 64, enabling the extraction of more abstract and discriminative features. A second max-pooling operation further compresses the temporal dimension and improves robustness to temporal variability.

To reduce overfitting, a dropout layer (rate 0.2) [27] is applied after the convolutional stages. The resulting feature maps are then flattened and passed to a fully connected layer with ReLU activation, where local time–frequency features are integrated into a global representation suitable for classification. Batch normalisation [28] is applied to stabilise training and improve convergence, followed by an additional dropout layer (rate 0.4) to reduce overfitting.

The network terminates with a softmax output layer, which produces a probability distribution over the target classes. Training is carried out using the Adam optimiser [23], and optimisation is performed by minimising the categorical cross-entropy loss, which is appropriate for multi-class classification tasks.

3.3. Threat Model and Scope

We consider an adversary who can observe both (i) the screen of the target machine and (ii) the corresponding audio signal during a typing session, as is the case for attendees of an online lecture, a screen-shared meeting or a publicly streamed tutorial. The adversary does not require physical proximity, a hidden microphone, or the ability to install software on the victim’s device; the audio is assumed to be captured via an ordinary laptop or smartphone microphone and is possibly subject to lossy compression and standard conferencing denoising, consistent with the setting of [4,5,8]. Ground-truth labels for training are obtained exclusively from characters that the victim voluntarily renders on the shared screen (terminal or editor output during coding tutorials); sensitive input that is masked on the screen—e.g., password fields—is outside the scope of the labelling pipeline, but once a classifier has been trained its operation is audio-only, so the attacker can subsequently recognise masked or entirely off-screen input from acoustic captures in the same recording environment.

For clarity, we distinguish two operational phases: (a) a training-data acquisition phase, in which the adversary needs both video and audio and uses OCR to derive labels; and (b) a deployment phase, in which the trained model is applied to fresh audio captures and no video is required. Our present study focuses on phase (a) and establishes the feasibility of scalable labelled data collection; the entropy analysis in the Password

Entropy Reduction subsection quantifies the consequences of phase (b) for password security. For the password-entropy analysis, we assume a comparable recording environment across the training-data-acquisition and deployment phases for a given target. This per-target assumption is distinct from the cross-victim, cross-keyboard transfer, which shows the pretrained feature extractor itself generalises across individuals and devices. Characters that require the Shift key—uppercase letters and shifted symbols—are recognised by the OCR stage and mapped back to the corresponding unshifted physical key using the victim’s known software keyboard layout, so they are retained in the training alphabet rather than excluded; the password-entropy analysis is computed with respect to the 94-symbol printable-ASCII alphabet.

Consistent with prior work in this area [1–3,5], the adversary’s goal is to recover typed content from the acoustic channel; our specific contribution addresses the scalable acquisition of labelled training data rather than the attack model itself.

3.4. Data Collection and Labelling

Audio was recorded with the open-source Audacity software, version 3.3.3, at a sampling rate of 44.1 kHz and exported as 32-bit PCM wave files. For each lecture, we captured a synchronised screen capture and audio file. Prior to click detection, each signal is normalised to have unit root-mean-square amplitude to reduce variance in recording volume. Following the method described in *Don’t Skype & Type!* [4], we compute the short-time Fourier transform (STFT) over 10 ms windows and sum the magnitude spectra to obtain an energy envelope. Formally, let $X[n, f]$ denote the STFT magnitude at frame n and frequency bin f ; the per-frame acoustic energy envelope is

$$e[n] = \sum_{f=1}^F |X[n, f]|, \quad (1)$$

where F is the number of frequency bins. A keystroke press event is detected at frame n whenever the envelope exceeds an adaptive threshold,

$$e[n] > \tau, \quad \tau = \mu_e + \beta \sigma_e, \quad (2)$$

where μ_e and σ_e are the mean and standard deviation of the envelope $e[n]$ over the recording and β is a sensitivity coefficient (set to $\beta = 3$ in our experiments) that trades off missed detections against false detections. Once an onset is detected, we extract the subsequent 250 ms of audio as the waveform of the keystroke. When keystrokes are closely spaced, the extracted window is shortened to avoid overlap. Each cropped keystroke is stored in an individual file, resulting in two distinct datasets recorded from two different typists on two different keyboards. The first dataset (pretraining) consists of 125 samples per key collected from Typist A on a Keychron Q1 Pro (QMK/VIA, US ANSI layout) and is used for model training (pretraining of the generalised base model). The second (target) dataset consists of 25 samples per key collected from a different individual, Typist B, on a Cooler Master MK730 (GKCL1-DE, DE ISO layout) in an independent session and is used to evaluate transfer-learning performance under limited-data conditions. Transfer-learning experiments draw subsets of $\{13, 20, 25\}$ samples per key from this second dataset. This makes the transfer evaluation a cross-victim and cross-device test rather than a same-session re-split. In parallel, screen recordings are sampled at 30 frames per second and processed with the Tesseract OCR engine [29] to extract the characters that appear in terminal or editor windows. By aligning the click timestamps with the appearance of characters on screen, we assign a ground-truth label to each audio segment. The pipeline automatically flags and logs low-confidence OCR outputs for optional manual correction.

3.5. Feature Extraction and Data Augmentation

Each labelled audio segment is transformed into a mel-spectrogram [17,18] using a Hann window of length 1024 samples (≈ 23 ms) and a hop length of 225 samples (5 ms) as suggested by recent practical attacks [5]. We employ 64 mel bands covering the 200–8000 Hz range, which balances resolution with the relatively short duration of keystroke sounds. For comparison, we experimented with the 128-band, 25 ms window commonly used in speech recognition and found that reducing the number of bands had little impact on classification accuracy while lowering memory requirements. The resulting 64×50 spectrograms are then converted to decibel scale using `librosa.power_to_db` [30]. To encourage the model to generalise, we apply data augmentation: (1) time shifting each waveform by up to 2 ms to simulate variations in typing cadence; and (2) masking 10% of both the time and frequency bins of the mel-spectrogram (SpecAugment) [21]. In our implementation, the masked values are replaced with the global mean of the spectrogram, which forces the network to rely on context rather than isolated frequency peaks.

3.6. Model Training

The dataset is partitioned into 80% training, 10% validation, and 10% test splits, with the test set kept completely unseen during model development. Class labels are encoded using a label encoder followed by one-hot encoding. Model performance during training is monitored on the validation set to guide model selection and prevent overfitting.

To obtain robust performance estimates under limited-data conditions, stratified five-fold cross-validation is applied to the combined training and validation data. For each fold, a new model instance is trained and evaluated on the corresponding validation split, ensuring balanced class distributions across folds. Training is conducted for up to 50 epochs with a batch size of 32, using the Adam optimiser with a learning rate of 10^{-3} . The model achieving the lowest validation loss within each fold is retained.

Performance is evaluated using top- k accuracy ($k = 1\text{--}5$) and macro-averaged F1-score, complemented by entropy-based metrics to assess prediction confidence and information concentration. Top- k accuracy counts a prediction as correct whenever the ground-truth key appears among the model's k highest-probability outputs, whereas macro-F1 averages per-class F1-scores with equal weight and is therefore insensitive to class frequency imbalance. Entropy-based metrics are computed from the softmax output distribution and summarise how sharply the classifier commits to a single key. Cross-validation results are reported as mean $\pm$ standard deviation across folds, computed over the five stratified splits described above.

Following cross-validation, a final model is trained on the full training and validation set and evaluated on the held-out test set, providing an unbiased estimate of generalisation performance on unseen keystroke recordings.

Training was performed on a MacBook Pro equipped with an Apple M2 processor and 8 GB of RAM. No discrete GPU was used; all experiments were executed using the default TensorFlow backend without explicit GPU or Neural Engine acceleration. Each training run of 50 epochs required less than 2 min, resulting in a total training time of about 8 min when including five-fold stratified cross-validation and final model retraining. Training proceeded without early stopping and exhibited stable convergence. Final evaluation was conducted on a held-out test set that was not used during training or validation.

For reproducibility, the full software stack consisted of Python 3.10, TensorFlow 2.15, NumPy 1.26, librosa 0.10 and scikit-learn 1.3. A fixed random seed (42) was used for dataset shuffling, stratified k -fold splitting, and network weight initialisation; the same seed was applied to the NumPy and TensorFlow random number generators. No learning-rate scheduler was used—the Adam learning rate was held constant at 10^{-3} throughout each

training run. The initial train/validation/test partition is stratified by class in addition to the five-fold cross-validation, so that every key is represented proportionally in each split. A complete `requirements.txt`, the `environment.yml`, the hyperparameter grid-search CSV, the random seed and scripts to regenerate every reported number are included in the accompanying code repository.

The high classification accuracy observed at this stage (exceeding 99%) is largely a consequence of training and evaluation on a relatively large and homogeneous dataset, with five-fold cross-validation ensuring a stable and reliable performance estimate. It should be emphasised that the primary objective of this training phase was not to demonstrate ultimate generalisation performance, but rather to obtain a well-generalised base model. This model is intended to capture robust and transferable feature representations that can subsequently be reused in transfer-learning scenarios, where only a limited number of task-specific samples are available. Consequently, the reported results should be interpreted in the context of model pretraining for transferability rather than as a definitive measure of real-world performance.

3.7. Transfer Learning

Transfer learning [24,31] adapts a CNN pretrained on one typist and keyboard to a target dataset recorded from a different typist on a different keyboard. The objective is to test whether the spectro-temporal features learned by the frozen backbone transfer across individuals and devices, so that a new target is handled by re-fitting only a lightweight classification head from a small number of samples.

The pretrained network is loaded and used as a fixed feature extractor by freezing all convolutional layers [31]. The original classification layer is removed, and a new task-specific classification head is attached to the output of the penultimate layer. This head consists of a fully connected layer with ReLU activation, followed by dropout for regularisation and a softmax output layer corresponding to the number of target classes. During adaptation, only the parameters of this newly added head are updated, while the pretrained convolutional backbone remains unchanged.

Preserving the initial convolutional layers is motivated by their role in learning generic spectro-temporal features of keystroke sounds, such as transient onsets, local frequency patterns, and time–frequency correlations present in mel-spectrogram representations. These features are largely invariant across users, keyboards, and recording conditions and therefore transfer effectively to new scenarios [24,31]. By freezing these layers, the model retains its previously learned generalisation capability, while the classification head adapts to user- and session-specific characteristics.

This design directly probes generalisation: the backbone never sees the target typist or keyboard, so any accuracy on the target is attributable to transferable features rather than memorised session characteristics. Fitting only the compact head both limits overfitting under scarce data and confines victim- and device-specificity to a small, cheaply retrained component.

3.8. Hyperparameter Search and Model Selection

To ensure that the network architecture and optimisation settings were well suited to the problem, a grid search over multiple hyperparameter combinations was conducted. The search space included the number of convolutional filters in the two convolutional blocks $(f_1, f_2) \in \{(32, 64), (64, 64), (64, 128)\}$, the number of units in the fully connected layer $d \in \{128, 256\}$, the dropout rate after the dense layer $\in \{0.3, 0.4, 0.5\}$, and the learning rate of the Adam optimiser $\in \{10^{-3}, 3 \times 10^{-4}, 10^{-4}\}$. The dropout rate after the convolutional blocks was kept fixed (0.2) to maintain consistent regularisation in the early layers.

Each candidate configuration was trained for up to 50 epochs using a stratified train/validation/test split, and model selection was performed based on validation macro-F1-score, which provides equal weighting across all classes. This criterion was chosen to avoid bias toward frequently occurring keys and to ensure balanced classification performance.

Table 3 summarises the hyperparameters explored and their meanings. Table 4 lists the five best configurations ranked by validation macro-F1.

Table 3. Hyperparameters explored in the grid search. Each parameter controls the model capacity or regularisation.

Parameter	Meaning
Filters (f_1, f_2)	Kernels in successive convolutional blocks controlling capacity
Dense units d	Size of the fully connected layer before the softmax output
Dropout (conv.)	Dropout rate after convolutional blocks (fixed at 0.2)
Dropout (dense)	Dropout rate after the dense layer
Learning rate	Initial step size of the Adam optimiser

Table 4. Top hyperparameter configurations (sorted by validation macro-F1) for a grid search over 54 combinations of filters, dense units, dropout rates and learning rates. LR denotes learning rate.

Config	Filters	Dense Units	Dropout (Dense)	LR	Val. Macro-F1	Test Macro-F1	Test Top-3 Acc.
13	(32, 64)	256	0.40	10^{-3}	0.9914	0.9656	0.9957
49	(64, 128)	256	0.50	10^{-3}	0.9908	0.9785	0.9957
52	(64, 128)	128	0.50	10^{-3}	0.9904	0.9771	0.9914
1	(32, 64)	128	0.30	10^{-3}	0.9874	0.9744	0.9742
16	(32, 64)	256	0.50	10^{-3}	0.9871	0.9699	0.9957

Among these, configuration 13 with filters (32, 64), 256 dense units, a dense dropout of 0.4 and a learning rate of 10^{-3} achieved the highest validation macro-F1 (0.9914) and tied for the best top-3 test accuracy (0.9957). Configuration 49 achieved the highest test macro-F1 (0.9785) with the same top-3 test accuracy, but at the cost of a substantially larger parameter count. Balancing validation performance and test generalisation, we adopt configuration 13 for all subsequent experiments. It is worth noting that the choice of convolutional filter sizes has a substantial impact on model complexity. Configurations using (32, 64) filters result in approximately 13.2 million trainable parameters, whereas configurations with (64, 128) filters increase the number of trainable parameters to approximately 26.4 million. This significant increase in model size strengthens the case for selecting a simpler architecture, as it offers a more favourable balance between representational capacity, computational efficiency, and robustness, particularly in scenarios with limited training data and when transfer learning is employed.

3.9. Evaluation Metrics

To evaluate classification performance, we report several complementary metrics. Accuracy measures the proportion of correct predictions and is defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP , TN , FP and FN denote true positives, true negatives, false positives and false negatives. Precision (Pr) and Recall (Rcl) quantify how well the model avoids false positives and false negatives, respectively, and are given by $\text{Pr} = TP/(TP + FP)$ and $\text{Rcl} = TP/(TP + FN)$. The F1-score combines precision and recall as their harmonic mean,

$$F1 = \frac{2 \text{Pr Rcl}}{\text{Pr} + \text{Rcl}}.$$

We average the F1-score over classes (macro-F1), which assigns equal weight to each key regardless of its sample count and is therefore insensitive to any class imbalance. This metric provides a robust measure of per-class performance and is used to select the best hyperparameters. We also report top- k accuracy, defined as the fraction of samples for which the true class appears in the model's top- k predictions. In our experiments, we focus on top-3 accuracy ($k = 3$), reflecting an attacker who can try three candidate keys per keystroke.

3.10. Use of Generative AI Tools

In accordance with the journal's policy on generative artificial intelligence (GenAI), we disclose that a GenAI tool (OpenAI GPT-4o) was used exclusively to improve the English language and readability of author-written text. GenAI was not used for the study design, data collection, generation or labelling of the datasets, implementation or training of the models, production of figures, or the analysis and interpretation of the results. All AI-assisted passages were subsequently reviewed, verified, and edited by the authors, who take full responsibility for the content of this article. Full product details of the tool are provided in the Acknowledgments.

4. Results

4.1. Audio Signal and Feature Extraction

The feature-extraction procedure itself is described in Section 3.5; the figures below illustrate the result of applying that procedure to real keystroke recordings in our dataset.

Figure 2 shows a typical waveform of the letter “a”. The signal exhibits two peaks corresponding to the press and release of the key, and the noise floor is relatively low. Figure 4 displays the mel-spectrogram of the same press, highlighting energy concentrations at specific frequencies. To improve generalisation, Figure 5 illustrates a masked mel-spectrogram after applying SpecAugment, where random time and frequency bands are zeroed out.

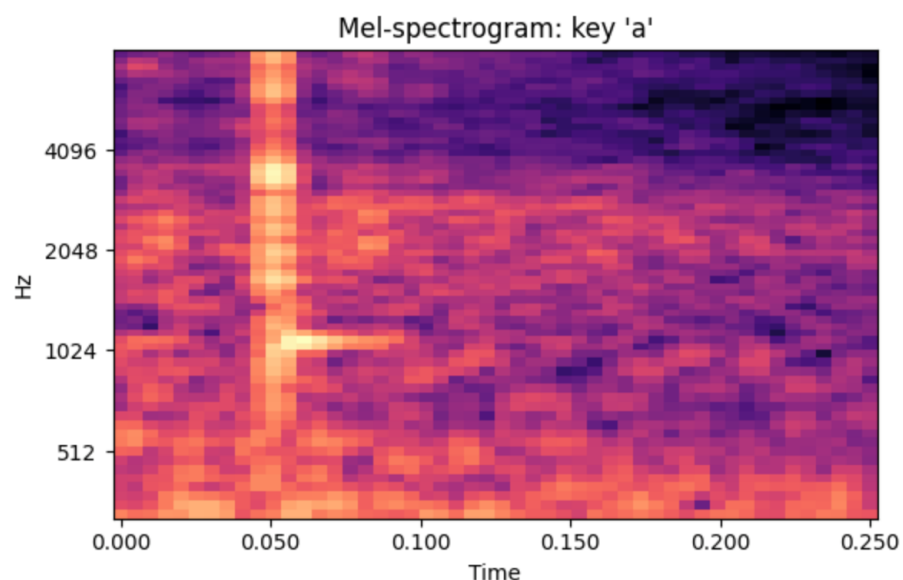


Figure 4. Mel-spectrogram of the same key press. The frequency axis is mapped to the mel scale, yielding a perceptually meaningful representation suitable for convolutional processing.

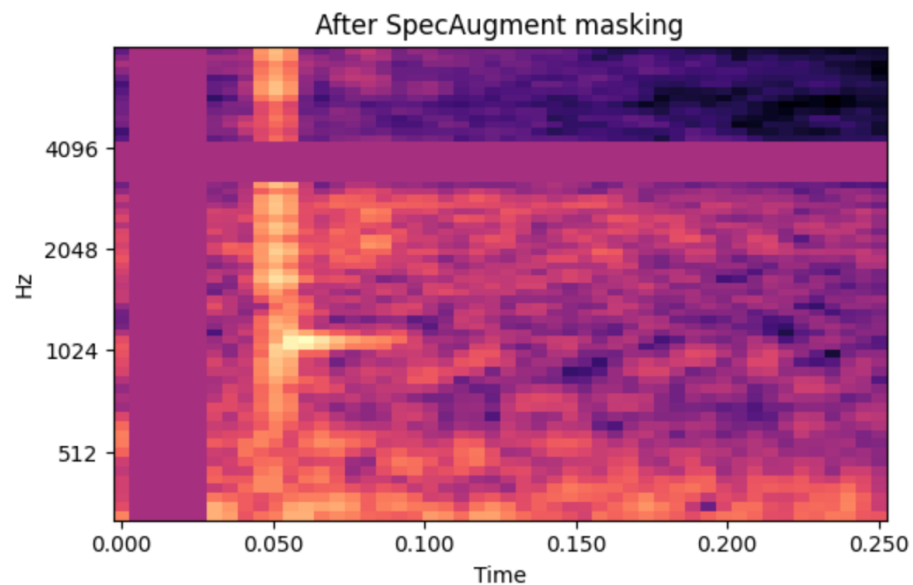


Figure 5. Masked mel-spectrogram produced by SpecAugment. Random time and frequency bands are masked to encourage the model to learn robust features.

4.2. Training

The training and validation loss curves obtained during the training of the main generalised model are shown in Figure 6. A rapid decrease in training loss is observed within the first few epochs, followed by stable convergence towards near-zero values. The validation loss exhibits a similar trend, decreasing steadily and closely tracking the training loss throughout the training process. The absence of a significant divergence between the two curves indicates stable optimisation and limited overfitting. These results confirm that the model learns robust and transferable feature representations, which are later reused in the transfer-learning stage.

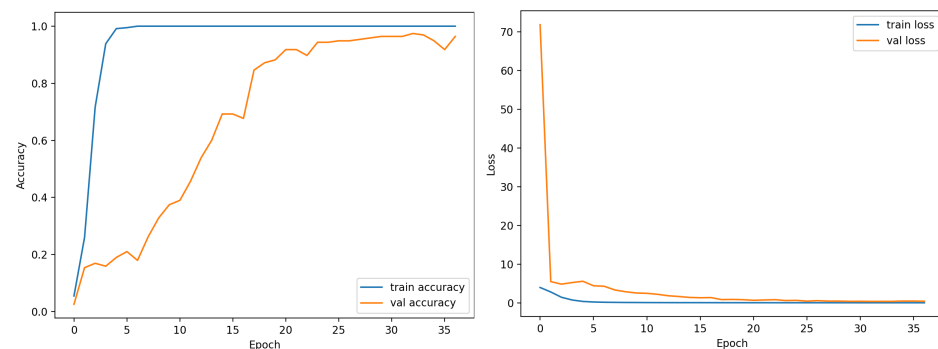


Figure 6. (left) Training and validation accuracy, (right) Training and validation cross-entropy loss for the base model—Typist A, keyboard A.

The training and validation accuracy and loss curves for the transfer-learning stage are shown in Figure 7. In this experiment, the model is fine-tuned using a substantially reduced dataset consisting of only 20 samples per key. As training progresses, both training and validation losses decrease steadily, indicating stable convergence despite the limited amount of data. Notably, the validation accuracy consistently exceeds the training accuracy. This behaviour is expected, as regularisation mechanisms such as dropout are active during training but disabled during validation, causing the model to operate under more challenging conditions when learning. During validation, the full network capacity is utilised, resulting in higher measured accuracy. This effect is further amplified by the

use of a pretrained feature extractor, which provides strong prior representations that generalise well even when fine-tuned on a small, session-specific dataset. Overall, the curves demonstrate effective transfer learning without signs of instability or overfitting.

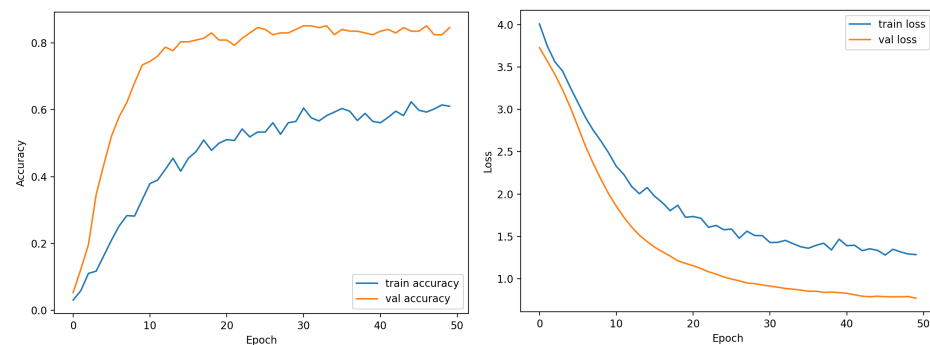


Figure 7. (left) Training and validation accuracy, (right) Training and validation cross-entropy loss for the transfer-learning stage—Typist B, keyboard B. The steady decline in loss mirrors the increase in accuracy and illustrates stable optimisation.

Transfer learning was performed by reusing a pretrained convolutional feature extractor while retraining only a compact classification head on the target dataset. The resulting model comprises approximately 13.2 million parameters, of which 41,668 parameters are trainable, while the remaining parameters are frozen. This corresponds to less than 1% of the total model parameters, substantially reducing the risk of overfitting when adapting the model to a limited dataset. The transfer-learning stage was conducted for 50 epochs using the Adam optimiser. Owing to the small number of trainable parameters, convergence was achieved rapidly. The adapted model demonstrates strong performance on the held-out test set, confirming that the pretrained feature representations are well suited for efficient adaptation to user- and session-specific data. Retraining converged rapidly and was completed in under one minute on the same hardware, reflecting the small number of trainable parameters and the efficiency of the proposed approach.

4.3. Evaluation

To establish an upper-bound performance reference, the classifier was trained from scratch using a large and homogeneous dataset consisting of 125 samples per key, without applying transfer learning. This setting represents an idealised scenario in which sufficient labelled data are available for each class.

The trained model achieves very high classification performance across all top- k metrics. A top-1 accuracy of 98.1% is obtained, indicating near-perfect single-guess classification. When allowing multiple hypotheses, performance rapidly saturates, reaching 99.4% top-2 accuracy and 100% top-3, top-4, and top-5 accuracy, demonstrating that the correct key is almost always ranked among the top predictions.

In addition to accuracy, entropy-based metrics further confirm the model's strong discriminative capability. The mean entropy at top-1 is 0 bits, corresponding to maximum confidence in the predicted class, which translates to an entropy reduction of 5.285 bits relative to a uniform prior over the evaluation alphabet (i.e., the bits of information the classifier contributes beyond a random guess). As k increases, entropy rises gradually, while entropy reduction remains substantial, exceeding 4.96 bits even at top-5. This indicates that uncertainty remains tightly constrained even when multiple candidate keys are considered.

Overall, these results show that training without transfer learning can achieve near-perfect performance when a sufficiently large number of samples per key is available. However, this data requirement is substantial, motivating the transfer-learning approach

explored in subsequent sections, which aims to retain high accuracy under significantly reduced training data availability.

While the results obtained with a large training dataset provide an important performance reference, such data availability is often unrealistic in practical attack scenarios. In real-world settings, especially for targeted or user-specific attacks, only a limited number of keystroke samples can typically be collected from a single user and a single recording session. This constraint motivates the evaluation of the proposed approach under severely reduced training data conditions, where the ability to adapt a pretrained model using a small number of samples becomes a critical factor.

The impact of the number of training samples per key on classification performance is illustrated in Figure 8, which reports top- k accuracy on the held-out test set for varying dataset sizes. For all experiments, transfer learning was performed using the same pretrained base model, while only the number of samples used for fine-tuning was progressively increased. Importantly, the test set was kept identical across all configurations, ensuring a fair and consistent evaluation of the effect of training data size.

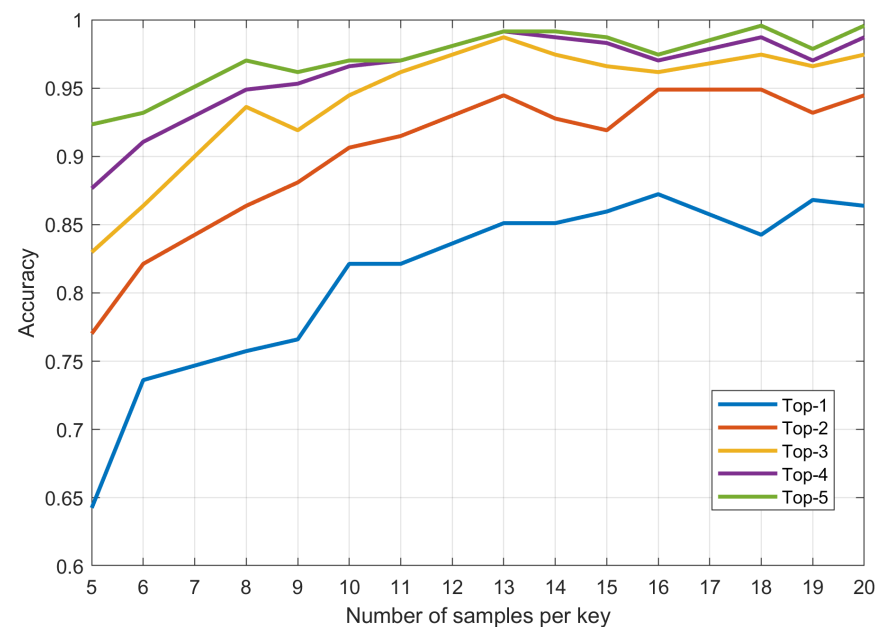


Figure 8. Top- k test accuracy versus number of training samples per key obtained by fine-tuning the same pretrained model using transfer learning.

As expected, classification accuracy improves as additional samples per key are introduced, reflecting improved adaptation of the model to the target user and session characteristics. However, the performance gain exhibits early saturation. With only 13 samples per key, the model already achieves approximately 85% for top-1, 95% top-3, and 99% top-5 accuracy on the test set, demonstrating strong generalisation even under severe data constraints.

Further increases in the number of training samples lead to only marginal improvements, particularly for higher top- k metrics. This behaviour indicates that the pretrained feature extractor captures highly discriminative and transferable representations, allowing effective fine-tuning with a small amount of user-specific data. Overall, these results emphasise the key advantage of the proposed approach: reliable keystroke classification can be achieved with minimal retraining effort, making the method practical for real-world deployment scenarios where collecting large labelled datasets per user is not feasible.

Operational interpretation of the transfer-learning accuracy. The top-1 accuracy of $\approx 85\%$ obtained with only 13 samples per key (Figure 8) should be read in the context of

the intended attack rather than as a stand-alone recognition score. First, a realistic acoustic adversary is not restricted to a single guess per keystroke; the operationally relevant quantity is top- k accuracy for small k , and at the same 13-sample point the model already reaches $\approx 95\%$ top-3 and $\approx 99\%$ top-5. As the Password Entropy Reduction analysis below shows, even $k = 3$ collapses the per-character search space to $\log_2 3 \approx 1.58$ bits, so an 85% top-1/95% top-3 classifier is already highly damaging. Second, typed content is highly redundant: source code and natural language carry strong n -gram and syntactic structure, so most of the residual $\sim 15\%$ first-guess errors are recoverable by a downstream language-model or dictionary step, exactly as exploited in [2,3,9]. Third, the 13-sample setting is deliberately the most data-starved configuration; raising the budget to 20–25 samples per key—still only a few minutes of a single lecture—increases top-1 accuracy further (Figure 8). We therefore regard $\approx 85\%$ top-1 under 13-shot adaptation as sufficient for the intended low-data, single-victim scenario, and we do not claim the 98% top-1 of the large-data upper bound for this few-shot regime.

To further evaluate the effectiveness of transfer learning, additional comparison experiments were conducted by training the proposed CNN from scratch using the same number of training samples per key as in the corresponding transfer-learning experiments. Figure 9 presents the obtained results. It can be observed that transfer learning consistently outperforms training from scratch when only a limited number of user-specific training samples is available. This improvement is achieved because the pretrained network already contains generalised acoustic feature representations learned from a larger source dataset. During transfer learning, these pretrained feature-extraction layers remain frozen, while only the newly added classification head is optimised for the target user. Consequently, the model requires substantially fewer training samples to achieve high classification accuracy, while simultaneously reducing the risk of overfitting. In contrast, training from scratch must learn both low-level acoustic features and the final classifier solely from the limited target-user data, making the learning process considerably less data-efficient. These results demonstrate that transfer learning is particularly suitable for realistic attack scenarios where only a small amount of labelled target-user data can be collected.

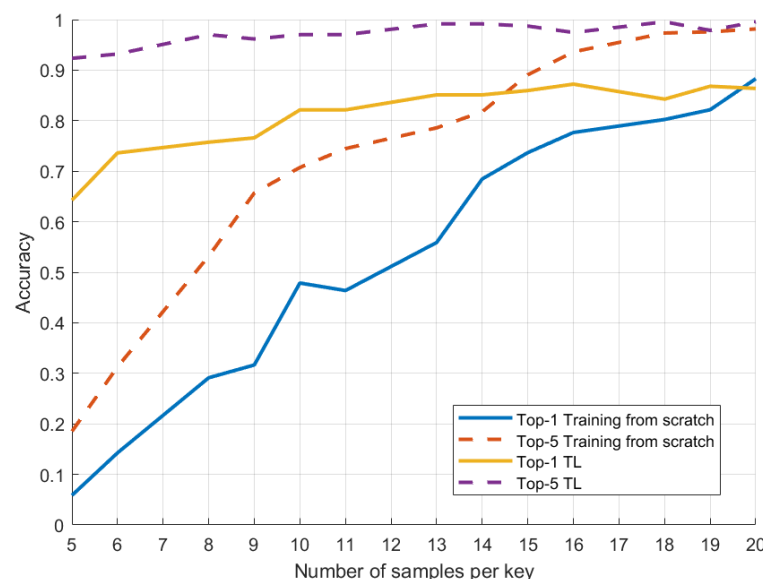


Figure 9. Top-5 test accuracy as a function of the number of training samples per key in the target session, comparing the proposed transfer-learning approach (pretrained convolutional backbone frozen, only the classification head retrained) with the same CNN architecture trained from scratch on the target data alone. Under identical training conditions, transfer learning consistently outperforms training from scratch in the limited-data regime, with the largest margin at the smallest sample counts.

4.4. Comparison with a State-of-the-Art Deep-Learning Attack

To complement the qualitative positioning in Table 2, we report a quantitative comparison against a representative state-of-the-art deep-learning acoustic side-channel attack. We take the CoAtNet-based attack of Harrison et al. [5] as the comparison method: it shares our threat model (per-keystroke classification from click acoustics on a mel-spectrogram front-end) and reports some of the strongest published recognition accuracies. Because no trained checkpoint is publicly released, we retrained a faithful CoAtNet-1 implementation on our dataset and evaluated it under the same train/test methodology used for our own model (identical file-level stratified split, seed 42), with each method retaining its own native feature front-end (CoAtNet: 64×64 log-mel, hop 300, 100–2500 Hz; ours: 64×50 mel, hop 225, 200–8000 Hz). For our own network, we reuse the base-model results already reported in Section 4.3 rather than retraining it for this comparison. The two methods are contrasted in Table 5.

Table 5. Quantitative comparison against a retrained CoAtNet-1 baseline. The figures for our network are the base-model results reported in Section 4.3; only CoAtNet-1 was retrained on our dataset and under the same train/test methodology. Best results are marked in bold.

Method	Top-1	Top-3	Macro-F1	Params
CoAtNet-1 (Harrison) [5]	96.41	98.97	96.45	24,041,755
Our CNN (base model)	98.1	100.0	96.56	13,170,191

In this comparison, our compact CNN attains higher top-1 accuracy (98.1% vs. 96.41%, a +1.7 percentage-point margin) and perfect top-3 (100% vs. 98.97%) at a comparable macro-F1, while using roughly half the parameters (13.2 M vs. 24.0 M) and correspondingly lower training and inference costs. This supports our design choice of a compact CNN over a heavier convolution–attention architecture for this task: in the limited-data regime that characterises the attack, the smaller model matches or exceeds the transformer-hybrid at a fraction of the cost. Data efficiency is quantified separately: transfer learning retains $\approx 85\%$ top-1 with as few as 13 samples per key.

4.5. Mechanistic Analysis of Feature Discriminability

Under matched (within-session) conditions, the classifier separates keys essentially perfectly, so raw accuracy alone reveals little about which keys the model finds hard. We therefore characterise discriminability through the model’s internal structure using two complementary analyses: the nearest competitor (the second-highest softmax class per keystroke) and the pairwise distances between class centroids in the penultimate 256-dimensional embedding space. Both agree on a clear, physically interpretable pattern: confusability is dominated by physically adjacent keys. The closest pairs in embedding space are adjacent digits (5/6, 3/4, 2/3, 6/7; centroid distances 15.4–16.5), adjacent home-row letters (d/f, s/d) and adjacent top-row letters (e/r, t/r), and the same neighbours (x/y, b/n, o/p, j/k, c/v) dominate the runner-up distribution (Figure 10). This reflects the physical origin of the signal: neighbouring keys share the most similar strike position and mechanical response and thus emit the most similar acoustic signatures.

The pattern is consistent with the amplitude-invariant effect sizes in Table 1: the i/1 pair (large effect sizes, $|d| \geq 1.5$) is among the least confusable keys, whereas the comma/period pair (weak separators, $|d| \leq 0.6$) is among the closest pairs in the learned feature space. Discriminability therefore scales with acoustic effect size, and the residual similarity the network encodes concentrates on precisely the pairs Table 1 identifies as acoustically subtle; these are resolved at decision time under matched conditions and surface as error only under harder settings, which explains the differences observed

across experimental settings. The drivers of the transfer-learning data efficiency reported above follow the same logic: the convolutional backbone learns generic spectro-temporal primitives—transient press/release onsets, local time–frequency energy patterns, and harmonic structure—that are physically determined by switch mechanics and largely invariant across typists, keyboards, and sessions [24,31], so session- and device-specific variation is confined to a compact classification head comprising under 1% of the parameters, for which a handful of samples per key suffices while the expensive feature representation is reused rather than relearned.

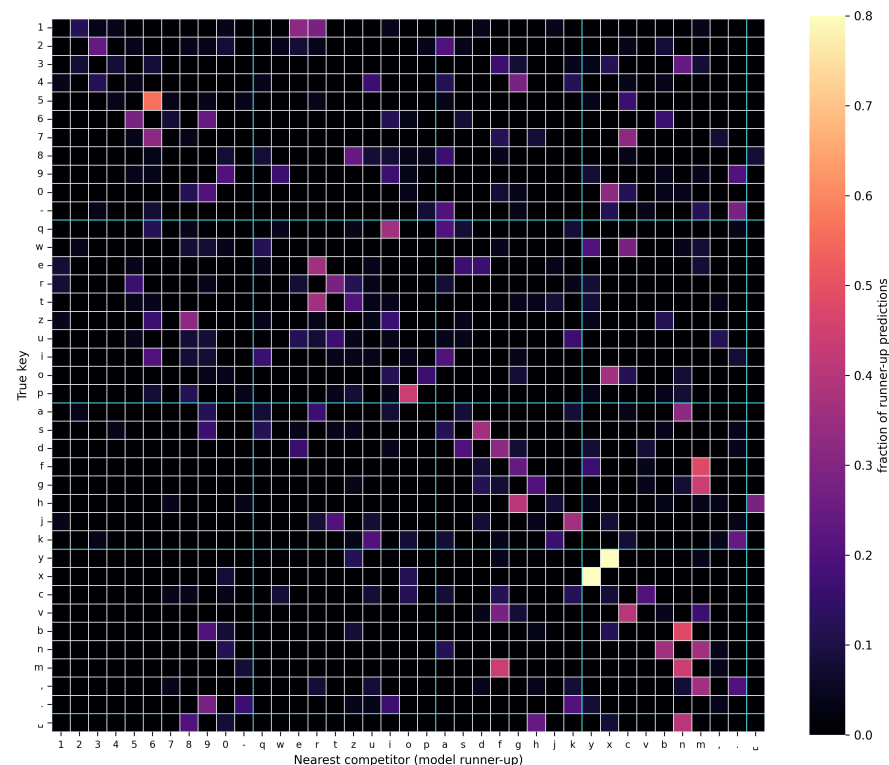


Figure 10. Nearest-competitor structure of the CNN, with keys ordered by physical keyboard position (lines separate the number, top, home, and bottom rows). For each true key (row), the cell shows the fraction of test keystrokes for which each other key (column) was the model's second-highest-probability prediction (top-1 accuracy is $\approx 100\%$, so the diagonal is zero by construction). Confusability concentrates along each row's local diagonal—i.e., on physically adjacent keys (e.g., the home-row band s–d–f–g–h–j–k, the number-row cluster 5–6–7, and the bottom-row pair x/y)—consistent with adjacent keys producing the most similar acoustic signatures and with the amplitude-invariant effect sizes in Table 1.

4.6. Password Entropy Reduction

The effectiveness of the attack can be quantified by analysing the reduction in password entropy. A random password of length L drawn uniformly from an alphabet of size A has Shannon entropy [32]

$$H(L) = L \log_2 A, \quad (3)$$

where L is the password length in characters and A is the number of distinct symbols in the alphabet. For an eight-character password with $A = 94$ printable ASCII characters, $H = 8 \log_2 94 \approx 52.44$ bits.

Suppose the model ranks the correct key among the top- k predictions with probability p , and an adversary tries at most k candidates per keystroke. Following the entropy-based adversarial model of Zhuang et al. [2], the effective per-password entropy becomes

$$H'(L) = L[p \log_2 k + (1 - p) \log_2 (A - k)], \quad (4)$$

where $p \in [0, 1]$ is the top- k classification accuracy, k is the number of candidate keys the adversary tests per keystroke, and A and L are as defined in Equation (3). The first bracketed term accounts for keystrokes whose true key falls within the top- k set (search reduced to k options), and the second for the residual uncertainty when the true key is missed. As a conservative reference point, setting $p = 0.95$ and $k = 5$ yields $H'(8) \approx 21.9$ bits, corresponding to a reduction of nearly 60%.

Our empirical evaluation shows that the classifier achieves **98.1% top-1 accuracy, 99.4% top-2 accuracy, and 100% top-3 accuracy** on a held-out test set. This implies that the correct key is always contained within the top three predictions. Under this setting, the attacker needs to try at most $k = 3$ candidates per keystroke with probability $p \approx 1$.

Substituting $p = 1$ and $k = 3$ into the entropy model gives an effective entropy per character of

$$\log_2 3 \approx 1.58 \text{ bits.}$$

For an eight-character password, this results in

$$H'(8) = 8 \log_2 3 \approx 12.6 \text{ bits,}$$

which corresponds to a reduction of approximately 39.8 bits (about 76%) relative to the brute-force baseline.

This substantial decrease demonstrates that highly accurate top- k classification dramatically reduces the effective search space for password recovery, even when the attacker is limited to a small number of guesses per keystroke.

Practical implications for authentication. The entropy reduction translates directly into a concrete guessing advantage. When the correct key is guaranteed within the top- k candidates, recovering an L -character password reduces to enumerating at most k^L full-password hypotheses instead of A^L . For our measured operating point ($k = 3$, $p \approx 1$) and an eight-character password, this is $3^8 = 6561$ candidates—down from $94^8 \approx 6.1 \times 10^{15}$ —and constitutes a search that can be exhausted essentially instantly offline and still remain feasible against online services that lack strict rate-limiting or lockout. At the more conservative $k = 5$, $p = 0.95$ operating point, the probability of recovering all eight characters correctly is $p^8 \approx 0.66$, i.e., roughly two of every three attempts yield the exact password within $5^8 \approx 3.9 \times 10^5$ guesses. These figures assume independent per-keystroke errors and are therefore an upper bound on attacker effort; adding language or syntax priors (as in [2,3]) would lower it further. The consequence is that acoustic leakage of this magnitude effectively breaks password-only authentication for the targeted user and motivates defence-in-depth measures—multi-factor authentication, hardware-backed passkeys, and aggressive online-guess rate-limiting—whose security does not depend on the secrecy of the typed characters.

Why a language model does not tighten these estimates. Our entropy analysis deliberately adopts the random-password model: a password drawn uniformly at random from the 94-symbol printable-ASCII alphabet, giving $H = 8 \log_2 94 \approx 52.4$ bits for an eight-character password. This is the standard worst-case model from the defender's perspective, and it is exactly the regime in which linguistic constraints provide no advantage: a high-entropy random string carries no exploitable n -gram or lexical structure, so a language-

model prior is effectively uniform over its support and does not reweight the acoustic top- k candidates. Formally, $H(P \mid \text{acoustics}, \text{LM}) = H(P \mid \text{acoustics})$ for such randomly generated passwords P , so the reduction we report—from ≈ 52.4 to ≈ 12.6 bits under the acoustic top-3 attack—is attributable solely to the acoustic channel and is not further improved by text correction. For human-chosen passwords, passphrases, and natural-language text, the situation only improves for the attacker: linguistic constraints reduce the effective search space, so combining the acoustic classifier with a language model (as in [2,3,9]) can only decrease the residual entropy relative to our reported values. Our acoustic-only estimates therefore already provide a conservative upper bound on attacker effort for structured secrets, and the correct and complete measure for strong, randomly generated ones. This does not diminish the value of language models elsewhere in the pipeline: while an LLM cannot reduce the entropy of a high-entropy random secret, it is a natural aid to the labelling stage, where large language models are highly effective at post-correcting noisy OCR transcriptions and at helping to synthesise and validate training text, thereby improving the quality of the automatically labelled data. We regard this LLM-assisted labelling as a promising direction and leave its integration to future work.

4.7. Video Analysis for Automatic Labelling

Our automatic labelling pipeline yields insights into the visual component of the attack. We emphasise that the task solved by our video component is fundamentally easier than the one solved by prior video-based attacks such as *ClearShot* [26]: *ClearShot* reconstructs keystrokes from video of the physical keyboard and the user's fingers, a vision problem with substantial ambiguity; our pipeline only needs to OCR the text that has already been rendered on the screen by the operating system, which is typographically clean and can be read with near-perfect accuracy. The contribution therefore does not lie in novel computer vision but in using the easy visual signal as a label source for the hard acoustic inference task.

The labelling pipeline is semi-automatic: OCR produces a draft label for every detected click, and the operator may confirm, edit, or discard each draft before it enters the training set. Because the operator only verifies machine-proposed labels rather than transcribing audio from scratch, the labelling cost drops from hours of manual annotation per session to a few minutes of review. The resulting labels are thus treated as ground truth by construction, which is what lets the downstream acoustic classifier be trained on reliable targets without large-scale hand annotation.

To align clicks with on-screen characters, we crop the recorded frames in time and space. Temporal cropping discards frames more than ± 500 ms from each detected click; spatial cropping restricts OCR to a user-selected region covering the active terminal or editor pane; and layout detection is bootstrapped from the frequency of specific Croatian QWERTZ characters (e.g., š, č, ć, đ, ž) that unambiguously identify the layout. Figure 11 compares frames immediately before and after a key press during a programming lecture. The difference image highlights the region where the typed character appears. To reduce false detections due to cursor movement or scrolling, we crop the frames in time and space: temporal cropping discards frames far from the click, and spatial cropping isolates the console or editor window (Figure 12). After extracting the region of interest, we detect the keyboard layout and map screen characters to physical keys. Figure 13 illustrates the layout-detection and character-to-key mapping step. The same mapping mechanism extends in principle to other layouts (e.g., QWERTY, AZERTY); we evaluate Croatian QWERTZ here and leave a systematic multi-layout evaluation to future work.



Figure 11. Frames before (**top**) and after (**centre**) a key press and their difference (**bottom**). The difference image highlights the newly typed character, enabling precise alignment between audio clicks and on-screen input.

Two properties of this pipeline keep the visual labels reliable. First, because each label is read from a residual (frame-difference) image, the newly typed glyph appears as bright strokes on an otherwise black background; this high-contrast, machine-rendered white-on-black text is close to the ideal operating point for OCR and is far easier than the scanned-document setting for which OCR engines are usually benchmarked—on clean,

high-contrast printed text the current Tesseract engine (v5, LSTM) attains roughly 95–97% character-level accuracy in recent benchmarks [29], and our residual pre-processing places the labelling at the easy end of this range. Second, characters that require the Shift key (uppercase letters and shifted symbols) are recognised as well: because the acoustic label space is defined over physical keys, each recognised shifted glyph is mapped back to the unshifted physical key that produces it using the victim's known software keyboard layout, so Shift-produced characters are retained in the labelling rather than excluded.

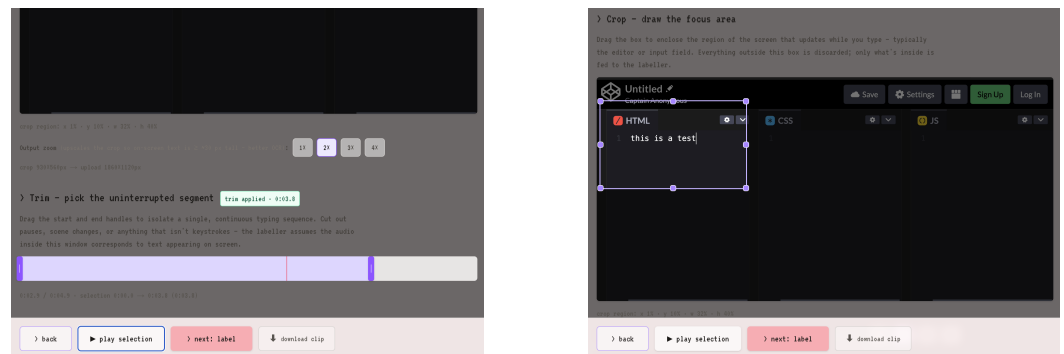


Figure 12. (Left) Temporal cropping trims the video using a video trim tool to keep only the portion around the click event, removing unnecessary frames before and after it. (Right) Spatial cropping resizes the crop rectangle to include only the region of interest (e.g., the code editor) for OCR processing.

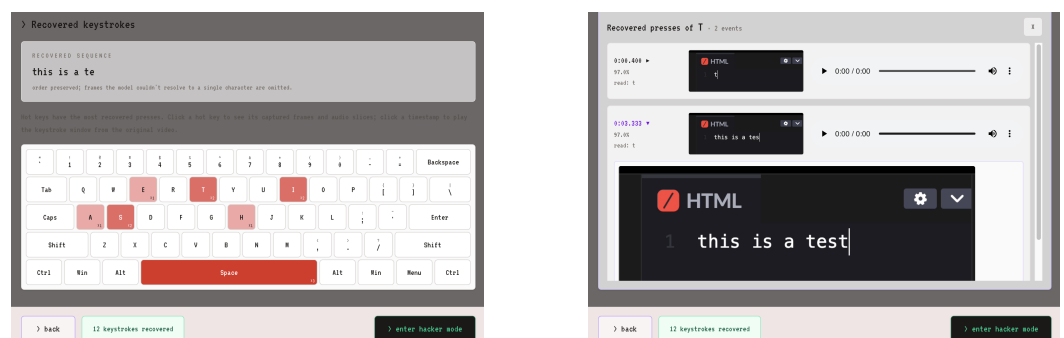


Figure 13. (Left) Automatic detection of the US QWERTY keyboard layout from the on-screen virtual keyboard. The highlighted keys represent the recovered keystrokes, while the recovered character sequence is displayed above the keyboard. (Right) Recovered keystroke events for the selected key. Each event includes the timestamp, the extracted video frame, and the corresponding audio clip used by the model. Selecting an event displays an enlarged view of the captured frame for inspection.

4.8. Robustness of the Labelling Channel to Voice Interference

A practical concern for the labelling pipeline is background speech, which is intrinsic to the online-lecture setting because the lecturer typically narrates while typing. To test whether narration corrupts the acoustic channel, we recorded a session, overlaid synthetic speech on approximately half of it, and then removed the speech with a WebRTC voice-activity-detection (VAD) detect-and-mute stage before running keystroke detection. Table 6 compares keystroke recovery before the interference (the clean reference) with recovery after the voice was added and then removed.

This behaviour follows directly from the different spectro-temporal signatures of the two sound sources. Human speech is quasi-periodic and low-frequency: it is built on a fundamental of roughly 85–255 Hz with harmonics and formants that concentrate most of its energy below about 3–4 kHz, and it is sustained over the tens to hundreds of milliseconds of a voiced phoneme. A keystroke, by contrast, is an impulsive broadband click: its energy is spread across the whole 200–8000 Hz analysis band—including the

mid–high region (roughly 1–5 kHz) that Table 1 identifies as carrying the key-discriminative information—and it lasts only a few milliseconds. Because the two sources differ both in time (sparse impulses versus sustained voicing) and in spectral centre of mass (a high-frequency transient versus a low-frequency harmonic structure), the VAD stage can gate the voiced segments while leaving the broadband keystroke transients in the surrounding non-voiced frames essentially intact. Only keystrokes that happen to coincide with a voiced segment are removed together with it, which is exactly the loss observed in Table 6. The degradation is therefore a bounded, explainable recall effect of detect-and-mute—not label corruption and not model fragility. These figures come from a single recording of one paragraph; a fully defensible study would repeat the measurement across several recordings and typists, but the methodology transfers unchanged.

Table 6. Effect of overlaid speech and its removal on keystroke recovery. After detect-and-mute, keystrokes that do not temporally coincide with speech are almost entirely recovered; the loss is confined to keystrokes that overlap the voice.

Setting	Keystrokes Recovered
Clean recording (reference)	100%
Voice added, then VAD-removed—keystrokes not overlapping speech	93%

5. Discussion

Our study confirms that the acoustic emanations of keyboard presses carry enough information to reconstruct typed text with high accuracy. By combining automatic labelling from video analysis with deep learning on mel-spectrogram features, we created a scalable attack that is effective in real-world online lecture scenarios. The key novelty is the removal of manual annotation: previous works either assumed labelled training data or used language models to infer labels post hoc, whereas we derive labels directly from the victim’s screen via OCR. This simplifies data collection and allows the attack to adapt to different keyboard layouts and languages.

Compared to state-of-the-art attacks, our model performs competitively. Harrison et al. reported 95% accuracy using a smartphone microphone placed 17 cm from the laptop [5], whereas our model achieves roughly 98% using a laptop-grade Audacity recording made during an online programming lecture, with a larger and more homogeneous training dataset available through our automatic labelling pipeline; the two numbers therefore refer to different recording setups and should not be interpreted as a direct head-to-head comparison. Although Ayati et al. achieved improved robustness by leveraging transformers and large language models [9], their method still presumes manually labelled training data; likewise, *Taheritajar and Rahaeimehr* [10] and *Fürst and Aßmuth* [15] require either per-user typing-pattern profiling or substantial unsupervised audio, whereas our pipeline avoids both by pairing OCR-derived labels with standard-microphone audio. Our automatic labelling pipeline could be combined with transformer-based classifiers in future work. Furthermore, the integration of OCR allows us to learn the mapping from physical key positions to characters, enabling attacks on different keyboard layouts (e.g., QWERTY, QWERTZ or AZERTY). Our experiments use Croatian QWERTZ keyboards, but the pipeline is generic.

There are limitations. First, the success of OCR depends on the clarity of the recorded screen; poor resolution or rapid typing can lead to mislabelled training samples. We mitigated this by flagging uncertain cases for manual review and found that only a small fraction of samples required intervention. Second, our dataset contains mainly mechanical keyboards recorded in quiet environments; although the model shows some robustness

to noise, background sounds such as talking or music may reduce accuracy. Defending against acoustic side-channel attacks can be achieved by adding noise (e.g., white noise or background music), using soft-switch keyboards that emit less distinctive sounds, or randomising typing by using on-screen keyboards or password managers. For the specific online-lecture and teleconferencing setting that motivates this paper, stream-level defences are particularly attractive: recent conferencing clients bundle keystroke-specific noise-suppression and speech-enhancement stages (for example, Krisp-style DNN denoisers and the native noise-suppression used by Zoom, Google Meet and Microsoft Teams) that can strongly attenuate the transient click energy the classifier relies on, and a “keystroke suppression” toggle should arguably be enabled by default whenever a lecturer types code on a shared screen. An orthogonal consideration is typing rhythm: Taheritajar and Rahaeimehr [10] show that inter-keystroke timings alone carry recognisable user-specific structure, so deliberately randomised typing cadence (as produced by password managers that inject keystrokes with jittered delays) yields an additional layer of defence. Beyond these heuristic countermeasures, learned defences are emerging: EveGuard [13] generates human-imperceptible audio adversarial perturbations that cause an eavesdropping classifier to fail while preserving intelligibility for legitimate listeners, a direction directly applicable to the shared-audio lecture setting studied here. Zhuang et al. emphasised that the diversity of keyboards and users affects sound patterns [2]; we observe similar variability and suggest that training on multiple keyboards can improve generalisation, an observation also borne out by Halevi and Saxena [14], who quantified how typing style alone can degrade per-key accuracy from the hunt-and-peck regime down to the touch-typing regime.

Robustness to Audio Dropouts and Conferencing Artefacts

A concern specific to the teleconferencing setting is that the audio stream is not delivered losslessly: variable-bitrate compression, jitter-buffer under-runs, packet loss and—most importantly—voice-activity-gated noise suppression can intermittently attenuate or drop the very transient onsets on which the classifier relies. We distinguish three root causes: (i) network-level loss, where RTP packets are dropped or arrive too late and the client conceals the gap; (ii) codec-level distortion, where low-bitrate Opus or AAC encoding smears the short broadband click; and (iii) enhancement-level suppression, where DNN denoisers and voice-activity-detection (VAD) gates classify an isolated keystroke as non-speech noise and remove it. The third cause is particularly relevant here because production denoisers are explicitly trained to attenuate transient non-speech events: RNNoise and comparable real-time enhancers list keyboard and mouse clicks among their target noise classes [33], so a keystroke-suppression stage that is beneficial for call quality acts directly against the attacker.

Rather than measure a single per-session dropout rate—which would be specific to one network, client and codec configuration and therefore of limited generality—we bound the effect using published conferencing quality-of-service (QoS) figures and treat the per-keystroke corruption probability ρ as a swept parameter. Standard QoS guidance places the boundary of good interactive audio at roughly 1% packet loss, with noticeable degradation in the 1–5% range and severe disruption above 5% [34]. Independent network-level measurement studies of production platforms (Zoom, Webex and Google Meet) confirm that, while median loss on healthy links is well below 1%, sessions on congested residential and mobile links routinely enter this 1–5% regime [35]. We therefore analyse $\rho \in \{1\%, 2\%, 5\%, 10\%\}$, spanning nominal to adverse conditions, and treat the upper end as a conservative worst case rather than a typical value. This range also subsumes enhancement-level suppression: a keystroke removed by a VAD gate is, for the purposes of the classifier, indistinguishable from a lost packet and is captured by the same ρ .

We model each keystroke as being independently corrupted—dropped or rendered unclassifiable—with probability ρ . A corrupted keystroke cannot be recovered from acoustics alone, so the expected per-keystroke top- k accuracy degrades from a_k to $(1 - \rho) a_k$, and the probability that an L -character password is captured in full is $(1 - \rho)^L$. Table 7 instantiates this model with our measured operating point ($a_1 = 98.1\%$, $a_3 = 100\%$) for an eight-character password. Because the full-capture probability is $(1 - \rho)^L$, even modest per-keystroke loss compounds sharply over a full password: at $\rho = 1\%$, the attacker still captures a complete eight-character string 92% of the time, but this falls to 43% at $\rho = 10\%$. Per-keystroke top- k accuracy, by contrast, degrades only linearly in ρ and remains high ($\geq 88\%$ top-1, $\geq 90\%$ top-3) across the whole range, so individual-key recognition is far more fault-tolerant than full-password capture.

Table 7. Analytical effect of the per-keystroke corruption probability ρ (lost packet, codec smearing, or VAD suppression) on attack performance, under the independent-corruption model $a_k \rightarrow (1 - \rho) a_k$ and full-capture probability $(1 - \rho)^L$. Values use the measured operating point $a_1 = 98.1\%$, $a_3 = 100\%$ and password length $L = 8$. The ρ range is taken from published conferencing-QoS targets and measurement studies [34,35], not from our own measurements.

ρ	Eff. Top-1 $(1 - \rho) a_1$	Eff. Top-3 $(1 - \rho) a_3$	Full 8-Char Capture $(1 - \rho)^8$	Regime
0%	98.1%	100.0%	100.0%	lossless (reference)
1%	97.1%	99.0%	92.3%	good link
2%	96.1%	98.0%	85.1%	acceptable
5%	93.2%	95.0%	66.3%	degraded
10%	88.3%	90.0%	43.0%	adverse (worst case)

Two mitigation strategies follow directly from the model and validate the fault-tolerance of the pipeline without requiring lossless capture. First, dropout-aware training: augmenting the training data with simulated packet loss and VAD gating—randomly zeroing a fraction of onsets and applying codec round-trips—hardens the feature extractor against these artefacts, in the same spirit as the SpecAugment masking already employed [21]; because our augmentation pipeline already zeroes time–frequency bins, extending it to zero whole onsets at rate ρ is a one-line change that makes the training distribution match the deployment channel. Second, redundancy at capture and decision time: because a lecture repeats many instances of each key, aggregating m independent acoustic observations of the same on-screen character by majority vote over the softmax outputs recovers most single-instance losses—the probability that all m copies are corrupted falls as ρ^m , so at $\rho = 10\%$, just $m = 2$ repetitions cut the effective per-key loss to 1% and $m = 3$ to 0.1%. Capturing the raw local microphone in parallel with the conferencing stream—feasible whenever the attacker controls the receiving endpoint—side-steps enhancement-level suppression entirely. These strategies place our pipeline within the broader robustness literature: augmentation against channel artefacts parallels the augmentation-based robustness of [25], while the dual perspective—an attacker hardening against dropouts versus a defender injecting adversarial perturbations to induce them [13]—frames dropout tolerance as one side of an adversarial-robustness problem. A systematic per-platform (Zoom, Google Meet, Microsoft Teams) characterisation of the empirical dropout distribution and an end-to-end evaluation of dropout-aware training against it remain important directions for future measurement work.

6. Conclusions

This work presents an end-to-end acoustic side-channel attack on keyboards that enables the automatic collection and labelling of training data from online videos. By combin-

ing OCR-derived ground-truth characters with detected keystroke sounds, a scalable and adaptable pipeline is established that removes the need for manual annotation. Using mel-spectrogram representations and a convolutional neural network classifier, high keystroke recognition accuracy is achieved under realistic online lecture and conferencing conditions.

Experimental results demonstrate that near-perfect classification performance can be obtained when sufficient training data are available, confirming that keyboard acoustic emanations contain substantial information about typed content. More importantly, the study shows that transfer learning enables effective keystroke classification even when only a small amount of training data is collected from a new, previously unseen typist and keyboard during a short recording session.

The approach couples a transferable pretrained backbone—shown to generalise across victims and keyboards—with a lightweight, target-specific head adapted from minimal data. The combination of automatic labelling, transfer learning, and targeted fine-tuning significantly lowers the barrier for conducting effective acoustic side-channel attacks.

Overall, the results highlight the severity of acoustic information leakage in modern computing environments and demonstrate that even short recording sessions can suffice to reconstruct sensitive input with high confidence. This underscores the need for increased awareness of acoustic side-channel threats in online communication and shared environments, as well as the importance of considering such risks in the design and use of input devices.

Author Contributions: Conceptualisation, D.V. and T.P.; methodology, D.V. and M.B.; software, D.V., T.P., M.B. and I.S.; validation, M.B. and I.S.; formal analysis, D.V. and T.P.; investigation, D.V.; resources, T.P.; data curation, D.V.; writing—original draft preparation, D.V.; writing—review and editing, M.B., I.S. and T.P.; supervision, T.P.; project administration, T.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union (NextGenerationEU) under the Croatian Recovery and Resilience Plan 2021–2026 (NRRP), through the University of Split institutional project “Advanced modeling and measurement procedures in engineering multiphysics problems (MULTIMOD)—IP-UNIST08”, approved by the Ministry of Science, Education and Youth of the Republic of Croatia.

Data Availability Statement: The source code, the OCR-aligned labelling pipeline, the trained CNN models (TensorFlow SavedModel format), the hyperparameter grid-search CSV, the full requirements.txt/environment.yml, and the labelled keystroke-audio dataset generated in this study have been released to facilitate reproducible research. The code repository is available at <https://github.com/k3rnel-pan1c-ksd/automated-keystroke-acoustic-attack> (accessed on 24 July 2026). Raw audio recordings of lectures have been released in anonymised form (speech segments removed), and a replay-style synthetic keystroke dataset has been provided for users who cannot use the raw recordings.

Acknowledgments: During the preparation of this manuscript, the authors used the generative AI tool ChatGPT (OpenAI, <https://chat.openai.com>; model version GPT-4) solely for the purpose of modifying and improving the English language and readability of author-written text. The authors have reviewed and edited all output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Worked Mel-Spectrogram Conversion

For the keystroke `i_1.wav` (44.1 kHz, 11,025 samples, 250 ms), the STFT uses a 1024-sample Hann window (23.2 ms) and a 225-sample hop (5.1 ms), giving 513 rFFT bins spaced 43.07 Hz apart and a 513×50 power spectrogram. Applying the 64 Slaney mel filters over 200–8000 Hz yields a 64×50 mel spectrogram. As an example, mel band

$m = 33$ (triangular support 1914.7/2002.2/2093.7 Hz) covers four rFFT bins (1938–2067 Hz); at the click-peak frame $t = 10$, the band energy is

$$S[33, 10] = \sum_f w_{33}(f) |X(f, 10)|^2 = 6.148 \times 10^{-2}. \quad (\text{A1})$$

With the clip maximum $\text{ref} = 4.972 \times 10^{-1}$, the decibel value is

$$10 \log_{10} \left(\frac{6.148 \times 10^{-2}}{4.972 \times 10^{-1}} \right) = -9.08 \text{ dB}, \quad (\text{A2})$$

matching the full `power_to_db` output at that cell. The corresponding spectral-drift comparison for both key pairs is reported in Table 1; the full computation is reproduced in the accompanying notebook.

References

1. Asonov, D.; Agrawal, R. Keyboard acoustic emanations. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P)*; IEEE: Piscataway, NJ, USA, 2004; pp. 3–11. [\[CrossRef\]](#)
2. Zhuang, L.; Zhou, F.; Tygar, J.D. Keyboard acoustic emanations revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, USA, 7–11 November 2005; pp. 373–382. [\[CrossRef\]](#)
3. Berger, Y.; Wool, A.; Yeredor, A. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VI, USA, 30 October–3 November 2006; pp. 245–254. [\[CrossRef\]](#)
4. Compagno, A.; Conti, M.; Lain, D.; Tsudik, G. Don't Skype & Type!: Acoustic eavesdropping in Voice-over-IP. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 703–715. [\[CrossRef\]](#)
5. Harrison, J.; Toreini, E.; Mehrnezhad, M. A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards. In *Proceedings of the 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*; IEEE: Piscataway, NJ, USA, 2023; pp. 270–280. [\[CrossRef\]](#)
6. Slater, D.; Novotney, S.; Moore, J.; Morgan, S.; Tenaglia, S. Robust keystroke transcription from the acoustic side-channel. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, San Juan, Puerto Rico, 9–13 December 2019; pp. 776–787. [\[CrossRef\]](#)
7. Bai, J.-X.; Liu, B.; Song, L. I Know Your Keyboard Input: A Robust Keystroke Eavesdropper Based on Acoustic Signals. In *Proceedings of the 29th ACM International Conference on Multimedia (MM)*, Chengdu, China, 20–24 October 2021; pp. 1239–1247. [\[CrossRef\]](#)
8. Wang, X.; Liu, Y.; Li, S.-C. Deep Learning Enabled Keystroke Eavesdropping Attack over Videoconferencing Platforms. In *Proceedings of the IEEE INFOCOM 2023—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–2. [\[CrossRef\]](#)
9. Ayati, S.A.; Park, J.H.; Cai, Y.; Botacin, M. Making Acoustic Side-Channel Attacks on Noisy Keyboards Viable with LLM-Assisted Spectrograms' "Typo" Correction. In *Proceedings of the 19th USENIX WOOT Conference on Offensive Technologies (WOOT '25)*, Seattle, WA, USA, 11–12 August 2025; USENIX Association: Berkeley, CA, USA, 2025.
10. Taheritajar, A.; Rahaeimehr, R. Acoustic Side Channel Attack on Keyboards Based on Typing Patterns. In *Cryptology and Network Security (CANS 2025)*; Lecture Notes in Computer Science; Kim, Y., Miyaji, A., Tibouchi, M., Eds.; Springer: Singapore, 2026; Volume 16351. [\[CrossRef\]](#)
11. Dang, J.; Chen, L.; Wu, J.; Lin, R.; Wang, B.; Wang, Y.; Wang, L.; Zhu, N.; Wang, T. Diff-LMM: Diffusion Teacher-Guided Spatio-Temporal Perception for Video Large Multimodal Models. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, QC, Canada, 16–22 August 2025; pp. 873–881. [\[CrossRef\]](#)
12. Quattrone, B.; Badr, Y. A Survey on Acoustic Side-Channel Attacks: An Artificial Intelligence Perspective. *J. Cybersecur. Priv.* **2026**, *6*, 6. [\[CrossRef\]](#)
13. Chang, J.-W.; Sun, K.; Xia, D.; Zhang, X.; Koushanfar, F. EveGuard: Defeating Vibration-based Side-Channel Eavesdropping with Audio Adversarial Perturbations. In *Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P)*; IEEE: Piscataway, NJ, USA, 2025; pp. 4534–4552. [\[CrossRef\]](#)
14. Halevi, T.; Saxena, N. A Closer Look at Keyboard Acoustic Emanations: Random Passwords, Typing Styles and Decoding Techniques. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, Seoul, Republic of Korea, 2–4 May 2012; pp. 89–90. [\[CrossRef\]](#)

15. Fürst, D.; Aßmuth, A. Practical Acoustic Eavesdropping on Typed Passphrases. In Proceedings of the 16th International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2025), Valencia, Spain, 6–10 April 2025.
16. Taheritajar, A.; Mahmoudpour Harris, Z.; Rahaeimehr, R. A Survey on Acoustic Side Channel Attacks on Keyboards. In *Information and Communications Security (ICICS 2024)*; Lecture Notes in Computer Science; Springer: Singapore, 2024. [CrossRef]
17. Stevens, S.S.; Volkman, J.; Newman, E.B. A Scale for the Measurement of the Psychological Magnitude Pitch. *J. Acoust. Soc. Am.* **1937**, *8*, 185–190. [CrossRef]
18. Davis, S.B.; Mermelstein, P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [CrossRef]
19. Hershey, S.; Chaudhuri, S.; Ellis, D.P.W.; Gemmeke, J.F.; Jansen, A.; Moore, R.C.; Plakal, M.; Platt, D.; Saurous, R.A.; Seybold, B.; et al. CNN Architectures for Large-Scale Audio Classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2017; pp. 131–135. [CrossRef]
20. Abdel-Hamid, O.; Mohamed, A.; Jiang, H.; Deng, L.; Penn, G.; Yu, D. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2014**, *22*, 1533–1545. [CrossRef]
21. Park, D.S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B.; Cubuk, E.D.; Le, Q.V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 2613–2617. [CrossRef]
22. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
23. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
24. Kong, Q.; Cao, Y.; Iqbal, T.; Wang, Y.; Wang, W.; Plumbley, M.D. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2880–2894. [CrossRef]
25. Hama Rawf, K.M. Improved CoAtNet for Robust Acoustic Side-Channel Attack Classification on Keyboards. *Int. J. Inf. Secur.* **2026**, *25*, 26. [CrossRef]
26. Balzarotti, D.; Cova, M.; Vigna, G. ClearShot: Eavesdropping on Keyboard Input from Video. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P)*; IEEE: Piscataway, NJ, USA, 2008; pp. 170–183. [CrossRef]
27. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958. Available online: <https://jmlr.org/papers/v15/srivastava14a.html> (accessed on 24 July 2026).
28. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*; PMLR: Lille, France, 2015; Volume 37, pp. 448–456. Available online: <https://proceedings.mlr.press/v37/ioffe15.html> (accessed on 24 July 2026).
29. Smith, R. An Overview of the Tesseract OCR Engine. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*; IEEE: Piscataway, NJ, USA, 2007; Volume 2, pp. 629–633. [CrossRef]
30. McFee, B.; Raffel, C.; Liang, D.; Ellis, D.P.W.; McVicar, M.; Battenberg, E.; Nieto, O. Librosa: Audio and Music Signal Analysis in Python. In Proceedings of the 14th Python in Science Conference (SciPy), Austin, TX, USA, 6–12 July 2015; pp. 18–25. [CrossRef]
31. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3320–3328.
32. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
33. Valin, J.-M. A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement. In *Proceedings of the IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5. [CrossRef]
34. International Telecommunication Union. ITU-T Recommendation G.1010: End-User Multimedia QoS Categories; ITU-T: Geneva, Switzerland, 2001. Available online: <https://www.itu.int/rec/T-REC-G.1010> (accessed on 24 July 2026).
35. Chang, H.; Varvello, M.; Hao, F.; Mukherjee, S. Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet. In Proceedings of the 21st ACM Internet Measurement Conference (IMC), Online, 2–4 November 2021; pp. 216–228. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.