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

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

Posted Date: 19 June 2026

doi: 10.20944/preprints202606.1478.v1

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

 doi:10.20944/preprints202606.1478.v1



*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

* Correspondence: toni.perkovic@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. Our dataset contains 50 unique keys from standard QWERTZ keyboards recorded during real programming lectures. 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 analyzing 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 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 [7]. Related work also investigates end-to-end transcription settings (joint detection and recognition) and robustness

 doi:10.20944/preprints202606.1478.v1



to overlapping keystrokes and background noise [5], as well as improved acoustic modelling for keystroke eavesdropping with robust features and classification pipelines [6,8]; more recent directions combine visual transformers and large language models to correct mispredictions under noise [12], or exploit user-specific typing patterns instead of raw click acoustics [11].

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 [1,7,8]. 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.

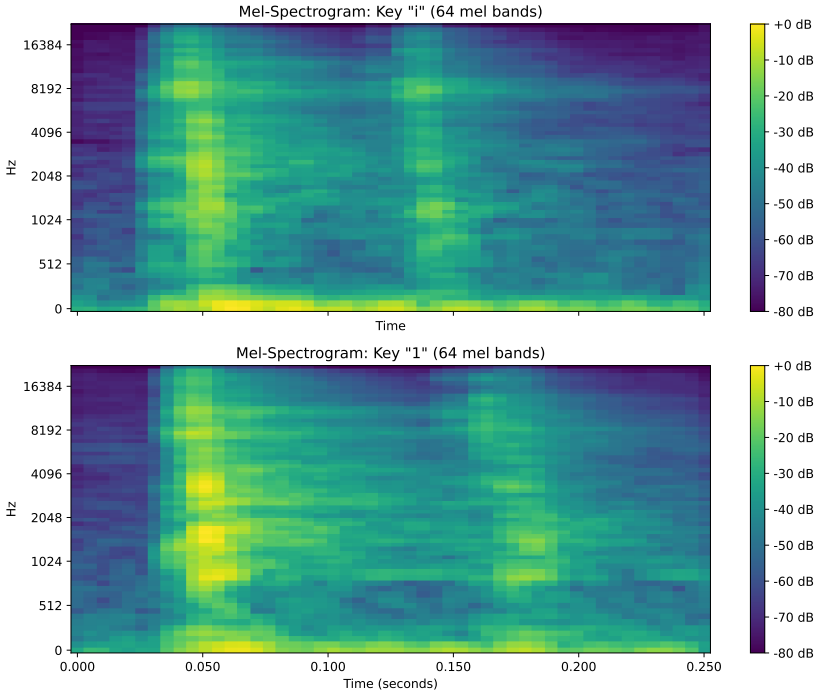
This paper addresses the labelling bottleneck by introducing an automated pipeline that *extracts labelled keystrokes from video recordings* that contain both (i) the screen content and (ii) the corresponding audio. We leverage OCR over video frames to recover the character sequence as it appears on 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 initializing the network from a previously trained, generalized 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 [1,7] or focuses on different threat models (e.g., dictionary recovery, end-to-end transcription, or purely VoIP capture) [3–5], our pipeline explicitly targets *scalability and realism* in dataset creation while maintaining a practical acoustic attack setting.

*Motivating Analysis: Acoustic Differences Between Similar Keys*

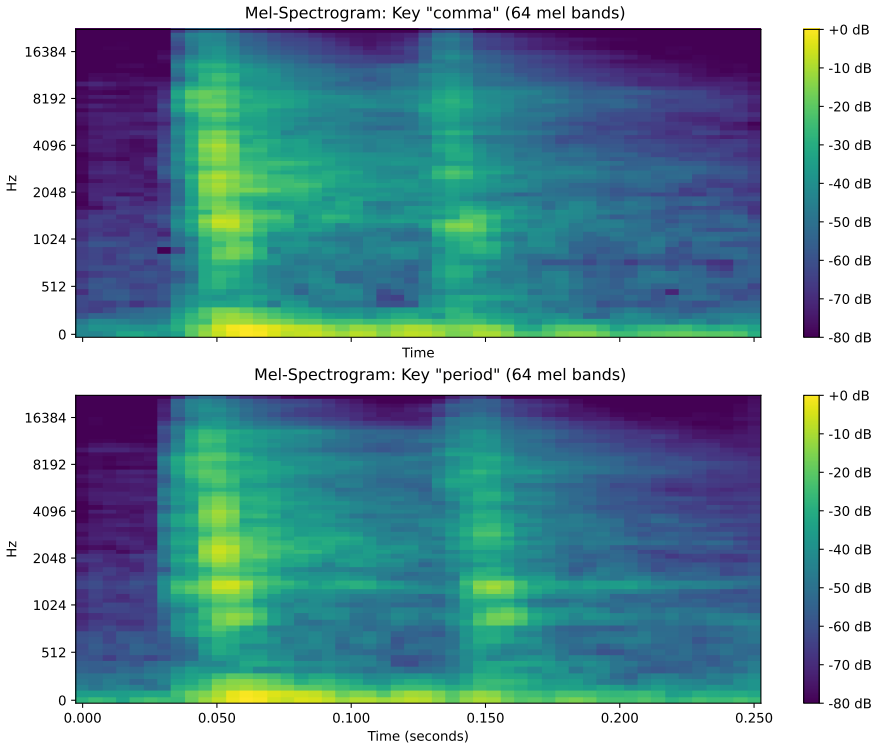
To illustrate why keystroke classification is feasible, we compare the acoustic signatures of visually similar keys. Figure 1 shows mel-spectrograms for two key pairs: the lowercase letter `i` versus the digit `1` and the comma versus the period. In both cases the upper panel displays the first key and 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. Likewise, the period has greater low-frequency content and shorter duration than the comma.

 doi:10.20944/preprints202606.1478.v1






**(a)** Mel-spectrograms of `i` (top) and `1` (bottom)




**(b)** Mel-spectrograms of comma (top) and period (bottom)

**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.

We also computed simple statistical features—including spectral rolloff, root-mean-square (RMS) energy, dominant frequency, zero-crossing rate, and spectral centroid—for each key. The resulting feature vectors differ significantly between paired keys. For instance, the digit `1` has a higher zero-crossing rate and spectral centroid than the letter `i`, reflecting its increased high-frequency energy. Similarly, the

 doi:10.20944/preprints202606.1478.v1



period shows higher peak amplitude and power spectral density (PSD) mean than the comma, indicating stronger overall energy. These differences provide empirical evidence that individual keystrokes emit distinguishable acoustic patterns, motivating our use of mel-spectrograms as a discriminative feature representation.

## 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 formalizing dictionary-style attacks that combine acoustic classification with structured search over likely strings [3]. Halevi and Saxena [26] 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.* [14].

### *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 [7,8]. In parallel, end-to-end transcription work emphasizes that realistic settings must handle imperfect keystroke detection, overlap, and environmental noise, shifting evaluation from isolated classification toward pipeline robustness [5]. 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 [16], and Davis and Mermelstein [17] 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 [13,25]. 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 [9].

### *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 [6,7], a property of convolutional architectures first formalised for image recognition by LeCun *et al.* [23] and subsequently shown to be highly effective on mel-spectrogram inputs for speech and general audio [13,25]. For optimisation, Adam is commonly used for stable convergence on noisy gradients and large datasets

 doi:10.20944/preprints202606.1478.v1



[10]. While alternative architectures (e.g., recurrent or attention-based models) can improve robustness in some settings [5], and transfer from large-scale pretrained audio models offers a complementary direction [28], our focus is on a practical and reproducible pipeline where the main novelty is scalable dataset creation rather than architectural complexity.

### 2.5. Positioning Our Contribution

Table 1 positions our work against representative prior studies. In contrast to work that assumes labelled keystroke audio collection per target device/user [1,7,8], work that emphasises dictionary or end-to-end transcription under different assumptions [3,5], and to purely video-based keystroke-inference approaches such as *ClearShot* [27], 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 [12]: 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 [6,7,13,25] 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 1.** 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) [5] | 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) [6] | Near-field mic | Yes | Robust features + ML/DL | Key classification | Generalization remains challenging across keyboards/users |
| Harrison et al. (2023) [7] | 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 [1,6–8]. In practice, such labelled datasets are difficult to obtain because typed passwords or sensitive input are typically masked, and manual annotation of long

 doi:10.20944/preprints202606.1478.v1



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.

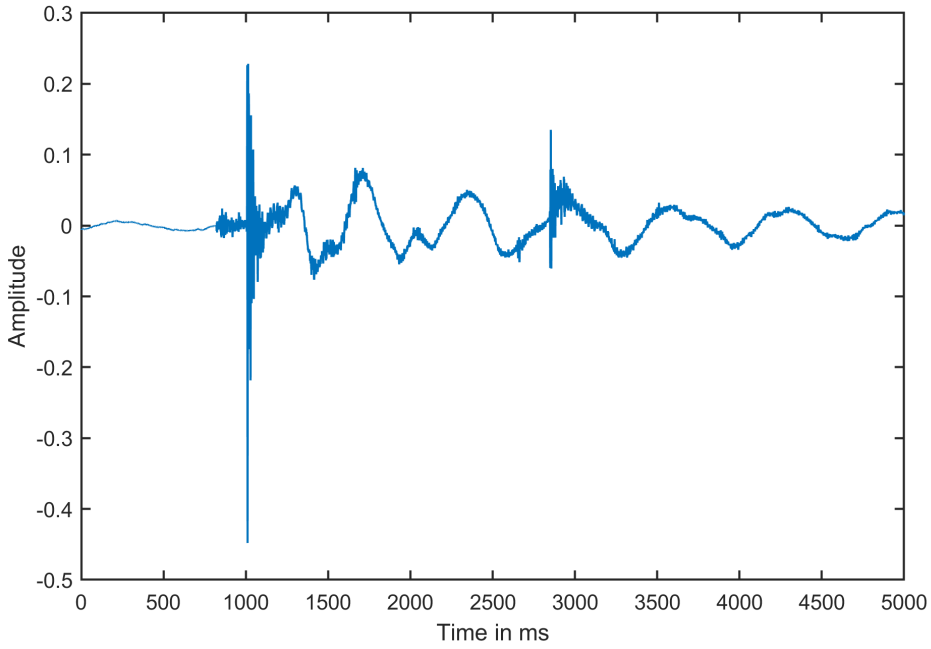



**Figure 2.** Waveform of a single key press. The raw audio signal contains two peaks for pressing and releasing the key, but the high-dimensional waveform is not directly suitable for CNN input.

### *3.2. Convolutional Neural Networks for Keystroke Classification*

A convolutional neural network (CNN) [23] 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 [13,25]. 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.

 doi:10.20944/preprints202606.1478.v1



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

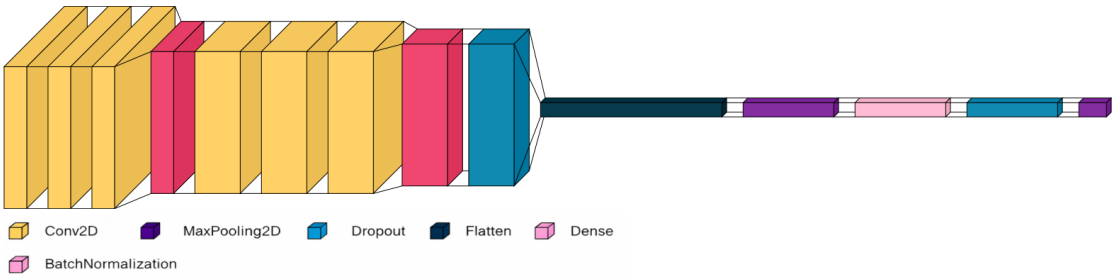



**Figure 3.** Visual representation of the convolutional neural network architecture used for keystroke classification.

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 $3 \times 3$ convolution is used to capture joint time–frequency patterns, followed by $3 \times 1$ and $1 \times 3$ 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.

A max-pooling layer with a pooling size of $(2 \times 1)$ 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) [20] 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 [21] 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 [10], and optimisation is performed by minimising the categorical cross-entropy loss, which is appropriate for multi-class classification tasks.

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 [4,7,8]. 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

 doi:10.20944/preprints202606.1478.v1



analysis in the Password Entropy Reduction subsection quantifies the consequences of phase (b) for password security. We assume the same keyboard and a comparable recording environment across phases, consistent with the transfer-learning setting described in the Transfer Learning subsection of this section. Keys that require modifier chords (e.g., Shift+letter) are not enumerated separately in the training alphabet; handling uppercase and symbol characters that require Shift is left to future work, but does not affect the password-entropy analysis, which is computed with respect to the 94-symbol printable-ASCII alphabet.

Consistent with prior work in this area [1–3,7], 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.

### *3.3. Data Collection and Labelling*

Audio was recorded with the open-source `Audacity` software 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. A press event is detected whenever the energy exceeds a threshold, and we extract the subsequent 100 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. The first dataset consists of 125 samples per key collected from a single keyboard–user setup and is used for model training (pretraining of the generalised base model). The second dataset contains 25 samples per key from an independent session with the same setup and is used to evaluate the transfer-learning performance of the proposed approach under limited-data conditions; transfer-learning experiments draw subsets of $\{13, 20, 25\}$ samples per key from this second dataset. In parallel, screen recordings are sampled at 30 frames per second and processed with the Tesseract OCR engine [19] 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.

### *3.4. Feature Extraction and Data Augmentation*

Each labelled audio segment is transformed into a mel-spectrogram [16,17] using a Hann window of length 1024 samples ($\approx$ 23 ms) and a hop length of 225 samples (5 ms) as suggested by recent practical attacks [7]. 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 $64 \times 25$ spectrograms are then converted to decibel scale using `librosa.power_to_db` [18] (see Listing 1 in the accompanying code repository). 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) [9]. 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.

### *3.5. 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.

 doi:10.20944/preprints202606.1478.v1



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 ($k = 1$–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 minutes, resulting in a total training time of about 8 minutes 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.

### *3.6. Transfer Learning*

Transfer learning [22,28] is employed to adapt a convolutional neural network pretrained on a larger keystroke dataset to a target dataset obtained from a single user and a single recording session. In this setting, the objective is not to optimise performance on a highly generalised target dataset, but rather to efficiently adapt a previously generalised model to a narrowly defined acquisition condition with limited data availability.

The pretrained network is loaded and used as a fixed feature extractor by freezing all convolutional layers [22]. The original classification layer is removed, and a new task-specific classification head

 doi:10.20944/preprints202606.1478.v1



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 [22,28]. By freezing these layers, the model retains its previously learned generalisation capability, while the classification head adapts to user- and session-specific characteristics.

This strategy is particularly suitable for the considered use case, where the target dataset is intentionally non-generalised. Rather than preventing overfitting to a diverse dataset, the focus is on leveraging a robust, pretrained representation to enable accurate classification with minimal retraining effort under constrained data conditions.

### *3.7. 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 $\in \{0.2,0.3,0.4\}$ , and the learning rate of the Adam optimiser $\in \{10^{-3}, 3\times10^{-4}, 10^{-4}\}$. 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 2 summarises the hyperparameters explored and their meanings. Table 3 lists the five best configurations ranked by validation macro-F1.

**Table 2.** 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 |

Among these, configuration 13 with filters $(32,64)$, 256 dense units, a dense dropout of 0.4 and learning rate $10^{-3}$ 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 6.6 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.

 doi:10.20944/preprints202606.1478.v1



**Table 3.** 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 |

### *3.8. 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,

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

Because our dataset has imbalanced class frequencies, we average the F1-score over classes (*macro-F1*), which assigns equal weight to each class regardless of its frequency. 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.

## 4. Results

### *4.1. Audio Signal and Feature Extraction*

The feature-extraction procedure itself is described in Section 3.4; 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.

 doi:10.20944/preprints202606.1478.v1



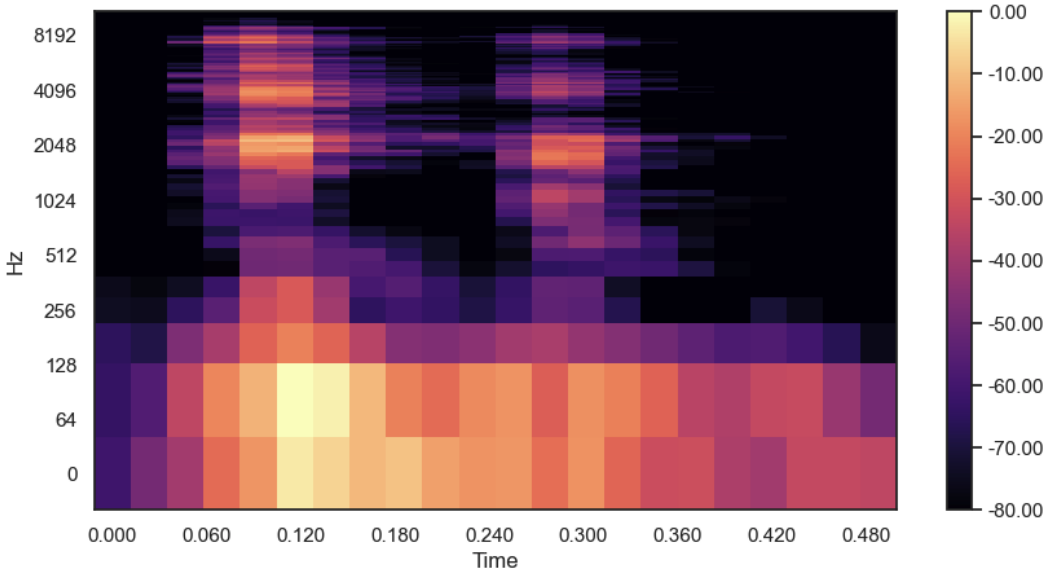



**Figure 4.** Mel-spectrogram of the same key press. The frequency axis is mapped to the mel scale, yielding a perceptually meaningful representation suitable for convolutional processing.

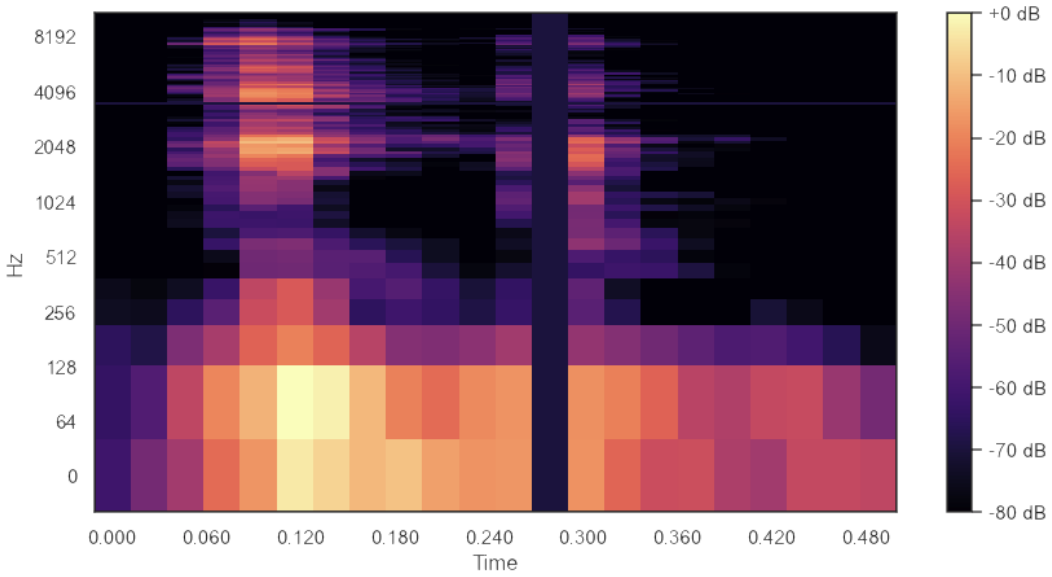



**Figure 5.** Masked mel-spectrogram produced by SpecAugment. Random time and frequency bands are masked to encourage the model to learn robust features.

### *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.

 doi:10.20944/preprints202606.1478.v1



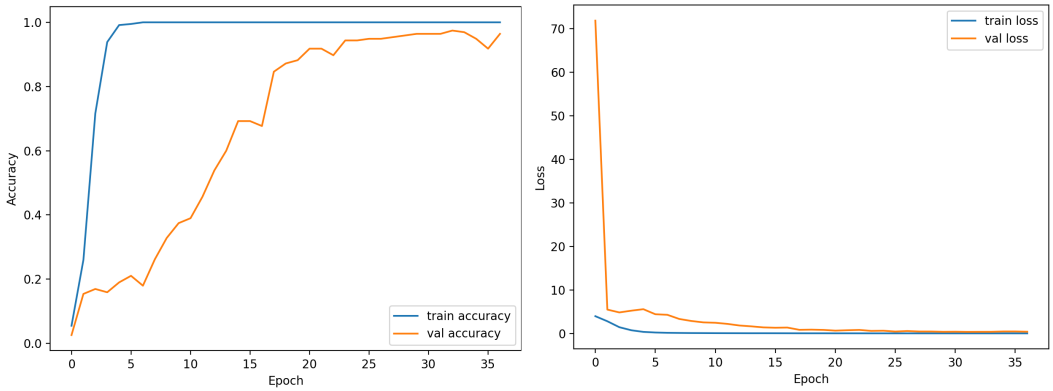



**Figure 6.** (left) Training and validation accuracy, (right) Training and validation cross-entropy loss

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 pre-trained 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.

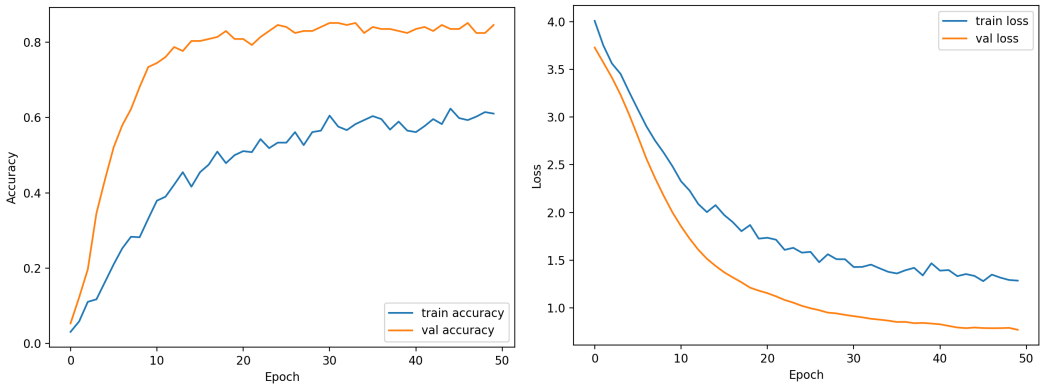



**Figure 7.** (left) Training and validation accuracy, (right) Training and validation cross-entropy loss for the transfer-learning stage. The steady decline in loss mirrors the increase in accuracy and illustrates stable optimisation.

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 6.63 million parameters, of which 22,575 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.

 doi:10.20944/preprints202606.1478.v1



*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.

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 accuracy 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.

 doi:10.20944/preprints202606.1478.v1



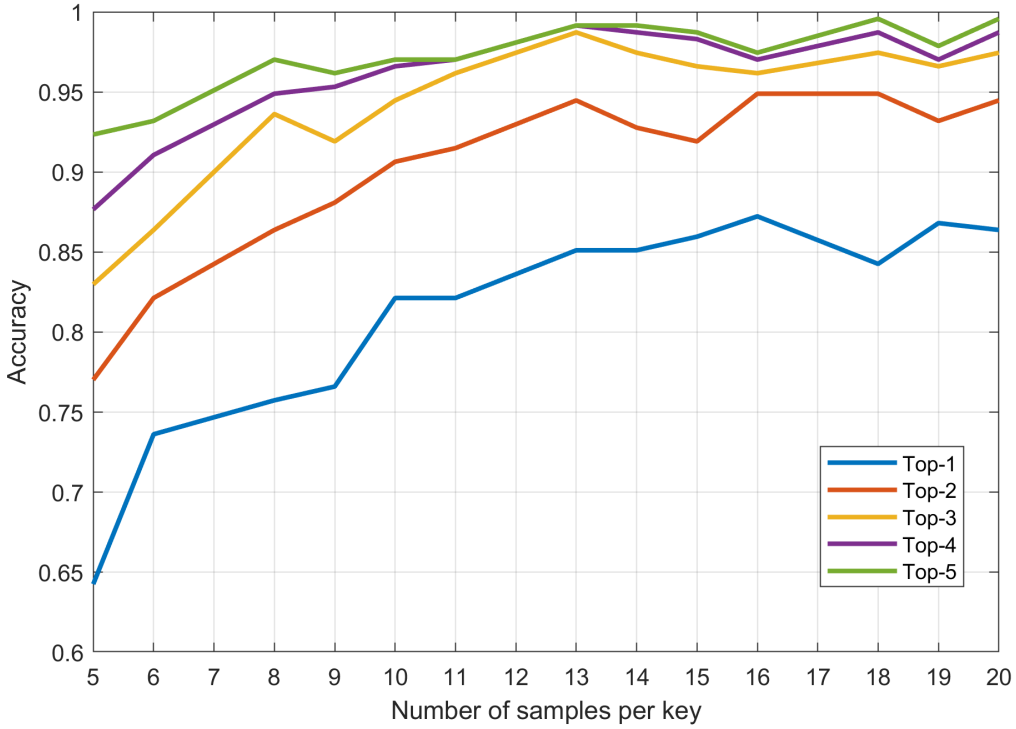



**Figure 8.** top-$k$ test accuracy versus number of training samples per key obtained by fine-tuning the same pretrained model using transfer learning.

### *4.4. 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 $H(L) = L \log_2 A$ [24]. 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 entropy becomes

$$H'(L) = L\big[p \log_2 k + (1-p) \log_2 (A-k)\big].$$

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.

 doi:10.20944/preprints202606.1478.v1



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.

### *4.5. 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* [27]: 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 near-perfect to read. 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 $\pm 500$ 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 9 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 10). After extracting the region of interest, we detect the keyboard layout and map screen characters to physical keys. Figure 11 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..

 doi:10.20944/preprints202606.1478.v1




**Figure 9.** Frames before (left) and after (centre) a key press and their difference (right). The difference image highlights the newly typed character, enabling precise alignment between audio clicks and on-screen input.

 doi:10.20944/preprints202606.1478.v1



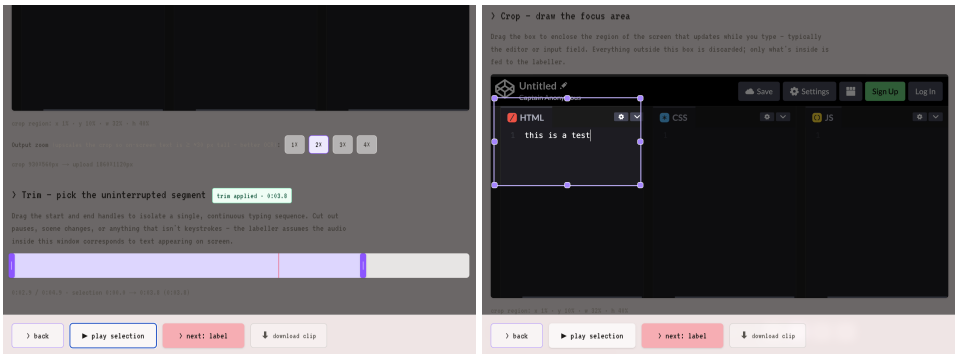



**Figure 10.** Left: temporal cropping removes frames far from the click event to reduce noise. Right: spatial cropping isolates the region of interest (e.g., code editor) for OCR processing.

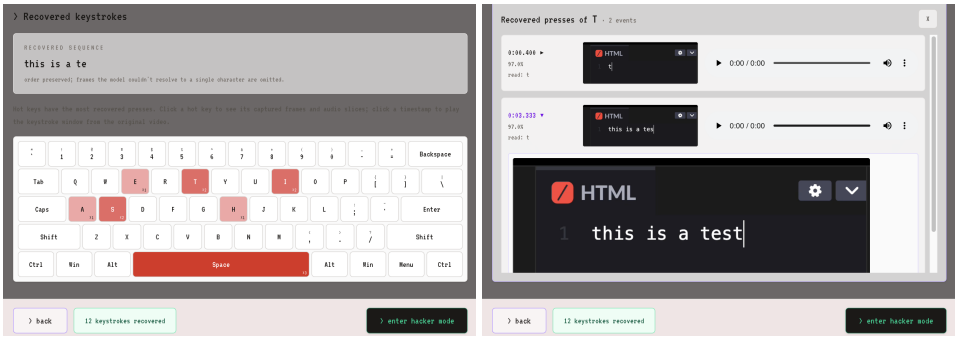



**Figure 11.** Left: automatic detection of the keyboard layout (US QWERTY) from the on-screen virtual keyboard. Right: mapping of recognised characters to physical keys, which enables the attack to work across different layouts.

## 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 [7], 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 [12], their method still presumes manually labelled training data; likewise, *Taheritajar and Rahaeimehr* [11] 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 mis-labelled training samples. We mitigated this by flagging uncertain cases for manual review and found that only a small fraction of samples required inter-

 doi:10.20944/preprints202606.1478.v1



vention. 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 [11] 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. *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* [26], who quantified how typing style alone can degrade per-key accuracy from the hunt-and-peck regime down to the touch-typing regime.

An important outcome of this study is the observation that high attack performance can be retained even when only a small number of training samples are available, provided that transfer learning is employed. While the initial model is trained on a relatively large and homogeneous dataset to learn general acoustic–spectral representations of keystrokes, this "generalised" model is not intended to directly operate across all keyboards, users, and recording conditions. Instead, it serves as a robust feature extractor that can be efficiently adapted to a specific target setup.

Our experiments demonstrate that fine-tuning the pretrained model with data obtained from a single user and a single recording session yields strong performance, even when the number of samples per key is severely limited. In particular, using as few as 13 samples per key is sufficient to achieve high top-$k$ accuracy on unseen test data, indicating that the learned representations transfer effectively to the target domain. This substantially reduces the data collection effort and enables practical attacks within short observation windows, such as a single online lecture or meeting.

This result reframes the objective of the proposed approach. Rather than aiming to construct a universally generalised keystroke classifier, the method follows a two-stage strategy: first, learning generic acoustic features from a broader dataset, and second, rapidly adapting the model to a specific victim, keyboard, and environment using minimal additional data. In this context, overfitting to the target setup is not a limitation but an intended property, as the goal is to maximise reconstruction accuracy for a particular user rather than to generalise across heterogeneous conditions.

Future improvements could further strengthen this approach by training the base model on a more diverse set of keyboards, users, recording environments, and realistic background noises. Increasing variability during pretraining would likely yield a more robust and transferable feature extractor, thereby improving adaptation performance under even more constrained data scenarios. Such extensions would enhance the generalisation capacity of the base model while preserving the efficiency and practicality of the transfer learning stage.

## 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 combining 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.

 doi:10.20944/preprints202606.1478.v1



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 single user and keyboard during a short recording session. By fine-tuning a pretrained, generalised base model, high top-$k$ accuracy is maintained with significantly reduced data requirements, making the attack practical in scenarios where extensive data collection is not feasible.

The proposed approach is not aimed at producing a universally generalised model that works across all keyboards and users. Instead, it outlines a practical methodology for rapidly adapting a pretrained model to a specific victim, keyboard, and environment, achieving usable results with minimal effort. 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:** Conceptualization, 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 are released to facilitate reproducible research. The code repository is available at https://github.com/k3rnel-pan1c-ksd/automated-keystroke-acoustic-attack. Raw audio recordings of lectures are released in anonymised form (speech segments removed), and a replay-style synthetic keystroke dataset is provided for users who cannot use the raw recordings.

**Acknowledgments:** During the preparation of this manuscript, the authors used OpenAI's ChatGPT for the purposes of modifying and improving the English language. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

## References

1. Asonov, D.; Agrawal, R. Keyboard acoustic emanations. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P)*; 2004; pp. 3–11. doi:10.1109/SECPRI.2004.1301311.
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)*; 2005; pp. 373–382. doi:10.1145/1102120.1102169.
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)*; 2006; pp. 245–254. doi:10.1145/1180405.1180436.
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)*; 2017; pp. 703–715. doi:10.1145/3052973.3053005.
5. 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)*; 2019; pp. 776–787. doi:10.1145/3359789.3359816.

 doi:10.20944/preprints202606.1478.v1




6. 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)*; 2021; pp. 1239–1247. doi:10.1145/3474085.3475539.
7. Harrison, J.; Toreini, E.; Mehrnezhad, M. A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*; 2023; pp. 270–280. doi:10.1109/EuroSPW59978.2023.00034.
8. Wang, X.; Liu, Y.; Li, S.-C. Deep Learning Enabled Keystroke Eavesdropping Attack over Videoconferencing Platforms. In *IEEE INFOCOM 2023 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*; 2023; pp. 1–2. doi:10.1109/INFOCOMWKSHPS57453.2023.10225861.
9. 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 Interspeech 2019*; 2019; pp. 2613–2617. doi:10.21437/Interspeech.2019-2680.
10. Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*; 2015. arXiv:1412.6980. https://arxiv.org/abs/1412.6980.
11. Taheritajar, A.; Rahaeimehr, R. Acoustic Side Channel Attack on Keyboards Based on Typing Patterns. *arXiv preprint* 2024, arXiv:2403.08740. https://arxiv.org/abs/2403.08740.
12. 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)*; USENIX Association, 2025. arXiv:2504.11622. https://arxiv.org/abs/2504.11622.
13. 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. doi:10.1109/TASLP.2014.2339736.
14. Taheritajar, A.; Mahmoudpour Harris, Z.; Rahaeimehr, R. A Survey on Acoustic Side Channel Attacks on Keyboards. *arXiv preprint* 2023, arXiv:2309.11012. https://arxiv.org/abs/2309.11012.
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, April 2025. arXiv:2503.16719. https://arxiv.org/abs/2503.16719.
16. Stevens, S. S.; Volkmann, J.; Newman, E. B. A Scale for the Measurement of the Psychological Magnitude Pitch. *J. Acoust. Soc. Am.* **1937**, *8*, 185–190. doi:10.1121/1.1915893.
17. 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. doi:10.1109/TASSP.1980.1163420.
18. 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)*; 2015; pp. 18–25. doi:10.25080/Majora-7b98e3ed-003.
19. Smith, R. An Overview of the Tesseract OCR Engine. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*; IEEE, 2007; Volume 2, pp. 629–633. doi:10.1109/ICDAR.2007.4376991.
20. 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.
21. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*; PMLR, 2015; Volume 37, pp. 448–456. arXiv:1502.03167.
22. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*; 2014; Volume 27, pp. 3320–3328. arXiv:1411.1792.
23. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324. doi:10.1109/5.726791.
24. Shannon, C. E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423; 623–656. doi:10.1002/j.1538-7305.1948.tb01338.x.
25. 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.; Slaney, M.; Weiss, R. J.; Wilson, K. CNN Architectures for Large-Scale Audio Classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; 2017; pp. 131–135. doi:10.1109/ICASSP.2017.7952132.

 doi:10.20944/preprints202606.1478.v1




26. 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)*; 2012; pp. 89–90. doi:10.1145/2414456.2414509.
27. 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)*; 2008; pp. 170–183. doi:10.1109/SP.2008.20.
28. 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. doi:10.1109/TASLP.2020.3030497.